Computing with the AMSTRAD

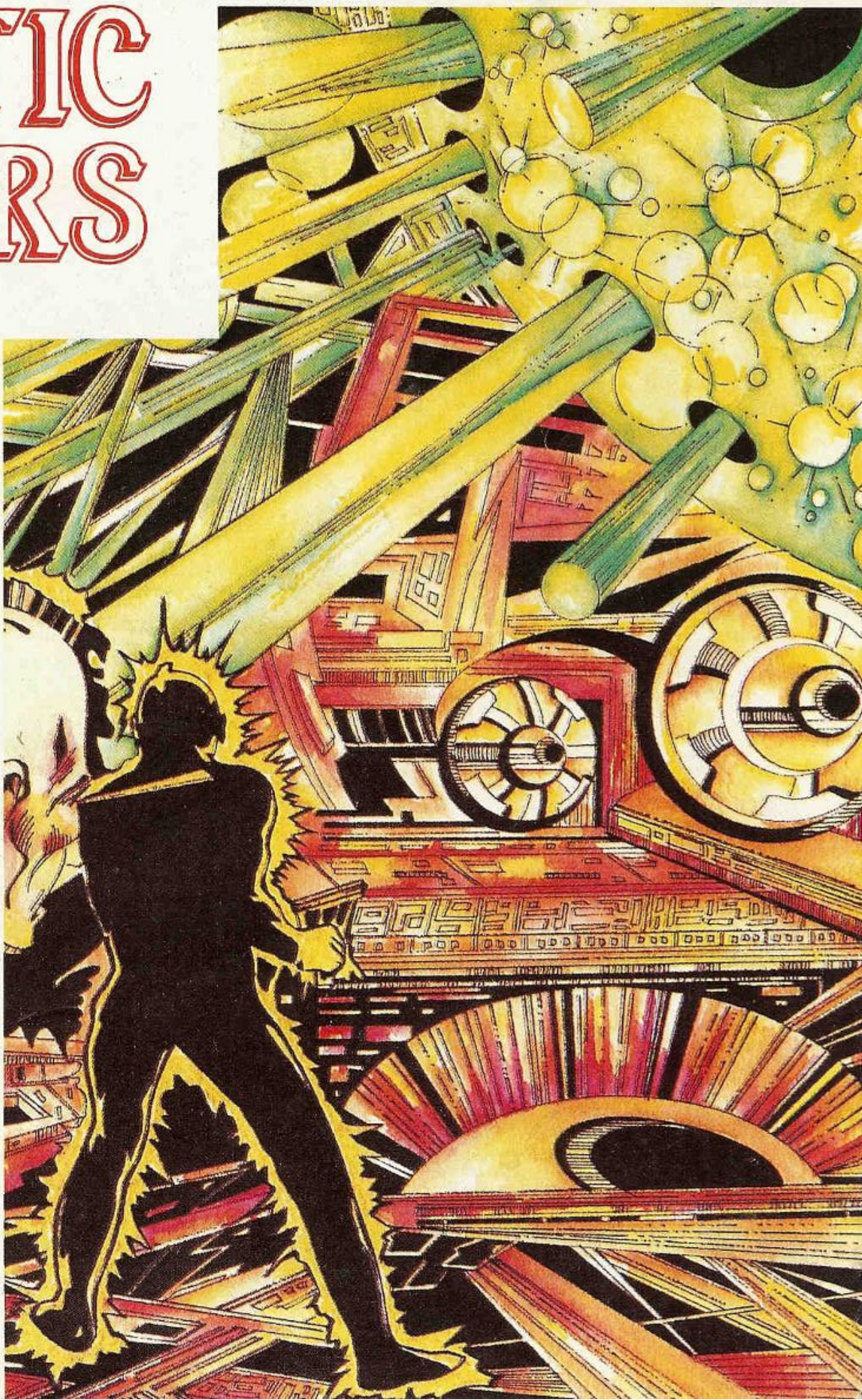
March, 1987
$4.50
A Database Publication

The independent magazine for Amstrad computer users

Registered by Australia Post - Publication No. VBP7811

GALACTIC INVADERS

GAME OF THE MONTH



Lot's more inside....

We've got series on Sound, Graphics, Basic, Machine Code, CP/M and Public Domain.

There's a full assembler listing, utility and game reviews, hardware reviews and more...

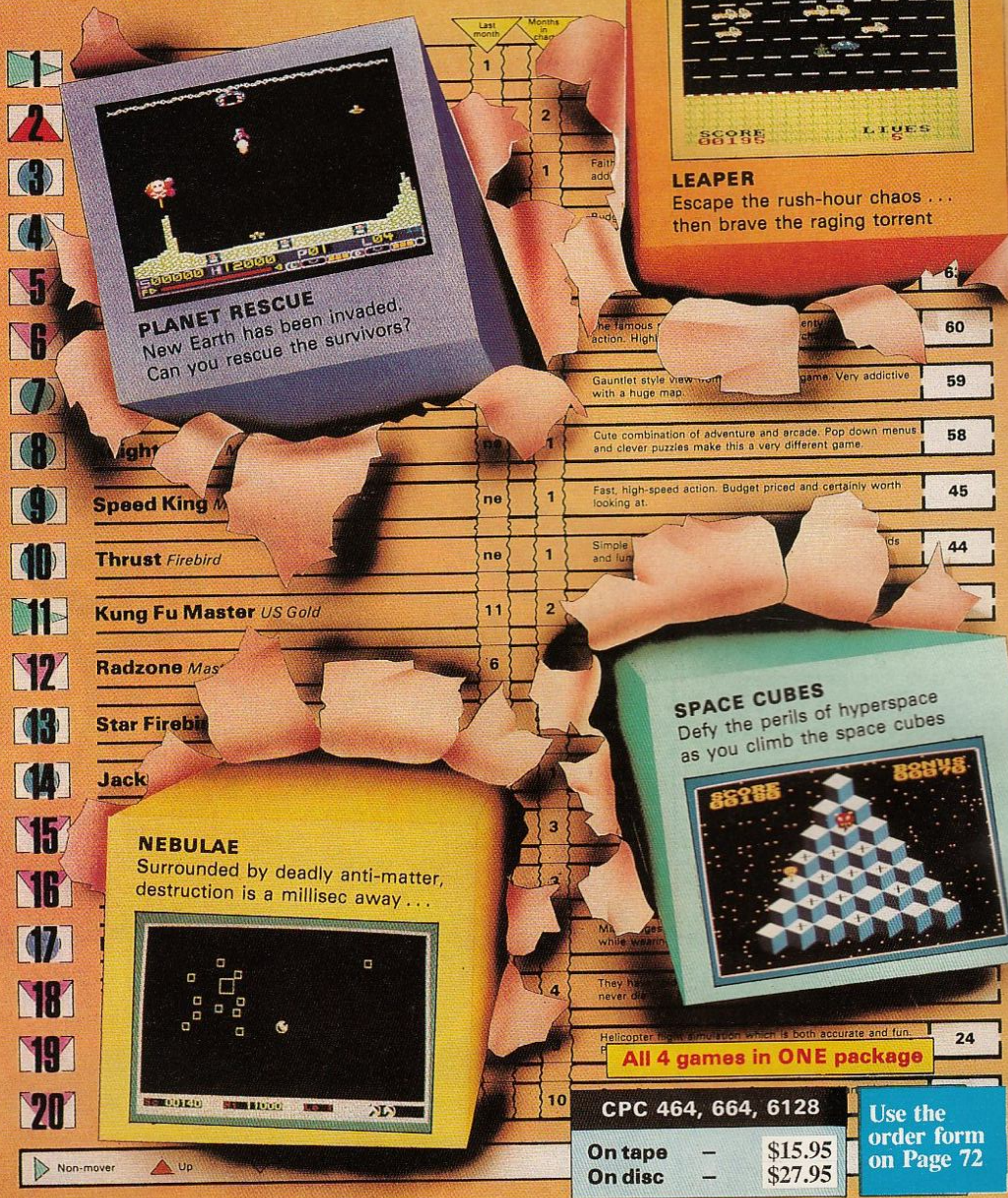
Our sixteen page business section checks out PCW software, there's your letters, puzzles, Ready Reference chart, Amstrad Analysis and of course, the latest from the U.K.

If that's not enough we've got Mail order offers and an article on the Amstrad's interrupts!

Get in NOW!

Chart Busters

Four action-packed classics to challenge the most expert gamester



Last month
Months in chart
Night
Speed King
Thrust Firebird
Kung Fu Master US Gold
Radzone
Star Firebi
Jack
Faith add
Bud
he famous ent action. High ch
Gauntlet style view game. Very addictive with a huge map.
Cute combination of adventure and arcade. Pop down menus and clever puzzles make this a very different game.
Fast, high-speed action. Budget priced and certainly worth looking at.
Simple ids and fun
Ma ges while wearin
They h never di
Helicopter simulation which is both accurate and fun.
63
60
59
58
45
44
24
SCORE 00195
LIVES 5
BONUS
SCORE
Non-mover
Up

PLANET RESCUE
New Earth has been invaded.
Can you rescue the survivors?

LEAPER
Escape the rush-hour chaos . . .
then brave the raging torrent

SPACE CUBES
Defy the perils of hyperspace
as you climb the space cubes

NEBULAE
Surrounded by deadly anti-matter,
destruction is a millisec away . . .

All 4 games in ONE package

CPC 464, 664, 6128

On tape	–	$15.95
On disc	–	$27.95

Use the order form on Page 72

Learning CAN be fun

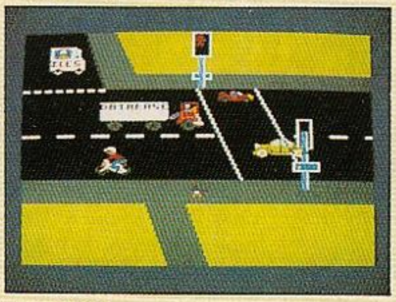
- Use your Amstrad to teach and amuse your children at the same time.
- Three packages crammed full of educational programs – and so easy to use!
- Each program has been educationally approved after extensive testing in the classroom.

ONLY
$15.95 - Tape
$27.95 - Disk

FUN SCHOOL!
10 programs for 8-12 year olds
Amstrad CPC 464, 664, 6128
FUN SCHOOL!
Amstrad CPC 464, 664, 6128
programs for 5-8 year olds
Amstrad CPC 464, 664, 6128

Ages 2-5

Alphabet
Colours
Counting
House
Magic Garden
Matchmaker
Numbers
Pelican
Seaside
Snap



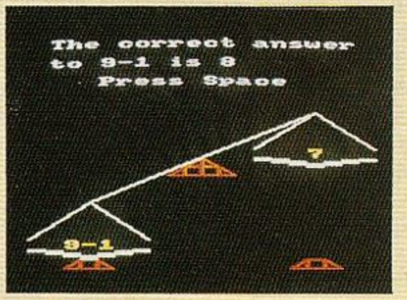
PELICAN
Teach your children to cross the road safely at a Pelican crossing

HOUSE
Select the colours to draw a house – hours of creative entertainment

Ages 5-8

Balance
Castle
Derrick
Fred's Words
Hilo
Maths Test
Mouser
Number Signs
Seawall
Super Spell

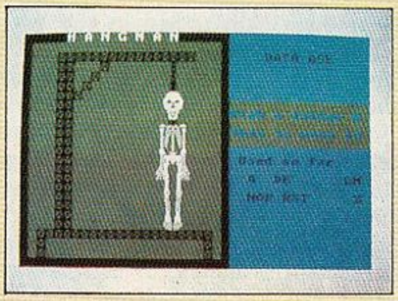
NUMBER SIGNS
Provide the correct arithmetic sign and aim to score ten out of ten



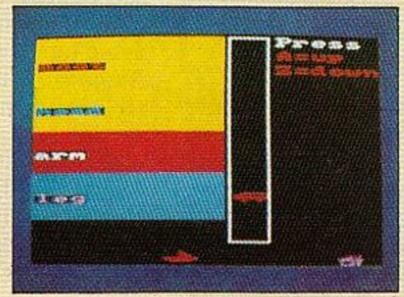
BALANCE
Learn maths the fun way. Type in the answer to balance the scales

Ages 8-12

Anagram
Codebreaker
Dog Duck Corn
Guessing
Hangman
Maths Hike
Nim
Odd Man Out
Pelmanism
Towers of Hanoi



HANGMAN
Improve your child's spelling with this fun version of the popular game



ODD MAN OUT
Find the word that does not fit – before your time runs out

To order, please use the form on Page 72

Icon's adventure

Icon Jon

$27.95- Tape Only

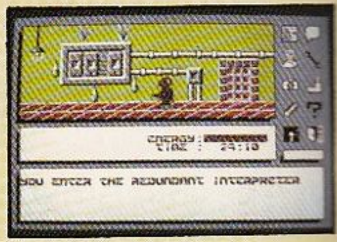
FOR many years I was under the misguided impression that at the heart of a computer lay a soulless mass of solid state electronics. I was wrong. Each component, variable and icon has thoughts and feelings. This is a story of one such icon by the name of Jon.

"Shut down in 30 minutes" was the message which went round the computer. Icon Jon's love of life had grown too strong – he didn't want to be reset. So with only 30 minutes remaining you must guide Jon through the computer's internals in a desperate attempt to find a new host computer.

Icon Jon is an arcade adventure game with the emphasis on the adventure aspect. The screen display is divided into several windows, the topmost displaying a four colour picture of Jon and his surroundings.

The next window down shows the time remaining and the items which Jon is carrying while the bottom window informs you of the room which you have just entered along with any incidental comments.

Finally, to the right of the screen there is a block of 10 icons which allow you to interact with Jon. The status icon tells you the percentage of the adventure you have completed while the chat icon allows you to talk to passing components and persuade them to help you. The action and manipulate icons allow you to collect, examine and use the various objects which



YOU ENTER THE REDUNDANT INTERPRETER

you will encounter.

Well programmed with a multitude of useful options, Icon Jon is a simple to operate yet very taxing adventure game. **Jon Revis**

Sound	6
Graphics	8
Playability	8
Value for money	8
Overall	8

Be a detective

L'Affaire Vera Cruz

$29.95 - Tape
$44.95 - Disk

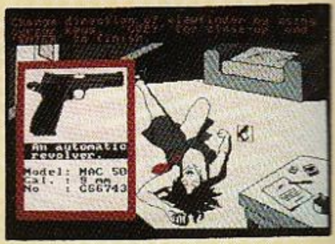
A BEAUTIFUL brunette lies sprawled on the apartment floor. The pool of blood surrounding her suggests that she is no longer alive. The evidence points to suicide, but your instincts tell you that something is not quite right.

The investigation takes place in two separate stages. Having loaded part one of the game you are presented with a highly detailed drawing of the scene of the crime. As you cannot touch anything until the forensic people have done their bit you limit yourself to taking photographs of the scene.

A small red square representing the viewfinder of your camera can be moved around the screen. By placing it on an object such as a matchbox and pressing the copy key a blown up photo is displayed. You are advised to examine everything in the apartment in minute detail and make notes on all you find. Once you are satisfied that you have not missed a clue then you can load the next part of the program.

You are now sitting at your desk in St Etienne. Using your typewriter you list the objects which you found at the scene of the crime.

As the French police force is a very high tech organisation your investigations are performed from the comfort of your desktop terminal. Via the computer network you are able to contact other police services, justice departments, prison administrations, and the national police.


Change direction of viewfinder by using cursor keys. 'COPY' for close-up and [illegible] to finish

An automatic revolver.
Model: MAC 50
Cal. : 9 mm
No : C56743

L'Affaire Vera Cruz is not a game but a real life criminal investigation. It is worth every penny and is highly recommended.

Carol Barrow

Sound	6
Graphics	9
Playability	9
Value for money	9
Overall	9

Atlantic antics

Virgin Atlantic Challenge game

$27.95 - Tape Only

IN the summer of 1986 Richard Branson and his crew broke the record for crossing the Atlantic Ocean in a boat. This program released by Virgin is a fun simulation of that event.

Richard Branson's crew have abandoned him on Challenger 2, which is divided into three main areas, the cockpit, the computer room, and the living and engine room.

To complete the game successfully you must divide your attentions equally among avoiding icebergs, feeding Richard, locating the fuel tankers and running the Virgin empire.

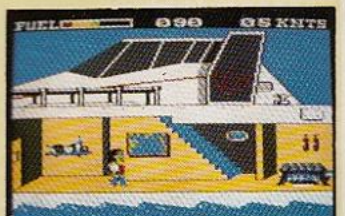
Any spare time is best spent eating, which is achieved by standing in the galley and pressing the fire button. A calorie meter will begin to rise as you eat. If you leave Mr Branson too long without food he will turn a funny shade of green and slow down a great deal.

An information line at the bottom of the screen constantly flashes warnings of obstacles in the water and incoming telexes.

One blatant omission is that the offending obstacles cannot be seen from the cockpit. One minute the view is perfectly clear, the next you're paying Davey Jones a visit.

Telexes are answered by running to the onboard computer. You are presented with a business proposition which requires a yes or no answer, your aim being to make money.

The Virgin Atlantic Chal-



lenge Game should keep you amused for an hour or so. The graphics are quite pretty but that's about as much as I can say in its favour.

Steve Brook

Sound	5
Graphics	7
Playability	6
Value for money	6
Overall	6

Computing With The Amstrad

March 1987

Published monthly by Strategy Software under license from Database Publications Ltd.

Telephone: [002] 29 4377
Telex: AA 58134
Attn. HT163

Mail: P.O. Box 11
Blackmans Bay
Tasmania 7152

Annual Subscription (12 issues)

Australia:	$ 40
South Pacific (inc. NZ)	$A60

'Computing With The Amstrad' welcomes program listings and articles for publication. Material should be typed or computer printed, and preferably double spaced. Program listings should be accompanied by cassette tape or disk. Please enclose a stamped addressed envelope or the return of material cannot be guaranteed. Contributions accepted for publication by Database Publications or its licensee will be on an all-rights basis.

Contents

SEPERATE BUSINESS INDEX - PAGE 41.

MACHINE CODE PROGRAMMING - in the Raw!

Roland Waddilove takes the strain out of machine code programming with this Z80 assembler for the Amstrad

If you intend following Mike Bibby's excellent introduction to machine code programming, then by you will probably be looking round for some sort of assembler to take all the hard work out of writing machine code programs.

As you probably know, a machine code program consists of a series of binary numbers in the range 0 to 255 (which we usually enter in hexadecimal).

This makes a program very difficult to read. What does &26, &2A mean? Very little, I should imagine, unless you know all the opcodes off by heart.

Assembly language is far easier to digest. A mnemonic is used to represent each machine code instruction. For example, the code above can be written as:

```
LD H,&2A
```

which is far more meaningful. It's not perfect, but is's a big improvement.

```
.score=&2A
LD H,score
```

is even better.

What an assembler does is to convert these assembly language mnemonics into machine code for you. There is no need to look up the individual codes and type them in as hex numbers.

You will find programs are far easier to write and far simpler to debug if they don't work first time (and they rarely do) and by using labels to represent constants and addresses machine code can become quite readable.

RAW, the assembler presented here, will allow you to write in assembly language.

When it is run, the mnemonics and labels will be converted into Z80 machine code which will be stored starting at any address not occupied by the assembler itself.

As the assembler is in Basic, it resides at the bottom of the memory, so it's convenient to place the code near the top.

HIMEM can be moved down if necessary to create space for the object (machine) code.

The assembly language program (source code) or the machine code generated by it can be saved to disk or tape, with or without the assembler program itself.

The code can afterwards be CALLed (there is no need for an assembler once the object code has been produced).

RAW will make such a difference to your programming that once you have used this assembler I can guarantee you will never do it by hand again.

All instructions and labels are placed in DATA statements before the assembler. Instructions can be 1,2 or 3 part.

Only one instruction per line is allowed and it must be typed in upper case with one space between the first and second part (if any), and a comma between the second and third parts (if any). No extra spaces are allowed.

Labels must be in lower case only.

For example:

RET	1 part instruction.
DJNZ loop	2 part instruction with label, 1 space.
LDA,(IX+4)	3 part instruction,1 space, 1 comma.

Himem will usually be set to reserve space for the code. Use MEMORY as normal.

```
10 MEMORY &7FFF:REM&8000
on is available for code
```

The variable printer used by the assembler can be set to direct the listing to the printer on the second pass. *printer=0* turns printer off, *printer=1* turns printer on.

```
20 printer=0:REM printer
off
```

Note that there are no spaces between the label, equals, and the number.

The first instruction must be ORG: the code is assembled at this address.

```
30 DATA ORG &8000
```

A label must be preceded by a full-stop when it is defined.

Labels must be used for relative jjumps.

A label can be used instead of a number, but must be positive.

They can be set to the current address by simply defining a label, as in line 70 of Program I.

```
10 REM PROGRAM I
20 DATA .print = &BB5A
30 DATA .offset=5
60 REM
70 DATA .loop
80 DATA LD A,(IX+offset)
90 DATA CALL print
95 DATA JR Z,loop
```

Alternatively they can be given a positive value (address or constant), by use of=, as in lines 20 and 30 of the same program.

Numbers can be decimal, hexa-decimal binary or labels as in Program II. They must, however, be positive.

```
10 REM PROGRAM II
100 DATA LD A, 27
110 DATA LD B, &1B
120 DATA LD C, %11011
130 DATA .number=27
140 DATA LD D, number
```

Single bytes (DEFB), double bytes (DEFW) and strings (DEF$) can be placed at the current address. Space can be reserved for data (DEFS followed by the number of bytes required). See Program III for an example:

```
10 REM PROGRAM III
150 DATA DEFB &FF
160 DATA DEFW number
170 DATA DEF$ 'Assembler'
190 DATA DEFS 150
200 DATA . number=1000
```

Comments can be included in the listing by placing them on a separate line and starting them with a ',(on the same key as 7).

Run the assembler. It will stop with an "Origin?" message.

Type AUTO and press the small Enter key. Throughout the program use the small Enter key instead of the large one.

The full-stop next to the small Enter key can be used instead if a comment is needed on the next line.

The source listing can be saved with the assembler or the assembler can be deleted and the source code saved.

The assembler can be merged with the source code when loading.

```
load "assembler"
merge "code"
```

There is a bug in the disk filing system which sometimes prevents two programs merging successfully.

If you have any problems, try re-saving one.

Alternatively, one could be saved as an Ascii file.

Machine code can be saved or loaded as a binary file, for example:

```
save "code", b, &8000, &100
```

The program is quite well structured with hardly a GOTO to be seen.

Subroutines are separated by lines with a single colon for clarity.

Line numbers start at 5000 to allow room for the assembler text.

Some of the lines are very long. If they will not fit in, then you've put in some additional spaces somewhere. The program was written in Mode 2 so it makes more sense if read in that mode.

The classes of instruction are read into an array and the first part of the instruction, *textl$*, is checked against this.

If a match is found, then ON GOSUB is used to select the correct subroutine.

The names of all the labels are stored in the array *label$(50)*, and their values in *value(50)*.

As you can see from the listing, several instructions are assembled by the same subroutine. This is possible because of the similarity in the bit pattern between instructions.

BIT, RES and SET are a good example of this similarity.

The left two bits are 01, 10 and 11, and the rest are the same.

So by setting a variable the same subroutine can be used for all three.

When you have finished entering the program, test it on all the groups of instructions to make sure it is OK.

All instructions are assembled correctly and the program has been used to assemble many routines.

If it stops with an error report first check the DATA statements to see if they are correct, then the appropriate subroutine.

If you find the prospect of typing in such a long complex program too daunting, remember it is available on the monthly tape of listings.

Any errors that occur when using this are caused by an instruction being entered incorrectly.

Please note the rules above about spaces and so on. They are quite rigid.

Apart from that you'll find RAW to be friendly, functional and extremely flexible. Happy assembling!

Program listing follows overleaf:

```
5000 REM ======= assembler ========
5010 REM By R.A.Waddiove
5020 REM(c)Computing With The Amstr
ad
5030 MODE 1:INK 0,0:BORDER 0
5040 GOSUB 5330:'initialise
5050 FOR pass=1 TO 2
5060 IF pass=2 AND printer=1 THEN l
isto=8
5070 CLS:RESTORE:READ textl$
5080 IF LEFT$(textl$,3)<>"ORG" THEN
 PRINT "Origin ?":END
5090 x$=MID$(textl$,INSTR(textl$,"
")+1):GOSUB 7570:pc=x
5100 PRINT #listo,"RAW Assembler V.
3":PRINT #listo:PRINT #listo,"Pass.
..";pass;TAB(20);textl$:PRINT #list
o
5110 WHILE textl$<>"END"
5120 PEN 1:READ textl$:textr$="":IF
 LEFT$(textl$,1)="'" THEN PRINT #li
sto,TAB(20)MID$(textl$,2):GOTO 5120
5130 PRINT #listo,HEX$(pc);":";:PEN
 3
5140 byte1=-1:byte2=-1:byte3=-1:byt
e4=-1:index=-1
5150 i%=-1
5160 FOR j%=0 TO 69
5170 IF INSTR(textl$,type$(j%))=1 T
HEN i%=j%
5180 NEXT
5190 IF i%=-1 THEN PRINT:PRINT text
l$;"??":END
5200 IF i%<36 THEN ON i% GOSUB 6860
,7010,7230,6460,7110,6160,6380,6500
,6420,6420,6420,6420,6380,6380,6760
,6380,5980,6380,5900,6380,6380,6010
,6060,6760,6420,6420,6420,6420,6260
,6210,5590,6420,6420,6420,6420
5210 IF i%>35 THEN i%=i%-35:ON i% G
OSUB 6420,6380,6510,6420,6420,6110,
6420,6420,5520,5520,7130,6340,6420,
6420,6650,6380,6380,6640,6420,6670,
6380,6660,6380,6420,5490,6990,6380,
7140,6600,6620,6630,6480,6490,5420
5220 IF index>-1 THEN x=index:GOSUB
 7350
5230 IF byte1>-1 THEN x=byte1:GOSUB
 7350
5240 IF byte2>-1 THEN x=byte2:GOSUB
 7350
5250 IF byte3>-1 THEN x=byte3:GOSUB
 7350
5260 IF byte4>-1 THEN x=byte4:GOSUB
 7350
5270 PEN 2:PRINT #listo,TAB(20);tex
tl$;:IF textr$="" THEN PRINT #listo
 ELSE PRINT #listo,",";textr$
5280 WEND
5290 NEXT pass
5300 PEN 1
5310 END
5320 :
5330 REM === initialise ===
5340 DIM type$(69),label$(50),value
(50),cond$(7)
5350 RESTORE 7810:FOR i%=0 TO 69:RE
AD type$(i%):NEXT
5360 RESTORE 7830:FOR i%=0 TO 7:REA
D cond$(i%):NEXT
5370 KEY 139,CHR$(13)+"DATA   "
5380 KEY 138,CHR$(13)+"REM    "
5390 numl=0
5400 RETURN
5410 :
5420 REM === def ===
5430 IF MID$(textl$,4,1)="S" THEN x
$=MID$(textl$,6):GOSUB 7570:pc=pc+x
:RETURN
5440 IF MID$(textl$,4,1)="$" THEN F
OR j%=INSTR(textl$,"'")+1 TO LEN(te
xtl$)-1:POKE pc,ASC(MID$(textl$,j%)
):pc=pc+1:NEXT:RETURN
5450 x$=MID$(textl$,INSTR(textl$,"
")+1):GOSUB 7570
5460 IF MID$(textl$,4,1)="B" THEN b
yte1=x ELSE GOSUB 7480:byte1=lb:byt
e2=hb
5470 RETURN
5480 :
5490 REM === rst ===
5500 x$=MID$(textl$,5):x=VAL("&"+x$
)\8:byte1=VAL("&X11"+BIN$(x,3)+"111
"):RETURN
5510 :
5520 REM === push/pop ===
5530 x$=RIGHT$(textl$,2)
5540 IF x$="IX" OR x$="IY" THEN ind
ex=-&DD*(x$="IX")-&FD*(x$="IY"):x$=
"HL"
5550 IF x$="AF" THEN byte1=-&F5*(IN
STR(textl$,"U")>0)-&F1*(INSTR(textl
$,"O")>0):RETURN
5560 GOSUB 7730:IF INSTR(textl$,"PO
P") THEN byte1=VAL("&X11"+x$+"0001"
) ELSE byte1=VAL("&X11"+x$+"0101")
5570 RETURN
5580 :
5590 REM === ld ===
5600 READ textr$
5610 IF INSTR(textl$,"IX")+INSTR(te
xtr$,"IX") THEN index=&DD
5620 IF INSTR(textl$,"IY")+INSTR(te
xtr$,"IY") THEN index=&FD
5630 IF INSTR(textl$,"(")+INSTR(tex
tr$,"(")=0 GOTO 5810:'no brackets
5640 IF INSTR(textl$,"(") GOTO 5720
:'left bracket
5650 IF textr$="(HL)" THEN x$=RIGHT
$(textl$,1):GOSUB 7690:byte1=VAL("&
X01"+x$+"110"):RETURN
5660 x$=textr$:IF x$="(BC)" OR x$="
(DE)" THEN byte1=-&A*(x$="(BC)")-&1
A*(x$="(DE)"):RETURN
5670 IF INSTR(textl$,"HL")+INSTR(te
xtl$,"I") THEN byte1=&2A:x$=MID$(te
xtr$,2,LEN(textr$)-2):GOSUB 7570:GO
SUB 7480:byte2=lb:byte3=hb:RETURN
5680 IF INSTR(textr$,"I") THEN x$=R
IGHT$(textl$,1):GOSUB 7690:byte1=VA
L("&X01"+x$+"110"):x$=MID$(textr$,5
,LEN(textr$)-5):GOSUB 7570:byte2=x:
RETURN
```

```
5690 IF LEN(textl$)=4 THEN byte1=&3
A:x$=MID$(textr$,2,LEN(textr$)-2):G
OSUB 7570:GOSUB 7480:byte2=lb:byte3
=hb:RETURN
5700 byte1=&ED:x$=RIGHT$(textl$,2):
GOSUB 7730:byte2=VAL("&X01"+x$+"101
1"):x$=MID$(textr$,2,LEN(textr$)-2)
:GOSUB 7570:GOSUB 7480:byte3=lb:byt
e4=hb:RETURN
5710 :
5720 IF LEFT$(textr$,1)="I" OR text
r$="HL" THEN byte1=&22:x$=MID$(text
l$,5,LEN(textl$)-5):GOSUB 7570:GOSU
B 7480:byte2=lb:byte3=hb:RETURN
5730 IF INSTR(textl$,"(I") THEN GOT
O 5790
5740 x$=MID$(textl$,5,1):IF x$="B"
OR x$="D" THEN byte1=-&2*(x$="B")-&
12*(x$="D"):RETURN
5750 x$=textr$:IF x$="BC" OR x$="DE
" OR x$="SP" THEN byte1=&ED:GOSUB 7
730:byte2=VAL("&X01"+x$+"0011"):x$=
MID$(textl$,5,LEN(textl$)-5):GOSUB
7570:GOSUB 7480:byte3=lb:byte4=hb:R
ETURN
5760 IF INSTR(textl$,"H")=0 THEN by
te1=&32:x$=MID$(textl$,5,LEN(textl$
)-5):GOSUB 7570:GOSUB 7480:byte2=lb
:byte3=hb:RETURN
5770 x$=textr$:IF x$>="A" AND x$<="
L" THEN GOSUB 7690:byte1=VAL("&X011
10"+x$):RETURN
5780 byte1=&36:GOSUB 7570:byte2=x:R
ETURN
5790 IF textr$>="A" AND textr$<="L"
 THEN x$=textr$:GOSUB 7690:byte1=VA
L("&X01110"+x$):x$=MID$(textl$,8,LE
N(textl$)-8):GOSUB 7570:byte2=x:RET
URN
5800 byte1=&36:x$=MID$(textl$,8,LEN
(textl$)-8):GOSUB 7570:byte2=x:x$=t
extr$:GOSUB 7570:byte3=x:RETURN
5810 IF RIGHT$(textl$,1)="I" OR tex
tr$="I" THEN byte1=&ED:byte2=-&57*(
textr$="I")-&47*(textr$="A"):RETURN
5820 IF RIGHT$(textl$,1)="R" OR tex
tr$="R" THEN byte1=&ED:byte2=-&5F*(
textr$="R")-&4F*(textr$="A"):RETURN
5830 IF INSTR(textl$,"I") THEN byte
1=&21:x$=textr$:GOSUB 7570:GOSUB 74
80:byte2=lb:byte3=hb:RETURN
5840 IF textr$="HL" OR LEFT$(textr$
,1)="I" THEN byte1=&F9:RETURN
5850 IF MID$(textl$,LEN(textl$)-1,1
)>" " THEN x$=RIGHT$(textl$,2):GOSU
B 7730:byte1=VAL("&X00"+x$+"0001"):
x$=textr$:GOSUB 7570:GOSUB 7480:byt
e2=lb:byte3=hb:RETURN
5860 x$=RIGHT$(textl$,1):GOSUB 7690
5870 IF textr$>="A" AND textr$<="L"
 THEN byte1=VAL("&X01"+x$):x$=textr
$:GOSUB 7690:byte1=VAL("&X"+BIN$(by
te1,5)+x$):RETURN
5880 byte1=VAL("&X00"+x$+"110"):x$=
textr$:GOSUB 7570:byte2=x:RETURN
5890 :
5900 REM === ex ===
5910 READ textr$
5920 IF textr$="IX" THEN index=&DD
5930 IF textr$="IY" THEN index=&FD
5940 IF INSTR(textl$,"S") THEN byte
1=&E3:RETURN
5950 IF textr$="HL" THEN byte1=&EB
ELSE byte1=&8
5960 RETURN
5970 :
5980 REM === djnz ===
5990 byte1=&10:x$=MID$(textl$,INSTR
(textl$," ")+1):GOSUB 7570:GOSUB 75
20:byte2=x:RETURN
6000 :
6010 REM === im ===
6020 byte1=&ED:x=VAL(RIGHT$(textl$,
1))
6030 byte2=-&46*(x=0)-&56*(x=1)-&5E
*(x=2)
6040 RETURN
6050 :
6060 REM === in ===
6070 READ textr$
6080 IF textr$="(C)" THEN byte1=&ED
:x$=RIGHT$(textl$,1):GOSUB 7690:byt
e2=VAL("&X01"+x$+"000"):RETURN
6090 byte1=&DB:x$=MID$(textr$,2,INS
TR(textr$,")")-2):GOSUB 7570:byte2=
x:RETURN
6100 :
6110 REM === out ===
6120 READ textr$
6130 IF INSTR(textl$,"C") THEN byte
1=&ED:x$=textr$:GOSUB 7690:byte2=VA
L("&X01"+x$+"001"):RETURN
6140 byte1=&D3:x$=MID$(textl$,6,INS
TR(textl$,")")-6):GOSUB 7570:byte2=
x:RETURN
6150 :
6160 REM === call ===
6170 x$=RIGHT$(textl$,2):GOSUB 7410
6180 IF x=-1 THEN x$=MID$(textl$,6)
:GOSUB 7570:GOSUB 7480:byte1=&CD:by
te2=lb:byte3=hb:RETURN
6190 byte1=VAL("&X11"+BIN$(x,3)+"10
0"):READ textr$:x$=textr$:GOSUB 757
0:GOSUB 7480:byte2=lb:byte3=hb:RETU
RN
6200 :
6210 REM === jr ===
6220 x$=RIGHT$(textl$,2):GOSUB 7410
6230 IF x=-1 THEN x$=MID$(textl$,4)
:GOSUB 7570:GOSUB 7520:byte1=&18:by
te2=x:RETURN
6240 byte1=VAL("&X001"+BIN$(x,2)+"0
00"):READ textr$:x$=textr$:GOSUB 75
70:GOSUB 7520:byte2=x:RETURN
6250 :
6260 REM === jp ===
6270 IF INSTR(textl$,"X") THEN inde
x=&DD
6280 IF INSTR(textl$,"Y") THEN inde
x=&FD
6290 IF INSTR(textl$,"(") THEN byte
1=&E9:RETURN
6300 x$=RIGHT$(textl$,2):GOSUB 7410
```

```
6310 IF x=-1 THEN x$=MID$(textl$,4)
:GOSUB 7570:GOSUB 7480:byte1=&C3:by
te2=lb:byte3=hb:RETURN
6320 byte1=VAL("&X11"+BIN$(x,3)+"01
0"):READ textr$:x$=textr$:GOSUB 757
0:GOSUB 7480:byte2=lb:byte3=hb:RETU
RN
6330 :
6340 REM === ret ===
6350 IF textl$="RET" THEN byte1=&C9
 ELSE x$=RIGHT$(textl$,2):GOSUB 741
0:byte1=VAL("&X11"+BIN$(x,3)+"000")
6360 RETURN
6370 :
6380 REM === odds & ends ===
6390 x$=textl$:byte1=-&1F*(x$="RRA"
)-&F*(x$="RRCA")-&37*(x$="SCF")-&3F
*(x$="CCF")-&2F*(x$="CPL")-&27*(x$=
"DAA")-&F3*(x$="DI")-&FB*(x$="EI")-
&D9*(x$="EXX")-&76*(x$="HALT")-&17*
(x$="RLA")-&7*(x$="RLCA")
6400 RETURN
6410 :
6420 x$=textl$:x=-&A9*(x$="CPD")-&6
7*(x$="RRD")-&B9*(x$="CPDR")-&A1*(x
$="CPI")-&B1*(x$="CPIR")-&AA*(x$="I
ND")-&BA*(x$="INDR")-&A2*(x$="INI")
-&B2*(x$="INIR")-&A8*(x$="LDD")-&B8
*(x$="LDDR")-&A0*(x$="LDI")-&B0*(x$
="LDIR")
6430 byte2=x-&44*(x$="NEG")-&BB*(x$
="OTDR")-&B3*(x$="OTIR")-&AB*(x$="O
UTD")-&A3*(x$="OUTI")-&4D*(x$="RETI
")-&45*(x$="RETN")-&6F*(x$="RLD")
6440 byte1=&ED:RETURN
6450 :
6460 REM === and/sub/xor ===
6470 byte=&A6:GOTO 6530:'and
6480 byte=&96:GOTO 6530:'sub
6490 byte=&AE:GOTO 6530:'xor
6500 byte=&BE:textl$="CP "+MID$(tex
tl$,3):GOTO 6530:'cp
6510 byte=&B6:textl$="OR "+MID$(tex
tl$,3):GOTO 6530:'or
6520 :
6530 IF INSTR(textl$,"I") GOTO 6570
6540 IF INSTR(textl$,"HL") THEN byt
e1=byte:RETURN
6550 x$=RIGHT$(textl$,1):IF (x$>="A
" AND x$<="L") AND INSTR(textl$,"&"
)=0 THEN GOSUB 7690:byte1=(byte AND
 &X11111000)+VAL("&X"+x$):RETURN
6560 byte1=byte OR &X1000000:x$=MID
$(textl$,5):GOSUB 7570:byte2=x:RETU
RN
6570 IF INSTR(textl$,"IX") THEN ind
ex=&DD ELSE index=&FD
6580 byte1=byte:x$=MID$(textl$,9,IN
STR(textl$,")")-9):GOSUB 7570:byte2
=x:RETURN
6590 :
6600 REM === sla/sra/srl/rlc/rl/rrc
/rr ===
6610 byte=&26:GOTO 6690:'sla
6620 byte=&2E:GOTO 6690:'sra
6630 byte=&3E:GOTO 6690:'srl
6640 byte=&6:GOTO 6690:'rlc
6650 byte=&16:textl$="RL "+MID$(tex
tl$,3):GOTO 6690:'rl
6660 byte=&E:GOTO 6690:'rrc
6670 byte=&1E:textl$="RR "+MID$(tex
tl$,3):GOTO 6690:'rr
6680 :
6690 IF INSTR(textl$,"I") GOTO 6730
6700 byte1=&CB
6710 IF INSTR(textl$,"HL") THEN byt
e2=byte:RETURN
6720 x$=RIGHT$(textl$,1):IF x$>="A"
 AND x$<="L" THEN GOSUB 7690:byte2=
(byte AND &X11111000)+VAL("&X"+x$):
RETURN
6730 IF INSTR(textl$,"IX") THEN ind
ex=&DD ELSE index=&FD
6740 byte1=&CB:x$=MID$(textl$,9,INS
TR(textl$,")")-9):GOSUB 7570:byte2=
x:byte3=byte:RETURN
6750 :
6760 REM === inc/dec ===
6770 IF INSTR(3,textl$,"I") THEN GO
TO 6820
6780 byte=-1*(textl$<"I")
6790 IF INSTR(textl$,"(H") THEN byt
e1=&34 OR byte:RETURN
6800 IF RIGHT$(textl$,2)<"!" THEN x
$=RIGHT$(textl$,1):GOSUB 7690:byte1
=byte OR VAL("&X00"+x$+"100"):RETUR
N
6810 byte=-8*(textl$<"I"):x$=RIGHT$
(textl$,2):GOSUB 7730:byte1=byte OR
 VAL("&X00"+x$+"0011"):RETURN
6820 IF INSTR(textl$,"IX") THEN ind
ex=&DD ELSE index=&FD
6830 IF INSTR(textl$,"(") THEN byte
=-1*(textl$<"I"):byte1=&34 OR byte:
x$=MID$(textl$,9,INSTR(textl$,")")-
9):GOSUB 7570:byte2=x:RETURN
6840 byte=-8*(textl$<"I"):byte1=&23
 OR byte:RETURN
6850 :
6860 REM === label ===
6870 PEN 1:k%=-1:x$=MID$(textl$,2)
6880 IF INSTR(x$,"=")>0 THEN x$=LEF
T$(x$,INSTR(x$,"=")-1)
6890 FOR j%=0 TO numl
6900 IF x$=label$(j%) THEN k%=j%
6910 NEXT
6920 IF k%=-1 THEN k%=numl:numl=num
l+1
6930 IF numl>50 THEN PRINT:PRINT "T
oo many labels":STOP
6940 label$(k%)=x$
6950 IF INSTR(textl$,"=")=0 THEN va
lue(k%)=pc:RETURN
6960 x$=MID$(textl$,INSTR(textl$,"=
")+1):GOSUB 7570:value(k%)=x
6970 RETURN
6980 :
6990 REM === sbc/adc ===
7000 byte=&X10011110:GOTO 7020:'sbc
7010 byte=&X10001110:GOTO 7020:'adc
7020 READ textr$
7030 IF INSTR(textl$,"HL") THEN x$=
textr$:GOSUB 7730:byte1=&ED:byte2=V
AL("&X01"+x$+"0010")+(8 AND textl$<
"S"):RETURN
7040 IF textr$>="A" AND textr$<="L"
 THEN byte=byte AND &X11111000:x$=t
extr$:GOSUB 7690:byte1=byte+VAL("&X
"+x$):RETURN
7050 IF textr$="(HL)" THEN byte1=by
te:RETURN
7060 IF INSTR(textr$,"I") GOTO 7080
7070 byte1=byte OR &X1000000:x$=tex
tr$:GOSUB 7570:byte2=x:RETURN
7080 IF INSTR(textr$,"IX") THEN ind
ex=&DD ELSE index=&FD
7090 byte1=byte:x$=MID$(textr$,5,IN
STR(textr$,")")-5):GOSUB 7570:byte2
=x:RETURN
7100 :
7110 REM === bit/set/res ===
7120 byte=&X1000000:GOTO 7150:'bit
7130 byte=&X10000000:GOTO 7150:'res
7140 byte=&X11000000:GOTO 7150:'set
7150 READ textr$:x=VAL(RIGHT$(textl
$,1)):byte2=byte+VAL("&X"+BIN$(x,3)
+"000")
7160 IF INSTR(textr$,"I") GOTO 7200
:'ix/iy
7170 byte1=&CB
7180 IF textr$="(HL)" THEN byte2=by
te2+VAL("&X110"):RETURN
7190 x$=textr$:GOSUB 7690:byte2=byt
e2+VAL("&X"+x$):RETURN
7200 IF INSTR(textr$,"IX") THEN ind
ex=&DD ELSE index=&FD
7210 byte3=byte2+VAL("&X110"):byte1
=&CB:x$=MID$(textr$,5,INSTR(textr$,
")")-5):GOSUB 7570:byte2=x:RETURN
7220 :
7230 REM === add ===
7240 READ textr$
7250 IF INSTR(textl$,"Y") THEN inde
x=&FD
7260 IF INSTR(textl$,"X") THEN inde
x=&DD
7270 IF INSTR(textl$,"L") OR INSTR(
textl$,"I") THEN x$=textr$:GOSUB 77
30:byte1=VAL("&X00"+x$+"1001"):RETU
RN
7280 IF textr$>="A" AND textr$<="L"
 THEN x$=textr$:GOSUB 7690:byte1=VA
L("&X10000"+x$):RETURN
7290 IF textr$="(HL)" THEN byte1=&8
6:RETURN
7300 IF INSTR(textr$,"I") GOTO 7320
:'adc a,(ix/iy+n)
7310 byte1=&C6:x$=textr$:GOSUB 7570
:byte2=x:RETURN
7320 IF INSTR(textr$,"Y") THEN inde
x=&FD ELSE index=&DD
7330 byte1=&86:x$=MID$(textr$,5,INS
TR(textr$,")")-5):GOSUB 7570:byte2=
x:RETURN
7340 :
7350 REM === print hex$(x) ===
7360 POKE pc,x:pc=pc+1
7370 IF x<16 THEN PRINT #listo,"0";
7380 PRINT #listo,HEX$(x);" ";
7390 RETURN
7400 :
```

```
7410 REM === get code for condition
 x$ -> x ===
7420 x=-1
7430 FOR j%=0 TO 7
7440 IF x$=cond$(j%) THEN x=j%
7450 NEXT
7460 RETURN
7470 :
7480 REM === x -> lowbyte,hibyte ==
=
7490 lb=x-256*INT(x/256):hb=INT(x/2
56)
7500 RETURN
7510 :
7520 REM === calculate jump ===
7530 IF x>255 THEN x=x-pc-2:IF x>25
5 THEN PRINT textl$;" Too far":END
7540 IF x<0 THEN x=x+256
7550 RETURN
7560 :
7570 REM === x$,number/label -> x,n
umber ===
7580 WHILE x$<"!":x$=MID$(x$,2):WEN
D
7590 IF LEFT$(x$,1)="&" THEN x=VAL(
x$)-65536*(VAL(x$)<0):RETURN
7600 IF LEFT$(x$,1)>="0" AND LEFT$(
x$,1)<="9" THEN x=VAL(x$):RETURN
7610 IF LEFT$(x$,1)="%" THEN x=VAL(
"&X"+MID$(x$,2)):RETURN
7620 IF pass=2 THEN x=-1 ELSE x=0:R
ETURN
7630 FOR j%=0 TO numl
7640 IF label$(j%)=x$ THEN x=value(
j%)
7650 NEXT
7660 IF x<0 THEN PRINT textl$;" Lab
el??":END
7670 RETURN
7680 :
7690 REM === x$,register -> x$,bina
ry code ===
7700 x$=BIN$(ASC(x$)-66-8*(x$="A")+
2*(x$="H")+5*(x$="L"),3)
7710 RETURN
7720 :
7730 REM === x$,register pair -> x$
,binary code ===
7740 IF x$="BC" THEN x$="00"
7750 IF x$="DE" THEN x$="01"
7760 IF x$="HL" OR x$="IX" OR x$="I
Y" THEN x$="10"
7770 IF x$="SP" THEN x$="11"
7780 RETURN
7790 :
7800 REM === classes ===
7810 DATA END,..,ADC,ADD,AND,BIT,CAL
L,CCF,CP,CPD,CPDR,CPI,CPIR,CPL,DAA,
DEC,DI,DJNZ,EI,EX,EXX,HALT,IM,IN,IN
C,IND,INDR,INI,INIR,JP,JR,LD,LDD,LD
DR,LDI,LDIR,NEG,NOP,OR,OTDR,OTIR
7820 DATA OUT,OUTD,OUTI,POP,PUSH,RE
S,RET,RETI,RETN,RL,RLA,RLCA,RLC,RLD
,RR,RRA,RRC,RRCA,RRD,RST,SBC,SCF,SE
T,SLA,SRA,SRL,SUB,XOR,DEF
7830 DATA "NZ"," Z","NC"," C","PO",
"PE"," P"," M"
```

END

Get the facts at your fingertips with the fifth of our Amstrad reference charts

STRING$ allows you to create long strings made up of single characters. If you want a row of plusses you'd use:

```
PRINT STRING$(20,"+")
```

Notice that it only works with single characters.

```
character$=STRING$(10,"ab")
PRINT character$
```

LEFT$ slices off part of a string. As you might guess, it returns the leftmost part.

```
PRINT LEFT$("remainder",6)
```

results in "remain" appearing on the screen. LEFT$ has taken the leftmost six characters from "remainder", ignoring the rest. Try:

```
word$="thisbitthatbit"
PRINT LEFT$(word$,7)
```

and see what you get. The number after the string to be sliced decides how many characters are taken. What if this number is greater than the number of characters in the string to be sliced?

```
PRINT LEFT$("four",5)
```

shows that only the four characters are produced. In effect, the string is left intact.

RIGHT$ follows on from LEFT$, slicing off the right hand part of the string.

```
word$="leftrights"
PRINT RIGHT$(word$,6)
```

The number inside the brackets specifies the character at which the slicing is to start - measured inwards from the end of the string to be sliced. It is the number of characters to be lifted from the right hand part of the original string.

```
PRINT RIGHT$("12345678",2)
```

if the number specified is greater than the length of the string, the string remains intact.

```
PRINT RIGHT$("five",6)
```

SPACE$ allows you to create long lines of spaces. You'll see that:

```
PRINT SPACE$(11)
```

is a lot clearer than:

```
PRINT "           "
```

Although you can't see them, the spaces are there.

```
seeing$ = "is" + SPACE$(12) + "believing"
PRINT seeing$
```

MID$ is, in effect, a combined RIGHT$ and LEFT$. With it you can not only decide where the slice is to start, but also how characters of the original string are to be taken.

```
PRINT MID$("level",2,3)
```

should leave you with "eve" while:

```
PRINT MID$("madam",2,4)
```

should produce "adam".

The first number following the specified string indicates where the slice is to begin. The second number decides how long that slice will be.

```
word$="level":nextword$="ning all"
combined$=MID$(word$,2,3)+nextword$
PRINT combined$
```

If the final figure is omitted you just get the rest of the string from the start character.

```
PRINT MID$("rotor",2)
```

gives "otor".

If the character specified is greater than the number of characters in the string to be sliced then you get nothing.

```
slice$=MID$("123456",7,3)
PRINT LEN(slice$)
```

If you ask for more characters than the original string can supply, the micro does the best it can.

```
PRINT MID$("123456",3,5)
```

An eventful way to get your kicks...

ROLAND WADDILOVE discusses the use of interrupts fom machine code

AN interrupt is a signal sent to the Z80, the microprocessor at the heart of the Amstrad, informing it that its attention is required elsewhere immediately.

It stops what it is currently doing, carries out the process needing attention, then returns to it's previous task, carrying on where it left off.

Interrupts are used for many things. Perhaps the most obvious one is the internal clock, read by the Basic pseudo-variable TIME. Every 300th of a second the clock is incremented. Interrupts also occcur every 50th of a second so that the screen can be updated and the keyboard read, and every 100th of a second so sounds can be dealt with and envelopes processed.

As you can see, interrupts enable the Amstrad to appear to be carrying out more than one task at once. We don't notice any of these background tasks being performed as they are dealt with quickly and efficiently and in such a manner so as not to disturb the foreground task the Amstrad is running.

One of the machine's most powerful features is it's ability to handle interrupts from Basic. This is a major innovation, and is quite a rarity among home micros. For example, it is possible to arrange for a subroutine to be executed every five seconds.

If you haven't yet come across Basic interrupts then I would recommend that you look them up, they can be very useful. The associated Basic commands are AFTER, EVERY and REMAIN.

These are excellent, but how can the machine code programmer use the interrupt facility? It's not all that difficult, and once you get the hang of it you'll find the options available from machine code far more extensive than from Basic.

Handling interrupts can be quite a complicated process, but fortunately the operating system provides an easy-to-use prepackaged form of interrupt known as an event. Events are more flexible than hardware interrupts and there are fewer restrictions.

The time interrupts for the operating system provide three sources of "kicks" for events, and each source has an associated queue. When a particular event is kicked it is placed in the appropriate queue, so the event routine may not be called immediately.

The three sources of kicks are the fast ticker, ticker and frame flyback interrupts.

Fast ticker events are kicked every 1/300th of a second. Ticker events have a timer which is decremented every 1/50th of a second and when the timer reaches zero the event is called. It can then be automatically reset. Frame flyback events are kicked every 1/50th of a second.

There are three classes of events.

Express asynchronous events are called immediately during interrupt processing, but these are not normally used.

Normal asynchronous events are the most flexible type. When kicked they are placed in an interrupt queue to be processed when the operating system has finished its own interrupts. There are few restrictions, and the routine may take as long to run as is needed.

Synchronous events, when kicked, are placed on a separate queue according to a priority which they are given. The foreground program must poll the queue to see if there are any events outstanding, and process them if there are.

We're now going to look at how normal asynchronous events are processed and see what they can be used for. A couple of simple routines will be placed on the ticker list and we'll set them off at regular intervals.

The operating system requires a small block of memory as workspace for each event routine. We are going to use ticker events, and the block needed for each event is 13 bytes long.

The block is in two parts, the first six bytes being the ticker block and the last seven is the event block. (See Figure I.) The tick chain holds the address of the next tick block, if there is one. The tick count is the timer for the event and is decremented every 50th of a second. When it reaches zero the event routine is called.

Bytes	0/1	Tick chain
Bytes	2/3	Tick count
Bytes	4/5	Recharge count
Bytes	6/7	Chain
Byte	8	Count
Byte	9	Class
Bytes	10/11	Routine address
Byte	12	ROM select

Figure I: Workspace alteration

The recharge count is the value the timer is reset to after the event has been kicked.

Chain is used to store the event's position in the queue and count is the number of kicks received. The class I'll come back to. The routine address and the ROM select byte point to the entry address of the routine, which can be in ROM or RAM.

In order for our interrupt routine to be called we need to initialize a ticker block. &BCEF is called with the address of the event block in the HL register pair, the event address in the DE pair, the ROM select address in C and the class in B. We can ignore the ROM select address as the routine isn't in a ROM.

Bit 0	Near/far address
Bits 1-4	Synchronous event priority
Bit 5	Must be Zero
Bit 6	Express event
Bit 7	Asynchronous event

Figure II: Class byte

The class is bit significant - see Figure II. We are going to place the routines in the central 32k of RAM. This is classed as a near address, so bit 0 is set. The event is asynchronous, so bits 1-4 are meaningless. Most events will be normal and don't need urgent attention, so bit 6 is zero. It's an asynchronous event so bit 7 is set.

The first four lines of Program I shows the code to initialize the event block. The next four lines show how the tick block may be added to the ticker list.

```
RAW Assembler V.3

Pass... 2               ORG &8000

                        Initialise event
8000:21 28 80           LD HL,eventblock
8003:06 81              LD B,%10000001
8005:11 1D 80           LD DE,event
8008:C3 EF BC           JP &BCEF

                        Add to tick list
800B:21 22 80           LD HL,tickblock
800E:11 32 00           LD DE,50
8011:01 32 00           LD BC,50
8014:C3 E9 BC           JP &BCE9

                        Remove from list
8017:21 22 80           LD HL,tickblock
801A:C3 EC BC           JP &BCEC

                        Event routine
801D:                   .event
801D:3E 07              LD A,7
801F:C3 5A BB           JP &BB5A

                        Workspace
8022:                   .tickblock
8022:                   DEFS 6
8028:                   .eventblock
8028:                   DEFS 7
802F:                   END
```

Program I

HL holds the tick block address, DE the initial timer value, BC the recharge value, the routine calls &BCE9 and returns. Now every 50th of a second the timer is decremented and the routine called when it is zero. As we set it to 50 initially this will take exactly one second.

The event routine itself is very simple. It just outputs CHR$(7) by loading the A register with 7 and jumping to &BB5A, the same as CALL1 and RETURN. So every second the micro will beep no matter what it is doing unless interrupts have been disabled for some reason.

Program II shows a slightly more complex event routine. As you can see the initialization is identical, and, apart from the different timer values, so is the routine to add it to the ticker list.

```
RAW Assembler V.3

Pass... 2               ORG &8030

                        Initialise event
8030:21 61 80           LD HL,eventblock
8033:06 81              LD B,%10000001
8035:11 4D 80           LD DE,event
8038:C3 EF BC           JP &BCEF

                        Add to tick list
803B:21 5B 80           LD HL,tickblock
803E:11 05 00           LD DE,5
8041:01 05 00           LD BC,5
8044:C3 E9 BC           JP &BCE9

                        Remove from list
8047:21 5B 80           LD HL,tickblock
804A:C3 EC BC           JP &BCEC

                        Event routine
804D:                   .event
804D:3E 44              LD A,68
804F:CD 1E BB           CALL &BB1E
8052:C8                 RET Z
8053:                   .loop
8053:3E 46              LD A,70
8055:CD 1E BB           CALL &BB1E
8058:28 F9              JR Z,loop
805A:C9                 RET

                        Workspace
805B:                   .tickblock
805B:                   DEFS 6
8061:                   .eventblock
8061:                   DEFS 7

8068:                   END
```

Program II

The event routine reads the keyboard and freezes the program - Basic or machine code - if the Tab key is pressed. It waits until the Caps Lock key is pressed before continuing the program.

This is quite a useful little routine. It's quite handy to be able to freeze your favorite arcade game when the phone rings, for example, or just to have a breather.

If you want to use this do make sure it's tucked out of the way of your game. If you hide it in the function key buffer at about &B460 it will even freeze a fair proportion of commercial software.

```
10 REM PROGRAM III
20 FOR i=0 TO 90
30 READ byte$
40 POKE &8000+i,VAL("&"+byte$)
60 NEXT
70 DATA 21,28,80,06,81,11,1D,80,C3
80 DATA EF,BC,21,22,80,11,32,00,01
90 DATA 32,00,C3,E9,BC,21,22,80,C3
100 DATA EC,BC,3E,07,C3,5A,BB,00,00
110 DATA 00,00,00,00,00,00,00,00,00
120 DATA 00,00,00
130 REM
140 DATA 21,61,80,06,81,11,4D,80,C3
150 DATA EF,BC,21,5B,80,11,05,00,01
160 DATA 05,00,C3,E9,BC,21,5B,80,C3
170 DATA EC,BC,3E,44,CD,1E,BB,C8,3E
180 DATA 46,CD,1E,BB,28,F9,C9
```

Program III ↑

It's then possible to step through the program little by little and see how it works.

If you haven't got an assembler to hand then Program III will poke the relevant data to &8000.

So far I've not covered how to remove an event from the ticker list. It's quite simple. Just load the HL register pair with the address of the tick block and CALL &BCEC.

After assembling the routines CALL &8000 and CALL &8030 to initialize the events. CALL &800B and CALL &803B to enable them, CALL &8017 and CALL &8047 to disable them.

Program IV can be used to study the tick blocks and event blocks, *i* in line 60 should point to the start of the block. It prints out the contents of the event and tick blocks so you can watch the operating system actually running the events.

Once you've got the hang of events, you'll be amazed how often they pop up as an ideal solution to programming problems. If you're not using interrupts, you're not using your Amstrad to the full!

Program IV ↓

```
10 REM PROGRAM IV
20 MODE 1:INK 0,0:BORDER 0
30 PEN 1:PAPER 3:LOCATE 9,3
40 PRINT " TICKER AND EVENT BLOCK "
50 PAPER 0
60 i=&8022:j=i+2
70 LOCATE 1,7
80 PRINT TAB(9)"Tick chain";:GOSUB 19
0
90 PRINT TAB(9)"Tick count";:GOSUB 19
0
100 PRINT TAB(9)"Recharge Count";:GOS
UB 190
110 PRINT TAB(9)"Chain";:GOSUB 190
120 PRINT TAB(9)"Count";:GOSUB 180
130 PRINT TAB(9)"Class";:GOSUB 180
140 PRINT TAB(9)"Routine address";:GO
SUB 190
150 PRINT TAB(9)"ROM";:GOSUB 180
160 PEN 3
170 LOCATE 28,9:PRINT "&";HEX$(PEEK(j
)+256*PEEK(j+1),4):GOTO 170
180 GOSUB 200:PRINT "&";HEX$(PEEK(i),
2):PRINT:i=i+1:PEN 1:RETURN
190 GOSUB 200:PRINT "&";HEX$(PEEK(i)+
256*PEEK(i+1),4):PRINT:i=i+2:PEN 1:RE
TURN
200 PEN 2:WHILE POS(#0)<28:PRINT ".";
:WEND:PEN 3:RETURN
```

MUSIC COMPOSER - Superb fun for all the family. Suitable for the beginner or experienced musician. Clear graphics displays the music as it is composed. Full editing facilities. Compositions may be joined together and/or saved on tape/disk.

TAPE $17.95
DISK $29.95

See Page 72 for ordering information

DATABASE/MAILING LIST
Easy to use and flexible electronic card index. Up to 50 items per record. Up to 360 records. Fast sort and search facilities. Simple label printing routine with multiple copy facility. Large files may be split into smaller files.

TAPE $29.45
DISK $41.45

EASIDATA III is a completely revised and vastly improved version of our machine-code database. This new version includes: sorting on up to 9 levels at any one time; searching on any combination of fields; a powerful report generator allowing you to produce reports in the layout you require; facilities to restructure your database at any time.

TAPE $41.45
DISK $53.45

EASIWORD is an incredibly fast and powerful professional Word Processor. It is easy to use and has all the facilities you would expect of the modern office Word Processor. There is a print module which will allow you to set up 52 of your own print commands, customizing **EASIWORD** for use with your own printer.

TAPE $37.95 DISK $49.95

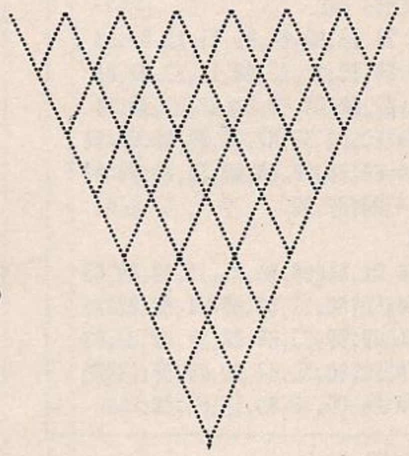
Strategy Software

EASIDATA II - The ultimate database program for your CPC464. Up to 32 items per record. Up to 1,000 records. Full search facilities, including wild card. Records automatically inserted in correct order. Built-in report generator allows simple word processing with mail merge and label printing.

TAPE ONLY $29.95

Amstrad Analysis

Trees



Analysed by
Trevor Roberts

THIS month we use our Amstrad to draw trees. Starting with one point, or node, at the bottom of the screen we draw lines to the left and right.

The ends of each of these lines also split into two and so on. At each split, the changes in the x and y coordinates are constant, resulting in a regular pattern.

It's a simple idea, but not particularly easy to program. Trees shows one method. Can you make it more efficient?

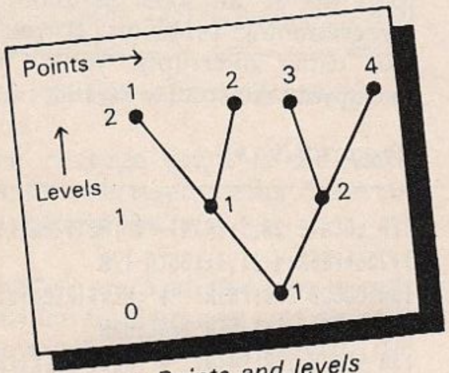


Figure I: Points and levels

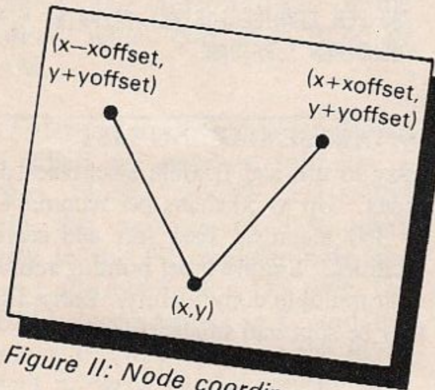
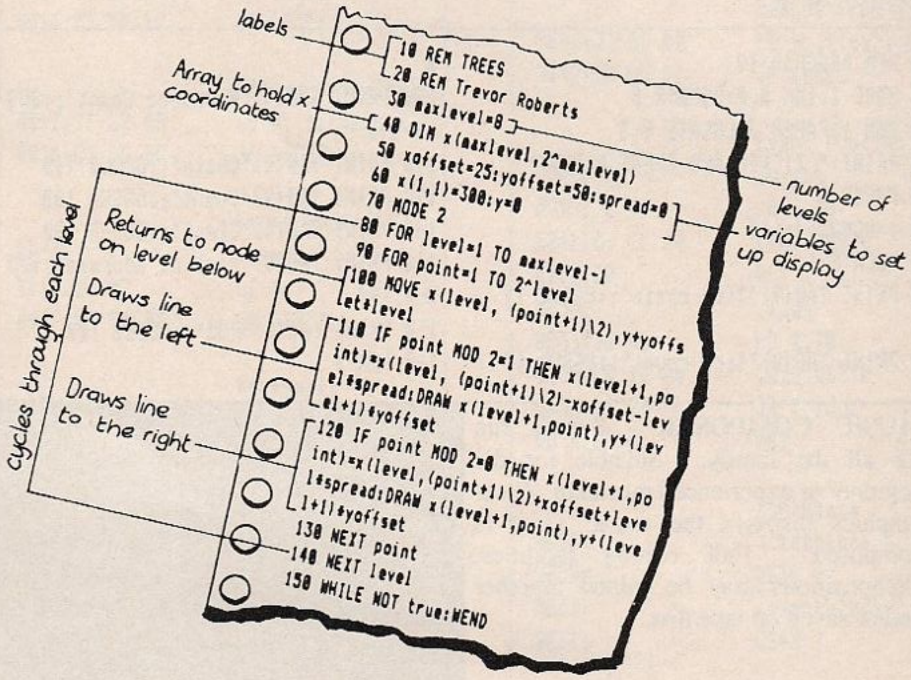


Figure II: Node coordinates


labels
Array to hold x coordinates
number of levels
variables to set up display
cycles through each level
Returns to node on level below
Draws line to the left
Draws line to the right

```
10 REM TREES
20 REM Trevor Roberts
30 maxlevel=8
40 DIM x(maxlevel,2^maxlevel)
50 xoffset=25:yoffset=50:spread=0
60 x(1,1)=300:y=0
70 MODE 2
80 FOR level=1 TO maxlevel-1
90 FOR point=1 TO 2^level
100 MOVE x(level, (point+1)\2),y+yoffs
et*level
110 IF point MOD 2=1 THEN x(level+1,po
int)=x(level, (point+1)\2)-xoffset-lev
el*spread:DRAW x(level+1,point),y+(lev
el+1)*yoffset
120 IF point MOD 2=0 THEN x(level+1,po
int)=x(level,(point+1)\2)+xoffset+leve
l*spread:DRAW x(level+1,point),y+(leve
l+1)*yoffset
130 NEXT point
140 NEXT level
150 WHILE NOT true:WEND
```

10,20 REMS that identify the program.

30 The variable *maxlevel* selects the number of levels (each line splits at a level). Have a look at Figure I to see how the levels and the number of points on them are related.

40 Horribly, this sets up a two dimensional array to hold all the possible x coordinates of the nodes. This is terribly wasteful as not all the levels have the maximum number of points. This array makes room for lots of non-existent points. Can you think of a better way of doing things?

50 Holds values for the changes in x and y coordinates for each succeeding level. *spread* just allows a little bit of variance in the way the lines split. Try giving it non-zero values and see what happens. Also vary *xoffset* and *yoffset*. Why not have them changing randomly?

60 Sets the coordinates for the initial point on level 0.

80-140 Form a FOR ... NEXT loop which cycles once for each of the levels. Each time the coordinates for all the nodes on that level will be calculated.

90-130 Make up a second loop which does the job of working out all the coordinates of the nodes of a level and draws the lines joining them. If you look at Figure I you should see that there are 2 to the power of *level* points on each level.

100 Ensures that the graphics cursor returns to the appropriate node. Leave it out and see the result.

110 Draws the line to the left from the node. Looking at Figure I you'll see that all the odd numbered points are on the left hand of a split. The MOD checks this and draws a line to it from the node below.

120 Deals with the right splits. Figure II shows how the coordinates relate.

150 An endless loop that keeps the ready message at bay.

Tame your loops with a crafty counter

By PETE BIBBY

Observant readers may have noticed that Mike Bibby has been replaced by Pete Bibby as the author of this series. It's not a misprint, nor a sign of a putsch at Computing with the Amstrad.

It's more a sort of negative nepotism. Mike's done the first six months articles and handed them over to me with a "now get out of that!"

The result is that I've been reading through the back issues trying to find out what he's covered. And very interesting it is. So, from now on I'm assuming that you know all about CLS, NEW, LIST and PRINT and that if I tell you to GOTO you won't take offence.

You should know your RUNs from REMs, and be able to use INPUT WHILE WENDing your way round loops and taking the TAB LETs as usual. IF you understand all that, THEN you shouldn't have any problems following the rest of the series.

I'll be sticking to Mike's easy-to-follow style and the emphasis will still be on a hands-on approach. And that's where you come in.

Don't just read about it, try it out on the micro. And don't stick to the programs I give you. Try varying them or making up your own. And don't take anything I say for granted, test it out in practice. If you want to know "What if..." try it out on the Amstrad.

And now to begin. Type in Program I and see what it does.

```
10 REM Program I
20 counter=1
30 PRINT counter
40 LET counter=counter+1
50 IF counter <=25 THEN
   GOTO 30
```

It's not exactly the wold's most fascinating piece of programming but it does cover some important points. With what you've done already you shouldn't have any difficulty in seeing how it works.

Line 20 sets up the variable *counter*, giving it the value 1 while line 30 displays it. Line 40 adds one to *counter* storing the result back in *counter*. This means that *counter* is now one bigger than it was when it was displayed.

Now comes the clever bit. The IF of the next line checks to see if *counter* is less than or equal to 25. If it is, the GOTO after the THEN sends the program back to line 30 and the whole proocess begins again.

When *counter* is greater than 25 the loop comes to an end. Technically the program is said to drop out of the loop. It then goes on to process the next line. In this case there isn't one, so things grind to a halt.

The program is an example of what's known as a conditional loop, the condition being that *counter* must be less than or lequal to 25 for the loop to cycle. The result is that the numbers 1 to 25 are printed out and then the program ends. Can you guess the value of *counter* when the program ends? Try adding:

```
60 PRINT "At the end
counter is "counter
```

and you'll find that the result is 26. You may have thought that it should be 25 but a quick look at the condition of line 50 should disabuse you.

As the condition is less than or equal to 25, when *counter* is 25 the program will still obey the GOTO. It loops back to line 30 and prints and so *counter* now has a value of 26 when it comes to the test of line 50. As 26 is greater than 25 the test is failed, the condition is false and the GOTO ignored. The program drops out of the loop and, finding no more lines to process, ends.

Try changing line 50 to:

```
50 IF counter <25 THEN GOTO 30
```

or

```
50 IF counter=25 THEN GOTO 30
```

and try to figure out what's happening. As you can see, seemingly small changes in the conditions of loops can have big effects on the way the program works.

You'll find that a lot of the problems you get when you use IFs and GOTOs to create loops lie in the use of the wrong conditions. You have to get them dead right which, in more complicated programs isn't always easy.

Now having read all that lot you should have no difficulty in understanding why Program II says HELLO 25 times.

```
10 REM Program II
20 counter=1
30 PRINT "HELLO"
40 LET counter=counter+1
50 IF counter <=25 THEN GOTO 30
```

Program II

Use your knowledge to produce five HELLOs, then five thousand. Or maybe three hundred and five. Should the condition be less than 306 or less than or equal to 305 or will both do the job or will neither? Try it and see. When you've finished playing around, run Program III.

The output from this is exactly the same as that from Program I, yet it's one line shorter. How is this done? The answer lies in the FOR of line 20 and the NEXT of line 40. These form what is known as a FOR...NEXT loop.

All the program lines that come between the FOR and the NEXT

```
10 REM Program III
20 FOR counter=1 TO 25
30 PRINT counter
40 NEXT counter
```

Program III

that follows it become part of the loop. In fact they're known as the body of the FOR...NEXT loop. And, as we've come to expect with the loops that we've already dealt with, it cycles over and over.

Each time round the lines making up the body of the loop are preformed. In this case there's only one line sandwiched between the FOR and the NEXT. This is line 30 which prints the value of *counter*.

Obviously, the value of *counter* is varying as the loop progresses but how? Previously we've always had something like:

```
counter=counter+1
```

but there's nothing like that in program III. And how does the loop know when to finish? And, while we're at it, what's the initial value of *counter*?

The answers lie in line 20 where the FOR is followed by:

```
counter=1 TO 25
```

It's this line that tells the Amstrad the initial value of *counter* which, in this case, is one. As well as this it automatically adds one to it's value each time round the loop and also determines how many times it will cycle. Let's look at it more closely.

As we've seen, the FOR and NEXT bracket the body of the loop, the lines that are to be repeated, Notice that this case it's *counter*. This is grandly known as the loop control variable, and it's this that decides how many times the loop will cycle.

The control variable starts out at the value assigned to it after the FOR and each time round the loop it is increased in value by one. The loop stops when the control variable exceeds the value specified by the figure after the TO.

In other words, the control variable acts as a counter. If we wanted to do something 10 times we could use our fingers as a control variable. Looking at it as a FOR...NEXT loop, we'd probably have something like:

```
FOR finger=1 TO 10
do something
NEXT finger
```

By the time we've used all 10 fingers we'll have done whatever it was we were doing ten times. The finger has acted as a control variable ranging from 1 TO 10.

In Program III we have the Line:

```
FOR counter=1 TO 25
```

This tells the Amstrad that it is to repeat all the lines up until the following NEXT. The loop control variable is to be counter, it's initial value will be 1 and each time round the loop this is to be increased by one. The loop is to cycle until *counter's* final value is greater than 25.

The first time round the loop counter is 1. Then the NEXT sends the program back to the FOR and *counter* becomes 2, then 3 as the loop progresses and so on until it's 25.

Now when the NEXT sends the program back to the FOR the 1 that is added to *counter* gives it the value 26. This exceeds the limit put on it by the TO so this time the loop is skipped, the Amstrad going to the line after the NEXT. In this case there isn't one, so the program finishes.

The result is that all the numbers between 1 and 25 are printed, as the loop cycles FOR values of *counter* 1 TO 25. Try adding:

```
50 PRINT "The final value of counter is "counter
```

and you'll see that it really has a final value of 26.

Compare Program I with Program III. I think that you'll agree that Program III is a lot clearer. With the FOR...NEXT structure it's easy to see that the loop will cycle 25 times. With Program I you're never quite sure if it's going to be 24

```
10 REM Program IV
20 FOR counter=1 TO 25
30 PRINT "HELLO"
40 NEXT counter
```

Program IV

or 25 times. So the lesson is that FOR...NEXT loops make things a lot clearer. They're also a lot easier to adjust then loops cobbled together with GOTOs.

Take a look at Program IV which does the same job as Program II but uses a FOR...NEXT loop.

You still get your 25 HELLOs on screen just as with Program II. Notice, however, how easy it is to get 500 or 5,000 of them. And to get 305 all we do is change one line.

```
20 FOR counter=1 TO 305
```

gives us what we want. As you can see it's a lot easier than messing about with Program II.

FOR...NEXT loops are not only simple, they're also powerful. Take a look at Program V and you'll see what I mean.

Again the loop cycles 25 times, but

```
10 REM Program V
20 FOR counter=1 TO 25
30 PRINT counter, counter*2,
   counter*counter
40 NEXT counter
```

now the line that makes up the body of the loop is different. It not only prints the value of *counter*, it also doubles it and shows its square. Try altering the line so that it gives the cube of *counter* or adds five each time round. Simple isn't it?

Before we leave Programs III and IV just see if you can spot any major difference between the way that they work. Don't worry if you can't, it's fairly obscure.

The answer is that Program III not only uses *counter* to control the number of loop repititions but also uses *counter* in the body of the loop. Program V does the same thing.

Program IV however doesn't use *counter* in the main body of the loop, it just uses it to keep track of the cycles. As I said, it's an obscure point but one worth bearing in mind.

If you use the loop control variable inside the main body of the loop be careful that you don't change it or you'll upset the loop's counting.

```
25 counter=counter+3
```

in Program III will show you what I mean. Although it's fairly obvious in this case, in longer programs it can occasionally be a problem. Beware!

So far all our FOR...NEXT loops have had the control variable going up in steps of 1 each time round the loop. It doesn't always

```
10 REM Program VI
20 FOR counter=0 TO 20 STEP 2
30 PRINT counter
40 NEXT counter
```

Program VI

have to be like this. You can make the change any number you want using the aptly-named STEP command. Program VI uses it to display all the even numbers between 0 and 20.

It's simple command to use. As you can gather from the program, the figure after the STEP decides how much the control variable is to be incremented at the end of each cycle. In this case *counter* is increased by 2 each time round.

What would happen if the STEP was 4? Try and figure it out before you attempt it on the micro.

Program VII shows what happens when the STEP is 3.

```
10 REM Program VII
20 FOR counter=0 TO 20 STEP 3
30 PRINT counter
40 NEXT
```

Program VII

There's no problem with most of the output. It starts at 0, and then goes up in threes until it reaches 18. But why stop at 18? If you add:

```
50 PRINT "The final value of
counter is "counter
```

to the end of the program you should see why.

When *counter* is 15 the NEXT adds 3 to it, making it 18 and then sends the loop back to line 20. Line 30 prints out the 18 and the program again comes to the NEXT.

Here *counter* is again increased by three, making it 21 and the program goes back to the FOR.

Now when the micro checks, *counter* is outside the allowed range (0 TO 20) so the loop is skipped. Hence 18 is the last number you see printed.

Try changing the STEP to 5 or 8 and see if you can explain the results.

You may have noticed that in the last program the NEXT of line 40 didn't have a *counter* by it but it still worked. From this you can see that you don't always have to label the NEXT with the loop control variable. Having said that I'd advise you to always do so, it makes debugging programs a lot easier.

All this is all right if you want the numbers produced by a loop to increase, but what if for reasons best known to yourself, you want the numbers to decrease in value?

Program VIII shows one way of doing this.

```
10 REM Program VIII
20 FOR counter=0 TO 15
30 PRINT 15-counter
40 NEXT counter
```

Here *counter* is still going from zero upwards by one each time, its range spanning 0 TO 15. However, instead of just printing out the value of *counter*, now we take it away from 15 each time round the loop. The first time round *counter* is 0 so 15 is displayed. The seond time *counter* has increased to 1 so 14 (15-1) appears with 13 (15-2) following it on the next cycle.

As counter ranges from 0 to 15, so the result of the subtraction goes from 15 to 0. Can you make it go down in twos?

Program IX shows an easier way of counting down.

```
10 REM Program IX
20 FOR counter=15 TO 0 STEP -1
30 PRINT counter
40 NEXT counter
```

The secret lies in the STEP parameter of -1. Each time round the loop -1 is added to *counter*. The result of adding -1 each time round the loop is that *counter* is decreased.

Notice that the figures on either side of the TO have changed to cope with this. They now go from an initial value of 15 to a final one of 0.

Try using other negative steps and see what happens. They're quite simple to use, but make sure that the start and finish values match the steps.

If you don't understand that try Program X and you'll see what I mean.

```
10 REM Program X
20 FOR counter= 2 TO 1
30 PRINT "That's silly!"
40 NEXT counter
```

Try:

STEP -1

or

FOR counter=1 to 2

to correct it.

By now you should be fairly at home with FOR...NEXT loops, so program XI should hold no fears for you. You should be able to figure out that the last number displayed will be 9.

```
10 REM Program XI
20 FOR counter=1 TO 10 STEP 2
30 PRINT counter
40 NEXT counter
```

Notice that when the STEP is positive the loop finishes when the control variable exceeds the limit set.

If you've any doubts:

50 PRINT "The last value is "counter

should convince you of the fact as 11 is certainly greater than the 10 after the TO.

However, when the STEP is negative as in Program XII things are different. The loop finishes when the control variable is less than the figure after the TO.

```
10 REM Program XII
20 FOR counter=10 TO 1 STEP -2
30 PRINT counter
40 NEXT counter
```

Again:

50 PRINT "The last value is "counter

added to the end of the program should convince you. There's no doubt that 0, the final value of *counter* is less than 1, the limit set in line 20.

Why not write your own programs using negative and positive steps? And why not use fractional values instead of the whole numbers we have been doing? And after all this, if you have any time left, can you explain Program XIII?

```
10 REM Program XIII
20 FOR outer=1 TO 3
30 PRINT "This is outer loop
"outer
40 FOR inner=1 TO 3
50 PRINT TAB(15) "This is
inner loop"inner
 60 NEXT inner
70 NEXT outer
```

What's happening here? All will be explained....

Like the parts of a Russian doll, each loop must be completely contained by the other if your program is to fit together properly. So ...

Don't get your NEXTs in a twist!

We've had a look at the way we can use FOR ... NEXT loops to repeat a body of code for a fixed number of times. We saw that we could use the same program lines over and over again by sandwhiching them between a FOR and a NEXT.

This not only increases the power of a program but also saves a lot of typing. Program XIV should hold no difficulties for you.

```
10 REM PROGRAM XIV
20 FOR row=1 TO 10
30 PRINT "#";
40 NEXT row
```

This is a very elementary use of the FOR ... NEXT loop formed by lines 20 and 40. The body of the loop is line 30. This prints a hash on the screen each time the loop cycles.

The apostrophe at the end of the line makes sure that the hashes are "glued" together, one after the other. Leave it out and see what happens.

The number of times the loop cycles is determined by the loop control variable *row*. This starts with a value of 1 and is incremented by 1 each time the program reaches a NEXT.

When row is eventually incerased to 11 the loop finishes, having cycled 10 times. The result is a line of 10 hashes stretched across the screen.

Don't just run the program and leave it at that, though. Try varying the control variable *row* to produce lines of 20 or 30 hashes. Notice that:

FOR row=10 TO 19

or:

FOR row=99TO 90 STEP -1

also produces 10 hashes. You don't always have to have 1 as the initial value of the loop control variable.

The main point to grasp is that every piece of code between the FOR and the NEXT is repeated. It doesn't matter if its a PRINT command or a LET or whatever. While the loop is still in operation everything inside the FOR ... NEXT's boundaries is performed repeatedly.

This is so even if it's another FOR ...NEXT loop that makes up the body of the loop. Program XV shows you what I mean.

Lines 30 to 50 should hold no mysteries, they're the same as the FOR ... NEXT loop of the previous program. As we saw before, that particular loop will cycle 10 times and produce a line of 10 hashes. However with Program XV we get

```
10 REM PROGRAM XV
20 FOR freshrow=1 TO 5
30 FOR row=1 TO 10
40 PRINT "#";
50 NEXT row
60 PRINT CHR$ (13)
70 NEXT freshrow
```

Program XV

five rows of 10 hashes. How has that happened?

The answer lies in the FOR ... NEXT loop formed by lines 20 and 70. By looking at line 20 we can see that this loop will cycle five times while the control variable *freshrow* takes values from 1 to 5.

Never mind about the stuff in the middle. For the moment just concentrate on that outer loop cycling five times.

Now, as we've seen before, everything inside a FOR ... NEXT loop is repeated as the loop cycles. So the body of the loop - that's everything between the FOR of line 20 and the NEXT of line 70- will be repeated five times.

However, instead of just being a PRINT command as in previous programs, the body of the loop is now another FOR ... NEXT loop. And, as we've seen this FOR ... NEXT loop wants to cycle 10 times. This is, in fact what happens.

Each time the outer loop, controlled by *freshrow*, repeats once the inner loop is performed in its entirety. That is, for each cycle of the outer loop, the inner loop, controlled by *row*, repeats 10 times producing the familiar row of 10 hashes.

As the outer loop cycles five times in all, and as the inner loop cycles 10 times for each time round the outer loop, we get five rows of 10 hashes.

This is an example of what is known as nested loops. In this case there is one FOR .. NEXT loop (lines 30-50) nested inside another (lines 20-70). Each time the outer loop cycles once the inner loop cycles for its full quota of repetitions.

Again, don't be content with the example program, try varying it. What would lines like:

```
20 FOR freshrow=10 TO 15
30 FOR row=10 TO 5 STEP -1
```

or:

```
10 FOR freshrow=10 TO 50 STEP 5
20 FOR row=15 to 21 STEP 2
```

produce? Did the number of rows of hashes and the number of hashes in each row tally with your expectations?

Notice the care you have to take with your control variables to avoid having more cycles of each loop that you really wanted.

Incidentally, can you see the point of line 60? It's there to undo the effects of the semi-colon at the end of line 40 when the program leaves the inner loop. It does this by producing what's known as a carriage return. Try leaving it out and see what happens.

Armed with our new-found mastery of nested loops we can turn back to Program XIII and see how it works.

```
10 REM PROGRAM XIII
20 REM
30 FOR outer=1 TO 3
40 PRINT "This is outer
loop  "outer
50 FOR inner=1 TO 3
60 PRINT TAB (15) "This is
inner loop" inner
70 NEXT inner
80 NEXT outer
```

Program XIII

As you can see, it consists of two loops, one nested inside the other. The inside loop is formed by lines 50 to 70 and has control variable *inner*. This takes values from 1 to 3 and each time it cycles the "inner loop" message is displayed.

The outer loop is formed by lines 30 and 80 and, unsurprisingly has control variable *outer*. This, too, ranges from 1 to 3. However, each time this outer loop cycles *it not only prints out a message but also performs all three cycles of the inner loop.*

By the time the outer loop has completed its quota of repeats the inner loop will have been through it's full quota three times.

Let's follow the program as it runs. Lines 10 and 20 are just REMs, which the Amstrad ignores. Line 30 tells the micro that it is starting a loop and that the variable *outer* is to be used to control the loop. Initially this has the value 1.

The program now goes on to line 40, and displays the message:

```
This is outer loop 1
```

Line 50 marks the start of another FOR ... NEXT loop. This has it's control variable *inner* taking values from 1 to 3. At this stage the Amstrad gives *inner* the value 1 and goes on to line 60. This results in the message:

```
This is inner loop 1
```

and the program goes on to the NEXT of line 70. Here *inner* is increased by 1 and the program goes back to the FOR of line 50. Since *inner* is within the limits set, control goes to line 60 and the message:

```
This is inner loop 2
```

appears on the screen.

Again the NEXT is met at line 70, *inner* is increased to 3 and the program goes back to line 50. Line 60 now prints:

```
This is inner loop 3
```

and the program comes across the NEXT of line 70 again. Now, however, when *inner* is increased to 4 the conditions of the loop are exceeded. The program drops out of the loop and reaches line 80. Here it finds the NEXT of the outer loop.

Now *outer* is increased to 2 and the program goes back to line 30. Line 40 displays:

```
This is outer loop 2
```

and line 50 takes the program back into the inner loop, producing the three "inner" messages as before.

Once this is done the program drops out of the inner loop and again encounters the NEXT of line 80, *outer* now becomes 3 and the program goes back to line 30 for the final time. When *outer* is 4 the program finished.

Don't worry if you don't grasp it in all its complexity. Nested loops take a bit of getting used to, but once you've played around with them for a while they become second nature.

All you have to grasp is that for each single cycle of the outer loop the inner loop goes through all its cycles. Adding a line like:

```
55 PRINT "outer="outer :"inner="inner
```

might make things clearer.

While we're still with Program XIII I should point out that you don't need to put the control variables after the NEXT. Program XVI shows this.

```
10 REM PROGRAM XVI
20 FOR outer=1 TO 5
30 PRINT "This is outer
loop"outer
40 FOR inner=1 TO 4
50 PRINT TAB(15)"This is inner
loop"
60 NEXT
70 NEXT
```

Program XVI

However, while the Amstrad may be clever enough to keep track of things, you might not be. Take my advice, and always put in the variable, at least until you've got your grogram working properly.

Notice that Program XVI is a variant of Program XIII. There have only been two minor alterations to parts of lines, but look at the difference in output. As you might appreciate, nested FOR ... NEXT loops are both flexible and powerful.

So far we've only nested one FOR ... NEXT loop inside another. It's perfectly possible to have three or more loops nested, one inside the other rather like those Russian dolls on the front of spy stories. They work in much the same way but are easier to understand in practice than in theory. Add:

```
45 FOR middle=1 TO 3
47 PRINT TAB (8)"This is
middle loop "middle
75 NEXT middle
```

to Program XIII and you'll see what I mean. And notice the vast increase in output for just three extra lines. As I said, these nested loops are powerful.

There's one thing to be wary of when you messing around with nested loops. Don't get your NEXT's in a twist. Program XVII shows this very well.

```
10 REM PROGRAM XVII
20 FOR outer=1 TO 3
30 PRINT "This is outer loop
"outer
40 FOR inner=1 TO 3
50 PRINT TAB (15)"This is inner
loop" inner
60 NEXT outer
70 NEXT inner
```

Program XVII

When you run this you'll get the error message:

```
Unexpected NEXT in 70
```

glaring at you from the screen. It won't take you long to realize that you've confused the computer by getting the control variables in lines 60 and 70 the wrong way round. They should have been:

```
60 NEXT inner
70 NEXT outer
```

If you really want to test your understanding of nested loops can you explain why changing the last lines to:

```
60 NEXT outer
70 NEXT outer
```

should produce the message:

```
This is outer loop 1
```

before the error message?

But enough of what can go wrong, it's too theoretical. After all, you and I don't make mistakes, do we? We're too busy doing important things like producing triangles of hashes. Have a look at Program XVIII.

```
10 REM PROGRAM XVIII
20 FOR length=1 TO 10
30 FOR row=1 TO length
40 PRINT "#";
50 NEXT row
60 PRINT CHR$ (13)
70 NEXT length
```

Program XVIII

While the output is nothing to rave about, the program does contain some interesting points. By now you should be able to recognize the two sets of nested loops and the CHR$(13) of line 60 should hold no fears.

The outer loop with control variable *length* ranging from 1 to 10 is fairly straightforward. The inner loop is rather different. Here the control variable *row* goes from 1 to *length*. Now *length* is changing each time round the outer loop.

The first time round *length* is 1, so the control variable of the inner loop goes from 1 to 1. This results in the first, solitary hash.

The next time round the outer loop, *length* is 3, so the inner loop produces a trio of hashes as *row* goes from 1 to 3. By the time that *length* is 10, 10 hashes are displayed.

The point to grasp is that the number of repetitions of the inner loop depends on the control variable of the outer loop. The loops are not only nested, the outer controls the inner.

Can you alter the program so that it produces the same triangle but upside down? Program XIX show how it's done.

```
10 REM PROGRAM XIX
20 FOR length=10 TO 1 STEP -1
30 FOR row=1 TO length
40 PRINT "#";
50 NEXT row
60 PRINT CHR$(13)
70 NEXT length
```

Here *length* is descreasing each time round the outer loop, hence the reducing number of hashes in each line.

Once you're sure that you understand the last two programs have a go at producing the mirror image of Program XIX's hashes.

It's not quite as simple as the previous programs, but if you remember how LOCATE works you should have no problem seeing how it works. Program XX shows how it's done.

```
10 REM PROGRAM XX
20 CLS
30 FOR row=1 TO 10
40 FOR length=1 TO row
50 LOCATE 11=length, row
60 PRINT"#"
70 NEXT length
80 NEXT row
```

Program XX

Now can you turn it upside down? The alterations are:

```
30 FOR row=10 to 1 STEP-1
50 LOCATE 11-length, 11-row
```

And that's it for this month. I'll leave you to mess around with nested loops. Keep on trying to produce patterns of hashes.

While it's hardly the most important use of your Amstrad, you'll be amazed at how it will increase your grasp of nested loops.

And when you get tired of that, can you figure out what's happening in ProgramXXI?

```
10 REM PROGRAM XXI
20 CLS
30 hash$="#"
40 FOR row=1 TO 10
50 LOCATE 11-row, row
60 PRINT hash$
70 hash$=hash$+"#"
80 NEXT row
```

It produces the same output as Program XX, but there's only one loop instead of two. How's that happened? The answer lies with string variables, and we'll be looking at them next time.

TASWORD 464
THE WORD PROCESSOR

WE HAVE A LIMITED NUMBER LEFT AND ARE SELLING THESE AT ONLY $36.95 ON TAPE AND $48.95 ON DISK. MANUAL IS COMPLETE WITH ADDENDUM TO ENABLE TRANSFER TO DISK. HURRY! THESE WON'T LAST.

TASWORD
464

The Word Processor

A
Tasman Software Program
for the
Amstrad CPC 464

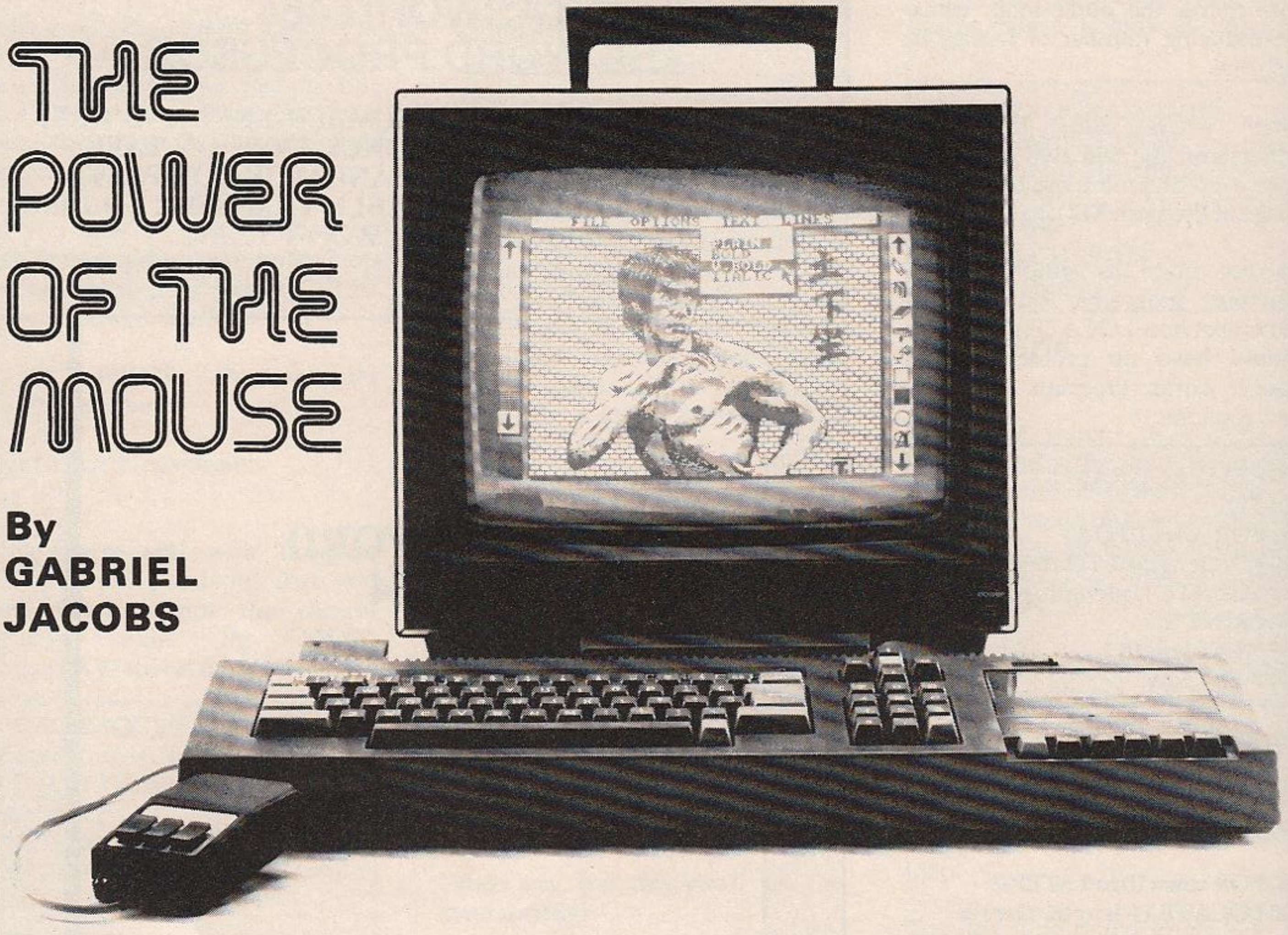
Strategy
Software

BOX 11, BLACKMANS BAY, TASMANIA 7152

See order Form on Page 72 for further details

THE POWER OF THE MOUSE

By GABRIEL JACOBS



Product: AMX Mouse. Price: $ 199 for cassette, disk, Mouse and interface.

If you still think a mouse is just a furry mammal you haven't been keeping up with hardware developments over the last few years. A mouse is also a small plastic box with one or more buttons on top, and a tracker-ball mechanism underneath.

It is linked to a computer via a cable - the mouse's tail, if you like - and the general idea is that it can be used for many operations instead of the keyboard.

When the mouse is pushed along a flat surface the tracker-ball rotates two disks set at right angles to each other. A light shines though perforations in the disks, and the pulsed signals are received by an encoder.

The pulses are sent to the computer, where they are interpreted as the length and direction of the tracker-ball movement, and a symbol - an arrow or a pencil - moves around the screen accordingly.

The buttons are there to executecommands, for example to make a selection from a menu when the symbol is pointing to an item on it, or to pick up a shape and move it elsewhere on the screen.

When the first mouse gingerly crept into the marketplace with Apple's fabulous but expensive Lisa (a machine which, like Concorde, was too far ahead of it's time to be a commercial success), it was thought by many to be something of a gimmick.

But Apple stuck to their guns and put it out as standard with the Macintosh. The species came to be accepted as a desirable business tool, and has proliferated, as you would expect mice to do, on a number of machines.

Executives which for one reason or another are averse to the keyboard seem to love the little creatures. Good data entry requires accurate, rapid touch-typing, but many executives need do no more than select a few options, plus maybe a bit of cutting and pasting. A mouse is ideal. Perhaps equally important, it's also fun.

The AMX Mouse will interface with the Amstrad 464, 664, or 6128, taking its power from the monitor. It comes with simple pushconnectors - no hard-wiring is involved - a range of software on cassette which is easily transferred to disk with the routines provided, and a fully documented and readable manual.

As a piece of hardware the product is not quite as well designed as some of its high-class competition. It does not feel as solid, nor does it fit as comfortably in the hand as, say, MicroSoft's No. 5 mouse for the IBM PC.

But it is reasonably well made, and in particular its three buttons, Execute, Move and Cancel, react to the lightest of touches, while having just enough travel to allow a finger to rest on them without activating them unintentionally.

Provided that the surface it moves over is fairly even and not too highly polished it produces a smooth and responsive movement of the screen pointer.

A mouse can only be used after loading the special software it requires, in this case AMX Control. It sets up an extended set of bar commands, then sits in memory allowing other programs to be loaded alongside it.

Some of the commands are concerned with pointer control, and include a |STEP command which allows you to alter the number of pulses generated for a given amount of movement. This can turn the mouse into a precise pointing device, far more accurate than light pens or digitizers in the same price bracket.

As a bonus, AMX Control also gives you a set of commands which offer quick and easy ways of improving screen presentation. With a single command you can cover the screen with a desk-top background. You can produce overlapping windows which the mouse will treat as separate screen areas, changing its pointer symbol as it enters them.

Most important of all, you can load and display previously designed icons - graphic representations of ideas, such as a tiny picture of a waste bin which might be used as the symbol for deleting files. When the pointer symbol is moved to an icon the Execute button will generate the code the icon has been set to represent.

'The new release of AMX Art is in glorious color, though you're still stuck with shades of grey on the printer'

All the AMX Control commands can be incorporated into your own basic or machine-code programs, and two of them can also be used very simply with commercial software. One converts the movement of the tracker-ball into cursor codes rather than graphics coordinates.

The other can be used to re-configure the buttons to return values different from the defaults - Return, Copy and Delete, for instance, or codes corresponding to the function keys. The commands cannot be guaranteed to work with every commercial program, but there were no problems with those I tested, including Tasword.

AMX Control must also be installed before running the customized application software included with the mouse, that is to say AMX Art, Pattern Designer and Icon Designer. In many ways this software is the most impressive part of the package, and certainly the one new users will be most concerned with.

● AMX Art is a graphics program, driven by a series of pull-down mouse-controlled menus, which allows you to create pictures on the screen, save them to tape or disk, and dump them to an Epson or compatible printer.

When the AMX Mouse was first made available for Amstrad machines late last year, the graphics were monochrome. The new release of AMX Art is in glorious color, though of course you're still stuck with shades of grey at the printer.

The drawing area, which takes up most of the screen, is flanked on the left by a scrolling window from which you can choose any of 32 patterns in the wide range of colors offered. This is like having an artist's palette at your disposal, but with paint that can automatically produce hatching, dots, and stripes.

You simply move the pointer over the pattern you want, and press Execute to confirm your choice. The current pattern is displayed in a box at the bottom of the window, and can then be used in the drawing area. But first the drawing mode has to be selected.

This is done by moving the pointer to another scrolling window on the right-hand side of the drawing area which contains icons representing the various modes available - Spray, Fill, Pencil, Rubber, Circle, Box, Text, and the like.

Having selected the mode, again with the Executive button, and re-entered the drawing area, the pointer symbol changes to conform with the current setting. So if you have chosen Spray the pointer symbol becomes a little spray gun, and you can spray the current pattern anywhere in the drawing area to your heart's content.

It is a simple matter to create varied and interesting designs. Within a few minutes of loading the program for the first time I had concocted a colorful work of abstract art, topped with some self-congratulatory text in large bold italics, the whole set in a background of a delicate shade of rose pink.

I admit that none of my subsequent attempts turned out to be either more aesthetically pleasing than that first one, or recognizable as true-to-life pictures, but that reflects my own artistic limitations rather than the quality of the program.

However, anyone with even a scraping of talent will be able to create effective paintings which would be much harder to achieve with real pencils, paint, brushes and paper.

With a judicious combination of the Spray Gun, the Paint Roller and the Pencil, complex and detailed designs can be built up, changed and re-changed until they are satisfactory.

There is even a Zoom facility which magnifies a selected area of the screen so that it can be edited pixel by pixel.

● Pattern Designer is meant to be used in conjunction with AMX Art, and allows you to create your own set of up to 32 patterns, again by turning individual pixels on or off.

Using the mouse and a series of icons you can draw patterns on a blank 16 x 16 grid, edit existing ones, see what they will look like when reduced in size to the normal screen display, move half-baked ideas to a Scratchpad, put the ones you're happy with into a temporary window before saving them, in fact do just about anything with them.

● Icon Designer works on exactly the same principles, but has some extra facilities for disk users, allowing greater flexibility in loading and saving. The program is meant primarily for designing icons to be incorporated into your own routines, so for some users it will be the most important module.

'anyone with even a scraping of talent will be able to create effective paintings which would be much harder to achieve with real pencils, paint, brushes and paper'

Icon-driven programs are just about the most effective way of providing a friendly interface between man and computer - the AMX software itself is proof enough of that. Writing this kind of interface is normally a complex task, but with the AMX Mouse and Icon Designer, it really is a cinch.

Many manufacturers give prominence to the small footprint of their computers - the amount of room they take up on a desk- leaving that much more space available for documents, books, reading lamps, and other paraphernalia.

Yet the manufacturers of mice (or mouses) meant to be used with those same machine proclaim the enormous benefits of their product, but admit the little footprints happen to require a fairly large uncluttered desk space. You have to decide where your priorities lie.

If desk space is not a serious problem the AMX Mouse is a superb buy.

At $199 it is not cheap in relation to the cost of your machine, but it is good value for money as mice go and the bundled software is magnificent.

I can't think of another add-on in the same price range which will give you as much enjoyment combined with as much practical potential.

Added to this is what is almost certain to come. The AMX Mouse for the BBC Micro has been around for some time, and has generated a good deal of software, from extra utilities to mouse-driven music editors, both commercially and as listings in magazines.

We can already see the beginnings of this on Amstrad machines. MiniOffice 2 and Electric Studio's Light Pen software have been configured to be used with the AMX Mouse. And at the time of writing, AMX are putting the finishing touches to some new utilities, including extended commands for AMX Art such as Rotate, Shrink, Stretch and Mirror.

The AMX Mouse's future on the Amstrad looks as if it will be as rosy as that first picture I created.

Psst...

Into Music on your Amstrad?

Check these out:

Musico

Available on tape only, Musico is an easy to use system for creating 3 part music and sound effects. The results of your musical genius may be used in your own programs by saving your music file and then merging it with any Basic program of your own design.

Drumkit

Drumkit is a percussion instrument simulator with eleven standard percussion sounds plus three user-defined sounds. Drumkit uses the Amstrad sound chip and no extra hardware is required.Other features include:

- up to 10 rhythm sequences
- sequences can be joined to-gether
- icon control of functions
- 3 channel support
- rhythm grid
- save/load to/from cassette

Drumkit is only available on tape for CPC 464 models.

Easy Music

For disk users Easy Music combines Musico and Drumkit together with the ease of disk operation for use on all CPC computers with disk drives

Sole Australian and New Zealand distributor - Strategy Software See Page 72 for ordering info.

Just bluffing ..

... but Aleatoire reveals there can be more to random choice than you might expect

MANY books have been written about bluff and some attempts have been made to treat the subject mathematically - this approach comes under the General Theory of Games and reveals a number of apparent paradoxes.

For example, given limited time or resources, it is often more effective to confuse your opponent rather than try to improve your own position, and that the *best* way to confuse him is to make random decisions.

To give a working example, consider the following very simple card game for two players. Remove 11 cards from the pack - usually the ace to jack of a suit. Shuffle them and deal five cards each. The object is to guess the identity of the undealt card.

The players pick up their cards and play alternately. At each turn a player can either guess the hidden card - winning or losing immediately - or he can ask his opponent if he has a certain card. If the opponent has that card then he must tell the truth. To cheat is to lose. Irrespective of the answer, it is then the opponent's turn to guess or ask.

Now if you always try to gain information, that is always ask for a card not in your hand, then you will not be very successful in this game. You must occasionally bluff, otherwise whenever asked for a card he does not have your opponent will know the card and win.

However you cannot afford to bluff too often because, providing your bluffs are ignored, you are learning nothing about your opponent's hand.

So you must bluff some of the time and you must assume that your opponent is doing the same.

This game was invented by R. Isaacs back in 1967. He showed that whether you bluff or not, and whether you should suspect your opponent of bluffing, are both functions of the number of unknown cards each player holds.

These optimal strategies are encoded in the two data tables in Program I which are read into the two arrays *ibluff* and *hebluffs*.

ibluff gives the percentage of times the program should bluff in the 25 possible situations. Assume it has five cards and the opponent has four cards - this happens when the program ignores a possible bluff by its opponent at his very first turn.

At line 380 the number 27 is picked up and if 27 >=RND(1)*100, that is 27 per cent of the time, then the program will bluff - ask for a card in its hand not previously mentioned.

If, however, 27 < RND(1)*100, that is 73 per cent of the time, then the program goes to line 430 and makes a genuine guess at a card the opponent may have.

The second array, *hebluffs*, works in the same way and is used on the occasions when the program does not have the requested card. This is more complicated because, if it assumes a bluff and the opponent only has one card left, then the program must also assume it knows the hidden card.

There are two other situations where the program will guess the card.

● It didn't bluff and got "no". It waits for its turn and then says it must be the last card it asked for. Note that it doesn't remember what it was.

. The program has no cards left - this is a forced risk because if the program doesn't make the guess then the opponent will win at his next turn.

Type in the program including the debug lines 340 to 370. When you have got it working note that you enter a *negative* number to make your guess and, if the program always goes first, it should win about 55 per cent of the games providing that you mix your bluffs in the same way as the program.

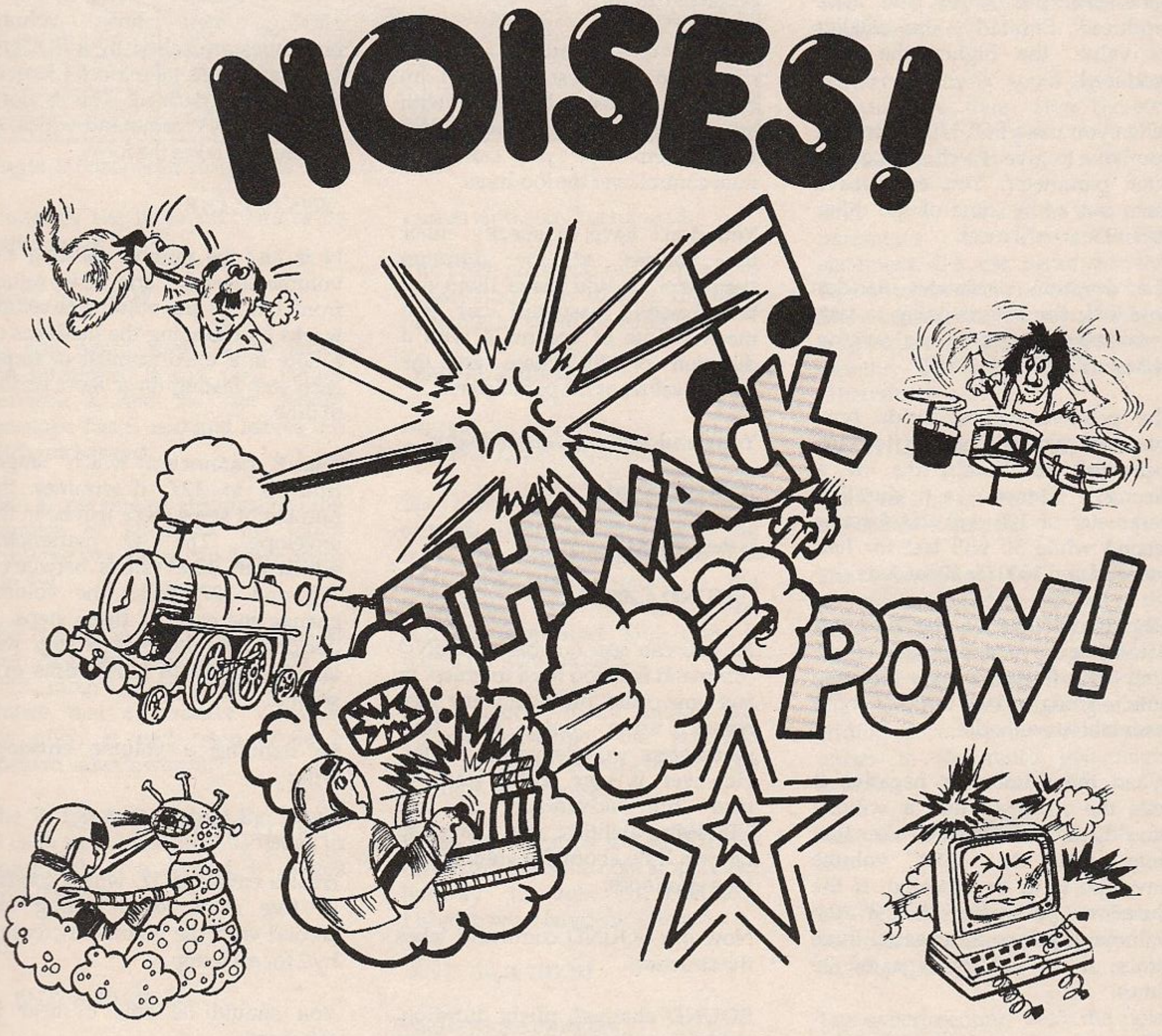
If, however, you are always honest you should beat the program 60 per cent of the time. This is because it is set to assume that you bluff occasionally. An interesting improvement is to make the program observe how often you bluff and let it modify the *hebluffs* array accordingly.

If you never bluff then this array should eventually become full of 100s - if you always bluff then the *hebluffs* array contents should all drop to zero.

Exactly how you observe and modify is up to you, but it is a nice example of a learning program that not only alters its own behavior but should also make you change yours.

N.B: Enter a lower case "y" for "yes", anything else means you don't have the requested card. ☞

```
10 REM PROGRAM I
20 REM Aleatoire
30 REM (c)Computing with the Amstr
ad
40 MODE 1
50 DIM ibluff(5,5),hebluffs(5,5),d
eck(11):true=-1:false=0:GOTO 80
60 guess=INT(RND(1)*11+1):IF deck(
guess)<his GOTO 60
70 deck(guess)=-deck(guess):RETURN
80 FOR his=1 TO 5:FOR mine=1 TO 5:
READ ibluff(his,mine):NEXT mine:NE
XT his
90 DATA 33,50,50,56,58
100 DATA 26,33,37,40,43
110 DATA 19,26,29,31,33
120 DATA 16,21,23,25,27
130 DATA 12,17,19,21,22
140 FOR his=1 TO 5:FOR mine=1 TO 5
:READ hebluffs(his,mine):NEXT mine
:NEXT his
150 DATA 50,50,41,38,33
160 DATA 33,33,28,25,22
170 DATA 38,32,27,24,21
180 DATA 32,30,26,23,21
190 DATA 32,29,25,23,21
200 wins=0:games=0:mine=1:his=2:it
=3
210 FOR i=1 TO 11
220   deck(i)=it
230 NEXT i
240 PRINT"You have cards    ";
250 FOR i=1 TO 10
260   card=INT(RND(1)*11+1): IF de
ck(card)<>it THEN 260
270   deck(card)=mine

280   IF i>5 THEN deck(card)=his:P
RINT card;" ";
290 NEXT i
300 PRINT
310 win=false:bluffing=false:hehad
it=true:mycards=5:hiscards=5
320 IF RND(1)<0.5 THEN 470
330 PRINT
340 FOR i=1 TO 11
350   IF ABS(deck(i))=his THEN PRI
NT i
360   IF deck(i)=-mine     THEN PRI
NT "     ";i
370 NEXT i
380 IF ibluff(hiscards,mycards)<RN
D(1)*100 THEN 430
390 bluffing=true          :REM Ask
 for one of my own cards
400 guess=INT(RND(1)*11+1): IF dec
k(guess)<>mine THEN 400
410 deck(guess)=-mine
420 mycards=mycards-1: GOTO 440
430 bluffing=false: GOSUB 60: hisc
ards=hiscards-1
440 PRINT "Have you got card ";gue
ss;
450 INPUT; answer$:PRINT
460 IF LEFT$(answer$,1)="y" THEN h
ehadit=true ELSE hehadit=false
470 INPUT"Your turn ";request
480 IF request>0 THEN 510
490 IF ABS(deck(-request))<>it THE
N win=true:REM He guessed wrong
500 GOTO 740
510 IF NOT bluffing AND NOT hehadi
t THEN 700
520 IF ABS(deck(request))<>mine TH
EN 600
530 PRINT"Yes I have card ";reques
t
540 IF deck(request)>0 THEN mycard
s=mycards-1
550 deck(request)=-mine
560 IF mycards>0 THEN 690
570 GOSUB 60: PRINT"I must guess "
;guess
580 IF ABS(deck(guess))=it THEN wi
n=true:REM I guessed right
590 GOTO 740
600 REM Is he bluffing?
610 IF hebluffs(hiscards,mycards)<
RND(1)*100 THEN 650: REM Assume bl
uff else
620 PRINT"You are NOT bluffing. I
guess that is the card"
630 IF ABS(deck(request))=it THEN
win=true:REM He wasnt bluffing
640 GOTO 740
650 REM I think he is bluffing
660 PRINT"No, I do not have card "
;request
670 deck(request)=-ABS(deck(reques
t))
680 hiscards=hiscards-1
690 IF bluffing OR hehadit THEN 71
0
700 PRINT"It should be the last on
e I asked for":win=true:GOTO 740
710 IF hiscards>0 THEN 330: REM Do
nt know yet
720 GOSUB 60:PRINT"The card  must
be ";guess
730 IF ABS(deck(guess))=it THEN wi
n=true ELSE PRINT"You just bluffed
 me!"
740 i=0
750 i=i+1: IF ABS(deck(i))<>it THE
N 750
760 PRINT"The card  was the ";i:ga
mes=games+1
770 IF win THEN wins=wins+1:PRINT"
I win" ELSE PRINT"I lose"
780 PRINT"My average is ";wins/gam
es*100;"%":PRINT:GOTO 210
```



Part V of NIGEL PETERS' series on coaxing sounds from the CPC464

WE'VE covered a lot of ground in the first four articles of this series, so this month we'll take a little time to reconsider what we've learned.

The SOUND command is the one that tells the Amstrad to make a noise. Not only that, it controls the type of noise made, how long it lasts and how loud it is.

At its simplest, the structure of the SOUND command is:

SOUND channel, pitch, duration, volume.

You can hear what it sounds like by keying in:

```
SOUND 1, 200, 100, 7
```

By comparing it to the basic structure, you'll see that the channel parameter is 1. The Amstrad has three sound channels - channel A, channel B, and channel C. Each can make a noise independently of the other, so you can have three different notes playing at the same time.

In the above example the channel parameter was 1, so channel A made the sound. If it had been 2, channel B would have produced the noise, while 4 would select channel C.

There is a lot more to the channel parameter (which we'll come to next month) but for the present we'll just stick to the values 1, 2 and 4 selecting channels A, B and C respectively.

The next parameter we come to controls the pitch, deciding how high or low the note can be. It can take whole number values from 0 to 4095.

The larger the value of the pitch parameter the lower the note produced. Similarly, the smaller its value, the higher the note produced.

When you use a SOUND command you have to give it a channel and a pitch parameter. You can't leave them out as in some of the other parameters we'll meet.

The duration parameter decides how long the note is going to last. It can take integer values ranging between 32767 and -32768.

The positive values decide how long the note is going to last, measured in hundredths of a second. Hence a duration parameter of 100 will last for one second while 50 will last for half a second and 1000 for 10 seconds.

Negative values of the duration parameter have a different effect. They still determine how long the note is going to last, but in a more roundabout method.

When the number is negative it tells the Amstrad that a volume envelope is to be used to alter that note and that that volume envelope is to be repeated. If the duration parameter is -3 the volume envelope is repeated three times. If it's -20 it's repeated 20 times.

When a negative duration parameter is used the length of the note produced by the SOUND command depends on the number of times the envelope is repeated. Each volume envelope lasts for a fixed time, decided in its definition.

So if a particular volume envelope lasts for two seconds and the duration parameter is -3 then the whole note will last six seconds. The repeated volume envelope determines the length of the whole note.

We'll have more to say about volume envelopes later.

The volume parameter speaks for itself. Normally it can range from 0 (silence) to 7 (loudest) but if a volume envelope is used the range becomes 0 to 15.

There's no difference in the maximum loudness. Without an envelope 7 is just as loud as 15 with an envelope. It's just that with a volume envelope you can have finer control over the loudness.

You don't have to specify either the volume or the duration parameter. If you leave them out the Amstrad assumes that you mean a note of volume 4 with a duration of 20. These are the default values of the parameters.

You should now understand why:

```
SOUND 4,200
```

is the same as

```
SOUND 4,200,20,4
```

As you can see, our basic SOUND command isn't too hard to grasp. In fact you could produce some nice noises using only the four parameters we've covered so far. However if you really want to take full advantage of the CPC464's abilities you have to know a little about the volume and tone envelopes.

Now our SOUND command takes the structure:

SOUND channel, pitch, duration, volume, volume envelope, tone envelope

As you can see, we've added two more parameters. The volume envelope parameter can take values between 1 and 15. These numbers refer to previously defined volume envelopes.

A parameter of 2 will bring volume envelope number 2 into play while a parameter of 5 invokes the volume envelope labeled number 5. These volume envelopes determine how the loudness of the note produced by SOUND command varies as the note plays.

There is a volume envelope 0. This is the default envelope and plays for two seconds at the volume specified in the SOUND command.

Having seen how volume envelopes are called by a SOUND command, let's take a brief look at how they're defined. This is done using the ENV command which, at its simplest, takes the form:

```
ENV N, P, Q, R
```

N is just the number labeling the volume envelope, and takes values from 1 to 15. the volume envelope works by changing the loudness of a note in a fixed number of steps, each step lasting for a short period of time.

The P parameter, which ranges from 0 to 127, determines the number of steps there will be in the envelope. The Q parameter, which can take values between -128 and 127, picks the volume change for each of these steps. R decides how long each step will last, measured in hundredths of a second.

So defining a volume envelope with:

```
ENV 12, 5, 2, 100
```

creates envelope 12, which consists of five steps, each lasting one second with the volume increasing by 2 for each step.

You should be able to hear its effect on:

```
SOUND 4, 200, 500, 3, 12
```

Table I shows the parameter ranges for the ENV command.

You'll probably remember that our previous envelope definition only referred to one section of a volume envelope. In fact you can define up to five sections in a volume envelope and the corresponding formula is:

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

The pitch or tone envelope is very similar to the volume envelope except that it affects the highness or lowness of a note, not it's

Parameter	Number N	Number of steps in section O	Volume change per step P	Time length of each step R
Range	1 to 15	0 to 127	-128 to 127	0 to 255

Table 1: Parameter ranges for ENV command

loudness. The basic structure of its definition is:

ENT S,T,V,W

As you might guess, S just labels the envelope. It can take values between 1, and 15. The default envelope has S as 0 and leaves the pitch unchanged.

The T parameter decides on the number of steps in the pitch envelope. Its value can range from 0 to 239.

The change in pitch that occurs at each of these steps is given by V. This ranges from -128 to 127. Notice that a negative V raises the pitch at each step while a positive value lowers it.

The W parameter gives the length of each step. Measured, as usual, in hundredth of a second, it ranges from 0 to 255.

So:

ENT 13,10,20,100

defines pitch envelope number 13. This has 10 steps, each lasting for one second. At each step 20 is added to the pitch parameter of the SOUND command. This results in a note descending in pitch.

Listen to its effect on:

SOUND 2,500,1000,7,0,13

Like the volume envelope before it, the pitch envelope can have up to five sections, each using the above parameters. This more comprehensive definition is:

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

Table II shows the parameter ranges for the ENT command.

The pitch envelope can be made to repeat simply by making the label negative. Listen to the effect that:

ENT -5,5,-20,100

has on:

SOUND 1,600,2500,7,0,5

The pitch envelope lasts for five seconds, then repeats itself as the noise keeps on going. Notice that while the label number in the definition is negative, the SOUND command still calls it with a positive number.

Also notice that, unlike the volume envelope, the repeated pitch envelope has no effect on the duration of the noise.

When the duration in the SOUND command ends the note stops, even halfway through a repetition. Listen to what happens when:

ENT -14,3,100,50

has its wicked way on:

SOUND 2,1000,400,7,8,14

Before we take our leave of the envelopes notice that by themselves ENV and ENT are mute. They don't make a noise, they just affect the noises made by SOUND commands.

As you can see from the above, we've come a long way in four articles. Our simple SOUND command has grown from four fairly obvious parameters to six, with ENT and ENV thrown in for good measure!

However I think you'll agree that if you take them step by step they're a lot simpler than they look at first glance.

And now I'll add one last parameter to the SOUND statement. It's the parameter that tags onto the end and allows us to make noise. Yes, I know that's all we've been doing for the past four months, but this is a rather different noise. Try:

SOUND 2,300,1000,7,0,0,13

and hear the effect, Not what you might have expected.

The explanation lies in the last parameter tagged on the end of the SOUND command. This is the aptly named noise parameter. It can take values from 1 to 15 (0 is the default and does nothing) adding a variety of semi-random noises to the notes you might expect.

Now our SOUND command looks like:

SOUND channel, pitch, duration, volume, volume envelope, tone envelope, noise

Try experimenting with the noise parameter to see what it can do. Use a note like:

SOUND 1,200,200,7

and then add noise to it with lines such as:

SOUND 1,200,200,7,0,0,1

and

SOUND 1,200,200,7,0,0,15

Parameter	Number S	Number of steps in section T	Pitch cnage per step V	Time length of each step W
Range	1 to 15 (-ve for repeat)	0 to 239	-128 to 127	0 to 255

Table II: Parameter ranges for ENT command

	Channel	Pitch	Duration	Volume		Volume envelope	Pitch envelope	Noise
				Without envelope	With envelope			
Range	1 = A 2 = B 4 = C	0 to 4095	32767 to -32768	0 to 7	0 to 15	1 to 15	1 to 15	1 to 15
Default	none	none	20	4	12	0	0	0

Table III: Parameter ranges for SOUND command

Use all the noise values between 1 and 15 and then try it out with pitch and volume envelopes and different pitch and duration parameters. You'll come across all sorts of marvelous sound effects (Send them in to Postbag if you've got any you want to share.)

Program I is my version of a steam engine.

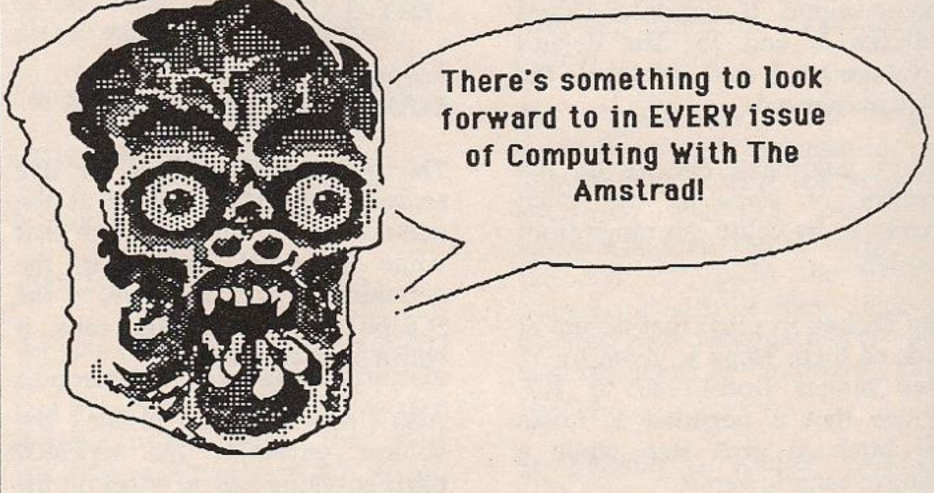
And that's it for this month. I've summed up all our SOUND parameters in Table III and leave you to experiment with noise and noises.

One last point. Remember from the first article I told you to get rid of any odd sounds by entering:

```
SOUND 129,100,0,0
```

into your Amstrad? What does a channel parameter of 129 mean? We'll be coming to that next month.

```
10   REM Program 1
20   REM Steam Engine
30   pitch=1
40   FOR duration=100 to 20
     STEP -10
50    SOUND 1, pitch, duration,
     7,0,0,1
60   SOUND 1, pitch, duration,
     7,0,0,15
70   NEXT duration
80   WHILE NOT true
90   SOUND 1, pitch, duration,
     7,0,0,1
100  SOUND 1, pitch, duration,
     7, 0,0,15
110  WEND
```


There's something to look forward to in EVERY issue of Computing With The Amstrad!

Next month in CWTA....

All the regular series on Graphics, Sound, Machine Code, CP/M, Public Domain PLUS Software Survey, Your letters, 16 page Business section and more!

There's an arcade Maths game and our arcade game of the month is Radiation Leak - it's a beauty!

Desktop publishing is all the rage and we investigate what's happening on the CPC and PCW machines.

For the utilities fans we have a complete listing of 10 machine code RSXs to help you with scrolling your screens.

There's much, much more in EVERY issue of Computing With The Amstrad - DON'T MISS IT!

LOCATE that mysterious text screen – and open lots of windows

WE HAVE already looked at all the colors available on the Amstrad and the ways we could fill our pens with them using the ink command.

So far, however, though this series is titled graphics, I haven't really explained what graphics, as opposed to text, is.

In fact most of our examples have concerned straightforward text operations rather than graphics. Or, to use more accurate terminology, they have involved the text screen rather than the graphics screen.

You see, although I haven't made it explicit, when you switch on your Amstrad, there are two different "screens" present.

To put it simply, if not entirely accurately, there is the text screen, which you use for writing words on, and the graphics screen on which you draw pictures.

The reason you haven't noticed these two screens so far is that when you switch on or reset, the text and graphics screens overlap - both filling the entire screen.

Let's prove there really are two screens. Reset your micro with Ctrl + Shift + Escape so that the start up message appears. Next enter:

```
CLS
```

It won't surprise you that the screen now clears leaving you with the Ready prompt at the top.

Reset the micro once more and this time enter:

```
CLG
```

Again the screen clears, but notice - the Ready prompt is now halfway down the screen. Never mind why for the moment - the point is that it shows there's a difference between CLS and CLG.

You see,

```
CLS
```

means "clear the text screen"

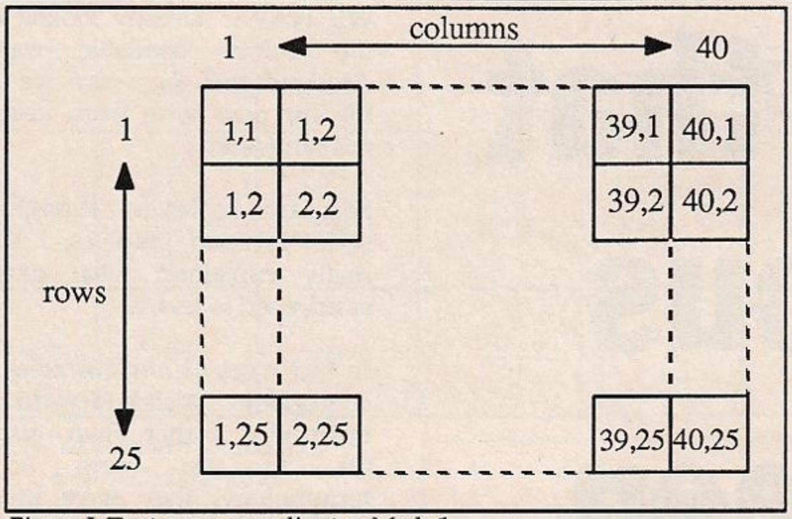


Figure I: Text screen coordinates, Mode 1

whereas

CLG

means "clear the graphics screen". See, there are two screens!

Later on we'll see how we can separate them entirely, or overlap them in various ways by limiting them to screen areas picturesquely called windows.

For the moment, however, we'll leave them both occupying the whole of the screen. To avoid the schizophrenia this may induce we'll consider each screen separately starting this month with the text screen.

The text screen is the screen that our

PRINT "Hello"

writes characters onto. As we've seen, the size of the text screen - that is, the number of characters we can fit on it - varies from mode to mode.

Each mode allows 25 rows of characters. But Mode 1 allows us 40 characters a row, Mode 0 a measly 20 and Mode 2 a generous 80 characters
per row.

We can consider each character to occupy a different "cell" of the screen, each mode giving us a different number of cells.

There's also a coordinate system for these cells, which we've been using to some extent when we've used TAB. The first cell on a row is referred to as number 1, the second is number 2 and so on. Thus:

PRINT TAB(5) "#"

would put an asterisk in the fifth column, or cell or a row, whereas

PRINT TAB(10) "#"

would place it in the tenth column.

Program I shows how we can print a simple diagonal of asterisks on the screen using TAB. It's in Mode 1. Try altering line 20 to put the program in Modes 0 and 2, and notice the effects.

The diagonal appears to go further

```
10 REM Program I
20 MODE 1
30 FOR loop% = 1 to 20
40 PRINT TAB(loop%) "#"
50 NEXT loop%
```

Program I

across the screen in Mode 0 since there are fewer cells. What cells there are, are larger, however, so the asterisks appear larger.

We say that Mode 0 has coarser resolution than Mode 1. This may seem a drawback, but the advantage is that in this mode you can have 16 colors on screen at a time.

In Mode 2, though, the diagonal fails to travel even halfway across the screen. This is because there are now 80 cells, or columns, in a row. And, of course, the cells are smaller.

**By
MICHAEL NOELS**

Mode 2 has a finer resolution than Mode 1l. The drawback is that you can only have two colors on the screen at once.

Experiment with TABing off the line - for example try:

PRINT TAB(44) "#"

in Mode 1. As you'll find, the TAB "wraps" round onto the same line to place the asterisk in column 4.

Up to now we've limited our coordinate system to determining the column in a row at which an asterisk appears. By using LOCATE, however, we can have far more control over the position of our asterisk - we can choose both the row and column at which it appears.

Take a look at Figure I, which shows how this text screen coordinate system works in Mode 1. Each cell has its own pair of coordinates, the row it is on followed by the column it is in - both separated by a comma.

So in Mode 1 the top left hand cellof the text screen would be denoted by 1,1 and the bottom right hand cell by 40,25.

```
10 REM PROGRAM II
20 MODE 1
30 FOR loop% = 1 to 20
40 LOCATE loop%, loop%
50 PRINT "#"
60 NEXT loop%
```

Program II

Let's introduce the idea of a text cursor. This is an imaginary sort of marker to show where the next character is going to be printed on the screen. If you enter:

```
PRINT "Hello": PRINT "There"
```

the words will appear on different lines. We say that, after printing Hello, the text cursor moves to the beginning of a new row.

If, however, you enter:

```
PRINT "Hello";: PRINT "There"
```

the words will appear "glued" together on the same line. We say that the semi-colon causes the text cursor to stay on the same line.

We could describe TAB(n) as moving the text cursor to column n on the current line.

LOCATE allows us to shift the text cursor and hence the position of the next character printed to wherever we want on the text screen.

We follow LOCATE with the coordinate of the cell we want to place the cursor at. So LOCATE will always be followed by two figures separated by a comma. The first specifies the column and the second the row at which we want the text cursor to appear.

Try the following in Mode 1:

```
CLS: LOCATE 20,5: PRINT "#"
```

As you'll see, the asterisk appears in column 20 of the fifth row. Follow this up with:

```
LOCATE 5,20: PRINT "#"
```

This time your asterisk appears in column 5 of the twentieth row. Make sure you get your numbers the right way round!

Actually here's an easy way to remember which comes first, the C for column comes before R for row in the alphabet - as in CR (Carriage Return).

Try Program II. This should give identical results to Program I. As loop% increases, LOCATE moves the text cursor both to the next row and the next column before printing the asterisk.

In Program I TAB(loop%) moved the cursor to the next column as loop% increases. Moving to the next row is automatically taken care of by having no semi-colon at the end of PRINT statement.

```
10 REM PROGRAM III
20 MODE 1
30 offset% = 21
40 FOR loop% = 1 to 20
50 LOCATE loop%, offset% -loop%
60 PRINT "#"
70 NEXT loop%
```

Program III

By the way, notice that while you can get away with:

```
PRINT TAB(5) "Hello"
```

you can't have:

```
PRINT LOCATE 5,5 "Hello"
```

That is, LOCATE needs to be in its own separate statement. Also, it doesn't have brackets round its numbers, as does TAB.

Since Program II gives identical results to Program I and is one line longer, you may be wondering why we need LOCATE. Well, have a look at Program III.

Being able to specify the row allows you to start at the bottom of the screen and work up, effectively reversing the direction of the diagonal.

We accomplish this in line 50 by making the value of the row selected by LOCATE, offset% - loop%.

Initially offset% has the value 21, so the first time through the loop, we locate our asterisk at 1,21-1,that is, 1,20, the final character on the twentieth row.

Next time through, loop% is 2, so we locate our asterisk at 2,21-2,=>2,19 - the second character on the 19th row. Next time through it will be at 3,18 and so on, building up our reverse diagonal.

```
10 REM PROGRAM IV
20 MODE 1
30 offset% = 21
40 FOR loop% = 1 to 20
50 LOCATE loop%,loop%
60 PRINT "#"
70 LOCATE loop%, offset% -loop%
80 PRINT "#"
90 NEXT loop%
100 WHILE INKEY$="":WEND
```

Program IV

Program IV is a compilation of Programs II and III that prints a cross of asterisks on the screen. Program V goes even further.

See if you can work out what it does before you run it. A thorough understanding of Program IV will help.

Incidentally, see what happens if the row or column numbers you give LOCATE are too large. For example, in Mode 1 you could try:

```
LOCATE 49,32: PRINT "#"
```

You'll find that specifying an over-large column number simply places the text cursor in the first column of the next line. Specifying an over-large row number in effect specifies the bottom of the screen.

```
10 REM PROGRAM V
20 MODE 1
30 offset% = 21
40 FOR multiple% = 0 to 20
50 FOR loop% = 1 to 20
60 LOCATE multiple%#20+loop%,
   loop%
70 PRINT "#"
80 LOCATE multiple%#20+loop%,
   offset% - loop%
90 PRINT "#"
100 NEXT loop%
110 NEXT multiple%
120 WHILE INKEY$="":WEND
```

Program V

The meaning of over-large depends, of course, on the mode you're in. For instance in Mode 2 specifying a column number 49 is well within scale, since we have 80 characters on a row.

Figures II and III show the sizes of the Mode 0 and Mode 2 screens respectively. Watch out for problems with LOCATE when you change mode!

To tie all this in with what we've already learned, have a look at Program VI. This prints out a series of randomly colored spaces. One point to note is that a space is a character that's completely background - so to change its color we use paper not ink.

```
10 REM PROGRAM VI
20 MODE 1
30 WHILE -1
40 color = RND(1) # 3 +1
50 PAPER color
60 PRINT" ";
70 WEND
```

Program VI

Program VII merges these ideas with that of LOCATE, printing random colors at random points on the screen. Notice that, though I'm in Mode 1, I've limited row from 1 to 24, and column from 1 to 39 - that is, onebelow their maximum value. This is to prevent the screen scrolling.

```
10 REM PROGRAM VII
20 MODE 1
30 WHILE -1
40 color = RND(1)#3+1
50 row = RND(1)#24+1
60 column = RND(1)#39+1
70 LOCATE column, row
80 PAPER color
90 PRINT" ";
100 WEND
```

Program VII

Pretty, isn't it?

And now for something completely different. Reset your micro (which ensures you're in Mode 1) and enter:

WINDOW 10,20,12,16

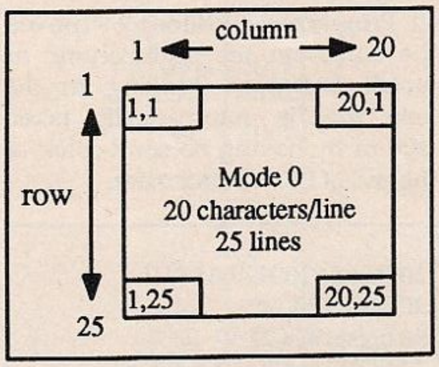


Figure II: Text screen Mode 0

The first thing you'll notice is that the Ready prompt appears in a peculiar place - towards the middle of the screen. What we've done is to restrict the text screen to a window of the screen - the rectangle shown in Figure IV.

To see this more clearly, enter:

PAPER 3

Notice that your typing doesn't appear "where it should".

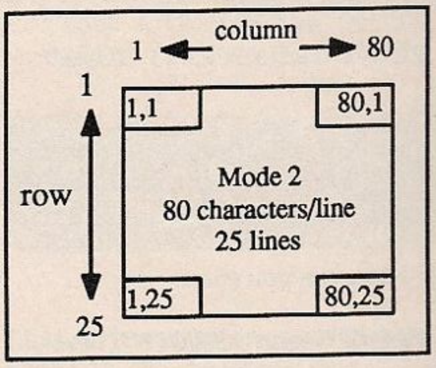
To really see the window, enter CLS and the area of the window we've defined will be cleared to red.

Notice also that the command that defined the window is still on the screen - it didn't get cleared. You see, now we've limited the text screen to our window the effect of CLS is also limited to that window.

To prove my point about the text and graphics screens being different, enter CLG and you'll see the whole screen clear this time. After all, we haven't done anything to affect the graphics screen, have we?

WINDOW works like this: The first and second numbers define the left and right hand side of the window respectively, in terms of the text screen's coordinate system. The third and fourth number define the top and bottom of the window.

So assuming you're still in Mode 1, to restore our window to the full



text screen, we would enter:

WINDOW 1,40,1,25

(The 40 would be 20 in Mode 0 and 80 in Mode 2, of course.)

Before you do that try using PRINT, TAB, LOCATE, PAPER and PEN in the window we've created. You'll find that these behave exactly as you'd expect, except for their being on a smaller screen - the text in the window even scrolls when it's full!

Note, though, that LOCATE and TAB are relative to the new window. That is, to place an asterisk in the top lefthand corner of the new window enter:

LOCATE 1,1 : PRINT " #"

You may have thought you need LOCATE 10,12 to do this - after all, that's what you needed when using the old screen.

Once you're in a new window, though, the coordinates for LOCATE "start again" with 1,1 as the top left hand corner, as Figure V shows. Notice, by the way, our window is 5 lines of 11 characters - not 4 x 10 as the numbers might lead you to expect.

Try experimenting with various windows - you'll soon get the hang of them, and you'll find that they do, indeed, behave exactly as mini text screens.

To get back to the normal Mode 1 text screen you could enter:

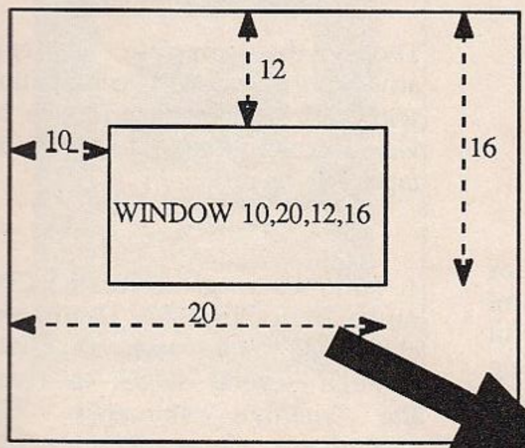


Figure IV: Defining a window

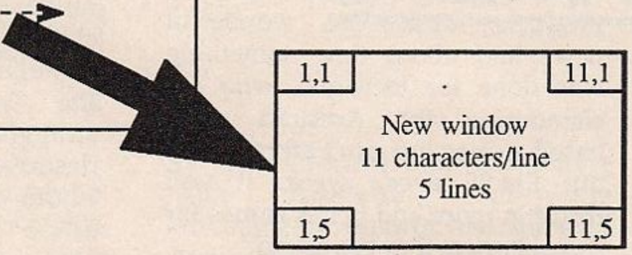


Figure V: Co-ordinates in new widow

```
10 REM PROGRAM VIII
20 KEY 139,"MODE 1: CALL
   &BC02: PEN1: PAPER 0" +
   CHR$(13)
30 MODE 0
40 WHILE -1
50 WINDOW RND(1)*19+1,
     RND(1)*19+1,
     RND (1)*24+1,
     RND(1)*24+1
60 PAPER INT(15*RND(1)+1):
   CLS
70 WHILE INKEY$="": WEND
80 WEND
```

Program VIII

WINDOW 1,40,1,25

as described above. A quicker way is to simply change mode. You may be wondering by now why we would want to use a window. After all, it just restricts the screen, limiting our possibilities. We've already learned how to put text where we want LOCATE on the original screen, so why bother?

Part of the answer is that we can have more than one text window on the screen at once, which allows us to create many special effects.

For instance, the micro might reserve one section of the screen for questions, and print our responses and the correct answer in separate windows.

Often this approach is used in adventure games. One window, occupying the top half of the screen describes the locations while another, taking up the bottom half, is reserved for your moves, inventories and so on.

We'll be looking at how to use multiple windows in more detail later in the series. This month, though, I'll leave you with an example of using a single window to effect.

One window can be far more effective than you think - in fact, the BBC Micro has only one. For instance, you can use it as a quick way of drawing rectangles. Simply define a window of the appropriate size, set PAPER to the color you want and clear the screen. Hey presto - a rectangle!

As we'll see, you could use this technique to construct rough and ready bar charts.

For the moment have a look at Program VIII. Each time you press a key you'll get a randomly defined rectangle (line 50) of a random color (line 60).

Notice how line 20 sets up the small Enter key to return things to normal once you're out of the program.

See if you can use this window technique to draw three equal squares - colors red, yellow and cyan - in Mode 1.

If you want to investigate the effect of WINDOW further you could try adding a line defining a window to the example programs we've already covered in the series - remember to put the line after the mode change.

That's all for this month. We'll soon be learning lots more about windows, but for the moment I suggest you try consolidating what we've already covered by incorporating the ideas in your own programs. Good luck!

Genesis Adventure Creator is now available!

Priced at only $39.95, Genesis offers a complete system for writing Text/Graphic adventures complete with Music or Sound Effects at any location. Features include: Text compression, synonyms for commands and objects, sentence analyzer, multiple graphic windows, variable screen modes, extensive graphic commands and storage of text and graphics on disk for large, disk-based adventures. Ideal for the novice wishing to create his/her own adventures for own use or re-sale. Genesis creates stand-alone adventures which may be sold without royalty payments. Genesis is used by commercial software houses in the U.K.

Genesis is available on disk only and is suitable for CPC 464, 664 and 6128 computers. A complete manual is supplied together with a sample adventure which can be completed by following the documentation supplied.

Sole Australian and New Zealand distributor: Strategy Software

See Page 72 for ordering details.

Software boom on the way?

STAND by for a possible games software explosion following the 40 per cent reduction in the price of 3in disks for the Amstrad CPC and PCW.

Software houses contacted by Computing with Amstrad are predicting a boom in disk-based software sales, leading in turn to significant increase in the number of entertainment programs produced for Amstrad machines.

The retail price of Amstrad's 3in CF-2 blank disks has been slashed from £5 to £2.99 , and there are hopes the price may fall even further eventually.

The high price of disks has been a major factor in limiting the output of entertainment software for Amstrad micros. But that is all set to change dramatically.

John Ainsworth, Amsoft marketing director, said: "There's no doubt more software developers will be drawn into the Amstrad market by the drop in disk prices.

"It will stimulate an enormous demand for disk-based games, and those firms who've got their wits about them and can move fast will do very well indeed."

Patricia Mitchell, product manager for Virgin Games and Leisure Genius, said: "This is very good news for us because our research shows a high proportion of Amstrad users prefer disks to tapes, so we will be able to provide them with more and cheaper games in the future."

Frank Brunger, marketing director of Ariolasoft, said: "We've always supported 3in disk software and lately have been increasing our output of programs for the CPC.

"The new price means we can look toward reducing the cost of our games and providing more titles".

Farah Jones, spokeswoman for NMC which handles Konami products, said: "It's wonderful news and about time something was done for teenagers who are clamouring for Amstrad disk-based games but can't aford to lay out £14.95 every week. It will result in more and better games for them at prices they can afford."

Ian Andrew, marketing director of Incentive, said: "Because the cost of disks was depressing the market we originally decided to bring out our two latest Amstrad releases, Apacke Gold and Winter Wonderland, on tape only.

"Now we will probably bring out disk versions as well. It's a very encouraging situation."

Jeffrey Green, general manager of Advantage, said: "This will undoubtedly boost the market for our adventure games compilations for the CPC and PCW.

"And in turn it means more authors will become interested in writing for the Amstrad machine".

At press time we were unable to contact anyone at Mitsubishi-AWA to ask for their comments. It would appear that the only person qualified to speak on this matter was on annual leave until early February.

Games round-up

LATEST version of it's bestselling arcade-style Repton games series has been launched by Superior Software for the CPC range.

Much larger than its predecessors, Repton 3 offers 24 screens and new features allowing the user to design his own screens and monsters, and even re-define the characters.

Those who complete the epic adventure can enter a competition with cash and other prizes to be won. Prices will be about $29.95 on tape, $45 on disk.

*****

A NEW battle game for the CPC range from Bug-Byte, Deathwake challenges the user to fight through several stages to reach the enemy's mountain top stronghold. Bombers, U-boats, destroyers and minefields contribute to the action. Price: app. $14.95

*****

A FOLLOW-UP to the cartoon game Redhawk has been released for the CPC series by Melbourne House.

Kwah incorporates enhanced graphics, improved sound effects, a score-meter and direction indicator as well as a devious and intricate plot involving a greater degree of character interaction.

The program follows Kevin Oliver in his confrontations with the sinister Dr. Lee as he tries to discover the secret of the past he no longer remembers.

*****

A NEW racing game for the CPC range has been released by English Software.

Described by the copany as "a major step forward in racing/driving games for Amstrad computers", Elektraglide features perspective scrolling landscapes and 3D hazards appearing from the skies. Price on tape $27.95, disk $45

*****

A VERSION of the bestselling arcade game Z has been released for the CPC range by Rino Marketing. Converted by top programmer Steve Evans, the game features graphics of near arcade standard together with impressive backdrops. Price $24.95 on tape, $45 on disk.

PC1512...fan being fitted to keep everyone happy

Amstrad nails PC rumours

AMSTRAD has moved quickly to stamp on damaging rumours about the PC1512 that have been circulating in the industry and reported in some sections of the computer press - particularly the Sunday Times.

Described as "assuming the proportions of a dirty tricks campaign", the smears claimed the new machine overheats easily, won't run Token Ring and the IBM PC network, and has been rejected by ICI evaluators "because of its problems".

The slurs have been totally rebutted by Amstrad in a detailed statement supported by ICI's director of information technology Derek Seddon and leading figures in the retail and distribution sectors of the micro market.

IBM has entered the controversy to protect itself against claims that its employees are involved in the anti-Amstrad campaign.

A spokeswoman told Computing with Amstrad: "We are investigating reports that some of our sales staff have been telling dealers the Amstrad PC1512 melts when Token Ring network cards are inserted in it.

"I can state categorically that disparaging the competition is a strict contravention of IBM business conduct guidelines."

Meanwhile, to further stifle the snipers, Amstrad has announced it is now fitting fans to all top-of-the-range PC1512 hard disc models.

Floppy disk models can be supplied with a fan fitted as a optional extra for $60.

Sales and marketing director Malcolm Miller described the move as "a marketing tactic - Amstrad has simply decided to satisfy those potential customers who feel more secure with an integral cooling system.

"The PC1512 range has been greeted with universal praise, both in terms of performance and value for money.

"While deep and incisive trials have given the machine a clean bill of health, there are those who have suggested that it runs hot and they would like to see a fan fitted.

"Frankly, we are getting very irritated with this rumor. We could tell every potential corporate customer that it is a lot of nonsense, but that would take a lot of time and effort.

"It is easier to fit the fan and be done with it".

Alan Sugar commented: "This fan is a waste of money, but it will keep some people happy.

"I recommended that operators switch the fan off. It'll save electricity and won't make any difference to the machine's operation".

University picks PC

AMSTRAD has made a breakthrough into the lucrative higher education market with the PC1512.

Aston University has placed a large order for the machines to run alongside its existing Macintosh and IBM PCs.

The micros will be installed over the next 12 months, and the first batch has already been delivered to lecturers.

The university says the PC1512 will eventually be widely used in student teaching, research and administration.

It will also play a major part in the electronic mail strategy being devised by the university.

The objective is to have a terminal on the desk of every lecturer and administrator linked into the central computer systems, which include two DEC VAX8650s. The PC1512 links to the DEC equipment via a VT 100 emulator.

SOFTWARE AIDS CHARITY

A NEW public domain screen editor has been launched for the PCW to boost the funds of a leading charity.

The author, Dr. Ronald Yorston, has asked everyone who makes a copy to donate at least $25 to War on Want.

At the same time he has banned the use of software by the military.

'Boost software standards' call

A CALL has gone out to entertainment software houses to improve their standards to cater for the increasingly sophisticated Amstrad computer user.

It comes from Stewart Bell, the managing director of the recently formed UK subsidiary of MicroProse, the number one developer of simulation software for the home computer market in the United States.

"The problem is that there are too many software houses around who are apparently working on the assumption that end users have IQs less than orang-outangs", says Stewart Bell.

"However they couldn't be more wrong. For just as machines are becoming more sophisticated, so too are the users. And no more so than in the case of Amstrad people".

The end result of all this, according to Stewart Bell, is that the entertainment side of the Amstrad computer range is not realizing its full potential.

Naturally as the UK boss of MicroProse - the company known for its award winning entertainment simulation software - Stewart Bell is not at all pleased with this situation.

For his company claims its product is in the Rolls Royce category.

Producing only a handful of titles each year, MicroProse spends up to $ 1 million on the development of each, so guaranteeing each title will be a blockbuster.

Now Stewart Bell's job is to hammer home the message to micro owners not to be put off purchasing entertainment software just because they may unwittingly have bought shoddy products in the past.

"Thanks to the authenticity and accuracy of our simulations, users get literally hundreds of hours of pleasure out of our product", he says.

"So you can understand why we become a little irritated when our software occasionally turns up on the shelves next to the cheap and nasties".

MicroProse has announced its first titles for the Amstrad range.

It has launched two of its most famous packages - Silent Service and Acro Jet - for the CPC machines. Both are priced at $29.95 on cassette and $45 on disk.

Silent Service is a World War II submarine similation while Acro Jet recreates in exact detail intricate high speed aerial manoeuvres.

Also in the pipeline is F-15 Strike Eagle.

In all, six major titles from MicroProse are available for the new PC1512, ranging from Spitfire Ace to Crusade in Europe. Price range is $55 to $75.

APPROVED COPIER GOES ON SHOW

THE sixth Amstrad Computer Show at the Novotel, London, last month was the scene of a highly important computer industry milestone.

For the first time an officially approved software copying device was on sale.

It is seen as a landmark in responsible relations between micro manufacturers and software producers on one hand, and peripherals suppliers on the other, in an area often plagued with discord and mistrust.

And it is a significant victory for the joint anti-piracy policies of Amstrad and show organiser Database Exhibitions.

The Mirage Imager not only has the blessing of Amstrad and Database Exhibitions, it will be used as a benchmark by which other copying devices will be evaluated.

The policy of both Amstrad and Database is not to allow aids to software piracy anywhere near the official Amstrad Computer Shows.

Products which fall into this category are hardware imagers which allow the computer to be frozen at a given point and a snapshot copy taken, and disk-to disk and tape-to-disk copiers. Mirage Imager merits Amstrad and Database approval because it incorporates a serial number process and programs saved are unreadable unless the device is present.

The Amstrad Computer Show, which ran from January 9 to 11, was the biggest yet, with a record number of new products and post-Christmas bargains galore.

Advanced Memory Systems unveiled at least four new products. With large quantities of the PC1512 beginning to arrive in the UK, there will be no shortage of software or peripherals support for the new machine.

And the drop in 3in disk prices is expected to result in a flood of new entertainment software.

Comms package

TWO leading suppliers are combining to produce a high specification comms pack for the PC1512.

Pace Micro Technology, famous for its Nightingale and Series Four modems, has designed a new modem for use with the new machine. The package will be capable of supporting V21 and V23 speeds, as well as auto/answer dial, and offer full Hayes compatability.

BUSINESS COMPUTING WITH THE AMSTRAD

MARCH 1987

FULL AMSTRAD RANGE OVER 220 SOFTWARE TITLES IN STOCK AT ALL TIMES!

Mail order and Telephone specialists

Computer Oasis

(09) 385 1885

- AMSTRAD DUST COVERS
- DISK WIZARD
- GOLDMARK T.I.E.S.
- MONITOR EXTENSION LEADS
- DISCMASTER
- SPEEDTRANS

tech-soft

Computer Wholesale

(09) 385 1765

324 STIRLING HIGHWAY, CLAREMONT, WA 6010

Desk top publishing boom forecast

AMSTRAD is set to double sales of its computers come the 1990s by riding the next boom predicted for the industry - desk top publishing. The forecast comes from Nick Pearson, managing director of Advanced Memory Systems, the company that has just released it best selling Pagemaker package for the CPC range.

"Desk top publishing is the next stage in the computer revolution after word processing", he told Computing with the Amstrad. "A major report emanating from the United States has stated that it will double the sales of personal computers in the next decade. And with Amstrad being a market leader already, there is little doubt that this is the least the company will achieve".

The AMX Pagemaker for the Amstrad combines a typesetter, graphics and word processor on two disks for about $350. It lends itself to letters, posters, price lists, news sheets, in fact anything with graphics and text that needs to be created on A4 size paper.

"We have already seen enormous interest in this product", said Pearson, "but fascinating and useful as it is this is only the start. Desk top publishing is about to transform the computer business itself as more and more advances take place".

Pearson predicts that DIY publishing will have a dramatic impact because it will make a significant contribution to cost cutting for businesses around the world. "If you can imagine not having to pay huge printing bills, then you will appreciate what I mean", he said.

"Just think of the money involved in all those glossy annual reports that are produced for companies. Well within a relatively short time, desk top publishing will have become so advanced that the creative office manager or secretary will be able to produce them - on their own".

Nor will the impact stop there. "It would be possible for every street to have its own newspaper because of desk top publishing packages. Even Eddie Shah with his love of new technology couldn't possibly conceive all the exciting breakthroughs that are just around the corner".

The PCW8256 version of AMX Pagemaker will be out in the autumn.

Gestation . .

THE long awaited firmware catalog for the CPC6128 will now be available in a matter of weeks, according to an Amstrad official. Cliff Lawson revealed this during a question and answer session at the Amstrad Computer Show.

But what was the reason for the delay?

"Someone had a baby", he disclosed.

Silicon disks get new OS

A NEW operating system from DK'Tronics enables the firm's silicon disk to work as Drive C with CP/M+ on the CPC464 and CPC6128. The new software also allows users with both the 256k memory and the silicon disk to configure the system as 442k silicon disk.

It automatically probes the machine to find out how much memory is fitted and sets the system accordingly.

The enhanced version has 128 directory entries with 2k block size as in the PCW8256 RAM disk, and automatic initialization and configuration to use the maximum amount of expansion RAM - including the additional RAM provided by the 256k RAM expansion.

This allows three possible configurations: silicon disk, expansion RAM or both expansion RAM and silicon disk RAM.

Customers who want to update their 1.0 version of the silicon disk operating system should return it to DK'Tronics. The silicon disk operating ROM has also been released as a product on its own, enabling users of the 256k bank switched memory to use it as a 254k disk from Basic CP/M 2.2 or 190k with CP/M+. Price - about $75.

In addition DK'Tronics has updated the software supplied with its memory expansions to allow a 61k TPA area with CP/M 2.2. A free upgrade service is being offered for this facility to customers returning their bank switching software to the company.

Training

TWO firms have joined forces to open 75 "PC Learning Centres" to train business users on the PCW8256 and PCW8512.

Headline Communication and RT Computer Training will base their centres on an existing network of secretarial training venues. New courses have been designed which embody the techniques of Headline's Reel-Time audio tutors.

Headline has recently published a range of Reel-Time audio tutorials for the PCW8256 and PCW8512 covering Locoscript, CP/M and Basic, and Supercalc 2.

☞

A division of Reach-a-Teacha, which has been offering audio-visual training packages for more than 12 years, RT Computer Training has a chain of 90 franchised secretarial training centres.

Database for tyros

Using a simple menu system, File Manager from Sandpiper Software is designed to enable first time users to create a database system within hours rather than days.

The program allows the user to begin with simple systems and then expand into fully interactive databases with automatic transaction processing and report generation.

The database is fully relational and interactive, permitting function processing, rapid screen creation and screen formatting.

An interface is built into the system for the creation of external list files for processing by other software systems, and an eight character encryption code is available which allows full data security.

Data usage is minimized typically by two-to-one compression of all numeric data into the record held on disk.

A launch date is imminent and File Manager, claimed by Sandpiper to be as powerful as systems costing more than $1250, is expected to cost around $375.

IBM loses out

A LEADING financial software house that made its name in the IBM PC market now finds that this section of its business is overshadowed by demand from Amstrad users.

Compact Software, the specialist company set up by accountants, is currently achieving record sales thanks to the PCW8256.

It recently announced it had notched up the 2,000th sale of its accounts package for the machine.

"We now find that the Amstrad side of our business is outselling IBM," says Danny Lindsey, a Compact director, "so we are obviously very well pleased with the market".

The company has recently reached agreement with three major distributors - GEM, Proteus and New Star - to handle its range for the Amstrad market.

Nor has its success linked to Amstrad been confined to the UK. Compact has just revealed that it has signed distribution agreements in Australia and Spain. These involve Proteus, Compact's Rita Award winning package, which is a report and program generator.

"Our newly appointed distributors out there are very enthusiastic about Nucleus", said Danny Lindsey, "but even more so about the future of the PCW in their respective markets".

Nor does Compact share the concern of some companies that the impending Amstrad IBM clone may damage the PCW market. "If this does happen, it will not effect us at all", insists Danny Lindsey.

"All our product can be upgraded to IBM compatibles. After all that's the very market in which we started out".

Latest Pascal

OXFORD Pascal - the fully extended implementation of standard Pascal - is now available for the CPC 6128 and the PCW 8256.

Systems Software (Oxford) offer the compiler, which supports every feature of Pascal, for app. $65, including a comprehensive tutorial and reference manual, making it suitable for those new to the language as well as the experienced programmer.

Oxford Pascal, running under CP/M+, features a range of extensions designed to get the most out of the Amstrad models including hex constants and I/O, bit manipulation, random access files and program chaining.

Delta 1 gets a boost

THE outstanding success of the PCW8256 has given a new lease of life to Delta 1, the earliest version of Compsoft's best selling Delta database management system.

High Street demand for the Amstrad machines, coupled with a major Compsoft advertising campaign, has generated massive sales for the package.

Delighted Compsoft marketing manager Wendy Berry told Business Computing with the Amstrad: "There's a new attitude among micro buyers. Database software is now the in thing so we've increased our output of Delta and appointed additional distributors, particularly those experienced in High Street retailing".

Launched in 1983, Delta 1 was developed for the earlier generation of 8 bit micros and had been superseded by versions for 16 bit computers, the latest being Delta 4. But the arrival on the scene of the 8 bit PCW8256 has resurrected Delta 1 and, says Berry, "demand has been mushrooming since 1st November".

Please note that all prices shown are approximate and are calculated at 2.5 times the U.K. retail - this may vary considerably from what you pay in Australia or New Zealandas no allowance is made for duty..

Simple method of replacing manual files

GABRIEL JACOBS reviews Cardbox, the flexible database which is easy to use but suffers some limitations for business use

THERE are database systems available for the PCW8256 and the CPC6128 which stretch the machines to their limits. Cardbox is not one of them, but if all you're after is something to replace an old manual filing system, and if you prize simple virtues, it could be a suitable choice.

Cardbox simulates a physical collection of filing cards. The format of the cards can be designed to your own specifications, and they are displayed on the screen one at a time. You can browse through a file, type information on to a new blank card and edit existing ones, just as you would with a conventional card-index system.

The benefit in doing this electronically lies in the ability to manipulate data in ways which pose problems with many physical record system, whether or not the information is held on cards.

Take the published telephone directory, from which it is virtually impossible to find people's names or addresses if all you know is their telephone number.

Nearly all electronic databases allow a degree of flexibility in data retrieval. Some will perform amazing feats, though often at the cost of complex procedures which have to be written by the user before the operation can be carried out.

Cardbox, on the other hand, needs no programming since it is entirely interactive, and has been designed above all for ease of use while still offering search procedures adequate for many applications.

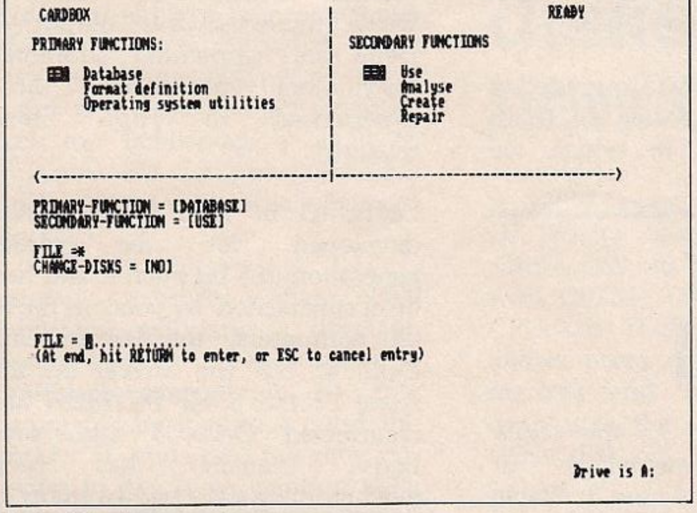
Its most impressive search feature is the way in which it allows selection to be taken many levels deep, in the form of a hierarchy, without the process becoming unmanageable.

For example, you might request an initial search for cards containing a particular word. From this, a temporary file, and those which match the new search criteria will then make up a second temporary file, or Level 2.

You may then decide to request a new search on the entire original file, and the cards found could be added, say, to those at Level 2. You can go on like this with as many permutations as you like, down to a theoretical limit of Level 99.

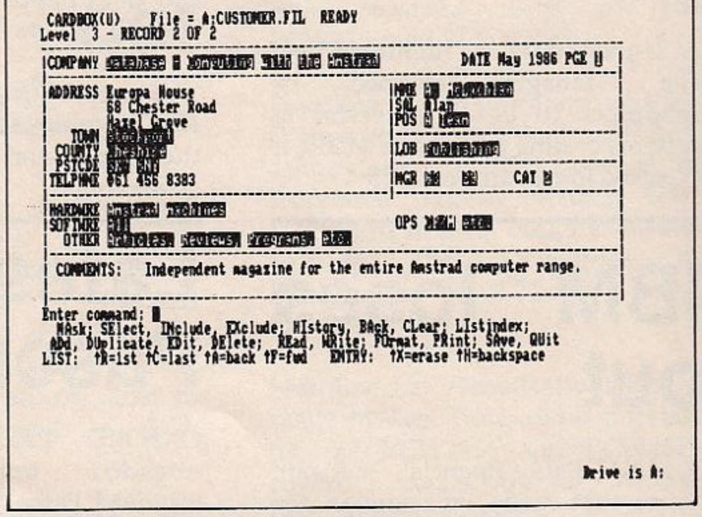
Cardbox, in fact, works on roughly the same principles as the old fashioned system using numbered holes punched round the edges of record cards. After clipping selected holes to leave only a half-moon shape you inserted a knitting needle through the cards at one of the numbered points, then lifted it up.

In this way you left in the box all the cards which had that number clipped, while the rest dangled from the knitting needle. ☞



```
CARDBOX                              |                                READY
                                     |
PRIMARY FUNCTIONS:                   | SECONDARY FUNCTIONS
                                     |
   ▮▮▮ Database                      |  ▮▮▮ Use
       Format definition             |      Analyse
       Operating system utilities    |      Create
                                     |      Repair
                                     |
(------------------------------------|------------------------------------)

PRIMARY-FUNCTION = [DATABASE]
SECONDARY-FUNCTION = [USE]

FILE =*
CHANGE-DISKS = [NO]


FILE = ▮...........
(At end, hit RETURN to enter, or ESC to cancel entry)


                                                              Drive is A:
```

Typical Cardbox record card



```
CARDBOX(U)    File = A:CUSTOMER.FIL  READY
Level  3 - RECORD 2 OF 2
-------------------------------------------------------------------------
|COMPANY Database & Computing With The Amstrad        DATE May 1986 PGE U |
|-------------------------------------------------------------------------|
|ADDRESS Europa House                          |NME Mr [illegible]        |
|        68 Chester Road                       |SAL Alan                  |
|        Hazel Grove                           |POS E Ed                  |
|   TOWN Stockport                             |--------------------------|
| COUNTY Cheshire                              |LOB Publishing            |
| PSTCDE SK7 5NY                               |--------------------------|
|TELPHNE 061 456 8383                          |MGR [illegible]   CAT [illegible] |
|----------------------------------------------|--------------------------|
|HARDWRE Amstrad Machines                                                 |
|SOFTWRE All                                    OPS [illegible]           |
|  OTHER Articles, Reviews, Programs etc                                  |
|-------------------------------------------------------------------------|
| COMMENTS:   Independent magazine for the entire Amstrad computer range. |
|                                                                         |
|-------------------------------------------------------------------------|
Enter command: ▮
  MAsk; SElect, INclude, EXclude; HIstory, BAck, CLear; LIstindex;
  ADd, DUplicate, EDit, DElete;  REad, WRite; FOrmat, PRint; SAve, QUit
LIST:  ↑R=1st ↑C=last ↑A=back ↑F=fwd   ENTRY:  ↑X=erase ↑H=backspace


                                                              Drive is A:
```

Typical record card in brief format for a mail list

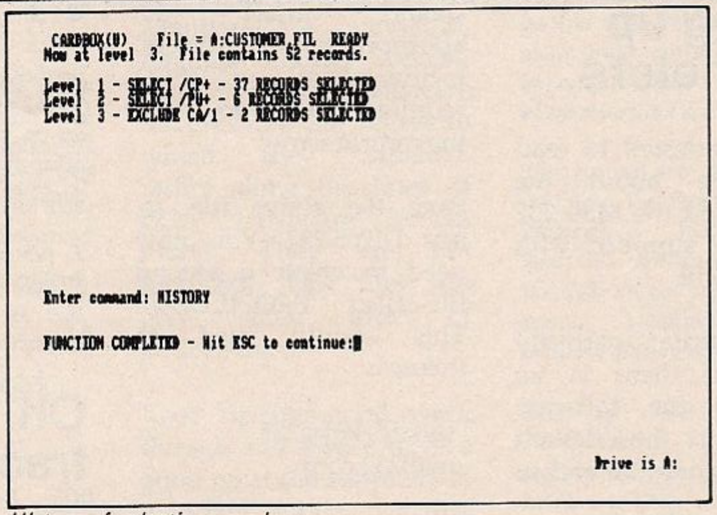
The ones left in the box were the equivalent of Cardbox's Level 1, and to get to the next level you simply inserted another knitting needle at an appropriate number. There was no need to keep the cards in any particular order, only the right way up.

It need hardly be said that Cardbox offers greater sophistication than the rather primitive punched record card. True, the program can't compete in search speed with the almost instantaneous knitting needle, but words can be indexed, making for rapid selection.

And it offers facilities such as searching for a particular string of characters within a word, taking care of spelling uncertainties with the use of wildcard characters, and displaying the selection criteria and the number of hits made at each level.

It also lets you change your mind without too much bother, particularly when it comes to entering data. For instance, if you find that an entry is too long to fit into a previously defined field ·an area of the card set aside for it) it is relatively easy to make an extension by redesigning a new card format.

In fact, any number of different formats can be associated with a file, and they can include windows for viewing or printing out selected fields.



```
 CARDBOX(U)    File = A:CUSTOMER.FIL  READY
Now at level  3.  File contains 52 records.

Level  1 - SELECT /CP+ - 37 RECORDS SELECTED
Level  2 - SELECT /PU+ - 6 RECORDS SELECTED
Level  3 - EXCLUDE CA/1 - 2 RECORDS SELECTED




Enter command: HISTORY

FUNCTION COMPLETED - Hit ESC to continue:█



                                                    Drive is A:
```

History of selections made

This kind of flexibility is very much in Cardbox's favor - some packages expect you to define once and for all the length of all fields before entering any data, or face a cumbersome linking and copying process.

Some even allocate disk space for each potential record of a newly created file, so a record containing only a few bytes of data takes up the same amount of room on disk as one filled to capacity. Cardbox does not pad out its records to keep them all the same length, so each occupies the minimum possible disk space.

Other Cardbox plus points include the ability to create disk files in a variety of formats which can be read by other application programs such as word processors, an uncomplicated printout facility, and a repair or rebuilt feature which will do its best to recover data from a corrupted file.

So far, so good. Unfortunately that's not the end of the story.

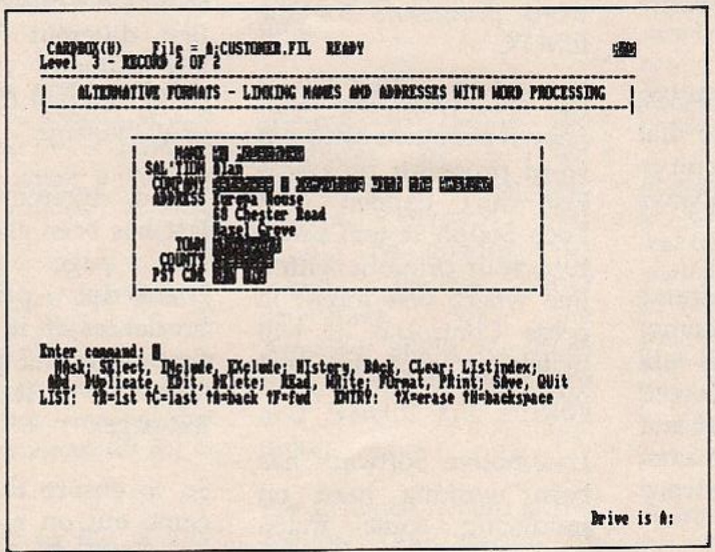
The limits of database packages are often quantified in terms of maximum data space. Cardbox is limited to 65,000 records and a maximum file size of 16 megabytes, but with a 3 in disk such values are irrelevant.

Its upper limits of 26 fields and 1,404 characters per record might be an important consideration in some circumstances, but on the whole Cardbox's real limitations lie in what it will not do, rather than in the amount of data it will hold.

It will not sort cards into alphabetical order on a key field, as many database systems do. Of course, as with punched cards, the order of the records is immaterial for the purposes of a search, but there are instances when an alphabetical list is useful, even essential.

It will not perform any numerical operations, and so cannot be used for accounts. You can't even search for numerical values corresponding with criteria such as greater than or less than, or select cards with dates falling within a given range.

So although Cardbox is easy to use, although it comes fully configured for both the 8256 and the 6128, and although the documentation and demonstration files are beyond criticism, $295 for such a limited program does not seem good value for money.



```
 CARDBOX(U)    File = A:CUSTOMER.FIL  READY                    [redacted]
Level  3 - RECORD 2 OF 2
--------------------------------------------------------------------
|   ALTERNATIVE FORMATS - LINKING NAMES AND ADDRESSES WITH WORD PROCESSING   |
|-------------------------------------------------------------------|

          ---------------------------------------------
          |     NAME [redacted]                        |
          | SAL'TION Alan                              |
          |  COMPANY [redacted]                        |
          |  ADDRESS Eureka House                      |
          |          68 Chester Road                   |
          |          Hazel Grove                       |
          |     TOWN [redacted]                        |
          |   COUNTY [redacted]                        |
          |  PST CDE [redacted]                        |
          |--------------------------------------------|

Enter command: █
  MAsk; SElect, INclude, EXclude; HIstory, BAck, CLear; LIstindex;
 ADd, DUplicate, EDit, DElete;  REad, WRite; FOrmat, PRint; SAve, QUit
LIST:  ↑R=1st ↑C=last ↑A=back ↑F=fwd   ENTRY:  ↑X=erase ↑H=backspace



                                                     Drive is A:
```

Cardbox opening menu

Setting up parameters

I WAS interested to read the article about the CPS8256 and the Mail 232 software supplied with the PCW 8256.

As the writer correctly points out there is no facility in the software for setting the default values of baud rate and so on.

I calculate that setting the communication parameters for accessing Telecom Gold and identifying yourself to the computer requires a minimum of 46 keystrokes - more if you have a long password.

However judicious use of the SETKEYS.COM command file enables this to be reduced to just four keystrokes.

The first step is to set up a key definition file, as described on page 108 of the CP/M manual. To do this you can use the file editor RPED.BAS supplied on the system disk. A file suitable for Telecom Gold should read as follows:

```
02 S "^'#82'"
E #82 "call AA^M"

00 S "^'#84'"
E #84 "id BBB123^M"
73 S "^'#86'"
E #86 "XXXXXXX^M"
03 N "^'#8D7"
E                    #8D
"^Z^V^V^V^V^V^V^V^'
30
'^Z^V^V^V^V^V^V^V^'3
0^V^V^
V^'30'^V^V^["
```

In the above AA stands for your system number, BBB123 for your customer identity and XXXXXXX for your password. You should insert the appropriate characters. Incidentally RPED requires Extra+U to create the control arrow.

Save the above file as, say DEYS.232. You now need to create a second file called PROFILE.SUB This should read as follows:

```
setkeys keys.232
mail232.com
```

and your comms disk should have on it:

CP/M+,
MAIL 232.COM,
SUBMIT.COM,
SETKEYS.COM,
KEYS.232 and
PROFILE.SUB.

When you boot the disk Mail232 will load and run itself automatically, but here is where the clever bit begins.

You have re-defined the Paste key to deliver all the keystrokes needed to set the transmission parameters, so press Paste.

The framing menu appears, and like lightning the Tx and Rx baud rate will set themselves up to 300, the word length will become seven bits and the parity will change to even.

The menu vanishes, leaving you ready to dial up BT Gold if you want to check you can always press f1.

Once the mainframe responds with the prompt PAD> press f2 - this will automatically send call AA. When required f4 and f6 will send your customer identity and password respectively.

David Starer.

● *An excellent idea, and one which can of course be taken further. For example, codes for VT 52 emulation and appropriate log-on strings can be put into other keys which are unused in Mail232.*

I personally always name as SENDASC all Ascii files to be sent by e-mail, since they will subsequently be deleted.

On the right track.

I FOUND the comparison of Locomotive Software's Loco-Script with Micropro's Wordstar fascinating.

The chief test is the effectiveness of a word processor. LocoScript has been designed from the outset to be natural to use. You tell it what you want to do and it both does it and remembers what you wanted to do.

Then when you change your text later Locoscript already knows how you want it laid out. Also at no time does LocoScript leave a document incorrectly laid out.

We at Locomotive Software strongly believe this to be the best approach - one that is used by the best of today's word processors for the IBM PC.

In this way the well known problems with old word processor programs just can't happen with Loco Script. It will never ruin your printout with a line where you forgot to press Control+B, it will match or exchange text over a next line.

Locomotive Software has been working hard on producing some much requested add-ons to LocoScript. We would expect the first of these, LocoMail and LocoSpell, to be available by the time this letter is published.

LocoMail will provide sophisticated mailshot and form filling features. LocoSpell, which we have developed in conjunction with Longman's dictionaries division, will provide an on-line 33,000 word dictionary, and a much larger dictionary for more serious use.

Both will of course run alongside LocoScript and operate directly from LocoScript's menus.

Finally I should like to correct both Mr. Villiers' letter and your reply in the July issue concerning pagination of single page document.

LocoScript provides very flexible features for headers and footers, and we have tried to make these work as naturally as possible.

Mr. Villiers is completely wrong when he suggests that LocoScript reads a document backwards. What it does do is to allow you to have different headers and footers on different pages first different from rest, last different from rest, odds different from evens or all the same.

When different first or last has been chosen on a one page document LocoScript prints the header as if it were the first page and prints the footer as if it were the last page.

So to ensure that footers come out on a one page document do not disable the footers on the last

page. This is covered on page 107 of the user guide - in the supplement for the first edition.
Howard Fisher,
Locomotive Software Ltd.

Accounting program

We are a small business, running an Amstrad 8512, and are having difficulty finding a suitable database.

We have tried Cardbox, which would be perfect, but it does not have an arithmetic function.

Can you recommend a database program and in particular have you any experience of a program called Delta 1.25? Also, have you any recommendations for accounting programs?

J. M. Donlan.

● *Sage produces a data management system called simply Database, which has some calculation facilities and which may well suit you.*

Delta 1.25 is produced by Compsoft, costs about $250 and is fairly powerful transactional database which will handle all kinds of data management applications, including integrated accounts, stock records, and so on.

Transactional means that sub-records can be attached to each record - in the case of Delta, up to eight - giving a kind of simulated multi-file handling capability.

Delta has a good label printing module (up to five labels across a page, with three fields per label line) and will link quite happily to mail merge programs and word processors, but not to LocoScript.

It also saves its files in DIF format, which means that it can read in data from most up-market CP/M spreadsheets.

However be warned that Delta is not for beginners who may expect instantaneous results. It takes some getting used to, and in fact is a set database on some university courses on information systems.

Database v spreadsheet

I READ with interest Jo Stork's article in the September issue of Business Computing with the Amstrad about spreadsheets and databases.

The impression that I was left with after reading the article was that the Cambase database and the Supercalc II spreadsheet were, if not of comparable quality, at least in the same league.

I find this a long way from my experience - the Supercalc spreadsheet is miles ahead of anything the database system can even begin to think about, let alone accomplish.

For business users or serious home users a more complete survey of packages is required, not just a review of the ones that happen to land on your desk at a point in time.

You have made an attempt to define the attributes of a self respecting database system, perhaps you should be goading the brand leaders into providing for the Amstrad along the lines that Digital Research and Sorcim are already doing.

For instance, where is the Ashton Tate version of Dbase ll or III for the Amstrad?

Another thought, establish some benchmarks for software features and the degree to which they satisfied, rather along the lines of PCW hardware benchmarks. They are not foolproof, but they do give some idea of performance.

Keep up the good work though and please have a good go at the suppliers of manuals. Again referring to the comparison of Supercalc with Cambase, why Cambase should be allowed to get away with such a user-hostile set of duplicated sheets structured in such a peculiar fashion and not get hauled over the coals for it I cannot imagine.

J. A. Cornish.

● *I apologize for implying that Supercalc II is on a par with Cambase. I tried to say that in my opinion Supercalc II is the finest spreadsheet I know of for the Amstrad and that I find databases are potentially far more useful than spreadsheets.*

Supercalc II is without equal in any commercial environment, whereas Cambase, which is very good indeed, does face some stiff opposition.

This difference is that it is more application-specific, since its very sophistication limits it to more complex tasks. I fear I did not differentiate clearly between my swings and roundabout.

Secondly, you suggest a survey of application packages. This sounds fine in theory but worthwhile road tests require 4 to 12 hours depending on the features offered. Were I doing this work full-time and also prepared to forego all of life's pleasures, I might make a decent job of such a survey.

A dry table of features would not only be worthless, but could be misleading. Handbooks, user friendliness plus a whole raft of systems design factors such as audit trails are ultimately more important than whether the maximum record size is 512 or 1 k.

Turning to the brand leading packages, the suppliers face two problems. Just because Amstrads are so successful in the UK do not forget their real impact worldwide is still minimal, albeit growing fast.

Most of these systems come from America, where until the start of this year Amstrads were unknown. Only when Amstrads become a major worldwide micro producer will the suppliers consider conversions other than CP/M copies on 3 inch disks.

The second problem is that many of these packages do not make practical sense when only 178k of file space is available. Many finance systems need a couple of 360k disks just to hold the programs.

The arrival of the PCW512 and its adequate storage, plus the rapidly rising sales curves, may still not be enough to make the Americans show an interest because if the IBM PC clone is genuinely compatible, they will be in the Amstrad market anyway.

Like the survey, benchmarks also sound an attractive proposition, and I will try to come up with some. However this is not as easy as it may appear.

The problem is to produce a test which not only stretches the software but can also be performed by all packages of that type.

In the case of a ledger posting system standard data would be required, and masses of it.

I have still not solved how to produce such data when product A needs six input fields in sequence X and product B requires eight in sequence Y.

Even assuming this were possible, would it be meaningful if during a batch run system A updates only five totals while system B updates eight? I am not rubbishing the idea, far from it, but the more I feel the *practical difficulties will prove insurmountable.*

As far as handbooks are concerned, I honestly believed I was not only giving their creators a hard time but also becoming a bore on the subject. Nevertheless I shall be devoting part of a future column to this matter.

I trust that this reply clarifies any misconceptions I may have caused, as well as outlining my views on the topics you raised.
Jo Stork.

Unfair to LocoScript?

HAVING read your September 1986 issue I feel a need to make a comment or two about the article by Gabriel Jacobs entitled LocoSCript v Wordstar.

Firstly, in Table I there appeared to be irregularities in the scoring of points between LocoScript and Wordstar.

In my opinion it was inconsistent and unfair, especially when Loco-Script received as its' highest score 15 and Wordstar received 25 for an average.

This may be due to the fact that there was no criteria laid down for the scoring. If there had been I think that PCW owners would have accepted a defeat by Wordstar because we know what we have in LocoScript.

The second point I would like to make concerns the Cut key which Gabriel Jacobs forgot to mention. I realize that the high-flying want a word processor that will do anything at a single command, but as someone once said "you pay your money..."

I would like to thank Gabriel Jacobs for pointing out the bugs in early versions of LocoScript, something of which I knew nothing. I checked my version and sure enough it was in need of refreshment.

After making a phone call to Amstrad I was told that if I returned my systems disk - side 1 and 2 only - they would indeed replace it free of charge and I have done this at a great rate of knots.

R. Griffiths.

● *Firstly, my scoring system was based on a personal judgement taking into account my own needs. I said so in the article and pointed out that readers' scores might well be different from mine.*

I don't see how such personal judgements can be subject to a charge of inconsistency.

What criteria can there be for judging two high-quality packages, other than what you yourself feel about them? My criteria were bound to be impressionistic.

Secondly, on the delete-in-one-go question see the reply to M. Searle.

Thirdly, Amstrad told me that all they required was a letter marked LocoScript. I've checked with them again and they confirm that directive.
Gabriel Jacobs

Deleting text

I BOUGHT your magazine for the first time because of Gabriel Jacobs' excellent comparison of the Amstrad WP with Wordstar - and because I disagreed with two of his criticisms.

I find the Cut key can delete text very flexibly if used with the Word, Line, Doc/Page and Para keys and also with the Find key to define other portions of the text - with the facility of viewing and amending if necessary the extent of the deletion before confirming or canceling the action.

I seldom use the Del keys except to correct spelling mistakes as I'm typing in text.

Also, although I agree it would be nice to have a dedicated Word key on the keyboard, with experience I find there is sufficient flexibility in moving the cursor by using the Line/End of line key along with Shift and Alt, or the cursor direction keys along with the Shift key, to make good use of the Word/Char key.

I'd certainly not like to change the simplicity of the existing Copy and Paste keys.

As for response time, I find the speed of editing or opening new files while printing invaluable in preparing standard forms and letters for the coming day's work, and the ability to edit a file while it is being printed is another great timesaver.

Two operating questions I'd like to ask: What is the difference between the Unit and the Doc keys? (when experi-menting they seem to have the same effect on the cursor).

And how, without switching off and starting over again, does one abort from a Print instruction? Or is this something else which is taken care of in version 1.2?

Incidentally, you and Gabriel Jacobs didn't say that the version number is shown on the title page, which is probably the most reliable way of finding out which version you are using.

Also in your replies in Postbag you mention seeing demonstrations of various software programs to enable a sensible selection of a program to fit your own requirements, but where can this be done?

Stores say they don't have this facility. I feel I've certainly had my money's worth from this month's magazine and if next months Business Computing is as good you'll have a new subscriber.

Margaret Searle.

● *Yes, the Cut key can be used effectively with a selected range of deletion. Like many of LocoScript's features, this one was taken from Dec's VAX/VMS EDT program. But EDT also has Delete Line and Delete Word facilities. All I was saying was that it would be useful to have these in LocoScript too.*

I agree that Paste and Copy are two of the strengths of LocoScript. But I still think, from a purely ergonomic point of view, that cursor movements ought as far as possible to be contained in unshifted keystrokes.

It would hardly make Past and Copy less simple if they were in shifted position.

There is no difference between the Unit and Doc keys if you haven't set any markers. LocoScript will try to find a Unit, starting from the cursor position to the end of the document.

If no marker has been set it will go to the end of the document while looking for one, thus giving the same effect as the Doc key.

To set a marker, use the Set menu or + UT (the plus sign on the left of the spacebar, of course). To delete it use the Del key.

To abort a Print instruction, press Prt then f7 for a 'clear and re-set'. This works on all versions of LocoScript.

The reason why I didn't mention looking at the Disk Management screen to see which version of LocoScript you have is that a Start of the Day disk, with all the system files copied on to it - the one the manual recommends you should use - may not contain a disk-identity label, unless you have added it in yourself.

The most reliable way of telling which version you have is therefore to try out one of the new features of Version 1.2.

Dealers are not all they should be, particularly since High Street stores have started selling micros in a big way. You have to be strong and insist on demos before buying. If everyone did the service we would get would be that much better.

Glad you like Business Computing with the Amstrad. We do our best to please.
Gabriel Jacobs.

System Disks Bugs

I recently purchased a PCW 8256 and have found quite a few faults with the programming, especially in the keyboard.

Before I bought it I read that Amstrad had experienced some problems with the system disks supplied with the 8256, and that if anyone had bought one of the 'bugged' programs, they would replace it free of charge.

Can you help me?

E.W. Thorley

● *The original version of LocoScript did contain some bugs and some omissions.*

It was impossible, for example, to print a document from any page and to generate a straight Ascii file - in some circumstances the program actually crashed.

Amstrad quickly corrected the errors and offered a new version - 1.2 - free of charge to existing PCW owners, together with an upgraded CP/M Plus - version 1.4 - on side B of the new Locoscript disk.

However, since you bought your PCW recently it's unlikely that you have been supplied with an early version of the software - check that your Locoscript disk is labeled V1.2.

In any case the bugs and omissions from the early version did not affect the keyboard. In your case it sonds as if the hardware is at fault.

I suggest that you take your Locoscript disk back to your dealer and try it on a PCW 8256 that he should have on display. If the same bugs recur obviously there is something wrong with your software.

If not, the hardware is indeed the culprit. In either case the dealer should provide replacement or repair under the guarantee.

Use bigger files with Word Star

Here's a little known technique, not included in the Pocket Word Star documentation, for increasing maximum file size safely.

Normally a WordStar file cannot be bigger than about a third of available disk space because it also requires a third for its working temp (.$$$)file, and a third for its back-up (.BAK) file. The figures depend to some extent on the amount of RAM available, but a third is not usually far out.

If you run out of disk space while editing, WordStar will delete the old one, but this is obviously dangerous. Should something go wrong at that moment you'd have no back-up file perhaps no file at all if there's been a mains spike or the computer has been powered down for some reason.

However if you have two drives - either two physical drives, or one physical drive and a RAM disk as on the PCW 8256 - available disk space, and therefore safe maximum file size, can be increased by swinging from drive to drive with each save. The method allows WordStar to create its back-up file on a drive not being used for the file itself.

Suppose you've opened a file on drive A. The idea is to create the back-up file on drive B at the first save (with Ctrl + KS), to write the file itself to drive B at the next save with the back-up now on drive A, and so on, backwards and forwards until a Ctrl + KD, KX or KQ is issued.

The net result will be that you can have files which are two-thirds the size of the free disk space, rather than only a third.

You achieve this by including the second drivename after the name of the document to be edited when you take the D or N options at the opening menu. Put a space between the filename and the second drivename, like this:

```
A:MYDOC.TXT B:
```

It will be displayed on the status line which, when the main file is being written to drive B, will read:

```
B:MYDOC.TXT A:
```

so you know precisely where you're up to.

Unless you have an 8512 or an 8256 with double physical drives, and therefore in effect three drives, this method assumes that the file size is not greater than the amount of space left on the system disk, where the Pocket WordStar command overlay files are held.

In any case, always open the file initially on the drive with the most room available, so that the temp file will be created there.

Getting to grips with foreign accents - the PCW way

WHAT do Portuguese love letters, Danish dairies and Italian novels all have in common? Why Joyce, of course. One of the lesser publicized aspects of the PCW is its ability to run a variety of foreign language sets under CP/M, and its access to a wide range of foreign characters and accents under LocoScript thanks to its additional shift keys of Alt and Extra.

I remember a writer friend of mine being overjoyed when he discovered, not that he could successfully print out the manual's recommended letter to Fred, but that he could print in italics, complete with accents, the phrase *coup de théâtre*.

Apart from its low-cost attraction to anyone doing any kind of word processing, these foreign characters make the machine particularly appealing to anyone involved in translation work of any kind. Mogens Bech runs a Danish translation service, and was immediately won over by the PCW's potential.

"In Denmark the price of the machine to us is the same as that of a good electronic typewriter, but for that of course you get much much more. We had previously been using Brother daisywheel typewriters, on which there is no difficulty in using the different language typefaces that we need for our business, but we wanted to upgrade to full word processing facilities. Our only other choice when the PCW came along was to opt for an IBM system which would cost up to five times as much.

"I haven't seen any word processing system that serves us as well in Denmark. The machine is imported by a Danish company and re-programmed so that the main keyboard is in Danish, but the other character sets and accents are still available as in the English version. The re-programming has not been too wonderful though, and I think Amstrad should look into that.

"I represent a group of translators who work mainly on business material in most of the European languages, and in addition to my own machine I have instigated the sale of a further 16 machines to other translators.

"We translate a lot of technical documents to do with food technology plants, dairies and other aspects of Danish industry. If a company is hoping to sell machinery or even build a complete plant abroad, we would translate the documents from Danish into French, Spanish or whatever language was wanted".

Mr. Bech's only major complaint over the print quality produced by the PCW printer, which he says is acceptable, but for professional purposes the machine really needs to be connected up to a printer producing letter quality rather than merely near letter quality results. His company, the Alliance Sprogservice of Aarhus in Denmark, has got the elusive interface but it is not proving compatible with the printer they want to link up to. "Still", he says, with great resignation, "I am sure we will fix it in time".

From the industrial to the intellectual, and author Stuart Hood, whose first task on his new PCW was to translate into English a novel by the Italian writer and film-maker Pasolini. Though Stuart didn't buy the PCW specifically for this purpose, he nevertheless found it useful to have a range of accents on the keyboard.

Although he is translating into English, many of the proper names remain unchanged and need to be accented and this can be done on the screen, where on previous translations done on the typewriter he would have to go through an entire typescript and add the accents in by hand. Another unexpected bonus was the Exchange facility.

"There was a term in the book which I wasn't quite sure how to translate, so I did it one way, but then when towards the end I decided it ought to be translated slightly differently, I was able to go through the manuscript and change every occurrence, which was delightfully easy.

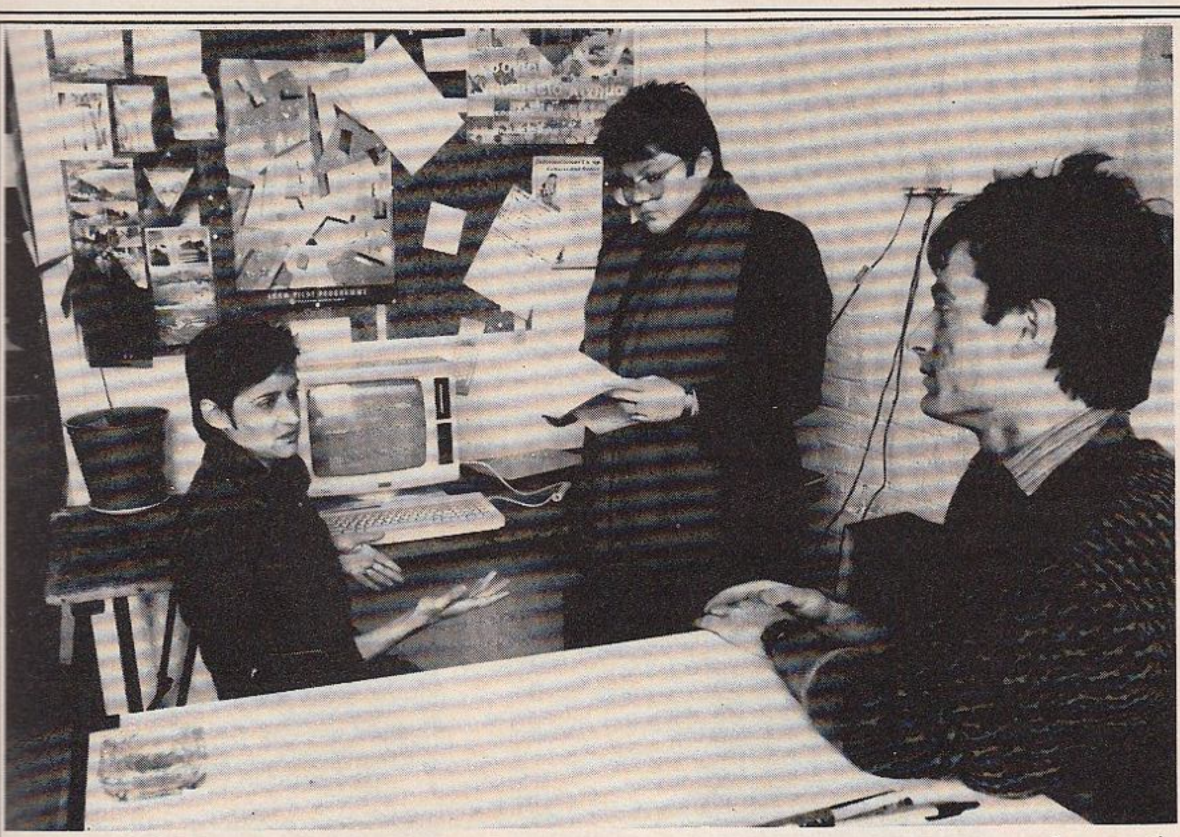
"The machine does make all my work easier. What I tend to do is write a preliminary version of something and run this off in draft mode, which is very quick, so it provides me with an in-between translation if you like. I then go back and correct this on the screen before printing out the final high quality version for sending off to the publisher.

"Many of the changes I make are only small - where I decide later that a different meaning would be more appropriate for one particular word on a page. Now I don't have to re-type the entire page.

"It took me somewhere between 10 and 24 days to get to grips with the machine, partly because of the infernal manual, but it's no real problem now. I found no problem in adapting from the typewriter to working at a keyboard and straight on to a screen - it would defeat the object of it if you didn't use it as it's meant to be used".

'In companies where work is intensive but funds are limited the PCW is a godsend'

One of the delights of the PCW is



Members of the Triangle Translation co-operative at work in their North London offices.

that there is no one way in which it is meant to be used, and its potential is only just starting to be realized. In companies where work is intensive but funds are limited, the PCW is a godsend, or in the case of the Triangle Translations cooperative it was sent by the London Cooperative Enterprise Board, which bought 10 PCWs for distribution among the co-operatives that they advise and fund.

Triangle was lucky enough to get one of them, and though the machine is versatile it can't as yet help them with the Chinese, Vietnamese, Hindu, Urdu and other texts that they deal with on behalf of ethnic groups in north London.

By Mike Gerrard

It helps considerably with the other side of their work, however, which tends to be banking, legal, trade union and general business translations both into and out of the various European languages - not without a few difficulties, however, as Jutta Schettler explains:

"One thing that irritates me is that in German if you want to put an Umlaut over a capital letter the machine shrinks the letter slightly to make room for the Umlaut. Mostly this is fine, except for the capital letter O, which it shrinks to lowercase size so that it looks as if you've made a mistake", she said.

Another complaint is that if there were just another three characters on the keyboard, the PCW would provide a full Turkish alphabet, while the Greek alphabet is some way short of being complete, being limited at present to those characters having mathematical relevance. Another of the members of the Triangle cooperative, Stephan Jaeger, points out that there is also a symbol missing from the alleged full French character set:

"It won't do the character which combines an o and an e into one letter, although the similar one which a and e is there".

Despite the complaints Triangle is delighted with the versatility of the machine. "If you have a piece of text that has to go out in few different languages", says Jutta Schettler, "it looks good because you can maintain the same layout and so each translation will more or less correspond, and that is very difficult to do using a typewriter.

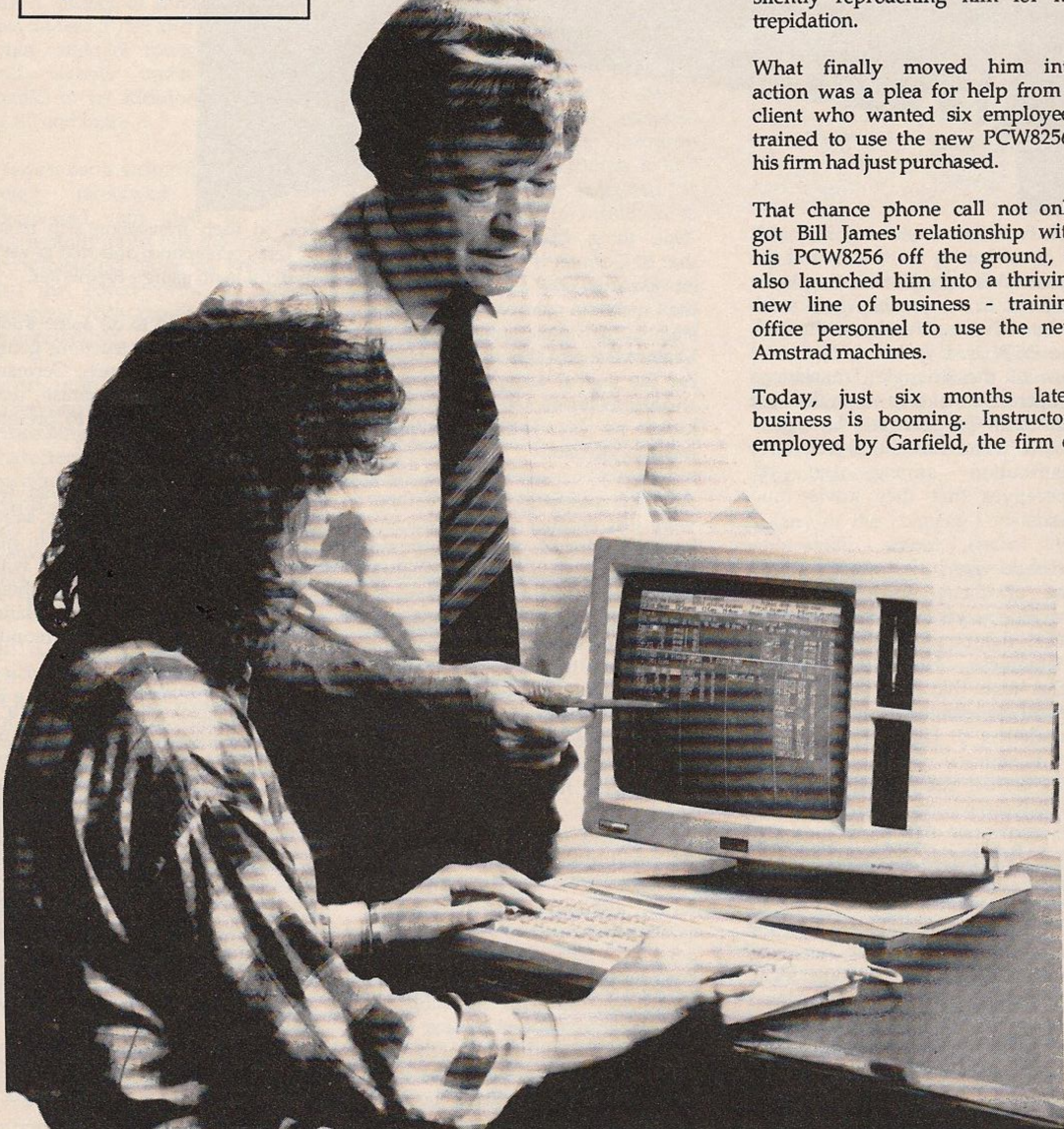
"We recently had to do some trade union work with instructions going out to drivers in French, German, Italian, English and Spanish, and it was good to be able to present those in a uniform manner.

"Lots of the translations that we do are also very much the same every month, and here you can combine the translation work with the word processing without having to re-type a lot of the same material month after month, which we did before".

One job which didn't turn into a monthly commission, although no one at the cooperative knows why, is the love letter mentioned at the start of this article. A Romeo had returned from a summer holiday, and wanted to write a letter to the love of his life in her native Portuguese, which Triangle was happy to help him with. That Romeo never returned, but if any of you out there have similar requirements this summer Triangle will be happy to oblige. And if you've ever wondered what the price of love is, it's about $100 per 1,000 words. Even with Joyce's help.

PCW8256 was the start of something big for Bill James

By Tony Leah



With 30 years experience dealing in office equipment behind him, Bill James was somewhat surprised to find himself behaving like a bashful typist toward his newly acquired PCW8256.

After rushing out to buy one of the first models available - "because I thought it was the best thing since sliced bread" - he'd only had the machine out of its box twice in a period of several weeks. The rest of the time it just sat there, silently reproaching him for his trepidation.

What finally moved him into action was a plea for help from a client who wanted six employees trained to use the new PCW8256s his firm had just purchased.

That chance phone call not only got Bill James' relationship with his PCW8256 off the ground, it also launched him into a thriving new line of business - training office personnel to use the new Amstrad machines.

Today, just six months later, business is booming. Instructors employed by Garfield, the firm of which James is managing director, are teaching PCW8256 skills to the uninitiated in all parts of the country from Aberdeen to Bristol.

One typical week recently saw Grayfield clients in Uxbridge, Battersea, Southampton, Birmingham, Wellingborough and Maidenhead receiving individual training in their own offices, while one-day group courses were taking place at venues in London, Manchester, Birmingham and Inverness.

Grayfield's new "classroom" image is a far cry from when the company was formed five years ago to provide contract drafting services to firms in the oil industry. But with most of these companies based in Aberdeen or London, Grayfield people have always done plenty of travelling.

It wasn't long before the company moved into computer aided design and became involved in providing training facilities to drawing office personnel unfamiliar with computers and CAD. Grayfield also supplied office consumables, electronic typewriters and word processors, offering training for purchasers of the latter equipment.

It was one of these former word processor training clients who called Bill James last December with the question that pointed Grayfield in a fresh direction: "Can you train six typists on my new Amstrad PCW8256 machines?"

Galvanized by the request, James ripped the packing off his own PCW8256 and, aided by Grayfield's senior word processor instructor Merilyn Large, embarked on a 10 day, 100 hour project to prepare a "logical, intensive one day training syllabus for LocoScript training".

The following week the company tried out its training scheme on the six typists and there was delight - and relief - all round when, at the end of the day, the girls said they felt in control of the Amstrads instead of being afraid of them.

The same syllabus is still used today with only minor modifications.

'...at the end of the day the girls said they felt in control of the Amstrads instead of being afraid of them'

It is an intensive course of instruction and it is assumed that all those who take it are familiar with a standard typewriter keyboard and can produce a typewritten letter whether by touch or using two fingers.

Two types of training are offered by Grayfield. There are classes in London every Wednesday and Thursday, and in other major centres to suit demand. This usually averages two a month in Birmingham and Manchester and one a month in Bristol, Glasgow and Aberdeen.

The alternative to classes is individual tuition in the client's own office using his PCW and company stationery for practical purposes.

Whether the client has one or several employees to be trained is of no consequence - he has sole use of a Grayfield instructor for the day. "Flexibility is the name of the game - a lesson we learned when dealing with the oil industry", says Bill James.

Grayfield fixed its training charges to take into account the comparatively low selling price of the PCW8256. "The standard going rate elsewhere for computer and word processor training ranges from $300 to $500 a day", says James.

"To charge this figure for a machine which costs only $1500 would not appear very attractive to the prospective trainee. In view of this our class training cost has been fixed at $135 a head and individual training at $235 ".

A help line is available to all who have completed LocoSCript training with Grayfield, should any subsequent problems arise. "As only three query calls have been received I assume the other participants have either become proficient operators or they've thrown their PCWs out of the window", says James.

The company helps clients prepare complicated templates for stationery, a free service to all who have completed a course of tuition. And, as a result of requests from satisfied trainees it now provides supply, installation and training facilities for accounts and payroll packages, filing systems, customer record systems, mailing systems and telex packages.

James finds it personally satisfying to compare the apprehensive faces of new students with their expressions at the end of the day "when realization has dawned that they can now make the Amstrad sing and dance".

Having started his working life 30 years ago as a typewriter and accounting machine salesman, Bill James retains a keen interest in the marketing of new technology. As a result he has prepared a short demonstration program to show what he feels are the best sales points for retailers who stock the PCW8256.

He has made a practice of posing as a potential PCW8256 buyer when visiting Amstrad dealers who aren't specialist computer outlets.

"The average hi-fi salesman has never sold a business machine and often has little working knowledge of typewriters", James says. "Consequently his so-called sales demonstration is abysmal and he is not usually able to demonstrate the best selling points of the PCW8256 to the potential buyer.

"The customer departs little wiser and a potential sale is lost. Multiply this 200 times or so for the hi-fi shops in the home counties region alone and the amount of lost sales is considerable.

"Yet the PCW8256 is an easy machine to explain with a 10 to 15 minute demonstration which show off all its best points, after which a sale can often be closed on the spot".

Software selection: Setting the vital parameters

WHILE I was walking around the last Amstrad show, I saw some business software that was grossly overpriced as well as some more which was so awful that I believe it should be sold with garlic if financial blood-sucking is not to follow its purchase.

Nevertheless, I was most impressed with the variety of good business programs available, especially so when I consider that the Amstrad is one of the newer home computers.

It therefore astounded me to overhear visitors repeatedly complaining that there was a vast range of software aimed at the amusement market while the commercial sector was very poorly served. I had hitherto been convinced that many potential users were merely finding difficulty in making wise selections from what was available.

These comments are a representative sample.

"I could write better in a morning ..."

"That package costs more than my printer!"

"It's a bit elementary, what about ...?"

Despite the evidence of my ears, I was not convinced that there was a shortage, but the experience made me reconsider my opinions.

Probably nothing would have come of these musings if it had not been for the fact that a couple of days later the editor wanted by comments on two business programs submitted by readers. I was at first reluctant to do so.

While I am more than happy to respond to general queries, it is extremely time-consuming to perform a thorough test on business software, and the last thing I desired was to set a potentially open-ended precedent. Nevertheless I owed him a few favors and thus agreed.

I am glad I did, because the exercise proved just how little the average businessman knows about the problems of software producers.

Amateur attitudes

If you understand these problems not only will you accept that it is often unrealistic to expect to find a system which is a perfect match with your organization's requirements, but also you'll improve your selections from what is on the market. I'll use several personal, amateur and professional examples to illustrate this point.

The software producer's lot is not a happy one. Both amateur submissions demonstrated different aspects of the same fundamental fact. This is that system creators are faced with a complex series of mutually exclusive choices while requiring considerable amounts of time, money, talent and not a little luck before any profit is realized.

The first of these submissions was a suite of four programs intended for listing in this magazine. Unfortunately while a delight in every respect, they were far too long for these pages. I suggested that the package could be a commercial success if marketed properly.

The other offering was very short and intended for sale in the High Street. The creator was seeking our advice on how to go about commercializing his product. You can therefore imagine my surprise on finding that it kept crashing my system.

Even assuming it can be made to run reliably, I doubt the venture would every be profitable. Nevertheless the concept has some merits, therefore if my suggestions are followed you may yet see the program appear as a listing in Computing with the Amstrad.

In summary, I had received two programs, both of which were originally targeted for totally inappropriate slots in the market.

Professional attitudes

Moving into the marketplace generates a variety of problems which frequently have little to do with one's systems and programming abilities, hence the flak I sometimes receive from producers of software I have reviewed in this column.

No matter what I write, they complain. If they do not agree with my comments they cite satisfied clients. If they like my remarks they still object to my not giving them a quote they can use for marketing.

I fully understand their concerns. If you consider that over 90 per cent of all projected business systems never hit the shop shelves, and then remember that the creator's problems are usually multiplied once retailing starts, it's obvious that they will become excessively protective. Not only do I sympathize with them, I know their problems first hand. Nevertheless since you are good enough to read this column, my role cannot be to provide advertising copy, but to guide the general reader.

Design problems - phase 1

Let us look at how the examples

sent in by two of the readers demonstrated some of the problems the prospective software tycoon faces. While explaining why it is so difficult to buy packages that precisely fit your needs, this will also serve as a warning if you should ever feel you can do better.

The first problem is that it is all too easy to believe that just because you are delighted with your brainchild the world will beat a path to your door in order to obtain a copy. The following two factors are frequently overlooked:

● If you create a system to suit your organization it will almost invariably incorporate elements which are highly specific to your requirements. These factors may well render it awkward or even unusable in different situation. For example an invoicing system developed in a small foundry may be utterly useless for a tailor and a payroll system incorporating piece rates may prove clumsy in firms where everyone is on a fixed weekly rate.

● The creator is the person most familiar with a system's operation and this invariably disguises the quirks which have been introduced. Even assuming they maintain a perfectly logical approach, this is no guarantee others will find it equally user-friendly.

The next problem to be overcome is that merely making the software more general or simpler to use, whether by ejecting the site specific features or by eliminating the more complex tasks, does not guarantee greater user acceptance.

A depressingly large percentage of purchasers select their software on the basis of which has the most features. This is despite the old maxim: *"The greater the number of options the system has the more clumsy will be the software"*.

Allowing for some suites using more efficient coding than others, this maxim holds true both in terms of the software's performance and its user-friendliness.

Equally true is that the more the software offers the more the costs must rise. Not only will the package be more expensive, but more significantly the costs of your time as you learn the system, set up its files and enter the data will mount dramatically.

This is why I do not hesitate to repeat in this column that you should select your systems on the basis that performing 80 per cent of the necessary tasks efficiently and simply, thereby not building up a resistance to keeping up to date, is vastly preferential to using a system offering 30 per cent more facilities than you need in order to meet every requirement.

JO STORK
examines the problems inherent in matching business software to your organization's requirements

If you find a system which offers precisely 100 per cent of the features you require, then this is probably owing to no more than good fortune that your needs are so standard or close to the organization the designer had in mind.

My guess is that the show visitors were actually complaining that they could not find software which was a perfect match, rather than there was very little there.

Software authors are constantly trying to second guess what these standard system needs are for any given application. Fifty per cent of the potential offerings fall by the wayside at this stage, merely because they fail to do so.

Design problems - phase 2

Having finally steered a course through these conflicting requirements, the authors meet a totally different set of problems. In many cases they are not equipped to handle them. Some are:

● The need for market research to determine the sales levels that may be achieved at any given selling price.

● The need for accurate production costings at each sales level. It is depressing how many times these are grossly underestimated by persons whose primary expertise is computing. Whether they wish to recoup their systems and programming effort is for them to decide.

Many software writers keep the eventual selling price low by charging these development costs to the original organization for whom the system was first written. In other words they make no attempt to recover these costs. In government circles this is often called creative accounting.

● Having done this work they may, if they are lucky, have a series of revenue and cost estimates which show an acceptable level of profit at some level of sales. A decision now has to be taken whether to proceed with a production and marketing operation. If this point is at 100,000 sales the decision is quite different to one at 750 units.

● The program may need speedy conversion to run on other micros if a reasonable return on investment becomes possible.

By now a further 30 per cent of potential suppliers have dropped out of the contest for your money.

If you believe that all that is needed now is to deliver a fully tested product which does not crash reviewer's systems, please think again. Leaving aside obvious details such as documentation, there are still a whole pack of commercial hazards to be overcome.

More problems

A further 10 per cent will have fallen by the wayside at the coding stage, thus your selection will be limited to those suppliers who can handle commercial chores which have nothing to do with computers.

A brief sample includes keeping production and administration costs under control, ensuring quality is maintained, ensuring sales meet the desired level and responding to user queries.

These tasks have been responsible for more ulcers than anything else I can think of, since if any are not completely mastered all the earlier profit estimates will become totally null and void.

You now understand the pitfalls of becoming a software writer and also why at times I must, on your behalf, disagree with some of the judgements they have made.

With this set of guidelines we can now turn to studying how they apply to software you may be considering. Before reviewing two products currently on the market it will be easier to consider a couple of common and essential programs no matter what the organization's business - a word processor and a spreadsheet.

Personal example

Let me cite a classic case of practising what I preach. It demonstrates how small can be beautiful.

Apart from my 6128 and 664, I need four other computers including an IBM PC with more memory than my Amstrads can store on their twin-disks.

Even though the PC has a very powerful word processor these articles plus the bulk of the documents, handbooks and reports which are my main business were until recently produced using a very basic Amstrad word processor on the 664.

It had less than half the functions of my IBM software, did not provide spooling and was significantly slower when cutting and pasting, but I preferred it because it was exceptionally easy to use.

I am also told that the IBM's keyboard is vastly superior to the Amstrad's but I am not a good enough typist to say I can notice much difference.

Only three occasionally-needed facilities were missing. When these were essential I obviously used the PC. I considered switching to a ROM-based word processor, but decided against it. I would have gained the speed and one of the facilities but lost the advantages of simplicity - and by now considerable familiarity.

When the 6128 version of my software was produced I did upgrade. I gained two of the missing features and did not lose the user-friendliness. Since this conversion I have not had occasion to use the PC word processor at all.

This fortuitous set of circumstances does not always apply, even with such general software. I frequently need to process spreadsheets. The market leader for the Amstrad is certainly easy to use, and can hold matrices far larger than I will ever need. Nevertheless, I normally use the PC for this work.

The reason is quite simple. Many of the cells require mathematical functions which are far more complex than can be handled by the Amstrad software. I am forced to use a much more complex and therefore clumsy spreadsheet than I would like in order to gain this mathematical ability. The speed gain from using a PC is marginal with small spreadsheets.

Conclusion

As ever the secret of wise selection is to ensure that the vital criteria are ideally matched. Any other requirements which can be met should be regarded as bonuses, provided that the do not thereby also add to the drawbacks.

While this makes it far easier to achieve for general applications, for example basic word processors and spreadsheets, the principles do not change just because one turns to more specific applications.

Hopefully these criteria are now understood and you will understand the reasoning behind my reviews of the following products.

Next month in Business Computing With The Amstrad:

We review PlannerCalc and Scratchpad Plus, show you how to breathe life into dull stationery - there's more on Mini Office II and don't miss our special desk-top publishing feature. Naturally we'll have more of your letters as well as the latest news from here and overseas. See you next month!

GALACTIC INVADERS

By
ROBIN NIXON

REMEMBER the good old days when all there was in the way of video games was ping pong? And then Space Invaders appeared and you couldn't get on a machine for love nor money.

Well, for all nostalgia lovers, Computing with the Amstrad proudly presents a re-vamped version of this classic game. It is written in 99 per cent machine code with only the title screen in Basic.

As ever, your mission is to prevent Earth from being invaded by hostile aliens, your only defences being five space cruisers and your own reflexes.

The machine code is held in the data statements starting at line 590. If you renumber the program you must change the value of lin in line 550 to the first line of data. This is to ensure that the internal data validation routine works correctly.

Because this sort of validation routine is not 100 per cent infallible make sure you save the program before you run it.

USEFUL POKES

&86C4	This location contains the number of lives. If you poke it with more than 10 you will destroy the display.
&86C1	Level of difficulty. Try poking it with 100!
&8198	Alien's speed. I bet you can't shoot them all if this is poked with 1
&8200 &8293	Both these locations contain the speed at which your ship moves. The larger the number poked the slower it goes. Both locations must contain the same number.
&80EC &8142 &86BF	All these locations must contain the same number. A multiple of 10 between 10 and 80. This is the Number of aliens you start up with on screen.

```
100 REM ***********************
110 REM *                     *
120 REM * Galactic  Invaders *
130 REM *                     *
140 REM *   By Robin Nixon    *
150 REM *                     *
160 REM * (c) Computing with *
170 REM *    the  Amstrad     *
180 REM *                     *
190 REM ***********************
200 REM
210 MODE 0:CALL &BC02:DEFREAL p
220 DEFINT a-o,q-z
```

☞

```
230 FOR x=1 TO 5:POKE x+&86CB,0:NEXT
:POKE &86C1,5:POKE &86C4,5:POKE &86B
F,20:POKE &86BD,0
240 CALL &BB4E
250 hs=0
260 MEMORY &7FEF
270 GOSUB 550:CLS
280 ENV 1,0,15,2,1,0,1,5,-3,15
290 ENV 2,0,15,1,1,0,1,5,-3,7
300 ENV 3,0,15,50,1,0,1,5,-3,50
310 ENT 1,15,-20,1,15,15,1
320 CALL &BC02
330 BORDER 0
340 PLOT 1000,1000,2:MOVE 0,0:DRAW 6
38,0:DRAW 638,398:DRAW 0,398:DRAW 0,
0
350 PEN 1:LOCATE 2,2:PRINT"GALACTIC
 INVADERS"
360 PEN 3:LOCATE 7,5:PRINT"CONTROLS"
370 PEN 6:LOCATE 4,7:PRINT"Z left X
right"
380 LOCATE 6,9:PRINT"SHIFT fire"
390 PEN 9:LOCATE 6,13:PRINT"HIGH SCO
RE"
400 hs$=STR$(hs):hs$=STRING$(6-LEN(h
s$),"0")+RIGHT$(hs$,LEN(hs$)-1)+"0":
PEN 11:LOCATE 8,15:PRINT hs$
410 PEN 2:LOCATE 2,19:PRINT"(c) Comp
uting with"
420 LOCATE 5,21:PRINT"the  Amstrad"
430 PEN 10:LOCATE 4,23:PRINT"By Robi
n Nixon"
440 WHILE INKEY$<>"":WEND:WHILE INKE
Y$="":WEND:CLS
450 BORDER 1:INK 0,0:INK 5,1:INK 13,
6
460 PLOT 0,0,13:DRAW 638,0:PLOT 638,
0,3:DRAW 638,398:DRAW 0,398:DRAW 0,0
:MOVE 0,366:DRAW 638,366
470 LOCATE 9,2:PEN 4:PRINT"Score";:L
OCATE 19,2:PEN 1:PRINT"0";
480 CALL &7FF0
490 IF PEEK (&869A)<2 THEN FOR x=1 T
O 2000:NEXT:CLS:GOTO 450
500 FOR x=1 TO 200:BORDER RND*27:INK
 RND*15,RND*27:NEXT:CALL &BC02:CLS
510 IF PEEK (&86FD)=1 THEN GOTO 530
520 GOTO 450
530 sc=0:FOR x=0 TO 4:sc=sc+10^x*(PE
EK(&86CC+x)):NEXT:IF sc>hs THEN hs=s
c
540 GOTO 320
550 MODE 2:PRINT"Checking data...":l
in=590:p=&7FF0:FOR x=1 TO 365:a=0:FO
R y=1 TO 8
560 READ d$:d=VAL("&"+d$):POKE p,d:p
=p+1:a=a+d
570 NEXT:LOCATE 1,3:PRINT"Line";lin;
:READ c$:c=VAL("&"+c$):IF c<> (a MOD
 &100) THEN PRINT"incorrect!":END
580 PRINT"ok":lin=lin+10:NEXT:MODE 0
:RETURN
 590 DATA F3,ED,5F,32,C5,86,21,55,32
 600 DATA CF,22,08,86,21,34,C2,22,B8
 610 DATA 0A,86,3E,01,32,92,86,3E,57
 620 DATA 00,32,9C,86,32,FB,86,32,39
 630 DATA FC,86,32,FD,86,06,50,21,AE
 640 DATA 0C,86,36,01,23,10,FB,21,18
 650 DATA 9F,86,06,0A,36,00,23,10,9E
 660 DATA FB,21,01,D0,E5,FD,E1,3A,EA
 670 DATA C4,86,47,C5,ED,4B,00,87,15
 680 DATA 11,02,87,CD,DC,84,01,06,CE
 690 DATA 00,FD,E5,E1,09,E5,FD,E1,8F
 700 DATA C1,10,E8,3E,02,CD,72,BB,F3
 710 DATA 06,04,2E,12,DD,21,C7,86,95
 720 DATA C5,E5,7D,CD,6F,BB,DD,7E,79
 730 DATA 00,C6,30,CD,5A,BB,E1,2D,E6
 740 DATA DD,23,C1,10,EB,3E,00,32,2C
 750 DATA 9A,86,CD,BB,81,ED,5F,3F,B4
 760 DATA DE,00,30,03,CD,C5,85,3A,62
 770 DATA FB,86,A7,28,36,FE,01,20,A5
 780 DATA 05,2A,F5,86,18,15,FE,02,D7
 790 DATA 20,05,2A,F7,86,18,0C,FE,EE
 800 DATA 03,20,05,2A,F9,86,18,03,EC
 810 DATA 2A,F7,86,22,FF,85,23,23,93
 820 DATA 22,02,86,CD,D9,85,3A,FB,0A
 830 DATA 86,3C,FE,05,20,02,3E,01,26
 840 DATA 32,FB,86,ED,5F,32,C5,86,7C
 850 DATA 3A,9A,86,FE,02,20,51,2A,F5
 860 DATA 08,86,ED,4B,40,89,11,42,E2
 870 DATA 89,CD,DC,84,E5,21,EC,86,2E
 880 DATA CD,B3,85,E1,3E,00,32,BD,13
 890 DATA 86,3A,C4,86,3D,32,C4,86,C3
 900 DATA A7,20,2C,3E,14,32,BF,86,BC
 910 DATA 3E,05,32,C4,86,32,C1,86,38
 920 DATA DD,21,C7,86,FD,21,CC,86,BB
 930 DATA 06,05,DD,7E,00,FD,77,00,DA
 940 DATA DD,36,00,00,DD,23,FD,23,33
 950 DATA 10,F0,3E,01,32,FD,86,C9,BD
 960 DATA ED,5F,3F,DE,01,30,08,CD,6F
 970 DATA 68,84,3E,00,32,FB,86,3A,17
 980 DATA BD,86,21,BF,86,BE,20,2B,B2
 990 DATA 3E,00,32,BD,86,3A,BF,86,32
1000 DATA C6,0A,32,BF,86,FE,5A,20,BF
1010 DATA 0D,3E,14,32,BF,86,3A,C1,D1
1020 DATA 86,C6,02,32,C1,86,2A,08,F9
1030 DATA 86,ED,4B,00,87,11,02,87,DF
1040 DATA C3,DC,84,2A,0A,86,3A,9A,B1
1050 DATA 86,FE,01,C2,80,81,3E,02,88
1060 DATA 32,98,86,CD,BB,81,3E,00,97
1070 DATA 32,9A,86,32,98,86,2A,0A,D6
1080 DATA 86,01,50,00,09,22,0A,86,92
1090 DATA 3A,C3,86,3D,32,C3,86,A7,E2
1100 DATA 20,24,3A,98,86,3C,FE,02,D8
1110 DATA 20,02,3E,00,32,98,86,3E,EE
1120 DATA 07,32,C3,86,E5,21,E3,86,F1
1130 DATA CD,B3,85,E1,3A,92,86,A7,DF
1140 DATA 28,03,23,18,01,2B,22,0A,BE
1150 DATA 86,3E,42,CD,1E,BB,CA,6D,E3
1160 DATA 80,FB,C9,DD,21,0C,86,21,F5
1170 DATA 62,86,3A,98,86,A7,28,03,12
1180 DATA 21,72,86,E5,FD,E1,3E,00,1A
1190 DATA 32,5C,86,2A,0A,86,E5,3E,F1
```

```
2420 DATA 4B,D0,88,11,D2,88,CD,DC,B7
2430 DATA 84,18,94,C5,D5,F5,DD,E5,81
2440 DATA FD,E5,CD,AA,BC,FD,E1,DD,D0
2450 DATA E1,F1,D1,C1,C9,3A,FB,86,E8
2460 DATA A7,20,3C,21,A0,D0,22,D7,8D
2470 DATA 85,3E,01,32,FB,86,C9,00,40
2480 DATA 00,2A,D7,85,3A,FC,86,A7,E9
2490 DATA 28,0D,2B,7D,FE,A1,20,12,AE
2500 DATA 3E,00,32,FC,86,18,0B,23,38
2510 DATA 7D,FE,E7,20,05,3E,01,32,F8
2520 DATA FC,86,22,D7,85,ED,4B,00,38
2530 DATA 00,11,00,00,CD,DC,84,C9,07
2540 DATA 32,CF,54,C0,01,01,01,01,19
2550 DATA 01,01,01,01,01,01,01,01,08
2560 DATA 01,01,01,01,01,01,01,01,08
2570 DATA 01,01,01,01,01,01,01,01,08
2580 DATA 01,01,01,01,01,01,01,01,08
2590 DATA 01,01,01,01,01,01,01,01,08
2600 DATA 01,01,01,01,01,01,01,01,08
2610 DATA 01,01,01,01,01,01,01,01,08
2620 DATA 01,01,01,01,01,01,01,01,08
2630 DATA 01,01,01,01,01,01,01,01,08
2640 DATA 01,01,01,01,00,00,00,00,04
2650 DATA 00,00,60,87,90,87,C0,87,45
2660 DATA 60,87,90,87,C0,87,60,87,2C
2670 DATA 90,87,F0,87,20,88,50,88,0E
2680 DATA F0,87,20,88,50,88,F0,87,6E
2690 DATA 20,88,80,88,80,88,80,88,C0
2700 DATA 80,88,80,88,80,88,80,88,20
2710 DATA 80,88,01,00,01,00,06,00,10
2720 DATA 00,00,00,00,00,00,00,00,00
2730 DATA 00,00,00,00,00,00,00,00,00
2740 DATA 00,00,00,00,00,00,00,00,00
2750 DATA 00,00,00,00,00,00,00,00,00
2760 DATA 00,00,00,00,00,00,00,14,14
2770 DATA 00,05,00,07,05,00,00,00,11
2780 DATA 00,00,00,00,00,00,00,00,00
2790 DATA 00,84,01,00,00,00,03,07,8F
2800 DATA 64,00,81,02,01,2C,01,00,15
2810 DATA 07,64,00,82,00,00,D0,07,C4
2820 DATA 00,07,08,00,81,03,01,00,94
2830 DATA 00,03,07,E8,03,90,89,00,0E
2840 DATA 8A,70,8A,00,00,00,00,00,84
2850 DATA 0D,07,00,00,00,28,00,00,3C
2860 DATA 00,00,00,00,28,00,00,00,28
2870 DATA 00,00,05,2D,00,00,00,00,32
2880 DATA 00,05,25,00,00,00,00,00,2A
2890 DATA 10,30,00,00,00,00,00,10,50
2900 DATA B0,00,00,00,00,00,46,E4,DA
2910 DATA 02,00,00,00,0A,52,F0,02,50
2920 DATA 0A,00,00,03,52,58,03,02,BC
2930 DATA 00,00,03,03,03,03,02,00,0E
2940 DATA 00,01,40,C0,01,00,00,00,02
2950 DATA 40,00,80,40,00,00,00,00,00
2960 DATA 00,80,00,00,00,00,00,00,80
2970 DATA 08,05,00,80,00,40,00,00,CD
2980 DATA 40,00,80,00,00,CC,CC,CC,24
2990 DATA 00,00,98,C3,64,00,00,C9,88
3000 DATA C0,C6,00,00,40,C3,80,00,09
3010 DATA 00,14,00,28,00,00,14,3C,8C
3020 DATA 28,00,00,00,00,00,00,00,28
3030 DATA 08,05,00,40,C0,80,00,00,8D
3040 DATA 40,00,80,00,00,50,F0,A0,A0
3050 DATA 00,00,50,0C,A0,00,00,78,74
3060 DATA F0,B4,00,00,14,24,28,00,04
3070 DATA 00,44,3C,88,00,00,CC,00,D4
3080 DATA CC,00,00,00,00,00,00,00,CC
3090 DATA 08,05,00,F0,00,F0,00,00,ED
3100 DATA E5,CF,DA,00,00,11,CC,22,8D
3110 DATA 00,00,C7,CC,8A,00,00,45,62
3120 DATA 0C,CB,00,00,14,CC,28,00,DF
3130 DATA 00,3C,00,28,00,00,00,00,64
3140 DATA 3C,00,00,00,00,00,00,00,3C
3150 DATA 08,05,00,00,C0,00,00,00,CD
3160 DATA 40,00,80,00,00,CC,CC,CC,24
3170 DATA 00,00,98,C3,64,00,00,C9,88
3180 DATA C0,C6,00,00,40,C3,80,00,09
3190 DATA 00,14,00,28,00,00,3C,00,78
3200 DATA 3C,00,00,00,00,00,00,00,3C
3210 DATA 08,05,00,C0,00,C0,00,00,8D
3220 DATA 40,00,80,00,00,50,F0,A0,A0
3230 DATA 00,00,78,0C,B4,00,00,50,88
3240 DATA F0,A0,00,00,14,24,28,00,F0
3250 DATA 00,44,3C,88,00,00,44,CC,18
3260 DATA 88,00,00,00,00,00,00,00,88
3270 DATA 08,05,00,F0,F0,F0,00,00,DD
3280 DATA 45,CF,8A,00,00,11,CC,22,9D
3290 DATA 00,00,45,CC,CB,00,00,C7,A3
3300 DATA 0C,8A,00,00,14,CC,28,00,9E
3310 DATA 00,14,00,3C,00,00,3C,00,8C
3320 DATA 00,00,00,00,00,00,00,00,00
3330 DATA 09,05,00,00,00,00,00,00,0E
3340 DATA 00,00,00,00,00,00,00,00,00
3350 DATA 00,00,00,00,00,00,00,00,00
3360 DATA 00,00,00,00,00,00,00,00,00
3370 DATA 00,00,00,00,00,00,00,00,00
3380 DATA 00,00,00,00,00,00,00,00,00
3390 DATA 0F,02,28,00,80,00,28,00,E1
3400 DATA 80,00,28,00,80,00,00,00,28
3410 DATA 00,00,00,00,00,00,00,00,00
3420 DATA 00,00,00,00,00,00,00,00,00
3430 DATA 08,01,00,00,A0,A0,A0,A0,89
3440 DATA 80,80,00,00,00,00,00,00,00
3450 DATA 08,05,00,00,40,00,00,00,4D
3460 DATA 40,00,10,00,00,88,20,88,80
3470 DATA 00,00,44,44,00,00,00,40,C8
3480 DATA 68,40,00,00,20,C4,00,00,8C
3490 DATA 00,44,94,00,00,00,80,88,E0
3500 DATA 20,00,00,00,00,00,00,00,20
3510 DATA 0F,02,00,00,00,00,00,00,11
3520 DATA 00,00,00,00,00,00,00,00,00
3530 DATA 00,00,00,00,00,00,00,00,00
3540 DATA 00,00,00,00,00,00,00,00,00
3550 DATA 07,02,00,00,00,00,00,00,09
3560 DATA 00,00,00,00,00,00,00,00,00
3570 DATA 0D,06,00,C0,00,88,40,80,1B
3580 DATA 00,80,44,CC,00,80,00,00,10
3590 DATA A0,00,A0,00,00,50,28,08,C0
3600 DATA 78,00,00,00,14,14,00,00,A0
3610 DATA 00,44,00,08,44,00,00,CC,5C
3620 DATA 04,04,44,88,00,44,00,08,20
3630 DATA 44,00,00,00,14,14,00,00,6C
```

```
3640 DATA 00,50,28,08,78,00,00,00,F8
3650 DATA A0,00,A0,00,00,80,44,CC,D0
3660 DATA 00,80,00,C0,00,88,40,80,88
3670 DATA 0C,08,00,00,50,A0,50,A0,F4
3680 DATA 00,00,00,00,F0,CC,CC,F0,78
3690 DATA 00,00,00,00,E4,8C,4C,D8,94
3700 DATA 00,00,00,00,CC,0C,0C,CC,B0
3710 DATA 00,00,00,CC,CC,CC,CC,CC,FC
3720 DATA CC,00,00,D8,E0,F0,D0,E0,24
3730 DATA E4,00,00,CC,CC,CC,CC,CC,E0
3740 DATA CC,00,00,00,CC,0C,0C,CC,7C
3750 DATA 00,00,00,00,C4,8C,4C,C8,64
3760 DATA 00,00,00,40,80,CC,CC,40,98
3770 DATA 80,00,00,C0,00,89,46,00,0F
3780 DATA C0,00,00,80,00,CC,CC,00,D8
3790 DATA 40,00,00,00,00,00,00,00,40
3800 DATA 00,00,00,00,00,00,00,00,00
3810 DATA 0C,08,00,00,F0,00,00,F0,F4
3820 DATA 00,00,00,00,A0,CC,CC,50,88
3830 DATA 00,00,00,00,E4,8C,4C,D8,94
3840 DATA 00,00,00,00,CC,0C,0C,CC,B0
3850 DATA 00,00,00,CC,CC,CC,CC,CC,FC
3860 DATA CC,00,00,C8,F0,D0,E0,F0,24
3870 DATA C4,00,00,CC,CC,CC,CC,CC,C0
3880 DATA CC,00,00,00,CC,0C,0C,CC,7C
3890 DATA 00,00,00,00,C4,8C,4C,C8,64
3900 DATA 00,00,00,00,80,CC,CC,40,58
3910 DATA 00,00,00,00,80,89,46,40,8F
3920 DATA 00,00,00,40,80,CC,CC,40,98
3930 DATA 80,00,00,00,00,00,00,00,80
3940 DATA 00,00,00,00,00,00,00,00,00
3950 DATA 0C,08,00,F0,00,00,00,00,04
3960 DATA F0,00,00,50,A0,CC,CC,50,C8
3970 DATA A0,00,00,00,E4,8C,4C,D8,34
3980 DATA 00,00,00,00,CC,0C,0C,CC,B0
3990 DATA 00,00,00,CC,CC,CC,CC,CC,FC
4000 DATA CC,00,00,D8,D0,E0,F0,D0,14
4010 DATA E4,00,00,CC,CC,CC,CC,CC,E0
4020 DATA CC,00,00,00,CC,0C,0C,CC,7C
4030 DATA 00,00,00,00,C4,8C,4C,C8,64
4040 DATA 00,00,00,00,80,CC,CC,40,58
4050 DATA 00,00,00,00,C0,89,46,C0,4F
4060 DATA 00,00,00,00,40,CC,CC,80,58
4070 DATA 00,00,00,00,00,00,00,00,00
4080 DATA 00,00,00,00,00,00,00,00,00
4090 DATA 0C,08,00,00,00,00,00,00,14
4100 DATA 00,00,00,00,00,00,00,00,00
4110 DATA 00,00,00,00,00,00,00,00,00
4120 DATA 00,00,00,00,00,00,00,00,00
4130 DATA 00,00,00,00,00,00,00,00,00
4140 DATA 00,00,00,00,00,00,00,00,00
4150 DATA 00,00,00,00,00,00,00,00,00
4160 DATA 00,00,00,00,00,00,00,00,00
4170 DATA 00,00,00,00,00,00,00,00,00
4180 DATA 00,00,00,00,00,00,00,00,00
4190 DATA 00,00,00,00,00,00,00,00,00
4200 DATA 00,00,00,00,00,00,00,00,00
4210 DATA 00,00,00,00,00,00,00,00,00
4220 DATA 00,00,00,00,00,00,00,00,00
4230 DATA 00,00,00,00,00,00,00,00,00
```

PIPELINE

COMPUTER ACCESSORIES

P.O. BOX 288, MORISSET, NSW 2264
TELEPHONE: [049] 732754

MasterCard

ALL PRICES INCLUDE P&P WITHIN AUSTRALIA

SOFTWARE

BONZO SUPER MEDDLER from Nemesis - $25.00 (C)
Developed as a dedicated tape to disk utility, BSM will transfer all standard Basic, Binary & Ascii files, BSM will also handle some Header-less files and Flashloaders. Comes with comprehensive instructions and additional utility functions.

BONZO CLONE ARRANGER $22.00 (C)
Designed primarily for archival purposes BCA will automatically transfer a full disk to tape. BCA has additional functions including rapid format, directory utility and a disk copier which can read those 'funny' formats using 41 or 42 tracks.
Both BSM and BCA are easily transferred to disk.

BONZO NEWS - $2.00
Issue six has 6 pages of news, hints and tips on the use of BSM and BCA.

PRIDE UTILITES software - prices for limited time only.

ODD JOB	$36.50	TRANSMAT (D)	$36.50
PRINTER PAC II (D)	$36.50	PRINTER PAC II (C)	$27.50
F.I.D.O. (D)	$34.00	SUPERSPRITES (D)	$34.00
SUPERSPRITES (C)	$24.00	SYSTEM X (D)	$25.00
SYSTEM X (C)	$16.00		

EDUCATIONAL

JUNIOR PRIMARY ACTIVITIES	(Disk only)	$42.00
LOGIC GAMES	(7 plus - Disk only)	$42.00
WORD GAMES	(6-12 - Disk only)	$42.00
STORY BOOK	(Mid- primary WP Disk only)	$42.00
SPELLING TUTOR	(7-12 - Disk only)	$42.00
NUMBER FACTS	(Disk or Tape)	$28.00
WORD SEARCH	(Disk or Tape)	$28.00

BOOKS

UNDERSTANDING AND EXPANDING YOUR AMSTRAD	$32.00
THE AMSTRAD DISC COMPANION	$28.00
POWERFUL PROGRAMMING FOR AMSTRADS	$25.00
MASTERING THE AMSTRAD PCW 8256/8512	$32.00
ADVANCED AMSTRAD CPC6128 COMPUTING	$28.00

HARDWARE

KDS MINI-MAX MODEM 300/300 1200/75 75/1200	$288.00
KDS SERIAL INTERFACE	$158.00
CONNECTING LEAD FOR MODEM/RS232	$14.50
KDS 8-BIT PRINTER PORT	$54.50
KDS POWER CONTROLLER	$135.00
KDS PRINTER 'T' SWITCH	$70.00
KONIX SPEED KING JOYSTICK	$35.00

TRADE ENQUIRIES WELCOME

PIP, your friendly in-house software switch

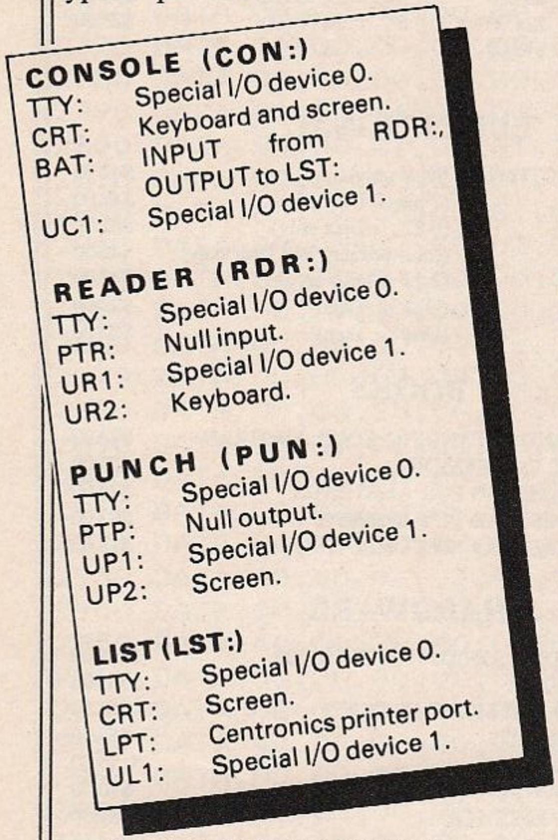
Part IV of COLIN FOSTER'S exploration of CP/M 2.2

CP/M performs all it's input/output between different logical devices. These are:

CON:	The console
LST:	List device
PUN:	Punch device
RDR:	Reader device
A:,B:	Disk drives

The PUN: and RDR: device names are a hangover from the days when computers used high speed punched paper tape for loading and saving data.

Nowadays most computers use these names to refer to the RS232 serial communication ports which are used to talk to other computers via modems, and to some fancy types of printers.



CONSOLE (CON:)

TTY:	Special I/O device 0.
CRT:	Keyboard and screen.
BAT:	INPUT from RDR:, OUTPUT to LST:
UC1:	Special I/O device 1.

READER (RDR:)

TTY:	Special I/O device 0.
PTR:	Null input.
UR1:	Special I/O device 1.
UR2:	Keyboard.

PUNCH (PUN:)

TTY:	Special I/O device 0.
PTP:	Null output.
UP1:	Special I/O device 1.
UP2:	Screen.

LIST (LST:)

TTY:	Special I/O device 0.
CRT:	Screen.
LPT:	Centronics printer port.
UL1:	Special I/O device 1.

Figure I: Possible logical/physical device assignments

PUN: is used for output, LST: can only be used for output, but CON: deals with both input from the keyboard and output to the screen.

Each of the logical devices, other than the disks, has associated with it four physical devices. Exactly what these are varies from computer to computer, as they depend on what facilities a particular mav=chine can provide.

So on a large computer the list output LST: could be directed to any of four different printers connected to different ports.

Our Amstrads are unfortunately not this powerful - we don't even have an RS232 interface built into the computer as most CP/M machines do. However all things are possible as far as add-on peripherals manufacturers are concerned.

The physical devices which the Amstrad BIOS can assign to each of the logical devices are detailed in Figure I. The two special I/O devices (0 and 1) are normally the two RS232 serial channels provided by Amstrad's own add-on interface.

If you buy anyone else's interface make sure that either it is exactly compatible with Amstrad's or that you are supplied with RSX software and instructions to integrate it into CP/M by installing new device drivers and patching the BIOS jump block appropriately, otherwise you will not be able to use it easily with CP/M.

The null output provides a useful way of 'throwing away' program output we don't want by directing it to the punch device and assigning PUN: to this b'black hole'. Similarly, the null inout simply returns the end-of-file character and so is useful as a dummy input when testing programs.

The current device assignments can be determined and changed by using - you guessed it - STAT.COM! Type:

A> stat dev:

and STAT will tell you the current assignments. Normally on an Amstrad, the defaults are:

```
CON:= CRT:
         (Keyboard and Screen)
PUN:=                    TTY:
         (RS232 channel 0 output)
RDR:          =          TTY:
         (RS232 channel 0 input)
LST:          =          LPT:
         (Centronics port)
```

Typing a command in the form:

A>stat <logical device> = <physical device>

will re-assign the physical device with which I/O is performed when data is sent to/read from the specified logical device. The command:

A>stat val:

will remind you of all legal STAT commands, including the logical and physical device names. For example:

A> stat lst:=crt:

will force output which the program tries to send to the printer to the screen instead - useful for a final check of the output from a program to avoid wasting paper on mistakes.

A>stat con:=tty

is apparently fatal - the machine appears to die. Can you work out why? Well, this assignment tells CP/M to take all the input from the RS232 interface instead of the keyboard, and send all the output to the RS232 port instead of the screen. Thus the console is 'locked out'.

This type of assignment is used when you want to control your Amstrad from another computer via a modem or network - admittedly not an every day operation for most of us. The only way out of this is from the Amstrad end is to hit CTRL+Shift+ESC and re-boot CP/M.

Once we have selected physical devices using STAT we can transfer data between any logical devices using the transient utility PIP.COM - CP/M's Peripheral Interchange Program. This might best be thought of as a software switch - a sort of telephone exchange for data. PIP accepts commands in the form of:

A>pip <destination>=<source>

If you want to do several things one after the other PIP can also be used interactively by just typing:

A>pip

without any command. PIP then prompts you for commands with a #. For example:

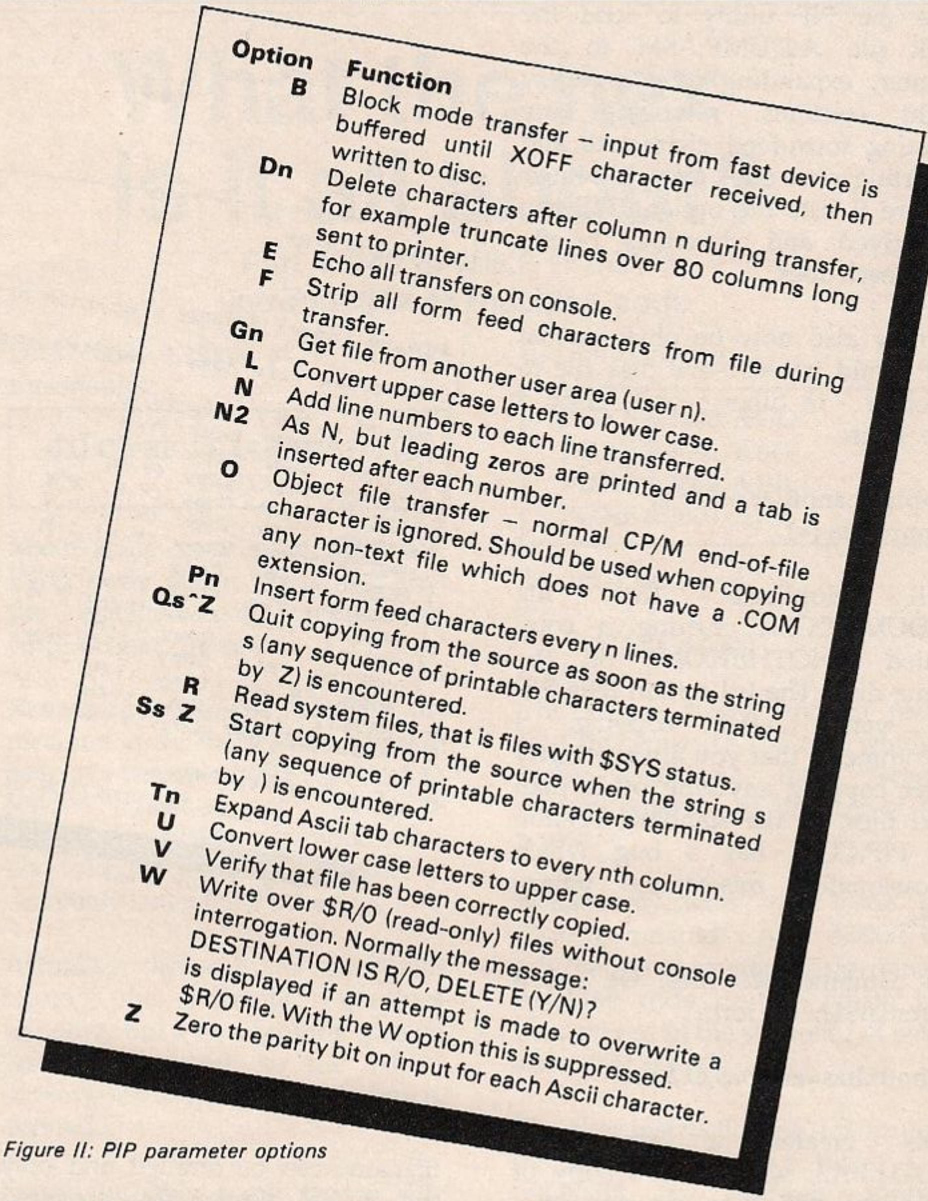
A>pip
#lst:=con:

turns your Amstrad into a primitive typewriter. Whatever you type on the keyboard is sent to the printer as well as the screen. This can be quite useful for bashing out quick memos, shopping lists and so on. Note that the Enter key now only returns the cursor to the start of the line - press CTRL+J to feed a new line.

To finish we must type CTRL+Z, CP/M's 'end-of-file' charcter to tell PIP we have reached thend of our input, now try:

#a:rubbish.txt=con:

This will let you type into the computer exaclty as before, but this time whatever you type will be sent to the disk file



Option	Function
B	Block mode transfer – input from fast device is buffered until XOFF character received, then written to disc.
Dn	Delete characters after column n during transfer, for example truncate lines over 80 columns long sent to printer.
E	Echo all transfers on console.
F	Strip all form feed characters from file during transfer.
Gn	Get file from another user area (user n).
L	Convert upper case letters to lower case.
N	Add line numbers to each line transferred.
N2	As N, but leading zeros are printed and a tab is inserted after each number.
O	Object file transfer – normal CP/M end-of-file character is ignored. Should be used when copying any non-text file which does not have a .COM extension.
Pn	Insert form feed characters every n lines.
Qs^Z	Quit copying from the source as soon as the string s (any sequence of printable characters terminated by Z) is encountered.
R	Read system files, that is files with $SYS status.
Ss Z	Start copying from the source when the string s (any sequence of printable characters terminated by) is encountered.
Tn	Expand Ascii tab characters to every nth column.
U	Convert lower case letters to upper case.
V	Verify that file has been correctly copied.
W	Write over $R/O (read-only) files without console interrogation. Normally the message: DESTINATION IS R/O, DELETE (Y/N)? is displayed if an attempt is made to overwrite a $R/O file. With the W option this is suppressed.
Z	Zero the parity bit on input for each Ascii character.

Figure II: PIP parameter options

CTRL+Z as before to tell PIP that you have finished. If you now type:

#con:=a:rubbish.txt

the process will be reversed - PIP will send the entire contents of A:RUBBISH.TXT to your screen, just like a CCP TYPE command.

PIP.COM is an extremely powerful and useful program, coming into it's own when you want to move files around, split them, search them for certain characters or strings, or concatenate files.

You can, for example, add to the end of an existing file by typing on the keyboard, add on input from the RS232 interface, then stick another file on the end, and so on.

RUBBISH.TXT on drive A: instead of to the printer.

This can be useful if you don't have an editor readily to hand and you want to create a small text file for some reason. Note, if you haven't already, that absolutely everything you type is sent out again - including delete characters.

Many of these functions are controlled by option parameters which we can give PIP along with out commands, in the form:

A>pip <destination> = <source>, <source>, <source>[options]

or

#<destination> = <source>, <source>, ..., <source> [options]

For example, the command:

A>pip lst:=a:dump.asm[t8fp64]

tells the PIP utility to send the disk file A:DUMP.ASM to the printer, expanding tabs to every eight columns, removing any existing form-feed characters and inserting new ones every 64 lines. Figure II lists the options PIP can be given and describes briefly what each does.

It may also now be obvious that PIP could transfer one disk file to another - in other words, copy a file. Thus:

A>pip a: another.one = a: dump.com [v]

will duplicate the file A:DUMP.COM, creating a copy called ANOTHER.ONE on the same disk. The [v] option tells PIP to verify after copying. I recommend that you always verify after copying anything other than text files, as the standard version of PIP.COM has a bug which occasionally mis-copies binary files.

To combine two files, we use a command of the form:

#both.bas=ex1.bas,ex2.bas

This creates a new file, BOTH.BAS, which has a copy of EX2.BAS added to the end of a copy of EX1.BAS - the original files are not changed.

To copy a file from one disk to another, type a command such as:

A>pip b:=a:dump.asm

PIP can therefore be used as a replacement for the Amstrad COPYFILE utility. Unfortunately it is NOT possible to use PIP to copy files from one disk to another on a single drive system.

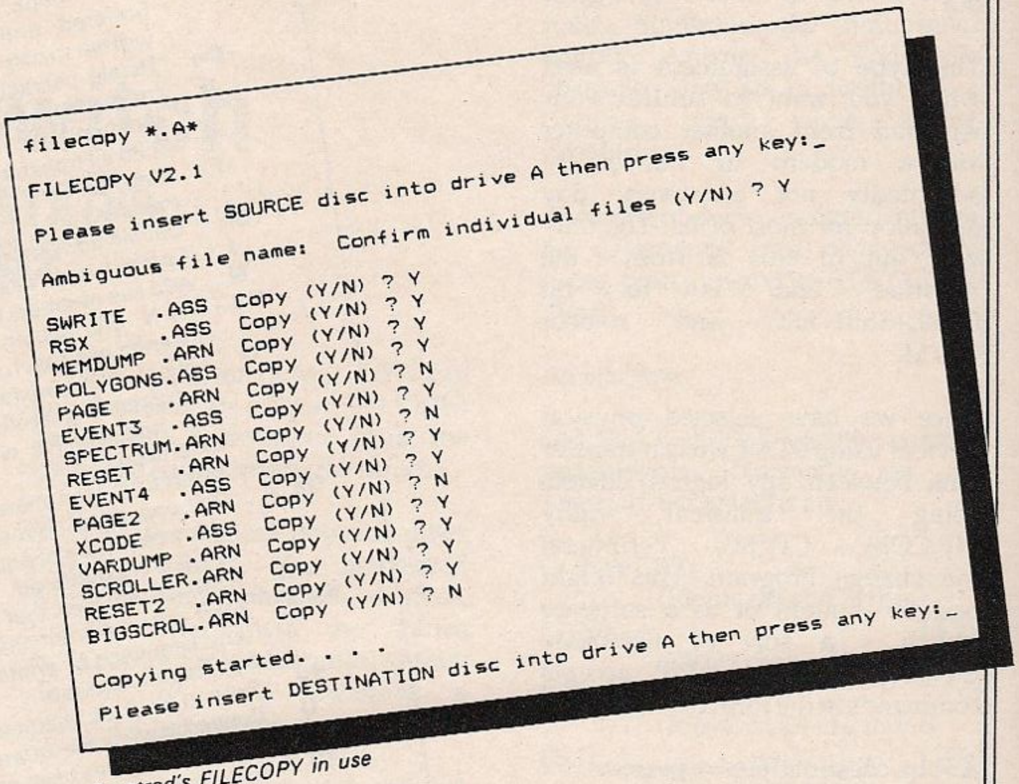
The Amstrad utility FILECOPY will do this (sometimes) but has known bugs which appear when you try to copy large files. This problem can be avoided by using one of the better copying programs available in the public domain.

Notice that if you do not want a different destination filename from the source the destination filename can be omitted and only the logical device (b:) specified explicitly. Also PIP accepts ambiguous filenames for copying - for example:

A>pip b:=a:#.com[v]

will copy and verify all files with a .COM extension from drive a: to drive b:

● Next month we'll see how programs can alter device assignments directly, and go on to look more closely at the facilities provided by the standard BDOS.



```
filecopy *.A*

FILECOPY V2.1

Please insert SOURCE disc into drive A then press any key:_

Ambiguous file name:  Confirm individual files (Y/N) ? Y

SWRITE  .ASS  Copy (Y/N) ? Y
RSX     .ASS  Copy (Y/N) ? Y
MEMDUMP .ARN  Copy (Y/N) ? Y
POLYGONS.ASS  Copy (Y/N) ? Y
PAGE    .ARN  Copy (Y/N) ? N
EVENT3  .ASS  Copy (Y/N) ? Y
SPECTRUM.ARN  Copy (Y/N) ? N
RESET   .ARN  Copy (Y/N) ? Y
EVENT4  .ASS  Copy (Y/N) ? Y
PAGE2   .ARN  Copy (Y/N) ? N
XCODE   .ASS  Copy (Y/N) ? Y
VARDUMP .ARN  Copy (Y/N) ? Y
SCROLLER.ARN  Copy (Y/N) ? Y
RESET2  .ARN  Copy (Y/N) ? Y
BIGSCROL.ARN  Copy (Y/N) ? N

Copying started. . . .

Please insert DESTINATION disc into drive A then press any key:_
```

Amstrad's FILECOPY in use

Don't forget our exclusive Public Domain column elsewhere in this issue! - this month's disk contains approximately 300k of titles available in the Public Domain. Read Shane Kelly's column and find out all about it.

CP/M Public Domain disks Volumes 1,2 and 3 are still available - see order form on page 72

What the hex is it all about?

Part Three of MIKE BIBBY's introduction to machine code

LAST month we looked at hexadecimal numbers - hex for short - and ran two very simple machine code programs. When counting in hexadecimal we went 1,2,3, ...9,A,B ...F.

To avoid confusion with our more usual denary numbers, we prefix hex numbers with the & sign, so strictly we should talk of &F.

If we wanted to count one more than &F we carried on to the 16s column. That is:

&F + 1 = &10

This 16s column meant that:

&23 = 2*16 + 3 = 35

and

&AE = 10*16 + 14 = 174

With the 16s column we can write the value of any byte. The maximum value a byte can store is 255, &FF in hexadecimal. Once our 16s column was full, we carried one to a 256s column, and after that a 4096s column.

We found that a four column (or digit) hex number was all that we needed to specify one of the Z80's memory addresses. For example, the highest memory location possible, 65535, would be &FFFF.

We split these four digit addresses into two sets of two hex digits - that is, into two bytes. The "larger" pair (the 4096s and 256s columns) formed the hi byte and the "smaller" pair (the 16s and 1s columns) formed the low byte. For example, &CDEF has hi byte &CD and low byte &EF.

Even if hex numbering doesn't come naturally to you as yet, I'm sure you've got enough of a feel by now to follow the rest of this series.

The machine code routines we covered last month were simple enough. Here's one of them:

```
CD DB BB C9
```

Even if you followed the last article closely, the above routine probably doesn't mean too much to you.

Let's have a look at it with its mnemonics:

```
CD DB BB   CALL &BBDB
C9         RET
```

Makes much more sense, doesn't it? Particularly when I tell you that the machine code routine at &BBDB clears the screen.

Remember, though we write the machine code the Z80 handles in hex, it's much easier for us to follow the more readable mnemonics. Using mnemonics this way is known as using Assembler or Assembly language.

Actually, the call to & BBDB clears the current graphics window, but I'm assuming you've just switched on, so the whole screen goes, as no special window is defined.

You'll probably recall that CALL (opcode &CD) tells the micro to "GOSUB" to an address specified by the following two bytes.

Notice that though the address of the routine is &BBCD the bytes appear low byte first - that is &CD followed by &BB. This is standard for the Z80.

There's a text screen equivalent of this routine at &BB6C. It clears the text window which, since we haven't set any, in effect clears the whole screen.

If we were going to use this routine instead of the first, our code would now read:

```
CD 6C BB C9
```

In assembler this would be:

```
CD 6C BB   CALL & BB6C
C9         RET
```

To put this routine into your micro at memory location &3000 onwards use the following:

```
POKE &3000, &CD
POKE &3001, &6C
POKE &3002, &BB
POKE &3003, &C9
```

To run it, we enter:

```
CALL &3000
```

The screen clears, and to prove that this time it's the text window we've cleared, the "Ready" prompt appears at the top of the screen.

Remember, when we issue the Basic command CALL &3000 we tell the micro to start performing a machine code routine which has been stored in the sequence of bytes starting at &3000.

The first instruction of this routine (&CD) tells the micro to "GOSUB" to another routine which is located at &BB6C. The micro knows we mean this address because it's stored in the two bytes directly following the &CD - low byte first.

When it returns from this routine, which is already written in the firmware, it looks at the next byte after the two specifying the address to find its next instruction.

In the above program this is &C9, which means RETurn to where you came from - in this case Basic.

All this poking bytes into memory one by one is a rather messy business, though. Surely there's an easier way to enter our hexadecimal numbers?

Program I, called Hexer, is the answer. It is what is known as a hexadecimal loader, storing the hex number you type in into memory.

However, it's much more than that, as you'll see. In fact it's vital that you have this program on tape if you're going to follow this series.

I suggest you type it in very carefully and save it before you try running it. Even if it's entered correctly, you might wipe it our by careless testing - so save it first!

Of course you could save yourself the bother by getting this month's tape - it's on there along with all the other programs from this month's issue. (See Page 72).

When you run it you'll see displayed:

```
You may:
1. Enter code
2. Examine code
3. Alter code
4. Run code
5. End program.
```

To choose any of these options you simply press the number corresponding to it. Option 5 should be self-explanatory. As the reason we wrote it was to enter code, choose option 1. You'll see displayed:

```
Start address?
```

This is asking where you want your machine code program to start from - that is where you want the sequence of bytes to begin.

Now this program, for reasons we'll explain later, is tailored to start machine code programs at &3000 for preference.

You can, of course, override its wishes by entering the address you wish to start at. For the moment though simply press Enter and Hexer will default to an address of &3000.

Next, you'll be prompted:

```
byte?
```

Here the program is asking you what hexadecimal number you want storing in memory, starting at the address you've just selected.

To obtain the previous program that cleared the text screen we enter the following sequence of bytes:

```
CD 6C BB C9
```

So in response to *byte* type in:

```
CD
```

and press Enter. Notice, you don't need the & in front of CD - Hexer does it for you. &CD will now have been stored at &3000.

You'll immediately be prompted for another byte:

```
byte?
```

Enter 6C, which will be stored in &3001. After the next two prompts enter BB followed by C9. These bytes will be stored at &3002 and &3003 respectively.

Our machine code program will now have been stored in memory, starting at &3000. However you'll still be left with the:

```
byte?
```

prompt. Simply enter S for stop (or Sease, as our illiterate editor would have it) and you'll return to the options menu.

We've now entered our code into memory, but don't be tempted to go straight to option 4 to run it.

As you'll discover all too soon, simple typing errors can play havoc with machine code programs.

I haven't protected the program against silly mistakes such as entering XX as a byte. As I mentioned last month, check and re-check before pressing Enter - if you do you'll avoid making such silly mistakes.

However you might accidentally misread E8 for F8, say, so you need some way to examine memory to see if your code is in correctly.

Option 2 allows you to do this. Select it and, when you're asked for the start address, press enter once more to select the default value. You'll see displayed something like:

```
2FF8 31 0 34 8 0 0 0 0
```

This option displays a sequence of bytes in memory a row at a time, eight bytes to the row. The four digit hex number on the left is the memory address of the first byte in that row.

So in the above case we know that &2FF8 has &31 stored in it (notice the numbers are in hex).

&2FF8 has &31 stored in it (notice the numbers are in hex).

&2FF9 has 0 stored in it, &2FFA (remember we count 8,9,A!) has &34 in it, &2FFB has &8 and so on to the last figure of the row, which is the contents of memory location &2FFF.

This time we've defaulted to &2FF8 instead of &3000 - and it's &3000 we're interested in looking at, since that's where our program is stored.

Well so far we've displayed the row of eight bytes from &2FF0 to &2FFF. Actually, the next byte along would be &3000 since:

```
&2FFF +
    1
-----
&3000
```

To prove it, press the space bar. If all is well you should see displayed something like:

```
3000 CD 6C BB C9 0 0 0 0
```

with the first four bytes holding the routine we've entered.

Check carefully that all is well with those first four bytes - they're our machine code program. If these aren't what you intended to enter you'll have to leave this option (by typing S once more) then re-enter the code via option 1. It's only four bytes, so it's no great hardship. Don't do that yet though.

You may be wondering why I chose to start the display at &2FF8 when it's really &3000 onwards we want, to show our programs. Well I'm going to use &2FF8 onwards as a sort of showcase where I can display the result of the programs resident in &3000 on - so this is quite convenient.

If this bothers you simply enter 3000 when you're prompted for the start address. Notice that, as with all Hexer inputs you don't need to specify hexadecimal with "&". Hexer expects hex!

Let's assume that our code is now properly entered and checked. We should have returned from displaying memory to our main menu by entering S. (Alternatively, we could have kept on displaying successive sets of 8 bytes by repeatedly pressing space.)

To run the code we choose option 4. Again, you'll be prompted for a start address. If you simply press Enter you'll default to &3000, our preferred start.

Do so now and you'll see the text screen clear.

Running a machine code program should only be done as a last resort. Unless you're 100 per cent sure that you've got it right don't try it - you're liable to wipe our program, Hexer and all with just a little error.

So if you've selected option 4 to run the code, and decide to back out when you get the start address prompt, simply enter X for eXit and you'll get back to the main menu without running anything.

The same holds for any of hexer's address prompts - X for eXit will take you back to the main menu. Similarly, once you're in option 1 to 3, S for Stop will return you to the menu.

If you've been following, you should have the following program in memory, starting at &3000:

CD 6C BB C9

This isn't too much different from our program to clear the graphics screen which, you may recall, was:

CD DB BB C9

In fact we've only to alter one byte - the low byte of the address called. From the main menu choose option 3. You will receive the prompt "Alter from?" Once more you can enter your own start address. In this case though, simply press Enter and it will default to &3000.

You'll see displayed:

3000 CD?

This tells you that the contents of &3000 are &CD (the opcode for CALL). We don't want to alter this, so just press Enter and it will stay the same. We're now at the next byte in memory and will see displayed:

3001 6C?

This is the byte we wish to alter, so type in its new value, DB then press Enter. You'll then move onto the next location:

3002 BB?

We don't want to alter this, or any subsequent bytes, so enter S and return to the main menu.

Points to note about the alter memory option:

- *To leave a byte unchanged simply press Enter.*
- *To alter a byte, type in its new hex value.*
- *To quit the option type S.*

Having altered our code, check it with option 2 (examine code) to ensure we've got the correct code in memory. Finally, run it with option 4

If you cast your mind back to the first article in the series, we discussed how machine code programs were really about shifting data bytes around in memory.

We likened it to the movement of trains between railway stations, and pointed out that, just as railway networks have junctions, so in a sense does our memory map of locations.

Instead of moving bytes of information directly from memory location to memory location we use these junctions or registers as halfway houses. We move data from memory to these registers, then from these registers to the new memory locations.

One of the most frequently used, and most versatile registers, is the A register, or accumulator as it is also known.

As with other memory locations it can hold one byte of information. The difference is that the A register is actually inside the Z80, so the microprocessor can get at - or process - the information in the A register in its own special ways.

In other words, there are things you can do to a data byte when it's inside A that you can't do while it's out in the sticks in memory.

So a lot of our programs will tend to involve bringing data into A and similar registers, working on it, then shunting it out.

For the moment though, we won't be doing anything so sophisticated, we're just going to try one or two simple tricks with A.

The first involves a firmware routine hidden at &BB5A. This prints on the screen the Ascii character whose code is stored in A.

By calling this routine we can print out any Ascii character we want on the screen - provided we've managed to get its code into the A register.

Actually, putting a number into the A register isn't that hard. Its mnemonic is:

LD A,n

where LD stands for LoaD and n is the hex number you're loading into A. So:

LD A,&2A

means LoaD the A register with the hex value 2A.

Now the opcode for this is &3E. You need to follow it immediately with the hex byte you wish to load into it. So:

LD A,&2A

would translate as

3E 2A

42(&2A) happens to be the code for an asterisk, so to put an asterisk on the screen we need to:

- LoaD the Accumulator with &2A

- CALL the "print out A" routine at &BB5A
- RETurn from our routine.

Assuming you store your code at &3000, this translates as follows:

address	hexcode	mnemonics
3000	3E 2A	LD A, &2A
3002	CD 5A BB	CALL&BB5A
3005	C9	RET

Notice that the address of the routine is in low byte, hi byte order, and that the routine itself terminates with a RET.

To create the routine via Hexer we must enter the following code (via option 1) at the default address of &3000:

3E 2A CD 5A BB C9

As you can see, it's imply the hex part of the assembler listing, byte by byte. Examine the code carefully, and, if all is well, run it.

If you now look at the screen you'll see a solitary "unexplained" asterisk - your machine code put it there!

Of course, if you'd loaded the A register with a different number a different character would have appeared. Suppose for example we'd loaded the accumulator with &41, the assembler listing would look like:

address	hexcode	mnemonics
3000	3E 41	LD A, &41
3002	CD 5A BB	CALL &BB5A
3005	C9	RET

The only byte different is that at memory location &3001 - this now holds &41 instead of &2A.

Instead of entering the whole routine again, we can simply alter this byte with option 3 - Alter code. Choose this option, then &3001 as the start address (no need for the preceeding "&"). You'll see displayed:

3001 2A ?

confirming that the byte has remained unchanged from its previous value of &2AA (Ascii for "*"). As this is the byte we want to change, enter 41, and, after the next prompt, enter S to stop.

If you now examine memory, the code should appear as:

```
10 REM     Hexer
20 REM   Mike Bibby
30 REM  (c) Computing
40 REM with the Amstrad
50 CLEAR
60 ON ERROR GOTO 10
70 size = HIMEM
80 MEMORY &2FF8
90 WHILE -1
100 PRINT: PRINT "You may:-":PRINT
110 PRINT "1. Enter code"
120 PRINT "2. Examine code"
130 PRINT "3. Alter code"
140 PRINT "4. Run code"
150 PRINT "5. End program":PRINT
160 b$ = INKEY$:IF b$="" GOTO 160
170 IF INSTR("12345",b$)=0 GOTO 160
180 b = VAL(b$) : ON b GOSUB 210,400,
550,350,200
190 WEND
200 END
210 INPUT "Start Address"; start$
220 IF start$="" THEN start$="3000":P
RINT:PRINT"Start address = &3000"
230 start = VAL( "&"+start$ )
240 code$="":PRINT
250 WHILE code$ <> "S" AND code$ <>"s
"
260 INPUT "byte"; code$
270 IF code$="" THEN PRINT CHR$(11);:
 GOTO 260
280 IF code$="S" OR code$="s" THEN GO
TO 330
290 code$ = "&"+code$
300 code = VAL(code$)
310 POKE start, code
320 start = start + 1
330 WEND
340 RETURN
350 INPUT "Start Address"; start$
360 IF start$ = "" THEN start$ = "300
0"
370 start = VAL( "&" + start$ )
380 CALL start
390 RETURN
400 INPUT "Start Address"; start$
410 IF start$ = "" THEN start$ = "2FF
8"
420 start = VAL( "&" + start$ )
430 A$ = CHR$(32): PRINT
440 WHILE A$<>"s" AND A$<>"S"
450 PRINT RIGHT$( "    "+HEX$(start),4
);" ";
460 FOR loop = 0 TO 7
470 code$="  " + HEX$( PEEK( start+lo
op) )
480 PRINT RIGHT$(code$,3);
490 NEXT
500 PRINT
510 A$=INKEY$ : IF A$="" GOTO 510
520 start = start + 8
530 WEND
540 RETURN
550 INPUT "Alter from"; start$
560 IF start$ = "" THEN start$ = "300
0"
570 start = VAL("&"+start$)
580 code$ = "": PRINT
590 WHILE code$<>"S" AND code$<>"s"
600 PRINT RIGHT$("    " + HEX$(start),
4);" ";RIGHT$("  "+HEX$(PEEK(start)),
2);" ";
610 INPUT code$
620 IF code$="S" OR code$="s" THEN GO
TO 670
630 IF code$ = "" THEN GOTO 660
640 code$ = "&"+code$
650 POKE start,VAL(code$)
660 start = start + 1
670 WEND
680 RETURN
```

Program I: Hexer

3000 3E 41 CD 5A BB C9

On running the code you'll see an A appear, which isn't surprising as &41 - which you loaded the A register with - is 65 in denary (4*16 + 1), the Ascii code for A. Try altering the program so it prints out B, C and so on.

Finally, load A with 7 and CALL &BB5A. Can you explain what happens? If not, try altering the number you load A with to &C and run the program.

If you're still not sure, the answer lies in the table of Ascii codes in chapter 9 of the User Instructions. 7 is Ascii for a bleep and &C is the code for CLS!

Well, this month we've run a few more programs, learned how to use the A register and familiarized ourselves with a powerful tool for working in machine code - Hexer.

Admittedly, the programs aren't all that spectacular, but if you can follow what's going on in them - as I'm sure you can - you're well on your way to learning what machine code is all about.

● *Next month more on the A register and its close relatives.*

Shane Kelly, our *Write-Hand-Man* takes a *Knife* to the *Torch*!

Or is it put's the Torch to the Knife? - who writes these headlines anyway?

The Knife is a disk editor and I know of no quicker way to reduce important files to a collection of busted bytes than to play around with a disk editor. If you're a novice then prepare a copy of a disk to practice on. ABOVE ALL, READ THE MANUAL!!!

It's such a pity that the manual does not tell you to read the disk file named READ.ME but I suppose it's obvious to everyone except me. I do think they could have included a little addendum to the manual to take care of this as there are some nice extra's in the READ.ME file. Such as both programs now work under CPM+. I would have thought this would be a natural for a full blown hype from the sales department. Anyway, enough gripes of the niggling kind.

This is an excellent package that can do just about anything to the data on your disk. I say package because the contents of the disk are quite comprehensive and include:

a) KNIFE.COM -A program that is user friendly, colorful and works with no trauma at all.

b) KNIFE2.COM -A program that is less friendly, more serious and not so colorful. It works with no trauma also.

Both of the above will work with CPM 2.2 & 3.0, but KNIFE2.COM requires installing with the supplied KNCONFIG.COM. This is one of those 'what key would you like for this function?' programs and anybody who has installed any other program under CPM will be able to use it easily.

c) KNCONFIG.COM -See above

d) KNIFE2J.COM -The JOYCE disk editor (PCW 8000 series)

e) COPY.COM-A copy program that works like PIP, but can check with you if you want to confirm the transfer.

f) GENSUB.COM -A program that will generate submit files.

g) PAGE.COM -A program to page output (ascii files) to a CPM logical device.

h) DIR.COM -A more informative DIR command.

i) UNERA.COM - As the name states - undelete the file.

j) WDEL.COM - Selective deletion of files.

This is a fairly comprehensive package that is cheap, works well, and is available now.

Just one of the fun things that you can do with this program is to customize your disk software by changing sign-on messages. It also provides security for your disks as you can put your name and/or some form of I.D. anywhere on the disk. All my important disks have this feature.

I am very impressed with this package and I can recommend it to anybody as the only disk/section editor they will ever need.

Once in a while I come across a program or concept that is truly useful. By that I mean that I actually use this product because it is accessible, easy,fast and useful.

Write-hand-man is a 'pop-up' utility that is readily available when running your chosen application - simply press the trigger character and up pops the menu. Make your choice from this menu and then return to your application as if you had never been away. That, I imagine was how this utility was supposed to work and to a large extent it does, but depending on what application you are running you may encounter difficulties as I did with the screen not being exactly as you left it. To be fair, there are two versions of WHM one of which claims to refresh the screen after you exit, but I could not get it working with any of the terminal configurations supplied.

What you get in your package is two versions, one for CPM2.2 and one for CPM3.0 and within these two versions you also get the screen refresh version (WHMT.COM). Also included are all the applications, a configuration program, a manual and instructions on how to write your own applications. The applications supplied are:

The notepad - a 32*8 screen within the screen. Notes may be written and read and cut and pasted into the application although this occurs very slowly as the notes are cut into the key macros memory (see later) and then transferred to the current application. All controls are WORDSTAR compatible.

The phonebook - 32*12 screen that has all the A's and B's on one page. A bit small really, but if you can limit yourself to those important numbers only you will it useful.

The calendar - a 14 day appointment diary. It has a roll forward option where the next week can become this week and so free the next for re-use.

The dir - just that - a directory of your disk(s) and user areas.

The view program - view ascii files (in 32*B line format). lines long than 32 characters may be truncated or not.

The calculator - a handy four function calc with 14 digit range.

Keys - a function similar to the programmable keys of the AMSTRAD. Key memory is used in the cut and paste option of notepad and view and for storing results in calculator which can then be pasted to an application later.

Hex - a small four function hexa-decimal calculator - handy for programmers.

Also included are SWAP and ASCII, the former a multi-task program that swaps out your current application and while it has limitations, is useful and the latter is ascii character table.

WHM is an attempt to simulate the pop-up utilities of larger 16 bit machines and within the confines of the restricted CPM address space it works very well. It is also good to see a package that has specific instructions for the AMSTRAD range of computers.

This product is useful, has capability for expansion, is well presented and it works well on the AMSTRADS. BUY IT ! !

The TORCH is a tutorial program that purports to teach CP/M to novices. If you wish to learn how to use CP/M 2.2 then this is an excellent program, but for CP/M+, forget it. There is no difference in the information presented for the different operating systems which would lead one to believe that there is no appreciable difference between the two. This is simply not the case. This is the major fault with Torch. Two minor faults were the choice of colors under CP/M 2.2 and the ridiculously tiny print of the documentation which appears to consist of 4 A4 size pages reduced to fit on one A4 size page.

Both versions present a menu page where you are given the choices of PART 1, which is basic CP/M operations like loading CP/M, the filing system, naming files, drive logging and other basic functions and PART 2 which is broken into sections covering built-in commands, transient commands and utility programs. From the main menu you may also browse through the part that happens to be loaded at that time. PART 1 is under TORCH1A (for CP/M+ it is TORCH1V) and PART 2 is under TORCH2A (TORCH2V for CP/M+). You must change disk sides to access part 2. The browse option lets you read the 'title page' of each sub-heading and if invoked from the PART 1 or 2 selection menu allows you to select more detail about a particular subject that has caught your eye. As you reach the end of each section you are given the choice of recapping the section, printing notes on the subject or going back to the main menu to access the next item. The printed notes are just the highlights and would help as a fast reference as without them you would need to shelve the current application, get the TORCH disk and boot the program then search until you find the section that you need. Hardly convenient, is it?

The thought that kept crossing my mind was that a good reference book was still going to be necessary.

Also supplied with the TORCH was a utility called the WAND which allows you set up menus of your most commonly used programs. These programs may spread themselves across many disks and the WAND can handle that with no problems. It has trouble with those programs that require a different command line to entered on each invocation but other than that it works extremely well at hiding the unfriendly user interface of CP/M.

I was more impressed with the WAND than the TORCH and I think that devoting three quarters of the (sparse) documentation to the WAND reflects the relative merits of these two programs.

Shane Kelly

Public Domain

by

Shane Kelly

It has been pointed out to me that I keep haranguing you lot out there to tell me what public domain software you want and as yet I have not given you the catalogs to choose from. That is remedied this month.

On side a of the disk is CPMUG.CAT and SIGØ.CAT. These are the CP/M user group catalog and the Special Interest Group (Micro-computers) catalog part Ø and on side b is part 1 of the SIG catalog. This is not the complete catalog and I will present updates as I get them. Don't despair, as there is about 300k of lists for you to get through and choose from.

There are many thousands of programs in the public domain. Some are absolutely useless to as AMSTRAD owners and others are inform-ative and useful.

It is these latter ones that I find are most useful to me in my own programming endeavors. You can be browsing a listing and see one way of doing something that is different from the way you usually do it and it turns out to be better in the long run, either saving memory or operating faster. I have picked up many useful routines from doing this and all have meant a saving in my time.

All this is leading up to saying that you should certainly use any Public Domain programs that you want, but don't be content just to use them, probe and explore how they work - if they don't do what you want then change them to suit yourself. It is a learning experience that is particularly rewarding. And, who knows, you may even be

Continued from Page 70

a contributor to the library of public domain programs your-self one day.

To let me know your preference drop a line to me via this magazine and I will attempt to obtain the volume requested and check it out for operation on your stated machine. As with all PD programs there are no guarantees and the time delay could be significant, so if you're desperate don't bother, try other avenues first.

Warning - do not ring the people at the magazine with queries about public domain software as a severely chewed ear-hole about support for free software is all you will get. If you have problems with this PD software first read the article accompanying it again, then any .DOC files on the disk and then go to your local user group. If none of the above work, then write me a letter detailing your problem and I will see what I can do for you. Once again, no promises and no guarantees.

One of the questions that I am asked frequently is where do I get all this public domain software? Well, a lot of it came from the EASTERN AMSTRAD USER GROUP in Vic with special thanks to the prez, Mr. Tony Blakemore and the CP/M guru Mr. Bob Smits. Various cont-ributions from other members are ack-nowledged with thanks. Several bulletin boards have been raided (but only those at 1200 Baud because it's horrendously expensive at 300Baud) and to the SYSOPS of these systems my thanks also.

The other question that I get asked is why does it cost $19.95 a disk? The breakdown goes like this - 10 dollars or so for the disk, which leaves 9.95. Then there's postage (I don't know how much OZ Post charge) then the magazine publishers have to pay someone to copy the discs and someone has to pay for the PD discs originally. My expenses also have to be paid. The other factor is that these programs are checked to see if they run on the AMSTRAD range and altered to suit if at all possible. As far as I am concerned I think the price is fair when compared with other suppliers such as SELECT who charge $15 per disk PLUS $3 post and packing per order PLUS 20% sales tax which makes each disk $21 and that's on a 5 1/4" disk!

Moans and groans aside, what have I got planned for the future I hear you ask. (OK, so you didn't ask!) Next month the small C compiler looks favorite at this stage, but if I can get hold of a text editor and a few format utilities (and I can get them working!) I'll be doing that instead.

Right that's it for this month, but if you wish to contact me urgently I'm on VIATEL MAIL BOX NUMBER 534876510.

Public Domain Disk
Volume 1

NuSweep file manip-ulation program, Cheque-book program (CP/M 3.0 only), Sort for Ascii files, File comparator, plus various other useful file and disk utilities.

CAT #: 2801

Public Domain Disk
Volume 3

PD3 features a full featured assembler and disassembler, again with full documentation - one of the best around and fully configured for your Amstrad.

CAT #: 2803

Public Domain Disk
Volume 2

All you need to get you communicating with the outside world. Modem 9 with full documentation.

CAT #: 2802

Public Domain Disk
Volume 4

Huge (300k) catalog of CP/M User Group soft-ware available. You'll spend weeks just drooling over this lot - write and tell us what you'd like to see on coming disks.

CAT #: 2804

See overleaf for ordering information

SOFTWARE ON TAPE

CAT #	TITLE	PRICE
1001	TASWORD	$36.95
1006	TOOLBOX	$19.95
1007	FLEXIFREND	$19.95
1008	GRASP	$19.95
1009	CHAOS FACTOR	$15.95
1010	MUZICO	$17.95
1011	DRUMKIT	$16.95
1012	MUSIC COMPOSER	$17.95
1013	EASIDATA II	$29.45
1014	EASIDATA III	$41.45
1015	DATABASE/MAIL LIST	$29.45
1022	QWERTY	$14.95
1024	MYRDDIN FLIGHT	$17.95
1030	AMS-FORTH	$25.00
1201	PLAN-IT (CPC)	$36.95
1202	MINI-OFFICE II	$36.95
1203	MAGIC SWORD	$21.95
1204	FUN SCHOOL (2-5)	$15.95
1205	FUN SCHOOL (5-8)	$15.95
1206	FUN SCHOOL (8-12)	$15.95
1207	CHARTBUSTERS	$15.95
1208	CLASSIC GAMES	$15.95

SOFTWARE ON DISK

CAT #	TITLE	PRICE
2001	TASWORD	$48.95
2009	CHAOS FACTOR	$27.95
2012	MUSIC COMPOSER	$29.95
2014	EASIDATA III	$53.45
2015	DATABASE/MAIL LIST	$41.45
2016	EASI-WORD COMBO	$49.95
2017	GENESIS	$39.95
2018	EASY MUSIC	$34.95
2019	POT POURRI VOL. 1	$19.95
2020	POT POURRI VOL. 2	$19.95
2022	QWERTY	$26.95
2024	MYRDDIN FLIGHT	$29.95
2201	PLAN-IT	$48.95
2202	MINI-OFFICE II	$48.95
2203	MAGIC SWORD	$33.95
2204	FUN SCHOOL (2-5)	$27.95
2205	FUN SCHOOL (5-8)	$27.95
2206	FUN SCHOOL (8-12)	$27.95
2207	CHARTBUSTERS	$27.95
2208	CLASSIC GAMES	$27.95
2301	PLAN-IT (PCW)	$47.95

PUBLIC DOMAIN DISKS

CAT #	TITLE	PRICE
2801	PD VOL. 1	$19.95
2802	PD VOL. 2	$19.95
2803	PD VOL. 3	$19.95
2804	PD VOL. 4	$19.95

ORDER FORM

CAT #	TITLE	PRICE
	TOTAL	

Bankcard ☐ Mastercard ☐ Visa ☐

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Signed ________________ Expiry ☐☐☐☐

Name ________________________

Address ______________________

________________ 'Phone ________

State ________ Postcode ____________

MAIL ORDERS TO:

STRATEGY SOFTWARE
P.O. BOX 11
BLACKMANS BAY
TASMANIA 7152

ENQUIRIES [002] 294377
ORDERS [008] 030930

ORDER TOLL-FREE WITH YOUR
VISA, MASTERCARD OR BANKCARD
SEE SIDE PANEL NEXT PAGE

SUBSCRIPTIONS

CAT #	TITLE	PRICE
5001	MAGAZINE ONLY	$40.00
5002	MAGAZINE + TAPE	$80.00
5003	MAGAZINE + QUARTERLY DISK	$105.00

BACK ISSUES

CAT #	TITLE	PRICE
6008	CWTA PREMIERE EDITION	$4.50
6009	CWTA SEPTEMBER ISSUE	$4.50
6010	CWTA OCTOBER ISSUE	$4.50
6011	CWTA NOVEMBER ISSUE	$4.50
6012	CWTA DECEMBER ISSUE	$4.50
6101	CWTA JANUARY ISSUE	$4.50
6102	CWTA FEBRUARY ISSUE	$4.50

TAPES

CAT #	TITLE		PRICE
7008	DIAMOND DIG	(8/86)	$7.50
7009	ICE FRONT	(9/86)	$7.50
7010	da BELLS	(10/86)	$7.50
7011	DISCMAN	(11/86)	$7.50
7012	ROBOT RON	(12/86)	$7.50
7101	OTHELLO	(1/87)	$7.50
7102	SPACE BASE	(2/87)	$7.50

DISKS

CAT #	TITLE	PRICE
7051	7008/7009/7010 AS ABOVE	$19.95
7151	7011/7012/7101 AS ABOVE	$19.95

BOOKS

CAT #	TITLE	PRICE
3001	AMSTRAD HANDBOOK	$ 9.95
3002	AMSTRAD COMPUTING	$17.95

ORDERING INFORMATION

1. MAIL ORDER

Should you wish to order by mail but not wish to deface your magazine please photocopy P.72 or handwrite your order being careful to include **all** information requested on the order form. Please make sure you have enclosed your name and address (you'd be surprised!) and the correct amount for the goods you require.

If sending a cheque or ordering using Visa, Mastercard or Bankcard please ensure that the date on your cheque is valid (i.e. 1987 not 1986) and that your credit card has not expired.

2. TELEPHONE ORDER [008] 030930

This month we have installed a new toll-free order line. Please follow the instructions below carefully **before** ringing. Note that this number will only be answered by a machine and cannot be used for general enquiries or messages. Anything other than an order will be ignored - you have been warned!

A) Complete the order form on Page 72 as though you were going to order by mail. Do not wait until ringing before deciding which titles you require or trying to find your credit card. The answering machine in use is voice activated and any pause over a couple of seconds will result in the machine hanging up on you.

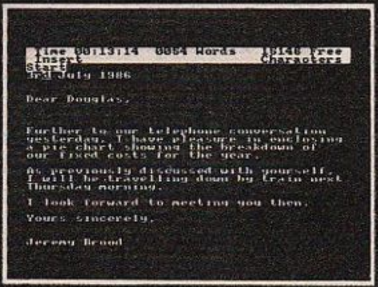
B) When the machine answers, read the order from your order form slowly, clearly and distinctly giving all the information you have written down. Where possible leave a telephone number just in case we can't understand or hear your order.

C) This service will be in operation 24 hours a day, every day of the year.

D) Allow 28 days for the delivery of your order - orders which we cannot despatch within 2-3 days of receipt will be advised of the likely delay by mail.

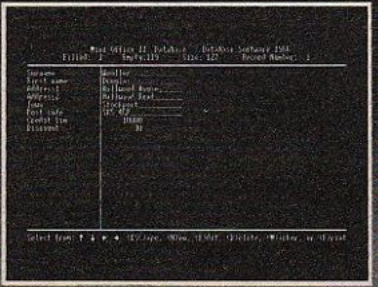
All prices include postage & packing

All this in just



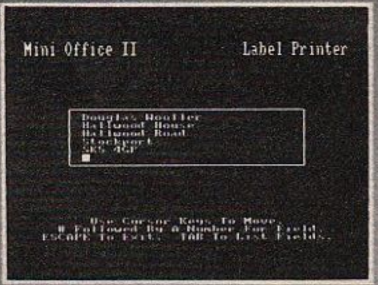
```
 Time 00:13:14  0054 Words    18140 Free
 Insert                        Characters
Start
3rd July 1986

Dear Douglas,

Further to our telephone conversation
yesterday, I have pleasure in enclosing
a pie chart showing the breakdown of
our fixed costs for the year.

As previously discussed with yourself,
I will be travelling down by train next
Thursday morning.

I look forward to meeting you then.

Yours sincerely,

Jeremy Brood
```

WORD PROCESSOR
Compose a letter, set the print-out options using embedded commands or menus, use the mail merge facility to produce personalised circulars – and more!



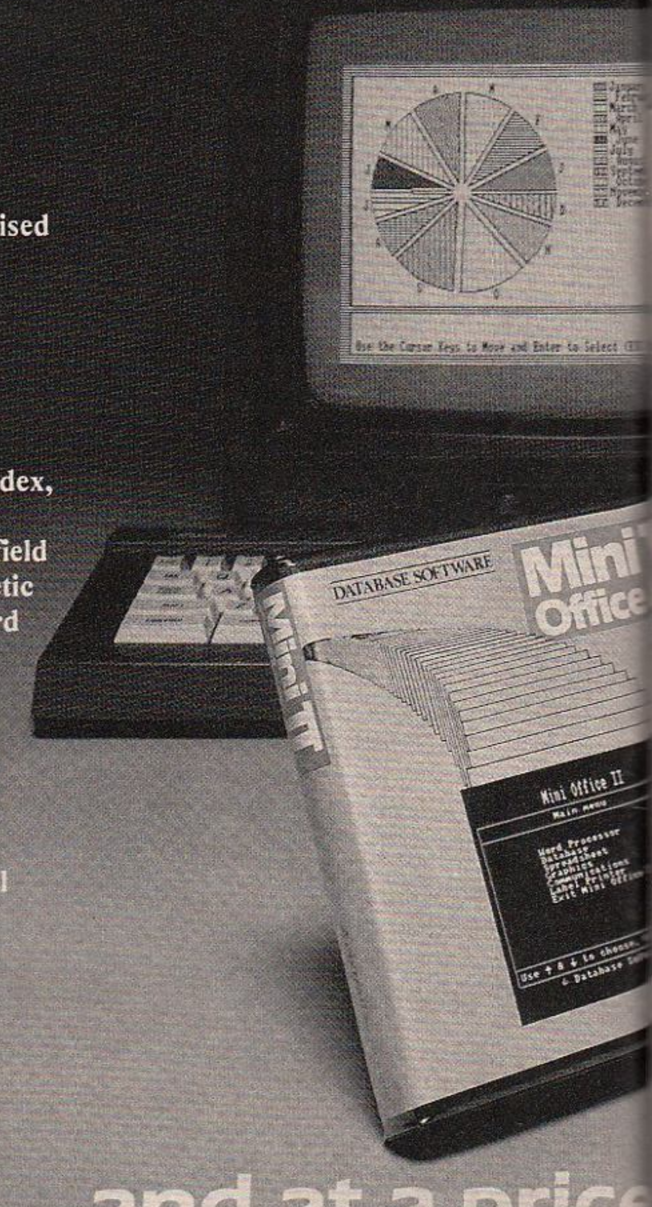
```
        Mini Office II Database   Database Software 1986
    Filled: 1    Empty:119     Size: 127     Record Number: 1

Surname          Woolley
First name       Douglas
Address1         Hallwood House
Address2         Hallwood Road
Town             Stockport
Post code        SK5 4GP
Credit lim          10000
Discount               30

Select from: [illegible]
```

DATABASE
Build up a versatile card index, use the flexible print out routine, do powerful multi-field sorting, perform all arithmetic functions, link with the word processor – and more!



```
Mini Office II                Label Printer

          Douglas Woolley
          Hallwood House
          Hallwood Road
          Stockport
          SK5 4GP

            Use Cursor Keys To Move.
        # Followed By A Number For Field.
     ESCAPE To Exit.   TAB To List Fields.
```

LABEL PRINTER
Design the layout of a label with the easy-to-use editor, select label size and sheet format, read in database files, print out in any quantity – and more!



...and at a price

Mini Office II offers the most comprehensive, integrated suite of programs ever written for the Amstrad – making it the most useful productivity tool yet devised.

A team of leading software authors were brought together to devote a total of 26 man years of programming to the development of Mini Office II. What they have produced is a package that sets new standards in home and business software.

The sample screenshots above illustrate just a few of the very wide range of features, many of which are usually restricted to software costing hundreds of pounds. Most are accessed by using cursor keys to move up and down a list of options and pressing Enter to select.

Is it that easy to use? Several leading reviewers have already sung its praises on this very point.

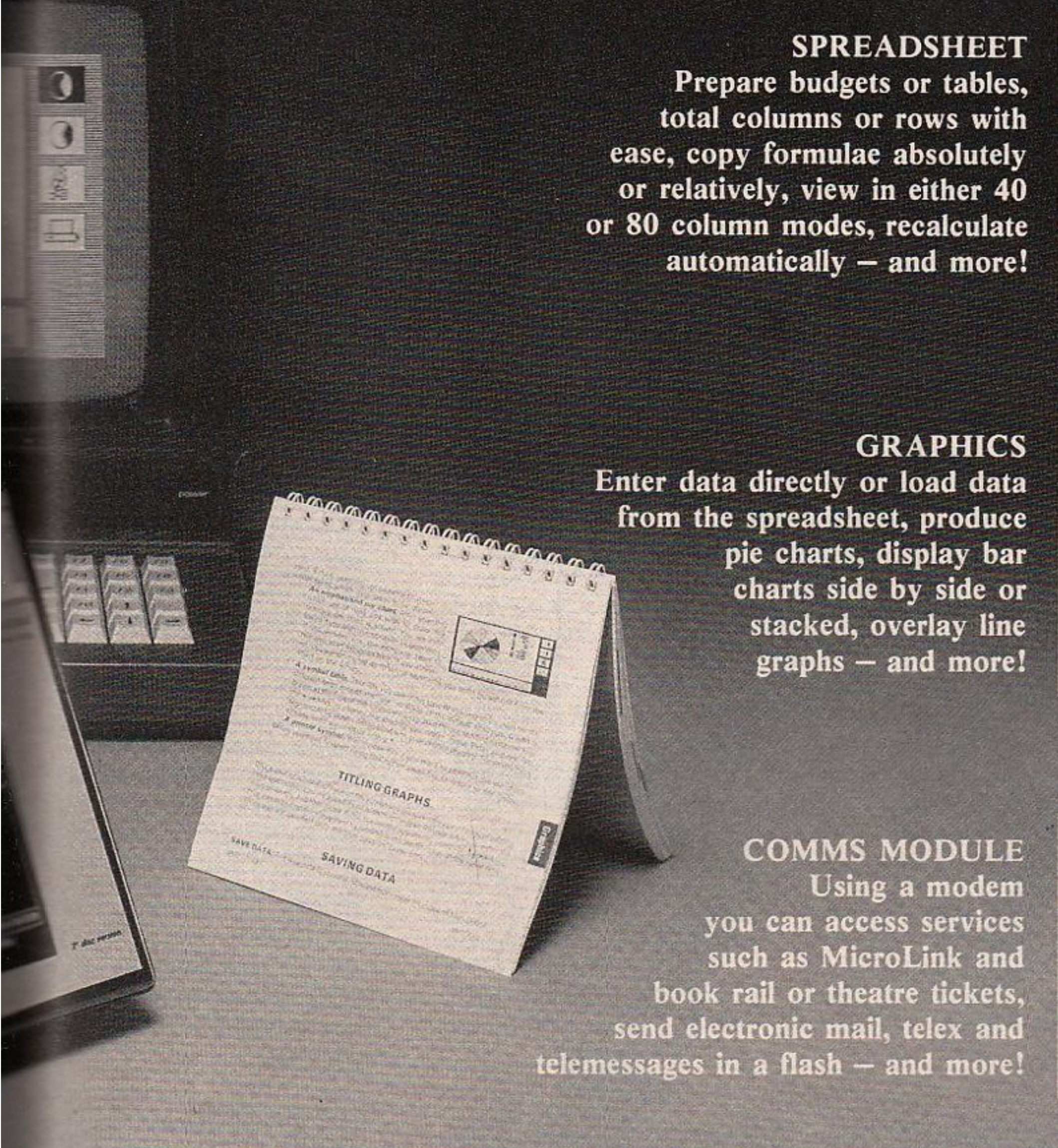
Yet possibly the best advertisement for Mini Office II is that it comes from the same stable that produced the original Mini Office package back in 1984.

That was so successful it was shortlisted in two major categories of the British Microcomputing Awards – the Oscars of the industry – and sold in excess of 100,000 units!

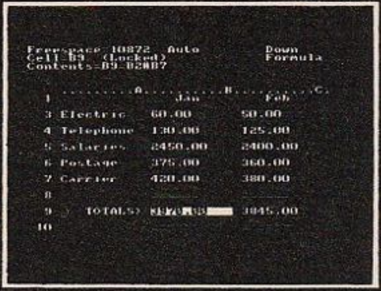
It was up to Mini Office II to take over where the first Mini Office left off, with 32 extra features, two additional modules, a program to convert existing Mini Office files to Mini Office II format, and a 60 page, very easy to follow manual.

This is the package thousands of Amstrad owners have been waiting for – and at a price everyone can afford!

ONE package!

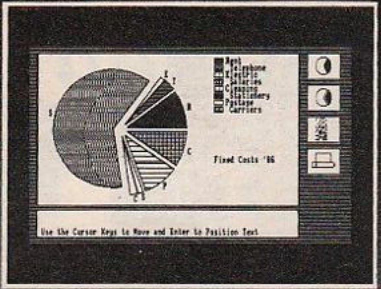


SPREADSHEET
Prepare budgets or tables, total columns or rows with ease, copy formulae absolutely or relatively, view in either 40 or 80 column modes, recalculate automatically – and more!



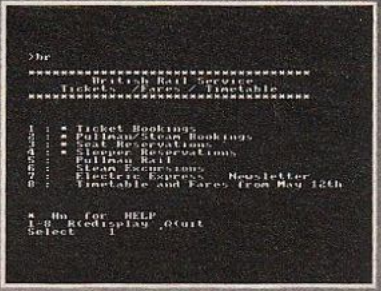
```
Freespace 10872  Auto          Down
Cell B9  (Locked)              Formula
Contents=B9-B2@B7

   ...........A..............B.............C.
  1                Jan              Feb
  3 Electric     60.00            50.00
  4 Telephone   130.00           125.00
  5 Salaries   2450.00          2400.00
  6 Postage     375.00           360.00
  7 Carrier     420.00           380.00
  8
  9     TOTALS> [illegible]      3845.00
 10
```

GRAPHICS
Enter data directly or load data from the spreadsheet, produce pie charts, display bar charts side by side or stacked, overlay line graphs – and more!



Use the Cursor Keys to Move and Enter to Position Text

COMMS MODULE
Using a modem you can access services such as MicroLink and book rail or theatre tickets, send electronic mail, telex and telemessages in a flash – and more!



```
>br

************************************
      British Rail Service
   Tickets  /Fares / Timetable
************************************

1 : * Ticket Bookings
2 : * Pullman/Steam Bookings
3 : * Seat Reservations
4 : * Sleeper Reservations
5 :   Pullman Rail
6 :   Steam Excursions
7 :   Electric Express   Newsletter
8 :   Timetable and Fares from May 12th

*  Hn  for  HELP
1-8  R(edisplay ,Q(uit
Select    1
```

that can't be matched!

Here's what some independent reviewers say about Mini Office II:

Menus throughout the program were abundant and well structured, allowing complex choices to be made with hardly a glance at the pocket-sized spiral-bound manual, and offering a wealth of user-defined functions ... For the money it really is unbeatable. Dedicated word processors, spreadsheets and comms packages may offer some extra facilities, but some are so full of whistles and bells you may find yourself forever in the manual or even not using half of their power. What you get with Mini Office II is a valient attempt to provide an all-embracing small business package at a down to earth price; it fulfills all its promises, and there's nothing cut-rate about the facilities it provides. – **Tubelink/Viewfax 258**

The original Mini Office has been recognised as an astonishing bargain for anyone looking for a low-cost introduction to serious software. Now the four programs which made it up have been substantially enhanced and two new programs have been added. The result, Mini Office II, is more than just an introduction to serious software applications. It's good enough to fulfill the entire software needs of many a small business! ... The range of features in each program is astonishing for a package of this cost ... The spreadsheet is every bit as good as a program such as MasterCalc which costs more than the entire Mini Office II ... This delightful little program [Graphics] allows you to produce professional looking charts to illustrate a set of figures. – **Amstrad Action**

The word processor is literally overflowing with excellent features ... The database is very easy to use yet extremely powerful ... The spreadsheet program is the next little gem ... Having used Mini Office II for the past few days I cannot praise it too highly. – **Popular Computing Weekly**

Amstrad CPC 464, 664, 6128

Cassette	$36.95
3″ disc	$48.95

DATABASE SOFTWARE

order form on Page 72

Your'e never too young to play a Magical Adventure on the Amstrad CPC range....

The Magic Sword
Kristin and Martin Hollis
The computer adventure of Princess Poppy and the Frog Prince
A Database publication
Read the book – then play the game!

Based on the style of the classic computer adventures - but written so that even small children can learn to find their way around, encouraged by colorful graphics and exciting sound effects.

Each pack contains a 48 Page full-color story book PLUS a full-length multi-location adventure for only $24.95 on tape and $34.95 on disk.

See order form on Page 72 for further details.

Read the book - then play the game!

Skill rewarded

Thrust

$9.95 - Tape Only

THE rebels intend to launch an attack against the Empire and to this end have captured several starships, but without drive units. Donning your Luke Skywalker gear you must steal new units from the Empire's storage planets.

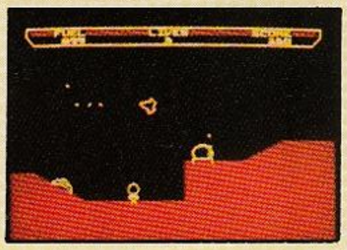
Your space craft, buildings and gun emplacements are displayed as high resolution line drawings.

The game is played as a series of missions of increasing complexity. In the first the drive unit is situated on the planet's surface, along with a fuel dump, gun emplacement and nuclear reactor. In subsequent missions the drive will be located in underground caverns.

Fuel can be taken on board by hovering over the dump and activating the tractor beam. To achieve the delicate hovering manoeuvre you need to point the ship away from the planet and use short bursts on the thruster to counteract the effects of gravity.

Having located the drive unit the tractor beam can be used to tow it behind your ship and this is where the fun really begins. The drive unit hangs beneath the ship like a pendulum, attached to the ship by a solid bar.

Each movement of your craft will cause a realistic movement of the pendulum. As a pilot you must fly as smoothly as possible, counteracting every swing of the drive unit. Should the swing become uncontrollable you will be dragged into a cavern wall.



Thrust has got what it takes to become a classic computer game – inherent simplicity combined with handsome rewards for the skillful player.

Jon Revis

Sound	*8*
Graphics	*9*
Playability	*10*
Value for money	*10*
Overall	*10*

Sticky search

Mission Elevator

$29.95 - Tape
$44.95 - Disk

FBI headquarters has been taken over by enemy agents. If the US government does not agree to their demands then a bomb will be detonated, destroying the building and killing everyone inside. Agent Trevor (that's you) must penetrate the building and make his way to the 62nd floor to defuse the bomb.

The building is divided into units of eight floors accessed by a series of lifts. The floors are separated by locked emergency exits, and the porter who holds the key has gone into hiding.

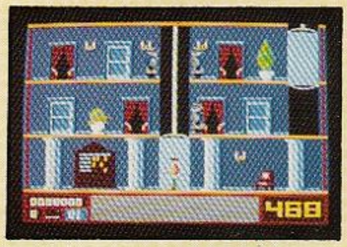
You must therefore search the building room by room to find him while avoiding the enemy agents.

The bomb can be made safe by entering a sixteen part code. Two sections of this code are to be found in each block of the building, hidden under items of furniture or behind paintings.

Every floor of the building is crawling with baddies, which you can dispose of using a well timed flying kick or, more usually, a slug from your 45.

The graphics employed in this game are brightly coloured but a little chunky. The bad guys are drawn as shady characters in grey overcoats and hats while you are a rather pathetic stick man. This is disappointing as great care has obviously been taken to design the other characters and objects.

The game is a complex arcade adventure with many items to examine and an infinite number of spies to



shoot. It may not get to number one in the charts but it will certainly keep you occupied for many a frustrating hour.

Steve Brook

Sound	*7*
Graphics	*8*
Playability	*8*
Value for money	*8*
Overall	*8*

Glamour is gone

Return to Oz

$29.95 - Tape
$44.95 - Disk

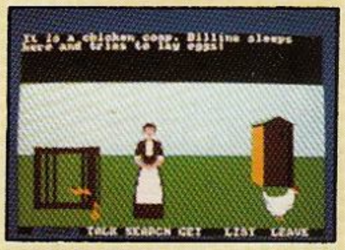
LONG after the excitement of your first adventure in Oz has died away you find yourself dreaming of Oz once more. Your friends, the tin man, scarecrow, and cowardly lion are in trouble and somehow you must find your way back to Oz to help them.

Return to Oz is an adventure game of sorts for children. In a normal adventure half the fun of the game is spending many hours trying out different commands and possibilities until you stumble across one which works. But you are deprived of this pleasure in Return to Oz as the whole game is menu driven.

The screen is divided into three windows. The top one describes the location and provides replies to your queries and the large middle window displays a picture of your surroundings complete with all people and objects. The final window is a single line at the bottom of the screen showing the menu options available.

The Look command draws a box around the first object on the screen. You press the spacebar to move the box to the object or person you wish to look at then press Return and read the response in the top window.

If there are no objects which can be looked at then the look command will have no effect. Other commands include Talk, Search, Get, List, and Leave. These too use the box principle, unfortunately the game soon degenerates into a pattern of using every command on every object on



It is a chicken coop. Billina sleeps here and tries to lay eggs!

the screen.

This is the first adventure game for children that I have seen on the Amstrad, and I'm afraid to say that it is disappointing and youngsters will quickly lose interest.

Carol Barrow

Sound	*0*
Graphics	*6*
Playability	*6*
Value for money	*6*
Overall	*6*

Dangerous mission

Impossible Mission

$29.95 - Tape
$44.95 - Disk

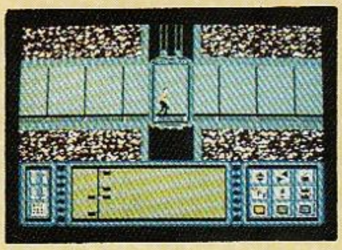
PROFESSOR Elvin Atombender, well known computer genius and psychopath, has hacked into the military computers of the major nations. Within six hours he will have cracked the launch codes and started World War Three. This is how long you are allocated to penetrate his underground stronghold and find the control centre.

The professor's stronghold consists of 32 rooms which are interconnected by numerous lift shafts and you begin the game in one of the lifts. Exits to your left and right can be explored or you can pull back on the joystick and the lift will descend to the next level.

Your footsteps have a wonderful metallic ring to them as you leave the lift and run along the corridors. Running off the screen causes the adjoining room to be displayed. This is made up of a series of catwalks, some of which can be accessed using mini lifts. All of the rooms are patrolled by the professor's robots which are sensitive to movement and armed with high voltage electrodes.

To enter the professor's control room you must have a nine digit password. By completing a series of sub-puzzles you will be given each digit in sequence. Puzzle sections are hidden in the furniture to be found in each of the rooms, and so are passwords which can be entered into one of the professor's security terminals. These will immobilise the robots or reset the lifts to their original positions.

As secret agents go you are poorly equipped – one pocket computer to help you solve the puzzles, and your own athletic prowess.



Impossible Mission is a very good conversion from the Commodore original.

James Riddell

Sound	7
Graphics	8
Playability	8
Value for money	8
Overall	8

Dan back in style

Dan Dare: Pilot of the Future

$29.95 - Tape
$44.95 - Disk

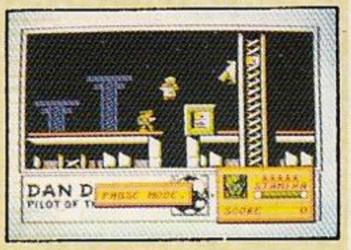
WHEN I was a lad Dan Dare fought weekly battles against the evil Mekon on the front cover of the Eagle comic. Now, courtesy of Virgin Games, Dan Dare "Pilot of the future" enters the computer age with a bang.

The Mekon, Dan's arch enemy, has launched a hollowed-out asteroid in the direction of Earth and unless we comply with his demands the planet is doomed. We have only one chance of survival – Dan Dare and his faithful companion Digby.

Dan and Digby have flown to the asteroid in an attempt to explode it before it reaches Earth. As they landed they were ambushed and Digby was captured and imprisoned at the far side of a large chasm. Dan must search the asteroid for the four bridging units required to span the gap.

As he locates each of them he is allowed deeper into the asteroid to face ever increasing dangers.

The graphics in the game make those of Sorcery look amateurish. The interior of the asteroid is supported by a complex combination of columns and girders. Every feature is drawn complete with shadow and highlights. The cartoon characters which represent Dan and the various types of Treen (the Mekon's followers) are truly incredible.



Dan Dare is a truly excellent game that I can thoroughly recommend.

James Riddell

Sound	8
Graphics	10
Playability	10
Value for money	10
Overall	10

Real puzzler

Split Personalities

$29.95 - Tape
$44.95 - Disk

MANY of the world's most famous celebrities have gone to pieces and you must put them back together again.

A remarkable likeness of a famous person has been split into 20 separate pieces which have then been thoroughly mixed up. The area on which you must reconstruct the image is a 5 x 5 grid.

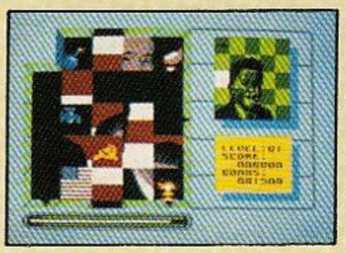
To the right of the playing area is a small picture of the completed puzzle. Whenever the cursor moves over a piece on the playing area a section of this small picture will be illuminated indicating its correct position on the grid.

When the game begins the playing area is empty and pieces can be brought on to the grid by moving the cursor to the top left-hand corner and pressing the fire button. Once on the grid the pieces are moved around by placing the cursor on the selected piece, pressing fire, and moving the stick in the desired direction.

Important features of the playing area are the three sliding sections of wall. These open and close at random and can be used to dispose of unwanted pieces.

Apart from the sections which construct the character's face there are also bonus pieces. These are related images such as the Soviet and US flags which, when thrown together, will earn you bonus points.

Occasionally a bomb will appear and if you fail to push it through a gap in the wall within five seconds you will lose a life. You are allowed three lives to complete the puzzle, and you will need all of them.



Split Personalities is a combination of excellent graphics and infuriating puzzles. If you are a puzzle nut then you won't find a better game.

Steve Brook

Sound	8
Graphics	8
Playability	8
Value for money	8
Overall	8

Galactic battles

StarStrike II

$29.95 - Tape
$44.95 - Disk

USING the Mark 1 Starstrike fighters the Federation drove the outsiders into deep space. With the development of the infinitely more powerful Mark 2 Starstrike ships they intend to neutralise the Outsiders' home planets.

The Outsider's system is based around five stars and their associated 22 planets. The planets fall into three categories – military, industrial and agricultural. Each has its own key point which must be knocked out in order to neutralise the planet.

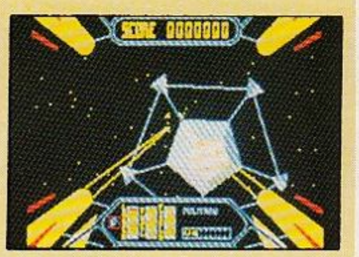
The game begins with your ship on board a Federation support module. From here you select which of the five star systems you wish to visit and after a short hyperspace jump you arrive at your destination.

All planets have several defence shields surrounding them but every shield contains a small opening to allow the entry of the Outsider's own ships.

An alternative route is to destroy the wheel shaped space station which orbits many of the planets. To do this you must shoot the five pods which ring the station, complete a tricky docking manoeuvre and then emerge from the rear exit of the station's hangar after immobilising its controls.

Once through these outer defences you encounter the planet's orbital fighters, very impressive looking in realistic 3D.

Having eliminated the fighters you go down to the planet's surface and shoot up



the ground defences.

The gameplay is nicely varied and the graphics very good but it is such a pity that the ship handles as though it is flying through porridge.

James Riddell

Sound	*8*
Graphics	*9*
Playability	*7*
Value for money	*8*
Overall	*8*

No trivial queries

Trivial Pursuit

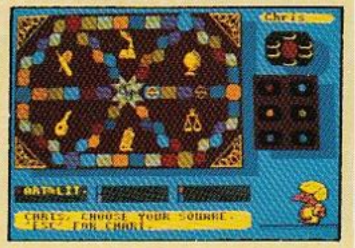
$44.95 - Tape
$59.95 - Disk

If you are part of the "in crowd" then you will have spent many an evening wracking your brains playing Trivial Pursuit, rather than getting tiddly down at the Dog and Duck. Now a computerised version has been released by Domark.

After loading, you are requested to enter the names of between one and six players. A comprehensive options menu allows you to alter several of the games features. A timer can be used to limit the time taken by a player to answer a question, and you have the option of using a dwarf called TP as question master.

When the game commences a very impressive board is displayed. Instead of rolling a die TP throws a dart at the numbers one to six and all possible moves are then highlighted on the board. You move your token to your chosen square and one of the six subject categories will be illuminated.

The scene now changes to TP's study where it's time to put your thinking cap on. Questions can be of three types: A straight forward written question as in the board game, a picture question for which a projection screen is lowered to display an image, or a musical question for which TP turns on his cassette player to play a tune. Answers are not typed into the computer but spoken out loud, and you press the fire button or spacebar to reveal the correct answer. You are then required to enter whether or not you got



it right.

Trivial Pursuit is a computer program of the highest calibre, though I doubt that it will ever replace the real thing.

Carol Barrow

Sound	*8*
Graphics	*9*
Playability	*8*
Value for money	*7*
Overall	*8*

Icy fate awaits

Frost Byte

$27.95 - Tape
$39.95 - Disk

DEEP beneath the frozen surface of the planet Cosmia, helpless Kreezers are being captured and eaten by the ice monsters. One little Kreezer called Hickey has escaped from his cage to make a bid for the surface and freedom.

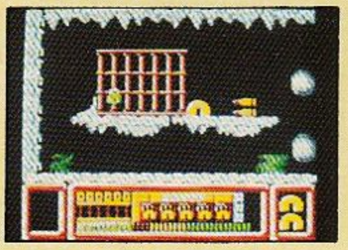
This square-ended caterpillar, which moves around like a slinky spring, can move left, right and jump. He can also pick up a variety of objects which he will find in the caverns. Just outside Hickey's cage there are two bullets which provide him with an initial supply of ammunition.

Other essential items are the diamonds which are to be found in the caverns. These come in three different varieties and will endow Hickey with special powers – red ones boost his speed, blue ones increase the height to which he can jump and the green ones allow him to fall further without fatal results.

Each screen is patrolled by numerous ice monsters who make life as difficult as possible for a runaway Kreezer. Some must be shot, but others are impervious to bullets and must be tackled with cunning and split second timing.

Although Hickey only appears to jump straight up and down don't be fooled into thinking that these are his only movements, they're not as you will discover for yourself.

The graphics used in the game, although very colourful, do give the odd flicker now and then. Also the text used on both the title and final score



screens is almost illegible.

With a little tidying up Frost Byte would make quite an attractive addition to anyone's software collection.

Jon Revis

Sound	*7*
Graphics	*7*
Playability	*8*
Value for money	*7*
Overall	*7*

... the COMPLETE personal organiser

Now there's a simple way to keep track of your money, plan your budgets, sort out your files and manage your time far more effectively.

PlanIt's three main modules – Personal Accounts, Financial Diary and Card Index – take care of all your day-to-day activities and help you rationalise your future financial position.

And there are two extra utilities – a Loan Calculator and a Calendar – to complete this remarkable package.

Personal Accounts Gives you up-to-the minute facts about your financial position at any time. Keeps separate accounts of your banking, cash transactions, credit card payments. Allows 24 individual accounts, up to nine different credit cards (and warns you when you reach your cash limit) and as many as 400 different transactions a month. Sets up your standing orders. Automatically updates relevant accounts with each transaction.

Card Index Create your own address book, phone directory, tape library title list. Use the flexible editor to enter or amend data. Sort and search. Call up detailed reports on contents in any form. Produce mailing labels on your printer.

Financial Diary All the features of the best desktop diary – plus much more. Enter up to 15 items per day and have them automatically sorted in time order. Add your expenses and have them totalled in separate categories. Speed search for entries, then mark them for future manipulation or replication.

DATABASE SOFTWARE

To order, please use the form on Page 72