APRIL, 1987

$4.50

A Database Publication

Computing with the AMSTRAD

The independent magazine for Amstrad computer users

Registered by Australia Post – Publication No. VBP7811

Page
Screen
Cutout
Do you want this Page 2 (YES/NO)
Mouse User
Rodent makes headlines in national magazine...
See you through the window!
AMSTRAD
DISC DRIVE
CPC6128

Desktop publishing on the CPC & PCW

Game of the Month: Smiley and the Grumpies

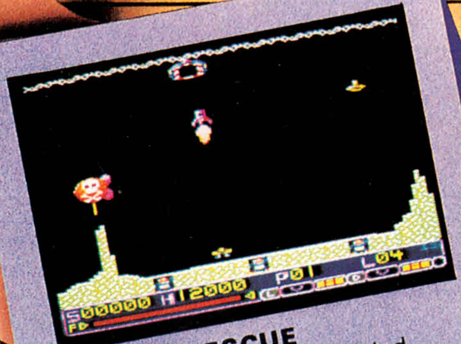
Desktop Publishing: We review Pagemaker, Fleet Street Editor and show you how to create stationery with your PCW

There's more on Basic, CP/M, Sound Graphics, Machine Code, Public Domain and Sound. There's the latest from the U.K., your letters, Software Survey

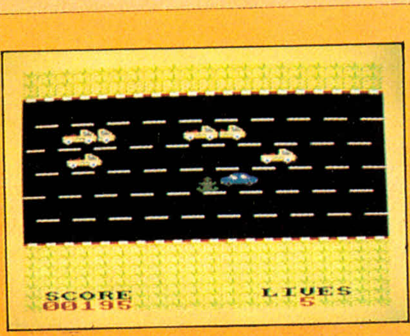
We also review Planner Calc and Scratchpad Plus

Chart Busters

Four action-packed classics to challenge the most expert gamester



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

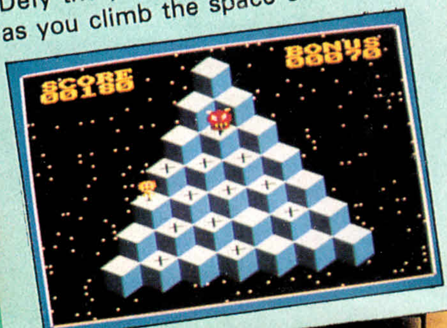


LEAPER
Escape the rush-hour chaos . . .
then brave the raging torrent

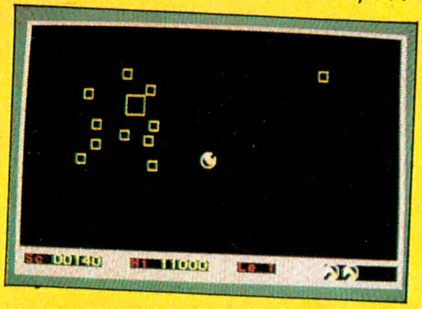
		Last month	Months in chart		
1		1			
2			2		
3			1	Faith add	
4				Budg	
5					6
6				he famous enty action. Highl ch	60
7				Gauntlet style view fro game. Very addictive with a huge map.	59
8	ight M	ne	1	Cute combination of adventure and arcade. Pop down menus and clever puzzles make this a very different game.	58
9	**Speed King** *M*	ne	1	Fast, high-speed action. Budget priced and certainly worth looking at.	45
10	**Thrust** *Firebird*	ne	1	Simple ids and lun	44
11	**Kung Fu Master** *US Gold*	11	2		
12	**Radzone** *Mas*	6			
13	**Star Firebi**				
14	**Jack**		1		
15			3		
16			3		
17				Mi ges while wearin	
18			4	They ha never di	
19				Helicopter flight simulation which is both accurate and fun. P	24
20			10		

▷ Non-mover △ Up

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

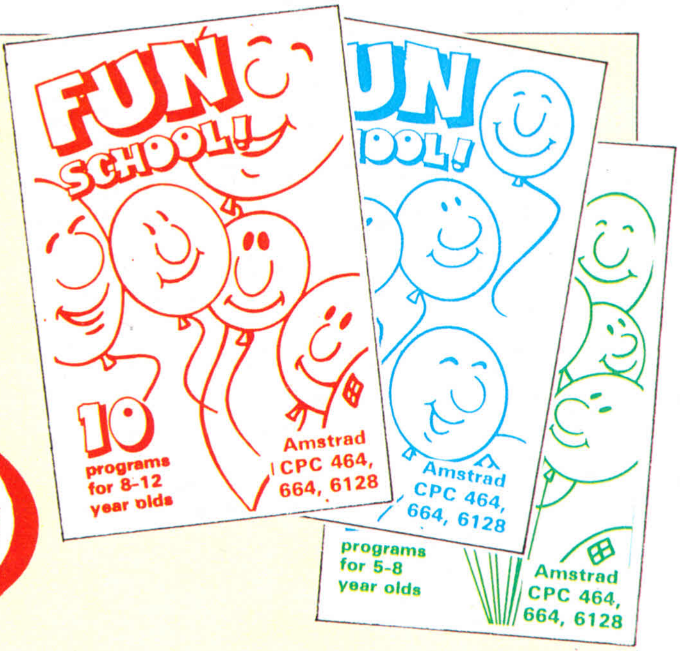
On tape	–	$15.95
On disc	–	$27.95

Use the order form on Page 72

Learning CAN be fun

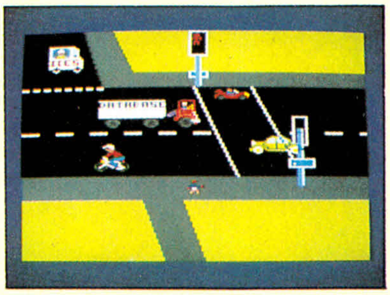
- Use your Amstrad to teach and amuse your children at the same time.
- Three packages crammed full of educational programs – and so easy to use!
- Each program has been educationally approved after extensive testing in the classroom.

ONLY
$15.95 - Tape
$27.95 - Disk

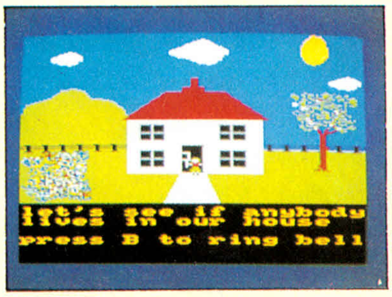

FUN SCHOOL!
10 programs for 8-12 year olds
Amstrad CPC 464, 664, 6128
programs for 5-8 year olds
Amstrad CPC 464, 664, 6128
Amstrad CPC 464, 664, 6128

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

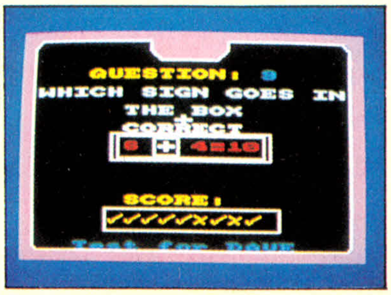


let's see if anybody lives in our house
press B to ring bell

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

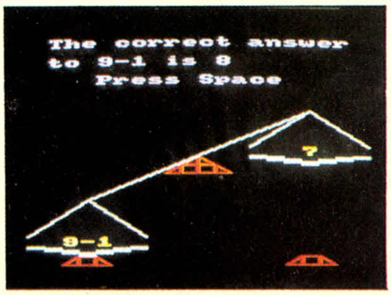
Ages 5-8

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



QUESTION: 9
WHICH SIGN GOES IN THE BOX
CORRECT
6 + 4=10
SCORE:
✓✓✓✓✓X✓X✓

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

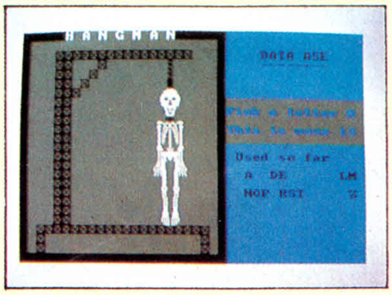


The correct answer to 9-1 is 8
Press Space
?
9-1

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

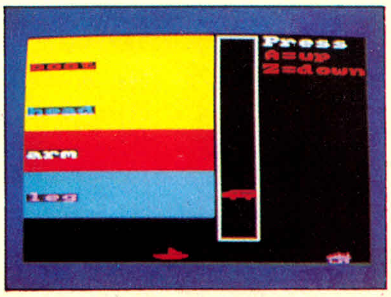
Ages 8-12

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



HANGMAN
DATA ASE
Used so far
A DE LM
NOP RST Z

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



Press
A=up
Z=down
head
arm
leg

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

To order, please use the form on Page 72

Rich pickings

INHERITANCE (PANIC IN LAS VEGAS)

TAPE	$24.95
DISK	$37.50

WHILE sitting alone in your flat pondering your dire financial situation you receive a telegram. Your aunt has died and left you her fortune – on condition that you can repeat her famous feat of winning $1,000,000 during one night in Las Vegas!

The game consists of three separate programs, each of which must be completed successfully before you can load in the next.

Part one is set in your block of flats. You make your way down from the seventeenth floor while avoiding your creditors.

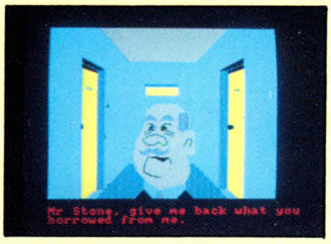
On your way to the ground floor you will encounter nine creditors who will each demand something from you. The trick is to know what items are wanted by which characters.

The pictures of the people are excellent and should provide a clue as to what they may have loaned you.

Successful completion of stage one gives you the codeword which is demanded by program two. This also means that you don't have to play every section of the game on future occasions.

Part two is set at the airport where I arrived with a wallet, watch, banana, comb and $130. As I passed through the doors my wallet vanished – I don't know why.

Eventually in part three you arrive in Las Vegas. The screen displays a map of the city, your holdall and your money. Using the cursor you select a casino from the map and once inside you can play the one armed bandits, Boule, or Craps. All games are big bold and great fun to play.



Mr Stone, give me back what you borrowed from me.

Inheritance is a deceptively difficult and excellently animated adventure/strategy game. I recommend it wholeheartedly. **Jon Revis**

Sound	6
Graphics	9
Playability	10
Value for money	10
Overall	9

Ball control

REVOLUTION

TAPE	$24.95
DISK	$37.50

TIME and again throughout history it has been demonstrated that the simplest designs are the best.

Take one bouncing ball and a couple of sugar lumps, mix with stunning graphics and you have the most addictive game of the year: Revolution.

The action takes place in what resembles a multi-storey car park. Each level is designed around a 5x5 grid pattern but some of the squares that make up the grid are missing.

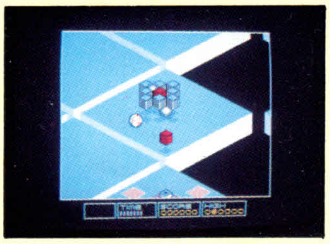
These provide convenient holes for an unwary ball to fall to it's death. You control the ball, which is capable of four heights of bounce in any of eight directions, though you cannot alter course in mid-flight.

Before beginning a level you are shown a map of its layout. This indicates the location of the missing tiles and a number of red spots, which represent the puzzles you must solve.

Each puzzle square contains two red cubes. Touch one and it turns white for several seconds. All you have to do is touch the second cube within this time limit and both cubes will vanish.

Solve all four puzzles on the level before your time expires and you can move up to level two.

As you might expect the cubes are normally in awkward positions, possibly even separated by skid grids and double bounce squares. A skid grid will kill your bounce completely but if it has an arrow on it you will be whisked off in the appropriate direction.



Revolution requires a combination of excellent ball control and animal cunning - buy it!

Carol Barrow

Sound	7
Graphics	10
Playability	10
Value for money	10
Overall	9

Spelling it rite

BETTER SPELLING

TAPE	$24.95
DISK	$37.50

IF you wish you could spell better but don't know how to go about it, then Better Spelling is for you.

Although it is designed for 9-14 year olds, it includes many words which I have seen adults spell wrongly.

Better Spelling is divided into 16 sections each demonstrating a certain spelling rule or grouping words which have irregular plurals, past tenses, and so on.

Throughout each section the number of right and wrong answers is shown on the screen together with the total number of questions asked – up to 50.

The time taken to work through each is shown on the screen, so you can try to improve your typing speed as well as your spelling.

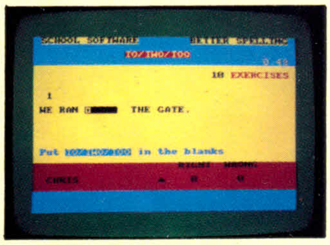
The first two sections are relatively easy – they are all fairly common plurals which most top junior children should know.

The third section on irregular plurals is more difficult, for instance, analyses, nuclei, larvae, and so on – it may help to know some Latin!

The sections on prefixes and suffixes require some vocabulary knowledge as the pupil has to choose the correct prefix or suffix from a given list and add these to the base words.

The last six sections are on common spelling errors – a sensible idea.

My overall verdict is that Better Spelling succeeds in its aim to improve spelling by reinforcing skills. As such it is certainly an asset for the busy teacher.



SCHOOL SOFTWARE BETTER SPELLING

TO/TWO/TOO

0.42

10 EXERCISES

1

WE RAN ▮ THE GATE.

Put TO/TWO/TOO in the blanks

RIGHT WRONG

CHRIS ▲ 0 0

Carole Sillers

Sound	n/a
Graphics	n/a
Educational value	8
Value for money	8
Overall	8

Computing With The Amstrad

April 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:	$ 45
South Pacific (inc. NZ)	$A65

'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.

'Computing With The Amstrad' is an independant publication and neither Amstrad Consumer Electronics plc or Amsoft or their distributors are responsible for any of the articles in this issue or for any of the opinions expressed.

Contents

SEPERATE BUSINESS INDEX - PAGE 41.

THE THINGS THAT STRINGS ARE MADE OF ...

THIS month we're going to be taking a look at string variables and exploring some of the Basic commands used to create and manipulate them.

You'll remember that string variables are the ones that end in the dollar sign, $. They hold groups of letters, numbers, punctuation marks and spaces, all lumped together as one.

To be slightly formal, we can store the word CATS in the string variable moggy$ using the following assignment statement:

```
LET moggy$ = "CATS"
```

After this, a quick

```
PRINT moggy$
```

will result in

```
CATS
```

appearing on screen.

Of course, we don't need the LET, but we do need the inverted commas. These are the delimiters, the things that mark the beginning and the end of the string. Try entering:

```
moggy$ = CATS
```

and see what you get.

You can have numbers making up the strings as:

```
figure$ = "1234"
PRINT figures
```

and

```
number$ = "0567"
PRINT number$
```

shows. However, these are string variables and you can't do arithmetic with them. Try:

```
PRINT number$*figure$
```

or

```
PRINT figure$-number$
```

and you'll see what I mean. The type mismatch error message means just that. You've used the wrong type of variables.

Having said that you can't do maths with strings, there is a way you can manipulate them that looks uncannily like addition. This is where you join or "concatenate" two strings together. To show what I mean, set up two strings variables such as:

```
first$ = "first"
second$ = "second"
```

and then enter:

```
join$ = first$+second$
```

The plus sign looks like we're doing an addition but how do you add first to second? What actually happens is that the two are joined together, as you'll see if you enter:

```
PRINT join$
```

and get:

```
firstsecond
```

in reply. This shows that the two strings have been strung together to make a third string.

Try:

```
PRINT figure$+numbers
```

and see the result. The answer to this "sum" is greatly different from the answer to:

```
PRINT 1234 + 0567
```

The first answer is a concatenated string, the second a number. It's just the same as the difference between:

```
PRINT 2+2
```

and

```
PRINT "2"+"2"
```

Remember, where string variables are concerned, the plus sign means "join together", not "add".

String concatenation lies behind last time's Program IX, this month's Program I.

```
10 REM PROGRAM I
20 CLS
30 asterisk$="*"
40 FOR row=1 TO 10
50 LOCATE 11=row,row
60 PRINT asterisk$
70 asterisk$=asterisk$+"*"
80 NEXT row
```

Take a close look at line 70. This takes the string variable asterisk$ and adds - or, rather, concatenates - an asterisk to it. The resulting string is stored back in asterisk$, now one character longer.

As the FOR...NEXT loop cycles, so asterisk$ grows in length, resulting in the triangle of asterisks.

Let's now go back to our original string. If you've reset your micro, recreate the variable with:

```
moggy$="CATS"
```

Have you noticed that when we used:

```
PRINT moggy$
```

we got the answer CATS and not "CATS" as we might have expected? The quotation marks have disappeared. The point is that the inverted commas are there to mark the ends of the string, not to be part of the string itself.

What if we had wanted them to appear? Could we do it by putting the whole thing in inverted commas? Try it and see. Unless your Amstrad's very different from mine, I think that you'll find that:

```
moggy$=""CATS""
```

results in a syntax error message. Don't despair though, there is a way of doing it making use of Basic's CHR$ function. But before we can do this we have to learn about something called the Ascii code.

As you probably know, your Amstrad works by numbers. Everything does, from flashing an angry syntax error message to attacking Earth with aliens in an arcade game is done by numbers. Even when it's dealing with words as in:

```
PRINT "CATS"
```

it does it by numbers. Every character has it's own code number.

The code for A is 65, while a question mark is represented by the number 63. All the letters, numerals 0-9 and punctuation marks have their own code numbers listed in a table known as the Ascii code.

For what it's worth, Ascii - pronounced "askey" - stands for the American Standard Code for Information Interchange. Table I shows a brief summary of the more useful codes.

The full set of codes is shown in Appendix III of the User Instructions. It's not exactly good reading but browse through it sometime and get an idea of how it's laid out.

So, to recap, each character you see on the Amstrad's screen has a number that represents it. The capital letters have the Ascii codes 65 to 90. You can convert these codes to their characters using the Basic function CHR$ mentioned earlier. Try entering:

```
PRINT CHR$(65)
```

and you'll have a capital A on the screen. It'll probably come as no surprise to find that:

```
PRINTCHR$(66)
```

produces B or that:

```
PRINT CHR$ (67)
```

gives C. Once you've grasped how the CHR$ function converts Ascii into alphabet you'll be able to follow such masterpieces as Program II.

```
10 REM PROGRAM II
20 PRINT CHR$(67);
30 PRINT CHR$(65);
40 PRINT CHR$(84);
50 PRINT CHR$(83)
```

I hope that you're feeling outraged by the indiscriminate use of PRINTs in the last program. We don't have to use a separate PRINT for each CHR$, we can string them all together as in:

```
PRINT CHR$(67);CHR$
(65); CHR$(84);CHR$(83)
```

Now you see where the term string comes from ...

So far, we've only used the Ascii codes in the range from 65 to 90.

Program III uses a FOR...NEXT loop to show the characters whose codes go from 32 to 126.

Here we not only have capital letters, there are also punctuation marks, lower case letters, numbers and even a space - 32. All these are the things that strings are made of.

```
10 REM PROGRAM III
20 FOR ascii=32 TO 126
30 PRINT CHR$(ascii);" ";
40 NEXT ascii
```

So using CHR$ and the relevant Ascii code we can create any string. In fact, the Amstrad has a whole set of graphics characters available using Ascii codes. You'll see these if you change the figure at the end of line 20 to 255.

If you want to know more about these characters I refer you to Computing with the Amstrad's excellent series on graphics from Geoff Turner and Michael Noels.

However, for the moment, let's just look at the capital letters produced by Program IV:

```
10 REM PROGRAM IV
20 FOR capitals = 65  TO  90
30 PRINT CHR$(capitals);" ";
40 NEXT capitals
```

Each time round the FOR...NEXT loop, capitals increases in value, ranging from 65 to 90. The result is that the CHR$ of line 30 prints out the whole of the alphabet in turn using capital letters.

Program V does exactly the same thing but in a rather better way.

```
10 REM PROGRAM V
20 offset=64
30 FOR letter=1 TO 26
40 PRINT CHR$(offset+letter);"
   ";
50 NEXT letter
```

Here, after offset has been set to 64 in line 20, the loop control variable letter ranges from 1 to 26. Line 40 sees the current value of letter added to the value of offset to produce an Scii code for the CHR$ to process.

This will range from 65, when offset is 1, to 90, when offset is 26 and so the upper case letters appear. But, if the result is the same as in Program III, why bother to rewrite it?

The answer is that I find a loop going from 1 to 26 producing the alphabet more intelligible than one going from 65 to 90 to the same end. Also, look how easy it is to produce lower case letters using the offset method.

```
10 REM PROGRAM VI
20 offset=96
30 FOR letter= 1 TO 26
40 PRINT CHR$ (offset+letter);"
   ";
50 NEXT letter
```

Notice how little Program VI differs from Program V, yet look at the difference in output. Here, having offset as 96 ensures that the values CHR$ works on go from 97 to 122.

These are the Ascii codes for the lower case letters, hence the differing output. Can you modify the program to produce the numbers 0 to 9? The codes range from 48 to 57.

To save yourself the bind of looking up the Ascii code for each character, Amstrad Basic has a very useful function, the aptly named ASC. This takes a character and returns its Ascii code. So:

```
PRINTASC("A")
```

returns 65 while:

```
PRINT ASC("a")
```

gives 97. You can use string variables inside the brackets as:

```
inside$="x"
PRINT ASC(inside$)
```

will show. Also ASC clearly differentiates between numbers and strings as shown by the differing results of:

```
PRINT ASC(7)
```

and

```
PRINT ASC("7")
```

Bear in mind that ASC only works on the first letter of a string. While it's perfectly allowable to have something like:

```
PRINT ASC("CAT")
```

you only get the code returned for the first letter. In other words,

```
PRINT ASC("XYZ")
```

gives exactly the same result as:

```
PRINT ASC("X")
```

the Y and Z being left out in the cold.

However, ASC is a lot more than just a quick way of getting an Ascii code. It can be useful in 'mug trapping' - catching user errors - as Program VII shows.

As you'll have found out if you've run it - and if you haven't, you should have - the program only accepts upper case letters. Line 40 checks the Ascii value of entry$. Only values in the range 65 to 90 produce the upper case alphabet, so if ASC(entry$) is below or above this value there's been an erroneous input. This is another way of saying someone's made a mistake or is trying to crash your program.

```
10 REM PROGRAM VII
20 PRINT "Enter an uppercase
   letter";
30 INPUT entry$
40 IF ASC(entry$)<65 OR
   ASC(entry$)>90 THEN CLS:
   PRINT "I said an uppercase
   letter": PRINT: GOTO 20
50 PRINT "Well done!"
```

The GOTO then sends the program back to line 20 for another try. Only when the Ascii code of entry$ is in the upper case range does the program get to the final message.

However, don't you think that Program VII might be a bit fierce? After all, someone might have put in p when they meant P. Rather than have the micro point out their error - possibly putting them off computers for life - why not have the Amstrad do it for them?

After all, it only takes an offset of 32 to allow for the 32 characters between an upper case letter and its lower case counterpart. Program VIII shows how it's done.

```
 10 REM PROGRAM VIII
 20 PRINT "Enter a letter ";
 30 INPUT entry$
 40 ascii=ASC(entry$)
 50 REM check if in letter range
 60 IF ascii<65 OR ascii>122
    OR
    (ascii>90 AND ascii <97)
    THEN GOTO 20
 70 REM if lower case subtract
    offset
 80 IF ascii>90 THEN ascii=ascii-
    32
 90 entry$=CHR$(ascii)
100 PRINT entry$
```

Here the Ascii value of entry$ is held in ascii. Line 60 checks that entry$ is either upper or lower case. If it isn't the mugtrap has the user trying again.

By the time the program gets to line 80, entry$ must be one or the other. Here it's tested and, if it's lower case - a code greater than 90 - 32 is taken away to make it upper case. In effect, ASC is allowing your Amstrad to correct human errors.

However, that's just one way of solving the problem and in some Basics it's the only way you have. Locomotive Basic has two functions UPPER$ and LOWER$, as you might guess, does exactly the opposite. Numbers and punctuation marks are left unchanged. After all, what is the capital form of 2? Try examples such as:

```
PRINT UPPER$ ("hijkl")
PRINT LOWER$ ("ABC12def")
PRINT UPPER$ ("pqr$*1KL")
```

and you'll soon get the grasp of them. You can use string variables inside the brackets if you want. Enter:

```
mixed$="aaCCff%%990 N"
PRINT UPPER$(mixed$)
```

if you doubt me. Program IX shows UPPER$ being used in a more efficient version of Program VIII.

```
10 REM PROGRAM IX
20 PRINT "Enter a letter ";
30 INPUT entry$
40 ascii=ASC(entry)
50 REM check if in letter range
60 IF ascii <65 OR ascii>122
   OR
   (ascii>90 AND ascii <97)
   THEN GOTO 20
70 PRINT UPPER$(entry$)
```

As you can see, lines 70 to 100 of the old program have been replaced by one line using UPPER$. Can you alter line 70 so that only lower case letters are displayed?

Before we leave the Ascii code I want to deal briefly with the codes in the range 0 to 31. These codes are rather different from the other codes we've used so far.

All the codes in the range 32 to 126 produce output on the screen when used with CHR$(). The codes from 0 to 31 don't display the character set but they do affect the micro. They're what are known as control codes, and that's what they do, they control the micro. Try:

```
PRINT CHR$(12)
```

and see, or rather, don't see what happens. As you'll have seen, or not, as the case may be, 12 is the control code for clearing the text screen. In effect it's the same as CLS.

Try:

```
PRINT CHR$(7)
```

and you'll hear what for tradition's sake is know as the bell.

Chapter 9 of the User Instructions gives all the control codes and their uses. Try them all and see if you can figure out what's happening.

I particularly like codes 8,9, 10 and 11 which move the text cursor backwards, forwards, down and up one character space respectively. You can have a lot of fun with them. Try to explain what's happening with:

```
PRINT "CATS";CHR$(8);CHR$(32)
```

You can even incorporate them inside string variables by adding-or, rather, concatenating - them together just like normal strings. You can see what I mean by entering:

```
 blank$ = "CATS"+CHR$(8)+CH
R$(8)+CHR$(8)+CHR$8)+CHR$(
32)+CHR$(32)+CHR$(32)+CHR$
(32)
```

After this, the string variable moggy$ contains four characters, four control codes and four spaces. Now when you

```
        PRINT blank$
```

you'll see nothing as the four backspaces overwrite CATS.

Don't worry too much if you don't grasp control codes straight away. Like everything else on the Amstrad, understanding comes with practice.

Just so long as you have the idea that numbers or Ascii codes can represent characters that's all you need to know for the time being.

Before we leave CHR$ entirely, do you remember our problem with "CATS"? Ascii codes come in handy here. Enter:

```
        moggy$=CHR$(34)+"CATS
        "+CHR$(34)
```

and then:

```
        PRINT moggy$
```

to get the sought-after

```
        "CATS"
```

It should come as no surprise that the Ascii code for inverted commas is 34. So, a cunning use of CHR$ allows you to display characters in a way impossible from the keyboard. Similarly,

```
        PRINT CHR$(123)
```

and

```
        PRINT CHR$(125)
```

give the curly brackets not found on any key.

And finally, how long is a piece of string? That's not such a silly question as it might seem. As you'll find out in the next couple of months, we do cut our strings into pieces - it's known as string slicing - and it's important to know their length. Because of this Basic has the function LEN.

It's not hard to use. Suppose, for reasons I can't imagine, you wanted to find the length of the string ABC using your Amstrad. All you'd have to do is enter:

```
        PRINT LEN ("ABC")
```

and 3 is returned as ABC is is three characters in length. It's hardly a shock, is it? More realistically, you might want to know the length of a string variable which could be changing all the time during the running of a program. Set up a string variable with:

```
        your choice$="whatever"
```

and

```
        PRINT LEN(your choice$)
```

will tell you the number of characters it contains.

As I said, LEN is fairly straightforward but there are a couple of special cases to watch out for. The length of a space is 1, not 0 as you might think. If you don't believe me, enter:

```
        PRINT LEN (" ")
```

and see for yourself. Remember, spaces count as one character, so:

```
        gap$=Hello Mum"
        PRINT LEN(gap$)
```

gives the answer 9, not 8.

Another special case is that of the null string, the string that contains nothing. Set one up with:

```
        null$=""
```

and find its length with:

```
        PRINT LEN(null$)
```

It makes sense that the answer is 0. After all, it contains no characters.

While it may seem a bit strange having a string that contains nothing, it comes in very handy as the control condition of a WHILE...WEND loop when slicing strings. Have a look at Program X.

```
10 REM PROGRAM X
20 entry$=""
30 WHILE LEN(entry)<>4
40 PRINT "Enter a four letter
   word ";
50 INPUT entry$
60 WEND
70 PRINT entry$
```

This is just a mugtrap using LEN to ensure that words of the right length are entered.

Line 20 sets entry$ to the null string. This isn't strictly necessary as the Amstrad assumes a string is the null string until told differently. However, it is good programming practice, making the listing more intelligible and, so, easier to debug.

I'll leave you to figure out how the rest of the program works and set this problem - The program is satisfied with 1234 but this isn't a word. Can you do anything about that?

Now let's string along a little bit further ...

If I want you to find the LEN of a concatenation of CHR$ I hope you'll understand what I'm Asciiing you for.

```
10 REM PROGRAM XI
20 entry$=" "
30 WHILE LEN(entry$)<>4 OR
   VAL(entry$)<>0
40 PRINT "Enter a four letter
   word ";
50 INPUT entry$
60 WEND
70 PRINT entry$
```

And if you've understood all that, Program XI - subtly different from Program X - should cause you no fears.

The problem was that this program, designed to allow only four letter words, accepted 1234 as input. The challenge was to mugtrap it so 1234 was not allowed. I hope you were able to see that a line such as:

```
55 IF ASC(entry$)<65 THEN
   GOTO 40
```

partially solved the problem. Now the program won't accept entry$ beginning with numbers but will accept things such as q234. We'll be able to deal with this when we come across more of the string-handling commands in a future article.

For now, we'll take a further look at string variables and seeing how they can be changed into numbers and vice versa. We'll also be learning why we should want to do it in the first place.

Before we go any further, however, it's important that you grasp the difference between a number and a string. If you can't explain the different results obtained from:

```
PRINT 2+2
```

and

```
PRINT "2+2"
```

then you'd better re-read some of the very first articles. However by now most of you will see that the first PRINT takes the 2+2 as numbers, adds them and displays the answer.

The second PRINT finds the inverted commas and displays, unchanged, everything that follows until it comes across another set of inverted commas.

To be formal, 2+2 is treated as an expression, or sum, while "2+2" is treated as a string, the inverted commas being the delimiters. As you'll remember, a string is just a collection of letters punctuation marks, spaces and numbers, all treated as one lump. That may not be a technically exquisite description but it works.

We tend to think of strings as collections of letters such as:

```
name$="Peter"
```

where the string variable name$ holds the letters that make up my christian name. However, combinations of letters and numbers such as:

```
name$="C3PO"
```

are allowed, and even strings that consist completely of figures, as:

```
name$="666"
```

shows. It's strings made up of combinations of letters and numbers, and the Basic functions that deal with them, that we're concentrating on this time.

PETE BIBBY solves a four letter word problem in our series for beginners

The first function we'll look at is STR$. This is used to change a number into a string. To convert the number 3 into a string variable called fred$ we use:

```
fred$=STR$(3)
```

When you enter:

```
PRINT fred$
```

you'll find that it now contains 3. Notice, however, that it's a string variable. The Amstrad won't like it if you try to do maths with it.

```
PRINT fred$*3
```

results in the "type mismatch" error message, showing you you've tried to use a string as a number.

STR$ not only works on numbers, it will work on numeric variables (variables that hold numbers) and expressions as well. Hence:

```
number=24
number$=STR$(number)
```

puts 24 into the string variable number$ while:

```
result$=STR$(72/12)
PRINT result$
```

shows that the string variable result$ holds the character for 6.

Notice that in the last case the expression inside the brackets is worked out before STR$ converts the lot to a string.

Using STR$ on negative numbers helps to highlight the difference between strings and numerics. It should come as no surprise to find that entering:

```
abc=1.23
fred$=STR$(abc)
```

results in the string variable fred$ holding the string -1.23.

There's one point to bear in mind when using STR$ to turn numbers into strings - spaces count as characters. If you find the length of fred$ with:

```
PRINT LEN(fred$)
```

it's given as five. This makes sense as the string consists of the negative sign and the decimal point as well as the figures. All count as characters.

However, set up a string with:

```
abc=123
fred$=STR(fred$(abc)
```

and then find the length of fred$ with:

```
PRINT LEN(fred$)
```

The result is 4, and not 3 as you might expect. This because, for reasons best known to Locomotive, who wrote the Basic, STR$ puts a space in front of any positive number it turns into a string. Hence the length of fred$ is 4, made up of one space followed by three figures. This invisible extra character can lead to problems if you're not careful when using STR$.

And if you've followed all that, try explaining the result of:

```
PRINT LEN(abc)
```

The more thoughtful, or perhaps

cynical, reader may be wondering what the point of all this is. After all, you can put a number into a string without using STR$. You can use an INPUT, as Program XII shows.

This amazingly trivial program asks you for a number and a street name and then prints out the address. It's hardly epic programming but it does make a point or two, so let's take a closer look at it.

Line 20 asks for the house number and stores it in the string variable number$. There we have an example of a number being turned into a string courtesy of INPUT.

```
10 REM  PROGRAM XII
20 INPUT "House number?",
   number$
30 INPUT "Street name?", street$
40 address$ = number$ +
   CHR$(32) + street$
50 PRINT "The address is ";
   address$
```

The next line puts the street name in street$ and line 40 concatenates (joins) the two together, putting the result in address$. The CHR$(32) sandwiched between them is just there to separate the number from the name.

Notice that address$ is holding two pieces of information - the street and the house number - in just one variable. What took two variables to hold is now contained in one. There'll be more about this later.

As you'll find out if you try:

```
PRINT number$+2
```

while number$ may hold a number, you can't do sums with it. In the program above this may be all right but for more complicated examples we may need to use the house number in our calculations.

Program XIII is just such a one. It asks for a street and number as before, but now the FOR...NEXT loop ensures that the next 25 addresses on that side of the road are displayed. It's very similar to Program XII, but notice that now the house number is held in the numeric variable *number*. This is because we'll be using *number* to determine the values that loop control variable takes.

```
10 REM PROGRAM XIII
20 INPUT "House number? ",
   number
30 INPUT "Street name? ",
   street$
40 FOR loop=number TO
   number+50 STEP 2
50 address$ = STR$(loop) +
   CHR$(32) + street$
60 PRINT "The address is ";
   address$
70 NEXT loop
```

Try changing all the *numbers* to *number$* and see what happens. The program crashes because you've tried to do calculations on a string. Never mind that the string may contain a number, you can't use it as a number, only as a string.

Line 50 has us using STR$ for the first time in a program. Here it takes the value of loop (increasing by 2 each time round the loop) and turns it into a string courtesy of STR$. This is then promptly joined with a space and street$ as before and stored in address$. Line 60 prints out the information stored in address$.

Of course it's not a very practical program, but it does have potential. It should only take a little imagination to see how it could be used as the basis of a larger program which kept track of, say, a paper round. Instead of displaying all the addresses on screen they could be printed out or, better, saved to tape or disk for further use.

Before we leave Program XIII I'd like to make two points about it. The first is that, unlike Program XII, the street number is kept in a numeric variable *number* and not a string variable, *number$*.

While it doesn't make too much difference in this case, I much prefer Program XIII's way of arranging the variables. After all, you expect to find a number in a numeric variable, while a string might contain all sorts of characters.

This may not be a problem in the above examples, but in long, complicated programs keeping numbers in numeric variables can save some elementary but time consuming errors.

The second point is that after what I've just said about keeping them separate, I've used STR$ to make number into a string! This may seem a bit contradictory but it's not, honestly.

While I advocate keeping numbers as numerics and non-numerics as strings when you're using them, I don't object to numbers being turned into strings so they can be stored more efficiently.

In Program XIII address$ manages to hold the information from two variables (number, street$) in one variable. This can be quite a saving, as Program XIV shows.

This gem of the programmer's art calculates the wages for three employees using a rather strange formula. The pay is $1000 for each year of age. No doubt in this concern they expect the work to kill you off young!

```
10 REM PROGRAM XIV
20 FOR loop=1 TO 3
30 INPUT "Name? ",name$
40 INPUT "Age? ",age
50 wage=age*1000
60 PRINT name$,age,wage
70 NEXT loop
```

'The trade off between flexibility and efficiency becomes a headache'

You should have no difficulty following how it works. The main body of the program is a FOR...NEXT loop which cycles three times. Each time round the loop an employee's name is entered and stored in name$. Similarly the employee's age is held in age. Line 50 calculates the pay by multiplying age by 1000 and putting the result in wage. Line 60 prints out the details.

Can you arrange them a little more neatly, maybe in columns with headings?

As you'll have seen if you've run it - and if you haven't you should have - the program works. But, having said that, it's all you can say about it. Does it really need three separate variables to keep track of things?

Line 60 is just printing out a simple message but it has to search for three variables to find the data it wants. Program XV does the same job, but does it in a different way.

```
10 REM PROGRAM XV
20 FOR loop=1 TO 3
40 INPUT "Name? ",name$
50 INPUT "Age? ",age
60 wage=age*1000
70 record$ = name$ + STR$(age) +
   STR$(wage)
80 PRINT record$
90 NEXT loop
```

This program only uses one variable, record$, to keep track of all the information about an employee. Each time round the FOR...NEXT loop lines 40 and 50 use INPUT to store the employee data in name$ and age. The next line calculates the pay as before, recording it in wage. Line 70 is the one that makes the difference.

Here STR$ is used to turn the numeric variables age and wage into string variables. These are immediately concatenated with name$ and the whole lot is stored in record$. So three pieces of information are now held in one variable instead of three variables as before.

It's a lot more efficient way of storing data, although you might have a bit of trouble adding table headings to neaten the display. And efficient as it is, beware of one thing.

When line 80 displays record$ the employee's name, age and wage are neatly separated by gaps. This is because the STR$ function of line 70 has introduced a space before each of the figures. In this case it's worked out, but if you use this method of holding data in one string, beware.

If you concatenated two normal strings they'll just be "glued" together and there'll be no space to show where the "join" is.

If you don't see what I mean, change line 70 to:

```
70 record$ = name$+STR$(age)
   +"anystring" + STR$(wage) +
   "anystring"
```

Now when you run Program XV you'll see that the information is messed up. It's difficult to see the ends of the age and the wage figures. The fields of some of the records, as they are known, are joined.

So bear in mind that a few spaces added during a concatenation might make things more intelligible. We'll be dealing with these fields when we come to using the LEFT$, RIGHT$ and MID$ functions.

While Program XV is arguably more efficient than Program XIV, it doesn't have its flexibility. With Program XIV it would be easy to print out the employee data in the order age, name, wage if we wanted it that way. We'd just change the order of the variables in line 60, leaving the main structure of the program intact.

With Program XV, however, we'd have to change line 70, altering the way the data is stored in record$. This problem can be eased using some of the string-handling techniques we haven't covered yet, but even so the changes aren't that simple.

So while the program may be efficient, it's not so flexible. You'll find as your programming experience grows that this trade off between flexibility and efficiency becomes a regular headache. It's up to each programmer to choose, though as computers get faster and memories larger and cheaper I suspect flexibility will become prized over efficiency.

For the time being, however, just notice how STR$ has been used to turn numbers into strings and store them in another string.

Information from several variables is compacted into one string variable. Later on we'll learn how to search these data strings for their information.

For the moment see if you can neaten up the display of Program XV. Is there any way of getting rid of the annoying questions that come between the display of the contents of record$? One solution lies with the control characters we dealt with last time. The following additions to Program XV will make things neater. Can you explain how they work?

```
 15 record$=" "
 70 record$ = record$ + CHR$(13) +
    CHR$(10) + name$ + STR$(age)
    + STR$(wage)
 80 CLS
100 PRINT record$
```

Now have a look at Program XVI, a yet more efficient version of Programs XIV and XV. It is shorter and uses fewer variables than the other two, yet does the same job. As you can see, there's no variable name, record$ being used instead. Similarly, rather than have a separate line and variable for calculating and storing the wage, STR$ is used with the expression age*1000 in line 50.

```
10 REM PROGRAM XVI
20 FOR loop=1 TO 3
30 INPUT "Name? ",record$
40 INPUT "Age? ",age
50 record$ = record$ + STR$(age) +
   STR$(age*1000)
60 PRINT record$
70 NEXT loop
```

While this may be a more efficient program, I don't like it all that much. Not only has it lost its flexibility, it's also a lot harder to understand. Dropping the variables name$ and wage tends to hide where things are happening.

When I come back to look at the program in a month's time how long will it take me to find where the wages are calculated? And if I start messing around with the program, will I realize that the record$ of line 30 contains a completely different set of information from the record$ of line 50?

I think of the three I prefer Program XV. It does the job, as far as I'm concerned, it does it at an acceptable level of efficiency, flexibility and comprehensibility. Also, its method of storing numbers in strings brings us onto the next topic.

So far we've been busy converting numbers into strings using STR$. Using it we've seen that we can combine a lot of numeric and string variables into one long string variable packed with information.

But what if, once we've converted our numbers into string, we want the numbers back again? Is there a way for converting strings, or parts of them, back into numbers? It's a good question.

Of course, the ASC function we covered earlier could be said to convert a string to a number, but that's not what we want. While:

```
PRINT ASC("A")
```

may give 65, so does:

```
PRINT ASC("AB")
```

and:

```
PRINT ASC("ABC")
```

Similarly,

```
PRINT ASC("1")
```

gives the same result as:

```
PRINT ASC("12")
```

which is hardly going to be much use extracting numbers from where we've stored them in strings. What we want is the aptly named VAL function. This turns the numeric part of a string back into a number again so:

```
PRINT VAL("12")
```

gives 12 and, to show that we really have transformed it into a number and that you can do maths with it, try:

```
PRINT VAL("12")*12
```

which gives 144.

For VAL to work on a string that string has to begin with a plus or minus sign or a number. If it begins with anything else, you get a 0 for your trouble. So:

```
PRINT VAL ("-56.67")
```

gives -56.67 while:

```
age$="+7"
age=VAL(age$)
PRINT age
```

gives 7. Incidentally, this last example shows that VAL can work on a string variable.

Beware, however, of strings that contain expressions, as VAL only works on the first number. Hence:

```
sum$="25+45"
PRINT VAL(sum$)
```

only gives 25 as the answer. And remember that it must start with a number or plus or minus sign. Try:

```
nono$="*123"
returned=VAL(nono$)
PRINT returned
```

and you'll get 0 returned. Similarly:

```
PRINT VAL ("BODGER")
```

gives 0 as it starts with a letter. Notice the difference between:

```
PRINT VAL("PETE34")
```

and:

```
PRINT VAL("34PETE")
```

The first returns 0 as the string starts with a letter, while the second returns 34. VAL takes all the numbers it can and then ignores the rest.

The VAL function allows another partial solution to the problem posed by Program XI. Using it, a line like:

```
55 IF VAL(entry$)<>0 THEN
   GOTO 40
```

solves the problem as efficiently as using ASC.

And that's all we're going to cover this time. We've seen how to use STR$ to convert numbers into strings and VAL to do the reverse. We've also come across an extremely compact way of storing information, one which we'll be dealing with a lot more.

See if you can use what we've covered in the past couple of months to create your own program to store and display information. And, if you've got any time left after that, can you mugtrap the above programs and maybe make Program XVI even more efficient (and less comprehensible)?

That should keep you busy until next time, when we'll leave strings for a while and READ all about DATA.

Stop Press

Just as we were going to press we learned from the U.K. that Database Publications, publishers of Computing With The Amstrad have purchased the U.K. magazine, 'AMTIX'.

It is not yet clear what effect this will have on Computing With The Amstrad but we are told that it is highly likely that AMTIX will be incorporated into Computing With The Amstrad.

As Computing With The Amstrad is already the world's largest selling Amstrad magazine based on Audited figures for the period July to December 1986, it would seem that incorporation of AMTIX into CWTA would give that magazine an unassailable lead in sales to Amstrad owners.

The Managing Director of Database Publications, Mr. Derek Meakin, has reportedly not completed his shopping list for Amstrad mags (he already owns Amstrad Computer User and Amstrad Professional Computing) and is said to have his eye on a publication aimed at the more serious side of Amstrad computing.

More Amstrad machines coming

AMSTRAD has confirmed it is working on additions to its PC and PCW lines for release during the second half of this year.

In keeping with its traditional policy, the company is against revealing details of the new machines until they are ready for release.

But sources close to the development team told Computing with the Amstrad that the new PC will be even cheaper than the current range of IBM compatibles. This is because it will contain more Amstrad-designed components than the present models which rely on third party suppliers for vital elements.

Alan Sugar believes he can build an even more cost effective machine by having greater control over component design and manufacturing.

For instance, Amstrad is said to be keen to produce its own hard disk drive - to replace the unit it currently buys from Tandon - and disk controller.

There will certainly be revised versions of the PC1512 board, and there is talk of an enhanced graphic adapter card for higher resolution and more colors.

The new PC is not expected to make its debut until late in the year, but there should be a new "improved" PCW this summer, probably incorporating a better quality printer.

Amstrad is planning a major thrust in the printer market with as many as four new models, not necessarily solely for its own machines.

A company spokesman would only say that: "There are a number of new machines under development at the moment, some further ahead than others.

"There will be additions to the PC and PCW ranges, though these products are some way away from completion.

"You won't see them during the first half of 1987, that's for sure".

What is a near certainty is that any new machines from the Amstrad stable will be cheaper than current models.

"If people think we're competing aggressively now, they'd better watch out in the coming months", Alan Sugar has warned.

PC PASSES POLY'S TOUGH TEST

THE Amstrad PC1512 has been given its toughest test so far by students at Leicester Polytechnic - and has passed with flying colors.

Sixteen of the new machines were installed in the commercial laboratory of the School of Mathematics, Computing and Statistics.

They were used continuously, 12 hours a day, five days a week by students - mostly running standard business software like Gem, Wordstar and dBase II. At the end of the first month of this gruelling test, principal lecturer Dr. Peter Thewlis said: "In terms of doing a job and value for money they're excellent.

"The PC1512s have proven their reliability. Three has been only one fault - a minor problem with a screen that was soon rectified.

"If they continue to be so reliable and the price remains roughly the same, we'll certainly be buying many more".

Leicester Polytechnic bought the 16 machines - part of a total order for 20 - in order to meet an increase in student numbers which meant that more teaching micros were needed.

The PC1512s are being used to give students a glimpse of the kind of commercial software - like Open Access, dBase II, WordStar and SuperCalc -that they will meet in the commercial world. The Polytechnic has also ordered Turbo Pascal and Turbo Prolog that will run on the machines.

Also under test is Novell's Pro Net-10 token and ring local area network which is running on a network of four Amstrad double floppy disk PC1512s with a modified hard disk PC1512 acting as file server. If tests prove successful, the 10Mbit per second LAN will be installed permanently.

Dr. Thewlis said the Polytechnic has experienced no difficulties in networking the PCs. "We've added the Pro Net cards to the networked 1512s and they've been fine", he says.

LOTUS LINE

AMERICAN developer Lotus has announced a new graphics program, Freelance Plus, to work with its 1-2-3 and other packages on the PC1512.

Also new from Lotus are HAL, a natural language interface, applications generator T-A-C, and stock market data extraction utility Signal.

New labels

THE Vonsoft Group has entered the Amstrad entertainment software market with two new labels - Vonsoft and Frozen Images.

There are 20 new releases promised for the first four months of this year, the first being graphic adventure game Arena for the CPC.

There is a cash price of at least $ to be won in connection with Arena, which costs $ on cassette and $ on disk.

Looking for writers

PUBLISHER OF award-winning entertainment programs for the CPC and PC, MicroProse has appointed Simon Barnard as software development manager.

Formerly with Activision as software producer, Barnard told Computing with the Amstrad: "I want to hear from program writers with new ideas or finished products.

"We are hoping to tap the rich resources of UK software authors in order to expand our range of products, particularly strategic simulations like Crusade in Europe".

French micros hit

UNBEATABLE competition from Amstrad machines has been blamed for lay-offs at leading French computer manufacturer Thomson.

The firm has announced the closure of its micro-manufacturing plant at St Pierre Montlimart, resulting in the loss of 450 jobs.

Thomson is expecting a loss of nearly $9 million on its computer operations because of inroads into it's home market made by imports of Amstrad micros in particular.

Plan Your Life.....

Three programs for the price of one from Database Software enable PCW and CPC owners to manage their time and money more effectively. The Planit personal organizer lets users control several complex aspects of their daily lives with a few keystrokes.

Personall Accounts maintains separate up-to-the-minute records of banking, cash transactions and credit card payments.

It can hold up to nine separate credit card records and can cope with 400 different transactions a month. It also sets up standing orders, automatically uipdates accounts with each transaction and even tells when credit limits are reached.

Financial Diary can accept up to 15 desktop diary entries a day and automatically sorts them into time order.

Personal expenses are totalled in separate categories and you can speed search for entries then mark them for future manipulation or replication.

With the Card Index you can create your own address book, telephone directory, or tape library title list.

A flexible editor allows you to enter and amend data, sort and search, ask for detailed reports on contents, and produce mailing labels on your printer.

Extra utilities include a loan calculator and calendar. The PCW version costs $57.95 on disk, CPC versions $36.95 on tape and $48.95 on disk. A special 'Australianized' version is distributed in Australia and New Zealand exclusively by Strategy Software.

Amstrad business sales doubled

AMSTRAD's share of the UK business micro market has more than doubled since the arrival of the PC1512.

And now the company is threatening IBM's traditional leadership in this sector.

A new industry survey reveals that in just one month Amstrad sales to business rocketed from 11 per cent of the market to 23 per cent.

And analysts predict that the next round of business micro sales figures will show a further increase in Amstrad's share.

Computer industry market research firm Romtec reports that Amstrad is now number two to IBM in the business sector - and closing the gap rapidly.

In the same period that Amstrad's share of the market more than doubled, IBM's share dropped from 38 to 31 per cent.

The Amstrad boom has pushed Apricot - with a 10 per cent share - into third place.

Romtec says that in the latter part of last year the UK business micro market as a whole grew by 18 per cent, with Amstrad accounting for five sixths of the increased sales.

"On the limited evidence we have so far Amstrad is primarily boosting market size rather than taking sales from other vendors", says Romtec.

Production rate for the PC1512 currently stands at 70,000 units a month - 45 per cent of them hard disk models - with no sign of demand letting up.

Grange Hill goes on CPC

LONG-RUNNING and often controversial TV series Grange Hill has been made into an adventure game for the CPC by Quicksilva.

The program is based on After Hours, one of the Grange Hill novels written by Phil Redmond who devised the series which began 10 years ago last month.

Quicksilva's version features leading characters Hollo and Gonch trying to break into school to retrieve a stereo that has been confiscated by a teacher.

There are plenty of puzzles and twists to the storyline, which has been devised by Deux Ex Machina and iD author Colin Jones.

Grange Hill - The Computer Game for the CPC range costs $19.95 on tape and $32.50 on disk.

CPC printer driver

QUALITAS 464 is a new printer driver for the CPC range from Seven Stars which gives near-letter quality on ordinary dot-matrix printers.

Professional features include equal-space justification and proportional character widths.

Five fonts in a variety of business styles are supplied. And each font can be modified or new ones designed using an editor included with the package.

Each font contains 96 characters and, depending on the amount of free memory, an alternate font can be loaded to give an additonal 96 - such as in the case of italics.

Several print modes enhance the versatility of each font, including double-height, double-width, half-width, subscript, superscript, emphasized and underlined. All except double-height are selected using standard Epson codes.

As Qualitas 464 intercepts all output to the printer it can be used with popular word processors like Protext, Tasword 464 and 464-D and also as a stand-alone program for printing Basic listings or disc files.

Suitable for all printers which are fully compatible with the Epson RX-80, Qualitas 464 costs $25 on tape, $32.50 on disk.

PageMaker deal

PRODUCER of the Vidi video digitizer for the CPC range, Rombo Productions has concluded a marketing deal with Advanced Memory Systems.

As a result Vidi is being bundled with AMX PageMaker to create MagazineMaker, a new pagemaker-digitizer package.

The Vidi contains an overlay file which enables AMX PageMaker users to digitize directly into their documents.

The interactive program supplied provides access to all Vidi facilities from joystick or keyboard.

MagazineMaker costs $349.95, Vidi on its own costs $239.95.

All AMX products are available from Strategy Software by mail order.

Looking through some new windows

MICHAEL NOELS

continues his series Part IV

LAST month we looked at text windows - a way of giving our text screen a new set of dimensions. For example, we could restrict text to the left hand side of the screen by defining a window limited to that area. Program I does this.

Line 30 defines the window we want. The first two figures give the number of the left and rightmost columns respectively.

Since we're in Mode 1 the rightmost column is normally column number 40. In line 30 we've restricted the window to half the screen by setting the rightmost column to 20.

The last two figures of line 30 specify the top and bottom row of the new window respectively. Normally these are rows 1 and 25. As you can see, we've confined the new window between rows 10 and 24. Figure 1 shows what's happened.

```
10 REM PROGRAM I
20 MODE 1
30 WINDOW 1,20,10,24
40 PAPER 2
50 PEN 3
60 CLS
70 LOCATE 3,8
80 PRINT"The Text Window"
90 WHILE INKEY$ = "": WEND
```

Since we're going to be dealing with various windows, changing pens and generally being very busy fiddling with our screen during this article, I suggest you set up the small Enter key as follows:

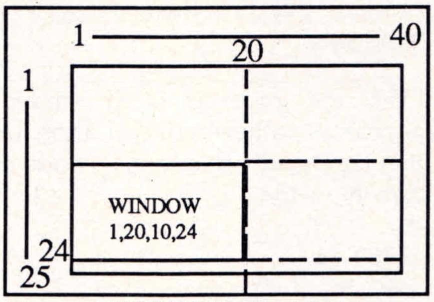


Figure I: WINDOW 1,20,10,24 in Mode 1

KEY 139, "CALL &BC02: CALL &BB4E : CLS" + CHR$(13)

On pressing the small Enter key

● CALL &BC02 will restore the pens to their default inks.

● CALL & BB4E will get rid of all text windows and set PAPER to zero and PEN to 1.

The effect is to restore our screen to the state it's in at switch-on, before we started meddling. Incidentally, be sure to use capitals where I have - Amstrad Basic won't translate these from lower case as it will other keywords, since they are in quotes.

Set up this key and press it between each program - or your latest program may suffer from the left-over effects of an earlier one.

Now have a look at Program II. In line 30, instead of:

WINDOW

we've got:

WINDOW #0

Also PAPER, PEN, CLS and LOCATE have acquired #0 as well. Yet if you run Program II you'll see that the output is identical to Program I. So what's going on?

Well, you may remember that last month I hinted that you could have several text windows on the screen at once. In fact, you can have eight distinct windows defined at any one time.

To distinguish between the windows we give them numbers, #0 to #7. We also use these hash (#) numbers to determine the windows that we want our PAPER, PEN, CLS, LOCATE and PRINT commands to effect - as we'll see later.

```
10 REM PROGRAM II
20 MODE 1
30 WINDOW #0,1,20,10,24
40 PAPER #0,2
50 PEN #0,3
60 CLS #0
70 LOCATE #0,3,8
80 PRINT"The Text Window"
90 WHILE INKEY$ = "": WEND
```

At the moment, though, you're probably wondering why the #0s in Program II haven't made any difference. The reason is that window zero is the default window. That is, if we just say WINDOW without specifying any hash number, the micro takes it for granted that we mean WINDOW #0.

The same goes for PEN, PAPER. If you don't specify the hash number the Amstrad assumes you mean the command to be effective on WINDOW #0.

Notice that in line 80 of Program II I haven't specified #0 - it doesn't matter of course, since it defaults to window zero anyway. For neatness' sake, though, you might want to replace it with:

```
80 PRINT #0,"The Text Window"
```

The INKEY$ in line 90 of both programs is just to delay the appearance of the Ready prompt.

Program III actually proves that windows other than WINDOW #0 exist. Here all my commands are aimed at window one. Apart from the #1 replacing the #0, the listings of Programs II and III are identical. The output of the program is not, however, Run Program III and see.

```
10 REM PROGRAM III
20 MODE 1
30 WINDOW #1,1,20,10,24
40 PAPER #1,2
50 PEN #1,3
60 CLS #1
70 LOCATE #1,3,8
80 PRINT #1,"The Text Window"
90 WHILE INKEY$ = "": WEND
```

Notice anything? The window is in the same place, the color's the same and the message identical - so what's different?

Actually nothing differs until you press a key and drop through the WHILE...WEND of line 90. You'll then notice that the Ready prompt appears at the top left hand corner of the screen, not inside the window we've defined, as before.

The reason is that the micro ends all its own messages - such as Ready and Press PLAY - to us via window zero. This is also the window in which our commands to the micro, such as LOAD and SAVE, appear.

As Program III doesn't alter window zero, as do Programs I and II, the Ready prompt appears on the "normal" screen. You might like to alter Program III so that you're using WINDOW #5 or some such. Remember, the eight windows are numbered #0 to #7.

So far we've just defined one window at a time on the screen, although we used different window numbers. In Program IV we have two distinct windows on the screen at once.

Enter Program IV, but see if you can predict its outcome before you run it. Assuming you remembered to press the small Enter key before running it you should see the messages of lines 80 and 100 printed out in their respective windows.

```
10 REM PROGRAM IV
20 MODE 1
30 WINDOW #0,1,20,10,24
40 WINDOW #1,21,40,10,24
50 PEN #0,,2
60 PEN #1,3
70 LOCATE #0,3,8
80 PRINT #0,"The Text Window"
90 LOCATE #1,3,8
100 PRINT #1, "The Other
    Window"
110 WHILE INKEY$ = "": WEND
```

LIST the program - it should appear in rather cramped form in the lefthand window, window zero. Now try:

LIST #1

The listing will now appear in window one on the right. So you can LIST to different windows.

Let's have a closer look at Program IV: The two windows (zero and one), defined by lines 30 and 40, are shown in Figure II.

Line 50 chooses PEN 2 for the writing in WINDOW #0 with:

50 PEN #0,2

This gives us cyan text in window zero (assuming we haven't been playing with INK and changing the inks assigned to the pens).

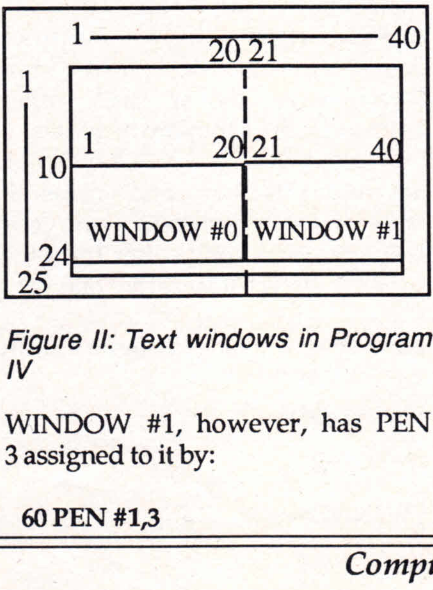


Figure II: Text windows in Program IV

WINDOW #1, however, has PEN 3 assigned to it by:

60 PEN #1,3

This will give us red text in window one.

An important point to remember is that LOCATE works relative to its own window. That is, the coordinate system for each LOCATE #n starts from the top left hand corner of WINDOW #n.

So:

70 LOCATE #0,3,8

will locate the text cursor at the third column and eighth row of window zero. On the other hand:

90 LOCATE #1,3,8

will locate the text cursor at the third column and eighth row of window one. Actually, each window has its own text cursor so really I should say that:

90 LOCATE #1,3,8

locates window one's text cursor at the third column, eighth row of window one.

The Amstrad itself keeps track of where each cursor is in its window. To prove the cursors really are separate, add lines:

```
105 PRINT #0, TAB(9) "zero";
106 PRINT #1, "one"
```

As you'll see, fiddling with the position of the text cursor in one window doesn't affect its position in the other. Mind you, the messages don't really give an idea of the extent of the windows. Adding the following lines should do so:

```
45 PAPER #0,3: CLS #0
46 PAPER #1,1: CLS #1
```

After you've played with Program IV to your heart's content, enter:

CLS #2

Well, it's not surprising is it? We haven't defined window two!

A point about terminology. So far we've said that PRINT #n prints in WINDOW #n. Actually we should say that PRINT #n prints to STREAM #n (which just happens to flow into WINDOW #n).

Streams are just distinct channels the micro uses to separate its INPUT and OUTPUT flows. There are nine output streams:

0-7 Text screen streams - the windows.
8 Printer stream.
9 Cassette output.

The input streams are:
0-7 Keyboard, prompts to relevant windows.
8 Keyboard, prompts to printer stream.
9 Cassette input.

Those of you lucky enough to have printers will have used these concepts when listing with:

LIST #8

Others may have experienced the ideas when using cassette or disk files with:

INPUT #9
PRINT #9

Anyway, enough of that. In this article we're sticking to windows, with output streams 0 to 7. So let's have a look at all eight windows at once. Run Program V.

```
10 REM PROGRAM V
20 MODE 0
30 PAPER 10: CLS
40 FOR pane% = 3 TO 17 STEP 2
50 number% = ( pane% -3 ) / 2
60 WINDOW #number%,
   pane%,pane%+1,1,25
70 PAPER #number%,number%+1
80 CLS #number%
90 PEN #number%, number%
100 LOCATE #number%,1,12
110 PRINT #number%, number%
120 NEXT pane%
130 WHILE INKEY$= "" :WEND
```

I leave it to you to work out the gory details - suffice it to say that the windows are labeled 0 to 7.

By the way, the effect of the program is to leave 10 strips on the screen. Don't worry, there are really only eight windows. The left and rightmost strips are left over from when we cleared the whole screen before defining our windows.

If you don't believe me, add the following to the program:

```
140 FOR loop% = 0 TO 7
150 PAPER #loop%, 10
160 CLS #loop%
170 WHILE INKEY$ = "":
    WEND
180 NEXT loop%
```

The dance of the eight veils, as it's known around the office, will clear a window to the original background color each time a key is pressed.

What happens if windows overlap? It's quite logical - they share. For example, if windows one and two overlap, clearing either clears the overlapping zone.

Program VI illustrates three overlapping windows. As you'll see, the last one cleared is the only one to survive intact. If you really want to test your skill with windows, try to reverse the way these colored "cards" are dealt.

Of course the ultimate overlapping is when they nest inside one another. Take a look at Program VII. It should be easy enough to follow.

One of the major uses of streaming text to windows is to separate input and output - you use one window to display the computer's messages to you, and another for your own replies.

```
10 REM PROGRAM VI
20 MODE 1
30 hor% = 5 : across% =20
40 ver% = 3 : down% = 12
50 FOR pane% = 0 TO 2
60 WINDOW #pane%,
   hor%,hor%+across%,ver%,
   ver%+down%
70 PAPER #pane%, pane% + 1
80 CLS #pane%
90 hor% = hor% + 5
100 ver% = ver% + 4
110 NEXT pane%
120 CALL &BC02
130 CALL &BB4E
```

Adventure games often use windows to give a split-screen format, reserving the top of the screen for room descriptions and the bottom for commands and so on.

Program VIII shows an approach to a "guess the number game" using three windows. The first is for your input and the micro's prompts. The other two display your previous guesses. Those greater than the target number are shown in the left hand window and those less in the right.

As it stands, it needs a bit more work on it. However it does show how careful choice of windows can add considerably to a program's appearance and ease of use.

Now suppose we've defined two windows with WINDOW #0 and WINDOW #1. All subsequent PRINT #0 and so on refer to window zero while all PRINT #1 refer to window one.

```
10 REM PROGRAM VII
20 MODE 1
30 hor% - 4: across % = 32
40 vert% = 4: down% = 16
50 FOR pane% = 2 TO 0 STEP -1
60 WINDOW #pane%, hor%,
   hor% + across%, vert%, vert%
   + down%
70 PAPER # pane%, pane%+1
80 CLS #pane%
90 hor% = hor% + across%/4:
   across% = across% / 2
100 vert% = vert%+ down%/4:
    down% = down%/2
110 NEXT pane%
120 PEN 3
```

Amstrad Basic has a WINDOW SWAP command that allows you to completely interchange the two windows. For example, if you entered:

WINDOW SWAP 0,1

the screen area that was labeled as window one magically becomes window zero and vice versa. So, after the swap:

PRINT #0,"ABC"

```
10 REM PROGRAM VIII
20 WHILE -1
30 GOSUB 120:REM Initialise
40 WHILE target% <> number%
50 GOSUB 260:REM input number
60 IF target% = number% THEN GOSUB 350
: REM Correct guess
70 IF target% < number% THEN GOSUB 430
: REM Too high
80 IF target% > number% THEN GOSUB 500
: REM Too low
90 WEND
100 WEND
110 END
120 REM *** Initialise ***
130 REM *** Set Mode   ***
140 REM *** and windows***
150 MODE 1
160 PAPER 2:CLS
170 WINDOW #1,9,16,3,14
180 PAPER #1,1:PEN #1,3: CLS #1
190 WINDOW #2,25,31,3,14
200 PAPER #2,3:PEN #2,1: CLS #2
210 WINDOW #3,4,36,19,24
220 PAPER #3,0:PEN #3,1: CLS #3
230 LOCATE #1,2,12: LOCATE #2,2,12
240 target% = RND(1)*100 + 1
250 RETURN
260 REM *** Input Number ***
270 REM *** Range 1 - 100***
280 CLS #3
290 LOCATE #3,6,3
300 number% = 0
310 WHILE number%<1 OR number%>100
320 INPUT #3,"Number"; number%
330 WEND
340 RETURN
350 REM *** Correct Guess ***
360 CLS #3
370 LOCATE #3,5,2
380 PRINT #3, "Well done." number% "is
 correct."
390 LOCATE #3,3,5
400 PRINT #3, "Press Space for another
 game."
410 WHILE INKEY(47)<>0: WEND
420 RETURN
430 REM *** Too high ***
440 CLS #3
450 LOCATE #3,5,2
460 PRINT #3,"Wrong!" number% "is too
big."
470 PRINT #1, TAB(2) number%
480 GOSUB 570 :REM Get Space
490 RETURN
500 REM *** Too low ***
510 CLS #3
520 LOCATE #3,5,2
530 PRINT #3,"Wrong!" number% "is too
small."
540 PRINT #2, TAB(2) number%
550 GOSUB 570 :REM Get Space
560 RETURN
570 REM *** Wait for Space ***
580 LOCATE #3,3,5
590 PRINT #3, "Press Space for another
 guess."
600 WHILE INKEY(47)<>0:CALL &BB18: WEND
610 RETURN
```

Program VIII

will cause ABC to appear in what was the previous window one.

Really, all that's happened is an exchange of labels. Mind you, considering that this will affect all subsequent PEN, PAPER and PRINT statements involving these labels, the effect can be quite impressive!

Program IX shows the results of window swapping. Each time you press space the text in the window interchanges as the labels to the windows are swapped (Line 230).

```
10 REM PROGRAM IX
20 MODE 1
30 WINDOW #0,1,20,10,24
40 WINDOW #1,21,40,10,24
50 WHILE -1
60 GOSUB 80
70 WEND
80 REM Message Routine
90 CLS#0:CLS#1
100 PRINT #0:PRINT #0:PRINT #0
110 PRINT #0,"Here we have two"
120 PRINT #0,"text windows."
130 PRINT #0,"This is window 0"
140 PRINT #1,"This is window 1"
150 PRINT #1,"Now let's do a"
160 PRINT #1,"window swap with"
170 PRINT #1
180 PRINT #1,"WINDOW SWAP 0,1"
190 PRINT #1
200 PRINT #1,"Press Space when"
210 PRINT #1,"you are ready."
220 WHILE INKEY(47)=-1:WEND
230 WINDOW SWAP 0,1
240 RETURN
```

Program IX

I can't stress too strongly that it's only the numbers labeling the windows that are swapped. The actual attributes of PEN, PAPER and so on associated with that window still remain with that window under its new label.

```
10 REM PROGRAM X
20 MODE 1
30 WINDOW #0,1,20,10,24
40 WINDOW #1,21,40,10,24
45 PEN #0,2: PEN #1,3
50 WHILE -1
60 GOSUB 80
70 WEND
80 REM Message Routine
90 CLS#0:CLS#1
100 PRINT #0:PRINT #0:PRINT #0
110 PRINT #0,"Here we have two"
120 PRINT #0,"text windows."
130 PRINT #0,"This is window 0"
140 PRINT #1,"This is window 1"
150 PRINT #1,"Now let's do a"
160 PRINT #1,"window swap with"
170 PRINT #1
180 PRINT #1,"WINDOW SWAP 0,1"
190 PRINT #1
200 PRINT #1,"Press Space when"
210 PRINT #1,"you are ready."
220 WHILE INKEY(47)=-1:WEND
230 WINDOW SWAP 0,1
240 RETURN
```

Program X

Program X illustrates this. Text in the left hand window appears in cyan, whereas that in the right hand window appears in red.

Now you might suppose that when you did the window swap in 230 the colors would also swap. Not so. It's only the labeling number that changes, not the physical characteristics - size, pen/paper colors, cursor position and so on remain unchanged for that window.

One important use of WINDOW SWAP is so that our windows can share subroutines. The subroutine beginning at line 160 of Program XI contains a rather primitive text centering routine which operates on window zero by default.

Having defined two windows (lines 40,50) we can then use this routine to centre text in both. It works straight away for the text in window zero (lines 70 to 90).

To get it to operate on the second window we perform a window swap line 100 so that it's now referred to as WINDOW #0. Then we can centre our text as normal (lines 110 to 130).

```
10 REM PROGRAM XI
20 MODE 1
30 wide% = 20
40 WINDOW #0,1,20,10,24
50 WINDOW #1,21,40,10,24
60 PEN #0,2: PEN #1,3
70 text$ = "Centered": GOSUB 160
80 text$ = "in": GOSUB 160
90 text$ = "First": GOSUB 160
100 WINDOW SWAP 0,1
110 text$ = "Centered": GOSUB 160
120 text$ = "in": GOSUB 160
130 text$ = "Second": GOSUB 160
140 WHILE INKEY$="":WEND
150 END
160 REM Centring Routine
170 position% = (wide%-LEN(text$))\2
180 PRINT TAB(position%) text$
190 RETURN
```

Program XI

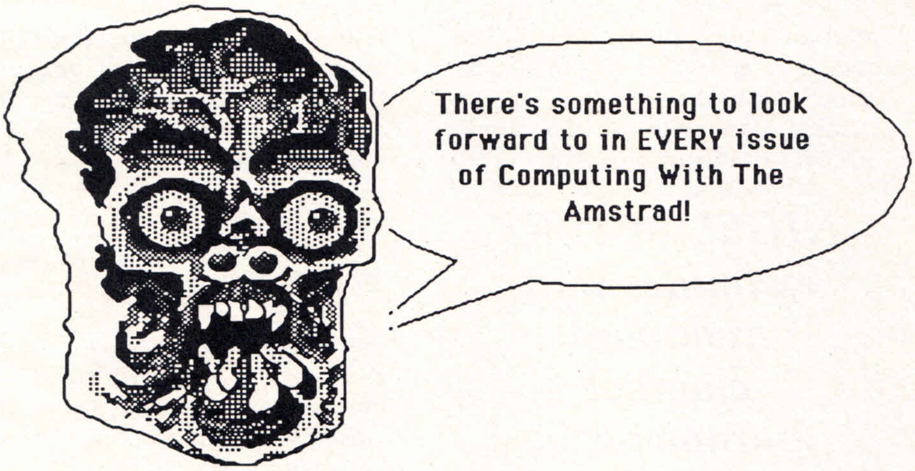
So far I've only swapped between one and zero. Of course WINDOW SWAP works between any pair of windows. For example you could add the following line to Program VIII:

225 WINDOW SWAP 1,2

Now the guesses greater than the target number will appear in the right hand window and those that are less in the left hand one.

If you are feeling adventurous you could try window swapping with an undefined window!

That's all for this month. Next it's the graphics screen.


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

You wanna play a note? Then get in the queue

NIGEL PETERS examines that awkward channel parameter in Part VI of his series on CPC464 sounds

FIRST the good news. This months we're going back to our good old basic SOUND command.

Made up of:

SOUND channel, pitch, duration, volume

It ignores the envelopes and noise, so things should be a lot simpler. The bad news is that the channel parameter isn't quite the simple beastie that I told you it was.

However, for the time being, let's stick to the old formula where a channel parameter of 1 produces a noise on channel A, while 2 uses channel B, and 4 plays on channel C.

Let's play a note on channel A with:

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

This produces a note of pitch 200 which lasts for one second and is played at full volume. To play the same note on channel B we'd use:

```
SOUND 2,200,100,7
```

and on channel C:

```
SOUND 4,200,100,7
```

Exciting stuff isn't it? Using these three commands you may think that:

```
SOUND 1,200,100,7:
SOUND 2,200,100,7:
SOUND 4,200,100,7
```

would produce the same note at the same time on each of the channels. You'd be right.

But what does:

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

produce? Try it and see. If you can find any difference between that noise and the one produced by the previous set of three sound commands you've got better ears than I have. (Which probably explains a lot about this series!)

The point is that:

```
SOUND 1,200,100,7:
SOUND 2,200,100,7:
SOUND 4,200,100,7
```

and:

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

produce the same noise. The question is why? It must be something to do with the 7 in the channel parameter of the single SOUND command, but what?

After all, there are only three channels and we've got numeric labels for each of them. So what's happening? How do we get three notes playing at once from a single SOUND command?

Well, take a look at the channel parameter of:

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

again. Notice that the sound is played on channels A, B and C. Now we know that the number labels for these channels are 1,2 and 4. And guess what 1+2+4 adds up to?

Yes, it's the 7 that we had in the channel parameter of the solitary but very productive SOUND command.

From this you should be able to see that when we want a SOUND command to play a note on more than one channel, we add together the individual channel numbers to form a new, combined number.

When the Amstrad finds this number it realizes that you want the note to play on more than one channel and takes the appropriate musical action.

In the case of:

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

the micro came across 7 in the channel parameter and immediately realized that this wasn't the normal 1, 2 or 4 it might have expected.

Being a computer, and therefore good at maths, it then broke the number down into the component 1=2+4 and played the note on channels A, B and C.

So, by adding together the channel parameters we can make one SOUND command to do the work of two or three.

To play a note on channels A and C at the same time, we'd use a channel parameter of 5(1+4) as in:

```
SOUND 5,300,50,6
```

while to play the same note on

channels A and B we'd use a parameter of 3 (1+2).

Table I shows how the differing channel parameters bring different combinations of channels into play.

Now let's leave combined channel parameters and get back to our simple SOUND commands.

Try entering:

```
SOUND 1,200,100,7:
SOUND 1,300,100,7
```

and see, or rather, hear what happens. You get a one-second note at pitch 200, followed by a one second note at pitch 300. This is more or less what you'd expect.

The Amstrad comes to the first SOUND command, processes it and starts playing the note.

While this note is playing, the micro comes to the second SOUND command and processes that. Now it has a problem.

The second SOUND command has told it to play a note on channel A but, in the very fast world of micro electronics, the first note is still playing.

What can it do? Should it interrupt the first note? Or should everything grind to a halt until the first note is finished and the second SOUND can be processed?

The answer is that the Amstrad lets the first note carry on playing for its full duration and pops the second on to what is known as a queue.

When the first note is finished, the micro looks at the queue, finds the second note stored there, and plays it.

The good thing about the queue is that the program can carry on doing other things, rather than waiting for SOUNDs to be processed.

If you don't follow that, try:

```
SOUND 4,1000,500,5:
SOUND 4,100,500,5:
PRINT "The Amstrad gets on
with other things"
```

Here you might expect from the order of the lines that you'll hear the first sound, then the second, then see the message.

In fact what happens is that the micro start the first sound, puts the second on the queue for channel C and then goes on to the next statement.

This means that the first note is heard, the message appears and only when the duration of the first note is finished do we hear the second note.

The queue is used for storing notes while the micro gets on with processing any Basic statements that follow.

There is a separate sound queue for each channel, each totally independent of each other.

These queues, however, are limited in length. There is only room for one note to be playing and four notes to be stored on the queue. After that you've got problems, as shown by Program I.

As you can see, the FOR...NEXT loop is trying to play ten notes, one after another. After each note a message is to be printed.

It seems fairly straightforward, but if you expected a note to be played, then a message, then another note, then another message, you'll have been disappointed.

The first time round the loop the Amstrad plays the note and prints the message.

However, the next four times round the loop the first note hasn't finished playing.

What happens is that the four notes produced by these cycles are popped on to the queue waiting for their chance.

There are no problems with the PRINT commands, however. There's nothing to stop these appearing as normal, so we get the first five messages appearing before the first note has finished!

```
10 REM PROGRAM
20 FOR note= 1 TO 10
30 SOUND 1,100#note,100,7
40 PRINT "Note number "note
50 NEXT note
60 PRINT "The program is
   finished..."
70 PRINT "the music lingers on."
```

Now the Amstrad has problems. There's a note sounding, and there are four notes on the queue, waiting to play.

The queue is full, there's no more room, yet on the sixth time round the loop, line 30 is telling it to make a noise! Something has to give.

When the first note stops sounding, the second comes off the queue and is played. The remaining three notes "shuffle" along the queue leaving a space at the end.

Now the Amstrad can carry on, putting the sixth note on the end of the queue and printing the appropriate message.

The trouble is that the next time round the loop the micro comes up against the same problem, a SOUND to be processed and no room on the queue.

Again the program grinds to a halt until a space is free.

You can see that this stopping and starting explains why the first five messages appear in a flash while the next five come at one-second intervals as the notes change.

As I said before, there is a queue for each channel, each queue holding four notes as well as the one currently playing on that channel.

Each channel queue operates independently.

Having said that, though, once you attempt to put a note on a channel queue that's already full, then everything comes to a halt until a space appears for it.

In this way one channel's problems can mess up another channel.

Program II shows how a note can sound one channel even though another has a full queue. Can you explain why?

```
10 REM PROGRAM II
20 SOUND 1,100,5000,7
30 FOR note=1 TO 10
40 SOUND  2,100#note,100,7
50 PRINT "Note number "note
60 NEXT note
```

Leaving queues for a moment, listen to Program III.

```
10 REM PROGRAM III
20 SOUND 1,239,100,5
30 SOUND 1,190,100,5
40 SOUND 1,213,100,5
50 SOUND 1,239,100,5
```

This consists of a four-note "tune". Each note is played on channel A one after the other, each note lasting for one second. The whole tune takes four seconds. Now bend your ear to the charms of Program IV.

```
10 REM PROGRAM IV
20 SOUND 2,119,100,7
30 SOUND 2,127,100,7
40 SOUND 21,159, 200,7
```

This is made up of three notes played one after the other on channel B. The first two notes each last for one second, the last one for two seconds. Again, the whole tune lasts for four seconds.

Now, in the interests of harmony and to excite music-lovers everywhere, let's combine the two tunes. One way of doing this is shown in Program V:

```
10 REM PROGRAM V
20 SOUND 2,119,100,7
30 SOUND 1,239,100,5
40 SOUND 2,127,100,7
50 SOUND 1,190,100,5
60 SOUND 2,159,200,7
70 SOUND 1,213,100,5
80 SOUND 1,239,100,5
```

The way that the SOUND

commands are arranged reflects the structure of the combined tunes. Line 20 starts a one-second note on channel B while line 30 starts another one-second note playing on channel A.

Since they are both on different channels, they are played simultaneously, and so a pleasing harmony is heard. (Well, I think it's pleasing.)

The next two lines 70 and 80 putting two one-second notes on channel A.

The whole tune lasts four seconds, and, as we've kept below six notes on each channel, we've no queue problems.

Program VI plays exactly the same melody even though the SOUND commands are in a different order.

```
10 REM PROGRAM VI
20 SOUND 1,239,100,5
30 SOUND 1,190,100,5
40 SOUND 1,213,100,5
50 SOUND 1,239,100,5
60 SOUND 2,119,100,7
70 SOUND 2,127,100,7
80 SOUND 2,159,200,7
```

Here all the channel A sounds come before the channel B sounds in the listing.

The micro works its way through first four notes, playing one and putting he other three on the channel A queue.

Then it comes to the three SOUND commands for channel B. It processes these, playing the first and popping the other on to the queue for channel B.

This all happens so quickly that we hear the notes on channels A and B start at the same time, even though the statements that produce the notes are four lines apart.

To be really accurate, the Amstrad 40 and 50, putting these notes on the queue before it reaches the first channel B note and plays it.

This means that the notes don't start at exactly the same time. Having said that, I can't hear the difference and I doubt if anyone else can.

Previously we've had an example of how too many SOUNDs can mess up a Basic program, causing it to pause.

Program VII shows how the Amstrad getting on with Basic can mess up the pattern of SOUND commands that go together to make up tunes.

```
10 REM PROGRAM VII
20 SOUND 1,239,100,5
30 SOUND 1,190,100,5
40 FOR delay=1 TO 200: NEXT
   delay
50 SOUND 1,213,100,5
60 SOUND 1,239, 100,5
70 SOUND 2,119,100,7
80 SOUND 2,127,100,7
90 SOUND 159,200,7
```

You'll see that apart from line 40 this is exactly the same as the previous program. Yet the tune is destroyed.

What's happened is that the micro has processed lines 20 and 30, playing one note and popping the next on the channel A queue.

Then it hits the delay loop and hurtles round 40 times doing nothing but taking up time.

Once the loop is finished, the program goes on to lines 50 and 60. popping these notes on the queue.

Finally the micro reaches line 70 and starts playing the first note on channel B, putting the next two on the channel B queue.

The problem arises because the first note on channel A has been playing for some time before the first note on channel B is started.

Previously they began at the same time (or as near as made no difference). Now, however , the delay loop has slowed down the program so much that by the time the micro reaches line 70 everything is out of step.

Try changing the length of the delay loop to see the effect.

You might say: "Well don't stick delay loops in your tunes and everything will be all right". It's a fair point, but remember we can only have five notes at a time on a channel without causing problems.

However, any decent tune has a lot more than five notes so if we want musical accompaniment to our programs we have to grab five notes, do something else, then grab other notes as space appears in the queues.

The delay loop in Program VII symbolizes these "something else's".

As you can see, while the program is doing these other things, tunes can get out of step. Don't worry, though, we're stuck to a repertoire of five-note ditties.

Program VIII shows how to overcome the problem.

As you can hear, the tune is intact,

```
10 REM PROGRAM VIII
20 SOUND 17,239,100,5
30 SOUND 17,190,100,5
40 FOR delay=1 TO 200:NEXT
   delay
50 SOUND 17,213,100,5
60 SOUND 1,239,100,5
70 SOUND 10,119,100,7
80 SOUND 10,127,100,7
90 SOUND 10,159,200,7
```

It takes two to make a rendezvous, and both have to recognize each other despite the delay loop. The secret lies in the funny-looking channel parameters.

Remember how we can add together our channel numbers to form new, more powerful channel parameters?

Well we can use the same sort of technique to ensure that notes in different channels are playing in step. We just add 8, 16 or 32 to the appropriate channel parameter as needed.

Suppose we wanted a note on channel A to be played starting at exactly the same time as a note of a different pitch on channel B.

What we do is to cause the notes to "rendezvous" with each other by adding the relevant number to the channel parameter.

To get a note on channel A to coincide with a note on channel B, so we add 16 and 1 to get a new parameter 17.

So:

```
SOUND 17,100,100,5
```

produces a note on channel A that will only play when there's a note on channel B.

That seems simple enough, so, if you haven't already, enter:

```
SOUND 17,100,100,5
```

and listen to what happens.

Nothing should happen, as adding 8 to the parameter has told the micro to wait for a note on channel B and there isn't one, yet. So let's give it a note on channel B with:

```
SOUND 2,4000,100,5
```

You'll probably be disappointed that all you hear is the low note on channel B. The high note waiting on channel A hasn't put in an appearance, even though we've given it a note on channel B. Has something gone wrong?

The answer is that it's our fault. We've added 16 to the first note's channel parameter to tell it to wait for a note on channel B but we haven't marked out which note on channel B.

It takes two to make a rendezvous and both have to recognize each other.

Not only must the first note have a number added to its channel parameter, the second one has to as well.

It's rather like two people meeting for the first time. One will have a bowler hat on, the other will be carrying a copy of The Times.

In the example above, the note on channel A was wearing its bowler hat (17) but the note on channel B wasn't carrying its copy of The Times. As a result, they passed each other by.

What we should have done was add 8 to the channel B note's parameter. If my maths is correct 2+8 gives 10 so:

```
SOUND 10,4000,100,5
```

should not only produce a low note but also bring the high note on channel A out of hiding.

Try it and see.

To sum up, two notes can be made to start at the same time or rendezvous by adding the appropriate values to their channel parameters.

To get a note to wait for one on channel A before we start, we add 8.

For a rendezvous with channel B we use 16 while a date with channel C has us adding 32.

Table II summarizes the situation.

Addition	Rendezvous with
8	Channel A
16	Channel B
32	Channel C

Using this, we can have a note on channel A coinciding with one on channel C using:

```
SOUND 33,600,100,7
```

and:

```
SOUND 34,3500,100,7
```

Similarly, to link notes on B and C we use:

```
SOUND 34,12,100,7
```

with:

```
SOUND 20,3750,100,7
```

Using your new-found knowledge, you should now be able to see how the 17's and 10's of Program VIII work to overcome the delay loop.

The note on channel A can't start playing until it comes across another marked for rendezvous on channel B. This means that the notes stay in step.

And that's it for this month. We'll be exploring the channel parameter in more detail next time.

Until then, play around with the values given in Tables I and II and see how they work in practice.

You may find that it helps to have set up the small Enter key with:

```
KEY 139, "SOUND 135,0,0,0,0" + CHR$(13)
```

Now when the sound channels get out of hand just press the small Enter and all the garbage will be cleared. I'll explain why in a future issue.

And, when you're more familiar with rendezvous, perhaps you'll be able to see why Program VIII suffers a bit from "overkill". Have fun.

SCROLLER

Get things moving with these routines by TERENCE BRATLEY

SCROLLER is a collection of 10 machine code routines for CPC Amstrads, two of which scroll a message on to the screen from either left or right while the other eight can be used to scroll off any defined section of the screen horizontally, vertically or diagonally.

The screen scrolling routines can be used whenever you wish to remove part or all of the screen in a novel way instead of simply removing it by a CLS command directed to a particular window or the whole screen. The main use will be in games programs to give special effects.

The routines scroll vertically one screen row at a time and horizontally in steps of one screen byte - the screen is 80 bytes wide regardless of the current screen mode.

This does not give the highest resolution possible, but it is much faster and perfectly adequate. As the bytes are scrolled off, the background color can be changed to any color you wish or will be replaced with PEN 0 by default.

To make it easy to use, these routines in a Basic program are in the form of RSX commands, so no knowledge of machine code is required. However, for those who would like to understand how they work - and possibly adapt them - there is an assembly listing of the machine code in Program II, produced using Maxam.

There are only four actual directions in which you can scroll the screen memory - up, down, left and right - but by calling, for example, up then right one after the other in a loop the screen memory will appear to scroll diagonally.

The assembly listing consists of three main sections.

The first sets up and logs on the new RSX commands.

In the second there are eight routines which are entered with the IX register pointing to the following parameters:

IX+0 column
IX+2 row
IX+4 width
IX+6 height
IX+8 pen

The A register holds the number of parameters on entry - 4 or 5. The first thing the routines do is call SETUP which uses the parameters to find screen addresses and the new background color. If there are only four parameters the new background color is set to PEN 0 by default.

For the diagonal scrolls the height and width of the defined box are compared and the lower of these two values is used as a loop counter to perform the correct number of scrolls to clear the box.

All the RSXs call the required scroll routines repeatedly using the width value for horizontal scrolls and the height value for vertical scrolls.

In the third section four scroll routines are set up which are called by the RSXs to do a single scroll in one of the four main directions. The horizontal scroll routines do a one byte scroll left or right and the vertical ones do a one row scroll up or down. You can call them as many times as required to clear the defined box.

The RSX commands and their associated parameters are:

|NORTH, |SOUTH, |EAST,
|WEST, |NEAST, |NWEST,
|SEAST, |SWEST

followed by p, h, w, r, c where:

p is the pen color to replace the old background color - optional,

h is the height of the box to scroll in rows - 1 to 25,

w is the width of the box to scroll in screen bytes - 1 to 80 as in Mode 2,

r is the top row number of the box - 1 to 25 and

c is the top column of the box where columns correspond to the number of screen bytes across the screen - 1 to 80 as in Mode 2.

To scroll the whole screen in a northwesterly direction the command would be |NWEST,0,25,80,1,1 which would leave the background as PEN 0.

It is important to note here that the height and width must always be at least one, as a box must have two dimensions and care should be taken that the defined box does not exceed the screen limits. Also you can omit the first parameter p and the new background will be PEN 0 by default.

The two routines to scroll any message you choose on to the screen are also in the form of RSX commands. If anything they are easier to use than the previous eight, as they only require two parameters.

The RSX commands are |MESLFT which scrolls the message from right to left , and |MESRIT which scrolls the message from left to right.

Before they are called, a string variable must be set up which contains the message to be scrolled. For simplicity we'll call it a$. This is the first parameter to follow the RSX command.

The second parameter required is the row number where you wish the message to be scrolled - from 1 to 25, top to bottom of the screen. Program I shows how to use these two commands.

```
10 a$="Computing with the
   Amstrad"
20 |MESLFT,@a$,1
30 a$="1234567890"
40 |MESRIT,@a$,25
```

Note how the string is defined immediately before the command and that the string parameter after the RSX command is preceded by a @ to tell the operating system to pass the address of the variable.

Going back to the assembly listing again - Program III - these routines use LFTSCL and RITSCL to scroll the line left or right and the routine FNDSTR finds the address of the string descriptor. The string descriptor consists of three bytes - byte 0 - length of the string, and bytes 1 and 2 - address where the string is stored.

The contents of byte 0 is used to print the correct number of characters in the string and the DE register is used as a pointer to the string so that the characters can be printed by the TXT OUTPUT routine at &BB5A.

Scrolling a message on entails three steps. Firstly print a character at the edge of the screen, then scroll the line by the width of one character and finally find the next character to print and go back to the first step.

This sequence is performed by RITMES and LFTMES in the assembly listing. There is also a pause routine to slow things down a bit.

While the message is scrolling you cannot input a keystroke to perform some new action, but a way around this is to define another string holding the message, take one character from it at a time and put it into a$ so that a$ is only one character long. Now the keyboard can be read after every character is scrolled.

This leads to jerky scrolling because of the delay caused by returning to Basic between characters. It would be better to test for a keypress within the machine code at RITMES or LFTMES.

Now we'll get down to entering the machine code. Type in Program III and run it. When you are sure you have made no errors save the code as:

SAVE "scroller.bin",b,40000,800

To use it in your own programs first set HIMEM to 39999 and load the code at 40000 as it is not relocatable. Then to set up the RSX commands simply CALL 40000 - you can now use the extra commands.

The demonstration program - Program IV - will give you an idea of what these routines can do. After that you're only limited by your own imagination.

Program II

```
ORG 40000:nolist

.SETRSX
LD BC,RSXTAB:LD HL,WORK:JP #BCD1
WORK:DEFS 4
.RSXTAB:DEFW NAMTAB
JP NORTH:JP SOUTH:JP EAST:JP WEST
JP NEAST:JP SEAST:JP SWEST:JP NWEST
JP MESLFT:JP MESRIT
.NAMTAB
DEFM "NORT":DEFB "H"+#80
DEFM "SOUT":DEFB "H"+#80
DEFM "EAS":DEFB "T"+#80
DEFM "WES":DEFB "T"+#80
DEFM "NEAS":DEFB "T"+#80
DEFM "SEAS":DEFB "T"+#80
DEFM "SWES":DEFB "T"+#80
DEFM "NWES":DEFB "T"+#80
DEFM "MESLF":DEFB "T"+#80
DEFM "MESRI":DEFB "T"+#80
DEFB 0

NORTH:CALL SETUP
LD A,(HEIGHT):LD B,A
.NLP:PUSH BC:CALL UPSCL:POP BC
DJNZ NLP:RET
SOUTH:CALL SETUP
LD A,(HEIGHT):LD B,A
.SLP:PUSH BC:CALL DWNSCL:POP BC
DJNZ SLP:RET
EAST:CALL SETUP
LD A,(WIDTH):LD B,A
.ELP:PUSH BC:CALL RITSCL:POP BC
DJNZ ELP:RET
WEST:CALL SETUP
```

```
LD A,(WIDTH):LD B,A
.WLP:PUSH BC:CALL LFTSCL:POP BC
DJNZ WLP:RET
NEAST:CALL SETUP:CALL COMPAR
.NELP:PUSH BC
CALL UPSCL:CALL RITSCL
POP BC:DJNZ NELP:RET
SEAST:CALL SETUP:CALL COMPAR
.SELP:PUSH BC
CALL DWNSCL:CALL RITSCL
POP BC:DJNZ SELP:RET
SWEST:CALL SETUP:CALL COMPAR
.SWLP:PUSH BC
CALL DWNSCL:CALL LFTSCL
POP BC:DJNZ SWLP:RET
NWEST:CALL SETUP:CALL COMPAR
.NWLP:PUSH BC
CALL UPSCL:CALL LFTSCL
POP BC:DJNZ NWLP:RET
COMPAR:LD A,(HEIGHT):LD B,A
LD A,(WIDTH):CP B:RET NC
LD B,A:RET
let LFTSCL=$
CALL #BD19
LD HL,(TOPLFT)
LD A,(HEIGHT):LD B,A
LFTLP1:PUSH BC  ;rows
LD B,8          ;pixel lines
.LFTLP2
PUSH BC:PUSH HL:PUSH HL:POP DE
INC HL
LD A,(WIDTH):DEC A:LD C,A
LD B,0
LDIR
DEC HL
LD A,(NEWCOL):LD (HL),A
POP HL
LD A,H:ADD A,8:LD H,A
POP BC
DJNZ LFTLP2
LD BC,#C050:ADD HL,BC
POP BC:DJNZ LFTLP1:RET
let RITSCL=$
CALL #BD19
LD HL,(TOPRIT)
LD A,(HEIGHT):LD B,A
RITLP1:PUSH BC
LD B,8
.RITLP2
PUSH BC:PUSH HL
PUSH HL:POP DE
DEC HL
LD A,(WIDTH):DEC A:LD C,A:LD B,0
LDDR
INC HL
LD A,(NEWCOL):LD (HL),A
POP HL:LD A,H:ADD A,8:LD H,A
POP BC
DJNZ RITLP2
LD BC,#C050:ADD HL,BC
POP BC:DJNZ RITLP1:RET
```

```
let UPSCL=$
CALL #BD19
LD HL,(TOPLFT)
LD A,(HEIGHT):DEC A:LD B,A
.UPLP1
PUSH BC
LD B,8
.UPLP2
PUSH BC:PUSH HL
PUSH HL:POP DE
LD BC,#50:ADD HL,BC
LD A,(WIDTH):LD C,A:LD B,0
LDIR
POP HL:LD DE,2048:ADD HL,DE
POP BC:DJNZ UPLP2
LD DE,#C050:ADD HL,DE
POP BC:DJNZ UPLP1:CALL CLRLIN:RET
let DWNSCL=$
CALL #BD19
LD HL,(BOTLFT)
LD A,(HEIGHT):DEC A:LD B,A
.DWNLP1:PUSH BC
LD B,8
.DWNLP2:PUSH BC
PUSH HL:PUSH HL
LD DE,#50:ADD HL,DE:EX DE,HL
POP HL
LD A,(WIDTH):LD C,A:LD B,0
LDIR
POP HL:LD DE,2048:ADD HL,DE
POP BC:DJNZ DWNLP2
LD DE,#BFB0:ADD HL,DE
POP BC:DJNZ DWNLP1
LD DE,#50:ADD HL,DE
CALL CLRLIN:RET
CLRLIN:LD B,8
CLRLP:PUSH BC:PUSH HL
PUSH HL:POP DE:INC DE
LD A,(NEWCOL):LD (HL),A
LD A,(WIDTH):DEC A:LD C,A:LD B,0
LDIR
POP HL:LD DE,2048:ADD HL,DE
POP BC:DJNZ CLRLP:RET
FNDPOS:PUSH DE
LD HL,#C000-#50
LD DE,#50
.FNDLP1:ADD HL,DE:DJNZ FNDLP1
DEC C:LD E,C:LD D,0:ADD HL,DE
POP DE:RET
.SETUP:CP 5:JR Z,ENCINK
XOR A:JR SETCOL
.ENCINK:LD A,(IX+8):CALL #BC2C
.SETCOL:LD (NEWCOL),A
.GETTOP:LD B,(IX+2)
LD C,(IX+0):PUSH BC
CALL FNDPOS:LD (TOPLFT),HL
POP BC
RITADD:PUSH BC
LD A,(IX+4):LD (WIDTH),A
DEC A:ADD A,C:LD C,A
CALL FNDPOS:LD (TOPRIT),HL
```

```
BOTADD:POP BC
LD A,(IX+6):LD (HEIGHT),A
DEC A:DEC A:ADD A,B:LD B,A
CALL FNDPOS:LD (BOTLFT),HL
RET
TOPLFT:DEFW 0
TOPRIT:DEFW 0
BOTLFT:DEFW 0
HEIGHT:DEFB 0
.WIDTH:DEFB 0
NEWCOL:DEFB 0
.BYTES: DEFB 0
FNDSTR:LD L,(IX+2)
LD H,(IX+3):LD C,(HL):LD B,0
INC HL:LD E,(HL):INC HL:LD D,(HL)
RET
.SETVAL
XOR A:LD (NEWCOL),A
INC A:LD (HEIGHT),A
LD A,80:LD (WIDTH),A
RET
MESLFT:CALL SETVAL
LD H,0:LD L,(IX+0):DEC L
CALL #BC1A:LD (TOPLFT),HL
LD A,B:LD (BYTES),A
CALL #BC11:AND A:JR Z,LMODE0
CP 1:JR Z,LMODE1
LD A,80:JR LFTWID
LMODE1:LD A,40:JR LFTWID
LMODE0:LD A,20
.LFTWID
CALL FNDSTR
LD H,A:LD L,(IX+0):JP LFTMES
.MESRIT
CALL SETVAL
CALL #BC11:AND A:JR Z,RMODE0
CP 1:JR Z,RMODE1
LD A,79:JR RITWID
RMODE1:LD A,39:JR RITWID
RMODE0:LD A,19
.RITWID
LD H,A:LD L,(IX+0):DEC L
CALL #BC1A:LD (TOPRIT),HL
LD A,B:LD (BYTES),A
CALL FNDSTR
EX DE,HL:ADD HL,BC:DEC HL:EX DE,HL
LD L,(IX+0):LD H,1:JP RITMES
.LFTMES
PUSH BC:PUSH DE:PUSH HL
CALL #BB75:LD A,(DE):CALL #BB5A
LD HL,(TOPLFT)
PUSH BC
LD A,(BYTES):LD B,A
BYTLP1
PUSH BC:PUSH HL
CALL LFTSCL:CALL PAUSE
POP HL:POP BC
DJNZ BYTLP1
POP BC:POP HL:POP DE
INC DE
POP BC:DEC BC
```

```
LD A,B:OR C:JR NZ,LFTMES
RET
RITMES:PUSH BC:PUSH DE:PUSH HL
CALL #BB75:LD A,(DE):CALL #BB5A
LD HL,(TOPRIT)
PUSH BC
LD A,(BYTES):LD B,A
BYTLP2
PUSH BC:PUSH HL
CALL RITSCL:CALL PAUSE
POP HL:POP BC
DJNZ BYTLP2
POP BC:POP HL:POP DE
DEC DE
POP BC:DEC BC
LD A,B:OR C:JR NZ,RITMES
RET
.PAUSE:PUSH BC
CALL #BC11:AND A:JR Z,MOD0
CP 1:JR Z,MOD1
LD BC,1500:JR WAIT
MOD1:LD BC,1500:JR WAIT
MOD0:LD BC,1000
WAIT:DEC BC:LD A,B:OR C:JR NZ,WAIT
POP BC
RET
```

Program III

```
10 REM Program III - CPC only
20 MEMORY 39999
30 address=40000
40 FOR a=0 TO 79
50 sum=0:READ code$,sum$
60 FOR b=1 TO 19 STEP 2
70 n=VAL("&"+MID$(code$,b,2))
80 POKE address,n
90 sum=sum+n:address=address+1
100 NEXT
110 IF sum<>VAL("&"+sum$) THEN PRINT
"Data error in line ";130+a*10
120 NEXT
130 DATA 014D9C21499CC3D1BCFC,53C
140 DATA A64D9C6D9CC3A09CC3AF,609
150 DATA 9CC3BE9CC3CD9CC3DC9C,720
160 DATA C3ED9CC3FE9CC30F9DC3,6DB
170 DATA 869EC3B79E4E4F5254C8,547
180 DATA 534F5554C8454153D457,417
190 DATA 4553D44E454153D45345,3FF
200 DATA 4153D453574553D44E57,423
210 DATA 4553D44D45534C46D44D,404
220 DATA 45535249D400CD209E3A,3CC
230 DATA 669E47C5CD8B9DC110F9,5CF
240 DATA C9CD209E3A669E47C5CD,56B
250 DATA BC9DC110F9C9CD209E3A,5B1
260 DATA 679E47C5CD5B9DC110F9,5A0
270 DATA C9CD209E3A679E47C5CD,56C
280 DATA 2B9DC110F9C9CD209ECD,5B3
290 DATA 209DC5CD8B9DCD5B9DC1,5FD
300 DATA 10F6C9CD209ECD209DC5,5A9
310 DATA CDBC9DCD5B9DC110F6C9,67B
320 DATA CD209ECD209DC5CDBC9D,600
330 DATA CD2B9DC110F6C9CD209E,5B0
340 DATA CD209DC5CD8B9DCD2B9D,5D9
350 DATA C110F6C93A669E473A67,4B6
360 DATA 9EB8D047C9CD19BD2A60,563
370 DATA 9E3A669E47C50608C5E5,4A0
380 DATA E5D1233A679E3D4F0600,3AA
390 DATA EDB02B3A689E77E17CC6,5A2
400 DATA 0867C110E50150C009C1,400
410 DATA 10DBC9CD19BD2A629E3A,4BB
420 DATA 669E47C50608C5E5E5D1,57E
430 DATA 2B3A679E3D4F0600EDB8,3A1
440 DATA 233A689E77E17CC60867,46C
450 DATA C110E50150C009C110DB,47C
460 DATA C9CD19BD2A609E3A669E,4D2
470 DATA 3D47C50608C5E5E5D101,4B8
480 DATA 5000093A679E4F0600ED,2DA
490 DATA B0E111000819C110E811,38D
500 DATA 50C019C110DECDF29DC9,5FD
510 DATA CD19BD2A649E3A669E3D,44A
520 DATA 47C50608C5E5E5115000,40A
530 DATA 19EBE13A679E4F0600ED,466
540 DATA B0E111000819C110E711,38C
550 DATA B0BF19C110DD11500019,3B0
560 DATA CDF29DC90608C5E5E5D1,693
570 DATA 133A689E773A679E3D4F,395
580 DATA 0600EDB0E111000819C1,377
590 DATA 10E6C9D521B0BF115000,485
600 DATA 1910FD0D59160019D1C9,355
610 DATA FE052803AF1806DD7E08,35E
620 DATA CD2CBC32689EDD4602DD,4EF
630 DATA 4E00C5CD0F9E22609EC1,46E
640 DATA C5DD7E0432679E3D814F,468
650 DATA CD0F9E22629EC1DD7E06,4BE
660 DATA 32669E3D3D8047CD0F9E,3F1
670 DATA 22649EC9D0C227C080C2,5A8
680 DATA 01500004DD6E02DD6603,2E8
690 DATA 4E0600235E2356C9AF32,2F8
700 DATA 689E3C32669E3E503267,39F
710 DATA 9EC9CD789E2600DD6E00,4BB
720 DATA 2DCD1ABC22609E783269,403
730 DATA 9ECD11BCA7280CFE0128,43A
740 DATA 043E5018063E2818023E,16E
750 DATA 14CD6A9E67DD6E00C3EC,54A
760 DATA 9ECD789ECD11BCA7280C,4F6
770 DATA FE0128043E4F18063E27,23B
780 DATA 18023E1367DD6E002DCD,317
790 DATA 1ABC22629E7832699ECD,476
800 DATA 6A9EEB092BEBDD6E0026,483
810 DATA 01C3159FC5D5E5CD75BB,5F4
820 DATA 1ACD5ABB2A609EC53A69,48C
830 DATA 9E47C5E5CD2B9DCD3E9F,5CE
840 DATA E1C110F4C1E1D113C10B,5F8
850 DATA 78B120D8C9C5D5E5CD75,6AB
860 DATA BB1ACD5ABB2A629EC53A,4E0
870 DATA 699E47C5E5CD5B9DCD3E,5C8
880 DATA 9FE1C110F4C1E1D11BC1,694
890 DATA 0B78B120D8C9C5CD11BC,554
900 DATA A7280EFE01280501DC05,2EB
910 DATA 180801DC05180301E803,209
920 DATA 0B78B120FBC1C9000000,3D9
```

Program IV

```
10 REM Program IV - CPC only
20 REM Scroller Demonstration
30 MEMORY 39999
40 LOAD "SCROLLER.BIN",40000
45 CALL 40000
50 MODE 0
60 a$="SCROLLER"+SPACE$(5)
70 |MESLFT,@a$,2
80 a$=SPACE$(8)+"by"
90 |MESRIT,@a$,4
100 a$="T.A.Bratley"+SPACE$(4)
110 |MESLFT,@a$,6
120 a$=SPACE$(2)+"Copyright 1986"
130 |MESRIT,@a$,8
140 a$="Presented by"+SPACE$(3)
150 |MESLFT,@a$,10
160 a$=SPACE$(4)+"COMPUTING"
170 |MESRIT,@a$,12
180 a$="WITH THE"+SPACE$(6)
190 |MESLFT,@a$,14
200 a$=SPACE$(5)+"AMSTRAD"
210 |MESRIT,@a$,16
220 a$="Press any Key"+SPACE$(3)
230 |MESLFT,@a$,18
240 a$=SPACE$(4)+"For Demo"
250 |MESRIT,@a$,20
260 WHILE INKEY$="":WEND
270 |NWEST,6,10,40,1,1
280 |SWEST,11,10,40,1,40
290 |SOUTH,1,10,20,11,1
300 |NORTH,2,10,20,11,20
310 |EAST,10,5,40,11,40
320 |WEST,12,5,40,16,40
330 |NWEST,0,25,40,1,1
340 |NEAST,0,25,40,1,40
350 a$="DEMO OVER"+SPACE$(5)
360 |MESLFT,@a$,10
370 a$=" Any key to repeat "
380 |MESLFT,@a$,20
390 WHILE INKEY$="":WEND
```

An extra facility for Word Count

HERE are some light modifications I have made to John Milsom's useful Word Count program from the December 1986 issue of Computing with the Amstrad.

I decided to alter it in order to provide a running word count as found in several word processors I have used.

Add the following to the end of line 210:

```
:CPOS$=esc$+CHR$(B9)+
CHR$(32)+CHR$(108)
```

and the following to the end of line 2160:

```
:PRINT CPOS$;:PRINT USING
"####";wc%
```

This will provide a running word count in the top righthand corner of the screen.

The position can be varied by altering the figures 32 and 108 in line 210 - see page 58 of the CP/M+ Operating System manual supplied with the PCW.

Also by adding a semicolon to the end of line 550 the "Words ... being counted" message will appear on the top line of the screen.

The modification does slow things down a little.

I timed a count of the READ.ME file from the LocoScript disc as one minute 42 seconds without the changes, and one minute 59 seconds with them - 17 seconds on a 7Kb file when converting to Ascii.

By the way, I found that the feature to hide the program files from the selection list did not work.

This is because the FIND$ function always returns a 12 character string.

The filenames contained within the brackets in line 220 must be padded out with spaces in the form BASIC.COM and WCOUNT .BAS in order for the function to work. - **Richard Gagen**

● *We're sure our readers will find this extra facility useful. Regarding the FIND$ function you're quite right.*

Somewhere during processing the spaces were stripped out from the filenames and this will have the effect you describe. Congratulations on being so observant.

Still counting

THANKS for the Basic program in your December 1986 issue that counts the number of words in a LocoScript document.

Here is an alternative method using LocoScript itself to count the words. Although extremely slow it does the job and can be left running on its own without further input.

First copy your document to drive M, so that access is faster and you do not accidentally corrupt your original document.

Now edit the header of this copy and set the number of lines in the body of the page to 50 - this is to make counting easier.

Make sure the page number starts at 1. Use the exchange key to change all carriage return effectors in the document to a space and get rid of any codes that force a page break.

Now for the all important Exchange command. Change all single spaces to a Return. You will see LocoScript putting each word - together with any punctuation - on a new line.

When it has finished position the cursor on the last line containing a word and make a note of the page and line number that the cursor is on.

The number of words works out as (PAGE NO.-1) * 50 + LINE NO.

Whatever method of word counting is used I recommend keeping a record of the number of words, together with the date and time, on line 3 of the LocoScript Identify Text option.

If you then make a minor amendment to the document you can adjust this total and avoid a recount. - Peter Hayden

● *It works! It's amazing what human ingenuity can achieve - no wonder we got to the moon.*

Still it may be better to wait for a proper add-on, which could well be quicker than counting an 80k file.

ALL programs printed in this magazine are exact reproductions of listings taken from running programs which have been thoroughly tested.

However, on the rare occasions when mistakes occur corrections are published as a matter of urgency. Should you encounter error messages when you type in a program, they will almost certainly be the result of your own typing mistakes.

Unfortunately we can no longer answer personal programming queries concerning these mistakes. Of course letters about suggested errors will be investigated without delay, but any replies found necessary will only appear in the mail pages.

Tape files to disk

About a year ago I bought a cassette version of the original Mini Office.

I put it to good use, but tired of the tedious loading time. So, having fitted a disk drive to my 464, I invested in Mini Office II.

I have tried using the Convert program supplied to get my Mini Office tape files into the disk version without success. Is it possible?

I must congratulate you on the Mini Office II package. It is all you claim it to be and superb value.

In particular I find the spreadsheet program versatile and easy to use.

I have easily adapted it to a

simple cash book with which I can pit my CPC's wits against those of my bank's computer. - **C. A. Watts**

● *The Convert program on Mini Office II is designed to make files created by the original Mini Office compatible with the new version. However the Mini Office II disk is looking for a disk file and you will first have to buy a utility that will transfer your tape files to disk.*

Cure for Da Bells

I RECENTLY sent off to you for some cassettes which correspond to your magazines. One of them, Da Bells, seems to be faulty. It loads in for play, then after 10 minutes the screen prints "Memory full in 440"or a variety of different line numbers.

Then the game has to be loaded again, losing all high scores gained during play. Is my cassette faulty? - **C. Cloud**

● *No it's not supposed to happen and the following lines will cure it. Just make the necessary alterations and resave the game in its corrected form. Not all versions of this game suffer from the problem you mention so check if the version you have on tape has already had these corrections made.*

```
180 GOTO 350
390 IF INKEY(47)=0 THEN 560
470 GOTO 190
730 GOTO 420
```

Problem wrapped up

WHILE writing an adventure game I came across the need to find an easy way of preventing words wrapping round on to the next line of the display.

My solution was to write a program to display the text word by word, checking beforehand whether each word would fit on to the line or not. The following is the finished subroutine and how to use it.

You must define the string you want to be printed as rd$, send the stream number in d and then call the subroutine. The variable c is the maximum number of characters per line which must be changed for each Mode. - **Stephen D. Wardell**

```
10 REM String Formatter
20 REM by Stephen Wardell
30 d=0
40 rd$="This test shows how the
word wrap subroutine works!"
50 GOSUB 5000
60 END
70 :
5000 REM
5010  a=0:b=0:a$="",b$="":c=40
5020 FOR a=1 TO (LEN(rd$)+1)
5030 b$=b$+a$
5040 IF a$=CHR$(46) THEN
GOSUB 5110
5050 IF a$=CHR$(32) THEN
GOSUB 5110
5060 a$=MID$(rd$,a,1)
5070 NEXT a
5080 PRINT #d,b$
5090 b$="":b=0
5100 RETURN
5110 REM
5120 IF b+LEN(b$)<=c THEN
PRINT #d,b$;:ELSE PRINT
#d,"":PRINT #d,b$;
5130 b=b+LEN(b$)
5140 b$+"":b=0
5150 RETURN
```

New routine

I HAVE discovered a firmware routine which is not documented in the 464 firmware guide.

It is the string input routine as used by Basic and the entry in the jumpblock after the JUMPER (&BD37).

The address of the routine is at &BD3A.

Entry conditions. HL contains address of 255 byte buffer.

Exit conditions. If user pressed Enter, Carry is true, A contains the last character typed and H contains the buffer address.

If user pressed Escape, Carry is false, A contains the last character typed and HL contains the buffer address.

All other registers are preserved. This routine is useful as the Del, CLR and cursor keys perform their correct operations.

If there are any characters in the buffer before the routine is called they will be printed up, unless the first character is a null. This may prove useful for editing purposes. - **Matthew Western**

Exposed negatives

WHEN I ask my 6128 to print a large hex number it gives a wrong negative answer.

I know how to allow for this, but is it safe to use a line such as:

```
FOR J=&A000 TO &A00F
```

or would the micro take these values as being negative?

Also is there any byte saving technique which can be used to shorten programs in Basic?

I have found that if a value is used several times, then by using a variable for the value the stored lines of Basic are shorter but the length of the listing may be increased.

For instance if the value 9 is used a lot use A=9, then in the listing whenever 9 is required use the variable A. - **H.P. Walters**

● *All the Amstrads use 16 bit signed integers, so the largest possible integer is 32767 or &7FFF.*

Any number greater than this is considered negative - &FFFF is -1, &FFFE is -2 and so on. &A000 is -24576, but you can still use these large numbers in the way you asked.

In answer to your second point the only efficient way to keep programs shorts is to use short variable names and multi-statement lines.

Space oddity

I WRITE with regard to your news item "Beyond boldly goes" in the October 1986 issue of Computing with the Amstrad, which refers to the forthcoming Star Trek game from Beyond Software.

It states "Players take on the roles of the seven leading protagonists, including Captain Kirk, Dr. Spock and Scotty."

Star Fleet records show that the well known baby doctor Spock is not among our staff.

However, we do have a Vulcan science officer serving aboard who answers to the name of Mr. Spock. I'm sure he will welcome this new game and the interest it will generate in not only his chose speciality - computers - but also Star Fleet itself, with its long and honored traditions. - **Admiral Mark Slade**

● *What an embarrassing misprint! Still it shows that our editorial staff are prepared to 'boldly go where no man has gone before'. Beam us up Spotty!*

Spider's Joy!

THIS year I decided to upgrade my home micro system to the Amstrad CPC6128 and I took out an annual subscription to Computing with the Amstrad.

We, my wife and daughter aged 10, avidly check each month to see what games are offered for typing in and so far have working copies of the dozen or so available this year. They type them in and I correct their typing mistakes. We all play them.

The offering Spider's Web soon joined the rest hence my letter. Wouldn't it be nice if joystick control were available?

In Basic the INKEY keyword works with a variable argument as well as the integers used in the program and that is the theory I used for the following changes:

```
860 IF INKEY (kf)<>0.................
870 IF INKEY (kr)=0...................
880 IF INKEY (kl)=0 ..................
890 IF INKEY (ku)=0 ..................
900 IF INKEY (kd)=0 ..................
```

```
2570 PRINT:PEN 1:PRINT"Press
joystick 'fire' or spacebar"
2580 IF INKEY (66)=0 THEN END
ELSE a$=INKEY$:IF a$=""
THEN 2580
2585 kf=47: kr=63: kl=71: ku=28:
kd=30: IF ASC(a$)=88 THEN
kf=76: kr=75: kl=74: ku=72: kd=73:
RETURN ELSE RETURN
```

and in lines 1080, 1100, 1200, 1220, 1300, 1320, 1410 and 1430 replace all instances of:

```
IF INKEY(47)=0
```

with:

```
IF INKEY(kf)=0
```

Finally a question. Most games written in Basic with or without "go faster" machine code additions also seem to need various calls to firmware to keep things in order. For example Spider's Web line 170 uses CALL&BB03. I have a firmware book which says this - KM RESET - restores the keyboard system to a standard state. Are these calls really necessary? - **Brian Windley**

● *This is just one of many letters we've had from people about our published listings. It's very gratifying to hear from anyone who will vouch for them. An awful lot of work goes into ensuring that they will work if typed in correctly by our readers. Brian seems to have found an ideal system involving the lady members of his family. Perhaps it's the feminine touch at the keyboard that's the answer.*

With regard to your question about the calls - yes these are necessary. The call to &BB03 clears the keyboard buffer of the garbage which tends to build up when playing games.

Checking data hints

I HAVE typed in a lot of your games and utilities and am delighted to report that they all run after hours of lengthy debugging sessions of my own typing errors.

In achieving this I found that most of my mistakes were made in the numerous lines of data statements in some of your recent games.

I found these errors by carefully checking against the magazine listing - every individual entry from a cluttered screen.

I'm sure everyone agrees that this is a tedious method just to find possibly one simple typing error, but it's necessary when typing in long listings.

Are there any simple hints that can make the job of checking out these data lines a little easier?

I realize that in some listings writers include checksum routines to save checking time, but it's the final search that I find so tedious. Surely there's a simpler way. - **Gwynn Jones**

● *We agree that checking data statements can be a tedious business if you are going to look at every entry to find a missing number, even if you know the area of code to search.*

The following hints might make life a little easier for you, but of course success will depend on the type of error originally reported.

To find missing data try defining a window the same width as the magazine listing - (say) 37 - and half the depth of the screen using:

```
WINDOW #2,1,37,1,12
```

Now you can list the offending lines using:

```
CLS #2:LIST #2
```

and all you need to check is the last character of each line.

This trick is useful for searching for a missing items as the end character of a particular line will be different if an item is missing.

You could define the remaining half of the screen as another window using:

```
WINDOW #1,1,40,13,25
```

and use this for normal listing purposes using:

```
CLS #1,LIST #1
```

to look at other lines at the same time.

Of course an apparent missing item of data could be the result of typing in a period instead of a comma between the characters.

A mistake such as 26.01 in stead of 26,01 will result in a READ of one number, but will not show using the hint above because it takes up the same amount of room on a line.

To find this particular error, and others, you'll have to go a stage further and print the data to the screen as the program is reading it.

Then you'll either see that offending code shouting at you from the screen, or at least see which piece of data was last read correctly before the error message appears.

Using August's Diamond Digger as an example we'll assume you had an 'out of data' error at line 2160 which reads:

```
2160 check=0:FOR f=0 TO
 135:READ x$:POKE &9C40+f,
 VAL("&"+x$) etc
```

Try editing this to include the following extra code between READ x$: and POKE &9C40......

```
PRINT x$: WHILE INKEY$="":
WEND:
```

It will print the first item of data on the screen and wait for you to press a key before moving on to the next and printing each successive item in a column down the screen.

The typing error mentioned earlier will easily stand out as it scrolls up the screen as a four digit decimal number in a column of two digit numbers.

In addition to this error wrong entries such as an O for a zero or a B for a 8 seem to stand out more clearly when they are printed to the screen in isolation using this method, rather than reading a long list across the screen including all the commas. Hope this helps.

Commands

I OWN an Amstrad CPC464 without a disc drive and I have tried using the commands CLOSEIN, CLOSEOUT, OPENIN, AND OPENOUT without success so far.

I would be grateful if you would help me out with this subject. - **P. Cross**

● *In order to explain the use of the four commands you mention, here is a short program which creates a file called DATA. Line 10 to 70 store the numbers 1 to 10 in the file and lines 100 to 150 read them back afterwards and print them to the screen. Remember to rewind the tape after saving the data.*

```
10 REM File Handling
20 OPENOUT "DATA"
30 FOR i=1 TO 10
40 WRITE #9,i
50 NEXT
60 CLOSEOUT
70 PRINT "File saved ..."
80 PRINT "Press a key>"
90 IF INKEY$="" GOTO 90
100 OPENIN "DATA"
110 FOR i=1 TO 10
120 INPUT #9,number
130 PRINT number
140 NEXT
150 CLOSEIN
160 END
```

GAMES CAN HELP TOO ! !

All the community is now being challenged by computers in everyday life and any chance to have a successful experience on a computer is important.

Don't under-estimate the benefit of using games to ease the initial process of computer understanding.

"Computer-Knowledge-Poor" people or the inexperienced "user" may need the lower threat of entertainment to accustom themselves to the screen, keyboard, disk drive jungle.

Just as much thought is needed to choose the right "game" program must fulfill the interest needs of the person learning computer usage. Worth investigating are the adventure and strategy type games as these can be easily grasped in concept as novels and board games.

Unstructured time on the computer is often needed by post school-age people to remove some of the threat from the situation. Games decrease this threat by excusing failure during familiarization of the keyboard without the added problems of codes, keywords and concepts.

Initial learning can be hampered if the computer being used appears formidable. Even a business computer can become an adventure into inter-active novels, doodle pads, board games and many other worlds before the 200 page manual of techno-lingo is tackled.

Quite often this period of enjoying the computer can sustain 'users' through frustration later.

If the computer is not fully used in your situation it is just like a blank book, it can become a different tool than its normal 'serious' self. So if you have people over your shoulder as you use the beast expressing either doubts or envy find a program which you can LOAD for them to experience.

Gordon Pearce, Project Officer Waverley Community Youth Support Scheme (Vic).

Auto-Boot Is out!

I own an Amstrad CPC-464 and recently purchased a DDI-1 Disk Drive and interface.

After transferring all my Basic programs to disk I found that I had quite a few, and wrote a menu program for them.

This works very well, but I have tried writing a COM file to immediately boot my menu program on power-up.

This is where my trouble starts - all my program does is go into AMSDOS. It doesn't load my menu program.

How do I go about creating a boot file? Could you possibly write an article on bootstrapping your own programs? **George Fontanini**

● *The only Basic programs that you can Auto-boot using CP/M are those written using a CP/M - based Basic such as Mallard.*

I'm afraid the simple answer is that you are going to have to resort to running your menu program as normal.

Harnessing the SPA

WE saw last month how physical ports and device drivers are assigned to CP/M's logical devices, CON:, LST:, PUN; and RDR:, using STAT. However it is also possible for programs which you write to determine and change device assignments directly.

CP/M maintains the assignments in a variable, IOBYTE, at address 3 in the System Parameter Area (SPA).

Each logical device has a field of two bits in the IOBYTE which specifies which of the four possible physical devices it is currently assigned to. This is shown clearly in Figure I.

To discover what the current assignments are you simply read the IOBYTE and decode the values of the four fields. Similarly, to change one or more fields directly all you need to do is write the appropriate bit pattern to the IOBYTE.

The SPA, which lives at addresses 0000h to 00FFh, right at the bottom of memory, contains a lot of other information which is maintained by CP/M for programs to make use of.

Figure II shows the layout of the SPA. Let's see what it means.

Addresses 0,1 and 2 contain a Z80 JP instruction, whose destination is the warm boot entry point to the BIOS. So to terminate a program properly when it finishes (or to restart CP/M if your program detects an error of some sort) you simply have to execute a JP 0000h (or RST 00h) instruction. CP/M will take care of the rest.

Address 3, as we have seen already, is the IOBYTE, Address 4 contains a variable called DRVUSR. The lower four bits of this byte tell us which disk drive is currently logged as the default drive - a value of zero would mean drive a:, a value of one would mean drive b:, and so on.

Note that CP/M can handle up to 16 disk drives, however for reasons best known to themselves Amstrad decided that we would have to be content with a maximum of only two when they designed the hardware.

Part V of COLIN FOSTER's exploration of CP/M 2.2

The top four bits of the byte at address 4, again considered as a value of 0-15, specify the current user number. This allows us to partition a disk into up to 16 user areas, each of which is invisible to all the others.

As a moment's thought will show, this is virtually useless on an Amstrad due to the small size of our disks - we would have to keep several copies on the same disk of any program we wanted to use (one in each user area) as no user area can access programs or files in another, and we'd soon run out of space.

You can change user areas from command level by using the CCP USER command; try it and see how restrictive it is. It is a feature useful only on larger machines with lots of storage capacity on hard disks and we will always work in User 0.

Addresses 5, 6 and 7 of the SPA contain another Z80 JP instruction. This time its destination is the BDOS entry point. This is our link with the operating system whenever we want it to carry out any functions for us. Quite simply,

Bit	7	6	5	4	3	2	1	C
Field	LST:		PUN:		RDR:		CON:	
Value:								
00:	TTY:		TTY:		TTY:		TTY:	
01:	CRT:		PTP:		PTR:		CRT:	
10:	LPT:		UP1:		UR1:		BAT:	
11:	UL1:		UP2:		UR2:		UC1:	

Figure I: Structure of IOBYTE

ADDRESS	CONTENTS	
&0000- &0002	JPWBOOT (RST 0).	
&0003	IOBYTE	
&0004	DRVUSR.	
&0005- &0007	JP BDOS.	
&0008- &000F	RST 1	Reserved for firmware functions.
&0010- &0017	RST2	
&0018- &001F	RST3	
&0020- &0027	RST4	
&0028- &002F	RST5	
&0030- &0037	RST6	Unassigned - user RST.
&0038- &003F	RST7	Reserved for interrupts.
&0040- &004F	Reserved by DR LOGO; available for use by other programs.	
&0050- &005B	Reserved.	
&005C-&007F	Default FCB.	
&0080- &00FF	Default DMA sector buffer.	

Figure II: Contents of the System Parameter Area (SPA)

the BDOS provides ready made routines which programs should use to perform all of their input/output.

We can tap the resources of the BDOS by making function calls. The mechanism for this is straightforward - we load some of the processor registers with any information the BDOS will need to carry out the operation, we load register C with the number of the function we want to use, and execute a CALL 0005 instruction. We'll return to this in more detail next month.

Addresses 8 through &3F contain the Z80 RST vectors all used by the firmware (not by CP/M) except for RST 6 at addresses &30 to &37. This is an undedicated user RST, and is available for use by any program. (The CP/M utility DDT uses it as a breakpoint instruction when debugging programs>)

RST7 (addresses &38 to &3F) is used by the firmware for interrupts. The processor normally runs in interrupt Mode 1, so address &38 is the entry point for hardware interrupts.

If the firmware cannot account for an interrupt occurring, it will assume that it is external and that a user routine will handle it. To invoke this, a call is made to address &3B, so if external devices which can interrupt are in use then addresses &3B to &3F must contain code to vector the call to the appropriate routine.

Addresses &40 to &4F are normally reserved in CP/M systems as scratchpad workspace for the BIOS, however, the Amstrad BIOS does not use these bytes and so they are available to your programs.

It should be noted that DR LOGO does use these bytes for its own purposes, so if you are thinking of installing an RSX under Logo make sure it doesn't corrupt these bytes.

Addresses &50 to &5B are not used by CP/M but should normally be considered to be reversed.

The block of SPA memory running from &5C up to &7F makes up the default File Control Block, or FCB, which the CCP sets up for use by a transient program. If your program does not require it as an FCB it is available as free memory.

The 128 bytes which comprise the upper half of the SPA, from &80 to &FF make up the default Disk Memory Access (DMA) buffer used by the BDOS to buffer records to and from disk.

Both of these areas of the SPA are only used when we read or write disk files, and we'll return to them when we discuss disk operations in a later article. For the moment, you can ignore them.

Next month we'll start to develop example routines to explore the available BDOS function calls and make use of them. In order to write these routines and test them you must have the use of an editor program with which to enter the source text and an assembler to create the machine code.

Unfortunately you cannot easily use an AmsDOS editor/assembler such as Hisoft's DevPac or C.W.T.A.'s RAW for this as AmsDOS.BIN binary files are not compatible with CP/M's. .COM machine code files - AmsDOS sticks a 128 byte header at the start of its disk files to maintain compatibility with tape.

One way round this if you don't have a CP/M assembler would be to use your AmsDOS assembler to create the .BIN disk file on disk, then go into CP/M and use a debugger such as DDT (which we'll discuss in detail next month) to load the code and move it down in memory by 128 bytes.

You could then exit the debugger and SAVE your program again, this time as a .COM file. A bit contrived, but better than nothing.

However, there really is no excuse for not having a CP/M editor and assembler - even if you can't afford one such as Hisoft's DevPac-80, there are several equally good programs available in the public domain for little more than the cost of a disk.

In the examples in these articles I'll be using a public domain Z80 assembler called ZSM, available from the CP/M Users Group, but any other Z80 assembler will do just as well.

It is worth noting that your distribution disk actually contains a rather nasty line editor, ED.COM, and an assembler ASM.COM, which is so old that it creaks. Explaining the complex and mysterious ways ED has to be coaxed into action would take several articles in itself. Any book on CP/M will describe it fully.

Again, there really is no need to ever use it as better editors are available in the public domain. ASM is of little use unless you are fluent in Intel 8080 assembler - CP/M was originally designed for computers which used the now obsolete 8080 microprocessor, and ASM will not understand Z80 assembler source.

'there really is no excuse for not having a CP/M editor and assembler'

The 8080 executes a subset of the Z80's instruction set, so our smart Z80 will run machine code written for 8080s without any problem - indeed, CP/M itself is written in 8080 code.

However the assembler mnemonics used to write source programs for 8080s, which are all that ASM will accept, are totally different to those for the Z80 which you are used to.

Figure III is a cross-reference table with the Z80 instructions which the 8080 can execute, and the equivalent mnemonic syntax which programs such as ASM and DDT will accept.

● Next month, we'll look at how the CP/M debugger DDT.COM works, and start developing routines to make use of BDOS function calls.

Add these titles to your CP/M disks - FREE!

CP/M users are fortunate in having a wealth of public domain software. However sorting out the best of it, and customizing it for your Amstrad, can be rather difficult for the newcomer.
As a service to our readers this month's Public Domain disk will contain - in addition to the main iems (Small 'C' compiler) - three invaluable CP/M public domain utilities - an enhanced directory lister, an intelligent erase facility and an unerase command.

Full details of these vital CP/M routines are given here.

D.COM

This program is an enhanced directory lister, designed to replace the more limited CCP command DIR. As with all CP/M programs, it is loaded from disk and executed simply by typing its name at the CCP prompt, that is:

```
A>d
```

This is the simplest command D.COM can be given, and will cause the program to print on the screen a list of all the files on the currently logged-in disk (a: in this case), in a similar form to the AMSDOS "CAT" command.

The list is sorted into alphabetical order across the screen and the space, in kilobytes, which each file takes up on disk is also displayed. Finally an information line is printed which tells us both the total space used on the disk and the amount remaining free.

D.COM will also accept a filename as a parameter in exactly the same way as DIR> This may contain wildcard characters. For example, the command:

```
B>d example.001
```

will only list the information relevant to the file EXAMPLE.001 on the disk in drive B: whereas the command:

```
B>d a:#.com
```

will display all the files on the disk in drive a: which have an extension of .COM.

D.COM is designed to run under CP/M 2.2. It will run under CP/M Plus, but will not give the correct result for space remaining free on the disk.

ERQ.COM

This is a replacement for the CCP command ERA. It is used in exactly the same way, but allows you to have second thoughts. For example, typing the command:

```
A>erq b:#.doc
```

will display the names of all files on the disk in drive b: which have an extension of .DOC, and will prompt you for each file with a question mark. Typing y will erase the file and typing n will leave it alone and move on to the next file. Typing anything else will abort the command.

This facility not only allows you to have second thoughts, but also gives you an easy way of erasing several files selectively without having to type each name separately - for example, typing the command:

```
A>erq #, #
```

is not as deadly as it appears - it simply lets you browse through all the files on the disk and selectively delete whichever you want.

ERQ will run under both CP/M 2.2 and CP/M Plus.

UNERASE.COM

This program does exactly what its name suggests. If you ever accidentally erase a file - it can happen, even with ERQ - then run:

```
A>unerase <filename>.<ext>
```

to bring it back from the dead. The restrictions are as follows:

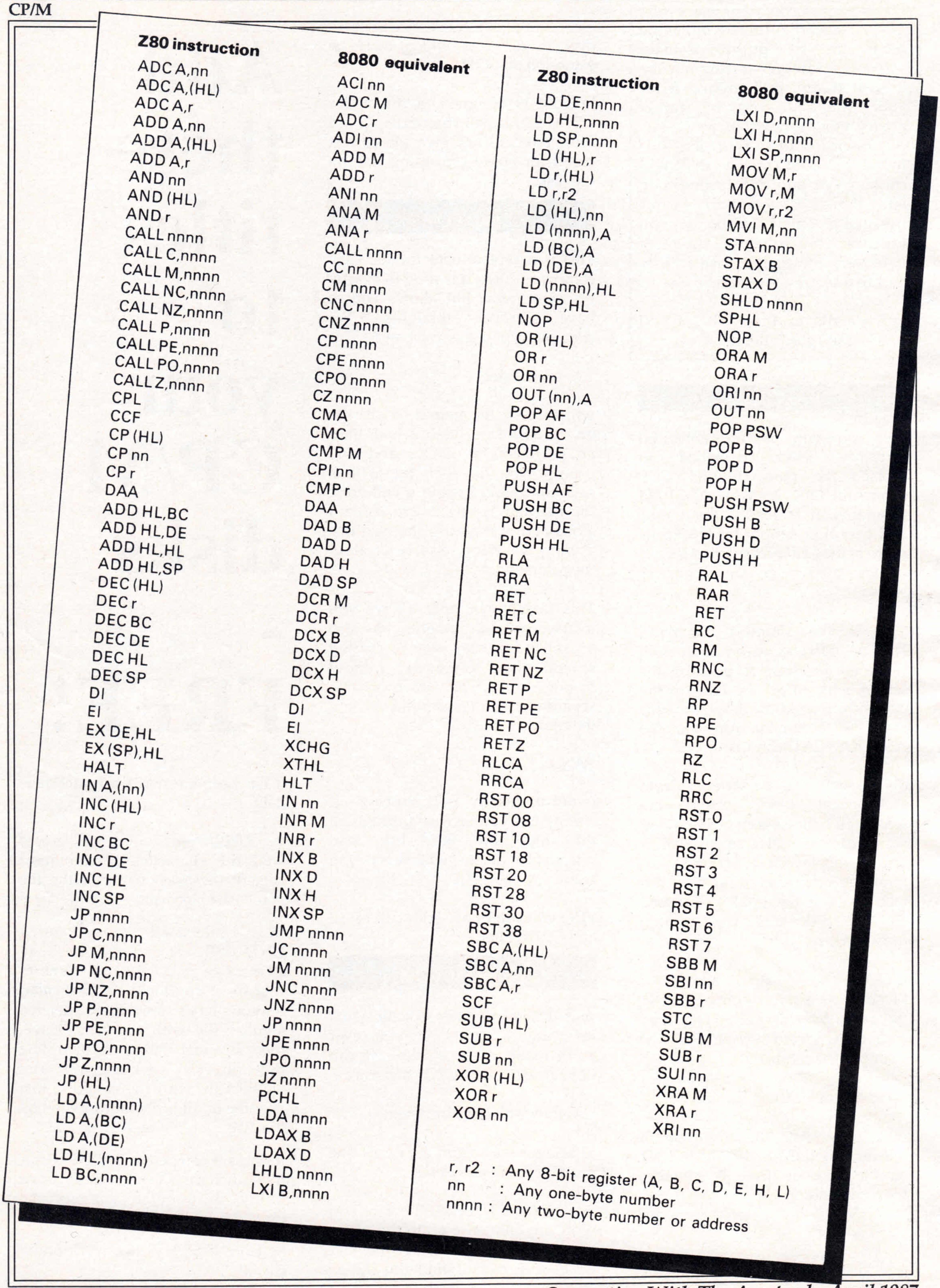
✎ You must UNERASE before you write anything else on to the disk - if you do not, you may find that UNERASE cannot find the file as it has been overwritten by the new data.

✎ UNERASE will not accept wildcard characters - you must specify the exact name of the file you wish to recover.

✎ Understand that using UNERASE at all carries an element of risk - do not come to rely on it routinely. In particular, if you have previously deleted another file with the same name, there is a real chance of corrupting the directory when you run UNERASE, with the resulting loss of some or all of the information on the disk.

If you are competent in such things, it is much safer to use a disk repair utility to examine and fix the directory entries directly.

✎ UNERASE will not work under CP/M Plus.



Z80 instruction	8080 equivalent
ADC A,nn	ACI nn
ADC A,(HL)	ADC M
ADC A,r	ADC r
ADD A,nn	ADI nn
ADD A,(HL)	ADD M
ADD A,r	ADD r
AND nn	ANI nn
AND (HL)	ANA M
AND r	ANA r
CALL nnnn	CALL nnnn
CALL C,nnnn	CC nnnn
CALL M,nnnn	CM nnnn
CALL NC,nnnn	CNC nnnn
CALL NZ,nnnn	CNZ nnnn
CALL P,nnnn	CP nnnn
CALL PE,nnnn	CPE nnnn
CALL PO,nnnn	CPO nnnn
CALL Z,nnnn	CZ nnnn
CPL	CMA
CCF	CMC
CP (HL)	CMP M
CP nn	CPI nn
CP r	CMP r
DAA	DAA
ADD HL,BC	DAD B
ADD HL,DE	DAD D
ADD HL,HL	DAD H
ADD HL,SP	DAD SP
DEC (HL)	DCR M
DEC r	DCR r
DEC BC	DCX B
DEC DE	DCX D
DEC HL	DCX H
DEC SP	DCX SP
DI	DI
EI	EI
EX DE,HL	XCHG
EX (SP),HL	XTHL
HALT	HLT
IN A,(nn)	IN nn
INC (HL)	INR M
INC r	INR r
INC BC	INX B
INC DE	INX D
INC HL	INX H
INC SP	INX SP
JP nnnn	JMP nnnn
JP C,nnnn	JC nnnn
JP M,nnnn	JM nnnn
JP NC,nnnn	JNC nnnn
JP NZ,nnnn	JNZ nnnn
JP P,nnnn	JP nnnn
JP PE,nnnn	JPE nnnn
JP PO,nnnn	JPO nnnn
JP Z,nnnn	JZ nnnn
JP (HL)	PCHL
LD A,(nnnn)	LDA nnnn
LD A,(BC)	LDAX B
LD A,(DE)	LDAX D
LD HL,(nnnn)	LHLD nnnn
LD BC,nnnn	LXI B,nnnn
LD DE,nnnn	LXI D,nnnn
LD HL,nnnn	LXI H,nnnn
LD SP,nnnn	LXI SP,nnnn
LD (HL),r	MOV M,r
LD r,(HL)	MOV r,M
LD r,r2	MOV r,r2
LD (HL),nn	MVI M,nn
LD (nnnn),A	STA nnnn
LD (BC),A	STAX B
LD (DE),A	STAX D
LD (nnnn),HL	SHLD nnnn
LD SP,HL	SPHL
NOP	NOP
OR (HL)	ORA M
OR r	ORA r
OR nn	ORI nn
OUT (nn),A	OUT nn
POP AF	POP PSW
POP BC	POP B
POP DE	POP D
POP HL	POP H
PUSH AF	PUSH PSW
PUSH BC	PUSH B
PUSH DE	PUSH D
PUSH HL	PUSH H
RLA	RAL
RRA	RAR
RET	RET
RET C	RC
RET M	RM
RET NC	RNC
RET NZ	RNZ
RET P	RP
RET PE	RPE
RET PO	RPO
RET Z	RZ
RLCA	RLC
RRCA	RRC
RST 00	RST 0
RST 08	RST 1
RST 10	RST 2
RST 18	RST 3
RST 20	RST 4
RST 28	RST 5
RST 30	RST 6
RST 38	RST 7
SBC A,(HL)	SBB M
SBC A,nn	SBI nn
SBC A,r	SBB r
SCF	STC
SUB (HL)	SUB M
SUB r	SUB r
SUB nn	SUI nn
XOR (HL)	XRA M
XOR r	XRA r
XOR nn	XRI nn

r, r2 : Any 8-bit register (A, B, C, D, E, H, L)
nn : Any one-byte number
nnnn : Any two-byte number or address

Welcome to the fifth in our series of public domain disks. So far I have concentrated on programming utilities as this type of program is to be found in great abundance. This month's offering is no different. It is the Small C compiler from volume 224 of the SIG/M public library. This programming language is now the 'in language' as far as the main stream of computing is concerned

To give you a short history of this very versatile programming language, it started out as an extension of BCPL (Basic Combined Programming Language) in the Bell Labs in the USA. As you may already know, Bell labs is the birthplace of UNIX, the operating system that is now being provided on such heavyweight machines as IBM's new 80286 PC. UNIX and the C language are now inseparable as most of UNIX is written in C. The standard is 'The C Programming Language' by Kernighan and Ritchie. The reason for C's popularity is it's versatility coupled with it's stability to be used for both systems level programming and application programs.

I am not seriously suggesting that you should consider writing applications in Small C, but as an introduction to the C language it has two things going for it - it is cheap and it includes floating point mathematics routines. Small C has a number of functions missing that are considered essential as part of the standard. This makes it limited for anything other than experimentation with the language.

Before playing with this Small C compiler I was not really interested in learning the language as I thought that I could do all that I wanted in assembler without excessive development time. I was wrong. I now realize that C can do all that assembler can do, faster and with less development time and usually (more importantly) with less bugs first time round. The standard C for CP/M is the BDS C package which is still available from some retailers. Failing that, there is the HISOFT package written for the CPC range of computers. The Hisoft implementation is integer only but a lot can be done with an integer package providing you don't try to write an accounting system in it! There are about 30 disks of C programs in the C User Group Library and a number of other programs in the SIG/M library so there is no shortage of examples if you decide to investigate this language further.

Incidentally, I have often heard it said that the public domain software is no use for serious applications. On the whole, I would have to agree with that statement BUT, I feel that the value of the PD software is in the feel it gives you for the way in which software can be written (the user interface, I suppose) and for what can actually be done under the constraints of a 'bare bones' operating system like CP/M. Also for programmers, as against users, there are many implementations of languages, routines, file-handling and communications to name just a few, all of which I learn from, and all of which can help to show you how to (or how not to ! !) implement the idea on your own machine.

Now, back to this month's disk. The Small C compiler produces an assembly language file in a suitable syntax for ZMAC.COM which is a macro assembler that produces an object code file that is in the correct format for ZLINK.COM which produces a .COM file (finally!). Both ZMAC and ZLINK are supplied on the disk as are the all important IOLIB, PRINTF1 and PRINTF2 libraries along with ARGS, TRANSCEN and FLOAT libraries. These are all essential for the operation of the compiler and their use is explained (partially) by the documentation files also supplied on the disk. There is also a test file that warrants close examination.

A typical session with the compiler goes something like this:- write your source file using your editor, compile it, correct mistakes and repeat until no errors at compile time. ZMAC your source code until there are no errors. ZLINK the necessary modules and then run your .COM file. Note the results, and start all over again because you won't have got it right the first time unless you are exceptionally lucky and then I don't want to hear about it because I'm never that lucky! There is going to be a shortage of disk space for users with 3 inch drives, so as your first exercise you should store the .DOC files on another disk and wipe them off your work disk. Both ZMAC and ZLINK produce listing files which go on the default drive, but these can be skipped, saving some space for your source file. Included on the disk is the submit file TESTCOM.SUB which you use from the A> prompt with TESTCOM filename - note no file type should be specified. This file takes care of wiping out the .PRN and .MAP files produced during the compilation process.

For an introduction to the C language you could not go past the excellent series 'C for Smarties' that has been running for some time in Your Computer magazine. Back issues, or, I believe, photocopies of specific articles are available for a price and could be a worthwhile investment before committing yourself to the purchase of a fairly expensive commercial compiler. Alternatively, both McGills newsagency and the Technical Book and Magazine company here in Melbourne have a large range of C books, including a handbook for this Small C. As you will know, any computing book is expensive, but for around $27 you can get Practical C, which is written for standard C, but with particular emphasis on Hisoft C as it is on Amstrad and Spectrum home computers. I bought it and can recommend it for novices.

Well, that's all for this month. I am still trying to get hold of a decent editor for those of you who don't want to spend your life trying to fathom WordStar and if I can get such an animal, it will be featured next month. Failing that, a graphics package for Epson compatible printers looks promising and I will shortly be featuring the Original Adventure by Crowther and Woods. Incidentally, this month's disk contains a great game, suitable for CPM 3.0 users (PCW's included).

Shane Kelly

MTBASIC

Multitasking BASIC Compiler

Softaid's MTBASIC is the Basic Compiler for MS-DOS, PC-DOS and CP/M-80 Systems

Features

MULTITASKING - Multitasking is an important new feature for microcomputers. Before MTBASIC, multitasking was only availabe on mini's and mainframes. With MTBASIC, up to 10 Basic programs (tasks) can run concurrently, giving minicomputer performance to the micro user. Using MTBASIC's multitasking features, you can create games that continue to compute even while awaiting input from the user, create your own print spoolers, and write nearly any application requiring multiple simultaneous operations. MTBASIC performs all of the context switching; no special hardware or software is needed. Although MTBASIC does support all hardware interrupt modes, it will automatically generate software interrupts if no source of hardware interrupts exists. A complete set of Basic statements to control tasks and multitasking is provided.

WINDOWING - MTBASIC can create up to ten windows on the screen at the same time. Each separate task, or multiple windows can be assigned to each task. One window can be overlayed by another, and overlayed windows can later be restored, making pop up and pop down menus a breeze to implement. A complete set of MTBASIC window control statements are provided. Windowing works equally well with all operating systems.

FAST CODE - MTBASIC generates highly optimized object code. Byte's September, 1981 Sieve runs in 26 seconds on a 4 Mhz Z-80 (10 iterations). Interpreters take over 1400 seconds. MTBASIC typically compiles over 100 Basic statements per second, so most programs compile in a second or two and then run 50 times faster than interpreted Basic code.

INTERACTIVE - Because MTBASIC runs interactively, like an interpreter, debugging is quick and easy. To the programmer, MTABASIC appears to be a tremedously fast interpreter. MTBASIC programs are entered with line numbers and each line's syntax is checked as it is entered. Unlike interpreted Basics, however, when RUN is typed, MTBASIC compiles the program to object code and executes it. The code generated by MTBASIC can be stored to disk files and executed stand alone. No runtime license is required to distribute object code.

Technical Data

MTBASIC produces ROMable code. Stand alone code, stored to disk file, can be downloaded to ROM to run in target systems. No runtime license is required to distribute the ROMed code.

FUNCTIONS: Multi-line, user defined functions, ACOS, ADR, ASC, ASIN, ATAN, BAND, BOR, BXOR, (binary AND, OR & XOR), CHR$, CONCAT$, COS, CVI, CVS, ERR AND ERR$ (disk error processors), EXP, INP, KEY, LEN, LOG, MID$, MKI$, MKS$, PEEK, RND, SIN, SQR, STR$, TAN, VAL.

STATEMENTS: CALL (Call an assembly routine), CANCEL (stop a task), CHAIN, CLOSE (file), CLS, CODE (in line machine code), CURSOR, DATA, DEF (define function), DEFSEG (8088 only), DELETE (file), ERASE, EXIT (leave a task), FEND (end function definition), FIELD, FILE, FOR/NEXT, PRINT (formated printing - more powerful than PRINT USING), GOSUB, GOTO, IF-THEN, INPUT, INPUT$, INTEGER, INTMODE (set Z-80 interrupt mode), INTON & INTOFF (interrupts on/off), JVECTOR & VECTOR (hardware interrupt vector definitions), OFF ERROR, ON ERROR, OPEN, OUT, POKE, PRINT, PUT, RANDOMIZE, READ, REAL, REM, RETURN, RGET, RUN (start a task), SEEK (random files), STOP, STRING, TASK, TRACEON, TRACEOFF, WAIT (suspend a task for a specified amount of time), WFRAME (draw an outline around a window), WINDOW (create a window), WSAVE (save a window), WSELECT (select a window), WUPDATE (update a window).

MTBASIC supports single precision floating point with 6 decimal digits of accuracy, long variable names, sequential and random disk files, strings and string arrays, and automatic mixed mode arithmetic.

FULLY RECURSIVE - MTBASIC is fully recursive, permitting development of sophisticated algorithms which previously could only be coded in PASCAL.

Specially configured for your Amstrad and available by mail-order only from Strategy Software -

Make Your Amstrad Fly!

The MTBASIC package includes all the necessary software to run in interpreter or compiler mode, an installation program (so any system can use windowing), three demonstration programs, and a comprehensive manual.

$125

Strategy Software
P.O. Box 11
Blackmans Bay
Tasmania 7152
[002] 29 4377

$125

Available NOW! - See Page 72 for ordering information

BUSINESS COMPUTING WITH THE AMSTRAD

APRIL 1987

EVEN WITH AN EXPANDED BUSINESS SECTION THIS MONTH WE WERE UNABLE TO FIT IN THE LATEST U.K. NEWS AND YOUR LETTERS - OUR APOLOGIES , THEY'LL RETURN NEXT MONTH.

PlannerCalc

GABRIEL JACOBS looks at a spreadsheet that, while long in the tooth, can be considered a good starter pack for the small businessman. And at an attractive price, too.

PLANNERCALC, which is a spreadsheet written originally for micros such as the Radio Shack Model II and the Superbrain, dates from early 1982. Although it never became one of the giants of business software it held its own, being adequate for many applications, and was an obvious candidate for a PCW resurrection.

Like all spreadsheets PlannerCalc works with a grid of cells (boxes), each identified by a column and row number, and with the screen acting as a window on to the worksheet. When values have been entered into the cells calculations can be carried out to give totals, averages, maxima, mimima and so on.

In addition to the usual arithmetic and logical operators, and basic functions such as AVErage, SUM OF and MAX and MIN, PlannerCalc has NPV (Net Present Value, CUMulative total, and the more unusual GREATER OF, LESSER OF, and GROW BY (automatic incrementation).

Conditional statements - IF ... THEN ... ELSE - are also available. It is not well endowed for scientific and statistical calculations, having only LN (Natural Log) and EXPonent, but its main market is the small businessman, not the mathematical modeller.

Setting up a worksheet means creating a template of commands entered by row - in PlannerCalc called lines - or column. A typical series of line commands might look something like:

```
LINE 1 SALES = 90,550,72,647,28,20,
2,987
LINE 2 DISCOUNT = IF LINE 1>= 500
THEN LINE # .10
LINE 3 INVOICE = SALES - DISCOUNT
```

Lines 2 and 3 tell the program to calculate a discount of 10 per cent on each item of data in Line 1 - each item being a value in a column - if it is greater than or equal to $500, or else to invoice the item at the original figure. A printout will then show each sale, the discount it has attracted and the amount to be invoiced.

The same kinds of calculation can be carried out on columns rather than rows and can become very complex with multiple conditions, discontinuous logic, incremented growths and the like.

All this may seem daunting to an inexperienced user, and it has to be said that to give of its best PlannerCalc does expect some understanding of programming procedures.

However that is true of most spreadsheets which offer facilities over and above the most basic ones, and in any case the commands soon become familiar territory.

This is especially true of PlannerCalc because for the most part it uses an English-like syntax. Commands such as COPY COLUMN 6 TO COLUMN 15, SUM OF LINE 1 THRU LINE 9 or DECIMALS ARE 3 can be understood even by complete beginners.

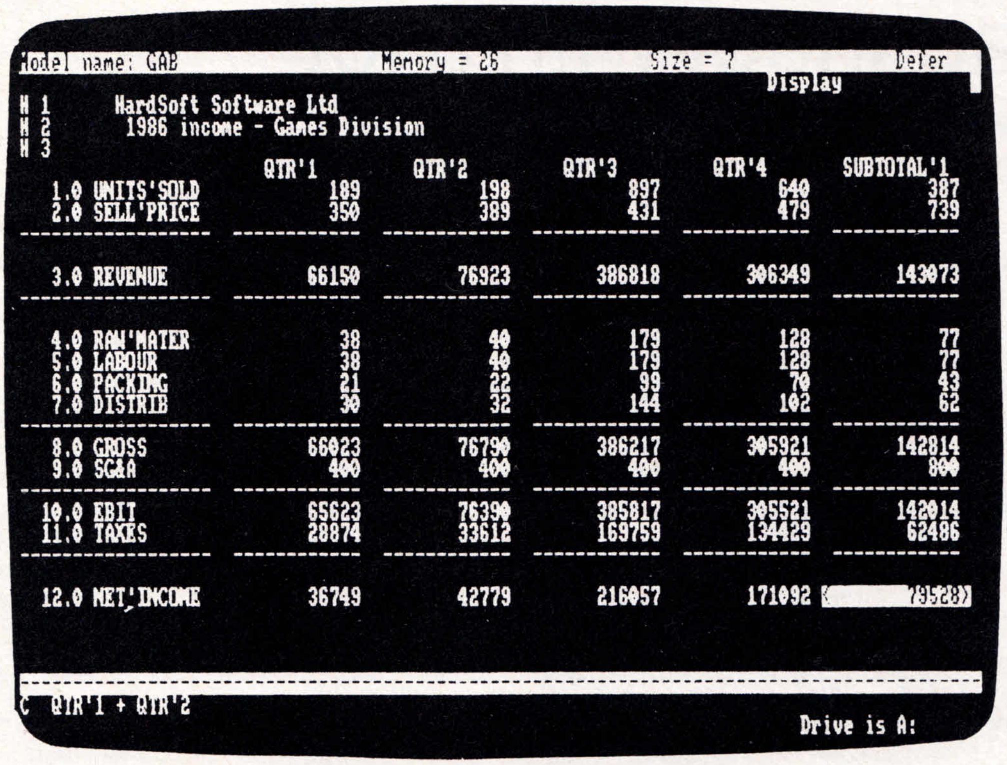
Columns and rows can be identified in commands by names you give them (called Labels) rather than numbers, and this also helps to make formulae comprehensible. The system does have its disadvantages however. For instance, care has to be taken to change all appropriate formulae if a column or row is re-labeled - the formula PROFIT = INCOME - EXPENSES will not be calculated if you have re-labeled EXPENSES to read COSTS.

Designing a hard copy printout of all or part of an electronic worksheet is never an easy task, and PlannerCalc can offer just as much of a challenge in this area as other spreadsheets, particularly if you want the hard copy to differ significantly from the screen display.

PLANNERCALC SPECIFICATIONS

Maximum Number of Rows	512
Maximum Characters per Row	80
Maximum Number of Columns	128
Column Width	2 to 30 characters
Maximum Number of heading lines	10
Maximum Heading Width	80 characters
Maximum Line & Column Labels	12 characters
Maximum Number of Decimals	28
Significant Digit Range	2 to 10

Standard editing/format facilities
Standard functions plus NPV, GREATER OF, LESSER OF, GROW BY
Restricted set of statistical/math functions
Absolute and Relative referencing
Vertical and Horizontal split screen windowing (maximum two windows)
Run-time data input
Choice of Matrix model or Worksheet printouts



```
Model name: GAB              Memory = 26          Size = 7            Defer
                                                             Display
H 1      HardSoft Software Ltd
H 2       1986 income - Games Division
H 3
                       QTR'1         QTR'2         QTR'3         QTR'4     SUBTOTAL'1
    1.0 UNITS'SOLD       189           198           897           640          387
    2.0 SELL'PRICE       350           389           431           479          739
----------------- ------------  ------------  ------------  ------------  ------------

    3.0 REVENUE        66150         76923        386818        306349       143073
----------------- ------------  ------------  ------------  ------------  ------------

    4.0 RAW'MATER         38            40           179           128           77
    5.0 LABOUR            38            40           179           128           77
    6.0 PACKING           21            22            99            70           43
    7.0 DISTRIB           30            32           144           102           62
----------------- ------------  ------------  ------------  ------------  ------------
    8.0 GROSS          66023         76790        386217        305921       142814
    9.0 SG&A             400           400           400           400          800
----------------- ------------  ------------  ------------  ------------  ------------
   10.0 EBIT           65623         76390        385817        305521       142014
   11.0 TAXES          28874         33612        169759        134429        62486
----------------- ------------  ------------  ------------  ------------  ------------

   12.0 NET'INCOME     36749         42779        216057        171092 <      79528>


---------------------------------------------------------------------------------------
C  QTR'1 + QTR'2
                                                                     Drive is A:
```

Sample PlannerCalc screen

However, the program handles simple formats as easily as many of its younger competitors, while still allowing underlining, blank lines and so on. You can also choose between worksheet format, which is basically an extended screen dump, or report format.

One of the advantages of electronic spreadsheets is that they allow the user to make plans and predictions, and to ask What if?, either directly in the best packages, by using combinations of conditional statements in a worksheet, or by putting projected or hypothetical values into a temporary copy of it.

PlannerCalc is far from being in the top league in this respect. It does however have an interactive run-time planner not found on even some of its latest rivals.

The facility enables models to be set up with cells programmed to prompt the user for an input before a calculation takes place, something which could be invaluable in consultancy business, say, where on-the-spot costings may have to be made.

There is an option to defer calculation, which is useful if you have a lot of data to input and don't want to wait for a complete recalculation at each entry. Horizontal or vertical split-screen windowing allows two distant parts of a worksheet to be scrolled independently on the same screen.

Another feature is relative column referencing for calculations which take into account values in columns other than the current one- a bonus in such things as cash flow analysis.

' Some applications have to be designed to fit the spreadsheet instead of the other way around'

In some ways however PlannerCalc shows its age. For example, it will only replicate in one dimension - a whole column or a whole row, cells will not accept text and there is no facility for protecting individual cells or parts of worksheets by locking them.

Much worse is that formulae are always taken to apply to an entire row or column, so that single cells cannot perform independent calculations as they can in nearly all modern spreadsheets.

In practice this means that with PlannerCalc some applications have to be designed to fit the spreadsheet instead of the other way round - a case of the tail wagging the dog.

The seriousness of such deficiencies will of course depend on individual requirements, and they have to be set against PlannerCalc's clear user-interface which includes a context-sensitive Help facility, its comprehensive documentation and above all its comparatively low cost of $99.

Another boon is that it is upwardly compatible with it big brother MasterPlanner, priced at $175. This has many extra features, including multiple windowing, an overspill-to-disk facility for large worksheets, mathematical modeling functions and, for an extra $75, a library of customized applications.

With the possibility of such expansion PlannerCalc has to be considered a good starter pack, and if you never need to upgrade you'll probably have had value for money.

If you do outgrow PlannerCalc both the data you have entered and the development skills you have acquired will be portable. This is a significant consideration in choosing a package.

Gabriel Jacobs

Spreadsheet with a lot to offer

Gabriel Jacobs reviews Scratchpad Plus

CAXTON Software has chosen a curious name - Scratch Pad Plus - for its spreadsheet program. It sounds as if it might refer to some kind of electronic notebook - in fact there is an ideas processor called Scratchpad, produced by Innovative Software, which runs on the BBC Micro.

But there's no obvious connection between making rough jottings - which is what a scratchpad is used for - and working with a spreadsheet.

Very briefly for the uninitiated, a spreadsheet consists of a grid of cells, each identified by a column number and a row number, with the screen acting as a window moving over the grid. Calculations can be carried out on values either entered directly into the cells, or deduced from formulae. For example, a formula something like:

AVG (B1 - B9)

entered into cell C1 - Column C, Row 1 - will display in that cell the average of all the values in the block of cells between B1 and B9.

Now Caxton describes ScratchPad as an "enhanced" spreadsheet, and the blurb in the introduction to the manual proclaims categorically that the program differs from all other spreadsheets on the market in three distinct respects.

First it uses virtual memory - overspill to disk - so grid size is limited only by disk capacity. Secondly it allows variable grid dimensions, in other words the number of rows and column need not be fixed in advance, thus saving on memory.

Thirdly, it offers multiple windowing as shown in Figure I. This enables you to view simultaneously and scroll independently as many separate parts of a worksheet as will fit on the screen - and therefore, incidentally, to lock labels in a variety of ways so that they are always displayed precisely as you want them - as opposed to the usual limit of two windows on a screen split either horizontally or vertically.

However these undoubted virtues do not explain the program's name. Nor, despite Caxton's unqualified claims, are they unique to ScratchPad Plus. To take just one example, MasterPlanner from Comshare, which runs on the PCW, has both multiple windows and virtual memory.

As for variable grid dimensions, some spreadsheets - admittedly running 16 and 32 bit machines - go one step further, using what is called a sparse-filled matrix, in which empty cells consume practically no memory.

Nevertheless ScratchPad Plus turns out to be in my view just about the best spreadsheet available for Amstrad machines. And at $175 - which includes a free copy of Caxton's Smartkey, a handy keyboard-customization program - it represents real value for money.

Those brought up on an industry standard spreadsheet like SuperCalc and Multiplan will quickly recognize when they use Scratch Pad Plus that it has been partly modeled on these packages - and you will not be surprised to learn, therefore, that it can easily read in and create SDF and DIF data interchange files.

But while incorporating the best ideas of its predecessors the package has also been endowed with many extra features, quite apart from the three already mentioned. What we have here is a powerful business tool which should satisfy the needs of most users.

For a start it has a wide range of mathematical functions and even a facility for reversing the order of calculation - that is, columns before rows - so fairly complex models can be handled.

It has a sort option which will arrange sections of a worksheet into ascending or descending alphabetical or numerical order and an interactive lookup function for finding values greater than given search criterion.

There is also an intelligent replication facility, which translates formulae copied to another part of a worksheet so that they bear the same relationship to the new section as they did to the old an an option which enables one worksheet to be merged into another.

The grid itself it totally flexible. Any cell can contain text, values, a formula or a function call. Text automatically spills over into adjacent cells, so global column width need not be adjusted to accommodate it.

Values can be prefixed with a $ sign, shown with commas - for example, as 1,100 - and have their decimals truncated in the display without affecting the accuracy of a calculation.

Formulae can range from a simple total to a complex series of conditional functions dependent on other formulae in blocks of distant cells.

Yet with all that, editing and formatting a worksheet on the screen are child's play. There are similar commands for changing the contents of cells, for inserting and deleting rows and columns, for protecting cells against unauthorized or accidental changes and so on. Also an on-line Help facility is always there for quick reference.

Above all, hard copy printouts have been made as easy as they could possibly be. To print out a section of a worksheet you merely select page parameters from a print menu, and specify the rectangle of cells to be printed.

If you design your printout carefully, all well and good. If you don't, ScratchPad Plus will make a praiseworthy effort to produce a readable output.

FUNCTIONS
Arithmetic operators plus modulus, exponentiation, and the following:

ABSolute
ATN (arc tangent)
AVG (average)
CHOOSE(1,x,y, ...)
(depending on 1, returns one of the other arguments in the list x,y ...)
COSine
COUNT (counts the number of entries in a block of cells)
EXPonent
FACtorial
IF ... THEN
INTeger
LOG
LOOKUP (compares values in two tables)
MAXimum
MINimum
NPV (Net Present Value)
PI
SINe
SQR
SUM
TANgent

If a worksheet will not fit across a page the program will divide it into blocks by intelligently wrapping columns, then print these blocks one below the other. And of course you can print out a worksheet showing formulae rather than results, a useful option when building complicated models.

One big advantage for PCW owners - no doubt a sign of the times, and maybe of the future - is that ScratchPad Plus comes ready implemented for the machine, whereas on all other machines the program has to be installed with special procedures.

The implementation takes into account the non-standard PCW screen size, allowing a bigger window on to the grid, and the keyboard has been fully configured in a CP/M Setkeys file.

For example, the function keys and the dedicated word processing keys can be used for frequently required commands, which means that single keystrokes replace most of the double-key operations normally expected by the program.

Furthermore the documentation, which includes a handy prompt card and would be just about perfect if an index had been included, is oriented very much towards the PCW. There are helpful comments throughout for PCW users, and where a machine is used to exemplify a point the PCW is invariably the one chosen.

This does not mean that CPC6128 owners should spurn ScratchPad Plus. Life is just made that much easier for the rapidly increasing number of people using the PCW for business purposes.

All this is not to say the Scratch Pad Plus is beyond criticism. It does have its failings, and some of them would have been relatively easy to correct with a little more thought in the design of the program. Perhaps the worst shortcoming is that if the current worksheet has not been saved there is no trap to warn the user that loading a new one will overwrite it.

FEATURES
Full editing, replication and formatting.
Full cell referencing
Nested function calls and conditional statement
Virtual memory facility
Variable or fixed dimension worksheet
Worksheet merging
Multiple windows
Deferred and forced calculations
Reversible order of calculation.
Ascending or descending sorts.
Intelligent printout facility.
On-line Help screens.

The other major weakness is slow disk access, particularly in loading and saving. This can be largely overcome on the PCW by using the RAM disk, as is suggested in the manual.

But a word of warning is necessary here - the RAM disk can certainly be used safely for the ScratchPad Plus command files, but in my opinion it should not be used to speed up access time with large worksheets which require the virtual memory facility.

This is far too dangerous a practice, since a mains spike or a disk full error could spell disaster.

These negative points however have to be set against the overall quality of the program, it's price and the enormous amount it has to offer on the positive side.

LIMITS

Max rows	**Disk capacity**
Max columns	**Disk capacity**
Max. no. cells in memory	**5,300 on the PCW**
Max column width	**84**
Max decimal places	**9**
Max label size per cell	**39 characters but can extend into adjacent cells**
Max formula size	**35 characters**

Let your PCW breath art into dull stationary

In the first of two articles, Bruce Hugman looks at the low-cost print option available with the PCW and some simple design skills.

IF you ever look enviously at the designer stationery used by big firms don't despair - with a PCW and a little ingenuity you can be up there with the best of them.

Whether you're simply wanting to impress your friends or more seriously to market your services as a small business, you can produce some very effective results. And with the new range of typesetting from disk services, the sky's the limit.

That awful letter template that pops up every time you use Loco Script in your first few days has a lot to answer for in limiting your imagination to one rather restrictive format.

The secret is to experiment. Use all the options - try the address in the centre, using bold, double size print, upright or italic, possibly with a space between each letter for a trading name. Or have the heading justified to the left with a line under it the width of the print area and the telephone number on the right.

For decoration use underline, rows of dashes or equals signs or graduated lines of Os in subscript, normal and double size. Figure I shows you the idea with just a couple of examples of letterheads.

In a good design, space is as important as text, so plan the general arrangement on the page and the use of space as carefully as you plan the arrangement of the characters themselves.

You'll probably be using A4 paper, so there's plenty of room to let your design breathe on the page. And when you set out a letter or an order or whatever, use the whole page - don't let your text be cramped into the top half.

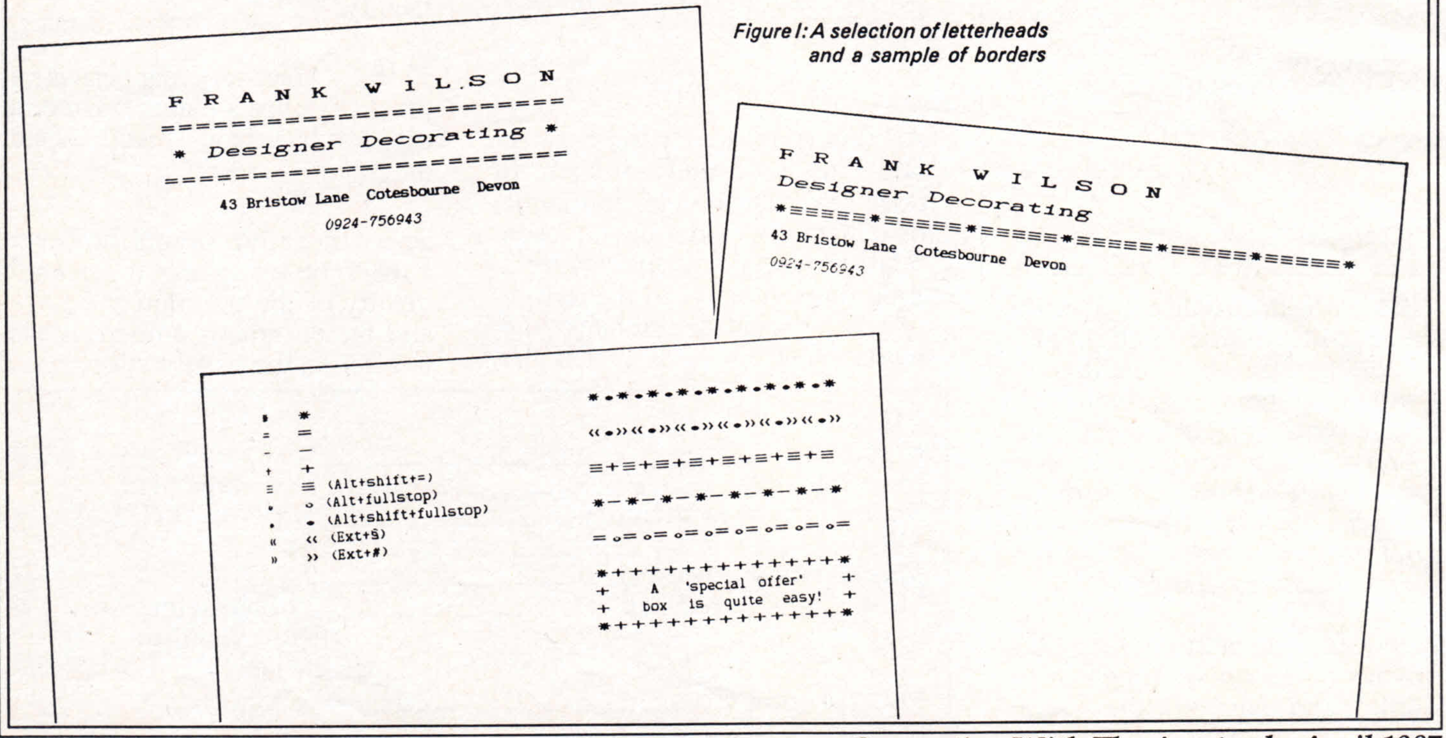
For very short communications one and a half line space is attractive with double that between paragraphs.

The address doesn't have to be at the top of the page - you can have your name or company name at the top and the address and phone number at the bottom. If you've got some of the software that turns documents through ninety degrees you can even have it down the side of the page.

Once you've found something that pleases your eye you can store it as a group template and your creation will be ready for instant use. It can also be handy to save the design as a block so you can insert it for one-off items in other groups (f7 Modes, insert text option).

If you require invoices, order forms, statements or sheets for estimates, you can develop your basic design as a new document for that particular purpose - adding "Order Number", "Terms strictly 30 days" or whatever you require.

You can store that as a group template, or keep one master



```
F R A N K   W I L S O N
========================
* Designer Decorating *
========================
43 Bristow Lane  Cotesbourne  Devon
0924-756943
```

```
F R A N K   W I L S O N
Designer Decorating
*=====*=====*=====*=====*=====*=====*=====*
43 Bristow Lane  Cotesbourne  Devon
0924-756943
```

```
*    *                                      *•*•*•*•*•*•*•*•*
=    =
-    —                                      «•»«•»«•»«•»«•»«•»
+    +
≡    ≡  (Alt+shift+=)                       ≡+≡+≡+≡+≡+≡+≡+≡
•    ₒ  (Alt+fullstop)
•    •  (Alt+shift+fullstop)                *—*—*—*—*—*—*—*
«    «  (Ext+§)
»    »  (Ext+#)                             =ₒ=ₒ=ₒ=ₒ=ₒ=ₒ=ₒ=ₒ=

                                            *+++++++++++++*
                                            +   A  'special offer'  +
                                            +   box  is  quite  easy!  +
                                            *+++++++++++++*
```

Figure I: A selection of letterheads and a sample of borders

version of it in a particular file group and make a copy each time you want to use it giving the copy an appropriate document name.

The invoice pro-forma shown in Figure II was stored as a group template, and the letter heading was pre-printed using the excellent Polyprint font generator from Arcom.

Keeping invoices in this way is particularly useful because you can run off a draft quality hard copy for your manual records if you wish. You can also rename each file - adding a single letter z for example - when the bill is paid. If your filenames include a date reference you can very quickly check which accounts are outstanding.

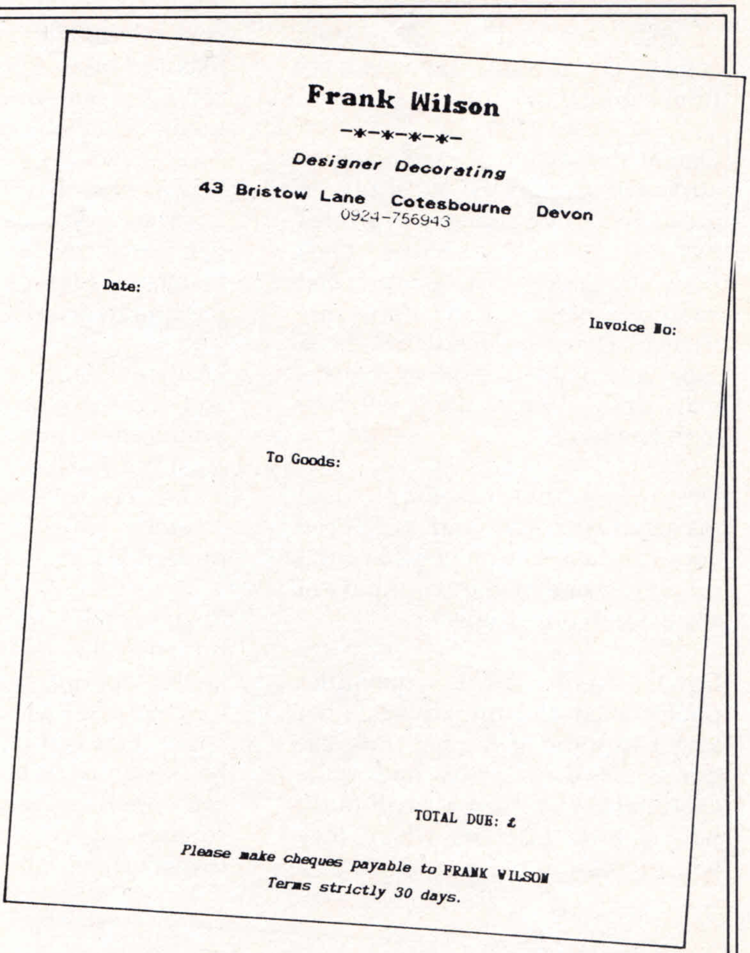
For a volume business you are likely to be using a database for invoicing but the pre-printed headed sheets may be just as useful then as databases don't generally provide good headline typefaces.

You can even produce a quite passable calling card by designing an attractive label layout. Print as many as you require (using copy and paste), perhaps on colored labels, then stick them carefully on plain or colored card with a border half a centimetre or so larger than the label. Figure III shows you the idea.

The end result of all these operations can again be greatly improved by using enhanced print software like Tasprint 8000 or Polyprint as can be seen from Figure IV.

Again you can store master copies on disk and use them as you require, with the related word processing program, or if you feel happier with LocoScript you can print a stock of sheets for future use.

Photocopying is another option - but it's usually pricy and the results are not always very clean or sharp - but the best photocopying is excellent for small quantities.



Frank Wilson
-*-*-*-*-
Designer Decorating
43 Bristow Lane Cotesbourne Devon
0924-756943

Date:

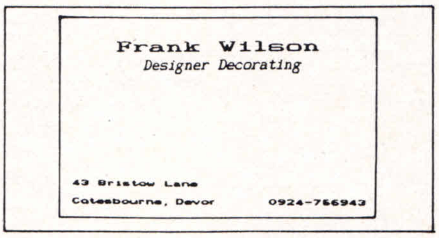
Invoice No:

To Goods:

TOTAL DUE: £

Please make cheques payable to FRANK WILSON
Terms strictly 30 days.

Figure II: A typical invoice pro forma



Frank Wilson
Designer Decorating

43 Bristow Lane
Cotesbourne, Devor 0924-756943

Figure III: You can even produce your own calling card

You can also have available the amazing luxury of pre-printed continuous stationery if you're going to be doing multiple mailings. Run the enhanced print letter/invoice/estimate head on as many sheets as you'll require, re-boot with LocoScript and re-feed your continuous pre-printed sheets when you need them.

The enhanced-print software output is pretty impressive but some typefaces are much better than others - I find Polyprint Broadway, Flash bold and Microgramma particularly effective - but they are clearly not up to print standard.

The examples show the weakness of some of the larger letters, W for instance. You may feel that your letterhead requires something a bit more polished.

The next cheap option - indeed much cheaper than buying print programs - is to set all or part of your copy in Letraset or one of the other dry-print options available from most stationers and graphic supply shops.

But it requires some practice to get a really professional effect with dry-print and all too often material set carelessly looks as though the alphabet has been shaken on to the page from a pepper-pot.

It's not that difficult to get it right. Plan the layout carefully, draw guidelines with a soft pencil and rub off the letters ensuring that every one just touches the line you can see through the backing-paper. It's important to get the proportional spacing right - an i should occupy much less space than an m but letters should not touch each other.

The higher quality versions (like Letraset) give much more visible guidance for the exact positioning of characters and the end result can be very effective as you can see from Figure V.

One of the secrets of producing an attractive design is to study in great detail every piece of printed material you come across. Look how the letters are spaced and analyze items you feel are particularly impressive. Look especially at the layout of modern magazines - they often set very high standards.

Once you've practised and mastered the art, and produced your finished version you can get it printed at one of the thousands of small, local copy-shops.

Don't have large quantities photocopied and my advice is not to got to some of the big franchise chains as their prices are quite exorbitant. You need a small outfit with a litho machine where they will process your art-work on to a paper printing plate. Prices are likely to be in the region of $35 or $50 for a thousand sheets and less for the next thousand.

When you have your stock of letterheads you can adapt them as you require as order forms, invoices, estimates and so on using the options described earlier.

Leaflets, brochures, newsletters and advertisements can all be produced using the techniques already described. If you're setting a lot of text, splitting it into columns can give it a very professional appearance.

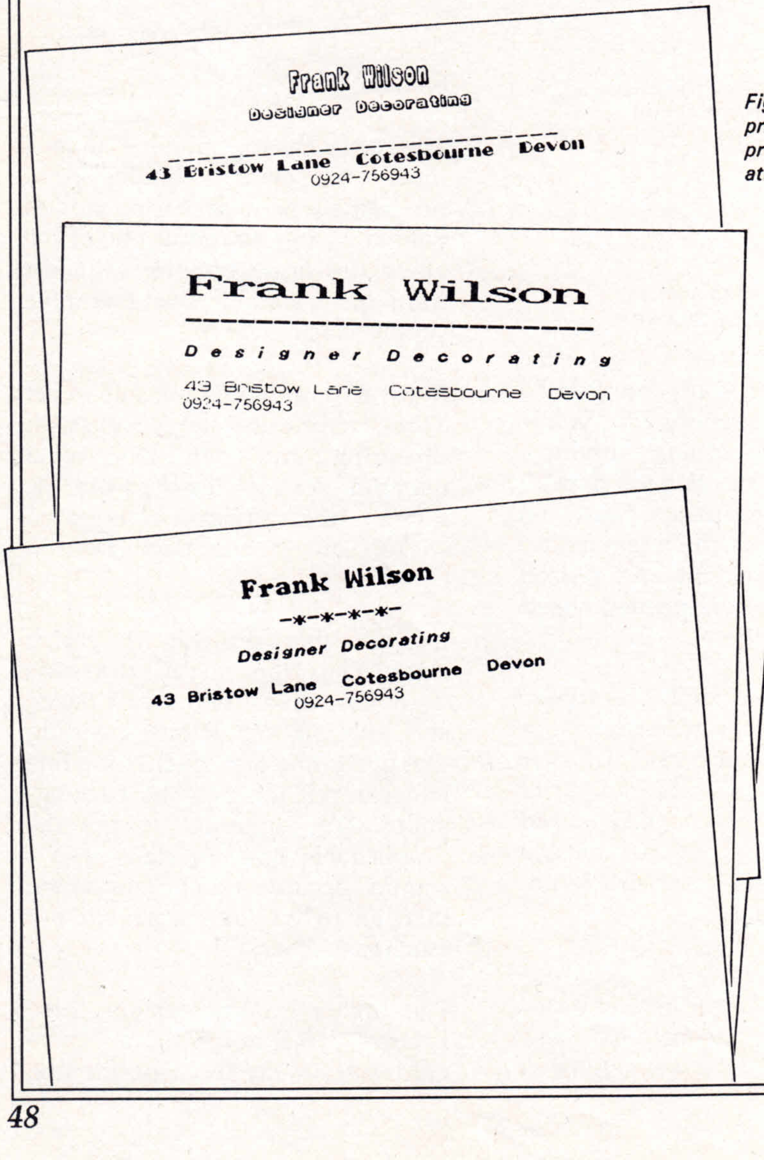
Reduce the width of text by bringing the right margin in (f2 layout option), but don't make it too narrow as the padded-out spaces between words are likely to be unsightly. With LocoScript you can produce only one justified column at a time while CPC owners have the benefit of the AMX Pagemaker to produce whole pages with more than one column at once.

The next stage is then to paste up your text (and pictures, enhanced print or Letraset headlines and so on) to make your finished art-work. Photocopying and litho plate making for printing both work on the principle of photographing an image.

If the image is completely flat and clean it doesn't matter if it's made up of a jigsaw of bits and pieces all stuck to a piece of board - as long as the board and the paper the text is on are all pure white and completely flat. Paste-up artists use aerosol adhesive which allows you to re-position material with ease.

For the highest quality results from your art-work you really need to have a photo-mechanical transfer (PMT) made - that is simply a fine photographic image of the original which is used to make the printing plate. Some printers will do that automatically, but you may not need to go to that trouble and expense.

● *That's a first glimpse of the amazing range of possibilities using the PCW as the starting point. Next month we'll be taking a look at the coming revolution which will be transforming the whole world of print and publishing in the next year or so - for international corporations and Amstrad owners alike.*

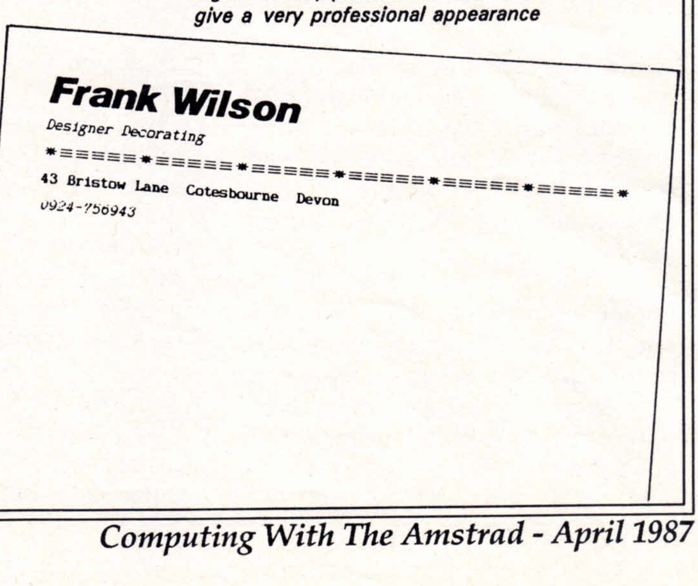


Frank Wilson
Designer Decorating
43 Bristow Lane Cotesbourne Devon
0924-756943

Frank Wilson
Designer Decorating
43 Bristow Lane Cotesbourne Devon
0924-756943

Frank Wilson
-*-*-*-*-
Designer Decorating
43 Bristow Lane Cotesbourne Devon
0924-756943

Figure IV: Enhanced print software can produce some very attractive results



Frank Wilson
Designer Decorating
==========*=====*=====*=====*=====*=====*
43 Bristow Lane Cotesbourne Devon
0924-756943

Figure V: Dry print rub-off letters can give a very professional appearance

What desktop publishing is all about

David Andrews reviews Pagemaker

I WAS delighted to get my hands on a pre-release copy of Advanced Memory System's Pagemaker. Restricted to CPC users with disk drives, it's the latest in their collection of programs designed to complement the now famous AMX Mouse - a complete icon-driven typesetting and artwork package.

It allows you to produce A4 pages of combined text and graphics which you can edit in a variety of ways, save to disk and print when required. As you can see from the examples shown here, the results can be quite spectacular. To make the product available to a much wider user base, AMS has developed it to cater for the mouseless, with the software responding to either keyboard or joystick control.

Unfortunately in order to access Pagemaker's tremendous facilities, CPC464 and 664 owners will have to acquire at least 64k of add-on ram (AMS say that both the DK'tronics and the Vortex boards are suitable).

The main Mode 2 drawing screen can be thought of as a window that covers approximately one third the height and two thirds the width of an A4 page.

If you're not using a mouse, the default device, you select your method of input with either Control + K or J, though you can change these options at any time.

Three function keys - 4(execute), 5(move) and 6(cancel) substitute for the mouse buttons when operating with either keyboard or joystick. They also remain operative when using a mouse - convenient when working from the keyboard in Text mode.

Seven menu icons are displayed in a banner across the top left-hand side of the screen, representing the different modes operative within Pagemaker.

You move the pointer to an icon, click the execute button or press function key 4 and the empty space to the right of the icons fills with a sub-menu. This takes the form of cards in a filing system and clicking on these produces a drop-down menu of additional facilities where appropriate.

In Figure 1 you can see a simplified breakdown of the menu structure, created using Pagemaker itself. At the extreme right-hand side of the banner is a small icon containing the currently selected spray-paint pattern.

Clicking on this reveals the Quick-Click window, a set of icons which enable you to bypass the menu system. More of this later.

That's the main screen in a nutshell, but hidden away within the menu structure is a wide range of facilities. The easiest way to look at these is to access each mode in turn.

The first is Filing mode, selected by clicking the pointer on the disk drive icon - it's in this mode that files can be transferred to or from disk. These can be whole pages (64k), single screens (16k), or any size of small cutout or picture. AMS has included some sample files on the disk - two cutouts and a selection from the picture library are shown in Figure II.

Further options in Filing mode allow you to load Mode 0 and 1 screens previously created using AMX Art.

The second mode, Paste, provides a variety of facilities for manipulating images on the screen. Copy, Flip (top to bottom, left to right) and Rotate (90, 180 and 270 degrees) all allow you to duplicate any section of the screen.

Stretch gives you the option of instantly doubling or halving the size of pictures, or with the slightly more involved use of a "hover box", of filing an area of any size.

A Ghost facility allows drawings to be overlapped without blanking out the image underneath, and a Scroll option enables any selected rectangular area of the screen to be moved by single-pixel steps in any direction.

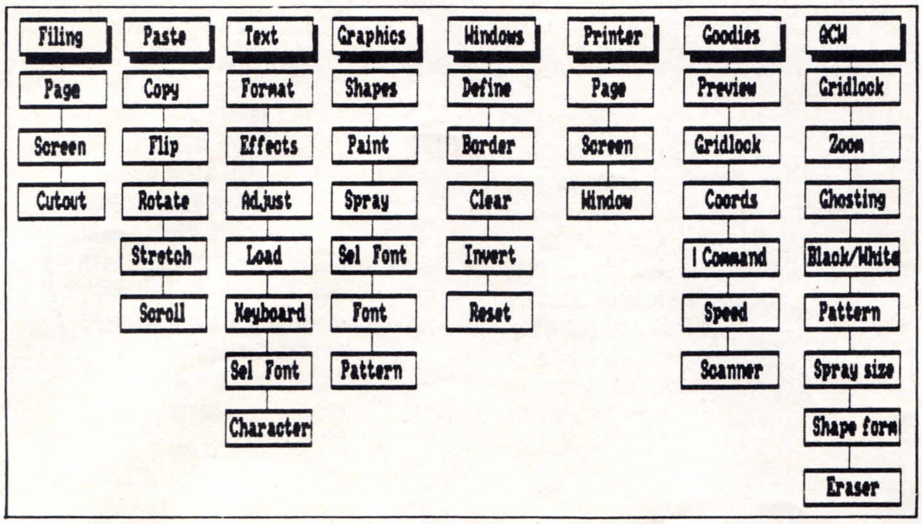


Figure I: A simplified breakdown of Pagemaker's menu system

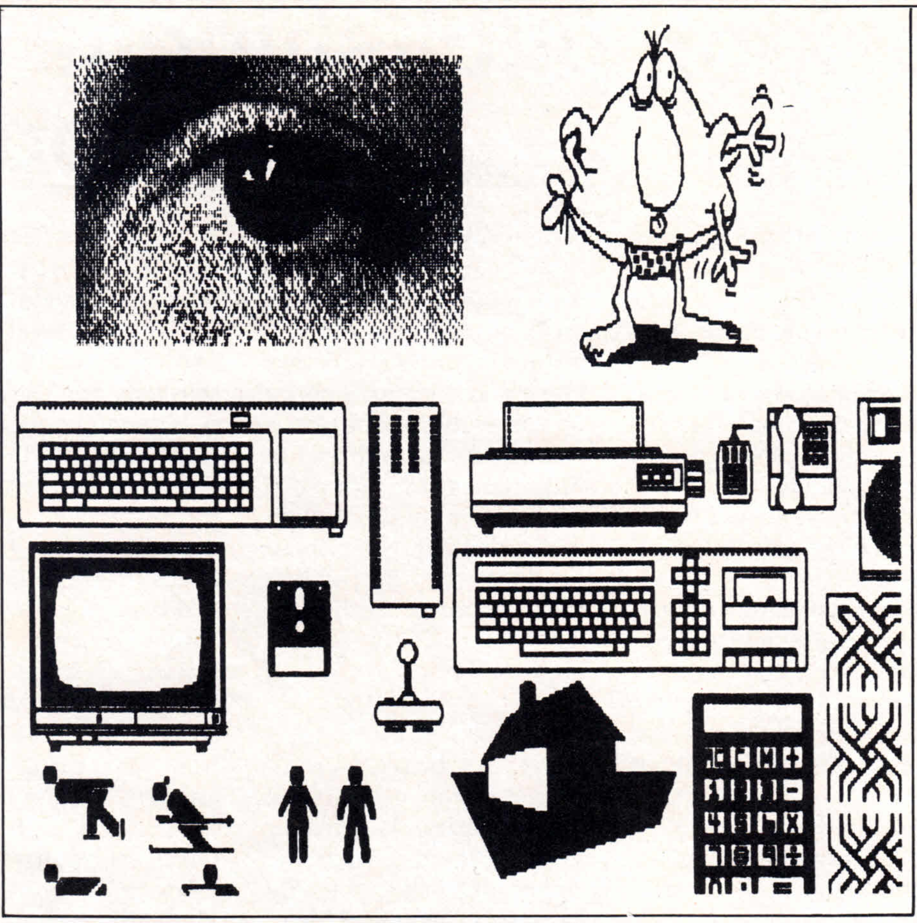


Figure II: Cutouts and some samples from the Clip Art library

Text mode is the heart of the Pagemaker system. Clicking on the Text icon produces a sub-menu of eight options for presenting your text in a versatile combination of sizes, formats and fonts.

You can elect to enter text directly from the keyboard or alternatively to load files previously saved to disk from a word processor. There's a sample Tasword file on the disk for you to experiment with, and Mini Office II Ascii files work equally well, as you can see from Figure III.

The Format facility provides a set of option which decide how your final layout will appear. The default is Literal with text printed exactly where you type it. But you can select Justify which ensures that ll lines are formatted to the same length, lining up both left and right margins. An alternative Right Justify option lines up the right-hand margin only, leaving the left margin ragged.

Selecting Word Wrap ensures that words do not split at the right-hand margin, while a very powerful Autoflow facility allows you to direct text to fill any enclosed area irrespective of its shape. Figure III also shows you the results of using these options.

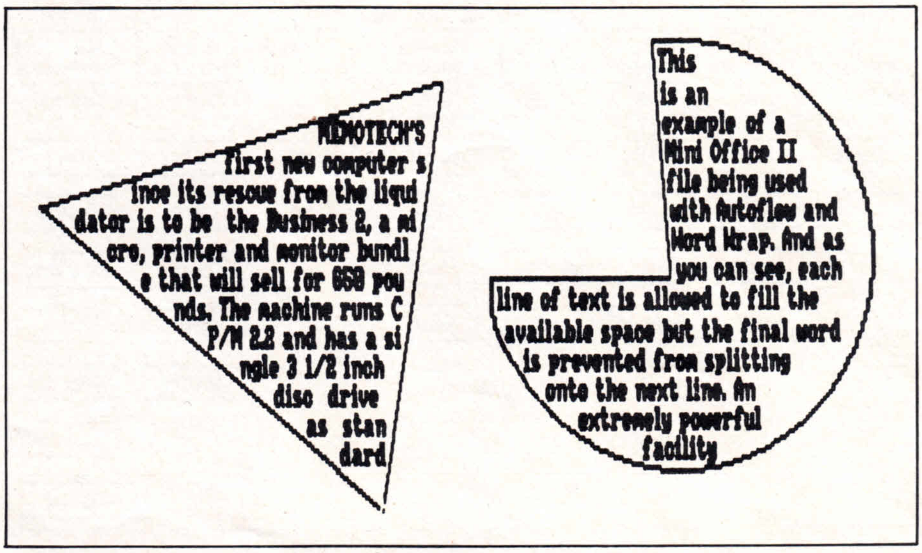
The Centre facility ensures that any subsequent output to the screen is centered to pixel accuracy between the cursor and the right margin - ideal for headlines or tables.

'If you are not satisfied with the fonts provided you can modify one or even design your own'

To provide some variety from the standard Amstrad font, the program holds in memory three alternatives accessed using the Sel Font icon. And the disk contains a further 19 fonts you can load into any of the three font stores to use when needed. You can view the currently selected font by clicking the pointer at the far right screen boundary.

If you are not satisfied with the fonts provided you can modify one or even design your own from scratch using the Definer option. This is a complete character designer in its own right, capable of producing not only letters but also patterns for the spray/paint routines and symbols for technical drawings. These can all be saved to disk for later recall.

The Adjust option is extremely flexible enabling you to increase the size of any individual



MEMOTECH'S
first new computer s
ince its rescue from the liqu
dator is to be the Business 2, a mi
cro, printer and monitor bundl
e that will sell for 650 pou
nds. The machine runs C
P/M 2.2 and has a si
ngle 3 1/2 inch
disc drive
as stan
dard

This
is an
example of a
Mind Office II
file being used
with Autoflow and
Word Wrap. And as
you can see, each
line of text is allowed to fill the
available space but the final word
is prevented from splitting
onto the next line. An
extremely powerful
facility

Figure III: Autoflow in action – first without, then with Word Wrap

letters, from a default 16x16 pixels to an enormous 128x64 - again ideal for creating headlines. Vertical and horizontal spacing can also be adjusted one pixel at a time to allow for very accurate text positioning. It can even be made negative to cater for overlapping text. You can see some of the available fonts and special effects in Figure IV.

Moving into Graphics mode provides you with facilities to produce artwork. They won't turn you into an artist overnight but with a little time and trouble you can achieve quite acceptable results.

The Shapes menu offers circles, ellipses (not implemented on the preview copy), triangles, boxes and lines. Simply click on the shape required, move the cursor to the desired position and click to set. At this point moving the cursor expands the shape to the size you require and a second lick sets it.

The Paint option allows you to fill any enclosed area with the currently selected pattern. To help you keep track of the pattern it is displayed as the Quick-Click Window icon at the top right of the Screen.

If you opt for the Spray facility the cursor becomes the nozzle of a very effective paint spray. You can use this to generate a fine mist, or alternatively a concentrated spray to paint areas of solid black or white.

The Patterns option produces a drop-down menu of facilities letting you modify the currently selected pattern. You can instantly Flip (T-B,L-R), invert, and even have at your disposal another simple yet effective character definer for editing the pattern.

It is in the Graphics mode that the Quick-Click Window comes into its own. It produces a window of 20 icons giving you immediate access to some of the more often-used facilities.

For instance there's a powerful Zoom option which instantly magnifies the small area of screen under the cursor and allows you to modify it pixel by pixel. There are icons to toggle the invisible gridlock and the ghosting facility, to select small, medium or large spray nozzles, open, filled or patterned shapes and the instant eraser - much-needed in my case.

The Windows mode also allows you to define various sized graphics or text windows, their main use being to compartmentalize a page into various columns and boxes.

The mode has three sub-menus: Clear and Invert are self-explanatory while Border gives a black-outlines window, the outline remaining on the screen once the window has been closed. While a window is active all text and graphics output is directed to it until you cancel the facility.

AMS's pride and joy is the Scanner option.

The Printer mode supports all Epson-compatibles and provides extremely versatile facilities to dump whole pages, screens, or graphics and text windows. You can select A4 or A5 output, in either draft, standard or NLQ format. The first is a high speed low quality dump ideal for checking what a page looks like; the second is a medium speed medium quality dump; the third is a slow but of very high quality.

The final option, Goodies, has a dual purpose icon that during the normal course of manipulating Pagemaker uses a small black rectangle to display the current position of the screen window relative to the page. As you scroll the page the little black rectangle moves accord-ingly so that you're constantly aware of how the current screen relates to the complete page.

When clicked the icon produces a sub-menu of seven option. The first, Preview, lets you see at a glance the contents of any completed pages saved to disk or the one currently in memory.

The gridlock option (not implemented on my preview copy) will allow you to change the step size of the invisible Gridlock grid - a system designed to assist in the correct positioning of text and drawings. When Gridlock is toggled to ON using the QCW cursor will move by the amounts preset here rather than by the single pixel default.

AMS's pride and joy is the Scanner option. With suitable equipment you can use Pagemaker's drawing screen to view digitized video input from either a camera or recorder. The image can then be frozen and the whole or any part of it saved as a cutout for illustrative purposes.

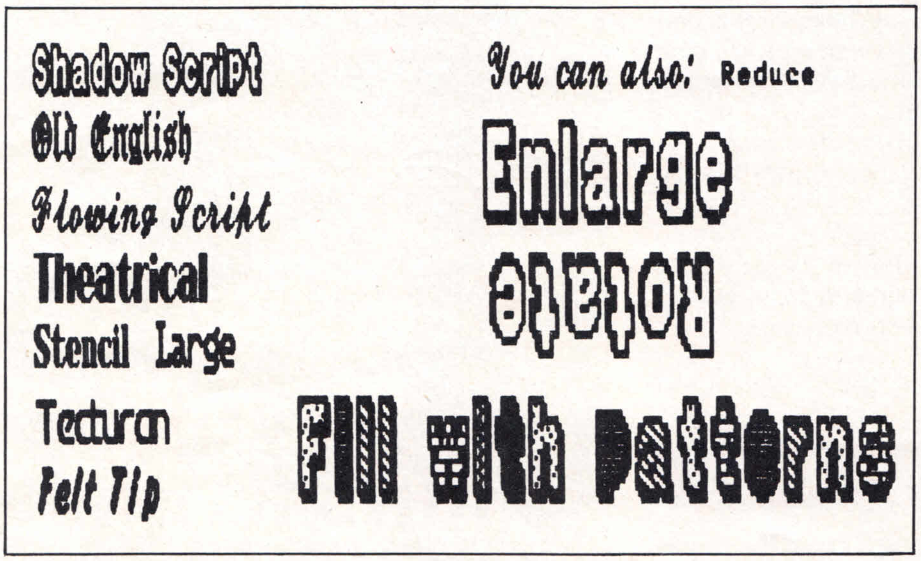


Figure IV: Some of the available fonts and special effects

I didn't have the necessary equipment (a video digitized) to test this but I did see a very impressive demonstration of its potential.

Other facilities in the sub-menu provide for increasing the speed of cursor movement (the default is slow), seeing an indication X,Y co-ordinates to help you position graphics, and entering RSX commands from within the software.

It is probably obvious by now that I really enjoyed using Pagemaker, but by the nature and size of the beast it's complex menu system can be a little daunting. Also, because of the amount of ram used by a page, the program can't be resident in memory all at once, so there's a lot of disk access and swapping between page disks and the resident system disk.

A newcomer might be put off by all this in the early stages but the technique is soon mastered. And the package is so user-friendly that the well written manual soon becomes redundant.

So who's going to benefit from such a package?

Well, anyone contemplating or currently producing low budget posters, restaurant or cafe menus, newsletters, advertisements, circulars ...things like that. A little time and effort with Pagemaker and you can have quite a high quality final product.

Beyond this, any enthusiastic home user could use the package to design a logo for a personalized letter heading, or some eye-catching handouts for church bazaars, fetes and dances.

Or why not let the kids help you design party invitations or create their own birthday or Christmas cards?

Obviously if you're going to reap the benefits from software like this you'll need access to a printer, but after that no holds are barred.

'It is probably obvious by now that I really enjoyed using Pagemaker'

Full marks then, to AMS for bringing out this first class package and for putting it within the grasp of those not fortunate enough to own a mouse. If you are restricted to the keyboard or joystick you'll still find it fun to use, although a mouse does give it that extra versatility.

Those who have already converted to rodent power and are familiar with AMX Art will adapt to Pagemaker in no time at all.

There were several irritating bugs in the pre-release version I was using but AMS have assured me that these will be ironed out by the time the production copies hit the shops.

It will then be a superb product, and I can wholeheartedly recommend it.

If you don't know what all the desktop publishing fuss is about go out and buy Pagemaker.

You won't regret it.

David Andrews

Commerce and the Church

While most lay people would not consider a church to be a business, today's clergy would almost certainly disagree. The stark reality is that to keep their clerical heads above water it is increasingly becoming necessary to adopt the latest commercial procedures. Here writer MIKE GERRARD examines the role a PCW is playing on one of the most famous churches in England, St. James's of Piccadilly, London.

NOT many rectors employ a public relations officer, but Donald Reeves is no ordinary rector, and his church, St. James's, which is just an arrow shot from the Eros statue in London's Piccadilly Circus, is no ordinary church.

Built in the 17th century to a design by Sir Christopher Wren, St. James's is a seven-days-a-week church, preaching Christianity in its widest possible sense. The forecourt, when not filled with market stalls and peace caravans, might be playing host to street theatre and clowns, while inside you'll find not only the regular church services but lunchtime concerts, talks on ecology, or meetings of the recently formed William Blake Society - the poet Blake was baptized at St. James's.

The church also boasts a coffee bar, bookshop, and even formed its own orchestra, while on the roof of his rectory (the last private dwelling on Piccadilly), Donald Reeves keeps beehives, and the honey produced, courtesy of nearby parks, has been considered good enough to be sold at Fortnum & Mason.

The newsletter listing the church's activities for just one month, The Whole Program, can be anything from four to eight A4 pages, and till recently was produced on a typwriter by the Visitors Ministry team: Sue Francis, Peter Tuffnell and Jo Berry. Until,that is, they heard the Gospel according to Alan Sugar and bought an Amstrad PCW.

Peter Tuffnell did refer to the way they decided to buy the Amstrad as "quite miraculous", which might be stretching things a little, though the coincidences were certainly remarkable.

"In addition to the newsletter", Peter explains, "we also run the bookshop, and that needed bringing into some sort of order, so although we had two main reasons for buying a computer we didn't do so simply because of the cost - we are a church, after all, and funds are always limited.

"But then we started seeing the ads for the Amstrad, which was much more affordable, and we couldn't decide whether we ought to get one or not. Then I remember saying one weekend about last February, 'Right, in three days' time we're going to make a decision, one way or the other.'

"And what happened in those three days was incredible. When Jo and I went home that evening we switched on the TV and the first word that we heard was Amstrad. Then a Welsh girl came into the bookshop and told us she was using an Amstrad to produce a newsletter and write a book about a visit she'd made to India. Someone else came into the bookshop to order something and when he saw how disorganized we were he told us we ought to get one of those new Amstrads, and then the guy from the place next door came in and raved about it too.

As no one we knew had ever mentioned the machine before, it did seem quite remarkable that suddenly everyone was talking about it and recommending it, so we went round the corner to Ryman's and bought one."

When the machine first arrived at St. James's it was put in one of the few reasonably peaceful rooms available to the Visitors Ministry team, a church tower that dated from 1684, although it's since been moved to the home of one of the team for convenience and space. None of them had any previous experience of computers, and all had to pitch in and learn how to use it from scratch.

"We could just about type," admits Sue Francis, "and that was the extent of it. We found the manual...well, let's say tricky, to be polite about it. You had to read page 250 before you could understand page three, except they don't tell you this on page three, of course. That kind of thing.

"But we found that using it was the best way to learn, and once we'd committed ourselves to producing our first edition of The Whole Program on the machine we simply had to find out how to use it.

"Just about everything had been covered by the time we got to the end of the first newsletter", Peter says, "and now the time we save is enormous. Before, my brother used to type it out for us and that would be a full day's work, sometimes more, on top of the more time-consuming task of gathering the information, checking it, making changes and so on.

"Now I do the typing in about four hours", says Sue, "and it's not just that we've halved the time on that, it's also that it's a lot less frustrating doing it. But in addition it's improved the quality of the newsletter no end".

"Yes", agrees Peter, "what we particularly like is that it allows you a great deal more creativity. Obviously you can do lots of things with the different characters, pitches, emphases and so on, that you can't do on a typewriter, but it also allows you to put a lot more thought into how your newsletter looks.

"You find yourself playing around with it to discover what it can do, which turns the work into a pleasure, and even with this playing around it still saves time. It is in fact a great deal of fun, and we find looking at the screen is a lot more inspiring than looking at a sheet of paper in the typewriter".

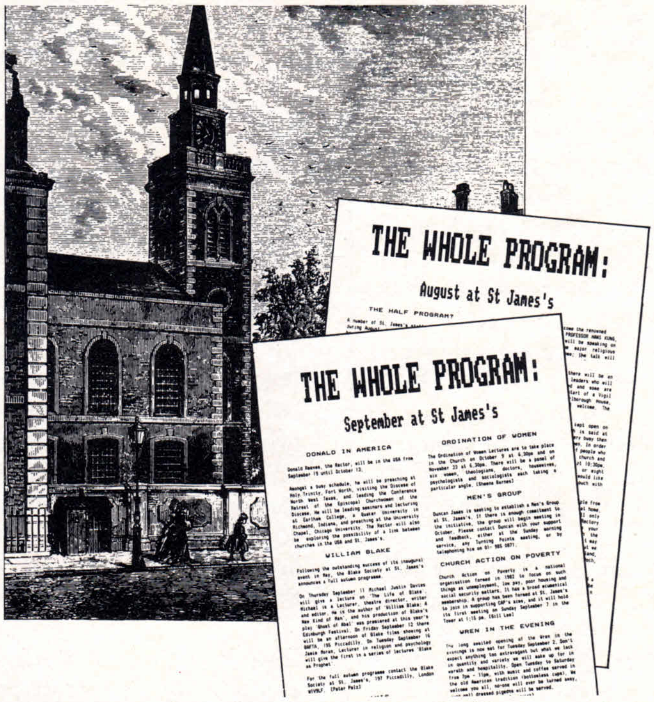
What the team would really like, though, would be the ability to work in two justified columns on screen simultaneously. Each page of The Whole Program consists of two columns of text, and a great deal of jiggling goes on to ensure that every column is the same length. Many devious tricks have been tried to produce a template that would allow them to divide each page into two justified halves, but it still eludes them.

Preparing a newsletter in this manner would also save them some cutting and pasting, as each column is printed out in NLQ then cut and pasted on to card in its finished form and passed to the printers who produce plates by photographic means and then print about 2,300 copies. Three hundred of these are sent out to a mailing list, the rest made available in the church itself.

The quality of the original is the key to the whole thing", Sue explains. "We particularly like the 15 pitch, but unfortunately our printers don't. We think it looks a lot better on the page, but it won't reproduce satisfactorily. Our mailing list is still very amateurish, and we know it. We do it through Loco-Script with a file of addresses which are printed out on to sheets of paper and then photocopied on to labels.

"Don't ask us to explain it any further, it's a bit of a Heath Robinson job, but it works, though obviously we'd like to start doing it properly. We've still only scratched the surface of everything yet, and haven't gone on to properly investigate other programs which might allow us to do even more".

"We haven't yet gone into the business functions", Peter adds, "though at the moment we're looking at Cambase to see if it will cope with the needs of our bookshop. We've had some hassles with the manual again, but it seems to be basically good.



THE WHOLE PROGRAM:

August at St James's

THE HALF PROGRAM?

THE WHOLE PROGRAM:

September at St James's

DONALD IN AMERICA

Donald Reeves, the Rector, will be in the USA from September 19 until October 13.

Amongst a busy schedule, he will be preaching at Holy Trinity, Fort Worth, visiting the Diocese of North West Texas, and leading the Conference Retreat of the Episcopal Churchwomen of the Diocese. He will be leading seminars and lecturing at Earlham College, a Quaker University in Richmond, Indiana, and preaching at the University Chapel, Chicago University. The Rector will also be exploring the possibility of a link between churches in the USA and St. James's.

WILLIAM BLAKE

Following the outstanding success of its inaugural event in May, the Blake Society at St. James's announces a full autumn programme.

On Thursday September 11 Michael Justin Davies will give a lecture on 'The Life of Blake'. Michael is a Lecturer, theatre director, writer and editor. He is the author of 'William Blake: A New Kind of Man', and his production of Blake's play 'Ghost of Abel' was premiered at this year's Edinburgh Festival. On Friday September 12 there will be an afternoon of Blake films showing at BAFTA, 195 Piccadilly. On Tuesday September 16 Jamie Moran, Lecturer in religion and psychology will give the first in a series of lectures 'Blake as Prophet'.

For the full autumn programme contact the Blake Society at St. James's, 197 Piccadilly, London W1V9LF. (Peter Pelz)

ORDINATION OF WOMEN

The Ordination of Women Lectures are to take place in the Church on October 9 at 6.30pm and on November 23 at 6.30pm. There will be a panel of six women, theologians, doctors, housewives, psychologists and sociologists each taking a particular angle. (Sheena Barnes)

MEN'S GROUP

Duncan James is seeking to establish a Men's Group at St. James's. If there is enough commitment to the initiative, the group will begin meeting in October. Please contact Duncan with your support and feedback, either at the Sunday morning service, any Turning Points meeting, or by telephoning him on 01- 985 0871.

CHURCH ACTION ON POVERTY

Church Action on Poverty is a national organisation formed in 1982 to focus on such things as unemployment, low pay, poor housing and social security matters. It has a broad ecumenical membership. A group has been formed at St. James's to join in supporting CAP's aims, and it will hold its first meeting on Sunday September 7 in the Tower at 1:15 pm. (Bill Lee)

WREN IN THE EVENING

The long awaited opening of the Wren in the evenings is now set for Tuesday September 2. Don't expect anything too extravagant but what we lack in quantity and variety we will make up for in warmth and hospitality. Open Tuesday to Saturday from 7pm - 11pm, with music and coffee served in the old American tradition (bottomless cups). We welcome you all, no-one will ever be turned away. well dressed pigeons will be served.

"We feel we have to check it thoroughly first because we carry something like 4 to 5,000 titles and if you're going to be keying all those in then you'd better make sure you're doing it into the right program. The PCW is the ideal size and price of machine for a small bookshop, and I'm sure if someone produced a software package to cater specifically for this type of business then it would do very well".

The arrival of the Amstrad hasn't been accompanied solely by songs of praise, however. The first machine they collected from Ryman's died on them after a week, and an instant replacement was made. This second machine now suffers from occasional paralysis, when the screen freezes and the cursor disappears, and it also seems wary of the Alt+Tab combination, used to produce a tab indent.

Sometimes when cutting part of the text after a tab indent the program throws a wobbly and the entire document will be printed out a letter at a time down the right-hand side of the screen. The people at St. James's are prepared to live with this, however, and are looking to the future and expanding the Amstrad's uses.

"We're very interested in networking", Peter Tuffnell says, "as the church here is a centre for so many activities, with lots of people interested in areas like the peace movement, ecology, healing and so on. It would be a great way of putting likeminded people in touch with each other.

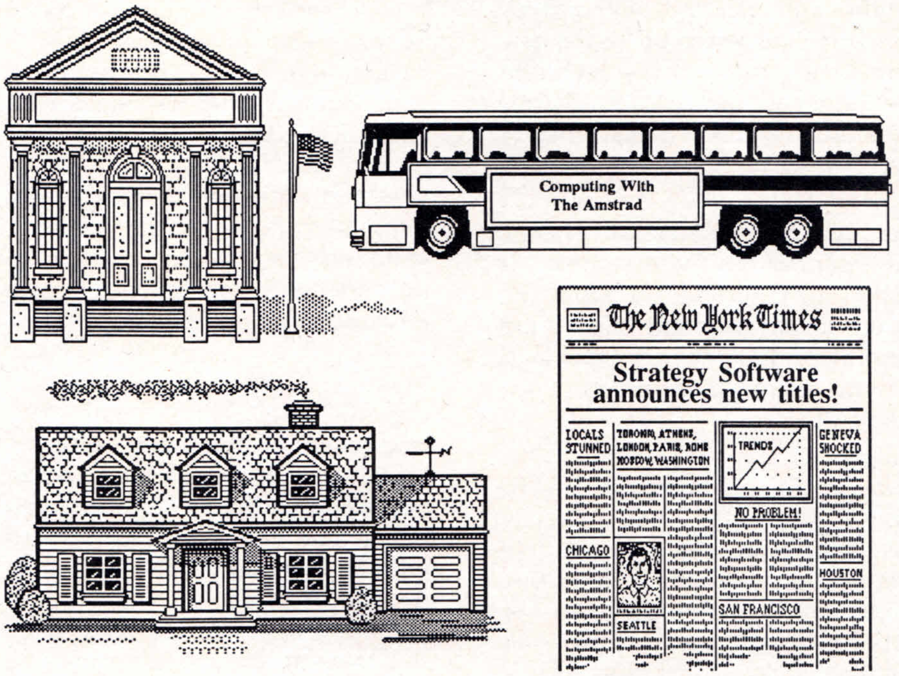
"And our dream would be to create public access to the computer. One of our jobs, in addition to the newsletter and bookshop, is to man the welcome desk, to be ready to help anyone who comes into the church for whatever reason, and believe me we do get all manner of people coming in, being right in the heart of London.

"I'd like all the information we have to be put on to the computer and be available to anyone at the press of a key. This could be advice for the homeless, on drugs, for personal problems, with lists of useful addresses and phone numbers, agencies, charities, whatever.

"There's such enormous scope. Just once in a while someone comes along in business with an idea that actually changes everyone's everyday life, and I really do think Alan Sugar has done it with the Amstrad PCW".

Hold the front page? Not really

MAX ALLEN considers what impact Fleet Street Editor will have on the desktop publishing market



THE latest entry into the desktop publishing market for the PC and compatibles, is Mirrorsoft's Fleet Street Editor. It allows you to create pages of combined text and graphics which you can format in a variety of ways, save to disk and print when required.

It is icon-drive, operating from either mouse or keyboard, and requires either a dual floppy or single floppy plus hard disk setup. It also needs a minimum 512k of ram - no problem with the Amstrad PC1512.

It is supplied as a set of four disks consisting of two system disks that differ in the screen driver supplied - for use with either IBM color graphics on the color PC (although the package doesn't use color) or the Hercules Graphics Card on the monochrome version, a font disk, and a graphics picture library disk.

When you boot up you're presented with the main screen which you can regard as a window covering approximately a quarter of an A4 page.

Displayed over the window are eight menu options, and apart from Art they all relate to processing text. When selected using the mouse pointer or the appropriate function key each produces a drop-down menu of further options and a tick against any item indicates that it is active.

Turning it off is simply a matter of moving a highlight bar to the alternative and clicking the button or pressing a key.

At the right hand edge of the screen is a set of 12 icons referred to as the Toolbox. You use one of the two letter As at the top to select either ordinary or graphics text mode.

The remaining icons are all directly connected with the graphics facilities. Alongside the icons is an elevator bar that allows you to scroll any part of the page into the screen window.

That sums up the presentation of the main screen, but its apparent simplicity conceals a wide range of facilities available to you at the press of a key or the mouse button.

The package defaults to text mode with the top letter A highlighted in the icon menu. You'll remain in this mode until such time as you select one of the Graphics options. But more of these later.

The first menu option is File. This provides a sub-menu from which you can create a new file, open an existing one, save the current file, preview pages in a reduced size, or output the current page to a printer.

The second option, Edit, provides facilities to manipulate text on the screen. When working with text you can input to the cursor directly from the keyboard and in this mode you have what is to all intents and purposes a word processor.

With the Get Text function you can also import files created using an external word processor such as WordStar, though only Ascii files can be read in, so in the case of WordStar you would have to create a non-document.

You can manipulate existing text using the Copy, Cut and Paste facilities by highlighting the section of text to be processed. To do this you place the cursor at the start of the text, then with the mouse button or key pressed, move to the end.

The section is automatically changed to inverse video and is then saved to the Edit "clip-board" for future use if required. This paste buffer acts like a temporary memory and always retains the latest block of text that you have highlighted. Only text highlighted in this way is available for any form of processing.

The Font menu displays by name all the fonts currently on the disk. You can see a sample of them in Figure II. The currently selected font can be changed by moving the highlight bar over the new name in the menu and clicking the button. You can then either type directly to the screen using the new font or, using the highlight facility mentioned earlier, change the font of text that's already there.

The Style menu is used to change the manner in which text is displayed. It provides a sub-menu of up to seven point sizes from 9(2mm caps) to 37(9mm caps), and two alternative styles of printing - bold or italic.

Once again highlighting can be used to change text already positioned on the screen. In combination with the Fonts menu some excellent results can be achieved for such things as headlines and enlarged captions.

The layout of a page determines how it will finally appear when printed, and there are several criteria to be considered here. For instance, you can choose between having your text in one, two, three or four columns. You might like it justified left or right, left and right, or centered. You may want to alter the line spacing and the distance between columns or headlines.

All this and more can be achieved using the Baseline and Align menus. Clicking on Layout in the Baseline sub-menu produces a window displaying all the information relating to the current settings and any of these can be changed to suit your own needs simply by pointing to the option, clicking the button and entering the new setting.

A very useful option in the Align sub-menu is Picturewrap. With it you can force any text to wrap round picture areas so that graphics do not obliterate adjacent text.

That covers most of the options governing text input. The one menu remaining is Art which allows you to import graphics files from disk. You can use other options on the sub-menu to cut, copy and paste, flip horizontal or vertical, and invert pen and ink colors.

When working in graphics mode existing text fades grey into a background plane and anything you draw or alter on the screen does not affect it.

It is here that the Toolbox comes into its own, providing facilities, although somewhat restricted, to produce and manipulate simple artwork.

There are line and variable-size box drawing options with four thicknesses of line to choose from. There's also a useful eraser and an outline and move option with which you can surround an area with an imaginary frame and push it around the page.

But that's it. No circle routine, no fill, not even a simple spray for texturing areas.

The graphics pictures supplied on disk with the package are superb - you can see some examples of them on page 55. But don't expect to be able to create anything like these using Fleet Street Editor itself - it's just not capable of it.

If you require work of this quality you'll have to buy it separately in the form of library disks of Clip-Art.

Alternatively, if you have an artistic bent you can use the PC's Gem Paint package or Mirrorsoft's The Art Studio to produce screen pictures which can then be changed into a form suitable for Fleet Street Editor using the Snapshot utility.

Other painting and drawing packages may be suitable but only time and experience will reveal which.

The A5 ring-bound manual is one of the best I've seen. The first section contains a novice's step-by-step guide to setting up the system that leaves nothing to chance. The second consists of a quick overview of many of the facilities and gets you using them within minutes.

There's a detailed description of all the features and menus, and finally a useful guide to planning your first publication gives you an idea of what can be achieved with some thought and patience.

It's well written and the layout makes it an extremely useful reference manual, although the efficient quick reference card included with the package soon makes the book redundant. Registering with Mirrorsoft entitles you to information on software updates, add-on packages and further hints and tips on using the software, as well as access to a free hotline for 90 days.

Fleet Street Editor will appeal to anyone contemplating or currently producing low-budget press releases, menus, newsletters, ads and the like, and it could even be put to good use creating leaflets and posters for church fetes, bazaars or dances.

Amstrad's low pricing policy is steadily bringing down the price of software for the PC, and at a shade under $375 Fleet Street, although a quality product, is rather expensive. What it does, it does well, but it could do more.

There are packages of the same ilk for the CPC and PCW machines that avoid having to resort to external software to create delightful graphics and they're pitched at a lower price level.

If you don't already have a PC you may consider the alternatives a more attractive proposition, particularly as the money saved on hardware and software would put you in the market for a good-quality printer.

If on the other hand you're already the proud owner of a PC, Fleet Street should have sufficient merit to appeal to you.

And with optional Laser printer software available, when running Fleet Street with a relatively low-cost laser printer such as the HP Laserjet or the Apple Laserwriter the quality of the finished page can approach that produced by professional equipment costing far more.

Graphics give the raw data real meaning

AS we've seen the Mini Office II package can take a lot of the drudgery out of handling figures and large amounts of data. However, it's fine to have all the numbers at your beck and call but it's no use if you can't visualize what they mean.

Here's where the Mini Office II Graphics module comes in. It's designed to take data and redisplay it as a series of graphics or diagrams so the implications can be understood at a glance.

What does it do? It can be used in one of two ways. You can either enter the data to be graphed directly from the keyboard, or load it from a file created in another module.

In either case up to three separate groups of data can be handled simultaneously. Mini Office II refers to these groups as data sets, and requests into which the data is to be placed for treatment.

If later you wish to process further data you can either use the Clear data option found on the Graphics module menu and start again, or over-write a data set by entering or loading as necessary.

The problem for the novice is that if you are not careful you can end up with an attempt to compare data in one data set with unrelated data in another.

If you do wish to use more than one data set you must ensure the compatibility of each set.

Comparing September, October and November results is no problem, but be careful if you try to compare something such as the drug dosage of various medications used for different ailments.

A final note is that it is pointless to try and graph values which are widely different in size. A data set consisting of the values 1000, 1000, 100, 10, 1, .1, .01, .001, .0001 would be impossible to accommodate given the size and resolution of the CPC screen.

We'll start by entering the data from the keyboard - the easiest way to explain.

Place the Mini Office II disk in the drive, type RUN "OFFICE", select graphics then Edit data.

You will be asked at this stage into which data set the figures you will enter are to be placed. To keep it simple at this point, in response to the prompt:

Edit Data
Data set 1, 2 or 3?

The screen will present you with two columns, the first of which, Fieldname, is for a brief description. It is pointless making this too long since it may be truncated depending on the room available on screen.

	Data set		
	1	2	3
Title	Value	Value	Value
A Cost	125	90	80
B Cost	35	60	40
C Cost	75	120	90
D Cost	60	70	80
E Cost	230	170	230
F Cost	90	140	120
G Cost	400	350	460
H Cost	250	270	270
I Cost	600	680	360
J Cost	360	230	240

Figure I: A sample data set

Do not worry about this since you will have ample opportunity later for placing all the text you may need.

The second column is where the value is entered. At this stage it is filled with zeroes, although if you had loaded data from a file the results would already be present.

Enter the data set 1 table as shown in Figure I. Use the cursor keys to move round the screen and the Delete key to correct any errors.

When you have finished press Escape and you will be returned to the Graphics menu. To protect the file immediately select Save data, to the prompt Are you Sure (Y/N)? reply Y, and to Data set 1, 2 or 3? reply 1. When you are requested for a filename, type GRAFDAT1 or any other valid name then press Enter.

An important point to make at this stage is that you must choose data set 1, since this is the only one which currently has data.

Before going any further, repeat the Edit data followed by the Save data procedures described above to load up data sets 2 and 3.

Notice how Mini Office II helps you by supplying you with the previously used Fieldnames, thereby minimizing the risk of the incompatibilities I mentioned earlier. Use the file names GRAFDAT2 and GRAFDAT3 or any others you wish.

Having done this and returned to the Graphics menu, select Bar chart and you will be presented with the area over which your data set will be graphed. Down the right hand side you will see five sketches called icons. These represent from top to bottom:

- Single bar chart.
- Multiple bar charts laid out side by side.
- Multiple bar charts laid out stacked one above the other.
- Print dump.
- Grid toggle, meaning draw a grid behind the chart.

Even though we have three data sets, we will use the single bar chart option for now, so press Enter as the cursor is located correctly on entry into this section of the module.

If you want to have a grid, move

the cursor to the grid position, press Enter and the word OFF will switch to ON.

Having selected single bar chart, you will be asked which data set you require.

Enter 1, then to the prompts 3D bar chart (y/n) and to Auto scale (y/n) reply Y and the graph will be drawn.

I suggest you always use auto scaling, since this leaves Mini Office II free to fit the graph properly on to the screen.

If you try it for yourself you may well have everything scrunched into a small part of the screen or what is worse, some unidentifiable data.

After drawing the graph you will see the prompt Text (y/n). Enter Y. To the prompt Underline (y/n) enter N and to Enter Text type in "Costs".

You will see at this point that the bottom left hand edge of your graph will have become somewhat distorted. Do not worry about this as it merely indicates the current position of the text you entered.

Use the cursor keys to reposition this area just under the Scale * 100 message. As it breaks clear of the graph the text will become legible. Once the text is in position, press Enter. It will be locked and you will be asked if there is more text to position.

If you reply N for No you will be able to use the cursor to select the print option since icon selection will become operational again.

Move the cursor to the printer symbol, press Enter and to the request Select printer type press A for a DMP1 or E for an Epson compatible and the graph as presented on the screen will be printed off. As soon as it is complete you can continue.

Before leaving the Graphics module for this month we will look at just a few of the other options. The first of these is printing off more than one data set.

While the sequence of commands is much longer than that given above, you will see that in principle it works in precisely the same way. Furthermore I have made it longer still by using more than one piece of text. Process the data you entered earlier by carrying out the following procedures.

Move the cursor to the side by side symbol and press Enter. To the prompt 2 or 3 sets of data press 3, to 3D Bar chart (y/n) press N, to Auto Scale (y/n) press Y, to Text (y/n) press Y and to Underline (y/n) press N.

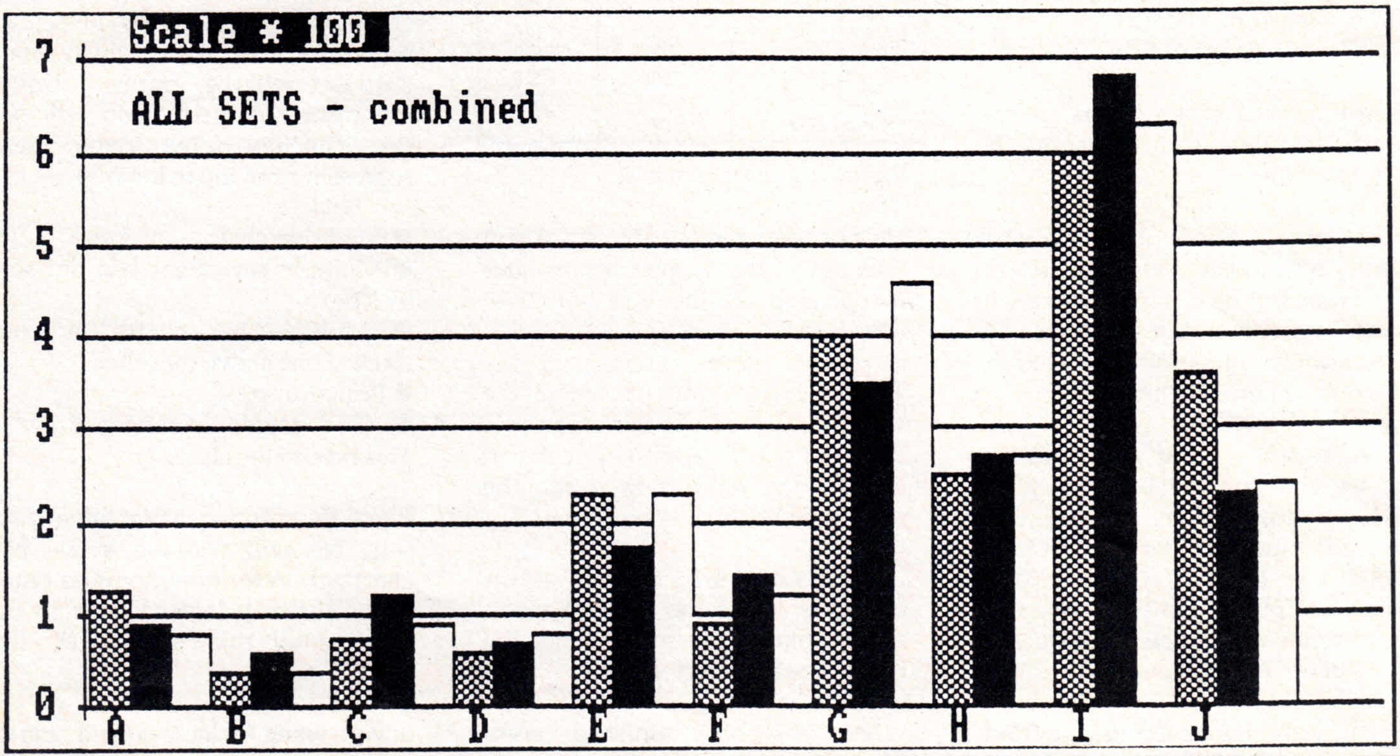
Then type "ALL SETS-combined", position the text and press Enter. To the prompt Text (y/n) press N.

Assuming that at some point you had toggled the Grid to ON, the result of these actions should look like the example shown in Figure II.

Finally for this month, on finishing the print press Escape and select Line Graph.

Note that the icons are very similar to that of the bar chart and virtually parallel to the ones we have met already. The only point to note is the third which has a plus sign. This only produces a single line, representing a cumulative total of the data sets as in the third example shown.

● *Next month we will examine the pie chart and also most importantly, how data may be transported from the Spreadsheet module of Mini Office II into the Graphics.*



AmsDOS disc commands at your fingertips in the seventh of our Amstrad quick reference charts

FILENAMES: The names of files saved on discs are in two parts. The first can be up to eight characters long, the second up to three. These parts have a fullstop, . , separating them.

Hence:

```
EXPENSES.1
TEST
A.END
```

are all valid filenames while:

```
OVERFLOWS.ONE
TEST.FOUR
```

aren't.

The last three letters of the filename are called the filetype. If you don't supply them, the Amstrad might, using the following filetypes:

.BAS Usual basic program.

.BIN Binary file.

.BAK Backup version created when you save a file with a previously used name. You can use your own filetypes. For example, AUG may suffix all files concerned with August, SEP those with September and so on.

When two drives are in use they are known as A and B. The micro looks to only one drive, known as the default drive, for a file. Unless it is changed, drive A is the default drive. To overcome this, filenames can be prefixed with A: or B: to pick the drive. A:JUNE.PAY refers to the file JUNE.PAY on the disc in drive A while B:JUNE.PAY means a file on that in drive B.

WILDCARDS: These are symbols used in place of part or all of a filename in order to select filenames of similar types or names. The question mark? stands for one character, while the asterisk * takes the place of several characters.

AMSDOS COMMANDS make use of two symbols, ¦ and @. Both are found on the key to the right of P.

TAPE OR DISC? Five commands decide whether tape or disc or a combination of both are used for input and output:

¦DISC.IN Discs to be used for input

¦DISC.OUT Discs to be used for output.

¦DISC Combines the above.

¦TAPE.IN Cassette input.

¦TAPE.OUT Cassette output.

WHICH DRIVE?

¦A Drive A default drive.

¦B Drive B default.

or you can use DRIVE with a string to pass data.

```
choice$="b"
¦DRIVE,@choice$
```

selects drive B as the default.

NEW NAME: REN is used to rename a file. This makes use of two strings to pass data. To rename the program AAAA.AAA to ZZZZ.ZZZ use:

```
past$="AAAA.AAA"
future$="ZZZZ.ZZZ"
¦REN,@future$,@past$
```

WHAT'S ON THE DISC? CAT gives you the directory.

¦DIR Gives CPM style directory.

```
name$="?.BAK"
¦DIR,@name$
```

uses a string, in this case *name$*, and a wildcard to display all the files with a name made up of a single character followed by the filetype BAK.

ERASE FILE: ¦ERA, with the filename held in a string, is used to wipe files from a disc. Wildcards can be used. To get rid of all files with the filetype DOG use:

```
destroy$="*.DOG"
¦ERA,@destroy$
```

Meet Smiley and beat the galloping Grumpies in ROLAND WADDILOVE's version of an old favourite

Introducing Smiley...

IN this colourful and compulsive game you have to guide Smiley round a maze picking up coloured buttons that are scattered about. The buttons belong to the Grumpies, who take exception to you pinching them from under their nose and chase you round the maze, trying to catch you.

If you collect all the buttons in the maze it is redrawn and the game starts all over again, except that you and the Grumpies move a little faster.

Yes, it's an Amstrad version of the old favourite – no collection is complete without it! There is a bonus which decreases with time, three lives and a keyboard or joystick option.

The program is fully structured with no GOTOs to confuse you. It is controlled by calling the various subroutines from lines 40 to 210 to print the instructions, draw the maze and move the characters.

Each subroutine has been given a title in a REM statement at the start of it so you know where to look if it does not work first time (it's nearly impossible to type in a program without making at least one mistake).

SUBROUTINES

Timer interrupt I haven't worked out how to set the clock yet so I have used Timer 1 to generate an interrupt and increment my own clock.

Bonus interrupt Timer 0 is the bonus counter, counting down to zero in 100 seconds. The bonus is printed in window 1 so as not to upset the print positions and colours of the other characters.

Initialise Dims the arrays, sets the colours and the high score. k% is the "Y" key.

Draw the maze Reads the data statements and adds 82 to the Ascii code of each character before printing.

Start positions Sets the start positions of the Grumpies and Smiley. Sets the bonus and the dots left. Reduces delays.

Set up Sets score and delays to initial values. Prints score and three Smileys.

Move man If it is not time then return. Sets the next time. Finds the new coordinates, and sets the flag (ok) if they are the same as one of the Grumpies. Prints at the new position, erases the old one. Increments score if a button has been picked up.

Move ghosts Calls ghost to move the Grumpies if it is time.

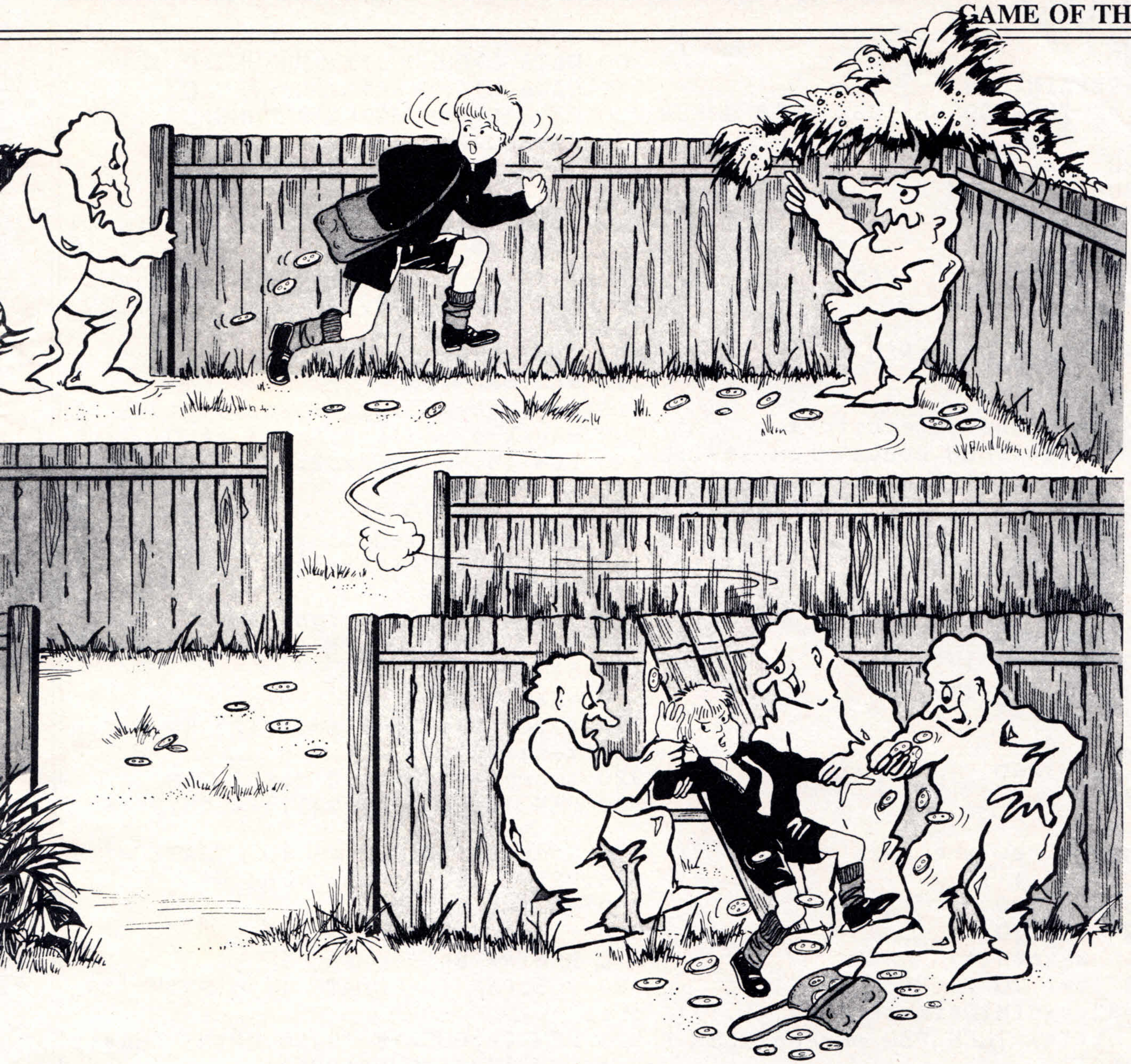
Ghost Looks where Smiley is and moves towards him if a wall is not in the way. Checks if he has been caught. Replaces button if necessary.

Caught Makes a rude noise and flashes several characters. Decreases lives.

Instructions The large title is drawn by printing it at the bottom in black, looking at the points and plotting it at the top of the screen. Prints the instructions, sets either keyboard (default setting) or joystick mode.

Next screen Prints screen completed in transparent mode. Adds the bonus to the score and increments the screen.

Game over Clears the screen by drawing black lines towards the centre. Prints scores. Asks if you want to play again.



VARIABLES

t%	Timer.
bonus%	Bonus.
maze$(30)	The maze.
ghost%(2,1)	Grumpies' positions.
gtime%(2)	Time when the Grumpies next have to move.
gdelay%(2)	Delay for the Grumpies.
i%,j%	Used in loops.
hi%	High score.
k%	General variable.
ok	A flag to show whether caught.
manx%,many%	Smiley's coordinates.
x%,y%	Temporary coordinates of Grumpies or Smiley.
score%	Score.
dots%	How many dots left.
lives	Number of lives left.
screen	Number of screen.
a%,b%,c%,d%	Values used by INKEY().
mtime%	The time Smiley can next move.
mdelay%	The delay for Smiley.

LISTING STARTS HERE:

```
10 REM ****  SMILEY  ****
20 REM *By R.A.Waddilove*
30 :
40 MODE 1:GOSUB 1400
50 MODE 0:GOSUB 300
60 WHILE k%=43
70 GOSUB 980
80 WHILE lives
90 GOSUB 400:GOSUB 810
100 EVERY 2,1 GOSUB 240
110 EVERY 50 GOSUB 270
120 WHILE ok AND dots%
130 GOSUB 1050:GOSUB 1160
```

```
140 WEND
150 t%=REMAIN(0)
160 IF ok THEN GOSUB 1660 ELSE GOSUB
 1310
170 WEND
180 GOSUB 1720
190 WEND
200 MODE 1
210 WHILE INKEY$<>"":WEND
220 END
230 :
240 t%=t%+1:RETURN :REM timer interu
pt
250 :
260 REM ** bonus interupt **
270 IF bonus% THEN bonus%=bonus%-1:L
OCATE #1,7,24:PRINT #1,bonus%:RETURN
 ELSE RETURN
280 :
290 REM ** Initialise **
300 DIM maze$(30),ghost%(2,1),gtime%
(2),gdelay%(2)
310 RESTORE 760
320 FOR i%=0 TO 15
330 READ j%:INK i%,j%
340 NEXT
350 BORDER 0
360 k%=43:hi%=0
370 RETURN
380 :
390 REM ** Draw the maze **
400 LOCATE 1,1
410 RESTORE 520
420 FOR i%=1 TO 23
430 READ maze$(i%)
440 FOR j%=1 TO 20
450 k%=82+ASC(MID$(maze$(i%),j%,1))
460 IF k%<147 THEN PEN screen ELSE P
EN screen+1
470 PRINT CHR$(k%);
480 NEXT
490 NEXT
500 RETURN
510 :
520 DATA DHHHHHHHHJDHHHHHHHHJ
530 DATA C>>>>>>>>CC>>>>>>>>C
540 DATA C>DHHHHJ>CC>DHHHHJ>C
550 DATA C>C====C>AG>C====C>C
560 DATA C>AHHHHG>>>>AHHHHG>C
570 DATA C>>>>>>>>DJ>>>>>>>>C
580 DATA C>DJ>DHJ>CC>DHJ>DJ>C
590 DATA C>CC>AHG>CC>AHG>CC>C
600 DATA C>AG>>>>>AG>>>>>AG>C
610 DATA C>>>>DHJ>>>>DHJ>>>>C
620 DATA AHHH>C=C>DJ>C=C>HHHG
630 DATA >>>>>C=C>CC>C=C>>>>>
640 DATA DHHH>C=C>AG>C=C>HHHJ
650 DATA C>>>>AHG>>>>AHG>>>>C
660 DATA C>DJ>>>>>DJ>>>>>DJ>C
670 DATA C>CC>DHJ>CC>DHJ>CC>C
680 DATA C>AG>AHG>CC>AHG>AG>C
690 DATA C>>>>>>>>AG>>>>>>>>C
700 DATA C>DHHHHJ>>>>DHHHHJ>C
710 DATA C>C====C>DJ>C====C>C
720 DATA C>AHHHHG>CC>AHHHHG>C
730 DATA C>>>>>>>>CC>>>>>>>>C
740 DATA AHHHHHHHHGAHHHHHHHHG
750 :
760 DATA 0,1,7,5,12,3,9,4,10,8,12,24
,6,11,26,18
770 :
780 DATA 19,2, 19,22, 2,22
790 :
800 REM ** Start positions **
810 RESTORE 780
820 mdelay%=mdelay%-1:mtime%=mdelay%
830 manx%=2:many%=2:ok=-1:t%=0
840 PEN 14:LOCATE 2,2:PRINT CHR$(224
)
850 FOR i%=0 TO 2
860 gdelay%(i%)=gdelay%(i%)-2*(i%+1)
:gtime%(i%)=gdelay%(i%)
870 READ x%,y%
880 ghost%(i%,0)=x%:ghost%(i%,1)=y%
890 PEN 11+i%:LOCATE x%,y%:PRINT CHR
$(225)
900 NEXT
910 maze$(2)=LEFT$(maze$(2),1)+" "+M
ID$(maze$(2),3)
920 score%=score%+10:dots%=183
930 PEN 14:LOCATE 15+lives,25:PRINT
" ";
940 bonus%=100:PEN #1,10:LOCATE #1,1
,24:PRINT #1,"Bonus:";bonus%
950 RETURN
960 :
970 REM ** Set up **
980 CLS:PEN 15:LOCATE 1,25:PRINT "Sc
ore:";0;
990 PEN 14:LOCATE 16,25:PRINT CHR$(2
24);CHR$(224);CHR$(224);
1000 gdelay%(0)=20:gdelay%(1)=40:gde
lay%(2)=80
1010 mdelay%=6:score%=0:bonus%=0:scr
een=1:lives=3
1020 RETURN
1030 :
1040 REM ** Move man **
1050 IF mtime%>t% THEN RETURN
1060 mtime%=t%+mdelay%:x%=manx%-(INK
EY(d%)>-1)+(INKEY(c%)>-1):y%=many%-(
INKEY(b%)>-1)+(INKEY(a%)>-1)
1070 IF x%=0 THEN x%=20
1080 IF x%=21 THEN x%=1
1090 IF (x%=ghost%(0,0) AND y%=ghost
%(0,1)) OR (x%=ghost%(1,0) AND y%=gh
ost%(1,1)) OR (x%=ghost%(2,0) AND y%
=ghost%(2,1)) THEN ok=0
1100 IF MID$(maze$(y%),x%,1)>">" THE
N RETURN
1110 PEN 14:LOCATE manx%,many%:PRINT
" ":LOCATE x%,y%:PRINT CHR$(224):man
x%=x%:many%=y%:IF MID$(maze$(y%),x%,
1)=" " THEN RETURN
```

```
1120 SOUND 129,30,5,15:dots%=dots%-1
:score%=score%+10:PEN 15:LOCATE 7,25
:PRINT score%;:maze$(y%)=LEFT$(maze$
(y%),x%-1)+" "+MID$(maze
$(y%),x%+1)
1130 RETURN
1140 :
1150 REM ** Move ghosts **
1160 FOR i%= 0 TO 2
1170 IF gtime%(i%)<t% THEN GOSUB 122
0
1180 NEXT
1190 RETURN
1200 :
1210 REM ** ghost **
1220 SOUND 130,370*i%+370,10,14:gtim
e%(i%)=t%+gdelay%(i%)
1230 y%=ghost%(i%,1)+(ghost%(i%,1)>m
any%)-(ghost%(i%,1)<many%):IF MID$(m
aze$(y%),ghost%(i%,0),1)>"?" THEN y%
=ghost%(i%,1)
1240 x%=ghost%(i%,0)-(ghost%(i%,0)<m
anx%)+(ghost%(i%,0)>manx%):IF MID$(m
aze$(y%),x%,1)>"?" THEN x%=ghost%(i%
,0)
1250 IF MID$(maze$(ghost%(i%,1)),gho
st%(i%,0),1)=">" THEN temp$=CHR$(144
) ELSE temp$=" "
1260 PEN screen:LOCATE ghost%(i%,0),
ghost%(i%,1):PRINT temp$:PEN 11+i%:L
OCATE x%,y%:PRINT CHR$(225):ghost%(i
%,0)=x%:ghost%(i%,1)=y%
1270 IF x%=manx% AND y%=many% THEN o
k=0
1280 RETURN
1290 :
1300 REM ** Caught **
1310 FOR i%=255 TO 128 STEP -1
1320 PEN INT(RND*15):LOCATE manx%,ma
ny%:PRINT CHR$(i%)
1330 CALL &BD19:SOUND 132,INT(RND*10
00)+1000,5,15
1340 NEXT
1350 lives=lives-1
1360 IF lives=0 THEN LOCATE 6,12:PEN
 14:PRINT CHR$(22);CHR$(1);"GAME  OV
ER";CHR$(22);CHR$(0)
1370 RETURN
1380 :
1390 REM ** Instructions **
1400 ENV 1,10,1,100,10,-1,20
1410 BORDER 0:INK 0,0:INK 1,0:INK 2,
6:INK 3,24
1420 PAPER 0:CLS:PEN 1:LOCATE 1,25:P
RINT "Smiley v The Grumpies";
1430 SOUND 1,60,1100,4,1:SOUND 2,80,
1100,2,1:SOUND 4,95,1100,0,1
1440 FOR x%=0 TO 336 STEP 2
1450 FOR y%=0 TO 16 STEP 2
1460 IF TEST(x%,y%) THEN PLOT x%+120
,366+y%*2,3:PLOT x%+120,368+y%*2,3
1470 NEXT
1480 NEXT
1490 LOCATE 1,25:PRINT SPACE$(25);
1500 INK 1,11:PEN 1:LOCATE 1,7
1510 PRINT "Guide Smiley around the
maze collecting":PRINT:PRINT"the col
oured buttons, but watch out for"
1520 PRINT:PRINT "the grumpies who t
ry to catch you..."
1530 PEN 3:PRINT:PRINT:PRINT
1540 PRINT "Press :-"
1550 PEN 1:PRINT:PRINT " K  for keyb
oard."
1560 PRINT:PRINT " J  for joystick."
1570 PEN 2:LOCATE 8,25:PRINT "< Pres
s space to start >"
1580 a%=69:b%=71:c%=39:d%=31
1590 WHILE INKEY(47)=-1
1600 IF INKEY(37)>-1 THEN LOCATE 1,1
7:PRINT CHR$(243):PRINT:PRINT " ":PE
N 3:LOCATE 1,22:PRINT "A...up   Z...
down   <...left   >...ri
ght":PEN 2:a%=69:b%=71:c%=39:d%=31
1610 IF INKEY(45)>-1 THEN :LOCATE 1,
22:PRINT SPACE$(40):LOCATE 1,17:PRIN
T " ":PRINT:PRINT CHR$(243):a%=72:b%
=73:c%=74:d%=75
1620 WEND
1630 PAPER 0
1640 RETURN
1650 :
1660 REM Next screen
1670 PEN 14:LOCATE 2,12:PRINT CHR$(2
2);CHR$(1);"Screen";screen;"complete
d";CHR$(22);CHR$(0)
1680 screen=screen+1:IF screen=10 TH
EN screen=1
1690 score%=score%+bonus:PEN 15:LOCA
TE 7,25:PRINT score%;
1700 RETURN
1710 :
1720 REM ** Game over **
1730 FOR t%=0 TO 5000:NEXT
1740 FOR x%=0 TO 320 STEP 4
1750 MOVE x%,0:DRAW x%,400,0
1760 MOVE 640-x%,0:DRAW 640-x%,400,0
1770 NEXT
1780 IF score%>hi% THEN hi%=score%
1790 PEN 14:LOCATE 1,3:PRINT STRING$
(20,"*")
1800 PEN 11:PRINT "Best score=";hi%
1810 PRINT:PRINT:PEN 12:PRINT "Your
score=";score%
1820 PRINT:PEN 14::PRINT STRING$(20,
"*")
1830 PEN 13:LOCATE 1,15:PRINT "Anoth
er game...?":k%=-1
1840 WHILE k%=-1
1850 IF INKEY(43)>-1 THEN k%=43
1860 IF INKEY(46)>-1 THEN k%=46
1870 WEND
1880 RETURN
```

Amstrad Analysis

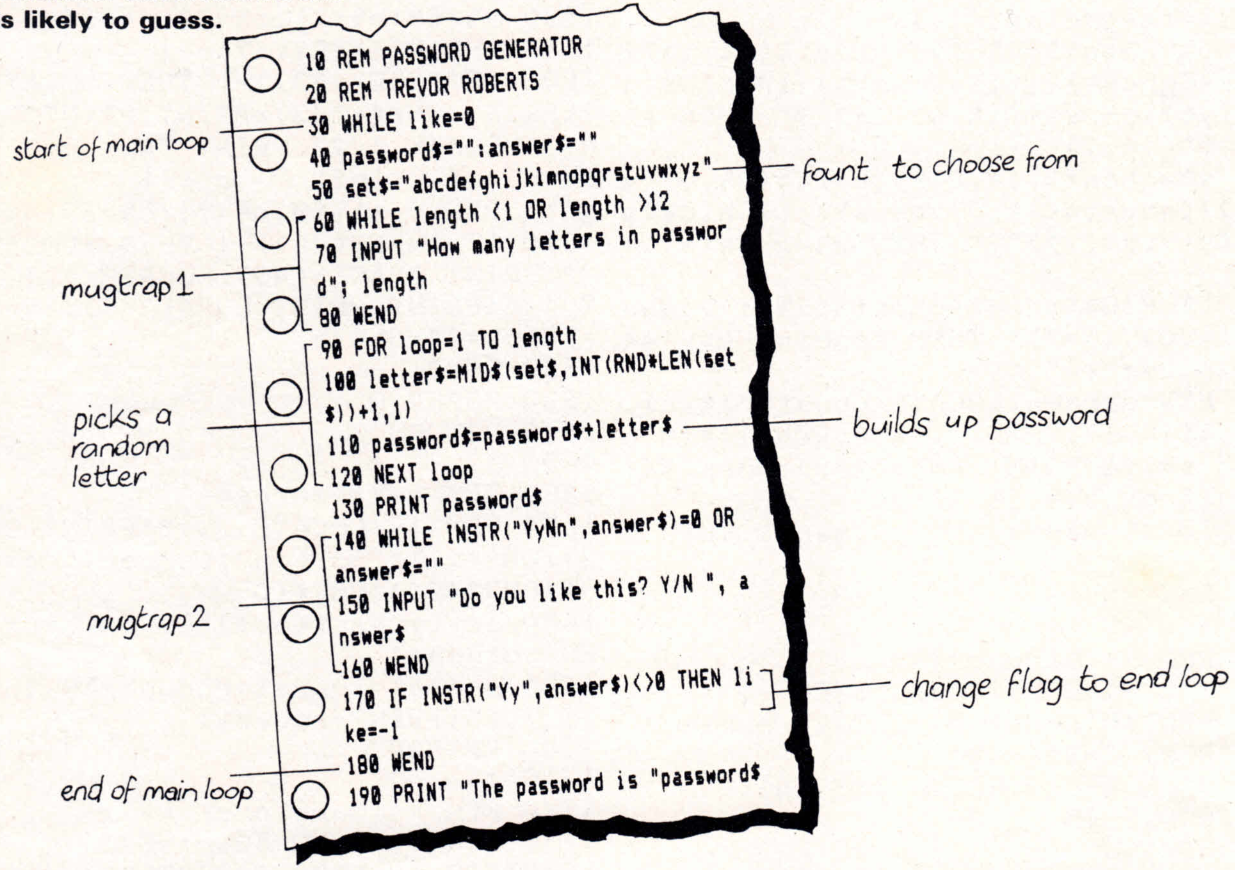
Password Generator

Analysed by Trevor Roberts

ONE of the first things you might like to try when you have created your first masterpiece is to protect it with a password routine. It sounds simple to think of a password but it can be surprisingly difficult.

You have to come up with a word that's not only memorable but also too hard for someone else to guess.

Password Generator is one way of overcoming the problem. Just run the program until you are presented with a word you can remember but which no one else is likely to guess.



```
10 REM PASSWORD GENERATOR
20 REM TREVOR ROBERTS
30 WHILE like=0
40 password$="":answer$=""
50 set$="abcdefghijklmnopqrstuvwxyz"
60 WHILE length <1 OR length >12
70 INPUT "How many letters in passwor
d"; length
80 WEND
90 FOR loop=1 TO length
100 letter$=MID$(set$,INT(RND*LEN(set
$))+1,1)
110 password$=password$+letter$
120 NEXT loop
130 PRINT password$
140 WHILE INSTR("YyNn",answer$)=0 OR
answer$=""
150 INPUT "Do you like this? Y/N ", a
nswer$
160 WEND
170 IF INSTR("Yy",answer$)<>0 THEN li
ke=-1
180 WEND
190 PRINT "The password is "password$
```

10,20 Title the listing and name the person responsible.

30-180 These lines form the major WHILE ... WEND loop of the program. Using the variable *like* as a flag, the loop keeps on cycling while *like* is false (0). Each time through a different password is produced.

40 This line initialises two variables, setting each to the null, or empty, string each time round the main loop. Leave it out and see what happens.

50 *set$* holds the complete set of letters that the program will choose from to make the password. In this case it's the alphabet but the more cryptically minded may like to use other combinations of characters.

60-80 These form a mugtrap. The user is prompted to enter the number of letters wanted in the password. The surrounding WHILE . . . WEND loop only allows the answer to lie between 1 and 12.

90-120 This FOR . . . NEXT loop cycles once for each letter of the password.

100 A mammoth line! All it does is select a letter at random from *set$* and store the result in *letter$*. Try doing the same thing with Ascii codes.

110 Each time round the loop the letter in *letter$* is added to the end of *password$*.

130 Displays the newborn password.

140-160 Another mugtrap. It asks whether the user accepts the password and stores the result in *answer$*. The conditions of the WHILE . . . WEND loop ensure that only the prompted answers are accepted.

170 If the answer is yes (Y or y) then *like* is set to −1 and the main loop comes to an end. If the answer was N or n, *like* stays the same.

190 When the program drops out of the loop the chosen password is displayed.

Mnemonics that make your code come to life

LAST month we saw that the Z80 has a special sort of memory location inside it called the A register, which can hold exactly one byte of data.

We saw how we could load it with any number in the range 0 to 255 with the instruction LD A which stands for LoaD the A register. For instance, to load the A register with &FF we would use the instruction:

LD A,&FF

The opcode for loading the A register with a number is &3E. We follow it immediately with the number we want loading into the A register. The above instruction would translate as:

3E FF

Let's have a look at a program, stored at &3000, that uses this opcode:

```
address hex code   mnemonics
3000     3E 07      LD A,7
3002     CD 5A BB   CALL &BB5A
3005     C9         RET
```

If you have Hexer, the hexadecimal loader we listed last month, you'll be able to load this into memory with the enter code option. Otherwise, you can poke the code into memory with:

```
POKE &3000,&3E
POKE &3001,&7
```

and so on.

Once you run the program - from Hexer or with a CALL&3000 - you'll be rewarded with a beep. You've loaded the A register with 7, the Ascii code for a beep, and then called the firmware routine at &BB5A which prints out the Ascii character in the A register.

Try altering the program so that it prints out various letters of the alphabet.

Notice that the address of the routine goes in lo byte followed by hi byte. That is, 5A followed by BB. This is the standard practice on the Z80.

The routine at &BB5A is, in a way, a primitive machine code equivalent of PRINT - handling only one character at a time. Let's have a look at a routine that corresponds to INPUT in a similar way.

```
address hex code   mnemonics
3000    CD 18 BB   CALL&BB18
3003    CD 5A BB   CALL &BB5A
3006    C9         RET
```

Here the routine at &BB18 waits for a key to be pressed, then stores its Ascii code in the A register. Note carefully, it's &BB18 not &BB1B. Don't mistake that last 8 for a B.

Having got the code in the A register, the program then calls the routine at &BB5A and prints the character out. If you run the program a few times you'll see that it indeed echoes onto the screen the key you've pressed.

So we've got two routines now, one to get a character into the A register and one to print it out.

We've already seen that, since the hexadecimal opcodes are a bit difficult for we humans to remember, we use mnemonics to simplify things. For example:

CALL &BB18

is far more readable than:

CD 18 BB

We can make things better still by giving our routines names, or labels as they're known. We could call the routine at &BB18 CharIn, for example, standing for Character In.

MIKE BIBBY takes the A train again in Part Four of his introduction to machine code and introduces loading bytes to and from memory locations

Similarly we could call the routine at &BB5A CharOut - no prizes for guessing why! The program above then becomes:

```
address   hex code   mnemonics
3000      CD 18 BB   CALLCharIn
3003      CD 5A BB   CALLCharOut
3006      C9           RET
```

Note that the actual code itself hasn't changed - it's still the same hex numbers. The mnemonic part, though, is much more readable.

When you've got a byte in the A register there are quite a lot of things you can do with it. Have a look at the following program:

```
address   hex code     mnemonics
3000      CD 18 BB   CALL CharIn
3003      3C         INC A
3004      CD 5A BB   CALLCharOut
3007      C9         RET
```

The instruction INC A is the interesting one. This takes the number in the A register and INCreases its value by one. Its opcode is &3C. What happpens in the above program is that the micro waits for you to press a key. It then stores the Ascii code of that character in the A register.

Next the INC A increases the

value of that Ascii number by one. Finally the character corresponding to the new, increased number is printed out on the screen.

So if you run the program and type A (Ascii code 65), a B will appear on your screen (Ascii code 66).

Why not ry altering the above program so that, having printed out the character with the INCreased Ascii code, you then INCrease the A register yet again and print that out?

Having learned that there's an INC A instruction that DECreases the A register by one. It's opcode is &3D.

It shouldn't be too hard to follow what the next program does:

```
address  hex code    mnemonics
3000     CD 18 BB    CALL CharIn
3003     3D          DEC A
3004     CD 5A BB    CALL CharOut
3007     C9          RET
```

Why don't you try using INC and DEC to give you ABC on the screen if you simply press the B key? You'll need to call CharIn, then decrease the A register, call CharOut, increase the A register and so on.

The Z80 has quite a few registers other than A, and we'll be meeting them next month. For the moment though, let's concentrate on the A register as it's rather more versatile than the others.

One thing we can use it for is to PEEK and POKE memory, You know that you can alter the contents of a memory location with POKE - that's how we loaded our first programs.

When you POKE you must specify two things:

- The address of the memory location you want altering.
- the value you wish to change it to.

Now a machine code equivalent of POKE would be:

LD (&2FF9),A

What this does is to take the number already in A and then load that number into the memory location specified.

As with POKE, we've needed two things:

- The address we wish to alter - in this case specified in brackets.
- The value we want it changing to - in this case it's supplied by the A register.

So:

LD (&2FF9),A

would read load the memory location &2FF9 with the contents of the A register. Its opcode is &32 and the hexadecimal translation of the above instruction would be:

32 F9 2F

Notice that we specify the location we want to load the A register into after the opcode in our usual lo byte, hi byte fashion.

The following program shows how we can use this idea:

```
address  hex code    mnemonics
3000     3E 20       LD A,&20
3002     32 F9 2F    LD (&2FF9),A
3005     C9          RET
```

If you run the program, then use Hexer to examine memory location &2FF9, you'll find that it has had &20 stored in it.

Just as we can POKE with the A register, so we can PEEK. The instruction:

LD A,(&2FF8)

will load the A register with the contents of memory location &2FF8, effectively PEEKing that location. It's opcode is &3A and the above instruction would translate into:

3A F8 2F

Notice once again the brackets around the memory location in the mnemonic. These are important - they mean that the instruction refers to the contents of the memory location specified within those brackets.

Here's a program that uses both peeking and poking:

```
address  hex code    mnemonics
3000     3A F8 2F    LD A,(&2FF8)
3003     32 F9 2F    LD (&2FF9),A
3006     C9          RET
```

This simply loads the contents of memory location &2FF8 into the A register, then loads memory location &2FF9 with the contents of the A register.

Use Hexer to enter the code into your micro, then enter two different numbers into locations &2FF8 and &2FF9.

Run your code, then use the Examine option - you'll find that the contents of &2FF9 have become the same as those of &2FF8.

Incidentally, this is why our Examine option defaults to &2FF8 and not &3000 where all our code has started. I'm going to use the locations &2FF8 to &2FFF to POKE the results of our machine code into, for examination after we've run our programs.

To use the jargon, I've reserved these bytes as program workspace.

So far we've used the INC instruction to increase the number in the A register by one. There is another way to do this:

```
address  hex code    mnemonics
3000     CD 18 BB    CALL CharIn
3003     C6 01       ADD A,1
3005     CD 5A BB    CALL CharOut
3008     C9          RET
```

The mnemonic ADD A, as its names implies, adds the single byte following the comma to the A register. The opcode for this is &C6. In fact the above program simply waits for a key press then prints out the character with the next higher Ascii code.

Notice that it takes one more byte than the version which just used INC A.

The advantage with ADD A, is

that we aren't restricted to adding one to the A register. We can add any number we want in the range 0 to 255. Try altering the program so you add, say, 2 to the Ascii value of the key pressed and so on.

We can use our poking techniques to store the results of our addition sums:

```
address   hex code    mnemonics
3000      3E 02       LD A,2
3002      C6 02       ADD A,2
3004      32 F8 2F    LD (&2FF8),A
3007      C9          RET
```

Use Hexer's examine option on location &2FF8 to see if the micro knows its sums. If you stick to small numbers you'll find it does. However, bigger numbers can cause problems, as the following shows:

```
address   hex code    mnemonics
3000      3E FF       LD A,&FF
3002      C6 01       ADD A,1
3004      32 F8 2F    LD A,1
3007      C9          RET
```

What we've done is to load the A register with 255 (&FF) and then add 1 to it. However if you run the code and then examine &2FF8, you'll find the value 0 stored there. What's going on?

The problem lies in the fact that the A register can only hold a single byte, which restricts it to numbers in the range 0 to 255. The answer to our sum is 256, which is too big to be stored in a single byte. So what the Z80 does is to start again from zero once it goes past 255.

So, as far as the A register's concerned:

255+1 = 0
255+2 = 1
255+3 = 2

and so on.

It's a bit like a car's odometer - once you go past 999,999 you start again at 0 .

I always visualize the numbers laid out in a circle, as in Figure I. Adding numbers takes you clockwise round the circle, subtracting numbers takes you anticlockwise.

When you're subtracting, things go awry when you get an answer less than 0. This time, as you'll see from Figure I, "going past" 0 gives us 255, not minus 1.

Thus, as far as the A register is concerned:

0 - 1 = 255
0 - 2 = 254

and so on.

The following program shows what happens:

```
address   hex code    mnemonics
3000      3E 00       LD A,0
3002      D6 01       SUB A,01
3004      32 F8 2F    LD (&2FF8),A
3007      C9          RET
```

We've used a new instruction, SUB A, which allows us to subtract the byte immediately following its opcode from the A register. Its opcode is &D6, so to subtract &7F from the A register, the hex code would be:

D6 7F

The previous program attempts to take 1 away from zero. If you run the code and then examine &2FF8, where the result of the subtraction is stored, you'll see that it has the value 255 (&FF).

You're going to get some pretty alarming answers to your sums unless you take this vehavious into account. Fortunately the Z80 has a warning signal to tell you something odd has occurred. It's called the carry flag. (You'll see exactly why in a later article).

Flags are very important to machine code programmers. We use them to check the state of our program - in much the same way as a doctor uses your pulse, blood pressure and so on as aids to his diagnosis.

However the numbers associated with flags don't vary as much as pulses and temperatures. A flag can have only two values: 0 and 1. (a nice two-state, or binary, system!)

When a flag has the value 1 we say that it is set. When it has the value 0 we say that it is clear.

The Z80 has several of these flags to help us keep track of what is going on, but for the moment we'll concentrate on teh carry flag. Now the Z80 uses the carry flag to warn us when our sums are going awry in the manner we've just seen. That is, when we're:

- Using ADD A and getting a result bigger than 255.
- Or using SUB A and getting a result less than 0.

In both of these cases the Z80 automatically sets the carry flag - that is, it gives the carry flag the value 1.

In fact every time you do an ADD A or a SUB A the Z80 adjusts the carry flag. If the sum has stayed "within bounds" - that is, hasn't gone over 255 or under 0, the carry flag is given the value 0. And, as we've said, if it does infringe the limits, the carry flag is set.

Then, depending on the condition of the flag, we can take appropriate action.

We use them a bit like Basic's IF statement. For example, IF the flag is not set THEN do this and so on.

However we don't tend to look directly at the flags and then decide what to do - we use instructions that automatically take the states of the flags into account.

To illustrate the use of flags we'll have to jump ahead a bit (no pun intended). Have a look at the following piece of code:

```
address   hex code    mnemonics
 3000     C3 00 30    JP &3000
```

Now this is an exceptionally silly piece of code. In fact it's the machine code equivalent of:

10 GOTO 10

except that if you run the code you'll have to reset your machine to get out of the endless loop.

You see, the JP opcode (&C3) tells the micro to continue the program from the address given in the two bytes immediately after that opcode (in lo byte, hi byte order). If you like, it tells the Z80 to JumP to that address and carry on from there. So:

C3 00 30

would tell the micro to go and do the program starting at memory location &3000. However, if you look at the program you'll see that it itself is stored at &3000. So it goes and does itself again. Then of course it goes and does itself again - and so on ad infinitum.

This type of jump is called an unconditional jump. That is, as soon as the micro gets to the jump instruction it does it.

There are ways, though, you can give the micro a "choice". The Z80 has a set of confitional jumps. Whether or not the micro does the jump depends on the state of the flags.

The one we're interested in at the moment is the JP NC instruction (opcode &D2).

JP NC stands for JumP if Not Carry. That is, jump to the address specified if the carry flag is not set, that is, it's clear. If the carry flag is set, however, the micro doesn't perform the jump - it simply continues, performing the next instruction.

We can use this instruction to create a loop, as the next program illustrates. It's our most complex program to date, so we'll examine it very carefully.

```
address  hex code    mnemonics
3000     3E 20       LD A,&20
3002     CD 5A BB    CALL CharOut
3005     C6 01       ADD A,1
3007     D2 02 30    JP NC,&3002
300A     C9          RET
```

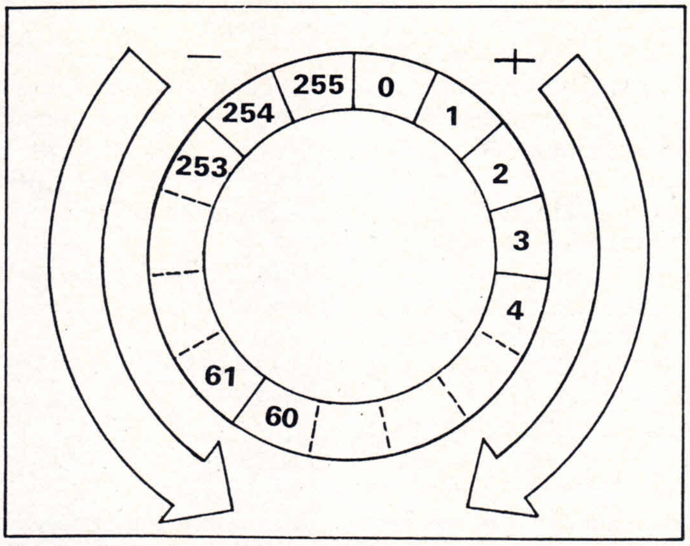


Figure 1: How the numbers cycle in a byte

The first thing we do is to load the A register with 32 (&20) - the Ascii code for a space. Then we call the CharOut routine to print the character in the A register on the screen - in this case simply a space.

The CharOut routine doesn't affect the value in the A register so this will still be 32 after the space has been printed. We then add one to the A register.

Now we come to the clever bit. The next instruction is:

JP NC,&3002

That is, if the carry flag isn't set, jump back to address &3002 and continue the program from there.

The carry flag won't be set, it will be clear, since we haven't gone over 255 in adding one to 32. So the program will jump to *3003 and do the instruction there.

As the instruction at this address calls the CharOut routine, the effect of taking the jump will be to print out the character whose Ascii code is 33(32 + 1), the contents of the A register.

The number in the A register is then increased to become 34 and, as this still doesn't set the flag, we again take the jump. The character with Ascii code 34 is then printed out, the number in the A register increased to 35, the jump again taken, and so on.

In fact this will continue looping round like this, printing our characters with increasing Ascii values until we have finally reached the character with Ascii value 255. Having printed it our, we add one to it, which should give us 256. As we've seen, the A register can't hold this. In fact it becomes 0. However the Z80 warns us by setting the carry flag.

This time, therefore, when we come to the jump we don't take it, because the carry flag is set. We "drop through" to the next instruction which, in this case, is RET.

The effect of this program is to print out all the characters with Ascii codes from 32 to 255. Incidentally, I started at 32 because the numbers below this are control codes, and can cause some rather funny effects.

Now you might like to try tidying the program up by making the first instruction a call to a routine to clear the screen (&BB6C). That's fine, but remember, if you do so, to alter tha address you're jumping to - the call to CharOut won't be at &3002 any more.

Just in case you're wondering if you can use INC A instead of ADD A,1 - you can't. Irritatingly INC A doesn't alter the carry flag as does ADD A.

Well that's all for this month. Next month we'll look at the other registers.

When the press use such words as 'Phenomenal', 'Outstanding', 'Ideal' and 'Worth Every Penny', they've obviously discovered something rather special.

But when that something special turns out to be a product in which they are already expert, then it must be something very special indeed.

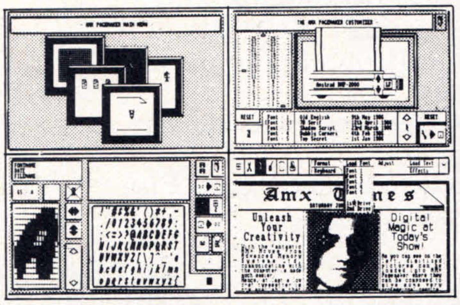
The object of their enthusiasm is AMX Pagemaker – a revolutionary software program that will produce newspapers, posters, leaflets, notices and hand-outs – in fact anything where text and graphics are required, to an extraordinary professional standard.

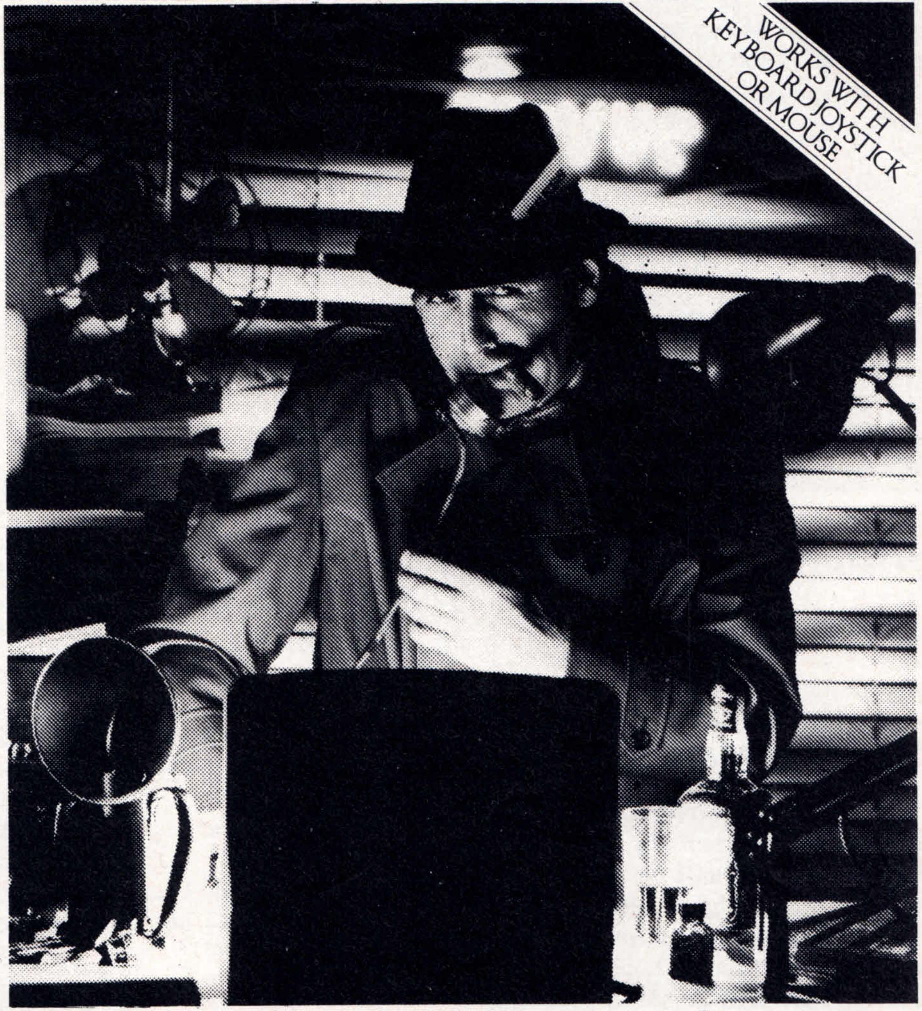
It's a complete graphics design system and word processor rolled into one. It has real time graphics with fast continuous scrolling up and down an A4 page and uses Mode 2, the highest graphics resolution on the Amstrad CPC computers.

READ ALL ABOUT IT.

You can type directly onto the screen, with any of the 16 typefaces supplied or design your own, alternatively, you can load in any ASCII file or a word processor file, from programs such as Tasword, Amsword, Maxam, or Protext, with fully automatic on-screen text formatting during loading.

'Word processing' facilities such as centering, ragged right and literal justification are all available. There is full pixel resolution control over text and graphics. Also included is a micro spacing facility.





The program that's making front page news.

EXTRA, EXTRA.

There are outstanding facilities for drawing, spraying and painting, using either the patterns supplied, or your own pattern designs. A screen conversion routine is included allowing screens created in Mode I and O to be used within the Pagemaker. The cut and paste facilities include copying, moving, rotating, stretching and a fantastic zoom is also available.

The previewer allows you to view three A4 pages at any time before work is output to a wide range of dot matrix printers including: Amstrad DMP 1000-2000, Epson FX/RX/LX/LQ, Canon PW-1080, Kaga KP810, Mannesman Tally MT-80+, Seikosha SP-1000A, Star Delta, Star SG10 and any that are compatible with the above.

The AMX Pagemaker requires: a) Amstrad CPC618 or b) Amstrad CPC664+64K Minimum add-on Ram or c) Amstrad CPC464+64K Minimum add-on Ram + disc drive, DK 'tronics Ram boards or compatible.

Let's leave the last word to the press.

*"Pagemaker" is phenomenal – it lends itself to creating anything where text and graphics are involved – notices, posters, leaflets, hand-outs, newsheets. Packages like this have been the province of the 16-bit micros until now, this product is worth every penny of £49.95."

AMX MAGAZINE MAKER– WE THOUGHT IT WAS ABOUT TIME WE PUT YOU IN THE PICTURE.

A combination of AMX Pagemaker and the AMX video digitiser. Using any video that provides a composite signal and the digitiser, images from a camera or TV can be converted into a graphic screen on the Amstrad Micro. They can then be used within AMX Pagemaker to illustrate magazines or newsletters. The digitiser connects into the expansion port and scans a complete picture in only 5 seconds.

A special print dump routine is also included with the driver programs. This is specially designed to produce fast, correctly proportioned pictures, with reduced 'Contouring' resulting in a very accurate reproduction of the image.

Features offered by this package include:

- Dot resolution 256 by 256
- Standard 1 volt composite video input
- 10 bit A/D convertor gives 32 grey scale output
- Low IC count
- Contrast and brightness control
- No external power unit required

These packages are your opportunity to join the desktop publishing revolution.

The AMX Pagemaker costs only $125.00 software is supplied on 3" disc and a fully illustrated operating manual, AMX Digitiser only $225.00 including software on 3" disc, and AMX Magazine Maker (including AMX Pagemaker and AMX Digitiser) at any $325.00

Now available direct from Strategy Software at U.K. prices!

CAT #	PRODUCT
4501	AMX MOUSE $175.00
4502	AMX PAGEMAKER $125.00
4503	AMX DIGITIZER $225.00
4504	MAGAZINE MAKER $325.00
4505	AMX MAX $49.95

Please use order form on Page 72

BANKCARD, MASTERCARD OR VISA ORDERS CALL 008 29 4377

This is the first in what we hope will be a regular section of Computing With The Amstrad. We hope to feature interesting stories and news events on Australian and New Zealand user groups in a similar way to our first story this month.If you have anything of interest to tell us please write - include photographs where possible or just send a copy of your club's newsletter highlighting the item you'd like featured.We don't intend to list user groups, meeting times etc. so please don't send this information unless there is a news item to go with it.We should also point out that user groups can qualify for discounts on products carried by Strategy Software, including this magazine. Only club office bearers should make enquiries please - not 12 year olds!

COMPUTER WORKSHOP IN HOBART

In December the Amstrad Club of Southern Tasmania held a one day workshop for people with disabilities. What was the background to this activity?

Persons with disabilities have the same needs and desires as other members of the community. However, these needs will still be very individual. Many people with disabilities have been asking questions about the potential for microcomputers to help them in their daily living, their hobbies and their work. For example, some of these people might find it difficult to write legibly, others may keep records for a club while others may be finding too much time on their hands.

These ideas combined with my interests in gadgets generally and my latest experiences with my newest 'toy' - an Amstrad Computer. I was also enthused to do something more positive when I was invited to participate in a Conference on Computers and Technology for the Disabled in Los Angeles last October. It seems that people with disabilities have less access than others to new technology to experiment and to try things out while not under any pressure.

So, through the Amstrad Club a general invitation was circulated to groups catering for people with disabilities and to individuals with a disability. Twelve people took advantage of the offer and this included a couple of deaf people, a couple with muscular dystrophy, a person with a visual problem, several with physical difficulties and several with an intellectual disability. It was a mixed group. All participants were adults and, in fact, that was a criteria for registration.

The organization was simplicity itself. Members of the Amstrad Club were asked to participate as tutors or facilitators - participation meant attendance for all or most of the day plus supply of one's computer and a variety of programs. This request was taken up by about nine or ten members including six youngsters (15 year olds).

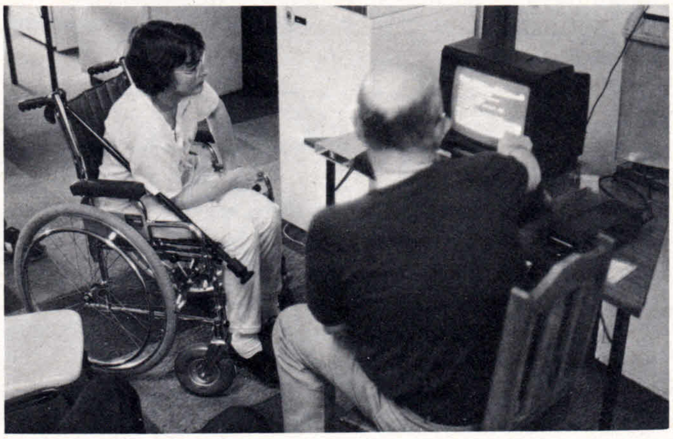
The planned program was to offer a wide variety of programs and an essential ingredient was long periods of "hands on". The plan worked.

We began the day with very little talk and insisted on no formality with "first names" for everyone. We started at 10.00 am and finished at 4.00 pm. There were eight computers available all day for the 12 key people. It must be said that two or three participants brought along a partner and so we had about two dozen people in the area during the workshop.

The general format was for a program to be introduced to the whole group e.g. a game involved with moving a balloon over a mountain and landing it on the other side. After a brief demo each person went back to their helper for the day and tried that game (program) or similar. Of course, games appealed to a number of participants but all types of programs were demonstrated - adventure games

- traditional games (chess, draughts ...)
- arcade games
- word games (crosswords)
- word processor
- database (family tree, club records)
- accounts
- miscellaneous fun like biorhythms, dates ...

We were able to demonstrate the potential of bulletin boards and use of a modem.



There was no recognition of disability or grouping. There was a need to offer sign language for the deaf participants and the person with a sight difficulty needed to get very close to the screen for the demonstration.

As the day proceeded, it became clear that a preference was being sought for certain types or groups of programs by most people. However many of the people recognised that we all can use a computer for recreation and for more formal work.

There are some observations to be made. The first must be that a computer club can offer a service to people outside the club and learn from the experience. The youngsters especially, coped with the necessity to interact with people in a situation quite new to them. The idea of a 'workshop' offered in an adult style to some people with disabilities was appreciated by those participants. The Amstrad as a computer is most admirable for the above tasks and the programs available fill all needs. The cost too is attractive.

As the organizer I am convinced that the home computer will offer a great deal to persons with disabilities in the same way that the computer is being so useful to other members of the community. I also believe that these kinds of workshops are not beyond the scope of most clubs and can be a very worthwhile activity.

It also seems to me that as a recreation tool it can be quite creative and can extend thinking activities for those people who are confined - the television can be quite mindless as a contrast. One of the participants at the workshop was particularly taken with the word games ... and then there can be chess, draughts, Othello, and all the adventure games.

The workshop finished at 4.00 pm. Most participants were still there and were asking about the date for the next workshop. To be honest, I was exhausted.

JOHN THORNE
John is Principal of the Lady Rowallan Centre for Hearing Impaired Children and Children with Language Disorders. He is also Chairman of the Disability Advisory Council of Tasmania.

Sydney Amstrad Computer Club

We still have a permanent meeting venue in the Newtown Area and our committee members will advise it's location for our meetings held on the 1st Saturday of the month and the programmers meeting on the 3rd Saturday of the month. Both meetings commence at 2-30 pm and the library opens at 2-00 pm. We have approx. Eighty (80) publications for financial members to borrow for a month at a time (free of charge) , at the beginning of 1987, and this number of books will increase as new publications are released.

We have found that members able to borrow books are in a better position to evaluate them before purchasing them for their own use, where before they were reluctant to buy books as they usually were not of the type they required at the time.

Kevin, would you please include us in your files and publish the information you see fit in your magazine "COMPUTING with the AMSTRAD" when you include an Australian User Club section in it. We are associated with the PCW AUSTRALIA GROUP but not with AM-USER's, and I direct country people and those with PCW's to their closest club.

Reed Walters
Honorary Secretary
[02] 5609487

**Public Domain Disk
Volume 1**

NuSweep file manipulation program, Chequebook 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 software 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 ORDER FORM

SOFTWARE ON TAPE

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)	$59.95
2401	MT-BASIC (CPC)	$125.00
2402	MT-BASIC (PCW)	$125.00

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
2805	PD VOL. 5	$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	$45.00
5002	MAGAZINE + TAPE	$90.00
5003	MAGAZINE + QUARTERLY DISK	$110.00

BACK ISSUES

CAT #	TITLE	PRICE
6008	CWTA PREMIERE EDITION	$4.95
6009	CWTA SEPTEMBER ISSUE	$4.95
6010	CWTA OCTOBER ISSUE	$4.95
6011	CWTA NOVEMBER ISSUE	$4.95
6012	CWTA DECEMBER ISSUE	$4.95
6101	CWTA JANUARY ISSUE	$4.95
6102	CWTA FEBRUARY ISSUE	$4.95
6103	CWTA MARCH ISSUE	$4.95

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
7103	SMILEY	(3/87)	$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.

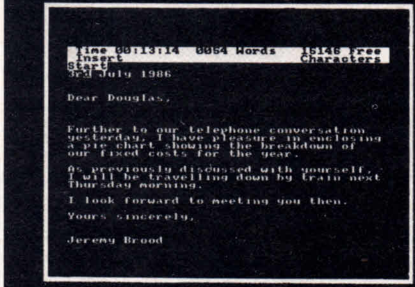
B) When the machine answers, read the order from your order form slowly, clearly and distinctly giving all the information you have written down. Where possible leave a telephone number just in case we can't understand or hear your order.

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.Back issue orders will be mailed at the same time as the current issue.

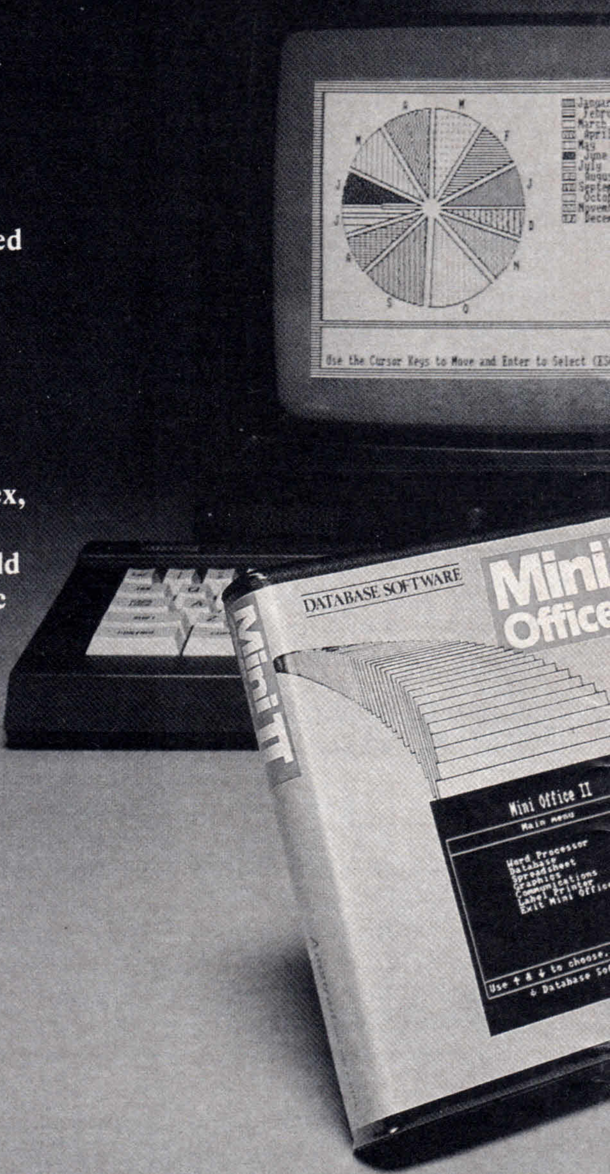
All prices include postage & packing

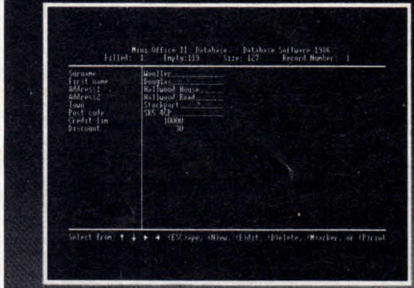
All this in just



```
Time 00:13:14  0054 Words   15146 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
```

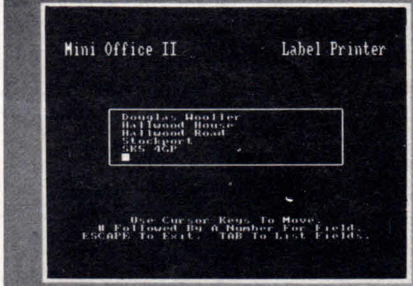
WORD PROCESSOR
Compose a letter, set the print-out options using embedded commands or menus, use the mail merge facility to produce personalised circulars – and more!





```
        Mini Office II  Database.   Database Software 1986
     Filled: 1    Empty:119     Size: 127     Record Number:  1
Surname         Wooller
First name      Douglas
Address1        Hallwood House
Address2        Hallwood Road
Town            Stockport
Post code       SK5 4GP
Credit lim           10000
Discount                30

Select from: ↑ ↓ ← → (ESC)ape, (N)ew, (E)dit, (D)elete, (M)arker, or (P)rint
```

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 Wooller
      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.

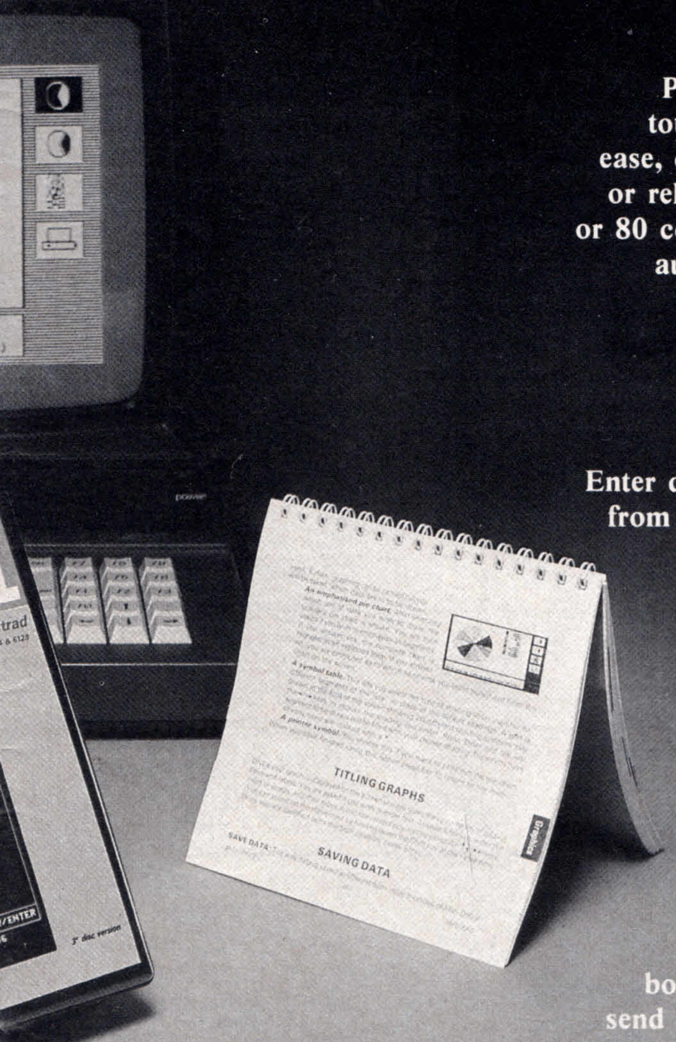
Yet possibly the best advertisement for Mini Office II is that it comes from the same stable that produced the original Mini Office package back in 1984.

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.

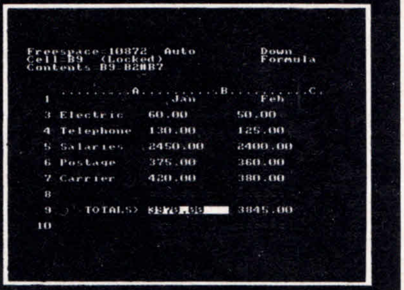
This is the package thousands of Amstrad owners have been waiting for – and at a price everyone can afford!

ONE package!



SPREADSHEET

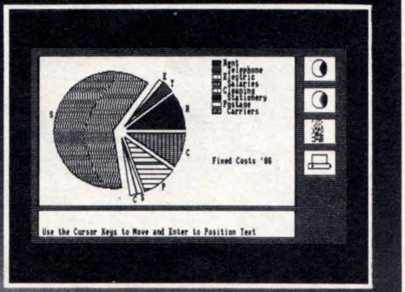
Prepare budgets or tables, total columns or rows with ease, copy formulae absolutely or relatively, view in either 40 or 80 column modes, recalculate automatically – and more!



```
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

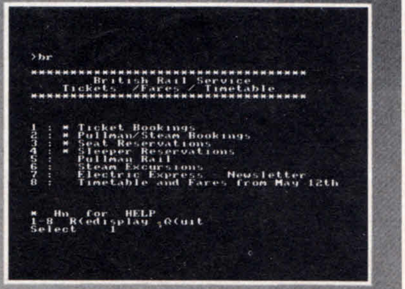
Enter data directly or load data from the spreadsheet, produce pie charts, display bar charts side by side or stacked, overlay line graphs – and more!



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**

Amstrad CPC 464, 664, 6128

Cassette	$36.95
3" disc	$48.95

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**

DATABASE SOFTWARE

order form on Page 72

Your'e never too young to play a Magical Adventure on the Amstrad CPC range....

Kristin and Martin Hollis
The Magic Sword
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!

SPACE SHUTTLE

TAPE $24.95
DISK $37.50

Real challenger

THE majority of you will have flown a computerised Spitfire or a jumbo jet at some time. So how about tackling the big one? There's a vacancy at Nasa for a space shuttle pilot – no previous experience required!

The objective of the game is quite simple. You must take off, orbit, launch a satellite, re-enter the atmosphere, and land safely. Actually achieving this is a little more difficult.

The game offers a choice of three types of flight mode. The first is autoflight in which ground control flies the shuttle by computer.

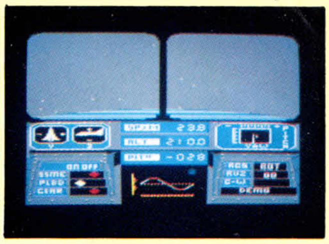
Option two is the simulation mode. The instructions claim that ground control looks after your fuel consumption, compensates for any stupid manoeuvres and overrides the majority of flight aborts which normally occur when you perform an action which would result in the shuttle's destruction.

The third flight mode is a genuine space shuttle mission. In this mode you receive no help at all from ground control. Your only indication of impending doom is a warning light on the console.

The screen shows the view from the cockpit and all the necessary instrumentation. A nice effect is the screen judder during lift off to simulate the power involved.

A pause key is provided so you can peek at the manual when things begin to go wrong.

The simulator is well designed, easy to learn but difficult to master. Beginners can get off the ground and into space with little trouble, but



launching and docking with satellites is trickier, and re-entry is a nightmare!

Space Shuttle provides an extremely challenging and novel variation on the flight simulator theme.

James Riddell

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

PRODIGY

TAPE $24.95
DISK $37.50

Playing Solo

MECHWORLD is a totally mechanised planet. The machine sorcerer Wardlock is at the forefront of research into organic life. The results of his genetic activities are held under strict security in his mechlab.

There are two prize specimens in the collection: Solo the Syntleman – a synthetic humanoid flesh form, and a recent addition named Nejo – a humanoid baby.

Your task is to guide our two space suited heroes through the four regions of the mechlab to collect four security units and take them to the control centre.

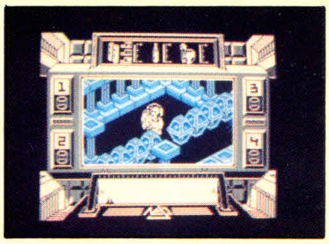
This is made more difficult by the constant need to feed Nejo. If that wasn't bad enough Nejo must be changed regularly.

You can control Solo with a joystick, but I preferred to use the keyboard. You can move him in four directions and fire his bubble gun.

You have no control over Nejo. If you don't walk too fast he will crawl along behind you, wiggling his rear end in true baby fashion.

Graphically the game is very like Knight Lore, all characters and backgrounds being drawn in 3D.

The baby idea gives the game a pretty high "cute" rating but apart from that it's nothing special. The action is



slow the outcome too random.

I'm afraid that it's all been done before, and far better.

Jon Revis

Sound	*6*
Graphics	*8*
Playability	*5*
Value for money	*6*
Overall	*6*

IT'S A KNOCKOUT

TAPE ONLY
$22.50

Knockout fun

It's a Knockout is the computer version of the multi-national TV program of the same name. It consists of five silly events and a mini marathon. The only thing missing is David Vine's commentary.

The first event is Flying Flans. Two of your team-mates launch flans into the air using a see saw and you must catch them on a tray.

Harlem Hoppers is event number two. Your female partner stands on a camel's head and drops a ball so it runs up and down the humps and finally flies into the air.

All you have to do is catch the ball. This wouldn't be too difficult were you not tethered to a length of elastic.

In Titanic Drop you slide down a rope from the bow of a ship. Below you in the water are several rubber rings. To score points you let go of the rope and land in one of the rings.

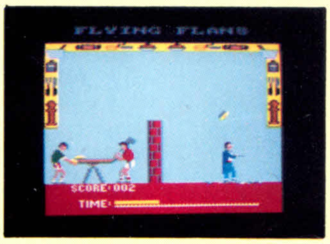
Diet of Worms is a stomach churner of a game – run around the farmyard collecting worms as they rise to the surface.

The final event is the obstacle race. You make your little men run by furiously waggling the joystick.

Sometime during these events you will be called upon to compete in the mini marathon – the Bronte Bash.

You control a crane from which is suspended a one ton weight. A dinosaur's head will appear at random from one of six craters in front of you. You move the crane left or right and try to drop the weight on to the unsuspecting dino.

Neither the graphics nor the animation could be described as state of the art. The characters are chunky and the



action slows down when there is more than one sprite on the screen.

In its favour the game can be played by up to six people and, more importantly, it's fun.

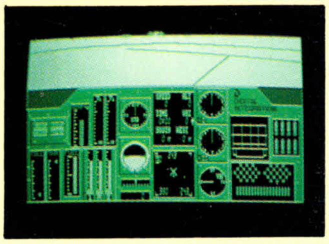
James Riddell

Sound	*7*
Graphics	*7*
Playability	*7*
Value for money	*7*
Overall	*7*

Chopper pilot

TOMAHAWK

PCW DISK $49.95



TOMAHAWK is a helicopter flight simulation game based on the McDonnell Douglas Apache. It comes with a comprehensive instruction manual and a leaflet about the helicopter. Both should be read to get the most out of the game.

The menu allows selection of either day or night-time mission, clear sky or cloudy (with various heights of cloud-base), cross-winds and turbulence or a smooth ride and four levels of pilot skill.

Flying a helicopter, if this simulation is to be believed, is easier than I thought. According to the leaflet, the Apache's advanced computer systems help a great deal and the throttle stays at maximum throughout.

Takeoff is achieved via a combined vertical lift and forward thrust control and the rudder selects the initial direction in which to move. All these are controlled by preselected keys.

There is a joystick option but it doesn't simulate real control very well. The effect is proportionate to the length of time it is held rather than the extent of movement.

I couldn't persuade my fingers of this, particularly when executing slow, steady turns requiring repeated movement – the cursor keys were much easier.

There are three different armaments, selected from the keyboard. A compass gives your heading and the track and bearing of three different enemy units or the nearest helipad. There is a choice of four combat missions. You must destroy eight ground units per map-sector and there are up to 128 sectors. At the same time enemy helicopters are on the attack and must be shot down.

Scenery – buildings, trees, hills, pylons, enemy units and the helipad – are all in 3D. The perspective of these objects changes as the craft moves over them or changes height.

These graphics are a bit rudimentary – simple wire frame objects, but the illusion of movement is well maintained.

The control panel, which takes up the lower half of the screen, is quite impressive. There is no sound other than a warning of excessive speed or torque.

This game suffers from confusion of intent – is it to be a high quality flight simulation or an attack and defence game?

If it is the former, the poor graphics, lack of sound and absence of any effect of wind and weather in the cockpit reduce its appeal. If the latter, there are better available.

A good arcade game automatically increases the levels of difficulty as play progresses – this does not. £19.95 seems a high price, but flight enthusiasts may well find it worth paying.

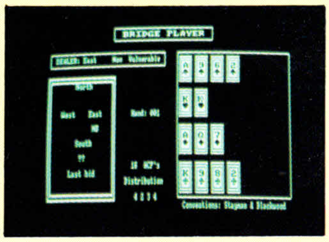
Michael Sterne

Sound	*3*
Graphics	*6*
Playability	*8*
Value for money	*6*
Overall	*6*

Opening bids

BRIDGE PLAYER

PCW DISK $49.95



BRIDGE Player from CP Software is loaded via CP/M. After loading you can choose, somewhat pointlessly, from four speeds – the speed does not affect playing skill.

Cards are dealt randomly unless you specify the point count, the only hand specification available. The program then continues random deals until it deals a hand with the required count. Don't choose a specification of low probability, there is no exit.

You bid and play by pressing appropriate keys. New bidding sequences can be called for and there is an exit facility. Scores (rubber bridge) are displayed at the conclusion of each hand.

Even at the simple, procedural level this program is irritating. The bidding sequence cannot be recapitulated (permitted within the rules) whereas the previous trick remains on display throughout the play of the next (against the rules).

You confirm bids or card-plays by pressing the spacebar. This is also the method of requiring the program to choose the next card to play, so an over-long key-press causes an unexpected play and mistaken key-presses cannot be corrected.

There is no way of claiming the remainder of the tricks when play is effectively at an end – you must play each hand, however boring, to the last card.

Some simple opportunities for practice have been missed. It would have been helpful to have been able to compare my choice of bid or play with the program's and an advantage to have been able to play with all hands open.

The program makes mistakes, even at an elementary level. It cannot distinguish between re-opening the bidding after two no-bids and an immediate intervention. It cannot unblock a suit and it has difficulty handling more subtle sequences too.

Contract Bridge is above all a social game. The interplay of the psyches of the four players and the quality of the relationships between the two pairs are a major part of its appeal. Bridge with only one player loses something, not least the fights between otherwise loving couples.

The famous bridge writer S.J. Simon invented a character called The Unlucky Expert who knew everything there was to know about bidding and play but who so misunderstood his fellows that he always ended a loser. One of his special characteristics was his penchant for following the rules come what may.

Bridge is a complex and subtle game, difficult to reduce to simple rules. The manual for this program is scanty, containing the most elementary rules for playing Acol, the bidding system on which the program is based.

It is those rules, slavishly followed, that determine the bidding sequences. I am sure bidding can be reduced to an algorithm and is inherently programmable but if the choices are over-simplified, the game is ruined.

Who might find the program worth buying? A beginner, perhaps, too shy to learn in company or a desperate but not excessively skilled enthusiast too isolated to find three other like-minded people.

Most players would do better to join a bridge club.

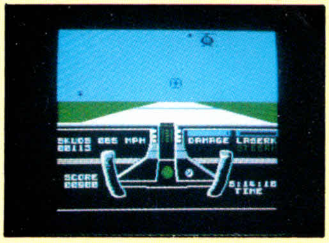
Michael Sterne

Sound	*N/A*
Graphics	*3*
Playability	*7*
Value for money	*5*
Overall	*5*

KNIGHT RIDER

TAPE $22.50
DISK $37.50

Rider misfires



MANY moons ago Ocean began advertising the computer game of the TV series Knight Rider. At long last it has arrived, though I'm not quite sure whether it was worth the wait.

As Michael Knight, you and Kitt the computerised car are in Atlanta on the trail of a gang of international terrorists. From the title screen you can choose one of three scenarios, or a fourth which is randomly generated using sections from the other three. Whichever plot you choose the game always begins with a map of the USA with ten cities marked.

Kitt scrolls a message from Devan across the top of the screen giving you your first port of call on the trail of the terrorists.

Firing up the jet engines you set out for your destination. The background design is very simplified – blue sky and green grass.

While travelling you will be attacked by an endless swarm of missile-launching enemy helicopters. There are two ways of tackling these. Firstly you can use the auto pilot to drive while you zap the helicopters with lasers. Unfortunately Kitt is not a very fast driver and the helicopters are difficult to shoot down.

The second and better course of action is for you to drive while Kitt shoots down the enemy.

Once at your destination you enter the terrorists' hide-out and make your way across the room. The rooms are viewed from above and are drawn in good detail.

Movement is rather slow and clumsy with joystick positioning being critical. Any diagonal movement of the stick results in Michael standing still, invariably losing a valuable life.

Knight Rider consists of a repeating sequence of two second-rate games. Don't be tempted by the glamour of the TV series.

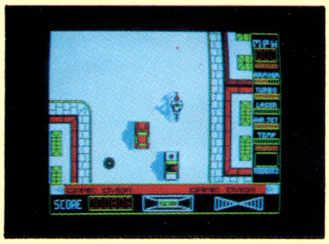
James Riddell

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

STREET HAWK

TAPE $22.50
DISK $37.50

Cops and robbers



STREET Hawk is a very special motor cycle – a kind of Knight Rider on two wheels. Playing the role of its rider Jesse Mach you must eliminate the city's criminals.

Speed is one of Street Hawk's major strengths. It is capable of 100 mph in normal mode and will scream up to 250 mph when you hit the turbo button.

Other niceties are its armour plating, lasers, and air jets which provide sufficient vertical thrust to get you airborne. Combine this with your forward motion and you can leap most vehicles.

The majority of Street Hawk's functions can be operated via keyboard or joystick. However, turbo and jump have to be selected on the keyboard.

The game's graphics are very pleasing. The city's streets are viewed from above using high resolution graphics and good colour.

All vehicles are drawn complete with shadows, which are used to good effect during the game. Grab a handful of throttle and the bike will do a wheelie which without a shadow would probably go unseen.

Down the side of the screen are indicators for such things as laser temperature, armour, turbo and air jet status. A single line of scrolling text alerts you to any criminal activity in the area.

The riding and zapping are interspersed with the occasional stick up. Pull up at the scene of a crime and the display changes to the view from Jesse's visor. Using your lasers you must fry the crooks as they run to their get-away van.

Your biggest headache is the local police force. They don't like other people doing their job and will do all they can to stop you.

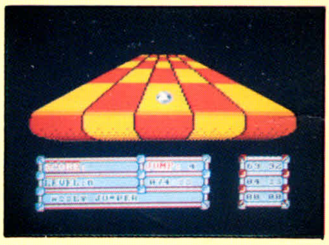
I had a great time weaving my way through the traffic at high speed. This is a really enjoyable game of cops and robbers. **Steve Brook**

Sound	6
Graphics	9
Playability	8
Value for money	8
Overall	8

TRAILBLAZER

TAPE ONLY
$24.95

Blazing a trail



LIFE is hard on the mega galactic trail. Only the elite rise to the dizzy heights of the trailblazers.

The action takes place on a gigantic conveyor belt in space and you are in control of a spinning football.

Whip the conveyor up to full speed and you'll soon be scoring points – the faster you go, the more you score. It is at this point that you notice occasional holes in the conveyor, which become more and more frequent as you progress into the game.

The belt is divided into coloured squares the significance of which soon becomes apparent. White squares cause you to bounce, which can often be used to your advantage.

Green squares stop you dead, which is a nuisance as some gaps must be jumped at full speed if you are to clear them. Worst of all are the blue squares – these reverse the left and right movements of the joystick.

There are two ways of playing the game – arcade mode, which starts you on level A with five balls, or the three course test, in which you select three levels from A to N.

Trailblazer is graphically stunning and so fast that you just wouldn't believe it!

Jon Revis

Sound	8
Graphics	10
Playability	10
Value for money	10
Overall	9

... 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.

DATABASE SOFTWARE

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.

To order, please use the form on Page 72