

60p

# YOUR COMPUTER

JANUARY 1983 BRITAIN'S BIGGEST-SELLING HOME COMPUTER MAGAZINE Vol. 3 No. 1

**How  
tomorrow's  
technology  
will turn today's  
micros into antiques**

**Reviews:  
Graphics  
tablets  
Spectrum  
software**

**BBC turtle graphics**

**Plus plenty of games and features for the  
ZX-81, Dragon, Vic, Atom and Atari**



JANUARY 1993						
M	T	W	T	F	S	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

**Win an Oric**



## Super Expander commands

**Auto** — displays and increments line numbers automatically. The user can specify the starting line number and the interval between lines, if not then the starting line defaults to 100 and the interval to 10.

**Renumber** — automatically rennumbers all the program lines including the Goto and Gosub references, starting at the line number specified and at the increment set by the user.

**Delete** — can delete a single line, or all the lines specified between a starting and finishing point, or all the lines from a starting point to the end of the program.

**Find** — searches the program for a specified Basic command, character, or character string, and displays all the lines in which it appears.

**Change** — searches for a Basic command or character string and replaces it with a new command or string.

**Edit** — changes from program mode to edit mode, which alters the function key commands.

**Key** — allows the user to alter the commands assigned to the function keys. The information assigned to any one key must be 10 characters long or less, and an automatic carriage return can be included in the function.

**Help** — displays the line in which an error has occurred during program execution and highlights the error in reverse characters.

**Dump** — displays all the variables and their values in the order in which they were defined, except those in arrays. The value of a variable can then be changed by over-writing it.

**Trace** — displays the program line number as it is being executed in a small window at the top right-hand corner of the screen. The Shift or Ctrl keys slow down the display if it is too fast. This is useful for finding infinite loops within a program.

**Step** — halts the program after each program instruction, displays the line numbers associated with that instruction and the first line number of the next instruction. The Shift or Ctrl keys cause the next instruction to be executed.

**Off** — cancels the Trace and Step modes.

**Prog** — changes the assignments of the function keys to normal Basic commands.

**Merge** — loads a previously stored program or subroutine from disc or cassette and incorporates it into the program already in the Vic's memory.

**Kill** — cancels the functions of the cartridge, but leaves the assignment of the function keys unaltered. It is necessary to inhibit the cartridge during normal program operation because memorising information for diagnostic messages such as Help increases execution time.

**Ctrl A** — scrolls up a listing.

**Ctrl Q** — scrolls down a listing.

**Ctrl L** — blocks out all the characters to the right of the cursor on a line.

**Ctrl N** — blocks out all the characters on the screen after the cursor.

**Ctrl U** — blanks out the line on which the character is positioned.

**Ctrl E** — inserts information between quotation marks on a program line.

# VIC-20 ADD

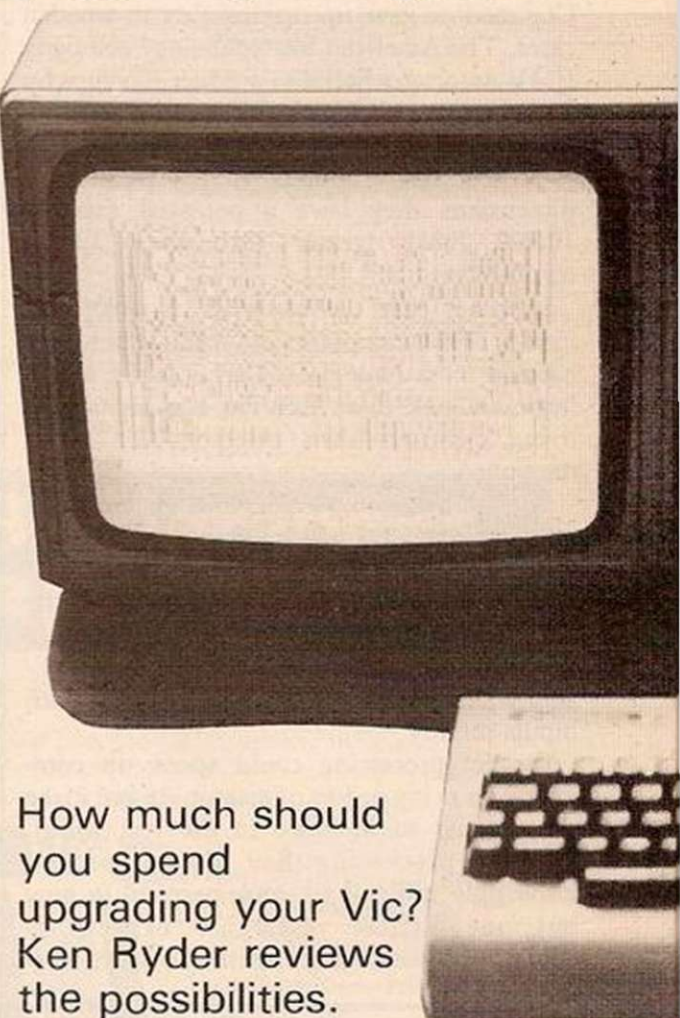
IF YOU WANT to upgrade your Vic-20, expanding the memory must be a priority. A screen expansion and some aids to programming in Basic and machine code are also often near the top of Vic-owners' shopping lists once they have tired of cartridge games.

With the Commodore expansion system a 3K, 8K or 16K memory pack can be plugged directly into the back of the Vic to give a maximum on-board memory of 21K, 19.5K of which is available for Basic programs. In order to expand to 32K a motherboard with six expansion slots can be plugged into the back of Vic allowing the 3K, 8K and 16K cartridges to be used together. This leaves three slots available for utility cartridges such as the programmer's aid, machine-code monitor, and games packs.

When the Vic has been fully expanded to 32K only 27.5 is available for Basic programs because the area of memory occupied by the 3K RAM pack is no longer used. However, it is still available for storage of machine-code programs, and for Peeking and Poking values to, so you could store an alternative character set or several screens of information there.

One of the main criticisms of the Vic is its 22x23 screen display, and a 40-column option has been long awaited. The Beebox was one of the first attempts, but was expensive and the manufacturer has now gone out of business. Stack Computer Services has stepped into the breach with a 40/80 column x 25 row monochrome card.

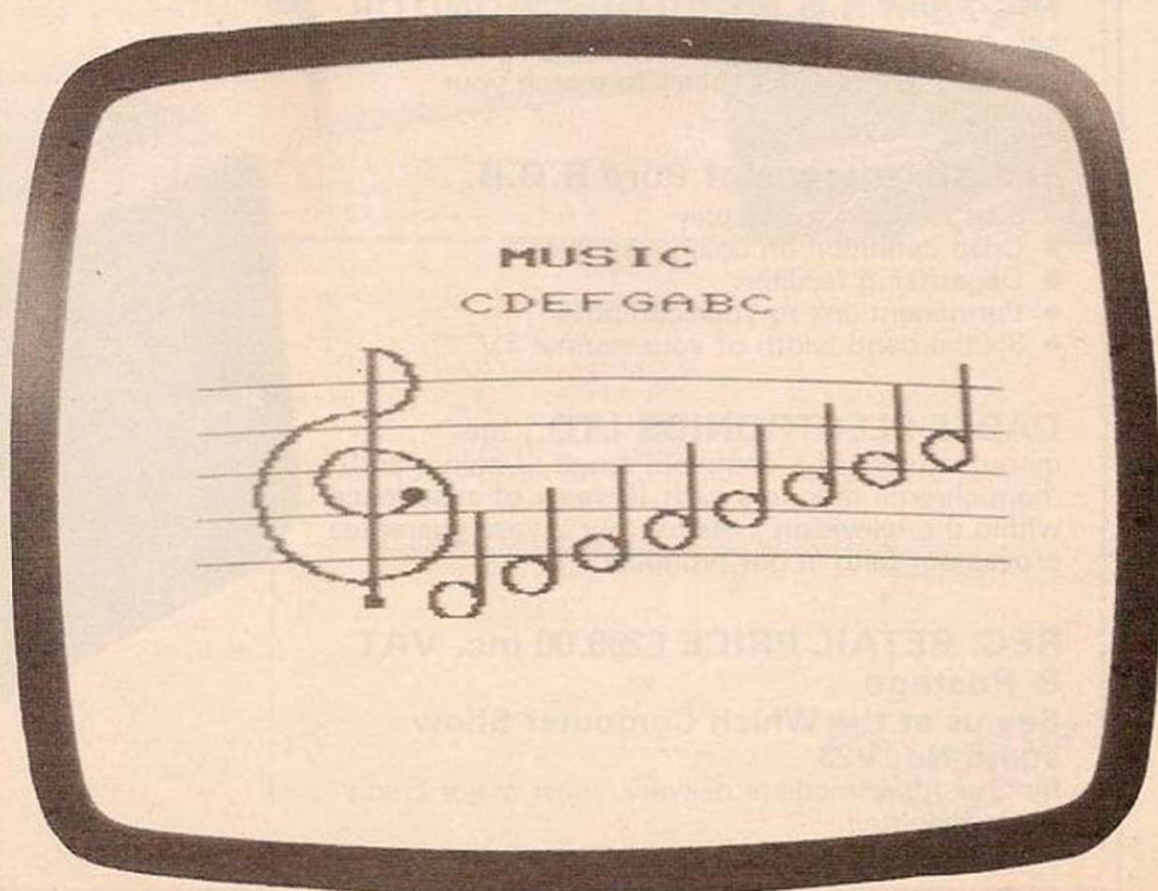
As soon as Commodore's Super Expander cartridge is plugged in you have 3K of extra RAM. The function keys assume eight single keystroke commands, which you can redefine if necessary. Nine new Basic commands are



How much should you spend upgrading your Vic? Ken Ryder reviews the possibilities.

available for plotting and colour control. Eighteen commands become available for the production of sound and music, and seven for reading values of sound and colour registers, including inputs from devices connected to the games port such as joysticks and light-pens.

The pre-assigned function-key commands







are Graphic, Color, Draw, Sound, Circle, Point, Paint and List. These may be changed using the Key command to any 128-character string, including a carriage return if required. If you do not want to use those graphics commands in a program, the function keys could be used to enter standard Basic commands instead such as Next, Goto, Print, and Gosub.

Unfortunately, the cartridge does not use the highest resolution available on the Vic,  $176 \times 184$ , but reduces the screen area slightly to 20 rows by 20 columns, giving a maximum resolution of  $160 \times 160$ . Presumably Commodore did this to reduce the screen memory overhead. Then for some reason they divided the screen into a  $1024 \times 1024$  co-ordinate system with its origin in the top left-hand corner of the screen. As there are only  $160 \times 160$  possible plotting positions this means that several co-ordinates occupy the same pixel.

Graphic selects one of the four graphics modes. These are: normal text, medium resolution,  $80 \times 160$ , multi-colour graphics allowing any of the 15 colours on the Vic to be used, high-resolution mode,  $160 \times 160$ , allowing only the eight keyboard colours to be used, and high-resolution multi-colour mode which allows the mixing of the high-resolution and medium-resolution multi-colour options. All of the modes except normal text require 3K RAM for the screen.

There are four colour registers available and a number from 0-15 can be stored in each,

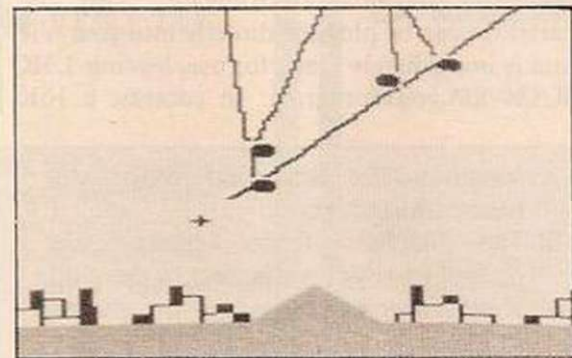


specifying a colour. Register 0 holds the screen colour, register 1 holds the border colour, register 2 holds the character colour used for plotting points and register 3 holds the auxiliary colour which also can be used for plotting, but only in multi-colour mode. The Color command sets the values for these registers.

Point plots a point on the screen at specified co-ordinates. Several points can be plotted using a single Point command, but all will be the same colour. The colours available and resolution depend upon the graphics mode selected. Region changes the character colour, register 2, to any of the eight keyboard colours in high-resolution mode, or any of the 16 colours in multi-colour mode.

Draw plots a straight line from one set of co-ordinates to another, for as many points as can be specified in an 88 character line. It can also be used to plot lines from the finishing points of previously drawn shapes and circles. Circle is an extremely powerful command allowing the user to draw circles, ellipses and arcs on the screen.

Paint fills an enclosed area with a colour starting at specified co-ordinates within the area. If the co-ordinates are outside the area, or the area is not fully closed then the whole



screen will be painted. Again the colours available depend upon the graphics modes.

Char permits strings of text to be displayed on the screen whilst in high-resolution or mixed mode. This facility is not available in medium-resolution multi-colour mode. ScnClr — clears the graphics screen.

The number and range of colours available in each graphics mode is a little complicated. The screen consists of  $20 \times 10$  double-height blank characters, and there is a colour block associated with each character. In multi-colour mode medium-resolution plotting can be performed in any of the colours specified in registers 1, 2 and 3 against the background

colour of register 0 and any of the 16 colours can be entered in each register. Thus four colours can appear on the screen at one time.

In high-resolution mode plotting can only be performed in the character colour registers, and only the eight keyboard colours can be used. During a program the character colour can be changed using the Region command, but if two points of different colours try to occupy the same colour block such as two lines intersecting, some interference will occur. With careful programming it is possible to use all eight colours on the screen at one time.

In high-resolution mixed mode only registers 0 and 2 can be used for plotting, and all 16 colours can be specified. If the colour set in register 2 is one of the keyboard colours, all points will be in high resolution. If it is one of the other colours, plotting will be in medium resolution. Again if points of different colours or resolutions try to occupy the same colour block some interference may be produced, but with care it is possible to use all 16 colours on the screen at one time.

The music capabilities of the Vic are greatly enhanced by the Super Expander cartridge. The Sound command controls four voices and their volume, and can be used to produce two or three note chords. Each voice is allocated a three-octave range, which at first sight appears to give nine octaves. However some of the octaves overlap giving a five-octave range.

Entering Music Mode with the Ctrl and  $\square$  keys, converts keys A, B, C, D, E, F, G, #, \$, P, Q, V, S, O, R. into musical commands. Keys A-G play the natural notes A-G. If # is pressed before the note a sharp is played, if \$ is pressed before the note then the note is flat. V controls the volume of the note and R is a rest or silent period. The duration of the note or rest is determined by tempo, T. Any of the four "voices" can be chosen with S, and any of the three octaves in its range with O. S302V5T9# A, plays A sharp of voice 3 in octave 2 at volume 5 for time 9; approximately 4 seconds.

All of the musical commands can be displayed on the screen by entering P, and this can be cancelled by Q. These commands can be entered directly at the keyboard, or they can be combined in a string and executed by a Print statement within a program.

There are also several commands which can be used to read the colour and sound registers, or test the condition of various peripherals

(continued on next page)



(continued from previous page)

that can be plugged into the games port. Rgr reads the mode set by the Graphic command and Rcolr reads the values of the registers set by the Colour command. Rdot reads the colour of any point on the screen and Rsnd reads the values of the four voice registers and their volume as set by the Sound command.

Rpot reads the values of any paddles or potentiometers connected to the user port and returns a number in the range 0-255. Rpen reads the X, Y, position of a light-pen on the screen and Rjoy reads the position of a switch-type joystick and its fire button.

Documentation consists of a 22-page booklet which describes each command complete with a small programming example. If you are thinking of buying a 3K expansion pack and want to use your Vic for graphics and sound, spend an extra £5 and buy the Super Expander cartridge — it is well worth it.

The Vic-20 Programmer's Aid Cartridge is a useful tool for writing, editing and debugging Basic programs. When first plugged into the Vic or expansion board it has no effect and must be initialised to gain access to the extra commands. This is achieved by typing SYS28681 followed by the Return key. Those of you familiar with the Vic memory map will realise that the area starting at location 28681 normally resides in an 8K block of RAM, if fitted. If you have a fully expanded Vic the last 8K block of RAM will be unavailable when using the Aid cartridge, allowing only Basic programs of 19.5K or less to benefit from it.

After initialisation the cartridge is in program mode and the function keys are assigned 12 useful Basic commands such as Goto and Gosub. Normally only eight function keys are available, the other four, F9-F12 are obtained by holding down the CTRL key and pressing the function keys. The Edit mode is entered by typing Edit, or by pressing the Ctrl and F1 keys together. In this mode the function keys are assigned special editing commands, which cannot be included in Basic programs such as Delete, Find and Step.

The function keys simply allow single-keystroke entry of commands and do not limit

the commands available. The user can even as sign his own commands to the function keys if desired using the Key command.

The cartridge comes complete with a 15 page instruction book describing each new command, together with a short programming example. The final section uses a dice-throwing program as an example of how to write and debug a Basic program using the features of the cartridge.

Anyone writing long complicated Basic programs of 19.5K or less will find this cartridge invaluable. The time and effort saved in program development could soon cover the cost.

Vicmon as the cartridge is affectionately referred to in the documentation, is similar to the Programmer's Aid, except that it simplifies the writing and debugging of assembly language programs rather than Basic. Typing SYS24576 followed by a Return initialises the cartridge, again the last 8K RAM block is occupied by Vicmon.

The function keys are not assigned any values. Upon initialisation the screen displays the contents of the 6502 registers, that is the program counter, status register, accumulator, index register X, index register Y and stack pointer. The commands offered are single characters followed by various parameters such as start or finish addresses, op-codes, operands and hex values. In fact all operands must be preceded by \$.

Vicmon should not be confused or compared with a full assembler/editor. It is a simple aid for the production of short machine-code programs or subroutines. For anyone who has tried hand assembly and a Basic loader for machine code this cartridge is heaven sent, reducing nervous breakdowns to a minimum. Although as all operands must be entered in hex, a decimal to hex conversion command would have been welcome, but I suppose you can't have everything.

At £35 this cartridge appears expensive when compared with some cassette-based assemblers, however, you are paying for the reliability of quality firmware. Also the cartridge can be plugged directly into your Vic and is immediately ready for use, leaving 3.5K RAM for your program. In contrast a 16K

RAM pack may be required to load a good software assembler.

The documentation assumes the same layout as the previous cartridges. It is aimed at the reader who is familiar with 6502 assembly language, but not an expert. The cartridge commands are presented in alphabetical order, leading to some page flicking, as some of the earlier commands refer to others further on in the text. Some important information, although obvious, is missing. For instance the M command can be used to create word tables or blocks of data as well as editing them; an example of the format required for each form of addressing would be beneficial; for conditional branching only the address to be branched to need be specified, this is only implied in an example; all two-bit addresses are entered MSB,LSB rather than LSB,MSB as in some assemblers.

Stack's 40-column card plugs directly into the back of the Vic or any expansion-board slot, and comes complete with its own video chip to control the display, 2K RAM for the screen memory, and the normal Vic/Pet character set in ROM. The card uses the Autostart facility of the Vic and if no action is taken at power-up the display is automatically in 40-column mode. Various key combinations during power-up will obtain the 80-column or normal Vic configurations. The display can be changed at any time, either directly from the keyboard or under program control.

The card comes complete with monitor and UHF output sockets, either of these together with the Vic's normal UHF output can be used to drive two separate televisions or monitors. For example a program listing or a table of results can be displayed on the 40/80-column screen whilst a graph is plotted on the other.

The card also offers a couple of other useful features. The screen can be set to give automatic line spacing, and if the normal Vic screen is not required the 1.5K memory normally allocated can be used for Basic program storage, also the lower 3K of RAM can be used for Basic giving the Vic a true 32K expansion potential. Obvious uses are word processing, communications, business applications and education.

## CONCLUSIONS

- Commodore's cartridges are well presented and constructed, the firmware is professional and bug-free.
- The approach of the documentation is far superior to that of the Vic-20 user manual, including contents, introduction, examples, summary and indices, but the demonstration programs could be better.
- The Super Expander Cartridge offers good value for money and should be purchased in preference to a plain 3K RAM pack. It would be a great advantage if the Super Expander functions could be included with 16K RAM instead.
- The Programmer's Aid offers many useful editing and debugging features, but can only be used to develop programs up to 19.5K in

length. This limitation may well restrict its market.

- The Machine Code Monitor will appeal to anyone wanting to develop short assembly language programs. It falls nicely between the two extremes of hand assembly/Basic loader, and a full assembler/editor. The cartridge limits the maximum possible program RAM to 22.5K but this should be no real restriction to assembly language programs.
- Stack's 40/80-column board is a welcome addition to the Vic range of accessories. However, the cost, monochrome display and inability to give high-resolution graphics, 320 x 192, may fail to appeal to the home user market.
- Vic owners should consider their expansion plans carefully. If you are

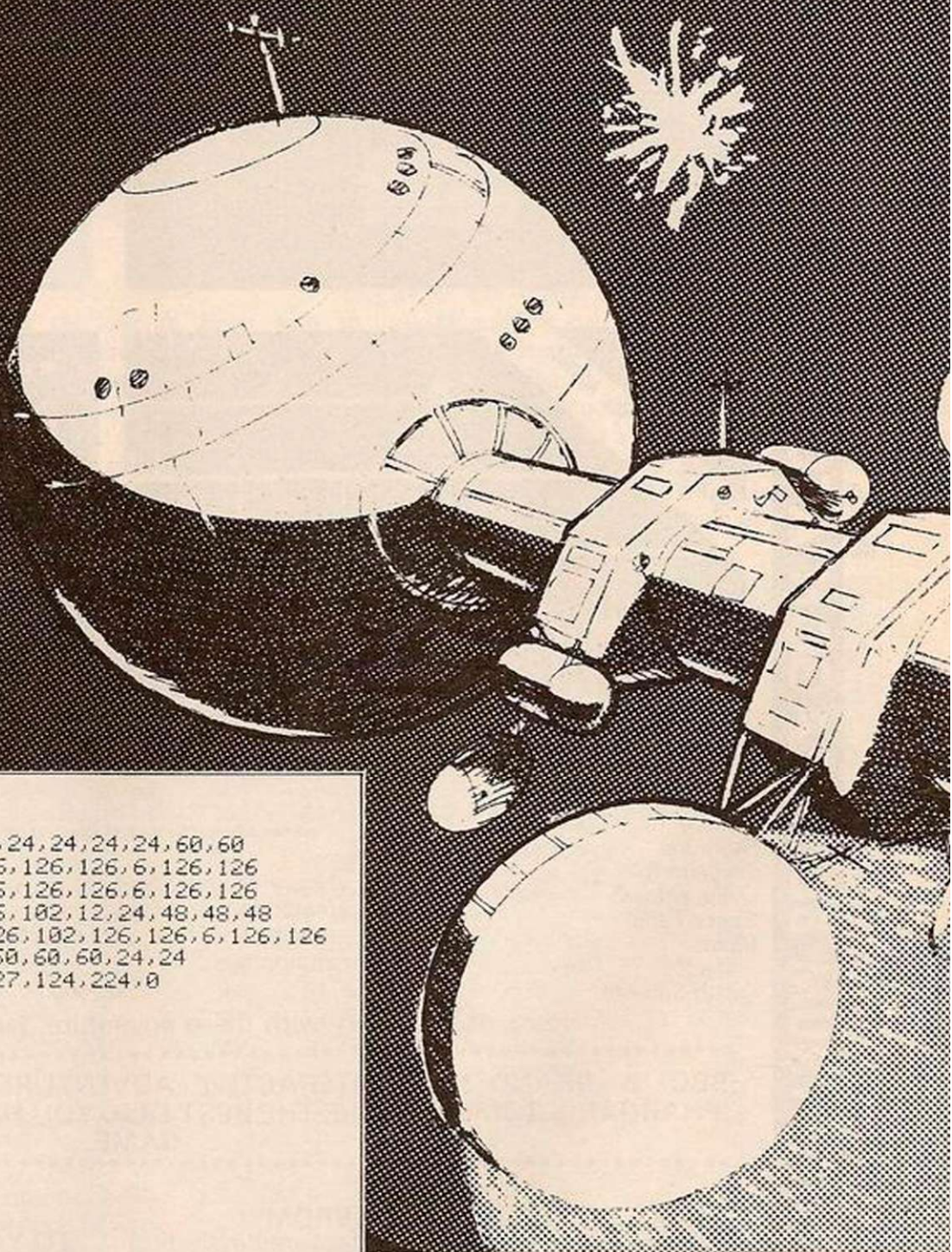
only interested in Basic programming and do not require high-resolution graphics, it seems pointless to buy a 3K RAM pack as it will not be available for Basic when expanding above 6.5K.

- Expanding above 19.5K may be a waste of money and RAM if you want to use the Programmer's Aid, because the top 8K of expansion area is occupied by this cartridge when in use. If you are only interested in assembly language then Vicmon may be all you need, plugged directly into the back of your Vic the 3.5K on board will go a long way.
- The Programmer's Aid, Super Expander and Machine Code Monitor each cost £34.95, 3K of RAM costs £29.95 and the Stack 40/80-column card costs £115.



# SPACE RUN

Flee through the cosmos in this engrossing game of pursuit written by David Browne. Malignant alien beings seek tirelessly to destroy your vessel.



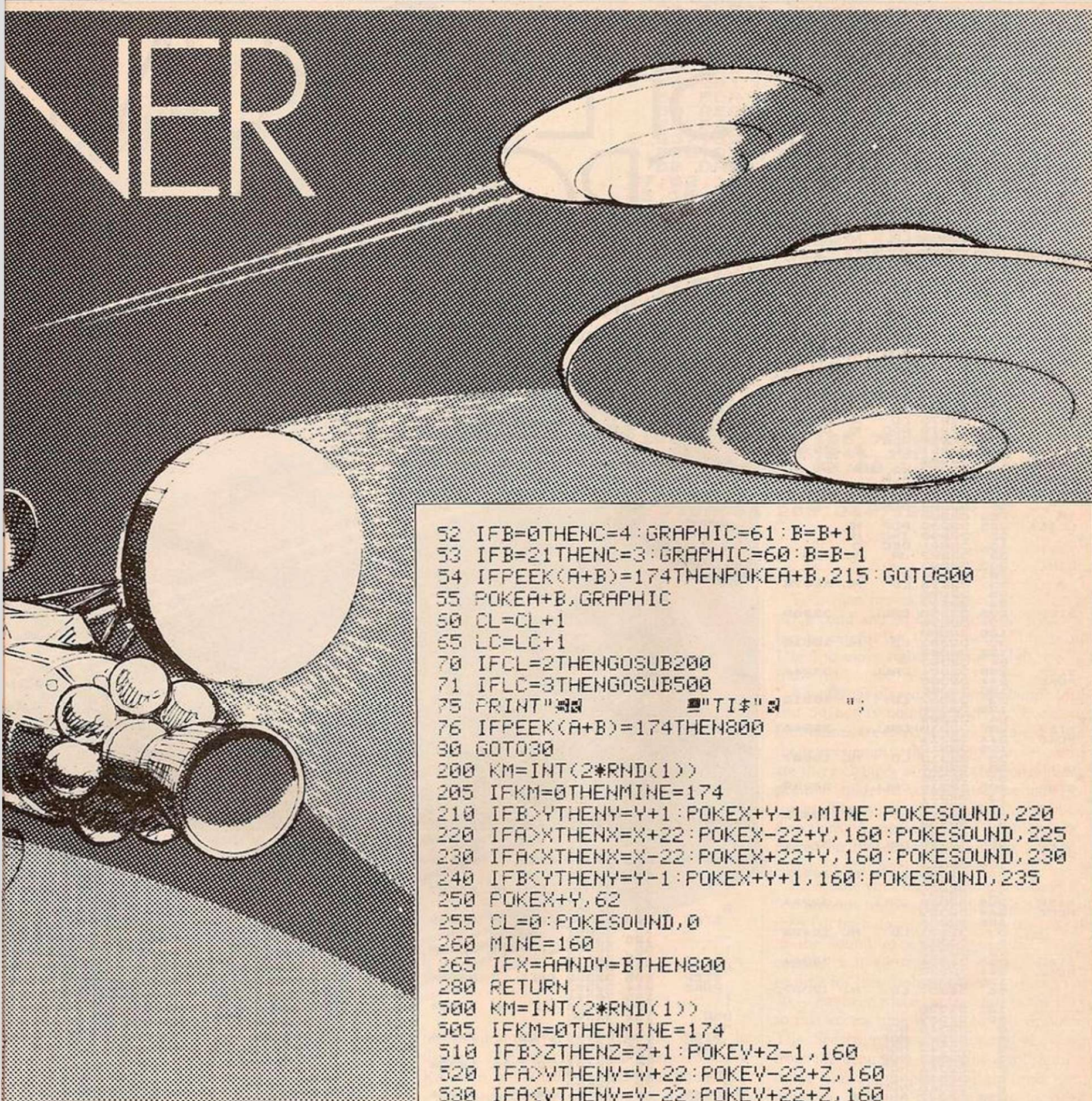
```
0 FORI=7552T07671:READA:POKEI,A:NEXT
1 DATA126,126,102,102,102,102,126,126,56,56,24,24,24,24,60,60
2 DATA126,126,6,126,126,96,126,126,126,126,6,126,126,6,126,126
3 DATA96,96,108,108,126,126,12,12,126,126,96,126,126,6,126,126
4 DATA126,126,96,126,126,102,126,126,126,126,102,12,24,48,48,48
5 DATA126,126,102,126,126,102,126,126,126,126,102,126,126,6,126,126
6 DATA24,24,60,60,60,126,126,66,66,126,126,60,60,60,24,24
7 DATA0,7,62,254,254,62,7,0,0,224,124,127,127,124,224,0
8 DATA0,60,126,171,213,126,60,0
10 PRINT"Q=Q";
11 PRINT"Q="";
12 FORT=1T020
13 PRINT"QI";
14 NEXTT
15 PRINT" ";
16 POKE36879,8:TI$="000000"
17 POKE36869,255
18 POKE8185,160
19 PRINT"QI"TI$";
20 GRAPHIC=209:A=7900
21 MINE=160:B=11
22 V=8142
23 Z=5
24 CL=0:POKE36878,15
25 X=8142:SOUND=36876
26 Y=15
27 LC=0
28 C=0
30 KEYS=PEEK(197)
41 IFKEYS=47THENC=1:GRAPHIC=59
42 IFKEYS=39THENC=2:GRAPHIC=58
43 IFKEYS=33THENC=3:GRAPHIC=60
44 IFKEYS=26THENC=4:GRAPHIC=61
45 IFC=1THENA=A+22:POKEA+22+B,160
46 IFC=2THENA=A-22:POKEA+22+B,160
47 IFC=3THENB=B-1:POKEA+B+1,160
48 IFC=4THENB=B+1:POKEA+B-1,160
50 IFA=7702THENC=1:GRAPHIC=59:A=A+22
51 IFA=8164THENC=2:GRAPHIC=58:A=A-22
```

THIS PROGRAM for the unexpanded Vic makes economical use of the user-defined character facility. Fifteen characters are defined in data statements and 10 of these redesign the numbers 0 to 9. It would be easier to read the definitions for numerals from the character set in ROM into RAM — but less pleasing to the eye.

## In hot pursuit

The object of the game is to manoeuvre your ship around the screen avoiding two saucers in hot pursuit. Use the keys Z and X to turn left and right, the function keys F1 and F3 for up and down. Watch out for the deep-space mines laid by the pursuing alien hostiles.





One problem with redefining a limited character set is that if the program crashes you will not be able to read the error message to find out at which line an error has occurred.

The solution here is to key in

POKE 36869,240

and then press return. This will send the Vic back to the standard character set in ROM and turn the garbage on the screen into letters. Be careful to key it in correctly since the letters you type will not appear as such on the screen.

In line 16 the border and screen colours are set by Poking the value 8 into 36879. Changing this value will set up other colour combinations.

```

52 IFB=0THENC=4:GRAPHIC=61:B=B+1
53 IFB=21THENC=3:GRAPHIC=60:B=B-1
54 IFPEEK(A+B)=174THENPOKEA+B,215:GOTO800
55 POKEA+B,GRAPHIC
56 CL=CL+1
65 LC=LC+1
70 IFCL=2THENGOSUB200
71 IFCL=3THENGOSUB500
75 PRINT"      "
76 IFPEEK(A+B)=174THEN800
80 GOTO30
200 KM=INT(2*RND(1))
205 IFKM=0THENMINE=174
210 IFB>YTHENY=Y+1:POKEY+Y-1,MINE:POKESOUND,220
220 IFA>XTHENX=X+22:POKEV+22+Y,160:POKESOUND,225
230 IFA<XTHENX=X-22:POKEV+22+Y,160:POKESOUND,230
240 IFB<YTHENY=Y-1:POKEV+Y+1,160:POKESOUND,235
250 POKEV+Y,62
255 CL=0:POKESOUND,0
260 MINE=160
265 IFX=ARNDY=BTHEN800
280 RETURN
500 KM=INT(2*RND(1))
505 IFKM=0THENMINE=174
510 IFB>ZTHENZ=Z+1:POKEV+Z-1,160
520 IFA>VTHENV=V+22:POKEV+22+Z,160
530 IFA<VTHENV=V-22:POKEV+22+Z,160
540 IFB<ZTHENZ=Z-1:POKEV+Z+1,MINE
550 POKEV+Z,62
555 LC=0
560 MINE=160
565 IFV=ARNDZ=BTHEN800
580 RETURN
800 POKEV+Z,209:POKEV+Y,209
801 POKEA+B,209
802 PRINT"*****GAME OVER"
803 PRINT"*****PUSH F7"
804 GETA$:IFA$="I"THENRUN
805 POKEA+B,215
806 FORT=1TO10:NEXTT
807 GOTO801

```



Special effects, user-defined graphics, Martin Howse opens up the Vic colour vista with his illuminating guide to high-resolution graphics.

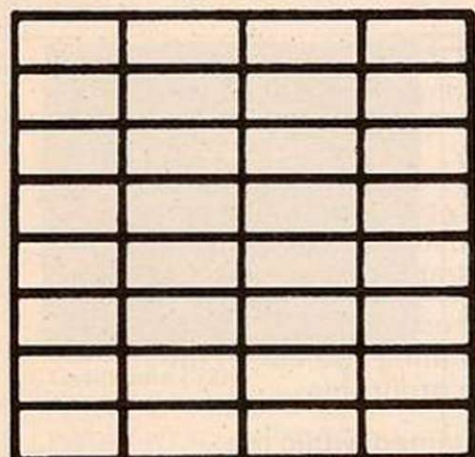


Figure 1.

MULTI-COLOUR GRAPHICS is a form of high-resolution graphics, and that is graphics defined on an eight by eight matrix of dots. However, it is different in that, unlike the high-resolution graphics which only have two colours for a dot on the matrix being lit or not lit, multi-colour characters have four colours.

There are a few limitations, in that the horizontal resolution is halved with multi-colour characters. That means that characters will be made on a four by eight matrix. Also, only four colours can be used; the border colour, the auxiliary colour, the character colour and the screen colour.

You cannot have many different characters with other colours because if you change the border, screen and auxiliary colours for a new character, the old character's colour will change. However, this may be used to produce special effects where many colours are changed very quickly.

To start defining a multi-colour character we must first decide on the colours. Since we can change the screen, border, character and auxiliary colours, we have a choice of four different colours which can be anything in the range of the Vic's first eight colours — except the border and auxiliary colours which may have 16 colours.

These colours are Poked into the Vic in the registers shown in table 1.

The codes of the character colours are found using the following table:—

- 0 Black
- 1 White
- 2 Red
- 3 Cyan
- 4 Purple
- 5 Green
- 6 Blue
- 7 Yellow

When the colours have been chosen and Poked into their appropriate memory location, you must start to design the character on a four by eight matrix. See Figure 1.

Once the colours have been put on the matrix where you want them, using, for example, A for auxiliary colour, B for border colour, C for character colour, you can set about coding the image. Each line of the

matrix will become a binary number which will be put into memory in decimal. You must go through each of the eight lines of the character matrix to do this.

To convert the line to a binary number you must start at the left-hand side of the line and work your way to the right. If you come to a position where you want a screen colour, write down 00; for a place with a character colour, 10; for a place with a border colour, 01; and for a place with an auxiliary colour, write down 11. You should end up with an eight-digit binary number, for example 00011100. Repeat this for all eight lines on the matrix. When you have finished coding the design, convert all your number to decimal and put them into memory using the following routine:

```
DATA 8 decimal numbers.
FOR F = 7168 TO 7175: READ A: POKE F, A:
NEXT
```

Now that it is in memory, we have to prevent it from being overwritten. Poke 52,28 is sufficient to do this. To use the character in

11101011 = 235

11101011 = 235

11111111 = 255

Then we Poke them into memory.

```
40 DATA 255,235,235,235,235,235,235,255
50 FOR F = 7168 TO 7175: READ 1: POKE F,A:
NEXT
```

Next we must protect the character.

60 POKE 52,28

And last we must make use of the character and Print it:

```
70 POKE 36869,255: PRINT"(shift clr-home)@"
```

To prevent the screen being messed up by "Ready" appearing in multi-colour lettering the following line may be added:

80 GOTO 80

When the program is Run a multi-colour box will appear on the screen. You will be able to define your own character as easily as this.

To end with, here is a program to demonstrate what happens when one colour code is changed.

```
10 PRINT "(shift clr-home)"
```

```
20 DATA 0,10,10,42,42,130,142,142
```

```
30 DATA 0,160,160,168,168,130,142,142
```

# MULTI-COLOUR VIC GRAPHICS

## Poke

Poke 36879, screen and border colours

Poke 36878, code of auxiliary colours (0-15) × 16

Poke 646, code of character colour (0-7) + 8

Poke 30720 + address of point, code of character colour.

## Comments

See Vic manual for combination of screen and border colours

Used to change the character colour for a PRINT statement.

If you are poking onto the screen use this.

Table 1.

memory you must change the value of 36869,255 to use the characters. You may have more than 1 character in memory. To do this add more data in the Data statement in the routine and increase the value of F, that is, increase 7175 in the previous routine. Here is an example program going through the steps required to generate a multicolour character.

First we choose the colours: white, black, red, blue and Poke them into memory.

```
10 POKE 36879,24 put white and black into memory
```

```
20 POKE 36878,2×16 put red into memory
```

```
30 POKE 646,14 put blue into memory
```

Then we design the character, which will be a simple multicolour box, see figure 2. Now convert it to binary.

A A A A = 11111111

A C C A = 11101011

A C C A = 11101011

A C C A = 11101011

A C C A = 11101011

A C C A = 11101011

A C C A = 11101011

A A A A = 11111111

Then we convert the binary numbers to decimal:

11111111 = 255

11101011 = 235

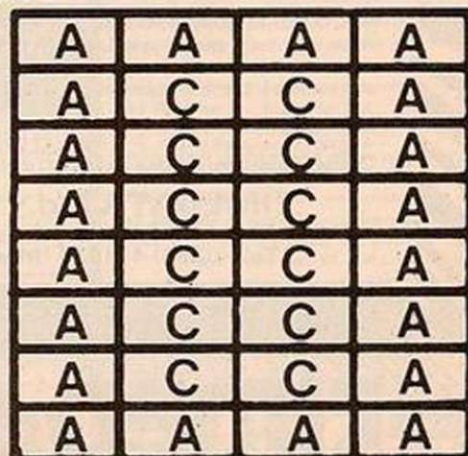
11101011 = 235

11101011 = 235

11101011 = 235

```
40 DATA 170,170,170,170,170,170,170,160
50 DATA 170,170,170,170,170,170,170,130
70 FOR F = 7168 TO 7199
80 READ A
90 POKE F,A
100 NEXT
110 FOR F = 7424 TO 7431: POKE F, PEEK
(25600 + F): NEXT
120 POKE 36869,255: POKE 52,28
130 FOR F = 1 TO 11
140 POKE 646,INT(RND(TI)×5)+10
150 PRINT "@A @A @A @A @A @A @A @A"
160 PRINT "BC BC BC BC BC BC BC BC"
170 NEXT
180 POKE 36878,(INT(RND(TI)×15)+1)×16
190 GOTO 180
```

Figure 2.





# SOFTWARE FILE

(continued from page 107)

```
770 IF N>255 THEN LET N=N-256
780 IF N<>62 THEN GOTO 750
790 POKE GP,0
800 PRINT AT 23,0; "YOU HAVE FA
```

```
FILED."
810 GOTO 630
1000 REM AUTOMATIC RUN
1010 SAVE "MAZE"
1020 RUN
```

## Slide show

Stewart Stallworthy,  
Rickmansworth,  
Hertfordshire.

**SPECTRUM**

THIS ROUTINE can be used to Poke to the screen any pre-defined picture or pictures. Whilst a Basic program would take approximately 60 seconds, this routine takes less than one second, and can therefore be usefully incorporated in Basic program. First draw a picture on the screen, then save it on tape —

```
SAVE "Picture" SCREEN
10 CLEAR 39997
```

```
20 FOR f=40000 TO 40048
30 INPUT a
40 POKE f,a
50 PRINT f,a
60 NEXT f
```

Run the program entering the appropriate decimal codes, and then Save on tape  
SAVE "Slide" CODE 40000,48

The next step is to load the picture bytes into memory. The routine is written to look for them at memory address 40100.

```
CLEAR 39997
LOAD "Slide" CODE 40000,48
LOAD "Picture" CODE 40100,6912
```

Now type

RANDOMISE USR 40000

and all should be revealed. Each picture takes 6912 bytes, and additional pictures may be located at other memory addresses, for example 47013 and 53926. If these are used, it will be necessary to make the routine look at the correct starting address, and this is done as follows. For picture starting at memory address 47013,

```
POKE 40008,165
POKE 40009,183
```

For picture starting at memory address 53926,

```
POKE 40008,166
POKE 40009,210
```

ADDRESS	MACHINE CODE	MNEMONIC	COMMENT	ADDRESS	MACHINE CODE	MNEMONIC	COMMENT
40000	38	LD H N		5	30	LD(NN)A	POKE 39998,A
1	28			6	52		
2	46	LD L N	6912 into HL	7	156		
3	1			8	24	JR DIS	Jump to 40013
4	34	LD(NN)HL	HL into memory	9	238		
5	62		at 39998 & 39999	40030	38	LD A(NN)	A=PEEK 39999
6	156			1	63		
7	17	LD DE NN	16384 into HL	2	156		
8	164			3	71	LD B A	B=A
9	156			4	61	DEC A	
40010	38	LD HL NN	16384 into HL	5	5	DEC B	
1	0			6	40	JRZ DIS	Jump to 40048 if
2	64			7	10		
3	26	LD A (DE)	A=PEEK DE	8	30	LD(NN)A	Poke 39999,A
4	119	LD (HL)A	POKE HL,A	9	63		
5	35	INC HL		40040	156		
6	19	INC DE		1	62	LD A N	255 into A
7	38	LD A(NN)	A=PEEK 39998	2	255		
8	62			3	50	LD(NN)A	Poke 39998,A
9	156			4	62		
40020	71	LD B A	B=A	5	156		
1	61	DEC A		6	24	JR DIS	Jump to 40013
2	5	DEC B		7	221		
3	40	JRZ DIS	Jump to 40030 if B=0	8	201	RET	Back to Basic
4	5						

## Sounds familiar

David Rees,  
Weybridge,  
Surrey.

**VIC-20**

IF YOU HAVE been gazing enviously all year at the BBC's Envelope command and have come home to the flat tones of a Vic-20, this program should cheer you up. It creates sounds at machine-code speed and gives an extra 11 registers for sound control.

Program 1 Pokes the machine code into memory, as the basic Vic has no assembler. Care must be taken when you type in the program, as one error in the code can cause the computer to stay in machine-code mode until you switch it off. The best method for dealing with this is to Save the program before you Run it. Then, if there is a mistake, you can load the program and check it through.

After running program 1, you can New it as

the machine code is all that is needed. It is located in the 256 bytes between 7424 and 7679, and the RAMtop has been moved down by this amount so that variables do not erase the program.

To use the command, you must first set the registers. There are 11 registers which have to be Poked to, and program 2 shows a convenient way of doing this.

The registers are as follows:  
DATA S,FR(1),FR(2),FR(3),T(1),T(2),T(3),  
V(1),V(2),V(3),DT

S is the voice chosen, 0 is the low sound, 1 is the medium sound, 2 is the highest sound and 3 is the white noise generator.

FR(1), FR(2) and FR(3) are the numbers that are added to the frequency of the voice chosen each cycle. If the number of the frequency rises above 255, the command wraps it around so that the value becomes 128 plus the value by which it exceeded 255. This means that if you want the command to play several scales of notes there will be no gap

in sound once the top of the scale is reached. The number that can be stored in these registers lies between 0 and 255.

T(1), T(2) and T(3) are the number of cycles through which the registers of each of the three parts are added to the sound and volume chosen. Again, the range of values lies between 0 and 255.

V(1), V(2) and V(3) are the numbers which change the volume of the sound each cycle. If 16 is Poked in here, volume does not change. A number lower than 16 lowers the volume, and a number greater than 16 raises it. The effective number can be calculated using 16-V(n).

DT is the delay time after each cycle is completed and can be between 0 and 255, with 0 having a delay time of 0.1ms. and the other numbers are measured in the computer in increments of 0.5ms.

If this seems complicated, program 3 gives the Basic equivalent of the main routine. You

(continued on next page)



# SOFTWARE FILE

(continued from previous page)

can also use this routine to demonstrate how much faster machine code is compared with Basic. Now all you have to do is type in SYS 7424 and the command will start. The registers will not change so SYS 7424 can be used any-

where and as many times as you want in a program. Registers can also be Poked individually so you can change a sound quickly.

You are now ready to create your own sound effects. The main advantage of this command is its speed. Basic cannot run faster than when

30 is Poked into the DT register. Lower numbers down to 5 can create fast, smooth sounds. When DT is 4 or less, sounds merge. These speeds are the most useful as you can mix tones and volume to create your own wave forms.

```
5 REM****PROGRAM 1****
10 POKE 51,255:POKE 52,28:POKE 55,255:POKE 56,28
20 FOR N=0 TO 152
30 READ A
40 POKE 7424+N,A
50 NEXT
60 DATA 24,173,252,29,41,31,141,252,29
70 DATA 173,253,29,41,31,141,253,29
80 DATA 173,254,29,41,31,141,254,29
90 DATA 173,245,29,41,31,141,245,29
100 DATA 169,10,133,1,169,144,133,2,169,0
110 DATA 141,241,29,141,242,29,169,246
120 DATA 133,251,169,29,133,252
130 DATA 172,245,29,177,1,172,241,29,24,113,251
140 DATA 144,3,24,105,128,24,172,245,29
150 DATA 145,1,172,241,29,200,200,200
160 DATA 238,242,29,200,200,200,177,251
170 DATA 24,109,14,144,233,15,24,184
180 DATA 141,14,144,172,255,29,192,0,240,8
190 DATA 162,120,202,208,253,136,208,248
200 DATA 172,241,29,200,200,200,177,251
210 DATA 205,242,29,208,181,238,241,29
220 DATA 169,3,205,241,29,208,6,169,0
230 DATA 141,241,29,96,169,0,76,46,29
999 REM****PROGRAM 2****
1000 FOR N = 0 TO 10:READ A:POKE 7669+N,A:NEXT N
1010 DATA 2,2,0,191,1,211,224,30,16,24,2
1020 SYS 7424:POKE 36876,0
```

```
99 REM****PROGRAM 3****
100 FOR A = 1 TO 3
110 FOR B = 1 TO T(A)
120 F = PEEK(36874 + S)+FR(A)
130 POKE 36874+S,(F AND 127)+128
140 V=PEEK(36878)-16+V(A)
150 POKE 36878,V AND 255
160 FOR T=0 TO DT:NEXT
```

TABLE 1

position	variable	range	function
7669	S	0-3	Voice choice(36874+S)
7670	FR(1)		Frequency + FR(n)
7671	FR(2)	0-255	each cycle
7672	FR(3)		
7673	T(1)		Number
7674	T(2)	0-255	of
7675	T(3)		cycles
7676	V(1)		Volume-16+V(n)
7677	V(2)	0-31	each
7678	V(3)		cycle
7679	DT	0-255	Delay time each cycle

EXAMPLES:

Helicopter  
1010 DATA 0,2,4,1,8,8,200,30,15,15,1  
1020 FOR N=0 TO 70:POKE 36874,240-N:SYS 7424:  
NEXT:POKE 36874,0

Laser Gun  
1010 DATA 2,2,4,1,8,8,200,30,16,0,2  
1020 POKE 36876,99:POKE 36878,15:SYS 7424

Footsteps(walking)  
1010 DATA 2,6,0,0,8,8,200,18,16,16,2  
1020 POKE 36878,0:FOR N=0 TO 10:SYS 7424:NEXT  
For running, use 1 as last number in data

Echo  
1010 DATA 2,0,127,1,0,6,9,16,16,16,0  
1020 FOR N=0 TO 9:POKE 36876,225  
1030 FOR A=0 TO 28:SYS 7424  
1040 NEXT A,N

## Field-gun

B Pearce,  
Bath,  
Avon.

BBC

THIS PROGRAM has been written for the BBC Model A Micro and uses almost all the available memory. Although it uses procedures which are, I believe, peculiar to the BBC Micro, there is no reason why it should not be adapted for Basic on other micros.

The game is for two players. There are two horizontal blocks representing a plain and a plateau, separated by mountains. The form of the mountains and the height of the plateau are both random, and the plateau may be right or left of the screen. Sited at a random position on the plain is a gun position, and another on the plateau. At the top of the screen a cross wind is specified, random left or right, random strength five to 40 mph in steps of 5 mph.

Player to start is specified, random left or right. The player is required to enter the gun elevation angle, which will be any angle between one to 90°, followed by muzzle velocity — any number from 1 to 20. On the second Return his gun fires a shell along the correct trajectory taking account of the effect of the wind. Each player fires in turn until one hits the others gun, when there is a flash and a bang. During the exchange, previous elevation and velocity settings are listed at each side of the screen for reference.

```
>L.L.
10REM "FIELD-GUN" by B.Pearce
20*TV255,1
30CLEAR:CLS:MODE4:VDU19,1,3,0:DIMA1(4):A%=32+4*RND(96):B%=832+4*RND(96):C%=256+4*RND(32)
40D%=INT(16*(RND(1)-0.5)):S=IFD%=0THEN40
50E%=2:F%=2:G%=RND(2):H%=1:I%=SGN(RND(1)-0.5):IFIX=1THEN80
60PROCFLAT(0,96,128,448):PROCSLOPE(448,1,832):PROCFLAT(832,96,C%,1280)
70MOVEAX,128:PROCFORT:MOVEBX,C%:PROCFORT:GOTO100
80PROCFLAT(1280,96,128,832):PROCSLOPE(832,-1,448):PROCFLAT(448,96,C%,0)
90MOVEAX,C%:PROCFORT:MOVEBX,128:PROCFORT
100PRINTTAB(1,1);"E V":PRINTTAB(33,1);"E V":IFD%>1THEN120
110PRINTTAB(12,1);"Wind ";D%:" mph":GOTO130
120PRINTTAB(12,1);"Wind ";D%:" mph ->"
130G%=3-G%:IFG%=2THEN150
140E%=E%+1:GOTO160
150F%=F%+1
160H%=H%+1:IFH%=2THEN170ELSE180
170IFG%=2THENPRINTTAB(35,10);"RIGHT":TAB(35,11);"FIRES":TAB(35,12);"FIRST"ELSEPRINTTAB(0,10);"
LEFT":FIRES":FIRST"
180IFH%=3THENPRINTTAB(0,10);"
"
190INPUTTAB(8,4)"Elevation (1-90) = "J%:IFJ%<10R3%>90THEN190
200IFG%=2THEN220
210PRINTTAB(0,E%);J%:GOTO230
220PRINTTAB(32,F%);J%
230PRINTTAB(8,4);"
"
240INPUTTAB(8,8)"Velocity (1-20) = "K%:IFK%<10R3%>20THEN240
250IFG%=2THEN270
260PRINTTAB(3,E%);K%:GOTO280
270PRINTTAB(35,F%);K%
280PRINTTAB(8,8);"
"
290IFG%=2THEN330
300IFIX=1THEN320
310MOVEAX,128:PROCSHOT(1,A%,128)
320MOVEAX,C%:PROCSHOT(1,A%,C%)
330IFIX=1THEN350
340MOVEBX,C%:PROCSHOT(-1,B%,C%)
350MOVEBX,128:PROCSHOT(-1,B%,128)
360IFG%=2THEN380
370IFAX>B%-16ANDAX<B%+16THEN390ELSE130
380IFAX>AX-16ANDAX<AX+16THEN390ELSE130
390PROCSOUND:PROCBANG:TIME=0:REPEAT:UNTILTIME=200:PRINTTAB(4,30);"Press SPACE BAR for another
game":c=GET:IFc=32THEN330
400DEFPROCFLAT(K,L,M,N):MOVEK,L:PLOT5,K,M:PLOT85,N,L:PLOT85,N,M:ENDPROC
410DEFPROCSLOPE(O,P,Q):FORR=1TO5:S%=0+64*P*R:TX=128+(C%-128)*R/5+(RND(128)-64)*R/2:PLOT85,SX,9
```



# SOFTWARE FILE

```

490 INPUT"      INPUT DESTRUCT LEVEL(1 TO 2)",DL
500 IF DL<1 OR DL>2 THEN 490
510 INPUT"      INPUT DIFFICULTY LEVEL(1 TO 5)",C
520 IF C<1 OR C>5 THEN 510
530 PRINT TAB(5,21);"PRESS F0 TO RESTART"
540 PRINTTAB(5,22);"PRESS ANY KEY TO START":ST#=GET$
550 ENDPROC
560 DEFPROC CRASH:SOUND0,-15,100,18
570 FOR CRASH=1 TO 10
580 FOR RIGHT=1 TO 10:VDU23;13,RIGHT,0;0;0:NEXT RIGHT
590 FOR LEFT=10 TO 1 STEP-1:VDU23;13,LEFT,0;0;0:NEXT LEFT
600 NEXT CRASH:ENDPROC
610 DEFPROC BOMB
620 FIRE=1:XB=X+2:YB=Y:IF XB>=20 THEN PROCDESTRUCT
630 ENDPROC
640 DEFPROC DESTRUCT
650 SOUND1,1,100,2:FIRE=0:PRINTTAB(XB,YB);DESTRUCT#:SOUND0,-5,100,2
660 ENDPROC
670 DEFPROC INIT:X=1:Y=3:FIRE=0
680 BOMB$=" "+CHR$10+CHR$8+CHR$232
690 AERO$=" "+" "+CHR$230+CHR$231
700 DROP$=" "+CHR$10+CHR$8+CHR$233
710 DROPT$=" "+CHR$10+CHR$8+CHR$234
720 ENDPROC
730 DEFPROC MOVE:X=X+1:IF X>=20 THEN X=0:Y=Y+1
740 ENDPROC
750 DEFPROC LAND
760 PRINTTAB(1,2);"WELL DONE":RESTORE780:FOR MUSIC=1 TO 10:READ A,B
770 SOUND2,-10,A,B:NEXTMUSIC:END:ENDPROC
780 DATA129,10,117,5,121,5,129,10,101,10
790 DATA121,5,129,5,137,5,145,5,149,5
800 DEFPROC DROP:XD=X+2:XDT=X+3:YD=Y-2:YDT=Y-2
810 FOR DELAY=1 TO 300:NEXT DELAY
820 YD=YD+1:PRINTTAB(XD,YD);DROP$
830 VDU26
840 IF POINT(XD*64+32,(29-YD)*32)=0 OR YD>=28 THEN 850 ELSE 820
850 YDT=YDT+1:PRINTTAB(XDT,YDT);DROPT$
860 VDU26
870 IF POINT(XDT*64+32,(29-YDT)*32)=0 OR YDT>=28 THEN END ELSE 850:ENDPROC

```

## Quick copy

Peter Hintjens,  
Edinburgh.

VIC-20

THIS BASIC program will load into a chosen area of memory a routine that, whenever Ctrl P is pressed, will produce a quick copy of the Vic screen.

The machine-code routine becomes part of the system interrupt — IRQ — and is called 60 times a second when it can look at the keyboard and take whatever action is necessary.

Enter, check and Save the program, then Run it. At the start you will be asked where the code should go. Normally this will be at the top of memory, but in some cases, for example when using machine code that needs this area, you will want to specify somewhere else. In the first case the program will lower the memory pointers to protect the code, but if you specify a location you must protect it as necessary. A useful free area of memory is the

3K expansion block, if both a 3K and 8K or 16K expansion are fitted.

As the program loads the machine-code data, you may get a number of error messages of the form

?DATA ERROR IN

showing that that data line has been incorrectly entered. If the check sum at the end of each data line fails to pick up the error, the total count — TT — should catch it. When any data error is found, a flag ER is set and the program continues, perhaps to find more errors. After the load, if ER is set then the run is aborted and the memory pointers restored to their initial value, line 570.

As the program is relocatable, certain values must be altered to suit its start — specifically the start of the IRQ wedge. The screen page is also Poked into the routine so that the code will run in any memory size.

When the data has been loaded and assuming that there have been no errors, you will be told the actual start of the routine.

Make a note of this number because to prime the code you must type: SYS, then the start address, then press Return. Pressing Stop/Restore will stop the effect of CtrlP.

You can alter the printing parameters — normally double-width and minimum line feed — by the following Pokes:

POKE (START)+99, 15;

POKE (START)+176, 15

for single-width printing and

POKE (START)+161, 15

for normal line feeds, useful when dealing only with screens of text. The machine-code routine will automatically print in upper or lower case as set by the shift keys.

Once you have the program working you may take out the data checks — remove the ninth data byte of each line and change lines as follows. 310 to 460 replaced by:

310 FOR J = 0 TO 207

320 READ DA:POKE PO + J, DA

330 NEXT J

Lines 540 to 570 would be deleted.

```

10 REM** VIC SCREEN DUMP **
20 REM** BY **
30 REM** P. HINTJENS **
40 REM
50 REM
60 M1 = PEEK (56): M2 = PEEK (55): REM** INITIAL MEMORY POINTERS
70 REM
80 REM** GET CODE START
90 REM
100 PRINT "WHERE SHOULD THE CODE RESIDE ?"

```

```

110 PRINT "01 - TOP-OF-MEMORY"
120 PRINT "02 - SOMEWHERE ELSE"
130 PRINT "00 CHOICE ?"
140 GET A$:IF A$ = "2" GOTO 240
150 IF A$ <> "1" GOTO 140
160 REM
170 REM** DEALLOCATE TOP OF MEMORY BY 256 BYTES
180 REM
190 PO = 256 * M1 + M2 - 256:POKE 56, M1 - 1:POKE 52, M1 - 1
200 PRINT "J":GOTO 270

```

(continued on page 117)