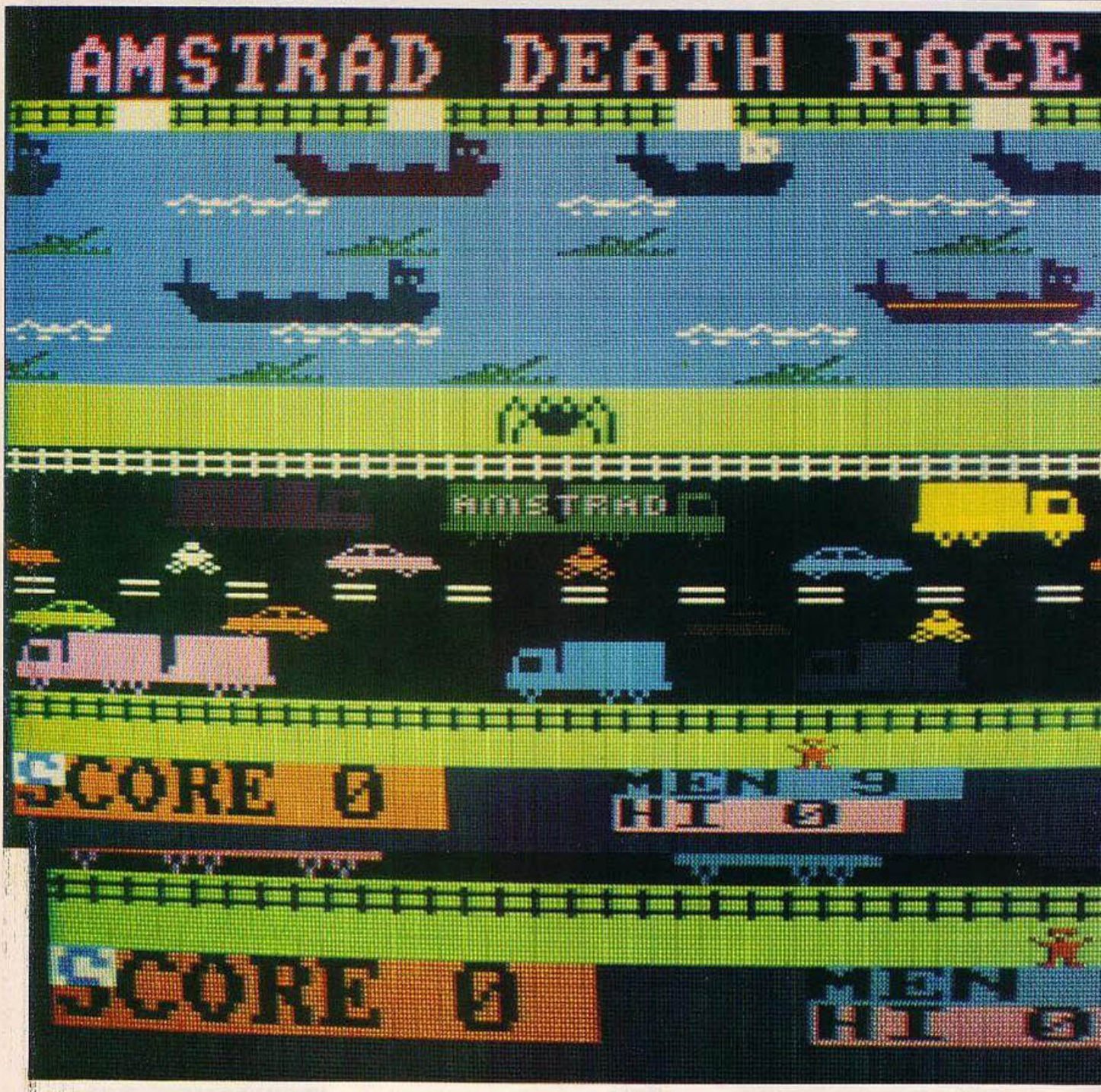



DEATH CHASE STARTED as a straightforward conversion attempt on a Spectrum program published by *Your Computer* some time ago. In the process some of the internal goings on of the Amstrad were revealed and the result is a playable and colourful game. This article gives the finished program and also details of screen display configuration and how to add extra commands to your Basic — lateral scrolling, enlarged printing and screen character reading.

In converting Spectrum Cross to Amstrad Death Race there were three fundamental problems to be faced. First the different size of the text screen, second the way the screen start moves around in memory, and third the way colour is mapped.

The Amstrad has three screen modes. Mode 2 is the high definition, two colour mode. Mode 1 is medium resolution with 40 columns of text in four colours. Mode 0 is the multi-colour mode with 20 column text. I decided to reduce the number of cars and lorries on the road to enable me to make the most of the colours.

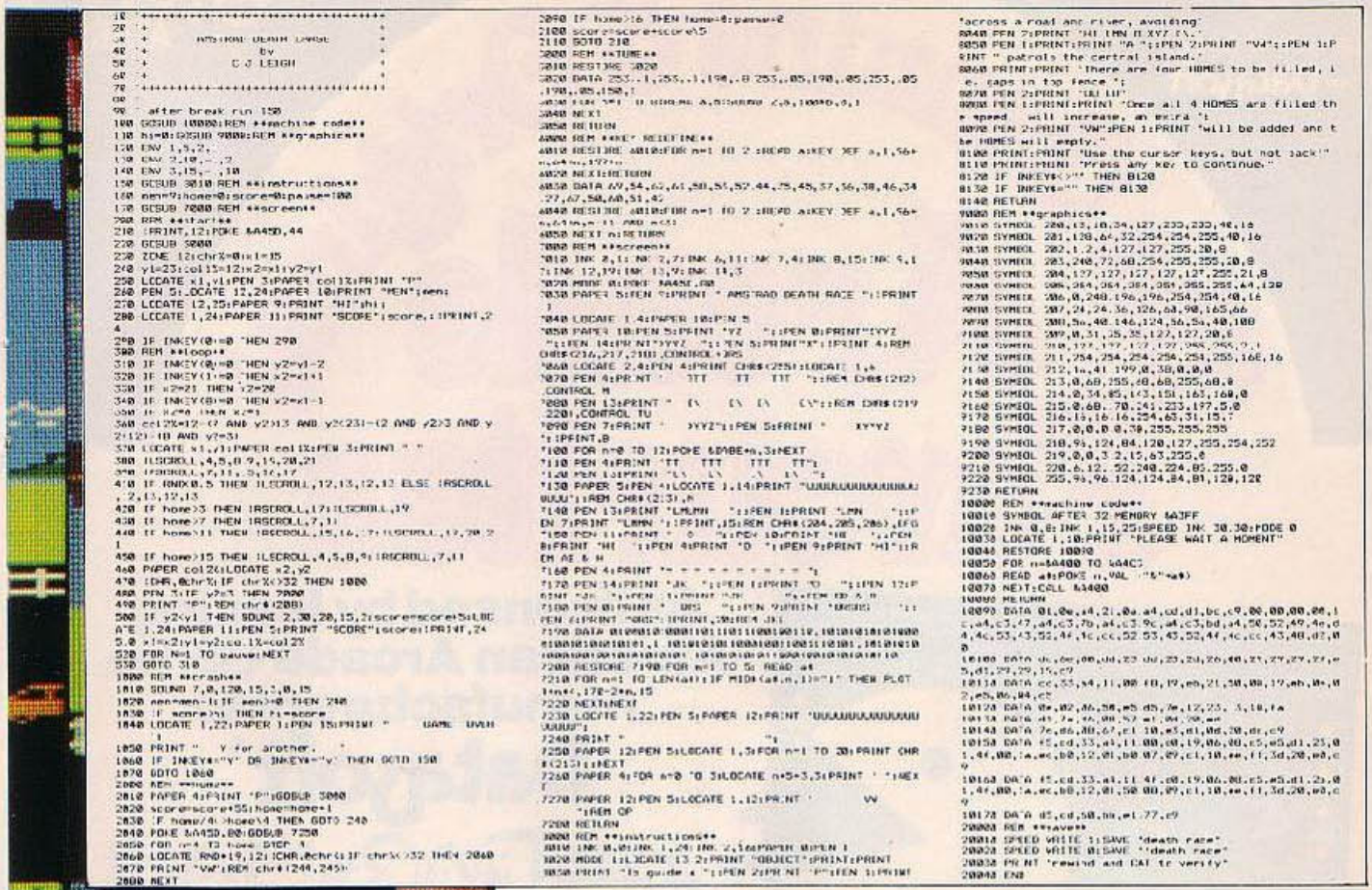
Although the number of columns would be less than the Spectrum's, the number of lines available was more — 25 instead of 22. Since a double height printing routine has long been one of my favourites, the answer was obvious.

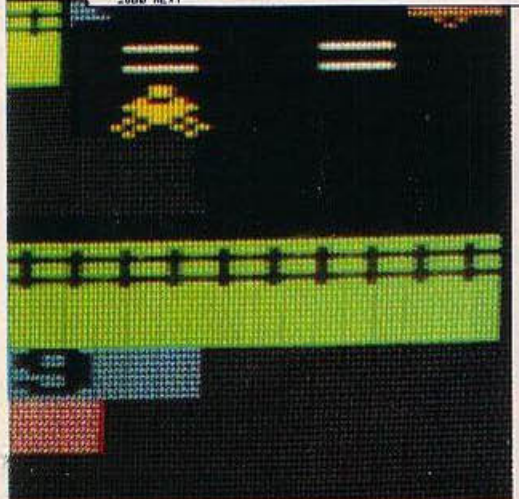
The screen display occupies the area of memory from hex &c000 up and is composed of 25 lines each with eight rows of 80 bytes. Some arithmetic tells you that there is a little memory left over at the end of the screen.

This fact, coupled with hardware scrolling of the screen when for example listing a program, means that the byte with address &c000 may not in fact be expected top left-hand corner. It could be anywhere, even off the screen altogether. However, after a mode change the screen is reset so that solves many of the problems.

Now supposing the top left hand corner is &c000 then we might expect the byte underneath to be &c050, but as on the Spectrum that is the top byte of the second line — pixel row nine. Thinking again and given that 25 lines times 80 bytes = &7d0 we might expect that second byte down to be &c7d0. In fact it is &c800. This tidy figure is arrived at by simply tagging roughly half a line on after the bottom of the screen.

Colour mapping is at its simplest in Mode 2. Here the eight pixel width of a character is matched by the eight bits in one byte. Hence 80 characters per line means 80 bytes across





# DEATH RACE

C J Leigh with a very colourful version of Frogger for the Amstrad.

the screen. Each bit reset is paper colour — usually pen 0 — and each bit set is usually pen 1.

In Mode 1 things are more difficult since the character width covers two display bytes with each pixel pen being represented by the state of two bits, so giving the four colours available. These bits are not consecutive but the byte is split in two with the most significant bit of each half representing the leftmost pixel of the byte.

In Mode 0 the same general idea is followed. The character width covers four bytes, each byte representing two pixels. The byte is split into four with the most significant bit of each quarter representing the left hand pixel of the pair. Each of the quarters contributes a bit to a 4 bit number with the bits weighted 1,4,2,8 indicating the pen number used.

Extensions to Amstrad Basic can be made very easily by setting up a name table and jump table as in listing 1 along with a four byte buffer and the log on routine at &a400. Each command must be preceded by a bar character available on the @ key.

The scrolling routines are straightforward whilst the double height printing routine is used after the Rom has printed to the screen. The machine code reads the bytes of colour off the screen and doubles them up. Screen character reading is done using a Rom system call.

Using machine code in this way is very attractive and is made even easier since it is possible to pass parameters from Basic to your own code and back again. These parameters are pointed to by the IX register pair with the number of parameters given in the A register.

In fact, the IX pair point to the low byte of the last parameter passed and is the base address of a parameter buffer set up for you by the system. If there is only one parameter it is available in the DE register pair and if the parameter is an integer variable preceded by @ then it is the address of the variable that is passed to the machine code. That allows values to be transferred back to Basic.

The graphics characters in the program listing may be entered most easily by entering and running the short routine at line 6000 first. This copies the Spectrum system of making the keyboard produce the appropriate graphic character when Control is pressed with one of the keys A to U.

The routine at 6040 returns the keys to normal if required. Follow the guidance of the

*(continued on page 91)*

(continued from page 89)

Rem statements when typing in the graphics. In addition you may find it easier to enter the machine code if you first enter Key 138,"," to change the decimal point on the numeric pad to a comma.

Finally, since the user graphics do not allow you to make the most of the Amstrad's colour, three different ways of going this have been introduced in the screen construction routine. They may be omitted if preferred since they slow down the screen printing.

To slow down the game increase the value of the pause used at the end of the main loop — set at line 160 — or to make it much easier remove some of the traffic. Every time you get four men home you will find extra spiders added and the speed will increase.

You can avoid the chore of typing in the program by sending £3 with your name and address to Death Race, 12 The Bassetts, Cashes Green, Stroud, Gloucestershire GL5 4SJ. Please make cheques payable to C J Leigh. ■

*Listing 1.*

```
                ORIGIN  A400H
0102A4  INITIAL LD BC,COMTAB
210A04          LD HL,BUFFER
CDD1BC          CALL BCD1H      ; tell the system
C9              RET
00000000BUFFER  DEFS 4          ; system work space
1CA4    COMTAB  DEFW NMETAB     ; command table
C347A4          JP PRINT
C37BA4          JP LSCROLL
C39CA4          JP RSCROLL
C3EDA4          JP CHR
505249  NMETAB  DEFM PRI        ; command names
4ED4            DEFM N,T+80H
4C534352        DEFM L,S,C,R
4F4CCC          DEFM O,L,L+80H
52534352        DEFM R,S,C,R
4F4CCC          DEFM O,L,L+80H
4348D2          DEFM C,H,R+80H
00              DEFB 0          ; end marker
DD6E00  ADDRESS LD L,(IX+0)     ; calculate screen start
DD23            INC IX
DD23            INC IX
2D              DEC L           ; start at line zero
2600            LD H,0
29              ADD HL,HL       ; multiply by two
29              ADD HL,HL
29              ADD HL,HL
29              ADD HL,HL
E5              PUSH HL
D1              POP DE
29              ADD HL,HL
29              ADD HL,HL
19              ADD HL,DE       ; times eighty
C9              RET
CD33A4  PRINT   CALL ADDRESS
1100F8          LD DE,F800H     ; bottom of first line
19              ADD HL,DE
EB              EX HL,DE        ; source
215000          LD HL,50H       ; move to next line
19              ADD HL,DE
EB              EX HL,DE        ; destination in DE
0E02            LD C,2          ; two lines
E5      HALF    PUSH HL
0604            LD B,4          ; four pixel rows
C5      ROW     PUSH BC
0E02            LD C,2          ; one row becomes two
0650    STRETCH LD B,50H        ; length of line
E5              PUSH HL
D5              PUSH DE
7E      NEXT    LD A,(HL)       ; read byte
12              LD (DE),A       ; write byte
23              INC HL
13              INC DE
10FA            DJNZ NEXT       ; next character
D1              POP DE
7A              LD A,D
D608            SUB 8           ; up a pixel line
57              LD D,A
E1              POP HL          ; recover read start
0D              DEC C
20ED            JRNZ STRETCH    ; do it again
7C              LD A,H
D608            SUB 8
67              LD H,A
C1              POP BC
10E3            DJNZ ROW        ; next read row
D1              POP DE          ; destination = source
0D              DEC C
20DC            JRNZ HALF       ; top half
C9              RET
F5      LSCROLL PUSH AF         ; save parameter count
CD33A4          CALL ADDRESS
1100C0          LD DE,C000H     ; start of top line
19              ADD HL,DE       ; screen address
0608            LD B,8
C5      BACK    PUSH BC
E5              PUSH HL
D1              POP DE
23              INC HL          ; HL to DE
014F00          LD BC,4FH       ; line length less one
1A              LD A,(DE)       ; save first byte
EDB0            LDIR            ; move line
12              LD (DE),A       ; wraparound byte
01B007          LD BC,7B0H
09              ADD HL,BC       ; next pixel row
C1              POP BC
10EE            DJNZ BACK
F1              POP AF
3D              DEC A           ; reduce parameter count
20E0            JRNZ LSCROLL    ; next line?
C9              RET
F5      RSCROLL PUSH AF
CD33A4          CALL ADDRESS
114FC0          LD DE,C04FH     ; end of top line
19              ADD HL,DE
0608            LD B,8
C5      AGAIN   PUSH BC
E5              PUSH HL
D1              POP DE
2B              DEC HL          ; HL to DE
014F00          LD BC,4FH
1A              LD A,(DE)
EDB8            LDDR
12              LD (DE),A
015008          LD BC,850H
09              ADD HL,BC
C1              POP BC
10EE            DJNZ AGAIN
F1              POP AF
3D              DEC A
20E0            JRNZ RSCROLL
C9              RET
D5      CHR     PUSH DE         ; save parameter address
CD60BB          CALL BB60H      ; read screen character
E1              POP HL
77              LD (HL),A       ; put value in variable
C9      END     RET
END A4C3H
```