VIDEO OUTPUT

One of the major problems on most home computers is that of producing the video display. Especially for a colour display of the resolution used in the Spectrum, data has to be copied from the video memory to the display at a continuous and fast rate. This poses problems when the CPU wants to read from the video memory at the same time as the ULA. Two devices cannot address different places in a memory chip simultaneously.

Most other computers use one of two methods to resolve this. In the first method the CPU has priority. This produces 'snow' on the screen, representing video information which was not displayed because the CPU was using the memory. Snow is annoying but the CPU runs at full speed. The second method gives the video circuit priority during a television field scan. This requires that the CPU should only be able to operate during the field sync time. Using this method does eliminate snow, but the computer operates very slowly. Neither of these methods is satisfactory.

Your Spectrum uses a very ingenious way to get over the problem. Let us assume that the ULA is accessing video memory. The CPU can simultaneously access the ROM or extra 32K of RAM without any bus contention occuring. The ULA and video memory have their address and data buses separated from the rest of the system by 330 ohm and 470 ohm series resistors respectively. Normally therefore, there are two separate systems operating independently of one other. The ULA outputting video, the Z80A operating BASIC. But then, the CPU may wish to access the 16K of RAM which the ULA is using, because this contains all of the BASIC system variables as well as the video memory. At this point, the ULA realises what is about to happen (by monitoring A14 and A15), so it hastily stops the CPU clock. The Z80A doesn't notice this because its only way of measuring time is by assuming that its clock input is constant. The ULA can then let the CPU have access to the memory for a few hundred nanoseconds when there is a brief gap in video output.

How does this novel design feature affect any programs which you may wish to write? In BASIC, it will always appear totally transparent to the user, but if you run a machine code program from the affected 16K of RAM, the timing of routines will not be constant. Normally this would not be important, however, routines with a critical timing loop in them, such as a BEEP routine, will not operate properly. The BEEP in BASIC does work properly because the machine code that operates it is running from the ROM.

A BIT ABOUT INTERRUPTS

If you start writing machine code routines, you may wish to use vectored interrupts. Whenever the CPU is interrupted (eg by a piece of your external hardware), you can make it run a routine whose address can be found at a position in memory pointed to by a 16 bit pointer. This 16 bit pointer is made up from the CPU I register contents and an 8 bit byte supplied by the interrupting device. The I register defines A8 — A15 in this case. Assume now that your interrupt routine address is held in the 16K of RAM used by the ULA. The I register will therefore contain a decimal number between 64 and 127 (A15 = 0,

A14 = 1). During every instruction cycle of the CPU, a refresh will occur. Refresh puts out the I register contents onto A8 — A15 and operates MREQ. This combination of signals confuses the ULA into expecting a read to or write from the video memory by the CPU. This doesn't occur so the ULA gets confused. So confused in fact that it omits some video output, causing snow on the screen!

Try running this small BASIC program to set the I register so that you can see the snow for yourself. Remember that values of I between 64 and 127 create snow.

```
10  CLEAR 32499
20  INPUT "Enter the value for I register ";v
30  POKE 32500,62 :REM LD A,v
40  POKE 32501,v
50  POKE 32502,237 :REM LD I,A
60  POKE 32503,71
70  POKE 32504,201 :REM RET
80  LET a = USR 32500
90  GO TO 20
```

Line 10 stops BASIC using memory above address 32499. Lines 30 — 70 put a machine code program into memory. You will be prompted for the value required in the I register. The machine code program then sets the I register when it is called from BASIC at line 80. Return from the machine code jumps to line 90, causing the whole BASIC program to be repeated.

CLOCKS

The ULA generates its own master clock. This master clock frequency is held constant at 14 MHz by the crystal X1. The 14MHz master clock is divided by 2 to produce the correct video dot frequency of 7MHz. Division by two again reduces the frequency to 3.5MHz which is fed to the Z80A. This 3.5MHz clock is not constant and can be stopped for short periods by the ULA to override the CPU during video memory accesses.

KEYBOARD AND CASSETTE INPUTS

The keyboard is dealt with in chapter 7. In summary, if you read in a byte from I/O address 254, bits D0 — D4 will hold the logic level at the KBD13 — KBD9 inputs to the ULA. These bits are set to 0 if a key is pressed and to 1 if it isn't. D6 holds the input level at the EAR input from your casette recorder.

BUZZER, CASSETTE & BORDER COLOUR OUTPUTS

Bits D2, D1 and D0 in a byte output to port 254 set the border colour. All at logic 1 generates white, all at logic 0 generates black.

Although the cassette and buzzer inputs and outputs are all connected to ULA pin 28 (see fig 7d), they operate independently. Bit D3 in output to port 254 drives the MIC socket, bit D4 drives the buzzer. This table explains why:

D4 (buzzer)	D3 (MIC)	Voltage at ULA pin 28
0	0	+0.75 volts
0	1	+1.3 volts
1	0	+3.3 volts