

Chapter 8 - The ULA Display

The ULA reproduces the picture model of the ZX Spectrum: a single 256×192 monochrome bitmap painted in pairs of colours chosen per 8×8 cell from a small palette. It is the simplest of the six picture sources, and the easiest to drive directly from BASIC. The ULA sits on compositor layer 15, so it draws on top of every other source except Voodoo 3D.

8.1 What the ULA can show

Item	Value
Display area	256×192 pixels
Character grid	32×24 cells of 8×8
Border	32 pixels on every side
Total frame	320×256 pixels
Colours	15 unique (8 base + 8 bright, but black has no bright form)
VRAM	6912 bytes (6144 bitmap + 768 attributes)
Flash rate	Toggle every 32 frames (about 1.6 Hz at 50 Hz)

Every pixel is either INK (foreground) or PAPER (background). The choice of INK and PAPER comes from an **attribute byte** that covers a whole 8×8 cell, so any two adjacent cells can use different colour pairs but pixels within one cell are limited to two colours. This is the Spectrum's famous **attribute clash**, and the ULA reproduces it exactly.

8.2 BASIC keywords

ULA introduces every ULA subcommand.

Form	Effect
ULA ON / ULA OFF	Enable / disable the chip.
ULA BORDER $colour$	Write $colour$ (0-7) to the border register.
ULA INK $colour$	Set the current foreground colour (0-7).
ULA PAPER $colour$	Set the current background colour (0-7).
ULA BRIGHT $flag$	Set the current bright flag (0 or 1).
ULA FLASH $flag$	Set the current flash flag (0 or 1).
ULA PLOT x,y	Set the bitmap pixel at (x, y) using the ZX-style address.
ULA CLS [$attr$]	Clear the bitmap to zero and fill the attribute area with $attr$ (default \$38 = white paper, black ink).
ULA ATTR $x,y,attr$	Write the attribute byte at character cell (x, y).

A minimal BASIC program that draws a red diagonal on a black background. The attribute byte chosen for ULA CLS decides the colour every PLOT will appear in, because PLOT only sets bits in the bitmap - the colour comes from the cell's attribute:

```

10 ULA ON
20 ULA CLS &H02           : REM black paper (bits 5-3 = 0), red ink (bits 2-0 = 2)
30 FOR I=0 TO 191
40 ULA PLOT I, I
50 NEXT I

```

To change colours under an existing picture, write new attribute bytes with ULA ATTR after plotting.

This version fills the attribute area with a bright colour grid, then plots diagonal pixels through those cells:

```

10 ULA ON
20 ULA CLS &H00
30 FOR Y=0 TO 23
40 FOR X=0 TO 31
50 A=&H40+((Y AND 7)*8)+(X AND 7)
60 ULA ATTR X,Y,A
70 NEXT X
80 NEXT Y
90 FOR I=0 TO 191
100 ULA PLOT I, I
110 ULA PLOT 255-I, I
120 NEXT I

```

The diagonals keep the same bitmap shape as they cross the screen, but their colours change at every 8 x 8 attribute cell.

ULA INK, ULA PAPER, ULA BRIGHT, and ULA FLASH set BASIC's working attribute state but do **not** touch the attribute area themselves. The state is consumed by routines that write characters or whole rectangles. To set the colour of a specific cell directly, use ULA ATTR.

8.3 The register block

The ULA's register block is small. It runs \$F2000-\$F2017, six 32-bit words. Only the low byte of each register is meaningful.

Address	Name	Purpose
\$F2000	ULA_BORDER	Border colour. Bits 0-2; upper bits ignored.
\$F2004	ULA_CTRL	Master enable and IRQ enable.
\$F2008	ULA_STATUS	Status bits (read-only).
\$F200C	ULA_ADDR_LO	Low byte of paged VRAM address latch.
\$F2010	ULA_ADDR_HI	Upper bits of the 13-bit VRAM address latch.
\$F2014	ULA_DATA	Read/write the VRAM byte at the latched address.

8.3.1 ULA_CTRL bits

Bit	Name	Meaning
0	ENABLE	Master enable.
1	VBLANK_IRQ_EN	Raise an interrupt when vertical blank begins.
2	AUTO_INC	Auto-increment the ULA_DATA latch after each access.

8.3.2 ULA_STATUS bits

Bit	Name	Meaning
0	VBLANK	Set during the vertical-blank interval.

8.3.3 The paged VRAM port

The ULA exposes its VRAM in two ways. The full 6912-byte VRAM window is mapped directly at \$FA000-\$FBAFF (see §8.4), but the chip also offers a paged port for CPUs that cannot reach that window cheaply:

1. Write the byte offset into the latch with ULA_ADDR_L0 and ULA_ADDR_HI (13-bit value, 0-6911).
2. Read or write ULA_DATA to access that byte.
3. If AUTO_INC is set in ULA_CTRL, the latch advances after every access.

This is the only path on chips with a small address space (for example the 6502 and Z80, which use the equivalent port-I/O mappings).

8.4 The VRAM aperture

The full 6912 bytes of VRAM are mapped at \$FA000-\$FBAFF for 32-bit and 64-bit CPUs. The layout is the Spectrum standard:

Range	Size	Contents
\$FA000-\$FB7FF	6144 B	Bitmap.
\$FB800-\$FBAFF	768 B	Attribute bytes.

8.4.1 Bitmap addressing

The bitmap is not stored in a simple row-major layout. For pixel (x, y) the byte address inside the bitmap is:

```
addr = ((y & $C0) << 5)      ; row group (0, 64, 128)
      | ((y & $07) << 8)     ; pixel row inside row group
      | ((y & $38) << 2)     ; character row inside row group
      | (x >> 3)             ; column
```

The bit inside the byte is $7 - (x \& 7)$ (bit 7 is the leftmost pixel of the eight). This is the same arrangement as the original Spectrum, and it is the reason a ULA PLOT keyword exists at all: the address arithmetic above is awkward to write each time by hand.

When you do want direct byte access, use POKE8 into the aperture. This listing draws a shallow red line by calculating the bitmap byte, reading it, setting one bit, and writing it back:

```
10 ULA ON
20 ULA CLS &H02
30 FOR X=0 TO 255
40 Y=INT(X*3/4)
50 A=((Y AND &HC0)*32)+((Y AND 7)*256)+((Y AND &H38)*4)+INT(X/8)
60 B=7-(X AND 7)
70 V=PEEK8(&H000FA000+A) OR 2^B
80 POKE8 &H000FA000+A,V
90 NEXT X
```

8.4.2 Attribute bytes

Each attribute byte covers one 8×8 cell, in row-major order:

```
bit 7 6 5 4 3 2 1 0
     F B P P P I I I
```

Bits	Field	Meaning
0-2	INK	Foreground colour, 0-7.
3-5	PAPER	Background colour, 0-7.
6	BRIGHT	Intensify both INK and PAPER.
7	FLASH	Swap INK and PAPER every 32 frames.

The eight base colours are:

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

BRIGHT brightens every colour except black, which is why the total comes to 15 unique colours and not 16.

8.4.3 Paged-port byte access

The paged VRAM port is slower than the direct aperture, but it is the common path for CPUs with a smaller address space. The latch is an offset inside ULA VRAM, not a full system address. This example turns on auto-increment, writes the first bitmap scanline as solid pixels, then writes the first attribute row as bright yellow ink on blue paper:

```
10 POKE32 &H000F2004,5           : REM ENABLE + AUTO_INC
20 POKE32 &H000F200C,0
30 POKE32 &H000F2010,0
40 FOR I=0 TO 31
50 POKE32 &H000F2014,&HFF
60 NEXT I
70 POKE32 &H000F200C,0
80 POKE32 &H000F2010,&H18
90 FOR I=0 TO 31
100 POKE32 &H000F2014,&H56
110 NEXT I
```

\$1800 is the start of the attribute section, so ULA_ADDR_HI receives \$18 and ULA_ADDR_LO receives 0.

8.5 Interrupts

The vertical-blank interval is the cleanest time to update VRAM, because the picture is not being scanned. Two ways to detect it:

- Poll ULA_STATUS bit 0.
- Enable VBLANK_IRQ_EN in ULA_CTRL; the CPU then receives an interrupt at the start of each VBlank.

The Spectrum-style INT vector for VBlank on the x86 coprocessor is \$20; other CPUs receive the interrupt through their normal controllers.

ULA_STATUS bit 0 is acknowledged by reading it. A polling loop therefore both waits for VBlank and clears the pending status:

```
10 ULA ON
20 IF (PEEK32(&H000F2008) AND 1)=0 THEN 20
30 ULA BORDER (PEEK32(&H000F2000)+1) AND 7
40 GOTO 20
```

The border cycles once per VBlank. For a non-interrupt program this is the safest time to update attributes or a small bitmap region.

8.6 Setup order, side effects, and limits

The shortest path to a Spectrum-style picture:

1. Write 1 to ULA_CTRL to enable the chip.
2. Clear the bitmap and write the attributes you want.
3. Plot pixels by writing into the bitmap with the formula above, or by setting one bit at a time through ULA_DATA.
4. Change the border colour with ULA_BORDER for cheap visual effects (loading-screen colours, music-driven flashes).

Important side effects and limits:

- Only the low byte of each ULA register is meaningful.
- ULA_BORDER uses bits 0-2; upper bits are ignored.
- ULA_STATUS bit 0 clears when read.
- ULA_DATA returns 0 for an out-of-range latch. Writes outside 0-6911 are ignored.
- With AUTO_INC set, the latch advances after each ULA_DATA read or write and wraps to 0 after byte 6911.
- The direct aperture supports byte, word, and double-word access, but bitmap work is easiest with POKE8 because every byte packs eight pixels.
- FLASH swaps INK and PAPER every 32 frames. It does not alter VRAM; it only changes how flagged attributes are drawn.
- Attribute clash is not an error. It is the intended rule: one attribute byte controls all 64 pixels in its cell.

The next chapter covers the Voodoo 3D rasteriser, the top of the compositor stack on layer 20.