

Chapter 5 - The VGA Controller

The VGA is the second of Intuition Engine's six picture sources. It sits on compositor layer 10, so it draws on top of the VideoChip (layer 0) and below every other source. Of all six chips, VGA is the easiest to drive from BASIC: most of the BASIC graphics keywords - SCREEN, CLS, PLOT, PALETTE, COLOR, LOCATE, SCROLL, VSYNC - operate on the VGA.

This chapter describes what VGA can show, where its memory lives, and how its registers work. The BASIC keywords are listed first because most readers will use them long before they touch the registers directly.

5.1 What VGA can show

VGA has four modes. You select one by writing its mode number to the VGA_MODE register, or with the BASIC SCREEN statement.

Mode value	Name	Resolution	Colours	Memory model
&H03	Text	80 × 25 characters (640 × 400 pixels)	16 fg, 8 bg	Character/attribute pairs in text buffer.
&H12	Mode 12h	640 × 480 pixels	16	Four bit-planes.
&H13	Mode 13h	320 × 200 pixels	256	Linear (chain-4).
&H14	Mode X	320 × 240 pixels	256	Four planes, unchained.

The mode names follow familiar VGA conventions, but the addresses and examples in this chapter are the ones to use on Intuition Engine.

5.2 BASIC keywords

Each of the keywords below has a one-line summary here and a full entry in Chapter 2.

Keyword	Effect
SCREEN n	Set VGA mode n. Enables VGA.
SCREEN ON / SCREEN OFF	Enable / disable VGA.
CLS [colour]	Clear the screen using the blitter.
PLOT x, y [, colour]	Plot a single pixel. Mode 13h only. Default colour is 15.
PALETTE i, r, g, b	Set DAC palette entry i to (r, g, b). Each component is 0-63.
COLOR fg [, bg]	Set the text-mode attribute byte used by PRINT. fg is 0-15; bg is 0-15.
LOCATE row, col	Place the text cursor at zero-based (row, col).
SCROLL dx, dy	Shift the displayed start address by dy rows and dx characters.
VSYNC	Wait for the vertical retrace by polling VGA_STATUS.

A minimal BASIC program that puts up Mode 13h and draws a single red pixel at the centre of the screen:

```

10 SCREEN &H13
20 PALETTE 1, 63, 0, 0
30 PLOT 160, 100, 1

```

Line 10 selects the 320 x 200 linear 256-colour mode and turns the VGA source on. Line 20 makes palette entry 1 bright red. Line 30 writes colour index 1 into the byte for pixel (160, 100). You should see one red point near the centre of the picture.

Text mode is just as direct:

```

10 SCREEN &H03
20 COLOR 14, 1
30 LOCATE 5, 10
40 PRINT "VGA TEXT"

```

Here SCREEN &H03 selects the 80 by 25 text buffer. COLOR 14, 1 sets yellow text on a blue background, and LOCATE moves the cursor before PRINT writes character/attribute pairs into \$B8000.

5.3 The register block

The VGA register block runs from \$F1000 to \$F13FF. The control registers are 32-bit words at 4-byte-aligned addresses. The direct palette window at \$F1100 is byte-addressed, so use POKE8 and PEEK8 there. The block is laid out as a set of small index/data pairs, each one mirroring a section of the IBM VGA register file.

5.3.1 Core control

Address	Name	Purpose
\$F1000	VGA_MODE	Mode select. Writing &H03, &H12, &H13, or &H14 chooses a mode.
\$F1004	VGA_STATUS	Status (read-only).
\$F1008	VGA_CTRL	Master enable. Bit 0 = on.

VGA_STATUS bits:

Bit	Name	Meaning
0	VSYNC	Vertical sync active.
1	HSYNC	Horizontal sync (approximate).
3	RETRACE	Vertical retrace active.

5.3.2 Sequencer

The sequencer governs how CPU writes are routed to the four bit planes. Use the index/data pair to read or write one of five sequencer registers; bit-plane mask has its own direct port.

Address	Name	Purpose
\$F1010	VGA_SEQ_INDEX	Index into the sequencer register file (0-4).
\$F1014	VGA_SEQ_DATA	Read/write the register selected by VGA_SEQ_INDEX.
\$F1018	VGA_SEQ_MAPMASK	Direct port for the map-mask register (plane write enable, bits 0-3).

5.3.3 CRT controller (CRTC)

The CRTC owns the cursor position, the display start address (for hardware scrolling), and the timing parameters. There are 25 CRTC registers; the most useful ones have direct ports.

Address	Name	Purpose
\$F1020	VGA_CRTC_INDEX	Index (0-24).
\$F1024	VGA_CRTC_DATA	Read/write the indexed register.
\$F1028	VGA_CRTC_STARTHI	High byte of display start address (page flip).
\$F102C	VGA_CRTC_STARTLO	Low byte of display start address.

Cursor position is held in CRTC indices \$0E (high) and \$0F (low). The cursor scan lines are indices \$0A (start) and \$0B (end). The LOCATE BASIC statement writes the cursor low and high bytes through VGA_CRTC_INDEX/DATA.

5.3.4 Graphics controller

The graphics controller decides how CPU bytes are combined with the existing VRAM bytes on a planar write.

Address	Name	Purpose
\$F1030	VGA_GC_INDEX	Index (0-8).
\$F1034	VGA_GC_DATA	Read/write the indexed register.
\$F1038	VGA_GC_READMAP	Direct port for the read-map select (plane read selector).
\$F103C	VGA_GC_BITMASK	Direct port for the bit-mask register.

5.3.5 Attribute controller

Holds the 16 colour-attribute palette entries and the text-mode attribute control bits.

Address	Name	Purpose
\$F1040	VGA_ATTR_INDEX	Index (0-20).
\$F1044	VGA_ATTR_DATA	Read/write the indexed register.

5.3.6 DAC and palette

The DAC stores 256 RGB triples, six bits per component. When the chip displays an indexed pixel, it looks up the triple, expands each component from six to eight bits ($(v \ll 2) \mid (v \gg 4)$), and sends the result to the compositor.

Address	Name	Purpose
\$F1050	VGA_DAC_MASK	Pixel mask applied before lookup.
\$F1054	VGA_DAC_RINDEX	Read index.
\$F1058	VGA_DAC_WINDEX	Write index.
\$F105C	VGA_DAC_DATA	DAC data port. Reads/writes one component per access in R, G, B order.
\$F1100-\$F13FF	VGA_PALETTE/VGA_PALETTE_END	Direct window into 256×3 bytes of palette RAM.

The DAC ports are a small state machine. Write the entry number to VGA_DAC_WINDEX, then write R, G, B in succession to VGA_DAC_DATA. The chip increments the write index after the third byte. The PALETTE BASIC keyword uses this sequence.

```

100 POKE32 &H000F1058, 1           : REM WINDEX = 1
110 POKE32 &H000F105C, 63          : REM red
120 POKE32 &H000F105C, 0           : REM green
130 POKE32 &H000F105C, 0           : REM blue

```

The same entry can also be written directly through the palette window:

```

200 POKE8 &H000F1100+1*3,63        : REM red
210 POKE8 &H000F1100+1*3+1,0       : REM green
220 POKE8 &H000F1100+1*3+2,0       : REM blue

```

Each component is a byte in the window and is clamped to six bits on write.

5.4 The VRAM regions

VGA uses two separate regions of main memory:

Address	Size	Used by
\$A0000	64 KB	Graphics modes (12h, 13h, Mode X).
\$B8000	32 KB	Text mode (03h).

The graphics region is divided into four 64 KB planes. The current mode decides how addresses in the window map onto those planes.

5.4.1 Mode 13h linear pixels

Mode 13h is the easiest bitmap mode. It is 320 x 200, one byte per pixel. The visible byte address is:

$$\$000A0000 + y * 320 + x$$

The byte value is a palette index. This BASIC program draws a yellow horizontal line by writing VRAM directly:

```

10 SCREEN &H13
20 PALETTE 4, 63, 63, 0
30 FOR X=40 TO 279
40 POKE8 &H000A0000 + 100*320 + X, 4
50 NEXT X
60 VSYNC

```

Internally Mode 13h uses chain-4 addressing: address bits 0-1 choose the plane, and address bits 2 and above choose the byte inside that plane. You normally do not need to think about that, because consecutive POKE8 addresses behave like a flat bitmap.

5.4.2 Mode X unchained pixels

Mode X is 320 x 240, one byte per pixel, but it is unchained. Pixel x, y lives in plane $x \text{ AND } 3$, at byte offset:

$$y * 80 + \text{INT}(x / 4)$$

Select the plane with VGA_SEQ_MAPMASK before writing the byte. This program draws a red horizontal line in Mode X:

```

10 SCREEN &H14
20 PALETTE 1, 63, 0, 0
30 FOR X=0 TO 319
40 POKE32 &H000F1018, 2^(X AND 3)
50 POKE8 &H000A0000 + 120*80 + INT(X/4), 1
60 NEXT X
70 VSYNC

```

5.4.3 Mode 12h bit planes

Mode 12h is 640 x 480 with four one-bit planes, giving colour indices 0-15. Each byte covers eight pixels. For pixel x, y , the byte offset is:

$$y * 80 + \text{INT}(x / 8)$$

The bit inside that byte is $7 - (x \text{ AND } 7)$. Plane 0 supplies colour bit 0, plane 1 supplies colour bit 1, and so on.

The key registers for planar writes are:

Register	Use
VGA_SEQ_MAPMASK	Bit mask of planes to write.
VGA_GC_READMAP	Plane selected by PEEK8 in planar modes.
VGA_GC_BITMASK	Bit mask of pixels affected inside the byte.
VGA_GC_MODE	Low bits select write mode 0-3; bit 3 selects read mode.

The write modes are:

Mode	Effect
0	Rotate the CPU byte, optionally use set/reset, combine with latches, then apply VGA_GC_BITMASK.
1	Write the current latch contents. Read before writing if you want useful latch data.
2	Expand the low four CPU bits across the four planes.
3	Use set/reset through the rotated CPU byte and VGA_GC_BITMASK.

Mode 12h is best for code that wants classic four-plane VGA effects. For ordinary BASIC drawing, Mode 13h is simpler.

This example writes one Mode 12h pixel directly. It sets colour index 5, which means plane 0 and plane 2 are set for the selected bit:

```

10 SCREEN &H12
20 X=100:Y=100
30 A=&H000A0000+Y*80+INT(X/8)
40 B=128/2^(X AND 7)
50 POKE32 &H000F103C,B
60 POKE32 &H000F1018,1
70 POKE8 A,255
80 POKE32 &H000F1018,4
90 POKE8 A,255
100 VSYNC

```

Line 50 selects the bit within the byte. Lines 60-90 write the same bit into plane 0 and plane 2, giving colour index 5.

5.4.4 Text buffer

In text mode the buffer holds pairs of bytes: character code, then attribute. The attribute's low nibble is the foreground colour (0-15); the next three bits are the background colour (0-7); the top bit, if set, makes the character blink (depending on the attribute-mode setting).

A bare `PRINT "HELLO"` writes characters into this buffer at the current cursor position using the attribute set by the most recent `COLOR` statement.

5.5 Hardware scrolling and page flipping

The pair of registers `VGA_CRTC_STARTHI` and `VGA_CRTC_STARTLO` holds the byte offset within VRAM at which the displayed picture begins. By changing this offset between frames you can:

- **Page flip.** Render to one half of VRAM while displaying the other, then swap by writing the new start address. The change takes effect on the next frame.
- **Scroll.** Shift the start address by one row's worth of bytes to scroll vertically; by one byte (Mode 13h) or one character (text mode) to scroll horizontally. The `SCROLL BASIC` keyword uses this in text mode.

The CRTC also exposes a line-compare register (index \$18) that splits the screen into two scrolling regions; this is the classic "status bar at the bottom" trick.

This BASIC program demonstrates text-mode scrolling through the `SCROLL` keyword:

```
10 SCREEN &H03
20 COLOR 15, 1
30 FOR I=0 TO 30
40 PRINT "SCROLL LINE "; I
50 NEXT I
60 VSYNC
70 SCROLL 0, 1
```

For raw register control, write `VGA_CRTC_STARTHI` and `VGA_CRTC_STARTLO` directly. A start address of 80 begins the display one text row later:

```
100 POKE32 &H000F1028, 0
110 POKE32 &H000F102C, 80
```

5.6 The vertical retrace

`VGA_STATUS` bit 0 is set whenever the chip is in the vertical retrace interval. The cleanest way to update VRAM without tearing is to poll this bit in a tight loop until it becomes set, then do your writes. The `VSYNC BASIC` keyword does the polling for you, with a one-million-iteration timeout so that a misconfigured chip cannot lock up the program:

```
10 VSYNC
20 REM update VRAM here
```

5.7 Putting it together

The shortest road to a picture on VGA is:

1. Pick a mode and write it to `VGA_MODE` (or use `SCREEN`).
2. Set `VGA_CTRL = 1` to enable the chip (`SCREEN` does this for you).
3. Load any palette entries you need through the DAC ports or the palette window (`PALETTE`).
4. Write pixels or characters into the appropriate VRAM window.
5. To animate, poll `VGA_STATUS` for the retrace or call `VSYNC` before each frame.

The next chapter covers the TED video chip, which produces the C16/Plus-4 picture model and lives on layer 12.