

# Chapter 7 - ANTIC and GTIA

---

ANTIC and GTIA together produce the Atari 8-bit picture model. They are two chips that share the same compositor layer (13) and the same picture; ANTIC fetches the playfield and runs the **display list**, while GTIA chooses colours and handles **player/missile graphics** (the chip's name for hardware sprites). Both chips live in adjacent register regions and are normally programmed together. They are still Intuition Engine bus devices: the display list, screen bytes, character set, bitmap data, and player/missile state are all written from the same BASIC prompt and the same main memory used by the other video cards.

## 7.1 What ANTIC and GTIA can show

---

Item	Value
Display area	320 × 192 pixels
Border	32 left, 32 right, 24 top, 24 bottom
Total frame	384 × 240 pixels
Colour registers	256-entry palette, register byte = HHHHLLLL
Picture modes	14 (text and bitmap, set per display-list entry)
Sprites	4 8-pixel players + 4 2-pixel missiles
Interrupts	Display-List Interrupt (DLI), Vertical-Blank Interrupt (VBI)

The chip's distinguishing feature is the **display list**: a short program in main memory that tells ANTIC, line by line, what kind of playfield to fetch and what to do at the end of each region. The list is what makes the screen of a typical Atari 8-bit game switch mode and scrolling speed every few scanlines.

## 7.2 BASIC keywords

---

The ANTIC and GTIA keywords introduce subcommands for each chip.

Form	Effect
ANTIC ON / ANTIC OFF	Enable / disable the chip.
ANTIC DLIST $addr$	Set the display-list pointer (writes the high and low byte registers).
ANTIC MODE $value$	Write $value$ to DMACTL.
ANTIC SCROLL $hscrol$ , $vscrol$	Write the horizontal and vertical fine-scroll registers (each 0-15).
ANTIC CHBASE $hi$	Write the character-set base (high byte).
ANTIC PMBASE $hi$	Write the player-missile base (high byte).
ANTIC NMI $mask$	Write $mask$ to NMIEN (bit 6 = VBI, bit 7 = DLI).
GTIA COLOR $reg$ , $value$	Write $value$ to colour register $reg$ (0-4 = COLPF0-3, COLBK; 5-8 = COLPM0-3).
GTIA PRIOR $value$	Write $value$ to PRIOR.

Form	Effect
GTIA PLAYER $n$ , $x$ [ , $size$ ]	Move player $n$ to horizontal position $x$ ; optionally set its size.
GTIA MISSILE $n$ , $x$	Move missile $n$ to horizontal position $x$ .
GTIA GRAFP $n$ , $bits$	Write 8-pixel pattern $bits$ to player $n$ .
GTIA GRAFM $bits$	Write the 4-missile pattern byte.
GTIA GRAC $T$ L $value$	Write the graphics-control byte.

A complete first picture needs three things: a display list, some screen bytes for that list to fetch, and GTIA colours. Type this to make a 40-column checkerboard from one blank character and one solid-block character:

```

10 DL=&H0200:SCR=&H0300:CH=&H0800
20 FOR R=0 TO 7
30 POKE8 CH+8+R,&HFF           : REM character 1 is a solid block
40 NEXT R
50 POKE8 DL+0,&H42             : REM mode 2 + LMS
60 POKE8 DL+1,SCR AND 255
70 POKE8 DL+2,INT(SCR/256)
80 FOR I=0 TO 22
90 POKE8 DL+3+I,2             : REM 23 more mode-2 rows
100 NEXT I
110 POKE8 DL+26,&H41           : REM JVB
120 POKE8 DL+27,DL AND 255
130 POKE8 DL+28,INT(DL/256)
140 FOR Y=0 TO 23
150 FOR X=0 TO 39
160 POKE8 SCR+Y*40+X,(X+Y) AND 1
170 NEXT X
180 NEXT Y
190 ANTIC CHBASE INT(CH/256)
200 GTIA COLOR 0,&H04
210 GTIA COLOR 1,&H9A
220 GTIA COLOR 4,&H00
230 ANTIC DLIST DL
240 ANTIC MODE &H22           : REM display-list DMA + normal width
250 ANTIC ON

```

The result is a blue-on-black ANTIC text screen. Nothing in that listing depends on any outside tool: the display list, the character set, and the screen matrix are all typed as bytes. The addresses are below \$1000 so the 16-bit ANTIC fetch address can reach them without overwriting BASIC's own work area.

## 7.3 The ANTIC register block

ANTIC's registers live at \$F2100-\$F213F. Every register is a 32-bit word at a 4-byte-aligned address; only the low byte of each is meaningful.

Address	Name	Purpose
\$F2100	ANTIC_DMAC $T$ L	DMA control.
\$F2104	ANTIC_CHACT $L$	Character control.
\$F2108	ANTIC_DLIS $T$ L	Display-list pointer, low byte.

Address	Name	Purpose
\$F210C	ANTIC_DLISTH	Display-list pointer, high byte.
\$F2110	ANTIC_HSCROL	Horizontal fine-scroll, 0-15 pixels.
\$F2114	ANTIC_VSCROL	Vertical fine-scroll, 0-15 lines.
\$F2118	ANTIC_PMBASE	Player-missile base address (high byte).
\$F211C	ANTIC_CHBASE	Character-set base address (high byte).
\$F2120	ANTIC_WSYNC	Write to halt the CPU until horizontal sync.
\$F2124	ANTIC_VCOUNT	Current scanline / 2 (read-only).
\$F2128	ANTIC_PENH	Light-pen X (read-only).
\$F212C	ANTIC_PENV	Light-pen Y (read-only).
\$F2130	ANTIC_NMIEN	NMI enable (bit 6 = VBI, bit 7 = DLI).
\$F2134	ANTIC_NMIST	NMI status (read); write to clear.
\$F2138	ANTIC_ENABLE	Bit 0 = video enable, bit 1 = PAL timing.
\$F213C	ANTIC_STATUS	Bit 0 = VBlank active (read-only).

### 7.3.1 DMACTL bits

Bit	Name	Meaning
0-1	Width	00 off, 01 narrow (128 colour clocks), 10 normal (160), 11 wide (192).
2	MISSILE	Enable missile DMA.
3	PLAYER	Enable player DMA.
4	PMRES	0 = double-line P/M, 1 = single-line.
5	DL	Enable display-list DMA. (Required to render anything.)

### 7.3.2 CHACTL bits

Bit	Name	Meaning
0	BLANK	Force blank in place of inverse video.
1	INVERT	Invert character cells (swap foreground/background).
2	REFLECT	Mirror character rows vertically.

## 7.4 The display list

ANTIC fetches the playfield by walking a small program in main memory. The program is stored as bytes; each byte is an **instruction**. ANTIC reads the next instruction after every region the previous one has finished.

### 7.4.1 Instruction encoding

Every instruction is one byte:

```

bit 7 6 5 4 3 2 1 0
   D L V H M M M M
   | modifiers |---mode---|

```

The low nibble (mode) is 0-15. The high nibble carries **modifiers** that change how the mode is fetched:

Bit	Modifier	Meaning
4	HSCROL	Enable horizontal fine-scroll for this region.
5	VSCROL	Enable vertical fine-scroll for this region.
6	LMS	The next two bytes hold a 16-bit fetch address; ANTIC reloads its screen pointer from them.
7	DLI	Fire a display-list interrupt at the end of this region.

Two special instructions live in the encoding:

Byte	Name	Effect
\$01	JMP	The next two bytes are an address; jump there.
\$41	JVB	Like JMP, but also wait for the next vertical blank.

## 7.4.2 The mode list

Modes 2-7 are character modes; modes 8-15 are bitmap modes; mode 0 is one or more blank scanlines.

Mode	Kind	Geometry per row
0	Blank	(opcode >> 4) + 1 blank scanlines (1-8).
2	Text	40 columns, 8 scanlines per row (the standard mode).
3	Text	40 columns, 10 scanlines per row.
4	Text	40 columns, 8 scanlines, multicolour.
5	Text	40 columns, 16 scanlines, multicolour.
6	Text	20 columns, 8 scanlines, 16-pixel-wide characters.
7	Text	20 columns, 16 scanlines.
8	Bitmap	40 pixels wide, 8 scanlines per row.
9	Bitmap	80 pixels wide, 4 scanlines.
10	Bitmap	80 pixels wide, 2 scanlines.
11	Bitmap	160 pixels wide, 1 scanline.
12	Bitmap	160 pixels wide, 1 scanline (alternate colour set).
13	Bitmap	160 pixels wide, 2 scanlines.
14	Bitmap	160 pixels wide, 1 scanline, 4 colours.
15	Bitmap	320 pixels wide, 1 scanline, 2 colours (GTIA modes).

### 7.4.3 A short example

This eight-byte display list shows one mode-2 region of text at the top of the screen, fires a DLI at its end, then jumps back to its own start and waits for the next frame:

addr	bytes	meaning
\$5000	\$42	mode 2, LMS modifier
\$5001	\$00 \$60	screen RAM starts at \$6000
\$5003	\$82	mode 2 with DLI at the end of the region
\$5004	\$02	another mode-2 region
\$5005	...	(more entries here)
\$50FE	\$41	JVB
\$50FF	\$00 \$50	jump back to \$5000 at next VBlank

Set the display-list pointer with ANTIC\_DLSTL/DLISTH and enable display-list DMA with bit 5 of DMACTL.

Display-list fetch addresses are 16-bit. Keep display lists and their screen data below \$10000 unless the CPU chapter you are using documents a wider ANTIC data path. LMS supplies a new fetch address; entries without LMS continue from the next byte after the previous region's data.

For text modes, one display-list entry consumes one row of screen codes. For bitmap modes, each scanline consumes bytes:

Modes	Bytes per scanline	Pixels per byte
8 and 15	40	8 one-bit pixels
9 and 10	20	4 two-bit pixels
11, 12, 13, and 14	40	4 two-bit pixels

### 7.4.4 A four-colour bitmap example

Mode 14 is a useful four-colour bitmap mode: each byte contains four two-bit pixels, read from left to right as bit pairs 7-6, 5-4, 3-2, and 1-0. The pair value selects COLPF0-COLPF3.

This example draws a safe top-of-screen ribbon with repeating four-colour vertical bars. It uses only the first sixteen scanlines, so the bitmap bytes still fit in the low-memory area that ANTIC can address from BASIC.

```

10 DL=&H0200:SCR=&H0300
20 POKE8 DL+0,&H4E           : REM mode 14 + LMS
30 POKE8 DL+1,SCR AND 255
40 POKE8 DL+2,INT(SCR/256)
50 FOR I=0 TO 14
60 POKE8 DL+3+I,&H0E
70 NEXT I
80 POKE8 DL+18,&H41
90 POKE8 DL+19,DL AND 255
100 POKE8 DL+20,INT(DL/256)
110 FOR Y=0 TO 15
120 FOR X=0 TO 39
130 A=SCR+Y*40+X
140 IF (X AND 1)=0 THEN POKE8 A,&H1B ELSE POKE8 A,&HE4
150 NEXT X
160 NEXT Y
170 GTIA COLOR 0,&H24
180 GTIA COLOR 1,&H46
190 GTIA COLOR 2,&H9A
200 GTIA COLOR 3,&HCE
210 GTIA COLOR 4,&H00
220 ANTIC DLIST DL
230 ANTIC MODE &H22
240 ANTIC ON

```

The top of the playfield becomes broad vertical bars made from the four playfield colours. \$1B is bit pairs 00,01,10,11; \$E4 reverses the order.

## 7.5 Interrupts

ANTIC raises two kinds of interrupt:

- **VBI** (Vertical-Blank Interrupt). Fires once per frame at the start of vertical retrace. Enable with bit 6 of NMIEN; status bit 6 of NMIST. Used for routine per-frame work such as timers, input scanning, and colour rotation.
- **DLI** (Display-List Interrupt). Fires at the end of any display-list region whose DLI modifier bit is set. Enable with bit 7 of NMIEN; status bit 7 of NMIST. Used for raster effects: change colours, scroll position, or character base mid-screen.

The CPU acknowledges either interrupt by writing any value to NMIST (also known as NMIRES), which clears both pending bits.

## 7.6 Wait for horizontal sync

Writing any value to ANTIC\_WSYNC halts the CPU until the next horizontal-retrace edge. This is the simplest way to time a register change to the start of the next scanline without polling.

On Intuition Engine, WSYNC also records the current GTIA background, player, and missile state for the scanline being advanced. That makes it useful from BASIC for raster-bar experiments:

```

10 ANTIC ON
20 FOR Y=0 TO 191
30 IF (Y AND 16)=0 THEN GTIA COLOR 4,&H36 ELSE GTIA COLOR 4,&H74
40 POKE32 &H00F2120,0 : REM WSYNC
50 NEXT Y

```

The next completed ANTIC frame uses alternating background colour bands. Use a display-list picture behind it if you want playfield graphics as well as bars.

## 7.7 The GTIA register block

GTIA's registers live immediately after ANTIC's, at \$F2140 through \$F21FB. Every register is again a 32-bit word at a 4-byte boundary, with only the low byte meaningful.

### 7.7.1 Colour registers

The playfield uses five colour registers; the four players and four missiles use four more. Players and missiles 0-3 share their colour register (one register per pair).

Address	Name	Used for
\$F2140	GTIA_COLPF0	Playfield colour 0.
\$F2144	GTIA_COLPF1	Playfield colour 1.
\$F2148	GTIA_COLPF2	Playfield colour 2.
\$F214C	GTIA_COLPF3	Playfield colour 3.
\$F2150	GTIA_COLBK	Background and border.
\$F2154	GTIA_COLPM0	Player/missile 0.
\$F2158	GTIA_COLPM1	Player/missile 1.
\$F215C	GTIA_COLPM2	Player/missile 2.
\$F2160	GTIA_COLPM3	Player/missile 3.

Each register holds an 8-bit colour byte in the form HHHLLLLL, where H selects one of 16 hues and L selects one of 16 luminance steps. The high-nibble hue names are:

High nibble	Hue
\$0	Grey
\$1	Gold
\$2	Orange
\$3	Red-orange
\$4	Pink
\$5	Purple
\$6	Purple-blue
\$7	Blue
\$8	Blue 2

High nibble	Hue
\$9	Light blue
\$A	Turquoise
\$B	Green-blue
\$C	Green
\$D	Yellow-green
\$E	Orange-green
\$F	Light orange

## 7.7.2 Control registers

Address	Name	Purpose
\$F2164	GTIA_PRIOR	Priority and GTIA-mode bits.
\$F2168	GTIA_GRCTL	Graphics control.
\$F216C	GTIA_CONSOL	Console switches (read).

PRIOR bits:

Bit	Name	Meaning
0-3	Mix	Player/playfield priority pattern (Atari standard).
4	MULTI	Enable multicolour players.
5	FIFTH	Treat missiles as a single fifth player.
6-7	GTIA mode	Select GTIA-only high-colour modes.

GRCTL bits:

Bit	Name	Meaning
0	MISSILE	Enable missile graphics.
1	PLAYER	Enable player graphics.
2	LATCH	Latch trigger inputs.

## 7.7.3 GTIA colour remapping bits

Bits 6 and 7 of PRIOR change how GTIA chooses colours for bitmap playfields. These modes are most useful with bitmap display list entries, because the source bytes then become colour data as well as pixel data.

PRIOR bits	Value	Colour rule
00	\$00	Normal ANTIC colour selection. One-bit pixels use COLPF0 and COLPF1; two-bit pixels use COLPF0-COLPF3.
01	\$40	GTIA luminance mode. The hue comes from COLPF1; the luminance comes from the low nibble of the source byte.
10	\$80	GTIA playfield-index mode. Set pixels choose one of COLPF0-COLPF3 from their bit or pair position.
11	\$C0	GTIA hue mode. The hue comes from the high nibble of the source byte; the luminance comes from COLBK.

This example uses mode 8 and PRIOR value \$40 to make vertical luminance bars. Each byte is an eight-pixel-wide bar; values 1-15 use the red hue from COLPF1 with different luminances, while value 0 remains the background colour.

```

10 DL=&H0200:SCR=&H0300
20 POKE8 DL+0,&H48           : REM mode 8 + LMS
30 POKE8 DL+1,SCR AND 255
40 POKE8 DL+2,INT(SCR/256)
50 POKE8 DL+3,&H41
60 POKE8 DL+4,DL AND 255
70 POKE8 DL+5,INT(DL/256)
80 FOR Y=0 TO 7
90 FOR X=0 TO 39
100 POKE8 SCR+Y*40+X,X AND 15
110 NEXT X
120 NEXT Y
130 GTIA COLOR 0,&H00
140 GTIA COLOR 1,&HA0
150 GTIA COLOR 4,&H02
160 GTIA PRIOR &H40
170 ANTIC DLIST DL
180 ANTIC MODE &H22
190 ANTIC ON

```

You should see red bars across the top of the playfield, stepping through the available luminance levels.

## 7.8 Players and missiles

Each of the four players is 8 pixels wide; each of the four missiles is 2 pixels wide. They can be drawn either from DMA buffers (under ANTIC's control, with DMA\_PLAYER/DMA\_MISSILE bits set) or by writing directly into the GTIA graphics registers.

Address	Name	Purpose
\$F2170-\$F217C	GTIA_HPOSP0-HPOSP3	Player horizontal position.
\$F2180-\$F218C	GTIA_HPOSM0-HPOSM3	Missile horizontal position.
\$F2190-\$F219C	GTIA_SIZEP0-SIZEP3	Player size. 0 = normal, 1 = double, 3 = quadruple.
\$F21A0	GTIA_SIZEM	All four missile sizes (2 bits each).
\$F21A4-\$F21B0	GTIA_GRAFP0-GRAFP3	Player graphics (8 pixels each).
\$F21B4	GTIA_GRAFM	Missile graphics (2 bits per missile).

### 7.8.1 Collisions

GTIA reports collisions in a 16-register read-only block at \$F21B8-\$F21F4. Each register's low nibble is a bit mask of the four object groups (COLPF0-3 for player-vs-playfield, players 0-3 for player-vs-player). The collisions latch from frame to frame until cleared by a write to GTIA\_HITCLR (\$F21F8).

This program draws a blue checkerboard playfield, then records a red player column and a green missile column across the visible frame:

```

10 DL=&H0200:SCR=&H0300:CH=&H0800
20 FOR R=0 TO 7
30 POKE8 CH+8+R,&HFF
40 NEXT R
50 POKE8 DL+0,&H42
60 POKE8 DL+1,SCR AND 255
70 POKE8 DL+2,INT(SCR/256)
80 FOR I=0 TO 22
90 POKE8 DL+3+I,2
100 NEXT I
110 POKE8 DL+26,&H41
120 POKE8 DL+27,DL AND 255
130 POKE8 DL+28,INT(DL/256)
140 FOR Y=0 TO 23
150 FOR X=0 TO 39
160 POKE8 SCR+Y*40+X,(X+Y) AND 1
170 NEXT X
180 NEXT Y
190 ANTIC CHBASE INT(CH/256)
200 GTIA COLOR 0,&H04
210 GTIA COLOR 1,&H9A
220 GTIA COLOR 4,&H00
230 GTIA COLOR 5,&H46           : REM player 0
240 GTIA COLOR 7,&HCE           : REM missile 2
250 GTIA GRACTL 3
260 GTIA PLAYER 0,110,1
270 GTIA MISSILE 2,190
280 FOR Y=0 TO 191
290 GTIA GRAFP 0,&H3C
300 GTIA GRAFM 4
310 POKE32 &H000F2120,0       : REM capture one scanline
320 NEXT Y
330 ANTIC DLIST DL
340 ANTIC MODE &H2E           : REM playfield + player + missile DMA
350 ANTIC ON

```

GRAFP 0,&H3C gives player 0 a centred six-pixel shape. GRAFM 4 enables missile 2. WSYNC advances the scanline capture, so the same player and missile shape is recorded down the frame.

## 7.9 Setup order, side effects, and limits

Use this order from a clean state:

1. Build the display list in low memory.
2. Build any character set, screen matrix, bitmap data, or player/missile scanline data it needs.
3. Set ANTIC\_CHBASE and ANTIC\_PMBASE if those features are used.
4. Set GTIA playfield, background, and player/missile colours.
5. Write the display-list pointer with ANTIC DLIST.
6. Write DMACTL with ANTIC MODE; bit 5 must be set for display-list DMA.
7. Enable the source with ANTIC ON.

Important side effects and limits:

- Only the low byte of each ANTIC and GTIA register is meaningful.
- ANTIC\_HSCROL and ANTIC\_VSCROL are masked to 0-15.

- ANTIC\_WSYNC is write-only; reads return 0.
- ANTIC\_STATUS reports VBlank and does not clear NMIST.
- ANTIC\_NMIST clears when any value is written to it.
- ANTIC\_VCOUNT returns the current scanline divided by two.
- ANTIC\_ENABLE bit 0 enables video; bit 1 selects PAL timing.
- Display-list execution stops after JVB, after the visible playfield is full, or after 1024 display-list entries.
- GTIA collision registers are read-only and remain latched until GTIA\_HITCLR is written.
- Player horizontal positions are Atari-style positions. A value of about 48 begins at the left edge of the visible playfield.

The next chapter covers the ULA, the Sinclair-style picture chip on layer 15.