GTIA2RGB

Hardware Reference Manual

GTIA2RGB Project

May 2026 — Draft

Contents

1 Overview

GTIA2RGB is a Gowin FPGA-based companion device for the GTIA chip in Atari XL and XE computers. The original GTIA remains in place; GTIA2RGB observes its inputs and outputs and produces a digital RGB signal (and optional 31 kHz VGA-rate scandoubled signal) in parallel, with a number of selectable quality improvements over the original analog video path.

Compatibility. Supported across the entire XL/XE family in both PAL and NTSC variants. Not every model/region combination has been individually tested, but the design has no model- or region-specific dependencies within that range. The 400/800 line is outside the scope of this manual.

SECAM machines use a different chip (FGTIA) with a different pinout and are *not* supported by this hardware; a SECAM-compatible variant would require a separate physical board.

1.1 Feature Summary

- Digital RGB output, with optional scandoubler to 31 kHz (VGA-compatible)
- Separate H/V sync or composite sync, selectable via DIP switch
- Selectable 15 kHz / 31 kHz output via DIP switch
- PAL and NTSC palettes derived from real-hardware measurements, switchable at runtime via OSD
- Optional inter-line colour blending (PAL-like; required for correct rendering of APAC, HIP, RIP, CIN, etc.)
- **ANTIC FIX**: corrects the broken sync ANTIC emits past scanline 240, allowing full 240-line output to be displayed by modern TVs and monitors
- Optional right-side hgate masking (**HGATE**) to hide the garbage data ANTIC fetches in WIDE mode
- Optional support for the GTIA luma-bit-0 (**LUMA0**) outside of GTIA '01' / BASIC 9
- Optional independent chroma for set/cleared pixels in hi-res modes (**HRBICOLOR**), instead of GTIA's COLPF2-only chroma
- On-screen menu (**OSD**) for switching options live
- Built-in buzzer reproducing GTIA-generated audio (keyboard click etc.)
- **COL80**: optional overlay that reinterprets ANTIC mode F scan lines as 80-column (or 64/88-column) character data, with per-character colour attributes and runtime-uploadable font

With its default settings GTIA2RGB simply adds a clean RGB output to an otherwise stock machine. ANTIC FIX is always active and has no toggle. The other features (HGATE, LUMA0, HRBICOLOR, blending, palette, buzzer) are configured through the OSD; the 80-column overlay is controlled through the register interface described in Section 5.

2 Device Identification

GTIA2RGB exposes three identification/version registers that are readable from the Atari at any time, irrespective of mode.

Address	On read	Notes
`$D01E`	`$0A` (constant)	Identification byte
`$D01C`	Firmware major version (decimal digit)	e.g. 1 for FW 1.0
`$D01D`	Firmware minor version (decimal digit)	e.g. 0 for FW 1.0

A standard GTIA returns `$0F` on a read from `$D01E`; GTIA2RGB returns `$0A` instead. This forms the basis of the recommended detection sequence:

```
lda $D01E
cmp #$0A
bne NOT_GTIA2RGB      ; $0F = stock GTIA, anything else = unknown
; GTIA2RGB detected
lda $D01C             ; -> major version
lda $D01D             ; -> minor version
```

Write side. These addresses retain their original GTIA write semantics in all modes *except* as noted in Section 5: writes to `$D01D` configure the 80-column overlay, and writes to `$D01E` upload font data when the overlay is enabled.

3 Hardware Configuration

3.1 DIP Switches

GTIA2RGB has two DIP switches on the board:

Switch	Position	Function
Near the GTIA socket	ON	15 kHz output (TV / SCART)
	OFF	31 kHz output (VGA, scandoubled)
Near the PCB edge	ON	Composite sync (CSYNC)
	OFF	Separate horizontal + vertical sync

3.2 OSD — On-Screen Display

The OSD allows live adjustment of most non-register-mapped options. The settings are saved when the OSD is exited and persist across power cycles.

3.2.1 Entering and Navigating the OSD

- **Enter:** hold START + SELECT + OPTION for 100 video frames (≈ 2.0 s PAL, ≈ 1.7 s NTSC).
- **OPTION:** step through options.
- **SELECT:** change the value of the current option.
- **START:** save and exit.
- **Auto-exit:** 500 frames after the last keypress (≈ 10 s PAL, ≈ 8.3 s NTSC) the OSD saves and closes automatically.

One frame = one ANTIC vertical period (1/50 s on PAL, 1/60 s on NTSC).

3.2.2 OSD Options

Option	Value	Meaning
PALETTE	PAL	Measured-PAL Atari palette
	NTSC	Measured-NTSC Atari palette
BLENDING	NONE	Each scan line rendered independently
	AUTO	Blend only between GTIA '01' / '11' lines and other modes (HIP/RIP/CIN/APAC stay correct, the rest stays sharp)
	FULL	Blend every line with the previous one (classic PAL television behaviour)
HGATE	OFF	Show the full ANTIC raster, including any right-side garbage in WIDE mode
	ON	Mask the right-side garbage
	USER	State follows bit 3 of `$D01D`
LUMA0	OFF	Force luma-bit-0 to zero (stock GTIA behaviour)
	ON	Use luma-bit-0 normally outside GTIA '01'
	USER	State follows bit 5 of `$D01D`
HRBICOLOR	OFF	Hi-res chroma taken from COLPF2 only (stock GTIA)
	ON	Set pixels can use COLPF1 chroma as well
	USER	State follows bit 4 of `$D01D`
BUZZER	OFF	Internal buzzer silent
	ON	Buzzer reproduces GTIA-generated sounds (chiefly the keyboard click)

Note on USER bits. Bits 3, 4 and 5 of `$D01D` are shared between the OSD 'USER' settings and the 80-column mode control (Section 5.2). When 80-column mode is enabled, these bits select the attribute interpretation and the chroma override and should not also be relied on to drive HGATE/HRBICOLOR/LUMA0.

4 Display-Quality Features

4.1 Scandoubler / 31 kHz Output

With the DIP switch in the 31 kHz position, GTIA2RGB doubles each 15 kHz NTSC/PAL line and emits a 31 kHz VGA-compatible signal. In this mode separate H and V sync should normally be used (second DIP switch in the off position). The 15 kHz path remains the natural choice for CRTs and SCART monitors.

4.2 ANTIC FIX

When ANTIC's display list runs past the normal active region (around scanline 240) the chip emits malformed sync. Most TVs and modern monitors react badly, often losing picture entirely. GTIA2RGB detects the condition and rewrites the sync signals on the fly, making it practical to use the full 240 active lines.

This feature is unconditionally active; there is no OSD or register control for it.

4.3 PAL / NTSC Palettes

The two built-in palettes are not synthesised — they were measured from several actual 600XL and 800XL machines (PAL and NTSC respectively) and quantised into the lookup table. The PALETTE OSD option selects which one is active.

4.4 Blending

PAL televisions blur chroma between adjacent lines; many NTSC sets do the same to a smaller degree. Several Atari "high-colour" tricks (APAC, HIP, RIP, CIN) depend on this blur to produce their extra colours. GTIA2RGB offers three modes; see the BLENDING entry in the OSD table above. Use AUTO when in doubt.

4.5 HGATE

In ANTIC WIDE mode the chip fetches extra bytes at the right edge of the screen, which beyond the genuine 384-pixel raster contain undefined data-bus contents. HGATE ON masks that strip; USER ties it to bit 3 of `$D01D`.

HGATE does not apply inside the 80-column overlay — the COL80 path discards the WIDE-mode tail bytes unconditionally, regardless of the HGATE setting.

4.6 LUMA0

Stock GTIA forces the least-significant luma bit to zero outside of GTIA mode '01' (BASIC 9), effectively halving the luma resolution in the other modes. LUMA0 ON enables the bit everywhere. USER ties it to bit 5 of `$D01D`.

4.7 HRBICOLOR

In stock GTIA hi-res modes the chroma comes from COLPF2 for both set and cleared pixels; COLPF1 only contributes luma. HRBICOLOR ON routes the COLPF1 chroma to set pixels as well, allowing genuinely two-coloured hi-res output. USER ties it to bit 4 of `$D01D`.

4.8 Buzzer

A small on-board buzzer reproduces the audio normally generated inside the original GTIA (mostly the keyboard click). It is muted entirely in the OFF setting.

5 80-Column Mode (COL80)

The 80-column overlay reinterprets ANTIC mode F scan lines as character data with per-character colour attributes. When the overlay is disabled (bit 6 of `$D01D` cleared), GTIA2RGB behaves as a normal GTIA with all the quality features above.

5.1 Operating Principle

ANTIC generates mode F scan lines (40 bytes each in NORMAL playfield width, 32 in NARROW, 44 visible in WIDE — see Section 5.5). GTIA2RGB captures these bytes and interprets them as 80-column character data rather than bitmap pixels. Each text row uses several mode F lines: two for character codes (left/right halves of the row), and additional lines for colour attributes depending on the selected mode. Blank-line instructions (`$10`–`$70`) provide the remaining scan lines needed to fill an 8-pixel-tall character cell.

GTIA2RGB latches one complete row of character and attribute data during the first 8 scan lines, then renders it during the following 8 scan lines while simultaneously latching the next row into a second buffer (double-buffered design). The first mode F instruction (`$4F` with LMS, or `$0F`) in the display list triggers the start of the 80-column display. If a subsequent group of 8 scan lines contains no mode F data, GTIA2RGB treats that row as inactive and displays the background colour.

5.2 Mode Control — `$D01D` (write)

Writes to `$D01D` are latched into the 80-column configuration register (bits 6:3 of the write data go into the 4-bit `col80_cfg`):

Bit	Name	Function
7	—	Reserved
6	COL80	80-column overlay enable (1 = on, 0 = plain GTIA)
5	ATTRHI	Attribute-mode select, high bit
4	CHROMA1	Bit 4: in Mode 1, independent foreground chroma (use COLPF1)
3	ATTRLO	Attribute-mode select, low bit
2–0	—	Reserved / original GTIA function

When COL80 is cleared, bits 3/4/5 may instead drive HGATE / HRBICOLOR / LUMA0 if those OSD options are set to USER; see Section 3.

5.2.1 Mode Selection

With COL80 set, bits 5 and 3 select the per-character attribute interpretation:

Mode	Bit pattern	`$D01D` value	Description
1	`x10x0xxx`	`$40`	Simple — no attributes
2	`x11x0xxx`	`$60`	CGA palette attributes
3	`x10x1xxx`	`$48`	Atari chroma + luma
4	`x11x1xxx`	`$68`	Full RGB

5.2.2 Independent Foreground Chroma (Bit 4)

Bit 4 affects Mode 1 only. Cleared (default), the foreground and the background share their chroma (from COLPF2), so characters and background differ only in luma. Set, the foreground takes its chroma from COLPF1 (shadow 709/`$02C5`); the background still comes from COLPF2 (shadow 710/`$02C6`). Useful for the classic green-on-black-style monochrome display without needing per-character attributes (`$D01D` = `$50`).

5.3 Font Upload — $D010, $D011, $D01E

The 2 KB font memory is 256 characters × 8 bytes. Each byte is one pixel row of one character (MSB = leftmost pixel; a set bit selects foreground, cleared selects background). The Atari accesses font memory through three I/O addresses:

Address	Direction	Function
$D010	Write	Sets the upper 8 bits of the 11-bit address; the lower 3 bits are forced to zero (effectively: address = character code × 8)
$D011	Write	Sets the lower 3 bits of the 11-bit address (i.e. the row within the current character)
$D01E	Write	Stores a byte at the current address and auto-increments the pointer

Font writes only take effect while COL80 is enabled. The data-write path is gated by bit 6 of $D01D; uploads attempted with COL80 cleared are dropped. This is also why $D01E keeps its normal read behaviour (returning the $0A ID byte) regardless of mode: only *writes* to $D01E are repurposed, and only when COL80 is on.

CPU writes only. Addresses $D010 and $D011 are also the targets of ANTIC's player/missile DMA (their original GRAFP3 and GRAFM function), and that DMA path remains live in 80-column mode — sprites still work normally. The COL80 font-address logic distinguishes the two write sources and ignores DMA writes entirely: only CPU stores update the upload pointer. It is therefore safe to upload a font with player/missile DMA active.

5.3.1 Write Protocol

1. Enable COL80 in $D01D (e.g. write $40).
2. Write the starting character code to $D010. This sets the write pointer to *code* × 8 (start of that character, row 0).
3. Optionally write a row index (0–7) to $D011 to patch a single row within the current character.
4. Write data bytes to $D01E. Each write stores the byte and advances the pointer; auto-increment wraps across character boundaries within the 2 KB space.

Uploading all 256 characters sequentially therefore reduces to "write 0 to $D010, write 2048 data bytes to $D01E".

The address for character c, row r is:

$$addr = c \times 8 + r \qquad (c = 0 \ldots 255,\ r = 0 \ldots 7)$$

5.3.2 Initial Font Contents

After power-on the font memory is undefined — there is no built-in default and no reset path other than power cycling, so any program that uses the 80-column overlay must upload its font at start-up. A convenient convention is to upload the Atari ROM character set remapped into ATASCII order; see Appendix A.1.

5.4 Display Modes

All four modes share the same overall structure: ANTIC mode F lines deliver data to GTIA2RGB, two mode F lines per row carry the character codes (left half then right half), and additional mode F lines carry attribute bytes when present. Each row ends with a blank-line instruction to pad the total scan-line count to eight.

The line counts below describe NORMAL playfield width; for NARROW and WIDE adjust the per-line byte counts as described in Section 5.5.

5.4.1 Mode 1 — Simple (No Attributes)

`$D01D` value:	`$40` (or `$50` for independent foreground chroma)
DL per row:	`$0F`, `$0F`, `$50` (+ LMS as needed)
Mode F lines:	2 (80 bytes: characters)
Blank lines:	6 (`$50`)
Bytes per row:	80

Mode 1 is 80-column monochrome text. Each character cell uses a fixed foreground/background colour pair from the Atari colour registers (or the COLPF1/COLPF2 split if bit 4 is set, see Section 5.2).

Mode F line	Bytes	Content
1	0–39	Character codes, columns 0–39
2	40–79	Character codes, columns 40–79

Character codes index the GTIA2RGB font memory directly; there is no hardware-imposed character ordering.

5.4.2 Mode 2 — CGA Palette Attributes

`$D01D` value:	`$60`
DL per row:	`$0F`×4, `$30` (+ LMS as needed)
Mode F lines:	4 (160 bytes: 80 chars + 80 attributes)
Blank lines:	4 (`$30`)
Bytes per row:	160

Mode 2 adds per-character colour attributes drawn from a 16-colour CGA-style palette.

Mode F line	Bytes	Content
1	0–39	Character codes, columns 0–39
2	40–79	Character codes, columns 40–79
3	80–119	Attributes, columns 0–39
4	120–159	Attributes, columns 40–79

Attribute byte: bits 7–4 select the background colour (0–15), bits 3–0 the foreground colour (0–15). Palette:

Index	Colour	Index	Colour
0	Black	8	Dark grey
1	Blue	9	Light blue
2	Green	10	Light green
3	Cyan	11	Light cyan
4	Red	12	Light red
5	Magenta	13	Light magenta
6	Brown	14	Yellow
7	Light grey	15	White

5.4.3 Mode 3 — Atari Chroma + Luma

`$D01D` value: `$48`
DL per row: `$0F`×6, `$10` (+ LMS as needed)
Mode F lines: 6 (240 bytes: 80 chars + 80 chroma + 80 luma)
Blank lines: 2 (`$10`)
Bytes per row: 240

Mode 3 carries native Atari chroma and luma per character, giving access to the full Atari 256-colour palette for both foreground and background of each cell independently.

Mode F line	Bytes	Content
1	0–39	Character codes, columns 0–39
2	40–79	Character codes, columns 40–79
3	80–119	Chroma, columns 0–39
4	120–159	Chroma, columns 40–79
5	160–199	Luma, columns 0–39
6	200–239	Luma, columns 40–79

Chroma byte: bits 7–4 = background hue (0–15), bits 3–0 = foreground hue (0–15). Hue 0 produces grey; 1–15 select the standard Atari hues. **Luma byte:** same nibble layout, but with luma values (0–15).

5.4.4 Mode 4 — Full RGB

`$D01D` value: `$68`
DL per row: `$0F`×8 (+ LMS as needed)
Mode F lines: 8 (320 bytes)
Blank lines: 0
Bytes per row: 320

Mode 4 provides per-character RGB colour control. All 8 scan lines of the row are consumed by data.

Mode F line	Bytes	Content
1	0–39	Character codes, columns 0–39
2	40–79	Character codes, columns 40–79
3	80–119	Red, columns 0–39
4	120–159	Red, columns 40–79
5	160–199	Green, columns 0–39
6	200–239	Green, columns 40–79
7	240–279	Blue, columns 0–39
8	280–319	Blue, columns 40–79

Each colour channel byte packs the same way as the Mode 3 attributes: bits 7–4 select the background intensity, bits 3–0 the foreground intensity (4 bits per channel, $16^3 = 4096$ colours per cell).

5.5 NARROW and WIDE — 64 and 88 Columns

GTIA2RGB honours the ANTIC playfield-width bits (`$D400` bits 1–0, DMACTL) inside the 80-column overlay. NARROW yields 64 columns per row, NORMAL gives the standard 80, and WIDE gives 88 (see the WIDE quirk below).

DMACTL 1–0	Bytes fetched	Bytes to GTIA	Text width
01 (NARROW)	32	32	64 columns
10 (NORMAL)	40	40	80 columns
11 (WIDE)	48	44	88 columns

The character-code mode F lines determine the row width; attribute mode F lines should be sent at the same width. Mixing widths between the two character lines of a single row is technically tolerated (total columns = first-line count + second-line count) but is not recommended.

The WIDE quirk. In WIDE mode ANTIC fetches 48 bytes from RAM per scan line but only forwards 44 of them to GTIA — the other four are dropped. Consequently each WIDE mode F line gives 44 character cells, two lines per row produce 88 columns, and there is a four-byte *hole* in screen memory between the bytes that drive successive scan lines. Without compensation this hole would appear as a four- column gap mid-row.

Recommendation: LMS on every mode F line in WIDE mode. Make each mode F instruction in the display list an LMS variant (`$4F`) pointing at the start of its own 44-byte buffer. This skips the dropped bytes entirely and lets the program lay out screen memory as contiguous 44-byte halves. Without per-line LMS the program must leave a 4-byte gap after every 44 visible bytes in screen memory.

Centring. The 64 and 80-column outputs occupy the same horizontal extent as a classic ANTIC 32-/40-column display. The 88-column output is shifted left, matching the off-centre placement of the underlying ANTIC WIDE raster.

5.6 Sprites in 80-Column Mode

Player/missile graphics remain available with the overlay enabled. Inside 80-column mode the sprites are rendered at half the original horizontal width (so the on-screen size matches the higher-density character grid).

Colour source. Sprite colours always come from the standard Atari sprite colour registers and are interpreted as Atari chroma + luma — regardless of which 80-column attribute mode is currently active. Sprites therefore do not see the CGA or RGB attribute interpretations.

Enabling. Sprites are visible only when `$D01B` (GTIACTL/PRIOR) is programmed to the bit pattern `00xx0001`: bit 0 set (players in front of playfield), bits 1, 2, 3, 6 and 7 cleared. Bits 4 (fifth-player enable) and 5 (multi-colour player) retain their original meaning and can be set freely.

Collisions. Collision detection is performed by the original physical GTIA chip — which is still in the system — and not by GTIA2RGB. The chip has no knowledge of the 80-column overlay or the half-width sprite rendering and continues to compute collisions exactly as it always has: against full-width sprites and against the original mode F bitmap interpretation of the playfield data — not against the displayed characters. The collision registers (M_nPF, P_nPF, M_nPL, P_nPL at `$D000`–`$D00F`) therefore report stock-GTIA values, which do not correspond to what is actually visible on screen in 80-column mode. Pixel-accurate collision against the character grid is not available.

5.7 ANTIC Notes

From ANTIC's point of view the overlay is just graphics mode F. In modes 3 and 4 the screen data for a single frame exceeds 4 KB, so the standard ANTIC 4 KB address-wrap applies in exactly the same way it would for a native 320×192 bitmap. Insert LMS instructions (`$4F`) wherever the address counter needs to be reset, just as in any high-resolution ANTIC graphics mode.

6 Quick Reference

6.1 Identification & Version Registers

`$D01E` (read)	`$0A`	GTIA2RGB ID byte (constant)
`$D01C` (read)	decimal digit	Firmware major version
`$D01D` (read)	decimal digit	Firmware minor version

6.2 80-Column Overlay Registers

`$D01D` (write)	—	80-column control (COL80, ATTRHI, CHROMA1, ATTRLO)
`$D010` (write)	—	Font address: upper 8 bits; lower 3 bits cleared
`$D011` (write)	—	Font address: lower 3 bits (row within character)
`$D01E` (write)	—	Font data + auto-increment (active only with COL80=1)

6.3 Mode Summary

Mode	Colour model	`$D01D`	MF	Blank	B/row	DL pattern (per row)
1	None (fixed)	`$40`	2	6 (`$50`)	80	`$0F`×2, `$50`
2	CGA 16-colour	`$60`	4	4 (`$30`)	160	`$0F`×4, `$30`
3	Atari chroma+luma	`$48`	6	2 (`$10`)	240	`$0F`×6, `$10`
4	RGB 4096-colour	`$68`	8	0	320	`$0F`×8

Byte counts are for NORMAL playfield width (80 columns). Multiply per-row byte counts by 32/40 (= 0.8) for NARROW (64 columns) or 44/40 (= 1.1) for WIDE (88 columns). In WIDE mode use an LMS (`$4F`) on every mode F line, see Section 5.5.

A Program Listings

A.1 Font Upload — ATASCII Order (MADS)

The following routine uploads the full Atari ROM character set (128 normal + 128 inverse = 256 characters) to the GTIA2RGB font memory, remapped to ATASCII order. It must be called *after* enabling 80-column mode (bit 6 of `$D01D`).

`ZPTR` is any available 2-byte zero-page location (e.g. `$CB`/`$CC`).

```
FONT_ADDR_HI = $D010    ; addr[10:3] (writing clears addr[2:0])
FONT_ADDR_LO = $D011    ; addr[2:0]  (row within current character)
FONT_DATA    = $D01E    ; data byte + auto-increment (needs COL80=1)

LOAD_FONT:
        lda #0
        sta FONT_ADDR_HI      ; start at char 0 (addr = 0)
        sta ZPTR              ; source page low byte = 0
        ldx #0
LF_BLOCK:
        lda LF_SRC_HI,x
        sta ZPTR+1            ; source page high byte
        lda LF_EOR,x
        sta LF_EOR_OP+1       ; self-modify: $00 or $FF
        ldy #0
LF_BYTE:
        lda (ZPTR),y
LF_EOR_OP:
        eor #0                ; $00 = normal, $FF = inverse
        sta FONT_DATA         ; write + auto-increment
        iny
        bne LF_BYTE           ; 256 bytes per page
        inx
        cpx #8                ; 8 pages = 2048 bytes
        bne LF_BLOCK
        rts

; Source pages (ROM $E000) remapped to ATASCII destination order.
; Pages 0-3: normal characters.  Pages 4-7: inverse (EOR $FF).
LF_SRC_HI: .byte $E2,$E0,$E1,$E3,$E2,$E0,$E1,$E3
LF_EOR:    .byte   0,  0,  0,  0,$FF,$FF,$FF,$FF
```

A.2 Detection Snippet (MADS)

```
DETECT_GTIA2RGB:
        lda $D01E
        cmp #$0A
        bne NOT_PRESENT       ; stock GTIA returns $0F
        ; GTIA2RGB present -- optional version read
        lda $D01C             ; major version digit
        sta FW_MAJOR
        lda $D01D             ; minor version digit
        sta FW_MINOR
        rts
NOT_PRESENT:
        rts
```