VERA module

Author: Frank van den Hoef.

This document describes the **V**ideo **E**nhanced **R**etro **A**dapter video-module.

# External address space

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Reg | Addr | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | $9F20 | VERA\_ADDR\_HI | Increment | | | | Address (18:16) | | | |
| 1 | $9F21 | VERA\_ADDR\_MID | Address (15:8) | | | | | | | |
| 2 | $9F22 | VERA\_ADDR\_LO | Address (7:0) | | | | | | | |
| 3 | $9F23 | VERA\_DATA1 | Data port 1 | | | | | | | |
| 4 | $9F24 | VERA\_DATA2 | Data port 2 | | | | | | | |
| 5 | $9F25 | VERA\_CTRL | RESET | - | | | | | | ADDRSEL |
| 6 | $9F26 | VERA\_IEN | - | | | | | SPRCOL | LINE | VSYNC |
| 7 | $9F27 | VERA\_ISR | - | | | | | SPRCOL | LINE | VSYNC |

When RESET is set to 1, the FPGA will reconfigure itself. All registers will be reset. The palette RAM will be set to its default values.

If ADDR\_SEL = 0, register 0/1/2 contain address of data port 1, otherwise register 0/1/2 contain address of data port 2.

After each access of the data port the address is increment by the value in the increment field.

Interrupts will be generated for the interrupt sources set in VERA\_IEN. VERA\_ISR will indicate interrupts that have occurred. Writing a 1 to a position in VERA\_ISR will clear that interrupt status.

# Internal address space

|  |  |
| --- | --- |
| Address range | Description |
| $00000 - $1FFFFF | Video RAM |
| $20000 - $207FF | PETSCII character ROM (upper-case) |
| $20800 - $20FFF | PETSCII character ROM (lower-case) |
| $40000 - $4000F | Layer 1 registers |
| $40010 - $4001F | Layer 2 registers |
| $40020 - $4002F | Sprite registers |
| $40030 - $4003F | Display composer registers |
| $40200 - $403FF | Palette |

# Registers

## Layer 1/2 registers

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Register | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | CTRL0 | MODE | | | VSCALE | | HSCALE | | EN |
| 1 | CTRL1 | - | | TILEH | TILEW | MAPH | | MAPW | |
| 2 | MAP\_BASE\_LO | MAP\_BASE\_LO (9:2) | | | | | | | |
| 3 | MAP\_BASE\_HI | MAP\_BASE\_HI (17:10) | | | | | | | |
| 4 | TILE\_BASE\_LO | TILE\_BASE\_LO (9:2) | | | | | | | |
| 5 | TILE\_BASE\_HI | TILE\_BASE\_HI (17:10) | | | | | | | |
| 6 | HSCROLL\_LO | HSCROLL (7:0) | | | | | | | |
| 7 | HSCROLL\_HI | - | | | | HSCROLL (11:8) | | | |
| 8 | VSCROLL\_LO | VSCROLL (7:0) | | | | | | | |
| 9 | VSCROLL\_HI | - | | | | VSCROLL (11:8) | | | |

Layer 1 registers can be accessed from memory location $40000.

Layer 2 registers can be accessed from memory location $40010.

The layer can be enabled or disabled by setting or clearing the **EN** bit.

The width and height of each rendered pixel can be controlled by the **HSCALE** and **VSCALE** field respectively. A range of 0-3 is available, which results in a pixel width or height of 1-4 pixels.

**MAP\_BASE**, which is composed of **MAP\_BASE\_HI** and **MAP\_BASE\_LO**, specifies the base address where tile map data is fetched from. (Note that the registers don’t specify the lower 2 bits, so the address is always aligned to a multiple of 4 bytes.)

**TILE\_BASE**, which is composed of **TILE\_BASE\_HI** and **TILE\_BASE\_LO**, specifies the base address where tile data is fetched from. (Note that the registers don’t specify the lower 2 bits, so the address is always aligned to a multiple of 4 bytes.)

**HSCROLL**, which is composed of **HSCROLL\_HI** and **HSCROLL\_LO**, specifies the horizontal scroll offset. A value between 0 and 4095 can be used. Increasing the value will cause the picture to move left, decreasing will cause the picture to move right.

**YSCROLL**, which is composed of **YSCROLL\_HI** and **YSCROLL\_LO**, specifies the vertical scroll offset. A value between 0 and 4095 can be used. Increasing the value will cause the picture to move up, decreasing will cause the picture to move down.

**MAPW**, **MAPH** specify the map width and map height respectively:

|  |  |
| --- | --- |
| Value | Map width / height |
| 0 | 32 tiles |
| 1 | 64 tiles |
| 2 | 128 tiles |
| 3 | 256 tiles |

**TILEW**, **TILEH** specify the tile width and tile height respectively:

|  |  |
| --- | --- |
| Value | Tile width / height |
| 0 | 8 |
| 1 | 16 |

### Layer display modes

Each layer supports a few different display modes, which can be selected using the **MODE** field:

|  |  |
| --- | --- |
| Mode | Description |
| 0 | Tile mode 1bpp (per-tile 16 color foreground and background color) |
| 1 | Tile mode 1bpp (per-tile 256 color foreground color and fixed background color 0) |
| 2 | Tile mode 2bpp |
| 3 | Tile mode 4bpp |
| 4 | Tile mode 8bpp |
| 5 | Bitmap mode 2bpp |
| 6 | Bitmap mode 4bpp |
| 7 | Bitmap mode 8bpp |

### Mode 0 – 16 color text mode

**MAP\_BASE** points to a tile map containing tile map entries, which are 2 bytes each:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Offset | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Character index | | | | | | | |
| 1 | Background color | | | | Foreground color | | | |

**TILE\_BASE** points to the character data. This data is organized as 8 bytes per character entry. Each byte represents 1 line of character data, where bit 7 represents the left-most pixel and bit 0 the right-most pixel. If the bit is set the foreground color is used, otherwise the background color. To use the built-in character set this can be set to $8000 for the upper case PETSCII font and to $8200 for the lower case PETSCII font. It is also possible to use a custom character set located in RAM.

### Mode 1 – 256 color text mode

**MAP\_BASE** points to a tile map containing tile map entries, which are 2 bytes each:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Offset | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Character index | | | | | | | |
| 1 | Foreground color | | | | | | | |

**TILE\_BASE** points to the character data. This data is organized as 8 bytes per character entry. Each byte represents 1 line of character data, where bit 7 represents the left-most pixel and bit 0 the right-most pixel. If the bit is set the foreground color is used, otherwise color 0 is used. To use the built-in character set this can be set to $8000 for the upper case PETSCII font and to $8200 for the lower case PETSCII font. It is also possible to use a custom character set located in RAM.

### Mode 2/3/4 – Tile mode 2/4/8bpp

**MAP\_BASE** points to a tile map containing tile map entries, which are 2 bytes each:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Offset | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Tile index (7:0) | | | | | | | |
| 1 | Palette offset | | | | V-flip | H-flip | Tile index (9:8) | |

**TILE\_BASE** points to the tile data. Each tile is 8x8 pixels. Depending on the mode each tile uses either 16, 32 or 64 bytes of memory.

Each pixel in the tile data gives a color index of either 0-3 (2bpp), 0-15 (4bpp), 0-255 (8bpp). This color index is modified by the palette offset in the tile map data using the following logic:

* Color index 0 (transparent) and 16-255 are unmodified.
* Color index 1-15 is modified by adding 16 x palette offset.

TODO: explanation of tile data memory organization

### Mode 5/6/7 – Bitmap mode 2/4/8bpp

**MAP\_BASE** points to the bitmap data.

**TILE\_BASE** isn’t used in these modes.

TODO: explanation of bitmap data memory organization

## Sprite registers

TBD

## Display composer

TBD

## Palette

The palette translate 8-bit color indexes into 12-bit output colors. The palette has 256 entries, each with the following format:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Offset | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | Green | | | | Blue | | | |
| 1 | - | | | | Red | | | |

At reset, the palette will contain a predefined palette:

Color indexes 0-15 contain the C64 color palette.

Color indexes 16-31 contain a grayscale ramp.

Color indexes 32-255 contain various hues, saturation levels, brightness levels.