SuperVidel

Memory Map

Rev 7

2016-07-13

History

Date	Version	Changed
12-03-12	1	Initial release
12-09-18	2	Changed Blit mode "0111" to (Src1.alpha*alpha_reg) from just alpha_reg.
12-09-23	3	Added Replacement RGB values in blitter reg 0x80010078.
13-01-19	4	Added firmware revision register at 0x8001007C.
13-08-18	5	Added new function of Done INT flag in reg 0x80010078.
15-12-06	6	Added Sblitter FIFO write port and status flags
16-07-13	7	Ethernet MAC core data buffers are removed from FW v010

DDR2 RAM

Address	Size	Access types	Description
0xA0000000	16MB	-	Shadow of ST-RAM in DDR2 SDRAM
0xA1000000	112MB	Byte/ Word/ Long Re+Wr	DDR2 SDRAM general purpose graphics RAM

Super Videl settings registers

Address	Size	Access types	Description
0x80010000	Long	Long	**General purpose settings/status register**

Bit	31:24
Name	Reserved
Read/ Write	R
Reset value	%00000000

Bit	23	22	21	20	19	18	17	16
Name	Reserved	DVI VSYNC int flag	DVI VSYNC int en	VGA VSYNC int flag	VGA VSYNC int en	HW revision	DVI protect	VGA protect
Read/ Write	R	R+W	R+W	R+W	R+W	R	R+W	R+W
Reset value	0	0	0	0	0	?	0	0

Bit	15	14	13	12	11	10	9:8
Name	DVI clk is from clock chip	DVI videl clk is 25.175 MHz	VGA clk is from clock chip	VGA videl clk is 25.175 MHz	Videl 25.175 MHz valid	Videl 32MHz valid	Monitor type
Read/ Write	R	R	R	R	R	R	R+W
Reset value	?	?	?	?	?	?	%10

Bit	7	6	5	4	3	2	1	0
Name	DVI connected	SV mode	DVI Hsync	DVI Vsync	VGA Hsync	VGA Vsync	DVI reset	VGA reset
Read/ Write	R	R	R	R	R	R	R+W	R+W
Reset value	0	1	?	?	?	?	1	1

VGA reset:
Set to '1' to hold VGA video system logic in reset. No screen is output on the VGA output.

DVI reset:
Set to '1' to hold DVI video system logic in reset. No screen is output on the DVI output.
VGA Vsync active:
Read-only status bit. Is '1' when a Vsync is in progress on VGA output.
VGA Hsync active:
Read-only status bit. Is '1' when a Hsync is in progress on VGA output.
DVI Vsync active:
Read-only status bit. Is '1' when a Vsync is in progress on DVI output.
DVI Hsync active:
Read-only status bit. Is '1' when a Hsync is in progress on DVI output.
SV mode:
Read-only status bit. Is '1' when the video system is currently in SuperVidel register mode, i.e. the SuperVidel registers were the last ones to be written, and the resolution etc is defined by the SuperVidel registers. If the Videl registers were the last ones to be written, the resolution output is defined by the Videl registers, and this bit is '0'. A copy of this bit can be found in the emulated Videl registers (**0xFFFF82C0.w bit 8**). When the user program reads out the Videl registers before changing resolution, this bit will be read too, and when the user program later restores the old Videl register settings, the status of this bit will be restored too. The SuperVidel will then know wether to restore the old SuperVidel resolution or the old Videl resolution.
DVI connected:
'1' if a powered-on DVI monitor is connected to the DVI port. '0' otherwise.
Monitor type:
Fake monitor type setting used only for the Videl compatiblity. The actual connected monitor type is NOT detected by the hardware. The monitor type only applies to the analog (VGA) output. Values:

Monitor type	Value
Monochrome	″00″
RGB - Colormonitor	″01″
VGA - Colormonitor	″10″
TV	″11″

Videl 32MHz valid:
Says if the input 32 Mhz clock from the falcon motherboard (or from CT60) is detected as running. But not if it is the correct frequency.
Videl 25.175MHz valid:
Says if the input 25.175 Mhz clock from the falcon motherboard is detected as running. But not if it is the correct frequency.
VGA Videl clk is 25.175MHz:
Says if the Videl 25.175MHz(1) clock is chosen instead of the 32MHz(0) for VGA clock selection MUX.
VGA clk is from clock chip:
The VGA clock net is driven by the CDCE925 clock chip(1), not a videl 25.175 or 32MHz clock (0).
DVI Videl clk is 25.175MHz:
Says if the Videl 25.175MHz(1) clock is chosen instead of the 32MHz(0) for DVI clock selection MUX.
DVI clk is from clock chip:
The DVI clock net is driven by the CDCE925 clock chip(1), not a videl 25.175 or 32MHz clock (0).

<table>
<tr><td></td><td></td><td></td><td>VGA protect:
If 1 it will protect the VGA output from switching to Videl register settings when Videl registers are written. On reset this is 0, so the Videl resolution will appear on VGA.
DVI protect:
If 1 it will protect the DVI output from switching to Videl register settings when Videl registers are written. On reset this is 0, so the Videl resolution will appear on DVI.
HW revision:
Is 1 if the SV hardware is a revision 1. Is 0 for a revision 2.
VGA/DVI VBL INT enable & Done INT flags
Writing VGA or DVI INT enable to 1 makes the vertical sync for VGA or DVI generate an interrupt request, and VGA or DVI Done INT flag will be set to 1. You must clear it in your handler by writing 0 to the right Done INT flag. Make sure you put your ISR pointer in vector number 0xC7 (address 0x31C) (for VGA) or 0xC8 (address 0x320) (for DVI) before enabling either interrupt type.</td></tr>
<tr><td>0x80010004</td><td>Long</td><td>Long
Wr only</td><td>CDCE925 PLL1-reg1 (DDR clock)
<table><tr><th>Bit</th><th>31:24</th><th>23:16</th><th>15:8</th><th>7:0</th></tr><tr><td>Name</td><td>Reg0x17</td><td>Reg0x18</td><td>Reg0x19</td><td>Reg0x1A</td></tr><tr><td>Read/Write</td><td>R+W</td><td>R+W</td><td>R+W</td><td>R+W</td></tr><tr><td>Reset value</td><td>0x01</td><td>0x06</td><td>0x90</td><td>0x13</td></tr></table>Use addr 0x17-0x1A (0x17 is MSB) output from CDCE925 ClockFinder program. Write this register first when setting PLL1.</td></tr>
<tr><td>0x80010008</td><td>Long</td><td>Long
Wr only</td><td>CDCE925 PLL1-reg2 (DDR clock)
<table><tr><th>Bit</th><th>31:8</th><th>7:0</th></tr><tr><td>Name</td><td>Reserved</td><td>Reg0x1B</td></tr><tr><td>Read/Write</td><td>R</td><td>R+W</td></tr><tr><td>Reset value</td><td>0x000000</td><td>0x47</td></tr></table>Use addr 0x1B (in LSB) output from the CDCE925 Clockfinder program. Write this register last when setting PLL1, to begin update automatically.</td></tr>
<tr><td>0x8001000C</td><td>Long</td><td>Long
Wr only</td><td>CDCE925 PLL2-reg1 (Video clock)
<table><tr><th>Bit</th><th>31:24</th><th>23:16</th><th>15:8</th><th>7:0</th></tr><tr><td>Name</td><td>Reg0x27</td><td>Reg0x28</td><td>Reg0x29</td><td>Reg0x2A</td></tr><tr><td>Read/Write</td><td>R+W</td><td>R+W</td><td>R+W</td><td>R+W</td></tr><tr><td>Reset value</td><td>0x00</td><td>0x00</td><td>0x00</td><td>0x00</td></tr></table>Use addr 0x27-0x2A (0x27 is MSB) output from CDCE925 ClockFinder program. Write this register first when setting PLL2.</td></tr>
<tr><td>0x80010010</td><td>Long</td><td>Long
Wr only</td><td>CDCE925 PLL2-reg2 (Video clock)
<table><tr><th>Bit</th><th>31:8</th><th>7:0</th></tr><tr><td>Name</td><td>Reserved</td><td>Reg0x2B</td></tr><tr><td>Read/Write</td><td>R</td><td>R+W</td></tr><tr><td>Reset value</td><td>0x000000</td><td>0x00</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td></td><td></td><td>Use addr 0x2B (in LSB) output from the CDCE925 Clockfinder program. Write this register last when setting PLL2, to begin update automatically.</td></tr>
<tr><td>0x80010014</td><td>Long</td><td>Long
Re+Wr</td><td><h2>VGA Screen address</h2>
<table>
<tr><th>Bit</th><th>31:27</th><th>26:1</th><th>0</th></tr>
<tr><td>Name</td><td>Reserved</td><td>Video address</td><td>0</td></tr>
<tr><td>Read/Write</td><td>R</td><td>R+W</td><td>R</td></tr>
<tr><td>Reset value</td><td>0b00000</td><td>0b00000000000000000000000000</td><td>0</td></tr>
</table>
Screen address is 27 bits, but bit 0 is always 0, so an odd adress is not possible. When reading this add 0xA0000000 (DDR2 base address) to get the real screen pointer.</td></tr>
<tr><td>0x80010018</td><td>Long</td><td>Long
Re+Wr</td><td><h2>VGA Horisontal data begin/end register</h2>
<table>
<tr><th>Bit</th><th>31:27</th><th>26:16</th><th>15:11</th><th>10:0</th></tr>
<tr><td>Name</td><td>Reserved</td><td>Hor. data begin</td><td>Reserved</td><td>Hor. data end</td></tr>
<tr><td>Read/Write</td><td>R</td><td>R+W</td><td>R</td><td>R+W</td></tr>
<tr><td>Reset value</td><td>0b00000</td><td>0b00000000000</td><td>0b00000</td><td>0b00000000000</td></tr>
</table>
Horisontal data begin:
Horisontal position on a scanline where the data window begins. Counted in video clock pulses (pixels). Divide the value by 2 after calculation.
Horisontal data end:
Horisontal position on a scanline where the data window ends. Counted in video clock pulses (pixels). Divide the value by 2 after calculation.</td></tr>
<tr><td>0x8001001C</td><td>Long</td><td>Long
Re+Wr</td><td><h2>VGA Horisontal sync register</h2>
<table>
<tr><th>Bit</th><th>31</th><th>30:27</th><th>26:16</th><th>15:11</th><th>10:0</th></tr>
<tr><td>Name</td><td>Polarity</td><td>Reserved</td><td>Hor. sync begin</td><td>Reserved</td><td>Hor. sync end</td></tr>
<tr><td>Read/Write</td><td>R+W</td><td>R</td><td>R+W</td><td>R</td><td>R+W</td></tr>
<tr><td>Reset value</td><td>1</td><td>0b0000</td><td>0b00000000000</td><td>0b00000</td><td>0b00000000000</td></tr>
</table>
Polarity:
Select sync polarity. '1' means sync pulse is active high.
Horisontal sync begin:
Horisontal position on a scanline where the hsync pulse begins. Counted in video clock pulses (pixels). Divide the value by 2 after calculation.
Horisontal sync end:
Horisontal position on a scanline where the hsync pulse ends. Counted in video clock pulses (pixels). Divide the value by 2 after calculation.</td></tr>
<tr><td>0x80010020</td><td>Long</td><td>Long
Re+Wr</td><td><h2>VGA Vertical data begin/end register</h2>
<table>
<tr><th>Bit</th><th>31:27</th><th>26:16</th><th>15:11</th><th>10:0</th></tr>
<tr><td>Name</td><td>Reserved</td><td>Vert data begin</td><td>Reserved</td><td>Vert data end</td></tr>
<tr><td>Read/Write</td><td>R</td><td>R+W</td><td>R</td><td>R+W</td></tr>
<tr><td>Reset value</td><td>0b00000</td><td>0b00000000000</td><td>0b00000</td><td>0b00000000000</td></tr>
</table>
Vertical data begin:
Vertical position (in scanlines) where the data window begins.
Vertical data end:</td></tr>
</table>

<table>
<tr><td></td><td></td><td></td><td>Vertical position (in scanlines) where the data window ends.</td></tr>
<tr><td>0x80010024</td><td>Long</td><td>Long
Re+Wr</td><td>

VGA Vertical sync register

<table>
<tr><th>Bit</th><th>31</th><th>30:27</th><th>26:16</th><th>15:11</th><th>10:0</th></tr>
<tr><th>Name</th><td>Polarity</td><td>Reserved</td><td>Vert. sync begin</td><td>Reserved</td><td>Vert. sync end</td></tr>
<tr><th>Read/Write</th><td>R+W</td><td>R</td><td>R+W</td><td>R</td><td>R+W</td></tr>
<tr><th>Reset value</th><td>1</td><td>0b0000</td><td>0b00000000000</td><td>0b00000</td><td>0b00000000000</td></tr>
</table>

Polarity:
Select sync polarity. '1' means sync pulse is active high.
Vertical sync begin:
Vertical position (in scanlines) where the vsync pulse begins.
Vertical sync end:
Vertical position (in scanlines) where the vsync pulse ends.
</td></tr>
<tr><td>0x80010028</td><td>Long</td><td>Long
Re+Wr</td><td>

VGA Last horisontal/vertical position register

<table>
<tr><th>Bit</th><th>31:27</th><th>26:16</th><th>15:11</th><th>10:0</th></tr>
<tr><th>Name</th><td>Reserved</td><td>Last hor. pos.</td><td>Reserved</td><td>Last vertical pos.</td></tr>
<tr><th>Read/Write</th><td>R</td><td>R+W</td><td>R</td><td>R+W</td></tr>
<tr><th>Reset value</th><td>0b00000</td><td>0b00000000000</td><td>0b00000</td><td>0b00000000000</td></tr>
</table>

Last horisontal position:
The last horisontal position (in pixels), i.e. the last value that the horisontal counter reaches before returning to 0. **Divide the value by 2 after calculation.**
Last vertical position:
The last vertical position (in lines), i.e. the last value that the vertical counter reaches before returning to 0.
</td></tr>
<tr><td>0x8001002C</td><td>Long</td><td>Long
Re+Wr</td><td>

VGA Data amount register

<table>
<tr><th>Bit</th><th>31:30</th><th>29:16</th><th>15:14</th><th>13:0</th></tr>
<tr><th>Name</th><td>Reserved</td><td>Virtual screen data</td><td>Reserved</td><td>Visible screen data</td></tr>
<tr><th>Read/Write</th><td>R</td><td>R+W</td><td>R</td><td>R+W</td></tr>
<tr><th>Reset value</th><td>0b00</td><td>0b00000000000000</td><td>0b00</td><td>0b00000000000000</td></tr>
</table>

Virtual screen data:
The number of bytes on a virtual screen row (the visible data + data to be skipped on each row). Note that bit 0 (at position 16 in the longword register) is always 0, so only even bytes are possible, i.e. whole 16-bit words.
Visible screen data:
The number of bytes on a visible screen row. Note that bits 3:0 are always 0, so only even 16 bytes are possible. Always round up to the nearest 16 bytes, since the extra read data will be discarded.
</td></tr>
<tr><td>0x80010030</td><td>Long</td><td>Long
Re+Wr</td><td>

VGA Video mode settings

<table>
<tr><th>Bit</th><th>31:6</th><th>5</th><th>4</th><th>3</th><th>2:0</th></tr>
<tr><th>Name</th><td>Reserved</td><td>Line doubling</td><td>Double pixel width</td><td>ST comp.</td><td>Video mode</td></tr>
<tr><th>Read/Write</th><td>R</td><td>R+W</td><td>R+W</td><td>R+W</td><td>R+W</td></tr>
<tr><th>Reset value</th><td>0b00000000000000000000000000</td><td>0b0</td><td>0b0</td><td>0b0</td><td>0b000</td></tr>
</table>
</td></tr>
</table>

			<table><tr><th>Bits 2:0</th><th>Video mode</th></tr><tr><td>0b000</td><td>1-bit bitplane (Falcon030 compatible)</td></tr><tr><td>0b001</td><td>2-bit bitplane (Falcon030 compatible)</td></tr><tr><td>0b010</td><td>4-bit bitplane (Falcon030 compatible)</td></tr><tr><td>0b011</td><td>8-bit bitplane (Falcon030 compatible)</td></tr><tr><td>0b100</td><td>8-bit chunky</td></tr><tr><td>0b101</td><td>16-bit highcolor (Falcon030 compatible)</td></tr><tr><td>0b110</td><td>reserved</td></tr><tr><td>0b111</td><td>32-bit truecolor (24-bit RGB + alpha byte)</td></tr></table> **ST compatibility mode:** Set to '1' to turn on ST compatibility mode. (No effect right now). **Double pixel width:** Set to '1' to double the pixel width, i.e. halve the amount of pixels horisontally. This is needed to get a 320 wide resolution with VGA timing (640x480). **Line doubling:** Set to '1' to make each visible line be drawn twice, i.e. halve the visible vertical resolution. This is needed to get a 240 high resolution with VGA timing (640x480).
0x80010034	Long	Long Re+Wr	## DVI Screen address <table><tr><th>Bit</th><th>31:27</th><th>26:1</th><th>0</th></tr><tr><td>Name</td><td>Reserved</td><td>Video address</td><td>0</td></tr><tr><td>Read/Write</td><td>R</td><td>R+W</td><td>R</td></tr><tr><td>Reset value</td><td>0b00000</td><td>0b00000000000000000000000000</td><td>0</td></tr></table> Screen address is 27 bits, but bit 0 is always 0, so an odd adress is not possible. When reading this add 0xA0000000 (DDR2 base address) to get the real screen pointer.
0x80010038	Long	Long Re+Wr	## DVI Horisontal data begin/end register <table><tr><th>Bit</th><th>31:27</th><th>26:16</th><th>15:11</th><th>10:0</th></tr><tr><td>Name</td><td>Reserved</td><td>Hor. data begin</td><td>Reserved</td><td>Hor. data end</td></tr><tr><td>Read/Write</td><td>R</td><td>R+W</td><td>R</td><td>R+W</td></tr><tr><td>Reset value</td><td>0b00000</td><td>0b00000000000</td><td>0b00000</td><td>0b00000000000</td></tr></table> **Horisontal data begin:** Horisontal position on a scanline where the data window begins. Counted in video clock pulses (pixels). **Divide the value by 2 after calculation.** **Horisontal data end:** Horisontal position on a scanline where the data window ends. Counted in video clock pulses (pixels). **Divide the value by 2 after calculation.**
0x8001003C	Long	Long Re+Wr	## DVI Horisontal sync register <table><tr><th>Bit</th><th>31</th><th>30:27</th><th>26:16</th><th>15:11</th><th>10:0</th></tr><tr><td>Name</td><td>Polarity</td><td>Reserved</td><td>Hor. sync begin</td><td>Reserved</td><td>Hor. sync end</td></tr><tr><td>Read/Write</td><td>R+W</td><td>R</td><td>R+W</td><td>R</td><td>R+W</td></tr><tr><td>Reset value</td><td>1</td><td>0b0000</td><td>0b00000000000</td><td>0b00000</td><td>0b00000000000</td></tr></table>

<table>
<tr><td></td><td></td><td></td><td>Polarity:
Select sync polarity. '1' means sync pulse is active high.
Horisontal sync begin:
Horisontal position on a scanline where the hsync pulse begins. Counted in video clock pulses (pixels). Divide the value by 2 after calculation.
Horisontal sync end:
Horisontal position on a scanline where the hsync pulse ends. Counted in video clock pulses (pixels). Divide the value by 2 after calculation.</td></tr>
<tr><td>0x80010040</td><td>Long</td><td>Long
Re+Wr</td><td><h2>DVI Vertical data begin/end register</h2>
<table>
<tr><th>Bit</th><th>31:27</th><th>26:16</th><th>15:11</th><th>10:0</th></tr>
<tr><th>Name</th><td>Reserved</td><td>Vert data begin</td><td>Reserved</td><td>Vert data end</td></tr>
<tr><th>Read/Write</th><td>R</td><td>R+W</td><td>R</td><td>R+W</td></tr>
<tr><th>Reset value</th><td>0b00000</td><td>0b00000000000</td><td>0b00000</td><td>0b00000000000</td></tr>
</table>
Vertical data begin:
Vertical position (in scanlines) where the data window begins.
Vertical data end:
Vertical position (in scanlines) where the data window ends.</td></tr>
<tr><td>0x80010044</td><td>Long</td><td>Long
Re+Wr</td><td><h2>DVI Vertical sync register</h2>
<table>
<tr><th>Bit</th><th>31</th><th>30:27</th><th>26:16</th><th>15:11</th><th>10:0</th></tr>
<tr><th>Name</th><td>Polarity</td><td>Reserved</td><td>Hor. sync begin</td><td>Reserved</td><td>Hor. sync end</td></tr>
<tr><th>Read/Write</th><td>R+W</td><td>R</td><td>R+W</td><td>R</td><td>R+W</td></tr>
<tr><th>Reset value</th><td>1</td><td>0b0000</td><td>0b00000000000</td><td>0b00000</td><td>0b00000000000</td></tr>
</table>
Polarity:
Select sync polarity. '1' means sync pulse is active high.
Vertical sync begin:
Vertical position (in scanlines) where the vsync pulse begins.
Vertical sync end:
Vertical position (in scanlines) where the vsync pulse ends.</td></tr>
<tr><td>0x80010048</td><td>Long</td><td>Long
Re+Wr</td><td><h2>DVI Last horisontal/vertical position register</h2>
<table>
<tr><th>Bit</th><th>31:27</th><th>26:16</th><th>15:11</th><th>10:0</th></tr>
<tr><th>Name</th><td>Reserved</td><td>Last hor. pos.</td><td>Reserved</td><td>Last vertical pos.</td></tr>
<tr><th>Read/Write</th><td>R</td><td>R+W</td><td>R</td><td>R+W</td></tr>
<tr><th>Reset value</th><td>0b00000</td><td>0b00000000000</td><td>0b00000</td><td>0b00000000000</td></tr>
</table>
Last horisontal position:
The last horisontal position (in pixels), i.e. the last value that the horisontal counter reaches before returning to 0. Divide the value by 2 after calculation.
Last vertical position:
The last vertical position (in lines), i.e. the last value that the vertical counter reaches before returning to 0.</td></tr>
<tr><td>0x8001004C</td><td>Long</td><td>Long
Re+Wr</td><td><h2>DVI Data amount register</h2>
<table>
<tr><th>Bit</th><th>31:30</th><th>29:16</th><th>15:14</th><th>13:0</th></tr>
<tr><th>Name</th><td>Reserved</td><td>Virtual screen data</td><td>Reserved</td><td>Visible screen data</td></tr>
</table></td></tr>
</table>

			<table><tr><td>**Read/Write**</td><td>R</td><td>R+W</td><td>R</td><td>R+W</td></tr><tr><td>**Reset value**</td><td>0b00</td><td>0b00000000000000</td><td>0b00</td><td>0b00000000000000</td></tr></table> **Virtual screen data:** The number of bytes on a virtual screen row (the visible data + data to be skipped on each row). Note that bit 0 (at position 16 in the longword register) is always 0, so only even bytes are possible, i.e. whole 16-bit words. **Visible screen data:** The number of bytes on a visible screen row. Note that bits 3:0 are always 0, so only even 16 bytes are possible. Always round up to the nearest 16 bytes, since the extra read data will be discarded.
0x80010050	Long	Long Re+Wr	## DVI Video mode settings <table><tr><td>**Bit**</td><td>**31:6**</td><td>**5**</td><td>**4**</td><td>**3**</td><td>**2:0**</td></tr><tr><td>**Name**</td><td>Reserved</td><td>Line doubling</td><td>Double pixel width</td><td>ST comp.</td><td>Video mode</td></tr><tr><td>**Read/Write**</td><td>R</td><td>R+W</td><td>R+W</td><td>R+W</td><td>R+W</td></tr><tr><td>**Reset value**</td><td>0b00000000000000000000000000</td><td>0b0</td><td>0b0</td><td>0b0</td><td>0b000</td></tr></table> <table><tr><td>**Bits 2:0**</td><td>**Video mode**</td></tr><tr><td>0b000</td><td>1-bit bitplane Falcon030 mode</td></tr><tr><td>0b001</td><td>2-bit bitplane Falcon030 mode</td></tr><tr><td>0b010</td><td>4-bit bitplane Falcon030 mode</td></tr><tr><td>0b011</td><td>8-bit bitplane Falcon030 mode</td></tr><tr><td>0b100</td><td>8-bit chunky</td></tr><tr><td>0b101</td><td>16-bit highcolor Falcon030 mode</td></tr><tr><td>0b110</td><td>reserved</td></tr><tr><td>0b111</td><td>32-bit truecolor (24-bit RGB + alpha byte)</td></tr></table> **ST compatibility mode:** Set to '1' to turn on ST compatibility mode. (No effect right now). **Double pixel width:** Set to '1' to double the pixel width, i.e. halve the amount of pixels horisontally. This is needed to get a 320 wide resolution with VGA timing (640x480). **Line doubling:** Set to '1' to make each visible line be drawn twice, i.e. halve the visible vertical resolution. This is needed to get a 240 high resolution with VGA timing (640x480).
0x80010054	Long	Long Re+Wr	## JTAG control register <table><tr><td>**Bit**</td><td>**31**</td><td>**30:16**</td></tr><tr><td>**Name**</td><td>JTAG enable</td><td>Magic key</td></tr><tr><td>**Read/ Write**</td><td>R+W</td><td>W</td></tr><tr><td>**Reset value**</td><td>0</td><td>%000000000000000</td></tr></table> <table><tr><td>**Bit**</td><td>**15:4**</td></tr><tr><td>**Name**</td><td>Reserved</td></tr></table>

			<table><tr><td>**Read/ Write**</td><td colspan="4">R</td></tr><tr><td>**Reset value**</td><td colspan="4">%000000000000</td></tr></table> <table><tr><td>**Bit**</td><td>**3**</td><td>**2**</td><td>**1**</td><td>**0**</td></tr><tr><td>**Name**</td><td>JTAG TCK</td><td>JTAG TMS</td><td>JTAG TDI</td><td>JTAG TDO</td></tr><tr><td>**Read/ Write**</td><td>R+W</td><td>R+W</td><td>R+W</td><td>R</td></tr><tr><td>**Reset value**</td><td>0</td><td>1</td><td>1</td><td>?</td></tr></table> **Magic key:** Must equal %101010101010101 to allow any changes to the JTAG control bits below, i.e. every write to this register that wishes to modify the JTAG bits must have this value at bits 30-16. **JTAG enable:** 0=JTAG interface disabled and JTAG pins in high impedance. 1=JTAG enabled and FPGA is driving JTAG pins towards PROM memories. **JTAG TDO:** When read, returns the level of the JTAG-TDO signal going from the PROMs to the FPGA. **JTAG TDI:** When written, controls the output level of the JTAG-TDI signal going from the FPGA to the PROMs. When read, returns the level of the internal JTAG TDI signal in the FPGA and not the level of the external signal between the FPGA and the PROMs. So this cannot be used to detect a possible external JTAG master trying to drive the signals. **JTAG TMS:** When written, controls the output level of the JTAG-TMS signal going from the FPGA to the PROMs. When read, returns the level of the internal JTAG TMS signal in the FPGA and not the level of the external signal between the FPGA and the PROMs. So this cannot be used to detect a possible external JTAG master trying to drive the signals. **JTAG TCK:** When written, controls the output level of the JTAG-TCK signal going from the FPGA to the PROMs. When read, returns the level of the internal JTAG TCK signal in the FPGA and not the level of the external signal between the FPGA and the PROMs. So this cannot be used to detect a possible external JTAG master trying to drive the signals.
0x80010058	Long	Long Re+Wr	**SuperBlitter source1 address register** <table><tr><td>**Bit**</td><td>**31:27**</td><td>**26:0**</td></tr><tr><td>**Name**</td><td>Reserved</td><td>Source address 1</td></tr><tr><td>**Read/ Write**</td><td>R</td><td>R+W</td></tr><tr><td>**Reset value**</td><td>0b00000</td><td>0b00000000000000000000000000</td></tr></table> The adress of the main data source, addressing bytes.
0x8001005C	Long	Long Re+Wr	**SuperBlitter source2 address register** <table><tr><td>**Bit**</td><td>**31:27**</td><td>**26:0**</td></tr><tr><td>**Name**</td><td>Reserved</td><td>Source address 2</td></tr></table>

			<table><tr><td>**Read/ Write**</td><td>R</td><td>R+W</td></tr><tr><td>**Reset value**</td><td>0b00000</td><td>0b00000000000000000000000000</td></tr></table> The adress of the secondary data source, addressing bytes. This source is only used when doing some kind of arithmetic/logic operations on src1 + src2 data before writing it to the destination.
0x80010060	Long	Long Re+Wr	**SuperBlitter destination address register** <table><tr><td>**Bit**</td><td>**31:27**</td><td>**26:0**</td></tr><tr><td>**Name**</td><td>Reserved</td><td>Destination address</td></tr><tr><td>**Read/ Write**</td><td>R</td><td>R+W</td></tr><tr><td>**Reset value**</td><td>0b00000</td><td>0b00000000000000000000000000</td></tr></table> The adress of the destination, addressing bytes.
0x80010064	Long	Long Re+Wr	**SuperBlitter line byte count register** <table><tr><td>**Bit**</td><td>**31:24**</td><td>**23:11**</td><td>**10:0**</td></tr><tr><td>**Name**</td><td>Alpha value for mode 0b0111</td><td>Reserved</td><td>Line byte count</td></tr><tr><td>**Read/ Write**</td><td>R</td><td>R</td><td>R+W</td></tr><tr><td>**Reset value**</td><td>0b00000000</td><td>0b0000000000000</td><td>0b00000000000</td></tr></table> **Line byte count:** The amount of bytes that are to be copied in a horizontal line, minus 1. To copy 1 byte, write 0 here. **Alpha value for mode 0b0111:** This 8-bit alpha value is used for the 0b0111 blitter mode, where each pixel in the entire Src1 block uses the same alpha value. Src2 uses (255-alpha).
0x80010068	Long	Long Re+Wr	**SuperBlitter source1 line offset register** <table><tr><td>**Bit**</td><td>**31:27**</td><td>**26:0**</td></tr><tr><td>**Name**</td><td>Reserved</td><td>Line offset</td></tr><tr><td>**Read/ Write**</td><td>R</td><td>R+W</td></tr><tr><td>**Reset value**</td><td>0b00000</td><td>0b00000000000000000000000000</td></tr></table> The amount of bytes that are to be added to the line start adress after a line has been copied, in source1, in order to reach the next line start address. Negative amounts (address decrements) can be realised by using numbers starting with '1' here (in bit 26).
0x8001006C	Long	Long Re+Wr	**SuperBlitter source2 line offset register** <table><tr><td>**Bit**</td><td>**31:27**</td><td>**26:0**</td></tr><tr><td>**Name**</td><td>Reserved</td><td>Line offset</td></tr><tr><td>**Read/ Write**</td><td>R</td><td>R+W</td></tr></table>

			<table><tr><td>**Reset value**</td><td>0b00000</td><td>0b00000000000000000000000000</td></tr></table> The amount of bytes that are to be added to the line start adress after a line has been copied, in source2, in order to reach the next line start address. Negative amounts (address decrements) can be realised by using numbers starting with '1' here (in bit 26).
0x80010070	Long	Long Re+Wr	**SuperBlitter destination line offset register** <table><tr><td>**Bit**</td><td>**31:27**</td><td>**26:0**</td></tr><tr><td>**Name**</td><td>Reserved</td><td>Line offset</td></tr><tr><td>**Read/ Write**</td><td>R</td><td>R+W</td></tr><tr><td>**Reset value**</td><td>0b00000</td><td>0b00000000000000000000000000</td></tr></table> The amount of bytes that are to be added to the line start adress after a line has been copied, in destination, in order to reach the next line start address. Negative amounts (address decrements) can be realised by using numbers starting with '1' here (in bit 26).
0x80010074	Long	Long Re+Wr	**SuperBlitter address mask and number of lines register** <table><tr><td>**Bit**</td><td>**31:24**</td><td>**23:16**</td><td>**15:12**</td><td>**11:0**</td></tr><tr><td>**Name**</td><td>Src1 address count modulo</td><td>Src2 address count modulo</td><td>Reserved</td><td>Number of lines</td></tr><tr><td>**Read/ Write**</td><td>R+W</td><td>R+W</td><td>R</td><td>R+W</td></tr><tr><td>**Reset value**</td><td>0b00000000</td><td>0b00000000</td><td>0b0000</td><td>0b000000000000</td></tr></table> **Src1 adress count mask:** A mask to apply to the source1 address counter while incrementing it during the write phase. The value set will make the adress wrap at that value. Only 2^n values are possible: 0,1,2,4,8...128. Note that we're counting 64-bit words here, not bytes, since the blitter works with 64-bit words internally. Also note that it is only the counter, that is added to the src1 base address, that is affected by this mask, not the src1 base address itself. Setting the mask to 0 will turn off masking of the address counter (but it can only count to 255 anyway). **Src2 address count mask:** The same as the src1 address mask above, but for src2. **Number of lines:** The amount of horizontal lines to do.
0x80010078	Long	Long Re+Wr	**SuperBlitter control and status register** <table><tr><td>**Bit**</td><td>**31:24**</td><td>**23:16**</td><td>**15:8**</td><td>**7**</td><td>**6**</td><td>**5**</td><td>**4:1**</td><td>**0**</td></tr><tr><td>**Name**</td><td>Replace Red</td><td>Replace Green</td><td>Replace Blue</td><td>Replace Enable</td><td>Done INT flag</td><td>INT enable</td><td>Blit mode</td><td>Busy (read) Start (write)</td></tr><tr><td>**Read/ Write**</td><td>R+W</td><td>R+W</td><td>R+W</td><td>R+W</td><td>R+W</td><td>R+W</td><td>R+W</td><td>R+W</td></tr><tr><td>**Reset**</td><td>0b0000</td><td>0b0000</td><td>0b0000</td><td>0b0</td><td>0b0</td><td>0b0</td><td>0b00</td><td>0</td></tr></table>

value	0000	0000	0000				00	(read), 0 (write)

Start/Busy:
When the Busy bit reads as '0' the Start bit may be set to '1' to start the SuperBlitter again. But make sure that the other SuperBlitter registers are properly set up first! While this bit is set to '1' the SuperBlitter is busy performing earlier copy operations.

Blit mode (Yes, the bits are not enumerated in the expected order..) :

Code	Blit mode
0b0000	*dest = src1*
0b0010	Reserved
0b0100	Reserved
0b0110	Reserved
0b1000	Reserved
0b1010	Reserved
0b1100	Reserved
0b1110	Reserved
0b0001	*dest = src1 if src2 != 0, else dest = dest* Each **byte** of src1 is written to dest if the corresponding byte of src2 is not 0. For 8-bit chunky mode: src1=src2=sprite data For 16-bit sprites: src1=sprite data, src2=mask data
0b0011	*dest = src1 * src1.alpha + src2 * (256-src1.alpha)* 32-bit pixel alpha blending. Each pixel is R,G,B,A. 8 bits per channel. Only the alpha channel of src1 is used.
0b0101	*dest = src1 if src1.bit5 == 1, else dest = dest* Each 16bit word of src1 is written to dest if its bit 5 is 1. This is bit 0 of the green channel in a 16bit mode.
0b0111	*dest = src1 * src1.alpha * alpha_reg +* *src2 * (256 - src1.alpha * alpha_reg)* 32-bit pixel alpha blending with global alpha effect too. Each pixel is R,G,B,A. 8 bits per channel. The global alpha value comes from the register at 0x80010064.
0b1001	Reserved
0b1011	Reserved
0b1101	Reserved
0b1111	Reserved

INT enable & Done INT flag
Writing INT enable to 1 makes the blitter generate an interrupt request when it completes a blitting job, and Done INT flag will be set to 1. You must clear it in your handler by writing 0 to Done INT flag. Make sure you put your ISR pointer in vector number 0xC6 (address 0x318) before starting a blitter job with interrupts enabled.

Since rev 6 firmware the Done INT flag can be written to '1' to make the blitter immediately generate a Done interrupt, **if** it was idle (if the blitter was busy with an operation nothing will happen). INT_enable must also be set for this to work.

Replace Enable and Replace Red, Green, Blue
This enables replacing of the RGB part of source1 in 32bit RGBA blitter modes (0b0011 and 0b0111) with the replacement register values Replace Red, Green, Blue.

<table>
<tr>
<td>0x8001007C</td>
<td>Long</td>
<td>Long
Re</td>
<td>

SuperVidel version register

Bit	31:10	9:0
Name	Reserved	Revision
Read/ Write	R	R
Reset value	0b00000000000000000000	?

Revision:
General revision of SV firmware.
</td>
</tr>
<tr>
<td>0x80010080</td>
<td>Long</td>
<td>Long
Re+Wr</td>
<td>

SuperBlitter register FIFO write port

Bit	31:2	1	0
Name	FIFO port bits 31:2	FIFO port bit 1 (write) FIFO full flag (read)	FIFO port bit 0 (write) FIFO empty flag (read)
Read/ Write	W	R+W	R+W
Reset value	-	0b0	0b0

This register is available from FW revision 9.

FIFO port:
Write to this 32bit port to queue SuperBlitter register settings. Starting a blitter operation requires nine register writes, starting at 0x80010058 and going up to and including 0x80010078, in that order. When the FIFO holds at least nine longwords and the blitter is idle, nine longwords will be read out from the FIFO to overwrite the registers at 0x80010058-0x80010078, and the blitter will be started, if the control register holds such settings.

FIFO empty flag:
Is '1' when the blitter FIFO holds less than 9 longwords.

FIFO full flag:
Is '1' when the blitter FIFO holds 500 or more longwords.
</td>
</tr>
</table>

Ethernet MAC core registers

Address	Size	Access types	Description
0x80012000-0x80012053	84 bytes	Long Re+Wr	**Ethernet MAC core control registers** See eth_speci.pdf, section 3.
0x80012400-0x800127FF	1024 bytes	Long Re+Wr	**Ethernet MAC core buffer descriptors** See eth_design_document.pdf: section 2.3.7.3, and eth_speci.pdf: section 4.2.2.
0x80012800-0x80013FFF	6144 bytes	Long Re+Wr	**Ethernet MAC core data buffers** 3 blocks of 2048 bytes form 4 packet buffers of 1536 bytes each. Note that writing this using other sizes than Long will corrupt data. *These buffers are removed from FW v10. The buffers then reside in DDR2 RAM instead.*

256 color palette

<table>
<tr><th>Address</th><th>Size</th><th>Access types</th><th>Description</th></tr>
<tr><td>0xFFFF9800</td><td>1024 bytes</td><td>Byte
Word
Long
Re+Wr</td><td>

256 color palette

Each longword entry looks like this:

<table>
<tr><th>Bit</th><th>31:24</th><th>23:16</th><th>15:8</th><th>7:0</th></tr>
<tr><th>Name</th><td>Red</td><td>Green</td><td>Reserved</td><td>Blue</td></tr>
<tr><th>Read/Write</th><td>R+W</td><td>R+W</td><td>R</td><td>R+W</td></tr>
<tr><th>Reset value</th><td>0x00</td><td>0x00</td><td>0x00</td><td>0x00</td></tr>
</table>

</td></tr>
</table>