A Simple Display Controller For SOC/Embedded Applications V 0.02

Register template to copy around the document

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Control registers

The control registers are defined for a 32 or 64 bit system. They are mostly 32 bits, but the addresses are spaced for a 64 bit memory access. This allows software to have few changes when moving from a 32 to 64 bit environment.

The following registers are extended from 32 to 64 bits when the architecture is changed from 32 to 64 bits.

- BASE = Address of display buffer in memory.
  - Note: LineInc is not expanded to 64 bits. No display buffer larger than 2^30 pixels/line is supported
- CURSOR = Address of displayed cursor bit definitions

For compatibility, The upper 32 bits of all other registers should be written to zero when in 64 bit mode. An implementation can/may ignore these bits on all but the BASE and CURSOR registers

Many registers are accessed at the beginning of VBLANK, a copy made, and that copy is used during the entire next frame. This prevents frame tearing (where part of the image abruptly changes displayed data), and allows the software to change registers anytime the frame is being displayed.

The following registers are copied at VBLANK rising edge, and the copies used during a displayed frame (Only active bits need be copied. The copies are not read back):

- CR
- CUR0
- CUR1
- CURFG
- CURBG
- BASE
- CURSOR

Register CR (offset 0x0000)

CR																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													WRC		EB	Vclk		INT		IV	IH	Pcnt						EN	CE	Mode	

- WRC Working Register Change
  - Defines when the working registers change:
    - Working Registers
      - Base
      - Cursor Registers
      - Lineinc
    - Coding
      - 00 = Vsize
      - 01 = VsyncStart
      - 10 = VsyncEnd
      - 11 = Vend
- EB Enable Cursor Blinking
  - This bit enables the cursor blinking function which Toggles the lower order bit of the cursor display data.
    - change/invert displayed data 00-01
    - swap Foreground and Background colors 10-11
- Vclk Vertical timing clock source
  - 0 = At Hsize -- (Vertical signals will change after last displayed pixel)
  - 1 = At HsyncStart
  - 2 = At HsyncEnd
  - 3 = At Hend
- INT interrupt generation
  - 0 = None
  - 1 = Vsync rising
  - 2 = Vblank rising
  - 3 = Hblank rising
- IV : Invert Vertical sync
- IH : Invert Horizontal sync
- Pcnt : pixel divider. Programs as n-1. This generates a pixel event used in timing horizontal items
- EN : enable the controller
- CE : Cursor enable
- Mode : Display format
  - 0 = 24 BPP (3 byte) – Extension
  - 1 = 32 BPP – base design
  - 2 = 16 BPP (565 RGB) – Extension
  - 3 = Text mode – Extension

Cursor register 0 CUR0 (offset 0x0008)

<table>
<tr><td colspan="32">CUR0</td></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="6">Reserved</td><td colspan="13">CURX</td><td colspan="13">CURY</td></tr>
</table>

- CURX : X cursor position
- CURY : Y cursor position

The display controller keeps the X and Y position of the displayed pixels (Spec uses Xcnt and Ycnt for explanation only). The CURX and CurXsize are compared to the current cursor position Xcnt. If the cursor is enabled and within range, a substitution is made with the cursor data. Selected by accessing two bits at bits 31-(Xcnt-CURX)*2 Xcnt

The two bits are coded as :

- 00 = Transparent, No displayed data substitution
- 01 = Inverted, Displayed data is inverted (R,G,B)
- 10 = Cursor Foreground color
- 11 = Cursor Background color

The CURY and CurYsize are used to see if the cursor is enabled, and on this line. If so, 64 bits are fetched from memory, saved internally, and used to display the cursor. The bits fetched correspond to the cursor row on this line. (Ycnt-CURY)

Cursor register 1 CUR1 (offset 0x0010)

<table>
<tr><td colspan="32">CUR1</td></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">Reserved</td><td colspan="6">CurBlink</td><td colspan="5">CurXsize</td><td colspan="5">CurYsize</td></tr>
</table>

- CurBlink : Number of frames to toggle cursor foreground and background
- CurXsize : Cursor width in pixels
- CurYsize : Cursor width in pixels

Cursor blinking must be enabled (CR → EB). If so, there is a bit which is used to toggle the low order bit of the cursor data. This is set to zero whenever the EB bit is changed. If EB is true, the number of vertical frames displayed is counted. Whenever it is greater than or equal to CurBlink, the counter resets to zero, and the toggle bit is inverted. This allows a simple blinking cursor, and also allows a graphics cursor to blink making it easier to visually locate on the screen.

Cursor Foreground CurFG (offset 0x0018)

CurFG																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Red								Green								Blue							

- Red : foreground pixel color
- Green : foreground pixel color
- Blue : foreground pixel color

Cursor Background CurBG (offset 0x0020)

CurBG																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Red								Green								Blue							

- Red : background pixel color
- Green : background pixel color
- Blue L background pixel color

Horizontal Size and End H1 (offset 0x0028)

H1																															
31	13	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						Hsize													Hend												

- Hsize : Number of displayed pixels per horizontal line
- Hend : Total number of pixels per horizontal line

Horizontal Sync Start and End (offset 0x0030)

H2																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						HorizSyncStart													HorizSyncEnd												

- HorizSyncStart : pixel position of Horizontal sync start
- HorizSyncEnd : pixel position of Horizontal sync end

Vertical Size and End V1 (offset 0x0038)

V1																															
31	13	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						Vsize													Vend													

- Vsize : Number of displayed lines per frame
- Vend : Total number of lines per frame

Vertical Sync Start and End (offset 0x0040)

V2																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						VertSyncStart													VertSyncEnd													

- VertSyncStart : line position of Vertical sync start
- VertSyncEnd : line position of Vertical sync end

BASE Address (offset 0x0048)

Base																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Base																															

- Base : starting address of displayed frame

The base address is the address in memory where the image is stored. The lower three bits of the address should always be zero. This allows 64 bit systems to fetch a single 64 bit word.

LineInc (offset 0x0050)

LineInc																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LineInc																															

- Lineinc : Amount to be added to start of a horizontal line address to get to the start of the next line.

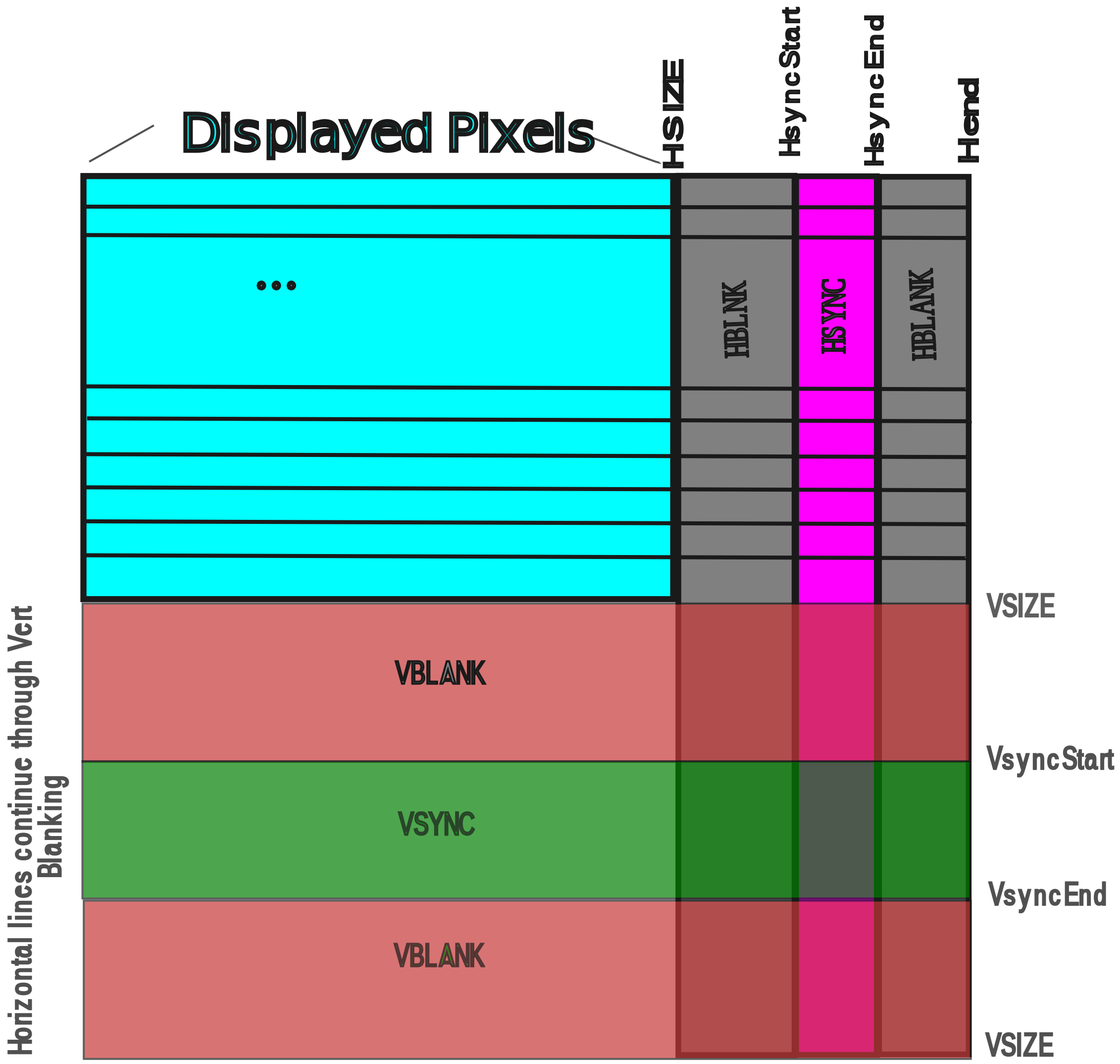
The horizontal address register starts at the beginning of the line. The register is updated on every Hblank signal where Vblank is 0 by adding the LineInc. This address is used to initialize a horizontal memory read pointer at the beginning of Hsync. The horizontal memory read pointer in incremented as data is fetched for the line. A FIFO is often involved as fetches are performed in bursts.

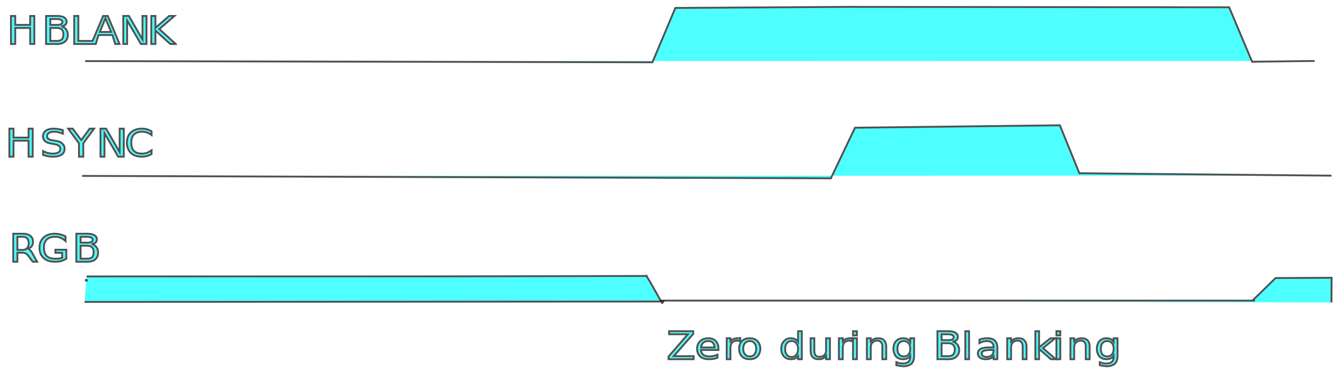
CURSOR (offset 0x0060)

CURSOR																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURSOR																															

- CURSOR : Location of up to 64 rows of 64 bit cursor data

This pointer is used with the CURX, CURY, Xcnt, Ycnt to fetch data during the horizontal blank time. The fetched data is stored in an internal controller register, and used to display the cursor. The data is stored with the HOB displayed first. If the system is 32 bits,


Displayed Pixels
HSIZE
HsyncStart
HsyncEnd
Hend
...
HBLNK
HSYNC
HBLANK
VSIZE
VBLANK
VsyncStart
VSYNC
VsyncEnd
VBLANK
VSIZE
Horizontal lines continue through Vert Blanking



Vertical changes on event defined in Vclk bits in CR register

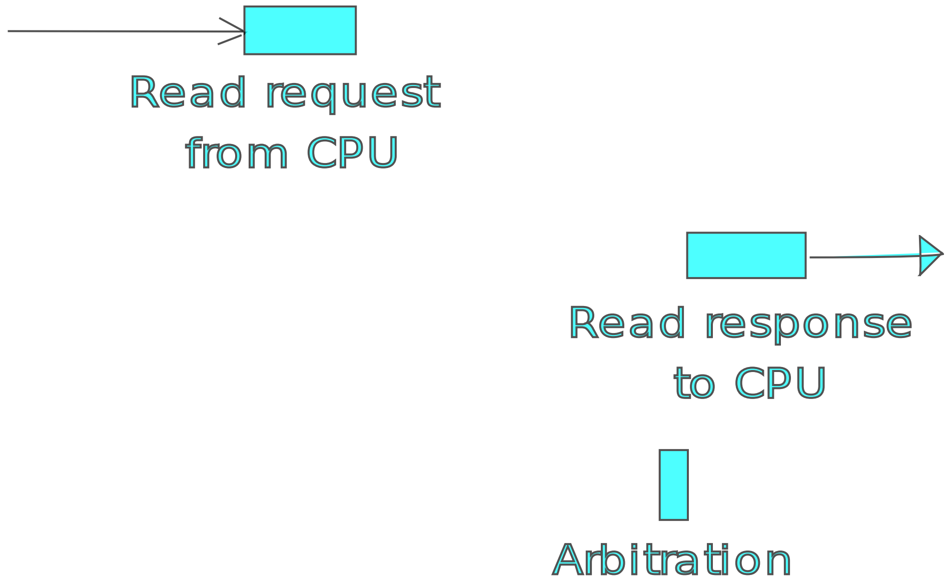
Bus interface

The bus is a simple message bus. It has the following signals:

Name	Dir	Width	Description
CMDIN	In	3	The command into a block: • 000 = No request • 001 = Data Phase • 010 = Read Request • 011 = Read Response • 100 = Write Request • 101 = Write Response • 110 = Read Error • 111 = Write Error
ADDRDATAIN	In	32/64	Address of request, or response
SELIN	In	1	This is the selected device
LENIN	In	2	Burst Length • 00 = 1 transfer • 01 = 4 transfers • 10 = 8 transfers • 11 = 16 transfers
CMDOUT	Out	3	The command from a block: • 000 = No request • 001 = Data Phase • 010 = Read Request • 011 = Read Response • 100 = Write Request • 101 = Write Response • 110 = Message Write • 111 = Message Response
ADDRDATAOUT	Out	32/64	Address or data for an output from the block
LENOUT	Out	2	Number of transfers for this request • 00 = 1 • 01 = 4 • 10 = 8 • 11 = 16
REQOUT	Out	2	Request bid • 00 = No Request • 01 = Low • 10 = Medium • 11 = High

Name	Dir	Width	Description
REQTAR	Out	4	A code for the device target. Used by the fabric, This code with the address selects the target device • 0 = Memory system • 1 = System Device (Interrupt controller, etc) • 2-15 = Other system busses and devices
ACKIN	In	1	Device can send the request

The interface has two one way paths. The input path receives requests and responses to the module. The output path sends requests, and responses from the module. The bus allows disconnected traffic. A read and response are not connected in time. However, the system limits a device to receiving one request at a time for most devices. The multiple request limiting is based on the REQTAR, and in the arbitrator.


Read request
from CPU
Read response
to CPU
Arbitration