

Intel® Iris® Xe and UHD Graphics Open Source

Programmer's Reference Manual

**For the 2020-2021 11th Generation Intel Xeon®, Core™, Celeron®,
Pentium® Gold Processors based on the "Tiger Lake" Platform**

Volume 12: Display Engine

December 2021, Revision 1.0



Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks

Customer is responsible for safety of the overall system, including compliance with applicable safety-related requirements or standards.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, with the sole exceptions that a) you may publish an unmodified copy and b) code included in this document is licensed subject to Zero-Clause BSD open source license (0BSD). You may create software implementations based on this document and in compliance with the foregoing that are intended to execute on the Intel product(s) referenced in this document. No rights are granted to create modifications or derivatives of this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Table of Contents

Display	1
Terminology.....	1
VGA and Extended VGA Registers.....	3
General Control and Status Registers	4
Sequencer Registers.....	8
Graphics Controller Registers	12
Attribute Controller Registers.....	24
VGA Color Palette Registers.....	30
CRT Controller Register.....	34
Display Audio Codec Verbs.....	55
Block Diagram.....	55
Codec Node Hierarchy	55
Programming.....	57
North Display Engine Registers.....	110
X ^e _D Overview.....	110
Mode Set.....	122
Clocks	168
Shared Functions	197
Central Power.....	208
Pipe	221
Planes	279
DSC.....	308
Transcoder.....	316
Audio	376
DisplayPort Transport	390
Digital Display Interface.....	390
Global Time Code (GTC)	419
Display Watermark Programming	421
Watermark Calculations.....	421
Memory Values.....	428
Retrieve Memory Latency Data.....	428
Memory Latency Data Definition	428



SAGV Block Time429

Display

Terminology

Term	Description
DP	DisplayPort
SST, DP SST	DisplayPort Single Stream Transport
MST, DP MST	DisplayPort Multi Stream Transport

Register Access Field	Description	Implementation
R/W (Read/Write)	The value written into this register will control hardware and is the same value that will be read.	Write data is stored. Read is from the stored data. Stored value is used to control hardware.
Reserved	Unused register bit. Don't assume a value for these bits. Writes have no effect.	Write data is ignored. Read is zero.
MBZ (Must Be Zero)	Always write a zero to this register.	May be implemented as Reserved or as R/W.
PBC (Preserve Bit Contents)	Software must write the original value back to this bit. This allows new features to be added using these bits.	May be implemented as Reserved or as R/W.
Read Only	The read value is determined by hardware. Writes to this bit have no effect.	Write data is ignored. Read is from a status signal or some other internal source.
Write Only	The value written into this register will control hardware. Reads return zero.	Write data is stored. Read is zero. Stored value is used to control hardware.
R/W Clear (Read/Write Clear)	Sticky status bit. Hardware will set the bit, software can clear it with a write of 1b.	Internal hardware events set a sticky bit. Read is from the sticky bit. A write of 1b clears the sticky bit.
R/W Set (Read/Write Set)	Sticky status bit. Software can set the bit with a write of 1b. Hardware will clear the bit.	A write of 1b sets a sticky bit. Internal hardware events clear a sticky bit. Read is from the sticky bit.
Double Buffered	Write when desired and the written value will take effect at the time of the update point specified in the 'Double Buffer Update Point'	Two stages of registers used. Write data is stored into first

Register Access Field	Description	Implementation
	<p>parameter.</p> <p>Reads will return the written value, which is not necessarily the value being currently used to control hardware. Some double-buffered registers have a corresponding "LIVE" read only register that provides the value being use to control hardware.</p> <p>Some have a specific arming sequence where a write to another register, specified in the 'Double Buffer Armed By' parameter, is required before the update can take place. Once the armed by register is written to, the written values of all registers controlled by that arming will take effect at the time of the double buffer update point. This is used to ensure atomic updates of several registers.</p> <p>Note: Once armed, by write to the armed by register, the registers controlled by this arming should not be changed until the double buffer update point is reached. If changed, this will disarm the sequence and will require another write to the armed by register to get it to the armed status again.</p>	<p>stage. Read is from the first stage stored data.</p> <p>First stage stored value is transferred to second stage storage at the double buffer update point.</p> <p>Second stage stored value is used to control hardware.</p> <p>Arm/disarm logic may be used for some registers to control the double buffer update point.</p>
Write/Read Status	The value written into this register will control hardware. The read value is determined by hardware.	<p>Write data is stored. Stored value is used to control hardware.</p> <p>Read is from a status signal or some other internal source.</p>

VGA and Extended VGA Registers

This section describes the registers and the functional operation notations for the observable registers in the VGA section. This functionality is provided as a means for support of legacy applications and operating systems.

It is important to note that these registers in general have the desired effects only when running VGA display modes. The main exceptions to this are the palette interface which allows real mode DOS applications and full screen VGA applications under an OS control running in high resolution (non-VGA) modes to access the palette through the VGA register mechanisms and the use of the ST01 status bits that determine when the VGA enters display enable and sync periods. Other exceptions include the register bits that control the memory accesses through the A000:0000 and B000:0000 memory segments which are used during operating system emulation of VGA for "DOS box" applications.

Some of the functions of the VGA are enabled or defeated through the programming of the VGA control register bits that are located in the MMIO register space.

Given the legacy nature of this function, it has been adapted to the changing environment that it must operate within. The three most notable changes are the addition of high resolution display mode support, new operating system support, and the use of fixed resolution display devices (such as LCD panels). Additional control bits in the PCI Config space will affect the ability to access the registers and memory aperture associated with VGA.

Mode of Operation	VGA Disable	VGA Display	VGA Registers	Palette (VGA)	VGA Memory	VGA Banking
VGA DOS	No	Yes	Yes	Yes	Yes	No
HiRes DOS	Yes	No	Yes	Yes	No	Yes
Fullscreen DOS	Yes/No	No/Yes	Yes	Yes	Yes	Yes
DOS Emulation	Yes	No	Yes	Yes	Yes	Yes

VGA Display Mode	Dot Clock Select	Dot Clock Range	132 Column Text Support	9-Dot Disable Support	Main Use
Native	VGA Clock Select	25/28 MHz	No	No	Analog CRT (VGA connector)
Centered	Fixed at display Requirements	Product Specific	No	Yes	Digital Display
Upper Left Corner	Fixed at display Requirements	Product Specific	No	Yes	Internal Panel

Native, Centered, and Upper Left Corner support varies from product to product.

Even in the native VGA display operational modes, not all combinations of bit settings result in functional operating modes. VGA display modes have the restriction that they can be used only when all other display planes are disabled.



These registers are accessed via I/O space. The I/O space resides in the PCI compatibility hole and uses only the addresses that were part of the original VGA I/O space (which includes EGA and MDA emulation). Accesses to the VGA I/O addresses are steered to the proper bus and rely on proper setup of bridge registers. Extended VGA registers such as GR10 and GR11 use additional indexes for the already defined I/O addresses. VGA register accesses are allowed as 8 or 16 bit naturally aligned transactions only. Word transactions must have the least significant bit of the address set to zero. DWORD I/O operations should not be performed on these registers.

Some products may support access to these registers through MMIO. The access method varies and is documented elsewhere.

General Control and Status Registers

The setup, enable, and general registers are all directly accessible by the CPU. A sub indexing scheme is not used to read from and write to these registers.

Various bits in these registers provide control over the real-time status of the horizontal sync signal, the horizontal retrace interval, the vertical sync signal, and the vertical retrace interval. The horizontal retrace interval is the period during the drawing of each scan line containing active video data, when the active video data is not being displayed. This period includes the horizontal front and back porches, and the horizontal sync pulse. The horizontal retrace interval is always longer than the horizontal sync pulse. The vertical retrace interval is the period during which the scan lines not containing active video data are drawn. This includes the vertical front porch, back porch, and the vertical sync pulse. The vertical retrace interval is normally longer than the vertical sync pulse.

ST00 - Input Status 0

Address: 3C2h

Default: 00h

Attributes: Read Only

Bit	Descriptions
7	CRT Interrupt Pending. This bit is here for EGA compatibility and will always return zero. The generation of interrupts was originally enabled, through bits [4,5] of the Vertical Retrace End Register (CR11). This ability to generate interrupts at the start of the vertical retrace interval is a feature that is typically unused by DOS software and therefore is only supported through other means for use under a operating system support. 0 = CRT (vertical retrace interval) interrupt is not pending. 1 = CRT (vertical retrace interval) interrupt is pending
6:5	Reserved. Read as 0s.
4	RGB Comparator / Sense. This bit is here for compatibility and will always return one. Monitor detection must be done through the programming of hotplug registers in the MMIO space. 0 = Below threshold 1 = Above threshold
3:0	Reserved. Read as 0s.

ST01 - Input Status 1

Address: 3BAh/3DAh

Default: 00h

Attributes: Read Only

The address selection is dependent on CGA or MDA emulation mode as selected via the MSR register.

Bit	Descriptions
7	Reserved (as per VGA specification). Read as 0s.
6	Reserved. Read as 0.
5:4	Video Feedback 1, 0. These bits are connected to 2 of the 8 color bits sent to the palette. Bits 4 and 5 of the Color Plane Enable Register (AR12) selects which two of the 8 possible color bits become connected to these 2 bits of this register. These bits exist for EGA compatibility.
3	<p>Vertical Retrace/Video.</p> <p>0 = VSYNC inactive (Indicates that a vertical retrace interval is not taking place).</p> <p>1 = VSYNC active (Indicates that a vertical retrace interval is taking place).</p> <p>VGA pixel generation is not locked to the display output but is loosely coupled. A VSYNC indication may not occur during the actual VSYNC going to the display but during the VSYNC that is generated as part of the VGA pixel generation. The exact relationship will vary with the VGA display operational mode. This status bit will remain active when the VGA is disabled and the device is running in high resolution modes (non-VGA) to allow for applications that (now incorrectly) use these status registers bits. In this case, the status will come from the pipe that the VGA is assigned to.</p> <p>Bits 4 and 5 of the Vertical Retrace End Register (CR11) previously could program this bit to generate an interrupt at the start of the vertical retrace interval. This ability to generate interrupts at the start of the vertical retrace interval is a feature that is largely unused by legacy software. Interrupts are not supported through the VGA register bits.</p>
2:1	Reserved. Read as 0s.
0	<p>Display Enable Output. Display Enable is a status bit (bit 0) in VGA Input Status Register 1 that indicates when either a horizontal retrace interval or a vertical retrace interval is taking place. This bit was used with the EGA graphics system (and the ones that preceded it, including MDA and CGA). In those cases, it was important to check the status of this bit to ensure that one or the other retrace intervals was taking place before reading from or writing to the frame buffer. In these earlier systems, reading from or writing to the frame buffer at times outside the retrace intervals meant that the CRT controller would be denied access to the frame buffer. Those behaviors resulted in either "snow" or a flickering display. This bit provides compatibility with software designed for those early graphics controllers. This bit is currently used in DOS applications that access the palette to prevent the sparkle associated with read and write accesses to the palette RAM with the same address on the same clock cycle.</p> <p>This status bit remains active when the VGA display is disabled and the device is running in high resolution modes (non-VGA) to allow for applications that (now considered incorrect) use these status registers bits. In this case, the status will come from the pipe that the VGA is assigned to. When in panel fitting VGA or centered VGA operation, the meaning of these bits will not be consistent with native VGA timings.</p> <p>0 = Active display data is being sent to the display. Neither a horizontal retrace interval or a vertical retrace interval is currently taking place.</p> <p>1 = Either a horizontal retrace interval (horizontal blanking) or a vertical retrace interval (vertical blanking) is</p>



Bit	Descriptions
	currently taking place.

FCR - Feature Control

Address: 3BAh/3DAh - Write; 3CAh - Read

Default: 00h

Attributes: Read/Write

The address used for writes is dependent on CGA or MDA emulation mode as selected via the MSR register. In the original EGA, bits 0 and 1 were used as part of the feature connector interface. Feature connector is not supported in these devices and those bits will always read as zero.

Bit	Descriptions
7:4	Reserved. Read as 0.
3	VSYNC Control. This bit is provided for compatibility only and has no other function. Reads and writes to this bit have no effect other than to change the value of this bit. The previous definition of this bit selected the output on the VSYNC pin. 0 = Was used to set VSYNC output on the VSYNC pin (default). 1 = Was used to set the logical 'OR' of VSYNC and Display Enable output on the VSYNC pin. This capability was not typically very useful.
2:0	Reserved. Read as 0.

MSR - Miscellaneous Output

Address: 3C2h - Write; 3CCh - Read

Default: 00h

Attributes: Read/Write

Bit	Descriptions
7	CRT VSYNC Polarity. This is a legacy function that is used in native VGA modes. For most cases, sync polarity will be controlled by the port control bits. The VGA settings can be optionally selected for compatibility with the original VGA when used in the VGA native mode. Sync polarity was used in VGA to signal the monitor how many lines of active display are being generated. 0 = Positive Polarity (default). 1 = Negative Polarity.
6	CRT HSYNC Polarity. This is a legacy function that is used in native VGA modes. For most cases, sync polarity will be controlled by the port control bits. The VGA settings can be optionally selected for compatibility with

Bit	Descriptions
	<p>the original VGA when used in the VGA native mode.</p> <p>0 = Positive Polarity (default).</p> <p>1 = Negative Polarity</p>
5	<p>Page Select. In Odd/Even Memory Map Mode 1 (GR6), this bit selects the upper or lower 64 KB page in display memory for CPU access:</p> <p>0 = Upper page (default)</p> <p>1 = Lower page.</p> <p>Selects between two 64KB pages of frame buffer memory during standard VGA odd/even modes (modes 0h through 5h). Bit 1 of register GR06 can also program this bit in other modes. This bit is would normally set to 1 by the software.</p>
4	<p>Reserved. Read as 0.</p>
3:2	<p>Clock Select. These bits can select the dot clock source for the CRT interface. The bits should be used to select the dot clock in standard native VGA modes only. When in the centering or upper left corner modes, these bits should be set to have no effect on the clock rate. The actual frequencies that these bits select, if they have any affect at all, is programmable through the PLL MMIO registers.</p> <p>00 = CLK0, 25.175 MHz (for standard VGA modes with 640 pixel (8-dot) horizontal resolution) (default)</p> <p>01 = CLK1, 28.322 MHz. (for standard VGA modes with 720 pixel (9-dot) horizontal resolution)</p> <p>10 = Was used to select an external clock (now unused)</p> <p>11 = Reserved</p>
1	<p>A0000-BFFFFh Memory Access Enable. VGA Compatibility bit enables access to video memory (frame buffer) at A0000-BFFFFh. When disabled, accesses to VGA memory are blocked in this region. This bit is independent of and does not block CPU access to the video linear frame buffer at other addresses.</p> <p>0 = Prevent CPU access to memory/registers/ROM through the A0000-BFFFF VGA memory aperture (default).</p> <p>1 = Allow CPU access to memory/registers/ROM through the A0000-BFFFF VGA memory aperture. This memory must be mapped as UC by the CPU.</p>
0	<p>I/O Address Select. This bit selects 3Bxh or 3Dxh as the I/O address for the CRT Controller registers, the Feature Control Register (FCR), and Input Status Register 1 (ST01). Presently ignored (whole range is claimed), but will "ignore" 3Bx for color configuration or 3Dx for monochrome.</p> <p>It is typical in AGP chipsets to shadow this bit and properly steer I/O cycles to the proper bus for operation where a MDA exists on another bus such as ISA.</p> <p>0 = Select 3Bxh I/O address (MDA emulation) (default).</p> <p>1 = Select 3Dxh I/O address (CGA emulation).</p>

In standard VGA modes using the analog VGA connector, bits 7 and 6 indicate which of the three standard VGA vertical resolutions the standard VGA display should use. Extended modes, including those with a vertical resolution of 480 scan lines, may use a setting of 0 for both of these bits. Different



connector standards and timing standards specify the proper use of sync polarity. This setting was "reserved" in the VGA standard.

Analog CRT Display Sync Polarities

V	H	Display	Horizontal Frequency	Vertical Frequency
P	P	200 Line	15.7 KHz	60 Hz
N	P	350 Line	21.8 KHz	60 Hz
P	N	400 Line	31.5 KHz	70 Hz
N	N	480 Line	31.5 KHz	60 Hz

Sequencer Registers

To access registers the VGA Sequencer Index register (SRX) at address 3C4h is written with the index of the desired register. Then the desired register is accessed through the data port for the sequencer registers at address 3C5.

SRX - Sequencer Index

Address: 3C4h

Default: 00h

Attributes: Read/Write

Bit	Description
7:3	Reserved. Read as 0s.
2:0	Sequencer Index. This field contains a 3-bit Sequencer Index value used to access sequencer data registers at indices 0 through 7.

SR00 - Sequencer Reset

Address: 3C5h(Index=00h)

Default: 00h

Attributes: Read/Write

Bit	Descriptions
7:2	Reserved. Read as 0.
1	Reserved. Reserved for VGA compatibility (was reset).
0	Reserved. Reserved for VGA compatibility. (was reset)

SR01 - Clocking Mode

Address: 3C5h (Index=01h)

Default: 00h

Attributes: Read/Write

Bit	Descriptions
7:6	Reserved. Read as 0s.
5	<p>Screen Off.</p> <p>0 = Normal Operation (default).</p> <p>1 = Disables video output (blanks the screen) and turns off display data fetches. Synchronization pulses to the display, however, are maintained. Setting this bit to 1 had been used as a way to more rapidly update and improve CPU access performance to the frame buffer during VGA modes. In non-VGA modes (VGA Disable=1), this bit has no effect. Before the VGA is disabled through the MMIO VGA control register, this bit should be set to stop the memory accesses from the display.</p> <p>The following sequence must be used when disabling the VGA plane.</p> <ol style="list-style-type: none"> 1. Write SR01 to set bit 5 = 1 to disable video output. 2. Wait for 100us. 3. Disable the VGA plane via Bit 31 of the MMIO VGA control register (location found in the MMIO display register programming specification).
4	<p>Shift 4.</p> <p>0 = Load video shift registers every 1 or 2 character clocks (depending on bit 2 of this register) (default).</p> <p>1 = Load shift registers every 4th character clock.</p>
3	<p>Dot Clock Divide. Setting this bit to 1 stretches doubles all horizontal timing periods that are specified in the VGA horizontal CRTIC registers. This bit is used in standard VGA 40-column text modes to stretch timings to create horizontal resolutions of either 320 or 360 pixels (as opposed to 640 or 720 pixels, normally used in standard VGA 80-column text modes). The effect of this is that there will actually be twice the number of pixels sent to the display per line.</p> <p>0 = Pixel clock is left unaltered (used for 640 (720) pixel modes); (default).</p> <p>1 = Pixel clock divided by 2 (used for 320 (360) pixel modes).</p>
2	<p>Shift Load. Bit 4 of this register must be 0 for this bit to be effective.</p> <p>0 = Load video data shift registers every character clock (default).</p> <p>1 = Load video data shift registers every other character clock.</p>
1	Reserved. Read as 0.
0	<p>8/9 Dot Clocks. This bit determines whether a character clock is 8 or 9 dot clocks long if clock doubling is disabled and 16 or 18 clocks if it is. This also changes the interpretation of the pixel panning values (see chart). An additional control bit determines if this bit is to be ignored and 8-dot characters are to be used always. The 9-dot disable would be used when doubling the horizontal pixels on a 1280 wide display or non-doubling on a 640 wide display. Panning however will occur according to the expected outcome.</p> <p>0 = 9 dot clocks (9 horizontal pixels) per character in text modes with a horizontal resolution of 720 pixels.</p> <p>1 = 8 dot clocks (8 horizontal pixels) per character in text or graphics modes with a horizontal resolution of</p>



Bit	Descriptions
	640 pixels.

SR02 - Plane/Map Mask

Address: 3C5h (Index=02h)

Default: 00h

Attributes: Read/Write

Bit	Descriptions
7:4	Reserved. Read as 0s.
3:0	<p>Memory Planes [3:0] Processor Write Access Enable. In both the Odd/Even Mode and the Chain 4 Mode, these bits still control access to the corresponding color plane.</p> <p>0 = Disable. 1 = Enable.</p> <p>This register is referred to in the VGA standard as the Map Mask Register.</p>

SR03 - Character Font

Address: 3C5h (index=03h)

Default: 00h

Attributes: Read/Write

In text modes, bit 3 of the video data's attribute byte normally controls the foreground intensity. This bit may be redefined to control switching between character sets. This latter function is enabled whenever there is a difference in the values of the Character Font Select A and the Character Font Select B bits. If the two values are the same, the character select function is disabled and attribute bit 3 controls the foreground intensity.

Bit 1 of the Memory Mode Register (SR04) must be set to 1 for the character font select function of this register to be active. Otherwise, only character maps 0 and 4 are available.

Bit	Descriptions																		
7:6	Reserved. Read as 0s.																		
3:2,5	<p>Character Map Select Bits for Character Map B. These three bits are used to select the character map (character generator tables) to be used as the secondary character set (font). The numbering of the maps is not sequential.</p> <table border="1"> <thead> <tr> <th>Bit [3:2,5]</th> <th>Map Number</th> <th>Table Location</th> </tr> </thead> <tbody> <tr> <td>00,0</td> <td>0</td> <td>1st 8KB of plane 2 at offset 0 (default)</td> </tr> <tr> <td>00,1</td> <td>4</td> <td>2nd 8KB of plane 2 at offset 8K</td> </tr> <tr> <td>01,0</td> <td>1</td> <td>3rd 8KB of plane 2 at offset 16K</td> </tr> <tr> <td>01,1</td> <td>5</td> <td>4th 8KB of plane 2 at offset 24K</td> </tr> <tr> <td>10,0</td> <td>2</td> <td>5th 8KB of plane 2 at offset 32K</td> </tr> </tbody> </table>	Bit [3:2,5]	Map Number	Table Location	00,0	0	1st 8KB of plane 2 at offset 0 (default)	00,1	4	2nd 8KB of plane 2 at offset 8K	01,0	1	3rd 8KB of plane 2 at offset 16K	01,1	5	4th 8KB of plane 2 at offset 24K	10,0	2	5th 8KB of plane 2 at offset 32K
Bit [3:2,5]	Map Number	Table Location																	
00,0	0	1st 8KB of plane 2 at offset 0 (default)																	
00,1	4	2nd 8KB of plane 2 at offset 8K																	
01,0	1	3rd 8KB of plane 2 at offset 16K																	
01,1	5	4th 8KB of plane 2 at offset 24K																	
10,0	2	5th 8KB of plane 2 at offset 32K																	

Bit	Descriptions		
	10,1	6	6th 8KB of plane 2 at offset 40K
	11,0	3	7th 8KB of plane 2 at offset 48K
	11,1	7	8th 8KB of plane 2 at offset 56K
1:0,4	Character Map Select Bits for Character Map A. These three bits are used to select the character map (character generator tables) to be used as the primary character set (font). The numbering of the maps is not sequential.		
	Bit [1:0,4]	Map Number	Table Location
	00,0	0	1st 8KB of plane 2 at offset 0 (default)
	00,1	4	2nd 8KB of plane 2 at offset 8K
	01,0	1	3rd 8KB of plane 2 at offset 16K
	01,1	5	4th 8KB of plane 2 at offset 24K
	10,0	2	5th 8KB of plane 2 at offset 32K
	10,1	6	6th 8KB of plane 2 at offset 40K
	11,0	3	7th 8KB of plane 2 at offset 48K
	11,1	7	8th 8KB of plane 2 at offset 56K

SR04 - Memory Mode Register

Address: 3C5h (index=04h)

Default: 00h

Attributes: Read/Write

Bit	Description
7:4	Reserved. Read as 0.
3	Chain 4 Mode. The selections made by this bit affect both CPU Read and write accesses to the frame buffer. 0 = The manner in which the frame buffer memory is mapped is determined by the setting of bit 2 of this register (default). 1 = The frame buffer memory is mapped in such a way that the function of address bits 0 and 1 are altered so that they select planes 0 through 3. This setting is used in mode x13 to allow all four planes to be accessed via sequential addresses.
2	Odd/Even Mode. Bit 3 of this register must be set to 0 for this bit to be effective. The selections made by this bit affect only non-paged CPU accesses to the frame buffer through the VGA aperture. 0 = The frame buffer memory is mapped in such a way that the function of address bit 0 such that even addresses select planes 0 and 2 and odd addresses select planes 1 and 3 (default). 1 = Addresses sequentially access data within a bit map, and the choice of which map is accessed is made according to the value of the Plane Mask Register (SR02).
1	Extended Memory Enable. This bit must be set to 1 to enable the selection and use of character maps in plane 2 via the Character Map Select Register (SR03). 0 = Disable CPU accesses to more than the first 64KB of VGA standard memory (default). 1 = Enable CPU accesses to the rest of the 256KB total VGA memory beyond the first 64KB.



Bit	Description
0	Reserved. Read as 0.

SR07 - Horizontal Character Counter Reset

Address: 3C5h (index=07h)

Default: 00h

Attributes: Read/Write

For standard VGAs, writing this register (with any data) causes the horizontal character counter to be held in reset (the character counter output will remain 0). It remained in reset until a write occurred to any other sequencer register location with SRX set to an index of 0 through 6. In this implementation that sequence has no such special effect.

The vertical line counter is clocked by a signal derived from the horizontal display enable (which does not occur if the horizontal counter is held in reset). Therefore, if a write occurs to this register during the vertical retrace interval, both the horizontal and vertical counters will be set to 0. A write to any other sequencer register location (with SRX set to an index of 0 through 6) may then be used to start both counters with reasonable synchronization to an external event via software control. Although this was a standard VGA register, it was not documented.

Bit	Description
7:0	Horizontal Character Counter.

Graphics Controller Registers

Accesses to the registers of the VGA Graphics Controller are done through the use of address 3CEh written with the index of the desired register. Then the desired register is accessed through the data port for the graphics controller registers at address 3CFh. Indexes 10 and 11 must only be accessed through the I/O space.

GRX - GRX Graphics Controller Index Register

Address: 3CEh

Default:

000UUUUUb (U=Undefined)

Attributes:

Read/Write

Bit	Description
7:5	Reserved. Read as 0.
4:0	Graphics Controller Register Index. This field selects any one of the graphics controller registers (GR00-GR18) to be accessed via the data port at address 3CFh.

GR00 - Set/Reset Register

Address: 3CFh (index=00h)

Default: 0Uh (U=Undefined)

Attributes:

Read/Write

Bit	Description
7:4	Reserved. Read as 0.
3:0	<p>Set/Reset Plane [3:0]. When the Write Mode bits (bits 0 and 1) of the Graphics Mode Register (GR05) are set to select Write Mode 0, all 8 bits of each byte of each memory plane are set to either 1 or 0 as specified in the corresponding bit in this register, if the corresponding bit in the Enable Set/Reset Register (GR01) is set to 1.</p> <p>When the Write Mode bits (bits 0 and 1) of the Graphics Mode Register (GR05) are set to select Write Mode 3, all CPU data written to the frame buffer is rotated, then logically ANDed with the contents of the Bit Mask Register (GR08), and then treated as the addressed data's bit mask, while value of these four bits of this register are treated as the color value.</p>

GR01 - Enable Set/Reset Register

Address: 3CFh (Index=01h)

Default: 0Uh (U=Undefined)

Attributes:

Read/Write

Bit	Description
7:4	Reserved. Read as 0.
3:0	<p>Enable Set/Reset Plane [3:0].</p> <p>This register works in conjunction with the Set/Reset Register (GR00). The Write Mode bits (bits 0 and 1) must be set for Write Mode 0 for this register to have any effect.</p> <p>0 = The corresponding memory plane can be read from or written to by the CPU without any special bitwise operations taking place.</p> <p>1 = The corresponding memory plane is set to 0 or 1 as specified in the Set/Reset Register (GR00).</p>



GR02 - Color Compare Register

Address: 3CFh (Index=02h)

Default: 0Uh (U=Undefined)

Attributes:

Read/Write

Bit	Description
7:4	Reserved. Read as 0.
3:0	Color Compare Plane [3:0]. When the Read Mode bit (bit 3) of the Graphics Mode Register (GR05) is set to select Read Mode 1, all 8 bits of each byte of each of the 4 memory planes of the frame buffer corresponding to the address from which a CPU read access is being performed are compared to the corresponding bits in this register (if the corresponding bit in the Color Don't Care Register (GR07) is set to 1). The value that the CPU receives from the read access is an 8-bit value that shows the result of this comparison, wherein value of 1 in a given bit position indicates that all of the corresponding bits in the bytes across all of the memory planes that were included in the comparison had the same value as their memory plane's respective bits in this register.

GR03 - Data Rotate Register

Address: 3CFh (Index=03h)

Default: 0Uh (U=Undefined)

Attributes:

Read/Write

Bit	Description
7:5	Reserved. Read as 0.
4:3	Function Select. These bits specify the logical function (if any) to be performed on data that is meant to be written to the frame buffer (using the contents of the memory read latch) just before it is actually stored in the frame buffer at the intended address location. 00 = Data being written to the frame buffer remains unchanged, and is simply stored in the frame buffer. 01 = Data being written to the frame buffer is logically ANDed with the data in the memory read latch before it is actually stored in the frame buffer. 10 = Data being written to the frame buffer is logically ORed with the data in the memory read latch before it is actually stored in the frame buffer. 11 = Data being written to the frame buffer is logically XORed with the data in the

Bit	Description
	memory read latch before it is actually stored in the frame buffer.
2:0	Rotate Count. These bits specify the number of bits to the right to rotate any data that is meant to be written to the frame buffer just before it is actually stored in the frame buffer at the intended address location.

GR04 - Read Plane Select Register

Address: 3CFh (Index=04h)

Default: 0UUh (U=Undefined)

Attributes:

Read/Write

Bit	Description
7:2	Reserved. Read as 0.
1:0	<p>Read Plane Select. These two bits select the memory plane from which the CPU reads data in Read Mode 0. In Odd/Even Mode, bit 0 of this register is ignored. In Chain 4 Mode, both bits 1 and 0 of this register are ignored. The four memory planes are selected as follows:</p> <p>00 = Plane 0 01 = Plane 1 10 = Plane 2 11 = Plane 3</p> <p>These two bits also select which of the four memory read latches may be read via the Memory read Latch Data Register (CR22). The choice of memory read latch corresponds to the choice of plane specified in the table above. The Memory Read Latch Data register and this additional function served by 2 bits are features of the VGA standard that were never documented.</p>

GR05 - Graphics Mode Register

Address: 3CFh (Index=05h)

Default: 0UUU U0UUb (U=Undefined)

Attributes:

Read/Write

Bit	Description
7	Reserved. Read as 0.

Bit	Description																																																																																																																																					
6:5	<p>Shift Register Control. In standard VGA modes, pixel data is transferred from the 4 graphics memory planes to the palette via a set of 4 serial output bits. These 2 bits of this register control the format in which data in the 4 memory planes is serialized for these transfers to the palette.</p> <p>Bits [6:5]=00</p> <p>One bit of data at a time from parallel bytes in each of the 4 memory planes is transferred to the palette via the 4 serial output bits, with 1 of each of the serial output bits corresponding to a memory plane. This provides a 4-bit value on each transfer for 1 pixel, making possible a choice of 1 of 16 colors per pixel.</p> <p>Serial</p> <table border="0" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding-right: 10px;">Out</td> <td style="padding-right: 10px;">1st</td> <td style="padding-right: 10px;">2nd</td> <td style="padding-right: 10px;">3rd</td> <td style="padding-right: 10px;">4th</td> <td style="padding-right: 10px;">5th</td> <td style="padding-right: 10px;">6th</td> <td style="padding-right: 10px;">7th</td> <td style="padding-right: 10px;">8th</td> </tr> <tr> <td></td> <td style="text-align: center;">Xfer</td> </tr> <tr> <td style="padding-right: 10px;">Bit</td> <td style="padding-right: 10px;">plane3</td> </tr> <tr> <td style="padding-right: 10px;">3</td> <td style="text-align: center;">bit7</td> <td style="text-align: center;">bit6</td> <td style="text-align: center;">bit5</td> <td style="text-align: center;">bit4</td> <td style="text-align: center;">bit3</td> <td style="text-align: center;">bit2</td> <td style="text-align: center;">bit1</td> <td style="text-align: center;">bit0</td> </tr> <tr> <td style="padding-right: 10px;">Bit</td> <td style="padding-right: 10px;">plane2</td> </tr> <tr> <td style="padding-right: 10px;">2</td> <td style="text-align: center;">bit7</td> <td style="text-align: center;">bit6</td> <td style="text-align: center;">bit5</td> <td style="text-align: center;">bit4</td> <td style="text-align: center;">bit3</td> <td style="text-align: center;">bit2</td> <td style="text-align: center;">bit1</td> <td style="text-align: center;">bit0</td> </tr> <tr> <td style="padding-right: 10px;">Bit</td> <td style="padding-right: 10px;">plane1</td> </tr> <tr> <td style="padding-right: 10px;">1</td> <td style="text-align: center;">bit7</td> <td style="text-align: center;">bit6</td> <td style="text-align: center;">bit5</td> <td style="text-align: center;">bit4</td> <td style="text-align: center;">bit3</td> <td style="text-align: center;">bit2</td> <td style="text-align: center;">bit1</td> <td style="text-align: center;">bit0</td> </tr> <tr> <td style="padding-right: 10px;">Bit</td> <td style="padding-right: 10px;">plane0</td> </tr> <tr> <td style="padding-right: 10px;">0</td> <td style="text-align: center;">bit7</td> <td style="text-align: center;">bit6</td> <td style="text-align: center;">bit5</td> <td style="text-align: center;">bit4</td> <td style="text-align: center;">bit3</td> <td style="text-align: center;">bit2</td> <td style="text-align: center;">bit1</td> <td style="text-align: center;">bit0</td> </tr> </table> <p>Bits [6:5]=01</p> <p>Two bits of data at a time from parallel bytes in each of the 4 memory planes are transferred to the palette in a pattern that alternates per byte between memory planes 0 and 2, and memory planes 1 and 3. First the even-numbered and odd-numbered bits of a byte in memory plane 0 are transferred via serial output bits 0 and 1, respectively, while the even-numbered and odd-numbered bits of a byte in memory plane 2 are transferred via serial output bits 2 and 3. Next, the even-numbered and odd-numbered bits of a byte in memory plane 1 are transferred via serial output bits 0 and 1, respectively, while the even-numbered and odd-numbered bits of memory plane 3 are transferred via serial out bits 1 and 3. This provides a pair of 2-bit values (one 2-bit value for each of 2 pixels) on each transfer, making possible a choice of 1 of 4 colors per pixel.</p> <p>Serial</p> <table border="0" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding-right: 10px;">Out</td> <td style="padding-right: 10px;">1st</td> <td style="padding-right: 10px;">2nd</td> <td style="padding-right: 10px;">3rd</td> <td style="padding-right: 10px;">4th</td> <td style="padding-right: 10px;">5th</td> <td style="padding-right: 10px;">6th</td> <td style="padding-right: 10px;">7th</td> <td style="padding-right: 10px;">8th</td> </tr> <tr> <td></td> <td style="text-align: center;">Xfer</td> </tr> <tr> <td style="padding-right: 10px;">Bit</td> <td style="padding-right: 10px;">plane2</td> <td style="padding-right: 10px;">plane2</td> <td style="padding-right: 10px;">plane2</td> <td style="padding-right: 10px;">plane2</td> <td style="padding-right: 10px;">plane3</td> <td style="padding-right: 10px;">plane3</td> <td style="padding-right: 10px;">plane3</td> <td style="padding-right: 10px;">plane3</td> </tr> <tr> <td style="padding-right: 10px;">3</td> <td style="text-align: center;">bit7</td> <td style="text-align: center;">bit5</td> <td style="text-align: center;">bit3</td> <td style="text-align: center;">bit1</td> <td style="text-align: center;">bit7</td> <td style="text-align: center;">bit5</td> <td style="text-align: center;">bit3</td> <td style="text-align: center;">bit1</td> </tr> </table>								Out	1st	2nd	3rd	4th	5th	6th	7th	8th		Xfer	Bit	plane3	3	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Bit	plane2	2	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Bit	plane1	1	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Bit	plane0	0	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Out	1st	2nd	3rd	4th	5th	6th	7th	8th		Xfer	Bit	plane2	plane2	plane2	plane2	plane3	plane3	plane3	plane3	3	bit7	bit5	bit3	bit1	bit7	bit5	bit3	bit1																																										
Out	1st	2nd	3rd	4th	5th	6th	7th	8th																																																																																																																														
	Xfer	Xfer	Xfer	Xfer	Xfer	Xfer	Xfer	Xfer																																																																																																																														
Bit	plane3	plane3	plane3	plane3	plane3	plane3	plane3	plane3																																																																																																																														
3	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0																																																																																																																														
Bit	plane2	plane2	plane2	plane2	plane2	plane2	plane2	plane2																																																																																																																														
2	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0																																																																																																																														
Bit	plane1	plane1	plane1	plane1	plane1	plane1	plane1	plane1																																																																																																																														
1	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0																																																																																																																														
Bit	plane0	plane0	plane0	plane0	plane0	plane0	plane0	plane0																																																																																																																														
0	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0																																																																																																																														
Out	1st	2nd	3rd	4th	5th	6th	7th	8th																																																																																																																														
	Xfer	Xfer	Xfer	Xfer	Xfer	Xfer	Xfer	Xfer																																																																																																																														
Bit	plane2	plane2	plane2	plane2	plane3	plane3	plane3	plane3																																																																																																																														
3	bit7	bit5	bit3	bit1	bit7	bit5	bit3	bit1																																																																																																																														

Bit	Description																																																																																																											
Bit 2	plane2 bit6	plane2 bit4	plane2 bit2	plane2 bit0	plane3 bit6	plane3 bit4	plane3 bit2	plane3 bit0																																																																																																				
Bit 1	plane0 bit7	plane0 bit5	plane0 bit3	plane0 bit1	plane1 bit7	plane1 bit5	plane1 bit3	plane1 bit1																																																																																																				
Bit 0	plane0 bit6	plane0 bit4	plane0 bit2	plane0 bit0	plane1 bit6	plane1 bit4	plane1 bit2	plane1 bit0																																																																																																				
<p>This alternating pattern is meant to accommodate the use of the Odd/Even mode of organizing the 4 memory planes, which is used by standard VGA modes 2h and 3h.</p> <p>Bits [6:5] = 1x</p> <p>Four bits of data at a time from parallel bytes in each of the 4 memory planes are transferred to the palette in a pattern that iterates per byte through memory planes 0 through 3. First the 4 most significant bits of a byte in memory plane 0 are transferred via the 4 serial output bits, followed by the 4 least significant bits of the same byte. Next, the same transfers occur from the parallel byte in memory planes 1, 2 and lastly, 3. Each transfer provides either the upper or lower half of an 8 bit value for the color for each pixel, making possible a choice of 1 of 256 colors per pixel. This is the setting used in mode x13.</p> <p>Serial</p> <table border="0" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;"></td> <td style="width: 10%;">Out</td> <td style="width: 10%;">1st</td> <td style="width: 10%;">2nd</td> <td style="width: 10%;">3rd</td> <td style="width: 10%;">4th</td> <td style="width: 10%;">5th</td> <td style="width: 10%;">6th</td> <td style="width: 10%;">7th</td> <td style="width: 10%;">8th</td> </tr> <tr> <td></td> <td></td> <td>Xfer</td> <td>Xfer</td> <td>Xfer</td> <td>Xfer</td> <td>Xfer</td> <td>Xfer</td> <td>Xfer</td> <td>Xfer</td> </tr> </table> <table border="0" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;"></td> <td style="width: 10%;">Bit</td> <td style="width: 10%;">plane0</td> <td style="width: 10%;">plane0</td> <td style="width: 10%;">plane1</td> <td style="width: 10%;">plane1</td> <td style="width: 10%;">plane2</td> <td style="width: 10%;">plane2</td> <td style="width: 10%;">plane3</td> <td style="width: 10%;">plane3</td> </tr> <tr> <td>Bit 3</td> <td></td> <td>bit7</td> <td>bit3</td> <td>bit7</td> <td>bit3</td> <td>bit7</td> <td>bit3</td> <td>bit7</td> <td>bit3</td> </tr> </table> <table border="0" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;"></td> <td style="width: 10%;">Bit</td> <td style="width: 10%;">plane0</td> <td style="width: 10%;">plane0</td> <td style="width: 10%;">plane1</td> <td style="width: 10%;">plane1</td> <td style="width: 10%;">plane2</td> <td style="width: 10%;">plane2</td> <td style="width: 10%;">plane3</td> <td style="width: 10%;">plane3</td> </tr> <tr> <td>Bit 2</td> <td></td> <td>bit6</td> <td>bit2</td> <td>bit6</td> <td>bit2</td> <td>bit6</td> <td>bit2</td> <td>bit6</td> <td>bit2</td> </tr> </table> <table border="0" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;"></td> <td style="width: 10%;">Bit</td> <td style="width: 10%;">plane0</td> <td style="width: 10%;">plane0</td> <td style="width: 10%;">plane1</td> <td style="width: 10%;">plane1</td> <td style="width: 10%;">plane2</td> <td style="width: 10%;">plane2</td> <td style="width: 10%;">plane3</td> <td style="width: 10%;">plane3</td> </tr> <tr> <td>Bit 1</td> <td></td> <td>bit5</td> <td>bit1</td> <td>bit5</td> <td>bit1</td> <td>bit5</td> <td>bit1</td> <td>bit5</td> <td>bit1</td> </tr> </table> <table border="0" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;"></td> <td style="width: 10%;">Bit</td> <td style="width: 10%;">plane0</td> <td style="width: 10%;">plane0</td> <td style="width: 10%;">plane1</td> <td style="width: 10%;">plane1</td> <td style="width: 10%;">plane2</td> <td style="width: 10%;">plane2</td> <td style="width: 10%;">plane3</td> <td style="width: 10%;">plane3</td> </tr> <tr> <td>Bit 0</td> <td></td> <td>bit4</td> <td>bit0</td> <td>bit4</td> <td>bit0</td> <td>bit4</td> <td>bit0</td> <td>bit4</td> <td>bit0</td> </tr> </table> <p>This pattern is meant to accommodate mode 13h, a standard VGA 256-color graphics mode.</p>										Out	1st	2nd	3rd	4th	5th	6th	7th	8th			Xfer		Bit	plane0	plane0	plane1	plane1	plane2	plane2	plane3	plane3	Bit 3		bit7	bit3	bit7	bit3	bit7	bit3	bit7	bit3		Bit	plane0	plane0	plane1	plane1	plane2	plane2	plane3	plane3	Bit 2		bit6	bit2	bit6	bit2	bit6	bit2	bit6	bit2		Bit	plane0	plane0	plane1	plane1	plane2	plane2	plane3	plane3	Bit 1		bit5	bit1	bit5	bit1	bit5	bit1	bit5	bit1		Bit	plane0	plane0	plane1	plane1	plane2	plane2	plane3	plane3	Bit 0		bit4	bit0	bit4	bit0	bit4	bit0	bit4	bit0							
	Out	1st	2nd	3rd	4th	5th	6th	7th	8th																																																																																																			
		Xfer	Xfer																																																																																																									
	Bit	plane0	plane0	plane1	plane1	plane2	plane2	plane3	plane3																																																																																																			
Bit 3		bit7	bit3	bit7	bit3	bit7	bit3	bit7	bit3																																																																																																			
	Bit	plane0	plane0	plane1	plane1	plane2	plane2	plane3	plane3																																																																																																			
Bit 2		bit6	bit2	bit6	bit2	bit6	bit2	bit6	bit2																																																																																																			
	Bit	plane0	plane0	plane1	plane1	plane2	plane2	plane3	plane3																																																																																																			
Bit 1		bit5	bit1	bit5	bit1	bit5	bit1	bit5	bit1																																																																																																			
	Bit	plane0	plane0	plane1	plane1	plane2	plane2	plane3	plane3																																																																																																			
Bit 0		bit4	bit0	bit4	bit0	bit4	bit0	bit4	bit0																																																																																																			
4	<p>Odd/Even Mode.</p> <p>0 = Addresses sequentially access data within a bit map, and the choice of which map is accessed is made according to the value of the Plane Mask Register (SR02).</p> <p>1 = The frame buffer is mapped in such a way that the function of</p>																																																																																																											

Bit	Description
	<p>address bit 0 is such that even addresses select memory planes 0 and 2 and odd addresses select memory planes 1 and 3.</p> <p>This works in a way that is the inverse of (and is normally set to be the opposite of) bit 2 of the Memory Mode Register (SR02).</p>
3	<p>Read Mode.</p> <p>0 = During a CPU read from the frame buffer, the value returned to the CPU is data from the memory plane selected by bits 1 and 0 of the Read Plane Select Register (GR04).</p> <p>1 = During a CPU read from the frame buffer, all 8 bits of the byte in each of the 4 memory planes corresponding to the address from which a CPU read access is being performed are compared to the corresponding bits in this register (if the corresponding bit in the Color Don't Care Register (GR07) is set to 1). The value that the CPU receives from the read access is an 8-bit value that shows the result of this comparison. A value of 1 in a given bit position indicates that all of the corresponding bits in the bytes across all 4 of the memory planes that were included in the comparison had the same value as their memory plane's respective bits in this register.</p>
2	<p>Reserved. Read as 0.</p>
1:0	<p>Write Mode.</p> <p>00 = Write Mode 0 - During a CPU write to the frame buffer, the addressed byte in each of the 4 memory planes is written with the CPU write data after it has been rotated by the number of counts specified in the Data Rotate Register</p>

Bit	Description
	<p>(GR03). If, however, the bit(s) in the Enable Set/Reset Register (GR01) corresponding to one or more of the memory planes is set to 1, then those memory planes will be written to with the data stored in the corresponding bits in the Set/Reset Register (GR00).</p> <p>01 = Write Mode 1 - During a CPU write to the frame buffer, the addressed byte in each of the 4 memory planes is written to with the data stored in the memory read latches. (The memory read latches stores an unaltered copy of the data last read from any location in the frame buffer.)</p> <p>10 = Write Mode 2 - During a CPU write to the frame buffer, the least significant 4 data bits of the CPU write data is treated as the color value for the pixels in the addressed byte in all 4 memory planes. The 8 bits of the Bit Mask Register (GR08) are used to selectively enable or disable the ability to write to the corresponding bit in each of the 4 memory planes that correspond to a given pixel. A setting of 0 in a bit in the Bit Mask Register at a given bit position causes the bits in the corresponding bit positions in the addressed byte in all 4 memory planes to be written with value of their counterparts in the memory read latches. A setting of 1 in a Bit Mask Register at a given bit position causes the bits in the corresponding bit positions in the addressed byte in all 4 memory planes to be written with the 4 bits taken from the CPU write data to thereby cause the pixel corresponding to these bits to be set to the color value.</p> <p>11 = Write Mode 3 - During a CPU write to the frame buffer, the CPU write data is logically ANDed with the contents of the Bit Mask Register (GR08). The result of this ANDing is treated as the bit mask used in writing the contents of the Set/Reset Register (GR00) are written to addressed byte in all 4 memory planes.</p>

GR06 - Miscellaneous Register

Address: 3CFh (Index=06h)

Default: 0Uh (U=Undefined)

Attributes:

Read/Write

Bit	Description
-----	-------------

Bit	Description
7:4	Reserved. Read as 0s.
3:2	<p>Memory Map Mode. These 2 bits control the mapping of the VGA address range for frame buffer into the CPU address space as follows:</p> <p>00 = A0000h - BFFFFh 01 = A0000h - AFFFFh 10 = B0000h - B7FFFh 11 = B8000h - BFFFFh</p> <p>This function is used in standard VGA modes, extended VGA modes (132 column text), and in non-VGA modes (hi-res). 132 column text modes are no longer supported.</p> <p>VGA aperture memory accesses are also controlled by the PCI configuration Memory Enable bit and MSR<1>.</p> <p>For accesses using GR10 and GR11 to paged VGA RAM or to device MMIO registers, set these bits to 01 to select the (A0000-AFFFF) range.</p> <p>The CPU must map this memory as uncacheable (UC).</p>
1	<p>Chain Odd/Even. This bit provides the ability to alter the interpretation of address bit A0, so that it may be used in selecting between the odd-numbered memory planes (planes 1 and 3) and the even-numbered memory planes (planes 0 and 2).</p> <p>0 = A0 functions normally. 1 = A0 is switched with a high order address bit, in terms of how it is used in address decoding. The result is that A0 is used to determine which memory plane is being accessed (A0=0 for planes 0 and 2 and A0=1 for planes 1 and 3).</p>
0	<p>Graphics/Text Mode. This is one of two bits that are used to determine if the VGA is operating in text or graphics modes. The other bit is in AR10[0], these two bits need to be programmed in a consistent manner to achieve the proper results.</p> <p>0 = Text mode. 1 = Graphics mode.</p>

GR07 - Color Don't Care Register

Address: 3CFh (Index=07h)

Default: 0Uh (U=Undefined)

Attributes:

Read/Write

Bit	Description
7:4	Reserved. Read as 0.
3:0	<p>Ignore Color Plane [3:0]. These bits have effect only when bit 3 of the Graphics Mode Register (GR05) is set to 1 to select read mode 1.</p> <p>0 = The corresponding bit in the Color Compare Register (GR02) will not be included in color comparisons.</p> <p>1 = The corresponding bit in the Color Compare Register (GR02) is used in color comparisons.</p>

GR08 - Bit Mask Register

Address: 3CFh (Index=08h)

Default: Undefined

Attributes:

Read/Write

Bit	Description
7:0	<p>Bit Mask.</p> <p>0 = The corresponding bit in each of the 4 memory planes is written to with the corresponding bit in the memory read latches.</p> <p>1 = Manipulation of the corresponding bit in each of the 4 memory planes via other mechanisms is enabled.</p> <p>This bit mask applies to any writes to the addressed byte of any or all of the 4 memory planes, simultaneously.</p> <p>This bit mask is applicable to any data written into the frame buffer by the CPU, including data that is also subject to rotation, logical functions (AND, OR, XOR), and Set/Reset. To perform a proper read-modify-write cycle into frame buffer, each byte must first be read from the frame buffer by the CPU (and this will cause it to be stored in the memory read latches), this Bit Mask Register must be set, and the new data then written into the frame buffer by the CPU.</p>

GR10 - Address Mapping

Address: 3CFh (Index=10h)

Default: 00h

Attributes:

Read/Write

This register must only be accessed using I/O operations.

Bit	Description
-----	-------------

Bit	Description
7:4	<p>Page Select Extension - Unused</p> <p>These bits form the upper bits of a 12-bit page selection value. When combined with the GR11 <7:0> bits they define the offset into stolen memory to the 64KB page that is accessible via the VGA Memory paging mechanism.</p> <p>These bits are ignored.</p>
3	<p>Reserved</p>
2:1	<p>Paging Map Target.</p> <p>When paging is enabled, these bits determine the target for data cycle accesses through the VGA memory aperture.</p> <p>VGA graphics memory starts from the base of graphics data stolen memory defined in the PCI configuration BDSM register.</p> <p>VGA display uses the first four 64KB pages of VGA graphics memory.</p> <p>00 = VGA Graphics Memory 01 = Reserved 10 = Reserved 11 = Reserved</p>
0	<p>Page Mapping Enable.</p> <p>This mode allows the mapping of the VGA memory address space.</p> <p>Once this is enabled, no VGA memory address swizzle will be performed, addresses are directly mapped to memory.</p> <p>A single paging register is used to map the 64KB [A0000:AFFFF] window. An internal address is generated using GR11 as the address lines extension to the lower address lines of the access A[15:2].</p> <p>When mapping is enabled, the B0000:BFFFF area must be disabled using GR06 <3:2> = 01.</p> <p>The use of addresses in the A0000-BFFFF range require that both the graphics device PCI configuration memory enable and MSR <1> be enabled.</p> <p>0 = Disable (default) 1 = Enable</p>

GR11 - Page Selector

Address: 3CFh (Index=11h)

Default: 00h

Attributes:

Read/Write

This register must only be accessed using I/O operations.

Bit	Description
7	Reserved
6:0	<p>Page Select.</p> <p>When concatenated with the GR10<7:4> bits, selects a 64KB window within target area when Page Mapping is enabled (GR10[0]=1).</p> <p>This requires that the graphics device PCI configuration space memory enable, the GR06<3:2> bits to be 01 (select A0000-AFFFF only), and the MSR<1:1> bit to be set.</p> <p>This register provides the Address[22:16] bits for the access.</p> <p>VGA paging of frame buffer memory is for non-VGA packed modes only and should not be enabled when using basic VGA modes.</p>

GR18 - Software Flags

Address: 3CFh (Index=18h)

Default: 00h

Attributes:

Read/Write

Bit	Description
7:0	<p>Software Flags. Used as scratch pad space in video BIOS. These bits are separate from the bits which appear in the MMIO space. They are used specifically by the SMI BIOS which does not have access to MMIO at the time they are required. These register bits have no effect on H/W operation.</p>



Attribute Controller Registers

Unlike the other sets of indexed registers, the attribute controller registers are not accessed through a scheme employing entirely separate index and data ports. Address 3C0h is used both as the read and write for the index register, and as the write address for the data port. Address 3C1h is the read address for the data port.

To write to the attribute controller registers, the index of the desired register must be written to address 3C0h, and then the data is written to the very same address. A flip-flop alternates with each write to address 3C0h to change its function from writing the index to writing the actual data, and back again. This flip-flop may be deliberately set so that address 3C0h is set to write to the index (which provides a way to set it to a known state) by performing a read operation from Input Status Register 1 (ST01) at address 3BAh or 3DAh, depending on whether the graphics system has been set to emulate an MDA or a CGA as per MSR[0].

To read from the attribute controller registers, the index of the desired register must be written to address 3C0h, and then the data is read from address 3C1h. A read operation from address 3C1h does not reset the flip-flop to writing to the index. Only a write to 3C0h or a read from 3BAh or 3DAh, as described above, will toggle the flip-flop back to writing to the index.

ARX - Attribute Controller Index Register

Address: 3C0h

Default:

00UU UUUUb (U=Undefined)

Attributes:

Read/Write

Bit	Description
7:6	Reserved. Read as 0s.
5	Video Enable. In the VGA standard, this is called the "Palette Address Source" bit. Clearing this bit will cause the VGA display data to become all 00 index values. For the default palette, this will cause a black screen. The video timing signals continue. Another control bit will turn video off and stop the data fetches. 0 = Disable. Attribute controller color registers (AR[00:0F]) can be accessed by the CPU. 1 = Enable. Attribute controller color registers (AR[00:0F]) are inaccessible by the CPU.
4:0	Attribute Controller Register Index. These five bits are used to select any one of the attribute controller registers (AR[00:14]), to be accessed.

AR[00:0F] - Palette Registers [0:F]

Address: Read at 3C1h and Write at 3C0h; (index=00h-0Fh)

Default: 00UU UUUUb (U=Undefined)

Attributes:

Read/Write

Bit	Description
7:6	Reserved. Read as 0.
5:0	<p>Palette Bits P[5:0]. In each of these 16 registers, these are the lower 6 of 8 bits that are used to map either text attributes or pixel color input values (for modes that use 16 colors) to the 256 possible colors available to be selected in the palette.</p> <p>Bits 3 and 2 of the Color Select Register (AR14) supply bits P7 and P6 for the values contained in all 16 of these registers. Bits 1 and 0 of the Color Select Register (AR14) can also replace bits P5 and P4 for the values contained in all 16 of these registers, if bit 7 of the Mode Control Register (AR10) is set to 1.</p>

AR10 - Mode Control Register

Address: Read at 3C1h and Write at 3C0h; (index=10h)

Default: UUh (U=Undefined)

Attributes:

Read/Write

Bit	Description
7	<p>Palette Bits P5, P4 Select.</p> <p>0 = P5 and P4 for each of the 16 selected colors (for modes that use 16 colors) are individually provided by bits 5 and 4 of their corresponding Palette Registers (AR[00:0F]).</p> <p>1 = P5 and P4 for all 16 of the selected colors (for modes that use 16 colors) are provided by bits 1 and 0 of Color Select Register (AR14).</p>
6	<p>Pixel Width/Clock Select.</p> <p>0 = Six bits of video data (translated from 4 bits via the palette) are output every dot clock.</p> <p>1 = Two sets of 4 bits of data are assembled to generate 8 bits of video data which is output every other dot clock, and the Palette Registers (AR[00:0F]) are bypassed.</p> <p>This bit is set to 0 for all of the standard VGA modes, except mode 13h.</p>
5	<p>Pixel Panning Compatibility.</p> <p>0 = Scroll both the upper and lower screen regions horizontally as specified in the Pixel Panning Register (AR13).</p> <p>1 = Scroll only the upper screen region horizontally as specified in the Pixel Panning Register (AR13).</p> <p>This bit has application only when split-screen mode is being used, where the display area is divided into distinct upper and lower regions which function somewhat like separate displays.</p>

Bit	Description
4	<p>Reserved. Read as 0.</p>
3	<p>Enable Blinking/Select Background Intensity.</p> <p>0 = Disables blinking in graphics modes, and for text modes, sets bit 7 of the character attribute bytes to control background intensity, instead of blinking.</p> <p>1 = Enables blinking in graphics modes and for text modes, sets bit 7 of the character attribute bytes to control blinking, instead of background intensity.</p> <p>The blinking rate is derived by dividing the VSYNC signal. The Blink Rate Control field of the VGA control register defines the blinking rate.</p>
2	<p>Enable Line Graphics Character Code.</p> <p>0 = Every 9th pixel of a horizontal line (i.e., the last pixel of each horizontal line of each 9-pixel wide character box) is assigned the same attributes as the background of the character of which the given pixel is a part.</p> <p>1 = Every 9th pixel of a horizontal line (i.e., the last pixel of each horizontal line of each 9-pixel wide character box) is assigned the same attributes as the 8th pixel if the character of which the given pixel is a part. This setting is intended to accommodate the line-drawing characters of the PC's extended ASCII character set -- characters with an extended ASCII code in the range of B0h to DFh.</p> <p>In some literature describing the VGA standard, the range of extended ASCII codes that are said to include the line-drawing characters is mistakenly specified as C0h to DFh, rather than the correct range of B0h to DFh.</p>
1	<p>Select Display Type.</p> <p>0 = Attribute bytes in text modes are interpreted as they would be for a color display.</p> <p>1 = Attribute bytes in text modes are interpreted as they would be for a monochrome display.</p>
0	<p>Graphics/Alphanumeric Mode. This bit (along with GR06[0]) select either graphics mode or text mode. These two bits must be programmed in a consistent manner to achieve the desired results.</p> <p>0 = Alphanumeric (text) mode.</p> <p>1 = Graphics mode.</p>

AR11 - Overscan Color Register

Address: Read at 3C1h and Write at 3C0h; (index=11h)

Default: UUh (U=Undefined)

Attributes:

Read/Write

Bit	Description
7:0	Overscan. These 8 bits select the overscan (border) color index value. The actual border color will be determined by the contents of the palette at the selected index. The border color is displayed between the end of active and the beginning of blank or the end of blank and the beginning of active on CRT type devices driven from the DAC output port. For native VGA modes on digital display ports, some devices have the option of including the border in the active region or not, depending on a control bit in the port control register. For centered VGA modes, the VGA control register determines if the border is included in the centered region or not. For monochrome displays, this value should be set to 00h.

AR12 - Memory Plane Enable Register

Address: Read at 3C1h and Write at 3C0h; (index=12h)

Default: 00UU UUUUb (U=Undefined)

Attributes:

Read/Write

Bit	Description															
7:6	Reserved. Read as 0.															
5:4	<p>Video Status Mux. These 2 bits are used to select 2 of the 8 possible palette bits (P7-P0) to be made available to be read via bits 5 and 4 of the Input Status Register 1 (ST01). The table below shows the possible choices.</p> <table border="1"> <thead> <tr> <th>Bit [5:4]</th> <th>ST01 Bit 5</th> <th>ST01 Bit 4</th> </tr> </thead> <tbody> <tr> <td>00</td> <td>P2 (default)</td> <td>P0 (default)</td> </tr> <tr> <td>01</td> <td>P5</td> <td>P4</td> </tr> <tr> <td>10</td> <td>P3</td> <td>P1</td> </tr> <tr> <td>11</td> <td>P7</td> <td>P6</td> </tr> </tbody> </table> <p>These bits are typically unused by current software; they are provided for EGA compatibility.</p>	Bit [5:4]	ST01 Bit 5	ST01 Bit 4	00	P2 (default)	P0 (default)	01	P5	P4	10	P3	P1	11	P7	P6
Bit [5:4]	ST01 Bit 5	ST01 Bit 4														
00	P2 (default)	P0 (default)														
01	P5	P4														
10	P3	P1														
11	P7	P6														
3:0	Enable Plane [3:0]. These 4 bits individually enable the use of each of the 4 memory planes in providing 1 of the 4 bits used in video output to select 1 of 16 possible															

Bit	Description
	<p>colors from the palette to be displayed.</p> <p>0 = Disable the use of the corresponding memory plane in video output to select colors, forcing the bit that the corresponding memory plane would have provided to a value of 0.</p> <p>1 = Enable the use of the corresponding memory plane in video output to select colors.</p> <p>AR12 is referred to in the VGA standard as the Color Plane Enable Register.</p>

AR13 - Horizontal Pixel Panning Register

Address: Read at 3C1h and Write at 3C0h; (index=13h)

Default: 0Uh (U=Undefined)

Attributes:

Read/Write

Bit	Description																																													
7:4	Reserved.																																													
3:0	<p>Horizontal Pixel Shift 3-0. This field holds a 4-bit value that selects the number of pixels by which the image is shifted horizontally to the left. This function is available in both text and graphics modes and allows for pixel panning.</p> <p>In text modes with a 9-pixel wide character box, the image can be shifted up to 9 pixels to the left. In text modes with an 8-pixel wide character box, and in graphics modes other than those with 256 colors, the image can be shifted up to 8 pixels to the left. A pseudo 9-bit mode is when the 9-dot character is selected but overridden by the VGA control bit.</p> <p>In standard VGA mode 13h (where bit 6 of the Mode Control Register, AR10, is set to 1 to support 256 colors), bit 0 of this register must remain set to 0, and the image may be shifted up to only 4 pixels to the left. In this mode, the number of pixels by which the image is shifted can be further controlled using bits 6 and 5 of the Preset Row Scan Register (CR08).</p> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <thead> <tr> <th colspan="5" style="text-align: center; background-color: #e1eef6;">Number of Pixels Shifted</th> </tr> <tr> <th style="text-align: center;">Bits [3:0]</th> <th style="text-align: center;">9-dot</th> <th style="text-align: center;">Pseudo 9-dot</th> <th style="text-align: center;">8-dot</th> <th style="text-align: center;">256-Color</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">1</td> <td style="text-align: center;">1</td> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> </tr> <tr> <td style="text-align: center;">1</td> <td style="text-align: center;">2</td> <td style="text-align: center;">2</td> <td style="text-align: center;">1</td> <td style="text-align: center;">Undefined</td> </tr> <tr> <td style="text-align: center;">2</td> <td style="text-align: center;">3</td> <td style="text-align: center;">3</td> <td style="text-align: center;">2</td> <td style="text-align: center;">1</td> </tr> <tr> <td style="text-align: center;">3</td> <td style="text-align: center;">4</td> <td style="text-align: center;">4</td> <td style="text-align: center;">3</td> <td style="text-align: center;">Undefined</td> </tr> <tr> <td style="text-align: center;">4</td> <td style="text-align: center;">5</td> <td style="text-align: center;">5</td> <td style="text-align: center;">4</td> <td style="text-align: center;">2</td> </tr> <tr> <td style="text-align: center;">5</td> <td style="text-align: center;">6</td> <td style="text-align: center;">6</td> <td style="text-align: center;">5</td> <td style="text-align: center;">Undefined</td> </tr> <tr> <td style="text-align: center;">6</td> <td style="text-align: center;">7</td> <td style="text-align: center;">7</td> <td style="text-align: center;">6</td> <td style="text-align: center;">3</td> </tr> </tbody> </table>	Number of Pixels Shifted					Bits [3:0]	9-dot	Pseudo 9-dot	8-dot	256-Color	0	1	1	0	0	1	2	2	1	Undefined	2	3	3	2	1	3	4	4	3	Undefined	4	5	5	4	2	5	6	6	5	Undefined	6	7	7	6	3
Number of Pixels Shifted																																														
Bits [3:0]	9-dot	Pseudo 9-dot	8-dot	256-Color																																										
0	1	1	0	0																																										
1	2	2	1	Undefined																																										
2	3	3	2	1																																										
3	4	4	3	Undefined																																										
4	5	5	4	2																																										
5	6	6	5	Undefined																																										
6	7	7	6	3																																										

Bit	Description				
	7	8	7	7	Undefined
	8	0	0	Undefined	Undefined

AR14 - Color Select Register

Address: Read at 3C1h and Write at 3C0h; (index=14h)

Default: 0Uh (U=Undefined)

Attributes:

Read/Write

Bit	Description
7:4	Reserved.
3:2	Palette Bits P[7:6]. These are the 2 upper-most of the 8 bits that are used to map either text attributes or pixel color input values (for modes that use 16 colors) to the 256 possible colors contained in the palette. These 2 bits are common to all 16 sets of bits P5 through P0 that are individually supplied by Palette Registers 0-F (AR[00:0F]).
1:0	Alternate Palette Bits P[5:4]. These 2 bits can be used as an alternate version of palette bits P5 and P4. Unlike the P5 and P4 bits that are individually supplied by Palette Registers 0-F (AR[00:0F]), these 2 alternate palette bits are common to all 16 of Palette Registers. Bit 7 of the Mode Control Register (AR10) is used to select between the use of either the P5 and P4 bits that are individually supplied by the 16 Palette Registers or these 2 alternate palette bits.

VGA Color Palette Registers

In devices that have multiple display pipes, there is one palette for each display pipe. These palettes are the same for VGA modes and non-VGA modes. Accesses through VGA register methods will read or write from the palette of the pipe selected through MMIO VGA control register.

For each palette, the color data stored in these 256 color data positions can be accessed only through a complex sub-addressing scheme, using a data register and two index registers. The Palette Data Register at address 3C9h is the data port. The Palette Read Index Register at address 3C7h and the Palette Write Index Register at address 3C8h are the two index registers. The Palette Read Index Register is the index register that is used to choose the color data position that is to be read from via the data port, while the Palette Write Index Register is the index register that is used to choose the color data position that is to be written to through the same data port. This arrangement allows the same data port to be used for reading from and writing to two different color data positions. Reading and writing the color data at a color data position involves three successive reads or writes since the color data stored at each color data position consists of three bytes.

To read a palette color data position, the index of the desired color data position must first be written to the Palette Read Index Register. Then all three bytes of data in a given color data position may be read at the Palette Data Register. The first byte read from the Palette Data Register retrieves the 8-bit value specifying the intensity of the red color component. The second and third bytes read are the corresponding 8-bit values for the green and blue color components respectively. After completing the third read operation, the Palette Read Index Register is automatically incremented so that the data of the next color data position becomes accessible for being read. This allows the contents of all of the 256 color data positions of the palette to be read in sequence. This is done by specifying only the index of the 0th color data position in the Palette Read Index Register, and then simply performing 768 successive reads from the Palette Data Register.

Writing a color data position, entails a very similar procedure. The index of the desired color data position must first be written to the Palette Write Index Register. Then all three bytes of data to specify a given color may be written to the Palette Data Register. The first byte written to the Palette Data Register specifies the intensity of the red color component, the second byte specifies the intensity for the green color component, and the third byte specifies the same for the blue color component. One important detail is that all three of these bytes must be written before the hardware will actually update these three values in the given color data position. When all three bytes have been written, the Palette Write Index Register is automatically incremented so that the data of the next color data position becomes accessible for being written. This allows the contents of all of the 256 color data positions of the palette to be written in sequence. This is done by specifying only the index of the 0th color data position in the Palette Write Index Register, and then simply performing 768 successive writes to the Palette Data Register.

DACMASK - Pixel Data Mask Register

Address: 3C6h

Default:

Undefined

Attributes:

Read/Write

Bit	Description
7:0	<p>Pixel Data Mask. In indexed-color mode, the 8 bits of this register are logically ANDed with the 8 bits of pixel data received from the frame buffer for each pixel. The result of this ANDing process becomes the actual index used to select color data positions within the palette. This has the effect of limiting the choice of color data positions that may be specified by the incoming 8-bit data.</p> <p>0 = Corresponding bit in the resulting 8-bit index being forced to 0.</p> <p>1 = Allows the corresponding bit in the resulting index to reflect the actual value of the corresponding bit in the incoming 8-bit pixel data.</p>

DACSTATE - DAC State Register

Address: 3C7h

Default:

00h

Attributes:

Read Only

Bit	Description
7:2	Reserved. Read as 0.
1:0	<p>DACState. This field indicates which of the two index registers was most recently written.</p> <p>Bits [1:0] Index Register Indicated</p> <p>00 = Palette Write Index Register at Address 3C7h (default)</p> <p>01 = Reserved</p> <p>10 = Reserved</p> <p>11 = Palette Read Index Register at Address 3C8h</p>



DACRX - Palette Read Index Register

Address: 3C7h

Default:

00h

Attributes:

Write Only

Bit	Description
7:0	Palette Read Index. The 8-bit index value programmed into this register chooses which of 256 standard color data positions within the palette are to be made accessible for being read from via the Palette Data Register (DACDATA). The index value held in this register is automatically incremented when all three bytes of the color data position selected by the current index have been read. A write to this register will abort an uncompleted palette write sequence. This register allows access to the palette even when running non-VGA display modes.

DACWX - Palette Write Index Register

Address: 3C8h

Default:

00h

Attributes:

Write Only

Bit	Description
7:0	Palette Write Index. The 8-bit index value programmed into this register chooses which of 256 standard color data positions within the palette are to be made accessible for being written via the Palette Data Register (DACDATA). The index value held in this register is automatically incremented when all three bytes of the color data position selected by the current index have been written. This register allows access to the palette even when running non-VGA display modes.

DACDATA - Palette Data Register

Address: 3C9h

Default:

Undefined

Attributes:

Read/Write

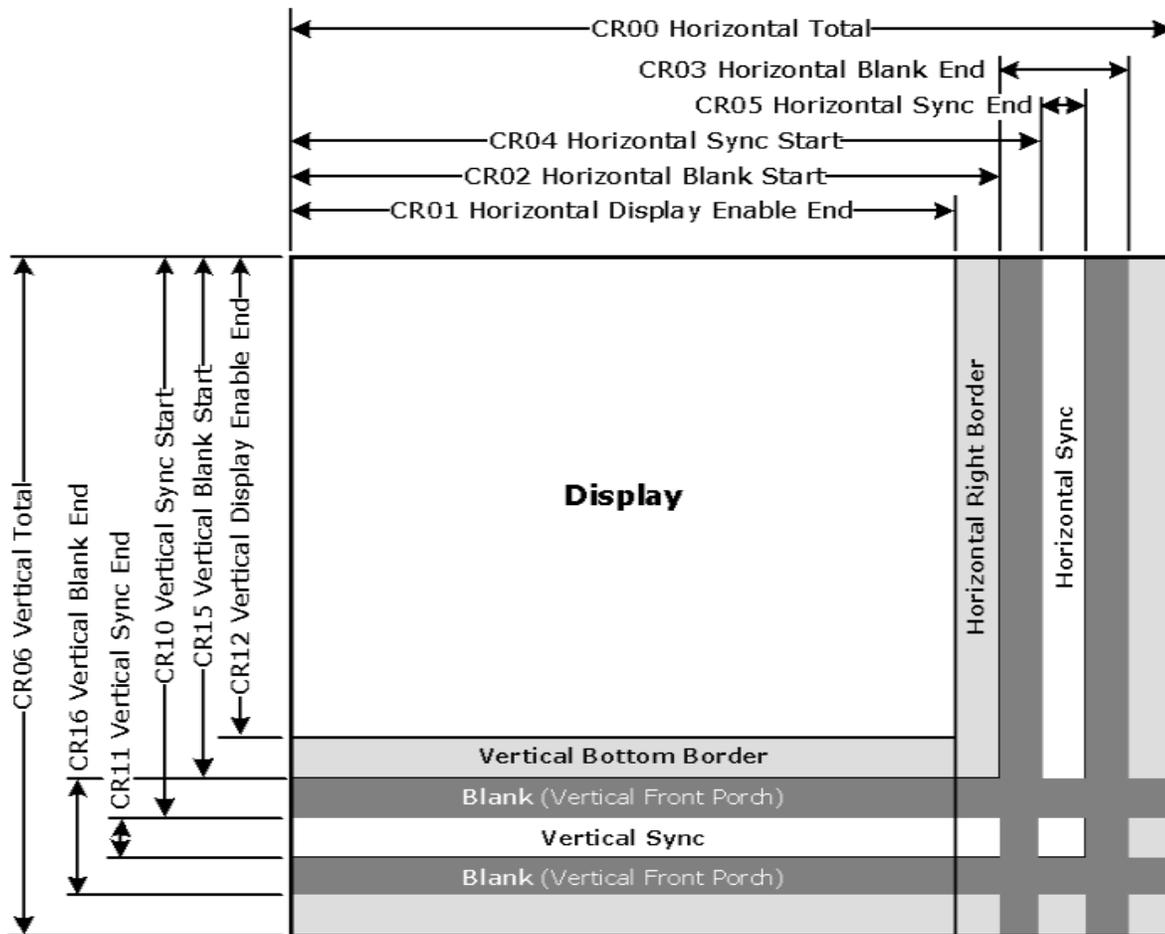
Bit	Description
7:0	<p>Palette Data. This byte-wide data port provides read or write access to the three bytes of data of each color data position selected using the Palette Read Index Register (DACRX) or the Palette Write Index Register (DACWX).</p> <p>The three bytes in each color data position are read or written in three successive read or write operations. The first byte read or written specifies the intensity of the red component of the color specified in the selected color data position. The second byte is for the green component, and the third byte is for the blue component. When writing data to a color data position, all three bytes must be written before the hardware will actually update the three bytes of the selected color data position.</p> <p>When reading or writing to a color data position, ensure that neither the Palette Read Index Register (DACRX) or the Palette Write Index Register (DACWX) are written to before all three bytes are read or written. A write to either of these two registers causes the circuitry that automatically cycles through providing access to the bytes for red, green and blue components to be reset such that the byte for the red component is the one that will be accessed by the next read or write operation via this register. This register allows access to the palette even when running non-VGA display modes. Writes to the palette can cause sparkle if not done during inactive video periods. This sparkle is caused by an attempt to write and read the same address on the same cycle. Some devices contain anti-sparkle circuits which will substitute the previous pixel value for the read output.</p>

CRT Controller Register

For native VGA modes, the CRTC registers determine the display timing that is to be used. In centered VGA modes, these registers determine the size of the VGA image that is to be centered in the larger timing generator defined rectangle.

The CRT controller registers are accessed by writing the index of the desired register into the CRT Controller Index Register at address 3B4h or 3D4h, depending on whether the graphics system is configured for MDA or CGA emulation. The desired register is then accessed through the data port for the CRT controller registers located at address 3B5h or 3D5h, again depending upon the choice of MDA or CGA emulation as per MSR[0].

The following figure shows display fields and dimensions and the particular CRxx register that provides the control.



B6781-01

Group 0 Protection: In the original VGA, CR[0:7] could be made write-protected by CR11[7]. In BIOS code, this write protection is set following each mode change. Other protection groups have no current use, and would not be used going forward by the BIOS or by drivers. They are the result of an industry fad some years ago to attempt to write protect other groups of registers; however, all such schemes were chip specific. Only the write protection (Group 0 Protection) is supported.

CRX - CRT Controller Index Register

Address: 3B4h/3D4h

Default:

0Uh (U=Undefined)

Attributes:

Read/Write

Bit	Description
7	Reserved. Read as 0.
6:0	CRT Controller Index. These 7 bits are used to select any one of the CRT controller registers to be accessed via the data port at location 3B5h or 3D5h, depending upon whether the graphics system is configured for MDA or CGA emulation.

CR00 - Horizontal Total Register

Address: 3B5h/3D5h (index=00h)

Default: 00h

Attributes:

Read/Write (Group 0 Protection)

Bit	Description
7:0	Horizontal Total. This register is used to specify the total length of each scan line. This encompasses both the part of the scan line that is within the active display area and the part that is outside of it. Programming this register to a zero has the effect of stopping the fetching of display data. This field should be programmed with a value equal to the total number of character clocks within the entire length of a scan line, minus 5.

CR01 - Horizontal Display Enable End Register

Address: 3B5h/3D5h (index=01h)

Default: Undefined

Attributes:

Read/Write (Group 0 Protection)

Bit	Description
7:0	Horizontal Display Enable End. This register is used to specify the end of the part of the scan line that is within the active display area relative to its beginning. In other words, this is the horizontal width of the active display area. This field should be programmed with a value equal to the number of character clocks that occur within the horizontal active display area, minus 1. Horizontal display

Bit	Description
	enable will go active at the beginning of each line during vertical active area, it will go inactive based on the programming of this register or the programming of the horizontal total (CR00) register. When this register value is programmed to a number that is larger than the total number of characters on a line, display enable will be active for all but the last character of the horizontal display line.

CR02 - Horizontal Blanking Start Register

Address: 3B5h/3D5h (index=02h)

Default: Undefined

Attributes:

Read/Write (Group 0 Protection)

Bit	Description
7:0	<p>Horizontal Blanking Start. This register is used to specify the beginning of the horizontal blanking period relative to the beginning of the active display area of a scan line. Horizontal blanking should always be set to start no sooner than after the end of horizontal active.</p> <p>This field should be programmed with a value equal to the number of character clocks that occur on a scan line from the beginning of the active display area to the beginning of the horizontal blanking.</p>

CR03 - Horizontal Blanking End Register

Address: 3B5h/3D5h (index=03h)

Default: 1UUU UUUUb (U=Undefined)

Attributes:

Read/Write (Group 0 Protection)

Bit	Description
7	<p>Reserved. Values written to this bit are ignored, and to maintain consistency with the VGA standard, a value of 1 is returned when this bit is read. At one time, this bit was used to enable access to certain light pen registers. At that time, setting this bit to 0 provided this access, but setting this bit to 1 was necessary for normal operation.</p>
6:5	<p>Display Enable Skew Control. Defines the degree to which the start and end of the active display area are delayed along the length of a scan line to compensate for internal pipeline delays. These 2 bits describe the delay in terms of a number character clocks.</p> <p>Bit [6:5] Amount of Delay</p> <p>00 = no delay</p>

Bit	Description
	01 = delayed by 1 character clock 10 = delayed by 2 character clocks 11 = delayed by 3 character clocks
4:0	<p>Horizontal Blanking End Bits [4:0]. This field provides the 5 least significant bits of a 6-bit value that specifies the end of the blanking period relative to its beginning on a single scan line. Bit 7 of the Horizontal Sync End Register (CR05) supplies the most significant bit.</p> <p>This 6-bit value should be programmed to be equal to the least significant 6 bits of the result of adding the length of the blanking period in terms of character clocks to the value specified in the Horizontal Blanking Start Register (CR02). End of blanking should occur before horizontal total.</p>

CR04 - Horizontal Sync Start Register

Address: 3B5h/3D5h (index=04h)

Default: Undefined

Attributes:

Read/Write (Group 0 Protection)

Bit	Description
7:0	<p>Horizontal Sync Start This register is used to specify the position of the beginning of the horizontal sync pulse relative to the start of the active display area on a scan line.</p> <p>This field should be set equal to the number of character clocks that occur from beginning of the active display area to the beginning of the horizontal sync pulse on a single scan line. Horizontal sync should always occur at least 2 clocks after the start of horizontal blank and 2 clocks before the end of horizontal blank. The actual start of sync will also be affected by both the horizontal sync skew register field and whether it is a text or graphics mode.</p>

CR05 - Horizontal Sync End Register

Address: 3B5h/3D5h (index=05h)

Default: 00h

Attributes:

Read/Write (Group 0 Protection)

Bit	Description
7	<p>Horizontal Blanking End Bit 5. This bit provides the most significant bit of a 6-bit value that specifies the end of the horizontal blanking period relative to its beginning. Bits [4:0] of Horizontal Blanking End Register (CR03) supplies the 5 least significant bits.</p>

Bit	Description										
	See CR03[4:0] for further details. This 6-bit value should be set to the least significant 6 bits of the result of adding the length of the blanking period in terms of character clocks to the value specified in the Horizontal Blanking Start Register (CR02).										
6:5	<p>Horizontal Sync Delay. This field defines the degree to which the start and end of the horizontal sync pulse are delayed to compensate for internal pipeline delays. This capability is supplied to implement VGA compatibility. These field describes the delay in terms of a number character clocks.</p> <table border="0"> <thead> <tr> <th>Bit [6:5]</th> <th>Amount of Delay</th> </tr> </thead> <tbody> <tr> <td>00</td> <td>no delay</td> </tr> <tr> <td>01</td> <td>delayed by 1 character clock</td> </tr> <tr> <td>10</td> <td>delayed by 2 character clocks</td> </tr> <tr> <td>11</td> <td>delayed by 3 character clocks</td> </tr> </tbody> </table>	Bit [6:5]	Amount of Delay	00	no delay	01	delayed by 1 character clock	10	delayed by 2 character clocks	11	delayed by 3 character clocks
Bit [6:5]	Amount of Delay										
00	no delay										
01	delayed by 1 character clock										
10	delayed by 2 character clocks										
11	delayed by 3 character clocks										
4:0	<p>Horizontal Sync End. This field provides the 5 least significant bits of a 5-bit value that specifies the end of the horizontal sync pulse relative to its beginning. A value equal to the 5 least significant bits of the horizontal character counter value at which time the horizontal retrace signal becomes inactive (logical 0). Thus, this 5-bit value specifies the width of the horizontal sync pulse. To obtain a retrace signal of W, the following algorithm is used: Value of Horizontal Sync start Register (CR04) + width of horizontal retrace signal in character clock units = 5 bit result to be programmed in this field</p>										

CR06 - Vertical Total Register

Address: 3B5h/3D5h (index=06h)

Default: 00h

Attributes:

Read/Write (Group 0 Protection)

Bit	Description
7:0	<p>Vertical Total Bits [7:0]. This field provides the 8 least significant bits of either a 10-bit or 12-bit value that specifies the total number of scan lines. This includes the scan lines both inside and outside of the active display area.</p> <p>In standard VGA modes, the vertical total is specified with a 10-bit value. The 8 least significant bits of this value are supplied by these 8 bits of this register, and the 2 most significant bits are supplied by bits 5 and 0 of the Overflow Register (CR07).</p>

CR07 - Overflow Register (Vertical)

Address: 3B5h/3D5h (index=07h)

Default: UU0U UUU0b (U=Undefined)

Attributes:

Read/Write (Group 0 Protection on bits [7:5, 3:0])

Bit	Description
7	<p>Vertical Sync Start Bit 9. The vertical sync start is a 10-bit that specifies the beginning of the vertical sync pulse relative to the beginning of the active display area. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Sync Start Register (CR10), and the most and second-most significant bits are supplied by this bit and bit 2, respectively, of this register. This 10-bit value should be programmed to be equal to the number of scan lines from the beginning of the active display area to the start of the vertical sync pulse. Since the active display area always starts on the 0th scan line, this number should be equal to the number of the scan line on which the vertical sync pulse begins.</p>
6	<p>Vertical Display Enable End Bit 9. The vertical display enable end is a 10-bit that specifies the number of the last scan line within the active display area. In standard VGA modes, the vertical display enable end is specified with a 10-bit value. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Display Enable End Register (CR12), and the most and second-most significant bits are supplied by this bit and bit 1, respectively, of this register. This 10-bit value should be programmed to be equal to the number of the last scan line within in the active display area. Since the active display area always starts on the 0th scan line, this number should be equal to the total number of scan lines within the active display area, minus 1.</p>
5	<p>Vertical Total Bit 9. The vertical total is a 10-bit value that specifies the total number of scan lines. This includes the scan lines both inside and outside of the active display area. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Total Register (CR06), and the most and second-most significant bits are supplied by this bit and bit 0, respectively, of this register.</p> <p>This 10-bit value should be programmed equal to the total number of scan lines, minus 2.</p>
4	<p>Line Compare Bit 8. This bit provides the second most significant bit of a 10-bit value that specifies the scan line at which the memory address counter restarts at the value of 0. Bit 6 of the Maximum Scan Line Register (CR09) supplies the most significant bit, and bits 7-0 of the Line Compare Register (CR18) supply the 8 least significant bits. Normally, this 10-bit value is set to specify a scan line after the last scan line of the active display area. When this 10-bit value is set to specify a scan line within the active display area, it causes that scan line and all subsequent scan lines in the active display area to display video data starting at the very first byte of the frame buffer. The result is what appears to be a screen split into a top and</p>

Bit	Description
	<p>bottom part, with the image in the top part being repeated in the bottom part. When used in cooperation with the Start Address High Register (CR0C) and the Start Address Low Register (CR0D), it is possible to create a split display, as described earlier, but with the top and bottom parts displaying different data. The top part will display what data exists in the frame buffer starting at the address specified in the two aforementioned start address registers, while the bottom part will display what data exists in the frame buffer starting at the first byte of the frame buffer.</p>
3	<p>Vertical Blanking Start Bit 8. The vertical blanking start is a 10-bit that specifies the beginning of the vertical blanking period relative to the beginning of the active display area. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Blanking Start Register (CR15), and the most and second-most significant bits are supplied by bit 5 of the Maximum Scan Line Register (CR09) and this bit of this register, respectively.</p> <p>This 10-bit value should be programmed to be equal to the number of scan lines from the beginning of the active display area to the beginning of the blanking period. Since the active display area always starts on the 0th scan line, this number should be equal to the number of the scan line on which the vertical blanking period begins.</p>
2	<p>Vertical Sync Start Bit 8. The vertical sync start is a 10-bit value that specifies the beginning of the vertical sync pulse relative to the beginning of the active display area. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Sync Start Register (CR10), and the most and second-most significant bits are supplied by bit 7 and this bit, respectively, of this register.</p> <p>This 10-bit value should be programmed to be equal to the number of scan lines from the beginning of the active display area to the start of the vertical sync pulse. Since the active display area always starts on the 0th scan line, this number should be equal to the number of the scan line on which the vertical sync pulse begins.</p>
1	<p>Vertical Display Enable End Bit 8. The vertical display enable end is a 10-bit value that specifies the number of the last scan line within the active display area. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Display Enable End Register (CR12), and the two most significant bits are supplied by bit 6 and this bit, respectively, of this register.</p> <p>This 10-bit or value should be programmed to be equal to the number of the last scan line within in the active display area. Since the active display area always starts on the 0th scan line, this number should be equal to the total number of scan lines within the active display area, minus 1.</p>
0	<p>Vertical Total Bit 8. The vertical total is a 10-bit value that specifies the total number of scan lines. This includes the scan lines both inside and outside of the active display area. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Total Register (CR06), and the most and second-most significant bits</p>

Bit	Description
	are supplied by bit 5 and this bit, respectively, of this register. This 10-bit value should be programmed to be equal to the total number of scan lines, minus 2.

CR08 - Preset Row Scan Register

Address: 3B5h/3D5h (index=08h)

Default: 0UUU UUUUb (U=Undefined)

Attributes:

Read/Write

Bit	Description																		
7	Reserved. Read as 0s.																		
6:5	<p>Byte Panning. This field holds a 2-bit value that selects number of bytes (up to 3) by which the image is shifted horizontally to the left on the screen. This function is available in both text and graphics modes.</p> <p>In text modes with a 9-pixel wide character box, the image can be shifted up to 27 pixels to the left, in increments of 9 pixels. In text modes with an 8-pixel wide character box, and in all standard VGA graphics modes, the image can be shifted up to 24 pixels to the left, in increments of 8 pixels. When the Nine dot disable bit of the VGA control register is set, the pixel shift will be equivalent to the 8-dot mode.</p> <p>The image can be shifted still further, in increments of individual pixels, through the use of bits [3:0] of the Horizontal Pixel Panning Register (AR13).</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th colspan="3" style="text-align: center;">Number of Pixels Shifted</th> </tr> <tr> <th style="text-align: center;">Bit [6:5]</th> <th style="text-align: center;">9-Pixel Text</th> <th style="text-align: center;">8-Pixel Text & Graphics</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">00</td> <td style="text-align: center;">0</td> <td style="text-align: center;">0</td> </tr> <tr> <td style="text-align: center;">01</td> <td style="text-align: center;">9</td> <td style="text-align: center;">8</td> </tr> <tr> <td style="text-align: center;">10</td> <td style="text-align: center;">18</td> <td style="text-align: center;">16</td> </tr> <tr> <td style="text-align: center;">11</td> <td style="text-align: center;">27</td> <td style="text-align: center;">24</td> </tr> </tbody> </table>	Number of Pixels Shifted			Bit [6:5]	9-Pixel Text	8-Pixel Text & Graphics	00	0	0	01	9	8	10	18	16	11	27	24
Number of Pixels Shifted																			
Bit [6:5]	9-Pixel Text	8-Pixel Text & Graphics																	
00	0	0																	
01	9	8																	
10	18	16																	
11	27	24																	
4:0	<p>Starting Row Scan Count. This field specifies which horizontal line of pixels within the character boxes of the characters used on the top-most row of text on the display will be used as the top-most scan line. The horizontal lines of pixels of a character box are numbered from top to bottom, with the top-most line of pixels being number 0. If a horizontal line of these character boxes other than the top-most line is specified, then the horizontal lines of the character box above the specified line of the character box will not be displayed as part of the top-most row of text characters on the display. Normally, the value specified by these 5 bits should be 0, so that all of the horizontal lines of</p>																		

Bit	Description
	pixels within these character boxes will be displayed in the top-most row of text, ensuring that the characters in the top-most row of text do not look as though they have been cut off at the top.

CR09 - Maximum Scan Line Register

Address: 3B5h/3D5h (index=09h)

Default: 00h

Attributes:

Read/Write

Bit	Description
7	<p>Double Scanning Enable.</p> <p>0 = Disable. When disabled, the clock to the row scan counter is equal to the horizontal scan rate. This is the normal setting for many of the standard VGA modes.</p> <p>1 = Enable. When enabled, the clock to the row scan counter is divided by 2. This is normally used to allow CGA-compatible modes that have only 200 scan lines of active video data to be displayed as 400 scan lines (each scan line is displayed twice).</p>
6	<p>Line Compare Bit 9. This bit provides the most significant bit of a 10-bit value that specifies the scan line at which the memory address counter restarts at the value of 0. Bit 4 of the Overflow Register (CR07) supplies the second most significant bit, and bits 7-0 of the Line Compare Register (CR18) supply the 8 least significant bits.</p> <p>Normally, this 10-bit value is set to specify a scan line after the last scan line of the active display area. When this 10-bit value is set to specify a scan line within the active display area, it causes that scan line and all subsequent scan lines in the active display area to display video data starting at the very first byte of the frame buffer. The result is what appears to be a screen split into a top and bottom part, with the image in the top part being repeated in the bottom part.</p> <p>When used in cooperation with the Start Address High Register (CR0C) and the Start Address Low Register (CR0D), it is possible to create a split display, as described earlier, but with the top and bottom parts displaying different data. The top part will display whatever data exists in the frame buffer starting at the address specified in the two aforementioned start address registers, while the bottom part will display whatever data exists in the frame buffer starting at the first byte of the frame buffer.</p>
5	<p>Vertical Blanking Start Bit 9. The vertical blanking start is a 10-bit value that specifies the beginning of the vertical blanking period relative to the beginning of the active display area. The 8 least significant bits of this value are supplied by bits [7:0] of the Vertical Blanking Start Register (CR15), and the most and second-most significant bits are supplied by this bit and bit 3 of the Overflow Register (CR07), respectively.</p> <p>This 10-bit value should be programmed to be equal to the number of scan line from</p>

Bit	Description
	the beginning of the active display area to the beginning of the blanking period. Since the active display area always starts on the 0th scan line, this number should be equal to the number of the scan line on which the vertical blanking period begins.
4:0	Starting Row Scan Count. This field provides all 5 bits of a 5-bit value that specifies the number of scan lines in a horizontal row of text. This value should be programmed to be equal to the number of scan lines in a horizontal row of text, minus 1.

CR0A - Text Cursor Start Register

Address: 3B5h/3D5h (index=0Ah)

Default: 00UU UUUUb (U=Undefined)

Attributes:

Read/Write

Bit	Description
7:6	Reserved. Read as 0.
5	Text Cursor Off. This text cursor exists only in text modes, so this register is entirely ignored in graphics modes. 0 = Enables the text cursor. 1 = Disables the text cursor.
4:0	Text Cursor Start. This field specifies which horizontal line of pixels in a character box is to be used to display the first horizontal line of the cursor in text mode. The horizontal lines of pixels in a character box are numbered from top to bottom, with the top-most line being number 0. The value specified by these 5 bits should be the number of the first horizontal line of pixels on which the cursor is to be shown.

CR0B - Text Cursor End Register

Address: 3B5h/3D5h (index=0Bh)

Default: 0UUU UUUUb (U=Undefined)

Attributes:

Read/Write

Bit	Description
7	Reserved. Read as 0.
6:5	Text Cursor Skew. This field specifies the degree to which the start and end of each horizontal line of pixels making up the cursor is delayed to compensate for internal

Bit	Description
	<p>pipeline delays. These 2 bits describe the delay in terms of a number character clocks.</p> <p>Bit [6:5] Amount of Delay</p> <p>00 = No delay</p> <p>01 = Delayed by 1 character clock</p> <p>10 = Delayed by 2 character clocks</p> <p>11 = Delayed by 3 character clocks</p>
4:0	<p>Text Cursor End. This field specifies which horizontal line of pixels in a character box is to be used to display the last horizontal line of the cursor in text mode. The horizontal lines of pixels in a character box are numbered from top to bottom, with the top-most line being number 0. The value specified by these 5 bits should be the number of the last horizontal line of pixels on which the cursor is to be shown.</p>

CR0C - Start Address High Register

Address: 3B5h/3D5h (index=0Ch)

Default: Undefined

Attributes:

Read/Write

Bit	Description
7:0	<p>Start Address Bits [15:8].This register provides either bits 15 through 8 of a 16-bit value that specifies the memory address offset from the beginning of the frame buffer at which the data to be shown in the active display area begins. (default is 0)</p> <p>In standard VGA modes, the start address is specified with a 16-bit value. The eight bits of this register provide the eight most significant bits of this value, while the eight bits of the Start Address Low Register (CR0D) provide the eight least significant bits.</p>

CR0D - Start Address Low Register

Address: 3B5h/3D5h (index=0Dh)

Default: Undefined

Attributes:

Read/Write

Bit	Description
7:0	<p>Start Address Bits [7:0] This register provides either bits 7 through 0 of a 16 bit value that specifies the memory address offset from the beginning of the frame buffer at</p>

Bit	Description
	<p>which the data to be shown in the active display area begins. (default is 0)</p> <p>In standard VGA modes the start address is specified with a 16-bit value. The eight bits of the Start Address High Register (CR0C) provide the eight most significant bits of this value, while the eight bits of this register provide the eight least significant bits.</p>

CR0E - Text Cursor Location High Register

Address: 3B5h/3D5h (index=0Eh)

Default: Undefined

Attributes:

Read/Write

Bit	Description
7:0	<p>Text Cursor Location Bits [15:8]. This field provides the 8 most significant bits of a 16-bit value that specifies the address offset from the beginning of the frame buffer at which the text cursor is located. Bit 7:0 of the Text Cursor Location Low Register (CR0F) provide the 8 least significant bits.</p>

CR0F - Text Cursor Location Low Register

Address: 3B5h/3D5h (index=0Fh)

Default: Undefined

Attributes:

Read/Write

Bit	Description
7:0	<p>Text Cursor Location Bits [7:0]. This field provides the 8 least significant bits of a 16-bit value that specifies the address offset from the beginning of the frame buffer at which the text cursor is located. Bits 7:0 of the Text Cursor Location High Register (CR0E) provide the 8 most significant bits.</p>

CR10 - Vertical Sync Start Register

Address: 3B5h/3D5h (index=10h)

Default: Undefined

Attributes:

Read/Write

Bit	Description
7:0	<p>Vertical Sync Start Bits [7:0]. This register provides the 8 least significant bits of a 10-bit that specifies the beginning of the vertical sync pulse relative to the</p>

Bit	Description
	<p>beginning of the active display area of a screen. In standard VGA modes, this value is described in 10 bits with bits [7,2] of the Overflow Register (CR07) supplying the 2 most significant bits.</p> <p>This 10-bit value should equal the vertical sync start in terms of the number of scan lines from the beginning of the active display area to the beginning of the vertical sync pulse. Since the active display area always starts on the 0th scan line, this number should be equal to the number of the scan line on which the vertical sync pulse begins.</p>

CR11 - Vertical Sync End Register

Address: 3B5h/3D5h (index=11h)

Default: 0U00 UUUU**b** (U=Undefined)

Attributes:

Read/Write

Bit	Description
7	<p>Protect Registers [0:7]. The ability to write to Bit 4 of the Overflow Register (CR07) is not affected by this bit (i.e., bit 4 of the Overflow Register is always writeable).</p> <p>0 = Enable writes to registers CR[00:07]. (default)</p> <p>1 = Disable writes to registers CR[00:07].</p>
6	<p>Reserved. In the VGA standard, this bit was used to switch between 3 and 5 frame buffer refresh cycles during the time required to draw each horizontal line.</p>
5	<p>Vertical Interrupt Enable. This bit is reserved for compatibility only. While this bit may be written or read, it's value will have no effect. VGA does not provide an interrupt signal which would be connected to an input of the system's interrupt controller. Bit 7 of Input Status Register 0 (ST00) originally indicated the status of the vertical retrace interrupt.</p> <p>0 = Enable the generation of an interrupt at the beginning of each vertical retrace period.</p> <p>1 = Disable the generation of an interrupt at the beginning of each vertical retrace period.</p>
4	<p>Vertical Interrupt Clear. This is reserved for compatibility only. VGA does not provide an interrupt signal which would be connected to an input of the system's interrupt controller.</p> <p>0 = Setting this bit to 0 clears a pending vertical retrace interrupt. This bit must be set back to 1 to enable the generation of another vertical retrace interrupt.</p>

Bit	Description
3:0	Vertical Sync End. This 4-bit field provides a 4-bit value that specifies the end of the vertical sync pulse relative to its beginning. This 4-bit value should be set to the least significant 4 bits of the result of adding the length of the vertical sync pulse in terms of the number of scan lines that occur within the length of the vertical sync pulse to the value that specifies the beginning of the vertical sync pulse (see the description of the Vertical Sync Start Register for more details).

CR12 - Vertical Display Enable End Register

Address: 3B5h/3D5h (index=12h)

Default: Undefined

Attributes:

Read/Write

Bit	Description
7:0	Vertical Display Enable End Bits [7:0]. This register provides the 8 least significant bits of a 10-bit value that specifies the number of the last scan line within the active display area. In standard VGA modes, this value is described in 10 bits with bits [6,1] of the Overflow Register (CR07) supplying the two most significant bits. This 10-bit value should be programmed to be equal to the number of the last scan line within in the active display area. Since the active display area always starts on the 0th scan line, this number should be equal to the total number of scan lines within the active display area, minus 1.

CR13 - Offset Register

Address: 3B5h/3D5h (index=13h)

Default: Undefined

Attributes:

Read/Write

Bit	Description
7:0	Offset Bits [7:0]. This register provides either all 8 bits of an 8-bit value that specifies the number of words or DWords of frame buffer memory occupied by each horizontal row of characters. Whether this value is interpreted as the number of words or DWords is determined by the settings of the bits in the Clocking Mode Register (SR01). In standard VGA modes, the offset is described with an 8-bit value, all the bits of which are provided by this register. This 8-bit value should be programmed to be equal to either the number of words or DWords (depending on the setting of the bits in the Clocking Mode Register, SR01) of frame buffer memory that is occupied by each horizontal row of characters.



CR14 - Underline Location Register

Address: 3B5h/3D5h (index=14h)

Default: 0UUU UUUUb (U=Undefined)

Attributes:

Read/Write

Bit	Description															
7	Reserved. Read as 0.															
6	<p>DWord Mode.</p> <p>0 = Frame buffer addresses are interpreted by the frame buffer address decoder as being either byte addresses or word addresses, depending on the setting of bit 6 of the CRT Mode Control Register (CR17).</p> <p>1 = Frame buffer addresses are interpreted by the frame buffer address decoder as being DWord addresses, regardless of the setting of bit 6 of the CRT Mode Control Register (CR17).</p> <p>This bit is used in conjunction with bits 6 and 5 of the CRT Mode Control Register (CR17) to select how frame buffer addresses from the CPU are interpreted by the frame buffer address decoder as shown below:</p> <table border="1"> <thead> <tr> <th>CR14[6]</th> <th>CR17[6]</th> <th>Addressing Mode</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>Word Mode</td> </tr> <tr> <td>0</td> <td>1</td> <td>Byte Mode</td> </tr> <tr> <td>1</td> <td>0</td> <td>DWord Mode</td> </tr> <tr> <td>1</td> <td>1</td> <td>DWord Mode</td> </tr> </tbody> </table>	CR14[6]	CR17[6]	Addressing Mode	0	0	Word Mode	0	1	Byte Mode	1	0	DWord Mode	1	1	DWord Mode
CR14[6]	CR17[6]	Addressing Mode														
0	0	Word Mode														
0	1	Byte Mode														
1	0	DWord Mode														
1	1	DWord Mode														
5	<p>Count By 4.</p> <p>0 = The memory address counter is incremented either every character clock or every other character clock, depending upon the setting of bit 3 of the CRT Mode Control Register.</p> <p>1 = The memory address counter is incremented either every 4 character clocks or every 2 character clocks, depending upon the setting of bit 3 of the CRT Mode Control Register. . This is used in mode x13 to allow for using all four planes.</p> <p>This bit is used in conjunction with bit 3 of the CRT Mode Control Register (CR17) to select the number of character clocks are required to cause the memory address counter to be incremented as shown, below:</p> <table border="1"> <thead> <tr> <th>CR14[5]</th> <th>CR17[3]</th> <th>Addressing Incrementing Interval</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>every character clock</td> </tr> <tr> <td>0</td> <td>1</td> <td>every 2 character clocks</td> </tr> <tr> <td>1</td> <td>0</td> <td>every 4 character clocks</td> </tr> <tr> <td>1</td> <td>1</td> <td>every 2 character clocks</td> </tr> </tbody> </table>	CR14[5]	CR17[3]	Addressing Incrementing Interval	0	0	every character clock	0	1	every 2 character clocks	1	0	every 4 character clocks	1	1	every 2 character clocks
CR14[5]	CR17[3]	Addressing Incrementing Interval														
0	0	every character clock														
0	1	every 2 character clocks														
1	0	every 4 character clocks														
1	1	every 2 character clocks														
4:0	Underline Location. This field specifies which horizontal line of pixels in a character box is to be used to display a character underline in text mode. The horizontal lines of pixels in a character box are numbered															

Bit	Description
	from top to bottom, with the top-most line being number 0. The value specified by these 5 bits should be the number of the horizontal line on which the character underline mark is to be shown.

CR15 - Vertical Blanking Start Register

Address: 3B5h/3D5h (index=15h)

Default: Undefined

Attributes:

Read/Write

Bit	Description
7:0	Vertical Blanking Start Bits [7:0]. This register provides the 8 least significant bits of a 10-bit value that specifies the beginning of the vertical blanking period relative to the beginning of the active display area of the screen. In standard VGA modes, the vertical blanking start is specified with a 10-bit value. The most and second-most significant bits of this value are supplied by bit 5 of the Maximum Scan Line Register (CR09) and bit 3 of the Overflow Register (CR07), respectively. This 10-bit value should be programmed to be equal the number of scan lines from the beginning of the active display area to the beginning of the vertical blanking period. Since the active display area always starts on the 0th scan line, this number should be equal to the number of the scan line on which vertical blanking begins.

CR16 - Vertical Blanking End Register

Address: 3B5h/3D5h (index=16h)

Default: Undefined

Attributes:

Read/Write

This register provides a 8-bit value that specifies the end of the vertical blanking period relative to its beginning.

Bit	Description
7:0	Vertical Blanking End Bits [7:0]. This 8-bit value should be set equal to the least significant 8 bits of the result of adding the length of the vertical blanking period in terms of the number of scan lines that occur within the length of the vertical blanking period to the value that specifies the beginning of the vertical blanking period (see the description of the Vertical Blanking Start Register for details).



CR17 - CRT Mode Control

Address: 3B5h/3D5h (index=17h)

Default: 0UU0 UUUUb (U=Undefined)

Attributes:

Read/Write

Bit	Description												
7	<p>CRT Controller Reset. This bit has no effect except in native VGA modes (non-centered).</p> <p>0 = Forces horizontal and vertical sync signals to be inactive. No other registers or outputs are affected.</p> <p>1 = Permits normal operation.</p>												
6	<p>Word Mode or Byte Mode.</p> <p>0 = The memory address counter's output bits are shifted by 1 bit position before being passed on to the frame buffer address decoder such that they are made into word-aligned addresses when bit 6 of the Underline Location Register (CR17) is set to 0.</p> <p>1 = The memory address counter's output bits remain unshifted before being passed on to the frame buffer address decoder such that they remain byte-aligned addresses when bit 6 of the Underline Location Register (CR17) is set to 0.</p> <p>This bit is used in conjunction with bits 6 and 5 of the CRT Mode Control Register (CR17) to control how frame buffer addresses from the memory address counter are interpreted by the frame buffer address decoder as shown below:</p> <p>CR14[6] CR17[6] Address Mode</p> <table><tbody><tr><td>0</td><td>0</td><td>Word Mode - Addresses from the memory address counter are shifted once to become word-aligned</td></tr><tr><td>0</td><td>1</td><td>Byte Mode - Addresses from the memory address counter are not shifted</td></tr><tr><td>1</td><td>0</td><td>DWord Mode - Addresses from the memory address counter are shifted twice to become DWord-aligned</td></tr><tr><td>1</td><td>1</td><td>DWord Mode - Addresses from the memory address counter are shifted twice to become DWord-aligned</td></tr></tbody></table>	0	0	Word Mode - Addresses from the memory address counter are shifted once to become word-aligned	0	1	Byte Mode - Addresses from the memory address counter are not shifted	1	0	DWord Mode - Addresses from the memory address counter are shifted twice to become DWord-aligned	1	1	DWord Mode - Addresses from the memory address counter are shifted twice to become DWord-aligned
0	0	Word Mode - Addresses from the memory address counter are shifted once to become word-aligned											
0	1	Byte Mode - Addresses from the memory address counter are not shifted											
1	0	DWord Mode - Addresses from the memory address counter are shifted twice to become DWord-aligned											
1	1	DWord Mode - Addresses from the memory address counter are shifted twice to become DWord-aligned											
5	<p>Address Wrap. This bit is only effective when word mode is made active by setting bit 6 in both the Underline Location Register and this register to 0.</p> <p>0 = Wrap frame buffer address at 16 KB. This is used in CGA-compatible modes.</p> <p>1 = No wrapping of frame buffer addresses.</p>												

Bit	Description															
4	Reserved. Read as 0.															
3	<p>Count By 2. This bit is used in conjunction with bit 5 of the Underline Location Register (CR14) to select the number of character clocks are required to cause the memory address counter to be incremented.</p> <p>0 = The memory address counter is incremented either every character clock or every 4 character clocks, depending upon the setting of bit 5 of the Underline Location Register.</p> <p>1 = The memory address counter is incremented either every other clock.</p> <table border="0" data-bbox="196 594 987 940"> <thead> <tr> <th>CR14[5]</th> <th>CR17[3]</th> <th>Address Incrementing interval</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>every character clock</td> </tr> <tr> <td>0</td> <td>1</td> <td>every 2 character clocks</td> </tr> <tr> <td>1</td> <td>0</td> <td>every 4 character clocks</td> </tr> <tr> <td>1</td> <td>1</td> <td>every 2 character clocks</td> </tr> </tbody> </table>	CR14[5]	CR17[3]	Address Incrementing interval	0	0	every character clock	0	1	every 2 character clocks	1	0	every 4 character clocks	1	1	every 2 character clocks
CR14[5]	CR17[3]	Address Incrementing interval														
0	0	every character clock														
0	1	every 2 character clocks														
1	0	every 4 character clocks														
1	1	every 2 character clocks														
2	<p>Horizontal Retrace Select. This bit provides a way of effectively doubling the vertical resolution by allowing the vertical timing counter to be clocked by the horizontal retrace clock divided by 2 (usually, it would be undivided).</p> <p>0 = The vertical timing counter is clocked by the horizontal retrace clock.</p> <p>1 = The vertical timing counter is clocked by the horizontal retrace clock divided by 2.</p>															
1	<p>Select Row Scan Counter.</p> <p>0 = A substitution takes place, where bit 14 of the 16-bit memory address generated of the memory address counter (after the stage at which these 16 bits may have already been shifted to accommodate word or DWord addressing) is replaced with bit 1 of the row scan counter at a stage just before this address is presented to the frame buffer address decoder.</p> <p>1 = No substitution takes place. See following tables.</p>															
0	<p>Compatibility Mode Support.</p> <p>0 = A substitution takes place, where bit 13 of the 16-bit memory address generated of the memory address counter (after the stage at which these 16 bits may have already been shifted to accommodate word or DWord addressing) is replaced with bit 0 of the row scan counter at a stage just before this address is presented to the frame buffer address decoder.</p> <p>1 = No substitution takes place. See following tables.</p>															

The following tables show the possible ways in which the address bits from the memory address counter can be shifted and/or reorganized before being presented to the frame buffer address decoder. First, the address bits generated by the memory address counter are reorganized, if need be, to accommodate



byte, word or DWord modes. The resulting reorganized outputs (MAOut15-MAOut0) from the memory address counter may also be further manipulated with the substitution of bits from the row scan counter (RSOut1 and RSOut0) before finally being presented to the input bits of the frame buffer address decoder (FBIn15-FBIn0).

Memory Address Counter Address Bits [15:0]

	Byte Mode CR14 bit 6=0 CR17 bit 6=1 CR17 bit 5=X	Word Mode CR14 bit 6=0 CR17 bit 6=0 CR17 bit 5=1	Word Mode CR14 bit 6=0 CR17 bit 6=0 CR17 bit 5=0	DWord Mode CR14 bit 6=1 CR17 bit 6=X CR17 bit 5=X
MAOut0	0	15	13	12
MAOut1	1	0	0	13
MAOut2	2	1	1	0
MAOut3	3	2	2	1
MAOut4	4	3	3	2
MAOut5	5	4	4	3
MAOut6	6	5	5	4
MAOut7	7	6	6	5
MAOut8	8	7	7	6
MAOut9	9	8	8	7
MAOut10	10	9	9	8
MAOut11	11	10	10	9
MAOut12	12	11	11	10
MAOut13	13	12	12	11
MAOut14	14	13	13	12
MAOut15	15	14	14	13

X = Don't Care

Frame Buffer Address Decoder

	CR17 bit 1=1	CR17 bit 1=1	CR17 bit 1=0	CR17 bit 1=0
	CR17 bit 0=1	CR17 bit 0=0	CR17 bit 0=1	CR17 bit 0=0
FBIn0	MAOut0	MAOut0	MAOut0	MAOut0
FBIn1	MAOut1	MAOut1	MAOut1	MAOut1
FBIn2	MAOut2	MAOut2	MAOut2	MAOut2

	CR17 bit 1=1	CR17 bit 1=1	CR17 bit 1=0	CR17 bit 1=0
	CR17 bit 0=1	CR17 bit 0=0	CR17 bit 0=1	CR17 bit 0=0
FBIIn3	MAOut3	MAOut3	MAOut3	MAOut3
FBIIn4	MAOut4	MAOut4	MAOut4	MAOut4
FBIIn5	MAOut5	MAOut5	MAOut5	MAOut5
FBIIn6	MAOut6	MAOut6	MAOut6	MAOut6
FBIIn7	MAOut7	MAOut7	MAOut7	MAOut7
FBIIn8	MAOut8	MAOut8	MAOut8	MAOut8
FBIIn9	MAOut9	MAOut9	MAOut9	MAOut9
FBIIn10	MAOut10	MAOut10	MAOut10	MAOut10
FBIIn11	MAOut11	MAOut11	MAOut11	MAOut11
FBIIn12	MAOut12	MAOut12	MAOut12	MAOut12
FBIIn13	MAOut13	MAOut13	RSOut0	RSOut0
FBIIn14	MAOut14	RSOut1	MAOut14	RSOut1
FBIIn15	MAOut15	MAOut15	MAOut15	MAOut15

CR18 - Line Compare Register

Address: 3B5h/3D5h (index=18h)

Default: Undefined

Attributes:

Read/Write

Bit	Description
7:0	<p>Line Compare Bits [7:0]. This register provides the 8 least significant bits of a 10-bit value that specifies the scan line at which the memory address counter restarts at the value of 0. Bit 6 of the Maximum Scan Line Register (CR09) supplies the most significant bit, and bit 4 of the Overflow Register (CR07) supplies the second most significant bit.</p> <p>Normally, this 10-bit value is set to specify a scan line after the last scan line of the active display area. When this 10-bit value is set to specify a scan line within the active display area, it causes that scan line and all subsequent scan lines in the active display area to display video data starting at the very first byte of the frame buffer. The result is what appears to be a screen split into a top and bottom part, with the image in the top part being repeated in the bottom part. (This register is only used in split screening modes, and this is not a problem because split screening is not actually used for extended modes. As a result, there is no benefit to extending the existing overflow bits for higher resolutions.)</p> <p>When used in cooperation with the Start Address High Register (CR0C) and the Start Address Low Register (CR0D), it is possible to create a split display, as described</p>

Bit	Description
	earlier, but with the top and bottom parts displaying different data. The top part will display whatever data exists in the frame buffer starting at the address specified in the two aforementioned start address registers, while the bottom part will display whatever data exists in the frame buffer starting at the first byte of the frame buffer.

CR22 - Memory Read Latch Data Register

Address: 3B5h/3D5h (index=22h)

Default: 00h

Attributes:

Read Only

Bit	Description
7:0	Memory Read Latch Data. This field provides the value currently stored in 1 of the four memory read latches. Bits 1 and 0 of the Read Map Select Register (GR04) select which of the four memory read latches may be read via this register.

CR24 - Toggle State of Attribute Controller Register

Address: 3B5h/3D5h (index=24h)

Default: 00h

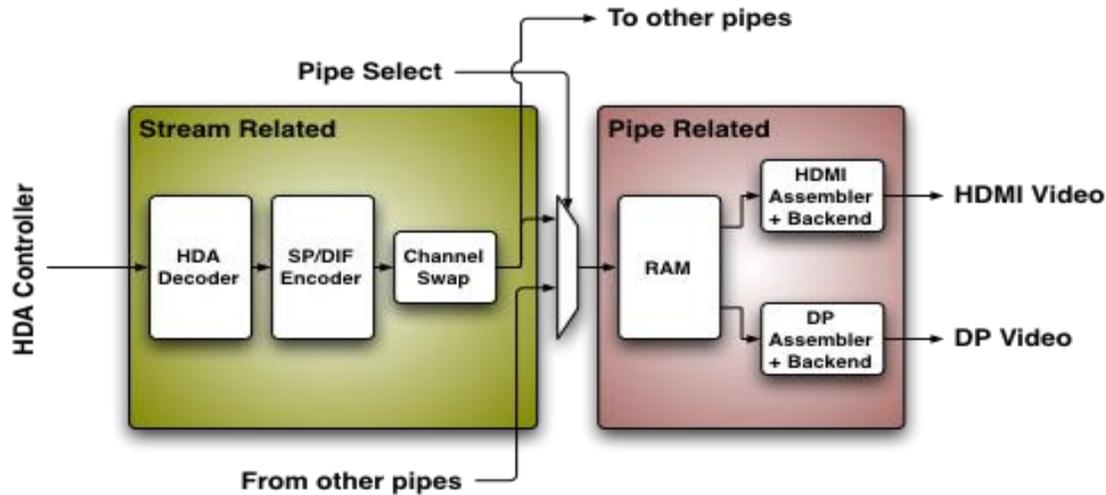
Attributes:

Read Only

Bit	Description
7	Toggle Status. Indicates where the last write to attribute register was to: 0 = index port 1 = data port
6:0	Reserved. Read as 0.

Display Audio Codec Verbs

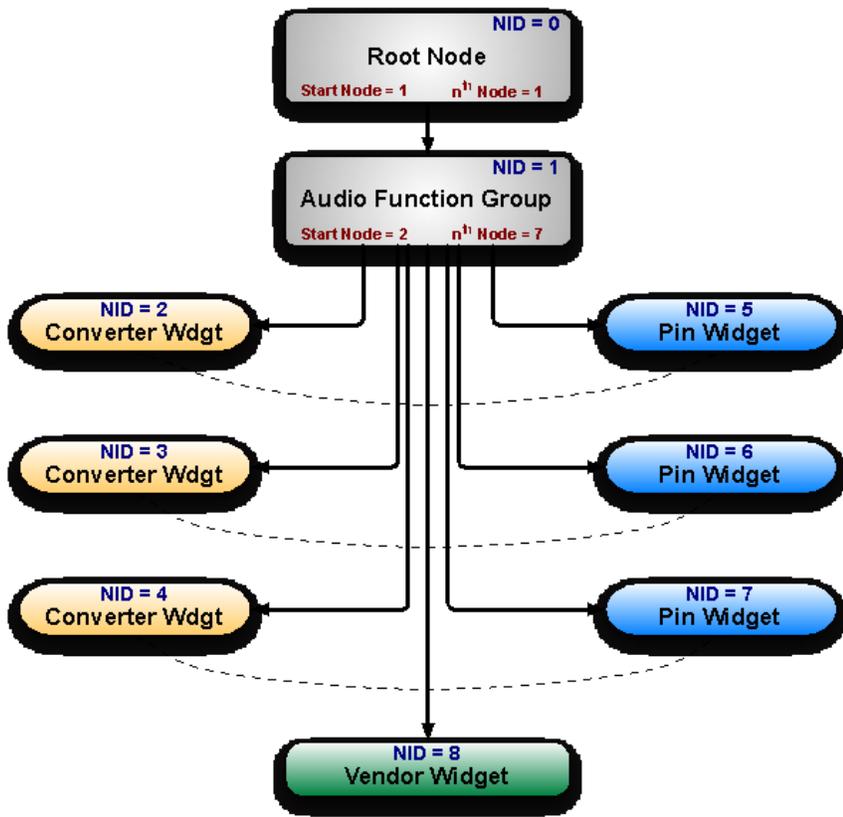
Block Diagram



Codec Node Hierarchy

The diagram below shows the hierarchy of the internal codec. The codec is presented as a single codec with multiple endpoints. By operating as a single codec, only one driver needs to be loaded on the system.

Inside the codec are three "converter widgets" and three "pin widgets", responsible for taking data from HD Audio DMA engines and placing into an HDMI/DP stream. Each pin widget has a 1-1 connection to a converter widget (as indicated by the dotted lines in the diagram).



Programming

Programming of the codec is performed by "verbs" as described in the HD Audio specification. These verbs travel over the internal HD Audio link at a rate of 1 verb per frame. A verb can either come from the CORB, with responses using the RIRB, or using an immediate command and response mechanism (ICR). Device 2 contains its own copy of an ICR mechanism as a back-door into the audio codec.

Verb Support

Verb ID		Verb Name/Description	Node ID							
Set	Get		01h	02h	03h	04h	05h	06h	07h	08h
2h	Ah	Stream Descriptor Format		Y	Y	Y				
3h	Bh	Set Amplifier Mute					Y	Y	Y	
-	F00h	Get Parameters	Y	Y	Y	Y	Y	Y	Y	Y
701h	F01h	Connection Select Control					Y	Y	Y	
-	F02h	Connection List Entry					Y	Y	Y	
705h	F05h	Power State		Y	Y	Y	Y	Y	Y	
706h	F06h	Channel and Stream ID		Y	Y	Y				
707h	F07h	Pin Widget Control					Y	Y	Y	
708h	F08h	Unsolicited Response Enable					Y	Y	Y	Y
-	F09h	Pin Sense					Y	Y	Y	
-	F0Dh	Digital Converter		Y	Y	Y				
70Dh	-	Digital Converter 1		Y	Y	Y				
70Eh	-	Digital Converter 2		Y	Y	Y				
-	F1Ch	Configuration Default					Y	Y	Y	
71Ch	-	Configuration Default Byte 0					Y	Y	Y	
71Dh	-	Configuration Default Byte 1					Y	Y	Y	
71Eh	-	Configuration Default Byte 2					Y	Y	Y	
71Fh	-	Configuration Default Byte 3					Y	Y	Y	
-	F20h	Subsystem ID	Y							
-	F21h	Subsystem ID	Y							
-	F22h	Subsystem ID	Y							
-	F23h	Subsystem ID	Y							
720h	-	Subsystem ID[7: 0]	Y							
721h	-	Subsystem ID[15: 8]	Y							
722h	-	Subsystem ID[23:16]	Y							
723h	-	Subsystem ID[31:24]	Y							
72Dh	F2Dh	Converter Channel Count		Y	Y	Y				
-	F2Eh	HDMI/DP Info Size					Y	Y	Y	



Verb ID		Verb Name/Description	Node ID							
730h	F30h	HDMI Info Index					Y	Y	Y	
731h	F31h	HDMI Info Data					Y	Y	Y	
732h	F32h	HDMI Info Transmit Control					Y	Y	Y	
734h	F34h	Converter Channel Map					Y	Y	Y	
735h	F35h	Device Select					Y	Y	Y	
-	F36h	Display Device List Entry					Y	Y	Y	
73Ch	73Ch	DisplayPort Stream ID					Y	Y	Y	
73Eh	-	Digital Converter 3	Y	Y	Y					
73Fh	-	Digital Converter 4	Y	Y	Y					
-	F80h	HDMI / DP Status								Y
781h	F81h	HDMI Vendor Verb								Y
782h	-	GTC Capture Trigger								Y
-	F83h	Captured Wall Clock Value								Y
-	F84h	Captured GTC Value								Y
-	F85h	Get GTC Offset Value								Y
785h	-	Set GTC Offset Value[7: 0]								Y
786h	-	Set GTC Offset Value[15: 8]								Y
787h	-	Set GTC Offset Value[23:16]								Y
788h	-	Set GTC Offset Value[31:24]								Y
789h	F89h	Converter Channel Count								Y

Parameter Support

Param ID	Parameter Name	Node ID									
		00h	01h	02h	03h	04h	05h	06h	07h	08h	
00h	Vendor ID	Y									
02h	Revision ID	Y									
04h	Subordinate Node Count	Y	Y								
05h	Function Group Type		Y								
08h	Audio Function Group Capabilities										
09h	Audio Widget Capabilities			Y	Y	Y	Y	Y	Y	Y	
0Ah	Sample Size, Rate CAPs			Y	Y	Y					
0Bh	Stream Formats			Y	Y	Y					
0Ch	Pin Capabilities						Y	Y	Y		
0Dh	Input Amp Capabilities										
0Eh	Connection List Length						Y	Y	Y		
0Fh	Supported Power States		Y								

Param ID	Parameter Name	Node ID									
		00h	01h	02h	03h	04h	05h	06h	07h	08h	
10h	Processing Capabilities										
11h	GPIO Count										
12h	Output Amp Capabilities						Y	Y	Y		
13h	Volume Knob Capabilities										
15h	Device List Length						Y	Y	Y		

Node ID Descriptions

Below is the description of the valid settings of the Vendor Verb 781h at node ID 02h for enabling the features of the Display Audio Codec. The bits 7:4 are not applicable when bit 0 is set to 1.

Vendor Verb 781h Bits					Node ID															
Port Select [7:4]	Reserved [3:2]	Enable DP1.2 [1]	Enable all pins and all Converters [0]	Description of the control bits	00h	01h	02h	03h	04h	05h	06h	07h	08h	09h	0Ah	0Bh	0Ch	0Dh	0Eh	0Fh
0000	0	0	0	Single Pin is exposed with DDIA on Node ID 4	Root	Function	Vendor	Converter1	DDIA	X	X	X	X	X	X	X	X	X	X	X
0001	0	0	0	Single Pin is exposed with DDIB on Node ID 4	Root	Function	Vendor	Converter1	DDIB	X	X	X	X	X	X	X	X	X	X	X
0010	0	0	0	Single Pin is exposed with DDIC on Node ID 4	Root	Function	Vendor	Converter1	DDIC	X	X	X	X	X	X	X	X	X	X	X
0011	0	0	0	Single Pin is exposed with USB C1 on Node ID 4	Root	Function	Vendor	Converter1	USB C1	X	X	X	X	X	X	X	X	X	X	X
010	0	0	0	Single	Ro	Functi	Vend	Convert	USB	X	X	X	X	X	X	X	X	X	X	X

Vendor Verb 781h Bits				Node ID																
Port Select [7:4]	Reserved [3:2]	Enable DP1.2 [1]	Enable all pins and all Converters [0]	Description of the control bits	00h	01h	02h	03h	04h	05h	06h	07h	08h	09h	0Ah	0Bh	0Ch	0Dh	0Eh	0Fh
0				Pin is exposed with USB C2 on Node ID 4	Root	Function	Vendor	Converter1	USB C2											
0101	0	0	0	Single Pin is exposed with USB C3 on Node ID 4	Root	Function	Vendor	Converter1	USB C3	X	X	X	X	X	X	X	X	X	X	X
0110	0	0	0	Single Pin is exposed with USB C4 on Node ID 4	Root	Function	Vendor	Converter1	USB C4	X	X	X	X	X	X	X	X	X	X	X
0111	0	0	0	Single Pin is exposed with USB C5 on Node ID 4	Root	Function	Vendor	Converter1	USB C5	X	X	X	X	X	X	X	X	X	X	X
1000	0	0	0	Single Pin is exposed with USB C6 on Node ID 4	Root	Function	Vendor	Converter1	USB C6	X	X	X	X	X	X	X	X	X	X	X

When Bit 0 is set to 1 of verb 781h then mapping as described below. Any converter can be mapped to any pin widget.

Node ID 0h = Root Node

Node ID 1h = Function Node

Node ID 2h = Vendor Widget

Node ID 3h = Converter 1

Node ID 4h = DDI A

Node ID 5h = Converter 2

Node ID 6h = DDI B

Node ID 7h = Converter 3

Node ID 8h = DDI C

Node ID 9h = Converter 4

Node ID Ah = USBC 1

Node ID Bh = USBC 2

Node ID Ch = USBC 3

Node ID Dh = USBC 4

Node ID Eh = USBC 5

Node ID Fh = USBC 6

Node ID 00h Root Node Verbs

The root node only contains a single verb - the "Get Parameters" verb at F00h.

F00h - Get Parameters

Parameters

Parameter	Symbol	Register Name
00h	PARAM_VID	Vendor ID
02h	PARAM_RID	Revision ID
04h	PARAM_SNC	Subordinate Node Count

Parameter 00h: VID - Vendor ID

Bit	Reset	Description
31:16	8086h	Vendor ID (VID): Indicates the 16-bit Vendor ID values used to identify the codec to the PnP subsystem.
15:00	<value in table below>	Device ID (DID): Indicates the 16-bit Device ID values used to identify the codec to the PnP subsystem.

Device ID
2812h



Parameter 02h: RID - Revision ID

Bit	Reset	Description
31:24	0	Reserved
23:20	1h	Major Revision (MJR): Indicates the major revision number (left of the decimal) of the High Definition Audio Specification to which the codec is fully compliant.
19:16	0h	Minor Revision (MNR): Indicates the minor revision number (right of the decimal) or "dot number" of the High Definition Audio Specification to which the codec is fully compliant.
15:08	00h	Revision ID (RID): Indicates the vendor's revision number for this given Device ID.
07:00	00h	Stepping ID (SID): Indicates optional vendor stepping number within the revision.

Parameter 04h: PARAM_SNC - Subordinate Node Count

Bit	Reset	Description
31:24	0	Reserved
23:16	0h	Starting Node Number (SNN): Indicates the first sub-node's ID is 01h.
15:08	00h	Reserved
07:00	01h	Total Number of Nodes (TNN): Indicates one sub-node

F37h GET CCF - Get Current Clock Frequency

Bits	Default	Description
31:6	0	Reserved
05	0	Current Clock 192 MHz (C192): Indicates the current clock is 192 MHz. Reserved for Display Codec
04	1	Current Clock 96 MHz (C96): Indicates the current clock is 96 MHz.
03	0	Current Clock 48 MHz (C48): Indicates the current clock is 48 MHz.
02	0	Current Clock 24 MHz (C24): Indicates the current clock is 24 MHz. Reserved for Display Codec
01	0	Current Clock 12 MHz (C12): Indicates the current clock is 12 MHz. Reserved for Display Codec
00	0	Current Clock 6 MHz (C6): Indicates the current clock is 6 MHz. Reserved for Display Codec

Node ID 01h Audio Function Group Verbs

Set Verb	Get Verb	Symbol	Name
-	F00h	GET_PARAM	Get Parameters
705h	F05h	SET_PS / GET_PS	Set Power State
-	F20h	GET_SSID	Get Subsystem ID
720h	F20h	SET_SSID0	Set/Get Subsystem ID
721h	F21h	SET_SSID1	Set/Get Subsystem ID
722h	F22h	SET_SSID2	Set//Get Subsystem ID
723h	F23h	SET_SSID3	Set/Get Subsystem ID

Set Verb	Get Verb	Symbol	Name
724h	F24h	SET CCF	Set/Get Current Clock Frequency

F00h Get Parameters

Parameter	Symbol	Register Name
04h	PARAM_SNC	Subordinate Node Count
05h	PARAM_FGT	Function Group Type
08h	PARAM_FGC	Function Group Capabilities
0Fh	PARAM_SPS	Supported Power States

Parameter 04h: PARAM_SNC - Subordinate Node Count

Bit	Reset	Description
31:24	0	Reserved
23:16	03h	Start Node Number (SNN): Indicates the start node number of widget or functional nodes in the Functional Group.
15:08	0	Reserved
07:00	07h	<p>Total Number of Nodes (TNN): Indicates 7 widgets in the Functional Group. (HDMI/DP converters (3) + HDMI/DP pins (3) + Vendor Defined Widget (1)).</p> <p>If Wigig is enabled in the vendor defined widget then the total number of nodes should be reported as 08h.</p> <p>Default for D11 is 9 [HDMI/DP converters (4) + HDMI/DP pins (5)]. If wireless widgets are enabled then total is 10 or 11.</p> <p>Default for D11p5 is 13 [HDMI/DP converters (4) + HDMI/DP pins (9)]. If wireless widgets are enabled then total is 14 or 15.</p>

Parameter 05h: PARAM_FGT - Function Group Type

Bit	Reset	Description
31:09	0	Reserved
08	0	Unsolicited Capable (UC): Not capable of generating an unsolicited response.
07:00	01h	Node Type (NT): Indicates Audio Function Group.

Parameter 08h: PARAM_FGC - Function Group Capability

Bit	Reset	Description
31:04	0	Reserved
03:00	00h	Output Delay (OD) Output Delay.



Parameter 0Fh: PARAM_SPS - Supported Power States

Bit	Reset	Description
31	1	Extended Power State Supported (EPSS): Indicates support for low power states
30	1	Clock Stop (CS): Indicates support for D3 when clock is stopped.
29:04	0	Reserved
03	1	D3 Supported (D3S): Indicates support for D3.
02	0	D2 Supported (D2S): Indicates no support for D2.
01	0	D1 Supported (D1S): Indicates no support for D1.
00	1	D0 Supported (D0S): Indicates support for D0.

Parameter 16h: PARAM_A2CAP - Azalia 2 Capabilities

Bits	Reset	Description
31:17	0	Reserved
16	0	Independent Codec Clock (ICC): When set, this indicates that the codec generates its own clock, which may drift from the link clock. When cleared, the codec's clock is locked to the link clock.
15:06	0	Reserved
05	0	Reserved for 192 MHz support.
04	1	96MHz Supported (S96): Indicates 96 MHz clock is supported.
03	1	48MHz Supported (S48): Indicates 48 MHz clock is supported. This bit must always be set.
02	0	24MHz Supported (S24): Indicates 24 MHz clock is supported. Reserved for Display Codec
01	0	12MHz Supported (S12): Indicates 12 MHz clock is supported. Reserved for Display Codec
00	0	6MHz Supported (S6) Indicates 6 MHz clock is supported. Reserved for Display Codec

705h SET_PS - Set Power State

Bits	Description
07:02	Reserved
01:00	Requested Power State (RPS): Only D0 (00) and D3 (11) may be requested

F05h GET_PS - Get Power State

Bits	Reset	Description
31:11	0	Reserved
10	0	Settings Reset (SR): Default values will not be changed. This bit will report 0 in all cases.
09	1	Clock Stop OK (CSOK): Clock stopping in D3 is OK
08	0	Error (ERR): No error will ever be reported.
07:06	0	Reserved
05:04	11	Actual Power State (APS): Indicates the current power state of the node.
03:02	0	Reserved

Bits	Reset	Description
01:00	11	Requested Power State (CPS): Reflects value written with SET_PS verb.

F20h GET SSID - Get Subsystem ID0

Bits	Reset	Description
31:00	80860101h	Subsystem ID (SSID): Reports the sub-system ID set via SET_SSIDx verbs.

720h SET SSID0 - Set Subsystem ID0

Bits	Description
07:00	Subsystem ID Bits [7:0]

721h SET SSID1 - Set Subsystem ID1

Bits	Description
07:00	Subsystem ID Bits [15:8]

722h SET SSID2 - Set Subsystem ID2

Bits	Description
07:00	Subsystem ID Bits [23:16]

723h SET SSID3 - Set Subsystem ID3

Bits	Description
07:00	Subsystem ID Bits [31:24]

724h SET CCF - Set Current Clock Frequency

Bits	Description
07:06	Reserved
05	Set Clock 192 MHz (S192): Set clock to 192 MHz.
04	Set Clock 96 MHz (S96): Set clock to 96 MHz
03	Set Clock 48 MHz (S48): Set clock to 48 MHz
02	Set Clock 24 MHz (S24): Set clock to 24 MHz
01	Set Clock 12 MHz (S12): Set clock to 12 MHz
00	Set Clock 6 MHz (S6): Set clock to 6 MHz



F24h GET CCF - Get Current Clock Frequency

Bits	Bits	Description
31:06	0	<i>Reserved</i>
05	0	Current Clock 192 MHz (C192): Indicates the current clock is 192 MHz.
04	0	Current Clock 96 MHz (C96): Indicates the current clock is 96 MHz.
03	0	Current Clock 48 MHz (C48): Indicates the current clock is 48 MHz.
02	0	Current Clock 24 MHz (C24): Indicates the current clock is 24 MHz.
01	0	Current Clock 12 MHz (C12): Indicates the current clock is 12 MHz.
00	0	Current Clock 6 MHz (C6): Indicates the current clock is 6 MHz.

7FFh SET Function Group Reset

Bits	Reset	Description
07:00	00h	<p>The Function Reset command causes the functional unit, and all widgets associated with the functional unit, to return to their power-on reset values. Note that some controls such as the Configuration Default controls should not be reset with this command. It is also possible that certain other controls, such as Caller-ID, should not be reset.</p> <p>This command does not affect the Link interface logic, which must be reset with the link RST# signal. Therefore, a codec must not initiate a Status Change request on the link.</p>

Audio Output Convertor Widget Verbs

Verb	Symbol	Verb Name
2h	SET_SDF	Set Stream Descriptor Format
Ah	GET_SDF	Get Stream Descriptor Format
F00h	GET_PARAM	Get Parameters
705h	SET_PS	Set Power State
F05h	GET_PS	Get Power State
706h	SET_CSID	Set Channel and Stream ID
F06h	GET_CSID	Get Channel and Stream ID
F0Dh	SET_DC1	Get Digital Converter
70Dh	SET_DC1	Set Digital Converter 1
70Eh	SET_DC2	Set Digital Converter 2
73Eh	SET_DC3	Set Digital Converter 3

Verb	Symbol	Verb Name
73Fh	SET_DC4	Set Digital Converter 4
72Dh	SET_CCC	Set Converter Channel Count
F2Dh	GET_CCC	Get Converter Channel Count

2hAh SETGET_SDF - SetGET Stream Descriptor Format

Bits	Reset	Description
31:15	0	Reserved
14	0	Sample Base Rate (SBR):
13:11	000	Sample Base Rate Multiplier (SBRM):
10:08	000	Sample Base Rate Divisor (SBRD):
07	0	Reserved
06:04	011	Bits / Sample (BPS): <ul style="list-style-type: none"> • 001b: Data is packed in memory in 16 bit containers on 16 bit boundaries • 010b: Data is packed in memory in 20 bit containers on 32 bit boundaries • 011b: Data is packed in memory in 24 bit containers on 24 bit boundaries • 100b: Data is packed in memory in 32 bit containers on 32 bit boundaries All other bit combinations reserved
03:00	1h	# Channels in Stream (NCS): 2 channels in each frame

F00h Get Parameters

Parameter	Symbol	Register Name
09h	PARAM_AWC	Audio Widget Capabilities
0Ah	PARAM_PSB	Parameter Sizes and Bit Rates
0Bh	PARAM_SF	Stream Formats
0Fh	PARAM_SPS	Power Supported States

Parameter 09h: AWC - Audio Widget Capabilities

Bits	Reset	Description
31:24	0	Reserved
23:20	0h	Widget Type (TYPE): Indicates this is an audio output widget
19:16		Sample Delay in Widget (DELAY):
15:13	011	Channel Count Extension (CCE): These three bits, combined with STRO, indicate that there are 8 channels supported.
11	0	L-R Swap (LRS): Indicates no left/right channel swap.
10	1	Power Control (PC): Indicates power state control
09	1	Digital (DIG): Indicates support for digital streams.
08	0	Connection List (CL): Indicates no connection list
07	0	Unsolicited Capable (UC): Indicates support for unsolicited responses.
06	0	Processing Widget (PW): Indicates no support for processing
05	0	Stripe (STRP): Indicates striping not supported.
04	1	Format Override (FO): Indicates support for formatting
03	1	Amp Parameter Override (APO): Indicates no amplifier support.
02	0	Out Amp Present (OAP): Indicates no output amplifier present.
01	0	In Amp Present (IAP): Indicates no input amplifier present.
00	1	Stereo (STRO): Indicates a stereo widget

Parameter 0Ah: PSB - PCM Sizes and Bit Rates

Bits	Reset	Description
31:21	0	<i>Reserved</i>
20	1	32-bit Support (B32): Indicates 32-bit samples supported
19	1	24-bit Support (B24): Indicates 24-bit samples supported
18	0	20-bit Support (B20): Indicates 20-bit samples supported
17	1	16-bit Support (B16): Indicates 16-bit samples supported
16	0	8-bit Support (B8): Indicates 8-bit samples not supported
15:12	0	<i>Reserved</i>
11	0	384 kHz Support (R12): Indicates 384 kHz not supported
10	1	192 kHz Support (R11): Indicates 192 kHz supported
09	1	176.4 kHz Support (R10): Indicates 176.4 kHz supported
08	1	96 kHz Support (R9): Indicates 96 kHz supported
07	1	88.2 kHz Support (R8): Indicates 88.2 kHz supported
06	1	48 kHz Support (R7): Indicates 48 kHz supported
05	1	44.1 kHz Support (R6): Indicates 44.1 kHz supported
04	1	32 kHz Support (R5): Indicates 32 kHz supported
03	0	22.05 kHz Support (R4): Indicates 22.05 kHz not supported
02	0	16 kHz Support (R3): Indicates 16 kHz not supported
01	0	11.025 kHz Support (R2): Indicates 11.025 kHz not supported
00	0	8 kHz Support (R1): Indicates 8 kHz not supported



Parameter 0Bh: SF - Stream Formats

Bits	Reset	Description
31:03	0	<i>Reserved</i>
02	1	AC3 Support (AC3): Indicates AC3 stream format is supported
01	0	Float32 Support (F32): Indicates float32 stream format not supported
00	1	PCM Support (PCM): Indicates PCM format is supported.

Parameter 0Fh: PARAM_SPS - Supported Power States

Bit	Reset	Description
31	1	Extended Power State Supported (EPSS): Indicates support for low power states
30:04	0	<i>Reserved</i>
03	1	D3 Supported (D3S): Indicates support for D3.
02	0	D2 Supported (D2S): Indicates no support for D2.
01	0	D1 Supported (D1S): Indicates no support for D1.
00	1	D0 Supported (D0S): Indicates support for D0.

705h SET_PS - Set Power State

Bits	Description
07:02	<i>Reserved</i>
01:00	Requested Power State (RPS): Only D0 (00) and D3 (11) may be requested

F05h GET_PS - Get Power State

Bits	Reset	Description
31:11	0	<i>Reserved</i>
10	0	Settings Reset (SR): Default values will not be changed. This bit will report 0 in all cases.
09	0	Clock Stop OK (CSOK): Clock stopping in D3 is not OK

Bits	Reset	Description
08	0	Error (ERR): No error will ever be reported.
07:06	0	Reserved
05:04	11	Actual Power State (APS): Indicates the current power state of the node.
03:02	0	Reserved
01:00	11	Requested Power State (CPS): Reflects value written with SET_PS verb.

706hF06h GETSET_CSID - GetSet Channel and Stream ID

Bits	Reset	Description
07:04	0h	Stream ID (SID): Link stream used by the converter for data output.
03:00	0h	Lowest Channel Number (LCN): Lowest channel used by the converter.

Digital Converter Verbs

F0Dh: GET_DC - Get Digital Converter

Bits	Reset	Description
31:24	0	Reserved
23	1	Keep Alive (KA): See SET_DC3.KA
22:20	0	Reserved
19:16	0h	IEC Coding Type (ICT): See SET_DC3.ICT
15	0	Reserved
14:08	00h	Category Code (CC): See SET_DC1.CC
07	0	Level (LVL): See SET_DC1.LVL
06	0	Professional (PRO): See SET_DC1.PRO
05	0	Audio is not PCM (AUDIO): See SET_DC1.AUDIO
04	0	Copyright (COPY): See SET_DC1.COPY
03	0	Pre-emphasis (PRE): See SET_DC1.PRE
02	0	Validity Configuration (VCFG): See SET_DC1.VCFG

Bits	Reset	Description
01	0	Validity (V): See SET_DC1.V
00	1	Digital Enable (DIGEN): See SET_DC1.DIGEN

70Dh: SET_DC1 - Set Digital Converter 1

Bits	Description
07	Level (LVL): S/PDIF IEC Generation Level.
06	Professional (PRO): When set, indicates professional use of channel.
05	Audio is not PCM (AUDIO): When set, data is non-PCM format.
04	Copyright (COPY): When set, copyright asserted.
03	Pre-emphasis (PRE): When set, enables filter pre-emphasis.
02	Validity Configuration (VCFG): Determines S/PDIF transmitter behavior when data is not being transmitted.
01	Validity (V): Affects the validity flag transmitted in each sub-frame, and enables S/PDIF transmitter to maintain connection during error or mute conditions.
00	Digital Enable (DIGEN): When set, enables digital content

70Eh: Digital Converter 2

Bits	Description
07	Reserved
06:00	Category Code (CC): S/PDIF IEC Category Code.

73Eh: Digital Converter 3

Bits	Description
07	Keep Alive
06:04	Reserved
03:00	IEC Coding Type

73Fh: Digital Converter 4

Bits	Description
07:00	Reserved

72DhF2Dh GETSET_CCC - GetSet Converter Channel Count

Bits	Reset	Description
07:04	0	Reserved
03:00	0000	Converter Channel Count 1 (0 th order)

Pin Widget Verbs

Set Verb	Get Verb	Symbol	Verb Name
3h	-	SET_AM	Set Amplifier Mute
-	Bh	GET_AM	Get Amplifier Mute
-	F00h	-	Get Parameters
701h	F01h	SET_CSC / GET_CSC	Set/Get Connection Select Control
-	F02h	-	Get Connection List Entry
705h	F05h	SET_PS / GET_PS	Set/Get Power State
707h	F07h	SET_PWC / GET_PWC	Set/Get Pin Widget Control
708h	F08h	SET_UE / GET_UE	Set/Get Unsolicited Response Enable
-	F09h	-	Get Pin Sense
71Ch	-	SET_CD0	Set Configuration Default Byte 0
71Dh	-	SET_CD1	Set Configuration Default Byte 1
71Eh	-	SET_CD2	Set Configuration Default Byte 2
71Fh	-	SET_CD3	Set Configuration Default Byte 3
-	F1Ch	GET_CD	Get Configuration Default
-	F2Eh	GET_HDIS	Get HDMI/DP Info Size
730h	F30h	SET_HII / GET_HII	Set/Get HDMI Info Index
731h	F31h	SET_HID / GET_HID	Set/Get HDMI Info Data
732h	F32h	SET_HITC / GET_HITC	Set/Get HDMI Info Transmit Control
733h	F33h	SET_PC / GET_PC	Set/Get Protection Control
734h	F34h	SET_CCM / GET_CCM	Set/Get Converter Channel Map
735h	F35h	SET_DS / GET_DS	Set/Get Device Select
-	F36h	GET_DDLE	Get Display Device List Entry
73Ch	F3Ch	SET_DPID / GET_DPID	Set/Get DisplayPort Stream ID



3h SET_AM - Set Amplifier Mute

Bits	Bits	Description
15	0	Set Output Amp (SOA):.
14	0	Set Input Amp (SIA):.
13	0	Set Left Amp (SLA):.
12	0	Set Right Amp (SRA):.
11:08	0h	Index (IDX):
07	1	Mute (MUTE): When set, amp muted.
06:00	0	<i>Reserved</i>

B8h GET_AM - Get Amplifier Mute

Bits	Bits	Description
31:08	0	Reserved
07	1	Mute (MUTE): When set, amp is muted.
06:00	0	Reserved

F00h Get Parameters

Parameter	Symbol	Register Name
09h	PARAM_AWC	Audio Widget Capabilities
0Ch	PARAM_PC	Pin Capabilities
0Eh	PARAM_CLL	Connection List Length
12h	PARAM_OAC	Output Amplifier Capabilities
15h	PARAM_DLL	Device List Length
0Fh	PARAM_SPS	Supported Power States

Parameter 09h: AWC - Audio Widget Capabilities

Bits	Reset	Description
31:24	0	Reserved
23:20	4h	Widget Type (TYPE): Indicates this is a pin complex widget
19:16	0	Sample Delay in Widget (DELAY): No delay through the pin widget.
15:13	011	Channel Count Extension (CCE): This field, combined with STRO, indicate 8 channels supported.
11	0	L-R Swap (LRS): Indicates no left/right channel swap.
10	1	Power Control (PC): Indicates power state control
09	1	Digital (DIG): Indicates support for digital streams.
08	1	Connection List (CL): Indicates a connection list
07	1	Unsolicited Capable (UC): Indicates support for unsolicited responses.
06	0	Processing Widget (PW): Indicates no support for processing
05	0	Stripe (STRP): Indicates striping not supported.
04	0	Format Override (FO): Indicates no support for formatting
03	1	Amp Parameter Override (APO): Indicates no amplifier override support.
02	1	Out Amp Present (OAP): Indicates no output amplifier present.
01	0	In Amp Present (IAP): Indicates no input amplifier present.
00	1	Stereo (STRO): Indicates a stereo widget

Parameter 0Ch: PC - Pin Capabilities

Bits	Reset	Description
31:28	0	<i>Reserved</i>
27	1	High Bit Rate (HBR): Indicates support for high bit-rate audio
26	0	<i>Reserved</i>
25	1	Multi Stream Capable (MSC): Indicates support for DisplayPort Multistream. Will be zero in vanilla mode.
24	1	DisplayPort (DP): Indicates support for DisplayPort
23:08	0	<i>Reserved</i>
07	1	HDMI (HDMI): Indicates support for HDMI
06:05	0	<i>Reserved</i>
04	1	Output Capable (OC): Pin is output capable
03	0	<i>Reserved</i>
02	1	Presence Detect Capable (PDC): Indicates capability for presence detection
01:00	0	<i>Reserved</i>

Parameter 0Eh: CLL - Connection List Length

Bits	Reset	Description
31:08	0	<i>Reserved</i>
07	0	Long Form (LF): Indicates connection list is short form
06:00	03h	Length (LEN): Indicates there is one item in the connection list.

Parameter 12h: OAC - Output Amplifier Capabilities

Bits	Reset	Description
31	1	Mute Capable (MC): Muting is capable on this pin
30:00	0	Reserved

Parameter 15h: DLL - Device List Length

Bits	Reset	Description
31:06	0	Reserved
05:00	00h	Length (LEN): Indicates no devices

Parameter 0Fh: PARAM_SPS - Supported Power States

Bit	Reset	Description
31	1	Extended Power State Supported (EPSS): Indicates support for low power states
30:04	0	Reserved
03	1	D3 Supported (D3S): Indicates support for D3.
02	0	D2 Supported (D2S): Indicates no support for D2.
01	0	D1 Supported (D1S): Indicates no support for D1.
00	1	D0 Supported (D0S): Indicates support for D0.

701hF01h SETGET_CSC - SetGet Connection Select Control

Bits	Reset	Description
07:00	00h	Connection Select Control (CSC):

F02h GET_CLE - Get Connection List Entry

Bits	Reset	Description
31:08	0	Reserved
07:00	Varies	Connection List Entry (CLE): 02h for NodeID 05h, 03h for NodeID 06h, and 04h for NodeID 07h.



705h SET_PS - Set Power State

Bits	Description
07:02	Reserved
01:00	Requested Power State (RPS): Only D0 (00) and D3 (11) may be requested

F05h GET_PS - Get Power State

Bits	Reset	Description
31:11	0	Reserved
10	0	Settings Reset (SR): Default values will not be changed. This bit will report 0 in all cases.
09	0	Clock Stop OK (CSOK): Clock stopping in D3 is not OK
08	0	Error (ERR): No error will ever be reported.
07:06	0	Reserved
05:04	11	Actual Power State (APS): Indicates the current power state of the node.
03:02	0	Reserved
01:00	11	Requested Power State (CPS): Reflects value written with SET_PS verb.

707hF07h SETGET_PWC - SetGet Pin Widget Control

Bits	Reset	Description
07	0	Reserved
06	1	Out Enable (OE): When set, the audio is enabled
05:02	0	Reserved
01:00	00	Encoded Packet Type (EPT):

708hF08h SETGET_UE - SetGet Unsolicited Enable

Bits	Description
07	Unsolicited Enable (UE): When set, unsolicited responses are allowed
06	Reserved
05:00	Tag (TAG):

F09h GET_PS - Get Pin Sense

Determining SST / MST Mode

The audio codec will use multi-stream based indexing (MST mode) as described in the following sections if the following conditions are met, else it will use Non-MST mode.

Conditions for MST Mode	
•	In register TRANS_DDI_FUNC_CTL , the bit-field "TRANS DDI Mode Select" is set to "DP MST" AND
•	In verb GetSet iDisp Codec Vendor Verb (781h), the bit-field "Enable DP1.2 Features" (bit 1) is set to 1b.

Verb Parameters (Non-MST Mode)

For non-MST mode, the F09h verb response will automatically provide the status of the single pipe associated with the given pin node.

Nothing needs to be set in the data byte of the F09h Get Pin Sense verb.

Bits	Reset	Description
07:00	0	Reserved

Verb Parameters (MST Mode)

For MST mode, the F09h verb response will provide the status of the pipe selected in the Pipe Select bit-field, among the pipe(s) associated with the given pin node.

The data byte of the F09h Get Pin Sense verb should be programmed to select the desired pipe as shown below.

Bits	Reset	Description
07:06	0	Reserved
05:00	0	Pipe Select: Set to 000000b for pipe A, 000001b for pipe B, 000010b for pipe C, 000011b for pipe D

Verb Response (All Modes)

The verb response for the F09h Get Pin Sense verb command contains the following data regardless of which mode (Non-MST or MST) is used.

Bits	Reset	Description
31	0	Presence Detect (PD): When set presence is detected on this pin.
30	0	ELD Value (ELDV):



Bits	Reset	Description
29	0	Inactive (INA):
28:00	28:00	Reserved

71Ch SET_CD0 - Set Configuration Default Byte 0

Bits	Description
07:04	Default Association (DA):
03:00	Sequence (SEQ):

71Dh SET_CD1 - Set Configuration Default Byte 1

Bits	Description
07:04	Color (COL):
03:00	Miscellaneous (MISC):

71Eh SET_CD2 - Set Configuration Default Byte 2

Bits	Description
07:04	Default Device (DD):
03:00	Connection Type (CT):

71Fh SET_CD3 - Set Configuration Default Byte 3

Bits	Description
07:06	Port Connectivity (PC): External connectivity of the pin complex. <ul style="list-style-type: none">• 00 = Connected to jack• 01 = No physical connection• 10 = Fixed function device (integrated speaker, mic, etc.)• 11 = Both a jack and internal connection

Bits	Description											
05:00	Location (LOC):											
		Bits 3:0										
	Bits 5:4	0h: N/A	1h: Rear	2h: Front	3h: Left	4h: Right	5h: Top	6h: Bottom	7h: Special	8h: Special	9h: Special	Ah-Fh Reserved
	00: External	Y	Y	Y	Y	Y	Y	Y	Y	Y		
	01: Internal	Y							Y	Y	Y	
	10: Separate Chassis	Y	Y	Y	Y	Y	Y	Y				
11: Other	Y						Y	Y	Y			

F1Ch GET_CD - Get Configuration Default

Bits	Description
31:30	Port Connectivity (PC): See SET_CD3.PC
29:24	Location (L): See SET_CD3.L
23:20	Default Device (DD): See Set_CD2.DD
19:16	Connection Type (CT): See Set_CD2.CT
15:12	Color (COL): See SET_CD1.COL
11:08	Miscellaneous (MISC): See SET_CD1.MISC
07:04	Default Association (DA): See SET_CD0.DA
03:00	Sequence (SEQ): See SET_CD0.SEQ

F2Eh HDMIDP Info Size

Bits	Reset	Description
31:08	0	Reset
07:00	Varies	Size (SZ): Indexes 0 - 3 return 1Eh, index 1000 returns 53h, others reserved.

F2Fh Get ELD Data

Parameter	Symbol	Register Name
07:0h	PARAM_INDXX	ELD DATA Index



Parameter nn: ELD Data

Bits	Reset	Description
31:00	0	ELD Data

730hF30h SETGET_HII - SetGet HDMI Info Index

Bits	Reset	Description																
07:05	000	Infoframe Packet Index (IPI): <table border="1"> <thead> <tr> <th>Value</th> <th>Name</th> <th>Value</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>000</td> <td>Audio</td> <td>011</td> <td>GP3</td> </tr> <tr> <td>001</td> <td>GP</td> <td>100</td> <td>GP4</td> </tr> <tr> <td>010</td> <td>GP2</td> <td>Others</td> <td>Reserved</td> </tr> </tbody> </table>	Value	Name	Value	Name	000	Audio	011	GP3	001	GP	100	GP4	010	GP2	Others	Reserved
Value	Name	Value	Name															
000	Audio	011	GP3															
001	GP	100	GP4															
010	GP2	Others	Reserved															
04:00	00h	Byte Offset Index Pointer (BOI):																

731hF31h SETGET_HID - SetGet HDMI Info Data

Bits	Reset	Description
07:00	00h	Data (DATA): Data at current index pointed to from SET_HII verb.

732hF32h SETGET_HITC - SetGet HDMI Info Transmit Control

Bits	Reset	Description
07:06	00	InfoFrame Control Current Indexed Frame (IFCCIF): <ul style="list-style-type: none"> • 00 = Disable Transmit • 01 = Reserved • 10 = Transmit Once • 11 = Best Effort
05:00	0	<i>Reserved</i>

733h SET_PC - Set Protection Control

Bits	Description
07:03	Unsolicited Response Sub Tag (URST): Subtag to use for unsolicited responses.
02	Reserved

734hF34h SETGET_CCM - GetSet Converter Channel Map

Bits	Reset	Description
07:04	0h	Converter Channel (CC):
03:00	0h	Slot (SN):

735h SET_DS - Set Device Select

Bits	Reset	Description
07:06	0	<i>Reserved</i>
05:00	00h	Device (D): 000000, 000001, 000010, 000011 (based upon number of devices present)

F35h: GET_DS - Get Device Select

Bits	Description
31:12	Reserved: Set to 0
11:06	SinK Device ID: Sink Device ID in the multi stream topology of the DP hierarchy. Device attached to Pipe A will have ID of "000000", PipeB will have "000001", Pipe C will have "000010" and Pipe D will have "000011".
05:00	Device (D): Device Entry index currently set

F36h GET_DDLE - Get Display Device List Entry

Bits	Bits	Description
31:16	0	<i>Reserved</i>
15	0	<i>Reserved</i>
14	0	IA of Entry 3
13	0	ELD V of Entry 3
12	0	PD of Entry 3
11	0	<i>Reserved</i>
10	0	IA of Entry 2



Bits	Bits	Description
09	0	ELDV of Entry 2
08	0	PD of Entry 2
07	0	<i>Reserved</i>
06	0	IA of Entry 1
05	0	ELDV of Entry 1
04	0	PD of Entry 1
03	0	<i>Reserved</i>
02	0	IA of Entry 0
01	0	ELDV of Entry 0
00	0	PD of Entry 0

73ChF3Ch SETGET_DPID - SetGet DisplayPort Stream ID

Bits	Reset	Description
07:03	00h	Tag (TAG): Represents the SSID that will go in the lower 5 bits of the SSID
02:00	000	Index (IDX): Pointer to program multiple SSID

Intel Vendor Widget Verbs

Set Verb	Get Verb	Symbol	Verb Name
-	F00h	GET_PARAM	Get Parameters
-	F80h	GET_HDPS	Get HDMI/DP Status
781h	F81h	SET_HVV / GET_HVV	Set/Get iDisp Codec Vendor Verb
782h	-	SET_GTCT	Set GTC Trigger
-	F83h	GET_CWC	Get Captured Wall Clock
	F84h	GET_CGTC	Get Captured GTC Value
	F85h	GET_GOF	Get GTC Offset Value
785h	-	SET_GOF0	Set GTC Offset Value Byte 0
786h	-	SET_GOF1	Set GTC Offset Value Byte 1
787h	-	SET_GOF2	Set GTC Offset Value Byte 2

Set Verb	Get Verb	Symbol	Verb Name
788h	-	SET_GOF3	Set GTC Offset Value Byte 3
789h	F89h	SET_GDI / GET_GDI	Set/Get GTC Offset Device Index

F00h Get Parameters

Parameter	Symbol	Register Name
09h	PARAM_AWC	Audio Widget Capabilities

Parameter 09h: AWC - Audio Widget Capabilities

Bits	Reset	Description
31:24	0	Reserved
23:20	Fh	Widget Type (TYPE): Indicates this is a vendor defined widget
19:00	0	Reserved

71Eh SET_GET_GFXMAILBOX - Set Get GFX MAILBOX Byte 2

71Eh: SET GFX MAILBOXM

Bits	Default	Description
07:00	00	Contents to be defined by GFX driver and audio driver

F1Eh: GET GFX MAILBOX

Bits	Default	Description
31:24	00h	Other values of 71F, verbs
23:16	00h	Contents to be defined by GFX driver and audio driver
15:00	0000h	Other values of 71D, 71C verb

728h SET CLOCK OFF - Set Clock Off Command

Bits	Description
07:00	Data is Irrelevant



708hF08h SETGET_UE - SetGet Unsolicited Enable

Bits	Description
07	Unsolicited Enable (UE): When set, unsolicited responses from GFX MAIL BOX register writes are allowed.
06	Reserved
05:00	Tag (TAG): Tag for GFX Mail box register unsol responses.

781hF81h GETSET_VV - GetSet iDisp Codec Vendor Verb

Bits	Default	Description
07:04	0	<p>Port Select: This field is programmed to select the port to be exposed to the inbox driver in the vanilla mode. Will not reset on double function group reset.</p> <p>0000 -> DDI A 0001 -> DDI B 0010 -> DDI C 0011 -> USBC1 0100 -> USBC2 0101 -> USBC3 0110 -> USBC4 0111 -> USBC5 1000 -> USBC6</p>
03:02	0	Reserved
01	0	Enable DP1.2 Features (EDP12): When set, DP1.2 features are enabled. Will not reset on double function group reset.
00	0	Enable all Pins and all Converter Widget (E3P): When set, all the pins and converter widgets are enabled and can respond to HD Audio Verbs. When cleared, only one Pin and one converter widgets are exposed. Pin selection is done programming bits 7:5. Will not reset on double function group reset.

782h SET_GTCT - Set GTC Trigger

Bits	Bits	Description
07:00	0	Any data: The value of this field is irrelevant. The access of the SET causes a capture to occur.

F83h GET_CGTC - Get Captured GTC Value

Bits	Bits	Description
07	0	GTC Value: 32-bit GTC value captured on the SET_GTCT verb

F84h GET_CWC - Get Captured Wall Clock Value

Bits	Bits	Description
31:00	0	Wall Clock Value: 32-bit wall clock value captured on the SET_GTCT verb

F85h GET_GOF - Get GTC Offset Value

Bits	Reset	Description
31:00	0h	Value (VAL): Reports the GTC Offset Value.

785h SET_GOF0 - Set GTC Offset Value Byte 0

Bits	Description
07:00	GTC Offset Value Bits [7:0]

786h SET_GOF1 - Set GTC Offset Value Byte 1

Bits	Description
07:00	GTC Offset Value Bits [15:8]

787h SET_GOF2 - Set GTC Offset Value Byte 2

Bits	Description
07:00	GTC Offset Value Bits [23:16]

788h SET_GOF3 - GTC Offset Value Byte 3

Bits	Description
07:00	GTC Offset Value Bits [31:24]



789hF89h SETGET_GDI - SetGet GTC Device Index

Bits	Reset	Description
07:06	0	Reserved
05:00	00h	Device (D): 000001, 000010 (based upon number of devices present) This is the pipe based. 000000 is pipeA and so on and so forth.

Intel Wireless Widget Verbs

Set Verb	Get Verb	Symbol	Verb Name
3h	-	SET_AM	Set Amplifier Mute
-	Bh	GET_AM	Get Amplifier Mute
-	F00h	-	Get Parameters
701h	F01h	SET_CSC / GET_CSC	Set/Get Connection Select Control
-	F02h	-	Get Connection List Entry
705h	F05h	SET_PS / GET_PS	Set/Get Power State
707h	F07h	SET_PWC / GET_PWC	Set/Get Pin Widget Control
708h	F08h	SET_UE / GET_UE	Set/Get Unsolicited Response Enable
-	F09h	-	Get Pin Sense
71Ch	-	SET_CD0	Set Configuration Default Byte 0
71Dh	-	SET_CD1	Set Configuration Default Byte 1
71Eh	-	SET_CD2	Set Configuration Default Byte 2
71Fh	-	SET_CD3	Set Configuration Default Byte 3
-	F1Ch	GET_CD	Get Configuration Default
-	F2Eh	GET_HDIS	Get HDMI/DP Info Size
733h	F33h	SET_PC / GET_PC	Set/Get Protection Control
734h	F34h	SET_CCM / GET_CCM	Set/Get Converter Channel Map
73Ch	F3Ch	SET_STRID / GET_STRID	Set/Get Widi Stream ID
740h-743h	F40h-F43h	SET / GET_PTSOFFSET	Set/Get PTS Offset Byte 0, 1, 2, 3
750h-753h	F50h-F53h	SET / GET Widi Buf LowBase Addr	Set/Get Widi Lower base address Byte 0, 1, 2, 3
754h-757h	F54h-F57h	SET / GET Widi Buf Upper Base Addr	Set/Get Widi Upper base address Byte 0, 1, 2, 3
760h-763h	F60h-F63h	SET / GET Widi Buf size	Set/Get Widi Buf Size Byte 0, 1, 2, 3
764h	F64h	SET / GET WDE TFD Size	Set/Get WDE TFD Size
770h	F70h	SET/GET Widi PID	Set/Get Widi PID
780h-783h	F80h-F83h	SET/GET WDE/WFA Header	Set/Get WDE/WFA Header
784h	F84h	SET/GET WDE Program Number	Set/Get WDE Program number

Set Verb	Get Verb	Symbol	Verb Name
785h-786h	F85h-F86h	SET/GET WDE STD Bit Rate	Set/Get WDE STD bit rate
787h	F87h	SET/GET WDE PKT PER FRM	Set/Get Packets per frame
788h	F88h	SET/GET Wireless Widget Mode	Set/Get Wireless Widget Mode (Wigig, Widi, WFA)

3h SET_AM - Set Amplifier Mute

Bits	Bits	Description
15	0	Set Output Amp (SOA):
14	0	Set Input Amp (SIA):
13	0	Set Left Amp (SLA):
12	0	Set Right Amp (SRA):
11:08	0h	Index (IDX):
07	1	Mute (MUTE): When set, amp muted.
06:00	0	<i>Reserved</i>

B8h GET_AM - Get Amplifier Mute

Bits	Bits	Description
31:08	0	<i>Reserved</i>
07	1	Mute (MUTE): When set, amp muted.
06:00	0	<i>Reserved</i>

F00h Get Parameters

Parameter	Symbol	Register Name
09h	PARAM_AWC	Audio Widget Capabilities
0Ch	PARAM_PC	Pin Capabilities
0Eh	PARAM_CLL	Connection List Length
12h	PARAM_OAC	Output Amplifier Capabilities
0Fh	PARAM_PS	Supported Power States

Parameter 09h: AWC - Audio Widget Capabilities

Bits	Reset	Description
31:24	0	Reserved
23:20	4h	Widget Type (TYPE): Indicates this is a pin complex widget
19:16	0	Sample Delay in Widget (DELAY): No delay through the pin widget.
15:13	011	Channel Count Extension (CCE): This field, combined with STRO, indicate 8 channels supported.
12	0	Content Protection Capable (CPC): Indicates pin capable of content protection.
11	0	L-R Swap (LRS): Indicates no left/right channel swap.
10	1	Power Control (PC): Indicates power state control
09	1	Digital (DIG): Indicates support for digital streams.
08	1	Connection List (CL): Indicates a connection list
07	1	Unsolicited Capable (UC): Indicates support for unsolicited responses.
06	0	Processing Widget (PW): Indicates no support for processing
05	0	Stripe (STRP): Indicates striping not supported.
04	0	Format Override (FO): Indicates no support for formatting
03	1	Amp Parameter Override (APO): Indicates no amplifier override support.
02	1	Out Amp Present (OAP): Indicates no output amplifier present.
01	0	In Amp Present (IAP): Indicates no input amplifier present.
00	1	Stereo (STRO): Indicates a stereo widget

Parameter 0Ch: PC - Pin Capabilities

Bits	Reset	Description
31:28	0	Reserved
27	1	High Bit Rate (HBR): Indicates support for high bit-rate audio
26	0	Reserved
25	0	MST capable (DP1.2)
24	1	DisplayPort (DP): Indicates support for DisplayPort
23:08	0	Reserved
07	1	HDMI (HDMI): Indicates support for HDMI
06:05	0	Reserved
04	1	Output Capable (OC): Pin is output capable
03	0	Reserved
02	1	Presence Detect Capable (PDC): Indicates capability for presence detection
01:00	0	Reserved

Parameter 0Eh: CLL - Connection List Length

Bits	Reset	Description
31:08	0	Reserved
07	0	Long Form (LF): Indicates connection list is short form
06:00	03h	Length (LEN): Indicates there is one item in the connection list.

Parameter 12h: OAC - Output Amplifier Capabilities

Bits	Reset	Description
31	1	Mute Capable (MC): Muting is capable on this pin
30:00	0	Reserved



Parameter 0Fh: PARAM_SPS - Supported Power States

Bit	Reset	Description
31	1	Extended Power State Supported (EPSS): Indicates support for low power states
30:04	0	<i>Reserved</i>
03	1	D3 Supported (D3S): Indicates support for D3.
02	0	D2 Supported (D2S): Indicates no support for D2.
01	0	D1 Supported (D1S): Indicates no support for D1.
00	1	D0 Supported (D0S): Indicates support for D0.

701hF01h SETGET_CSC - SetGet Connection Select Control

Bits	Reset	Description
07:00	00h	Connection Select Control (CSC):

F02h GET_CLE - Get Connection List Entry

Bits	Reset	Description
31:24	0	<i>Reserved</i>
23:16	100	Connection List Entry (CLE): Hardwired to Converter ID 4
15:08	011	Connection List Entry (CLE): Hardwired to Converter ID 3
07:00	010	Connection List Entry (CLE): Hardwired to Converter ID 2

705h SET_PS - Set Power State

Bits	Description
07:02	<i>Reserved</i>
01:00	Requested Power State (RPS): Only D0 (00) and D3 (11) may be requested

F05h GET_PS - Get Power State

Bits	Reset	Description
31:11	0	Reserved
10	0	Settings Reset (SR): Default values will not be changed. This bit will report 0 in all cases.
09	0	Clock Stop OK (CSOK): Clock stopping in D3 is not OK
08	0	Error (ERR): No error will ever be reported.
07:06	0	Reserved
05:04	11	Actual Power State (APS): Indicates the current power state of the node.
03:02	0	Reserved
01:00	11	Requested Power State (CPS): Reflects value written with SET_PS verb.

707hF07h SETGET_PWC - SetGet Pin Widget Control

Bits	Reset	Description
07	0	Reserved
06	1	Out Enable (OE): When set, the audio is enabled
05:02	0	Reserved
01:00	00	Encoded Packet Type (EPT):

708hF08h SETGET_UE - SetGet Unsolicited Enable

Bits	Description
07	Unsolicited Enable (UE): When set, unsolicited responses are allowed
06	Reserved
05:00	Tag (TAG):



F09h GET_PS - Get Pin Sense

Bits	Reset	Description
31	0	Presence Detect (PD): When set presence is detected on this pin.
30	0	ELD Value (ELDV):
29	0	Inactive (INA):
28:00	28:00	Reserved

71Ch SET_CD0 - Set Configuration Default Byte 0

Bits	Description
07:04	Default Association (DA):
03:00	Sequence (SEQ):

71Dh SET_CD1 - Set Configuration Default Byte 1

Bits	Description
07:04	Color (COL):
03:00	Miscellaneous (MISC):

71Eh SET_CD2 - Set Configuration Default Byte 2

Bits	Description
07:04	Default Device (DD):
03:00	Connection Type (CT):

71Fh SET_CD3 - Set Configuration Default Byte 3

Bits	Description
07:06	Port Connectivity (PC): External connectivity of the pin complex. <ul style="list-style-type: none">• 00 = Connected to jack• 01 = No physical connection• 10 = Fixed function device (integrated speaker, mic, etc.)• 11 = Both a jack and internal connection
05:00	Location (LOC):

Bits	Description											
	Bits 3:0											
Bits 5:4	0h: N/A	1h: Rear	2h: Front	3h: Left	4h: Right	5h: Top	6h: Bottom	7h: Special	8h: Special	9h: Special	Ah-Fh Reserved	
00: External	Y	Y	Y	Y	Y	Y	Y	Y	Y			
01: Internal	Y							Y	Y	Y		
10: Separate Chassis	Y	Y	Y	Y	Y	Y	Y					
11: Other	Y						Y	Y	Y			

F1Ch GET_CD - Get Configuration Default

Bits	Description
31:30	Port Connectivity (PC): See SET_CD3.PC
29:24	Location (L): See SET_CD3.L
23:20	Default Device (DD): See Set_CD2.DD
19:16	Connection Type (CT): See Set_CD2.CT
15:12	Color (COL): See SET_CD1.COL
11:08	Miscellaneous (MISC): See SET_CD1.MISC
07:04	Default Association (DA): See SET_CD0.DA
03:00	Sequence (SEQ): See SET_CD0.SEQ

F2Eh WIDI ELD Size

Bits	Reset	Description
31:08	0	Reset
07:00	Varies	Size (SZ) Reserved as we do not have Info frame support for WIdi widget. Will return the value 53h as the response.

F2Fh Get ELD Data

Parameter	Symbol	Register Name
07:0h	PARAM_INDX	ELD DATA Index



Parameter nn: ELD Data

Bits	Reset	Description
31:00	0	ELD Data

733h SET_PC - Set Protection Control

Bits	Description
07:03	Unsolicited Response Sub Tag (URST): Subtag to use for unsolicited responded.
02	Reserved
01:00	Content Protection Requested (CPR): <ul style="list-style-type: none">• 00 = Don't Care• 01 = Reserved• 10 = protection OFF• 11 = Protection ON

734hF34h SETGET_CCM - GetSet Converter Channel Map

Bits	Reset	Description
07:04	0h	Converter Channel (CC):
03:00	0h	Slot (SN):

73ChF3Ch SETGET_STRID - SetGet WIDI Stream ID

Bits	Reset	Description
07:00	00h	WDSTRMID (WDSTRMID): Represents the SSID that will go in the TS HEADER of PES and DATA packets

740hF40h SETGET PTS Offset Byte0

Bits	Reset	Description
07:00	00h	PTS B0(PTSB0): Bits 7:0 of the 32 bit 2's complement offset value. Should be added to the PTS value before sending to the circular buffer in the PES packet. Get verb will give the 32 bits value

741hF41h SETGET PTS Offset Byte1

Bits	Reset	Description
07:00	00h	<p>PTS B1(PTSB1): Bits 15:8 of the 32 bit 2's complement offset value. Should be added to the PTS value before sending to the circular buffer in the PES packet.</p> <p>Get verb will give the 32 bits value</p>

742hF42h SETGET PTS Offset Byte2

Bits	Reset	Description
07:00	00h	<p>PTS B2(PTSB2): Bits 23:16 of the 32 bit 2's complement offset value. Should be added to the PTS value before sending to the circular buffer in the PES packet.</p> <p>Get verb will give the 32 bits value</p>

743hF43h SETGET PTS Offset Byte3

Bits	Reset	Description
07:00	00h	<p>PTS B3(PTSB3): Bits 31:24 of the 32 bit 2's complement offset value. Should be added to the PTS value before sending to the circular buffer in the PES packet.</p> <p>Get verb will give the 32 bits value</p>

750hF50h SETGET Widi Buffer Base Address Byte0

Bits	Reset	Description
07:00	00h	<p>BLADDR Byte0(BLADDR0): Bits 7: 0 of the 32 bits of the lower base address of the widi circular buffer. This is a physical address of the Stolen memory. Address is cacheline aligned.</p>

751hF51h SETGET Widi Buffer Base Address Byte1

Bits	Reset	Description
07:00	00h	<p>BLADDR Byte1(BLADDR1): Bits 15: 8 of the 32 bits of the lower base address of the widi circular buffer. This is a physical address of the Stolen memory. Address is cacheline aligned.</p>



752hF52h SETGET Widi Buffer Base Address Byte2

Bits	Reset	Description
07:00	00h	BLADDR Byte2(BLADDR2): Bits 23:16 of the 32 bits of the lower base address of the widi circular buffer. This is a physical address of the Stolen memory. Address is cacheline aligned.

753hF53h SETGET Widi Buffer Base Address Byte3

Bits	Reset	Description
07:00	00h	BLADDR Byte3(BLADDR3): Bits 31:24 of the 32 bits of the lower base address of the widi circular buffer. This is a physical address of the Stolen memory. Address is cacheline aligned.

754hF54h SETGET Widi Buffer Upper Base Address Byte0

Bits	Reset	Description
07:00	00h	BUADDR Byte0(BUADDR0): Bits 7: 0 of the 32 bits of the upper base address of the widi circular buffer. This is a physical address of the Stolen memory. Address is cacheline aligned.

755hF55h SETGET Widi Buffer Upper Base Address Byte1

Bits	Reset	Description
07:00	00h	BUADDR Byte1(BUADDR1): Bits 15:8 of the 32 bits of the upper base address of the widi circular buffer. This is a physical address of the Stolen memory. Address is cacheline aligned.

756hF56h SETGET Widi Buffer Upper Base Address Byte2

Bits	Reset	Description
07:00	00h	BUADDR Byte2(BUADDR2): Bits 23:16 of the 32 bits of the Upper base address of the widi circular buffer. This is a physical address of the Stolen memory. Address is cacheline aligned.

757hF57h SETGET Widi Buffer Upper Base Address Byte3

Bits	Reset	Description
07:00	00h	BUADDR Byte3(BUADDR3): Bits 31:24 of the 32 bits of the Upper base address of the widi circular buffer. This is a physical address of the Stolen memory. Address is cacheline aligned.

760hF60h SETGET Widi Buffer Size Byte0

Bits	Reset	Description
07:00	00h	<p>BSIZE Byte0(BSIZE0): Bits 7: 0 of the 32 bits of the size of the widi circular buffer. Size is in number of cachelines. It should be multiple of 3 Cachelines. Total buffer size cannot exceed more than 4MBytes</p> <p>For Wigig mode this represents the number of TFDs in the buffer.</p>

761hF61h SETGET Widi Buffer Size Byte1

Bits	Reset	Description
07:00	00h	<p>BSIZE Byte1(BSIZE1): Bits 15:8 of the 32 bits of the size of the widi circular buffer. Size is in number of cachelines. It should be multiple of 3 Cachelines. Total buffer size cannot exceed more than 4MBytes.</p> <p>For Wigig mode this represents the number of TFDs in the buffer.</p>

762hF62h SETGET Widi Buffer Size Byte2

Bits	Reset	Description
07:00	00h	<p>BSIZE Byte2(BSIZE2): Bits 23:16 of the 32 bits of the size of the widi circular buffer. Size is in number of cachelines. It should be multiple of 3 Cachelines. Total buffer size cannot exceed more than 4MBytes</p> <p>For Wigig mode this represents the number of TFDs in the buffer.</p>

763hF63h SETGET Widi Buffer Size Byte3

Bits	Reset	Description
07:00	00h	<p>BSIZE Byte3(BSIZE3): Bits 31:24 of the 32 bits of the size of the widi circular buffer. Size is in number of cachelines. It should be multiple of 3 Cachelines. Total buffer size cannot exceed more than 4MBytes.</p> <p>For Wigig mode this represents the number of TFDs in the buffer.</p>

764hF64h SETGET WDE TFD size

764h: SET WDE TFD Size

Bits	Default	Description
07:04	00000	Reserved
03	0	This is an enable. If set, the tfd size is value is taken from here. Otherwise the tfd size is taken from WD_CNTRL register of MMIO.
02:00	000	<p>This is TFD size. Valid only for WiGig mode. The total circular buffer should be a multiple of this number.</p> <p>000b 512B [Default] default size of TFD is 512B</p> <p>001b 1024B TFD Size is 1024B</p> <p>010b 2048B TFD Size is 2048B</p> <p>011b 4096B TFD Size is 4096B</p> <p>100b 8192B TFD Size is 8192B</p> <p>Others: Reserved</p>

F64h: GET WDE TFD Size

Bits	Default	Description
31:04	0	Reserved
03	0	This is an enable. If set, the tfd size is value is taken from here. Otherwise the tfd size is taken from WD_CNTRL register of MMIO.
02:00	000	<p>This is TFD size. Valid only for WiGig mode. The total circular buffer should be a multiple of this number.</p> <p>000b 512B [Default] default size of TFD is 512B</p> <p>001b 1024B TFD Size is 1024B</p> <p>010b 2048B TFD Size is 2048B</p> <p>011b 4096B TFD Size is 4096B</p> <p>100b 8192B TFD Size is 8192B</p> <p>Others: Reserved</p>

770hF70h SETGET Widi Packet ID

Bits	Reset	Description
07:00	00h	PID byte0 (PID0): This goes in the bits 7:0 of the PID.

771hF71h SETGET Widi Packet ID

Bits	Reset	Description
07:05	00h	Reserved: MBZ
04:00	00h	PID byte1 These bits are bits 12:8 of the PID

780h SET WFAWDE Header

780h: SET WFA Header (WFA mode)

Bits	Reset	Description
07:00	00h	WFA HB0(WFAHB0): WFA Header Byte zero. Valid only in WFA mode. Otherwise behavior is unexpected

F80h: GET WFA Header (WFA mode)

Bits	Description
31:24	WFAHB3: Header Byte 3. Valid only when WFA mode is enabled
23:16	WFAHB2: Header Byte 2. Valid only when WFA mode is enabled
15:08	WFAHB1: Header Byte 1. Valid only when WFA mode is enabled
07:00	WFAHB0: Header Byte 0. Valid only when WFA mode is enabled

780h: SET WDE Header (Wigig Mode)

Bits	Reset	Description
07:04	00h	Format Type (FT): Indicates the format to use. These are the CT bits as defined in Display Port 1.2. The most common value will be 0001 => PCM.
03	0	High Bit Rate (HBR): Indicates whether the stream is HBR or not
02:00	000	Channel Count (CC): Indicates how many channels there are. The encoding is as follows: 000 Mono 001 2 Channel Stereo

Bits	Reset	Description
		010 3 Channels
		011 4 Channels
		100 5 Channels
		101 6 Channels
		110 7 Channels
		111 8 Channels

F80h: GET WDE Header (Wigig Mode)

Bits	Description
31:24	Reserved
23:16	Channel Allocation (CA): How are the channels allocated (L/R/etc.)
15	Reserved
14:12	Sampling Frequency (SF): Indicates the sampling frequency. The encoding is as follows: 000 - Reserved 001 - 32KHz 010 - 44.1KHz 011 - 48KHz 100 - 88.2KHz 101 - 96KHz 110 - 176.4KHz 111 - 192KHz
11:10	Reserved
09:08	Sampling length (SL): Indicates the sampling length. The encoding is as follows: 00 - Reserved (or 32 bits) 01 - 16bit 10 - 20bit 11 - 24bit
07:04	Format Type (FT): Indicates the format to use. These are the CT bits as defined in Display Port 1.2. The most common value will be 0001 => PCM.
03	High Bit Rate (HBR): Indicates whether the stream is HBR or not
02:00	Channel Count (CC): Indicates how many channels there are. The encoding is as

Bits	Description
	follows: 000 - Mono 001- 2 Channel Stereo 010 - 3 Channels 011 - 4 Channels 100 - 5 Channels 101 - 6 Channels 110 - 7 Channels 111 - 8 Channels

781h SET WFAWDE Header

781h: SET WFA Header (WFA Mode)

Bits	Reset	Description
07:00	00h	WFA HB1(WFAHB1) : WFA Header Byte One. Valid only in WFA mode. Otherwise behavior is unexpected

F81h: GET WFA Header (WFA Mode)

Bits	Description
31:24	WFAHB3 : Header Byte 3. Valid only when WFA mode is enabled
23:16	WFAHB2 : Header Byte 2. Valid only when WFA mode is enabled
15:08	WFAHB1 : Header Byte 1. Valid only when WFA mode is enabled
07:00	WFAHB0 : Header Byte 0. Valid only when WFA mode is enabled

781h: SET WDE Header (Wigig Mode)

Bits	Reset	Description
07	00h	Reserved
06:04	000	Sampling Frequency (SF): Indicates the sampling frequency. The encoding is as follows: 000 - Reserved 001 - 32KHz 010 - 44.1KHz

Bits	Reset	Description
		011 - 48KHz 100 - 88.2KHz 101 - 96KHz 110 - 176.4KHz 111 - 192KHz
03:02	00	Reserved
01:00	00	Sampling length (SL): Indicates the sampling length. The encoding is as follows: 00 - Reserved (32 bits) 01 - 16bit 10 - 20bit 11 - 24bit

F81h: GET WDE Header (Wigig Mode)

Bits	Description
31:24	Reserved
23:16	Channel Allocation (CA): How are the channels allocated (L/R/etc.)
15	Reserved
14:12	Sampling Frequency (SF): Indicates the sampling frequency. The encoding is as follows: 000 - Reserved 001 - 32KHz 010 - 44.1KHz 011 - 48KHz 100 - 88.2KHz 101 - 96KHz 110 - 176.4KHz 111 - 192KHz
11:10	Reserved
09:08	Sampling length (SL): Indicates the sampling length. The encoding is as follows: 00 - Reserved (32 bits)

Bits	Description
	01 - 16bit 10 - 20bit 11 - 24bit
07:04	Format Type (FT): Indicates the format to use. These are the CT bits as defined in Display Port 1.2. The most common value will be 0001 => PCM.
03	High Bit Rate (HBR): Indicates whether the stream is HBR or not
02:00	Channel Count (CC): Indicates how many channels there are. The encoding is as follows: 000 - Mono 001- 2 Channel Stereo 010 - 3 Channels 011 - 4 Channels 100 - 5 Channels 101 - 6 Channels 110 - 7 Channels 111 - 8 Channels

782h SET WFAWDE Header

782h: GET WFA Header (WFA Mode)

Bits	Reset	Description
07:00	00h	WFA HB2(WFAHB2) : WFA Header Byte two. Valid only in WFA mode. Otherwise behavior is unexpected

F82h: GET WFA Header (WFA Mode)

Bits	Description
31:24	WFAHB3 : Header Byte 3. Valid only when WFA mode is enabled
23:16	WFAHB2 : Header Byte 2. Valid only when WFA mode is enabled
15:08	WFAHB1 : Header Byte 1. Valid only when WFA mode is enabled
07:00	WFAHB0 : Header Byte 0. Valid only when WFA mode is enabled



782h: GET WDE Header (WiGig Mode)

Bits	Reset	Description
07:00	00h	Channel Allocation (CA): How are the channels allocated (L/R/etc.)

F82h: GET WDE Header (Wigig Mode)

Bits	Description
31:24	Reserved
23:16	Channel Allocation (CA): How are the channels allocated (L/R/etc.)
15	Reserved
14:12	Sampling Frequency (SF): Indicates the sampling frequency. The encoding is as follows: 000 - Reserved 001 - 32KHz 010 - 44.1KHz 011 - 48KHz 100 - 88.2KHz 101 - 96KHz 110 - 176.4KHz 111 - 192KHz
11:10	Reserved
09:08	Sampling length (SL): Indicates the sampling length. The encoding is as follows: 00 - Reserved (or 32 bits) 01 - 16bit 10 - 20bit 11 - 24bit
07:04	Format Type (FT): Indicates the format to use. These are the CT bits as defined in Display Port 1.2. The most common value will be 0001 => PCM.
03	High Bit Rate (HBR): Indicates whether the stream is HBR or not
02:00	Channel Count (CC): Indicates how many channels there are. The encoding is as follows: 000 - Mono 001- 2 Channel Stereo

Bits	Description
	010 - 3 Channels
	011 - 4 Channels
	100 - 5 Channels
	101 - 6 Channels
	110 - 7 Channels
	111 - 8 Channels

783h SET WFA Header

Bits	Reset	Description
07:00	00h	WFA HB3(WFAHB3): WFA Header Byte Three. Valid only in WFA mode. Otherwise behavior is unexpected

F83h: GET WFA Header

Bits	Description
31:24	WFAHB3: Header Byte 3. Valid only when WFA mode is enabled
23:16	WFAHB2: Header Byte 2. Valid only when WFA mode is enabled
15:08	WFAHB1: Header Byte 1. Valid only when WFA mode is enabled
07:00	WFAHB0: Header Byte 0. Valid only when WFA mode is enabled

784hF84h GETSET WDE Program Number Verb

Bits	Bits	Description
07:00	0	Program Number

785hF85h GETSET WDE STD Bit Rate Verb

785h: SET WDE STD Bit Rate

Bits	Bits	Description
07:00	0	Lower byte of WDE STD Bit Rate



F85h: GET WDE STD Bit Rate

Bits	Bits	Description
31:16	0	Reserved
15:08	0	Upper byte of WDE STD Bit Rate
07:00	0	Lower byte of WDE STD Bit Rate

786hF86h GETSET WDE STD Bit Rate Verb

786h: SET WDE STD Bit Rate

Bits	Bits	Description
07:00	0	Upper byte of WDE STD Bit Rate

F86h: GET WDE STD Bit Rate

Bits	Bits	Description
31:16	0	Reserved
15:08	0	Upper byte of WDE STD Bit Rate
07:00	0	Lower byte of WDE STD Bit Rate

787hF87h GETSET WDE PKT PER FRM Verb

787h: SET WDE PKT PER FRM

Bits	Default	Description
07:05	000	Reserved
04:00	00111	Valid only for WiGig mode. Packets per frame. Number of samples per packet are shown below. Legal values are: 00000: 1 packet = 192 samples per packet 00001: 2 packets = 96 samples per packet 00011: 4 packets = 48 samples per packet 00101: 6 packets = 32 samples per packet 00111: 8 packets = 24 samples per packet 01011: 12 packets = 16 samples per packet

Bits	Default	Description
		01111: 16 packets = 12 samples per packet 10011: 24 packets = 8 samples per packet 11111: 32 packets = 6 samples per packet

F87h: GET WDE PKT PER FRM

Bits	Default	Description
31:05	0	Reserved
04:00	00111	Packets per frame. Valid for Wigig mode only.

788hF88h GETSET Wireless Widget Mode

788h: SET Wireless Widget Mode

Bits	Default	Description
07:02	000	Reserved
01:00	00	Mode Selection 00: WiGig Mode 01: Widi Mode 10: WFA Mode 11: Reserved

F88h: GET Wireless Widget Mode Response data format

Bits	Default	Description
31:02	0	Reserved
01:00	00	Mode Selection 00: WiGig Mode 01: Widi Mode 10: WFA Mode 11: Reserved



North Display Engine Registers

This chapter contains the register descriptions for the display engine portion of a family of graphics devices.

These registers vary by devices within the family of devices, so special attention needs to be paid to which devices use which registers and register fields.

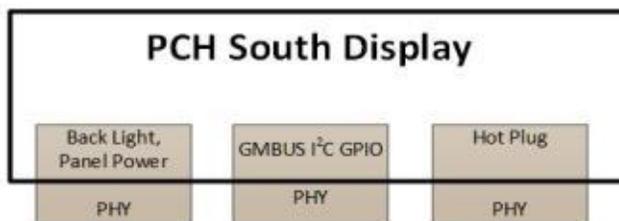
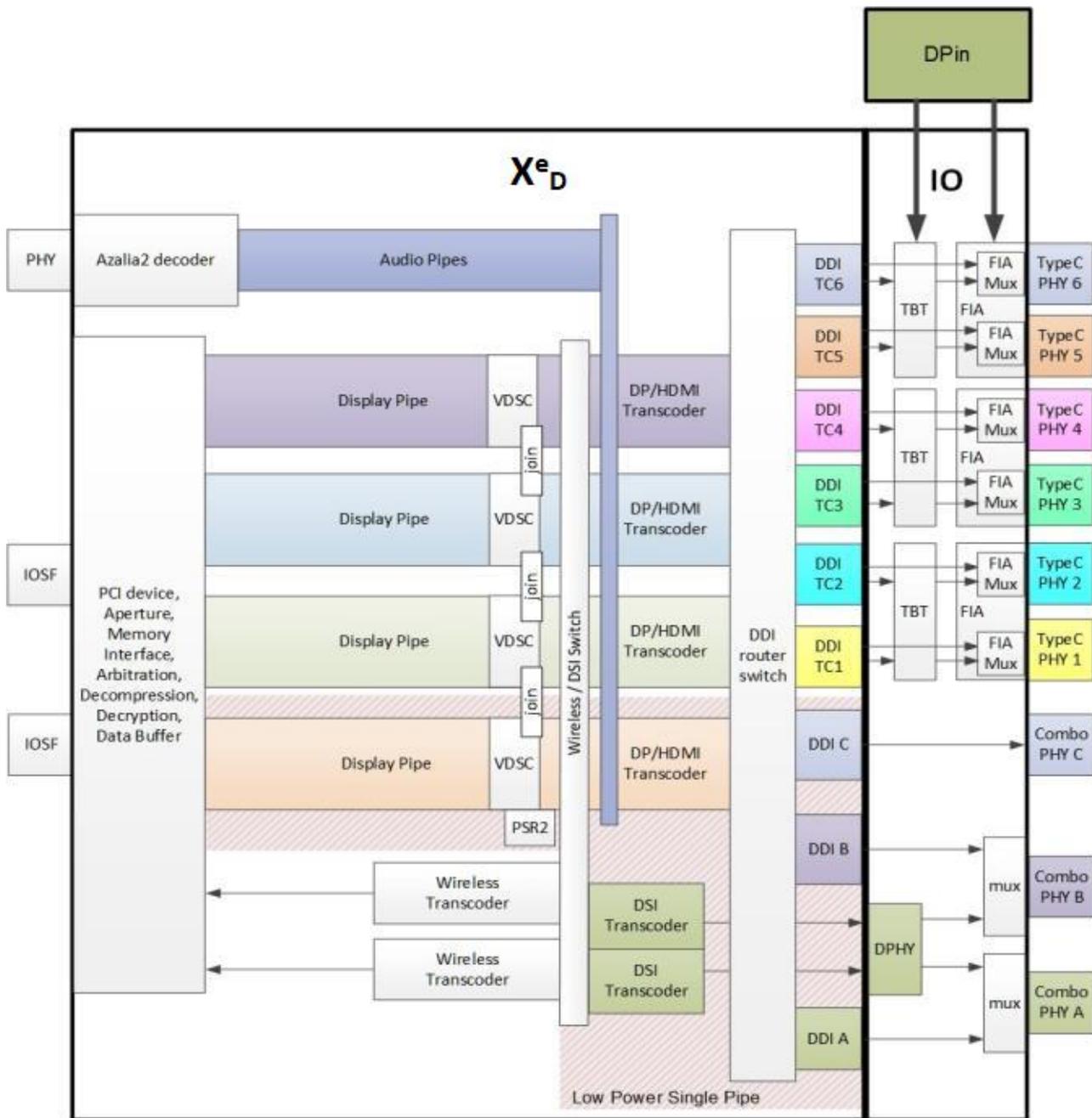
Different devices within the family may add, modify, or delete registers or register fields relative to another device in the same family based on the supported functions of that device.

X_eD Overview

Genesis

This project uses X_eD.

Block Diagram



Some SKUs may limit the number of PHYs connected at the package.



The front end of the display contains the pipes. The pipes connect to the transcoders. The transcoders, except for wireless, connect to the DDIs to drive the IO/PHY. Wireless writes back to memory. DPin allows external graphics DisplayPort output to be routed to typeC and thunderbolt.

General Capabilities

Four simultaneous displays (pipes A, B, C, D)

- 7 planes and 1 cursor per pipe
- Audio streams per pipe to go to external ports
- HDR support for 3 planes per pipe
- VESA DSC compression support for all pipes
- Post-DSC joining to connect 2 adjacent pipes for resolutions that require more bandwidth than one pipe can support
- Pipe A optimized for low power

External display connections

- 2 wireless
- 3 combo ports (DisplayPort, HDMI, or eDP)
- 6 USB Type C ports (DisplayPort alternate mode, DisplayPort over thunderbolt, native DisplayPort on legacy connector, native HDMI on legacy connector)
- Hotplug for type C
- AUX channels for DisplayPorts
- Multi-stream support for DisplayPorts
- Genlock support on all combo ports at display IP level but not at SoC level. Feature cannot be used.

Embedded/local display connections

- 3 combo ports that can support eDP, DSI, and external ports, but only one low power optimized port at a time
- 1 x8 MIPI DSI or 2 x4 MIPI DSI (shared pins on DDIA and DDIB)
- Combo IO shares pins between DSI0 and DDIA and between DSI1 and DDIB
- AUX channel for eDP
- PSR1, PSR2, and MSO (multi-segmented operation, chip on glass) for eDP

PCH

- Backlight modulation PWM, panel power sequencing, GMBUS I2C, non-typeC hot plug detection

Port Configurations

Port Type	# of ports ¹	Protocol	Maximum Speed	Maximum Resolution ²
Combo	1 port 8 lanes or 2 ports 4 lanes	MIPI DSI 1.3	2.5 Gbps	5k 60Hz compressed
"	3 ³	eDP 1.4b	HBR3 8.1 Gbps	5k 60Hz 24bpp
"	3	DP 1.4a	HBR2 5.4 Gbps	4k 60Hz 24bpp
"	3	HDMI 1.4	2.97 Gbps	4k 30Hz 24bpp
"	3	HDMI 2.0b	5.94 Gbps	4k 60Hz 24bpp
USB Type-C ⁴	6	DP 1.4a	HBR3 8.1 Gbps	8k 60Hz compressed, 5k 120Hz compressed
"	6	HDMI 1.4	2.97 Gbps	4k 30Hz 24bpp
"	6	HDMI 2.0b	5.94 Gbps	4k 60Hz 24bpp
Capture	2	N/A	N/A	Single display 4k 60Hz 30bpp Dual display 2560x1600 60Hz 30bpp

¹ Many SKUs will limit the number of ports connected at the package. Only one power optimized port for eDP.

² Resolution restrictions

- Resolutions supported, but not a guarantee of user experience.
- Thermal constraints may apply.
- Multi-display resolution support limited.
- Subject to memory bandwidth availability. Software configuration constrained to fit available bandwidth.
- These are single cable panel (non-tiled panel) resolutions. Panels with two cables may support higher resolution by using two ports.
- Higher bits per color may be possible at lower resolutions.
- Some resolutions require compression (VDSC). VDSC input and output bits per color are limited.
- Supports other resolutions and bits per color that fit the same bandwidth and size constraints.
- Software configuration constrained to fit available bandwidth with current memory configuration.

³ Only one power optimized port for eDP.

⁴ USB type-C PHY can support DP alternate mode, DP tunneled over thunderbolt, DP with legacy connector, or HDMI with legacy connector.



Multiple Display Resolution Support

This table gives the required resolution support. Actual results may have higher resolution, depending on exact details of the configuration, such as the number of planes enabled and pixel formats.

Number of simultaneous displays	Maximum Common Resolution
1	8k 60Hz HDR or 5k 120Hz HDR
2	5k 60Hz SDR or 4k 60Hz HDR
3 or 4	4k 60Hz HDR

All subject to resolution restrictions noted under the Port Configurations table.

HDR (High Dynamic Range) increases memory bandwidth compared to standard dynamic range (SDR).

Software does not have explicit restrictions for multiple display cases, just constraints to fit available bandwidth with current memory configuration.

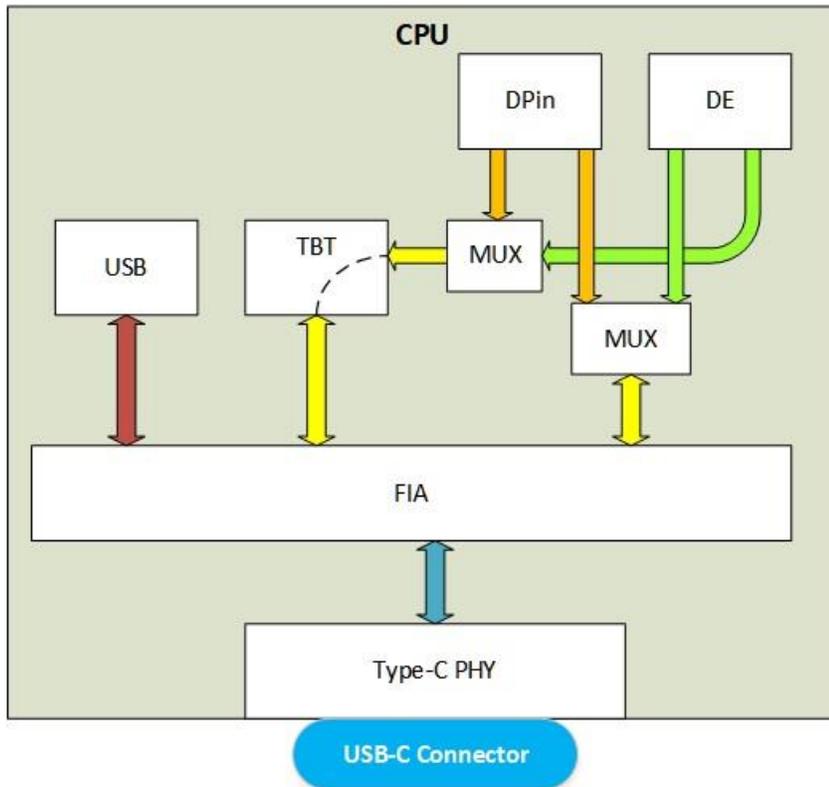
See the Resolution Support page for detailed restrictions and calculations.

Port Availability

Port	Usage	Capability	Alternate Names
DDI A	Combo port A	eDP, DP, HDMI, MIPI DSI 0	DDIA, Port A, Combo Port A, MIPIA, MIPI0
DDI B	Combo port B	eDP, DP, HDMI, MIPI DSI 1	DDIB, Port B, Combo Port B, MIPIB, MIPI1
DDI C	Combo port C	eDP, DP, HDMI	DDIC, Port C, Combo Port C
TC 1	Type C port 1	DP alternate mode, DP over Thunderbolt, native/legacy DP, native/legacy HDMI	Type C Port 1, TC1, USBC1
TC 2	Type C port 2	DP alternate mode, DP over Thunderbolt, native/legacy DP, native/legacy HDMI	Type C Port 2, TC2, USBC2
TC 3	Type C port 3	DP alternate mode, DP over Thunderbolt, native/legacy DP, native/legacy HDMI	Type C Port 3, TC3, USBC3
TC 4	Type C port 4	DP alternate mode, DP over Thunderbolt, native/legacy DP, native/legacy HDMI	Type C Port 4, TC4, USBC4
TC 5	Type C port 5	DP alternate mode, DP over Thunderbolt, native/legacy DP, native/legacy HDMI	Type C Port 5, TC5, USBC5
TC 6	Type C port 6	DP alternate mode, DP over Thunderbolt, native/legacy DP, native/legacy HDMI	Type C Port 6, TC6, USBC6

Some SKUs will limit which ports are connected in the die and package. Software should rely on hotplug to determine which ports are actually available.

USB TypeC



The USB type C ports are integrated into the CPU. Each type C port is shared between USB, thunderbolt (TBT), DPIn, and display engine.

FIA selects between the controllers and handles cable orientation swapping. Multiple controllers can share the connector simultaneously, but with reduced number of lanes for each.

Display engine has 4 output types for type C

- DisplayPort alternate mode: Display engine sends DisplayPort data to FIA to be muxed to the correct PHY lanes. DP can use 2 or 4 of the data lanes, depending on whether the connector is being shared with another controller.
- DisplayPort over thunderbolt: Display engine sends DisplayPort data to the thunderbolt controller, which combines up to 2 DisplayPort streams with other data streams and sends it to FIA to be muxed to the correct PHY lanes.
- Native HDMI: This is used for a legacy HDMI connector on a type C port. Display engine sends data to FIA to be passed through to the correct PHY lanes. There is no muxing with other controllers.
- Native DisplayPort: This is used for a legacy DisplayPort connector on a type C port. Display engine sends data to FIA to be passed through to the correct PHY lanes. There is no muxing with other controllers.

The type C sub-system IOM and power delivery controller determine the ownership of the type C connector. They interpret type C hotplug events and decide whether the hotplug should be routed to display engine or to one of the other controllers, and which lanes are available for DP alternate mode.



If the hotplug is directed to display engine it can take 3 paths

- Legacy hotplug interrupt: This interrupt comes through the south display hotplug detection and interrupt. This is used for the native HDMI or DP case with a legacy connector on a type C port.
- Thunderbolt hotplug interrupt: This interrupt comes through the north display thunderbolt hotplug control and interrupt. This is used for DP over thunderbolt.
- DP alternate hotplug interrupt: This interrupt comes through the north display typeC hotplug control and interrupt. This is used for DP alternate mode.

Depending on the type of interrupt, software needs to configure the display output to go to the correct destination.

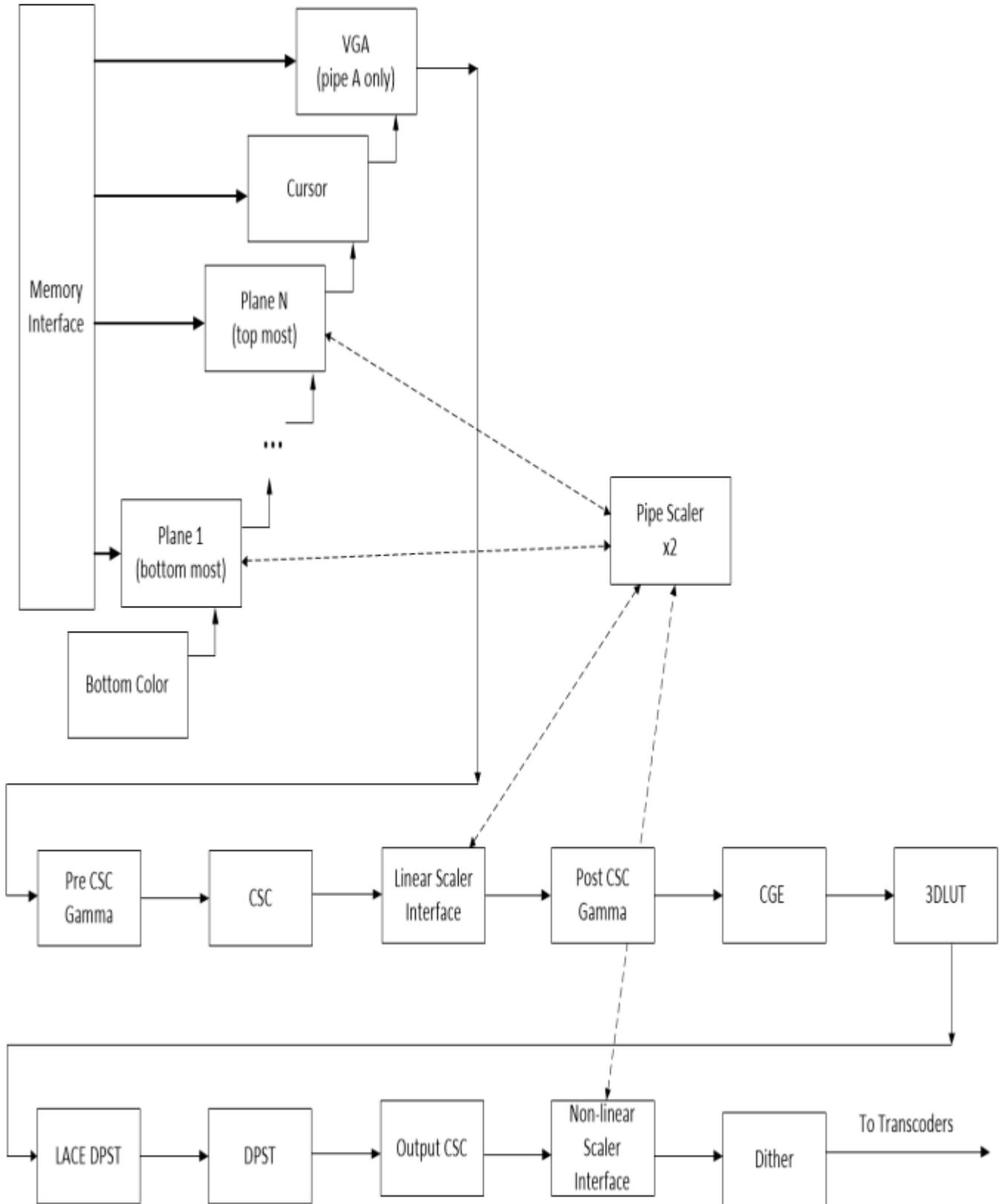
Power Wells

Display engine functions are spread across several different power gated (PG) wells that can be shut down to optimize power for different configurations.

Hardware can dynamically control some PGs for display power states. That is enabled through the DC State programming.

Refer to Sequences for Power Wells for more details on the functions in each power well and the sequences for enabling and disabling power.

Pipes





The display pipes contain the planes, blending, color measurement and adjustment, scaling, and dithering.

FBC and LDPST are supported only on pipe A.

3D LUT is supported only on pipes A and B.

The planes read data from memory, format it into pixels, and can apply color correction and scaling.

Each display pipe has 7 planes and a cursor.

The plane blending combines the output from all the planes following a fixed Z-order. Plane 1 is the bottom most plane and higher numbered planes stack on top of it. Cursor goes on top of all the planes.

Planes 1-3 support HDR. Planes 4-7 support SDR. Plane details are in the Universal Plane and Plane Capability and Interoperability sections.

The background color that is seen under the bottom most plane is programmable.

The blended pixels pass through several color correction functions and then output to the transcoders.

The pipes each have two pipe scalars. Each pipe scalar can be assigned to scale a plane or scale the blended output.

Transcoders

The display transcoders contain the timing generators, port encoders, DisplayPort transport control, Audio/Video mixers, Video Data Island Packet mixers, and Panel Self Refresh controllers.

Except for the WD (Wireless Display, Capture) transcoders, the transcoders convert pixel data to the appropriate format for the port, mix in data islands and audio, and output the data to the DDIs.

The WD transcoders convert pixel data to the appropriate format, then write it to system memory.

Transcoder A-D can support DP or HDMI, with audio.

Transcoders DSI* only support MIPI DSI.

Only transcoder A supports PSR2.

Only transcoder A supports eDP CoG.

The transcoders, except for wireless transcoders, can make use of VDSC compression.

Audio

The Azalia2 interface provides data to the audio codec.

The audio codec connects to the Audio/Video mixers in the transcoders.

The audio codec can also write data back to system memory for wireless audio.

DDIs (Digital Display Interfaces)

The DDIs contain port logic to interface to the DDI physical (PHY) layer.

The DDIs going to the combo PHY support lane reversal where the internal lane to package lane mapping is swapped. The DDIs going to the type C PHY rely on FIA to do lane reversal.

A Combo PHY capable of supporting DDI or MIPI DSI is attached to DDI A and DDI B. It can select between MIPI DSI output or DDI output. Dynamic switching between the two configurations is not supported.

Some of the DDI GPIO and HPD pins are reused for MIPI DSI panel controls.

DDI Lane Mapping:

Package Pin	DisplayPort Non-Reversed	DisplayPort Reversed	HDMI/DVI Non-Reversed	HDMI/DVI Reversed
DDI data 3	Main Link Lane 3	Main Link Lane 0	TMDS Clock	TMDS Data2
DDI data 2	Main Link Lane 2	Main Link Lane 1	TMDS Data0	TMDS Data1
DDI data 1	Main Link Lane 1	Main Link Lane 2	TMDS Data1	TMDS Data0
DDI data 0	Main Link Lane 0	Main Link Lane 3	TMDS Data2	TMDS Clock

MIPI DSI Lane Mapping:

Package Pin	MIPI DSI Connection
DDIA data 3	DSI 0 Data 3
DDIA data 2	DSI 0 Clock
DDIA data 1	DSI 0 Data 2
DDIA data 0	DSI 0 Data 1
DDIA AUX	DSI 0 Data 0
DDIB data 3	DSI 1 Data 3
DDIB data 2	DSI 1 Clock
DDIB data 1	DSI 1 Data 2
DDIB data 0	DSI 1 Data 1
DDIB AUX	DSI 1 Data 0

DDI and MIPI DSI Pin Sharing:

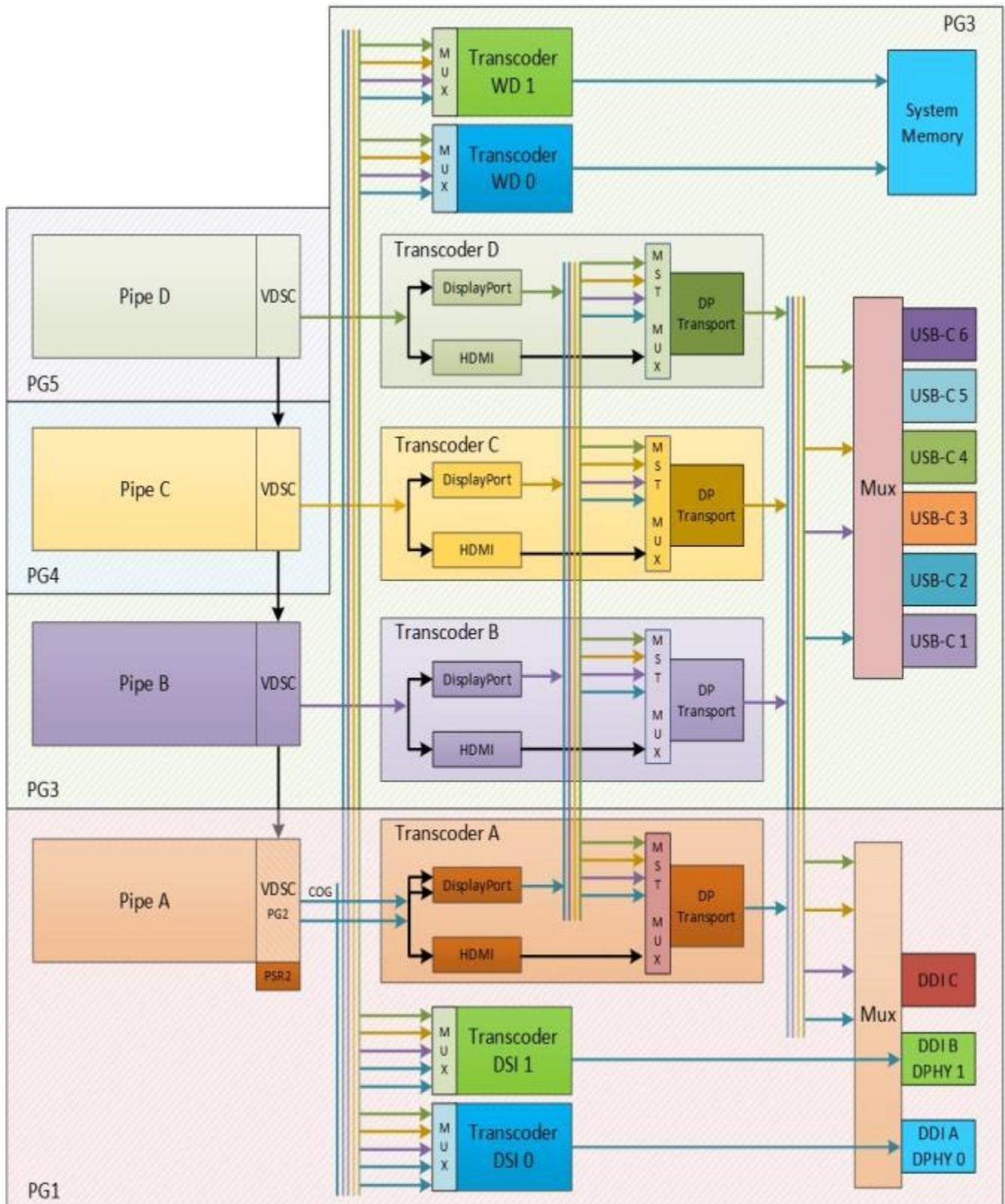
DDI pins	Non-MIPI DSI board	MIPI DSI 8 lane board	MIPI DSI 4 lane board
DDIA data and AUX	DDIA embedded DisplayPort	MIPI DSI 0	MIPI DSI 0
DDIB data and AUX	DDIB DisplayPort, HDMI, or DVI	MIPI DSI 1	DDIB DisplayPort, HDMI, or DVI
DDIA HPD	EDP HPD	MIPI panel RESET	MIPI panel RESET
DDIA DDC clock	Unused	MIPI panel AVEE	MIPI panel AVEE
DDIA DDC data	Unused	MIPI panel VIO	MIPI panel VIO



MIPI DSI Panel Pins Mapping to Display GPIO Pins

MIPI Panel Pin	GPIO Pin	Controls
RESET	DDSP_HPD_A	South display SHOTPLUG_CTL_DDI DDIA HPD Output Data
AVEE	DDPA_CTRLCLK	South display GPIO_CTL_1 (DDIA) clock
VIO	DDPA_CTRLDATA	South display GPIO_CTL_1 (DDIA) data
AVDD	L_VDDEN	South display panel power sequencing power enable
BKLTEN	L_BKLTEN	South display panel power sequencing backlight enable
PWM	L_BKLTCTL	South display backlight PWM output
DSI 0 TE	Utility Pin	UTIL_PIN_CTL
DSI 1 TE	DSI_DE_TE_2	Enabled automatically by DSI transcoder 1

Pipe to Transcoder to DDI Mappings





Twin modes are not supported. A pipe cannot drive more than one display.

A pipe cannot connect to more than one transcoder simultaneously.

With DisplayPort multistream it is possible to have multiple pipes/transcoders driving a single DDI. Multistream requires a transcoder to be enabled to drive the DP transport, even if the associated pipe streams are disabled.

Transcoders WD* and DSI* do not support multistream.

Two pipes can drive a single display through two transcoders and two DDIs that are joined in the panel (tiled display) or by using DSC compression and joining the pipe outputs to go to a single transcoder and DDI.

Transcoders A-D are tied to the respective Pipes A-D. Each pipe output can go to either the respective transcoder or to transcoders WD* or DSI*.

Pipe A VDSC outputs 2 streams to split the display for eDP Chip on Glass or MSO.

Transcoders A-D can connect to any DDI. Transcoders WD* can only go to system memory. Transcoders DSI* can connect only to DDIA and DDIB.

Mode Set

A mode set sequence is the programming sequence that must be followed when enabling or disabling output to a display. There are several different mode set sequences documented in the following sections. The sequence to use depends on which type of port is being enabled or disabled.

Sequences for Intel MIPI DSI IP

This folder contains the sequences for the Intel MIPI DSI IP.

Sequences for MIPI DSI

These sequences are used to enable and/or disable the new MIPI DSI ports and associated functions. Unless the sequence specifically talks about multi-port requirements (i.e. Dual Link mode) these sequences are for a given DSI port (i.e. either MIPI DSI port can be programmed with the below sequences).

Note that some of the registers outside of the DSI register space still refer to the old naming convention of MIPIA and MIPIC. The DSI ports are no longer associated with specific pipes and, so are referred to as DSI0 and DSI1. The mapping between the old naming convention and the new is: DSI0 = MIPIA; DSI1 = MIPIC

DSI Transcoder Enable Sequence

1. Initialize display (Sequences to Initialize Display)
2. Enable Power Wells
 - a. If Pipe A is not being used to drive MIPI DSI, then the power wells for those pipes will also need to be enabled (see Sequences for Power Wells)
3. Enable IO Power
 - a. Configure the mode of operation of the IO for MIPI DSI (via **DSI_IO_MODECTL**)
 - i. Note that setting the IO mode will assert the HS Word clock (a.k.a. the 1X clock) request to the IO. This will enable the IO to forward the HS Word clock back to the Display Engine
 - b. Enable **PWR_WELL_CTL_DDI** Power Requests for the DDI that will be used (DSI0 uses DDI A; DSI1 uses DDI B)
 - i. Note that the DPHY automatically powers up the AUX lane (DSI Data Lane 0) when it receives the power request with the IO mode set to MIPI DSI. Do **not** try to power on the AUX channel using the PWR_WELL_CTL_AUX register
 - c. Wait for PWR_WELL_CTL_DDI Power Status = Enabled
 - i. Timeout after 20us
4. Enable PLL
 - a. If PLL is not already enabled, follow Port Clock Programming enable sequence
 - b. Configure PLL to port mapping to direct the PLL output to the DDI
 - c. If not already done, gate the DDI clock associated with port being enabled (DPCLKA_CFGCR0)
 - i. If DSI 0 is being used, then set the DDIA Clock Off bit of DPCLKA_CFGCR0
 - ii. If DSI 1 is being used, then set the DDIB Clock Off bit of DPCLKA_CFGCR0
5. Configure and Enable DPHY
 - a. Configure PORT_CL_DW10 Static Power Down to power up all lanes of the DDI that will be used
 - i. See table below for the mapping of the DSI lanes to the Static Power Down bits
 - b. Configure the Lane sequencing of the Combo-PHY transmitters for MIPI DSI HS operation
 - i. Set the Loadgen Select (**PORT_TX_DW4**) for Transmit lanes 0, 1, and 3 to be secondary to the Aux lane (see Lane Sequence Programming table below)
 1. Note that Transmit lane 2 is the DSI Clock lane and it must be a primary lane (i.e. it cannot be secondary to any other lane) for proper operation
 - ii. Set the Latency Optimization (**PORT_TX_DW2** and **PORT_PCS_DW1**) for the Aux lane and Transmit lanes (see Lane Sequence Programming table below for values)
 1. The DSI Clock lane operates independent of the Data lanes, so there is no latency optimization needed

- c. Configure voltage swing and skew. Refer to DDI Buffer section for the programming not already specified above (i.e. the Loadgen Select and Latency Optimization should be used from here, all other DDI buffer programming can be taken from the DDI Buffer section).
 - i. Note that the voltage swing and skew programming is for the HS data only. The HW automatically handles the swing and skew for the LP data
 - d. Setup D-PHY timings
 - i. If SW is going to override any of the D-PHY timing parameters, then configure DPHY_CLK_TIMING_PARAM, DPHY_DATA_TIMING_PARAM, or DPHY_TA_TIMING_PARAM with the desired D-PHY timings. SW must also program the equivalent DSI_*_TIMING_PARAM registers with the appropriate overrides (see following step)
 - e. Program Tx Equalization for the port (**PORT_TX_DW0**), if necessary (determined by VBT)
 - i. Note that the programming should be applied across all lanes of the PHY, so Software can use the TXGRP offset to program all lanes at once.
 - f. If the link is operating at 1.5Gbps or below, then program the TX slew rate to 11b in **PORT_TX_DW1**, else program it to 0b0.
 - g. If the link is operating at 2.5Gbps, then program clock lane (lane 2) o_ldo_ref_sel_cri = 000011b in **PORT_TX_DW6_<lane 2 instance>**, else program it to 000000b.
 - h. Enable DDI Buffer and wait for DDI_BUF_CTL DDI Idle Status = 0b (Not Idle)
 - i. Timeout after 500us
6. Enable DSI Transcoder
- a. Setup link initialization/wakeup timings
 - i. Configure the DSI_T_INIT_PRIMARY with the Initialization time if different than the default 100us
 - ii. Configure the DSI_T_WAKEUP with the T_{WAKEUP} timing parameter if different than the default 1ms
 - iii. If Software overrode the transitional timing parameters for the clock or data lanes within the DPHY (i.e. DPHY_CLK_TIMING_PARAM/DPHY_DATA_TIMING_PARAM registers), then it must reflect those overrides within the DSI controller's timing parameter registers (**DSI_CLK_TIMING_PARAM**, or **DSI_DATA_TIMING_PARAM**)
 - b. Enable and configure the **UTIL_PIN_CTL** register as an input, if enabling DSI port 0 to be operating in Command Mode and the TE events will be received out-of-band
 - i. This IO is shared with Audio
 - c. Enable and configure UTIL2_PIN_CTL as an input, if enabling DSI port 1 to be operating in Command Mode and the TE events will be received out-of-band
 - i. This IO is shared with Genlock.
 - d. Configure DSI timeouts - **DSI_HTX_TO**, **DSI_CALIB_TO**, **DSI_LRX_H_TO**, **DSI_PWAIT_TO**, and **DSI_TA_TO**.
 - i. Note that the DSI_HTX_TO and DSI_CALIB_TO registers don't have to be programmed if Software is satisfied with the defaults.

- ii. The "Programming Reference" section below gives some guidelines on how the DSI_TA_TO and DSI_LRX_H_TO registers can be programmed.
 - e. Configure **TRANS_DSI_FUNC_CONF**
 - f. Configure **TRANS_DDI_FUNC_CTL2** Port Sync Mode Enable, if both DSI transcoders are going to operate in Dual Link mode
 - i. See "Dual Link Sequence" below for details
 - g. Configure and enable **TRANS_DDI_FUNC_CTL**
 - h. Wait for DSI Link Ready (TRANS_DSI_FUNC_CONF)
 - i. Timeout after 2X the value programmed in the DSI_T_WAKEUP register + 500us
 - ii. Note that the timeout duration is to account for the possibility that the DSI Link is in ULPS. When exiting ULPS the controller will have to serialize the wakeup of the Clock Lane and Data Lanes.
7. Program Panel
- a. Send a "Set Maximum Return Packet Size" DSI packet to the Panel. Refer to "Sending Commands to the Panel" section. This is a Short Packet format, only a Header credit will be needed to form the packet within the **DSI_CMD_TXHDR** register (with the Payload bit cleared). The packet is composed of:
 - i. Data Type: 0x37
 - ii. Virtual Channel: Panel specific
 - iii. Data: The Maximum Return Packet Size can be programmed for up to 32 bytes
 - b. Software will handle any additional panel setup here.
 - c. Software needs to ensure that all of the Panel programming has been dispatched before the Transcoder is enabled. To do this, the following sequence will need to be done:
 - i. Software will wait for all Header and Payload Credits to be released (DSI_CMD_TXCTL)
 - 1. Timeout after 100us
 - ii. Build a NOP DCS command using the DSI_CMD_TXHDR register
 - 1. The packet will use a Data Type of 0x05 (DCS Short Write - 0 Parameters)
 - 2. The Word Count/Parameter bytes will be set to all zeroes (NOP DCS command)
 - 3. Set the LPDT bit
 - iii. After loading the NOP with the DSI_CMD_TXHDR write, Software will wait for the following to be true:
 - 1. Wait for all Header Credits to be released
 - i. Timeout after 20us
 - 2. Wait for LP Tx in Progress bit of the **DSI_LP_MSG** register to be cleared
 - i. Timeout after 20us
 - iv. Software can now continue the enabling sequence
8. Enable Planes, Pipe, and Transcoder
- a. Configure and enable planes (VGA or hires)

- i. If VGA, clear VGA I/O register SR01 bit 5
 - ii. This can be done later, if desired
 - b. Enable Scaler, if needed (must be enabled for VGA)
 - c. Configure Transcoder Timings
 - i. Make sure to program TRANS_VBLANK appropriately
 - 1. The Vertical Blank End should always be programmed to be equal to the Vertical Total programming of TRANS_VTOTAL
 - 2. If the Display Stream Buffer (DSB) is not enabled, then the Vertical Blank Start should be programmed to be equal to the Vertical Active programming of TRANS_VTOTAL
 - 3. If the DSB is enabled, then the Vertical Blank Start may need to be pushed out
 - ii. If operating in Command Mode (a.k.a. DBI), then the Vertical and Horizontal totals should still be programmed even though the DSI transcoder is not truly maintaining these timings.
 - 1. Program the Horizontal Total to be Horizontal Active + 160
 - 2. Program the Vertical Total to include a Vertical blanking time that satisfies the following: $\text{ceiling}(400\mu\text{s} / \text{Line Time})$. See the "**Determining Vertical Blank in DBI**" section within the "Transcoder DSI Function" page for programming details.
 - iii. Software does not need to program the TRANS_*SYNC registers, if the DSI transcoder is operating in Command Mode
 - d. Configure and enable **TRANS_CONF**
 - i. Wait for the Transcoder State to be enabled, if additional Panel Commands need to be sent at this time
- 9. Turn on the backlight, if necessary

The following table shows the mapping of the Static Power Down bits within the PORT_CL_DW10 register and the PWR_WELL_CTL_DDI register to the physical I/O lanes used for a DSI transcoder's link.

Phys Uni-Dir Lane 3 (Bit 3)	Phys Uni-Dir Lane 2 (Bit 2)	Phys Uni-Dir Lane 1 (Bit 1)	Phys Uni-Dir Lane 0 (Bit 0)	Phys Bi-Dir Lane (PWR_WELL_CTL_DDI)	Notes
MIPI DSI Usage					
Data Lane 3	Clock Lane	Data Lane 2	Data Lane 1	Data Lane 0	Lane mapping
OFF	ON	OFF	OFF	ON	x1 DSI
OFF	ON	OFF	ON	ON	x2 DSI
OFF	ON	ON	ON	ON	x3 DSI
ON	ON	ON	ON	ON	x4 DSI

Lane Sequence Programming

AFE Lane	DSI Lane	Latency Optimization				Load Generation Select	
		PORT_TX_DW2		PORT_PCS_DW1		PORT_TX_DW4	
		Bits	Value ⁽¹⁾	Bits	Value ⁽²⁾	Bits	Value
Aux	Data Lane 0	[10:8]	'h5	[3:2]	'h0	[31]	0
0	Data Lane 1		'h5		'h1		1
1	Data Lane 2		'h5		'h1		1
2	Clock Lane		'h0		'h1		0
3	Data Lane 3		'h5		'h1		1

1. The Latency Optimization of the Clock Lane can be either left at its default value ('h0) or programmed to the same value as the other lanes. If programmed with the same value as the other lanes, then the Group access can be used for PORT_TX_DW2 programming
2. The Group access register for PORT_PCS_DW1 can be used to program the non-Aux AFE lanes.

DSI Transcoder Disable Sequence

Note that this sequence is agnostic to whether the transcoder is operating in the Video (i.e. DPI) or Command (i.e. DBI) mode of operation.

1. Turn off Backlight, if desired
2. Disable Planes, Pipe, and Transcoder
 - a. If VGA
 - i. Set VGA I/O register SR01 bit 5 for screen off
 - ii. Wait for 100us
 - b. Disable planes (VGA or hires)
 - c. Disable **TRANS_CONF**
 - d. Wait for off status in TRANS_CONF
 - i. Poll TRANS_CONF register or timeout after two frame times
 - e. Send any necessary commands to the Panel now
 - f. Make sure all commands have been delivered to the Panel
 - i. Software will wait for all Header and Payload Credits to be released (**DSI_CMD_TXCTL**)
 1. Poll DSI_CMD_TXCTL or Timeout after 100us
 - ii. Build a NOP DCS command using the DSI_CMD_TXHDR register
 1. The packet will use a Data Type of 0x05 (DCS Short Write - 0 Parameters)
 2. The Word Count/Parameter bytes will be set to all zeroes (NOP DCS command)
 3. Set the LPDT bit
 - iii. After loading the NOP with the DSI_CMD_TXHDR write, Software will wait for the following to be true:
 1. Wait for all Header Credits to be released
 1. Poll DSI_CMD_TXCTL or Timeout after 20us
 2. Wait for LP Tx in Progress bit of the DSI_LP_MSG register to be cleared
 1. Poll DSI_LP_MSG or Timeout after 20us
 - g. Put the Link in ULPS
 - i. Configure ULPS Type and Entry in **DSI_LP_MSG**
 - ii. Wait for the Link to enter ULPS (logged within the DSI_LP_MSG register) or an interrupt indicating ULPS entry is done
 1. Option to timeout after 10us
 - h. Disable **TRANS_DDI_FUNC_CTL**
 - i. Wait for 1us to allow the PPI lane enables to propagate to the DPHY
 - j. Disable TRANS_DDI_FUNC_CTL2 Port Sync Mode Enable, if both DSI transcoders are operating in Dual Link mode
 - k. Disable panel fitter, if enabled

3. Disable Port. Note that if the Port is going to be disabled, then the IO power will have to be disabled as well
 - a. Disable **DDI_BUF_CTL**
 - b. Wait for 8us or poll on DDI_BUF_CTL Idle Status for buffers to return to idle
4. Disable IO Power
 - a. Disable **PWR_WELL_CTL_DDI** IO Power Request for the DDI that was used
 - b. Wait for PWR_WELL_CTL_DDI Power Status = Disabled
 - c. Configure **DSI_IO_MODECTL** for DDI operation
5. Disable PLL
 - a. Configure PLL to port mapping to direct no clock to the DDI
 - b. If this PLL is no longer needed, disable
6. Disable Power Wells being used by the transcoder if there are no required resources within the power well

There are two levels of port disablement allowed by the IO when it is operating in the MIPI mode.

Level	ULPS	Disable Buffer	Disable IO Power	Notes
1	x			This will gate the clocks within the IO AFE. This is equivalent to disabling the buffer without disabling the IO power when the IO is operating as a DDI port.
2	x	x	x	This will power gate the IO

Note that it is invalid to put the Link into ULPS and then disable the Buffer without also disabling the IO power. E.g. the following flow is not allowed:

- Enter ULPS
- Disable Buffer
- Re-enable Buffer
- Exit ULPS

Dual Link Enable Sequence

When setting up two DSI transcoder's to be synchronized, there are two different configurations possible:

1. Single Pipe to two ports (Dual Link - Single Pipe)
2. Two Pipes to two ports (Dual Link - Double Pipes)

Notes:

- Currently, only Pipe A can support a Dual Link - Single Pipe configuration.
- The expectation from the DSI transcoders when they are synchronized is:
 - **Only a single PLL is being used to generate the 1X clock.**

- TE events will need to be supplied to both transcoders

The following table illustrates the programming of the TRANS_DDI_FUNC_CTL2 and the configuration of the DSI ports

Dual Pipe Sync	Port Sync Mode	DSI Port configuration
0	0	Port is independent
0	1	Dual Link - Single Pipe
1	0	Not valid
1	1	Dual Link - Dual Pipe

Regardless of the type of synchronization, the following enable sequence will be common:

1. Enable the Power Well. This step is only done once since both Primary and Secondary transcoders are in the same well.
2. Enable IO power. This step will have to be done for **each** IO port
 - a. Note that if enabling for Dual Link - Single Pipe (DLSP), then Software can send the Power Requests (PWR_WELL_CTL_DDI) to both IO ports at the same time and then wait for the Power Status = Enabled for both ports
3. Enable the PLL. This step is only done once since both ports should run off of the same PLL.
4. Configure and Enable DPHY will have to be done for **each** IO port
 - a. Again for DLSP, Software can enable both buffers at the same time via the DDI_BUF_CTL register and wait for both buffers to go to the Ready state

Beyond this point the enabling sequences will differ slightly.

Dual Link - Single Pipe (DLSP)

When setting up the DSI transcoders for dual link - single pipe mode, DSI Transcoder 0 will always be the primary and DSI Transcoder 1 will always be the secondary. Each transcoder will transmit 1/2 of the image to be displayed by the panel plus any overlap. It is beyond the scope of the DSI transcoder on which portion of the panel image it is transmitting.

The majority of the primary transcoder registers can be mirrored to the secondary transcoder when the Port Sync Mode is enabled within the TRANS_DDI_FUNC_CTL2 register of DSI Transcoder 0 (i.e. the **primary**). This allows Software to perform the enabling sequence for a single DSI transcoder (the primary). **However, Software must:**

1. Power up and enable **both** port IO's before moving to step 5 of the enabling sequence
2. Program the DSI0 transcoder (the primary) first. If Software is not taking advantage of the mirroring done in hardware, then it can program DSI1 (the secondary), but only after programming DSI0.

After the IO's for both ports have been powered up and enabled, Software can move to step 5 of the enabling sequence.

- Again, Software only needs to program the primary registers

- When programming the **TRANS_DDI_FUNC_CTL2** register set the Port Sync Mode Enable, but keep the Dual Pipe Sync disabled
- Configure the **PIPE_DSS_CTL1** register for Pipe A
 - Enable the Splitter
 - Note that Software **cannot** enable the small Joiner in DLSP
 - Select the Dual Link Mode of operation (Front-Back or Interleave)
 - Program the Overlap, if the overlap is non-zero
- Configure the **PIPE_DSS_CTL2** register for Pipe A
 - If compression is being used, then both Left and Right branch VDSC engines will need to be enabled
 - Configure the Right dual link buffer target depth, if required
- Program the DSI Input Select of the **TRANS_DDI_FUNC_CTL** register to Pipe A when enabling the function
- Wait for the DSI Link Ready (**TRANS_DSI_FUNC_CONF**) for **both** ports

Once both transcoder functions are enabled and ready, Software can move to programming the Panels.

- Software will have to do this for each Panel independently
- Software should finish initializing both Panels before moving on

After Software is finished initializing the Panels it can enable the Planes, Pipe A, and Transcoder. When programming the Horizontal Active size, Software will program it for **half** of the total horizontal active size across both Panels. **Software should not include any overlap within this programming** (HW will handle this)

Dual Link - Dual Pipe (DLDP)

When operating in the dual link - dual pipe configuration, each transcoder is being supplied by a separate and independent Pipe (any Pipe can drive a given transcoder as long as it's not the same Pipe).

Since both transcoders are being supplied by separate Pipes there are a few major differences from DLSP

1. The mirroring that occurs with DLSP does not happen (the Dual Pipe Sync bit of DSI 0 will disable it). This means that Software is responsible for programming both ports with identical timings
2. Either port can be disabled and/or enabled on the fly. When a port is enabled in the presence of an already running port Software can have Hardware automatically synchronize the newly enabled port to the already running port.
3. In Command Mode Software will have to initiate Frame Updates to each transcoder independently. The transcoders will ensure both ports start transmitting the frame after both ports have received their TE events

Because both transcoders are essentially independent yet synchronized, the concept of a primary and secondary port changes. A given transcoder that has its Port Sync enabled will draft off of the other port (i.e. the Primary). Not only will the Primary not know that it is being drafted off of by the other port, but it's possible for both ports to be subordinate to each other (i.e. both ports have the Port Sync bit set, so

there is no primary). The latter scenario will be the normal mode of operation when both ports are in Command Mode.

Synchronizing in Video Mode

There are a couple of ways to get the ports synchronized to each other in Video Mode

1. Bring both ports up at the same time where both ports are sync'ed to each other (i.e. both ports have their Port Sync Enable set)
2. Bring the ports up independently where the second port is subordinate to the first port

When bringing both ports up synchronized at the same time in Video Mode:

- Power up and enable both port IO's before moving to step 5
- Software will have to do the following for each transcoder
 - When programming the **TRANS_DDI_FUNC_CTL2** register set the Port Sync Mode Enable and the Dual Pipe Sync Enable
 - Configure the **PIPE_DSS_CTL1** register for the Pipe that will be feeding the transcoder, if needed
 - Software should not enable Splitting
 - Software can enable Joining
 - Configure the **PIPE_DSS_CTL2** register for the Pipe that will be feeding the transcoder, if compression is needed
 - Configure and enable **TRANS_DDI_FUNC_CTL**
 - The DSI Input Select should be different for each transcoder
 - Wait for the DSI Link Ready (**TRANS_DSI_FUNC_CONF**)
 - Program the Panel
 - Enable the Planes and Pipe, but stop short of enabling the Transcoder
 - Note that the Horizontal Active should be programmed for the active size of the given Panel
- When above programming for both transcoders has been complete, then enable both Transcoders (**TRANS_CONF**).
 - Wait for both Transcoder States to be enabled
- If allowing DC*co, then do that now

When bringing up a single port to be synchronized to an already running port in Video Mode:

- The port already running is enabled normally without Port Sync Mode being enabled
- The new port is enabled normally but it will enable the Port Sync Mode and the Dual Pipe Sync within the **TRANS_DDI_FUNC_CTL2** register

Synchronizing in Command Mode

When in Command Mode, if Software wishes to synchronize the frames between the two ports then it must enable the Port Sync for **both** ports (i.e. both ports have to have the Port Sync Mode and Dual Pipe Sync bits set) before sending the Frame Update Request to either port. The enabling sequence, otherwise, is not unique.

When the transcoders are sync'ed to each other in Command Mode, each transcoder will satisfy the following conditions before starting the transmission of the frame data:

1. This transcoder has met its minimum V. Blank requirements as programmed within the **TRANS_VTOTAL** register
2. This transcoder has received the TE event from its Panel (if TE Gating is enabled)
3. The remote transcoder has met the above two requirements

The transcoders do supply some hooks within the **DSI_CMD_FRMCTL** register for Software to initiate frame updates to the Panels.

- **Single Panel Update.** This attribute will allow a transcoder that is synchronized to another port to send out a frame update without requiring the other port to send a frame
- **Forward Frame Update Request.** This attribute will allow SW to send a single Frame Update Request to a single transcoder and have that transcoder forward the request to the other transcoder
- **Accumulate Frame Update Requests.** This attribute will enable the synchronized transcoders to synchronize the Frame Start to each Pipe

Initializing Panel in LP Escape Mode

Some Panels may require initialization (i.e. Panel programming) to be performed by the Host in the LP Escape mode without the Clock Lane running. After initialization is complete in the LP Escape mode and before the transcoder's timing generator is enabled (TRANS_CONF), the Panel may need the Clock Lane to be running continuously. The following sequence describes the flow needed to reliably perform these Panel requirements.

1. Software will enable the Power Wells, PLL, and IO Power per the above enabling sequence
2. When Enabling the DSI Port and D-PHY, Software will leave the Continuous Clock setting of the TRANS_DSI_FUNC_CONF register to its default value of "Always enter LP after Data Lanes"
 - a. Software will otherwise continue the enabling flow within this step normally where the DSI Function will be enabled at the end (TRANS_DDI_FUNC_CTL / TRANS_DSI_FUNC_CONF)
3. Software will perform the Program Panel step normally except it will set the "LPDT" attribute when writing to the DSI_CMD_TXHDR register.
4. When Software has finished sending all of the Panel initialization commands, it will ensure they have all been dispatched by doing the following serialization:
 - a. Software will wait for all Header and Payload Credits to be released (**DSI_CMD_TXCTL**)

- i. Timeout after 100us
- b. Build a NOP DCS command using the DSI_CMD_TXHDR register
 - i. The packet will use a Data Type of 0x05 (DCS Short Write - 0 Parameters)
 - ii. The Word Count/Parameter bytes will be set to all zeroes (NOP DCS command)
 - iii. Set the LPDT bit
- c. After loading the NOP with the DSI_CMD_TXHDR write, Software will wait for the following to be true:
 - i. Wait for all Header Credits to be released
 - 1. Timeout after 20us
 - ii. Wait for LP Tx in Progress bit of the **DSI_LP_MSG** register to be cleared
 - 1. Timeout after 20us
- 5. When Software has finished the above serialization it will program the Continuous Clock setting of the TRANS_DSI_FUNC_CONF register to the desired setting
- 6. When finished with the above sequence, Software can continue with the enabling sequence and go to the step to enable Planes, Pipe, and Transcoder

Note that this sequence is only for Panels that require initialization in the LP Escape mode. If the Panel does not have this requirement, then Software should perform the normal enabling sequence described above and send the Panel initialization programming in the HS mode.

Sending Commands to the Panel

Software has the ability to send command's to the Panel (e.g. Display Set Command packets) using some specific registers within the DSI transcoder. The DSI transcoder supplies Software with a pool of credits (both header and payload) that Software can use to inject commands onto the DSI Link to the panel.

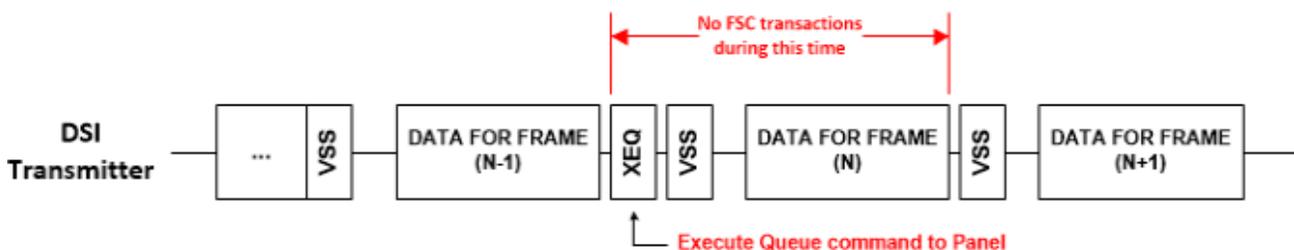
Command Injection Sequence

1. SW will read the DSI Command Transmission Control register (**DSI_CMD_TXCTL**) to determine how many credits are available.
 - a. SW must only use the available credits advertised within the register at the time of the read (i.e. it cannot combine the credits from a previous read)
2. If there are enough credits available for the packet it wishes to build then it will begin issuing writes to the Payload and/or the Header registers
 - a. If the packet contains a payload:
 - i. SW will use the Payload credits it read in step 1 to write to the Command Payload register (**DSI_CMD_TXPYLD**)
 - ii. Every write to DSI_CMD_TXPYLD will load the data written to the register into a payload queue within the transcoder which will consume a Payload resource (i.e. credit)
 1. HW will maintain this credit pool and reflect the current number of available credits within the DSI_CMD_TXCTL register

2. SW can also maintain the credits that it read in step 1 to pipeline the writes to this register, if it has not consumed all of the credits read from that step
- iii. SW will write the first DW of the payload first followed by the subsequent DW's of payload data
- b. If the packet does not contain a payload, or it has finished writing the payload data to the DSI_CMD_TXPYLD register, it will then write to the Command Header register (**DSI_CMD_TXHDR**) using a Header credit it read in step 1
 - i. If the packet contains a payload, then SW will:
 1. Program the Word Count (WC) field of the packet header to the number of **bytes** within the packet payload that was written to the Payload queue
 2. Set the Payload Present attribute
 - ii. Otherwise, SW will clear the Payload Present attribute when writing the Command Header register
 - iii. SW will also set the other transmission attributes it wishes to be applied to the packet
 1. The LPDT bit will indicate whether the packet should be sent to the Periphery in the High Speed (HS) or the Low Power (LP) mode
 2. The Vertical Blank Fence bit will determine whether the packet will be gated until the next Vertical Blank entry (for Video Mode) or the end of the current frame update is seen
 - iv. A write to DSI_CMD_TXHDR will load the data written to the register into a header queue within the transcoder which will consume a Header resource (i.e. credit)
 1. HW will maintain this credit pool and reflect the current number of available credits within the DSI_CMD_TXCTL register
 2. SW can also maintain the credits that it read in step 1 to continue issuing writes to this register, if it has not consumed all of the credits read from that step

Frame Synchronized Command Sequences

The MIPI DSI specification defines a mechanism for the Peripheral to synchronize commands to vertical blanking events internal to the panel. The types of commands that the Peripheral considers to be Frame Synchronized Commands (FSC) is panel specific, but the specification does have restrictions on how and when FSC transactions can be sent to the panel from the Host.



The Vertical Blank Fence attribute within the DSI_CMD_TXHDR is the mechanism within the DSI transcoder that Software can use to ensure these restrictions are met.

Command Mode

1. Load Execute Queue command (Data Type = 0x16) into the **DSI_CMD_TXHDR** register
2. Set the Frame Update Request bit of the **DSI_CMD_FRMCTL** (this can happen at any time after loading the Execute Queue command)
3. If there is another FSC transaction to send, then load it into the DSI_CMD_TXHDR register with the Vertical Blank Fence bit set

Video Mode

1. Load Execute Queue command (Data Type = 0x16) into the DSI_CMD_TXHDR register with the Vertical Blank Fence bit set
2. If there is another FSC transaction to send, then load it into the DSI_CMD_TXHDR register with the Vertical Blank Fence bit set

Low Power Messaging Sequence

1. SW will read the DSI Command Transmission Control register (**DSI_CMD_TXCTL**) to determine if there are any Header credits available.
 - a. Each LP message consumes a single Header credit
2. If there is a Header credit available, then Software will program the **DSI_LP_MSG** register with the LP message it wishes to send
 - a. If the LP message is either a Trigger or a ULPS Entry, then the respective message type will also need to be programmed
 - b. If the LP message is for a ULPS Entry, then Software will need to follow the "DSI Transcoder Disable Sequence" up until step 1.f
3. When HW is servicing the LP message it will advertise the following through the DSI_LP_MSG register:
 - a. For all LP messages the transcoder will set the "LP Tx in Progress" bit
 - b. The Direction of the Link is always advertised within the register. When doing a Bus Turn-Around, Software will be able to see when the Link changes direction
 - c. When doing a ULPS Entry, the register will advertise when the Link has entered the ULP state.
4. When HW is done servicing the LP message it will do the following (Software can key off of any of these to determine when the LP message is complete):
 - a. It will clear out the LP message request bit that Software set (i.e. ULPS Entry, Bus Turnaround, or Trigger Message)
 - b. Except for ULPS Entry messages, it will clear the "LP Tx in Progress" status bit
 - i. For ULPS Entry messages, this bit will not be cleared out until the DSI Function is disabled
 - c. For ULPS Entry messages:
 - i. The "In ULPS" bit will be set. This bit will be cleared when the DSI Function is disabled

- ii. If unmasked, the "ULPS Entry Done" interrupt will be set
- 5. When done with the ULPS entry, Software can continue with the DSI Transcoder Disable Sequence

Data Receipt Sequence

1. If SW is sending a read to the Panel, then it will follow the "Command Injection Sequence" above to place a read request within the header queue via a write to the **DSI_CMD_TXHDR**. Some notes on this step:
 - a. If Software is just looking for an error status from the Panel, then it can go directly to step 2
 - b. Before sending a read to the Panel, Software must have sent a "*Set Maximum Packet Size*" DSI packet (Data Type = 0x37) to the Panel to control the amount of data the Panel returns to the Host
 - c. The DSI transcoder will send the read command within the next Vertical blank line time that it sees, if operating in Video Mode. If Software wishes the command to be sent at the beginning of the next Vertical blank region, then it can set the Vertical Blank Fence bit when writing to the DSI_CMD_TXHDR
2. SW will write to **DSI_LP_MSG** to place a Bus Turn-Around (BTA) request within the header queue
 - a. Prior to this SW should have set up the following (see **Performing LP Transactions in Video Mode** below for more details):
 - i. The Turn-Around Timeout (**DSI_TA_TO**) to ensure the BTA request to the Panel proceeds in a timely manner
 1. SW will need to program the equivalent register within the Panel to a time that is less than what is programmed within the DSI_TA_TO register. The command needed to update any registers within the Panel it is beyond the scope of this flow.
 - ii. The maximum allowed time the Peripheral will be given to own the Link (**DSI_LRX_H_TO**)
 1. Again, SW will need to program the equivalent register within the Panel with a time that is less than what is programmed within the DSI_LRX_H_TO register
 2. If in Video Mode, the value should be programmed such that the time given to the panel is less than the time between Sync events
 - iii. The Peripheral Response Timeout (**DSI_PWAIT_TO**) to give the Panel some time before it has to send back a response.
 - iv. If the Panel does not support ECC, then both of the Single and Multi ECC Mask bits of the Interrupt Mask Register (**IMR**) should be set
 - v. If the Panel does not support CRC checksums, then the Checksum Error bit of the IMR should be set
 - b. Notes:
 - i. SW will need a Header Credit to place the Bus Turn-Around request within the Header queue

- ii. The DSI transcoder dispatches the commands it receives from Software via the DSI_CMD_TXHDR and DSI_LP_MSG registers in the order they are received
 - iii. The DSI transcoder will dispatch the BTA at the beginning of the next Vertical blank line time that it sees, if it is in Video Mode. In other words, it will not dispatch the read and the BTA within the same Vertical blank line time.
- 3. If SW wishes to have HW announce the receipt of the response from the Panel through an interrupt, then it has the following interrupts that it can use. To enable these interrupts, SW should clear the mask bit within the Interrupt Mask Register (IMR)
 - a. Rx Trigger: This is for non-TE triggers received from the Panel. If the Panel does not have any return data to send back to the Host, and it has not detected any errors since the last BTA, then it will only send a trigger message (most likely an Acknowledge trigger)
 - b. Rx Data/BTA Terminated: HW will generate this type of event when it either receives a response that SW can read from the **DSI_CMD_RXHDR** and/or **DSI_CMD_RXPYLD**, or the BTA was terminated
 - c. Error Report: If the Panel detected an error after the last BTA, then it will send an Acknowledge and Error Report packet back to the Host. The error report carried within this packet is sent directly to the interrupt register (the lower 16 bits of the interrupt registers). Depending on the severity of the error, the Peripheral may not send any other responses (i.e. a READ return packet).
- 4. When the response is received from the Panel, SW will be able to read the header of the response through the DSI_CMD_RXHDR register and the payload, if there is a payload, through the DSI_CMD_RXPYLD register
 - a. The payload of the response will be contained within a queue that will be unloaded for every read to the DSI_CMD_RXPYLD register. SW can control whether the DW of data is unloaded from the queue or not through the **DSI_CMD_RXCTL** register
 - b. The DSI_CMD_RXCTL register will also advertise the number of DW's available within the payload queue, so SW can use this to determine how many reads it needs to make to DSI_CMD_RXPYLD
- 5. If the Panel detected any errors since the last BTA, then it will send an Acknowledge and Error Report packet to the Host that will contain the errors that it detected. If the masks for these errors are cleared within the IMR, then SW will have these errors logged within the Interrupt Identity Register (IIR)

Send a Frame in Command Mode

1. During the enabling sequence, Software will:
 1. Setup the source of the Tear Effect (TE) events within the TRANS_DSI_FUNC_CTL register - in-band over the DSI Link or out-of-band through a GPIO.
 2. Choose how the start of the incoming frame pixels are to be handled by the DSI transcoder through the Mode of Operation of the TRANS_DSI_FUNC_CTL register - will it wait for a TE event to be received or not

2. To start a frame, Software will set the Frame Update Request bit of the DSI_CMD_FRMCTL. Hardware will clear this bit once it has begun processing the request.
 1. Note that if configured in Dual Link mode, then only the DSI_CMD_FRMCTL register of the primary port (DSI port 0) will initiate a frame update. However, if Software wishes to observe the Frame in Progress bit of the register, it should look at the secondary port's DSI_CMD_FRMCTL register (DSI port 1) since the secondary port will always be slightly behind the primary port.

Tear Effect (TE) Signaling Sequences

There are three different methods for the Host to get TE information from the Panel:

1. Polling the Panel to detect the current scan line (i.e. using DCS *get_scan_line* commands)
2. Receiving TE events out-of-band (over GPIO)
3. Receiving TE events in-band (over the DSI Link)

For the polling option, Software will simply send out DCS *get_scan_line* requests to the Panel (through DSI_CMD_TXHDR followed by a BTA request through DSI_LP_MSG) until the Display has reached a certain scan line. To receive this information through TE events, Software will need to program the Panel to send these events at specified times (through the DCS *set_tear_on* and *set_tear_scanline* commands). The following is the basic sequences needed to handle receiving TE events.

Receiving Out-of-Band TE Events

1. Software will configure the transcoder to take the TE events from the GPIO when configuring the TRANS_DSI_FUNC_CONF
 - a. Software will need to enable and configure the **UTIL_PIN_CTL** register as the TE input for DSI 0
2. During the Panel Programming section of the enabling sequence, Software will configure the Panel to send TE events at the desired times
3. Software will continue the enabling sequence and finally enable the transcoder
4. If TE events are gating:
 - a. Software will initiate the Frame Request (DSI_CMD_FRMCTL)
 - b. Transcoder hardware will automatically send the frame data to the Panel when it receives the TE event
 - c. Software will repeat with a new Frame Request
5. If TE events are not gating:
 - a. Software will initiate a Frame Request (DSI_CMD_FRMCTL)
 - b. Transcoder hardware will automatically send the frame data to the Panel
 - c. Software will repeat with a new Frame Request

Receiving In-Band TE Events

1. Software will configure the transcoder to take the TE events from the DSI receiver when configuring the TRANS_DSI_FUNC_CONF
2. During the Panel Programming section of the enabling sequence, Software will configure the Panel to send TE events (set_tear_on, set_tear_scanline DCS commands)
 - a. When operating in Dual Link, both ports will need to be programmed to send the TE triggers
 - b. After enabling the tearing on the Panel, a BTA should be sent to the Panel (DSI_LP_MSG) so that the Panel can Acknowledge that the set_tear_* DCS commands were received without any errors
3. Software will continue the enabling sequence and finally enable the transcoder
4. If TE events are gating:
 - a. Software will initiate the Frame Request (DSI_CMD_FRMCTL). If in Dual Link Mode, this is done on the Primary port (DSI 0).
 - b. Software will send a BTA request to the Panel (DSI_LP_MSG). If in Dual Link Mode, this is done on both ports.
 - c. Transcoder hardware will automatically send the frame data to the Panel when it receives the TE event and the Panel turns control back over to the Host
 - d. Repeat when the Frame is done (Frame In Progress de-asserts)
5. If TE events are not gating:
 - a. Software will initiate the Frame Request (DSI_CMD_FRMCTL). Again, if in Dual Link Mode, then this is performed on the Primary port.
 - b. Transcoder hardware will automatically send the frame data to the Panel
 - c. When the Frame is done (Frame In Progress de-asserts), Software will send a BTA request to the Panel (DSI_LP_MSG)
 - i. If in Dual Link Mode, then Software will monitor the Primary port's DSI_CMD_FRMCTL.Frame In Progress bit, and will send the BTA request on both ports
 - d. Repeat when the TE event is received (through either interrupt or DSI_CMD_RXCTL)

Periodic Command Mode

As described above, frame updates sent to the Panel in Command Mode are typically initiated by Software (i.e. Intel's graphics driver). When the Intel graphics driver is not present (e.g. the driver is uninstalled, the OS basic driver is present, or pre-boot time) there will be no frame update requests to the DSI transcoder which will lead to a stale screen. In this situation, the DSI transcoder will have the ability to automatically issue frame updates to the Panel by trapping off of GPIO TE events received from the Panel.

The requirements to enable this feature are:

1. The DSI transcoder will be configured to send Periodic Frame Updates (DSI_CMD_FRMCTL)
 - a. By default this feature will be disabled

2. The Panel will need to be configured to send TE events to the Host (i.e. set_tear_on, etc.)
3. The DSI controller will override the programming of the TE Source and TE Gating/Non-Gating when this mode of operation is enabled. The HW overrides will be:
 - a. The TE source will be taken from the GPIO. The DSI transcoder HW does not have a mechanism to send Bus Turn-Around's to the Panel
 - b. Has to be configured for non-gating Command mode

Switching between Video Mode (DPI) and Command Mode (DBI)

1. Disable the **TRANS_CONF** Transcoder Enable
2. Wait for off status in TRANS_CONF
 - i. Timeout if transcoder hasn't moved to the off status after two frame times
3. Change the TRANS_DSI_FUNC_CTL Mode of Operation to the new mode
4. Change any of the other transcoder settings in preparation for Command Mode
5. Re-enable the TRANS_CONF Transcoder Enable
6. If the new mode of operation is Command Mode, then Software can start transmitting frame updates to the Panel using the **DSI_CMD_FRMCTL** register (see "Send a Frame in Command Mode"). Otherwise, the transcoder will automatically start sending frames to the panel

Programming References

Programming references has been moved to the "Transcoder DSI Function" page

Sequences to Initialize Display

These sequences are used to initialize the display engine before any display engine functions can be enabled. To save power, do not initialize display if it will not be used, such as on "headless" systems.

Most display engine functions will not operate while display is not initialized. Only basic PCI, I/O, and MMIO register read/write operations are supported when display is not initialized.

Initialize Sequence

1. Enable PCH Reset Handshake
 - a. Set **NDE_RSTWRN_OPT** RST PCH Handshake En to 1b.
 - b. Configure south display Raw Clock before first enabling GMBUS, south display hotplug detection, or panel power sequencing.
2. Initialize all combo PHYs with Combo PHY DDI Buffer Combo PHY Initialization Sequence.
3. Enable Power Well 1 (PG1)
 - a. Poll for **FUSE_STATUS** Fuse PG0 Distribution Status = 1b.
 - Timeout and fail after 20 us.
 - b. Set **PWR_WELL_CTL** Power Well 1 Request to 1b.

- There are two sets of PWR_WELL_CTL registers for software use. It is expected that BIOS uses PWR_WELL_CTL1 and driver uses PWR_WELL_CTL2.
- c. Poll for **PWR_WELL_CTL** Power Well 1 State = 1b.
 - Timeout and fail after 30 us (38.4 MHz reference frequency) or 45us (24 MHz reference frequency).
 - Register DSSM Reference Frequency indicates the reference frequency.
- d. Poll for **FUSE_STATUS** Fuse PG1 Distribution Status = 1b.
 - Timeout and fail after 20 us.
 - Combo PHY Aux IO can be optionally enabled in parallel with PG1 following DDI AUX Channel Aux IO Power Enabling, or it can be enabled later when the Aux will be used.
- 4. Enable CD clock following the Sequences for Changing CD Clock Frequency
- 5. Enable first DBUF
 - a. Set **DBUF_CTL_<first DBUF>** DBUF Power Request to 1b.
 - b. Poll for **DBUF_CTL_<first DBUF>** DBUF Power State = 1b.
 - Timeout and fail after 10 us.
 - Additional DBUFs may be enabled later as needed for the display bandwidth and number of pipes enabled.
- 6. Setup MBUS. Refer to MBus page for the MBus credits programming.
- 7. Program BW_BUDDY registers. Refer to Arbiter page.

Un-initialize Sequence

Software should only run the un-initialize sequence as part of DC9. It can conflict with other DC states.

1. Disable all display engine functions using the full mode set disable sequence on all pipes, transcoders, ports, planes, and power wells above PG1.
2. Disable DBUFs
 - a. Clear **DBUF_CTL_*** DBUF Power Request to 0b.
 - b. Poll for **DBUF_CTL_*** DBUF Power State = 0b.
 - Timeout and fail after 10 us.
3. Disable CD clock following the Sequences for Changing CD Clock Frequency
4. Disable Power Well 1 (PG1) and Aux IO Power.
 - a. Follow AUX Channel Aux IO Power Disabling for every Aux that is powered up.
 - b. Clear **PWR_WELL_CTL_*** Power Well 1 Request to 0b.
 - c. Wait for 10us. Do not poll for the power well to disable. Other clients may be keeping it enabled.
5. Follow Combo PHY DDI Buffer Combo PHY Un-Initialization Sequence for all combo PHYs.

Sequences for DisplayPort

This topic describes how to enable and disable DisplayPort.

The DP transport is within the transcoder. Multi-stream requires a transcoder to be enabled to act as the primary transcoder, hosting the DP transport that consolidates the multiple streams. The primary transcoder's clocking, TRANS_DDI_FUNC_CTL, and DP_TP_CTL must be kept enabled while the multistream DP link is enabled, and not enabled and disabled with the video stream.

Enable Sequence

Display must already be initialized

DDIA Lane Capability Control must be configured prior to enabling any ports or port clocks

1. **Enable Power Wells**
 - a. Based on the resources to be used, enable the appropriate power wells following the Sequences for Power Wells
 - b. Type-C: Note that AUX power is required for running main link.
2. **If panel power sequencing is required - Enable Panel Power**
 - a. Enable panel power sequencing
 - b. Wait for panel power sequencing to reach the enabled state
3. **Type-C ports**
 - a. Skip this step for TBT.
 - b. Program DFLEXDPMLE.DPMLETC* to maximum number of lanes allowed as determined by FIA and panel lane count.
 - i. Fixed/legacy/static - Program the number of lanes as per DDI_BUF_CTL DP Port Width Selection
 - ii. Dynamic - Program the number of lanes as per DFLEXDPSP.DPX4TXLATC*
4. **Enable Port PLL**
 - a. If PLL is not already enabled, follow port clock programming sequence from Clocks section
 - b. If PLL to port mapping is flexible, configure PLL to port mapping to direct the PLL output to the DDI
5. **If IO power is controlled through PWR_WELL_CTL - Enable IO Power**
 - a. Skip this step if TBT.
 - b. If the DDI is going to combo PHY, the PHY must be initialized with the Combo PHY DDI Buffer Combo PHY Initialization Sequence
 - c. Enable PWR_WELL_CTL DDI IO Power Request for the DDI that will be used
 - d. Wait for PWR_WELL_CTL DDI IO Power Request = Enabled, timeout after 20 us (30us for type-C PHYs with fixed/static/native connectors)
6. **Program DP_MODE**
 - a. Skip this step for display attached to Combo PHY and TBT.

- b. Program PHY lane0 and lane1 DP_MODE registers according to TypeC PHY programming page.

7. **Enable and Train DisplayPort**

- a. If single stream or multi-stream primary transcoder - Configure Transcoder Clock Select to direct the Port clock to the Transcoder
- b. If single stream or multi-stream primary transcoder - Configure TRANS_DDI_FUNC_CTL DDI Select, DDI Mode Select & MST Transport Select
- c. Configure and enable DP_TP_CTL with link training pattern 1 selected
- d. Configure voltage swing and related IO settings.
 - i. Refer to Combo DDI Buffer section for display attached to Combo PHY.
 - ii. Refer to Type-C DDI buffer page for all other cases.
- e. Combo PHY: Configure PORT_CL_DW10 Static Power Down to power up the used lanes of the DDI.
- f. Configure and enable DDI_BUF_CTL
- g. Wait for DDI_BUF_CTL DDI Idle Status = 0b (Not Idle), timeout after 1 ms.
- h. Follow DisplayPort specification training sequence (see notes for failure handling)
- i. If DisplayPort multi-stream - Set DP_TP_CTL link training to Idle Pattern, wait for 5 idle patterns (DP_TP_STATUS Min_Idles_Sent) (timeout after 800 us)
- j. Set DP_TP_CTL link training to Normal.
- k. Configure and enable FEC if needed. NOTE: Refer to Bspec "DDI FEC" page for enabling protocol with FEC capable sink.

8. **If not in compliance mode: Enable Planes, Pipe, and Transcoder (repeat to add multiple pipes on a single port for multi-streaming)**

- a. If DisplayPort multi-stream - use AUX to program receiver VC Payload ID table to add stream
- b. Configure Transcoder Clock Select to direct the Port clock to the Transcoder. Skip this step if clock select already programmed above.
- c. Configure and enable planes (VGA or hires). This can be done later if desired.
- d. If VGA - Clear VGA I/O register SR01 bit 5
- e. Configure and enable VDSC if needed.
- f. Enable panel fitter if needed (must be enabled for VGA)
- g. Configure transcoder timings, M/N/TU/VC payload size, and other pipe and transcoder settings
- h. Configure TRANS_DDI_FUNC_CTL2 if port sync mode needs to be configured. Then configure and enable TRANS_DDI_FUNC_CTL.
 - i. Please refer to the note below for big joiner mode operation.
- i. Configure VRR if needed. (note: VRR needs to be programmed after TRANS_DDI_FUNC_CTL and before TRANS_CONF).
 - i. program TRANS_VRR_VMIN

- ii. program TRANS_VRR_VMAX
- iii. Enable push bit if required
- iv. enable TRANS_VRR_CTL
- j. If DisplayPort multistream - Clear ACT sent status in primary
- k. If DisplayPort multistream - Enable pipe VC payload allocation in TRANS_DDI_FUNC_CTL
- l. If DisplayPort multistream - Wait for ACT sent status in primary transcoder DP_TP_STATUS and receiver DPCD (timeout after >410us)
- m. For DP 1.4 MST with FEC, set bit [23] at applicable offset 0x420C0/0x420C4/0x420C8/0x420D8 to '1'.
- n. Configure and enable TRANS_CONF
- o. If panel power sequencing is required - Enable panel backlight

SRD and/or Audio can be enabled after everything is complete. Follow the audio enable sequence in the audio registers section.

Notes

Changing voltage swing during link training:

- Change the swing setting following the DDI Buffer section. The port does not need to be disabled.

Changing port width (lane count) or frequency during link training:

1. Follow Disable Sequence for DisplayPort to Disable Port.
2. For TBT follow the steps to turn off the clock for the port. Otherwise, if PLL frequency needs to change, follow the Disable Sequence for DisplayPort to Disable PLL.
3. For TBT, follow the steps to map the new frequency to the port and turn on the clock for the port. For all other port types, follow the Enable Sequence for DisplayPort to Enable PLL, using the new frequency settings.
4. Type-C ports: skip this step for TBT.
 - a. Program DFLEXDPMLE.DPMLETC* for the new number of lanes (down training)
 - b. Program PHY lane0 and lane1 DP_MODE registers for the new number of lanes (down training)
5. Follow the Enable Sequence for DisplayPort to Enable and Train DisplayPort, using the new port width settings.

If the mode set fails, follow the disable sequence to disable everything that had been enabled up to the failing point.

Enabling DisplayPort Sync Mode (non-MST)

See TRANS_DDI_FUNC_CTL2 Port Sync Mode Enable for restrictions.

1. Follow the enable sequence for the DisplayPort secondary, but skip the step that sets DP_TP_CTL link training to Normal (stay in Idle Pattern).
 - Set secondary TRANS_DDI_FUNC_CTL2 Port Sync Mode Primary Select and Port Sync Mode Enable before configuring and enabling secondary TRANS_DDI_FUNC_CTL.
2. Follow the enable sequence for the DisplayPort primary, but skip the step that sets DP_TP_CTL link training to Normal (stay in Idle Pattern).
3. Set DisplayPort secondary DP_TP_CTL link training to Normal.
4. Wait 200 uS.
5. Set DisplayPort primary DP_TP_CTL link training to Normal.

Software may need to use DOUBLE_BUFFER_CTL to ensure updates to plane and pipe registers will take place in the same frame.

For example: If pipe A and pipe B are synchronized together and software needs the surface addresses for two planes to update at the same time, software should use DOUBLE_BUFFER_CTL when writing the surface address registers for both planes, otherwise there is a possibility that the updates could be split across a vertical blank such that one plane would update on the current vertical blank and the other plane would update on the next vertical blank.

Enabling DisplayPort with Big Joiner

Big joiner (such as for 8K on a single port) uses two pipes to drive a single transcoder.

1. Follow the enable sequence through the steps to Enable and Train DisplayPort
2. Follow the steps to Enable Planes, Pipe, and Transcoder for pipe C, but do not configure or enable transcoder C.
3. Follow the steps to Enable Planes, Pipe, and Transcoder for pipe B and transcoder B.

Enabling DisplayPort Sync Mode (MST with 1 DDI)

General constraint is that secondary transcoder has to be enabled first before primary transcoder.

See TRANS_DDI_FUNC_CTL2 Port Sync Mode Enable for restrictions.

1. Follow MST enable sequence for the DisplayPort steps 1 to 7
2. Enable the secondary pipe following the step 8 of MST enable sequence
 1. Skip the steps to enable TRANS_DDI_FUNC_CTL2, TRANS_DPT_PAT, TRANS_DDI_FUNC_CTL and TRANS_CONF.
3. Repeat step 8 of MST enable sequence to enable the primary pipe.
 1. Skip the steps to enable TRANS_DPT_PAT, TRANS_DDI_FUNC_CTL and TRANS_CONF.
4. Set secondary TRANS_DDI_FUNC_CTL2 Port Sync Mode Primary Select and Port Sync Mode Enable before configuring and enabling secondary TRANS_DDI_FUNC_CTL. Configure and enable secondary TRANS_CONF and secondary TRANS_DPT_PAT.
5. Configure and enable primary TRANS_DDI_FUNC_CTL. Configure and enable primary TRANS_CONF and primary TRANS_DPT_PAT.

Enabling DisplayPort Sync Mode (MST with 2 DDIs)

General constraint is that secondary transcoder has to be enabled first before primary transcoder.

See TRANS_DDI_FUNC_CTL2 Port Sync Mode Enable for restrictions.

1. secondary DDI: Follow MST enable sequence for the DisplayPort steps 1 to 7
2. Enable the secondary pipe following the step 8 of MST enable sequence
 1. Skip the steps to enable TRANS_DDI_FUNC_CTL2, TRANS_DPT_PAT, TRANS_DDI_FUNC_CTL and TRANS_CONF.
3. primary DDI: Follow MST enable sequence for the DisplayPort steps 1 to 7
4. Repeat step 8 of MST enable sequence to enable the primary pipe.
 1. Skip the steps to enable TRANS_DPT_PAT, TRANS_DDI_FUNC_CTL and TRANS_CONF.
5. Set secondary TRANS_DDI_FUNC_CTL2 Port Sync Mode Primary Select and Port Sync Mode Enable before configuring and enabling secondary TRANS_DDI_FUNC_CTL. Configure and enable secondary TRANS_CONF and secondary TRANS_DPT_PAT.
6. Configure and enable primary TRANS_DDI_FUNC_CTL. Configure and enable primary TRANS_CONF and primary TRANS_DPT_PAT.

Disable Sequence

SRD and Audio must be disabled first. Follow the audio disable sequence in the audio registers section.

1. **If panel power sequencing is required - Disable panel backlight**
2. **If not in compliance mode: Disable Planes, Pipe, and Transcoder (repeat to remove multiple pipes from a single port for multi-streaming)**
 - a. If VGA
 - i. Set VGA I/O register SR01 bit 5 for screen off
 - ii. Wait for 100 us
 - b. Disable planes (VGA or hires)
 - c. Disable TRANS_CONF
 - d. Clear bit [23] to '0' at applicable offset 0x420C0/0x420C4/0x420C8/0x420D8 if set to '1'.
 - e. Wait for off status in TRANS_CONF, timeout after two frame times
 - f. If DisplayPort multistream - use AUX to program receiver VC Payload ID table to delete stream
 - g. If done with this VC payload
 - i. Clear ACT sent status in primary
 - ii. Disable VC payload allocation in TRANS_DDI_FUNC_CTL
 - iii. Wait for ACT sent status in DP_TP_STATUS (multi-stream use the primary transcoder DP_TP_STATUS) and receiver DPCD
 - h. Disable VRR, if enabled.
 - i. Disable push bit if enabled.

- j. If MST primary transcoder: Disable TRANS_DDI_FUNC_CTL and do not change DDI_Select
 - 1. All other transcoders: Disable TRANS_DDI_FUNC_CTL with DDI_Select set to None
 - k. Disable DSS_CTL1 Joiner enable (bit 30) if enabled
 - l. Disable DSS_CTL2 left VDSC (bit 31) and/or right VDSC (bit 30) if enabled
 - m. Disable panel fitter
 - n. If multi-stream secondary transcoder: Configure Transcoder Clock Select to direct no clock to the transcoder
3. **Disable Port (all pipes and VC payloads on this port must already be disabled)**
- a. If multi-stream primary transcoder: Set DDI_Select set to None
 - b. Disable DDI_BUF_CTL
 - c. Disable DP_TP_CTL (do not set port to idle when disabling)
 - d. Disable FEC if it is enabled.
 - e. Wait 8 us or poll on DDI_BUF_CTL Idle Status for buffers to return to idle
 - f. If single stream or multi-stream primary transcoder: Configure Transcoder Clock select to direct no clock to the transcoder
4. **If panel power sequencing is required - Disable Panel Power**
- a. Disable panel power sequencing
5. **If IO power is controlled through PWR_WELL_CTL - Disable IO Power**
- a. Skip this step for TBT. For all other port types, disable PWR_WELL_CTL DDI IO Power Request for the DDI that was used.
6. **Disable Port PLL**
- a. If PLL to port mapping is flexible, configure PLL to port mapping to direct no clock to the DDI
 - b. If this PLL is no longer needed, follow PLL disable sequence from Clocks section
7. **Disable Power Wells**
- a. Disable power wells that are no longer required, following the Sequences for Power Wells

Disabling DisplayPort Sync Mode

1. Follow the disable sequence for the DisplayPort secondary.
2. Follow the disable sequence for the DisplayPort primary.

Disabling DisplayPort Sync Mode (MST with 1 DDI)

1. Follow the disable sequence for the MST DisplayPort secondary pipe but keep secondary VC enabled. Step 2 of Disable Sequence.
2. Follow the disable sequence for the MST DisplayPort primary stream. Step 2 of Disable Sequence.
3. Disable secondary VC.

Disabling DisplayPort Sync Mode (MST with 2 DDIs)

1. Follow the disable sequence for the MST DisplayPort secondary stream. Step 2 of Disable Sequence.
2. Follow the disable sequence for the MST DisplayPort primary stream. Step 2 of Disable Sequence.

Disable Displayport with Big Joiner

1. Follow the steps to disable Planes, Pipe, and Transcoder for pipe B and transcoder B
2. Follow the steps to Disable Planes and Pipe for pipe C
3. Disable port associated with Transcoder B

Sequences for HDMI and DVI

This topic describes how to enable and disable HDMI and DVI.

Enable Sequence

Display must already be initialized

1. **Enable Power Wells**
 1. Based on the resources to be used, enable the appropriate power wells following the Sequences for Power Wells
 2. Type-C static connection: Enable AUX I/O Power. It is required for running the main link, even with HDMI.
2. **Type-C static connection: Program DFLEXDPMLE.DPMLETC* to maximum number of lanes of the port.**
3. **Enable Port PLL**
 - a. If PLL is not already enabled, follow port clock programming sequence from Clocks section
 - b. If PLL to port mapping is flexible, configure PLL to port mapping to direct the PLL output to the DDI
4. **If IO power is controlled through PWR_WELL_CTL - Enable IO Power**
 - a. Enable PWR_WELL_CTL DDI IO Power Request for the DDI that will be used
 - b. Wait for PWR_WELL_CTL DDI IO Power Request = Enabled, timeout after 20 us (30us for type-C PHYs with fixed/static/native connectors)
5. **Type-C static connection: Program Type-C PHY DP_MODE register for static x4**
6. **Enable Planes, Pipe, and Transcoder**
 - a. Configure Transcoder Clock Select to direct the Port clock to the Transcoder
 - b. Configure and enable planes (VGA or hires). This can be done later if desired.
 - c. If VGA - Clear VGA I/O register SR01 bit 5
 - d. Enable panel fitter if needed (must be enabled for VGA)
 - e. Configure transcoder timings and other pipe and transcoder settings

- f. Configure and enable TRANS_DDI_FUNC_CTL
 - g. Configure and enable TRANS_CONF
7. **Enable Port**
- 1. Configure voltage swing and related IO settings. Refer to the DDI Buffer section.
 - 2. If not type-C static connection, configure PORT_CL_DW10 Static Power Down to power up all lanes of the DDI.
 - 3. Configure and enable DDI_BUF_CTL
 - 4. Wait for DDI_BUF_CTL DDI Idle Status = 0b (Not Idle), timeout after 500 us.

Audio can be enabled after everything is complete. Follow the audio enable sequence in the audio registers section.

Notes

If the mode set fails, follow the disable sequence to disable everything that had been enabled up to the failing point.

Disable Sequence

Audio must be disabled first. Follow the audio disable sequence in the audio registers section.

1. **Disable Planes, Pipe, and Transcoder**
 - a. If VGA
 - i. Set VGA I/O register SR01 bit 5 for screen off
 - ii. Wait for 100 us
 - b. Disable planes (VGA or hires)
 - c. Disable TRANS_CONF
 - d. Wait for off status in TRANS_CONF, timeout after two frame times
 - e. Disable TRANS_DDI_FUNC_CTL with DDI_Select set to None
 - f. Disable panel fitter
2. **Disable Port**
 1. Disable DDI_BUF_CTL
 2. Wait 8 us or poll on DDI_BUF_CTL Idle Status for buffers to return to idle
 3. Configure Transcoder Clock Select to direct no clock to the transcoder
3. **If IO power is controlled through PWR_WELL_CTL - Disable IO Power**
 - a. Disable PWR_WELL_CTL DDI IO Power Request for the DDI that was used
4. **Disable Port PLL**
 - a. If PLL to port mapping is flexible, configure PLL to port mapping to direct no clock to the DDI
 - b. If this PLL is no longer needed, disable it
5. **Type-C static connection: Disable AUX I/O Power for the port**

6. **Type-C static connection: Clear DFLEXDPCSSS.DPPMSTC to '0' for the port**
7. **Disable Power Wells**
 1. Disable power wells that are no longer required, following the Sequences for Power Well

Sequences for WD

Enable Sequence

Display must already be initialized

1. **Enable Power Wells**
 1. Based on the resources to be used, enable the appropriate power wells following the Sequences for Power Wells
2. **Enable Planes, Pipe, and Transcoder**
 - a. Configure WD_STRIDE, WD_SURF, and WD_TAIL_CFG
 - b. Configure and enable planes (VGA or hires). This can be done later if desired.
 - c. If VGA - Clear VGA I/O register SR01 bit 5
 - d. Enable panel fitter if needed (must be enabled for VGA)
 - e. Configure transcoder timings and other pipe and transcoder settings
 - f. Configure and enable TRANS_WD_FUNC_CTL
 - g. Configure and enable TRANS_CONF

If the mode set fails, follow the disable sequence to disable everything that had been enabled up to the failing point.

Disable Sequence

If using Triggered Capture Mode (TRANS_WD_FUNC_CTL) wait for the frame to complete (polling WD_FRAME_STATUS Frame Complete or using the WD Frame Complete interrupt) before starting the disable sequence.

1. **Disable Planes, Pipe, and Transcoder**
 - a. If VGA
 - i. Set VGA I/O register SR01 bit 5 for screen off
 - ii. Wait for 100 us
 - b. Disable planes (VGA or hires)
 - c. If using triggered capture mode, wait for current capture to complete, and do not trigger any more captures
 - d. Disable TRANS_CONF
 - e. Disable TRANS_WD_FUNC_CTL
 - f. Disable panel fitter
2. **Disable Power Wells**



1. Disable power wells that are no longer required, following the Sequences for Power Wells

Sequences for Display C5 and C6

Display C5 (DC5) is a power saving state where hardware dynamically disables power wells and PLLs and saves the associated registers.

DC5 can be entered when software allows it, only certain power wells are enabled, and hardware detects that all pipes are disabled or low power pipe(s) are enabled with PSR active.

Display C6 (DC6) is a deeper power saving state where hardware dynamically disables power and saves the associated registers.

DC6 can be entered when software allows it, the conditions for DC5 are met, and the PCU allows DC6.

DC6 is required for S0ix.

The context save and restore program is reset on cold boot, warm reset, PCI function level reset, and hibernate/suspend.

Pipe 3D LUT contents are lost during DC5 and DC6.

Ports A IO/PHY programming is preserved when DC6 is enabled. The programming for other combo PHY ports may not be. After disabling DC6, before enabling the other ports, follow the combo phy initialization sequence and enable Aux IO Power for those ports.

Sequence to Allow DC5 or DC6

1. Load the correct stepping specific Display Context Save and Restore (CSR) program from the binary package.
 - a. Read the package header and extract the correct individual firmware. Binary package format details can be found in sections below.
 - b. Skip the header section at the start of the program binary.
 - c. Copy the payload into Display CSR Program Storage.
 - d. Perform the MMIO writes specified in the header section.
2. Configure display engine to disable power wells from table below, following the appropriate mode set disable sequences for any ports using those power wells. This can be done earlier if desired. DMC will prevent entry if these wells are not disabled.
 - Disable power wells PG3 and greater
3. Set display register 0x45520 bits [1:0] to 2'b11. The bits do not need to be cleared at any time.
4. Set DC_STATE_EN Dynamic DC State Enable = "Enable up to DC5" for DC5 or "Enable up to DC6" for DC6.

Programming Note	
Context:	Display C5 and C6
<p>Do not switch between "Enable up to DC5" and "Enable up to DC6" without moving to "Disable" and reloading the CSR program in between.</p> <p>Disable DC5/DC6 during mode set and re-enable after the mode set programming is completed.</p> <p>Disable DC5 and DC6 before a DDI AUX channel transaction is sent.</p> <p>MMIO accesses have more latency when DC5/DC6 is enabled. For optimal performance, disable DC5/DC6 when programming a set of registers and re-enable them after the programming is completed.</p>	

Sequence to Disallow DC5 and DC6

1. Set DC_STATE_EN Dynamic DC State Enable = "Disable".

MIPI DSI

Hardware does not support DC5 or DC6 with MIPI DSI enabled.

DMC Firmware Package

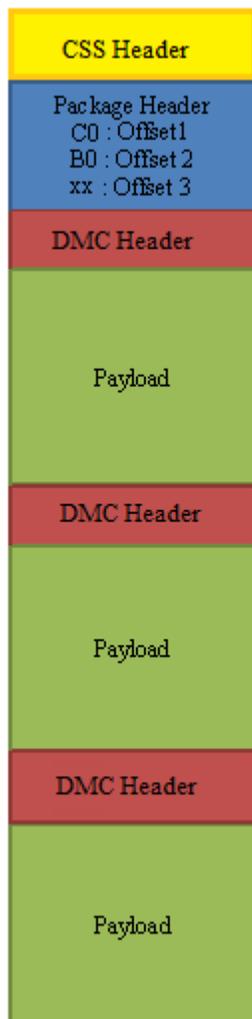
Display Micro-Controller firmware package includes all the firmwares that are required for different steppings of the product. The stepping dependent firmwares are all packaged and released as a single binary package. The package contains the CSS header, followed by the package header and the actual DMC firmwares.

Packaged firmware uses the following naming convention - <project>_dmc_ver<major>_<minor>.bin. The major version will get incremented whenever there is a change in the header layout and would require an update to the driver firmware loading module.

Major version 2

CSS Header	1.0
Package Header	2
DMC Header	3

Package Layout



CSS Header

```
typedef struct _CssHeader {
uint32_t moduleType; // 0x09 for DMC
uint32_t headerLen; // CSS header length in dwords
uint32_t headerVer; // 0x10000
uint32_t moduleID; // Not used
uint32_t moduleVendor; // Not used
uint32_t date; // YYYYMMDD(YYYY << 16 + MM << 8 + DD)
uint32_t size; // Total dmc fw binary size in dwords - (CSS_HeaderLen +
PackageHeaderLen + dmc FWsLen)/4
uint32_t keySize; // Not used
uint32_t modulusSize; // Not used
```

```
uint32_t exponentSize; // Not used
uint32_t reserved1[12]; // Not used
uint32_t version; // Major Minor
uint32_t reserved2[8]; // Not used
uint32_t uKernelHeaderInfo; // Not used
} CssHeader;
```

Package Header

Package header contains the firmware/stepping mapping table and the corresponding firmware offsets to the individual binaries, within the package. Mapping table will list the exceptions first, followed by the default entries. An Offset value of "0xFFFFFFFF" in the mapping table indicates that there is no firmware available/supported for that stepping. The offsets to the individual binary are DWord aligned. The first individual binary starts at an offset value of "0x00000000" after the CSS Header and the Package Header.

Stepping/Version mapping example

Stepping	FW Version
A1	1.1
B*	1.6
**	2.3

DMC IDs

DMC	DMC ID
Main	0
Pipe A	1
PipeB	2
PipeC	3
PipeD	4

```
typedef struct _PackageHeader {
uint8_t headerLen; // DMC package header length in dwords
uint8_t headerVer; // 0x02
uint8_t reserved[10]; // Reserved
uint32_t numEntries; // Number of valid entries in the FWInfo array below
struct _FWInfo_ {
uint8_t reserved1; // Reserved
uint8_t dmc_id; // DMC ID (Main-0, pipeA-1, pipeB-2, pipeC-3, pipeD-4)
```

```

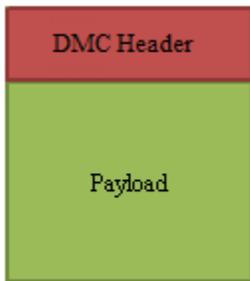
char stepping; // Stepping (A, B, C, ..., *). * is a wildcard
char substepping; // Sub-stepping (0, 1, ..., *). * is a wildcard
uint32_t offset; // FW offset within the package in dwords
uint32_t reserved2; // Reserved
} FWInfo[32];
} PackageHeader;

```

DMC firmware binary

Each individual DMC firmware binary has a header followed by a payload whose size is specified in the header section. Along with the version, length, firmware size etc. the header section also specifies a list of MMIO addresses and data. These MMIO write cycles, in the 0x80000 - 0x8FFFF address range, should be executed as part of the initial CSR program setup.

DMC firmware Layout



```

for i = 1 to <mmioCount>
Perform MMIO write to address <mmioaddr[i]> with data <mmiodata[i]>

```

DMC Header - Version 3

```

typedef struct _DMCHHeader {
    uint32_t signature; // 0x40403E3E
    uint8_t headerLen; // DMC specific header length in
dwords uint8_t headerVer; // 0x03 uint16_t
    dmccVer; // dmcc compiler version
    uint32_t project; // Major, Minor: 0x00090000, 0x00090001,
0x000A0000
    uint32_t fwSize; // Firmware program size in dw
    uint32_t fwVersion; // Major Minor
    uint32_t startMMIOAddr; // DMC RAM start MMIO address
    uint32_t reserved[9];
    uint8_t dfile [32]; // .d file name (encoded using
ascii - 40)

    uint32_t mmioCount; // number of mmio uint32_t mmioaddr[20]; // MMIO address uint32_t
mmiodata[20]; // MMIO data
} DMCHHeader;

```

Header field	Value
project	0x000C0000

eDP Sequences for Display C3 Clock Off

eDP Display C3 Clock Off (DC3co) is a power saving state where hardware dynamically disables the CDCLK PLL, DBUF power, and Panel Fitter power when PSR2 is idle between frame updates.

DC3co is entered when software allows it and hardware detects that PSR2 is idle.

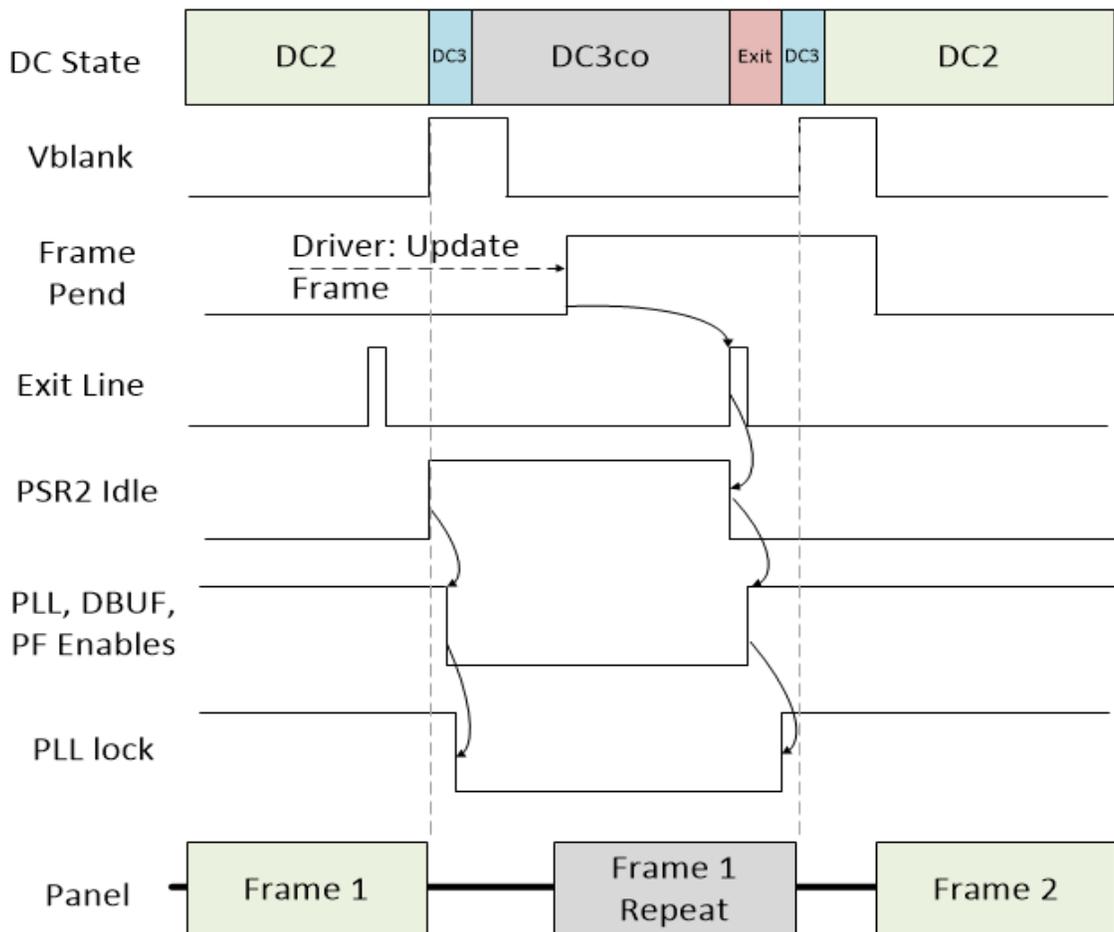
When the CDCLK PLL is disabled, display engine runs CDCLK from a slower SoC clock to maintain accessibility.

DC3co is exited when software disallows it or PSR2 becomes active to send an updated frame.

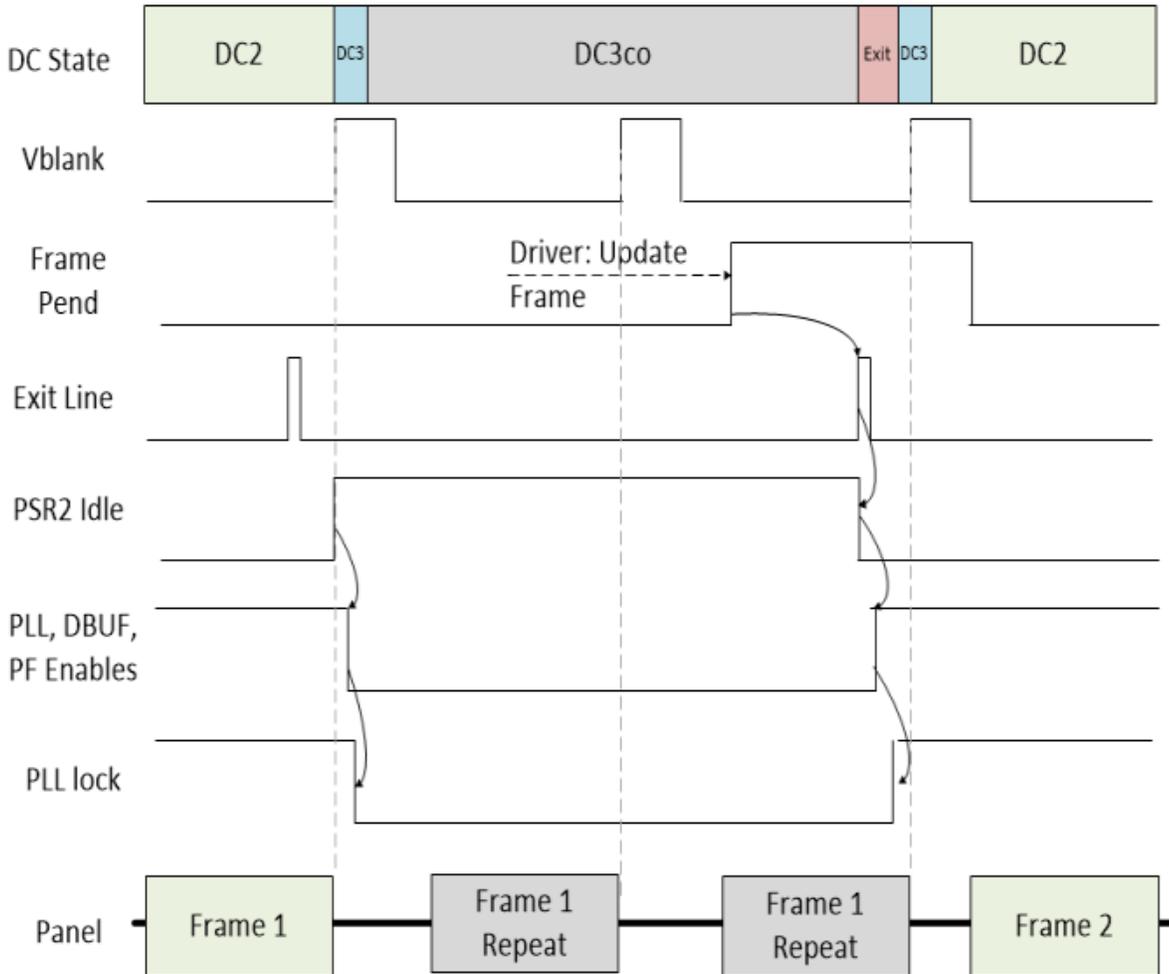
DC3co is controlled by a context save and restore program which has to be loaded before allowing DC3co and is reset on DC9, cold boot, warm reset, PCI function level reset, and hibernate/suspend.

DC3CO will not work with pipeA and DDIB combination. It only works with pipeA and DDIA.

DC3co with Single Idle Frame



DC3co with Multiple Idle Frames



Requirements

Audio codec idle and disabled.

External displays disabled. WD transcoders and DP/HDMI transcoders must be disabled.

Backlight cannot be driven from the display utility pin. It can be driven from the south display.

This feature should be enabled only on eDP.

DC5 and DC6 not allowed when this feature is enabled.

- DC5 and DC6 saves more power, but cannot be entered during usages like video playback because there are not enough idle frames in a row to meet most panel PSR2 deep sleep entry requirements (typically 4 frames). Software must select between DC5 and DC6 versus DC3co based on flip rate or other hints.

PSR2 deep sleep disabled (PSR2_CTL Idle Frames = 0000b).

Disable DC3co before mode set, or other Aux, PLL, and DBUF programming, and do not re-enable until after that programming is completed.

DC3co is only compatible with PSR2. Follow the sequences to enable and disable PSR2 and DC3co at the correct times.

Sequence to Allow DC3co

1. If not already loaded, load the correct stepping specific Display Context Save and Restore (CSR) program from the binary package.
 - a. Read the package header and extract the correct individual firmware. Binary package format details explained in the Sequences for Display C5 and C6 section.
 - b. Skip the header section at the start of the program binary.
 - c. Copy the payload into Display CSR Program Storage.
 - d. Perform the MMIO writes specified in the header section.
2. Configure display engine to meet the requirements for DC3co, following the appropriate disable sequences. This can be done earlier if desired.
 1. Disable DC states by programming DC_STATE_EN Dynamic DC State enable = 00b
 2. Disable PSR2 following the Panel Self Refresh section PSR2 disable sequence.
 3. Disable PSR2 deep sleep by programming PSR2_CTL Idle Frames = 0000b
 4. Enable PSR2 following the Panel Self Refresh section PSR2 enable sequence.
3. Configure and enable the PSR2 transcoder's Early Exit line (TRANS_EXITLINE)
 - a. DC3co exit time = Polling rate + instruction execution time + delays from driver MMIO access + DBUF enable time + PLL lock time + guardband = 200us
 - b. $\text{ROUNDUP}(\text{DC3co exit time} / \text{line time}) + 1$
4. Set DC_STATE_EN Display Clock Off Enable = 1b.

Sequence to Disallow DC3co

1. Clear DC_STATE_EN Display DC*CO State Status by writing a 0 to it.
2. Set DC_STATE_EN Display Clock Off Enable = 0b.
3. Wait DC3co exit time for disable to complete. See above for exit time latency calculation above.
 - This ensures exit has completed and it is safe to do things such as program PLL or DBUF registers.
4. Disable PSR2 following the Panel Self Refresh section PSR2 disable sequence.
5. Restore PSR2 deep sleep by programming correct allowed Idle Frames in PSR2_CTL.
6. If PSR2 is still needed, re-enable PSR2 following the Panel Self Refresh section PSR2 enable sequence.

Sequences for Display C9

Display C9 (DC9) is a power saving state where the display engine is powered off.



DC9 supports S0ix with more power savings than DC6.

Display software must follow certain programming sequences to allow or dis-allow DC9.

Hardware will dynamically enter and exit DC9 when allowed, saving and restoring some of the display state.

Sequence to Allow DC9

1. Follow Sequence to Disallow DC5.
2. Disable all display engine functions using the full mode set disable sequence on all pipes, transcoders, ports, planes, and power wells above PG1.
3. Disable and mask all graphics interrupts in north and south display.
4. Save state of MMIO display registers.
 - The exact registers depend on software policy.
 - Hardware will save and restore the PCI Config and DGunit registers
5. Follow Sequences to Initialize Display - Un-initialize Sequence.
6. Set DC_STATE_EN DC9 Allow to 1b.

Sequence to Disallow DC9

1. Clear DC_STATE_EN DC9 Allow to 0b.
2. Follow Sequences to Initialize Display - Initialize Sequence.
3. Restore state of MMIO display registers:
 - The exact registers depend on software policy.
 - The context save and restore program is reset on DC9 and has to be restored.
4. Enable and unmask graphics interrupts as needed.

X_{D+}^e Resolution Support

A display resolution is only supported if it meets all the restrictions below for Maximum Pipe Pixel Rate, Maximum Port Link Rate, Maximum Size, Maximum Bandwidth, and Maximum Watermark.

Core Display Clock (CDCLK)

Refer to the Clocks section for details of the frequencies.

Maximum Pipe Pixel Rate

The display resolution must fit within the maximum pixel rate output from the pipe.

For each enabled plane on the pipe {

 If plane scaling enabled {

 Plane Ratio = 1 / Plane down scale amount

Refer to the Watermark Calculations section, Scaling sub-section, for down scale calculation.

```

}
Else {
    Plane Ratio = 1
}
}

```

Pipe Ratio = Minimum Plane Ratio of all enabled planes on the pipe

If pipe scaling is enabled {

Pipe down scale amount = Horizontal down scale amount * Vertical down scale amount

Refer to the Watermark Calculations section, Scaling sub-section, for down scale calculation.

Pipe Ratio = Pipe Ratio / Pipe down scale amount

```

}

```

Pipe maximum pixel rate = 2 * CDCLK frequency * Pipe Ratio

YUV420 Full Blend Mode

Pipe Maximum Y channel pixel rate = 2 * CDCLK frequency * Y channel Pipe ratio

Resolutions Requiring Joined Pipes

For resolutions requiring multiple pipes to be combined together (pipe joining), the pixel rate seen by each pipe is 1/<# of joined pipes> of the pixel rate of the full resolution.

The overlapping excess horizontal pixels added for scaling smoothly across the seam between pipes do not impact the pixel rate.

For example: 7680x4320 CVT1.2 RB1 pixel rate is 2089.75 MHz. That is split across 2 pipes, so each pipe is 3840x4320 with a pixel rate of 1044.875 MHz.

Maximum Port Link Rate

The display resolution must fit within the maximum link rate for each port type.

Refer to the project overview sections for the maximum rates.

YUV420 Full Blend Mode

Maximum Port Link Rate = CDCLK frequency * Y channel Pipe Ratio * (bits per color / 8)

Maximum Size

There are limits on the maximum horizontal and vertical size.



Plane stride maximum listed in PLANE_STRIDE register.

Multiple pipes can be joined together side by side in the display engine to drive a single port or a tiled panel.

Unjoined pipe source size, plane size, and pipe active size maximum vertical 4096.
Unjoined pipe source size, plane size, and pipe active size maximum horizontal 5120.
Joined pipes active size maximum is 7680x4320, with each pipe and maximum horizontal plane size at 7680/<# of joined pipes> and vertical size at 4320, plus extra overlapping horizontal pixels for scaling excess.
PSR2 maximum pipe horizontal active size 5120 pixels. PSR2 has additional restrictions on minimum vblank size and horizontal total line time. See the PSR section for those.
LACE DPST maximum size 5120x3200 pixels.
Pipe scalers maximum horizontal source size 5120 pixels.

Maximum Bandwidth

Refer to the Bandwidth Restrictions section

Maximum Watermark

The display resolution must not exceed the level 0 maximum watermark value. See the section on Watermark Programming.

Example Display Resolution Capabilities

Refer to the project overview sections for the common resolutions that meet all the resolution restrictions.

Examples

Example pipe pixel rate

Plane 1 enabled at 64bpp and plane down scale amount 1.25, plane 2 enabled at 32bpp, no pipe scaling enabled, and CDCLK 312 MHz:

Plane 1 ratio = $1/1.25 = 0.8$

Plane 2 ratio = 1

Pipe ratio = $\text{Minimum}[1, 0.8] = 0.8$

Pipe maximum pixel rate = $2 * 312 \text{ MHz} * 0.8 = 499.2 \text{ MHz}$

Bandwidth Restrictions

There are bandwidth restrictions that limit the display resolution and configuration. A display resolution is only supported if it meets all the restrictions below for Maximum Memory Read Bandwidth found from the Available Memory Bandwidth Calculation and Required Memory Bandwidth Calculation, Maximum Data Buffer Bandwidth, and Maximum Pipe Read Bandwidth.

Maximum Memory Read Bandwidth

The display required memory bandwidth for a resolution configuration must not exceed the available memory bandwidth, considering efficiency and GV points.

Calculate the available memory bandwidth and the required memory bandwidth across all GV points. Refer to the SAGV chapter for the flow to find GV points and restrict those that do not provide enough available memory bandwidth. If no points provide enable available memory bandwidth, then the display configuration can not be supported.

Note: These calculations are valid even if GV points are disabled. They then use the single available GV point.

Available Memory Bandwidth Calculation

This calculates the memory bandwidth available for display. It considers the efficiency, impacted by memory type and configuration details and number of enabled planes, and differences between GV points.

The InitializeBandwidthAndPlanes function populates arrays with the bandwidth and number of planes that can be enabled in several groupings for each GV point. See the SAGV chapter for information on retrieving the memory configuration to populate the structs at the start of the function.

The CalculateMaxBW function returns the bandwidth available for a specified number of enabled planes and GV point.

```
#define MIN(a, b) (a < b ? a : b)
#define MAX(a, b) (a > b ? a : b)

inline int divide_roundup(int n, int d) {
    return (n + d - 1) / d;
}

inline int isYtile() {
    return TRUE if any plane is using tileY surface format, and do not count tile4 as tileY
}

struct PCODE_GLOBAL_INFO
{
    uint32_t DdrType : 4;
    uint32_t NumOfPolulatedChannels : 4;
    uint32_t NumOfEnabledQgvPoints : 4;
    uint32_t NumOfEnabledPsf0Points : 4;
};
PCODE_GLOBAL_INFO PcodeGlobalInfo();

struct PCODE_QGV_POINT
{
    uint32_t Dclk : 16; // in 16.66666MHz units
    uint32_t tRP : 8; // In DCLKs
    uint32_t tRCD : 8; // In DCLKs
    uint32_t tRDPRE : 8; // In DCLKs
    uint32_t tRAS : 9; // In DCLKs
}
```



```
};
PCODE_QGV_POINT PcodeQgvPoint(int point);

// Number of plane groups.
#define NUM_GROUPS 6

// Max number of Qclk GV points
#define NUM_SAGV_POINTS 4

// These state variables should be stored in a device context. The initialize function will
generate the records
int g22_deratedbw[NUM_SAGV_POINTS][NUM_GROUPS];
int h24_maxplanes[NUM_GROUPS];

// Find out the memory timings for each of the SAGV points and initialize the state variables
above.
void InitializeBandwidthAndPlanes()
{
    // Static config for all gens
    int c3_derating = 10;
    int c2_ipqdepthpch = 16;
    int c25_deprogbwplimit = 60; // %
    int c7_mempagesize = 16;
    int g4_numchannels;
    int g4_maxnumchannels;

    // The following require the appropriate SAGV timings, however they can be precalculated
for each of the SAGV points
    int g3_clk[NUM_SAGV_POINTS];
    int g7_trp[NUM_SAGV_POINTS];
    int g8_trcd[NUM_SAGV_POINTS];
    int g10_trdpred[NUM_SAGV_POINTS];
    int g11_trc[NUM_SAGV_POINTS];

    PCODE_GLOBAL_INFO mem_global_info = PcodeGlobalInfo();
    g4_numchannels = mem_global_info.NumOfPolulatedChannels;
    MEM_TYPE mem_type = (MEM_TYPE)mem_global_info.DdrType;

    for (uint32_t sagv = 0; sagv < NUM_SAGV_POINTS; sagv++)
    {
        if (sagv < mem_global_info.NumOfEnabledQgvPoints)
        {
            PCODE_QGV_POINT gv_point_info = PcodeQgvPoint(sagv);
            g3_clk[sagv] = (16667 * gv_point_info.Dclk + 500) / 1000;
            g7_trp[sagv] = gv_point_info.trp;
            g10_trdpred[sagv] = gv_point_info.trdpred;
            g11_trc[sagv] = gv_point_info.trp + gv_point_info.tras;
            g8_trcd[sagv] = gv_point_info.trcd;
        }
        else
        {
            // Initialize any undefined GV points to zero
            g3_clk[sagv] = 0;
            g7_trp[sagv] = 0;
            g10_trdpred[sagv] = 0;
            g11_trc[sagv] = 0;
            g8_trcd[sagv] = 0;
        }
    }

    // Detect the maximum clock from the SAGV points
    int clk_max = 0;
    for (uint32_t sagv = 0; sagv < NUM_SAGV_POINTS; sagv++)
        clk_max = MAX(clk_max, g3_clk[sagv]);
}
```

```

// Project specific config
int c4_deBurst;
int c14_displayrtids;
int c24_deprogbwlimit;
int g9_tBL;
int g6_channelwidth;
int g5_deinterleave;

c4_deBurst = 16;
c14_displayrtids = 256;
switch (mem_type)
{
case DDR4:
    g4_maxnumchannels = 2;
    g5_deinterleave = isYtile() ? 1 : 2;
    g9_tBL = isYtile() ? 8 : 4;
    g6_channelwidth = 64;
    break;
case DDR5:
    g4_maxnumchannels = 4;
    g5_deinterleave = isYtile() ? 1 : 2;
    g9_tBL = isYtile() ? 16 : 8;
    g6_channelwidth = 32;
    break;
case LPDDR4:
case LPDDR5:
    g4_numchannels *= 2; // convert to number of subchannels
    g4_maxnumchannels = 8;
    g5_deinterleave = isYtile() ? 2 : 4;
    g9_tBL = 16;
    g6_channelwidth = 16;
    break;
default:
    assert(0); // not supported
    break;
}

c24_deprogbwlimit = 25;
if (g4_numchannels < g4_maxnumchannels)
{
    // Halve the DE interleave if channels are not present
    g5_deinterleave = MAX(g5_deinterleave / 2, 1);
}

int g13_peakbw = g4_numchannels * (g6_channelwidth / 8) * clk_max;
int g14_maxdebw = MIN(c24_deprogbwlimit * 1000, (g13_peakbw * c25_deprogbwplimit) / 100);
int g17_ipqdepth = MIN(c2_ipqdepthpch, c14_displayrtids / g4_numchannels);

h24_maxplanes[0] = 0;
int c8_4kpagespermempage = c7_mempagesize / 4;
int c9_clperchperblock = (8 / g4_numchannels) * g5_deinterleave;
int g16_clperchgroup = c8_4kpagespermempage * c9_clperchperblock;

// Figure out the max plane groups
for (int i = 0; i < NUM_GROUPS; i++)
{
    int g18_clpchgroup = (c4_deBurst * g5_deinterleave / g4_numchannels) << i;
    if (i < NUM_GROUPS - 1)
    {
        if (g18_clpchgroup < g16_clperchgroup)

```

```

        {
            h24_maxplanes[i + 1] = (g17_ipqdepth - g18_clpchgroup) / g18_clpchgroup;
        }
        else
        {
            h24_maxplanes[i + 1] = 0;
        }
    }
    for (uint32_t sagv = 0; sagv < NUM_SAGV_POINTS; sagv++)
    {
        int g20_CT = MAX(g11_tRC[sagv], g7_tRP[sagv] + g8_tRCD[sagv] + (g18_clpchgroup - 1) *
g9_tBL + g10_tRDPRE[sagv]);
        int g21_BW = g18_clpchgroup * 32 * g4_numchannels * g3_clk[sagv] / g20_CT;
        g22_deratedbw[sagv][i] = MIN(g14_maxdebw, ((100 - c3_derating) * g21_BW) / 100);
    }
}

// This function will return the maximum bandwidth that can be consumed by display when the
specified number of planes are enabled
// at the specified SAGV point. It should be used at run time to limit the modes and the
planes allowed.
int CalculateMaxBW(int numPlanes, int point)
{
    for (int i = NUM_GROUPS - 1; i >= 0; i--)
    {
        if (numPlanes <= h24_maxplanes[i])
        {
            return g22_deratedbw[point][i];
        }
    }
    return g22_deratedbw[point][0];
}

```

Required Memory Bandwidth Calculation

This calculates the display required memory bandwidth.

Note on planar formats: The NV12 format counts as 2 planes of 2 Bpp each. The P01x formats count as 2 planes of 4 Bpp each. Vertical sub-sampling does not reduce this calculation because the bandwidth has to be counted over a few microsecond period.

DBUF maximum data buffer bandwidth MB/s = CDCLK frequency MHz * 64 Bytes

For each DBUF {

DBUF maximum pipe bandwidth MB/s = DBUF maximum data buffer bandwidth / number of enabled pipes using this buffer

For each pipe using this buffer {

Pipe cumulative bytes per pixel = 0

DBUF maximum plane bandwidth MB/s = DBUF maximum pipe bandwidth / number of enabled planes

For each plane enabled on the pipe { // cursor can be ignored because it only impacts bandwidth for a very short time

Plane required bandwidth MB/s = pixel rate MHz * source pixel format in bytes * plane down scale amount * pipe down scale amount

Display required memory bandwidth MB/s += Plane required bandwidth

Pipe cumulative bytes per pixel += plane source pixel format in bytes

If plane required bandwidth > DBUF maximum plane bandwidth {Return: Failure maximum data buffer bandwidth is exceeded}

}

If pipe cumulative bytes per pixel > (CDCLK frequency MHz / (pixel rate MHz * plane down scale amount * pipe down scale amount)) * 51.2 {Return: Failure maximum pipe read bandwidth is exceeded}

}

}

Return: Display required memory bandwidth MB/s

Maximum Data Buffer Bandwidth

The display resolution must not exceed the maximum bandwidth from each display data buffer (DBUF). There are multiple buffers that can be enabled, and planes can be allocated data blocks from multiple. See the Display Buffer Programming section for requirements on how and when to use multiple buffers.

This is calculated as part of the Required Memory Bandwidth Calculation where it returns a failure if maximum data buffer bandwidth is exceeded. The calculation for this check can be separated out if required.

Maximum Pipe Read Bandwidth

The display resolution must not exceed the maximum memory read bandwidth from each display pipe. This restriction is met by limiting the cumulative bytes per pixel for the pipe, which accounts for the limits of memory reads, translation reads, and data buffer reads.

This is calculated as part of the Required Memory Bandwidth Calculation where it returns a failure if maximum pipe read bandwidth is exceeded. The calculation for this check can be separated out if required.



Clocks

X_D Clocks

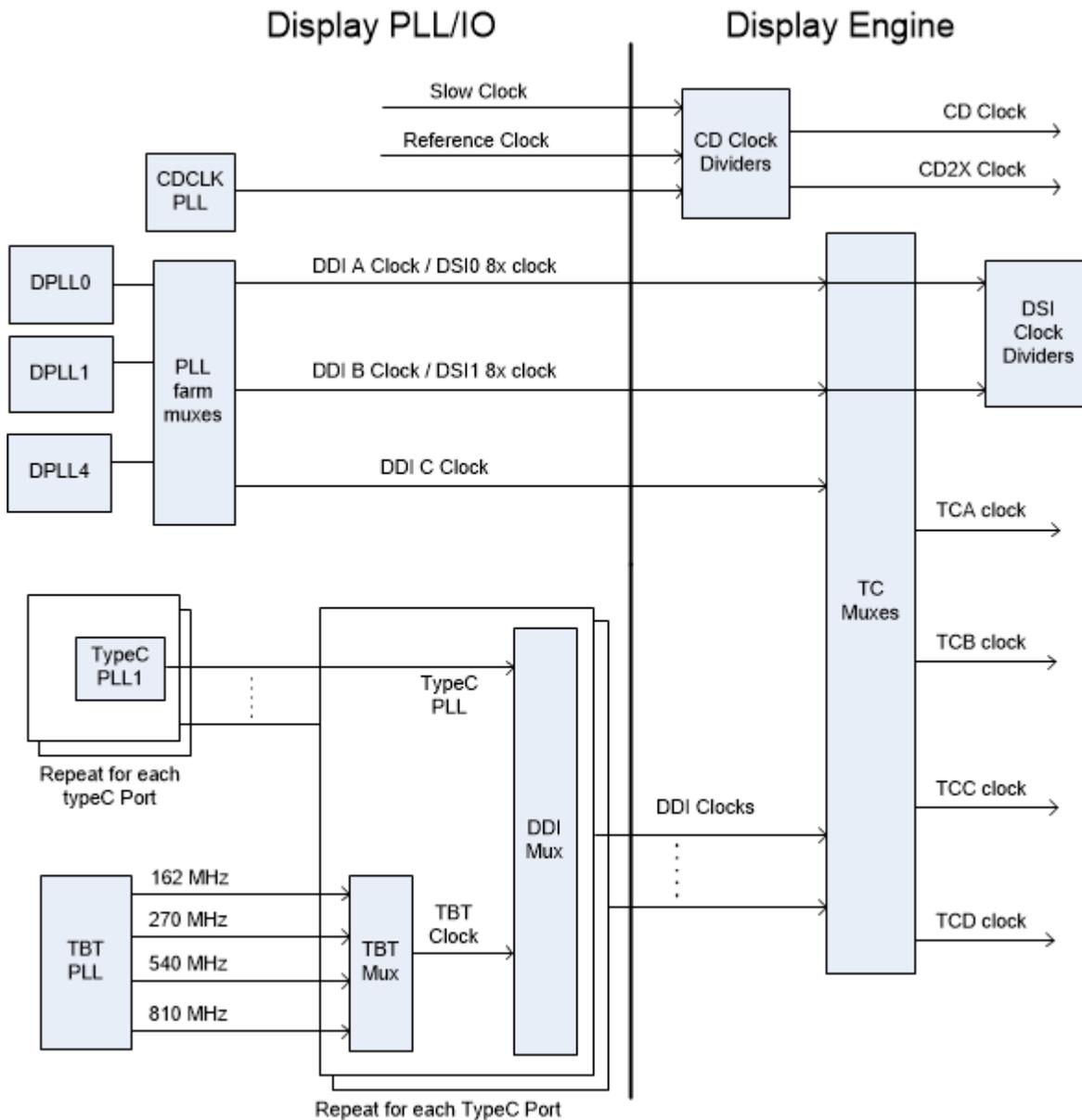
Registers

Registers
CDCLK_CTL
CDCLK_PLL_ENABLE
DPLL_CFGCR0
DPLL_CFGCR1
DPLL_SSC
DPCLKA_CFGCR0
DPLL_ENABLE
TRANS_CLK_SEL
DDI_CLK_SEL
TIMESTAMP_CTR
DKL_PLL_DIV0
DKL_PLL_DIV1
DKL_PLL_LF
DKL_PLL_FRAC_LOCK
DKL_REFCLKIN_CTL
DKL_CLKTOP2_HSCLKCTL
DKL_CLKTOP2_CORECLKCTL1

Overview of Display Clock Paths

The display engine clocking structure has multiple PLLs and clocks. The flow is from PLL to DDI (port) clock to transcoder clock.

PLL Arrangement



Display Engine Clock Reference

There is one display engine clock reference.

	Reference
Usage	Reference for the PLLs and for miscellaneous timers in display engine
Frequency	19.2, 24, 38.4 MHz Non-SSC (Register DSSM Reference Frequency indicates the frequency)
Default after reset	Enabled
Programming	Not programmable by display software.



Combo PHY PLLs

DPLLs are used for the combo PHY ports.

The PLL output frequencies are 5x the symbol/TMDS rate, which is 1/2 the bit rate.

The PLL output is divided by 5 to become the symbol/TMDS clock frequency used in the display engine.

Both edges of the PLL output are used in the I/Os to double the frequency to bit clock rate.

A single PLL output may be used by multiple ports simultaneously if those ports all require the same frequency and spread characteristics.

	Combo PHY PLLs (DPLLs)
Usage	Sources for DSI0, DSI1 clocks, and DDI clocks for the combo PHY ports (non-typeC)
Input	Reference clock
Frequency	Programmable eDP/DP link bit rates: 1.62, 2.16, 2.7, 3.24, 4.32, 5.4, 6.48*, 8.1* GHz, SSC and Non-SSC HDMI/DVI symbol rates: 20 to 600 MHz, Non-SSC** DSI 8x clock rates: 0.340 - 2.5 GHz, SSC and Non-SSC *OEM must use VBT to specify a maximum that is tolerated by the board design. *Frequencies between 500 and 533.2 MHz (exclusive) not supported
Default after reset	Disabled
Programming	Must be programmed by software when enabling and disabling a display output. See the section on Port Clock Programming. PLLs are automatically disabled and re-enabled by hardware for some power states.

Thunderbolt (TBT) PLL

DPLL2 is used for thunderbolt ports.

The PLL output is the symbol clock frequencies used in the display engine.

This PLL is used to deliver symbols from the display engine to the thunderbolt controller. The thunderbolt controller will re-clock and convert the symbols to the thunderbolt format.

The PLL outputs all 4 thunderbolt frequencies simultaneously.

	TBT PLL
Usage	Source for DDI clocks on typeC ports when using thunderbolt
Input	Reference clock
Frequency	PLL fixed frequency divided to give simultaneous DP thunderbolt link rates: 162, 270, 540, and 810 MHz, Non-SSC* *SSC, if needed, will be applied on the link by the thunderbolt controller
Default after reset	Disabled
Programming	Must be programmed by software when enabling and disabling a display output. See the section on Port Clock Programming.

TypeC PLLs

The TypeC PLLs are used for Type C ports. There are 2 TypeC PLLs per port, but display engine can only use one PLL, which the register offsets select.

The PLL output frequencies are 5x the symbol/TMDS rate, which is 1/2 the bit rate.

The PLL output is divided by 5 to become the symbol/TMDS clock frequency used in the display engine.

Both edges of the PLL output are used in the DDI I/Os to double the frequency to bit clock rate.

The TypeC PLL is also called a Dekel (DKL) PLL.

	TypeC PLL per port
Usage	Sources for DDI clocks on type C ports when using DP alternate mode or legacy DP or HDMI connectors
Input	Reference clock
Frequency	Programmable eDP/DP/DP-altmode link bit rates: 1.62, 2.7, 5.4, 8.1 GHz, SSC and Non-SSC HDMI/DVI symbol rates: 20 to 600 MHz, Non-SSC
Default after reset	Disabled

Programming	Must be programmed by software when enabling and disabling a display output. See the section on Port Clock Programming.
--------------------	---

DDI clocks

There is one DDI clock tied to each DDI port.

A single DDI clock output may be used by multiple transcoders simultaneously for DisplayPort Multi-streaming.

	Combo PHY (non-typeC) DDI clocks	TypeC DDI clocks
Usage	DDI ports I/O bit clock and symbol/TMDS clock, source for transcoder clocks	
Input	Programmable selection between the DPLLs	Programmable selection between TypeC PLLs and TBT PLL outputs
Frequency	DPLL frequency	TypeC PLLs or TBT frequency
Default after reset	Disabled	
Programming	Must be programmed by software when enabling and disabling a display. Programming is done through the DPLL, PORT_CLK_SEL, TypeC, and TBT registers. See the section on Port Clock Programming.	

Transcoder Clocks

There is one Transcoder clock tied to each display transcoder.

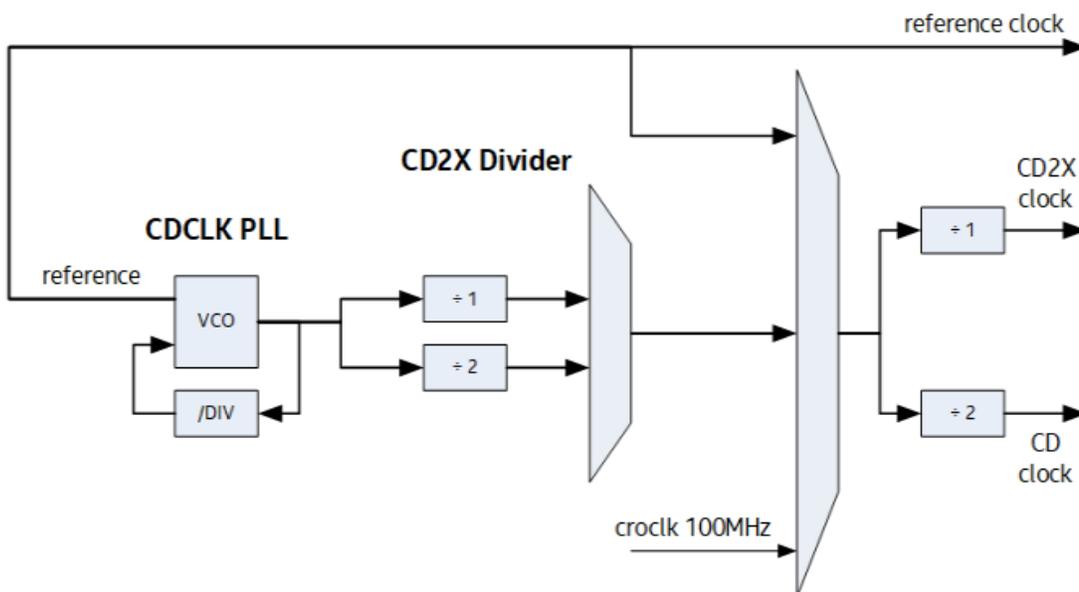
	Transcoder clocks
Usage	Transcoder symbol/TMDS clocks
Input	Programmable selection between DDI clocks
Frequency	PLL output frequency divided by 5
Default after reset	Disabled
Programming	Must be programmed by software when enabling and disabling a display. Programming is done through the TRANS_CLK_SEL registers. See the section on Port Clock Programming.

CD Clock

CD clock refers to the Core Display clock which includes the Core Display 1X Clock (CD clock, CDclk, cdclk, CDCLK) and the Core Display 2X Clock (CD2X clock, cd2xclk, CD2XCLK).

	CD clock
Usage	Clocking for most display engine functions
Input	CDCLK PLL output or reference clock.
Frequency	CDCLK PLL disabled - reference or slow clock divided by 2 CDCLK PLL enabled - Frequencies in table above
Default after reset	Running on the reference clock
Programming	Must be programmed by software when enabling and disabling a display. Programming is done through the CDCLK_CTL and CDCLK_PLL_ENABLE registers.

CD Clock Generation



CD Clock PLL

The CDCLK PLL is the main source for CD clock. A programmable divider inside the PLL controls the PLL frequency. The CD2X Divider after the PLL provides additional dividing to create the CD2X clock used by the display engine. The programmable divider value must be set to account for the reference frequency for in the **DSSM** Reference Frequency field.

When the CDCLK PLL is disabled, the CD clock runs at reference clock divided by 2 to maintain register accessibility. The slow clock (crock) is used by hardware for some power state transitions.



CDCLK PLL	
Alternate Name	ADPLL or DEPLL
Usage	Source for CD clock
Input	Reference clock
Frequency	Programmable - Frequencies in table below
Default after reset	Disabled
Programming	Must be programmed by software when enabling and disabling a display. May be automatically enabled and disabled by hardware for some power states. Programming is done through the CDCLK_PLL_ENABLE register.

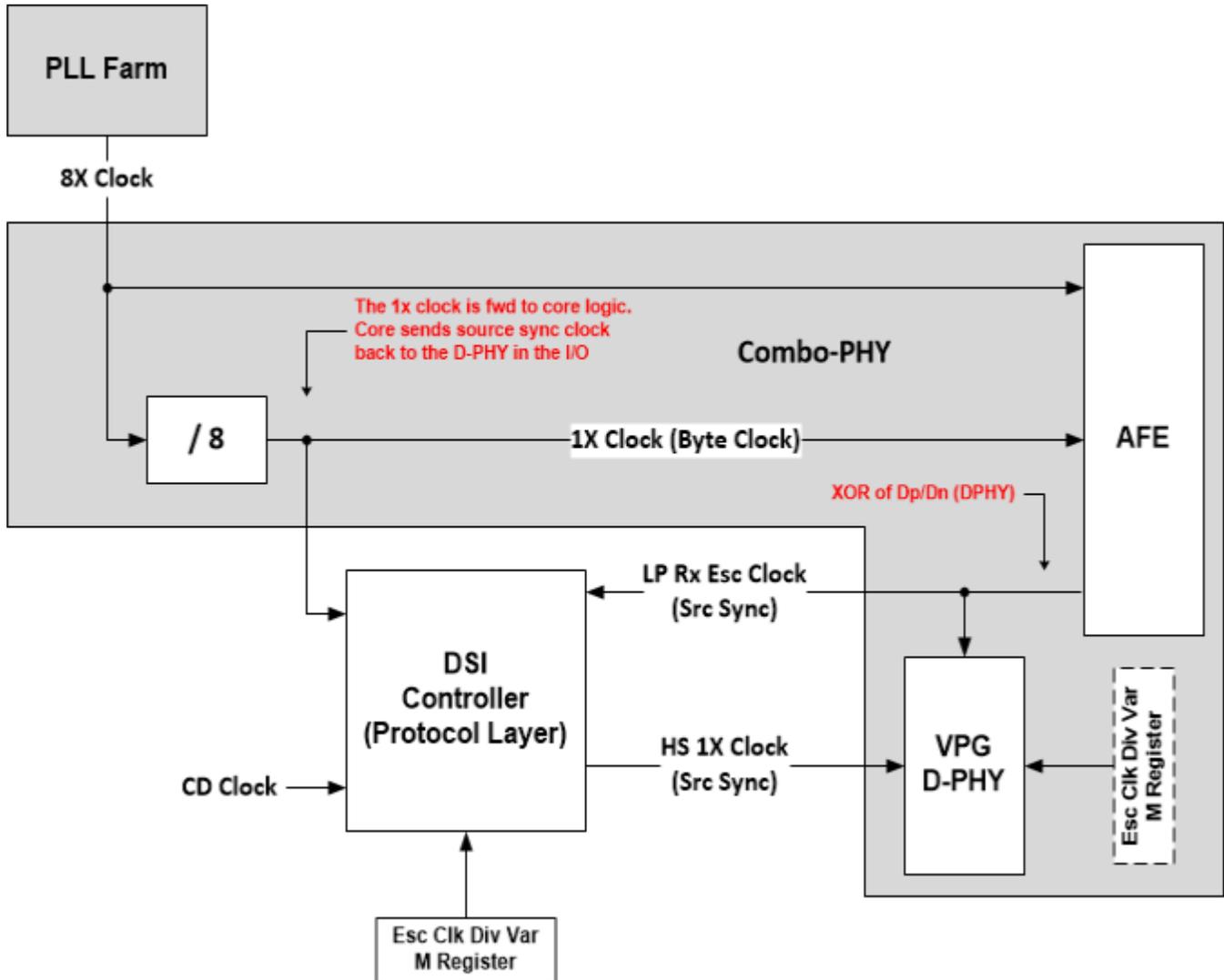
CDCLK PLL Ratio and Divider Programming and Resulting Frequencies

PLL Ratio for 19.2 MHz Reference	PLL Ratio for 38.4 MHz Reference	CD2X Divider	CDCLK MHz
18	9	1	172.8
20	10	1	192
32	16	1	307.2
68	34	2	326.4
58	29	1	556.8
68	34	1	652.8

PLL Ratio for 24 MHz Reference	CD2X Divider	CDCLK MHz
15	1	180
16	1	192
26	1	312
54	2	324
46	1	552
54	1	648

MIPI DSI Clocks

The MIPI DSI clocks are used for the DSI 0 and DSI 1 ports and transcoders.



The DSI port operates within several different clock domains.

- **CD Clock** - The DSI controller receives the Pixel and Software control within this clock domain. The DSI controller clock crosses this information over to the 1X clock domain.
- **8X Clock** - This clock domain is received from the PLL farm and used to transfer the serialized data across the Link (i.e. it is the per-Lane frequency of the DSI Link). Only the I/O block will receive this clock. The IO contains a programmable divider that will divide this clock down to the 1X clock needed by the DSI controller.
- **1X Clock (a.k.a. Byte clock, Word clock)** - This clock domain is used to transfer the HS parallel data (a byte) per Data Lane. Both the DSI Controller and the D-PHY will receive this clock
- **Escape Clock** - This clock is used for LP communication across the DSI Link. Per all MIPI physical layer specifications, this clock has a fixed frequency that should be as close, but not greater than, 20 MHz.
 - **Tx Escape clock** - The Display Engine does not use this clock. Instead it emulates it based off of the DSI_ESC_CLK_DIV register programming and the 1X clock that the Display receives



from the IO. All of the LP Tx signaling from the controller to the DPHY will be driven off of the 1X clock, but will transition at an effective Tx Escape clock frequency.

- **Rx Escape clock** - The IO XOR's together the LP signaling received from the Panel to generate this clock. The IO forwards this clock to the DSI controller

Notes:

1. The MCD block (shaded in grey) delivers a 1X clock to the DSI Controller within the Display Core and to the AFE within the MCD block. The DSI Controller forwards this 1X clock back to the D-PHY within the MCD block along with its data (i.e. the data is source synchronous). The D-PHY will work entirely within the forwarded 1X clock domain. The inlet to the AFE contains a FIFO to handle the phase crossing between the forwarded clock that the D-PHY is operating within and the 1X clock the AFE receives directly from the PLL.
2. The LP Tx Escape clock is derived from a variable M that Software programs within the DSI_ESC_CLK_DIV register. The divider used to live within the IO, so the variable M represents the divisor for generating a clock based off of the 8X clock. Since the divider is within the Display and the clock it is operating on is a 1X clock the divisor becomes "N" which is equal to ceiling(M/8).

	DSI Clocks
Usage	DSI I/O and Transcoder
Input	Programmable selection between the DPLLs
Frequency	PLL clock is the 8X clock. 8X clocks have variable divider to create ~20 MHz escape clocks. Fixed divides for 4X clock and 1X (byte) clocks.
Default after reset	Disabled
Programming	Must be programmed by software when enabling and disabling a display using MIPI DSI.

X_e Port Clock Programming

PLL and Clock Usage

This is for mapping the DPCLKA_CFGCR0 bit fields to the PLLs and port names used in display engine.

Note: SKUs with only 2 combo PHY ports should only use DPLL0 and DPLL1.

DPLL0 = PLL available for display combo PHY usage

DPLL1 = PLL available for display combo PHY usage

DPLL2 = TBTPLL, the PLL for thunderbolt

DPLL3 = Not available to display

DPLL4 = PLL available for display combo PHY usage (not available on Y/U SKUs)

DDIA Clock = DDIA and DSIO (combo port A) symbol clock

DDIB Clock = DDIB and DSII (combo port B) symbol clock

DDIC Clock = DDIC (combo port C) symbol clock

TC1 Clock = DDI USBC1 (type C port 1) symbol clock

TC2 Clock = DDI USBC2 (type C port 2) symbol clock

TC3 Clock = DDI USBC3 (type C port 3) symbol clock

TC4 Clock = DDI USBC4 (type C port 4) symbol clock

TC5 Clock = DDI USBC5 (type C port 5) symbol clock

TC6 Clock = DDI USBC6 (type C port 6) symbol clock

TypeC PLL = A PLL in the typeC port, more info in the TypeC PLL Programming section

PLL Frequency Changes

PLL frequency should not be changed while the PLL is enabled.

1. Follow PLL Disable Sequence
2. Follow PLL Enable Sequence using the new frequency

DisplayPort Mode Combo PHY Programming

This programming is for DPLLs when using eDP or DP on combo PHY.

DisplayPort Mode PLL Enable Sequence

1. Enable DPLL power in DPLL_ENABLE.
2. Wait for DPLL power state enabled in DPLL_ENABLE.
 - Should complete immediately.
3. Configure DPLL_SSC to set SSC enable/disable.
4. Configure DPLL_CFGCR0 to set DCO frequency.
5. Configure DPLL_CFGCR1 to set the dividers.
6. Read back DPLL_CFGCR0 or DPLL_CFGCR1, ignoring the data value, this is to ensure writes completed before the next step.
7. If the frequency will result in a change to the voltage requirement, follow the Display Voltage Frequency Switching - Sequence Before Frequency Change.
8. Enable DPLL in DPLL_ENABLE.
9. Wait for PLL lock status in DPLL_ENABLE.
 - Should complete within 50us.
10. If the frequency will result in a change to the voltage requirement, follow the Display Voltage Frequency Switching - Sequence After Frequency Change.
11. For each DDI that will use this DPLL
 - a. Configure DPCLKA_CFGCR0 to map the DPLL to the DDI.



- b. Configure DPCLKA_CFGCR0 to turn on the clock for the DDI. This step and the step before must be done with separate register writes.

DisplayPort Mode PLL Disable Sequence

1. For each DDI that was using this DPLL, configure DPCLKA_CFGCR0 to turn off the clock for the DDI.
2. If the frequency will result in a change to the voltage requirement, follow the Display Voltage Frequency Switching - Sequence Before Frequency Change.
3. Disable DPLL through DPLL_ENABLE.
4. Wait for PLL not locked status in DPLL_ENABLE.
 - Should complete within 1us.
5. If the frequency will result in a change to the voltage requirement, follow the Display Voltage Frequency Switching - Sequence After Frequency Change.
6. Disable DPLL power in DPLL_ENABLE.
7. Wait for DPLL power state disabled in DPLL_ENABLE.
 - Should complete immediately.

DisplayPort Mode PLL Values

Link Bit Rate GHz	Reference Clock MHz	DCO Integer	DCO Fraction	Pdiv	Kdiv	Qdiv Mode	Qdiv Ratio
5.4	24	151h	4000h	0010b (3)	001b (1)	0b	00h (1)
2.7	24	151h	4000h	0010b (3)	010b (2)	0b	00h (1)
1.62	24	151h	4000h	0100b (5)	010b (2)	0b	00h (1)
3.24	24	151h	4000h	0100b (5)	001b (1)	0b	00h (1)
2.16	24	168h	0000h	0001b (2)	010b (2)	1b	02h (2)
4.32	24	168h	0000h	0001b (2)	010b (2)	0b	00h (1)
6.48	24	195h	0000h	0010b (3)	001b (1)	0b	00h (1)
8.1	24	151h	4000h	0001b (2)	001b (1)	0b	00h (1)
5.4	19.2 or 38.4	1A5h	7000h	0010b (3)	001b (1)	0b	00h (1)
2.7	19.2 or 38.4	1A5h	7000h	0010b (3)	010b (2)	0b	00h (1)
1.62	19.2 or 38.4	1A5h	7000h	0100b	010b	0b	00h (1)

Link Bit Rate GHz	Reference Clock MHz	DCO Integer	DCO Fraction	Pdiv	Kdiv	Qdiv Mode	Qdiv Ratio
				(5)	(2)		
3.24	19.2 or 38.4	1A5h	7000h	0100b (5)	001b (1)	0b	00h (1)
2.16	19.2 or 38.4	1C2h	0000h	0001b (2)	010b (2)	1b	02h (2)
4.32	19.2 or 38.4	1C2h	0000h	0001b (2)	010b (2)	0b	00h (1)
6.48	19.2 or 38.4	1FAh	2000h	0010b (3)	001b (1)	0b	00h (1)
8.1	19.2 or 38.4	1A5h	7000h	0001b (2)	001b (1)	0b	00h (1)

Example of DisplayPort on DDIA using HBR 2.7 GHz link rate with SSC and reference 24 MHz

This example assumes DPLL0 is available.

1. Enable DPLL0 power in the DPLL0_ENABLE register
 - Power Enable = 1b (Enable)
2. Wait for DPLL0 power state enabled in DPLL0_ENABLE.
3. Configure DPLL0_SSC register
 - sscen = 1b (Enable)
4. Configure DPLL0 in the DPLL0_CFGCR0 register
 - DCO Fraction = 4000h
 - DCO Integer = 151h
5. Configure DPLL0 in the DPLL0_CFGCR1 register
 - Qdiv Ratio = 00h (1)
 - Qdiv Mode = 0b (Disable)
 - Kdiv = 010b (2)
 - Pdiv = 0010b (3)
6. Read back DPLL0_CFGCR0 or DPLL0_CFGCR1 to ensure writes completed
7. Display Voltage Frequency Switching - Sequence Before Frequency Change
 - Skipped if 270 MHz is low enough to not trigger an increase in the voltage requirement
8. Enable DPLL0 in the DPLL0_ENABLE register
 - PLL Enable = 1b (Enable)
9. Wait for PLL lock status in DPLL0_ENABLE
10. Display Voltage Frequency Switching - Sequence After Frequency Change
 - Skipped if 270 MHz is low enough to not trigger an increase in the voltage requirement



11. Configure DPLL0 mapping to DDIA in the DPCLKA_CFGCR0 register
 - DDIA Clock Select = 00b (DPLL0)
12. Turn on DDIA clock in the DPCLKA_CFGCR0 register
 - DDIA Clock Off = 0b (On)

HDMI Mode Combo PHY Programming

This programming is for DPLLs when using HDMI or DVI on combo PHY.

HDMI Mode PLL Enable Sequence

Sequence is the same as for DisplayPort, except SSC must be disabled.

HDMI Mode PLL Disable Sequence

Sequence is the same as for DisplayPort

Formula for HDMI Mode DPLL Programming

Reference frequency = 38.4, 24, or 19.2 MHz (Register DSSM Reference Frequency indicates the frequency). If reference frequency is 38.4, use 19.2 because the DPLL automatically divides that by 2.

Symbol clock frequency MHz = $\text{DCO Frequency} / (\text{Pdiv} * \text{Qdiv} * \text{Kdiv}) / 5$.

AFE clock = PLL output frequency = Symbol clock frequency MHz * 5 = $\text{DCO Frequency} / (\text{Pdiv} * \text{Qdiv} * \text{Kdiv})$.

Pdiv can be 2, 3, 5, or 7.

Kdiv can be 1, 2, or 3.

If Kdiv != 2, then Qdiv must be 1. Else Qdiv can be 1 to 255.

Minimum DCO frequency = 7,998 MHz

Maximum DCO frequency = 10,000 MHz

Midpoint DCO frequency = 8,999 MHz

Note: If the desired symbol clock frequency cannot be achieved with the valid values of P, Q, K, and DCO frequencies, use a different screen resolution with a different symbol clock frequency.

Note: HDMI symbol clock frequency can be different from the pixel rate.

HDMI Color Format	Bits per color	HDMI Symbol Clock	HDMI Pixel Rate
RGB or YUV444	8	1 * pixel rate	1 * symbol clock
RGB or YUV444	10	5/4 * pixel rate	4/5 * symbol clock
RGB or YUV444	12	3/2 * pixel rate	2/3 * symbol clock
YUV420	8	1/2 * pixel rate	2 * symbol clock
YUV420	10	5/8 * pixel rate	8/5 * symbol clock
YUV420	12	3/4 * pixel rate	4/3 * symbol clock

Algorithm to Find HDMI Mode DPLL Programming

1. Calculate AFE clock
2. For each legal divider ($P*Q*K$), find the DCO.
3. Select the divider that gives the DCO closest to the midpoint and fits within the minimum or maximum.
4. Find the P, Q, K values to create that divider.

Pseudo-code for HDMI Mode DPLL Programming

```

$freq = Desired symbol clock frequency in MHz

$div=0; $bestdiv=0; $dco=0; $bestdco=0; $dcocentrality=0; $Pdiv = 0; $Qdiv = 0; $Kdiv = 0;
$afeclk = 0;
$bestdcocentrality = 999999;
@dividerlist = (2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 24, 28, 30, 32, 36, 40, 42, 44, 48, 50,
52, 54, 56, 60, 64, 66, 68, 70, 72, 76, 78, 80, 84, 88, 90, 92, 96, 98, 100, 102, 3, 5, 7, 9,
15, 21);
$dcomin = 7998;
$dcomax = 10000;
$dcomid = ($dcomin + $dcomax) / 2;

$afeclk = 5 * $freq;

foreach $div (@dividerlist) {
    $dco = $afeclk * $div;
    if (($dco <= $dcomax) & ($dco >= $dcomin))
    {
        $dcocentrality = abs($dco-$dcomid);
        if ($dcocentrality < $bestdcocentrality)
        {
            $bestdcocentrality = $dcocentrality;
            $bestdiv = $div;
            $bestdco = $dco
        }
    }
}

if ($bestdiv != 0) # Good divider found
{
    if ($bestdiv % 2 == 0) # Even
    {
        if ($bestdiv == 2)
        {
            $Pdiv = 2;

```

```

        $Qdiv = 1;
        $Kdiv = 1;
    }
    elsif ($bestdiv % 4 == 0)
    {
        $Pdiv = 2;
        $Qdiv = $bestdiv / 4;
        $Kdiv = 2;
    }
    elsif ($bestdiv % 6 == 0)
    {
        $Pdiv = 3;
        $Qdiv = $bestdiv / 6;
        $Kdiv = 2;
    }
    elsif ($bestdiv % 5 == 0)
    {
        $Pdiv = 5;
        $Qdiv = $bestdiv / 10;
        $Kdiv = 2;
    }
    elsif ($bestdiv % 14 == 0)
    {
        $Pdiv = 7;
        $Qdiv = $bestdiv / 14;
        $Kdiv = 2;
    }
    else # odd
    {
        if ($bestdiv == 3 || $bestdiv == 5 || $bestdiv == 7)
        {
            $Pdiv = $bestdiv;
            $Qdiv = 1;
            $Kdiv = 1;
        }
        else # 9, 15, 21
        {
            $Pdiv = $bestdiv / 3;
            $Qdiv = 1;
            $Kdiv = 3;
        }
    }
    SUCCESS
    Program DPLL_CFGCR0 with Best DCO / Reference Frequency
    Program DPLL_CFGCR1 with Pdiv, Qdiv, and Kdiv
}
else # No good divider found
{
    FAIL, try a different frequency
}

```

Calculating Frequency from Divider Values

Symbol clock frequency in MHz = DCO divider * Reference Frequency in MHz / (5 * Pdiv * Qdiv * Kdiv)

DCO divider from DPLL_CFGCR0 DCO Integer + (DPLL_CFGCR0 DCO Fraction / 2¹⁵)

Pdiv from DPLL_CFGCR1 Pdiv

Qdiv from DPLL_CFGCR1 Qdiv Mode ? DPLL_CFGCR1 Qdiv Ratio : 1

Kdiv from DPLL_CFGCR1 Kdiv

Example of DVI on DDIB using 113.309 MHz symbol clock and reference 24 MHz

This example assumes DPLL1 is available.

Frequency programming algorithm finds Pdiv=2, Qdiv=4, Kdiv=2, DCO Frequency = 9064.72

DCO Integer = $\text{INT}(9064.72/24) = 377$

DCO Fraction = $\text{INT}(((9064.72/24) - \text{DCO Integer}) * 2^{15}) = 22,828$

1. Enable DPLL1 power in the DPLL1_ENABLE register
 - Power Enable = 1b (Enable)
2. Wait for PLL power state enabled in DPLL1_ENABLE.
3. Configure DPLL1_SSC register
 - sscen = 0b (Disable)
4. Configure DPLL1 in the DPLL1_CFGCR0 register
 - DCO Fraction = 592Ch (22828)
 - DCO Integer = 179h (377)
5. Configure DPLL1 in the DPLL1_CFGCR1 register
 - Qdiv Ratio = 04h (4)
 - Qdiv Mode = 1b (Enable)
 - Kdiv = 010b (2)
 - Pdiv = 0001b (2)
6. Read back DPLL1_CFGCR0 or DPLL1_CFGCR1 to ensure writes completed
7. Display Voltage Frequency Switching - Sequence Before Frequency Change
 - Skipped if 113.309 MHz is low enough to not trigger an increase in the voltage requirement
8. Enable DPLL1 in the DPLL1_ENABLE register
 - PLL Enable = 1b (Enable)
9. Wait for PLL lock in DPLL1_ENABLE
10. Display Voltage Frequency Switching - Sequence After Frequency Change
 - Skipped if 113.309 MHz is low enough to not trigger an increase in the voltage requirement
11. Configure DPLL1 mapping to DDIB in the DPCLKA_CFGCR0 register
 - DDIB Clock Select = 01b (DPLL1)
12. Turn on DDIB clock in the DPCLKA_CFGCR0 register
 - DDIB Clock Off = 0b (On)

MIPI DSI DPLL Programming

This programming is for DPLLs when using MIPI DSI.

Note that if the Driver wishes to maintain a single enabling sequence between DP/eDP and MIPI DSI, then the same enabling sequence defined for DP/eDP on combo PHY can be used for MIPI DSI. The only difference between the two sequences is:

1. The order of the DPLL mapping (DPCLKA_CFGCR0) with respect to the DPLL_ENABLE. The order of these two steps is unimportant
2. The programming of the *_ESC_CLK_DIV registers. The placement of the *_ESC_CLK_DIV registers can be placed anywhere within the sequence (i.e. at the beginning or end of the sequence) to facilitate a single function.

MIPI DSI DPLL Enable Sequence

1. Enable DPLL power in DPLL_ENABLE.
2. Wait for DPLL power state enabled in DPLL_ENABLE.
 - Should complete immediately.
3. Configure DPLL_SSC to set SSC enable/disable.
4. Configure DPLL_CFGCR0 to set DCO frequency.
5. Configure DPLL_CFGCR1 to set the dividers.
6. Read back DPLL_CFGCR0 or DPLL_CFGCR1, ignoring the data value, this is to ensure writes completed before the next step.
7. Configure DPCLKA_CFGCR0 to map the DPLL to the DDI/DSI.
 - Do not turn on the clock for the DDI. DSI has independent clock gating controlled by MIPI DSI mode programming.
8. Read back DPCLKA_CFGCR0, ignoring the data value, this is to ensure write completed before the next step.
9. Configure both **DSI_ESC_CLK_DIV** and **DPHY_ESC_CLK_DIV** registers. Both registers must be programmed with the same value.
- 10.
11. Read back both DPHY_ESC_CLK_DIV, ignoring the data value, this is to ensure write completed before the next step.
12. Enable DPLL in DPLL_ENABLE.
13. Wait for PLL lock status in DPLL_ENABLE.
 - Should complete within 50us.

MIPI DSI DPLL Disable Sequence

Sequence is the same as for DisplayPort, except there is not need to configure DPCLKA_CFGCR0 to turn off the clock for the DDI.

Formula for MIPI DSI DPLL Programming

Follow the same formula as for HDMI Mode. The AFE clock is the MIPI 8X clock.

Formula for MIPI DSI Escape Clock Programming

The MIPI DSI port operates with a constant frequency clock referred to as the Escape clock. The Escape clock should be programmed to be as close to, but not greater than 20 MHz. The Escape clock frequency will be determined by dividing the 8X frequency by a variable "M".

$$f_{esc} = f_{8x} / M \leq 20 \text{ MHz}$$

Solving for M supplies the programming necessary for the DSI_ESC_CLK_DIV and DPHY_ESC_CLK_DIV registers.

$$\text{Escape clock divider } M = \text{ceiling}(f_{8x} \text{ (MHz)} / 20 \text{ MHz})$$

Example for DSI0 8X 566.545 and reference 24 MHz

This example assumes DPLL0 is available.

8x clock = AFE Clock = 566.545 MHz

Frequency programming algorithm finds Pdiv=3, Qdiv=1, Kdiv=1, DCO Frequency = 8498.175

Escape clock divider M = ceiling(566.545 / 20 MHz) = 29

DCO Integer = INT(8498.175/24) = 354

DCO Fraction = INT(((8498.175/24) - DCO Integer) * 2¹⁵) = 2969

1. Enable DPLL0 power in the DPLL0_ENABLE register
 - Power Enable = 1b (Enable)
2. Wait for PLL power state enabled in DPLL0_ENABLE.
3. Configure DPLL0_SSC register
 - sscen = 0b (Disable)
4. Configure DPLL0 in the DPLL0_CFGCR0 register
 - DCO Fraction = 0B99h (2969)
 - DCO Integer = 162h (354)
5. Configure DPLL0 in the DPLL0_CFGCR1 register
 - Qdiv Ratio = 00h (1)
 - Qdiv Mode = 0b (Disable)
 - Kdiv = 001b (1)
 - Pdiv = 0010b (3)
6. Read back DPLL0_CFGCR0 or DPLL0_CFGCR1 to ensure writes completed
7. Configure DPCLKA_CFGCR0 DPLL0 mapping to DSI0 in the DPCLKA_CFGCR0 register
 - DDIA (DSI0) Clock Select = 00b (DPLL0)
8. Turn on DSI0 clock in the DPCLKA_CFGCR0 register
 - DDIA (DSI0) Clock Off = 0b (On)
9. Read back DPCLKA_CFGCR0 to ensure write completed

10. Configure MIPI escape clock dividers in DSI_ESC_CLK_DIV_0 and DPHY_ESC_CLK_DIV_0
 - Escape Clock Divider M = 0x01D (Divide by 29)
11. Read back DPHY_ESC_CLK_DIV to ensure write completed
12. Enable DPLL0 in the DPLL0_ENABLE register
 - PLL Enable = 1b (Enable)
13. Wait for PLL lock in DPLL0_ENABLE

Thunderbolt PLL Programming

This programming is for ports going to Thunderbolt. Those ports use the TBT PLL (a version of DPLL) shared across all thunderbolt ports. Thunderbolt supports a few fixed frequencies which are all produced simultaneously from parallel dividers after the PLL.

Thunderbolt PLL Enable Sequence

1. Enable PLL power in TBT_PLL_ENABLE (an instance of the DPLL_ENABLE register).
2. Wait for PLL power state enabled in TBT_PLL_ENABLE.
 - Should complete immediately.
3. Configure TBTPLL_CFGCR0 to set DCO frequency.
4. Read back TBTPLL_CFGCR0, ignoring the data value, this is to ensure writes completed before the next step.
5. Enable PLL in TBT_PLL_ENABLE.
6. Wait for PLL lock status in TBT_PLL_ENABLE.
 - Should complete within 50us.
7. If the frequency will result in a change to the voltage requirement, follow the Display Voltage Frequency Switching - Sequence Before Frequency Change.
 - Since the PLL produces multiple frequencies simultaneously, use the frequency selected for the port when determining the voltage requirement.
8. For each port that will use this PLL, program DDI_CLK_SEL to map the TBT clock to the port.
9. For each port that will use this PLL, configure DPCLKA_CFGCR0 to turn on the clock for the port.
10. If the frequency will result in a change to the voltage requirement, follow the Display Voltage Frequency Switching - Sequence After Frequency Change.
 - Since the PLL produces multiple frequencies simultaneously, use the frequency selected for the port when determining the voltage requirement.

Thunderbolt PLL Disable Sequence

- Since multiple ports can share the same PLL, the PLL is not disabled until after all the ports using it have turned off the clock.
1. If the frequency will result in a change to the voltage requirement, follow the Display Voltage Frequency Switching - Sequence Before Frequency Change.

2. For each port that was using this PLL, configure DPCLKA_CFGCR0 to turn off the clock for the port.
3. If the frequency will result in a change to the voltage requirement, follow the Display Voltage Frequency Switching - Sequence After Frequency Change.
4. Disable PLL through TBT_PLL_ENABLE.
5. Wait for PLL not locked status in TBT_PLL_ENABLE.
 - Should complete within 1us.
6. Disable DPLL power in TBT_PLL_ENABLE.
7. Wait for DPLL power state disabled in TBT_PLL_ENABLE.
 - Should complete immediately.

Thunderbolt PLL Divider Values

The thunderbolt PLL must always be configured to give a 1620 MHz output, depending on reference clock, using the values below.

Reference Clock	Pdiv	Kdiv	Qdiv	DCO MHz	DCO divided by Reference Clock MHz	DCO Integer hex	DCO Fraction hex
19.2 or 38.4 MHz	Not used	Not used	Not used	1620	84.375	0x54	0x3000
24 MHz	Not used	Not used	Not used	1620	67.5	0x43	0x4000

Thunderbolt Port Frequency Selection

Hardcoded dividers on the PLL 1620 MHz output produce 162, 270, 540, and 810 MHz symbol clock frequencies. The DDI_CLK_SEL register allows each port to select between those 4 frequencies.

Example of Thunderbolt on TC1 using 270 MHz symbol clock

This example assumes the reference frequency is 24 MHz.

1. Enable TBT_PLL power in the TBT_PLL_ENABLE register
 - Power Enable = 1b (Enable)
2. Wait for PLL power state enabled in TBT_PLL_ENABLE.
3. Configure TBTPLL_CFGCR0 register
 - DCO Fraction = 4000h
 - DCO Integer = 43h
4. Read back TBTPLL_CFGCR0 to ensure writes completed
5. Enable PLL in the TBT_PLL_ENABLE register
 - PLL Enable = 1b (Enable)
6. Wait for PLL lock in TBT_PLL_ENABLE
7. Display Voltage Frequency Switching - Sequence Before Frequency Change
 - Skipped if 270 MHz is low enough to not trigger an increase in the voltage requirement
8. Configure TC1 to use TBT 270 MHz in DDI_CLK_SEL_C register



- Clock Select = 1101b (TBT 270)
9. Configure DPCLKA_CFGCR0 to turn on TC1 clock
 - TC1 Clock Off = 0b (On)
 10. Display Voltage Frequency Switching - Sequence After Frequency Change
 - Skipped if 270 MHz is low enough to not trigger an increase in the voltage requirement

TypeC PHY PLL Programming for Displayport

NOTE: Display uses PLL1, not PLL0. The defaults of associated registers may be different.

TypeC PHY and PLL registers need to be accessed as per the instructions in Dekel PHY Access page.

Register programming for DP frequencies consists of 2 parts: rate independent programming and rate dependent programming.

DKL PLL Enable Sequence

1. Enable PLL power in MGPLL_ENABLE (an instance of the DPLL_ENABLE register).
2. Wait for PLL power state enabled in MGPLL_ENABLE.
 - Should complete within 10us.
3. Program PLL registers as given in tables below - rate independent first and then rate dependent.
4. Read back the last programmed PHY PLL register, ignoring the data value, to ensure writes completed before the next step.
5. If the frequency will result in a change to the voltage requirement, follow the Display Voltage Frequency Switching - Sequence Before Frequency Change.
6. Enable PLL in MGPLL_ENABLE.
7. Wait for PLL lock status in MGPLL_ENABLE.
 - Timeout and fail after **900us**.
8. If the frequency will result in a change to the voltage requirement, follow the Display Voltage Frequency Switching - Sequence After Frequency Change.
9. Program DDI_CLK_SEL to map the Type-C PLL clock to the port.
10. Configure DPCLKA_CFGCR0 to turn on the clock for the port.

DKL PLL Disable Sequence

1. Configure DPCLKA_CFGCR0 to turn off the clock for the port.
2. If the frequency will result in a change to the voltage requirement, follow the Display Voltage Frequency Switching - Sequence Before Frequency Change.
3. Disable PLL through MGPLL_ENABLE.
 - Should complete within 50us.
4. Wait for PLL not locked status in MGPLL_ENABLE.

5. If the frequency will result in a change to the voltage requirement, follow the Display Voltage Frequency Switching - Sequence After Frequency Change.
6. Disable PLL power in MGPLL_ENABLE.
7. Wait for PLL power state disabled in MGPLL_ENABLE.
 - Should complete immediately.

The following DP rate independent and rate dependent programming must check DFLEXDPSP[12] for IOM FW support.

MFD DP - Rate independent programming - 100 MHz reference clock

This table covers programming for cases where DisplayPort is sharing lanes with USB.

PHY Register Address	Controller Register Address	Block	Register Name	Register Value	
0x2200	Check DKL PHY register access page	PLL1	DKL_PLL_DIV0	0x50272228	0x50272228
0x2204	Check DKL PHY register access page	PLL1	DKL_PLL_DIV1	0x0CD8C40D	0x0CD8C40D
0x2208	Check DKL PHY register access page	PLL1	DKL_PLL_LF	0x00401300	0x00401300
0x220C	Check DKL PHY register access page	PLL1	DKL_PLL_FRAC_LOCK	0x8044B56A	0x8044B56A
0x2210	Check DKL PHY register access page	PLL1	DKL_SSC ⁽²⁾	0x800020FF	0x800020FF
0x2214	Check DKL PHY register access page	PLL1	DKL_BIAS	0xE0000000	0xE0000000
0x2218	Check DKL PHY register access page	PLL1	DKL_TDC_COLDST_BIAS	0x00000023	0x00000023
0x2228	Check DKL PHY register access page	PLL1	DKL_XXX_TDC_CRO	0x80008150	0x80008150
0x2244	Check DKL PHY register access page	PLL1	DKL_PLL1_CNTR_XXXX_SETTINGS	0x0004003F	0x0004003F
0x203C	Check DKL PHY register access page	CMN_DIG	DKL_CMN_DIG_PLL_MISC	0x00020000	0x00020000
0x02D4/0x12D4	Check DKL PHY register access page	PCS_Glue_Lane0/1	RTT_CR_SPARE	Do not program ⁽¹⁾	Do not program ⁽¹⁾

⁽¹⁾default applies

⁽²⁾SSC can be disabled as per requirement.



Native and ALT DP - Rate independent programming - 38.4 MHz reference clock

This table covers programming for cases where DisplayPort is not sharing lanes with USB (native and ALT modes).

PHY Register Address	Controller Register Address	Block	Register Name	Register Value		
0x2200	Check DKL PHY register access page	PLL1	DKL_PLL_DIV0	0x70272269	0x70272269	0x70272269
0x2204	Check DKL PHY register access page	PLL1	DKL_PLL_DIV1	0x0CDCC422	0x0CDCC422	0x0CDCC422
0x2208	Check DKL PHY register access page	PLL1	DKL_PLL_LF	0x00401300	0x00401300	0x00401300
0x220C	Check DKL PHY register access page	PLL1	DKL_PLL_FRAC_LOCK	0x8044B56A	0x8044B56A	0x8044B56A
0x2210	Check DKL PHY register access page	PLL1	DKL_SSC ⁽²⁾	0x401322FF	0x401322FF	0x401322FF
0x2214	Check DKL PHY register access page	PLL1	DKL_BIAS	Do not program ⁽¹⁾	Do not program ⁽¹⁾	Do not program ⁽¹⁾
0x2218	Check DKL PHY register access page	PLL1	DKL_TDC_COLDST_BIAS	Do not program ⁽¹⁾	Do not program ⁽¹⁾	Do not program ⁽¹⁾
0x2228	Check DKL PHY register access page	PLL1	DKL_XXX_TDC_CRO	Do not program ⁽¹⁾	Do not program ⁽¹⁾	Do not program ⁽¹⁾
0x2244	Check DKL PHY register access page	PLL1	DKL_PLL1_CNTR_XXXX_SETTINGS	Do not program ⁽¹⁾	Do not program ⁽¹⁾	Do not program ⁽¹⁾
0x203C	Check DKL PHY register access page	CMN_DIG	DKL_CMN_DIG_PLL_MISC	0x00000000	0x00000000	0x00000000
0x212C	Check DKL	CMN_ANA	DKL_REFCLKIN_CTL	0x00000101	0x00000303	0x00000101

PHY Register Address	Controller Register Address	Block	Register Name	Register Value		
	PHY register access page					
0x2130	Check DKL PHY register access page	CMN_ANA	DKL_CMN_ANA_DWORD28	0x14158888	0x14158888	0x14158888
0x02D4/0x12D4	Check DKL PHY register access page	PCS_Glue_Lane0/1	RTT_CR_SPARE	Do not program ⁽¹⁾	Do not program ⁽¹⁾	Do not program ⁽¹⁾

⁽¹⁾default applies

⁽²⁾SSC can be disabled as per requirement.

MFD/Native DP - Rate dependent programming

Register Name	8p1	5p4	2p7	1p62
CLKTOP2_HSCLKCTL	0000011D (default)	0x0000121D	0x0000521D	0x0000621D
CLKTOP2_CORECLKCTL1	10080510 (default)	10080510 (default)	0x10080A12	0x10080A12

TypeC PHY PLL Programming - algorithmic steps

NOTE: Display uses PLL1, not PLL0. The defaults of associated registers may be different.

If the desired symbol clock frequency cannot be achieved with the valid programming, use a different screen resolution with a different symbol clock frequency.

The following algorithmic steps apply for HDMI protocol and enumerated reference clocks. For DP protocol, refer to the tables above.

```

$symbol_frequency = Desired HDMI symbol clock frequency in MHz (example: 594 MHz for HDMI 2.0
4K 60 Hz)
$refclk_mhz = Reference frequency (19.2 MHz or 24.0 MHz or 38.4 MHz)
$ssc_en = SSC enable // DO NOT SELECT SSC $dp = 1 for DisplayPort, 0 for HDMI // DO NOT
SELECT DISPLAYPORT OPTION Set $dp = 0
$dp_dco = 8.1;
$dco_min_freq = $dp ? $dp_dco : $ssc_en ? 8 : 7.992;
$dco_max_freq = $dp ? $dp_dco : 10;
@div1_vals = ( 7, 5, 3, 2);
@div2_vals = ( 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 );
$success = 0;

$frequency = $symbol_frequency/100;

for $div1 (@div1_vals) {
  for $div2 (@div2_vals) {
    $dco = $div1 * $div2 * $frequency / 2;
    if ( $dco >= $dco_min_freq and $dco <= $dco_max_freq ) {
      Program DKL_REFCLKIN_CTL od_refclkin2_refclkmux_2_0 = 1
      Program DKL_CLKTOP2_HSCLKCTL od_clktop2_coreclk_inputsel = 1
    }
  }
}

```

```

        Program DKL_CLKTOP2_CORECLKCTL1 od_clktop2_coreclka_divratio = 5

        if ($div2 >= 2) {
            Program DKL_CLKTOP2_HSCLKCTL od_clktop_tlinedrv_clkssel_1_0 = 1
        } else {
            Program DKL_CLKTOP2_HSCLKCTL od_clktop_tlinedrv_clkssel_1_0 = 0
        }
        Program DKL_CLKTOP2_HSCLKCTL od_clktop_coreclk_inputsel = $dp ? 0b : 1b
        if ($div1 == 2) {Program DKL_CLKTOP2_HSCLKCTL od_clktop_hsdiv_divratio_1_0 = 0}
        if ($div1 == 3) {Program DKL_CLKTOP2_HSCLKCTL od_clktop_hsdiv_divratio_1_0 = 1}
        if ($div1 == 5) {Program DKL_CLKTOP2_HSCLKCTL od_clktop_hsdiv_divratio_1_0 = 2}
        if ($div1 == 7) {Program DKL_CLKTOP2_HSCLKCTL od_clktop_hsdiv_divratio_1_0 = 3}
        Program DKL_CLKTOP2_HSCLKCTL od_clktop_dsdiv_divratio = $div2 converted to 4 bit
binary;
        $target_dco_mhz = $dco * 1000;
        $success = 1;
        last;
    }
}
if ($success == 1) {last;}
}

if ($success == 0) {
    FAIL and try a different frequency
}

$mldiv = 2;           # pre-divider ratio
$ndiv = 1;           # reference clock divider
$tdc_res = 0.000003; # TDC resolution
$ssc_freq_mhz = .032; # SSC frequency
$ssc_stepnum = 32;   # SSC step num
$ssc_amp = .0047;    # SSC amplitude

# calculate the M2 ratio
$m2div = ($target_dco_mhz / ($refclk_mhz / $ndiv)) / $mldiv;

# 8 bits for M2
if ($m2div > 255) {
    FAIL and try a different frequency
}

# integer portion
$m2div_int = floor($m2div);

# fractional portion
$m2div_frac = floor(2**22 * ($m2div - $m2div_int));

# iref ndiv
$iref_ndiv = ($refclk_mhz > 80) ? 4 : ($refclk_mhz > 38) ? 2 : 1;

# TDC target count
$tdc_targetcnt = int(2/($tdc_res*8*50*1.1)/$refclk_mhz + 0.5);

# feed forward gain
$feedfwdgain = ($ssc_en || $m2div_frac > 0) ? floor($mldiv*(1/$target_dco_mhz)/$tdc_res) : 0;

# iref trim value
$iref_trim = (($refclk_mhz/$iref_ndiv) <= 19.2) ? 28 :
              (($refclk_mhz/$iref_ndiv) > 19.2 && ($refclk_mhz/$iref_ndiv) <= 25) ? 25 :
24;

# bias start-up pulse width

```

```

$iref_pulse_width = (($refclk_mhz/$iref_ndiv) > 20) ? 2 : 1;

# lf coeffs
$prop_coeff = ($target_dco_mhz >= 9000) ? 5 : 4;
$int_coeff = ($target_dco_mhz >= 9000) ? 10 : 8;

# ssc params
$ssc_stepsize = $ssc_en ? (floor($m2div * $ssc_amp / $ssc_stepnum * 2**10)) : 0;
$ssc_steplen = $ssc_en ? (ceil($refclk_mhz / ($ssc_freq_mhz / (2 * $ssc_stepnum))) : 0;
$ssc_steplog = log($ssc_stepnum)/log(2)-1;

# calculate register values (these are read-modify-write ops)
$div0 = ($int_coeff * 2**16) + ($prop_coeff * 2**12) + ($m1div * 2**8) + ($m2div_int);
$div1 = ($iref_trim * 2**16) + ($tdc_targetcnt);
$ssc = ($iref_ndiv * 2**29) + ($ssc_steplen * 2**16) + ($ssc_stepnum * 2**11) + ($ssc_en *
2**9);
$bias = ($m2div_frac > 0 ? 2**30 : 0) + ($m2div_frac * 2**8);
$tdc = ($ssc_stepsize * 2**8) + $feedfwdgain;

Program DKL_PLL_DIV0 = $div0 converted to 32 bit hex;
Program DKL_PLL_DIV1 = $div1 converted to 32 bit hex;
Program DKL_SSC = $ssc converted to 32 bit hex;
Program DKL_BIAS = $bias converted to 32 bit hex;
Program DKL_TDC_COLDST_BIAS = $tdc converted to 32 bit hex;

```

Calculating PLL Frequency from Divider Values

Symbol clock frequency in MHz = M1 * M2 * Reference Frequency in MHz / (5 * DIV1 * DIV2)

M1 from DKL_PLL_DIV0[i_fbprediv_3_0]

M2 from DKL_PLL_DIV0[i_fbdiv_intgr_7_0] + DKL_BIAS[i_fracnen_h] ? DKL_BIAS[i_fbdivfrac_21_0/2^22) : 0

DIV1 from DKL_CLKTOP2_HSCLKCTL[od_clktop_hsdiv_divratio]

DIV2 from DKL_CLKTOP2_HSCLKCTL[od_clktop_dsdiv_divratio]

HDMI Example

HDMI, 5.94 Gbps, SSC disabled, 38.4 MHz reference

Register	Default	Value to program
DKL_BIAS	0xde000000	0xc1000000
DKL_PLL_DIV0	0x50273269	0x50284274
DKL_PLL_DIV1	0x0cdcc522	0x0cdcc527
DKL_SSC	0x401322ff	0x400020ff
DKL_TDC_COLDST_BIAS	0x00000f23	0x0000004a
TAP_OVRD	0x50003100	0x50003100
DFX_TDC_CRO	0x80008130	0x80008130
CLKTOP2_HSCLKCTL	0x0000011d	0x0001111d
CLKTOP2_CORECLKCTL1	0x10080510	0x10080510



Display Voltage Frequency Switching

Display Voltage and Frequency Switching (DVFS) is used to adjust the display voltage to match the display clock frequencies. If voltage is set too low, it will break functionality. If voltage is set too high, it will waste power. The voltage rail for display is shared by the entire system agent, so it has a large power impact.

When changing clock frequencies, graphics software must inform the power controller of the display voltage requirement. The power controller tracks requests from all users of the voltage rail and sets voltage to accommodate all the requests. The voltage requirements are separated into discrete levels aligned to the frequencies of several clocks used by display.

Voltage Requirement Selection			
<p>The voltage requirement is specified by selecting from several discrete voltage levels. The power controller will map these levels to the actual voltage values, which are usually determined on a part by part basis during manufacturing.</p> <p>Only the CD clock and DDI clocks (max of all DDI clocks) are used to select the voltage levels. Other display clocks are either supported at all frequencies with the minimum voltage, or have their voltage requirements accounted for by non-graphics software.</p>			
Voltage Level	CD clock MHz	Max DDI symbol clock MHz	Comment
0	307.2, 312, or lower	≤ 594	No 5k or 8k
1	324 or 326.4	≤ 594	These display clocks are mainly for resolutions under 5k with multiple displays enabled (cdclk PLL divided by 2 to allow for smooth increase to max cdclk)
2	556.8 or 552	> 594	These display clocks are mainly for non-HDR 5K and 8k, or some cases with DP multistream or USB TypeC
3	652.8 or 648	> 594	These display clocks are mainly for HDR 5k and 8k
<p>If CD clock ≤ 312 MHz AND Max of DDI clocks ≤ 594 MHz, use level 0. Else If CD clock ≤ 326.4 MHz AND Max of DDI clocks ≤ 594 MHz, use level 1. Else If CD clock ≤ 556.8 AND Max of DDI clocks > 594 MHz, use level 2. Else, use level 3.</p> <p>Note: For these calculations, disabling a clock is the same as switching it to the lowest frequency.</p> <p>The sequences below are used when changing CD clock frequency and when changing DDI clock frequency during a mode set.</p>			

The following sequences are not used on their own. They are called as part of other sequences which change the display clock frequencies.

Sequence Before Frequency Change

- This sequence requests the power controller to raise voltage to the maximum
1. Ensure any previous GT Driver Mailbox transaction is complete.
 2. Write GT Driver Mailbox Data Low = 0x00000003.
 3. Write GT Driver Mailbox Data High = 0x00000000.
 4. Write GT Driver Mailbox Interface = 0x80000007.
 5. Poll GT Driver Mailbox Interface for Run/Busy indication cleared (bit 31 = 0).
 - Timeout after 150 us. Do not change CD clock frequency if there is a timeout.
 6. Read GT Driver Mailbox Data Low, if bit 0 is 0x1, continue, else go to step 2.
 - If the condition in step 6 is not satisfied after cycling through steps 2-6 for 3 ms (typically <200 us), timeout and fail and do not change clock frequency.

Sequence After Frequency Change

- This sequence requests the power controller to set the voltage to the selected level. The power controller may choose to keep the voltage higher to accommodate other users outside of display.
1. Write GT Driver Mailbox Data Low with the voltage level selection, following the table above.
 - For Level 0, write 0x00000000.
 - For Level 1, write 0x00000001.
 - For Level 2, write 0x00000002.
 - For Level 3, write 0x00000003.
 2. Write GT Driver Mailbox Data High = 0x00000000.
 3. Write GT Driver Mailbox Interface = 0x80000007.
 - There is no need for display software to wait for the voltage to adjust.

Sequences for Changing CD Clock Frequency

Restrictions
<p>The CD clock frequency impacts the maximum supported pixel rate and display watermark programming. The CD clock frequency must be at least twice the frequency of the Azalia BCLK.</p>

Sequence for Changing CD Clock Frequency

1. Unless changing only the CD2X Divider, disable all display engine functions using the full mode set disable sequence on all pipes, ports, and planes.
 - Includes Global Time Code
 - Display power wells may be left enabled
2. Follow the Display Voltage Frequency Switching - Sequence Before Frequency Change

3. Enable or change the frequency of CD clock
 - a. If enabling CDCLK PLL
 - i. Write CDCLK_PLL_ENABLE with the PLL ratio, but not yet enabling it.
 - ii. Set CDCLK_PLL_ENABLE PLL Enable
 - iii. Poll CDCLK_PLL_ENABLE for PLL lock
 - iv. Timeout and fail if not locked after 200 us
 - v. Write CDCLK_CTL with the CD2X Divider selection and CD Frequency Decimal value to match the desired CD clock frequency
 - b. If disabling CDCLK PLL
 - i. Clear CDCLK_PLL_ENABLE PLL Enable
 - ii. Poll CDCLK_PLL_ENABLE for PLL unlocked
 - iii. Timeout and fail if not unlocked after 200 us
 - c. If changing the CDCLK PLL frequency
 - i. Follow steps above for disabling CDCLK PLL.
 - ii. Follow steps above for enabling CDCLK PLL, using the new PLL ratio.
 - d. If changing only the CD2X Divider
 - i. Write CDCLK_CTL with the CD2X Pipe selection, CD2X Divider selection, and CD Frequency Decimal value to match the desired CD clock frequency
 - ii. If pipe is enabled, wait for start of vertical blank for change to take effect
4. Follow the Display Voltage Frequency Switching - Sequence After Frequency Change
5. Update programming of functions that use the CD clock frequency

If these features are not currently enabled, the programming can be delayed to when they are enabled.

- Wireless Display 27 MHz frequency in the WD_27_M and WD_27_N registers.
 - For CD clock 168 MHz, program M=9 and N=56 (decimal).
 - For CD clock 336 MHz, program M=9 and N=112 (decimal).
 - For CD clock 528 MHz, program M=9 and N=176 (decimal).
 - For CD clock 172.8 MHz, program M=45 and N=288 (decimal).
 - For CD clock 307.2 MHz, program M=45 and N=512 (decimal).
 - For CD clock 556.8 MHz, program M=45 and N=928 (decimal).
 - For CD clock 652.8 MHz, program M=45 and N=1088 (decimal).
 - For CD clock 180 MHz, program M=45 and N=300 (decimal).
 - For CD clock 312 MHz, program M=45 and N=520 (decimal).
 - For CD clock 552 MHz, program M=45 and N=920 (decimal).
 - For CD clock 648 MHz, program M=45 and N=1080 (decimal).
- Utility pin backlight frequency and duty cycle in the BLC_PWM_DATA register.

Resets

The north and south display engines are reset by PCI Function Level Resets (FLR) and the chip level resets.

The south display engine runs panel power down sequencing (if configured to do so by NDE_RSTWRN_OPT) before resetting.

NDE_RSTWRN_OPT

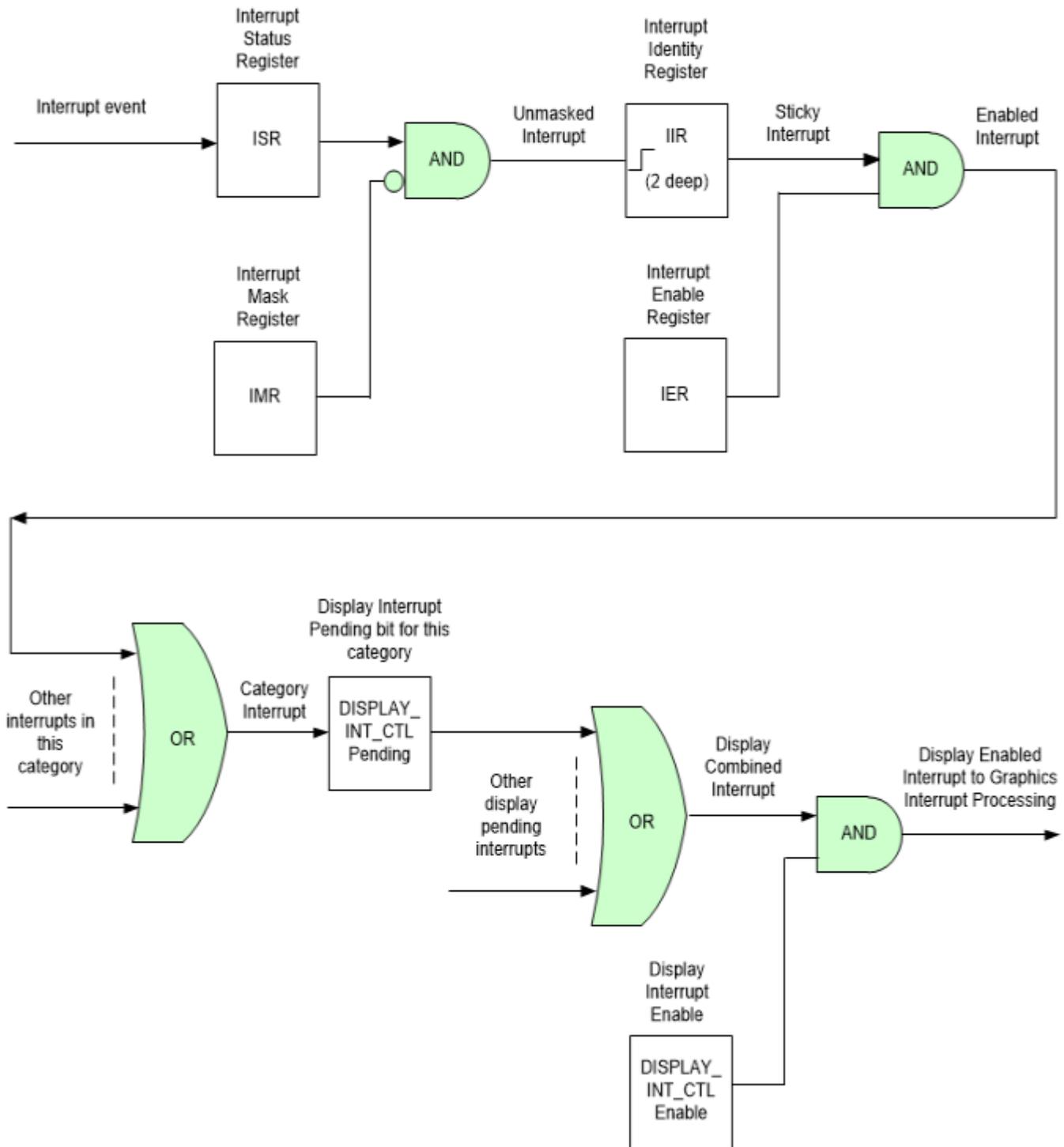
Shared Functions

Display Interrupts

Registers
DISPLAY_INT_CTL
DE Pipe Interrupt Definition
DE Port Interrupt Definition
DE Misc Interrupt Definition
Audio Codec Interrupt Definition
INTERRUPT Structure
DE HPD Interrupt Definition
Graphics Primary Interrupt
Graphics Interrupt Introduction

Interrupt Flow

First Level Interrupts in Display

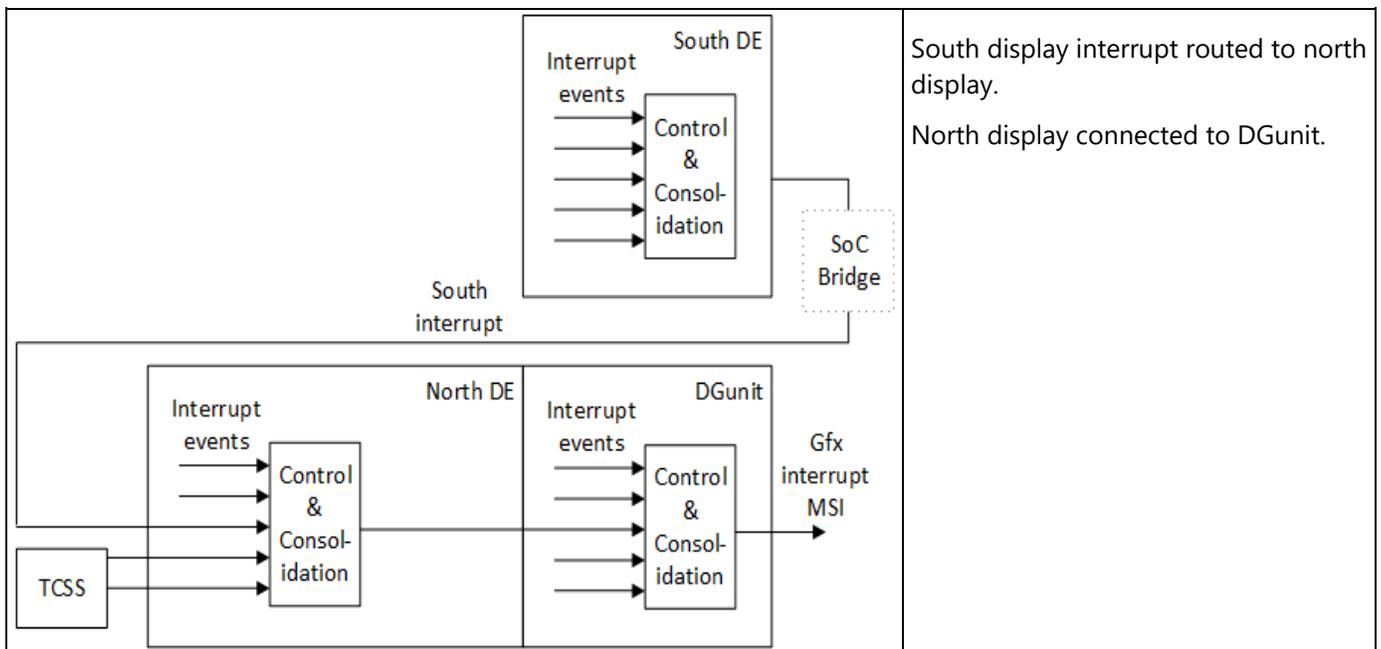


1. For every first level interrupt bit
 - a. The interrupt event comes in to the interrupt handling logic.
 - There may be more levels of interrupt handling behind each event. For example, the PCH Display interrupt event is the result of the SDE interrupt registers.
 - b. The interrupt event goes to the Interrupt Status Register (ISR) where live status can be read back.
 - The live status is mainly useful for hotplug interrupts where it gives the live state of the hotplug line.
 - The live status is not useful for pulse interrupt events due to the short period that the status will be present.
 - c. The interrupt event is ANDed with the inverted Interrupt Mask Register (IMR) to create the unmasked interrupt.
 - d. The unmasked interrupt rising edge sets the sticky bit in the Interrupt Identity Register (IIR).
 - The IIR can be cleared by writing a 1 to it.
 - The IIR can queue up to two interrupt events. When the IIR is cleared, it will set itself again if a second event was stored.
 - e. The sticky interrupt is ANDed with the Interrupt Enable Register (IER) to create the enabled interrupt.
2. All enabled interrupts are then ORed by category (Pipe, Audio, etc.) to create a category interrupt which is then visible in one of the Display Interrupt Control Register (DISPLAY_INT_CTL) pending category bits.
3. All pending interrupts are then ORed to create the display combined interrupt.
4. The display combined interrupt is ANDed with the Display Interrupt Enable (DISPLAY_INT_CTL Bit 31) to create the display enabled interrupt.
5. The display enabled interrupt then goes to graphics interrupt processing before eventually creating an interrupt message which will reach the OS.

Interrupt Service Routine

1. Read graphics primary interrupt register to find a display interrupt pending.
2. Write graphics primary interrupt register to clear display interrupt.
3. Disable Display Interrupt Control (can be done here, or anywhere before the final step)
 - Clear bit 31 of DISPLAY_INT_CTL.
 - This de-asserts the display enabled interrupt and is required in order for the final step to cause a new assertion when all interrupts are not cleared during the service routine.
4. Find the category of interrupt that is pending
 - Read DISPLAY_INT_CTL and record which interrupt pending category bits are set.
5. Find the source(s) of the interrupt and clear the Interrupt Identity bits (IIR)
 - Read the IIR associated with each pending interrupt category, record which bits are set, then write back 1s to clear the bits that are set.

- There can be up to 2 interrupts recorded per source, requiring multiple writes to the IIR to fully clear.
6. Process the interrupt(s) that had bits set in the IIRs.
 7. Optionally go to step 4 again to check for any new display interrupts that have happened during previous steps
 - This is optional since any new interrupts will also be found when they trigger the interrupt service routine again after the final step and then re-enter from the start of the routine.
 8. Re-enable Display Interrupt Control
 - Set bit 31 of DISPLAY_INT_CTL.
 - If interrupts were not fully cleared before this point, then the display enabled interrupt will re-assert and there will be a new display interrupt in graphics primary interrupt register.



MBus

Mbus is an internal ring type bus. Display high priority clients reside on this bus. The introduction of a ring architecture allow for easy addition/deletion of clients as well as improves Bandwidth. The various "boxes" are different types of tap points where display clients are plugged in.

MBus
MBUS_ABOX_CTL
MBUS_BBOX_CTL
MBUS_DBOX_CTL
MBUS_UBOX_CTL

MBus programming during display Initialization

Program credits in MBUS_ABOX_CTL, MBUS_ABOX1_CTL, MBUS_ABOX2_CTL

The MBus credits should be setup once with the following default values during the display initialization.

MBUS_ABOX_CTL for this pipe -> BT Credits Pool1 = 16

MBUS_ABOX_CTL for this pipe -> BT Credits Pool2 = 16

MBUS_ABOX_CTL for this pipe -> B Credits = 1

MBUS_ABOX_CTL for this pipe -> BW Credits = 1

DBUF_CTL for this pipe -> Tracker State Service = 8

The following programming must be done when enabling each pipe as a part of *configure other pipe settings* in the *Enable Planes, Pipe, and Transcoders* sequence.

MBUS_DBOX_CTL for this pipe -> A Credits = 2

MBUS_DBOX_CTL for this pipe -> BW Credits = 2

MBUS_DBOX_CTL for this pipe -> B Credits = 12

Fuses and Straps

FUSE_STATUS

DFSM

DSSM

DFSDONE

Render Response

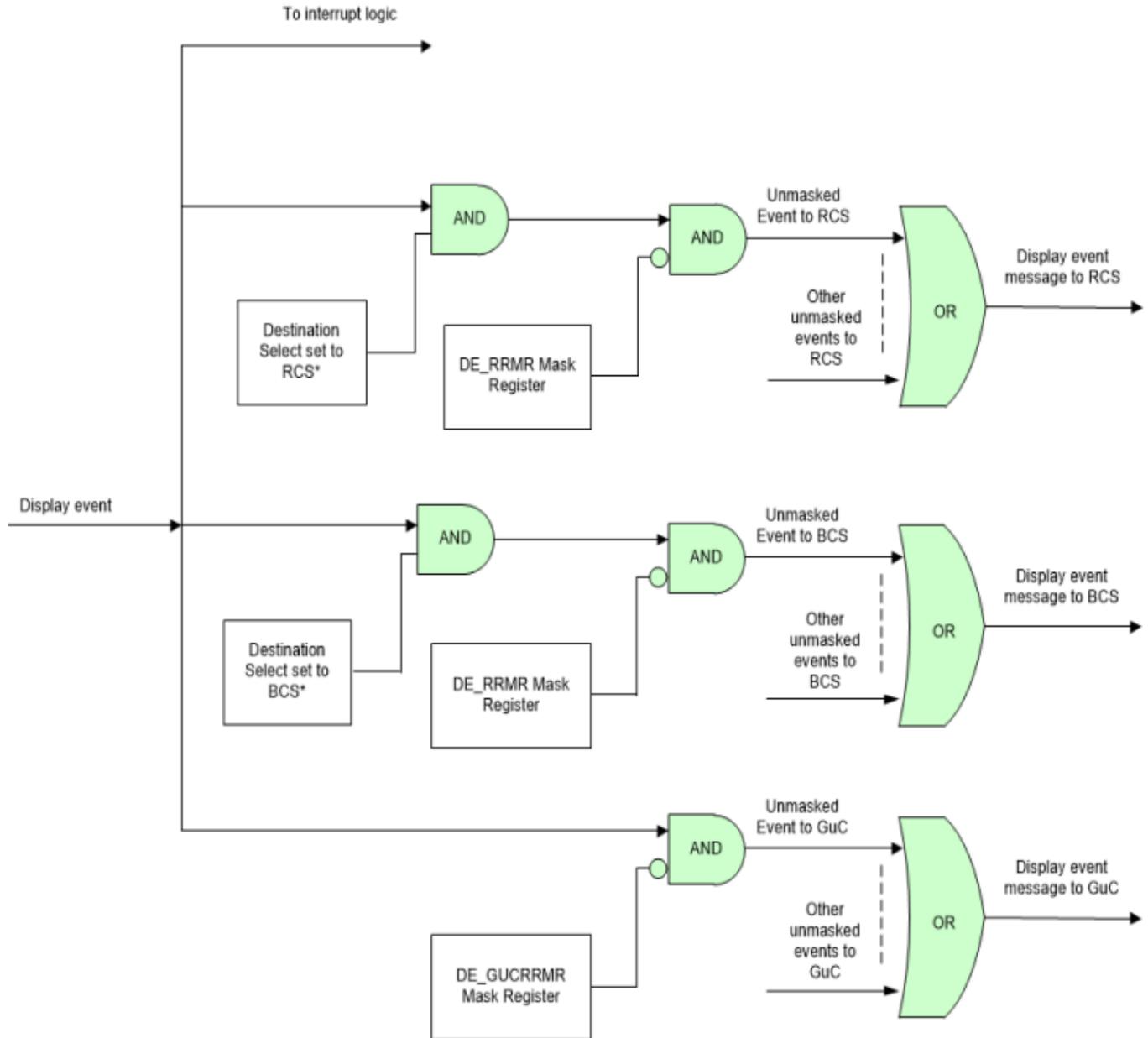
Display Engine Render Response Message Definition

DE_RRMR

DE_RRMR_DW1

DE_RRMR_DW2

DE_RR_DEST



*For Vblank events, destination select is configured in DE_RR_DEST to direct the event to RCS, BCS, or both at once.
 For MMIO triggered flip done events, destination select is configured in PLANE_SURF Ring Flip Source to direct the event to either RCS or BCS.
 For ring (AKA message) triggered flip done events, destination select is configured automatically by decoding whether the flip came from BCS or RCS.

GuC supports some additional events beyond what RCS and BCS support. In that case the event can feed into the GuC masking directly without branching to RCS and BCS.

Older documentation refers to RCS as CS.

Hot Plug Detection

This topic describes hot plug detection for type C ports with DP alternate mode or DP over thunderbolt. Hotplug detection for non-typeC is done through the south display.

Type C Port Hot Plug Detection

HOTPLUG_CTL

These registers are used for detecting hot plug on thunderbolt or type C (DP alternate). They will generate interrupts in DE_HPDP_INTERRUPT. HOTPLUG_CTL has the status bits to indicate if long or short pulses are detected. Do not use the interrupt ISR for finding the live connect state. It can become out of sync with the typeC sub-system after some power states and resets. Instead, use DFLEXDPSP TC Live State fields.

The south display hot plug detection registers are used for detecting hot plug for combo PHY ports and type C ports with non-type C connectors (legacy or static configuration).

HPD Interrupt Sequence

- This sequence is for identifying each type of hotplug. Follow interrupt documentation for how interrupt service routine should enable, disable, and clear interrupts.
1. Display interrupt received and DISPLAY_INT_CTL shows source of the interrupt is DE HPD Interrupts pending (north display HPD)
 2. Read DE_HPDP_INTERRUPT IIR to find if source port type is thunderbolt or typeC (DP alternate)
 3. Read <port type, TBT for thunderbolt, TC for DP alternate>_HOTPLUG_CTL to find the HPD Status
 - Long pulse, short pulse, both, or none
 4. If long pulse or both
 1. Read PORT_TX_DFLEXDPSP TC Live State to find the live state
 - Long pulse with live state = 1 indicates a connect
 - Long pulse with live state = 0 indicates a disconnect
 2. Follow the TypeC Programming connect flow or disconnect flow
 5. If short pulse and not HDMI
 1. Query the panel and then proceed as required by panel

Resets and Power States

Hotplug interrupts can be missed around resets and power states. When coming out of any state where HPD interrupts can be missed (cold boot, warm reset, S3/S4, DC9, Function Level Reset, handoff between VBIOS/GOP/EFI and driver, etc.) use the following sequence to enable HPD interrupts and find if any previous HPD was missed and needs to be serviced.

1. Enable HOTPLUG_CTL and configure interrupt related registers so that HPD interrupts will be received.



2. If HPD interrupt received after step 1, handle it as usual and abort the remaining steps for that port.
3. Process live state for each typeC port.
 - Read PORT_TX_DFLEXDPPMS DPPMSTC, PORT_TX_DFLEXDPCSSS DPPMSTC, and PORT_TX_DFLEXDPSP TC Live State.
 - Process connect or disconnect as per truth table, abort if HPD interrupt is received for this port and handle the disconnect/connect requested by the HPD.

The table below illustrates the next steps based on the live state and connection state.

DPSP TC Live State	DPPMS DPPMSTC	DPCSSS DPPMSTC	IOM State	Software Next Steps
00	0	0	Disconnect complete	Do nothing right now. IOM will send HPD if something is needed.
00	1	0	IOM action pending	Do nothing right now. IOM will send HPD if something is needed.
01	1	0	DP-alt connect pending	Follow connect flow for DP alternate.
01	1	1	DP-alt connect complete	Follow connect flow for DP alternate from the step after DPCSSS DPPMSTC is set to 1.
00	1	1	DP-alt disconnect pending	Follow disconnect flow for DP alternate.
10	x	x	TBT connect complete	Follow connect flow for thunderbolt.
00	0	1	Invalid	Follow disconnect flow for DP alternate.
01	0	0	Invalid	Do nothing right now
01	0	1	Invalid	Follow disconnect flow for DP alternate.
11	x	x	Invalid	Follow disconnect flow for DP alternate.

Refer to the TypeC Programming page for details on signaling between IOM and display driver.

Arbiter

ARB_CTL

ARB_CTL2

PIPE_ARB_CTL

BW_BUDDY0_CTL

BW_BUDDY1_CTL

BW_BUDDY2_CTL

BW_BUDDY1_PAGE_MASK

BW_BUDDY2_PAGE_MASK

Bandwidth Buddy Registers

These registers are used to improve bandwidth efficiency by pairing up memory requests with similar addresses. The mask register selects which address bits to compare when buddying up transactions. The mask must be programmed based on the memory type and memory controller attributes.

Abox1 and Abox2 are used for pixel data reads. Program the BW_BUDDY1 and BW_BUDDY2 registers.

Optimal programming of the BW_BUDDY_PAGE_MASK registers depends on the memory configuration used. Program the BW_BUDDY_PAGE_MASK according to the table below. For memory configurations not listed below, set the BW_BUDDY_CTL Disable bit.

MEMORY CHANNELS/Number of modules	MEMORY TYPE	BW_BUDDY_PAGE_MASK
1/1	DDR4	0x0000000F
1/1	DDR5	0x0000000F
2/1	LP4,LP5	0x0000001C
2/2	DDR4	0x0000001F
2/2	DDR5	0x0000001E
4/2	LP4,LP5	0x00000038

Data Buffer

DBUF_CTL

DBUF_STATUS

DBUF_ECC_STAT

DBUF_CTL2

See the Display Buffer Programming section for information on how to allocate the buffer to planes.

Backlight

This section refers to the CPU display backlight control over the utility pin. For PCH display backlight control, see South Display Engine Registers.

The backlight PWM output frequency is determined by the PWM clock frequency, increment, and frequency divider.

PWM output frequency = PWM clock frequency / PWM increment / PWM frequency divider

The frequency divider must be greater than or equal to the number of brightness levels required by software; typically 100 or 256.

Backlight Enabling Sequence

1. Enable utility pin, select PWM mode, and set polarity in UTIL_PIN_CTL Util Pin Enable, Util Pin Mode, and Util Pin Output Polarity.
2. Set frequency and duty cycle in BLC_PWM_DATA Backlight Frequency and Backlight Duty Cycle.
3. Enable PWM output and set granularity in BLC_PWM_CTL PWM Enable and PWM granularity.

...

4. Change duty cycle as needed in BLC_PWM_DATA Backlight Duty Cycle.

If needed, granularity and polarity can be programmed earlier in the enabling sequence.

Backlight Frequency Change Sequence

1. Disable PWM output in BLC_PWM_CTL PWM Enable.
2. Set new frequency and duty cycle in BLC_PWM_DATA Backlight Frequency and Backlight Duty Cycle.
3. Enable PWM output and set granularity in BLC_PWM_CTL PWM Enable and PWM granularity.

Backlight Registers

BLC_PWM_CTL

BLC_PWM_DATA

Miscellaneous Shared Functions

UTIL_PIN_CTL

UTIL_PIN_BUF_CTL

UTIL2_PIN_CTL

UTIL2_PIN_BUF_CTL

The utility pin(s) can be used for various functions, including MIPI DSI TE and genlock, on SoCs supporting those functions. The pins must be configured for those functions as part of the function enable and disable sequences.

AUDIO_PIN_BUF_CTL

Double Buffer Control

DOUBLE_BUFFER_CTL

Double Buffer Disabling/Stalling

Double buffer stalling (also called disabling) is used to synchronize register updates across multiple display resources for an atomic update. While stalled, double buffer registers can be written with new values, updating the 1st stage buffer, but the double buffering will not activate to transfer those values to the 2nd stage (live/working value) buffer to be used by the hardware resource. This allows for multiple register programming to cross the double buffer update point and keep the registers in sync.

Without double buffer stalling, updates to multiple registers can become un-synchronized if the double buffer update event (typically vblank rising edge) happens during the programming. Then some registers will have double buffer updates a frame before other registers and the screen contents will be incorrect.

The double buffer stalling is controlled by the global double buffer disable. Disabling double buffering means it is stalled.

- DOUBLE_BUFFER_CTL Global Double Buffer Disable register bit is used to stall double buffering in all pipes and transcoders.

Allow Double Buffer Stall

The double buffer stalling is also controlled by the Allow Double Buffer Update Disable register bit that each resource has in one of its control registers. The Allow Double Buffer Update Disable is useful if certain resources need to update freely and not be tied into the synchronizing of other resources, such as if cursor update needs to happen as soon as possible while some plane updates are being stalled to synchronize together.

The double buffering for a particular resource will only be stalled if that resource's Allow Double Buffer Update Disable is set and the global double buffer disable is set.

Details

This only stalls the double buffer update for periodic events, like the start of vertical blank. It does not change the behavior for constant events, like pipe not enabled.

Double buffer stalling does not stall asynchronous flips initiated by MMIO or command streamers.

Synchronous flips will not complete or give the flip done indication while double buffering is stalled for a plane. They will complete and give the flip done at the next double buffer update point after the double buffering is no longer stalled.

Sequence for synchronizing double buffer updates

1. If not already set, set the Allow Double Buffer Update Disable field for each resource to be synchronized together.
2. Set the Global Double Buffer Disable register bit. Double buffering is now stalled.
3. Program the registers that need to be synchronized together.
4. Clear the Global Double Buffer Disable register bit. Double buffering is now un-stalled. Any pending updates will take place at the next periodic update event.
5. Depending on configuration, variable refresh rate and PSR require extra programming to trigger a frame update after the registers have been updated. See the VRR and PSR programming pages.
6. If a resource no longer needs to be synchronized, clear the Allow Double Buffer Update Disable field for that resource. If the resource will continue to be synchronized, the field can remain set.

It is not necessary to set and clear the Allow Double Buffer Update Disable around every register update. Allow Double Buffer Update Disable only has to change if a resource's requirements for synchronization are changed.

Central Power

SAGV

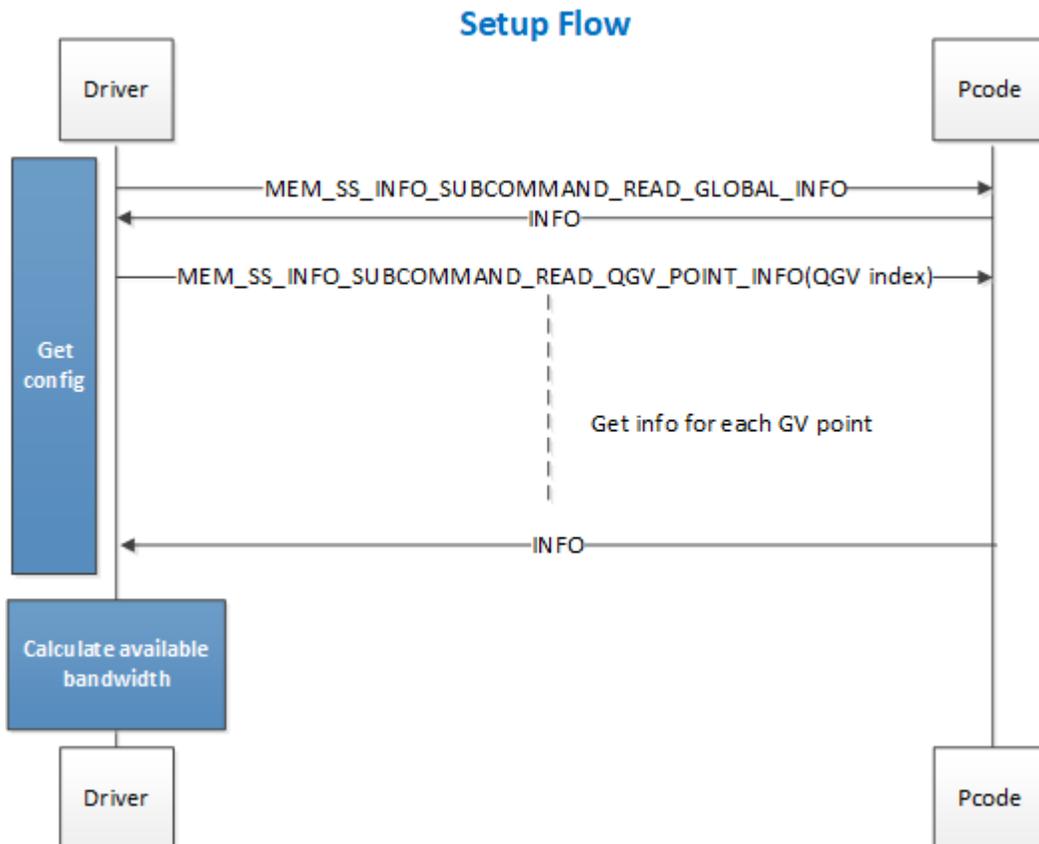
System Agent Geyserville (SAGV) dynamically adjusts the system agent voltage and clock frequencies depending on power and performance requirements. SAGV impacts display engine in two ways. SAGV point selections can limit the system memory bandwidth to display. SAGV transitions can temporarily block display engine access to system memory. Display software must restrict the SAGV point selection to control the bandwidth availability and setup watermarks to tolerate the temporary memory block.

SAGV Point Selection

Setup

Find the memory bandwidth availability for display. The results can be cached so that the flow does not have to be rerun.

The driver does not have to use the mailbox if it has other ways of finding the memory and GV point info.



1. Driver reads memory subsystem (MEM SS, MemSS) configuration information from Pcode using the GT Pcode mailbox command MAILBOX_GTDRIVER_CMD_MEM_SS_INFO, first reading the global info and number of GV points, then reading GV info for each GV point.
 - Described in the Mailbox Commands section
2. Driver uses the mailbox info to calculate the available memory bandwidth for each GV point and number of enabled planes.
 - Described in the Bandwidth Restrictions chapter, Available Memory Bandwidth Calculation section.

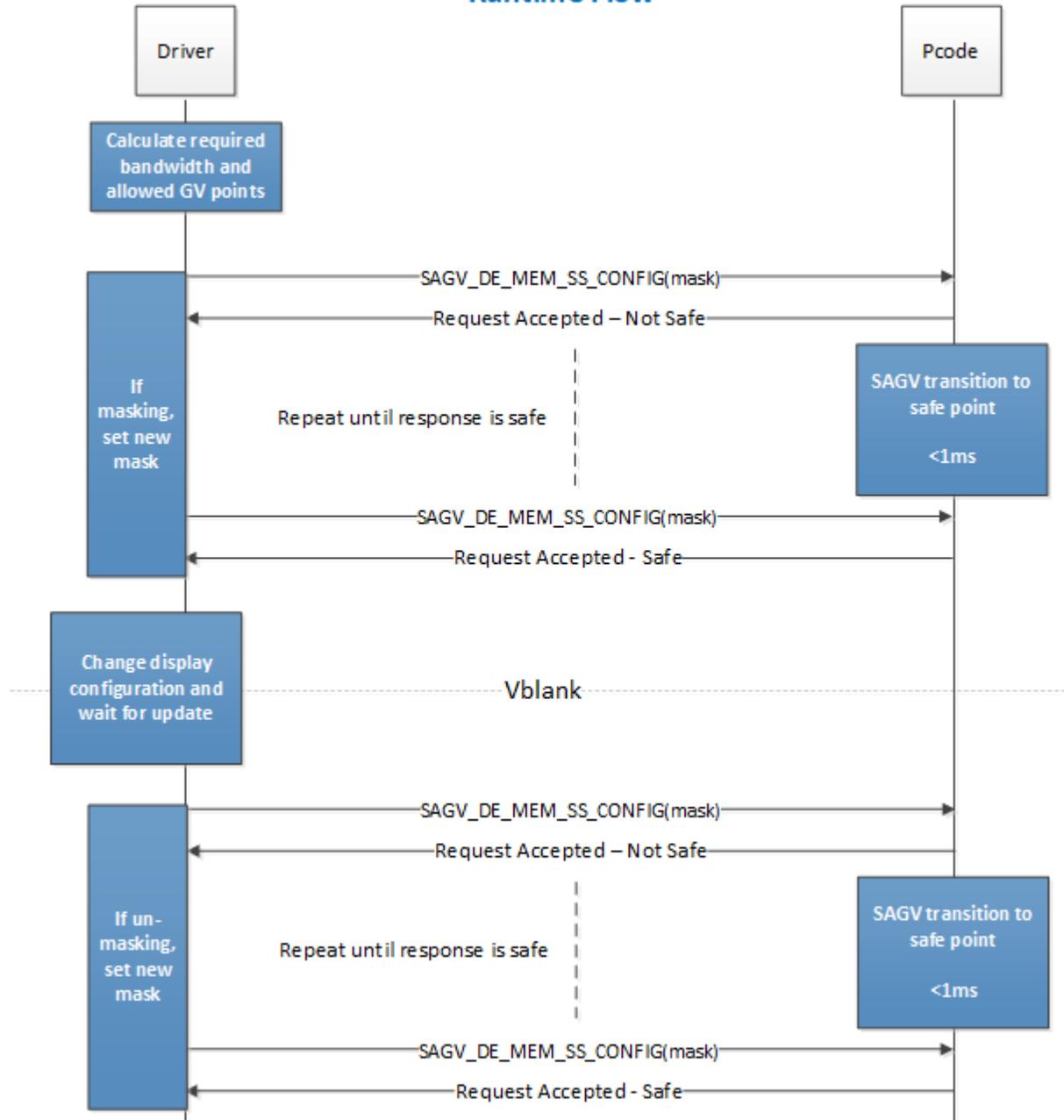
There are cases where the highest numbered GV point allows less bandwidth for display than a lower point.

Overclocking reprogramming of memory parameters dynamically (system running without a reboot when changes are applied) can result in a display underrun since the display driver will not know that the configuration has changed.

Runtime

Adjust the allowed GV points to match display configuration changes.

Runtime Flow



1. Before changing display configuration in a way that impacts required bandwidth or number of planes, driver calculates the new bandwidth requirement.
 - Described in the Bandwidth Restrictions chapter, Required Memory Bandwidth Calculation section.
2. Driver finds which GV points supply enough available bandwidth to meet the required bandwidth and which GV points do not supply enough and must be restricted (masked off).
 - At least one GV point of each type must always remain unmasked. The GV point of each type providing the highest bandwidth for display must always remain unmasked.

- Each cycle through this sequence can only mask or unmask points of a GV type, not do both for a single GV type. It is allowed for one GV type to unmask while another masks, then one cycle through this sequence will mask one type of GV and the unmask another type of GV.
 - If no GV point will provide enough bandwidth, then the display configuration must be constrained to fit within the available bandwidth.
 - If driver has lost track of what was previously masked off, it can mask off all but the highest bandwidth point of each type to get to a known safe setting, then cycle through the full sequence again to set the correct mask.
 - To disable SAGV when watermarks do not meet the SAGV block time requirement, mask off all the QGV points except for the point providing the highest bandwidth for display. To re-enable SAGV, unmask all of the QGV points that meet the bandwidth requirements for display. See the Watermark Calculations section for more info.
3. If masking GV points
 1. Driver uses GT Pcode mailbox command MAILBOX_GTDRIVER_CMD_SAGV_DE_MEM_SS_CONFIG to send Pcode the bitmask of restricted points and polls by repeatedly issuing the command with the same bitmask until receiving the acknowledge with point safe for each type, hitting a timeout after 1 millisecond, or receiving the error code.
 - Command described in the Mailbox Commands section
 - Pcode adjusts GV
 1. Pcode performs needed changes to ensure that all future transitions will be performed to the allowed states only.
 2. If pcode currently resides in a point which is restricted, it issues a GV transition to an allowed state.
 3. Pcode responds to the mailbox with point safe.
 - If only one type of GV is masking, keep the mask unchanged for other types.
 4. Driver changes the display configuration
 5. If unmasking GV points
 1. Driver waits for the configuration to be updated (typically at the next vertical blank). Driver can wait even longer, but should eventually complete unmasking in order to let GV move between points to give more optimal power and performance.
 2. Driver uses GT Pcode mailbox command MAILBOX_GTDRIVER_CMD_SAGV_DE_MEM_SS_CONFIG to send Pcode the bitmask of restricted points and polls by repeatedly issuing the command with the same bitmask until receiving the acknowledge with point safe for each type, hitting a timeout after 1 millisecond, or receiving the error code.
 - Command described in the Mailbox Commands section
 - Pcode adjusts GV
 1. Pcode performs needed changes to ensure that all future transitions will be performed to the allowed states only.

2. If pcode currently resides in a point which is restricted, it issues a GV transition to an allowed state.
 3. Pcode responds to the mailbox with point safe.
- If only one type of GV is unmasking, keep the mask unchanged for other types.

All points are unrestricted by default until driver requests otherwise or BIOS disables SAGV.

Pcode will reset the bitmask from display on the graphics device function level reset (FLR).

The BIOS mailbox for SAGV gets precedence over Display Mailbox for GV request.

Display driver must allow at least one GV point of each type at all times. Driver must not mask off all the points of any type.

Display driver must wait for the previous GV point restriction to finish before starting a new restriction.

Legacy Behavior

The legacy command MAILBOX_GTDRIVER_CMD_DE_LTR_SETTING has the effect of masking off all but the highest numbered QGV point. the command continues to be supported and can be used as a limited alternative to MAILBOX_GTDRIVER_CMD_SAGV_DE_MEM_SS_CONFIG. The highest numbered QGV point is not always the highest bandwidth for display, and masking off more QGV points than necessary is not optimal for power and performance, so the legacy command is of limited use and should be phased out by newer software. Eventually a future project may de-feature the legacy command.

Mailbox Commands

Mailbox Access Routine

1. Ensure any previous GT Driver Mailbox transaction is complete
2. Write GT Driver Mailbox Data0= <request data 31:0> and GT Driver Mailbox Data1= <request data 63:32>
3. Write GT Driver Mailbox Interface RUN_BUSY=1, PARAM2= <parameter2>, PARAM1= <parameter1>, COMMAND= <command>
4. Poll GT Driver Mailbox Interface for RUN_BUSY==0
 - Timeout and fail after 150 us
 - Response COMMAND encoding listed below
5. Read GT Driver Mailbox Data0= <response data 31:0> and Data1= <response data 63:32>

MAILBOX_GTRDIVER_CMD_MEM_SS_INFO

This mailbox command provides the MemSS information requested by display. Pcode collates this information from MemSS registers, parses display relevant information and passes it to display on query. It contains subcommands selected through the field PARAM1[15:8] of the INTERFACE register.

MAILBOX_GTRDIVER_CMD_MEM_SS_INFO_SUBCOMMAND_READ_GLOBAL_INFO

Subcommand used to read MemSS global configuration.

Mailbox COMMAND=0xD and PARAM1=0x0

Response Data Bits	Description
3:0	DDR Type; 0:DDR4, 1:DDR5, 2:LPDDR5, 3:LPDDR4, 4:DDR3, 5:LPDDR3
7:4	Number of populated channels
11:8	Number of enabled QGV points

If GV is fused disabled, BIOS should configure GV such that only 1 point will be reported.

MAILBOX_GTRDIVER_CMD_MEM_SS_INFO_SUBCOMMAND_READ_QGV_POINT_INFO

Subcommand used to read QGV point related timing info. Input is the QGV index following the number of enabled QGV points from the global configuration. Out of bounds QGV index results in error code MAILBOX_GTDRIVER_CC_ILLEGAL_DATA error.

Mailbox COMMAND=0xD and PARAM1=0x1 and PARAM2= <QGV index, counting from 0>

Response Data Bits	Description
15:0	Dclk in multiples of 16.6666 MHz
23:16	TRP in DCLKs
31:24	tRCD in DCLKs
39:32	TRDPRE in DCLKs
48:40	TRAS in DCLKs

Pcode reports the maximums values from across multiple memory channels.

MAILBOX_GTDRIVER_CMD_SAGV_DE_MEM_SS_CONFIG

This mailbox command provides pcode with the bitmap containing the list of allowed GV operation points. Value of 1 in the mask is restricting the associated GV point.

Mailbox COMMAND=0xE

Request Data Bits	Description
0	Restricted QGV point 0
1	Restricted QGV point 1
2	Restricted QGV point 2
3	Restricted QGV point 3
4	Restricted QGV point 4
5	Restricted QGV point 5
6	Restricted QGV point 6
7	Restricted QGV point 7
Other bits	Reserved: Must program all 0s



The response data indicates when a safe GV point has been reached.

Response Data Encoding
Bit 1:0
0x0 = Request accepted. QGV point safe.
0x1 = Request accepted. QGV point not safe yet. Poll again.
0x2 = Request rejected error. QGV point cannot be made safe. It may be blocked by a BIOS override or fuse. Retry with different points or fail.

The response command indicates if there were any errors.

Response COMMAND Encoding
00h = Success
03h = Illegal data or all GV points masked off
04h = Illegal subcommand
06h = Mailbox locked (BIOS loading not done or GV points not programmed or BIOS has only enabled a single GV point)
11h = Rejected (BIOS overriding the GV selection)

Pcode will quickly respond to accept the request, then take up to 1 millisecond (typically 100-200us) to move to a safe point. Driver has to poll Pcode with repeated MAILBOX_GTDRIVER_CMD_SAGV_DE_MEM_SS_CONFIG commands (with unchanged request data field) to find when the move to the safe point is complete.

MAILBOX_GTDRIVER_CMD_DE_LTR_SETTING (Legacy Command)

This mailbox command is the legacy method for disabling Qclk SAGV. It requests Pcode to move to the highest numbered enabled Qclk GV point and then hold there, equivalent to masking off all but one QGV point.

Mailbox COMMAND=0x21

Request Data Bits	Description
2:0	EL_THLD LTR Override: 0 = Disable Qclk GV 3 = Enable Qclk GV
3	Interlace Override: Unused must program with 0
4	VGA Override: Unused must program with 0
63:5	Reserved: Must program all 0s

The response data indicates when a safe Qclk point has been reached.

Response Data Encoding
0x0 = Request accepted. Qclk point not safe yet. Poll again.
0x1 = Request accepted. Qclk point safe.

Pcode will quickly respond to accept the request, then take up to 1 millisecond (typically 100-200us) to move to a safe point. Driver has to poll Pcode with repeated MAILBOX_GTDRIIVER_CMD_DE_LTR_SETTING commands (with unchanged request data field) to find when the move to the safe point is complete.

Frame Buffer Compression

FBC Registers

FBC_CFB_BASE

FBC_CTL

DPFC_CONTROL_SA

DPFC_CPU_FENCE_OFFSET

FBC Overview

Frame Buffer Compression (FBC) gives a lossless compression of the display frame buffer to save power by reducing system memory read bandwidth and increasing the time between display engine reads to system memory.

FBC is only available on specific plane(s), depending on project. See FBC_CTL for details. FBC compresses pixels for the plane(s) it is attached to as they are displayed. The compressed data is written into the Compressed Frame Buffer (CFB) in graphics data stolen memory. The compressed data is then read the next time the same line needs to be displayed. Changes to the display front buffer (currently displayed memory surface) through host aperture (GMADR) tiling fences (on projects that support that), render (RCS), and blitter (BCS) are tracked and cause the compressed lines to be invalidated and recompressed. Flips or changes to plane size and panning cause the entire buffer to be recompressed (nuke).

FBC Compression Limit

The FBC compression limit reduces the size of the Compressed Frame Buffer (CFB) by limiting which lines will be compressed. This is used when the graphics stolen memory available for the FBC CFB is smaller than the size of the original uncompressed frame buffer. There is also a FBC compressed vertical limit, listed in FBC_CTL, which is the maximum number of lines FBC can compress. Lines beyond the vertical limit do not need to be accounted for in the CFB size.

When the compression limit is 1:1, every line is written to the CFB, so the CFB width is the same as the original uncompressed frame buffer.

When the compression limit is 2:1, only lines that compress to 1/2 their original size will be written to the CFB, so the CFB width can be 1/2 the original uncompressed frame buffer.



When the compression limit is 4:1, only lines that compress to 1/4 their original size will be written to the CFB, so the CFB width can be 1/4 the original uncompressed frame buffer.

DSM allocation in bytes = min (640, ceiling (plane height / 4))* Compressed buffer stride in cachelines * 64

The 16bpp plane pixel format requires compression limit to be 2:1 or 4:1.

Compressed buffer stride in cachelines = at_least (ceiling [plane_width_in_pixels / (32 * compression_limit_factor)] * 8 + 1)

FBC Programming Overview

1. Set up the compressed frame buffer.
 - The compressed buffer resides in graphics data stolen memory.
 - The stolen memory must be contiguous and un-cached.
 - The stolen memory needed for compressed frame buffer must be greater or equal to CFB size (calculation above).
 - Manage the compressed buffer size at run-time by balancing other graphics memory needs with the FBC allocation, and implement appropriate memory needs prioritization schemes.
2. Tracking for CPU host front buffer modifications
 - CPU Host Front Buffer Tracking sequence below
3. Tracking for display front buffer rendering
 - Render Tracking sequence below
4. Tracking from display front buffer BLTs
 - Blitter Tracking sequence below

LRI commands to MSG_FBC_REND_STATE are used as part of the render and blitter tracking. Those LRIs must be followed SRM commands to the same address.

LRI to MSG_FBC_REND_STATE with data 0x00000004 tells FBC to nuke and invalidate the entire compressed buffer.

Render Tracking With Nuke

- Software must send the nuke LRI after each render to the display front buffer
1. Render commands that touch the display front buffer
 - a. Render submission
 - b. PIPE_CONTROL
 - c. LRI to MSG_FBC_REND_STATE with data 0x00000004 (nuke)
 - d. SRM to read MSG_FBC_REND_STATE and store to a scratch page
 2. More render commands that touch the display front buffer
 - a. Render submission
 - b. PIPE_CONTROL

- c. LRI to MSG_FBC_REND_STATE with data 0x00000004 (nuke)
 - d. SRM to read MSG_FBC_REND_STATE and store to a scratch page
3. Render commands that do not touch the display front buffer
 - a. Render submission
 - b. PIPE_CONTROL

Blitter Tracking With Nuke

- Software must send the nuke LRI after each BLT to the display front buffer. Never set 221d0h bit 3 (Address Valid for FBC).
1. BLT commands that touch the display front buffer
 - a. BLT submission
 - b. MI_FLUSH_DW
 - c. LRI to MSG_FBC_REND_STATE with data 0x00000004 (nuke)
 - d. SRM to read MSG_FBC_REND_STATE and store to a scratch page
 2. More BLT commands that touch the display front buffer
 - a. BLT submission
 - b. MI_FLUSH_DW
 - c. LRI to MSG_FBC_REND_STATE with data 0x00000004 (nuke)
 - d. SRM to read MSG_FBC_REND_STATE and store to a scratch page
 3. BLT commands that do not touch the display front buffer
 - a. BLT submission
 - b. MI_FLUSH_DW

MSG_FBC_REND_STATE address for pipe A FBC is 0x50380

CPU Host Front Buffer Tracking

- Host tracking must be setup by software, then hardware will track FBC line invalidations to the specified fence in the graphics aperture.
 1. Program Y offset from the CPU fence to the Display Buffer base in DPFC_CPU_FENCE_OFFSET.
 2. Program CPU fence ID and enable CPU fence tracking in DPFC_CONTROL_SA.
 3. Disable CPU Fence tracking in DPFC_CONTROL_SA if no tiling fence is mapped to display buffer.



Display Plane Enabling with FBC

- FBC has to be enabled after the attached display plane is enabled.
- This is the general sequence. See FBC_CTL for any workarounds.
 1. Enable display plane
 2. Enable FBC as described in FBC_CTL register

Display Plane Disabling with FBC

- FBC has to be disabled at least one frame before FBC attached display plane disabling.
- This is the general sequence. See FBC_CTL for any workarounds.
 1. Disable FBC as described in FBC_CTL register.
 2. Wait for at least one start of vblank for the disable double-buffering.
 3. Disable display plane.

Watermarks

The watermark registers are used to control the display to memory request timing. The watermarks must be programmed according to the Display Watermark Programming section.

Plane and cursor watermark registers are in the planes section.

WM_MISC

WM_LINETIME

DE_POWER1

DE_POWER2

DC States

DC_STATE_EN

Sequences for Power Wells

Registers

PWR_WELL_CTL

PWR_WELL_CTL_AUX

PWR_WELL_CTL_DDI

FUSE_STATUS

Functions Within Each Well

Except where noted, the registers for a function reside within the same power well as that function.

Note: Some views of the PRM will show a power domain within each register definition. Those domains may be incorrect.

- PG0 contains the functions for the graphics PCI device and bringing up the display
 - PCI
 - Clocks
 - The port PLLs are in the PLL and IO/PHY/AFE power domains and require power enabling which is explained in the mode set sequences
 - Shared Functions
 - Interrupts are in PG0, except for pipe interrupts which reside in the power wells associated with the pipes.
 - MBus, except for PIPE_MBUS_DBOX_CTL which reside in the power wells associated with the pipes.
 - Data Buffer (DBUF) registers are in PG0, but the function is in PG1.
 - Central Power, except for FBC which resides in PG1.
 - Top Level GTC. DDI Level GTC is in the power well associated with each DDI.
- PG1 contains the functions for pipe A display on eDP or MIPI DSI, without VDSC.
 - Data Buffer (DBUF) function is in PG1, but the registers are in PG0.
 - Pipe A and associated Planes, except for VGA.
 - Transcoder A, including PSR
 - Transcoder DSI
 - DDI A-C, including Aux. See note below about IOs.
 - FBC
 - DSS
- PG2 contains the functions for VDSC on pipe A
 - VDSC/joining for pipe A.
- PG3 the functions for pipe B, external displays, and VGA.
 - Pipe B and associated Planes and VDSC/joining
 - Audio
 - Transcoder WD
 - VGA. The VGA_CONTROL register is in PG0, but requires PG3 to be enabled before VGA is enabled.
 - Transcoder B
 - DDI USBC1-6, including Aux. See note below about IOs.
 - KVMR. Hardware will automatically enable PG2 and PG3 for KVMR.

- PG4 contains pipe C.
 - Pipe C and associated Planes and VDSC/joining
 - Transcoder C (registers reside in PG3, but access path goes through associated pipe)
- PG5 contains pipe D.
 - Pipe D and associated Planes and VDSC/joining
 - Transcoder D (registers reside in PG3, but access path goes through associated pipe)
- The port PLLs and IOs and Aux IOs are in the IO/PHY/AFE power domains and require power enabling which is explained in the mode set sequences.
 - For typeC ports, there are separate Aux channel IO power domains for thunderbolt (TBT fields) and for non-thunderbolt (USBC fields).

Enable Sequence

PG0 -> PG1 -> PG2 -> PG3 -> PG4 -> PG5

Power wells must be enabled sequentially and no intermediate power well can be skipped.

1. **PG0 is controlled by the CPU power controller and will be automatically enabled any time software accesses display**
2. **PG1 is enabled using** Sequences to Initialize Display
3. **To turn on VDSC with pipeA**
 - a. PG1 must be turned on first (display initialization sequence will turn PG1 on)
 - b. Enable PWR_WELL_CTL Power Well 2 Request
 - c. Wait for PWR_WELL_CTL Power Well 2 State == Enabled, timeout after 20 us
 - d. Wait for FUSE_STATUS FUSE PG2 Distribution Status == Done, timeout after 20 us
4. **To turn on PipeB, PG1 and PG2 must be turned on and then do the following to turn on PG3**
 - a. Enable PWR_WELL_CTL Power Well 3 Request
 - b. Wait for PWR_WELL_CTL Power Well 3 State == Enabled, timeout after 20 us
 - c. Wait for FUSE_STATUS FUSE PG3 Distribution Status == Done, timeout after 20 us
5. **To turn on PipeC, PG1, PG2, and PG3 must be turned on and then do the following to turn on PG4**
 - a. Enable PWR_WELL_CTL Power Well 4 Request
 - b. Wait for PWR_WELL_CTL Power Well 4 State == Enabled, timeout after 20 us
 - c. Wait for FUSE_STATUS FUSE PG4 Distribution Status == Done, timeout after 20 us
6. **To turn on PipeD, PG1, PG2, PG3 and PG4 must be turned on and then do the following to turn on PG5**
 - a. Enable PWR_WELL_CTL Power Well 5 Request
 - b. Wait for PWR_WELL_CTL Power Well 5 State == Enabled, timeout after 20 us
 - c. Wait for FUSE_STATUS FUSE PG5 Distribution Status == Done, timeout after 20 us

Disable Sequence

Power wells have to be turned off in the reverse order of the enabling. The following sequence should be followed to disable each power well

1. Disable PWR_WELL_CTL Power Well # Request
2. Wait for PWR_WELL_CTL Power Well # State == disabled, timeout after 20 us

Pipe

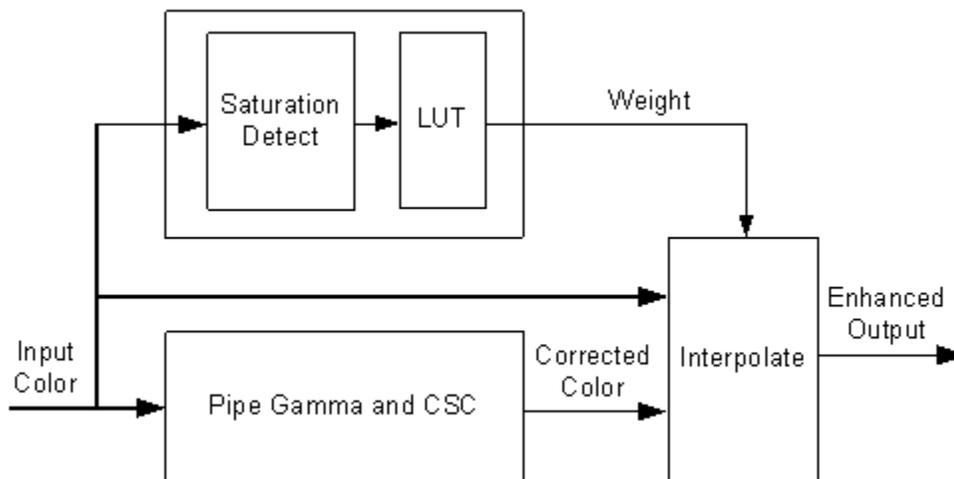
Pipe Color Gamut Enhancement

Pipe color gamut enhancement is used to enhance display of standard gamut content on wide gamut displays. It processes the color value from before and after the pipe gamma and color space correction blocks to create the color gamut enhanced output. The typical usage is to output the pipe gamma and CSC corrected color for areas of low saturated content and the input (not gamma or CSC corrected) color for areas of high saturated content. It is not recommended to use color gamut enhancement with wide gamut inputs.

CGE_CTRL

CGE_WEIGHT

The pipe Gamma and CSC must be programmed to either the split gamma mode or gamma after CSC mode when using pipe color gamut enhancement.



The saturation level of the pipe gamma and CSC input color is detected and used to index into a look up table (LUT) containing programmable weights. The saturation values are linearly distributed across the LUT indexes from the lowest index for lowest saturation to the highest index for highest saturation.

The enhanced output color is created by using the weight value to interpolate between the input color and corrected color. See the following table of weights to amount of input or corrected color used to create the enhanced output color.



Weighting of input and corrected colors

Weight from LUT	Amount of Input Color in Enhanced Output	Amount of Corrected Color in Enhanced Output
00 0000b (minimum)	0%	100%
...
00 1000b	25%	75%
...
01 0000b	50%	50%
...
01 1000b	75%	25%
...
10 0000b (maximum)	100%	0%

Example weight programming

CGE LUT Index	CGE Weight Value Decimal	CGE Weight Value Binary	CGE Weight Percent Input Color	CGE Weight Percent Corrected Color
0 (lowest saturation)	0	00 0000b	0%	100%
1	0	00 0000b	0%	100%
2	0	00 0000b	0%	100%
3	0	00 0000b	0%	100%
4	0	00 0000b	0%	100%
5	0	00 0000b	0%	100%
6	1.6	00 0010b	5%	95%
7	3.2	00 0011b	10%	90%
8	4.8	00 0101b	15%	85%
9	6.4	00 0110b	20%	80%
10	8.64	00 1001b	27%	73%
11	12.8	00 1101b	40%	60%
12	19.2	01 0011b	60%	40%
13	25.6	01 1010b	80%	20%
14	28.8	01 1101b	90%	10%
15	32	10 0000b	100%	0%
16 (highest saturation)	32	10 0000b	100%	0%

HDR

High Dynamic Range (HDR)

Key HDR features

- HDR mode supports up to 3 planes in each pipe.
- Tone mapping support in planes.
- Linear blending of HDR planes.
- Linear scaling support in planes and pipes.
- Dedicated chroma upsampler to handle YUV420.
- Programmable color space convertors in planes/cursor.
- Enhanced gamma mode for PQ encoding.

Hardware Capabilities

FP16 Normalizer:

FP16 Normalizer normalize the pixels to -1.0 to 1.0 range. FP16 Normalizer is programmed in the PLANE_PIXEL_NORMALIZE register. The programmed FP16 value gets multiplied with the pixel value for normalizing them. Out of band values get clamped to -1.0 to 1.0 value. The output of this block directly feeds into plane CSC block bypassing Chroma up-sampler, input CSC and de-gamma. The FP16 source content must be linear with no gamma encoding.

Chroma up-sampler:

HDR planes have a dedicated bi-linear Chroma up-sampler for converting P0xx/NV12 source pixel formats to YUV444. Supports up to 4k resolution. Chroma up-sampler is programmed in the PLANE_CUS_CTL register.

CUS supports 4 different chroma siting positions that can be programmed through the initial phase. For initial phase programming, refer PLANE_CUS_CTL.

Input CSC:

A programmable 3x3 color space converter generally used for converting YUV content to RGB. This operates in non-linear space. Plane input CSC is programmed in the PLANE_INPUT_CSC_* registers along with the enable bit in the PLANE_COLOR_CTL register.

Plane pre-CSC Gamma:

Plane de-gamma LUT is used for linearizing HDR or SDR source content with gamma. The 128 entries LUT is uniformly spaced with additional 3 entries for 1.0,3.0 and 7.0. Plane input CSC is programmed in the PLANE_PRE_CSC_GAMC_INDEX_ENH and PLANE_PRE_CSC_GAMC_DATA_ENH registers along with the enable bit in the PLANE_COLOR_CTL register.

Plane CSC:

The programmable plane CSC can be used for color space conversion of source content to BT.2020 or panel color gamut. Plane CSC is programmed in the PLANE_CSC_* registers along with the register bit in the PLANE_COLOR_CTL register.

Plane post-CSC Gamma:

In HDR mode, this block can be used for tone mapping/dynamic range adjustment of each source content to a common reference luminance range before blending. The LUT must be programmed to use "Multiply mode" in the PLANE_POST_CSC_GAMC_INDEX_ENH, PLANE_POST_CSC_GAMC_DATA_ENH and PLANE_COLOR_CTL registers.

Hardware computes the pseudo luminance of the incoming pixel using the following equation, uses it to index the LUT and compute an adjustment factor 'F'. Toned mapped output R, G and B values are computed by scaling the input R, G and B channel by 'F'.

$$L_{in} = 0.25 * \text{Red input} + 0.625 * \text{Green input} + 0.125 * \text{Blue input}.$$

Plane scaling:

Plane scaling supports both linear and non-linear scaling modes. The scaling mode is programmed in the PS_CTRL. In HDR mode, scaling and blending operations are generally performed in linear mode.

Linear Blending:

With precision improvements, hardware supports blending in linear mode. Linear blending is enabled by programming the HDR mode bit in the PIPE_MISC register.

Cursor:

For HDR usages, the cursor plane supports a programmable De-gamma LUT, Color Space Converter and a Luminance scaler. Luminance scaler scales each color component by a programmed 10 bit value fractional value.

Cursor de-gamma LUT is programmed in CUR_PRE_CSC_GAMMA_INDEX and CUR_PRE_CSC_GAMMA_DATA registers along with the enable bit in CUR_CTL. Cursor CSC is programmed in the CUR_CSC_COEFF register along with the enable bit in CUR_CTL. Luminance scaling is enabled and programmed in CUR_COLOR_CTL.

Pipe Scaler:

Pipe scaler supports linear scaling.

Pipe Gamma:

Supports a 12 bit multi-segmented gamma mode that provides high quality HDR PQ encoding. Refer to the "Pipe Palette and Gamma" page for details

Pipe LDPST

DPLC_CTL

DPLC_HIST_INDEX

DPLC_HIST_DATA

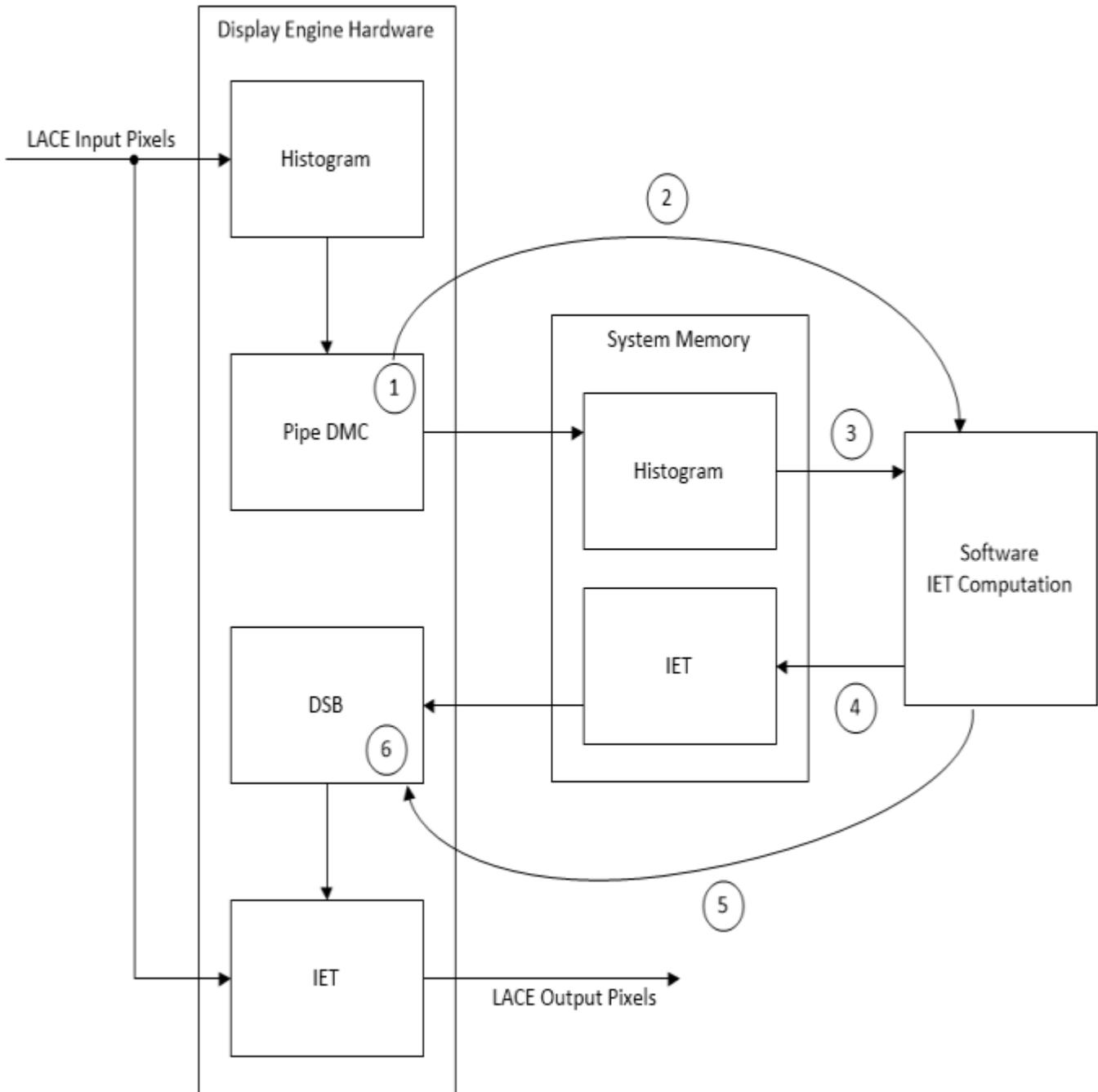
DPLC_IE_INDEX

DPLC_IE_DATA

Fast LACE Access

The fast LACE access improves the latency for programming LACE by moving histogram and image enhancement table (IET) programming to system memory instead of display engine MMIO. This also allows software to write updates only to IET tiles that have changes, instead of writing them to all IET tiles when there is any change.

1. DMC reads LACE histogram data and writes it into a system memory location indicated by display driver.
2. DMC triggers interrupt to driver.
3. Display driver reads histogram data and computes IET.
4. Display driver writes IET into system memory in DSB data format. If the same memory surface is used for multiple frames, driver can skip writing any tiles that do not need update since memory will hold the old values of those tiles.
5. Driver programs DSB to start loading IET.
6. DSB reads the IET data from system memory and writes into LACE IET.



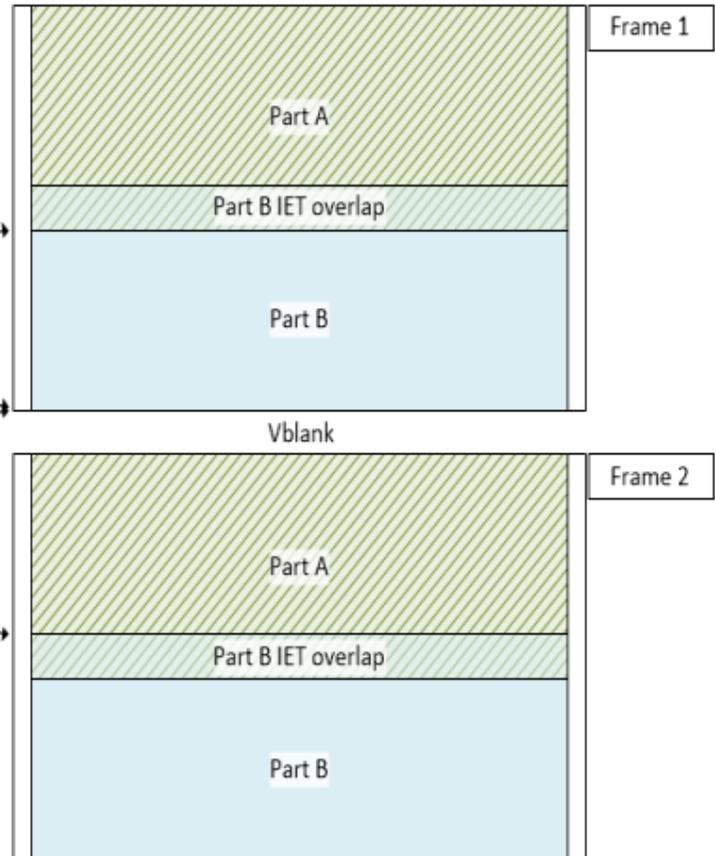
To reduce the latency between histogram computation and IET application to 1 frame, each active frame is broken roughly in half, into part A and part B. While hardware computes the histogram of one part, software works on computing IET for the other part. The size of part A and B is programmable in number of LACE tile rows. The part B IET overlaps part A by one tile row in order to provide a smooth transition across parts.

Part A processing:

1. Hardware completes Part A histogram
2. Hardware stores Part A histogram to memory
3. Hardware generates interrupt
4. Software calculates Part A IET, minus overlap, and stores to memory
5. Software programs DSB to load part A IET
6. Hardware loads Part A IET from memory

Part B processing:

1. Hardware completes Part B histogram
2. Hardware stores Part B histogram to memory
3. Hardware generates interrupt
4. Software calculates Part B IET, plus overlap, and stores to memory
5. Software programs DSB to load part B IET
6. Hardware loads Part B IET from memory



Registers

DPLC_PART_CTL

DPLC_PTRCFG

DPLC_PTRCFG_LIVE

DPLC_RDLENGTH

DPLC_RDLENGTH_LIVE

DPLC_FA_STATUS

DPLC_FA_IMR

DPLC_FA_IIR

DPLC_FA_SURF

DPLC_FA_SURF_LIVE

Part Start and End Programming

Tile Size = 256

Tile Columns = $\text{CEILING}(\text{Pipe Horizontal Source Size} / \text{Tile Size})$

Part A Start Tile Row = 1

Part A End Tile Row = $\text{FLOOR}(\text{Pipe Vertical Source Size} / 2 / \text{Tile Size}) + 1$



Part A Tile Rows = Part A End Tile Row

Part B Start Tile Row = Part A End Tile Row + 1

Part B End Tile Row = CEILING(Pipe Vertical Source Size / Tile Size)

Part B Tile Rows = 1 + Part B Start Tile Row - Part B End Tile Row

The row number programming is zero based, so subtract 1 from these values when programming PART CTL. Resolutions with less than 3 tile rows in part A or part B are not supported.

Example

5120x3200 pipe source size

Tile Size = 256

Tile Columns = 20

Part A Start Tile Row = 1 (program 0)

Part A End Tile Row = 7 (program 6)

Part A Tile Rows = 7

Part B Start Tile Row = 8 (program 7)

Part B End Tile Row = 13 (program 12)

Part B Tile Rows = 6

Pointer Config Programming

The pointer config and read length are used by hardware to read out the histogram at the end of each part.

Part A X, Y, and DW Indexes = 0.

Part B X and DW Indexes = 0. Part B Y Index = Part B Start Tile Row.

Part A read length = Part A Tile Rows * Tile Columns * DWords per tile

Part B read length = Part B Tile Rows * Tile Columns * DWords per tile

The index values are programmed in the pointer config register, using the same format as DPLC_HIST_INDEX.

DSB Programming

DPLC_IE_INDEX

Part A X, Y, and DW Indexes = 0.

Part B X and DW Indexes = 0. Part B Y Index = Part B Start Tile Row - 1.

DPLC_IE_DATA

Part A number of IE data DWords written by DSB = (Part A Tile Rows - 1) * Tile Columns * DWords per tile

Part B number of IE data DWords written by DSB = (Part B Tile Rows + 1) * Tile Columns * DWords per tile

Overall Sequence

1. Driver setup LACE with fast access

- a. Load DMC Fast LACE access program.
- b. Program DPLC_FA_SURF_PARTA with the memory base address for part A histogram.
- c. Program DPLC_FA_SURF_PARTB with the memory base address for part B histogram.
- d. Write DPLC_CTL to enable LACE function, Fast Access Enable, select enhancement mode. IE Enable is OFF.
- e. Write DPLC_PART_CTL, DPLC_RDLENGTH_LIVE, and DPLC_PTRCFG with partA/PartB programming.
- f. Write DPLC_FA_IMR to unmask histogram copy done interrupts. Other interrupts are optional for error handling.
- g. Configure pipe interrupts to unmask and enable LACE interrupt.
- h. Write DPLC_CTL to enable correction - IE Enable.

2. DMC execution triggered at the end of part A

- a. Read DPLC_RDLENGTH_LIVE_PARTA to obtain number of reads to issue.
- b. Move DPLC_PTRCFG_LIVE_PARTA to DPLC_HIST_INDEX to start histogram reads for part A.
- c. Read DPLC_HIST_DATA as many times as programmed in DPLC_RDLENGTH_LIVE_PARTA.
- d. Write as many cachelines to system memory for part A to the base address specified in DPLC_FA_SURF_PARTA.
- e. Flush writes with a read to the last write location, ignoring the read return value.
- f. Write DPLC_FA_STATUS to clear Part A Histogram Ready and set Part A Histogram Copy Done.

3. Driver

- a. Receive Part A Histogram Copy Done interrupt.
- b. Read histogram part A from system memory at the base address specified in DPLC_FA_SURF_PARTA.
- c. Calculate PART A IET.
- d. Configure DSB with IET part A programming.
- e. Poll for DSB Idle.
- f. Load DSB HTP.

4. DSB

- a. Write DPLC_IE_INDEX to start IET data load.
- b. Write DPLC_IE_DATA entries.
- c. Write DPLC_FA_STATUS to set Part A Load IE.

5. DMC execution triggered at the end of part B

- a. Read DPLC_RDLENGTH_LIVE_PARTB to obtain number of reads to issue.
- b. Move DPLC_PTRCFG_LIVE_PARTB to DPLC_HIST_INDEX to start histogram reads for part B.
- c. Read DPLC_HIST_DATA as many times as programmed in DPLC_RDLENGTH_LIVE_PARTB.
- d. Write as many cachelines to system memory for part B to the base address specified in DPLC_FA_SURF_PARTB.
- e. Flush writes with a read to the last write location, ignoring the read return value.
- f. Write DPLC_FA_STATUS to clear Part B Histogram Ready and set Part B Histogram Copy Done.



6. Driver

- a. Receive Part B Histogram Copy Done interrupt.
- b. Read histogram part B from system memory at the base address specified in DPLC_FA_SURF_PARTB.
- c. Calculate PART B IET.
- d. Configure DSB with IET part B programming.
- e. Poll for DSB Idle.
- f. Load DSB HTP.

7. DSB

- a. Write DPLC_IE_INDEX to start IET data load.
- b. Write DPLC_IE_DATA entries.
- c. Write DPLC_FA_STATUS to set Part B Load IE.

8. Go to 2

Fast Access Status and Interrupts

Histogram Ready

Status

Used by LACE to know when DMC has finished reading the histogram

Set by LACE

Clear by DMC after reading histogram

IIR

No current usage.

Set by LACE when FA status bit sets

Clear by driver on servicing interrupt

Load IE

Status

Used by LACE to know when to load the IE.

Set by DSB after copying IE into LACE

Clear by LACE after load complete

IIR

No current usage.

Set by LACE when FA status bit sets

Clear by driver on servicing interrupt

Histogram Copy Done

Status

Used by DMC to pass the copy done to the interrupt

Set by DMC after copying histogram to memory
 Cleared immediately by LACE

IIR

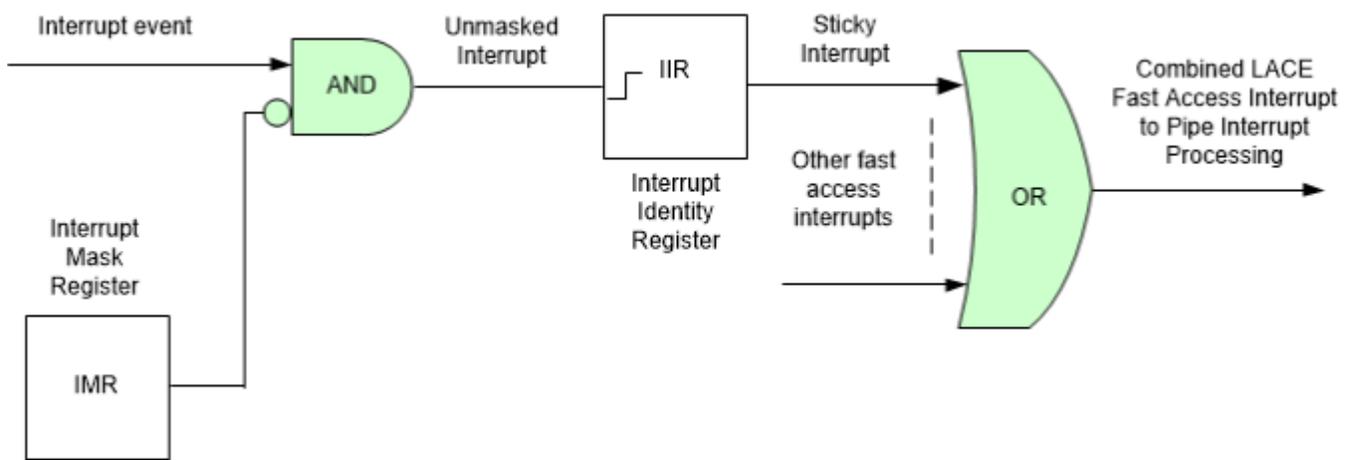
Used by driver to know when histogram is ready in memory and which part to work on
 Set by LACE when FA status bit sets
 Clear by driver on servicing interrupt

Histogram/IET Late/Overlap

IIR

Used by driver to detect error cases
 Set by LACE when error detected
 Clear by driver on servicing interrupt

Interrupt Structure



DSB Engine Programming

Display State Buffer Programming

A DSB (Display State Buffer) is a queue of MMIO instructions in the memory which can be offloaded to DSB HW in Display Controller. DSB HW is a DMA engine that can be programmed to download the DSB from memory. It allows driver to batch submit display HW programming. This helps to reduce loading time and CPU activity, thereby making the context switch faster.

It can be chiefly used as an extension of the current flip programming

There are three DSB DMA engines per pipe.

Different DSB's are required if different tasks need to be accomplished simultaneously

DSB's can access only the pipe, plane, and transcoder Data Island Packet registers.

DSB cannot access any registers outside pipe registers, including, but not limited to FBC, clocks, audio, Aux channel, Power Wells, HPD.

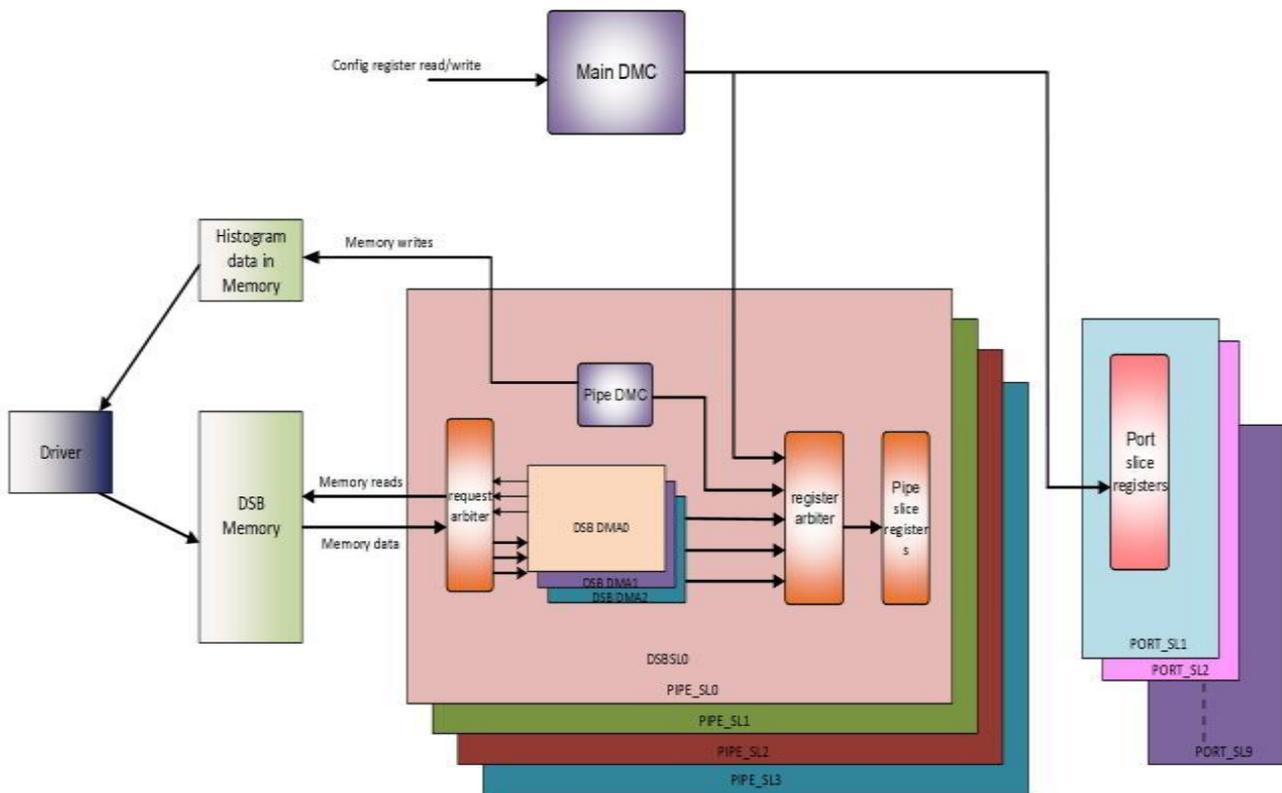
DSB HW can support only register writes (both indexed and direct MMIO writes). There are no registers reads possible with DSB HW engine.

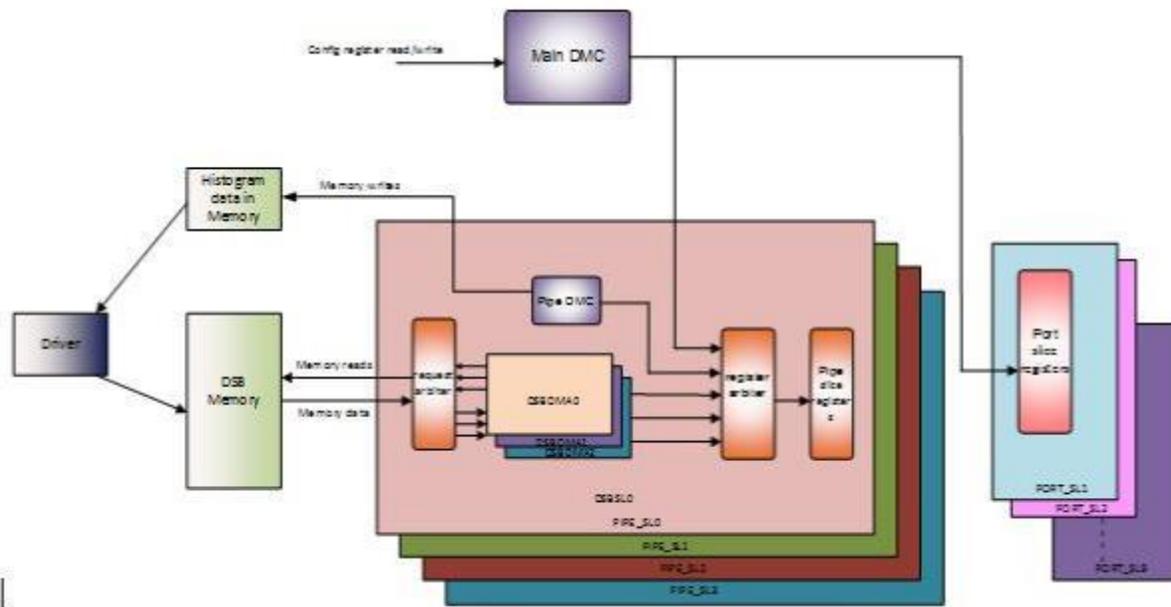
DSB HW has poll function capability.

DSB HW can be used in FAST LACE programming.

DSB HW can be used in HDR Meta data programming.

Below is the block diagram of the DSB HW and shows the general flow of the HW and SW flow. Driver will upload the Display State Buffer with instructions to program the Pipe registers.





Following registers will be used in the DSB HW. Software must ensure correct programming of these registers for the proper function of the HW. Illegal/incorrect programming of these register may result in unexpected behavior of hardware.

DSB HW Registers

DSB_CTRL

DSB_BUFRPT_CNT

DSB_HEAD_PTR

DSB_TAIL_PTR

DSB_MMIOCTRL

DSB_POLLMASK

DSB_POLLFUNC

DSB_INTERRUPT

DSB_PF_LN_LOWER

DSB_PF_LN_UPPER

DSB_PMCTRL

DSB_PMCTRL_2

DSB_CURRENT_HEAD_PTR

DSB_RM_TIMEOUT

DSB_RMTIMEOUTREG_CAPTURE

DSB_INTERRUPT

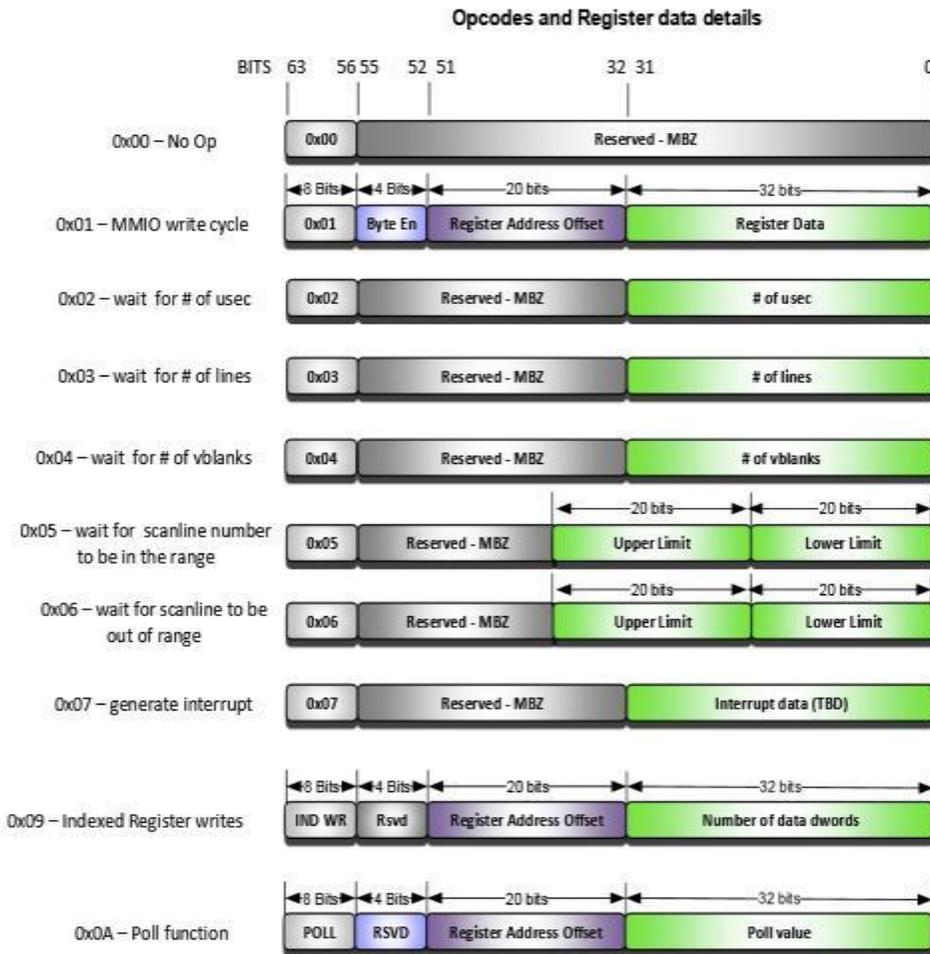
DSB_STATUS

PIPEDMC_CONTROL

PIPE_DMCSANLINECOMP

DSB Instructions

The following DSB instructions are supported by the DSB HW. Software must upload the instructions in the defined format shown below.



Below are some examples of the DSB instructions format in the memory.

20000000	10000000	09F7227c	00000003	00000008	00000007	00000006	00000005	00000004	00000003	00000002	00000001	09F7227c	00000008	0xh03000000
01F72250	8003801d	01F7224c	80020011	01F72248	80010009	01F72244	80004003	01F72240	80004003	01F721c8	00000000	00000000	30000000	0xh03000040
01F721cc	00002000	01F7219c	91440000	01F721e4	00000000	01F72268	00004007	01F7225c	8005c02f	01F72258	80050029	01F72254	80044023	0xh03000080
07000000	FFFFFFFF	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	01F72180	84000408	0xh03000100



General Programming Sequence of the DSB HW

1. Driver will form a display state buffer using the supported DSB instructions to program the display registers and upload it in the memory.
2. Driver must program the **DSB_CTRL** register to enable the DSB engine. SW may choose to start the DSB DMA engine to start the memory requests right away or wait for a vblank to trigger the DSB DMA engine. SW may also chose to repeat to play the the same buffer again on every vblank by setting re-iterate bit of the **DSB_CTRL** register. SW must follow the programming rules defined in the **DSB_CTRL** register bits.
3. Driver cannot update or modify the DSB registers within the DSB program.
4. Driver must program the **DSB_HEAD_PTR** with the first cacheline address of the program and **DSB_TAIL_PTR** with the last cacheline address of the DSB program plus one. DSB HW DMA engine will be triggered on the tail pointer update.
 - Head and Tail Pointers (32 bits each).
 - SW must make sure that there must atleast one cacheline in the DSB program when updating Head and Tail pointer. i.e. head and tail pointer cannot be equal.
 - Cacheline aligned graphics address.
 - TP points to last location+1 cacheline address.
 - SW should not modify these DSB registers when DSB is busy.
 - Each cacheline can have 8 instructions.
 - 64 bits per instruction.
 - Fill incomplete cachelines with NoOps
 - Cacheline data is in little Endian.
 - One DSB can write to the registers of the another DSB engine within the pipe.
5. Driver can poll on the DSB Status bit in the **DSB_CTRL** register to check if the DSB engine has completed. If DSB program has an interrupt instruction as the last instruction then an interrupt will generated to SW which can be used to check the status the DSB DMA engine.

- SW can enable the interrupts by programming the appropriate enabled in **DSB_INTERRUPT** register.
- A pipe interrupt at **DE_PIPE_INTERRUPT** will be generated when interrupts are correctly enabled in the IMR and IER registers. Bits 13,14,15 shows the status of interrupts from each DSB engine.
- SW must clear the sticky bits in the **DSB_INTERRUPT** first and then clear the ISR bits in the **DE_PIPE_INTERRUPT**

6. Each of the instructions are described below in detail. Each of the supported functions/capabilities are also described that SW can use to program DSB HW. Every instruction must start at a 64 bit boundary.

NOOPs

The opcode of this instruction is 0x00. All the instructions must start and end on the cacheline boundaries. NoOPs can be used to complete the cachelines. All the 64 bits of this instruction are 0's.

MMIO Writes

The opcode of this instruction is 0x01. Lower 32 bits of the instruction is the MMIO write data. Bits 51:32 is the offset address of the MMIO write. Bits 55:52 are the byte enables of the mmio write instruction.

Wait for number of Microseconds

The opcode of this instruction is 0x02. The lower 32 bits of the instructions will have the number of microseconds to wait. DSB HW will wait for the programmed number of micro seconds before processing the next instruction.

Wait for number of lines

The opcode for this instruction is 0x03. The lower 32 bits of the instruction has the number of lines to wait. DSB HW will wait for the programmed number of lines before processing the next instruction.

Wait for number of vblanks

The opcode for this instruction is 0x04. The lower 32 bits of the instruction has the number of vblanks to wait. DSB HW will wait for the programmed number of vblanks (pipe undelayed vblank) before processing the next instruction.

Wait for scanline number in range

The opcode for this instruction is 0x05. The lower 40 bits of the instruction has the upper and lower scanline numbers. DSB HW will wait for the scanline number to be within the programmed range of the line numbers before processing the next instruction.

Wait for scanline number out of range

The opcode for this instruction is 0x06. The lower 40 bits of the instruction has the upper and lower scanline numbers. DSB HW will wait for the scanline number to be out of the programmed range of the line numbers before processing the next instruction.

Generate Interrupt

The opcode for this instruction is 0x07. DSB HW, after decoding this instruction, will generate a interrupt when interrupt is enabled in **DSB_INTERRUPT** register and the IMR and IER register bits are correctly programmed in **DE_PIPE_INTERRUPT**.

Indexed MMIO register writes

The opcode for this instruction is 0x09. The lower 32 bits of this instruction will have the number of the data dwords of the indexed MMIO writes. Bits 51:32 of this instruction will have the address offset.

- The data dwords are followed by this instructions.
- If the number of the data dwords is odd then the last dword of the instruction will be 0's to complete the 64 bit instruction. Example is shown below



Example of Indexed Register Writes with odd number of dwords

Poll Function

The opcode for this instruction is 0x0A. Lower 32 bits of the instruction will have the poll value to be compared and bits 51:32 have the offset of the register that needs to be polled on.

- SW must program the **DSB_POLLMASK** before this instruction is used. Poll mask will have the enable on each of the 32 bits that the data will be compared against.
- SW must also program the **DSB_POLLFUNC** register prior to using the poll function instruction. Poll function has to be enabled before using the poll function.
 - The default value of number of micro seconds to wait before a poll retry is set to 2us.
 - The default number of retries is set to 50 times before a timeout or poll fails
- SW can chose to add these two registers as part of DSB program itself before using the POLL instruction. But SW must add 5 NOOPs after MMIO writes instructions to these (**DSB_POLLMASK** and **DSB_POLLFUNC**) registers to allow some time for DSB engine to process the register writes to itself.



Special programming

The following transcoder registers need to special programming if they are part of the DSB program in the buffer. SW must follow the sequence listed below in the DSB program before adding programming of these registers.

1. Set the non-posted bit 8 to 1 in the **DSB_CTRL** register as part of the DSB program.
2. Add four No-Op instructions.
3. Add the transcoder register programming.
4. Set the non-posted bit 8 to 0 in the **DSB_CTRL** register.
5. Add four No-Op instructions
6. Add the rest of the programming (if any).

The list of these special registers is below.

DDI_BUF_CTL
TRANS_DDI_FUNC_CTL2
TRANS_DDI_FUNC_CTL
VIDEO_DIP_CTL
VIDEO_DIP_PPS_DATA_*
VIDEO_DIP_ADAPTIVE_SYNC_DATA_*
TRANS_CONF
TRANS_STEREO3D_CTL
TRANS_DPT_PAT
TRANS_SET_CONTEXT_LATENCY
TRANS_VRR_CTL
TRANS_VRR_FLIPLINE
TRANS_VRR_VMAXSHIFT
TRANS_VRR_VMAX
TRANS_VRR_VMIN
TRANS_VRR_VSYNC
TRANS_LINKN1
SRD_CTL
SRD_PERF_CNT
PSR_MASK
DP_COMP_CTL
DP_TP_CTL
PSR2_CTL

DSB Atomic Usage

The following is the framework for using DSB to program pipe and plane registers to atomically update (all update together for the same frame). DSB can also be used to update registers non-atomically or to update single resources that are self-atomic, with simpler sequences that don't require any wait for vblank.

For this atomic usage, transcoder timings must be programmed to create a window between start of undelayed vblank and delayed vblank. The typical requirement is a 100 microsecond window to provide enough time for DSB to program all the pipe and plane registers. Smaller values may be used for testing. Larger values may exceed required time for filling the display pipeline during vblank when using reduced blanking. $\text{ROUNDUP}(\text{desired window time}/\text{line time}) = \text{number of lines to program delayed vblank larger than undelayed vblank}$.

Fixed Refresh Rate

This requires VRR to be disabled.

Before enabling DSB, set bit 23=1b in the following register for the pipe and DSB to skip waits when PSR is entered.

Pipe A: DSB0 0x70BF0, DSB1 0x70CF0, DSB2 0x70DF0

Pipe B: DSB0 0x71BF0, DSB1 0x71CF0, DSB2 0x71DF0

Pipe C: DSB0 0x72BF0, DSB1 0x72CF0, DSB2 0x72DF0

Pipe E: DSB0 0x73BF0, DSB1 0x73CF0, DSB2 0x73DF0

DSB will need to align instruction execution to the start of vblank. Either enable DSB_CTRL Wait for VBLANK or use the wait for vblank instruction.

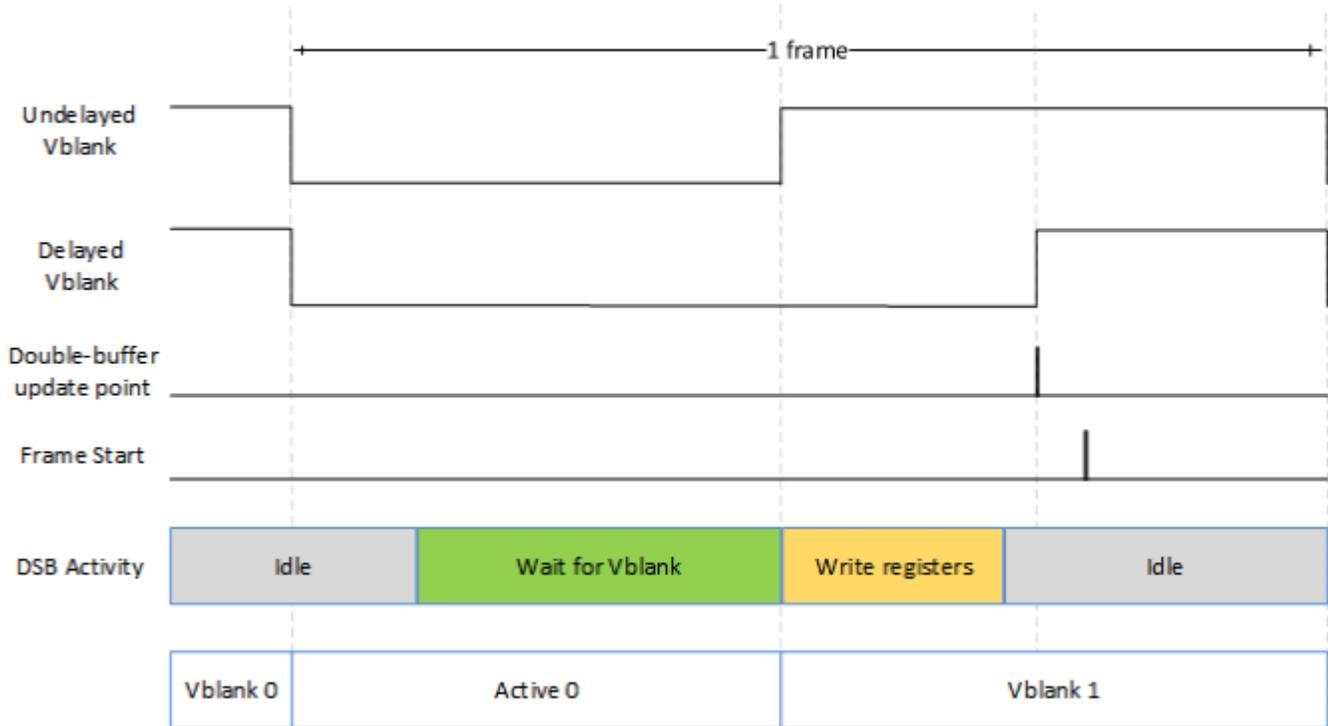
Follow the General Programming Sequence of the DSB HW to setup and initiate DSB.

Use the following DSB instructions

1. If DSB_CTRL Wait for VBLANK not enabled, Wait for 1 vblank.
2. Write 1 to DSB_STATUS bit 16 to clear any previous indication that DSB was busy during delayed vblank.
3. Write pipe and plane registers to be updated for the next frame.
 - Depending on the registers to write, this can include register writes and indexed register writes.
 - Requirements for writing some registers in certain orders to arm them for double-buffer updates are still required.
4. Optional: Generate Interrupt

After DSB completes, found through interrupt or polling on DSB status, read DSB_STATUS bit 16 to find if delayed vblank started before DSB finished. If it did, then there was an error and the pipe and plane programming may not be complete. Software may log the error and attempt recovery.

Example: Fixed Refresh Rate



Variable Refresh Rate

This requires VRR with push bit mode.

Before enabling VRR, set bit 31 in the following register for the transcoder attached to this pipe to enable VRR safe window generation.

Transcoder A: 0x420C0

Transcoder B: 0x420C4

Transcoder C: 0x420C8

Transcoder D: 0x420D8

Before enabling DSB, set bits 23=1b, 15:14=11b, bits 7:6=11b in the following register for the pipe and DSB to enable VRR safe window to be used instead of vblank and to skip waits when PSR is entered.

Pipe A: DSB0 0x70BF0, DSB1 0x70CF0, DSB2 0x70DF0

Pipe B: DSB0 0x71BF0, DSB1 0x71CF0, DSB2 0x71DF0

Pipe C: DSB0 0x72BF0, DSB1 0x72CF0, DSB2 0x72DF0

Pipe E: DSB0 0x73BF0, DSB1 0x73CF0, DSB2 0x73DF0

DSB will need to align instruction execution to the VRR safe window. Either enable DSB_CTRL Wait for VBLANK during DSB setup or use the wait for vblank instruction.

Follow the General Programming Sequence of the DSB HW to setup and initiate DSB.

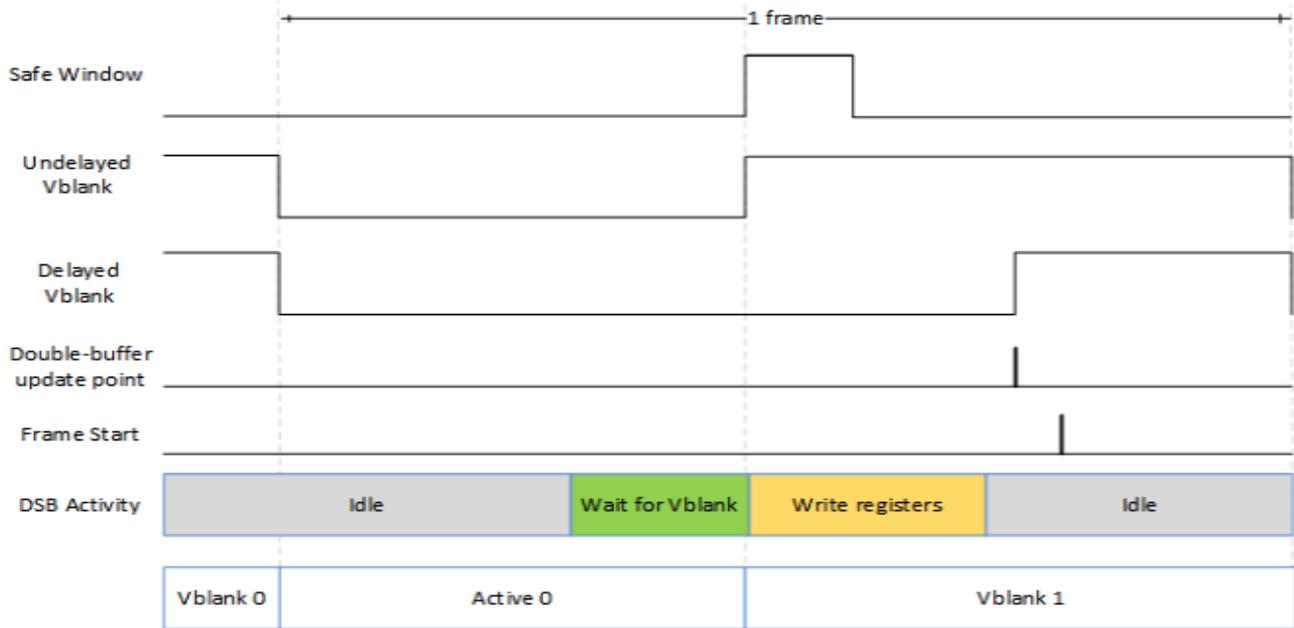
Use the following DSB instructions

1. If DSB_CTRL Wait for VBLANK not enabled, Wait for 1 vblank.
2. Write 1 to DSB_STATUS bit 16 to clear any previous indication that DSB was busy during delayed vblank.
3. Write 1 to transcoder push bit*. This step can be done here or in step 5. Initiating push at this point will allow the vblank to end sooner, but is different from the non-DSB programming flow that does push at the end.
4. Write pipe and plane registers to be updated for the next frame.
 - Depending on the registers to write, this can include register writes and indexed register writes.
 - Requirements for writing some registers in certain orders to arm them for double-buffer updates are still required.
5. Write 1 to transcoder push bit* if not done in earlier step.
6. Optional: Generate Interrupt

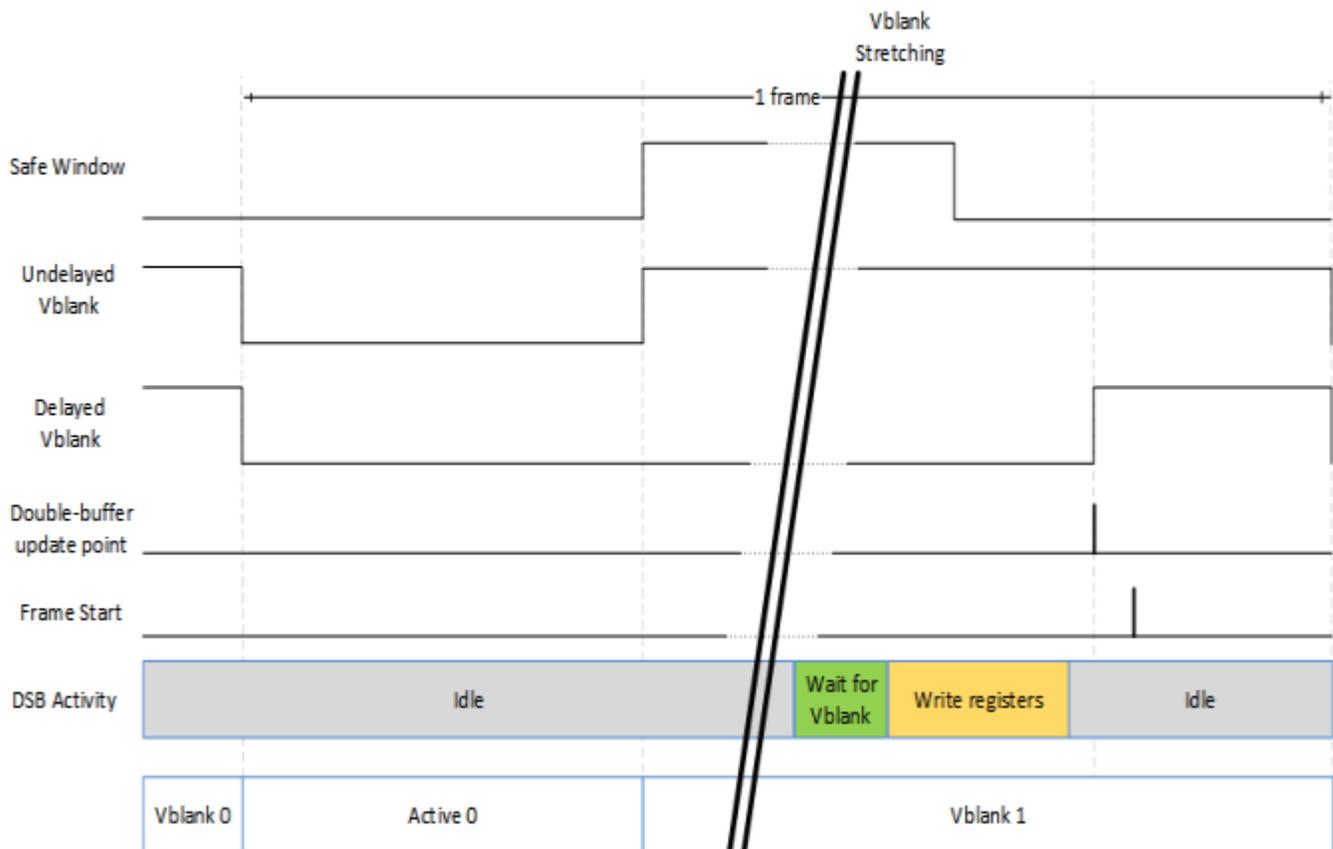
After DSB completes, found through interrupt or polling on DSB status, read DSB_STATUS bit 16 to find if delayed vblank started before DSB finished. If it did, then there was an error and the pipe and plane programming may not be complete. Software may log the error and attempt recovery.

*If driver knows that push is already guaranteed to be set outside of this DSB, such as by another DSB running in parallel, then this push can optionally be skipped. This requires very careful sequencing to ensure that this DSB does not get out of alignment with whatever is issuing the push.

Example: VRR – DSB Wait Starts Before Vblank



Example: VRR – DSB Wait Starts in Vblank



Adding Dynamic Metadata

If dynamic metadata will be sent just once, then it can be programmed together with the pipe and plane registers in the previous sequences.

Color Space Conversion

CSC_COEFF

CSC COEFFICIENT FORMAT

CSC_PREOFF

CSC_POSTOFF

CSC_MODE

OUTPUT_CSC_COEFF

OUTPUT_CSC_PREOFF

OUTPUT_CSC_POSTOFF

The high color channel is the most significant bits of the color. The low color channel is the least significant bits of the color. The medium color channel is the bits between high and low. For example: In



RGB modes Red is in the High channel, Green in Medium, and Blue in Low. In YUV modes, V is in the High channel, Y in Medium, and U in Low.

The color space conversion registers are double buffered and are updated on the start of vertical blank following a write to the CSC Mode register for the respective pipe.

The matrix equations are as follows:

$$\text{OutputHigh} = (\text{CoefficientRY} * \text{InputHigh}) + (\text{CoefficientGY} * \text{InputMedium}) + (\text{CoefficientBY} * \text{InputLow})$$

$$\text{OutputMedium} = (\text{CoefficientRU} * \text{InputHigh}) + (\text{CoefficientGU} * \text{InputMedium}) + (\text{CoefficientBU} * \text{InputLow})$$

$$\text{OutputLow} = (\text{CoefficientRV} * \text{InputHigh}) + (\text{CoefficientGV} * \text{InputMedium}) + (\text{CoefficientBV} * \text{InputLow})$$

Example programming for RGB to YUV is in the following table:

The input is RGB on high, medium, and low channels respectively and the desired YUV output is VYU on high, medium, and low channels respectively.

Program CSC_MODE to put gamma before CSC.

Program the CSC Post-Offsets to +1/2, +1/16, and +1/2 for high, medium, and low channels respectively.

The coefficients and pre and post offsets can be scaled if desired.

	Bt.601		Bt.709	
	Value	Program	Value	Program
RU	0.2990	0x1990	0.21260	0x2D98
GU	0.5870	0x0968	0.71520	0x0B70
BU	0.1140	0x3E98	0.07220	0x3940
RV	-0.1687	0xAAC8	-0.11460	0xBEA8
GV	-0.3313	0x9A98	-0.38540	0x9C58
BV	0.5000	0x0800	0.50000	0x0800
RY	0.5000	0x0800	0.50000	0x0800
GY	-0.4187	0x9D68	-0.45420	0x9E88
BY	-0.0813	0xBA68	-0.04580	0xB5E0

Example programming for YUV to RGB is in the following table:

The input is VYU on high, medium, and low channels respectively.

The output is RGB on high, medium, and low channels respectively.

Program CSC_MODE to put gamma after CSC.

Program the CSC Pre-Offsets to -1/2, -1/16, and -1/2 for high, medium, and low channels respectively.

The coefficients and pre and post offsets can be scaled if desired.

	Bt.601 Reverse		Bt.709 Reverse	
	Value	Program	Value	Program
GY	1.000	0x7800	1.000	0x7800
BY	0.000	0x0000	0.000	0x0000
RY	1.371	0x7AF8	1.574	0x7C98
GU	1.000	0x7800	1.000	0x7800
BU	-0.336	0x9AC0	-0.187	0xABF8
RU	-0.698	0x8B28	-0.468	0x9EF8
GV	1.000	0x7800	1.000	0x7800
BV	1.732	0x7DD8	1.855	0x7ED8
RV	0.000	0x0000	0.000	0x0000

Pipe 3D LUT

The 3D LUT is a pixel modification function which resides in the post blend color processing section of the display pipeline. It is used to apply non-linear transforms on each color component on a per pixel basis. Our LUT implementation uses a 17x17x17 three dimensional matrix of color points, with each point holding a 30 bit pixel value (10 bpc).

3D LUT functionality is supported only in pipe A and pipe B.

LUT_3D_CTL

LUT_3D_INDEX

LUT_3D_DATA

Programming

Enabling 3D LUT

Software should follow the sequence below.

1. Check if the "New LUT Ready" bit is clear. If set, software must wait till the bit is clear. LUT entries must not be change the LUT entries when the "New LUT Ready" is set.
2. Load the desired 3D LUT entries.
3. Set the "Enable" and the "New LUT Ready" bits.

The LUT buffer is double buffered. When 3D LUT functionality is enabled, the hardware observes the "New LUT Ready" bit on every vblank start. If the "New LUT Ready" bit is set, hardware loads the LUT entries in to its working RAM and clears the bit. The 3D LUT functionality works with programmed LUT values in the following frames until it gets disabled. When the "New LUT Ready" bit is clear, the software is allowed to modify the LUT entries.

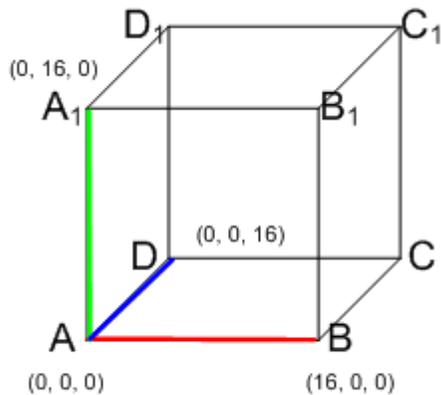
Disabling 3D LUT

1. Clear the "Enable" bit to '0'.

Programming 3D LUT Entries

The LUT array is accessed by an index/data register pair. The index register is read/writable and auto-increments after each read/write to the data register. Write '0' into the index register, followed by 4913 LUT entry writes to the data register.

Each LUT 3D entry is 30 bits and programmed as R10G10B10 (msb... lsb) value in the LUT_3D_DATA register. Since 10 bit values are used for all 17 points, the max value programmed is limited to 1023. A 1:1 mapping should use [0, 64, 128, 192, 256, 320, 384, 448, 512, 576, 640, 704, 768, 832, 896, 960, 1023].



The LUT entries should start at A and end at C₁ following the sequence specified below.

```

Iterate on Red axis from 0 - 16 {
  Iterate on Green axis 0 - 16 {
    Iterate on Blue axis 0 - 16 {
      program 3D LUT entry
    }
  }
}

```

Pipe DPST

Registers

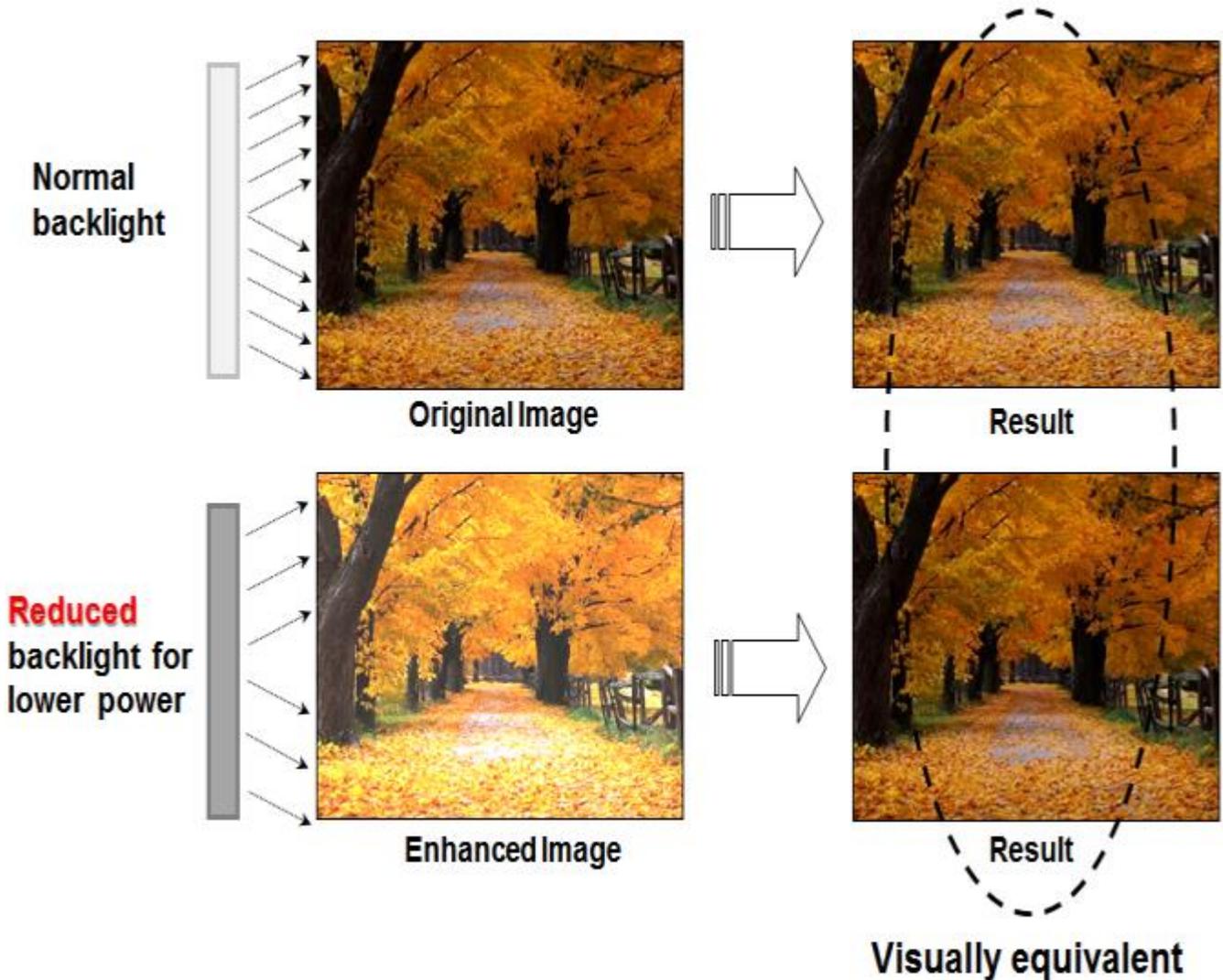
DPST_CTL

DPST_GUARD

DPST_BIN

Overview

Display Power Savings Technology (DPST) achieves significant platform average power savings by dynamically decreasing the display backlight brightness, while increasing the pixel values in the displayed image by a corresponding factor. The goal of DPST is to provide equivalent end-user-perceived image quality at a decreased backlight power level.



DPST generates statistics (histogram) for each image frame that is sent to the display. These statistics are used to determine if, and by how much, the backlight level can be reduced, saving backlight power. In order to maintain the same image brightness, the image pixel values are increased by an amount related to the backlight level reduction.

DPST is composed of three blocks.

Histogram Block

The hardware histogram block generates image statistics based on the pixel stream input. These statistics are used by the Processing block to determine how much the backlight level can be reduced.

The histogram block has the following requirements.

- Generate a histogram for each frame of display data
 - The histogram is generated based on the brightness of each pixel.
 - DPST_CTL Histogram Mode Select selects how the brightness is determined; either HSV $\max(\text{RGB}_{\text{in}})$, or the luma after converting RGB_{in} to YUV.
 - The histogram is composed of 32 bins with each bin covering a range of 8 values for 8 bit pixel component values. The first bin covers the values 0 thru 7.
 - If the pixel component values use more than 8 bits, the most significant 8 bits are used to form the histogram.
 - Each bin value has enough bits to count every pixel within an image.
 - Enabled by DPST_CTL IE Histogram Enable
- Two sets of histograms
 - The histogram currently being generated and a saved histogram from a previous frame.
 - The current histogram is moved to the saved histogram at the end of a frame in which a threshold event occurs.
 - Each bin of the saved histogram is readable by software.
 - Clear DPST_CTL Bin Register Function Select to TC
 - Wait for vertical blank for switch to TC mode, can skip if step 1 was done more than 1 vblank previously
 - Set DPST_CTL Bin Register Index to 0
 - Read DPST_BIN
 - If DPST_BIN Busy Bit is 1, go to step 3
 - Store DPST_BIN Data
 - Go to step 4 until all 32 bins are read
- Threshold register for comparing the histogram of the current frame to the histogram of the saved frame.
 - If any bin in the current histogram differs from the same bin in the saved histogram by more than the value in DPST_GUARD Threshold Guardband, then a threshold event is generated at the start of vertical blank.
 - DPST_GUARD Guardband Interrupt Delay specifies the number of consecutive frames the threshold must be exceeded before generating a threshold event.
 - This allows filtering out momentary variations from generating a threshold event.
 - DPST_GUARD Histogram Interrupt Enable enables the threshold event interrupt.
 - DPST_GUARD Histogram Event Status is a sticky bit that is set with the interrupt and must be cleared to receive more histogram events.

Enhancement Block

The hardware enhancement block adjusts the pixel values sent to the display, compensating for the brightness loss due to lowering of the display backlight level.

The enhancement block has the following requirements.

- Located after the histogram block
- Enabled by DPST_CTL IE Modification Table Enable
- Find the enhancement factor from a Look Up Table (LUT) with 33 entries
 - Switch (DPST_CTL Enhancement mode)
 - Case Direct: LUT address = most significant bits of RGB_{input}
 - Case Multiplicative: LUT address = most significant bits of HSV $\max(RGB_{input})$
 - Case Additive: LUT address = most significant bits of Y channel after converting RGB_{input} to YUV
 - DPST_CTL IE Table Value Format selects if the enhancement factor is a 1 integer and 9 fractional bits format, or a 2 integer and 8 fractional bits format.
 - The 2 integer and 8 fractional bits format allows for brightness increases nearly to 4x, but with reduced precision.
 - The final enhancement factor is derived by interpolating between the addressed LUT entry and the next entry, using the lower bits of the input.
 - Each entry of the LUT is programmable by software.
 - Set DPST_CTL Bin Register Function Select to IE
 - Wait for vertical blank for switch to IE mode, can skip if step 1 was done more than 1 vblank previously
 - Set DPST_CTL Bin Register Index to 0
 - Write enhancement factor to DPST_BIN Data
 - Go to step 4 until all 33 entries are written
 - The IE values are double-buffered and will update on the next start of pipe vblank. The values are not atomically updated or tied into the global double buffering control (disable/stall), so if programming straddles the pipe vblank start, some values will update in the upcoming frame and some in the frame after that.
- The enhancement factor modifies each input pixel component value with the method selected by DPST_CTL Enhancement Mode.
 - Direct lookup mode replaces the input pixel value with the enhancement factor.
 - Additive mode increases the input pixel value by the enhancement factor.
 - Multiplicative mode multiplies the input pixel value by the enhancement factor.

Processing Block

The software processing block responds to the histogram threshold interrupts, determines how much the backlight level can be reduced, then sets the backlight level and programs the Enhancement block.

The Processing block has the following requirements.

- Enable or disable the hardware blocks based on OS and user control inputs.
- Respond to the histogram threshold interrupts by reading the histogram and calculating a new backlight level based on statistics from the histogram.
 - The calculation is proprietary. In general, darker images will allow greater backlight reduction, and increased power savings.
 - The aggressiveness of backlight reduction can be controlled by the OS and user.
 - The final backlight reduction amount must be combined with backlight level requirements set by the OS, applications, and other power saving technologies.
- Program the Enhancement block to compensate for the brightness loss due to reducing the backlight level.
 - Because the maximum RGB component value is limited, not all pixel values can be perfectly compensated.
 - The number of pixel values which cannot be perfectly compensated is a function of the aggressiveness level.
- Phase-in the backlight level change and pixel enhancement gradually, in order to avoid flickering artifacts.

Pipe Palette and Gamma

The display palette provides a means to correct the gamma of an image stored in a frame buffer to match the gamma of the monitor or presentation device. Additionally, the display palette provide a method for converting indexed data values to color values for VGA and 8-bpp indexed display modes. The display palette is located after the plane blender. Using the individual plane gamma enables, the blended pixels can go through or bypass the palette on a pixel by pixel basis.

PAL_LGC

PAL_PREC_INDEX

PAL_PREC_DATA

PAL_GC_MAX

PAL_EXT_GC_MAX

GAMMA_MODE

PAL_EXT2_GC_MAX

PRE_CSC_GAMC_INDEX

PRE_CSC_GAMC_DATA

PAL_PREC_MULTI_SEG_INDEX

PAL_PREC_MULTI_SEG_DATA

If any gamma value to be programmed exceeds the maximum allowable value in the associated gamma register, then the programmed value must be clamped to the maximum allowable value.

Programming Modes

The display palette can be accessed through multiple methods and operate in one of four different modes as follows.

8 bit legacy palette/gamma mode:

This provides a palette mode for indexed pixel data formats (VGA and primary plane 8 bpp) and gamma correction for legacy programming requirements.

All input values are clamped to the 0.0 to 1.0 range before the palette/gamma calculation. It is not recommended to use legacy palette mode with extended range formats.

For input values greater than or equal to 0 and less than 1.0, the input value is used to directly lookup the result value from one of the 256 palette/gamma entries. The 256 entries are stored in the legacy palette with 8 bits per color in a 0.8 format with 0 integer and 8 fractional bits.

The legacy palette is programmable through both MMIO and VGA I/O registers. Through VGA I/O, the palette can look as though there are only 6 bits per color component, depending on programming of other VGA I/O registers.

Direct lookup (10 bit) gamma mode:

This provides the highest quality gamma for pixel data formats of 30 bits per pixel or less.

For input values greater than or equal to 0 and less than 1.0, the input value is used to directly lookup the result value from one of the first 1024 gamma entries. The first 1024 entries are stored in the precision palette with 10 bits per color in a 0.10 format with 0 integer and 10 fractional bits.

For input values greater than or equal to 1.0 and less than 3.0, the input value is used to linearly interpolate between the 1024th and 1025th gamma entries to create the result value. The 1025th entry is stored in the PAL_EXT_GC_MAX register with 19 bits per color in a 3.16 format with 3 integer and 16 fractional bits.

For input values greater than or equal to 3.0 and less than 7.0, the input value is used to linearly interpolate between the 1025th and 1026th gamma entries to create the result value. The 1026th entry is stored in the PAL_EXT2_GC_MAX register with 19 bits per color in a 3.16 format with 3 integer and 16 fractional bits.

All input values are clamped to the greater than -7.0 and less than 7.0 range before the gamma calculation.

For negative input values, gamma is mirrored along the X-axis, giving the same result as positive input values, except for a negative sign. When gamma input may be negative, the first gamma point should be programmed to a value of 0.0 in order to have a symmetric mirroring.

Interpolated gamma mode:

This mode uses up to 515 gamma entries and the gamma output gets computed through interpolation between the neighboring LUT entries.



The gamma correction curve is represented by specifying a set of gamma entry reference points spaced equally along the curve for values between -1 and 1. For extended values there is an extended gamma entry reference point at the maximum allowed input value.

For input values greater than or equal to 0 and less than 1.0, the input value is used to linearly interpolate between two adjacent points of the first 513 gamma entries to create the result value. The first 512 entries are stored in the precision palette with 16 bits per color in a 0.16 format with 0 integer and 16 fractional bits (upper 10 bits in odd indexes, lower 6 bits in even indexes). The 513th entry is stored in the PAL_GC_MAX register with 17 bits per color in a 1.16 format with 1 integer and 16 fractional bits.

For input values greater than or equal to 1.0 and less than 3.0, the input value is used to linearly interpolate between the 513th and 514th gamma entries to create the result value. The 514th entry is stored in the PAL_EXT_GC_MAX register with 19 bits per color in a 3.16 format with 3 integer and 16 fractional bits.

For input values greater than or equal to 3.0 and less than 7.0, the input value is used to linearly interpolate between the 514th and 515th gamma entries to create the result value. The 515th entry is stored in the PAL_EXT2_GC_MAX register with 19 bits per color in a 3.16 format with 3 integer and 16 fractional bits.

All input values are clamped to the greater than -7.0 and less than 7.0 range before the gamma calculation.

For negative input values, gamma is mirrored along the X-axis, giving the same result as positive input values, except for a negative sign. When gamma input may be negative, the first gamma point should be programmed to a value of 0.0 in order to have a symmetric mirroring.

To program the gamma correction entries, calculate the desired gamma curve for inputs from 0 to 3.0. The curve must be flat or increasing, never decreasing. For inputs of 0 to 1.0, multiply the input value by 512 to find the gamma entry number, then store the desired gamma result in that entry. For inputs greater than 1.0 and less than or equal to 3.0, store the result for an input of 3.0 in the 514th gamma entry.

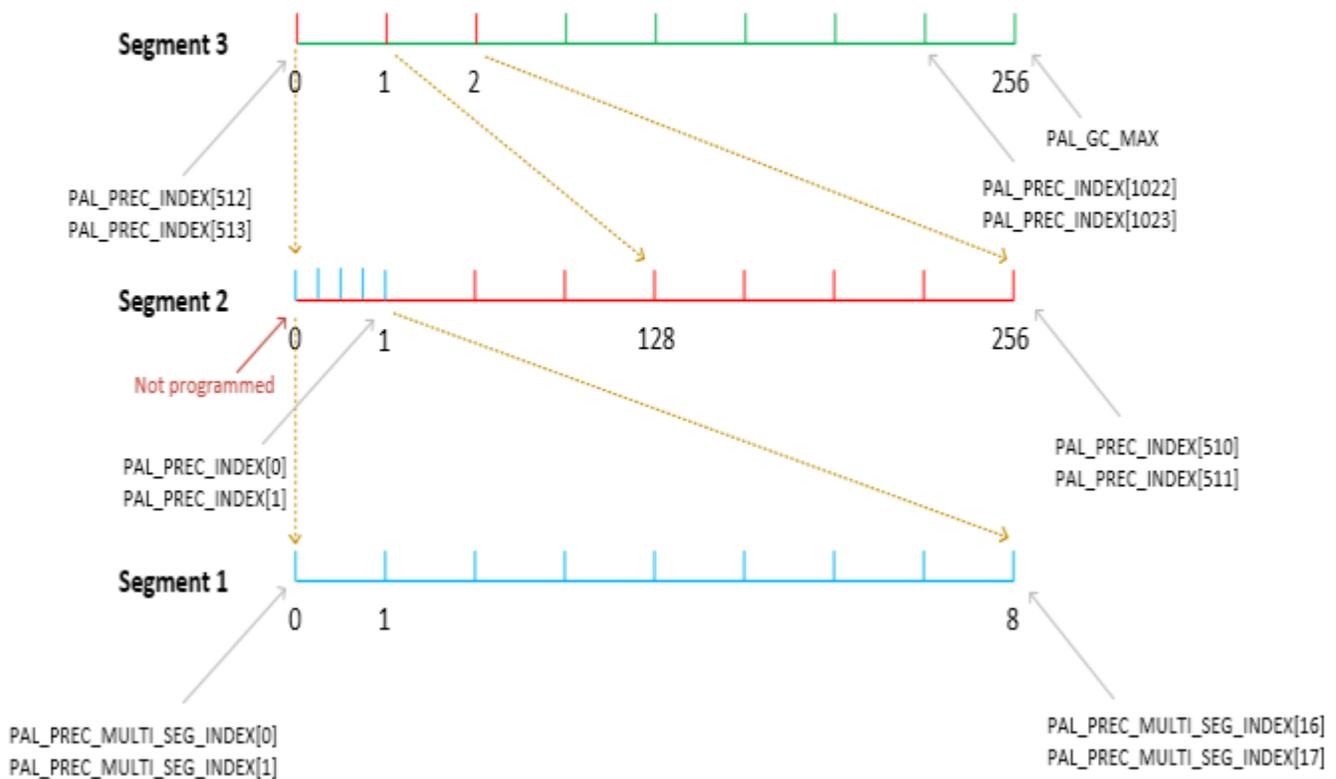
Multi-segmented interpolated gamma mode:

This provides the highest quality gamma for HDR with three segments, each addressing different parts of the gamma curve. The gamma output gets computed through interpolation between the neighboring LUT entries.

The gamma correction curve is represented by specifying a set of gamma entry reference points, spaced equally within each segment, along the curve for values between -1 and 1. For extended values there is an extended gamma entry reference point at the maximum allowed input value.

All input values are clamped to the greater than -7.0 and less than 7.0 range before the gamma calculation.

Segmented Gamma



For input values greater than or equal to 0 and less than 1.0, the input value is used to linearly interpolate between two adjacent points of the gamma entries to create the result value. The gamma entries in all segments are stored in the precision palette with 16 bits per color in a 0.16 format with 0 integer and 16 fractional bits (upper 10 bits in odd indexes, lower 6 bits in even indexes). The 513th entry is stored in the `PAL_GC_MAX` register with 17 bits per color in a 1.16 format with 1 integer and 16 fractional bits.

For input values greater than 1.0 and less than 3.0, the input value is used to linearly interpolate between the 513th and 514th gamma entries to create the result value. The 514th entry is stored in the `PAL_EXT_GC_MAX` register with 19 bits per color in a 3.16 format with 3 integer and 16 fractional bits. For input values greater than or equal to 3.0 and less than 7.0, the input value is used to linearly interpolate between the 514th and 515th gamma entries to create the result value. The 515th entry is stored in the `PAL_EXT2_GC_MAX` register with 19 bits per color in a 3.16 format with 3 integer and 16 fractional bits.

For negative input values, gamma is mirrored along the X-axis, giving the same result as positive input values, except for a negative sign. When gamma input may be negative, the first gamma point should be programmed to a value of 0.0 in order to have a symmetric mirroring.

To program the gamma correction entries, calculate the desired gamma curve for inputs from 0 to 7.0. The curve must be flat or increasing, never decreasing. For inputs of 0 to 1.0, program the gamma segments 1, 2 and 3. For inputs greater than 1.0 and less than or equal to 3.0, store the result for an input of 3.0 in the 514th gamma entry.

The Segment 1 or the super fine segment has 9 entries covering the input range from 0 to $1/(128*256)$ => $[0, 1/(8* 128*256), 2/(8* 128*256) \dots 1/(128*256)]$. The 9 entries of Segment 1 gamma values gets programmed using the PAL_PREC_MULTI_SEG* registers ranging from index 0 to index 17.

The Segment 2 or the fine segment has 257 entries covering the input range from 0 to $1/128$ => $[0, 1/(128*256), 2/(128*256) \dots 1/128]$. The Segment 2 gamma values gets programmed using the PAL_PREC* registers ranging from index 0 to index 510. The Segment 2 zero value is implied in hardware and is not explicitly programmed by software. Segment 2 first value is programmed in PAL_PREC* index 0 as shown in the figure above.

The Segment 3 or the coarse segment has 257 entries covering the input range from 0 to 1 => $[0, 1/256, \dots 255/256, 1.0]$. The first 256 entries of Segment 3 gamma values gets programmed using the PAL_PREC* registers ranging from index 512 to index 1023. The 257th value (1.0) is programmed in PAL_GC_MAX register.

Pipe Control

PIPE_SRC SZ

PIPE_SCANLINE

PIPE_SCANLINECOMP

PIPE_MISC

PIPE_FRMTMSTMP

PIPE_FLIPTMSTMP

PIPE_BOTTOM_COLOR

PIPE_FLIPCNT

PIPE_FRMCNT

PIPE_FLIPDONETMSTMP

PIPE_MISC2

PIPE_STATUS

Pixel Passthrough Operation

For modes of operation where the input image CRC needs to match the output image CRC (i.e. the pixels need to flow through the pixel pipe unmodified from frame buffer to the port), the following programming needs to be done:

1. Use only a single Plane with no cursor (**CUR_CTL**)
2. Use a fixed point, non-planar pixel format without Alpha (**PLANE_CTL**). Make sure the frame buffer format matches the port output format

3. Disable all color correction (i.e. CSC, Gamma, etc), LACE, Scaling, compression and dithering within the Plane and Pipe. See the other sub-sections of the Pipe/Planes chapters for more details:
 1. Universal Plane
 2. Color Space Conversion
 3. Pipe Color Gamut Enhancement
 4. Pipe Palette and Gamma
 5. Pipe LDPST
 6. Pipe 3D LUT
 7. Pipe DPST
 8. Pipe Scaler
 9. DSC (Display Stream Compression)
4. If the Plane that is being used (i.e. enabled) is an HDR Plane, then the HDR Mode bit within **PIPE_MISC** must be set. Otherwise, the HDR Mode bit of PIPE_MISC must be **cleared**.
5. Disable (i.e. truncate) Pixel Rounding (**PIPE_MISC**)

Pipe Scaler

Each scaler has its own set of registers.

Scaler
PS_PWR_GATE
PS_WIN_POS
PS_WIN_SZ
PS_CTRL
PS_VSCALE
PS_HSCALE
PS_VPHASE
PS_HPHASE
PS_ECC_STAT
PS_ADAPTIVE_CTRL
PS_COEF_INDEX
PS_COEF_DATA
SCALER_COEFFICIENT_FORMAT
PS_PROG_HSCALE
PS_PROG_VSCALE

Scaler Programmed Coefficients

Two sets of programmed coefficients are available for each scaler. The horizontal filter and vertical filter can be configured individually to use one of these 2 sets. When used for YUV planar format plane scaling, the Y and UV scalers can be configured individually to use one of the 2 sets. Scaler coefficients



are accessed through their respective index and data registers following the mapping shown below. The coefficients must be programmed in the SCALER_COEFFICIENT_FORMAT.

17 phase of 7 taps requires 119 coefficients in 60 dwords per set. The letter represents the filter tap (D is the center tap) and the number represents the coefficient set for a phase (0-16).

Coefficient Set		
Index Value	Data Value Coefficient 2	Data Value Coefficient 1
00h	B0	A0
01h	D0	C0
02h	F0	E0
03h	A1	G0
04h	C1	B1
...
38h	B16	A16
39h	D16	C16
3Ah	F16	E16
3Bh	Reserved	G16

Nearest-neighbor scaling (Integer scaling ratios)

Nearest neighbor scaling can be used to maintain the rendering intent of some classic games in integer upscaling scenarios. For enabling nearest-neighbor scaling, the scaler must be set to use "programmed" mode with the center tap (Dxx) values set to 1 and all other values set to 0. The following coefficients values must be used with the coefficients programmed in the SCALER_COEFFICIENT_FORMAT.

Coefficient Set		
Index Value	Data Value Coefficient 2	Data Value Coefficient 1
00h	B0 = 0	A0 = 0
01h	D0 = 1	C0 = 0
02h	F0 = 0	E0 = 0
03h	A1 = 0	G0 = 0
04h	C1 = 0	B1 = 0
...
38h	B16 = 0	A16 = 0
39h	D16 = 1	C16 = 0
3Ah	F16 = 0	E16 = 0
3Bh	Reserved	G16 = 0

Register Double Buffering

Prior to D11, the Scaler has the following double buffer trigger points:

1. When the Pipe is disabled:
 - a. Any write to the PS_CTRL register (will refer to this as Control DB trigger)
 - b. Any write to the PS_WINSZ register (will refer to this as WinSize DB trigger)
2. When the Pipe is enabled:
 - a. For registers that are part of the Control DB trigger group, the DB trigger point is on the rising edge of V. Blank when double buffering is armed
 - b. For registers that are part of the WinSize DB trigger group, the DB trigger point is after the Frame Start when double buffering is armed

From D11 and onwards, the Scaler has the following double buffer trigger points:

1. When the Pipe is disabled:
 - a. Any write to the PS_CTRL register (Control DB trigger)
 - b. Any write to the PS_WINSZ register (WinSize DB trigger)
2. When the Pipe is enabled, the DB trigger point is on the rising edge of V. Blank when double buffering is armed and double buffering is allowed (Allow Double Buffer Update Disable = 1 in PS_CTRL)

When the Pipe is enabled, any write to the PS_WINSZ will arm the double buffering for the Scaler registers.

Once double buffering is armed, the disarming point is dependent on the Display generation.

- Prior to D11: Double buffering is disarmed on the next WinSize DB trigger (i.e. after Frame Start)
- D11+ : Double buffering is disarmed on the next rising edge of V. Blank where double buffering is allowed

As implied above, the double buffering for each of the Scaler registers is dependent on the trigger group it is within (i.e. Control or WinSize).

Register	DB Trigger
PS_PWR_GATE	Control
PS_CTRL	Control
PS_VPHASE	WinSize
PS_HPHASE	WinSize
PS_ADAPTIVE_SET_*_CTRL	WinSize
PS_WINPOS	WinSize
PS_WINSZ	WinSize

Note that the Programmable Coefficient sets accessed through the PS_COEF_SET_*_INDEX and PS_COEF_SET_*_DATA registers are on the Control DB trigger.

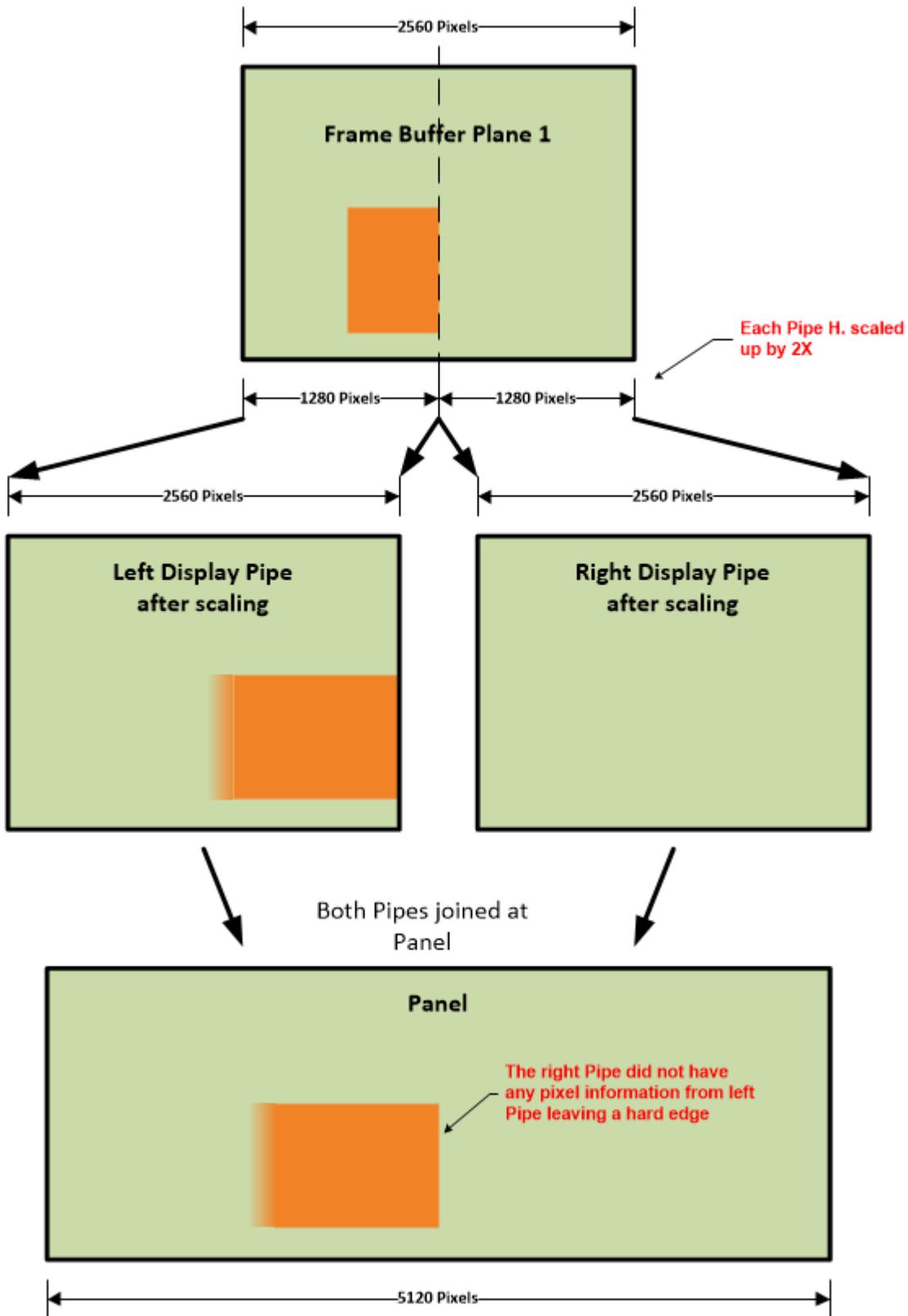
Programming Sequence

The below sequence should be followed when programming the Scaler to ensure proper double buffering:

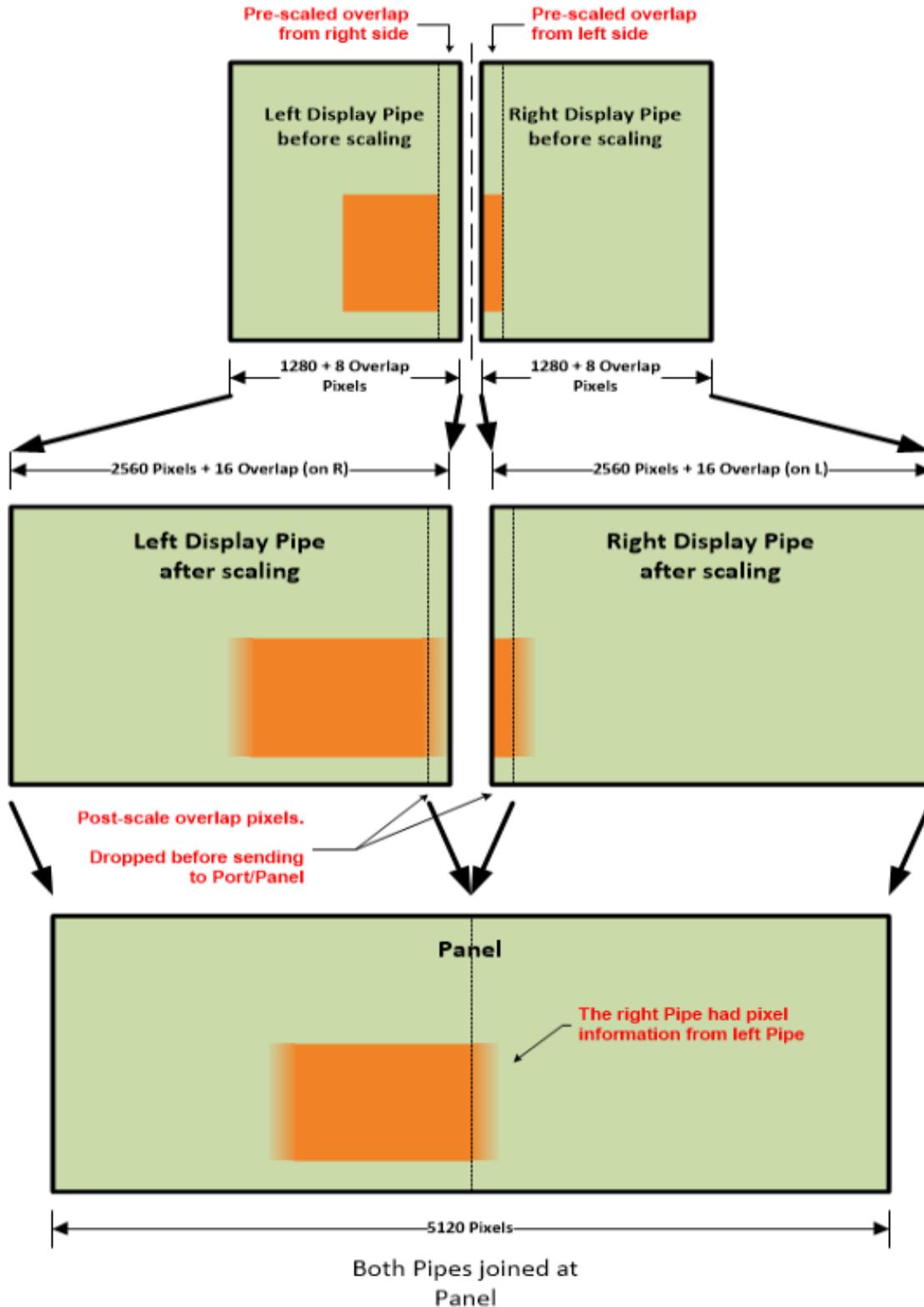
- Configure the Scaler's Programmable Coefficient sets through PS_COEF_SET_*_INDEX and PS_COEF_SET_*_DATA registers, if using Programmable Coefficients
- Configure power gating control of the Scaler SSA's (PS_PWR_GATE), if the Scaler is not already enabled
- Enable and/or configure Scaler (PS_CTRL), if needed
 - If the Scaler is not already enabled, the Scaler will begin the process of powering up the Scaler SSA's
 - Software doesn't have to write to this register to enable Control DB trigger registers (i.e. Programmable Coefficient sets) if the Scaler and Pipe are both enabled
- Configure initial phases (PS_*PHASE), adaptive control (PS_ADAPTIVE_SET_*_CTRL), and window position (PS_WINPOS), if needed
- Configure/write to the PS_WINSZ
 - For pre-D11 products, the write to this register should only be done outside of the V. Blank regime if the Pipe is enabled. There are no restrictions for D11+ products
 - If Software has changed any PS_* register programming and the Pipe is enabled, then Software has to write to this register regardless of whether the value is changing, or not.

Tiled Scaling (Seam Removal)

There are certain usage cases where an image from memory will be horizontally split across two Pipes, scaled up, and then joined at the Port/Panel. When the image is being horizontally scaled up across the seam of the split image if the Scalers within each Pipe do not have some additional pixels from the other Pipe's image, then an artifact at the seam can occur.



By adding some overlapping pixels of the split image around the seam, the Scalers within each Pipe will be able to correctly filter across the seam. At the end of the Pipe (before the image is delivered to the Port) the post-scaled overlap pixels will be dropped.



To perform the cross seam scaling, Software will be responsible for the following:

- It will calculate the pre and post scale excesses needed for each Pipe
- It will include the pre-scale excess within the pre-scaled Horizontal image sizes (e.g. the Horizontal Source Size of the PIPE_SRC SZ register)
- It will include the post-scale excess within the Scaler's Window Size register (PS_WIN_SZ)
- It will program the post-scale excess within the PIPE_SEAM_EXCESS register
- It will program the Horizontal Active size at the transcoder (TRANS_HTOTAL) to be the Panel width
 - This should **not** include any seam pixels

The following sections will discuss how Software will determine the pre/post excess sizes and the initial phase needed for the Scaler processing the right side image.

Location of Seam and Splitting Source Image

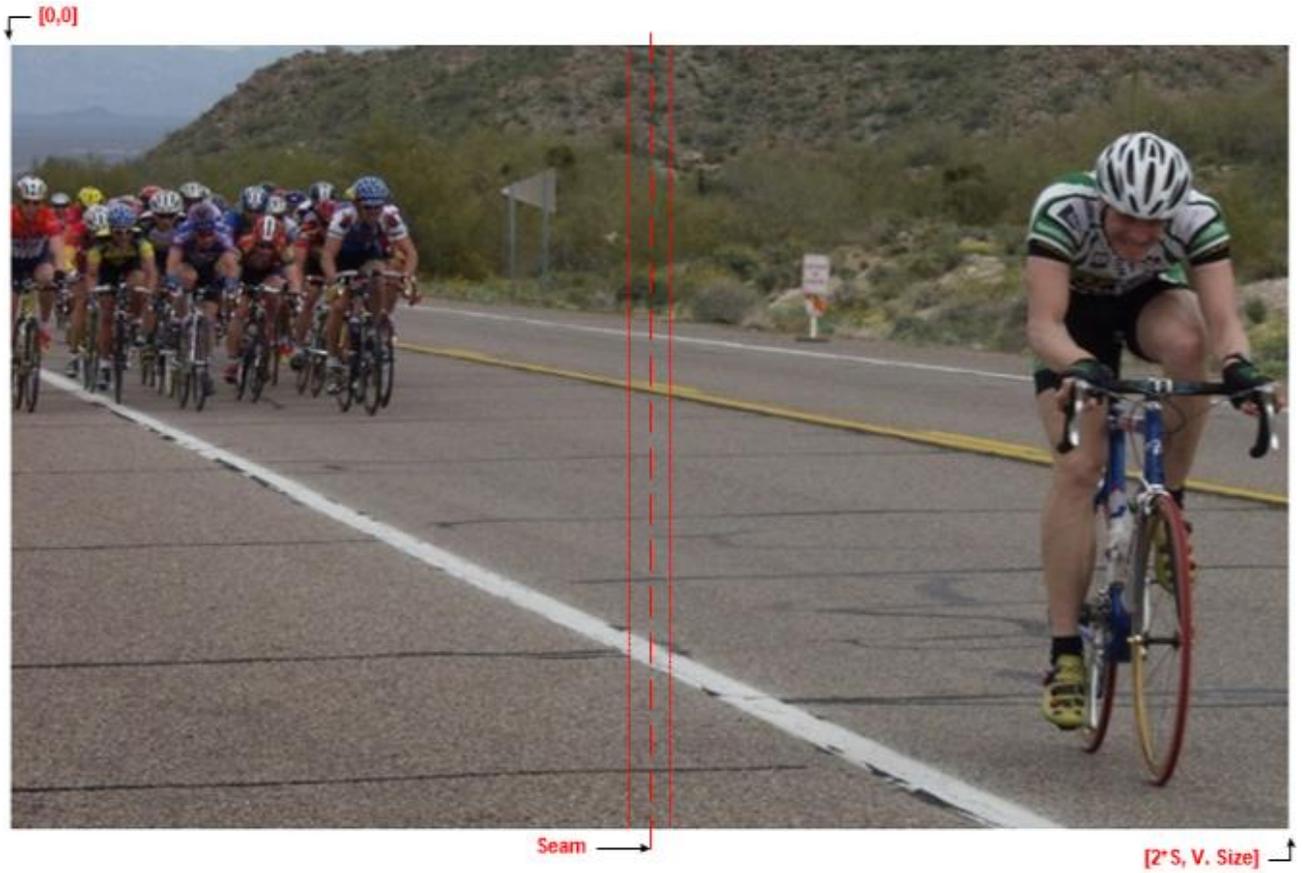
When the location of an image in memory is going to cross the seam between the two Pipes, then the PLANE_SIZE, PLANE_POS, and PLANE_OFFSET registers of the Planes carrying the image (one within each Pipe) will determine how that image in memory is split across the two Pipes.

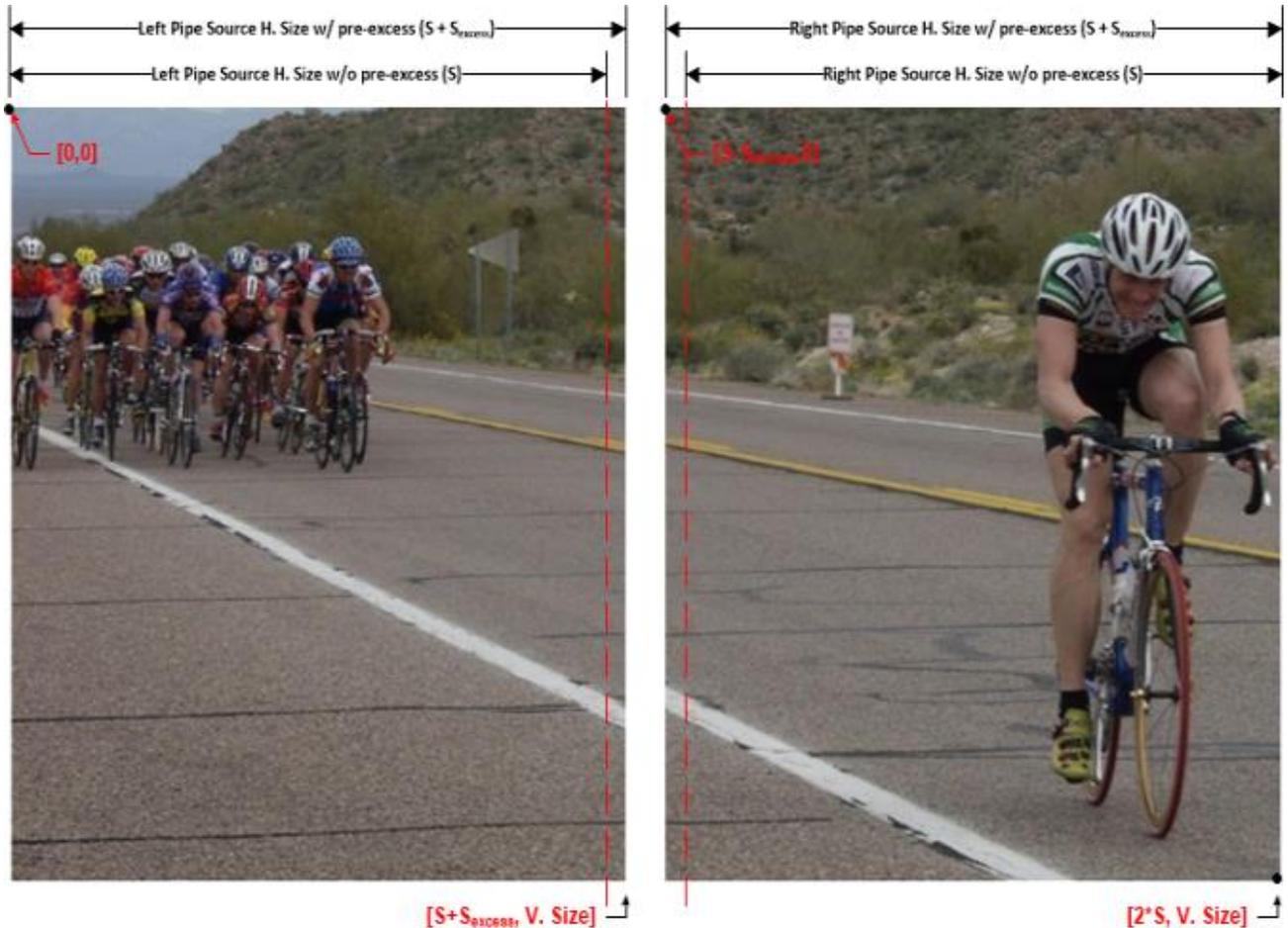
Seam Location

The location of the seam is relative to the left or right side of the Pipe Source Size window where the horizontal Pipe Source Size will include the pre-scaled excess (a.k.a. Source Excess, or S_{excess})

$$H. \text{ Pipe Source Size} = \text{Target H. Pipe Source Size (S)} + \text{Source Excess (S}_{\text{excess}})$$

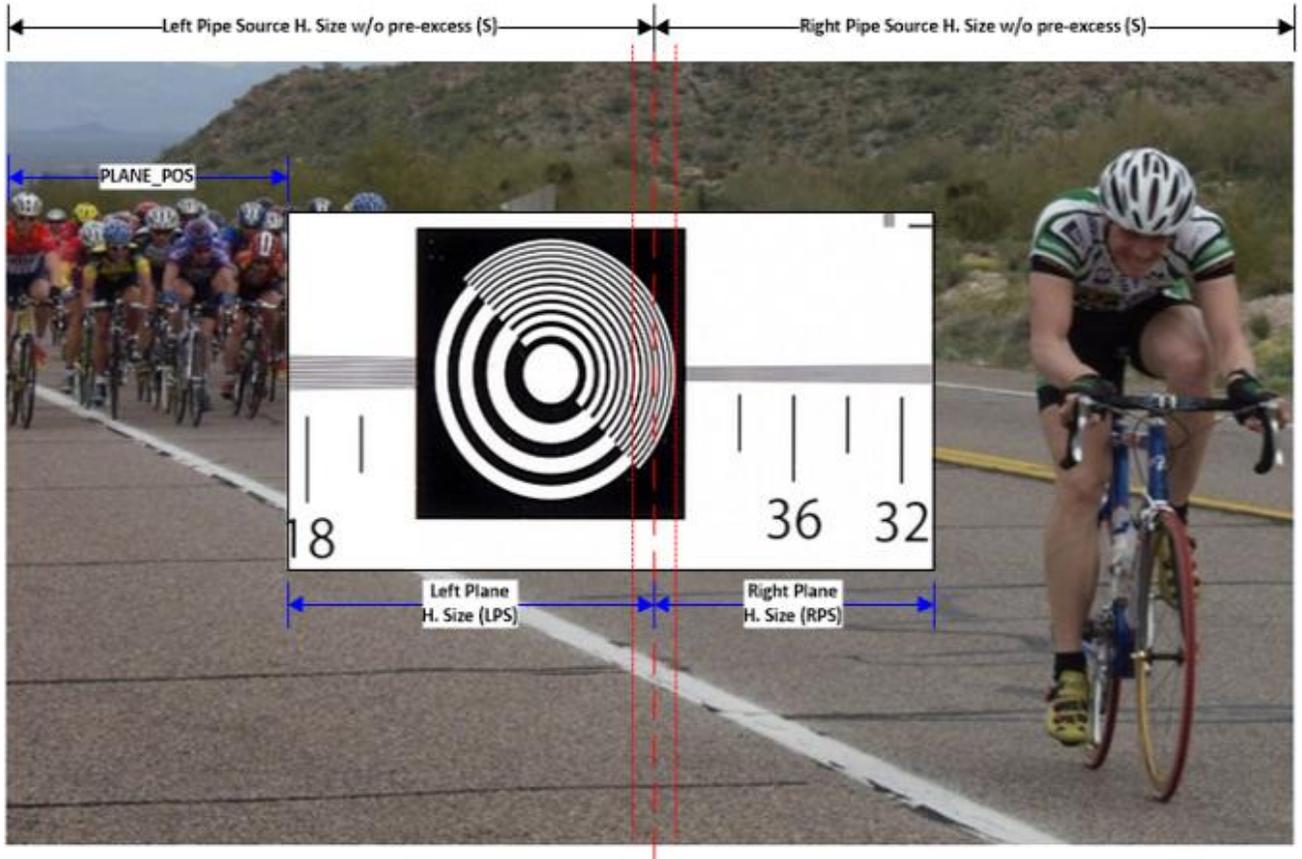
For the left-side Pipe, the seam will be located on the right-side of its source window, and for the right-side Pipe, the seam will be located on the left-side of its source window.





Note that this discussion assumes symmetric horizontal Pipe Source Sizes without the source excess (i.e. S) across both Pipes. This is not a requirement, but it will be left to the reader to extrapolate the equations within this section for asymmetric S terms.

Locating the seam within an image that spans the horizontal Pipe Source Size across both Pipes (e.g. the desktop/background image) is simply a function of the Pipe. But, if a Plane image is smaller than the combined horizontal Pipe Source Sizes, then the location of the seam is dependent on the offset of the Plane within the Pipe Source Window.



So, if the addition of the Plane Position (PP) and the Full H. Plane Size (HPS) is greater than S, then the following equations define where the seam is located within the image in memory.

$$\mathbf{HPS = Left\ Plane\ Size\ (LPS) + Right\ Plane\ Size\ (RPS)}$$

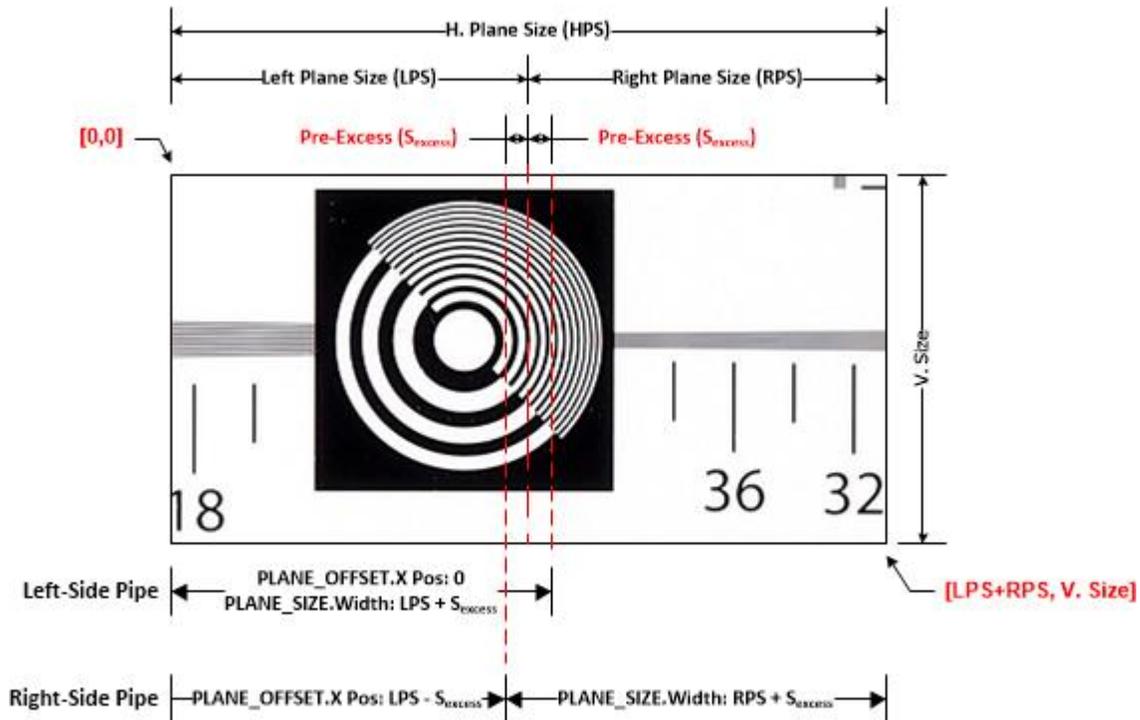
$$S = PP + LPS \Rightarrow \mathbf{LPS = S - PP}$$

$$\mathbf{RPS = HPS - (S - PP)}$$

Note that the Source Excess (S_{excess}) is a constant regardless of where the Plane image is located.

Splitting the Image

Now that the location of the seam is known, splitting the image across both Pipes can be performed. This is done using the PLANE_OFFSET and PLANE_SIZE registers.



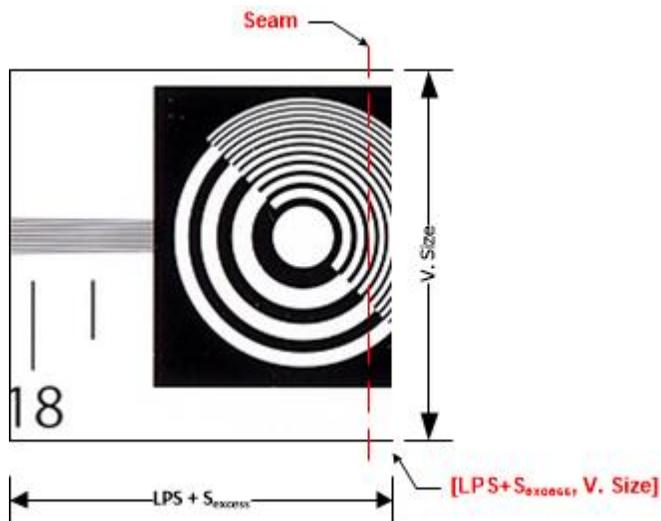
Using the figure above as a reference, to split the image in memory across the two Pipes.

Left PLANE_OFFSET:

- Starting X Position = H. Cropping Position (HCP) + 0
- Starting Y Position = V. Cropping Position (VCP) + 0

Left PLANE_SIZE:

- Width = Left Plane Size (LPS) + Pre-Excess Size (S_{excess})
- Height = V. Size

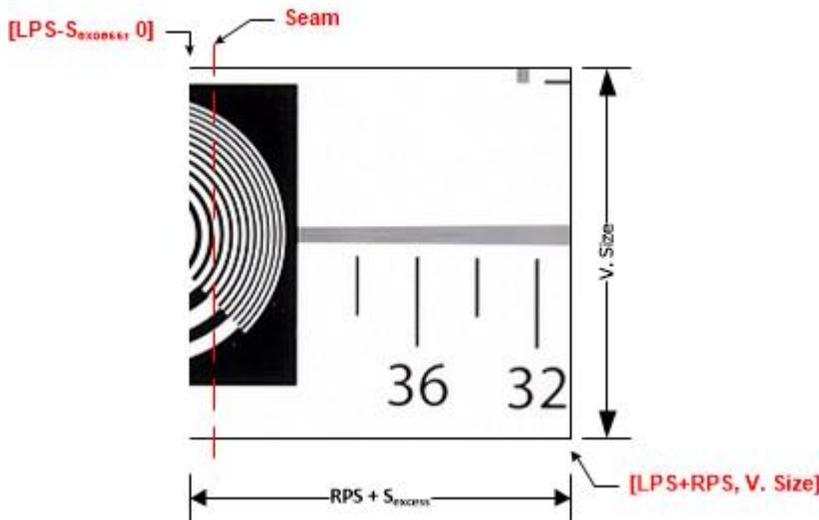


Right PLANE_OFFSET:

- Starting X Position = $HCP + LPS - S_{excess}$
- Starting Y Position = $VCP + 0$

Right PLANE_SIZE:

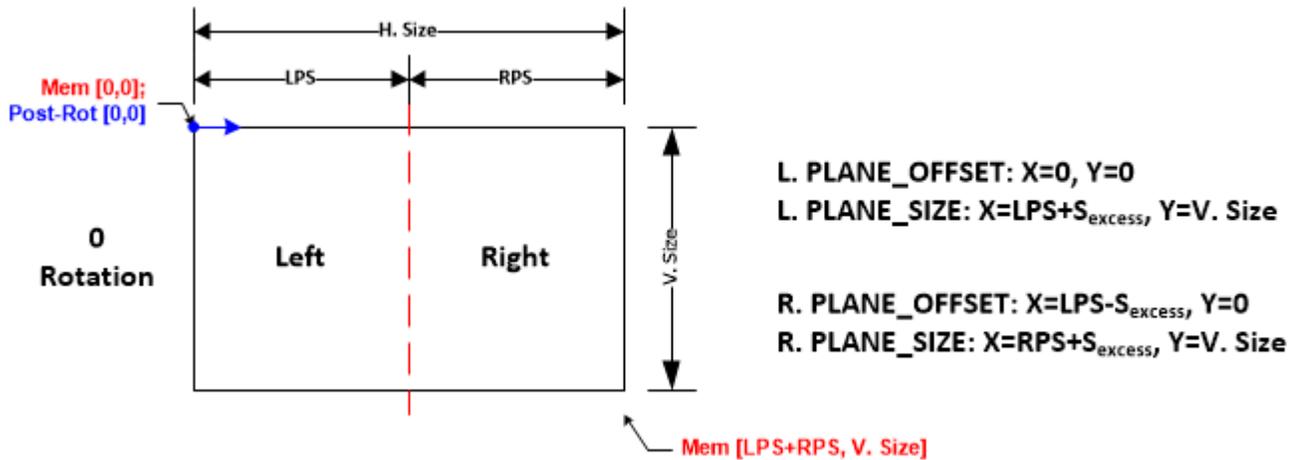
- Width = Right Plane Size (RPS) + S_{excess}
- Height = V. Size



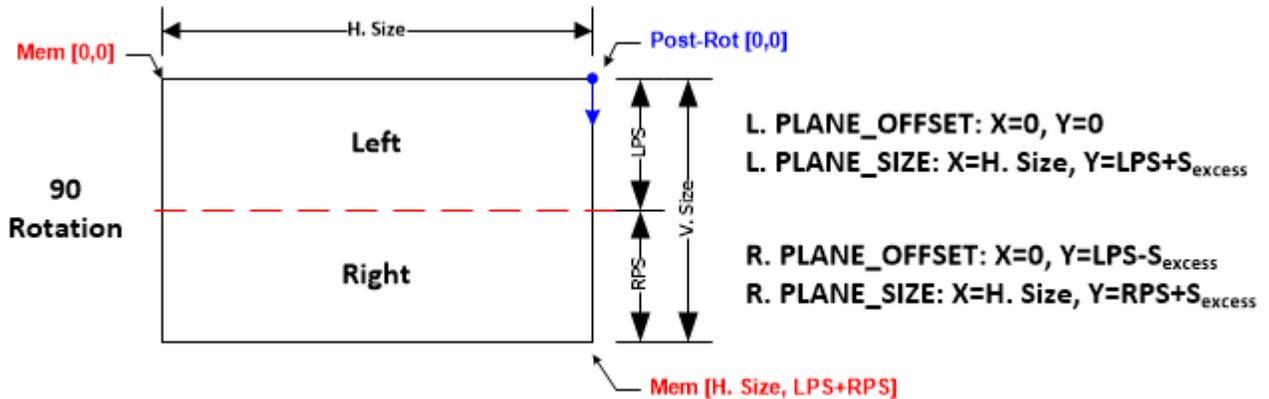
Rotation

The above discussion for splitting an image is when there is no rotation. If the image is to be rotated, then the programming of the PLANE_OFFSET and PLANE_SIZE needs to be adjusted accordingly. Since rotation is applied after the Plane Position and Size, the programming of these registers "rotates" to account for the image rotation.

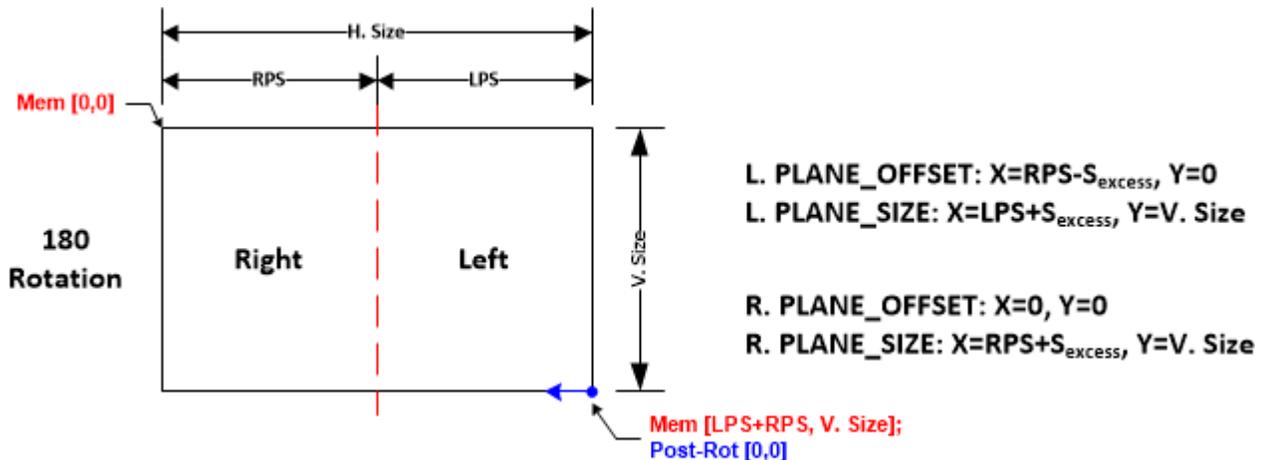
If we look at an image in memory that is going to be split without rotation, the regions of the image that are sent down each Pipe is shown below. Note that the blue dot and arrow represent the post rotation origin of the image and direction of scan.



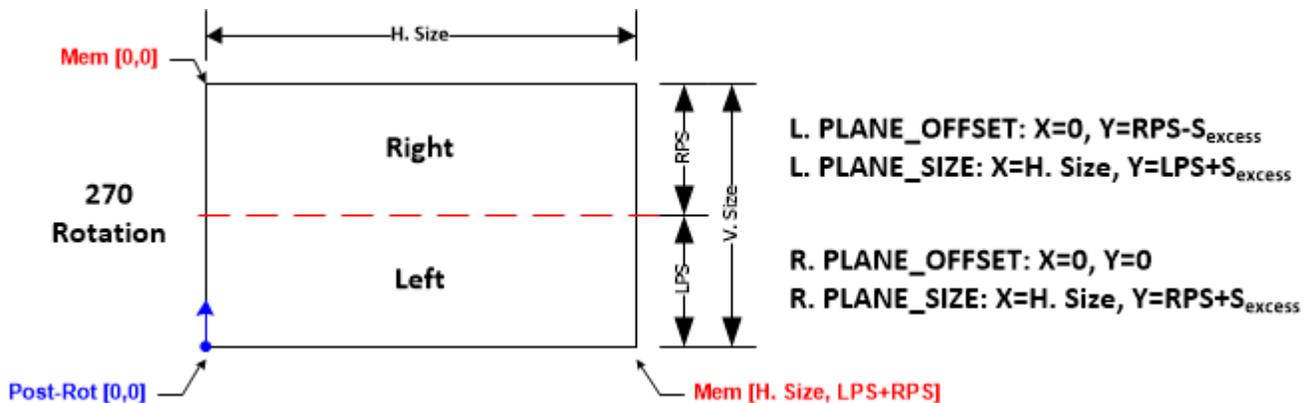
When the image is rotated by 90° , then the horizontal and vertical sizes programmed within the Plane/Pipe are swapped. So, the left and right regions swap as well.



When rotating another 90° (i.e. 180° rotation), then the left and right regions swap positions with respect to the no rotation scenario.



Finally, when the image is rotated 270° then the left and right regions swap positions with respect to the 90° rotation scenario.

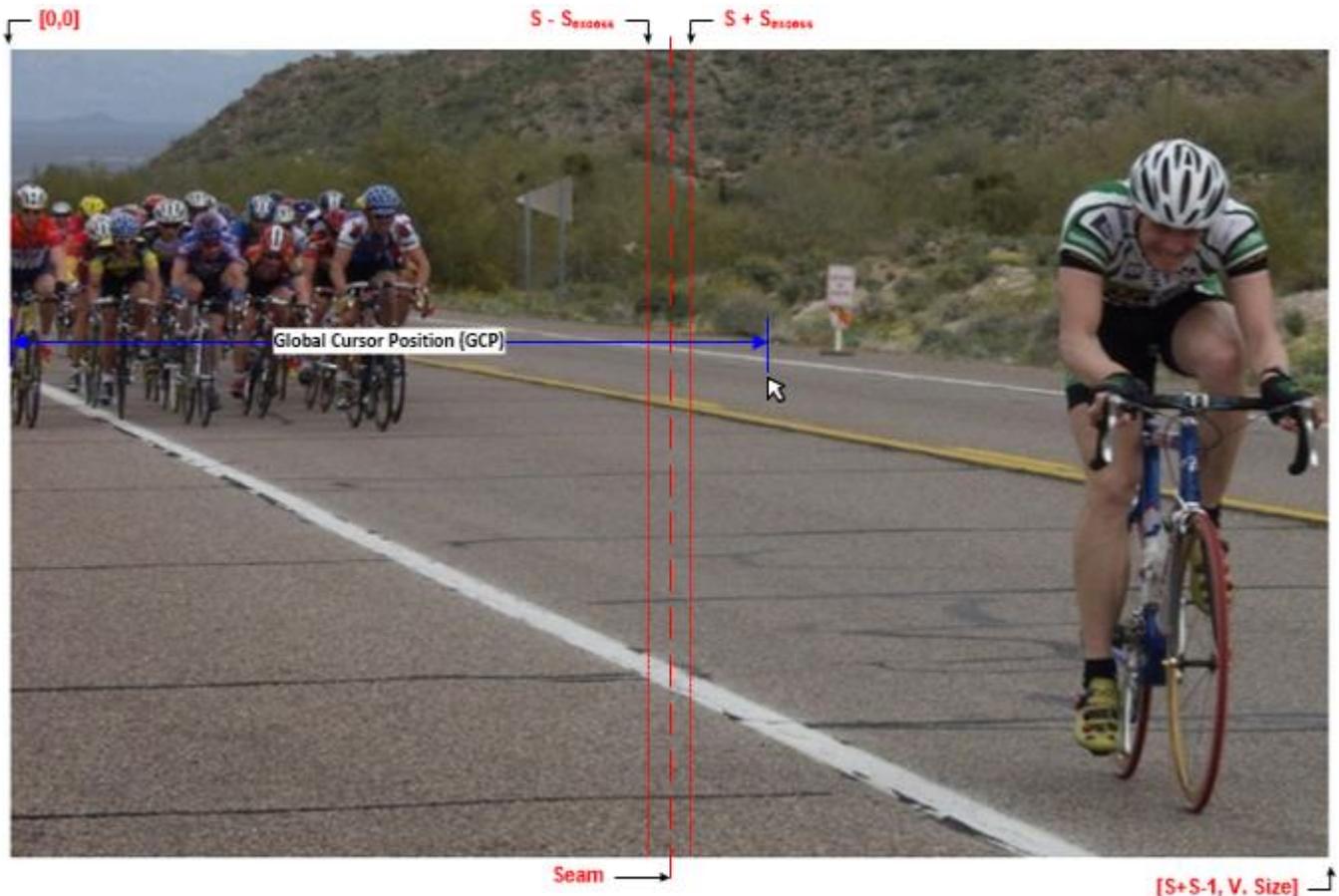


Note that Software should ensure that the horizontal Plane position for the image in the right Pipe is zero (i.e. there should be no positional offset of the image with respect to the seam)

Refer to the table within the Source Image Manipulation below for a summary of the Plane offset and size programming.

Cursor

The mechanism that Software uses to determine the horizontal position of the Cursor across both Pipes is beyond the scope of this document. However, to explain how the Cursor is enabled/programed around the seam, this document will use a global coordinate system that spans across both Pipe source windows.



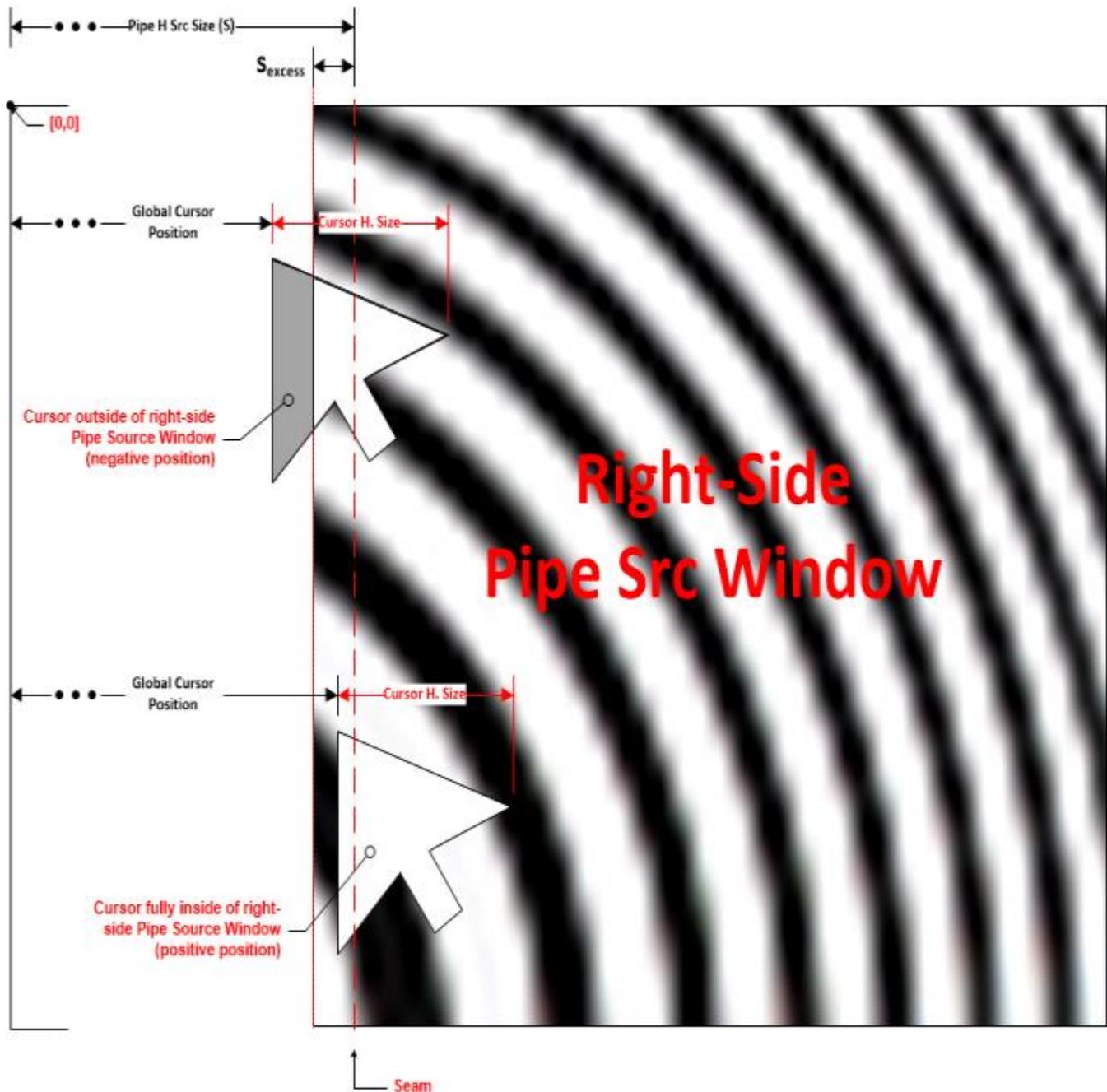
The horizontal position of the Cursor will be referred to as the global cursor position (GCP) that will span across both Pipe source windows from 0 to $(2*S)-1$.

When the Cursor is outside of the seam "window" ($S +/- S_{excess}$), then only one Pipe will have its Cursor enabled. As the Cursor moves into the seam window, then both Pipes will need to enable its Cursor. Determining when the Cursor is within this window is dependent on the global cursor position and the Cursor size (CS).

Global Cursor Position	Left Pipe Cursor	Right Pipe Cursor
$GCP < (S - S_{excess} - CS)$	Enabled	Disabled

$(S - S_{\text{excess}} - CS) \leq GCP < (S + S_{\text{excess}})$	Enabled	Enabled
$GCP \geq (S + S_{\text{excess}})$	Disabled	Enabled

The positioning of the Cursor will be dependent on where the Cursor is within the seam window. The left-side Cursor position is directly based off of the global cursor position, but the right-side Cursor is a little more involved.



The following table illustrates how to determine the Cursor position when it is crossing the seam.

Global Cursor Position	Left Pipe Cursor Position	Right Pipe Cursor Position
$GCP < (S - S_{\text{excess}} - CS)$	Magnitude = GCP Sign = SW determines	N/A (Disabled)
$(S - S_{\text{excess}} - CS) \leq GCP < (S - S_{\text{excess}})$	Magnitude = GCP Sign = Positive	Magnitude = $S - S_{\text{excess}} - GCP$ Sign = Negative
$(S - S_{\text{excess}}) \leq GCP < (S + S_{\text{excess}})$		Magnitude = $GCP - S - S_{\text{excess}}$ Sign = Positive
$GCP \geq (S + S_{\text{excess}})$	N/A (Disabled)	

Calculating Pre and Post Excess Sizes

To get an accurate sampling across the seam, the number of pre-scaled excess pixels (a.k.a. Source Excess, S_{excess}) needs to (at a minimum) fill the number of FIR taps to the left or right of the horizontal FIR's Center Tap.

$$S_{\text{excess}} \geq (\text{NUMTAPS} - 1)/2$$

Note that the Display Pipe requires an even number of line pixels, so the source excess may need to be rounded up to the next even number.

To determine the number of post-scaled excess pixels (a.k.a. Destination Excess, D_{excess}) we can use the following equation:

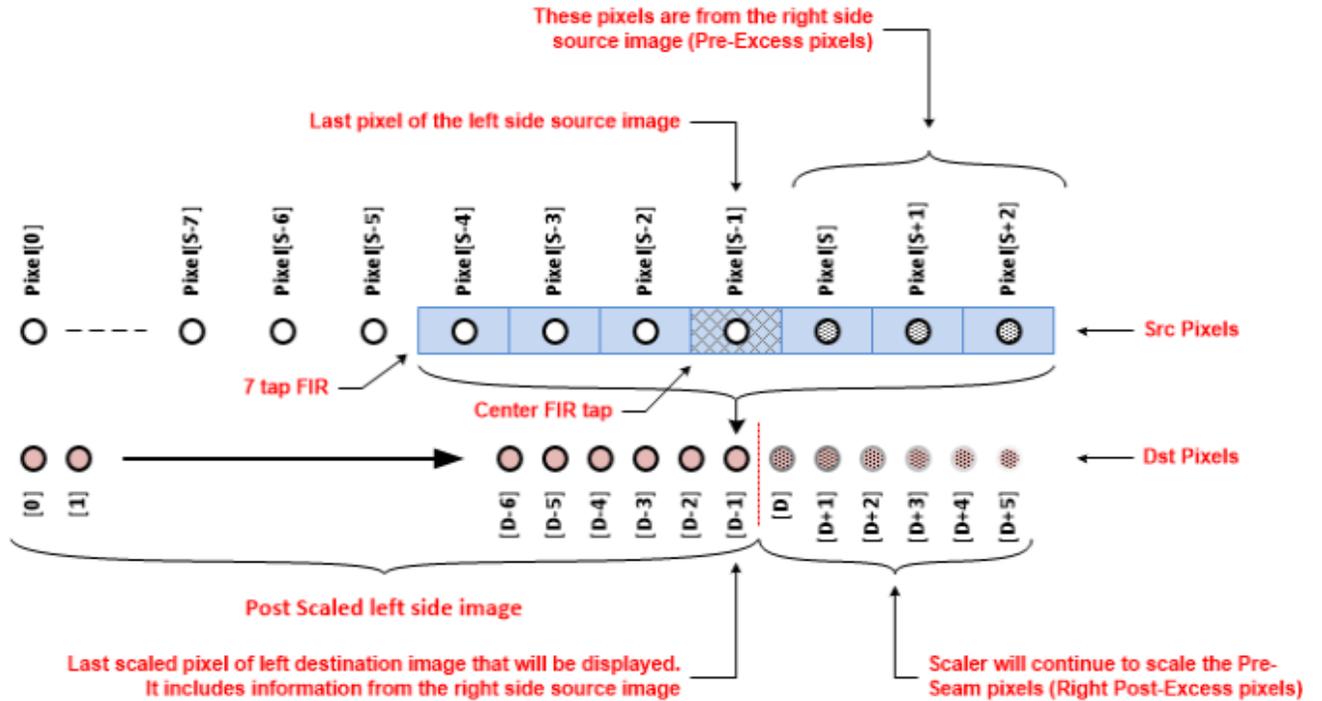
$$SF_{\text{target}} = (S + S_{\text{excess}}) / (D + D_{\text{excess}})$$

$$D_{\text{excess}} = \text{Round}(((S + S_{\text{excess}}) / SF_{\text{target}}) - D)$$

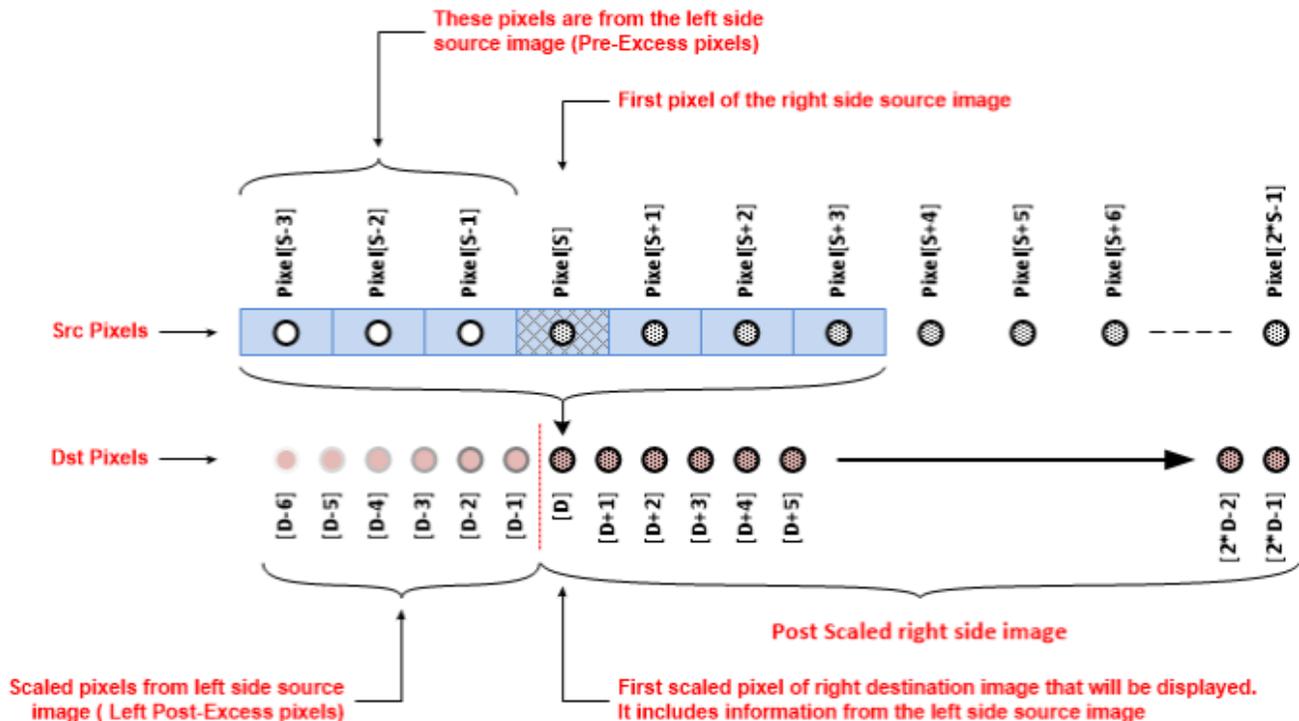
Where:

- S is the Source size in pixels per Pipe (one-based and without excess pixels)
- D is the Destination size in pixels per Pipe (one-based and without excess pixels)
- SF_{target} is the target Scale Factor (i.e. S/D)

Note that the D_{excess} must be a multiple of DISP_NUMPPC since the pixel dropping logic works at that granularity.



The above illustrates the Scaler output for the left side image (seam is on the right side). As the last pixel of the left side source image moves into the center tap of the Scaler's FIR, all of the taps to the right of the center tap contain pixels from the right side source image (Pixels S to $S+2$). The resulting destination pixel (this is the last displayable pixel) will contain all of the necessary information from the right side of the source image. All of the pixels beyond the last pixel of the destination image are the post excess pixels and those will be dropped.

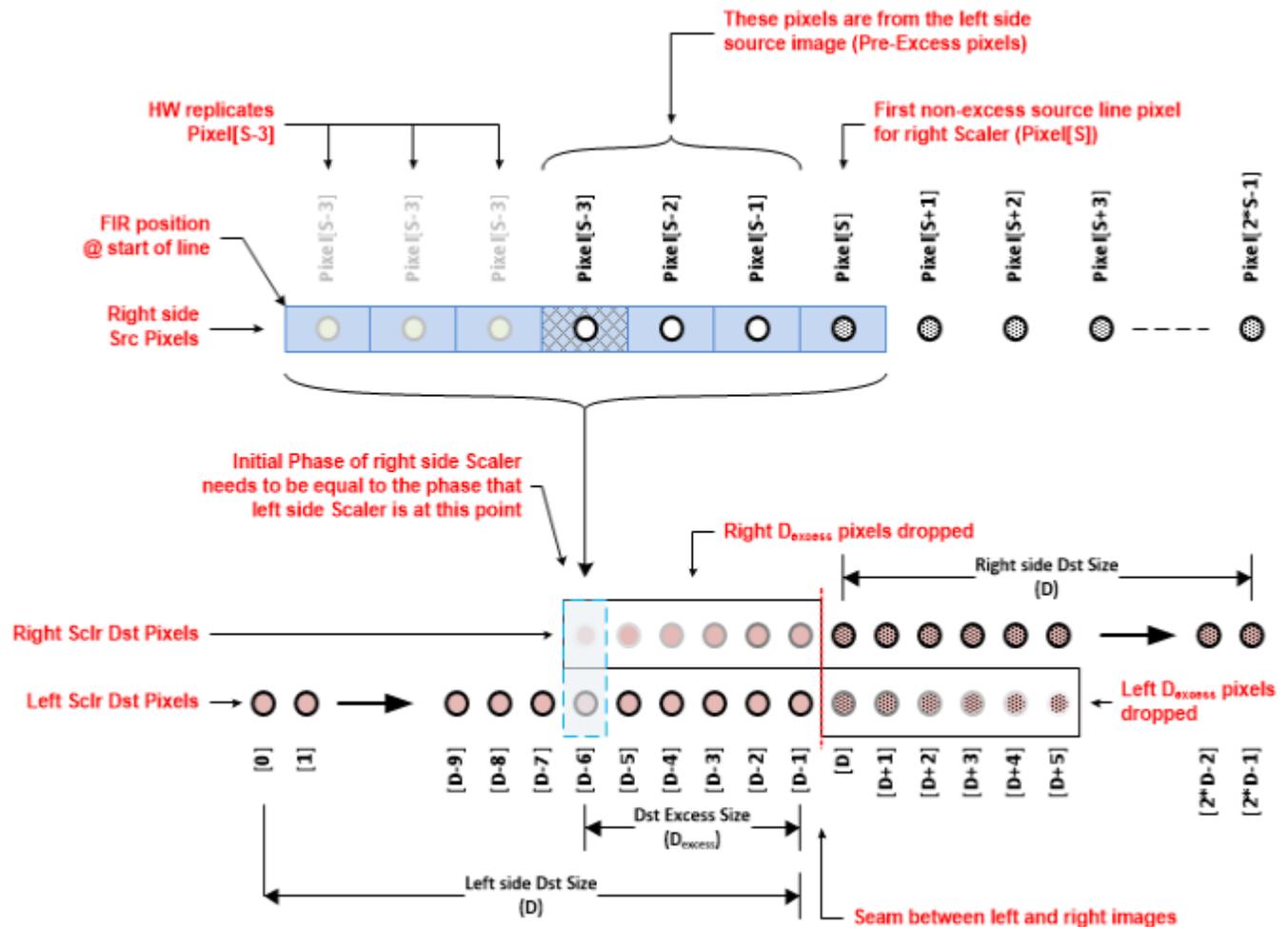


The above illustrates the Scaler output for the right side image (seam is on the left side). As the FIR progresses, the first pixel of the right side source image eventually moves into the center tap and all of the taps to the left of the center tap contain pixels from the left side source image (Pixels S-3 to S-1). All pixels up to this point are the post excess pixels that will be dropped. The resultant destination pixel when the first right side source image pixel makes it to the center pixel contains all of the necessary information from the left side of the source image.

Calculating Initial Phases

To ensure the image from both Pipes appears to be scaled by a single Scaler, the horizontal initial phase for the Scaler processing the right-side image will need to be set such that the phase of the horizontal FIR of the right Scaler picks up where the horizontal FIR of the left Scaler leaves off.

When the horizontal filter starts processing a line, it will load the first source pixel of the line into the Center Tap of the FIR and then apply an initial phase from that point (note that the Scaler can only apply positive initial phases). For the right-side Scaler, the first pixel of the source line will be the left most Source Excess pixel (i.e. a pixel from the left source image). To match the phase of the horizontal FIR of the right-side Scaler at the **beginning** of its source line with the phase of the horizontal FIR of the left-side Scaler at the **end** of its source line we need to determine the phase of the left-side Scaler's horizontal FIR ($D_{\text{excess}}-1$) pixels **before** it reaches the final Destination pixel ($D-1$).



The following equation is used to determine the phase that the right-side Scaler needs to start at.

$$P_{\text{right}} = SF_{\text{actual}} * (D - D_{\text{excess}})$$

Notes:

1. To get an accurate initial phase, need to use the actual scale factor (SF_{actual}) that the Scaler on the left is using. Depending on configuration, this is not always the target scale factor
 - a. If the Scaler HW is calculating the Scale Factor: $SF_{\text{actual}} = (S + [\text{Left, Right}] S_{\text{excess}}) / (D + D_{\text{excess}})$
 - b. If the Scaler HW is using the programmed Scale Factor: $SF_{\text{actual}} = PS_PROG_HSCALE$
 - i. The PS_PROG_HSCALE register should equal the target Scale Factor (SF_{target})
2. The Scale Factor has a precision of 2.14, regardless of how it is calculated. However, HW will automatically round the Scale Factor (Software should do the same)

The absolute phase difference between the left and the right will be used as the Horizontal Initial Phase (HIP) to be applied to either the left or right side.

$$\text{Absolute Horizontal Initial Phase} = | (S - S_{\text{excess}}) - P_{\text{right}} |$$

The side that the initial phase is applied will depend on which side is ahead of the other.

- If the left-side Scaler phase is less than or equal to the source pixel that the right-side Scaler starts at (i.e. $(S - S_{\text{excess}}) - P_{\text{right}}$ is positive) then the HIP will be applied to the left-side Scaler
- Otherwise, the HIP will be applied to the right-side Scaler

Notes:

1. Make sure to set the Initial Phase Trip bit of PS_HPHASE
2. The act of applying an initial phase will cause the image (on the left or the right) to appear to "shift". The shifting will be more evident for the left-side.

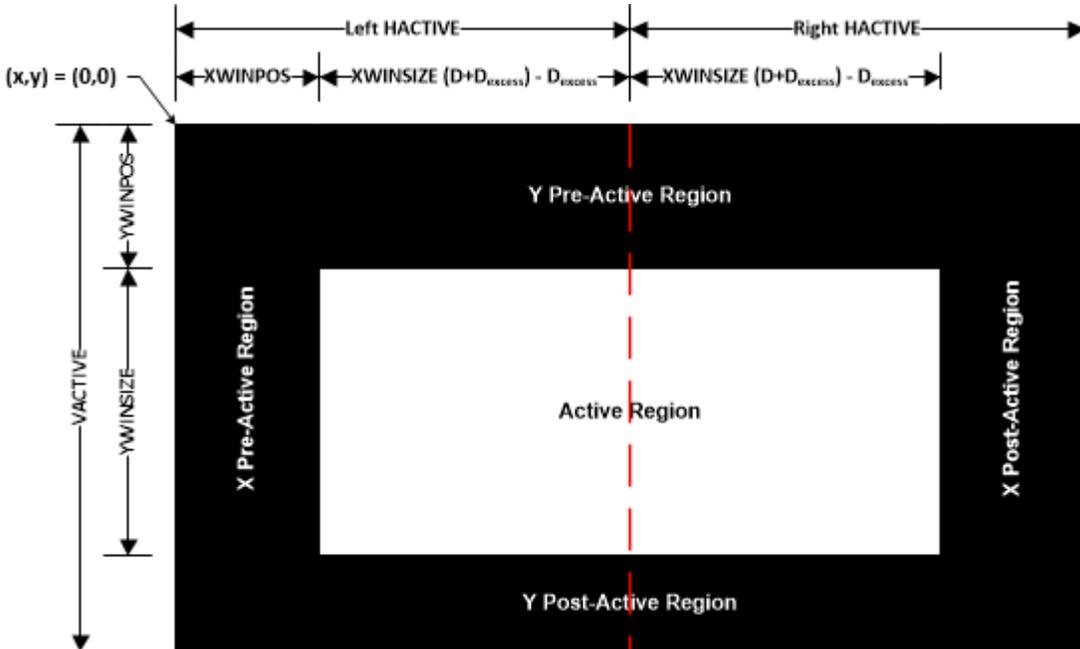
The following table illustrates an example of a 1920x1080 source image that is split across two Pipes with Pipe Source Sizes of $960 + S_{\text{excess}}$.

		Notes
Source Size (S)	960	Horizontal Pipe Source Size w/o S_{excess}
Source Excess (S_{excess})	4	$EVEN((NUMTAPS - 1)/2)$
Destination Size (D)	2576	Number of destination pixels per Pipe w/o D_{excess}
Target Scale Factor (SF_{target})	0.372670807	S/D
Actual Scale Factor (SF_{actual})	0.372497559	HW calculated by left Scaler: $(S + S_{\text{excess}}) / (D + D_{\text{excess}})$ Result is rounded to 2.14 precision
Destination Excess (D_{excess})	12	$ROUND(((S + S_{\text{excess}}) / SF_{\text{target}}) - D)$
Initial Phase needed on Right (P_{right})	955.0837402	$SF_{\text{actual}} * (D - D_{\text{excess}})$
First Src Pixel on Right	956	$S - S_{\text{excess}}$
Absolute Horizontal Initial Phase	0.9162598	$ (S - S_{\text{excess}}) - P_{\text{right}} $

Left Scaler Initial Phase	0.9162598	$(S - S_{\text{excess}}) \geq P_{\text{right}}$
Right Scaler Initial Phase	0.0	$(S - S_{\text{excess}}) \geq P_{\text{right}}$

Pillar Box Windowing

If pillar-box borders are going to be applied by the Scaler, then Software will need to account for this across both Pipes when programming the PS_WIN_POS register.



As we can see from the above illustration, the left-side border is applied by the PS_WIN_POS programming of the left-side Scaler and the right-side border is applied by the H. Active programming of the right-side transcoder.

Left Side:

- PS_WIN_POS.X Position = Left side border size
- TRANS_HTOTAL.H Active = Destination image size w/o excess (D) + Left border size

Right Side:

- PS_WIN_POS.X Position = 0
- TRANS_HTOTAL.H Active = Destination image size w/o excess (D) + Right border size

Summary:

The following is a summary of the above discussion that Software will need to perform for tiled Pipe scaling.

Some general notes:

1. Adding excess pixels for tiled Pipe scaling is only necessary when upscaling the tiled images. If downscaling the tiled images, then excess pixels do not need to be added.
2. This document assumes that the pre-excess horizontal Pipe Source Size (i.e. S) is symmetric across both Pipes (i.e. S is the same for both the left and right-side Pipes). It is not a requirement that this term be symmetric, but it will be left to the reader to extrapolate the equations within this document for asymmetric S terms.

Source Image Manipulation

- **S is the horizontal Pipe Source Size without S_{excess} pixels (one-based)**
- **$S_{\text{excess}} \geq (\text{NUMTAPS} - 1)/2$**
 - **NUMTAPS = 7**
 - **Round up the result to the nearest NUMPPC (NUMPPC=2)**
- **If the sum of the Plane Position (PP) and the Full H. Plane Size (HPS) is greater than S (i.e. (PP + HPS) > S):**
 - **HPS = Left Plane Size (LPS) + Right Plane Size (RPS)**
 - **LPS = S - PP**
 - **RPS = HPS - (S - PP)**

Note that the "Full H. Plane Size" is the size of the Plane image after rotation.

The following table summarizes how the Plane offset, position, and size programming changes with rotation. Some notes:

1. The programming reflected in this table is only applicable for Plane images that straddle the seam (i.e. (PO + HPS) > S)
2. The Plane offset programming in the table is not accounting for any image cropping that software may be applying to the image in memory
3. Software should ensure that the horizontal Plane position for the image in the right Pipe is zero (i.e. there should be no positional offset of the image with respect to the seam)
4. As the diagrams above illustrate, depending on rotation the LPS and RPS terms are either based off the horizontal or vertical size of the image in memory. The terms will be denoted with "h" or "v" subscripts to differentiate which size the term is based off.

Rotation	Side	Register	Field	Programming
0	Left	PLANE_OFFSET	Starting X Pos	0
			Starting Y Pos	0
		PLANE_SIZE	Width	$LPS_h + S_{\text{excess}}$
			Height	V. Size

	Right	PLANE_OFFSET	Starting X Pos	$LPS_h - S_{excess}$
			Starting Y Pos	0
		PLANE_SIZE	Width	$RPS_h + S_{excess}$
			Height	V. Size
90	Left	PLANE_OFFSET	Starting X Pos	0
			Starting Y Pos	0
		PLANE_SIZE	Width	H. Size
			Height	$LPS_v + S_{excess}$
	Right	PLANE_OFFSET	Starting X Pos	0
			Starting Y Pos	$LPS_v - S_{excess}$
		PLANE_SIZE	Width	H. Size
			Height	$RPS_v + S_{excess}$
180	Left	PLANE_OFFSET	Starting X Pos	$RPS_h - S_{excess}$
			Starting Y Pos	0
		PLANE_SIZE	Width	$LPS_h + S_{excess}$
			Height	V. Size
	Right	PLANE_OFFSET	Starting X Pos	0
			Starting Y Pos	0
		PLANE_SIZE	Width	$RPS_h + S_{excess}$
			Height	V. Size
270	Left	PLANE_OFFSET	Starting X Pos	0
			Starting Y Pos	$RPS_v - S_{excess}$
		PLANE_SIZE	Width	H. Size
			Height	$LPS_v + S_{excess}$

	Right	PLANE_OFFSET	Starting X Pos	0
			Starting Y Pos	0
	PLANE_SIZE	Width	H. Size	
		Height	$RPS_v + S_{excess}$	

Cursor

Global Cursor Position	Left Pipe Cursor	Right Pipe Cursor
$GCP < (S - S_{excess} - CS)$	Enabled	Disabled
$(S - S_{excess} - CS) \leq GCP < (S + S_{excess})$	Enabled	Enabled
$GCP \geq (S + S_{excess})$	Disabled	Enabled

Global Cursor Position	Left Pipe Cursor Position	Right Pipe Cursor Position
$GCP < (S - S_{excess} - CS)$	Magnitude = GCP Sign = SW determines	N/A (Disabled)
$(S - S_{excess} - CS) \leq GCP < (S - S_{excess})$	Magnitude = GCP Sign = Positive	Magnitude = $S - S_{excess} - GCP$ Sign = Negative
$(S - S_{excess}) \leq GCP < (S + S_{excess})$		Magnitude = $GCP - S - S_{excess}$ Sign = Positive
$GCP \geq (S + S_{excess})$	N/A (Disabled)	

Post-Excess

- **D is the number of non-excess destination pixels per Pipe (one-based)**
- **$SF_{target} = S/D$**
- **SF_{actual}**
 - **Scaler calculates Scale Factor: $(S + [Left, Right] S_{excess}) / (D + D_{excess})$**
 - **Programmed Scale Factor: SF_{target} (PS_PROG_HSCALE)**
 - **Hardware maintains the Scale Factor at a precision of 2.14. When the Scaler calculates the Scale Factor, it will round the result.**
- **$D_{excess} = ROUND(((S + Left S_{excess}) / SF_{target}) - D)$**
 - **Round up to the nearest NUMPPC**

Initial Phase

- $P_{right} = SF_{actual} * (D - D_{excess})$
- **Absolute H. Initial Phase = $| (S - S_{excess}) - P_{right} |$**
 - **Absolute HIP applied to left-side Scaler if $(S - S_{excess}) \geq P_{right}$**
 - **Absolute HIP applied to right-side Scaler if $(S - S_{excess}) < P_{right}$**

Impacts of Plane Scaling

If a Plane is being scaled across a seam, then the following variables will be impacted:

- The S_{excess} in the PLANE_SIZE will be unchanged
- The S_{excess} in the Pipe Source size will be equal to the D_{excess} from the Plane Scaler
- The D_{excess} at the outlet of the Pipe will be calculated based off the S_{excess} in the Pipe Source
- The Pipe Scaler's actual scale factor will be based off the S_{excess} in the Pipe Source and the D_{excess} at the outlet of the Pipe

Affected Registers

The following table summarizes the registers directly affected when multi-Pipe Scaling is being employed

Register	Field	Programming
PLANE_POS	X/Y Position	See above "Source_Image_Manipulation" section
PLANE_OFFSET	Start X/Y Position	See table above in "Source_Image_Manipulation"
PLANE_SIZE	Width/Height	See table above in "Source_Image_Manipulation"
PIPE_SRC_SIZE	H. Source Size	$S + S_{excess}$
PIPE_SEAM_EXCESS		Left Side: Right Excess Amount = D_{excess} Right Side: Left Excess Amount = D_{excess}
PS_WIN_SZ	X Size	$D + D_{excess}$
PS_HPHASE	RGB Initial H. Phase (Integer & Fraction)	Programming dependent on comparison of $(S - S_{excess})$ and P_{right} Note that the Initial H. Phase Trip bit should be set
PS_WIN_POS	X Position	Left Side: Left side border size, if a border is present Right Side: This should always be zero
TRANS_HTOTAL	H. Active	Left Side: $D +$ Left side border size (if border present) Right Side: $D +$ Right side border size (if border present)

Planes

Flips

Overview

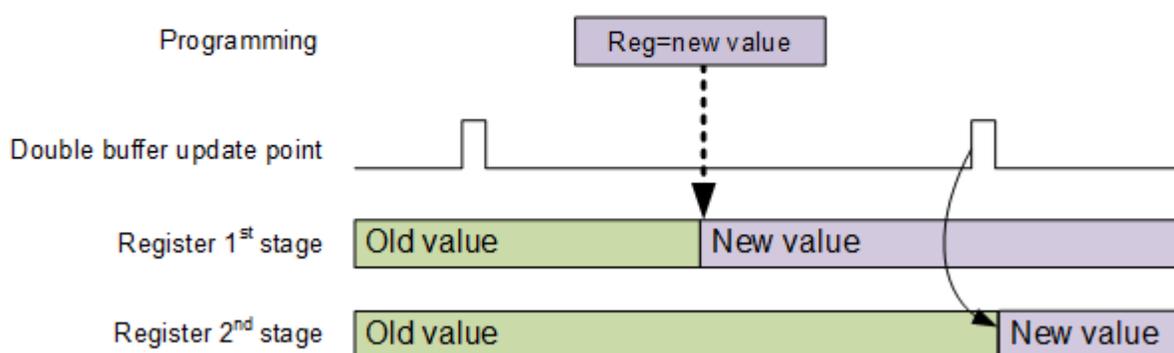
Display flips originally referred to switching between frame buffer surfaces to change the image. The term has evolved to refer more generally to an update to a plane configuration. Cursor configuration changes can also be considered to have flips, although it's more rare to describe a cursor update that way. Even updates to an enabled display pipe, such as new color correction settings, can be considered a flip, but usually only when combined with changes to the planes. A flip can be as small as updating one plane's surface address to a new frame buffer location, or as large as a synchronized update to planes, cursors, color correction, scaling, and transcoder metadata, across multiple running pipes.

Double Buffering

Most configuration that is allowed to be changed while a function is enabled, such as flips, is done through double buffered registers. Double buffer registers have two stages of registers to align the configuration update to a safe point, typically the start of vertical blank.

Writes and reads to double buffer registers access the first stage. The first stage is transferred to the second stage at the double buffer update point. The second stage is the "live value" that goes to the function being configured.

Basic Double Buffer Update



Some functions, such as planes, have read only registers to read the "live value" from the second stage.

Arming and double buffer stalling are used for some registers to control the double buffer update point for atomic updates across multiple registers and functions.

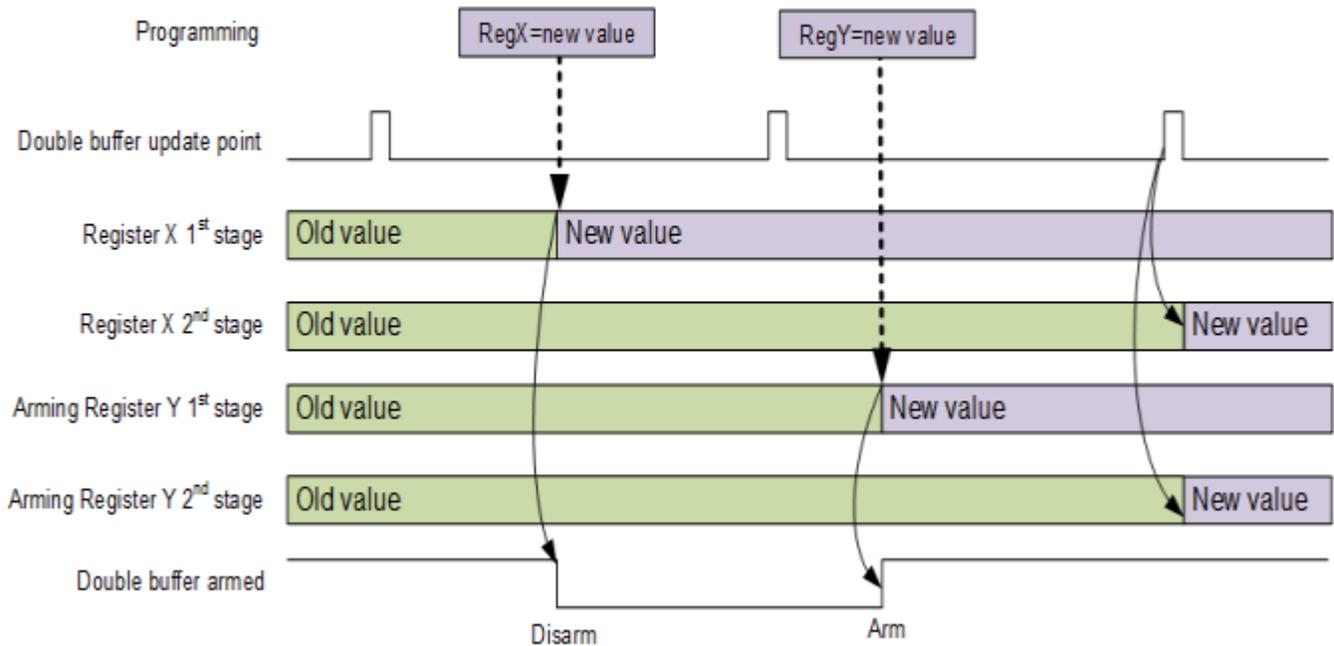
Atomic Updates

Updates to multiple registers of a single function can be done atomically using the arming mechanism that some functions, such as planes, use. One of the registers for the function is defined as the arming register, described in the double buffer fields of the registers. Writes to the arming register 1st stage will

arm the double buffer to update at the next double buffer update point. Writes to the other double buffered registers for that function will disarm and prevent updates at the double buffer point. The programming sequence is to write the disarming registers first and the arming register last, then all the registers update at the next double buffer update point.

Updates across multiple planes and other resources require double buffer stalling as explained in the Double Buffer Control section.

Armed Double Buffer Update



Note: Once armed, by writing to the armed by register, the registers controlled by this arming should not be changed until the double buffer update point is reached. If changed, this will disarm the sequence and will require another write to the armed by register to get it to the armed status again.

Flip Types

There are two broad types of flip; sync and async.

Synchronous Flips

Synchronous flips (sync flips) update the plane surface base address, and other double-buffered registers, at the frame boundary (start of next vertical blank). These can be used for updates across multiple functions, not just planes.

Asynchronous Flips

Asynchronous flips (async flips) update the plane surface base address as soon as possible (next TLB boundary reached or start of next vertical blank). This can cause the image to change mid-frame with a tear, but gives better responsiveness than sync flips.

Cursors cannot do async flips.

Asynchronous flip can update only the plane surface (PLANE_SURF).

Asynchronous flip completion time depends greatly on how much data has been prefetched for power savings, and can take up to 1 full frame to complete. For faster flip completion, disable FBC and render/unified compression and allocate a minimum amount of data buffer for the plane.

Flip Programming

There are two paths to programming a flip; MMIO and command ring.

MMIO Flips

MMIO writes may be used to flip.

General MMIO Sequence for Plane or Cursor Flips

1. If interrupt will be used to wait for the flip to complete, unmask and enable the flip done interrupt for the plane that will be flipped. See the Display Interrupts section. For cursors, use the vblank interrupt.
2. Write registers to update plane configuration fields besides the arming register, including PLANE_CTL Async Address Update Enable to select sync or async flip.
3. Arm the update: For planes, write PLANE_SURF with the plane surface address. For cursors, write CUR_BASE. For other resources, write the arming register as specified in registers for that resource. Some resources do not have arming.
4. Optional: Wait for the interrupt for the flipped plane. Not required if software uses other methods to find when the old surface is no longer being used.

Double buffering control does not apply to PLANE_SURF updates that occur when the plane is disabled so an interrupt event is generated immediately when the PLANE_SURF is written.

Command Ring Flips

The render command MI_DISPLAY_FLIP may be used to flip. Typically it is only used for plane surface address updates, but it can be extended to more complicated flips.

MI_DISPLAY_FLIP can be combined with LOAD_REGISTER_IMMEDIATE, which behaves like a MMIO write, to update other parts of a plane configuration.

General Command Sequence for Plane Flips

1. If MI_WAIT_FOR_EVENT will be used to wait for the flip to complete, unmask flip done for the plane that will be flipped in DE_RRMR using MMIO or LOAD_REGISTER_IMMEDIATE. See the Render Response section.
2. Optional: LOAD_REGISTER_IMMEDIATE to write registers to update plane configuration fields that are not included in the MI_DISPLAY_FLIP_COMMAND.
3. MI_DISPLAY_FLIP to update plane surface address and other configuration listed in the command details (stride, tile format, and more), and select sync or async flip. The flip is self arming.
4. Optional: MI_WAIT_FOR_EVENT to wait for the flip to complete for the flipped plane. Not required if software uses other methods to find when the old surface is no longer being used.
5. Re-mask the flip done in DE_RRMR using MMIO or LOAD_REGISTER_IMMEDIATE. It does not need to be immediately re-masked, but unmasked events will wake GT as they occur, so for improved power savings it is recommended to only unmask events as they are required.

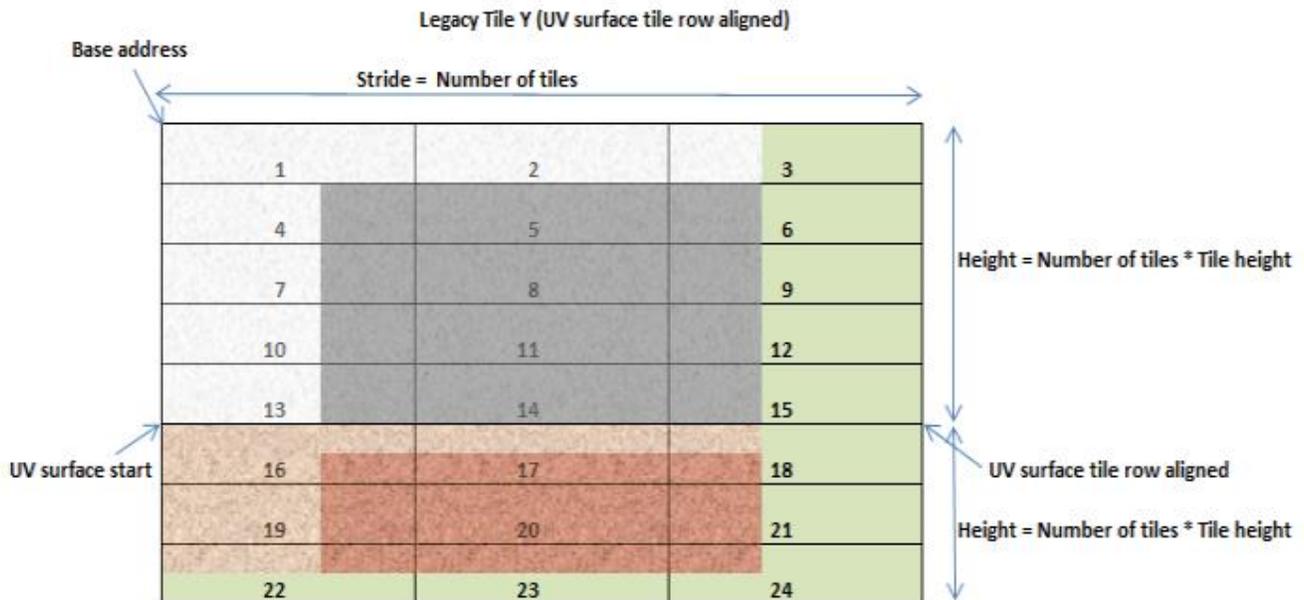
Flips to a plane that is disabled or on a disabled pipe will complete immediately and send the flip done.

Plane Planar YUV programming

YUV 420 hybrid planar formats (NV12, P0xx) have Y and UV surfaces stored separately. Display supports YUV 420 planar frame buffers in linear and tiled surface formats. Each display pipe can support up to two hybrid planar YUV 420 frame buffers. Display hardware uses two planes to handle each of these framebuffer - one for Y surface data and the other for UV surface. In addition to the two planes, a plane scaler must be enabled for YUV420 to YUV444 upsampling. See the Plane Capability and Interoperability section for UV and Y surface plane assignments.

Y and UV surface Base Address, Stride and Surface Offset (Start X and Y position) must be programmed separately for Y and UV planes. Software must make sure that the Y and UV plane programming gets applied together in the same frame. The Y and UV plane programming can be synchronized using the double buffer synchronization mechanism defined in DOUBLE_BUFFER_CTL. In the case of DisplayPort VRR mode (Variable Refresh Rate), the VRR push must be used for the Y and UV plane synchronization. Software must make sure that there is no push between the Y and UV plane flips.

With tiled surfaces, the UV surface must start on a new tile row.



Y Surface

Base Address, Stride and Surface Offset (Start X and Y position) should follow the standard plane programming.

UV Surface

The UV surface has some additional requirements.

Base Address:

For UV surfaces, the programmed Base Address should satisfy the following alignment requirements:

Linear:

The UV surface Base Address programmed in the PLANE_SURF register should be aligned to Stride in bytes of the Y surface * 64.

Tiled surfaces:

The start of the UV surface programmed in the PLANE_SURF register should be Tile Row Aligned.

Surface Offset:

The UV surface offset is programmed in the PLANE_OFFSET register. In the 0/180 rotate cases, the programmed X and Y start position should be half of the Y surface start X and Y positions.

For 0/180 rotate cases:

UV surface Start X Position = half of Y plane X start position

UV surface Start Y Position = half of Y plane Y start position

For 90/270 rotate cases:

UV surface Start X Position = (UV surface height in tiles * tile height) - UV plane Y start position (non-rotated) - UV plane height (non-rotated)

UV surface Start Y Position = UV plane Start X Position (non-rotated)

90/270 Rotation

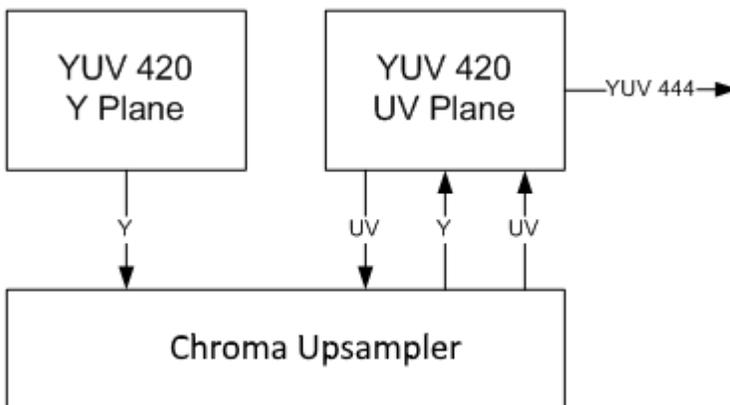
Plane 90/270 rotation support requires GTT remapping. GTT remapping allows the hardware to walk the pages sequentially. The Y and UV surfaces should be remapped separately, and the pages that contain portions of both Y and UV data will get GTT remapped twice, once for Y surface and the other for UV surface.

Upsampling and further processing

Planes 1-3 have dedicated chroma upsampler that is programmed in PLANE_CUS_CTL register.

The Y and UV plane pixels go through the chroma upsampler, gets upsampled, scaled and merged to form the YUV444 pixels for further processing.

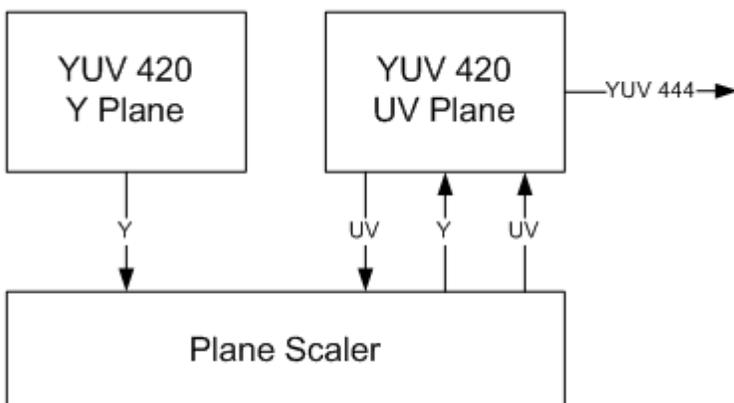
The Y plane binding is specified in the 'PLANE_CUS_CTL->Y Binding' field.



Planes 4-5 use the plane scaler for chroma upsampling.

The Y and UV plane pixels go through the plane scaler, gets upsampled, scaled and merged to form the YUV444 pixels for further processing.

The Y plane binding is specified in the 'PS_CTRL->Scaler Binding Y' field.



Destination keying, Color correction and Gamma must be programmed in the UV plane. Destination keying, Color correction and Gamma programming in the Y plane gets ignored. The UV plane position is used for blending order determination.

Flips

Since two different planes are involved in handling the pixels from a single planar YUV 420 frame buffer, we have to ensure that both the Y and UV planes' flip gets applied at the same time.

Global double buffer update disable mechanism can be used to synchronize and commit the flips atomically.

Follow the following sequence for CS, BCS or GUC flips

1. Use LRIs to set the 'Allow Double Buffer Update Disable' field for Y and UV planes.
2. Use LRI to set the 'Global Double Buffer Update Disable' field.
3. Use LRIs to update the 'Surface Base Address' and 'Stride' of the Y surface as required.
4. Send the flip messages for the UV surface.
5. Use LRI to clear the 'Global Double Buffer Update Disable' field.
6. Use LRIs to clear the 'Allow Double Buffer Update Disable' field for the Y and UV planes.

The flips will get applied in the following frame.

Follow the following sequence for MMIO flips

1. MMIOs to set the 'Allow Double Buffer Update Disable' field for Y and UV planes.
2. MMIO to set the 'Global Double Buffer Update Disable' field.
3. MMIOs to update the 'Surface Base Address' and 'Stride' of the Y surface as required.
4. MMIOs to update the 'Surface Base Address' and 'Stride' of the UV surface as required.
5. MMIO to clear the 'Global Double Buffer Update Disable' field.
6. MMIOs to clear the 'Allow Double Buffer Update Disable' field for the Y and UV planes.

The flips will get applied in the following frame.



Watermarks

Watermarks should be calculated and programmed for Y and UV planes separately. Refer to Watermarks page for further details.

Plane Capability and Interoperability

Plane Assignments and Capabilities

Plane capabilities:

- There are 7 planes + 1 cursor/pipe.
- Each planar YUV 420 surface uses 2 planes - Y and UV surfaces are handled by separate planes which gets combined later in the pipeline. Each pipe can support up to 2 YUV 420 planar surfaces.
- HDR mode supports up to 3 planes per pipe.

	Pipe A	Pipe B	Pipe C	Pipe D
Cursor	Cursor	Same as pipe A	Same as pipe A	Same as pipe A
Plane 7	Render Decompression Planar YUV 420 Y Surface	Same as pipe A	Same as pipe A	Same as pipe A
Plane 6	Render Decompression Planar YUV 420 Y Surface	Same as pipe A	Same as pipe A	Same as pipe A
Plane 5	Render Decompression Media Decompression Planar YUV 420 UV Surface OLED Compensation	Same as pipe A	Same as pipe A	Same as pipe A
Plane 4	Render Decompression Media Decompression Planar YUV 420 UV Surface	Same as pipe A	Same as pipe A	Same as pipe A
Plane 3	Render Decompression Media Decompression Planar YUV 420 UV Surface HDR	Same as pipe A	Same as pipe A	Same as pipe A

Plane 2	Render Decompression Media Decompression Planar YUV 420 UV Surface HDR	Same as pipe A	Same as pipe A	Same as pipe A
	Render Decompression Media Decompression Planar YUV 420 UV Surface HDR <u>Frame Buffer Compression</u>	Render Decompression Media Decompression Planar YUV 420 UV Surface HDR	Render Decompression Media Decompression Planar YUV 420 UV Surface HDR	Render Decompression Media Decompression Planar YUV 420 UV Surface HDR

Display Plane Mapping to Command Streamer Plane Number

Display Plane Name	Command Streamer Plane Number
Plane 1 A	Plane 1
Plane 1 B	Plane 2
Plane 1 C	Plane 3
Plane 2 A	Plane 4
Plane 2 B	Plane 5
Plane 2 C	Plane 6
Plane 3 A	Plane 7
Plane 3 B	Plane 8
Plane 3 C	Plane 9
Plane 4 A	Plane 10
Plane 4 B	Plane 11
Plane 4 C	Plane 12
Plane 5 A	Plane 13
Plane 5 B	Plane 14
Plane 5 C	Plane 15
Plane 6 A	Plane 16
Plane 6 B	Plane 17
Plane 6 C	Plane 18
Plane 7 A	Plane 19
Plane 7 B	Plane 20
Plane 7 C	Plane 21



Display Plane Name	Command Streamer Plane Number
Plane 1 D	Plane 22
Plane 2 D	Plane 23
Plane 3 D	Plane 24
Plane 4 D	Plane 25
Plane 5 D	Plane 26
Plane 6 D	Plane 27
Plane 7 D	Plane 28
N/A	Plane 29
N/A	Plane 30
N/A	Plane 31
N/A	Plane 32

Display Plane Mapping to Command Streamer Plane Number - Legacy Mode

Display Plane Name	Command Streamer Plane Number
Plane 1 A	Plane 1
Plane 1 B	Plane 2
Plane 1 C	Plane 3
Plane 2 A	Plane 4
Plane 2 B	Plane 5
Plane 2 C	Plane 6
Plane 3 A	Plane 7
Plane 3 B	Plane 8
Plane 3 C	Plane 9
Plane 4 A	Plane 10
Plane 4 B	Plane 11
Plane 4 C	Plane 12

Plane Feature Interoperability

Display Features / Surface Formats:

	RGB32 2:10:10 :10	RGB 32 8:8:8 :8	RGB 32-bit XR_BI AS 10:10: 10	RGB64 16:16:16 :16 FP16	RGB64 16:16:16 :16 UINT	RGB 16-bit 5:6:5	Index ed 8- bit	YUV4 44 8 bpc	YUV4 22 8 bpc	YUV4 20 - NV12	YUV4 20 - P0xx	YUV4 22 - Y210, Y212, Y216	YUV4 44 - Y410	YUV4 44 - Y412, Y416
Linear - 0/180 rotation	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Linear - 90/270 rotation														
X Tiling - 0/180 rotation	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
X Tiling - 90/270 rotation														
Y Tiling (legacy) - 0/180 rotation	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Y Tiling (legacy) - 90/270 rotation	✓	✓	✓			✓		✓	✓	✓	✓		✓	
Yf Tiling - 0/180 rotation														
Yf Tiling - 90/270 rotation														
Frame Buffer Compres sion		✓				✓ 90/270 rotatio n not suppor ted								
Render Compres sion	✓	✓		✓										
Media Compres sion	✓	✓		✓				✓	✓	✓	✓	✓	✓	✓

Plane Scaling	✓	✓		✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
Per-pixel Alpha	✓	✓	✓	✓	✓									
Color Keying (supported only in non-HDR mode)	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Horizontal Flip	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Async Flip	✓	✓	✓	✓	✓	✓								

Tiling Modes / Display Features:

	Rotation 0/180	Rotation 90/270	Horizontal Flip	Render Decompression	Media Compression	Frame Buffer Compression	Interlacing (Interlaced Fetch)
Linear	✓					✓	✓
X Tiling	✓		✓			✓	✓
Y Tiling Legacy	✓	✓ (some surface formats not supported)	✓	✓	✓	✓	
Yf Tiling							

Rotation Modes / Display Features:

	Interlacing	Frame Buffer Compression	Render Decompression	Media Compression
0/180 rotation	✓	✓	✓	✓
90/270 rotation		✓ RGB 16 bit format not supported		
Horizontal Flip	✓	✓	✓	✓

Render Decompression

GT Render engines uses a lossless scheme to compress the color Render targets. The goal of the compression is to reduce the memory bandwidth. The memory foot print increases slightly due to the need of the control surface.

- Decompression is supported with RGB8888, RGB1010102 and FP16 formats.

- Decompression is supported only with legacy tile Y or tile 4 surfaces.
- Decompression support is limited to left-right cache line pair mode. Top-bottom mode is not supported.
- Decompression is not supported with 90/270 degree rotation.
- Compressed displayable surfaces must be 16KB aligned and have pitches padded to multiple of 4 tiles.

Color Control Surface

The Color Control Surface (CCS) contains the compression status of the cache-line pairs. CCS stride is programmed separately independent of the main surface stride. The compression state of the cache-line pair is specified by 4 bits in the CCS. The address of CCS surface is specified as an offset from the start of the Render Target main surface.

Decompression Programming

When compressed Render targets are presented to Display, the display decompression must be enabled. Along with main surface programming, the following additional programming is required to enable the decompression

- **Decompression Enable**

Decompression is enabled by programming the 'Render Decomp' bit in the PLANE_CTL register.

- **Color Control Surface Distance**

The start of the CCS surface is programmed as the distance from the start of the main surface in the PLANE_AUX_DIST register. The CCS is always placed after the main surface and is 4K page aligned.

- **Clear color value**

When clear color is enabled in PLANE_CTL register, the clear color value must be programmed in the PLANE_CC_VAL register.

Media Decompression

The goal of the compression is to reduce the memory bandwidth. The memory foot print increases slightly due to the use of the control surface.

Display supports decompression of compressed media surfaces. 4 horizontally adjacent cache lines form a compression unit and its compression status is stored in the control surface (Tile Status Surface). The address of the control surface is specified as an offset from the start of the main surface. The control surface is always linear.

With Y tiling and planar YUV surfaces UV surface, the UV surface starts right after the Y surface and it can be in the middle on the tile row. The only guarantee is that the Y surface start will be 4 lines (cache line) aligned. Y and UV surfaces have their own individual control surface and they both have to be

programmed independently. In cases where the UV plane start is not aligned to the tile row start, the UV main surface start is programmed to the previous tile row aligned address with a plane offset to the start of the UV. In this case, the UV control surface will match the main surface start and hence will have the tile status for all compression units from the previously tile row aligned main surface address.

Restrictions
<ul style="list-style-type: none">• Only Y tiling is supported.• 90/270 rotation not supported.• Async flips not supported.

Decompression Programming

When compressed media targets are presented to display, the media decompression must be enabled. Along with main surface programming, the following additional programming is required for both Y and UV planes individually.

- **Decompression Enable**

Decompression is enabled by programming the 'Media Decomp' bit in the PLANE_CTL register.

- **Control Surface Distance**

The start of the control surface is programmed as the distance in bytes from the start of the main surface in the PLANE_AUX_DIST register. The control surface must always be placed after (not necessarily immediately) the main surface and must be 4K page aligned.

Control surface stride is not used for media compression.

Y and UV control surfaces must both be 4k aligned.



Plane Rotation Programming

This topic provides programming information for plane rotation.

The 180 rotation mode is unchanged and will continue to use the same programming. In the 180 rotation mode display hardware is responsible for walking the pages in the reverse order. The cacheline walk within the page is also reversed. The 90 and 270 rotation modes require more complicated page walk mechanism. The page walk is made transparent to the hardware by providing a different set of page translations (remapped GTT) for the same rendered surface. The remapping completely abstracts the page walk away from the hardware and the hardware walks the pages as if there is no rotation. Hardware is still responsible for handling the the walk within the page appropriately. Also, the 90, 270 Rotation requires a new parameter - surface height. The changes needed in the driver programming is discussed below.

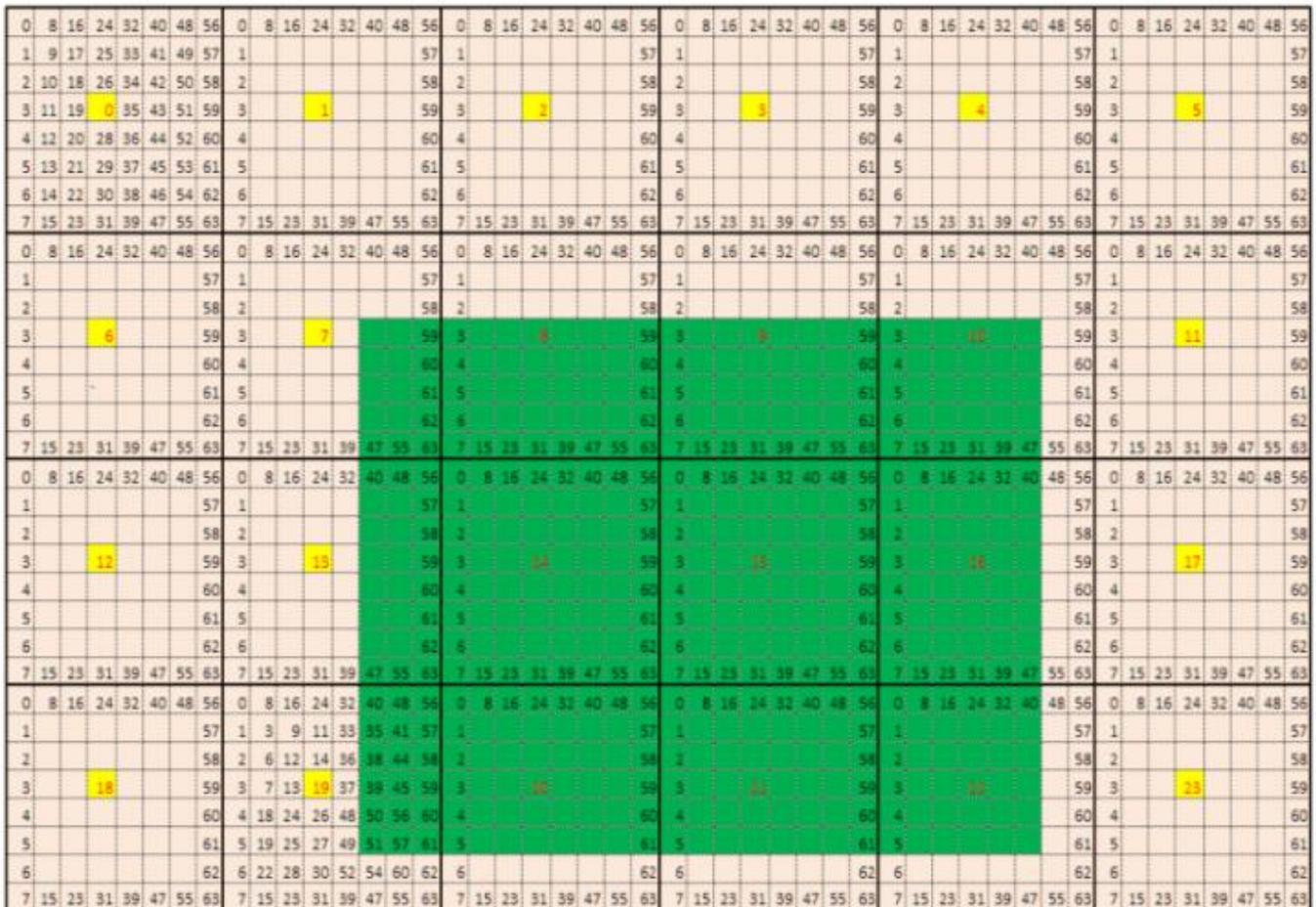
90 Rotation

For the 90 rotation programming, the plane parameters must use the following mapping

- Base address = New address (remapped GTT)

- Stride = Surface height in tiles
- Plane Width = Plane Height
- Plane Height = Plane Width
- X offset = (Surface height in tiles * tile height) - Plane Y offset - Plane Height [Note: the calculated X offset will always be ≥ 0 since Plane Y Offset + Plane Height \leq Surface Height in lines]
- Y offset = X offset
- Driver can enable the Scaler, as needed, to fit the rotated content in to the pipe active display area.

Sample frame buffer surface



Page access sequence in the rotated 90 mode

	8	16	24	32	40	48	56	0	8	16	24	32	40	48	56	0	8	16	24	32	40	48	56	0	8	16	24	32	40	48	56
1							57	1							57	1							57	1	9	17	25	33	41	49	57
2							58	2							58	2							58	2	10	18	26	34	42	50	58
3		18					59	3		12					59	3		8					59	3	11	19	0	35	43	51	59
4							60	4							60	4							60	4	12	20	28	36	44	52	60
5							61	5							61	5							61	5	13	21	29	37	45	53	61
6							62	6							62	6							62	6	14	22	30	38	46	54	62
7	15	23	31	39	47	55	63	7	15	23	31	39	47	55	63	7	15	23	31	39	47	55	63	7	15	23	31	39	47	55	63
0	8	16	24	32	40	48	56	0	8	16	24	32	40	48	56	0	8	16	24	32	40	48	56	0	8	16	24	32	40	48	56
1	3	9	11	33	35	41	57	1							57	1							57	1							57
2	6	12	14	36	38	44	58	2							58	2							58	2							58
3	7	13	19	37	39	45	59	3		13					59	3		7					59	3							59
4	18	24	26	48	50	56	60	4							60	4							60	4							60
5	19	25	27	49	51	57	61	5							61	5							61	5							61
6	22	28	30	52	54	60	62	6							62	6							62	6							62
7	15	23	31	39	47	55	63	7	15	23	31	39	47	55	63	7	15	23	31	39	47	55	63	7	15	23	31	39	47	55	63
0	8	16	24	32	40	48	56	0	8	16	24	32	40	48	56	0	8	16	24	32	40	48	56	0	8	16	24	32	40	48	56
1							57	1							57	1							57	1							57
2							58	2							58	2							58	2							58
3		20					59	3		14					59	3		8					59	3							59
4							60	4							60	4							60	4							60
5							61	5							61	5							61	5							61
6							62	6							62	6							62	6							62
7	15	23	31	39	47	55	63	7	15	23	31	39	47	55	63	7	15	23	31	39	47	55	63	7	15	23	31	39	47	55	63
0	8	16	24	32	40	48	56	0	8	16	24	32	40	48	56	0	8	16	24	32	40	48	56	0	8	16	24	32	40	48	56
1							57	1							57	1							57	1							57
2							58	2							58	2							58	2							58
3		21					59	3		15					59	3		9					59	3							59
4							60	4							60	4							60	4							60
5							61	5							61	5							61	5							61
6							62	6							62	6							62	6							62
7	15	23	31	39	47	55	63	7	15	23	31	39	47	55	63	7	15	23	31	39	47	55	63	7	15	23	31	39	47	55	63
0	8	16	24	32	40	48	56	0	8	16	24	32	40	48	56	0	8	16	24	32	40	48	56	0	8	16	24	32	40	48	56
1							57	1							57	1							57	1							57
2							58	2							58	2							58	2							58
3							59	3		16					59	3		10					59	3							59
4							60	4							60	4							60	4							60
5							61	5							61	5							61	5							61
6							62	6							62	6							62	6							62
7	15	23	31	39	47	55	63	7	15	23	31	39	47	55	63	7	15	23	31	39	47	55	63	7	15	23	31	39	47	55	63
0	8	16	24	32	40	48	56	0	8	16	24	32	40	48	56	0	8	16	24	32	40	48	56	0	8	16	24	32	40	48	56
1							57	1							57	1							57	1							57
2							58	2							58	2							58	2							58
3		22					59	3		17					59	3		11					59	3							59
4							60	4							60	4							60	4							60
5							61	5							61	5							61	5							61
6							62	6							62	6							62	6							62
7	15	23	31	39	47	55	63	7	15	23	31	39	47	55	63	7	15	23	31	39	47	55	63	7	15	23	31	39	47	55	63

Example

Let us assume the following display programming for a single pipe - single plane, Y tiled, non-rotated, 1920 x 1200, 4Bpp with the plane panned (100, 150), covering full active area and scaler not enabled.

GTT mapping

Here is a sample GTT mapping for 90 rotation mode.

Original GTT

Assumed Surface base = 0x200000

0x200000 = page 0

0x201000 = page 1

0x202000 = page 2

0x203000 = page 3

...

Remapped Display GTT - 90 rotation

Assumed new Surface base = 0x400000

0x400000 = page 18

0x401000 = page 12

0x402000 = page 6

0x403000 = page 0

Register programming for non-rotated scenario

- *PLANE_SURF*->Surface Base Address = 0x200000
- *PLANE_STRIDE*->Stride = 60 [(1920 * 4)/128] [(width *bpp)/tile width]
- *PLANE_SIZE*->Width = 1920
- *PLANE_SIZE*->Height = 1200
- *PLANE_OFFSET*->Start X Position = 100
- *PLANE_OFFSET*->Start Y Position = 150
- *Surface Height in tiles (assumed)* = 50 (allocated surface height in number of scan lines/tile height. For plane height = 1200, the surface height should be a minimum of 38 tiles (ceiling (1200/32)). When panning is used, the rendered frame buffer surface will be larger than the plane size. Here, let us assume that the rendered surface height in tiles = 50).

The programming changes to following for a 90 rotation scenario

- *PLANE_SURF*->Surface Base Address = 0x400000 [uses remapped GTT]
- *PLANE_STRIDE*->Stride = 50 [Surface height in tiles (assumed earlier)]
- *PLANE_SIZE*->Width = 1200 [non-rotated Height]
- *PLANE_SIZE*->Height = 1920 [non-rotated Width]
- *PLANE_OFFSET*->Start X Position = 250 [(50*32)-150-1200] [(Surface height * tile height) - non rotated Y position - non rotated Height]
- *PLANE_OFFSET*->Start Y Position = 100 [non-rotated X position]

The scaler should be programmed appropriately to fit the rotated plane in the pipe active area and the window position should be adjusted if it is desired to maintain the same apparent position on a physically rotated display.

270 rotation

Uses the same GTT remapping and register programming as 90 rotation mode with the Plane control register rotation mode set as 270.

NV12 (YUV 420) rotation

For NV12 90/270 rotation, the Y and UV surfaces should be treated as separate surfaces and thus the GTT remapping for rotation should be done separately.

Display Buffer Programming

Display Buffer Allocation

Allocation of the display buffer is programmable for each display plane, using the buffer start and buffer end values in **PLANE_BUF_CFG**.

Proper display buffer allocation is important for Display hardware to function correctly. Optimal allocation provides better display residency in memory low power modes. Display Buffer allocation must be recalculated and programmed when pipes/planes get enabled or disabled.

Display Buffer Size

Buffer Count	Total Buffer Size	Total Buffer Blocks	Blocks Available for Programming
2	1 MB (512 KB per buffer)	2048 (1024 per buffer)	2048

Each display buffer block is 8 cache lines.

Allocation Requirements

Allocation must not overlap between any enabled planes.

A minimum allocation is required for any enabled plane. See Minimum Allocation Requirements below.

A gap between allocation for enabled planes is allowed.

The allocation for enabled planes should be as large as possible to allow for higher watermarks and better residency in memory power saving modes.

Minimum Allocation Requirements

Allocation for each enabled plane must meet these minimum requirements.

Planes using Linear or X tiled memory formats must allocate a minimum of 8 blocks.

Planes using Y tiled memory formats must allocate blocks for a minimum number of scanlines worth of data. The formula and table of minimum scanlines is below.

Y tiled minimum allocation = Ceiling $[(4 * \text{Plane source width} * \text{Plane Bpp})/512] * \text{MinScanLines}/4 + 3$



Plane Bpp	Minimum Scanlines for Y Tile	
	0/180 Rotation	90/270 Rotation
1	8	32
2	8	16
4	8	8
8	8	4

YUV 420 Planar Surface Format	Bpp
NV12	1 Bpp for Y and 2 Bpp for UV
P0xx	2 Bpp for Y and 4 Bpp for UV

Multi-Buffer Enabling and Allocation Requirements

There are two display buffers DBUF_S1 and DBUF_S2.

Display Buffer	Buffer Start	Buffer End
DBUF_S1	0	1023
DBUF_S2	1024	2047

Enable DBUF_S1 with the display initialization sequence so it will be always available for VGA and backwards compatibility. Enable DBUF_S2 when any planes are allocated to it as per the following table.

The table ensures that pipes are using the closest DBUF when there are multiple pipes enabled.

Pipe and DBUF ordering: PipeD - DBUF_S2 - PipeC - PipeA - DBUF_S1 - PipeB

When a pipe is allowed to allocate from 2 DBUFs, a plane on that pipe may use allocation that straddles the 2 DBUFs.

Pipe D Planes DBUF Allocation	Pipe C Planes DBUF Allocation	Pipe A Planes DBUF Allocation	Pipe B Planes DBUF Allocation	Pipes with Enabled Planes
N/A	N/A	N/A	N/A	None or VGA
N/A	N/A	S1+S2	N/A	A
N/A	N/A	N/A	S1+S2	B
N/A	N/A	S2	S1	A+B
N/A	S1+S2	N/A	N/A	C
N/A	S2	S1	N/A	A+C
N/A	S2	N/A	S1	B+C
N/A	S2	S1	S1	A+B+C
S1+S2	N/A	N/A	N/A	D
S2	N/A	S1	N/A	A+D
S2	N/A	N/A	S1	B+D
S2	N/A	S1	S1	A+B+D
S2	S1	N/A	N/A	C+D

Pipe D Planes DBUF Allocation	Pipe C Planes DBUF Allocation	Pipe A Planes DBUF Allocation	Pipe B Planes DBUF Allocation	Pipes with Enabled Planes
S2	S2	S1	N/A	A+C+D
S2	S2	N/A	S1	B+C+D
S2	S2	S1	S1	A+B+C+D

Basic Allocation Method

These are basic methods that can be used for single and multi-pipe modes. For optimal power usage, the display driver can choose to use more advanced allocation techniques as desired.

Example Method 1:

Single Pipe

Enable display buffer(s). Refer to DBUF_CTL register for display buffer enabling.

If only DBUF_S1 is enabled

$$\text{TotalBlocksAvailable} = 1024$$

else if both DBUF_S1, DBUF_S2 enabled

$$\text{TotalBlocksAvailable} = 2048$$

Allocate a fixed number of blocks to cursors and then allocate the remaining blocks among planes, based on each plane's data rate.

$$\text{BlocksAvailable} = \text{TotalBlocksAvailable}$$

1. Allocate 32 blocks for cursor

The driver frequently enables and disables the cursor or changes the cursor pixel format. Fixed allocation is preferred for cursor to minimize the buffer re-allocation. More allocation might be required to support deeper low power states (based on the results of watermark calculations).

$$\text{CursorBufAlloc} = 32$$

$$\text{BlocksAvailable} = \text{BlocksAvailable} - 32$$

2. Check for minimum buffer requirement

For each enabled plane

If Y tiled

MinScanLines = Look up minimum scanlines needed from the table

$$\text{PlaneMinAlloc} = \text{Ceiling} [(4 * \text{Plane width} * \text{Bpp}) / 512] * \text{MinScanLines} / 4 + 3$$

Else

$$\text{PlaneMinAlloc} = 8$$

If sum of PlaneMinAlloc > BlocksAvailable

Enable DBUF_S2

BlocksAvailable = 2048 - CursorBufAlloc

If sum of PlaneMinAlloc > BlocksAvailable

Error - Display Mode can't be supported.

The driver can change the number of enabled planes or the plane configuration and rerun the algorithm.

3. Calculate Relative Data Rate for planes

In this step the driver may want to use the expected maximum plane source sizes so it does not have to reallocate for a plane that is changing size.

For each enabled plane

If PlaneScalerEnabled

*PlaneScaleFactor = (Plane width/Scaler window X size) * (Plane height/Scaler window Y size)*

Else

PlaneScaleFactor = 1

*PlaneRelativeDataRate = Plane height * Plane width * plane source bytes per pixel * PlaneScaleFactor*

4. Allocate blocks for enabled planes as per the Data rate ratio.

For each plane that needs allocation (PlaneBlockAllocFinal == false)

*PlaneBufAlloc = floor (BlocksAvailable * PlaneRelativeDataRate/Sum of PlaneRelativeDataRate of all planes that need allocation).*

**floor - rounds down to an integer value dropping the fractional part.*

5. Adjust for minimum allocation requirement

AdjustmentRequired = false

For each plane needs allocation (PlaneBlockAllocFinal == false)

If PlaneBufAlloc < PlaneMinAlloc

AdjustmentRequired = true

PlaneBufAlloc = PlaneMinAlloc

PlaneBlockAllocFinal = true

BlocksAvailable = BlocksAvailable - PlaneMinAlloc

If AdjustmentRequired = true

Go back to step 4

Multi-Pipe Option 1:

Enable both display buffers DBUF_S1 and DBUF_S2.

$TotalBlocksAvailable = 2048$

$NumPipes = Total\ number\ of\ display\ pipes\ in\ the\ hardware$

Allocate a fixed number of blocks to cursors, allocate $1/NumPipes$ of the remaining blocks to each pipe, then calculate each pipe individually as in the Single Pipe case.

1. Allocate 8 blocks for cursor per pipe

The driver frequently enables and disables the cursor or changes the cursor pixel format. Fixed allocation is preferred for cursor to minimize the buffer re-allocation. More allocation might be required to support deeper low power states (based on the results of watermark calculations).

For each enabled cursor

$CursorBufAlloc = 8$

$BlocksAvailable = TotalBlocksAvailable - (8 * NumPipes)$

2. Distribute the blocks equally among the pipes

$BlocksAvailable = BlocksAvailable/NumPipes$

3. Assign blocks to the planes

For each pipe

Perform Single Pipe sequence, starting from step 2.

Multi-Pipe Option 2:

Allocate a fixed number of blocks to cursors, use the Single Pipe case for all the other planes across all pipes.

Example Method 2:

This allocation is based on the Watermark calculations and helps to distribute the buffer more optimally to achieve consistent latency levels supported across all planes.

- For each enabled plane, calculate buffer allocation needed for all Latency levels 1 to 7.
- Calculate the total buffer allocation needed for each latency level by adding the individual allocation of all enabled planes.
- Choose the max latency level that can be supported with the available display buffer. For each enabled plane, program/enable all watermarks up to that latency level.
- Allocate the buffer to the planes as required by the latency level chosen.

Use method 1 to allocate any remaining buffer.



Buffer Allocation Re-distribution

When an additional pipe is getting enabled, or an existing pipe requires more buffer to support a new mode or is disabled, buffer reallocation may be necessary for proper display functionality. Whenever a portion of the allocated buffer is taken away from one pipe and allocated to a different pipe, the following sequence should be followed to make sure that there are no buffer allocation overlaps at any point of time.

1. *For each pipe whose allocation is reduced*
 - a. *Program the new buffer allocation.*
 - b. *Wait for VBlank of that pipe for new allocation to update.*
2. *For each pipe whose allocation is increased*
 - a. *Program the new buffer allocation.*
 - b. *Wait for VBlank of that pipe for new allocation to update.*

Display Buffer Allocation and Watermark Programming Prior to OS Boot

Basic programming of the display buffer and watermarks to allow limited display usage prior to OS boot: This will not allow package power saving states.

Supported usages:

- Up to 4 pipes enabled at once
- Up to one universal plane enabled per pipe. No cursor.
- Linear or Xtile memory
- Any RGB frame buffer pixel format 32bpp or less, without compression
- Any supported screen resolution
- Downscaling less than or equal to 12.5%

Enable DBUFs following the Multi-Buffer Enabling and Allocation Requirements.

Allocate 160 blocks per pipe

- Pipe A: 0-159, Pipe B: 160-319, Pipe C: 320-479, Pipe D: 480-639
- PLANE_BUF_CFG_<plane number>_A = 0x009F0000
- PLANE_BUF_CFG_<plane number>_B = 0x013F00A0
- PLANE_BUF_CFG_<plane number>_C = 0x01DF0140
- PLANE_BUF_CFG_<plane number>_D = 0x027F01E0

Set level 0 watermarks for any enabled plane to 160 blocks and 2 lines.

- PLANE_WM_<plane number>_<pipe>_0 = 0x800080A0

The higher level watermarks for any enabled plane must have bit 31=0 to keep the low power watermarks disabled.

VGA

The VGA Control register is located here. The VGA I/O registers are located in the VGA Registers document.

Register
VGA_CONTROL

Cursor Plane

Planes
CUR_CTL
CUR_BASE
CUR_POS
CUR_PAL
CUR_FBC_CTL
CUR_SURFLIVE
PLANE_BUF_CFG
PLANE_WM
DPRC_INSTANCES
CUR_COLOR_CTL
CUR_CSC_COEFF
CUR_PRE_CSC_GAMC_INDEX
CUR_PRE_CSC_GAMC_DATA

The CUR_CTL and CUR_FBC_CTL active registers will be updated on the vertical blank or when pipe is disabled, after the CUR_BASE register is written, or when cursor is not yet enabled, providing an atomic update of those registers together with the CUR_BASE register.

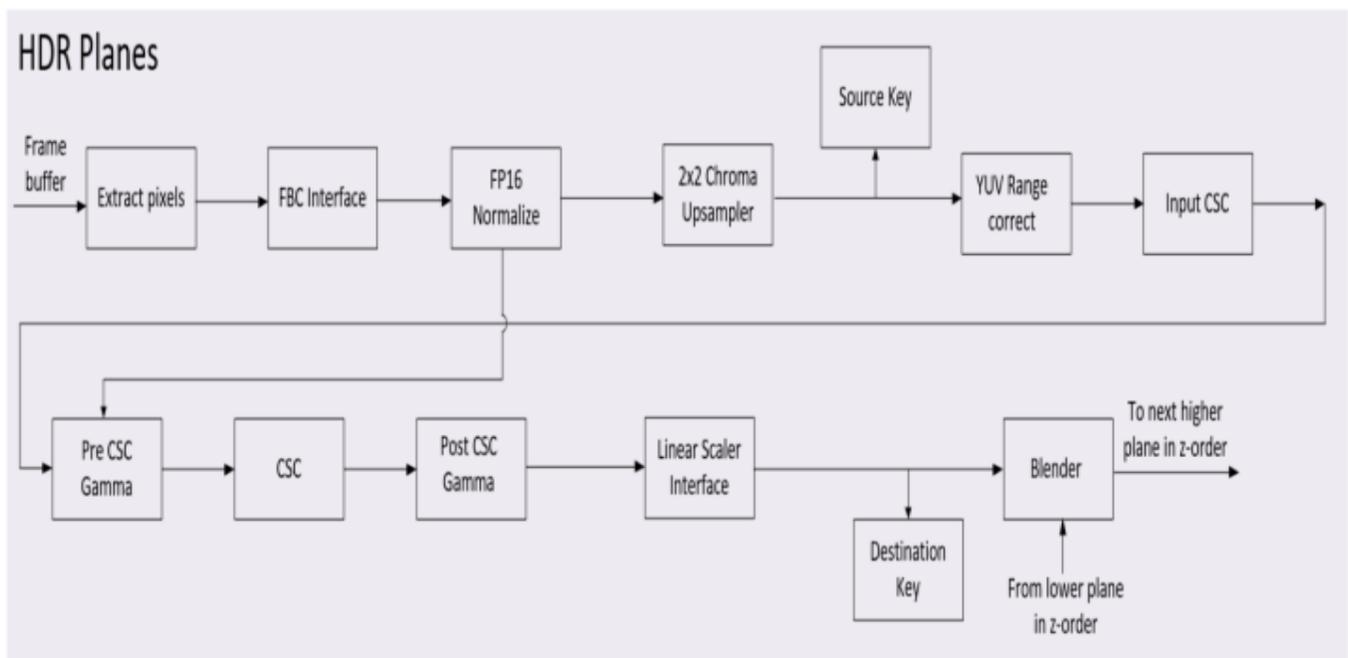
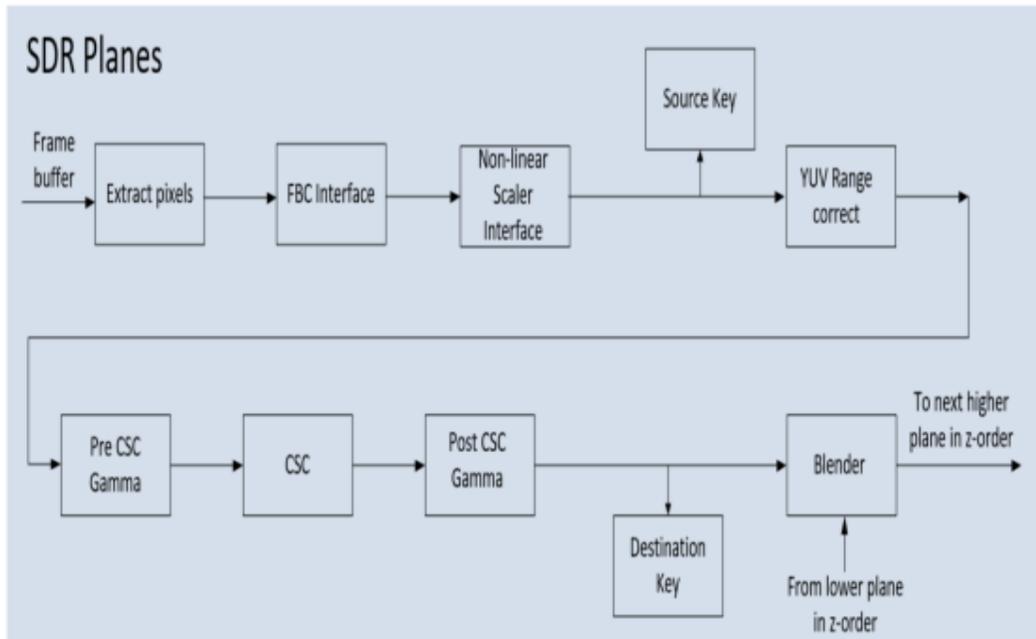
Universal Plane

Planes
PLANE_CTL
PLANE_STRIDE
PLANE_POS
PLANE_SIZE
PLANE_SURF
PLANE_LEFT_SURF
PLANE_AUX_DIST
PLANE_SURFLIVE
PLANE_WM
PLANE_BUF_CFG
PLANE_OFFSET
PLANE_KEYVAL

Planes
PLANE_KEYMSK
PLANE_KEYMAX
PLANE_PRE_CSC_GAMC_INDEX
PLANE_PRE_CSC_GAMC_DATA
PLANE_POST_CSC_GAMC_INDEX
PLANE_POST_CSC_GAMC_DATA
PLANE_COLOR_CTL
PLANE_PIXEL_NORMALIZE
PLANE_INPUT_CSC_COEFF
PLANE_INPUT_CSC_PREOFF
PLANE_INPUT_CSC_POSTOFF
PLANE_CSC_COEFF
PLANE_CSC_PREOFF
PLANE_CSC_POSTOFF
PLANE_PRE_CSC_GAMC_DATA_ENH
PLANE_PRE_CSC_GAMC_INDEX_ENH
PLANE_POST_CSC_GAMC_DATA_ENH
PLANE_POST_CSC_GAMC_INDEX_ENH
PLANE_CUS_CTL
PLANE_CC_VAL
PLANE_POST_CSC_GAMC_SEGO_DATA_ENH
PLANE_POST_CSC_GAMC_SEGO_INDEX_ENH
SEL_FETCH_PLANE_CTL

Many of the plane control active registers will be updated on the vertical blank or when pipe is disabled, after the surface base address register is written, or when the plane is not yet enabled, providing an atomic update of those registers together with the surface base address register.

Data Flow Through the Plane



FBC not on all planes or pipes.

HDR planes have higher bit precision throughout the processing stages.

PIPE_MISC HDR Mode must be enabled to get the higher precision output from the HDR planes, bypassing the SDR planes in blending.

Planes 1-3 are the HDR planes. The other planes are SDR planes.



Plane Pixel Formats

ARGB

Name	Alpha	Red	Green	Blue
RGB 32-bit 8:8:8:8 BGRA	31:24	23:16	15:8	7:0
RGB 32-bit 8:8:8:8 RGBA	31:24	7:0	15:8	23:16
RGB 32-bit 2:10:10:10 BGRA	31:30	29:20	19:10	9:0
RGB 32-bit 2:10:10:10 RGBA	31:30	9:0	19:10	29:20
RGB 64-bit 16:16:16:16 Float BGRA (FP16) Each component is 1:5:10 MSb-sign:exponent:fraction	63:48	47:32	31:16	15:0
RGB 64-bit 16:16:16:16 Float RGBA (FP16) Each component is 1:5:10 MSb-sign:exponent:fraction	63:48	15:0	31:16	47:32
RGB 64-bit 16:16:16:16 UINT BGRA Each component is 16 bit unsigned integer	63:56	47:32	31:16	15:0
RGB 64-bit 16:16:16:16 UINT RGBA Each component is 16 bit unsigned integer	63:56	15:0	31:16	47:32
RGB 32-bit XR_BIAS 2:10:10:10	31:30	9:0	19:10	29:20
16-bit BGR 5:6:5	N/A	15:11	10:5	4:0
64-bit formats supported only on the HDR planes.				

YUV 420 Planar

Name	Y	U	V
YUV 4:2:0 8 bpc - NV12	7:0	15:8	7:0
YUV 4:2:0 10 bpc - P010	15:6	31:22	15:6
YUV 4:2:0 12 bpc - P012	15:4	31:20	15:4
YUV 4:2:0 16 bpc - P016	15:0	31:16	15:0

YUV 422 Packed

Name	Y1	U	Y2	V
YUV 4:2:2 YUYV 8 bpc	7:0	15:8	23:16	31:24
YUV 4:2:2 UYVY 8 bpc	15:8	7:0	31:24	23:16
YUV 4:2:2 YVYU 8 bpc	7:0	31:24	23:16	15:8
YUV 4:2:2 VYUY 8 bpc	15:8	23:16	31:24	7:0
YUV 4:2:2 YUYV 10 bpc - Y210	15:6	31:22	47:38	63:54
YUV 4:2:2 YUYV 12 bpc - Y212	15:4	31:20	47:36	63:52
YUV 4:2:2 YUYV 16 bpc - Y216	15:0	31:16	47:32	63:48

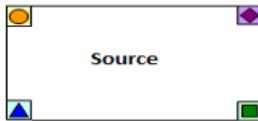
YUV 444 Packed

Name	Ignored	Y	U	V

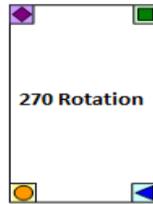
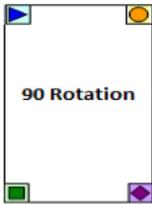
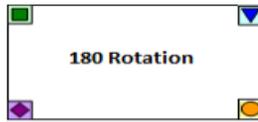
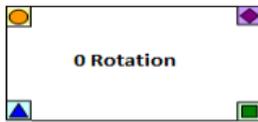
Name	Ignored	Y	U	V
YUV 4:4:4 8 bpc	31:24	23:16	15:8	7:0
YUV 4:4:4 10 bpc - Y410	31:30	19:10	9:0	29:20
YUV 4:4:4 12 bpc - Y412	63:52	31:20	15:4	47:36
YUV 4:4:4 16 bpc - Y416	63:48	31:16	15:0	47:32

Horizontal Flip

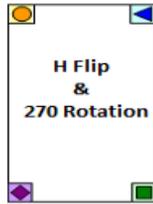
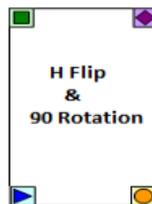
When plane horizontal Flip is enabled with rotation, the horizontal flip operation is logically performed first followed by the rotation operation. The sample results are shown below.



Rotation without Horizontal Flip



Rotation with Horizontal Flip





DSC

DSC Source Policies

Port Type	Source Policy
eDP	DSC output BPP is as defined in DPCD. If eDP Sink reports capability for multiple link rates, choose the optimal link rate based on bandwidth calculations.
DP	DSC is enabled only when link bandwidth requires it. DSC output BPP is set to the highest possible value within link bandwidth.
HDMI	DSC BPP and link bandwidth are selected as per the Specification (VIC).

Transport mandatory DSC features

The following features are options at DSC level but are mandatory at transport level.

Design requirements

Design parameter	RGB/YUV 444	YUV 422	YUV 420
slice width	no restriction	multiple of 2	multiple of 2
	minimum = 18	minimum = 36	minimum = 36
slice height	no restriction	no restriction	multiple of 2
Pixels in Slice	Must be >= 15000	Must be >= 30000	Must be >= 30000

VDSC compression engines do not support interlaced mode of operation for any transport protocol.

Below are the list of DSC encoder specific registers in Bspec. There are 2 copies of these registers, one for each VDSC engine.

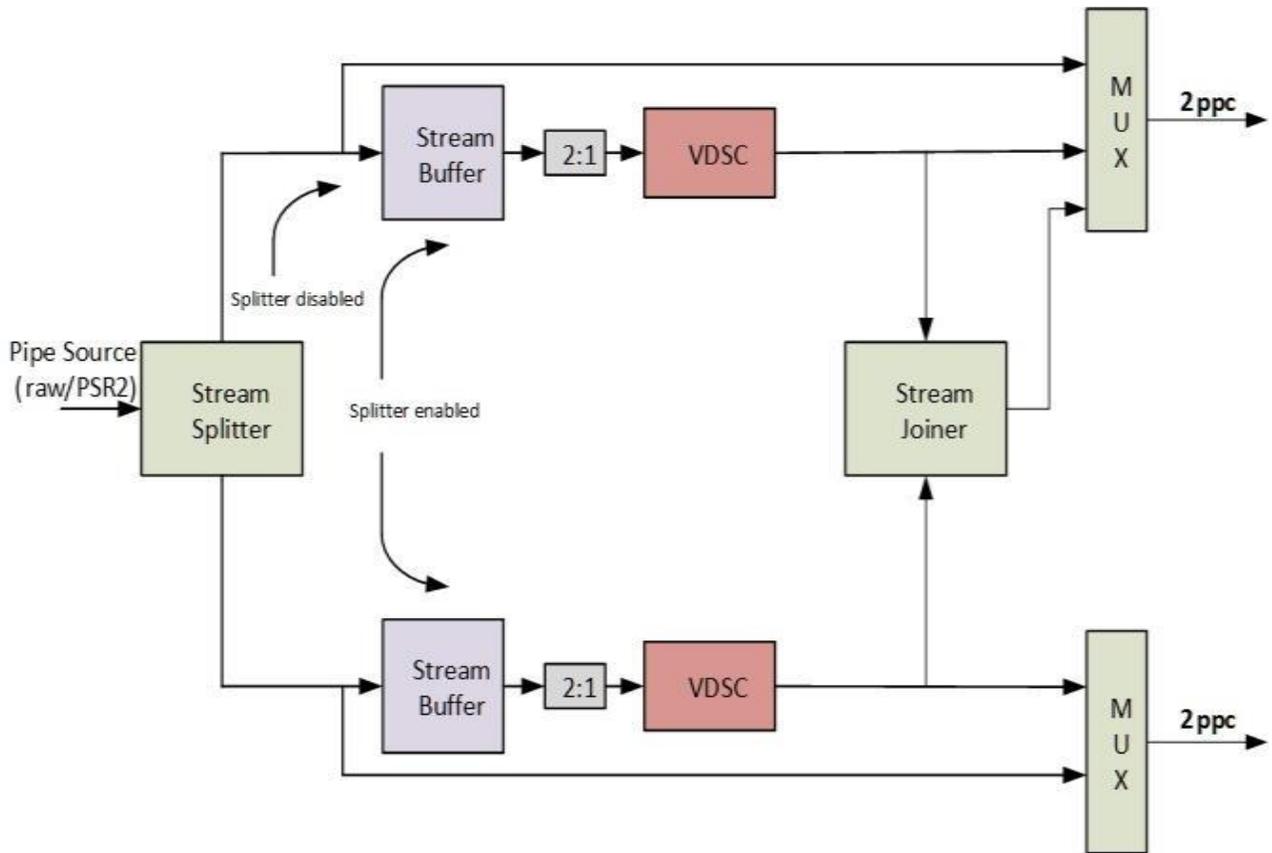
Register List
PIPE_DSS_CTL1
PIPE_DSS_CTL2
DSC_PICTURE_PARAMETER_SET_0 Field vbr_enable setting as "1" is not supported.
Set 0 Field "Allow double buffer update disable" is not from DSC standard.
DSC_PICTURE_PARAMETER_SET_1
DSC_PICTURE_PARAMETER_SET_2
DSC_PICTURE_PARAMETER_SET_3
DSC_PICTURE_PARAMETER_SET_4

Register List
DSC_PICTURE_PARAMETER_SET_5
DSC_PICTURE_PARAMETER_SET_6
DSC_PICTURE_PARAMETER_SET_7
DSC_PICTURE_PARAMETER_SET_8
DSC_PICTURE_PARAMETER_SET_9
DSC_PICTURE_PARAMETER_SET_10
DSC_PICTURE_PARAMETER_SET_11
DSC_RC_BUF_THRESH_0
DSC_RC_BUF_THRESH_1
DSC_RC_RANGE_PARAMETERS_0
DSC_RC_RANGE_PARAMETERS_1
DSC_RC_RANGE_PARAMETERS_2
DSC_RC_RANGE_PARAMETERS_3
DSC_PICTURE_PARAMETER_SET_12
DSC_PICTURE_PARAMETER_SET_13
DSC_PICTURE_PARAMETER_SET_14
DSC_PICTURE_PARAMETER_SET_15
DSC_PICTURE_PARAMETER_SET_16
Fields "slice_row_per_frame" and "slice_per_line" are not from DSC standard.

VDSC

VDSC feature compresses the raw pixel bytes into compressed byte stream as per VESA DSC specification. The pixel stream can be injected into VDSC as shown below at any of the 3 different input ports. In a typical use case, VDSC can split the pipe frame into left/right (front/back in MIPI) halves and uses 2 VDSC engines to compress each half and then joins the compressed byte streams from 2 VDSC engines into a single compressed byte stream. Each VDSC engine is configured with its own PPS register set.

Each VDSC operates with 1 ppc throughput and this becomes a limitation when the pixel clock is higher than the VDSC clock (cdclk). These cases are supported by running 2 VDSC engines in parallel, effectively realizing 2ppc throughput.



The following datapath options are supported in the VESA based compression engine. The top mux output is designated as left (front) and bottom mux output is designated as right (back) in the case of split streams. Stream joiner output can only feed into the top mux output. If the input frame is divided into an even number of slices (say, N) by enabling the splitter, then the corresponding branch parameters such as picture width and slice width need to be adjusted accordingly. For example, if the input frame (pipe source in the diagram above) is divided into 4 slices per scanline then the slice width on each branch will be $HACTIVE/4$ and the picture width on each branch will be $HACTIVE/2$.

Compression Engine Modes of Operation

Mode	Splitter Enabled	VDSC Enabled	Joiner Enabled	Notes
Bypass mode	No	No	No	Non-PSR2 stream eDP PSR2: same as bypass except for input source to DSC engine Output on left VDSC branch only.
VDSC enabled mode	No	Yes	No	MIPI: see the MIPI table below Pixel rate is limited w/o splitter, see resolution support page Output on left VDSC branch only.
Split mode	Yes	No	No	Required for eDP CoG/MSO without VDSC Required for MIPI dual link (front/back and interleaving) w/o VDSC Output on left and right VDSC branches.
Split/Compress mode	Yes	Yes	No	Required for eDP CoG/MSO with VDSC Required for MIPI dual link with VDSC Output on left and right VDSC branches.
Joiner enabled mode (small joiner)	Yes	Yes	Yes	Cannot work with CoG Output on left VDSC branch only.

Big Joiner

The above DSC engine can achieve compression over multiple pipes to compress 8K image size as shown below. A frame can be divided and processed by 2 adjacent pipes, compressed by their associated VDSC engines, then the compressed pixels streams from 2 pipes will be joined into a single compress pixel stream. In this 8K scenario the frame is compressed by a total of 4 VDSC engines in parallel. We have separate PPS DIP packet defined on per transcoder basis.

In the case of big joiner mode, video timings are determined by the attached transcoder.

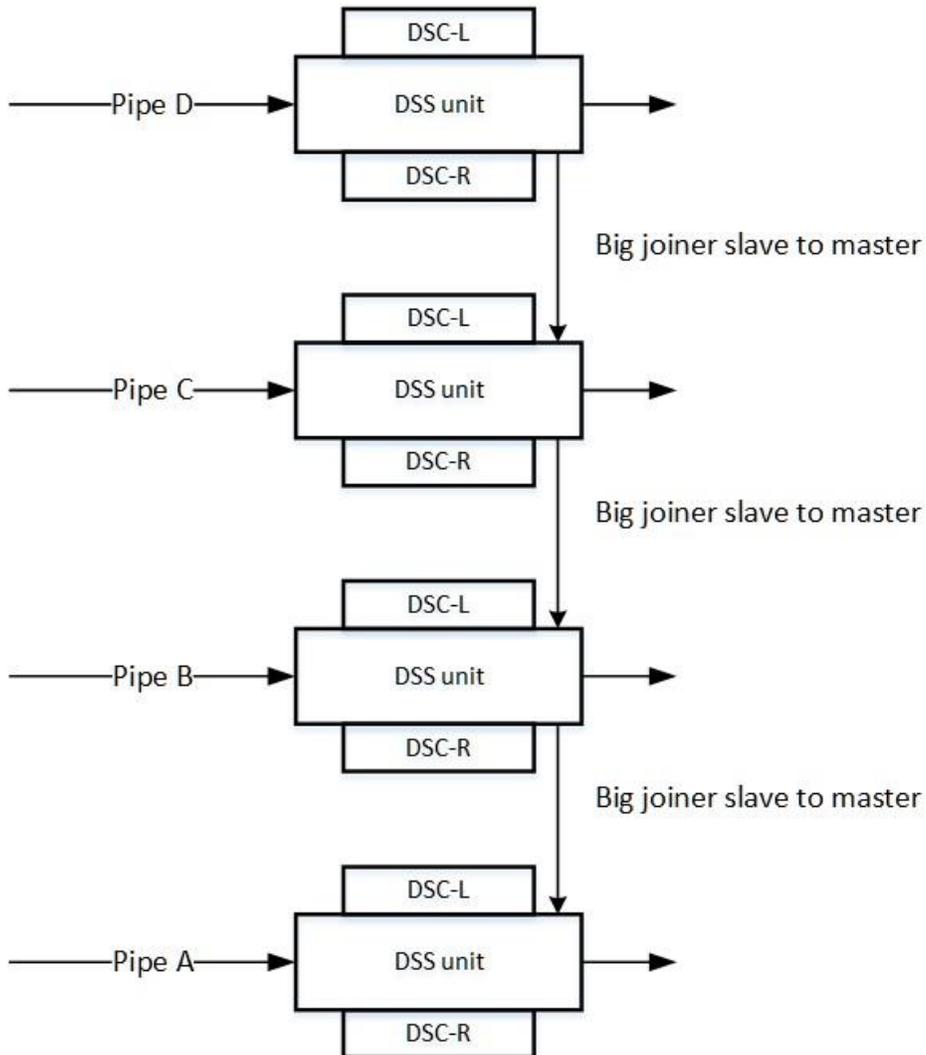
Big joiner mode cannot be used if compression is bypassed in VDSC.

Big joiner mode cannot be used if small joiner is not enabled.

The front-end display requirements in the case of big joiner mode are same as in dual port 8K configuration. Audio works same as in other display configurations.

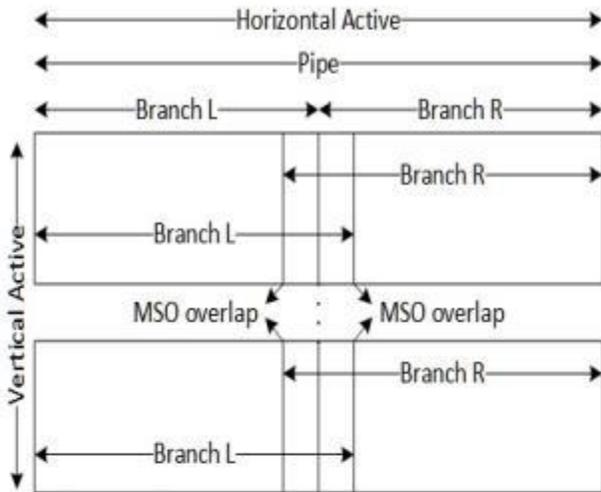
In DSC:

- Compression engine is moved to the pipe and all pipes support VDSC.
- Any two adjacent pipes can be joined together as shown below to support even higher resolutions (example: 8k).



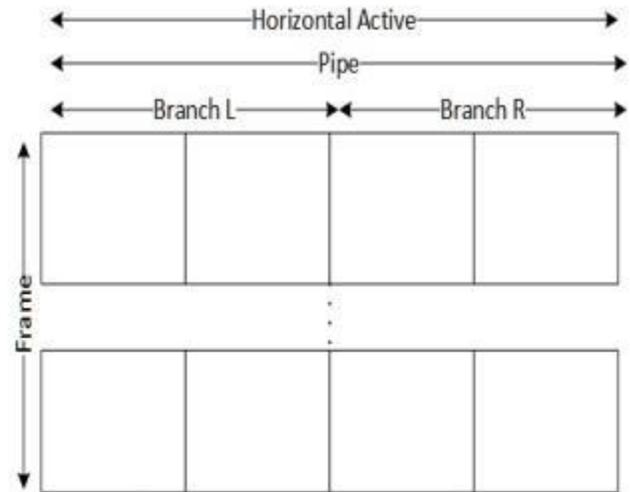
VDSC Slice Options

#1 No big joiner
2 horizontal slice
N vertical slice



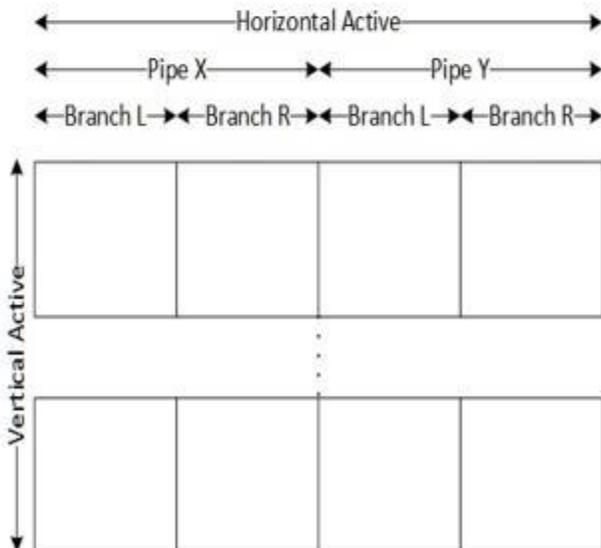
slice_per_line = 1
slice_row_per_frame=N

#2 No big joiner
4 horizontal slice
N vertical slice



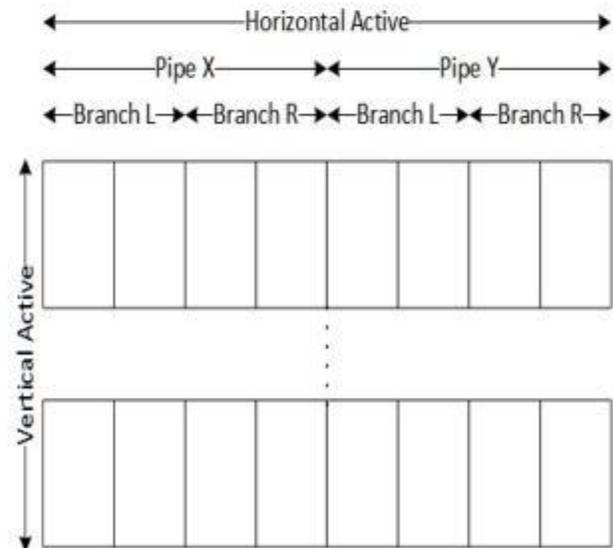
slice_per_line = 2
slice_row_per_frame=N

#3 Big joiner
4 horizontal slice
N vertical slice



slice_per_line = 1
slice_row_per_frame=N

#4 Big joiner
8 horizontal slice
N vertical slice



slice_per_line = 2
slice_row_per_frame=N

- Each DSC branch supports 1 or 2 horizontal slices.
- Both branches must be enabled with small joiner, so pipe output is 2 or 4 horizontal slices
- Final port output without big joiner (1 pipe alone) is 2 or 4 horizontal slices.
- Final port output with big joiner (2 pipes combined) is 4 or 8 horizontal slices.
- For non-MSO use cases, Horizontal active must be an integer multiple of slice width in pixels. For MSO use cases, overlap pixels need to be included.
- Each DSC branch $\text{slice_per_line} = \text{number of horizontal slices this DSC branch is processing} = \text{number of horizontal slices for full frame} / \text{number of DSC branches enabled for this video stream}$
- Minimum vertical slice (slice row) count is 1.
- Maximum vertical slice (slice row) count is 4095.
- Vertical active must be an integer multiple of slice height in lines.
- Each DSC branch $\text{slice_row_per_frame} = \text{number of vertical slices} = \text{vertical active} / \text{slice height}$
- VDSC spec implies that 108 lines is an optimal slice height, but any size can be used as long as vertical active integer multiple and maximum vertical slice count requirements are met.
- Each DSC branch $\text{pic_height} = \text{vertical active}$
- non-MSO: Each DSC branch $\text{pic_width} = \text{horizontal active} / \text{number of DSC branches enabled for this video stream}$. PPS transmitted to receiver must use full width.
- MSO: Each DSC branch $\text{pic_width} = \text{horizontal active} / \text{number of DSC branches enabled for this video stream} + \text{overlap pixels}$. PPS transmitted to receiver must use full width.

Programming considerations and restrictions

DP 1.4a RESTRICTION: For every VDSC instance, active video height must be an integer multiple of slice height.

DSC 1.2a slice height recommendation: Based on empirical data the optimal slice height is 108.

The default slice height can be set to a value that satisfies the above as follows.

slice height = ~108 and picture height = N * slice height where N is an integer.

Below are the checklist items to program VDSC.

- DSS registers to configure VDSC branches
- Picture parameter set register programming for each VDSC branch
- Link M/N programming for compression enabled use cases
- Transcoder data island packet for PPS

Note that there is no implicit order to program compression engine's PPS and DSS registers other than that they need to be programmed prior to a VBLANK.

DSI Configuration	Splitter	Joiner	DSC to DSI Mapping	Notes
Single link - single pipe	Yes	Yes	DSC (L) + DSC (R) -> DSIx	Small joiner output always on left VDSC branch.
Dual independent links	Yes	Yes	PipeX -> DSC (L) + DSC (R) -> DSI0 PipeY -> DSC (L) + DSC (R) -> DSI1	Each pipe will instantiate its own DSS unit to drive pixels to one of the DSI ports.
Dual link - single pipe	Yes	No	Pipe A DSC(L) - DSI0 Pipe A DSC(R) - DSI1	Both ports must be bound to the same pipe AND DSI0 must be configured for Port Sync Mode. Only pipe A can be used.

The following rules determine the highest output bpp that the design can support in various compression scenarios.

Pipe BW check: Pixel clock < PPC * CDCLK * number of pipes

Link BW check: output bpp < number of lanes * DDICLK * 8 / Pixel clock -- DisplayPort

Big joiner BW check
output bpp <= PPC * CDCLK * 24 bits / Pixel clock

Small joiner RAM check: output bpp <= joiner RAM (bits) / Horiz. width

Note that horizontal width has to account for overlap and dummy pixels in CoG use cases.

Greatest allowed DSC output BPP = INT(MIN (output BPP from Link BW check, output BPP from Big joiner BW check, output bpp from small joiner RAM check))

For cases where FEC is enabled, pixel clock is replaced by pixel clock/0.972261 in the above calculations.

Smaller Joiner RAM Size
7680 bytes per pipe, 15360 bytes for two joined pipes
Example:
CDCLK = 336 MHz, DDICLK = 810 MHz, PPC = 2 pixels/cycle, DP lanes = 4, input Pixel BPP = 30, Pixel clock (CVT) = 1026 MHz, Horiz. width = 3840 pixels
Greatest allowed DSC output BPP = INT(MIN (output BPP from Link BW check, output BPP from Big joiner BW check, output bpp from small joiner RAM check))
= INT(MIN (25.26, 15.71, 32))
= INT(15.71)
= 15



Design Support

With dual pipe configuration, HW is capable of supporting maximum of 8 slice per line. (Big joiner mode and dual pipe cases)

With single pipe configuration, HW is capable of supporting maximum of 4 slice per line. (Small joiner and single pipe cases)

Input pixel format	bpp	Max input width	Output bits per pixel ⁽¹⁾	DSC Standard
RGB444	24 and 30 bpp	5120 pixels per pipe, 8192 across two joined pipes	8 to 23 ⁽²⁾	v1.1
RGB444	24, 30 and 36 bpp	5120 pixels per pipe, 8192 across two joined pipes	8 to 23 ⁽²⁾	v1.1

⁽¹⁾BPP at max input width

⁽²⁾Use the rules above to calculate maximum bpp

⁽³⁾FEC is enabled. See the DDI FEC section for FEC support.

⁽⁴⁾ Fractional bpp mode enabled.

⁽⁵⁾ Some joiner modes limit output bpp to half of input bpp.

Transcoder

Transcoder DSI Function

The following registers are specific to the MIPI Display Serial Interface (DSI) ports. Both DSI Controller (i.e. the DSI Protocol Layer) and D-PHY (i.e. the DSI Physical Layer) registers are listed here.

Registers
DSI specific functional configuration registers:
TRANS_DSI_FUNC_CONF
Control for DSI protocol timers:
DSI_PWAIT_TO
DSI_HTX_TO
DSI_LRX_H_TO
DSI_TA_TO
DSI_CALIB_TO
DSI interrupt registers:
DSI_INTER_MSK_REG
DSI_INTER_IDENT_REG
DSI timing control registers:
DSI_CLK_TIMING_PARAM
DSI_DATA_TIMING_PARAM
DSI_TRIG_TX_TIME

Registers
DSI_ESC_CLK_DIV
DSI_T_INIT_PRIMARY
DSI_T_WAKEUP
DSI_IO_MODECTL
D-PHY control registers (Located in Combo-PHY):
DPHY_CLK_TIMING_PARAM
DPHY_DATA_TIMING_PARAM
DPHY_TA_TIMING_PARAM
DPHY_TRIG_EXT
DPHY_ESC_CLK_DIV
DSI command transmit/receive control:
DSI_CMD_FRMCTL
DSI_CMD_TXCTL
DSI_CMD_RXCTL
DSI_LP_MSG
DSI receive data (header and payload):
DSI_CMD_RXHDR
DSI_CMD_RXPYLD
DSI transmit data (header and payload):
DSI_CMD_TXHDR
DSI_CMD_TXPYLD
DC state control:

The following registers are common across all ports, in which the DSI port is one of them.

Registers
Transcoder Control
TRANS_CONF
TRANS_STEREO3D_CTL
Transcoder Timing
TRANS_HTOTAL
TRANS_HSYNC
TRANS_VTOTAL
TRANS_VSYNC
TRANS_SPACE
TRANS_VSYNCSHIFT
Transcoder DDI Function
TRANS_DDI_FUNC_CTL
TRANS_DDI_FUNC_CTL2

Programming Reference

Burst Mode Considerations

The MIPI DSI Burst Mode is a Video Mode concept where the transcoder delivers pixels to the Panel at a higher bandwidth than the pixel rate required to maintain the display timings. Note that there is no concept of Burst Mode in Command Mode since the Host is not maintaining the display timings and should be sending the pixels to the Panel as fast as possible.

Burst mode can be achieved in Video Mode by one of the following methods:

1. Program the Link speed to the minimum frequency needed to satisfy the resolution pixel rate and compress the pixel stream. This is assuming the minimum frequency is less than or equal to the maximum frequency supported.
2. Program the Link speed to a supported frequency that will generate a pixel rate greater than what is required to satisfy the resolution pixel rate

It is important to understand that the DSI transcoder uses the Horizontal Total, Sync Start, and Sync End programming (programmed in terms of pixels) to calculate the number of 1X clocks needed to attain the desired H. timings to the Panel. Hardware uses the following equation to perform this conversion:

$$\text{H. Time (in 1X clocks)} = \text{ceiling}(\text{H. Size} * \text{bpp}) / (\text{Operating Width} * 8)$$

Some notes on this equation:

1. The "bpp" term is the pixel format being transmitted to the Panel
2. 24bpp should be used for compressed pixel streams
3. Hardware does not know what frequency the Link is running at, so it is the responsibility of Software to program the Link frequency appropriately to meet the desired pixel rate

In a non-Burst Mode of operation, Software will program the Link speed to a frequency that will generate the pixel rate on the Link needed to satisfy the Panel's resolution. In this case the H. timings that Hardware calculates will correlate directly to the H. timings (in pixels) that Software programs. In other words, the 1X clock period is equal to the pixel rate needed for the resolution. However, for a Burst Mode method where the Link speed is running at a higher frequency, the period of the 1X clock is going to be less than the theoretical pixel rate period, so Software will need to account for that by adjusting the H. Total, Sync Start, and Sync End timings. In other words, the H. timings will need to be increased to offset the faster 1X clock frequency.

Calculating the Horizontal burst timings in pixels is simply taking the normal H. timing parameter and multiplying it by the burst ratio, where the burst ratio is:

$$\text{Burst Frequency Ratio} = f_{8X \text{ Burst}} / f_{8X \text{ Minimum}}$$

$$\text{H. Timing Burst} = \text{ceiling}(\text{H. Timing Non-Burst} * \text{Burst Frequency Ratio})$$

Example:

H. Active	3072	pixels
H. Blank	160	pixels
H. Front Porch	48	pixels
H. Sync	32	pixels
V. Active	1920	lines
V. Blank	55	lines
Frames per Second	60	fps
Pixel Format	18	bits per pixel
Operating Width	4	lanes
Pixel Frequency	382.99	Mega pixels per frame
8X Min Frequency	1723	MHz
8X Burst Frequency	2500	MHz
Burst Frequency Ratio	1.451	

$$\text{H. Total Burst} = \text{ceiling}((3072 + 160) * 1.451) = 4690 \text{ pixels}$$

$$\text{H. Sync Start Burst} = \text{ceiling}((3072 + 48) * 1.451) = 4527 \text{ pixels}$$

$$\text{H. Sync End Burst} = \text{ceiling}((3072 + 48 + 32) * 1.451) = 4574 \text{ pixels}$$

Programming Horizontal Timings in Video Mode when operating with compressed pixels

When a Horizontal resolution requires a Link speed that is greater than the maximum supported frequency, then compression can be used to reduce the Link speed to a frequency that is within a supported frequency. When the transcoder is operating within this regime, then similar to the Burst Mode described above, the Horizontal timings (H. Total, Sync Start, and Sync End) will need to be adjusted to account for a Link speed that is slower than what the non-compressed Link speed would need to be running at.

$$\text{H. Timing Compressed} = \text{ceiling}(\text{H. Timing Non-Compressed} * \text{Compression Frequency Ratio})$$

$$\text{Compressed Frequency Ratio} = f_{8X \text{ Compression}} / f_{8X \text{ Non-Compressed}}$$

Example:

H. Active	5120	pixels
H. Blank	160	pixels
H. Front Porch	48	pixels
H. Sync	32	pixels
V. Active	3200	lines
V. Blank	91	lines
Frames per Second	60	fps

Pixel Format	24	bits per pixel
Operating Width	4	lanes
Pixel Frequency	1042.5888	Mega pixels per second
8X Non-Cmp Frequency	6255.5	MHz
8X Cmp Frequency	2211.6	MHz
Cmp Frequency Ratio	0.3535	

H. Total Burst= ceiling((5120 + 160) * 0.3535) = 1867 pixels

H. Sync Start Burst = ceiling((5120 + 48) * 0.3535) = 1828 pixels

H. Sync End Burst = ceiling((5120 + 48 + 32) * 0.3535) = 1839 pixels

Determining Minimum Horizontal Blanking Regions

When in Video Mode, there are instances where the DSI controller does not have enough time to let the link transition to the LP state before the next HS burst. In these instances, the controller transmits a Blanking packet to keep the link in the HS state. However, Blanking packets are long packets (i.e. a packet header + payload + payload CRC) which means that the time between non-Blanking packet HS bursts (i.e. blanking regions) must be big enough to transmit a minimum sized Blanking packet.

Per the DSI specification, a long packet can carry zero bytes of payload (i.e. a packet header + payload CRC), but to simplify the hardware the transcoder expects the Blanking packet to carry a payload, and to keep the CRC boundary conditions to a minimum, the transcoder expects the payload to be **greater** than 4 bytes. Therefore, the minimum sized Blanking packet size that the controller can support is **11 bytes**.

In addition to the minimum Blanking Packet size, the blanking region must also account for protocol overhead (i.e. Pixel Packet header and payload CRC, and Sync Packets). The overhead within each horizontal blanking region differs slightly as shown within the following table.

Blanking Region	Size (Bytes)	Notes
Horizontal Front Porch	17	11B for minimum Blanking Packet, 6B for Pixel Packet header and payload CRC
Horizontal Sync	15	11B for minimum Blanking Packet, 4B for Horizontal Sync Start packet
Horizontal Back Porch	15-19	11B for minimum Blanking Packet, 4B for Horizontal Sync End packet, 4B for Pixel Packet header. Additional notes: <ol style="list-style-type: none"> 1. HW subtracts the Pixel Packet header from this region which is why the additional 4B is added to the minimum. 2. During Sync Event model the H. Sync End packet is not

Blanking Region	Size (Bytes)	Notes
		transmitted, so those 4B can be recovered.

Given the above, the number of byte clocks for a minimum sized blanking region can be calculated with the following equation.

$$\text{Min Blanking Region (Byte clocks)} = \text{Ceiling}(\text{Region Size} / \text{Operating Width})$$

The following table lists the blanking region size in number of Byte clocks across the different operating widths:

Blanking Region	x1	x2	x3	x4
Horizontal Front Porch	17	9	6	5
Horizontal Sync / Horizontal Back Porch (Event)	15	8	5	4
Horizontal Back Porch (Pulse)	19	10	7	5

All the horizontal timings to the transcoder's timing generator is defined in pixels, so the minimum blanking region needs to be translated to the pixel domain. Recall from above, we have the following equation to translate the horizontal timings from the pixel domain to the number of 1x (i.e. Byte) clocks.

$$\text{H. Time (in 1X clocks)} = \text{Ceiling}(\text{H. Size} * \text{Format bpp} / (\text{Operating Width} * 8))$$

$$\text{Minimum H. Size (in pixels)} = \text{Ceiling}(\text{Min Blanking Region} * \text{Operating Width} * 8 / \text{Format bpp})$$

Where:

1. The equation is assuming the minimum horizontal pixel size result is one-based. Software will need to convert the size to zero-based when programming the horizontal timing registers
2. "H. Time (in 1X clocks)" = "Min Blanking Region"
3. The numerator of the equation will be a constant for each operating width and represents the blanking region size in terms of bits

Determining Vertical Blank in DBI

When operating in Command Mode (i.e. DBI) the DSI transcoder is not responsible for the timings to the Panel, but it does need to maintain some timings to the Display Engine. Specifically, the Display Engine requires a certain amount of Vertical blanking time to fetch enough of the image from memory to supply the required bandwidth to the DSI transcoder. The minimum Vertical blanking time (in lines) that the Display Engine needs is:

$$\text{V. Blank} = \text{ceiling}(400\mu\text{s} / \text{Line Time})$$

The time it takes the DSI transcoder to transmit a line (i.e. Line Time) is given by the following equation:

$$\text{Line Time} = \# \text{ Byte clocks per line} * \text{Byte clock period}$$

Line Time = [(H. Total * Bytes per Pixel) / Operating Width] * Byte clock period

Example:

H. Total = 4256 pixels

8X Clock = 2500 MHz = 3.2ns Byte clock period

Pixel Format = RGB888 = 3 Bytes per Pixel

Operating Width = 4

Line Time = [(4256 * 3) / 4] * 3.2ns = 10.214us

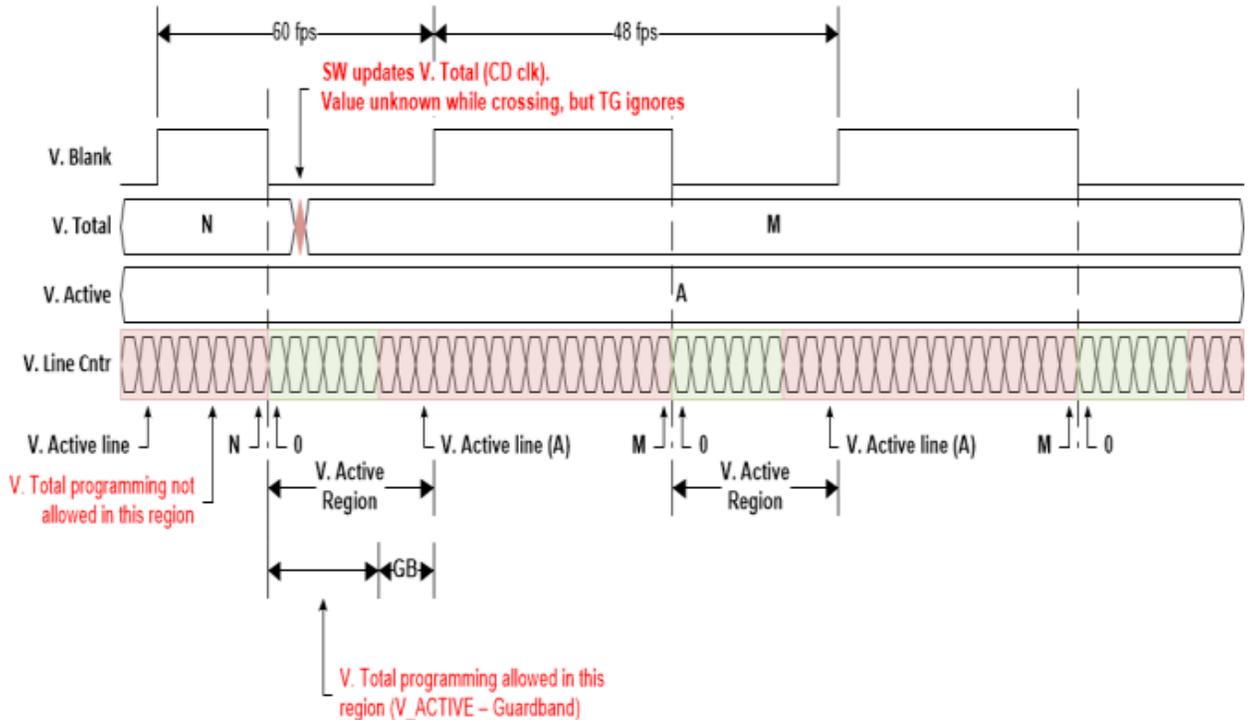
V. Blank = ceiling(400 / 10.214) = 40 lines

Extending Vertical Blank for Lower Frame Rates in DPI

A power saving option when the DSI transcoder is running in Video Mode is to extend the V. Blank time of the frames while running the clocks at the frequency required for the highest refresh rate. Normally, the transcoder expects the frame timings to be programmed before the timing generator is enabled and it expects the timings to remain unchanged while the timing generator is enabled, but for the Vertical timings if Software programs them at the right time, the transcoder should be able to handle the change seamlessly.

Specifically for this functionality, Software will be changing the V. Total field of the **TRANS_VTOTAL** register to either extend or reduce the V. Blank duration to get the desired frame rate. Since the DSI transcoder's timing generator only looks at this field when it is in the Vertical Back Porch, Software shall program the desired V. Total during the vertical active region.

To ensure the write to the TRANS_VTOTAL finishes before the V. Blank region, Software should ensure that it is not closer than 10 lines from V. Blank before issuing the write (i.e. lines 0 to (V. Active-10)). **Also, to ensure there is no glitching on the V. Active field of the TRANS_VTOTAL register, Software shall disable the byte enables for that portion of the register when re-programming the V. Total field.**



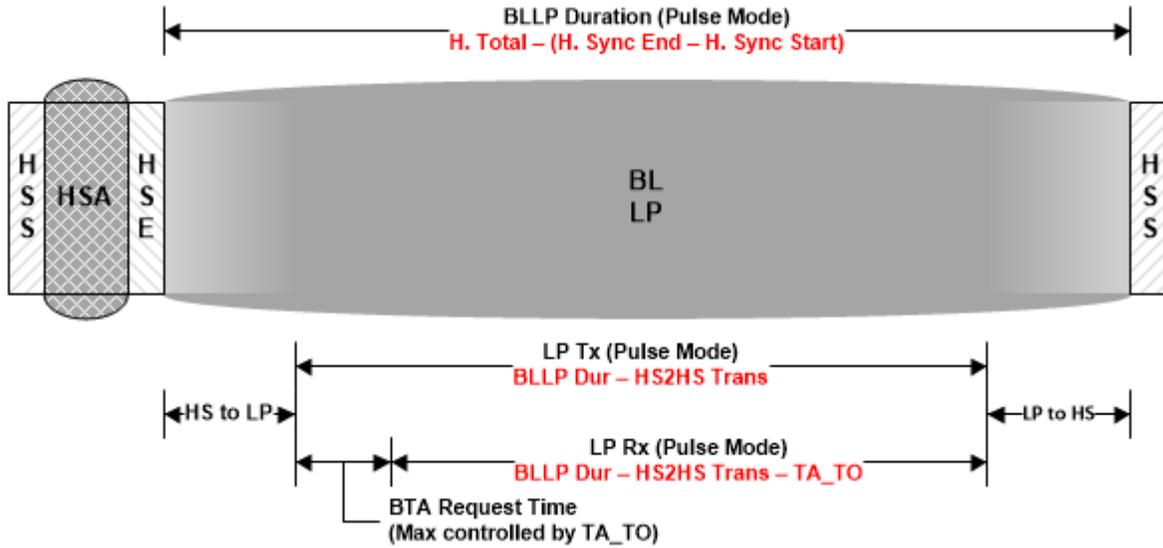
Performing LP Transactions in Video Mode

When in Video Mode the Host is responsible for transmitting the timing information to the Panel. Low Power communication is a very low bandwidth mode of operation on the Link (about 10 Mbps), so it is critical that there is enough time between scheduled synchronous timing packets to perform any LP transactions.

The following latencies need to be accounted for if a LP transaction is going to be performed:

- The available time between Sync packets in the Vertical blanking region
- HS to LP and LP back to HS Link transition latencies (HS to HS latency)
- Bus Turn-Around (BTA) latency, if doing a BTA
- LP transmission

The below diagram illustrates the durations available for LP communication within a Vertical blank line time.



The amount of time available for the LP transactions within the V. blanking region is dependent on whether the transcoder is in the Sync Pulse of Event mode.

Sync Mode	BLLP Duration
Pulse	H. Total - (H. Sync End - H. Sync Start)
Event	H. Total

Since the Horizontal timings are defined in terms of pixels, the BLLP Duration needs to be converted to the Byte clock domain using the following equation:

$$\text{Number of Byte Clocks} = \text{Ceiling}(\text{H. Size} * \text{Bytes per Pixel}) / \text{Operating Width}$$

The HS to HS transition latencies are given in the following table:

CLK Lane in LP?	HS2HS Transition Latency
Yes	$(N * 17) + 12$
No	$(N * 7.5) + 4$

Where N is the number of Byte clocks per Escape clock

$$N = \text{ceiling}(f_{\text{Link}} \text{ (MHz)} / 160)$$

Note that the above Link transition latencies assumes the default HW maintained timing parameters. If any of the timing parameters are being overridden by SW, than that needs to be accounted for.

So, given the above, the amount of time available for the actual LP transmissions is given as the following:

Direction of LP Transaction	Allowed Duration
Tx	BLLP Duration - HS2HS Latency
Rx	BLLP Duration - HS2HS Latency - Turnaround Timeout

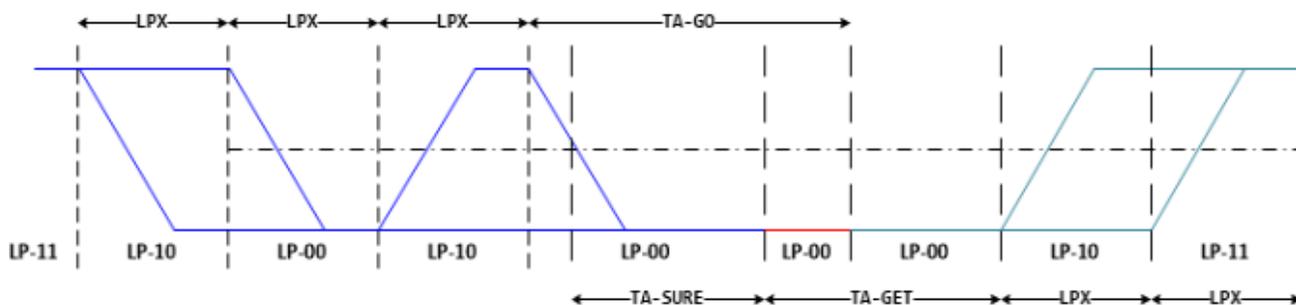
Some things to note on the above durations:

1. For the receive direction, the maximum amount of time given to perform a BTA is defined by the Turnaround Timeout (DSI_TA_TO). This timeout is in Escape clocks, so it will need to be multiplied by N
2. For the receive direction, Software should program the LP RX Timer Timeout (DSI_LRX_H_TO) to the allowed Rx duration.
3. For the receive direction, Software should program the Set Maximum Return Packet Size to account for the allowed Rx duration. Note that the value programmed should take into account the overhead of the Escape Mode entry, exit, packet header, and payload CRC.

Programming DSI_TA_TO and DSI_LRX_H_TO

The TA_TO and LRX_H_TO timers specify how long the DSI controller (i.e. the Host) will allow the panel to own the bus when there is a bus turn-around in progress.

For the turnaround time (DSI_TA_TO), the sequence to turn the Data Lane around is given below:



The table below shows the latency of a bus turn-around.

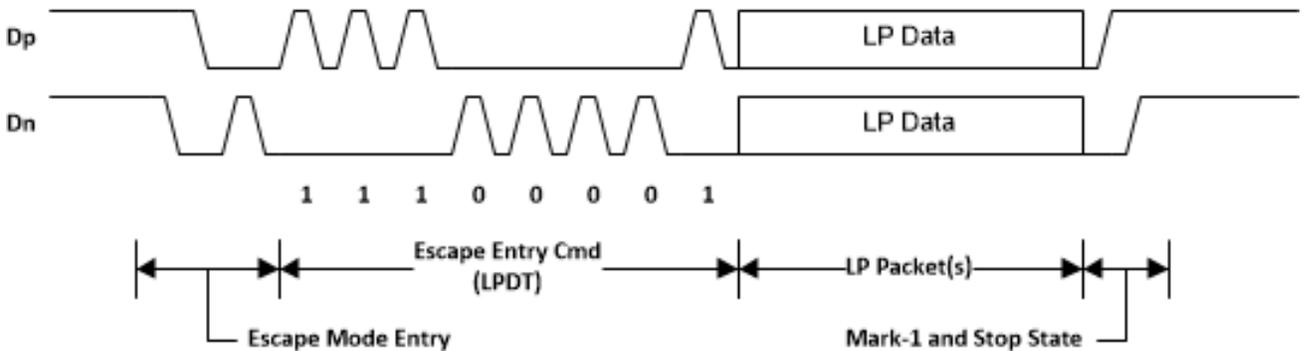
Sequence Step	HW Maintained ⁽¹⁾	SW Override ⁽¹⁾
LP10	1	1
LP00	1	1
LP10	1	1
V _{OL} ⁽²⁾	1	1
LP00 ⁽³⁾	6	TA-SURE + TA-GET

Sequence Step	HW Maintained ⁽¹⁾	SW Override ⁽¹⁾
LP10	1	1
LP11	1	1
Guardband ⁽⁴⁾	2	2
TOTAL:	14	8 + TA-SURE + TA-GET

Notes:

1. The latencies are in terms of Escape clocks (i.e. T_{LPX})
2. From when the Physical layer starts driving the TA-SURE step of the sequence (i.e. LP10 to LP-00), there is a short latency for the 1->0 transition to pass the V_{OL} threshold. This latency is less than one T_{LPX} , but since the granularity of the timeout timer is at the Escape clock granularity, this step is rounded up to one Escape clock.
3. TA-SURE, and TA-GET timing parameters are maintained by the Host and can, therefore, be overridden by Software. HW maintains a TA-SURE of 1 T_{LPX} and a TA-GET of 5 T_{LPX}
4. The guardband is to account for any clock crossings

For the amount of time that the panel should be allowed to own the link (DSI_LRX_H_TO), the timeout value is dependent on the maximum amount of data that the panel could have to send back to the Host. The worst case amount of data the panel can send back to the Host is a READ return followed by an Acknowledge and Error packet.



Sequence Step	Duration ^(1,2)
Escape Mode Entry	5
Escape Entry Cmd	16
LP Packets	READ Hdr: 64 READ Pyld: 512 ⁽³⁾ READ Pyld CRC: 32

Sequence Step	Duration ^(1,2)
	Ack & Error Hdr: 64 ⁽⁴⁾
Mark-1 and Stop State	2
Bus Turn-Around	14
Total:	709

Notes:

1. When in the Escape Control domain, it takes two Escape clocks (T_{LPX}) to transmit a given bit (e.g. the Escape Entry Command is 8 bits, so it takes $16 T_{LPX}$ to transmit the command)
2. This analysis is assuming the panel is not stalling
3. The Host can accept up to 32 bytes of READ payload data from the panel (i.e. the Set Maximum Return Packet Size can be set up to 32). Software can control how much READ data the panel sends for each BTA.
4. This analysis is assuming the panel sends the Acknowledge and Error packet immediately after the READ packet.

Transcoder Control

Control
TRANS_CONF
TRANS_STEREO3D_CTL

Transcoder LRR

LRR (Low refresh rate) is mainly about dynamically reducing panel refresh rate to lowest possible refresh rate or to media multiple refresh rate (e.g. 48Hz) for idle and media playback scenarios. LRR provides predictable VBI unlike flip based refresh rate. LRR feature relies on software programmed Vtotal functionality and can be enabled on VRR capable panels.

Entry Sequence

1. If VRR is enabled, driver disables VRR.
2. Driver updates Vtotal and Vblank-end to any refresh rate supported by the panel.

Exit Sequence

1. Driver updates Vtotal and Vblank-end to nominal refresh rate.

Here is an example scenario with idle and LRR.

1. Configure eDP/DP at 60 Hz.



2. LRR idle entry: Update Vtotal and Vblank-end for 40 Hz
3. LRR idle exit: Update Vtotal and Vblank-end for 60 Hz
4. Enable VRR and program flips
5. Disable VRR
6. LRR media playback entry: Program flip and then update Vtotal and Vblank-end for 48 Hz
7. LRR media playback exit: Update Vtotal and Vblank-end for 60 Hz

LRR Restrictions

- After push, software must wait for push done before starting another push.
- Global Double Buffer Disable functionality is not supported with LRR.
- No asynchronous flips are allowed when LRR is enabled.
- Interlaced mode, DRRS, Stereo 3D, and PSR2 are incompatible with LRR.
- Since panels may have a restriction on the maximum change in refresh rate between two consecutive frames, disabling LRR (TRANS_VRR_CTL[31]=0) does not immediately bring the current refresh rate to the nominal refresh rate.
- When using the transcoder port sync mode, LRR must use Pipeline Full Override Programmed Pipeline Full Line Count setting in TRANS_VRR_CTL.
- PSR is incompatible with LRR.

Transcoder Timing

Transcoder
TRANS_HTOTAL
TRANS_HBLANK
TRANS_HSYNC
TRANS_VTOTAL
TRANS_VBLANK
TRANS_VSYNC
TRANS_SPACE
TRANS_VSYNCSHIFT
TRANS_MULT

The transcoder timing generators create two versions of vertical blank (vblank) that are sent to the display pipes. The undelayed vblank asserts at the end of vertical active (TRANS_VTOTAL Vertical Active). The delayed vblank asserts at the vertical blank start (TRANS_VBLANK Vertical Blank Start) and must be programmed greater than or equal to the end of vertical active. Both de-assert at the end of vblank (TRANS_VBLANK Vertical Blank End), which must be programmed to match the vertical total (TRANS_VTOTAL Vertical Total).

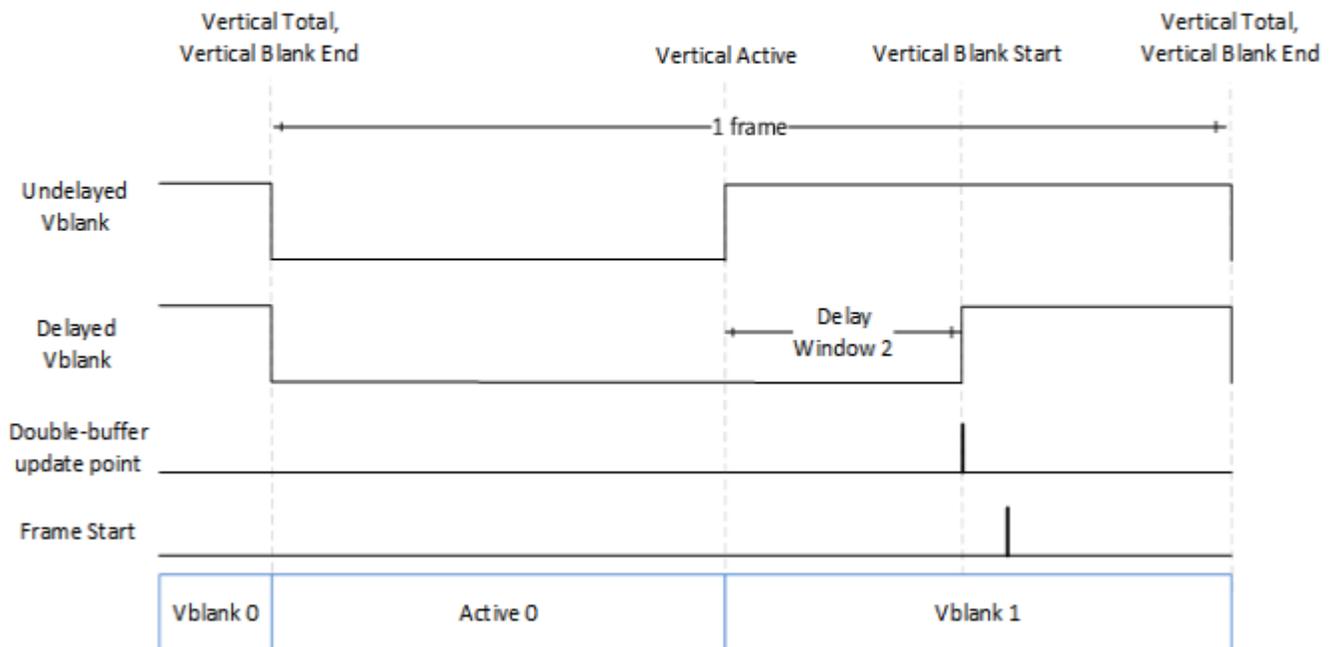
The rising edge of the delayed vblank is the double-buffer register update point. The frame start signal asserts after that point, then pixel processing begins for the next frame.

TRANS_VBLANK Vertical Blank Start can be programmed greater than TRANS_VTOTAL Vertical Active to create a window between start of undelayed vblank and delayed vblank where pixels are not being processed (called window 2) and programming can safely update double-buffered and non-double-buffered registers.

Window 2 size = TRANS_VBLANK Vertical Blank Start - TRANS_VTOTAL Vertical Active

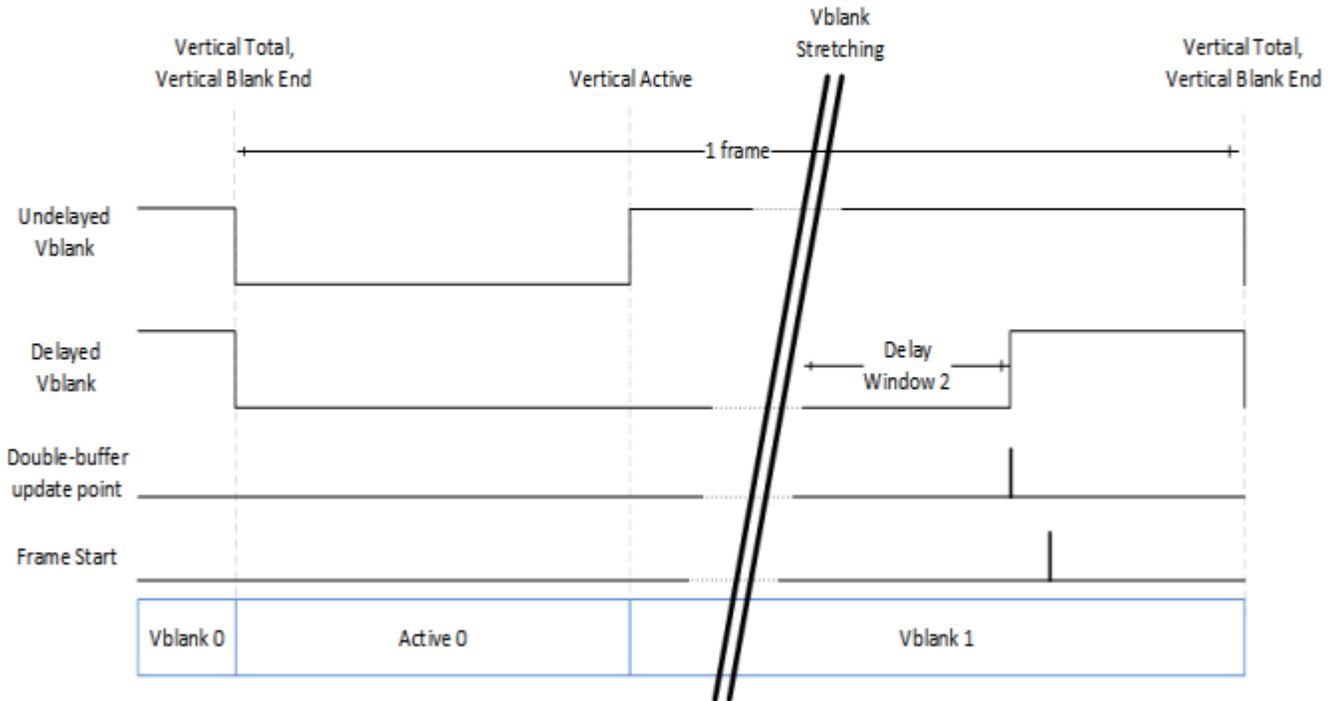
Window 2 is typically only configured to be greater than zero when DSB will be used to program display registers. Without DSB, driver MMIO programming usually is too slow to fit in window 2, and instead must update only the double-buffered registers and use capabilities like the double buffer update disable to ensure atomic updates, and then there is no need to delay the vblank. See the DSB Engine Programming for window 2 size requirements.

Delayed Vblank With Fixed Refresh Rate



When variable refresh rate is used, the vblank stretches while waiting for the indication to complete this vblank and start the next frame. The stretching happens before window 2.

Delayed Vblank With Variable Refresh Rate



Transcoder MN Values

There is one instance of these registers per each transcoder.

For dynamic switching between multiple refresh rates, M/N values may be reprogrammed on the fly. The link N should be programmed last to trigger the update of all the data and link M and N registers and then the new M/N values will be used in the next frame that is output.

DATAM

DATAN

LINKM

LINKN

Clocks:

ls_clk is the link symbol clock. i.e. 270 MHz for HBR.

strm_clk is the stream clock, which is the video pixel rate or dot clock.

cdclk is the core display clock.

The link only operates in Synchronous Clock mode. The link clock and stream clock are synchronous and the link M and N values stay constant for a given pixel rate.

Calculation of TU:

TU is the Transfer Unit used in SST.

TU size = 64 (recommended)

Calculation of Data M, and Data N:

Active/TU Size = Payload/Capacity = Data M/N

Note that for 4:2:0 format the number of bytes per pixel will be half the number of bytes of RGB 4:4:4 pixel.

Recommendation: In data M/N calculation, M should be rounded up and N should be rounded down.

Context:	Low Bandwidth SST over High BW link
Active/TU size = Data M/N	
Select link rate such that there is at least 1 active symbol in every TU for any given dot clock.	
An example where this check fails is 720p with YUV 4:2:0 pixel format over HBR3 channel:	
Active/64 = $(27 * 1.5) / (810 * 4)$ results in Active < 1.	

Compression Ratio (CR) = DisplayPort Compression enabled ? min(ratio1,ratio2) : 1

- ratio1 = Compressor BW / Link BW = $(cdclk * \text{bytes per pixel}) / (ls_clk * \text{number of lanes})$
- ratio2 = $(\text{Horizontal active in pixels} * \text{bytes per pixel} / 4) / ((\text{Horizontal active in pixels} * \text{bytes per pixel} / 8) + 2)$

Data M/N = $(strm_clk * \text{bytes per pixel}) / (CR * ls_clk * \text{number of lanes})$

The data M/N should be calculated as below for the eDP cases with VESA DSC.

First compute the compression ratio based on the DSC PPS parameters as follows.

without FEC:

Compression Ratio (CR) = DSC input bits per pixel / DSC output bits per pixel

Examples:

RGB444: Compression Ratio (CR) = $\text{bits_per_component} * 3 / \text{bits_per_pixel}$

If eDP CoG with VESA compression

Data M/N = $(strm_clk_{CoG} * \text{bytes per pixel}) / (CR * ls_clk * \text{number of lanes per segment})$

else if eDP with VESA compression (no CoG)

Data M/N = $(strm_clk * \text{bytes per pixel}) / (CR * ls_clk * \text{number of lanes})$

Note1: Data M/N remains unchanged between CoG and non-CoG because the reduction in the number of lanes is compensated by the reduction in stream clock.

Note2: For the CoG case above, the actual stream clock on pipe side = $(strm_clk_{CoG} * \text{number of segments})$.

Note3: $strm_clk_{CoG}$ is calculated considering a single segment



with FEC:

Compression Ratio (CR) = DSC input bits per pixel / DSC output bits per pixel

Examples:

RGB444: Compression Ratio (CR) = bits_per_component * 3 / bits_per_pixel

If eDP CoG with VESA compression

$$\text{Data M/N} = ((\text{strm_clk}_{\text{CoG}} * \text{bytes per pixel}) / (\text{ls_clk} * \text{number of lanes per segment})) * 1/(0.972261 * \text{CR})$$

else if eDP with VESA compression (no CoG)

$$\text{Data M/N} = ((\text{strm_clk} * \text{bytes per pixel}) / (\text{ls_clk} * \text{number of lanes})) * 1/(0.972261 * \text{CR})$$

Note1: Data M/N remains unchanged between CoG and non-CoG because the reduction in the number of lanes is compensated by the reduction in stream clock.

Note2: For the CoG case above, the actual stream clock on pipe side = (strm_clk_{CoG} * number of segments).

Note3: strm_clk_{CoG} is calculated considering a single segment

The data M/N should be calculated as below for the case FEC is enabled over the link in order to account for the 2.4% overhead.

$$\text{Data M/N} = ((\text{strm_clk} * \text{bytes per pixel}) / (\text{ls_clk} * \text{number_of_lanes})) * 1/(0.972261)$$

If compression is also enabled, then the following calculation should be used.

$$\text{Data M/N} = ((\text{strm_clk} * \text{bytes per pixel}) / (\text{ls_clk} * \text{number_of_lanes})) * 1/(0.972261 * \text{CR})$$

Calculation of Link M and Link N:

Link M/N = strm_clk / ls_clk - for non-CoG use cases

Link M/N = strm_clk_{CoG} / ls_clk - for CoG use cases

Recommendation: In Link M/N calculation, M should be rounded up and N should be rounded down.

Restriction on clocks and number of lanes:

- Number of lanes >= INT(strm_clk * bytes per pixel / ls_clk)

Restrictions on the Virtual Channel (VC) payload size in DisplayPort MST mode

- In a x1 lane config, each pipe stream on the link must use a VC payload size that is a multiple of 4.
- In a x2 lane config, each pipe stream on the link must use a VC payload size that is a multiple of 2.
- In a x4 lane config, each pipe stream on the link must use a VC payload size that is a multiple of 1.

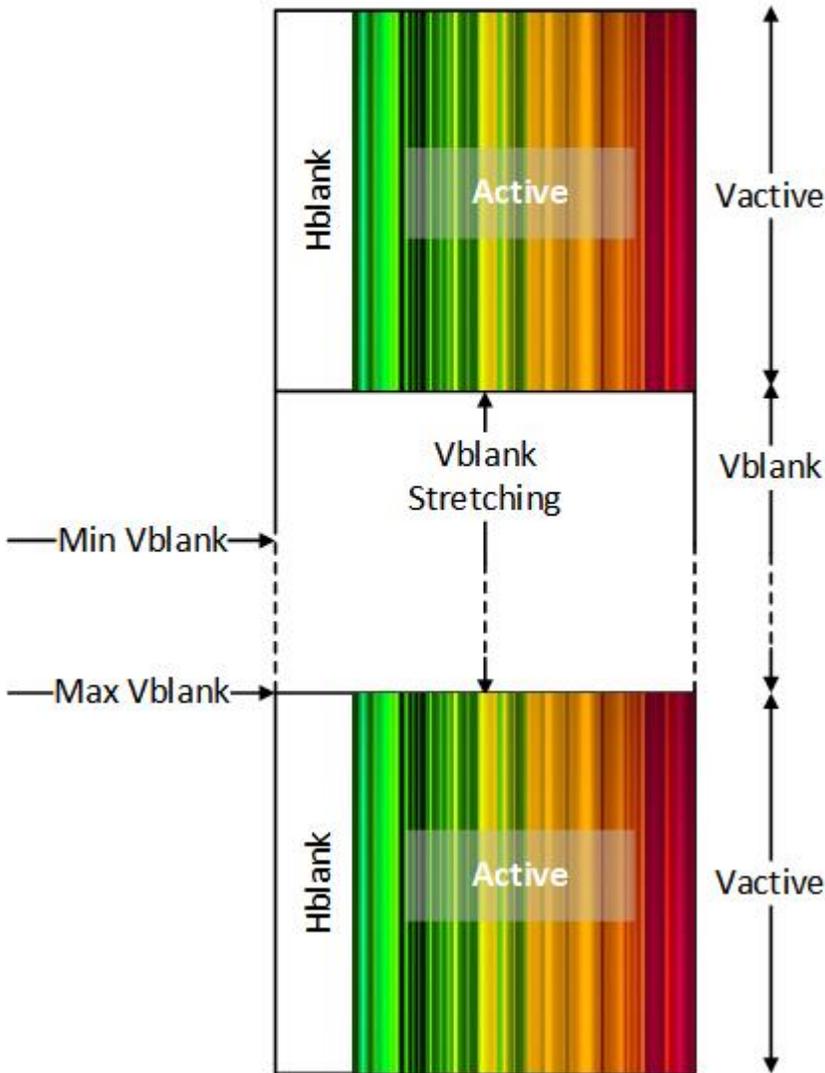
Transcoder VRR Function

VRR Registers
VRR Control TRANS_VRR_CTL
VRR Vmax TRANS_VRR_VMAX
VRR Vmin TRANS_VRR_VMIN
VRR Vmax-shift TRANS_VRR_VMAXSHIFT
VRR Status TRANS_VRR_STATUS
VRR Vtotal Previous TRANS_VRR_VTOTAL_PREV
VRR Flipline TRANS_VRR_FLIPLINE
VRR Status2 TRANS_VRR_STATUS2
TRANS_PUSH

VRR Overview

In the adaptive sync mode (a.k.a. VRR or Variable Refresh Rate), the display engine adapts itself to render speed by adjusting its refresh rate dynamically. Adaptive sync mode is supported both on eDP and DP. The dot clock is set to support the peak desired refresh rate for a given resolution and link clock. A minimum and maximum vertical blank (vblank) period is specified and the display hardware stretches or shrinks the actual vblank period based on the required refresh rate.

Variable Refresh Rate



VRR Details

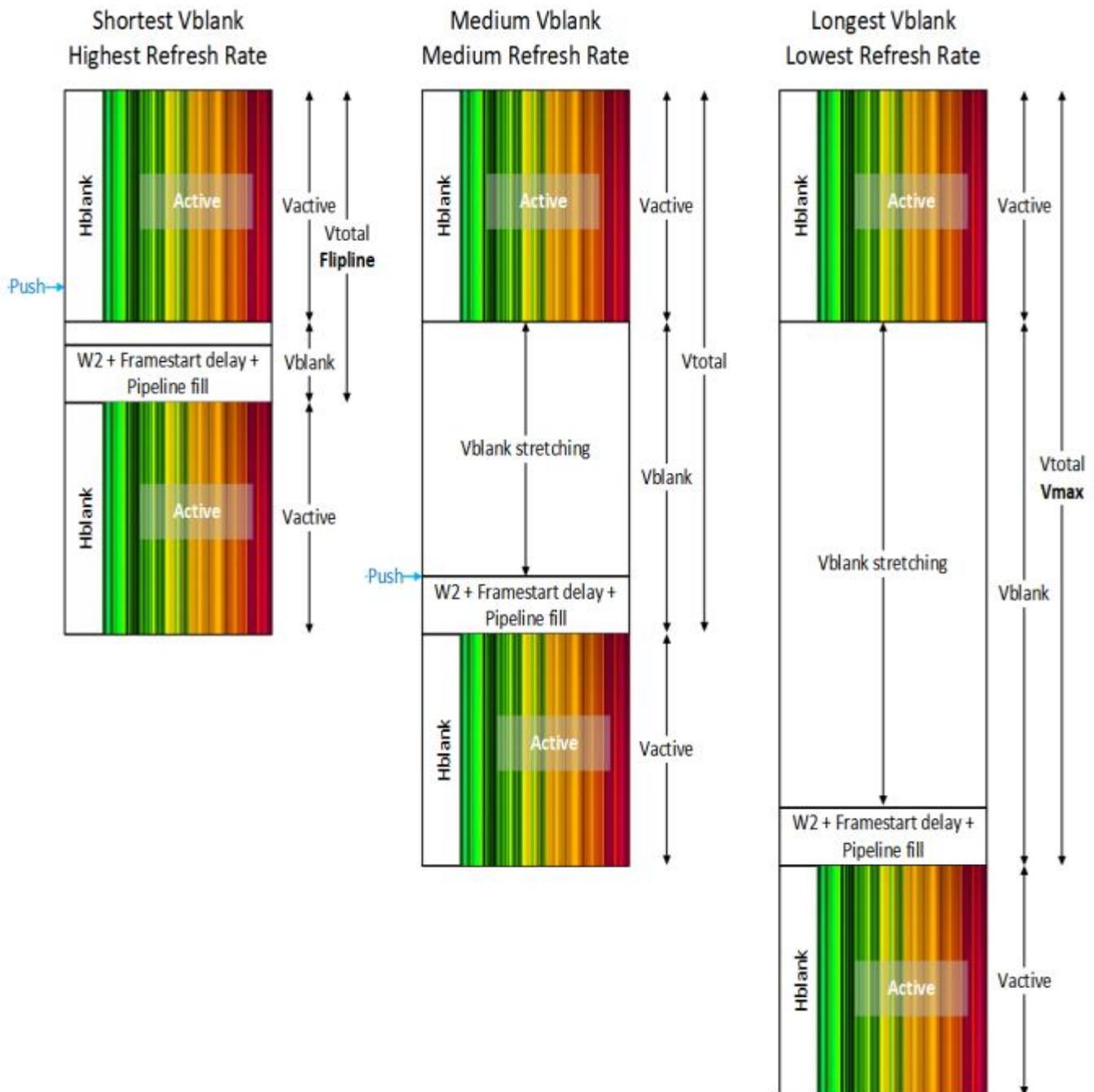
VRR stretches the beginning of vblank until the vblank needs to be terminated, then [delay window 2 \(W2\)](#) happens, double buffer registers are updated (sync flips complete), framestart is triggered (after a delay), the display pipeline fills, and vblank ends.

The vblank period is constrained by the registers specifying vertical total (V_{total}) lines maximum (V_{max}) and minimum (V_{min}). The V_{total} contains the V_{blank} and vertical active (V_{active}) region of each display frame. The minimum V_{blank} must at a minimum contain the framestart delay and pipeline fill.

When there is a frame update (flip or other update), software triggers the termination of vblank by setting the transcoder Send Push bit (push bit). The start of termination may be immediate, or delayed, depending on flip line and max shift. Hardware clears the push bit when the vblank termination begins. The vblank termination is the flip decision boundary, or latest time a push will be serviced. If a push is set after flip decision boundary, it will be held until the next vblank and serviced there.

Flip Line provides a frame by frame programmable limit on the minimum V_{total} . The flip line is programmed before the push. Hardware calculates the flip line decision boundary = flip line value - pipeline fill - framestart delay. If push happened before that boundary point, the vblank termination starts at that boundary point. If push happens after that boundary point, the vblank termination starts immediately.

When software does not trigger termination, the vblank automatically terminates when the programmed vertical max (V_{max}) will be reached. Hardware calculates the V_{max} decision boundary = V_{max} - pipeline fill - framestart delay, and starts termination at that boundary point.

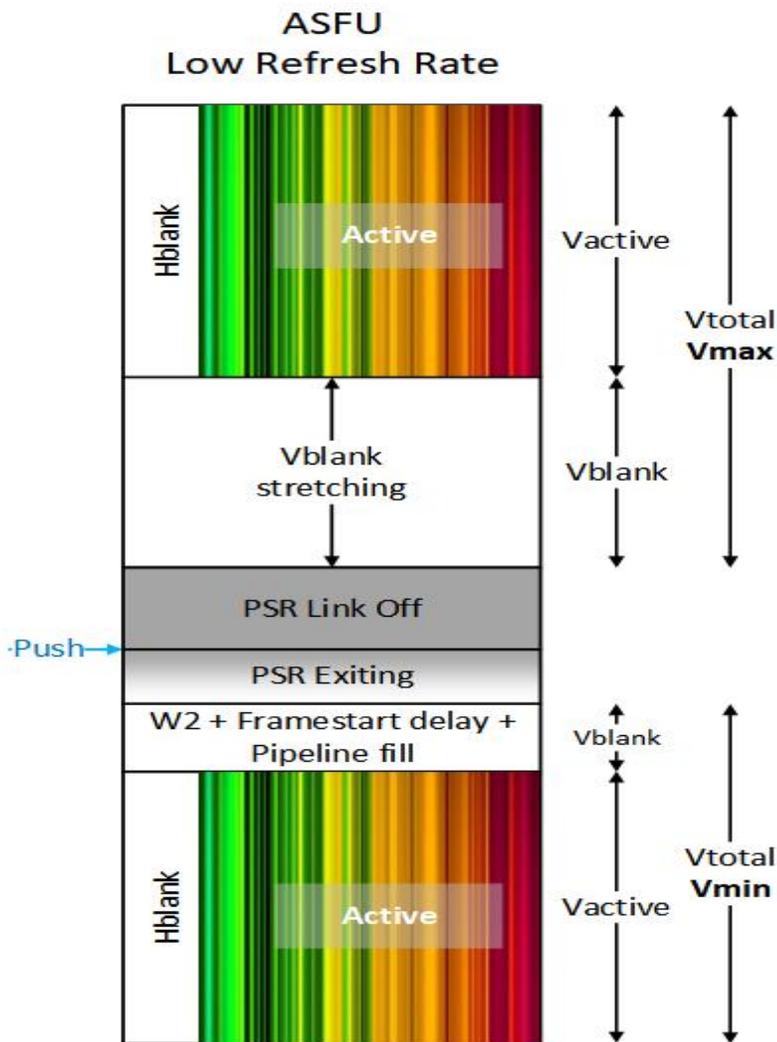


Double-Buffer Behavior

The push mode causes VRR to affect all pipe and plane double-buffered registers. As the vblank sent to the panel (timing generator vblank or undelayed vblank) stretches it delays the assertion of vblank sent to the pipe (delayed vblank), which is used as the double-buffer update for all of the pipe and planes. When VRR begins to terminate the vblank, **delay window 2 (W2)** happens, then the delayed vblank asserts, triggering the double buffer update before the framestart and pipeline fill. As a result, double-buffer registers will update after the push is set, or in a non-ASFU case, when v_{max} decision boundary is reached.

ASFU

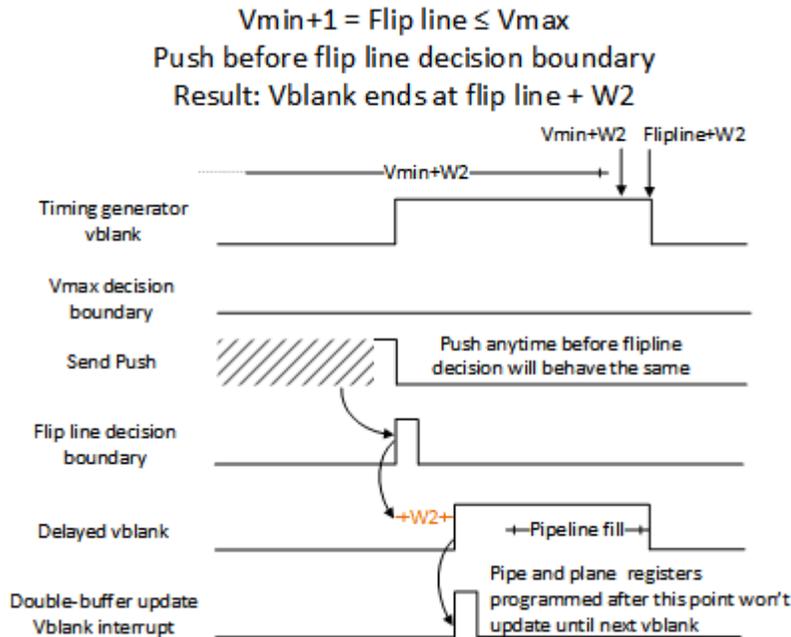
Adaptive Sync Frame Update (ASFU) changes the VRR behavior when max vertical is reached so that it enters PSR instead of terminating vblank and starting another frame. A later push then causes PSR exit with minimum vblank (flip line value ignored at PSR exit). The VRR behavior with push before V_{max} decision boundary is unchanged by ASFU.



VRR Scenarios

Minimum vblank

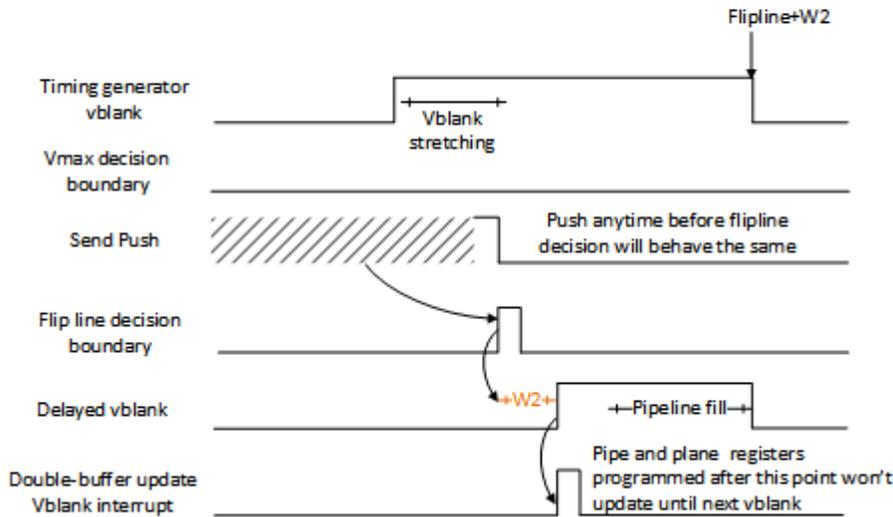
If push happens before flip line decision boundary and flip line value is $V_{min}+1$, the vblank is minimized and the plane and pipe double-buffer registers update at the start of timing generator vblank.



Medium vblank terminated at flip line

If push happens before flip line decision boundary and flip line value is greater than V_{min} and less than or equal to V_{max} , the vblank will stretch and the plane and pipe double-buffer registers update at the flip line decision boundary.

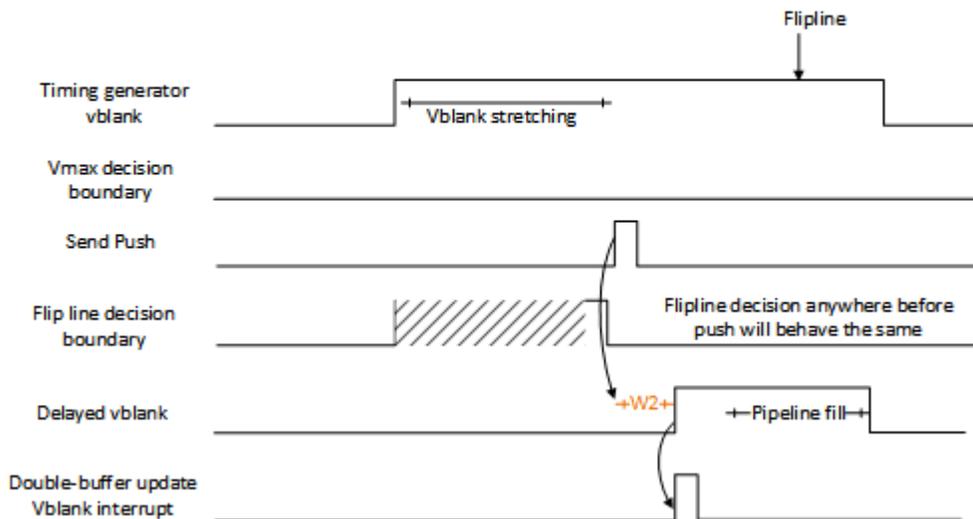
$V_{min} < \text{Flipline} < V_{max}$
 Push before flip line decision boundary
 Result: Vblank ends at flip line + W2



Medium vblank terminated at push

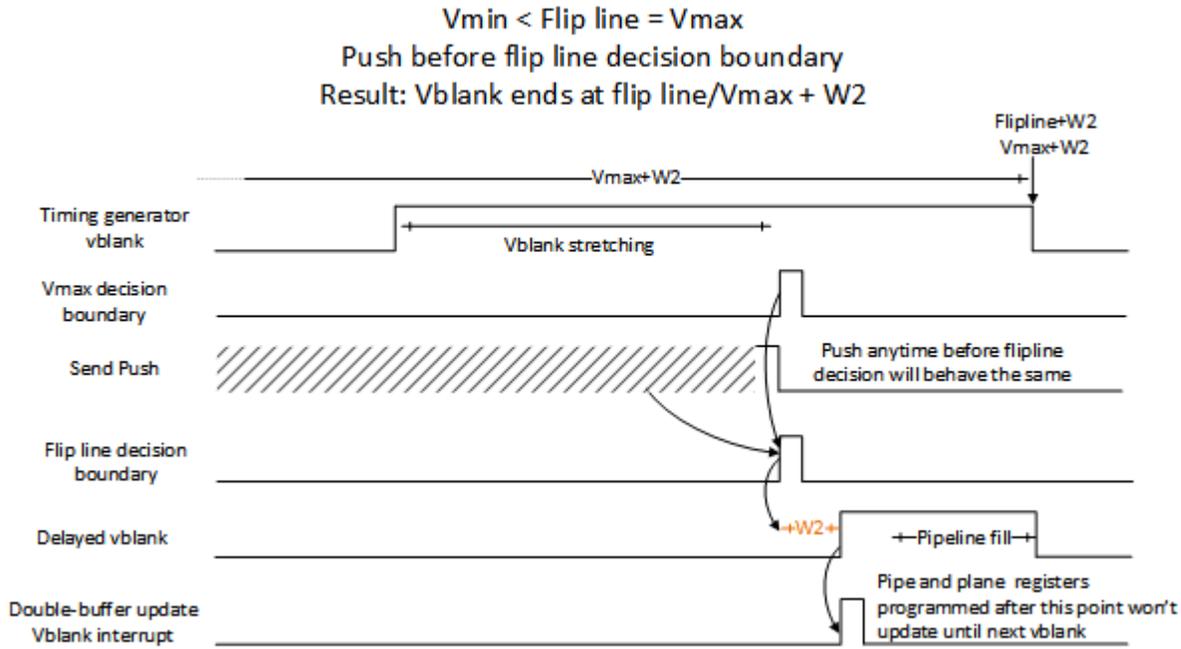
If push happens after flip line decision boundary and before Vmax decision boundary, the vblank will stretch and the plane and pipe double-buffer registers update at the push.

$V_{min} < \text{Flip line} < V_{max}$
 Push between flip line and Vmax decision boundaries
 Result: Vblank ends after flip line and before Vmax



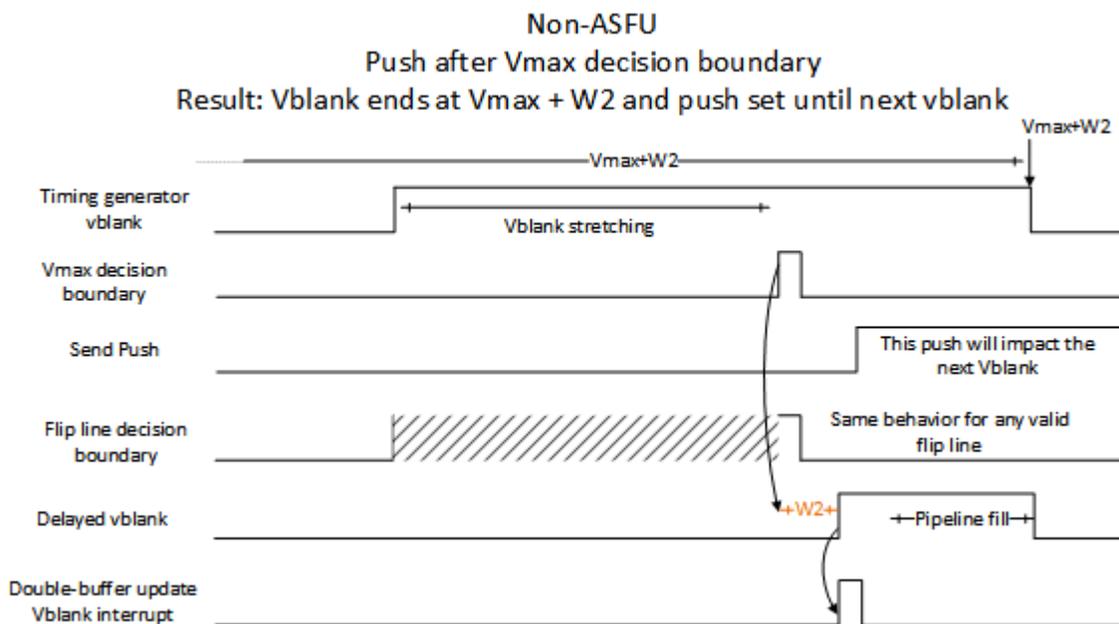
Maximum vblank terminated at flip line

If push happens before flip line decision boundary and flip line value is equal to V_{max} , the vblank will stretch to the max and the plane and pipe double-buffer registers update at the V_{max} and flip line decision boundary.



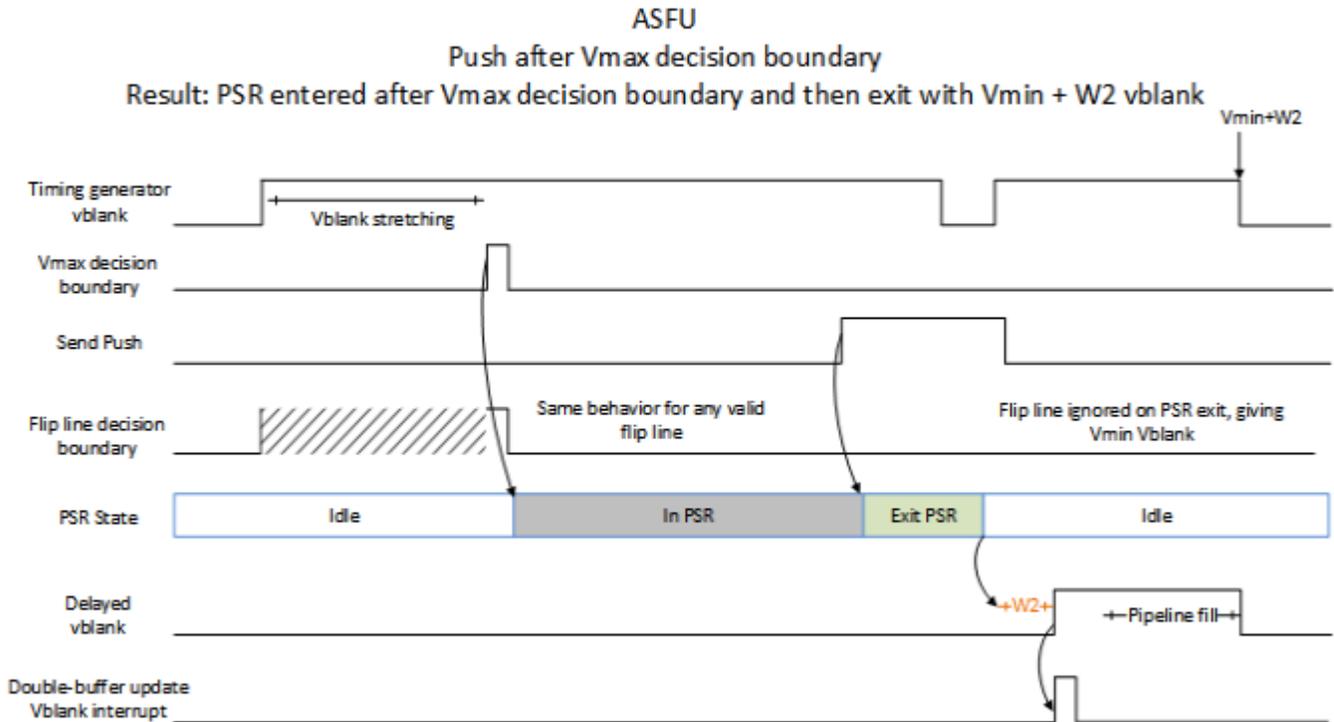
Maximum vblank terminated at V_{max}

If PSR is disabled and push happens after V_{max} decision boundary, the vblank will stretch and the plane and pipe double-buffer registers update at the V_{max} decision boundary.



Maximum vblank enters PSR

If PSR is enabled (for ASFU) and push happens after Vmax decision boundary, the vblank will stretch and enter PSR after the Vmax decision boundary. The push then causes PSR exit with minimum vblank (flip line ignored at PSR exit) and the plane and pipe double-buffer registers update at the start of vblank.



Programming Sequence

1. Enable VRR
 - This can be done after mode set (called VRR Enable Sequence 1), or during mode set in the step that configures transcoder timings and other pipe and transcoder settings so that VRR is enabled from the first frame (called VRR Enable Sequence 2).
1. Configure VRR timing and control registers
 - TRANS_VRR_CTL Pipeline Full Override must be set and the FrameStart to Pipeline Full LineCount programmed.
 - TRANS_PUSH Push Enable must be set.
 - If max shift, TRANS_VRR_CTL Ignore Max Shift and Flip Line Enable must be clear, else Ignore Max Shift and Flip Line Enable must be set.
2. If adaptive sync SDP use case
 - Program TRANS_VRR_VSYNC
 - Set "Adaptive Sync SDP Enable" in VIDEO_DIP_CTL register
3. Set TRANS_VRR_CTL VRR Enable
4. If ASFU, enable PSR - See Panel Self Refresh chapter

2. Screen updates, repeat as needed for each update
 1. Program the double-buffer registers that need updating
 - There will be a double-buffer update at Vmax decision boundary if ASFU is not used, so care must be taken to ensure multiple resource programming does not straddle the double-buffer update point and cause non-atomic updates.
 - Options
 - Use ASFU so that PSR is entered at Vmax.
 - Align programming to happen before the Vmax decision boundary.
 - Use double buffer disable/stall mechanisms to stall double buffering until all programming is complete.
 2. If not max shift, program Flip Line value in TRANS_VRR_FLIPLINE (this can be programmed multiple times before the next step).
 - Flip Line value cannot be changed between when the push is initiated and the push is done.
 3. Set TRANS_PUSH Send Push
 4. Hardware will clear the Send Push when the double buffer update happens
 5. Poll for TRANS_PUSH Send Push cleared
3. Disable VRR
 1. If ASFU, disable PSR - See Panel Self Refresh chapter
 2. Clear TRANS_VRR_CTL VRR Enable
 3. Stop setting push
 - Push is not needed when VRR is disabling because frames will terminate automatically.
 - Setting push after VRR is disabled will cause the push to be held until later VRR enabling. That is not validated and not supported.
 4. Poll for VRR live status indicating VRR has disabled.
 - VRR live status takes one frame to change after VRR mode is disabled.
 - Depending on VMAXSHIFT, the vblank length may keep changing for several frames after VRR mode is disabled, which is interrupted by disabling the transcoder or re-enabling VRR.
 - VRR double buffer update interrupt will remain active until VRR live status is de-asserted.
 1. TRANS_PUSH Push Enable can be cleared at this point or later
 2. VRR can be re-enabled by returning to step 1.
 3. If port will be disabled, continue to mode set disable sequence.
 - Hardware may be capable of transcoder disable with VRR enabled, but that is not validated and not supported.

VRR with Port Sync Mode

Port sync has multiple transcoders running synced together. The primary transcoder will send sync signals to the secondary transcoders.

Program the VRR registers the same on both transcoders when enabling VRR.

VRR_CTL VRR Enable and TRANS_PUSH Push Enable must be set on primary and secondary transcoders. The other registers may not all be necessary for the secondary transcoders, but programming them differently is not validated and not supported.

When sending screen updates, only the primary transcoder VRR needs to be updated. Only set push on the primary transcoder and only update the flip line value on the primary transcoder.

Starting VRR from Nominal Refresh Rate

1. Driver determines the panel timings and the highest, lowest, and nominal refresh rates to support
 2. Driver does mode set to the highest pixel rate and highest refresh rate timings (should be close to CVT1.2 RB), except it extends the vertical total so that the resulting refresh rate is nominal.
 3. Driver enables VRR with Vmax for the lowest refresh rate and Vmin for the highest refresh rate. Vmin = the vertical total from the mode set before extending to nominal refresh rate.
 4. Hardware will vary refresh rate between the Vmax and Vmin based on the timing of pushes.
 5. Hardware will also vary refresh rate between the Vmax and Vmin based on the VMAXSHIFT.
1. When VRR is disabled the refresh rate will return to Vtotal (nominal).
 - Steps 2 and 3 may be separated (VRR Enable Sequence 1) or combined so that VRR is enabled during the mode set (VRR Enable Sequence 2).
 - The pixel rate and horizontal timings are programmed to match the highest refresh rate and do not change.

Starting VRR from Maximum Refresh Rate

1. Driver determines the panel timings and the highest, lowest, and nominal refresh rates to support.
 2. Driver does mode set to the highest pixel rate and highest refresh rate timings (should be close to CVT1.2 RB).
 3. Driver enables VRR with Vmax for the lowest refresh rate and Vmin for the highest refresh rate. Vmin = vertical total from the mode set.
 4. Hardware will vary refresh rate between the Vmax and Vmin based on the timing of pushes and VMAXSHIFT.
1. When VRR is disabled the refresh rate will return to Vtotal (same as Vmin).
 2. It will take several frames to return, depending on VMAXSHIFT.
 - Steps 2 and 3 may be separated (VRR Enable Sequence 1) or combined so that VRR is enabled during the mode set (VRR Enable Sequence 2).

- The pixel rate and horizontal timings are programmed to match the highest refresh rate and do not change.

DPCD

Video timing information in the DP MSA (main stream attribute packet) data is designed for video modes where the parameters are static. For modes such as VRR that dynamically change the video timing, the main stream attribute fields cannot be used. Therefore, panel DPCD registers need to be appropriately programmed for VRR use cases.

VRR Restrictions
<ul style="list-style-type: none"> • Interlaced mode, DRRS, Stereo 3D, and PSR2 are incompatible with VRR.
<ul style="list-style-type: none"> • $V_{min} = \text{window 2} + V_{active} + \text{framestart delay} + \text{pipeline fill}$
<ul style="list-style-type: none"> • V_{max} must be greater than V_{min}. $V_{min} < V_{max}$
<ul style="list-style-type: none"> • After push, software must wait for flip done before starting another push.
<ul style="list-style-type: none"> • PSR is supported with VRR for the Adaptive Sync Frame Update (ASFU). • ASFU is mutually exclusive with max shift. • Flip Line must always be enabled when VRR is enabled, except if max shift is enabled. • Flip Line is mutually exclusive with max shift. • Flip Line value cannot be changed between when the push is initiated and the push is done. • Flip Line value must be less than or equal to V_{max}. $\text{Flip Line} \leq V_{max}$ • Flip Line value must be greater than V_{min}. $V_{min} < \text{Flip Line}$. <ul style="list-style-type: none"> • Flip Line value equal to V_{min} is not supported and causes vblank terminated by flip line to have an extra line.

Adjusting for Delayed Vblank Window 2

If vblank is delayed to create window 2 (see Transcoder Timings section), then VRR programming needs to be adjusted to account for window 2. Hardware does not calculate window 2 in the decision boundaries. Window 2 is inserted before the delayed vblank asserts, and is preserved when vblank stretches for VRR, so it has the impact of pushing out vblank by window 2 size. The window 2 size then needs to be subtracted when programming the VRR vblank affecting registers to get the desired result.

$VRR\ V_{min} = \text{Desired } V_{min} - \text{window 2 size}$

$VRR\ V_{max} = \text{Desired } V_{max} - \text{window 2 size}$

$\text{Flipline} = \text{Desired flipline} - \text{window 2 size}$

FrameStart to Pipeline Full LineCount must be calculated using the adjusted VRR V_{min} .

Transcoder Asynchronous Selective Update

This feature extends ASFU to partial frame updates and adds the capability of updating one region of the display using the ASFU framework. Only one update region per frame is supported and the capability is only supported with manual tracking mode/selective fetch. A Metadata packet will transfer the co-ordinates of the update region as a proprietary secondary data packet, prior to sending the active data for the frame. All the framework available with ASFU will continue to apply to ASU. If compression is enabled, ASU mandates that the PPS sent to the panel be fixed and it should not change on the fly. Hence, DSS control registers once configured, should not be modified for partial updates. ASU does not support CoG use cases.



- Dashed lines represent DSC slice
- Solid line represents full frame size
- Greyed out area not fetched
- Green area is updated region
- Pipe scaling would require fetch of extra slices around the updated region slices in order to have the data to fill scaler filters before entering the updated slices.

ASU flows

1. Mode set enable.
2. PSR enable, wait for 1 Vblank.
3. If selective update, SW updates pipe source size, Vblank start, Vactive and slice_row per frame as per SU size.
4. Set TRANS_PUSH send PUSH.
5. SW to poll for push done bit.
6. HW will consume pipe source size and slice_row per frame @VBI and Vblank start and Vactive @Vtotal.
7. For subsequent selective updates, SW to poll on scanline interrupt, 1st line and go to step (3).

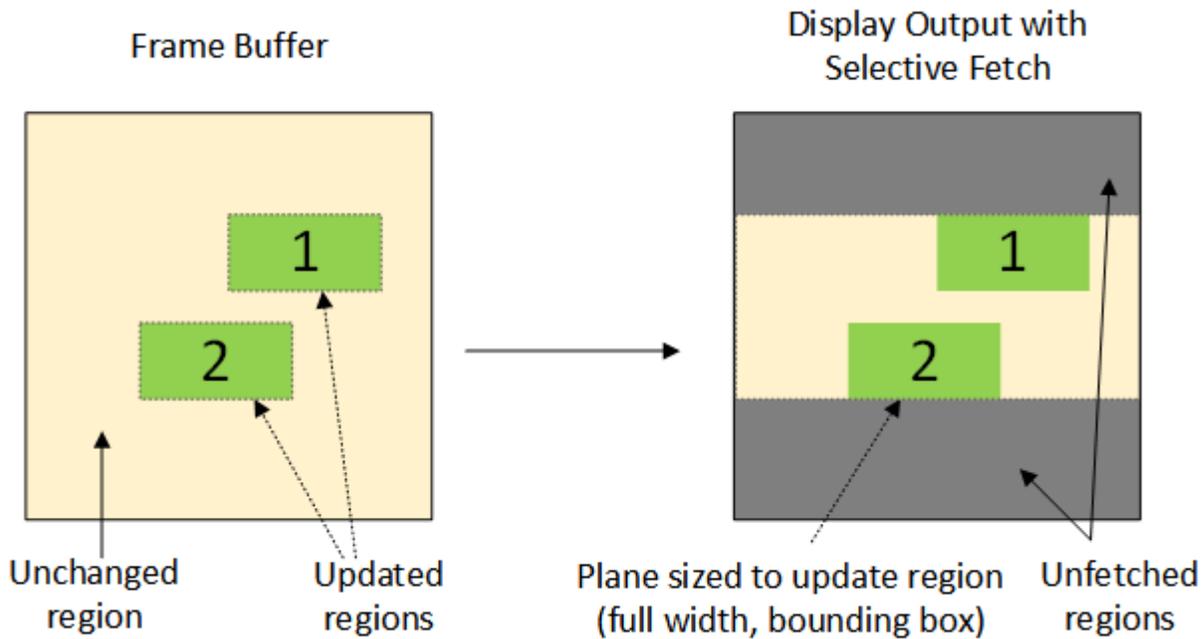
8. If PSR disable, wait for scanline interrupt, 1st line. Restore the original timing parameters.

The complete sequence is as follows.

1. Mode set enable
2. Push enable
3. DB Vactive enable
4. VRR enable
5. PSR enable, wait for 1 Vblank
6. Selective update
7. PUSH
8. Poll for PUSH done
9. SW to poll on scanline interrupt, 1st line for future selective updates
10. PSR disable
11. Restore original timing parameters
12. Program Vtotal to Vnominal.
13. VRR disable
14. DB Vactive disable
15. PUSH disable

Selective Fetch

Selective fetch is a power saving feature to reduce display engine use of memory bandwidth by only fetching (reading from memory) the updated regions of the frame buffer and sending those updated regions to a panel with a remote frame buffer and selective update capability. Software selects the amount of fetch by adjusting the size and position of the display planes to fit the updated regions in the frame, and configures panel selective update manual tracking to output that region to the panel. The result is that the bounding box around the updated regions is read from memory and delivered to the panel remote frame buffer and the other regions are not fetched and not sent to the panel, saving memory bandwidth, IO, and panel power.



Restrictions

- Supported with only full line width update region. Software must adjust the update region to full width.
 - Supported with only a single update region per frame. Software must create the bounding box around all regions.
 - Not supported with async flips. The plane size and position cannot be changed with async flips, so selective fetch cannot be used. Software must output a full frame for async flips.
 - Not supported with command streamer flips. Command streamer flips do not update the selective fetch plane registers without adding in extra `LOAD_REGISTER_IMMEDIATE` commands, which are too complicated.
 - Not supported with plane rotation. Software calculations to adjust update region coordinates are too complicated.
 - Not supported with plane or pipe scaling. Software calculations to account for extra lines of scaler filter input and adjusted scale factor and filter phase are too complicated.
 - Not fully supported with LACE/LDPST. The histogram accumulates across the frame and gives incorrect results when only part of the frame is updated. Software adjusts the histogram taking algorithm to get partial support.
-
- Not fully supported with DPST. The histogram accumulates across the frame and gives incorrect results when only part of the frame is updated. Software adjusts the histogram taking algorithm to get partial support.
-
- Supported with PSR2.
-
- Not supported with DSC.

Hardware

Hardware support of selective fetch is through the panel selective update feature with manual tracking, the planes and cursor selective fetch registers, and the full frame update triggers.

Software

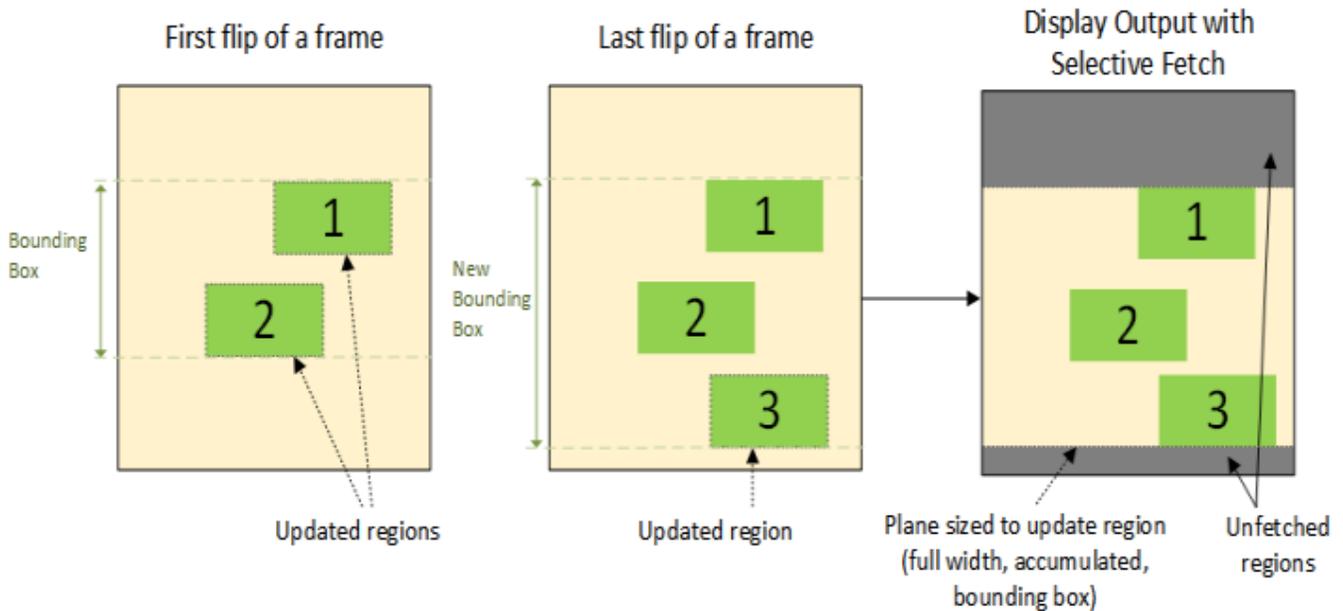
Software support of selective fetch

- Calculate the updated region
- Program the planes and cursor selective fetch registers to adjust the sizes, positions, and panning to fit the update region.
- Program the selective update manual tracking register to fit the update region.
- Program planes, cursor, and manual tracking together to atomically update.
- Trigger full frame updates when selective fetch cannot be used, such as when updating DPST image enhancement.

Update Region

Selective fetch requires software to know which areas of the screen are being updated. The update region coordinates are then used to adjust the planes and cursor to limit the fetch, and to notify the panel of which region has been selectively updated. The update region must be full width and just a single region per frame. Software must calculate the region as a bounding box to encompass all the updated sub-regions.

When there are multiple screen updates (flips) within a single frame, the update region can be different for each flip, but the final update region needs to be a bounding box that encompasses the update regions for all the flips in the frame. Hardware does not accumulate the update region across flips; it only uses the last programmed values. Software must either accumulate the update regions and program hardware with the accumulated result at each flip and reset accumulation at each frame, or trigger a full frame update when multiple flips happen within a single frame.



If there is a flip where the update region is unknown (e.g. application doesn't support this kind of tracking) or there is update that impacts the entire frame (e.g. pipe color correction programming), the update region is the full frame.

Update Region Alignment

Some features require the update region to be aligned to boundaries, then software needs to expand the update region to meet the alignment, increasing the amount of fetch.

For PSR2 selective update, the frame is divided into blocks of four scan lines each. The update region must be expanded so it aligns to the 4 line groups of transcoder vertical active.

Adjusting Plane and Cursor Size and Position

The selective fetch registers are an extra set of registers for each plane and cursor where software programs the adjusted size, position, offset (panning), and control (just the enable) to be used when doing a selective fetch. The regular (non SEL_FETCH) plane registers are used for full frame updates. Both sets must be updated with each flip, so that any frame will appear correct whether it is a selective fetch (uses the plane selective fetch registers) or a full frame fetch (uses the regular plane registers). Hardware will select between the sets of registers automatically. These registers are double-buffered and armed together with the regular plane and cursor registers.

SEL_FETCH_PLANE_CTL

PLANE_POS instances with SEL_FETCH prefix

PLANE_SIZE instances with SEL_FETCH prefix

PLANE_OFFSET instances with SEL_FETCH prefix

CUR_CTL instances with SEL_FETCH prefix

The plane selective fetch registers are programmed with the adjusted size and position of the planes, reduced down to the update region (must be the full width, accumulated, bounding box).

The plane adjustment calculations here are assuming the update region is relative to this plane's frame buffer. If the update is coming from the cursor or another plane, the position of cursor or other plane relative to this plane's frame buffer has to be calculated as the update region, or else the other plane and cursor changes can be simplified to do a full frame fetch if they are relatively infrequent. Other methods can be used to calculate using different starting coordinates.

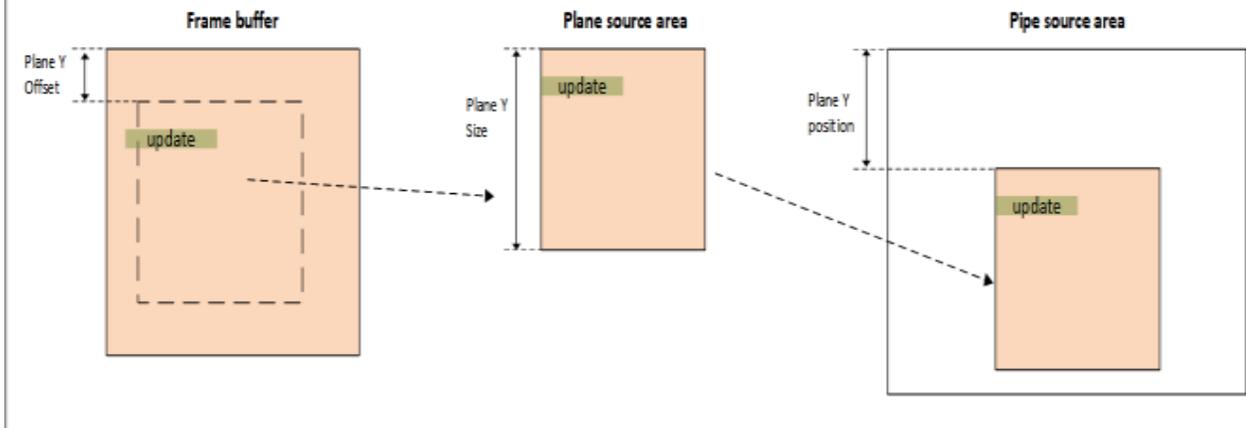
For each plane

- SEL_FETCH_PLANE_OFFSET Start Y Position = first line of update region relative to start of frame buffer for this plane
- Update offset within this plane = update region vertical offset in this plane = SEL_FETCH_PLANE_OFFSET Start Y Position - PLANE_OFFSET Start Y Position
- SEL_FETCH_PLANE_SIZE Height = vertical size of update region within this plane
- SEL_FETCH_PLANE_POS Y Position = PLANE_POS Y Position + Update offset within this plane
- SEL_FETCH_PLANE_CTL Selective Fetch Plane Enable = If update region is within this plane ? Enable : Disable
 - This plane is disabled when the update region is fully outside of this plane.
- The plane width/X/horizontal values of the position, size, and offset are not changed with selective fetch, so program those fields the same in the normal plane registers and the selective fetch plane registers.

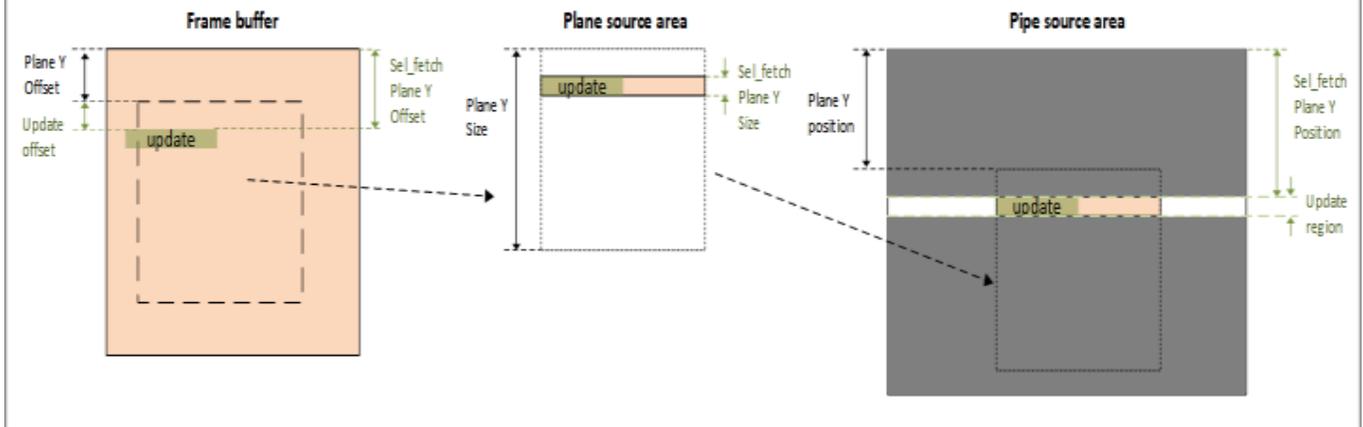
For cursor

- SEL_FETCH_CUR_CTL Cursor Mode Select = If update region, translated to pipe source coordinates, overlaps this cursor ? CUR_CTL Cursor Mode Select : Disable
 - Cursor size is not adjusted. Cursor is just disabled when the update region is fully outside of cursor.
 - Program the other fields in SEL_FETCH_CUR_CTL to match CUR_CTL.

Regular (full frame) Fetch

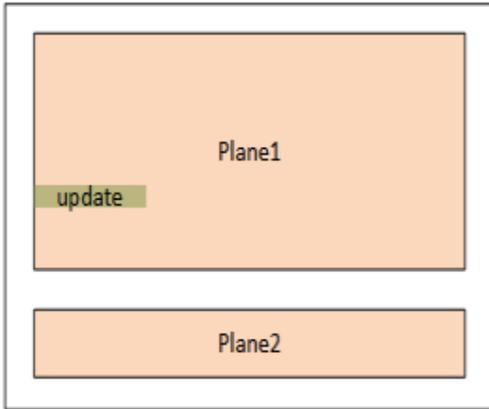


Selective Fetch



Multi-Planes Separate Update

Regular Fetch

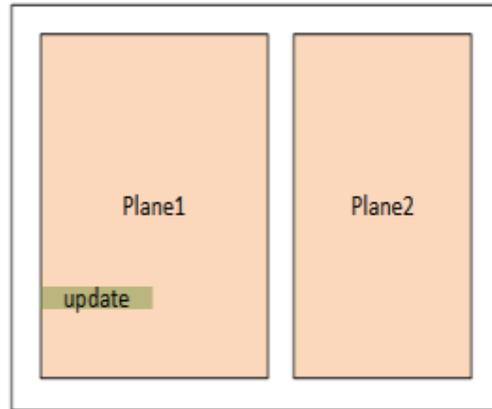


Selective Fetch

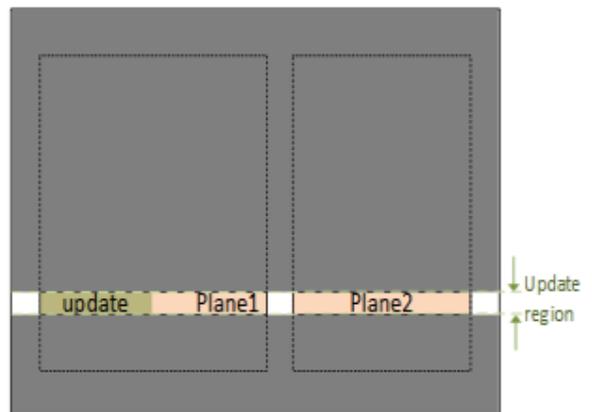


Multi-Planes Overlap Update

Regular Fetch

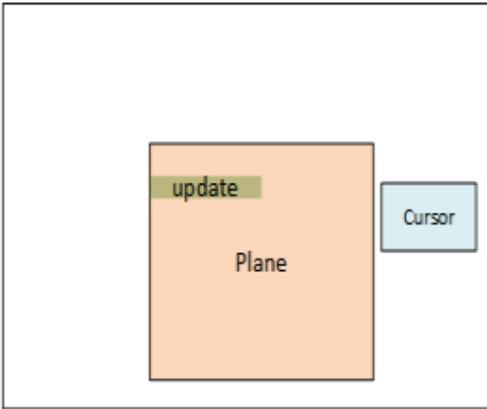


Selective Fetch

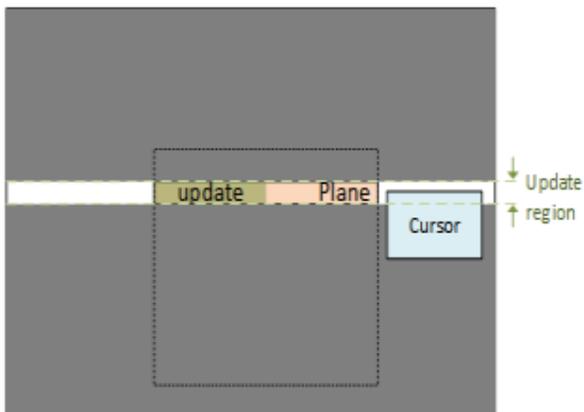


Cursor Inside Plane Update

Regular Fetch

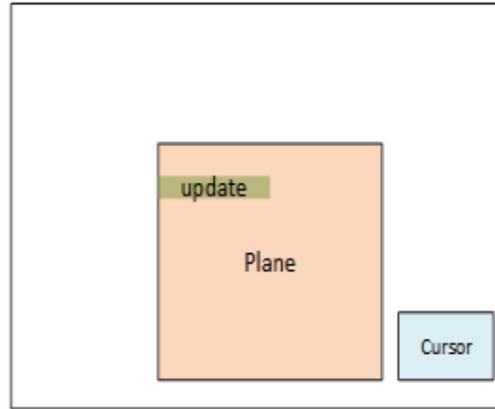


Selective Fetch

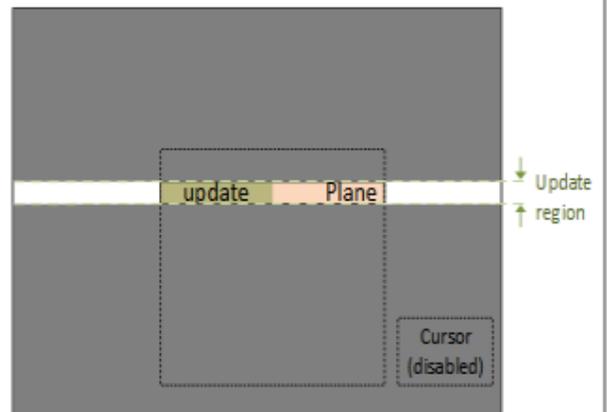


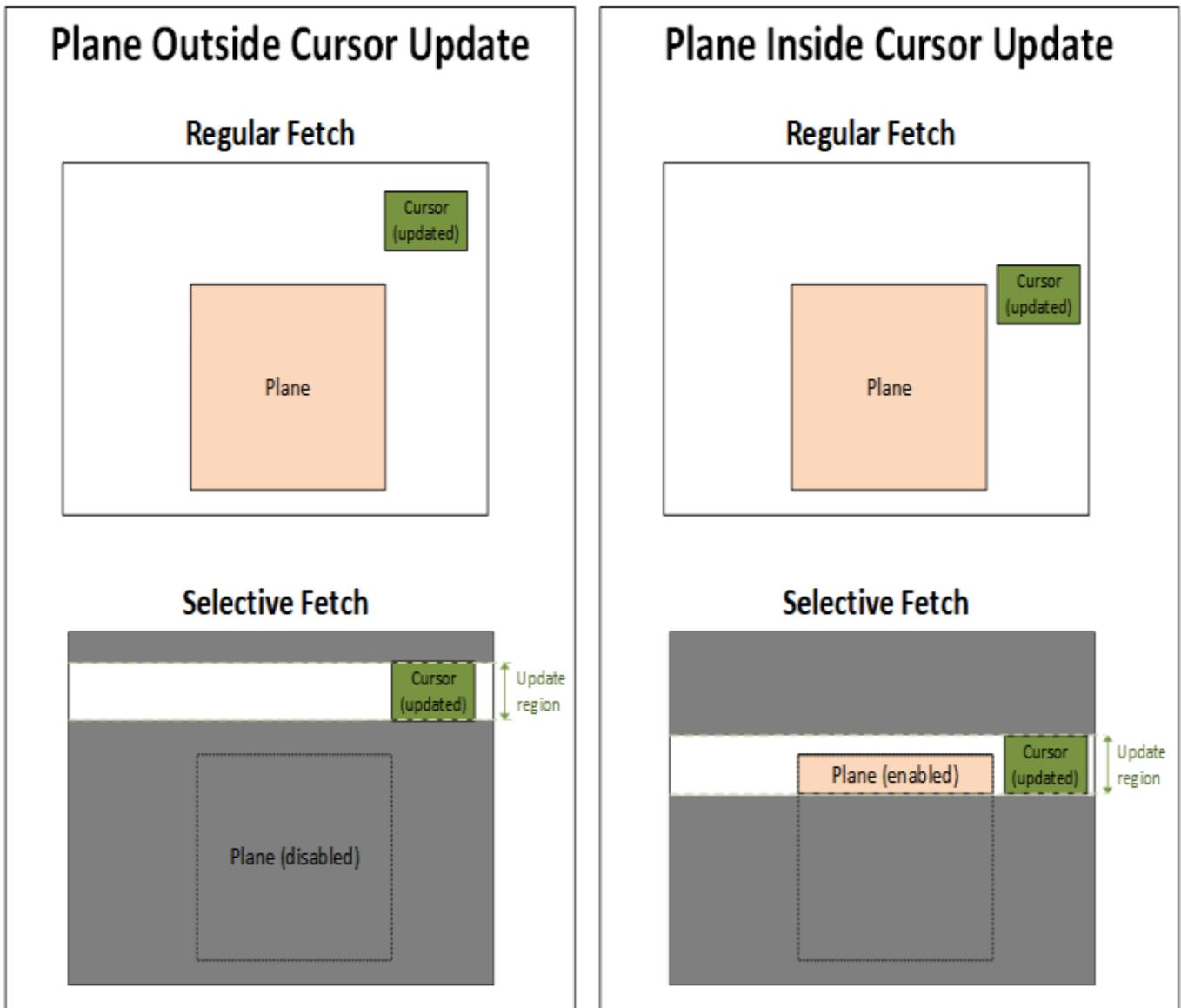
Cursor Outside Plane Update

Regular Fetch



Selective Fetch





Selective Update

The selective fetch programming must be combined with panel selective update (SU) programming so that only the selective fetch update region is sent to the panel.

Calculate the update region translated to transcoder vertical active coordinates, then program **PSR2_MAN_TRK_CTL** with that SU region.

- Depending on the project generation, the SU region address is programmed with blocks of either 4 line or 1 line granularity, and with different starting and ending behavior, as noted in the manual tracking register.
- PSR2_MAN_TRK_CTL SF Continuous Full Frame and SF Single Full Frame must be 0 for the SU region to be used.

Full Frame Updates

Sometimes software needs to send a full frame update, such as when async flips happen.

- Software sets PSR2_MAN_TRK_CTL SF Single Full Frame to trigger one full frame to be sent. Hardware will clear the bit after sending the frame.
- Software sets PSR2_MAN_TRK_CTL SF Continuous Full Frame to trigger full frames to be continuously sent until software clears the bit.
- The full frame fields do not cause a frame to be sent, they only set the frame to full size. To send a frame a flip or transcoder push is still needed.

When a full frame is sent, hardware uses the regular plane and cursor registers and not the SEL_FETCH registers, and sets the update region to the full frame, ignoring the programmed SU region.

Hardware will spontaneously send full frame updates on PSR exit, without software necessarily being aware of it, so the regular plane and cursor registers must always be programmed for the full frame.

DPST Handling

The DPST histogram is incorrect when used with selective fetch. DPST will usually give more power savings than selective fetch. Software can still use DPST with selective fetch, at least some of the time, by enabling the histogram only on selected frames (chosen periodically or algorithmically) and triggering selective fetch full frames while taking the histogram. Based on the histogram result, software can decide if the histogram needs to be taken more frequently for better power savings, with more full frames.

Selective Fetch Programming Sequence

1. DisplayPort enable mode set sequence.
2. For PSR2, configure PSR2_MAN_TRK_CTL SF Partial Frame Enable=1, and depending on project, PSR2_MAN_TRK_CTL PSR2 Manual Tracking Enable=1, then enable PSR2_CTL.
3. For each screen update
 1. If selective fetch for this frame, atomically update planes, cursor, selective fetch, and selective update registers, referring to the Adjusting Plane and Cursor Size and Position and Selective Update sections for the values.
 - All the registers must update atomically together in the same frame. Refer to the Double Buffer Control section, sequence for synchronizing double buffer updates.
 - The fields to allow double buffer stalling are located in PSR2_MAN_TRK_CTL, regular plane control, and regular cursor control registers.
 - The regular plane and cursor registers must be programmed for the full frame size because hardware can spontaneously send a full frame update.
 2. Else, send a full frame update, referring to the Full Frame Updates section.

YUV 4:2:0 Support

Overview

DP and HDMI ports support YUV 4:2:0 output.

YUV 4:2:0 has two modes of operation - bypass mode and full blend mode. Switching between full blend and bypass mode is supported at frame boundaries.

Enabling or disabling YUV 4:2:0 pipe output requires a mode-set.

Full blend mode (HDMI)

In this mode the complete display pipeline is enabled. Input data format can be YUV or RGB. Software has to ensure that the color space is YUV before post blender scaling, since the post blender scaler is only going to be doing the subsampling from YUV 4:4:4 to YUV 4:2:0 (Y & UV scaling). Post blender Y and UV scalers will be used in 5x3 configuration (horizontal taps = 5, vertical taps = 3). A packer block will pack the YUV 4:2:0 data as needed on the HDMI link.

YUV Full blend mode programming requirements (HDMI)

- A scaler should be enabled and bound to pipe.
- The pipe scaler supports downscaling to less than 1.5 (source/destination) in the horizontal direction. However, the pipe scaler can only support downscaling of 1.0 or less in the vertical direction (i.e. no downscaling) without the risk of underflows. A vertical scale factor between 1.0 and 1.5 (non-inclusive) could result in underflows.
- The pipe vertical active display size should be a multiple of 2.
- The horizontal window size must meet the HDMI restriction of a minimum of 128 Y component transfers per scan line.

Additional restrictions (HDMI)

- When YCbCr 4:2:0 pixel encoding is active, pixel repetition is not permitted.
- The transport of 4:2:0 pixels for interlaced video formats is not supported.
- The horizontal active, sync start, sync end, and total should be a multiple of 4 for 8/10/12 bpc.
-

Full blend mode (DP)

In the case of DP/eDP data will be formatted for YUV 4:2:0 before the pixel shifter. Operation is otherwise the same as for Full blend mode with HDMI.

Additional restrictions (DP)

- The transport of 4:2:0 pixels for interlaced video formats is not supported. The other HDMI additional restrictions do not apply.
- This mode is intended for use with external DP to HDMI converters, so VRR, MST, and DP port sync mode are not supported with it.

Bypass mode (HDMI)

In this mode the display pipeline is completely bypassed. YUV 4:2:0 planar data is fetched from memory, then packed according to the YUV 4:2:0 requirements of HDMI 2.0. All the display features in the pipeline including rotation, horizontal flip, blending, plane scaling, and pipe scaling, filtering, dithering, and range correction will not be supported and should not be enabled by software.

Plane size must match pipe active size when bypass mode is enabled.

Only one plane on the pipe can be enabled when bypass mode is enabled.

PIPE_MISC Pixel Rounding must be set to Truncate.

Use only plane 1 for UV and plane 6 for Y

Bypass mode (DP)

In this mode the display pipeline is completely bypassed. YUV 4:2:0 planar data is fetched from memory, then packed according to the YUV 4:2:0 requirements of DP 1.3. Operation and restrictions are otherwise the same as for Bypass mode with HDMI.

Transcoder Video Data Island Packet

Data Island Packet (DIP) is a mechanism that allows data to be sent over a digital port during blanking, according to the HDMI and DisplayPort specifications. This includes header, payload, checksum, and ECC information.

Each type of Video DIP will be sent once each frame while it is enabled.

Video DIP
VIDEO_DIP_CTL
VIDEO_DIP_DATA
VIDEO_DIP_GCP
VIDEO_DIP_ECC
VIDEO_DIP_DRM_DATA
VIDEO_DIP_DRM_ECC
VSC_EXT_SDP_CTL
VSC_EXT_SDP_CONF
VSC_EXT_SDP_HEADER
VSC_EXT_SDP_DATA

Supported DIPs

HDMI	DP	eDP
General Control Packet (GCP) Auxiliary Video Information (AVI) Source Product Description (SPD) Vendor Specific (VS) Gamut Metadata Packet (GMP)	Gamut Metadata Packet (GMP) Video Stream Configuration (VSC)	Video Stream Configuration (VSC)
DRM		
	Picture Parameter Set (PPS)	
		Gamut Metadata Packet (GMP)
	VSC Extension SDP (VSC EXT SDP)	VSC Extension SDP (VSC EXT SDP)

Construction of DIP for AVI, VS, or SPD (HDMI only):

Dword	Byte3	Byte2	Byte1	Byte0
0	Reserved	HB2	HB1	HB0
1	DB3	DB2	DB1	DB0
2	DB7	DB6	DB5	DB4
3	DB11	DB10	DB9	DB8
4	DB15	DB14	DB13	DB12
5	DB19	DB18	DB17	DB16
6	DB23	DB22	DB21	DB20
7	DB27	DB26	DB25	DB24
8 (RO)	Reserved	Reserved	Reserved	HB ECC
9 (RO)	DB ECC 3	DB ECC 2	DB ECC 1	DB ECC 0

HB = Header Byte, DB = Data Byte, RO = Read Only

Dword	Byte3	Byte2	Byte1	Byte0
0	DP: HB3 HDMI: Reserved	HB2	HB1	HB0
1	DB3	DB2	DB1	DB0
2	DB7	DB6	DB5	DB4
3	DB11	DB10	DB9	DB8
4	DB15	DB14	DB13	DB12
5	DB19	DB18	DB17	DB16
6	DB23	DB22	DB21	DB20
7	DB27	DB26	DB25	DB24



Dword	Byte3	Byte2	Byte1	Byte0
8	DB31	DB30	DB29	DB28
9 (RO)	Reserved	Reserved	Reserved	DP: Reserved HDMI: HB ECC
10 (RO)	DP: Reserved HDMI: DB ECC 3	DP: Reserved HDMI: DB ECC 2	DP: Reserved HDMI: DB ECC 1	DP: Reserved HDMI: DB ECC 0
11 (RO)	DP: HB ECC 3 HDMI: Reserved	DP: HB ECC 2 HDMI: Reserved	DP: HB ECC 1 HDMI: Reserved	DP: HB ECC 0 HDMI: Reserved
12 (RO)	DP: DB ECC 3 HDMI: Reserved	DP: DB ECC 2 HDMI: Reserved	DP: DB ECC 1 HDMI: Reserved	DP: DB ECC 0 HDMI: Reserved
13 (RO)	DP: DB ECC 7 HDMI: Reserved	DP: DB ECC 6 HDMI: Reserved	DP: DB ECC 5 HDMI: Reserved	DP: DB ECC 4 HDMI: Reserved

HB = Header Byte, DB = Data Byte, DP = DisplayPort, RO = Read Only

Construction of DIP for VSC (DisplayPort only):

Dword	Byte3	Byte2	Byte1	Byte0
0	HB3	HB2	HB1	HB0
1	DB3	DB2	DB1	DB0
2	DB7	DB6	DB5	DB4
3	DB11	DB10	DB9	DB8
4	DB15	DB14	DB13	DB12
5	DB19	DB18	DB17	DB16
6	DB23	DB22	DB21	DB20
7	DB27	DB26	DB25	DB24
8	DB31	DB30	DB29	DB28
9 (RO)	HB ECC 3	HB ECC 2	HB ECC 1	HB ECC 0
10 (RO)	DB ECC 3	DB ECC 2	DB ECC 1	DB ECC 0
11 (RO)	DB ECC 7	DB ECC 6	DB ECC 5	DB ECC 4

HB = Header Byte, DB = Data Byte, RO = Read Only

Construction of DIP for DRM (HDMI Only)				
DWord	Byte3	Byte2	Byte1	Byte0
0	RSVD	HB2	HB1	HB0
1	DB3	DB2	DB1	DB0
2	DB7	DB6	DB5	DB4
3	DB11	DB10	DB9	DB8
4	DB15	DB14	DB13	DB12
5	DB19	DB18	DB17	DB16

6	DB23	DB22	DB21	DB20
7	DB27	DB26	DB25	DB24
8 (RO)	RSVD	RSVD	RSVD	HB ECC
9 (RO)	DB ECC 3	DB ECC 2	DB ECC 1	DB ECC 0

The audio subsystem is also capable of sending Data Island Packets. These packets are programmed by the audio driver and can be read by in MMIO space via the audio control state and audio HDMI widget data island registers.

Video DIP data write sequence:

1. Wait for 1 VSync to ensure completion of any pending video DIP transmissions
2. Disable the video DIP being updated (disable VDSC before updating PPS DIP)
3. Program video DIP data buffer registers for DIP being updated
4. Enable the video DIP

For MMIO programming of dynamic DIPs, Video DIP write sequence is as follows.

1. Wait to be outside the VBlank region so the following programming does not happen during the VBlank.
2. Program video DIP data buffer registers for DIP being updated.

The video DIP data and ECC buffers may be read at any time.

DIP data buffer registers must be programmed with valid data before enabling the DIP.

Partial DIPs are never sent out while the port is enabled. Disabling the DIP at the same time it is being transferred will result in the DIP being completed before the function is disabled.

Shutting off the port on which DIP is being transmitted will result in partial transfer of DIP data. There is no need to switch off the DIP enable bit if the port transmitting DIP is disabled.

When disabling both the DIP port and DIP transmission, first disable DIP and then disable the transcoder before the transcoder clock select is set to none.

Enabling a DIP function at the same time that the DIP would have been sent out (had it already been enabled) will result in the DIP being sent on the following frame.

For HDMI, even if no DIP is enabled, a single Null DIP will be sent at the same point in the stream that DIP packets would have been sent.

Notes on DIP use cases

GMP DIP can be used to transmit the CEA infoframe SDP since it allows full packet header customization.

There is no assumption in HW on the content of the SDPs (except when PSR is enabled).

For all non PSR cases HW transmits SDP using data programmed in the registers.

The controller does not use the data internally.

PPS is a unique packet as it is 128 bytes of data. No other SDP has this length.



The GMP is what is recommended for HDR metadata.

The VSC may also need to be transmitted with HDR content as it contains colorimetry information.

VSC Extension SDP Packets

VSC_EXT_SDP_CTL

VSC_EXT_SDP_CONF

VSC_EXT_SDP_HEADER

VSC_EXT_SDP_DATA

VSC extension packets can support frame synchronous meta data up to 1K bytes. VSC extension SDP packets add capability to support HDR Dynamic Meta Data. HDR dynamic meta data is supported for full frame updates only.

Key Assumptions

- Software should always write metadata in multiples of 8 DWs i.e., enough data to form one SDP packet.
- Hardware does not support padding of incomplete SDP packets.
- DP driver software has to determine whether a particular audio mode can be supported when transmitting extension packets.
- When Audio is enabled Meta Data will be sent only in horizontal active period of vblank.
- When Audio is disabled Meta Data will be sent both in horizontal active and horizontal blanking period of vblank.
- DSB or software should program the metadata and/or set the buffer ready bits when the relevant frame has started.
- Transcoder sends metadata in vblank region as long as at least one buffer is not empty and respective buffer done indication is set.
- Software should not start a new chain in the buffer while the previous chain data is still pending.
- If the chain doesn't fit in one buffer then software can continue to write the data to second buffer in ping-pong fashion. However, packet straddling across buffers is not allowed.
- DSB can continue writing to a buffer until either the chain is complete or 1KB data is programmed.

- If software doesn't set Index auto increment bit then it should program control register with correct index value prior to each data buffer write.
- If software doesn't set Index auto increment bit and start writing from non-zero index then it is the responsibility of software to send as much data as it can fit in (256- Index) entries. Hardware will however start reading from index0 (and continue to read till the pointers match).
- Index value rolls over after reaching maximum possible value.

- Transcoder clock must be enabled while VSC_EXT_SDP_CTL is enabled or VSC_EXT_SDP_DATA is being written.

- TRANS_DDI_FUNC_CTL must be enabled while VSC_EXT_SDP_DATA is being written.

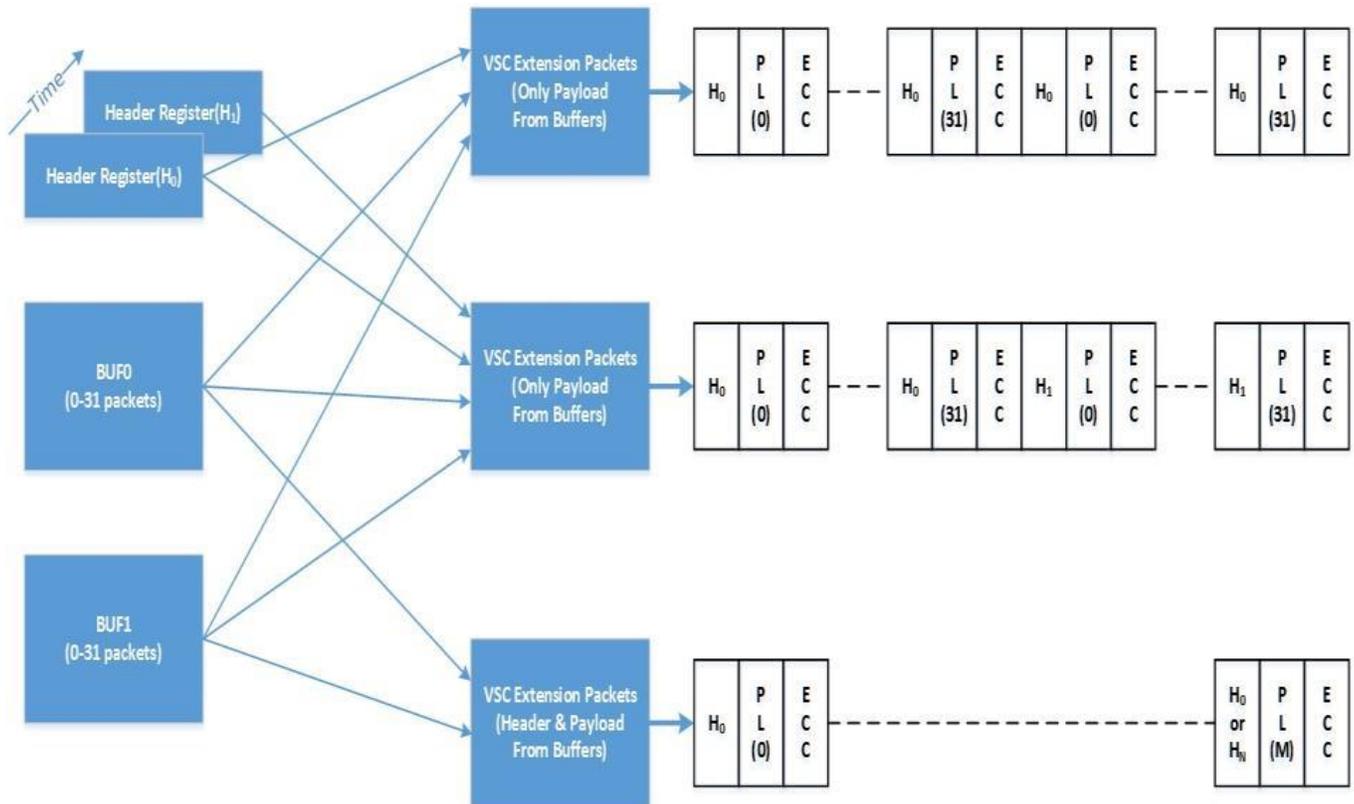
Programming sequence

There are 3 different ways of programming VSC extension packets based on the following considerations.

1. Packet header is from register and payload in buffer
2. Packet header and payload in buffer
3. Multiple VSC extension headers per frame with multiple chains per header

Type of Sequence	Max packets per Buffer
Only payload in buffer	32
Header and payload in buffer	28

The diagram below shows the final VSC extension packets from these 3 different programming methods.





Packet header is from register

In this use case, steps to program any buffer (BUF0 or BUF1) or as follows.

1. Program CTL Register - Auto Index Increment in VSC_EXT_SDP_CTL0 [14] (VSC_EXT_SDP_CTL1 [14]) or register bit.
2. Program BUF Register - Meta Data in VSC_EXT_SDP_DATA_0 (or VSC_EXT_SDP_DATA_1) register.
3. Set BUF ready bit - VSC_EXT_SDP_CTL0 [16] (or VSC_EXT_SDP_CTL1[16])

The complete sequence is as follows.

1. Program number of packets per chain in VSC_EXT_SDP_CONF[9:0] register.
2. Program Header values in VSC_EXT_SDP_HEADER register.
3. Program BUF0
4. Program BUF1 if needed (payload > 1 KB)
5. Repeat steps 3 and 4 after poll/wait for corresponding BUF empty indication - VSC_EXT_SDP_CTL0[24] (or VSC_EXT_SDP_CTL1[24])

Packet header is from buffer

The number of packets per chain is implicit in the buffer.

In this use case, steps to program any buffer (BUF0 or BUF1) or as follows.

1. Program CTL Register - Auto Index Increment in VSC_EXT_SDP_CTL0 [14] (VSC_EXT_SDP_CTL1 [14]) or register bit.
2. Program BUF Register - Header & Meta Data in VSC_EXT_SDP_Data0 (or VSC_EXT_SDP_Data1) register.
3. Set BUF ready bit - VSC_EXT_SDP_CTL0 [16] (or VSC_EXT_SDP_CTL1[16])

To set this up, program "middle of chaining" bit in the header to '1' and enable "block mode" in VSC_EXT_SDP_DATA_MISC[22].

1. Program BUF0
2. Program BUF1 if needed (payload > 1 KB)
3. Poll/Wait for BUF0 Empty indication - VSC_EXT_SDP_CTL0 [24] .
4. Poll/Wait for BUF1 Empty indication - VSC_EXT_SDP_CTL1 [24] if BUF1 is programmed.
5. Repeat steps 1 to 4 until entire metadata is transmitted.

Multiple headers per frame

In this use case, steps to program any buffer (BUF0 or BUF1) are as follows.

1. Program CTL Register - Auto Index Increment in VSC_EXT_SDP_CTL0 [14] (VSC_EXT_SDP_CTL1 [14]) or register bit.
2. Program BUF Register - Meta Data in VSC_EXT_SDP_Data0 (or VSC_EXT_SDP_Data1) register.

3. Set BUF ready bit - VSC_EXT_SDP_CTL0 [16] (or VSC_EXT_SDP_CTL1[16]) .

The driver will program multiple chains with different number of packets and different header for each chain. As it requires programming of multiple headers per frame, driver has to poll for Chain done bit to avoid overwriting header register before hardware consumes it. Polling of Chain done instruction should be inserted after the programming of entire meta data for each chain is complete.

1. Program number of packets per chain in VSC_EXT_SDP_CONF[9:0] register.
2. Program Header Register - Header values in VSC_EXT_SDP_HEADER register.
3. Program BUF0
4. Program BUF1 if needed (payload per chain > 1 KB)
5. Poll/Wait for BUF0 Empty indication - VSC_EXT_SDP_CTL0 [24] .
6. Poll/Wait for Chain Done indication - VSC_EXT_SDP_CONF[31]
7. If BUF0 is empty but Chain is not done, poll/wait for BUF1 Empty indication. Otherwise, go to step 8.
8. Repeat steps 1 to 7 above until entire meta data is programmed.

DSB side programming considerations

If DSB is being used to program metadata with polling for buffer empty, then the DSB polling timeout (100 uS) will need to be increased, recommending $\geq 500\mu s$ if metadata programming is started in vblank, or \geq frame time if metadata programming is started in vertical active.

Transcoder DDI Function

TRANS_DDI_FUNC_CTL

TRANS_MSA_MISC

TRANS_DDI_FUNC_CTL2

Panel Self Refresh

This section is about Panel Self Refresh (PSR). PSR1 at one point was named Self Refreshing Display (SRD).

PSR1 enable sequence:

- Prerequisite: The associated transcoder and port are running and Aux channel associated with this port has IO power enabled.
 - Prerequisite: The associated transcoder, port and at least one plane are running.
1. Configure FBC host and render tracking. The FBC function does not need to be enabled in FBC_CTL.
 2. Program Transcoder EDP VSC DIP data with a valid setting for SRD/PSR.
 3. Configure and enable SRD_CTL.

PSR1 disable sequence:

- Prerequisite: The associated transcoder and port are running.
1. Disable SRD_CTL.
 2. Wait for SRD_STATUS to show SRD is Idle. This will take up to one full frame time (1/refresh rate), plus SRD exit training time (max of 6ms), plus SRD aux channel handshake (max of 1.5ms).

Adaptive Sync Frame Update (ASFU) combines PSR link disable mode, VRR, and PSR single frame update. ASFU allows the vertical blank size to vary as it does for VRR, but then it will enter PSR link disable after the maximum vertical blank, and on PSR exit it can send a single frame before re-entering link disable.

ASFU enable sequence:

- Prerequisite: The associated transcoder and port are running.
 - Prerequisite: The associated transcoder, port and at least one plane are running.
1. If SRD_CTL Adaptive Sync Frame Update was not disabled in step 3 of the previous ASFU disable sequence, then disable it.
 2. Configure FBC host and render tracking. The FBC function does not need to be enabled in FBC_CTL.
 3. Enable VRR - see Transcoder VRR Function chapter
 4. Program Transcoder EDP VSC DIP data with a valid setting for SRD/PSR.
 5. Configure PSR/SRD Registers and set SRD_CTL Adaptive Sync Frame Update.
 6. Set masking bits for PSR to use the VRR push mode
 - PIPE_MISC[Change Mask for Register Write] to '1'
 - PIPE_MISC2[ASFU Flip exception] to '1'.
 - PIPE A: 0x420B0[11] to '1' to mask flips from being frame update events
 - PIPE B: 0x420B4[11] to '1' to mask flips from being frame update events
 - PIPE C: 0x420B8[11] to '1' to mask flips from being frame update events
 - PIPE D: 0x420BC[11] to '1' to mask flips from being frame update events
 7. Enable bit 31 of SRD_CTL.

Note that ASFU can be used together with VRR flip line. See the Transcoder VRR Function page for flip line programming since that can be used independent of PSR and ASFU.

ASFU disable sequence:

- Prerequisite: The associated transcoder and port are running.
 - Prerequisite: The associated transcoder, port and at least one plane are running.
1. Disable SRD_CTL bit 31.
 2. Wait for SRD_STATUS to show SRD is in Idle. This will take up to one full frame time (1/refresh rate), plus SRD exit training time (max of 6ms), plus SRD aux channel handshake (max of 1.5ms).
 3. Disable SRD_CTL Adaptive Sync Frame Update, or alternatively, SRD_CTL Adaptive Sync Frame Update can remain enabled until the next ASFU enable sequence is entered.
 4. Disable VRR - see Transcoder VRR Function chapter

PSR2 enable sequence:

Prerequisite: The associated transcoder and port are running and Aux channel associated with this port has IO power enabled.

Prerequisite: The associated transcoder, port and at least one plane are running.

1. Configure FBC host and render tracking. The FBC function does not need to be enabled in FBC_CTL.
2. Program Transcoder EDP VSC DIP header with a valid setting for PSR2 and configure VIDEO_DIP_CTL VSC fields.
3. Enable GTC/Aux Frame Sync, if required.
4. Program Aux control register Fast Wake Sync Pulse Count to 8 pulses.
5. Configure Idle Frame, Selective Update Tracking Enable, and Y-coordinate fields, and enable PSR2_CTL.

When configuring PSR2_CTL, also configure the Fast Wake and IO Buffer Wake fields

- IO buffer wake lines = ROUNDUP(PSR2 IO wake time / total line time in microseconds)
- Fast wake lines = ROUNDUP(10us Aux Sync Pattern + 32 microseconds Aux PHY_WAKE transaction time / total line time in microseconds)
 - eDP standard calculation for Aux maximum Fast Wake from FW_SLEEP: 8us Aux preamble + 4us PHY_WAKE pattern + 20us tFW_EXIT_LATENCY
- For both fields limit the minimum to 7 lines and maximum to 12 lines

PSR2 IO wake time = 10us PHY latency + 32us eDP standard maximum Fast Wake from FW_SLEEP

- eDP standard calculation for IO maximum Fast Wake from FW_SLEEP: 11us of valid 8b/10b symbols during Aux preamble and first part of PHY_WAKE pattern + 1us tML_PHY_LOCKsu + 20us tFW_EXIT_LATENCY = 32us

This maximum of 12 lines with 42 wake means PSR2 requires total line time of at least 3.5us.

Panels with relaxed requirements may allow the wake times to be customized.

PSR2 disable sequence:

1. Program PSR2_CTL to clear PSR2 Enable.
2. Disable GTC if required.
3. Wait for PSR2_STATUS to show PSR2 is Idle. This will take up to one full frame time (1/refresh rate), plus exit training time (max of 6ms), plus aux channel handshake (max of 1.5ms), plus one more full frame time if exit began while in a capture frame.
4. If not LRR: Program PSR2_CTL to clear Selective Update Tracking Enable.

Do not enable PSR2 if (TRANS_VBLANK Vertical Blank Start - TRANS_VBLANK Vertical Blank End) < PSR2_CTL Block Count Number value in lines.

This minimum number of lines is required to give time to wake the IO when there is an update starting from the first active line.

This minimum block count of 8 lines means PSR2 requires vblank to be at least 8 lines.

Registers
SRD_CTL
SRD_STATUS
SRD_PERF_CNT
PSR_IMR
PSR_IIR
PSR_MASK
PSR_EVENT
PSR2_CTL
PSR2_MAN_TRK_CTL
PSR2_SU_STATUS
PSR2_STATUS

Transcoder WD Function

Register Links
TRANS_WD_FUNC_CTL
WD_STRIDE
WD_SURF
WD_TAIL_CFG
WD_TAIL_CFG2
WD Interrupt Bit Definition
WD_IMR
WD_IIR
WD_FRAME_STATUS
TRANS_HTOTAL
TRANS_VTOTAL
TRANS_CONF
WD_27_M
WD_27_N

WD is a transcoder for capturing display pipe pixel output to memory. It is generally intended for wireless display, but can be used for other functions.

Like the transcoders for wired ports, WD has a timing generator that initiates each frame, and it formats pipe pixel data output. Unlike wired ports, it does not pass the formatted pixel data to DDI or other port logic, but instead writes it to memory.

The WD timing generator uses TRANS_HTOTAL Horizontal Active and TRANS_VTOTAL Vertical Active to define the active pixel area that is written to memory, with the WD_SURF and WD_STRIDE specifying the memory surface attributes.

The maximum resolution is 3840x2160 60Hz, with any capture color format.

Setup

Enable WD following the Sequences for WD - Enable Sequence, and set TRANS_WD_FUNC_CTL Triggered Capture Mode Enable before or at the same time the WD Function Enable it set. This will put WD into the Triggered Capture Mode where it will wait for a capture to be triggered.

Running (after setup)

1. Trigger a frame capture by writing TRANS_WD_FUNC_CTL with the intended Frame Number and Start Trigger Frame = 1. The vertical blank will start, causing double buffered registers to update, and then capture will begin.
2. Find when capture completes by polling WD_FRAME_STATUS Frame Complete or waiting for WD Frame Complete interrupt.
 - There is no hardware timeout for the triggered capture mode. If capture does not complete within 50 milliseconds, write 1 to TRANS_WD_FUNC_CTL Stop Trigger Frame, then correct the configuration before starting another capture.
3. If using WD_FRAME_STATUS Frame Complete to find the frame completion, write 1 to that field to clear it in preparation for the next frame.
4. If another frame is needed
 - a. Update any configuration that needs to be changed for the next frame; like WD_SURF, WD_STRIDE, or pipe and plane double-buffered registers.
 - b. Goto 1, using an incremented Frame Number

Do not trigger a frame capture in the last frame when disabling WD. Wait for the last capture frame to finish, or hang and timeout, before disabling WD.

The frame number is used to synchronize capture and encode. Software selects the frame number and starts the capture with it. Transcoder WD includes the frame number in the pointer message sent to the encoder. The encoder reads that to align itself to the correct frame. The frame number should increment on each captured frame.

Selective Update

If only a portion of the screen is changing, WD can be configured to selectively update just that region in the capture buffer, saving power and bandwidth. VDenc will process the full buffer, or a portion, depending on its requirements.

The first capture is full frame size to establish the base image, then subsequent captures can be a selective update region.

The selective update region must be 16 line aligned to match pointer granularity.



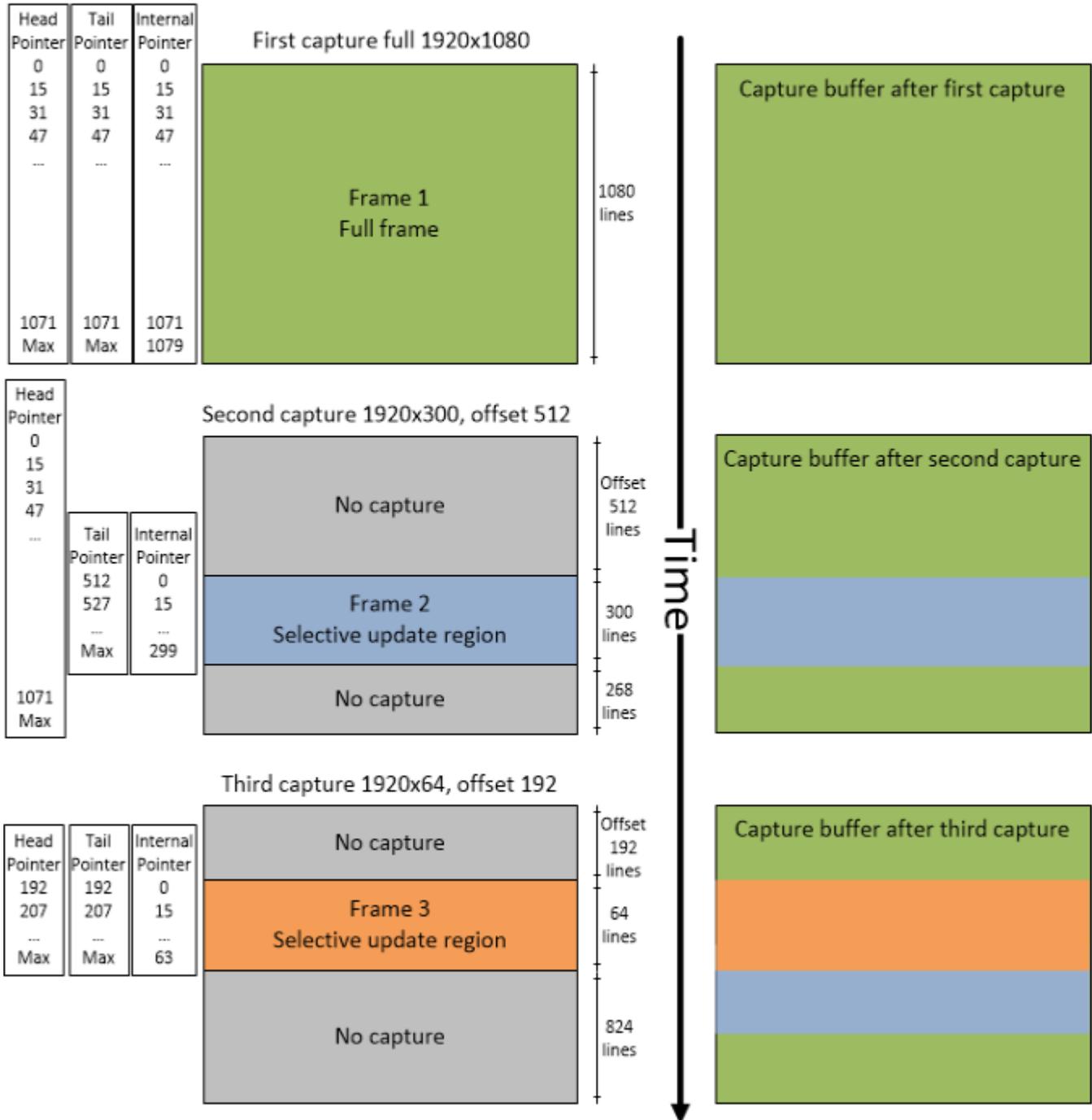
WD applies a programmable offset to the tail pointer to align the pointer to the selective update region within the full frame.

VDenc head pointer is always relative to the start of the full frame.

Programming for selective update region

- Program WD_TAIL_CFG2 Tail Pointer Offset to the number of lines offset from start of the frame to the selective update region, with 16 line granularity.
- Increase surface address to align to the start of the selective update region, with 16 line granularity.
- Reduce pipe and transcoder vertical sizes to match the size of the update region.
- Adjust plane size, position, and vertical panning to align with the update region.

Selective Update Example



Transcoder Port Sync

Feature Description

PORT SYNC is a transcoder level feature. This mode forces two or more transcoders to be in sync with one transcoder primary and one or more transcoder secondaries. In the case of DP/eDP, the primary is unaware that it is operating in Port Sync mode. Only the secondary is aware that it is operating in this mode. Hence, port sync mode is only enabled in the secondary transcoder.

Port Sync mode can be enabled with both DisplayPort SST and MST.

DP/eDP Port Sync Restrictions

1. The secondary and primary transcoders and associated ports must have identical parameters and properties.
2. They must have the same color format, link width (number of lanes enabled), resolution, refresh rate, dot clock, TU size, M and N programming, etc.
3. PSR/PSR2/ASFU/ASU would need to be disabled when port sync mode is enabled.
4. Port Sync Mode Primary Select must be programmed with a valid value when Port sync Mode is enabled.

Port Sync mode on MIPI

When setting up the DSI transcoders for dual link mode (a.k.a. Port Sync Mode) DSI Transcoder 0 is always the primary and DSI Transcoder 1 is always the secondary. The primary transcoder registers will be mirrored to the secondary transcoder when the Port Sync Mode is enabled within the TRANS_DDI_FUNC_CTL register of DSI Transcoder 0 (i.e. the **primary**). The secondary transcoder will ignore the programming within the majority of its registers when Port Sync Mode is enabled within the primary's TRANS_DDI_FUNC_CTL register. The only registers that will not be mirrored between the two transcoders are the registers that allow Software to initiate commands across the DSI Link to the Peripheral.

MIPI Port Sync Restrictions

1. Both DSI transcoders must be connected to the same PLL.

Transcoder Port Sync Support Table

Below is the complete list of primary and secondary transcoders that are supporting this feature.

The 1st column represents the primary transcoders and the 1st row represents the secondary transcoders.

Secondary ->	TC A	TC B	TC C	TC D	TC WD0	TC WD1	TC DS10	TC DS11
TC A	N	Y	Y	Y	N	N	N	N
TC B	Y	N	Y	Y	N	N	N	N
TC C	Y	Y	N	Y	N	N	N	N
TC D	Y	Y	Y	N	N	N	N	N
TC WD0	N	N	N	N	N	N	N	N
TC WD1	N	N	N	N	N	N	N	N
TC DS10	N	N	N	N	N	N	N	Y
TC DS11	N	N	N	N	N	N	N	N

Multichip Genlock

Cross-feature compatibility

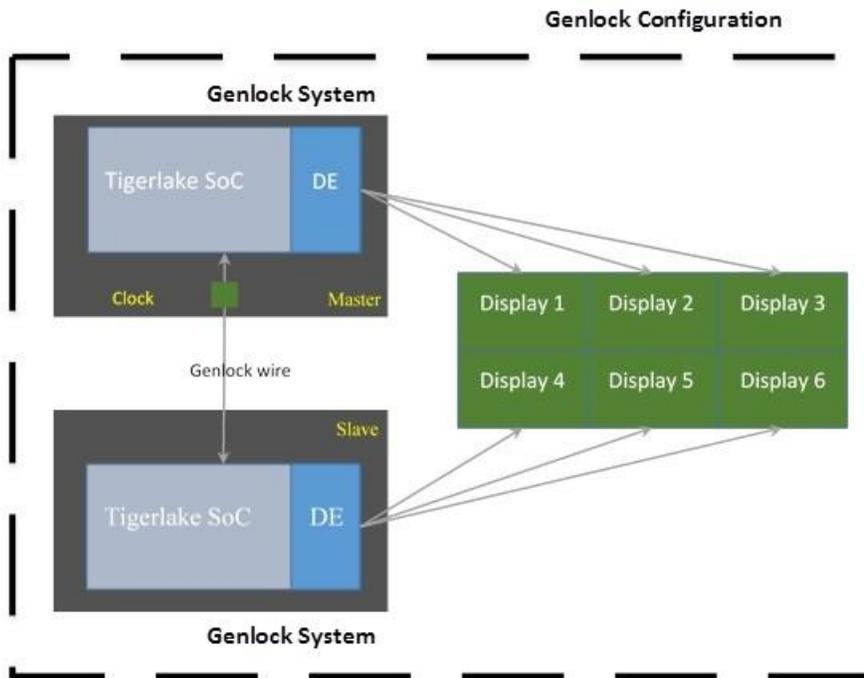
- Multichip genlock does not work with VRR (secondaries do not re-align vblank after enable mode set completes).

Description

With the multichip Genlock feature, platform/application software will synchronize frames (up to N-1 frame) across displays connected to multiple systems. The Nth frame synchronization is what HW portion of this Multi-chip Genlock will accomplish (i.e. all tiles display same scan line of same frame within +/- D pixels, as an example).

Below picture illustrates a wall of 2x3 displays connected to 2 systems in which one acts as genlock primary generating the clock and the other is genlock secondary receiving the clock from primary. DPLLs belonging to display 1, 2 and 3 are using the same PLL reference clock and that same reference clock propagates to DPLLs belonging to display 4, 5 and 6 over genlock wire as shown.

As part of multichip genlock configuration, each Genlock system below guarantees that (a) all display PLLs are running off of the same reference clock and (b) all displays receive their vertical reference (can be a separate wire or combined with reference clock) at the same time. However, flipping the right frame across displays is the responsibility of application software.



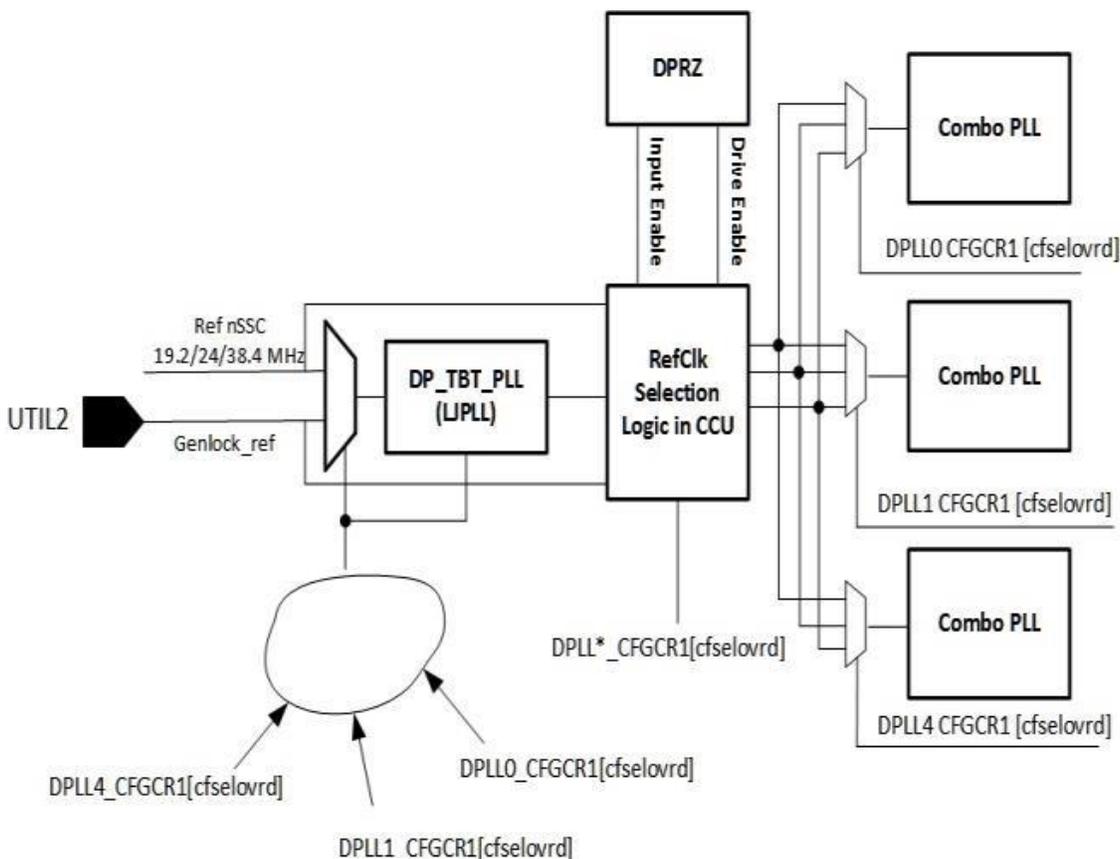
Key functional considerations

- South display is driving genlock enable and direction.
 - BIOS VBT can be configured to target unused GPIO pins.
- Within a genlock configuration, display driver will configure one transcoder as genlock primary (or secondary) and hooks up other transcoders as genlock secondaries.
- Only instances of identical platforms may be genlocked. Different types of platforms may not be mixed in a genlock topology.
 - For example, all platforms must use the same PLL reference clock frequency.
- All displays in a Multichip Genlock system must have the exact same timing requirement and support same transport protocol.
 - As an example, two panels with different 1080p timings are not allowed
- All genlock systems must train their links identically.
- Multichip Genlock is supported with both HDMI and DP (SST only) but not both on the same configuration. DP MST mode is not supported.
- Multichip genlock is not supported with software programmable Vtotal feature (LRR/LRR2).
- After genlock selection is made, chosen system enters genlock primary mode before any other system enters into genlock secondary mode.
 - Both primary and secondary ports finish training and have their transcoder enabled.
 - Primary pipe is enabled.
 - Secondary pipe is enabled.

- Runtime selection of Genlock primary or secondary requires a full mode set.
- Switching PLL reference for multichip Genlock operation requires mode set.
- Modeset across displays connected to the same genlock system can happen in multiple modeset calls.
- Genlock selection is not persistent across reboots.
- A genlock secondary will not generate vblank interrupts until after genlock sync is achieved.
- Genlock secondaries will exit first from genlock secondary mode to default mode before any genlock primary would.
- A genlock primary exits from genlock primary mode to default mode only after all genlock secondaries have switched to default mode.
- Multichip Genlock cannot be enabled at the same time port sync is enabled.
- Note: Genlock hardware may share pins with other mutually exclusive features as follows.

Genlock function	Pin	Pin functionality shared with
Vertical reference (frame sync)	Utility pin 1	MIPI DSI_DE_TE (TE1)
PLL reference clock	Utility pin 2	MIPI DSI_DE_TE2 (TE2)

The genlock reference clock generation for Combo PHY PLLs is shown below. As shown, DPLLs can optionally be filtered by TBT PLL.



Programming

Primary device

- Primary device refers to the display controller that contains the primary transcoder.
- Primary transcoder
 - Primary transcoder outputs the frame sync to be used by local or remote secondaries.
 - TRANS_DDI_FUNC_CTL2 Genlock Enable = 1
 - TRANS_DDI_FUNC_CTL2 Genlock Mode = 10 (Primary)
 - DPLL_CFGCR1[cfselovrd] = 00b (normal XTAL reference)
- Secondary transcoders
 - Secondary transcoders within the primary device are local secondaries.
 - Local secondaries receive frame sync from the primary transcoder in the same device and use the local PLL reference.
 - TRANS_DDI_FUNC_CTL2 Genlock Enable = 1
 - TRANS_DDI_FUNC_CTL2 Genlock Mode = 00 (Local Secondary)
 - DPLL_CFGCR1[cfselovrd] = 00b (normal XTAL reference)
- Pin configuration
 - Frame sync output from primary transcoder to secondary devices
 - UTIL_PIN_CTL Enable = 1
 - UTIL_PIN_CTL Direction = 0 (output)
 - UTIL_PIN_CTL Mode = 0110b (framestart/fsync)
 - UTIL_PIN_CTL Pipe Select = <pipe attached to the primary transcode>
 - Reference clock output to secondary devices
 - UTIL2_PIN_CTL Enable = 1
 - UTIL2_PIN_CTL Direction = 0 (output)

Secondary device

- Secondary device refers to any display controller that contains only secondary transcoders.
- Secondary transcoders
 - Secondary transcoders within the secondary device are remote secondaries.
 - Remote secondaries receive frame sync from the primary transcoder in the primary device and use the primary device PLL reference.
 - Secondary PLL is programmed only after receiving reference clock from the primary device.
 - TRANS_DDI_FUNC_CTL2 Genlock Enable = 1
 - TRANS_DDI_FUNC_CTL2 Genlock Mode = 01 (Remote Secondary)
 - DPLL_CFGCR1[cfselovrd] = 11b (filtered genlock reference)
- Pin configuration
 - Frame sync input from primary device
 - UTIL_PIN_CTL Enable = 1
 - UTIL_PIN_CTL Direction = 1 (input)
 - Reference clock input from primary device

- UTIL2_PIN_CTL Enable = 1
- UTIL2_PIN_CTL Direction = 1 (input)

Sequence to enable a DPLL using filtered genlock reference:

Starting from disabled TBT PLL and DPLL.

1. Configure genlock system to send the genlock reference into this chip
2. Write DPLL CFGCR1 1:0 = 11b (filtered genlock). This automatically sets TBT PLL dividers to the filtered mode and enables TBT PLL.
3. Enable TBTPLL.
4. Poll on TBT PLL lock assertion on the lock signal from CCU to display.
5. Configure DPLL dividers.
6. Enable DPLL.
7. Poll on DPLL lock assertion on the lock signal from CCU to display.

Sequence to disable the DPLL using filtered genlock reference:

1. Disable DPLL.
2. Poll on DPLL lock de-assertion on the lock signal from CCU to display.
3. Write DPLL CFGCR1 1:0 = 00b or 01b. This automatically disables TBT PLL.
4. Disable TBTPLL.
5. Poll on TBT PLL lock de-assertion on the lock signal from CCU to display

Configure genlock system to stop sending the genlock reference into this chip.



Audio

Audio Bios Programming Sequence

Audio Link Settings

When BIOS enables the audio link it must program both the audio controller and audio codec with the same BCLK and T-Mode settings before the link is enabled to ensure enumeration is successful.

In addition the audio codec "Detect Frame Sync Early" value needs to be set per-project in AUD_FREQ_CTL according to the table below.

- The audio codec link setup is performed using the **AUD_FREQ_CTL** register.
 - Program "BCLK" in bits 4:3
 - Program "T-Mode" in bits 15:14
 - Program "Detect Frame sync early" in bits 12:11
 - These settings need to be programmed **before** programming the **AUDIO_PIN_BUF_CTL** register with any analog IO settings.
- The audio controller setup is performed using controller mmio register LCTL (0xc84) and EM1 (0x1000).
 - Please consult with the Audio Controller HW team for official information.
- **For Sx and Connected Standby:**
 - GFX driver programs AUD_FREQ_CTL according to Table 2 below immediately after turning on the power well for audio (PG3).
 - GFX driver disables AUDIO_PIN_BUF_CTL bit 31 after turning off the power well for audio (PG3)

Table 1: Audio Power Wells

This table lists the power well containing the audio codec per-project.

Audio Power Well
PG3

Table 2: Audio Link Settings

The table below provides the project and stepping specific requirements for the "T-Mode" and "Detect Frame sync early" settings which must be used.

A0		B0 +	
T-Mode	Pull-In Clks	T-Mode	Pull-In Clks
4T	1	8T	0

Codec Verb Table

For each codec present on the High Definition Audio codec link, a corresponding pre-defined "Codec Verb Table" must be available to System BIOS. The Codec Verb Tables are based on codec specific information (coded datasheet) and platform design specific information (schematics) and are built by System BIOS writers and platform designers. The table contains a list of 32-bit Verbs (command and data payload) to be sent to the corresponding codec over the High Definition Audio codec link.

Below is a sample High Definition Audio Codec Verb Table for a platform with 1 codec at codec address 01h.

```

;Sample HIGH DEFINITION AUDIO Codec Verb Table
;Codec Address (CAAd) = 02h
;Codec Vendor: XYZ Company
;VenID DevID:
dd 12345678h
;-----
; FrontPanel_Supported? ; 1=Supported ,0=Not supported
db 01h
; # of Rear Panel Pin Complexes
dw 000Ch
; # of Front Panel Pin Complexes
dw 0002h
;-----

```

; Turn on the Audio IO buffer control register to enable the 3 pin link.

Set bit 31 to 1 of AUDIO_PIN_BUF_CTL register.

Note: Set the bit 15 of register offset 0x65F10h of the Display Audio offset. Wait for the Codec to generate the wake event to the controller.

Following verbs should be send to the codec using the PIO method described in the below sections 9.1.3.

VerbTable0:

;The Vendor Node ID and the Pin Node IDs may be updated per project.

;Please refer to the **Node ID descriptions** page to get the node IDs of the vendor node and each supported Pin, and replace the node IDs in the example below.

;All the Pin widgets present in the current project should be programmed with config data as below.

;Enable the third converter and Pin first (NID 02h)

0x20278101h



```
//  
// Audio Verb Table  
  
//  
// Pin Widget 0x4 - PORT DDI-A  
0x20471C10,  
0x20471D00,  
0x20471E56,  
0x20471F18,  
// Pin Widget 0x6 - PORT DDI-B  
0x20671C20,  
0x20671D00,  
0x20671E56,  
0x20671F18,  
// Pin Widget 0x8 - PORT DDI-C  
0x20871C30,  
0x20871D00,  
0x20871E56,  
0x20871F18,  
// Pin Widget 0xA - PORT USBC1  
0x20A71C40,  
0x20A71D00,  
0x20A71E56,  
0x20A71F18,  
// Pin Widget 0xB - PORT USBC2  
0x20B71C50,  
0x20B71D00,  
0x20B71E56,  
0x20B71F18  
// Pin Widget 0xC - PORT USBC3  
0x20C71C60,  
0x20C71D00,
```

```
0x20C71E56,  
0x20C71F18  
// Pin Widget 0xD - PORT USBC4  
0x20D71C70,  
0x20D71D00,  
0x20D71E56,  
0x20D71F18  
// Pin Widget 0xE - PORT USBC5  
0x20E71C80,  
0x20E71D00,  
0x20E71E56,  
0x20E71F18  
// Pin Widget 0xF - PORT USBC6  
0x20F71C90,  
0x20F71D00,  
0x20F71E56,  
0x20F71F18
```

Codec Initialization Programming Sequence

After System BIOS has determined the presence of High Definition Audio codecs, it must follow the programming sequence below to update the codec with the correct jack information specific to the platform for the High Definition Audio driver to retrieve and use later.

There are two ways to send verbs to and receive response data from codecs over the High Definition Audio codec link: using CORB/RIRB (Command Output Ring Buffer / Response Input Ring Buffer) or using the Immediate Command/Immediate Response register pair. The sequence below uses the latter which does not require the availability of a memory buffer.

- System BIOS should ensure that the High Definition Audio HDBAR D27:F0:10-17h contains a valid address value and is enabled by setting D27:F0:04h[1].
- System BIOS must ensure program as mentioned in section 9.6, and then the Controller Reset# bit of Global Control register in memory-mapped space (HDBAR+08h[0]) is set to 1b and read back as 1b.
- When clearing this bit and setting it afterward, System BIOS must ensure that minimum link timing requirements (minimum RESET# assertion time, etc.) are met.
- **Note:** To initialize the codec System BIOS should set the display mmio register 0x65F10 bit 15 to 1b. This bit needs to be set after the controller is brought out of reset.

- The codec requires **50 ms** to come out of reset prior to subsequent operations - System BIOS should wait for the Controller to detect the wake event and recognize the Codec.
- System BIOS can poll on display mmio register 0x65F10 bit 14 "Codec Sleep State" for value 0b (codec awake) with a timeout of **50 ms** before proceeding to additional steps.

For each High Definition Audio codec present as indicated by HDBAR + 0Eh[3:0], System BIOS should perform the codec initialization as described below:

1. Read the VendorID/DeviceID pair from the attached codec.
 - Verify that the ICB bit, HDBAR + 68h[0], is 0.
 - Write verb 200F0000h (dword) to the IC register, HDBAR + 60h, where: '2' (bits 31:28) represents the codec address (CAAd).
 - Program HDBAR + 68h[1:0] to 11b to send the verb to the codec.
 - Poll the ICB bit, HDBAR+68h[0] until it returns 0 indicating the verb has been sent to the codec. System BIOS may write HDBAR + 68h[0] to a 0 if the bit fails to return to 0 after a **50 us** timeout period.
 - If HDBAR + 68h[1] = 1b indicating the response data from the codec is now valid, read HDBAR + 64h; the data is the VID/DID value returned by the codec.
2. Check against internal list to determine if there is a stored verb table which matches the CAAd/VID/DID information.

Steps 1 and 2 are System BIOS implementation-specific steps and can be done in different ways. If a System BIOS has prior knowledge of a fixed platform/codec combination (e.g., for a System BIOS having 3 stored verb tables for 3 known codecs at known codec addresses on a known platform), a simple pre-defined codec-to-table matching can be used and steps 1 and 2 can be eliminated. For a System BIOS to support multiple codec/platform combinations, an internal match-list might be needed to match a platform/codec combination to a codec verb table.

3. If there is a match, send the entire list of verbs in the matching verb table one by one to the codec.
 - Verify the ICB bit, HDBAR + 68h[0] is 0.
 - Write the next verb (dword) in the table to HDBAR + 60h.
 - Program HDBAR + 68h[1:0] to 11b to send the verb to codec.
 - Poll the ICB bit, HDBAR + 68h[0] until it returns 0 indicating the verb has been sent to the codec. System BIOS may write HDBAR + 68h[0] to a 0 if the bit fails to return to 0 after a **50 us** timeout period.
 - Repeat the steps until all the verbs in the table have been sent.

Some verbs in the table may be dependent on certain platform-specific conditions. For example, for the sample table above, the verbs for Pin Complex 7 and 8 (NID=14,16 respectively) should be sent only if the Front Panel Jacks are present and connected on the platform, which may be indicated by a software flag that is controlled by a certain GPIO pin.

Audio Programming Sequence for Link Wakeup

The following audio programming sequences are to be used for preventing the Unsolicited responses when 3 pin link is awake.

Display Audio codec generates a wake event whenever the power well (PGx - power well in which Display Audio codec HW resides) is powered up. If the link is already running, this wake event is considered as unsolicited response by audio controller in PCH. This may sometimes be considered as unnecessary URs. To avoid such URs following programming should be followed by SW. This sequence assumes communication between Audio and GFX drivers without HW to indicate Audio codec power well status.

Power down sequence:

1. Unplug event
2. PD goes low
3. PG2 low
4. Audio link should go low (there should be a communication between GFX driver and Audio Driver to turn off the link)
 1. To turn off iDisp-A link:
 2. Power off iDisp-A codec. Follow the PW2 turn off sequence.
 3. In HD Audio Controller (PCH) Program LCTL1.SPA = 0.
 4. Wait for LCTL1.CPA = 0 to indicate the link has clock stopped.
 5. Clear bit 31 to 0 of AUDIO_PIN_BUF_CTL register.

Power up sequence:

1. Plug event
2. PG2 goes high
3. Audio link to be enabled (there should be a communication between GFX driver and Audio Driver to turn on the link)
 1. To turn on iDisp-A link:
 2. In HD Audio Controller (PCH) Program LCTL1.SPA = 1.
 3. Wait for LCTL1.CPA = 1 to indicate the link has clock running.
 4. Set bit 31 to 1 of AUDIO_PIN_BUF_CTL register.
4. Check WAKESTS[2] = 1 to indicate the codec wake up occurs
5. PD bit set
6. Codec awake. Continue with codec init.

Audio Programming Sequence

The following HDMI and DisplayPort audio programming sequences are to be used when enabling or disabling audio or temporarily disabling audio during a display mode set.

- The audio codec and audio controller disable sequences must be followed prior to disabling the transcoder or port in a display mode set.
- The audio codec and controller enable sequences can be followed after the transcoder is enabled and the port is enabled and completed link training (not sending training or idle patterns if DisplayPort).
- The audio controller and audio codec sequences may be done in parallel or serial. In general, the change in ELDV/PD in the codec sequence will generate an unsolicited response to the audio controller driver to indicate that the controller sequence should start, but other mechanisms may be used. SW should make sure to set the Inactive (IA) bit to 0 before setting PD to 1.

In addition, the Audio Bandwidth Checks page provides algorithms and calculations that must be followed during the enabling sequence for audio, to ensure there is adequate bandwidth available for audio transmission.

Disabling Audio

Audio codec disable sequence:

- Disable sample fabrication
 - Set AUD_MISC_CTRL Sample_Fabrication_EN (bit 2) to "0".
- Disable timestamps
 - Set AUD_CONFIG N_value_index (bit 29) to "0" for HDMI or "1" for DisplayPort.
 - Set N_programming_enable (bit 28) to "1"
 - Set Upper_N_value and Lower_N_value (bits 27:20, 15:4) to all "0"s.
- Disable ELDV and ELD buffer
 - Set AUD_PIN_ELD_CP_VLD ELD_valid (bit 0, 4, or 8 based on which port is used) to "0"
- Wait for 2 vertical blanks
- Optional: Disable audio PD (Presence Detect)
 - Software may choose to skip this in order to keep PD enabled during a resolution switch.
 - Set AUD_PIN_ELD_CP_VLD Audio_Inactive (bit 3, 7, or 11) to "1". SW does not need to set this bit to enable Inactive bit.
 - Set AUD_PIN_ELD_CP_VLD Audio_Output_Enable (bit 2, 6, or 10) to "0".

Audio controller disable sequence:

- Program Stream ID to 0 - Verb ID 706
- Disable audio info frames transmission - Verb ID 732
- Disable Digen - Verb ID 70D
- Program the codec to D3 state if needed.
- Audio driver may stop the audio controller DMA engine at this point if needed, but not required.

Enabling Audio

Pre-Enabling Notes:

The following tables (if present) list any project specific sequence requirements to be handled before initiating the audio codec enable sequence.

Programming Notes
<ul style="list-style-type: none"> No additional requirements specified.

Audio codec enable sequence:

- Enable audio Presence Detect
 - Set AUD_PIN_ELD_CP_VLD Audio_Inactive (bit 3, 7, or 11) to "0".
 - Set AUD_PIN_ELD_CP_VLD Audio_Output_Enable (bit 2, 6, or 10) to "1".
- Wait for 1 vertical blank
- Load ELD buffer and Enable ELDV
 - Set AUD_PIN_ELD_CP_VLD ELD_valid (bit 0, 4, or 8 based on which port is used) to "1".
- Enable timestamps
 - Set AUD_CONFIG N_value_index (bit 29) to "0" for HDMI or "1" for DisplayPort.
 - Set N_programming_enable (bit 28) to "0".
 - Program Upper_N_value and Lower_N_value (bits 27:20, 15:4) if a non-default N value is needed.
- Enable sample fabrication if this feature is needed
 - Set AUD_MISC_CTRL Sample_Fabrication_EN (bit 2) to "1".
- Enable sample fabrication if this feature is needed

Audio controller enable sequence:

- Program the codec to D0 state if in D3 state.
- Program Stream ID to non zero - Verb ID 706
- Enable audio info frames transmission - Verb ID 732
- Enable Digen - Verb ID 70D
- If audio controller DMA engine is stopped, audio driver can start the DMA engine at this point.

Audio Hblank early enable sequence:

- Bits 20:18 of the AUD_CONFIG_BE register have the Hblank early enable for each pipe for DP Audio. Bits 27:24 of the AUD_CONFIG_BE register have the Hblank early enable for each pipe for DP Audio. When DP Audio is enabled on that pipe, driver must set these bits when Audio presence detect is set for each of these pipe regardless of the video resolution. This programming is not needed for HDMI Audio.

- For the following cases of VDSC, 4K, 5K and 8K resolutions the following programming sequence needs to be performed by the driver before setting the Audio Presence detect of pipe for DP audio. This is not required for HDMI Audio.

A. Driver must Program "Hblank Early Enable for Pipe X" = 1b always

B. Determine required pixel clocks between hblank_early rise and hblank rise

```
link_clks_available = {ROUDDNDOWN[((h_total - h_active) * (link_clk /
pixel_clk) - 28)]}
```

```
link_clks_required = {ROUNDUP[(192000 / (refresh_rate * v_total))] *
((48/lanes) + 2)}
```

If `link_clks_available > link_clks_required`

- For Step C, use `hblank_delta = 32`

else

- For Step C, use `hblank_delta = {ROUNDUP[((5 / link_clk) + (5 /
cdclk)) * pixel_clk]}`

C. Determine hblank_early programming

```
tu_data = (pixel_clk * vdsc_bpp * 8) / (link_clk * lanes * fec_coeff)
```

```
tu_line = ((h_active * link_clk * fec_coeff)/(64 * pixel_clk))
```

```
link_clks_active = (tu_line - 1)*64 + tu_data
```

```
hblank_rise = ((link_clks_active + 6*{ROUDDNDOWN[(link_clks_active / 250)]} + 4) *
(pixel_clk/link_clk))
```

```
hblank_early_prog = {ROUNDUP[h_active - hblank_rise + hblank_delta]} (from
Step B)
```

if (`hblank_early_prog < 32`) then

- Program "Hblank_start count for Pipe X" to select value 32

elseif (`hblank_early_prog > 32 < 64`) then

- Program "Hblank_start count for Pipe X" to select value 64

elseif (`hblank_early_prog > 64 < 96`) then

- Program "Hblank_start count for Pipe X" to select value 96

elseif (`hblank_early_prog > 96`) then

- Program "Hblank_start count for Pipe X" to select value 128

D. Calculate samples per line required

```
samples_room = {ROUDDNDOWN[((h_total - h_active) * ( link_clk /
pixel_clk) - 12) / ( (48/lanes) + 2)]}
```

if (`samples_room < 3`) then

- Program "Number of samples per line for Pipe X" to select value == `samples_room`

else

- Program "Number of samples per line for Pipe X" to select value == default value (00b = "All Samples available in buffer")

The following variables are referred to in the equations:

Variable	Units	Note
<code>h_total</code>	pixels	Total horizontal pixels
<code>h_active</code>	pixels	Active horizontal pixels
<code>v_total</code>	pixels	Total vertical pixels
<code>refresh_rate</code>	Hz	Screen refresh rate
<code>input_bpp</code>	bits per pixel	Bits per color "bpc" setting * 8
<code>vdsc_bpp</code>	bits per pixel	If not using compression, <code>vdsc_bpp = input_bpp</code>
<code>fec_coeff</code>	constant	Currently 0.972261. See " Transcoder MN Values ".
<code>lanes</code>	lanes	1,2, or 4
<code>link_clk</code>	MHz	Link Rate / 10 (i.e. 162,270,540,810 MHz). See " Clocks " page per project.
<code>pixel_clk</code>	MHz	$h_total * v_total * refresh_rate$
<code>cdclk</code>	MHz	See " Clocks " page per project.
<code>link_clks_available</code>	link clocks	Link clocks within hblank available for audio use
<code>link_clks_required</code>	link clocks	Link clocks within hblank needed for audio each line

hblank_delta	pixel clocks	Pixel clocks to maintain between hblank_early_rise and hblank_rise
tu_data	link clocks	Link clocks per TU used for pixel data
tu_line	TUs	TUs required per horizontal line
link_clks_active	link clocks	Link clocks required to send active pixels per horizontal line
hblank_rise	pixel clocks	Pixel clock at which hblank will rise, accounting for FEC
hblank_early_prog	pixel clocks	Pixel clocks required to pull hblank_early back far enough from the end of h_active, to ensure there are hblank_delta pixel clocks before hblank rise.
samples_room	audio samples	Maximum number of samples which can fit within hblank, rounded down.

Audio Bandwidth Checks

This page provides the algorithms and calculations needed to ensure there is sufficient bandwidth available with the enabled video configuration for the transmission of audio.

These checks are broken down into DisplayPort and HDMI specific checks, and each section lists the criteria which determine when each check must be performed.

DisplayPort

The following section lists any required checks for audio bandwidth that are applicable in DisplayPort mode.

DP1 MST with DSC Enabled:

The following algorithm and calculation needs to be applied when DP1 MST + DSC + Audio is enabled.

Algorithm

1. Determine max_output_bpp that is supported per the current lanes, link rate, and video mode.
 - Bspec provides this calculation on the Display DSC page under "Programming Considerations and Restrictions"
2. GFX driver prunes all audio modes from the ELD except minimum audio (48KHz 2ch).
3. Check if minimum audio (48KHz 2ch) can work with max_output_bpp from step 1.

- Use Calculation 1 below, with `output_bpp = max_output_bpp`, `aud_freq_khz = 48`, `aud_chan = 2`, and rest of video settings.
 - IF YES -> Proceed to Step 4
 - IF NO -> Disable DSC

4. Determine the `min_output_bpp` required to support minimum audio (48KHz 2ch).

- For each `vdsc_bpp` in [8 to `max_output_bpp`]:
 - Use Calculation 1 below, with `output_bpp = vdsc_bpp[i]`, `aud_freq_khz = 48`, `aud_chan = 2`, and rest of video settings.
 - If `audio_supported = True`, `min_output_bpp = vdsc_bpp[i]`, loop can be stopped.
 - If `audio_supported = False`, continue loop with next `vdsc_bpp[i+1]`.

5. Enable DSC using at least `min_output_bpp` from Step 4.

Calculation 1

Inputs:

`link_rate_ghz`

`lanes`

`pixel_clk_mhz`

`output_bpp`

`hactive_pixels`

`hblank_pixels`

`aud_freq_khz`

`aud_chan`

Constants:

`mtp_size_clks` = 64

`link_overhead` = 0.03

`hblank_bytes_avail_overhead` = 48

`hblank_bytes_req_overhead` = 56

General Calc:

`link_clk_mhz` = (`link_rate_ghz` / 10) * 1000

`line_freq_khz` = (`pixel_clk_mhz` / (`hactive_pixels` + `hblank_pixels`)) * 1000

`pixel_bw_gbps` = (`pixel_clk_mhz` * (`output_bpp` / 8)) / 1000

`link_bw_gbps` = ((`link_clk_mhz` * `lanes`) * (1-`link_overhead`)) / 1000

`mtp_size_ns` = (`mtp_size_clks` / `link_clk_mhz`) * 1000

`hblank_size_ns` = (`hblank_pixels` / `pixel_clk_mhz`) * 1000

`vc_slots` = CEIL[64 * (`pixel_bw_gbps` / `link_bw_gbps`)]

`aud_samples_per_line` = CEIL[`aud_freq_khz` / `line_freq_khz`] + 1



Worst Case Hblank vs Audio BW:

```
hblank_reduced_ns      = hblank_size_ns - (((64 - vc_slots) / link_clk_mhz)*1000)
hblank_slots_full_mtps = FLOOR[ hblank_reduced_ns / mtp_size_ns ] * vc_slots
hblank_slots_partial_mtps = MIN[ CEIL[ ( MOD[ hblank_reduced_ns / mtp_size_ns ] * link_clk_mhz ) / 1000
], vc_slots]
hblank_slots           = hblank_slots_full_mtps + hblank_slots_partial_mtps
hblank_bytes_available = (hblank_slots * lanes) - hblank_bytes_avail_overhead
IF (aud_chan > 2):
    hblank_bytes_required = (aud_samples_per_line*10 + 4)*4
    + hblank_bytes_req_overhead
ELSE:
    hblank_bytes_required = (CEIL[aud_samples_per_line/2]*5 + 4)*4
    + hblank_bytes_req_overhead
```

```
audio_supported = IF hblank_bytes_available > hblank_bytes_required:
YES ELSE: NO
```

HDMI

The following section lists any required checks for audio bandwidth that are applicable in HDMI mode. Currently no audio bandwidth checks are required.

Audio Silent Stream Programming Sequence

The following sequence must be followed by the Display Audio Codec driver to enable/disable the silent stream. This should be part of stream enabling and disabling sequence as described in the Audio programming sequence page here .

Silent Stream disable flow:

1. Program channel index = 0xF, stream id = 0xF through 0x706 verb.
2. Wait for 100us
3. Program channel index = 0xF, stream id = 0 through 0x706 verb

Silent Stream enable flow:

1. Program channel index = 0xF, stream id = DMA stream id through 0x706 verb.
2. Wait for 100us
3. Program channel index = 0x0, stream id = DMA stream id through 0x706 verb

Audio stream playback disable sequence:

1. Program channel index = 0xF through 0x706 verb.
2. Program Stream ID to 0 - Verb ID 706

3. Disable audio info frames transmission - Verb ID 732
4. Disable Digen - Verb ID 70D
5. Program the codec to D3 state if needed.
6. Audio driver may stop the audio controller DMA engine at this point if needed, but not required.

Audio stream playback enable sequence:

1. Program the codec to D0 state if in D3 state.
2. Program Stream ID to non zero - Verb ID 706
3. Program channel index = "actual value" through 0x706 verb.
4. Enable audio info frames transmission - Verb ID 732
5. Enable Digen - Verb ID 70D
6. If audio controller DMA engine is stopped, audio driver can start the DMA engine at this point.

Audio Codec driver must ensure that there are no stream parameters that get modified when silent stream is enabled. Any stream parameters update can be done only after silent stream is disabled as mentioned above.

Audio Configuration

Registers
AUD_CONFIG
AUD_CONFIG_2
AUD_CONFIG_BE
AUD_MISC_CTRL
AUD_VID_DID
AUD_RID
AUD_M_CTS_ENABLE
Audio Power State Format
AUD_PWRST
AUD_EDID_DATA
AUD_FREQ_CNTRL
AUD_INFOFR
AUD_PIN_PIPE_CONN_ENTRY_LNGTH
AUD_PIPE_CONN_SEL_CTRL
AUD_DIP_ELD_CTRL_ST
AUD_PIN_ELD_CP_VLD



DisplayPort Transport

There is one instance of these registers per each DDI.

DP_TP_CTL

DP_TP_STATUS

Digital Display Interface

Combo PHY DDI Buffer

Overview

This section applies to the DDIs going to the combo PHY. DDIs going to FIA thunderbolt/typeC have separate programming.

Registers

DDI_BUF_CTL

PHY_MISC

PORT_COMP_DW0

PORT_COMP_DW1

PORT_COMP_DW3

PORT_COMP_DW8

PORT_COMP_DW9

PORT_COMP_DW10

PORT_CL_DW5

PORT_CL_DW10

PORT_CL_DW12

PORT_CL_DW15

PORT_CL_DW16

PORT_TX_DW1

PORT_TX_DW2

PORT_TX_DW4

PORT_TX_DW5

PORT_TX_DW6

PORT_TX_DW7

PORT_TX_DW8

PORT_PCS_DW1

PORT_PCS_DW9

The group access address can be used to simultaneously write the same value to a register that has instances in all 4 lanes. This is only for use when the same exact register value is applied to all 4 lanes. Reads using a port group address usually cannot return correct data. For read/modify/write to a group, the read should be to one of the lane addresses, then the write to the group address.

Combo PHY Comp Sources

Certain PHYs are connected to compensation resistors and act as compensation sources. Other PHYs and DPLLs sync to their sources to get compensation values.

A comp source PHY must be initialized before its comp sink PHYs are initialized or DPLLs are enabled, and source must be kept initialized while source or sinks are in use.

Source	Sink(s) to this Source
PHY A	DPLLs*, PHY B, PHY C

*DPLLs now have their own comp, but sink programming should be maintained for commonality with past projects.

Combo PHY Initialization Sequence

The PHY must be initialized before enabling combo PHY DDI IO power or Aux IO power. It can be initialized as early as the full display initialization sequence.

DDIA PHY initialization state will be preserved across DC6. Depending on project and SKU, other DDIs may not be preserved and then must be re-initialized for use after DC6 is disabled.

1. If **PORT_COMP_DW0** Comp Init == 1b, skip the rest of this sequence since it is already initialized
2. **DCC susclk divider programming**
PORT_TX_DW8
 Program PORT_TX_DW8[odcc_clk_sel] to "1" and PORT_TX_DW8[odcc_clk_div_sel[1:0]] to "00".
DCC continuous mode
PORT_PCS_DW1
 Program PORT_PCS_DW1[DCC Mode Select] to DCC continuous mode.
3. Clear **PHY_MISC_<DDI that maps to this PHY>** DE to IO Comp Pwr Down to 0b. Skip this step if this DDI does not have a PHY_MISC register instance.
4. Program procmon reference values in **PORT_COMP_DW1**, **PORT_COMP_DW9**, and **PORT_COMP_DW10**.
 - Procmon Reference Values



Voltage	0.85V	0.85V	0.95V	0.95V	1.05V	1.05V
Register/Process	dot-0	dot-1	dot-0	dot-1	dot-0	dot-1
PORT_COMP_DW1 [7:0]	0x00	N/A	0x00	0x00	0x00	0x00
PORT_COMP_DW1 [23:16]	0x00	N/A	0x00	0x00	0x00	0x44
PORT_COMP_DW9 [31:0]	0x62AB67BB	N/A	0x86E172C7	0x93F87FE1	0x98FA82DD	0x9A00AB25
PORT_COMP_DW10 [31:0]	0x51914F96	N/A	0x77CA5EAB	0x8AE871C5	0x89E46DC1	0x8AE38FF1

Voltage and process are found in **PORT_COMP_DW3**

5. If this PHY is a comp source, set **PORT_COMP_DW8** irefgen to 1b.
6. Set **PORT_COMP_DW0** Comp Init to 1b.
7. Set **PORT_CL_DW5** CL Power Down Enable to 1b.

Combo PHY Un-Initialization Sequence

The combo PHY only needs to be un-initialized for DC9.

DDIA PHY is the comp source, so it must not be un-initialized if other combo PHYs are in use.

1. Set **PHY_MISC** DE to IO Comp Pwr Down to 1b.
2. Clear **PORT_COMP_DW0** Comp Init to 0b.

If the PHY is needed again, follow the initialization sequence.

Voltage Swing Programming Sequence

This sequence is used to setup the voltage swing before enabling the DDI, as well as for changing the voltage during DisplayPort link training. The voltage swing values are listed in the tables below. Note that the MIPI DSI voltage swing programming is for the high speed data buffers and hardware automatically handles the voltage swing for the low power data buffers.

1. If port type is eDP or DP, set **PORT_PCS_DW1** cmnkeeper_enable to 1b, else clear to 0b.
2. Program loadgen select

Loadgen Values	Conditions		
Register	Bit rate <= 6 GHz and 4 lanes enabled	Bit rate <= 6 GHz and 2 or 1 lanes enabled	Bit rate > 6 GHz
PORT_TX_DW4_LN0 Loadgen Select	0	0	0
PORT_TX_DW4_LN1 Loadgen Select	1	1	0
PORT_TX_DW4_LN2 Loadgen Select	1	1	0
PORT_TX_DW4_LN3 Loadgen Select	1	0	0

- Group access cannot be used here since each lane can have a unique value, and any later writes to PORT_TX_DW4 must not use group access so that they don't overwrite the individual lane values.
3. Set PORT_CL_DW5 SUS Clock Config to 11b.
 4. Clear training enable to change swing values
 - Clear PORT_TX_DW5 TX Training Enable to 0b
 5. Program swing and de-emphasis
 - Values for each port type are listed in voltage swing programming tables below
 - Set PORT_TX_DW5 Scaling Mode Sel to 010b.
 - Set PORT_TX_DW2 Rcomp Scalar to 0x98 hex
 - Set PORT_TX_DW4 Post Cursor 2 to 0x00 hex
 - Set PORT_TX_DW5
 - Rterm Select to 110b
 - 3 Tap Disable to 1
 - Cursor program to 0
 - Coeff Polarity to 0
 - Program the other fields in PORT_TX_DW2, PORT_TX_DW4, PORT_TX_DW5, and PORT_TX_DW7 using the values specified in the table below.
 6. Set training enable to trigger update
 - Set PORT_TX_DW5 TX Training Enable to 1b

Voltage Swing Programming

Protocol	Voltage Swing Level ¹	Pre-emphasis Level ¹	Non-transition mV diff p-p	Transition mV diff p-p	Pre-emphasis dB	Swing Sel DW2 binary	N Scalar DW7 hex	Cursor Coeff DW4 hex	Post Cursor 1 DW4 hex	2 Tap Disable DW5
DP up to HBR	0	0	350	350	0.0	4'b1010	0x32	0x3F	0x00	0'b0
DP up to HBR	0	1	350	500	3.1	4'b1010	0x4F	0x37	0x08	0'b0
DP up to HBR	0	2	350	700	6.0	4'b1100	0x71	0x2F	0x10	0'b0
DP up to HBR	0	3	350	900	8.2	4'b0110	0x7D	0x2b	0x14	0'b0
DP up to HBR	1	0	500	500	0.0	4'b1010	0x4C	0x3F	0x00	0'b0
DP up to HBR	1	1	500	700	2.9	4'b1100	0x73	0x34	0x0b	0'b0
DP up to HBR	1	2	500	900	5.1	4'b0110	0x7F	0x2F	0x10	0'b0
DP up to HBR	2	0	650	700	0.6	4'b1100	0x6C	0x3C	0x03	0'b0
DP up to HBR	2	1	600	900	3.5	4'b0110	0x7F	0x35	0x0a	0'b0
DP up to HBR	3	0	900	900	0.0	4'b0110	0x7F	0x3F	0x00	0'b0
DP HBR2	0	0	350	350	0.0	4'b1010	0x35	0x3F	0x00	0'b0
DP HBR2	0	1	350	500	3.1	4'b1010	0x4F	0x37	0x08	0'b0
DP HBR2	0	2	350	700	6.0	4'b1100	0x63	0x2F	0x10	0'b0
DP HBR2	0	3	350	900	8.2	4'b0110	0x7F	0x2b	0x14	0'b0

Protocol	Voltage Swing Level ¹	Pre-emphasis Level ¹	Non-transition mV diff p-p	Transition mV diff p-p	Pre-emphasis dB	Swing Sel DW2 binary	N Scalar DW7 hex	Cursor Coeff DW4 hex	Post Cursor 1 DW4 hex	2 Tap Disable DW5
DP HBR2	1	0	500	500	0.0	4'b1010	0x47	0x3F	0x00	0'b0
DP HBR2	1	1	500	700	2.9	4'b1100	0x63	0x34	0x0b	0'b0
DP HBR2	1	2	500	900	5.1	4'b0110	0x7F	0x2F	0x10	0'b0
DP HBR2	2	0	650	700	0.6	4'b1100	0x61	0x3C	0x03	0'b0
DP HBR2	2	1	600	900	3.5	4'b0110	0x7B	0x35	0x0a	0'b0
DP HBR2	3	0	900	900	0.0	4'b0110	0x7F	0x3F	0x00	0'b0
DP HBR2	0	0	350	350	0.0	4'b1010	0x35	0x3F	0x00	0'b0
DP HBR2	0	1	350	500	3.1	4'b1010	0x4F	0x36	0x09	0'b0
DP HBR2	0	2	350	700	6.0	4'b1100	0x60	0x32	0x0D	0'b0
DP HBR2	0	3	350	900	8.2	4'b1100	0x7F	0x2D	0x12	0'b0
DP HBR2	1	0	500	500	0.0	4'b1100	0x47	0x3F	0x00	0'b0
DP HBR2	1	1	500	700	2.9	4'b1100	0x6F	0x36	0x09	0'b0
DP HBR2	1	2	500	900	5.1	4'b0110	0x7D	0x32	0x0D	0'b0
DP HBR2	2	0	650	700	0.6	4'b0110	0x60	0x3C	0x03	0'b0
DP HBR2	2	1	600	900	3.5	4'b0110	0x7F	0x34	0x0B	0'b0
DP HBR2	3	0	900	900	0.0	4'b0110	0x7F	0x3F	0x00	0'b0
eDP up to HBR2	0	0	200	200	0.0	4'b0000	0x7F	0x3F	0x00	0'b0
eDP up to HBR2	0	1	200	250	1.9	4'b1000	0x7F	0x38	0x07	0'b0
eDP up to HBR2	0	2	200	300	3.5	4'b0001	0x7F	0x33	0x0C	0'b0
eDP up to HBR2	0	3	200	350	4.9	4'b1001	0x7F	0x31	0x0e	0'b0
eDP up to HBR2	1	0	250	250	0.0	4'b1000	0x7F	0x3F	0x00	0'b0
eDP up to HBR2	1	1	250	300	1.6	4'b0001	0x7F	0x38	0x07	0'b0
eDP up to HBR2	1	2	250	350	2.9	4'b1001	0x7F	0x35	0x0a	0'b0
eDP up to HBR2	2	0	300	300	0.0	4'b0001	0x7F	0x3F	0x00	0'b0
eDP up to HBR2	2	1	300	350	1.3	4'b1001	0x7F	0x38	0x07	0'b0
eDP up to HBR2	3	0	350	350	0.0	4'b1001	0x7F	0x3F	0x00	0'b0
eDP up to HBR2 - optimized ²					0.0	4'b0110	0x7F	0x3F	0x00	0'b0
eDP HBR3	0	0	350	350	0.0	4'b1010	0x35	0x3F	0x00	0'b0
eDP HBR3	0	1	350	500	3.1	4'b1010	0x4F	0x37	0x08	0'b0

Protocol	Voltage Swing Level ¹	Pre-emphasis Level ¹	Non-transition mV diff p-p	Transition mV diff p-p	Pre-emphasis dB	Swing Sel DW2 binary	N Scalar DW7 hex	Cursor Coeff DW4 hex	Post Cursor 1 DW4 hex	2 Tap Disable DW5
eDP HBR3	0	2	350	700	6.0	4'b1100	0x71	0x2F	0x10	0'b0
eDP HBR3	0	3	350	900	8.2	4'b0110	0x7F	0x2b	0x14	0'b0
eDP HBR3	1	0	500	500	0.0	4'b1010	0x4C	0x3F	0x00	0'b0
eDP HBR3	1	1	500	700	2.9	4'b1100	0x73	0x34	0x0b	0'b0
eDP HBR3	1	2	500	900	5.1	4'b0110	0x7F	0x2F	0x10	0'b0
eDP HBR3	2	0	650	700	0.6	4'b1100	0x6C	0x3C	0x03	0'b0
eDP HBR3	2	1	600	900	3.5	4'b0110	0x7F	0x35	0x0a	0'b0
eDP HBR3	3	0	900	900	0.0	4'b0110	0x7F	0x3F	0x00	0'b0
HDMI			450	450	0.0	4'b1010	0x60	0x3F	0x00	0'b0
HDMI			450	650	3.2	4'b1011	0x73	0x36	0x09	0'b0
HDMI			450	850	5.5	4'b0110	0x7F	0x31	0x0E	0'b0
HDMI: ALS and Retimer option			650	650	0.0	4'b1011	0x73	0x3F	0x00	0'b0
HDMI			650	850	2.3	4'b0110	0x7F	0x37	0x08	0'b0
HDMI			850	850	0.0	4'b0110	0x7F	0x3F	0x00	0'b0
HDMI			600	850	3.0	4'b0110	0x7F	0x35	0x0A	0'b0
MIPI DSI			400	400	0.0	4'b0010	0x7F	0x3F	0x00	0'b1

¹The voltage swing level and pre-emphasis level values follow the naming used in the DisplayPort standard.

²eDP extra power optimized setting. This swing setting can be used for eDP HBR2 and lower frequencies if the motherboard has been designed to support it and it is specified through VBT or other OEM selection.

Program the voltage swing as listed in the table and additionally program PORT_CL_DW10 o_edp4k2k_mode_ovrd_en=1 and o_edp4k2k_mode_ovrd_val=1. Program o_edp4k2k_mode_ovrd_val=0 when using any other swing values. If link training fails with this swing, fall back to the regular eDP swing settings.

eDP panels may support lower power, low voltage, swing values using the "eDP" protocol values from the table or higher power, high voltage, swing values using the "DP" protocol values. The selection is generally an OEM decision configured in BIOS VBT.

TypeC PHY DDI Buffer

This programming is used for the typeC PHY when operating in DP alternate mode or the legacy HDMI or DP modes. This programming is not used for thunderbolt as the thunderbolt controller takes care of the PHY programming.



PHY Lane Configuration

The PHY contains 2 lanes, and each lane contains a primary transmitter (TX1) and a secondary transmitter/receiver (TX2 or TX/RX). A single PHY lane can support up to 2 DP or HDMI lanes. Two PHY lanes can be used to support 4 DP or HDMI lanes. Note that the RX function in TX2 is always disabled for DP use cases.

Driver must configure the DP mode to indicate which transmitter is used and voltage swing for each transmitter in use, within each lane in use.

Voltage Swing Programming Sequence

This sequence is used to setup the voltage swing before enabling the DDI, as well as for changing the voltage during DisplayPort link training.

1. Clear DKL_TX_PMD_LANE_SUS_LN0 register if display owns this PHY lane.
2. Clear DKL_TX_PMD_LANE_SUS_LN1 register if display owns this PHY lane.
3. Update registers DKL_TX_DPCNTL0, DKL_TX_DPCNTL1 and DKL_TX_DPCNTL2 registers with appropriate Vswing and de-emphasis levels from table below.
4. Set DKL_TX_DPCNTL2[dp20bitmode] to "0".

DP Voltage Swing Table

Voltage Swing	Pre-emphasis	Non-transition mV diff p-p	Transition mV diff p-p	Pre-emphasis dB	Voltage Level Swing Select	Pre Cursor	Post Cursor	
					DKL_TX_DPCNTL0[2:0] DKL_TX_DPCNTL1[2:0]	DKL_TX_DPCNTL0[17:13] DKL_TX_DPCNTL1[17:13]	DKL_TX_DPCNTL0[12:8] DKL_TX_DPCNTL1[12:8]	
					Vswing_Control_tx1 Vswing_Control_tx2	preshoot_control_i0_tx1 preshoot_control_i0_tx2	de_emphasis_control_i0_tx1 de_emphasis_control_i0_tx2	
							HBR/RBR	HBR3/HBR2
Level 0	0	400	400	0	7	0	0	0
Level 0	1	400	600	3.5	5	0	5	5
Level 0	2	400	800	6	2	0	B	B
Level 0	3	400	1200	9.5	0	0	18	19
Level 1	0	600	600	0	5	0	0	0
Level 1	1	600	800	3.5	2	0	8	8
Level 1	2	600	1200	6	0	0	14	14
Level 2	0	800	800	0	2	0	0	0
Level 2	1	800	1200	3.5	0	0	B	B
Level 3	0	1200	1200	0	0	0	0	0

HDMI Voltage Swing Table

Protocol	Voltage Swing and Pre-emphasis	HDMI Preset Level	Voltage Level Swing Select	Pre Cursor	Post Cursor
			DKL_TX_DPCNTL0[2:0] DKL_TX_DPCNTL1[2:0]	DKL_TX_DPCNTL0[17:13] DKL_TX_DPCNTL1[17:13]	DKL_TX_DPCNTL0[12:8] DKL_TX_DPCNTL1[12:8]
			Vswing_Control_tx1 Vswing_Control_tx2	preshoot_control_i0_tx1 preshoot_control_i0_tx2	de_emphasis_control_i0_tx1 de_emphasis_control_i0_tx2
HDMI	400mV / 0dB	1	7	0	0
HDMI	500mV / 0dB	2	6	0	0
HDMI : ALS	650mV / 0dB	3	4	0	0
HDMI	800mV / 0dB	4	2	0	0
HDMI : Re-timer	1000mV / 0dB	5	0	0	0
HDMI	Full / -1.5dB	6	0	0	5
HDMI	Full / -1.8dB	7	0	0	6
HDMI : CRLS	Full / -2dB	8	0	0	7
HDMI	Full / -2.5dB	9	0	0	8
HDMI	Full / -3dB	10	0	0	A

DKL_DP_MODE Programming

There are two DP_MODE registers per PHY, one for each PHY lane, containing x1 and x2 fields that are programmed to indicate which transmitter subsystem to enable. The values to program are derived from the pin and lane assignments and port width selection found in the following registers.

- PORT_TX_DFLEXPA1 DPPATC<this port> Display Port Pin Assignment for Type-C Connector
- PORT_TX_DFLEXDPSP DPX4TXLATC<this port> Display Port x4 TX Lane Assignment for Type-C Connector
- DDI_BUF_CTL_<this port> DP Port Width Selection

Orientation flip and lane reversal differences are cancelled out by hardware lane enable signaling shutdown of the unused LN0 or LN1.

DPPATC	DPX4TXLATC	Description	DP Port Width	DP MODE LN1		DP MODE LN0	
				x2_mode	x1_mode	x2_mode	x1_mode
0x1	0xf	A	x4	1	0	1	0
0x1	0xf	A flip	x4	1	0	1	0
0x1	0xf	A	x2	0	0	0	0
0x1	0xf	A flip	x2	0	0	0	0
0x1	0xf	A	x1	0	0	0	0
0x1	0xf	A flip	x1	0	0	0	0
0x1	0x5	A active	x2	0	0	0	0
0x1	0x5	A active flip	x2	0	0	0	0

DPPATC	DPX4TXLATC	Description	DP Port Width	DP MODE LN1		DP MODE LN0	
				x2_mode	x1_mode	x2_mode	x1_mode
0x1	0x5	A active	x1	0	0	0	0
0x1	0x5	A active flip	x1	0	0	0	0
0x2	0xc	B	x2	1	0	1	0
0x2	0x3	B flip	x2	1	0	1	0
0x2	0xc	B	x1	0	0	0	0
0x2	0x3	B flip	x1	0	0	0	0
0x2	0x4	B active	x1	0	0	0	0
0x2	0x1	B active flip	x1	0	0	0	0
0x3, 0x5	0xf	C/E	x4	1	0	1	0
0x3, 0x5	0xf	C/E flip	x4	1	0	1	0
0x3, 0x5	0xf	C/E	x2	1	0	1	0
0x3, 0x5	0xf	C/E flip	x2	1	0	1	0
0x3, 0x5	0xf	C/E	x1	0	1	0	1
0x3, 0x5	0xf	C/E flip	x1	0	1	0	1
0x4, 0x6	0xc	D/F	x2	1	0	1	0
0x4, 0x6	0x3	D/F flip	x2	1	0	1	0
0x4, 0x6	0xc	D/F	x1	0	1	0	1
0x4, 0x6	0x3	D/F flip	x1	0	1	0	1
0x0	n/a	Fixed/static	x4 or HDMI	1	0	1	0
0x0	n/a	Fixed/static reverse	x4 or HDMI	1	0	1	0
0x0	n/a	Fixed/static	x2	1	0	1	0
0x0	n/a	Fixed/static reverse	x2	1	0	1	0
0x0	n/a	Fixed/static	x1	0	1	0	0
0x0	n/a	Fixed/static reverse	x1	0	1	0	0

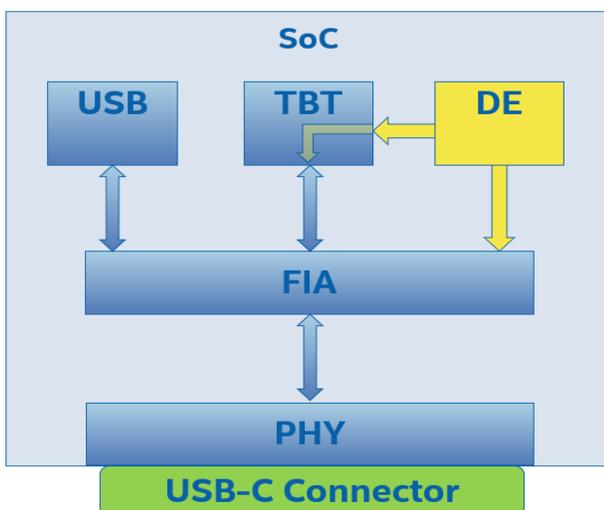
Decode of x1 and x2

DP MODE		Active TX
cfg_dp_x2_mode	cfg_dp_x1_mode	
0	0	TX1 (TX)
0	1	TX2 (TX/RX)
1	0	TX1 + TX2
1	1	TX1 + TX2

TypeC Programming

The USB 3.1, thunderbolt 3, and display (DE) controllers are integrated into the CPU. All these controllers connect to Flexi I/O Adapter (FIA) that muxes data and clocks between controllers and the PHY. The PHY contains the IOs and PLLs for the USB-C connector. Display Engine can also output to thunderbolt (TBT) which then goes to FIA and to PHY. During mode enumeration (DE to FIA path for non-TBT), display software takes into account not only the port bandwidth of the DP receiver (monitor) and any intervening DP re-timers, but also the number of DP lanes being driven out of a given USB Type-C port.

TBT does not impose restrictions on number of DP lanes that can be sourced. Display software reads the monitor capabilities, trains the link, and chooses number of DP lanes to be used per current software policies. However, PHY will not need to be programmed or trained. TBT will use a different PLL than the usual combo PHY DDI PLL or PLL, so software will need to be aware of the TBT connector.



Static/Fixed/Legacy/Native DP or HDMI Configuration

A native DP or HDMI connector can be attached to the TypeC PHY to support a static or fixed configuration instead of dynamic typeC. This configuration must be specified in IFWI strap bits which the SoC micro-controller uses to configure each port, and BIOS VBT which display software uses.

Type C DDI Sequence

The main Type-C flows of interest to display controller are

- PD connect flow
- PD disconnect flow
- PD Re-configuration flow

As part of PD connect flow, whenever a new device is connected over the Type-C connector, PD engine detects the presence of the device over the CC channel and identifies the connector orientation. It also configures PHY common lane and data lane to operate in DP mode. At the end of the PD connect flow,



SOC uC (typically called IOM) also programs FIA Pin Assignment registers, and programs FIA registers with pin assignment (PA), link width, live state, etc. Finally, PD connect flow sends HPD to DE.

The type-C PHYs can be shared between multiple controllers; display, USB, etc. As a result, handshaking through FIA registers is required around connect and disconnect to cleanly transfer ownership with the controller and set the type-C power state.

Display software most not use a disconnected port.

FIA registers updated during PD flows

Register	Function
PORT_TX_DFLEXP1	FIA arranges the 4 DP lanes in Type-C connector based on 6 possible arrangements called pin assignments A-F in VESA DP Type-C spec.
PORT_TX_DFLEXDPPMS PORT_TX_DFLEXDPCSSS PORT_TX_DFLEXDPS1	Communication between SoC uC and display software.
PORT_TX_DFLEXDPMLE1	Display software configures the number of lanes it is using

Port Mapping	
The following table highlights mapping from port to FIA instance.	
TypeC Port	2 Connector Modular FIA family
Port1	FIA1, connector 0
Port2	FIA1, connector 1
Port3	FIA2, connector 0
Port4	FIA2, connector 1
Port5	FIA3, connector 0
Port6	FIA3, connector 1
The mapping from connector to port is split into groups of 2. For example, the FIA1 DFLEXDPS1 register fields referring to connectors 0 through 1 map to ports 1 through 2 and the FIA2 DFLEXDPS1 register fields referring to connectors 0 through 1 maps to ports 3 through 4. The FIA register fields for connectors >1 are reserved.	

Type-C Aux Power Requirements

The aux power function is overloaded to control power states in the type-C subsystem.

Aux power must only be enabled while in the connect state. Do not enable Aux power before connect flow or after disconnect flow.

For DP-alternate and thunderbolt, Aux power must be enabled while using Aux channel or main link (enable mode set) and it should be disabled at other times to save power.

For static/fixed/legacy/native, for DP and for HDMI, it must be enabled during the connect flow and not disabled until disconnect flow.

For every typeC port (static/fixed/legacy/native and DP-alternate, but not thunderbolt), after enabling aux power request and receiving aux power ack, wait for uc_health bit (Devel PHY register DKL_CMN_UC_DW27 @BASE + 0x236c bit [15]) to assert. Timeout and fail after 10us.

Type-C Hot Plug Detection (HPD)

Type-C ports have separate hot plug detection paths for thunderbolt, DP alternate, and legacy HDMI or DP connectors (static/fixed/legacy/native connection).

The HPD control and status reside in 3 separate sets of display registers

1. Thunderbolt HPD is in the north display hotplug control and interrupt registers, TBT instances.
2. DP alternate HPD is in the north display hotplug control and interrupt registers, TC instances.
3. Legacy HDMI or DP hotplug is detected through the south display hotplug control and interrupt registers.

Display software enables HPD detection in all 3 sets, then responds to whichever interrupt is received. Based on the hotplug source, display software invokes one of the connect/disconnect flows below.

The DFLEXDPSP Live State fields provide the current connect or disconnect status for thunderbolt and DP alternate.

See the Hot Plug Detection page for details.

DP Tunneling - Integrated TBT behavior

On TBT integrated products, DP resource allocation is done by IOM. A USB-C port can connect to TBT discrete device (via a cable) and then a DP source will be allocated to a TBT host to be driven according to TBT protocol. DP resources are allocated according to the IOM "DP_RESOURCE_MNG" register. TBT CM (connection manager) will claim a DP resource via this register.

When IOM claim a resource to itself, TBT connection manager interprets that that corresponding resource is allocated and TBT cannot use it. IOM can also claim a resource even when TBT already claimed the same DP resource, which will resolve in TBT releasing it in favor of IOM.

HPD Processing

HPD messages from TBT to IOM is sent via a HW mechanism. IOM requires TBT to send HPD messages only when a DP resource is allocated to it. TBT CM will build a DP tunneling path only after it gets a resource. At HPD disconnect, TBT CM will tear down the DP connection path (no tunnel) and DP AUX should also be considered as disconnected. TBT will ignore all interface signals to display engine when there is no tunnel. IOM sends acknowledgement to TBT upon receiving HPD disconnect message.

If HPD connect event is detected while TBT port is enabled for DP tunneling a mode set disable is recommended, followed by re-evaluation and re-enabling of the TBT DP tunnel.



DP Connected Standby

DP connected standby is a low power mode in which monitors will be turned OFF actively by IOM (monitors are still physically connected).

NOTE: Registers mentioned below are within Type-C subsystem and not accessible to display software.

Entering connected standby mode

IOM will assert "dp_disconnect_req" on the DP_CONFIG register.

TBT LC (link controller) will detect the above bit assertion and will configure unplug to DP adaptor.

TBT CM will receive the un-plug packet and trigger the basic disconnect flow.

Exiting connected standby mode

IOM will de-assert "dp_disconnect_req" on the DP_CONFIG register.

LC will detect the above bit de-assertion and configure the plug of the DP adapter.

CM triggers the basic connect flow.

TBT DP Path Topology Retention

TBT can route any of its DP inputs to any of its endpoint monitors. In order to keep consistency of resources routing when entering/exiting standby modes, CM will register the routing configuration on its always ON memory. Registering the DP routing configuration should be done whenever a new path is built and CM will always check last state configuration prior to claiming a DP resource.

Whenever the TBT CM detects new plug-in monitor on one of its TBT ports it follows below flow.

- TBT CM will check its always ON memory for the last DP allocation of the new plug-in TBT port.
- TBT CM will check if DP resource is available and claim it, Otherwise claim for other DP resource. It is intended to restore to the state before entering a standby mode. IOM on its side should prevent DP ALT mode allocation to a source that was previously allocated to TBT.
- If monitors connectivity was physically changed between entering and exiting standby mode, TBT resource routing will not guarantee any sort of compatibility to pre-standby mode.
- If a new DP resource is allocated to TBT port, TBT CM should update the configuration on its always ON memory.

Connect Flows

These flows are used for completing a connection. They can be initiated upon receiving HPD connect interrupt or finding that a previous HPD connect was missed because of reset or power state. IOM FW will configure FIA and send the HPD assert message to display engine to trigger an interrupt. Display software can then complete the handshake and enable the port.

Note that display software can abort the connect flow if it does not intend to use the port main link or Aux channel. By aborting the connect, display software allows IOM to immediately change ownership of the port on a disconnect, without waiting for any display software response.

Display software is required to abort the connect if it knows it will be entering states where hotplug cannot be responded to for a long period, such as when GOP/VBIOS chooses not to enable a display on typeC, entering DC9, D3, S3, S4, and S5, and un-installing driver. Software can start the connect flow after exiting these states.

TCCOLD

The typeC subsystem can enter the TCCOLD power saving state when the state has been enabled by BIOS, there is no DP alternate or thunderbolt connection, and display software has not blocked it with a GT Driver Mailbox command. TCCOLD must be blocked by IOM or display software when display software accesses FIA or typeC PHY registers or uses Aux channel or the main link.

The general usage is for display software to use GT Driver mailbox to block TCCOLD before first accessing FIA registers during the hotplug and connect flows, then use GT Driver mailbox to unblock TCCOLD only after it is done accessing FIA registers and there is no longer a connection (connect abort complete or disconnect complete).

DP-alternate specific usage

Display software uses GT Driver mailbox to block TCCOLD before accessing FIA registers to during type-C hotplug detection.

If there is no longer a connection, display software uses GT Driver mailbox to unblock TCCOLD after finishing accessing FIA registers, and does not follow any connect flow.

If there is a DP-alternate connection, display software proceeds to the connect flow for DP alternate and leaves the mailbox in the TCCOLD blocking state.

Display software uses GT Driver mailbox to unblock TCCOLD after last accessing FIA registers when aborting the connect flow or completing the disconnect flow.

IOM will be doubly blocking TCCOLD while there is a receiver connected or once DFLEXDPPMS.DPPMSTC and DFLEXDPCSSS.DPPMSTC are both 1 in the connect flow until DFLEXDPCSSS.DPPMSTC is cleared in the disconnect flow, but display software will not rely on that.

Thunderbolt specific usage

Display software uses GT Driver mailbox to block TCCOLD before accessing FIA registers to find if a thunderbolt receiver is connected.

If there is no longer a connection, display software uses GT Driver mailbox to unblock TCCOLD after finishing accessing FIA registers, and does not follow any connect flow.

If there is a thunderbolt connection, display software proceeds to the connect flow for thunderbolt and leaves the mailbox in the TCCOLD blocking state.

Display software uses GT Driver mailbox to unblock TCCOLD when aborting the connect flow or completing the disconnect flow.

IOM and the thunderbolt controller will be doubly blocking TCCOLD while there is a receiver connected, but display software will not rely on that.

Static/fixed/legacy/native specific usage



Display software uses GT Driver mailbox to block TCCOLD before accessing FIA registers during the connect flow and leaves the mailbox in the TCCOLD blocking state.

Display software uses GT Driver mailbox to unblock TCCOLD after last accessing FIA registers when aborting the connect flow or completing the disconnect flow.

IOM is not aware of connection and will not be doing any TCCOLD blocking of its own.

GT Driver Mailbox to Block TCCOLD

This mailbox command triggers TCCOLD to exit and not re-enter.

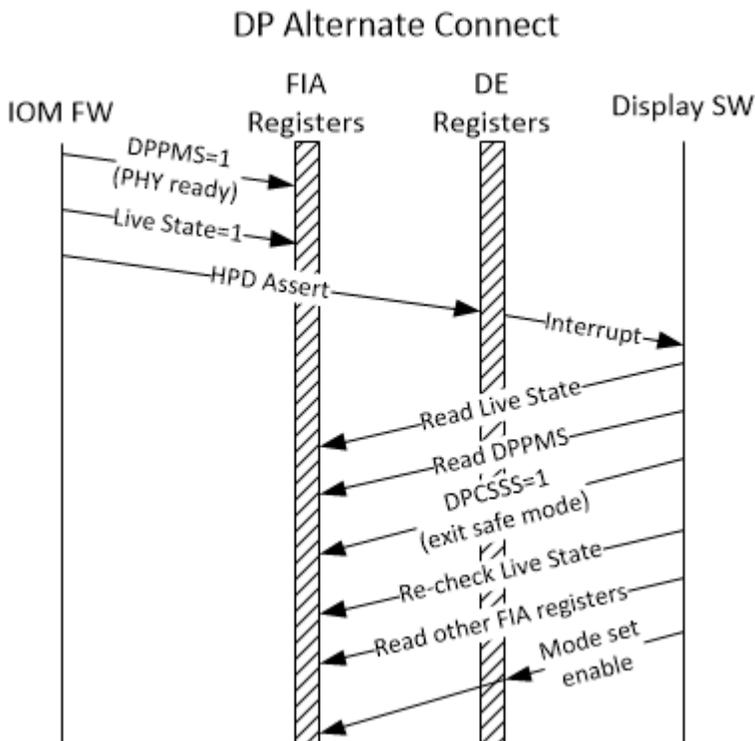
1. Ensure any previous GT Driver Mailbox transaction is complete.
2. Write GT Driver Mailbox Data Low = 0x00000000.
3. Write GT Driver Mailbox Data High = 0x00000000.
4. Write GT Driver Mailbox Interface = 0x80000026.
5. Poll GT Driver Mailbox Interface for Run/Busy indication cleared (bit 31 == 0).
 - Timeout and fail after 200 us. Do not use typeC ports if there is a timeout.
6. Read GT Driver Mailbox Data Low, if bit 0 is 0x0, continue, else go to step 2.
 - If the condition in step 6 is not satisfied after cycling through steps 2-6 for 600us total (typically <150 us), timeout and fail and do not use typeC ports.
 - Data 0 = not in TCCOLD. Data 1 = in TCCOLD.

GT Driver Mailbox to Unblock TCCOLD

This mailbox command allows TCCOLD to re-enter.

1. Ensure any previous GT Driver Mailbox transaction is complete.
2. Write GT Driver Mailbox Data Low = 0x00000001.
3. Write GT Driver Mailbox Data High = 0x00000000.
4. Write GT Driver Mailbox Interface = 0x80000026.
5. Poll GT Driver Mailbox Interface for Run/Busy indication cleared (bit 31 == 0).
 - Timeout and fail after 200 us. Do not use typeC ports if there is a timeout.

Connect Flow for DP Alternate

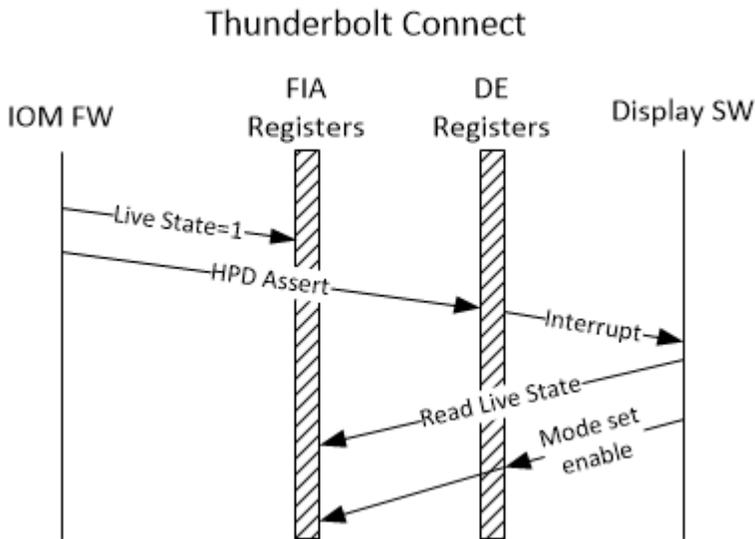


This flow is for DP alternate mode. PHY lane 0, lane 1, or both, will be used, depending on pin assignment.

1. If display software will not use the port main link or Aux channel
 1. Abort remainder of the flow
2. Else
 1. Display software uses GT driver mailbox to block TCCOLD
 2. Display software reads DFLEXDPPMS.DPPMSTC which should be '1' to indicate the SOC uC has switched the Lane into DP Mode, else abort connect flow.
 3. Display software sets DFLEXDPCSSS.DPPMSTC to '1' indicating that display controller is not in safe mode anymore.
 4. Display software reads DFLEXDPSP to verify port has not become disconnected
 - If Live State is "No HPD Connect for TypeC or TBT", nothing is connected, so abort the connect flow.
 - If Live State is "HPD Connect for TBT", clear DFLEXDPCSSS.DPPMSTC to '0' and go to the Thunderbolt connect flow.
 - If Live State is "HPD Connect for TypeC", proceed with DP Alternate connect flow.
 5. Display software reads lane assignment from DFLEXDPSP.DPX4TXLATC register.
 6. Display software issues AUX reads for EDID/DPCD.
 - Set DDI_AUX_CTL IO Select field to legacy

- AUX power is controlled through PWR_WELL_CTL_AUX non-Thunderbolt IO Power Request
 - See Type-C Aux Power Requirements for main link usage
7. Based on DFLEXDPSP.DPX4TXLATC and info from DPCD/EDID, display software knows the maximum number of lanes supported.
 8. If software and the OS decide to enable display output on this connected port, software performs the DP mode set enable sequence to bring display controller up and running.
- If connect flow is aborted at any point, clear DFLEXDPCSSS.DPPMSTC to '0' if it had been set, then use GT driver mailbox to unblock TCCOLD.

Connect Flow for Thunderbolt



This flow is for DP tunneling through thunderbolt. The thunderbolt controller will configure the PHY.

1. If display software will not use the port main link or Aux channel
 1. Abort remainder of the flow
 2. Else
 1. Display software uses GT driver mailbox to block TCCOLD
 2. Display software issues AUX reads for EDID/DPCD.
 - Set DDI_AUX_CTL IO Select field to TBT
 - AUX power is controlled through PWR_WELL_CTL_AUX TBT IO Power Request
 - See type-C aux power requirements for main link usage
 3. Based on info from DPCD/EDID, display software knows the maximum number of lanes supported.
 4. If software and the OS decide to enable display output on this connected port, software performs the DP mode set enable sequence to bring display controller up and running.
- If connect flow is aborted at any point, use GT driver mailbox to unblock TCCOLD.

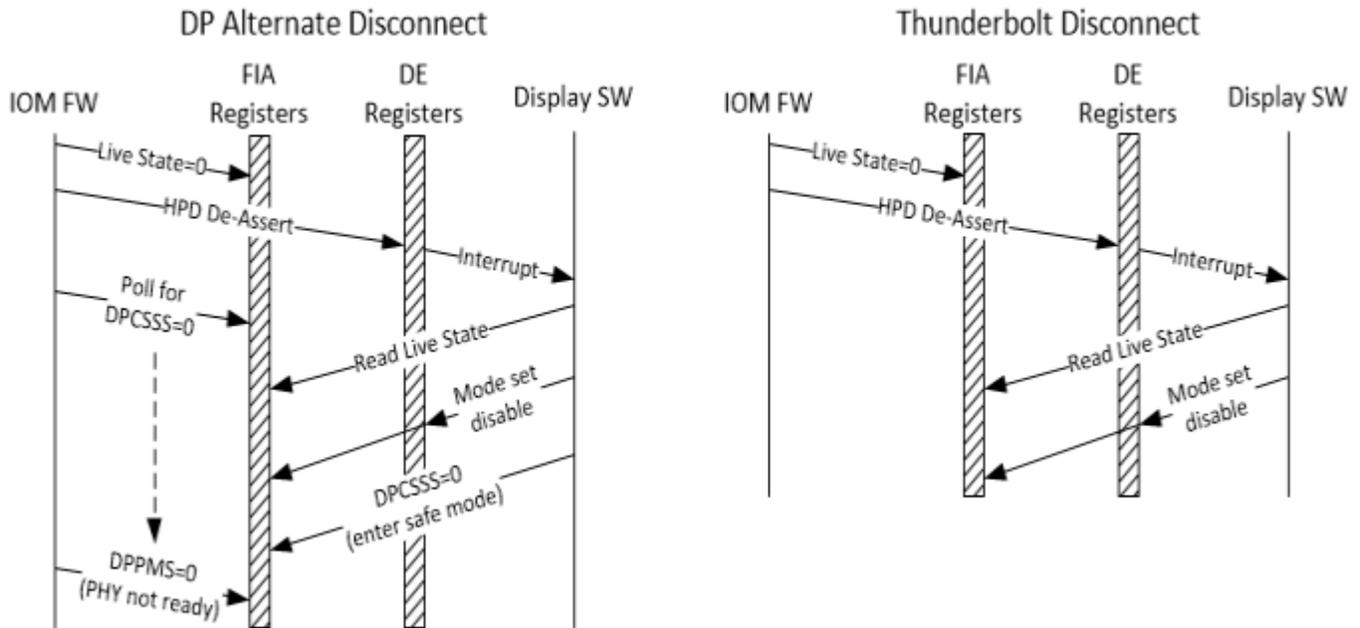
Connect Flow for Static/Fixed/Legacy/Native

This flow is for DP/HDMI as dedicated connection. PHY lane 0 and lane 1 are dedicated to this connection.

Note: IOM is not processing static/fixed/legacy/native connections, but the handshake with FIA DFLEX registers should be followed for consistency with DP alternate mode flow.

1. If display software will not use the port main link or Aux channel
 1. Abort remainder of the flow
2. Else
 1. Display software uses GT driver mailbox to block TCCOLD
 2. Display software reads DFLEXDPPMS.DPPMSTC which should be '1' to indicate the SOC uC has switched the Lane into DP Mode, else abort connect flow.
 3. Display software sets DFLEXDPCSSS.DPPMSTC to '1' indicating that display controller is not in safe mode anymore.
 4. Set PWR_WELL_CTL_AUX non-Thunderbolt IO Power Request.
 5. Poll for PWR_WELL_CTL_AUX non-Thunderbolt IO Power State == Enabled, timeout and fail after 1.5 milliseconds (typically takes <150us).
 6. Display software issues AUX reads for EDID/DPCD.
 - Set DDI_AUX_CTL IO Select field to legacy for DP. Don't care for HDMI.
 - See Type-C Aux Power Requirements for main link usage, even for HDMI.
 7. Based on info from DPCD/EDID, display software knows the maximum number of lanes supported for DP case. For HDMI, it is always 4 lanes.
 8. If software and the OS decide to enable display output on this connected port, software performs the DP or HDMI mode set enable sequence to bring display controller up and running.
- If connect flow is aborted at any point, clear DFLEXDPCSSS.DPPMSTC to '0' if it had been set, then use GT driver mailbox to unblock TCCOLD.

Disconnect Flow



This flow is used for completing a disconnection. This can be initiated upon receiving HPD disconnect interrupt, finding that a HPD disconnect was missed because of reset or power state, or for an early disconnect where software decides it will no longer require the port main link or Aux channel until the next connect flow or power state exit.

By giving an early disconnect, display software allows IOM to immediately change ownership of the port on a real disconnect, without waiting for any display software response.

Display software is required to give an early disconnect if it knows it will be entering states where hotplug cannot be responded to for a long period, such as DC9, D3, S3, S4, S5, and un-installing driver. Software can start the connect flow after exiting these states.

1. Display software follows the mode set disable sequence. This can be done prior to disconnect flow and skipped if there was never a mode set enable.
2. Display software disables Aux power and waits for Aux power state to disable. This cannot be done prior to mode set disable sequence because Aux power is required while Aux channel or main link are in use. Do not change the Aux control IO select field until the connect flow.
3. If not thunderbolt, display software clears `DFLEXDPCSSS.DPPMSTC` to '0' to tell PD FW that it had put the display Controller into Safe State.
 - Note that thunderbolt has no `DFLEXDPCSSS.DPPMSTC` handshake, so it is possible to receive a HPD connect interrupt before this disconnect flow is finished. Display software needs to complete the disconnect flow before moving to connect flow.
4. Display software uses GT driver mailbox to unlock TCCOLD

Link Training

- In Type-C, the connection defines the mode. Hence link training in Type-C is bound by pin assignment.
- It is ok for display engine to use fewer lanes than are available, same as if it were not a Type C connection.
- Display software is responsible for shutting off assigned but unused PHY lane(s) through DFLEXDPMLE.DPMLETC* as described in the mode set section.

Lane Configuration

Lane configuration in the case of Type-C connector (including non Type-C use cases) involves programming DE, FIA, and PHY registers. The programming is described in the mode set sequences and DDI Buffer sections.

IOM handles FIA and PHY register programming for lane configuration, except for shutting off unused lanes.

Lane reversal

Lane reversal is part of the service handled by the FIA, but only when using a type C connector.

With a legacy DP or HDMI connector, reversal is handled via the display engine DDI_BUF_CTL as selected by VBT i.e., it's a boot time configuration which will be done by display software. FIA is agnostic to any lane reversal occurring within the display controller.

Port clock programming

For DP alternate and static/fixed/legacy/native connections, a PLL in the PHY is used. For thunderbolt, a DPLL is used. The programming is described in the port clock programming section.

DDI Buffer specific registers

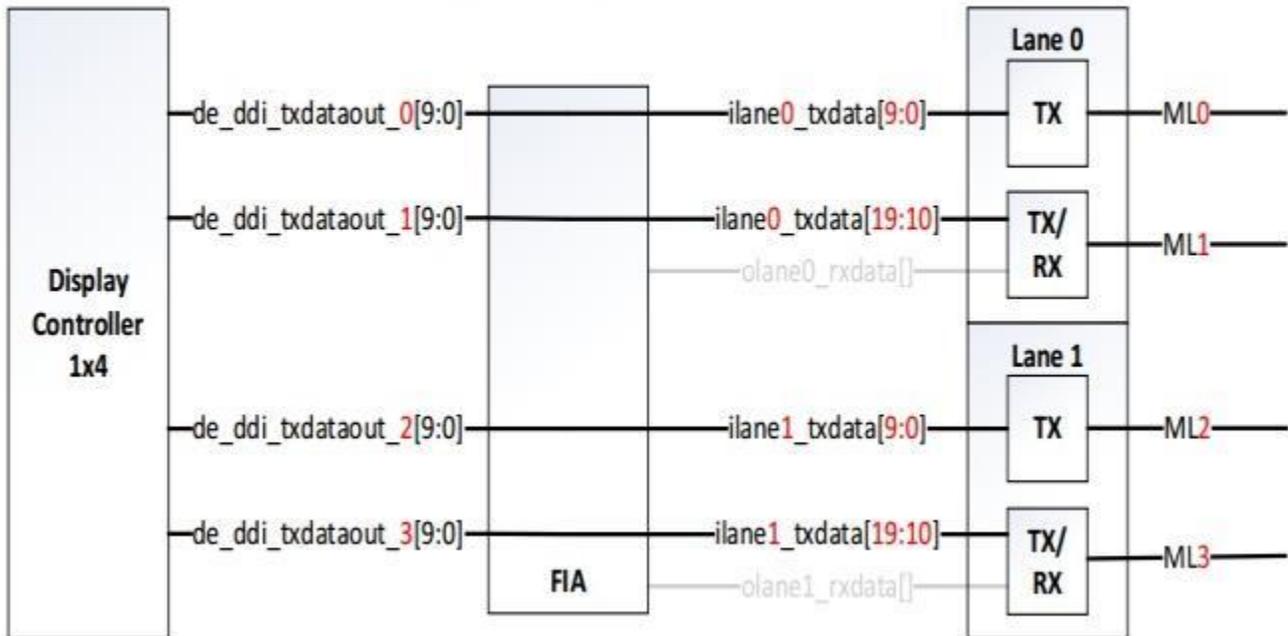
PHY registers are programmed to control voltage swing. The programming is described in the mode set sequences and DDI Buffer sections.

Pin Assignments

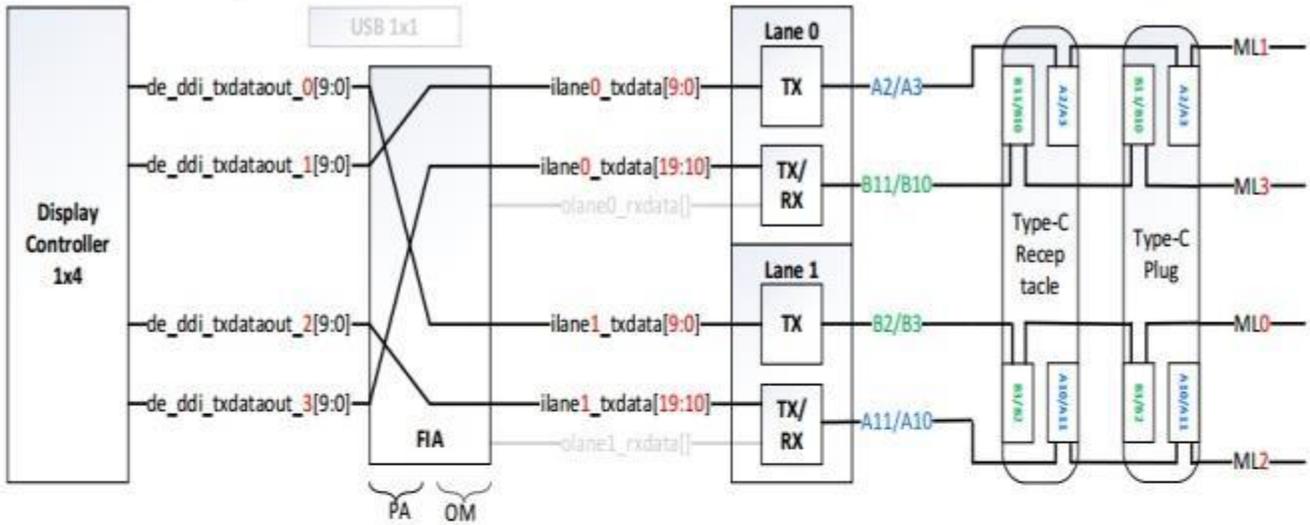
The following are the various FIA supported pin assignments, coming from the type-C specification. This shows the mapping of display data lanes through FIA to the PHY and the receiver, and the cases that limit display to 2 lanes because of using USB on the other lanes.

Please note that DE may use fewer lanes than are available, same as if it were not a Type-C connection.

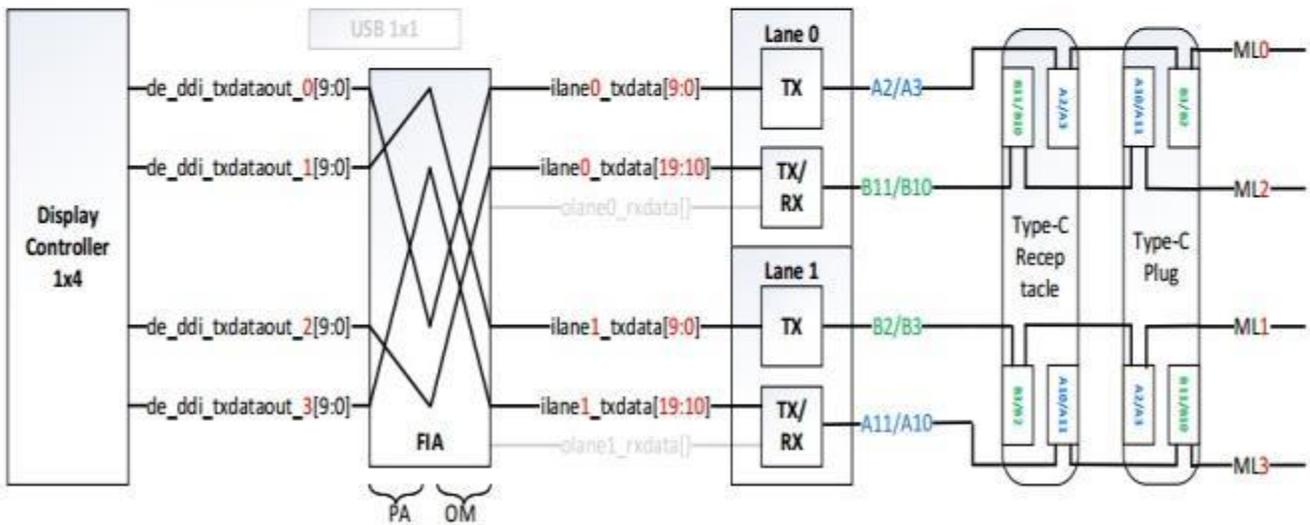
No Pin Assignment (Non Type-C DP)



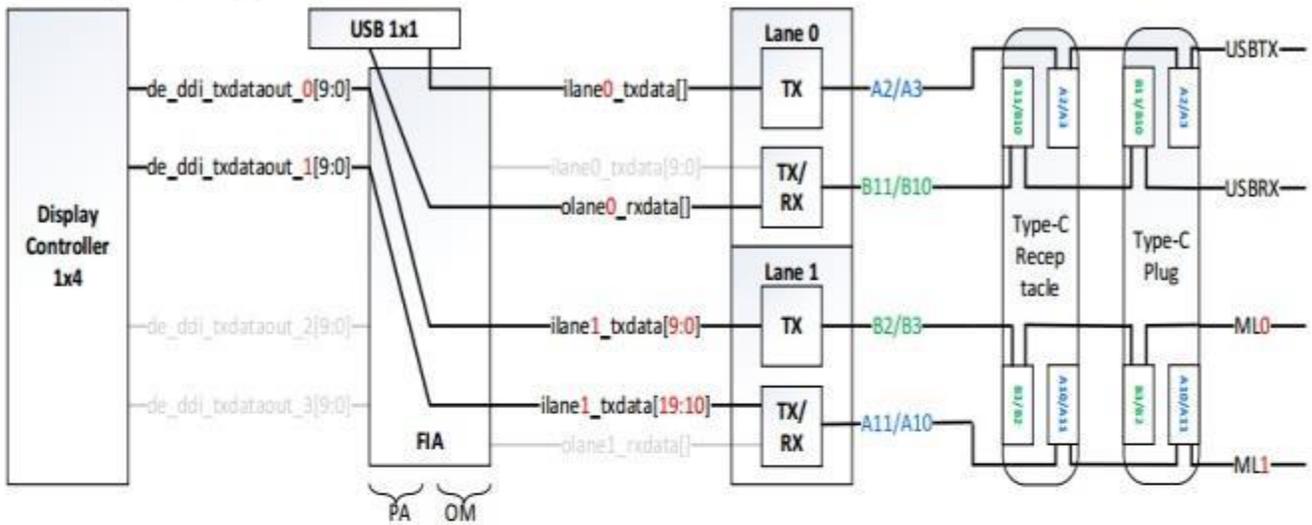
NON-flip: Pin Assignment A



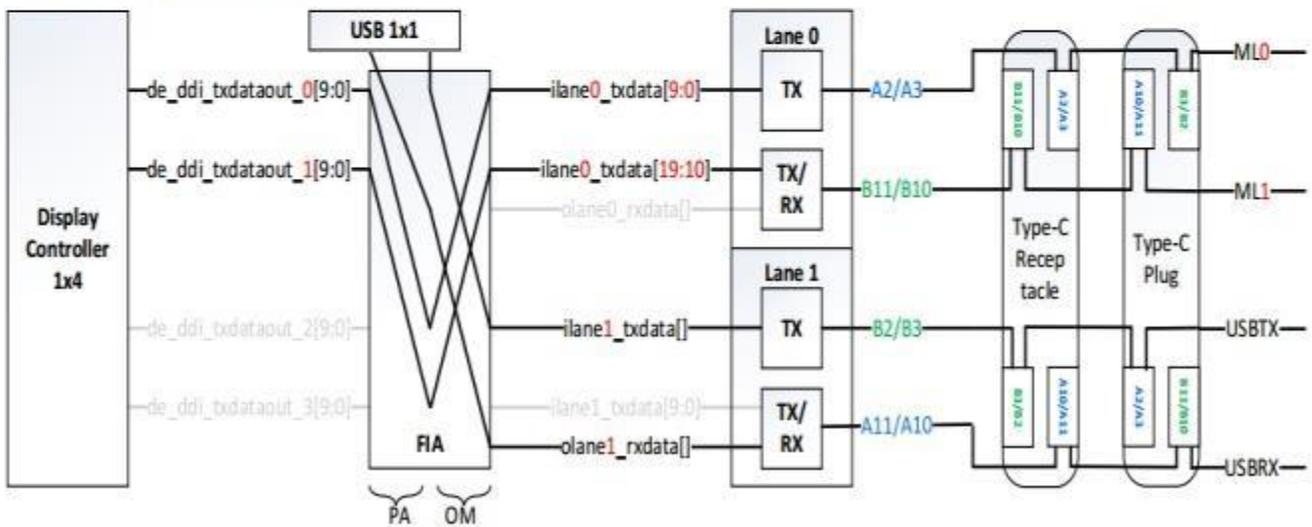
FLIP: Pin Assignment A



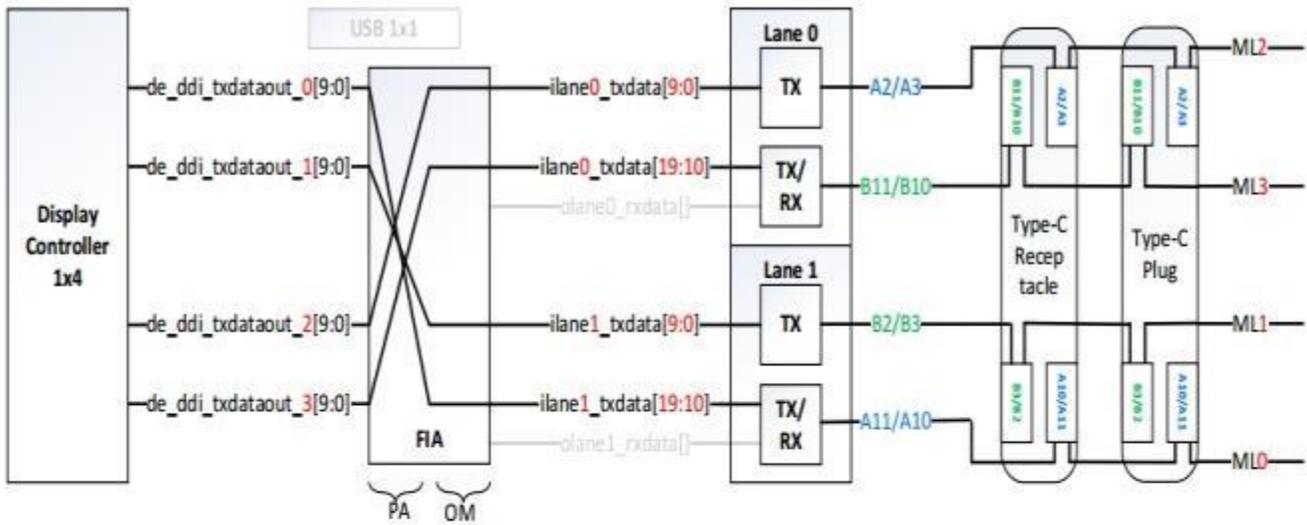
NON-flip: Pin Assignment B



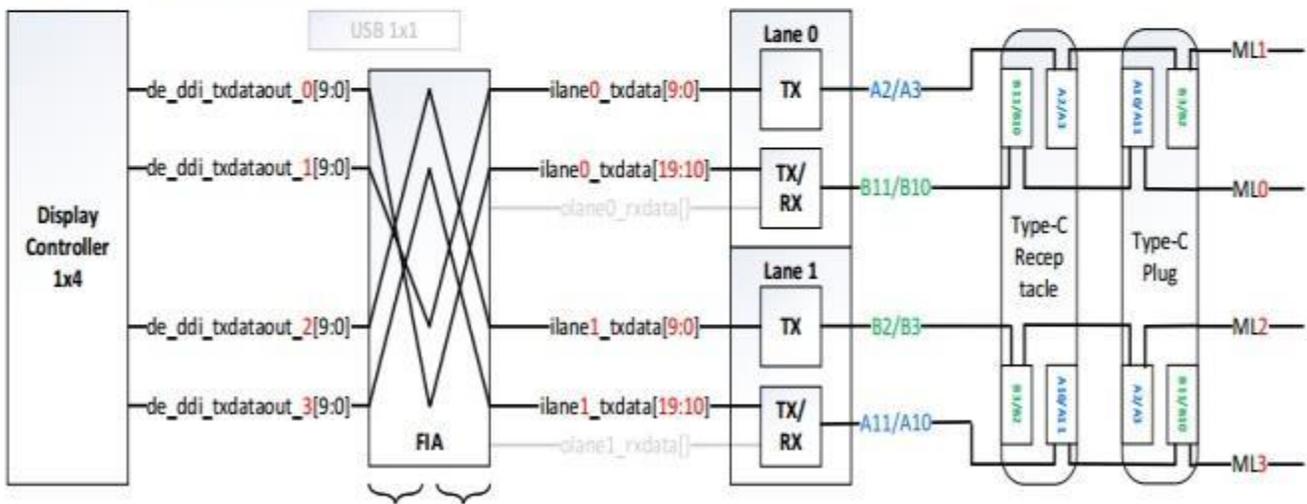
FLIP: Pin Assignment B



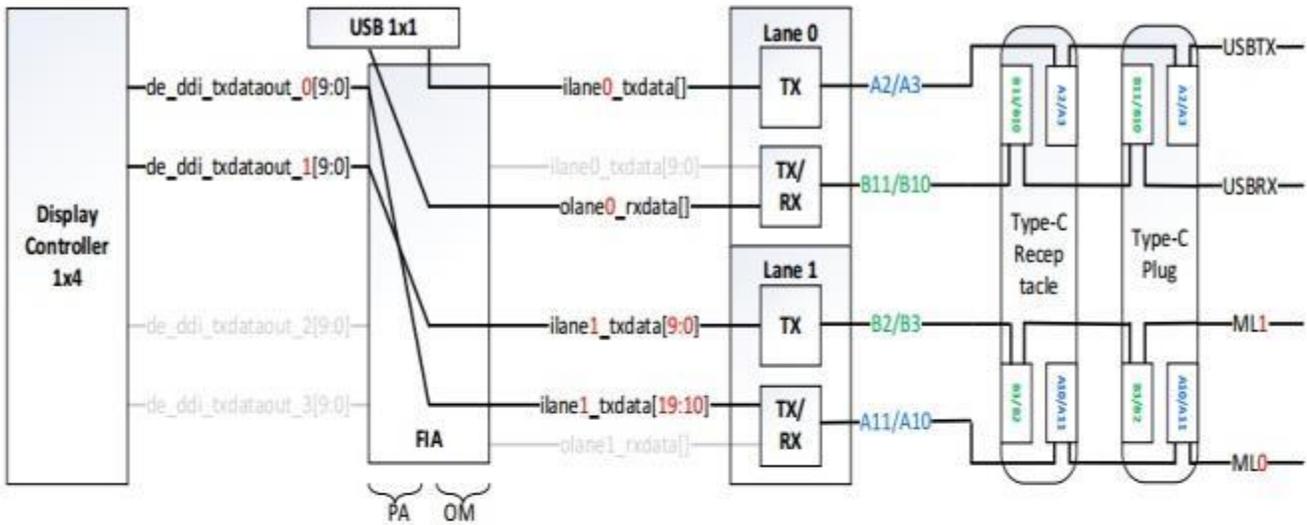
NON-flip: Pin Assignment C/E



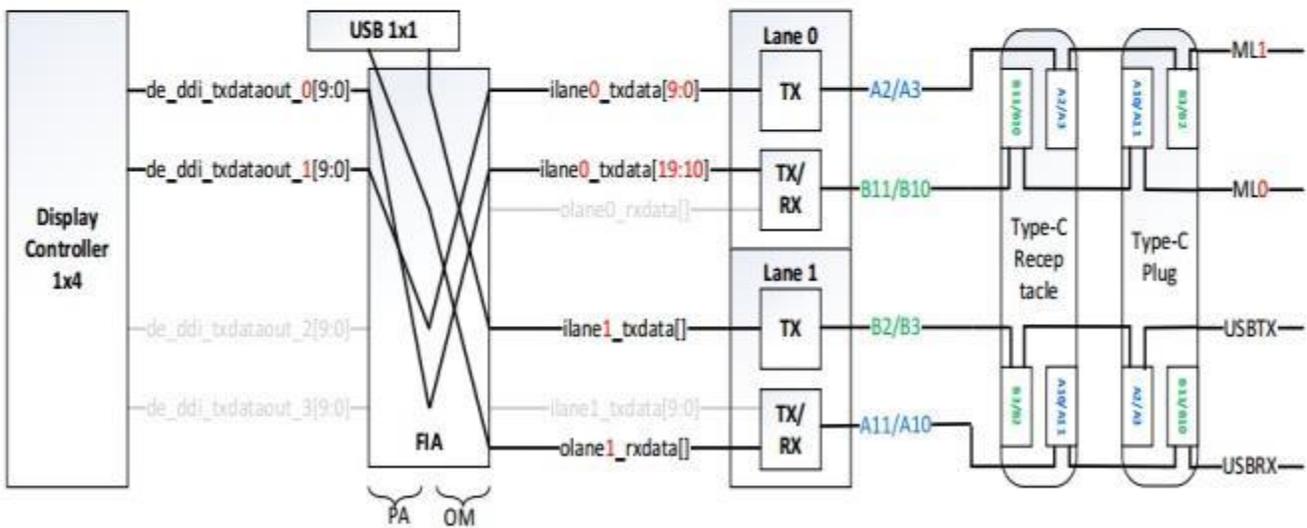
FLIP: Pin Assignment C/E



NON-flip: Pin Assignment D/F



FLIP: Pin Assignment D/F



Dekel PHY Register Access

Addressing Method

Each type-C port PHY is addressed through a 4KB aperture. Each PHY has more than 4KB of register space, so a separate index is programmed in HIP_INDEX_REG0 or HIP_INDEX_REG1, based on the port number, to set the upper 2 address bits that point the 4KB window into the full PHY register space.

Here is a programming example :

1. Assume Type C PHY #1 is allocated 16_8000-16_8FFFh.
2. Assume the Driver wants to write a Phy register 0x1234h.
3. Driver writes at (GTTMMADR+ HIP_INDEX_REG0[3:0] HIP_168_Index) = 01h
4. Driver writes at (GTTMMADR+16_8234h).
 - The write is forwarded to PHY #1 address [15:0] = 1234h

Note: Ignore any address defined within the PHY register. That is incorrect because the index is not accounted for.

PHY Base Addresses and Associated Index

Port #	PHY Base	Index Register and Field
5	0x16C000	HIP_INDEX_REG1 HIP_16C_Index
6	0x16D000	HIP_INDEX_REG1 HIP_16D_Index

MMIO Index Registers

HIP_INDEX_REG0

HIP_INDEX_REG1

PHY Registers

Register Name	PHY Building Block	PHY internal address	MMIO index value	MMIO address
DKL_PLL_DIV0	PLL2 common lane	0x2200	0x2	PHY Base + 0x200
DKL_PLL_DIV1	PLL2 common lane	0x2204	0x2	PHY Base + 0x204
DKL_PLL_LF	PLL2 common lane	0x2208	0x2	PHY Base + 0x208
DKL_SSC	PLL2 common lane	0x2210	0x2	PHY Base + 0x210
DKL_BIAS	PLL2 common lane	0x2214	0x2	PHY Base + 0x214
DKL_TDC_COLDST_BIAS	PLL2 common lane	0x2218	0x2	PHY Base + 0x218
DKL_PLL_FRAC_LOCK	PLL2 common lane	0x220C	0x2	PHY Base + 0x20C
DKL_REFCLKIN_CTL	CMN_ANA	0x212C	0x2	PHY Base + 0x12C
DKL_CLKTOP2_HSCLKCTL	CMN_ANA	0x20D4	0x2	PHY Base + 0x0D4
DKL_CLKTOP2_CORECLKCTL1	CMN_ANA	0x20D8	0x2	PHY Base + 0x0D8
DKL_CMN_UC_DW27	CMN_UC	0x236C	0x2	PHY Base + 0x36C
DKL_TX_DPCNTL0	TX Lane 0	0x02C0	0x0	PHY Base + 0x2C0
DKL_TX_DPCNTL1	TX Lane 0	0x02C4	0x0	PHY Base +

Register Name	PHY Building Block	PHY internal address	MMIO index value	MMIO address
				0x2C4
DKL_TX_DPCNTL2	TX Lane 0	0x02C8	0x0	PHY Base + 0x2C8
DKL_TX_FW_CALIB	TX Lane 0	0x02F8	0x0	PHY Base + 0x2F8
DKL_PMD_FORCE_PCS_IF_TX1	TX Lane 0	0x050C	0x0	PHY Base + 0x50C
DKL_PMD_FORCE_PCS_IF_TX1	TX/RX Lane 0	0x0104	0x0	PHY Base + 0x104
DKL_TX_PMD_LANE_SUS_LN0	TX Lane 0	0x0D00	0x0	PHY Base + 0xD00
DKL_TX_DW17	TX Lane 0	0x0DC4	0x0	PHY Base + 0xDC4
DKL_TX_DW18	TX Lane 0	0x0DC8	0x0	PHY Base + 0xDC8
DKL_TX_DPCNTL0	TX Lane 1	0x12C0	0x1	PHY Base + 0x2C0
DKL_TX_DPCNTL1	TX Lane 1	0x12C4	0x1	PHY Base + 0x2C4
DKL_TX_DPCNTL2	TX Lane 1	0x12C8	0x1	PHY Base + 0x2C8
DKL_TX_FW_CALIB	TX Lane 1	0x12F8	0x1	PHY Base + 0x2F8
DKL_PMD_FORCE_PCS_IF_TX1	TX Lane 1	0x150C	0x1	PHY Base + 0x50C
DKL_PMD_FORCE_PCS_IF_TX1	TX/RX Lane 1	0x1104	0x1	PHY Base + 0x104
DKL_TX_PMD_LANE_SUS_LN1	TX Lane 1	0x1D00	0x1	PHY Base + 0xD00
DKL_TX_DW17	TX Lane 1	0x1DC4	0x1	PHY Base + 0xDC4
DKL_TX_DW18	TX Lane 1	0x1DC8	0x1	PHY Base + 0xDC8
DKL_DP_MODE	ACU Lane 0	0x00A0	0x0	PHY Base + 0x0A0
DKL_DP_MODE	ACU Lane 1	0x10A0	0x1	PHY Base + 0x0A0

DDI AUX Channel

DDI_AUX_CTL

DDI_AUX_DATA

AUX IO Power

Each DisplayPort Aux channel has an Aux IO Power Request. If an Aux channel will not be used, it does not need to be powered up.

PSR spontaneously sends Aux transactions. If PSR is enabled on a port, then the associated Aux IO must be kept powered up.

For Type-C, Aux IO power use has extra requirements. Refer to Type-C Aux power requirements section under Type-C programming page.

AUX IO Power Enabling

1. For combo PHY (non-typeC PHY) Aux channel, the PHY must be initialized.
2. Set **PWR_WELL_CTL_AUX** Aux IO Power Request to 1b.
 - There are two sets of PWR_WELL_CTL_AUX registers for software use. It is expected that BIOS uses PWR_WELL_CTL_AUX1 and driver uses PWR_WELL_CTL_AUX2.
3. Poll for **PWR_WELL_CTL_AUX** Aux IO Power State = 1b.
 - Timeout and fail after 10 us.

AUX IO Power Disabling

1. Clear **PWR_WELL_CTL_AUX** Aux IO Power Request to 0b.

Wait for 10us. Do not poll for the power well to disable. Other clients may be keeping it enabled.

Type-C PHY Microcontroller Health

For every typeC port (static/fixed/legacy/native and DP-alternate, but not thunderbolt), after enabling aux power request and receiving aux power ack, wait for uc_health bit (Dekel PHY register DKL_CMN_UC_DW27 @BASE + 0x236c bit [15]) to assert. Timeout and fail after 10us.

AUX programming sequence

A general purpose AUX functional programming sequence is provided below.

AUX Functional Sequence

Step	Description	Register	Notes
1	Display must already be initialized.		Power well1 enabled cdclk enabled
1a	Aux power enabled		IO powered up as needed for Aux on this project
1b	Power well containing Aux logic powered up	PWR_WELL_CTL	High level power well partitioning shown in display overview diagram
2	Disable PSR1/SRD, PSR2 and GTC if they are enabled on this DDI. Disable DC5 and DC6.	DC_STATE_EN	Power wells will disable automatically for DC5 or DC6. PSR and GTC use Aux spontaneously.
3	Program AUX data registers.	DDI_AUX_DATA_*_[0-4]	
4	Program control to configure AUX and START transaction.	DDI_AUX_CTL_*	Timeout timer value must be at least 600us. To accommodate LT-tunable PHY Repeater AUX delay, AUX Reply Timeout must be programmed to the maximum value. Timer values: 0: 400 us 1: 600 us 2: 800 us 3: 4000 us START trigger: DDI_AUX_CTL_*[31]='1'
5	Wait for AUX transaction complete.		AUX Transaction complete interrupt if set OR when DDI_AUX_CTL_*[31:30] = '01'.
6	Check that receive data has no errors	DDI_AUX_CTL_*[25]	If set: write a '1' to clear this bit and skip reading AUX data registers.
7	Read AUX data register	DDI_AUX_DATA_*_[0-4]	Condition: Aux Channel Control Register Send/Busy bit is NOT asserted
8	Clear status flags	DDI_AUX_CTL_*[30]	Transaction done status

There is a field in DDI_AUX_CTL that must be programmed for the type C ports to select if the Aux transaction will go to thunderbolt

DDI FEC

Reed-Solomon code Forward Error Correction (FEC) function RS (254, 250), with a symbol size of 10 bits is capable of correcting up to two RS symbol errors per FEC block. Display controller when plugged to an FEC-capable DPRX and anticipates enabling FEC encoding sets the FEC_READY bit in the FEC_CONFIGURATION register (DPCD Address 00120h, bit 0) to 1 before initiating link training. Display controller needs to ensure completion of link training before starting FEC encoding. After link training is

complete, display controller, if it needs to enable FEC encoding, shall send an FEC_DECODE_EN sequence to indicate the start of FEC encoding. This prompts DPRX to enable FEC decoding.

When DSC is enabled, FEC shall also be enabled.

DP Support of FEC

DP Support
FEC is supported with DP SST and MST transport modes at all lane widths.

eDP Support of FEC

eDP Support	FEC without PSR	FEC with PSR1	FEC with PSR2
FEC is supported with eDP and all lane widths. PSR restrictions apply.	Yes	Yes	No

Global Time Code (GTC)

Global Time Code

Top Level GTC

GTC_CTL

GTC_DDA_M

GTC_DDA_N

GTC_LIVE

GTC Interrupt Bit Definition

GTC_IMR

GTC_IIR

DDI Level GTC

GTC_PORT_CTL

GTC_PORT_TX_CURR

GTC_PORT_TX_PREV

GTC_PORT_MISC

GTC Target Frequency Selection

For GTC top level logic, CDCLK is taken as an input and scaled to a "target frequency" which has a period which is an exact multiple of 0.5ns. This period is also known as the accumulator increment.

Once a target frequency + accumulator increment is selected, an M and N value can be picked and fine tuned to achieve the scaling from CDCLK to target frequency.



In order for the accumulated GTC Live value to match exactly with the real passage of time, the following must be true:

The target frequency selected must be the CLOSEST possible selection to CDCLK. This corresponds to rounding the accumulator increment to the NEAREST 0.5ns increment with respect to the period of CDCLK.

Example

	Frequency	Period
Given CDCLK	337.500 MHz	2.96 ns
1st closest target frequency/increment	333.333 MHz	3.00 ns
2nd closest target frequency/increment	400.000 MHz	2.50 ns
3rd closest target frequency/increment	285.714 MHz	3.50 ns
4th closest target frequency/increment	500.000 MHz	2.00 ns

Only using the 1st closest target frequency 333.333MHz / accumulator increment of 3.00 ns will result in the GTC Live Value correctly tracking the real passage of time.

Example Calculation Flow:

1. Find the period of CDCLK -- for 337.5 MHz = 2.96 ns.
2. Round to the nearest 0.5 ns -- 3.00 ns (this is your accumulator increment value).
3. Find the "target frequency" from the rounded period value of step 2 -- 333.333 MHz is the "target frequency".
4. Find the ratio of target frequency / CDCLK = 0.987654321.
5. Choose M and N to satisfy $M / N =$ same ratio as step 4.

If the results are not within the 1% error tolerance, multiply the accumulator increment used by 2 (i.e., use 1/2 of the target frequency). Re-select M and N as necessary to achieve the new target frequency as shown in the example below.

Example

	Frequency	Period	with 2x accumulator increment	New Period
Given CDCLK	652.800 MHz	1.531 ns		
1st closest target frequency/increment	500.000 MHz	1.500 ns	3.000ns	333.333 MHz

Example Calculation Flow:

1. Find the period of CDCLK -- for 652.800 MHz = 1.531 ns.
2. Round to the nearest 0.5 ns -- 1.500 ns
3. Multiply the above increment by 2 (this 3.00ns is your accumulator increment value).
4. Find the "target frequency" from the rounded period value of step 3 -- 333.333 MHz is the "target frequency".
5. Find the ratio of target frequency / CDCLK = 333.333 MHz / 652.800 MHz = 0.51062.

Choose M and N to satisfy $M / N =$ same ratio as step 5.

Display Watermark Programming

Watermark Overview

The display watermarks are used to control the display engine memory request behavior.

Description
The default settings of the watermark configuration registers will not allow the display engine to operate. The watermark values must be properly calculated and programmed in order to enable a display and achieve optimum power and performance. Incorrectly programmed watermark values can result in screen corruption.

The watermarks should be calculated and programmed when any of the watermark calculation inputs change. This includes planes enabling or disabling, plane source format or size changing, etc.

Besides programming the watermark registers, there are other display configuration requirements and registers that must be programmed in order for the display to operate in a low power mode, and there are memory controller configuration requirements which are not documented here.

Watermark Calculations

The display watermarks are calculated using information from the display configuration and memory latencies. The watermarks must be calculated and programmed before enabling a plane or changing a plane configuration.

YUV 420 planar surface format	Plane source bytes per pixel
NV12	1 Bpp for Y surface and 2 Bpp for UV surface. Watermark values must be calculated and programmed for Y and UV surfaces separately in their watermark registers.
P0xx	2 Bpp for Y surface and 4 Bpp for UV surface. Watermark values must be calculated and programmed for Y and UV surfaces separately in their watermark registers.

The ceiling function rounds any non-integer value up to the next greater integer. Example: $\text{ceiling}[0.3]=1$, $\text{ceiling}[2.1]=3$, $\text{ceiling}[4.8]=5$, $\text{ceiling}[4]=4$

Resolutions Requiring Combined Pipes

For resolutions requiring 2 or more pipes to be joined together inside display engine, each pipe processes only $1 / \langle \# \text{ of joined pipes} \rangle$ of the image. The pixel rate and horizontal total pixels seen by each pipe is $1 / \langle \# \text{ of joined pipes} \rangle$ of that for the full resolution of the panel.

The excess pixels added for scaling smoothly across the seam between pipes do not impact the watermark.

For example: 7680x4320 CVT1.2 RB1 pixel rate is 2089.75 MHz with horizontal total 7840. That is split across 2 pipes, so each pipe is 3840x4320 with a pixel rate of 1044.875 MHz and horizontal total 3920.



Watermark Algorithm

1. Retrieve memory latency values
 - See the Memory Values section to find the memory latency values
 - The memory values do not change after boot, so software may cache them to avoid re-reading
2. For each enabled pipe (run each time pipe configuration changes)
 - A. Calculate adjusted pipe pixel rate
 - I. Adjusted pipe pixel rate = pixel rate for the screen resolution
 - If there will be dynamic switching between refresh rates, either use the fastest pixel rate, or re-calculate using the current pixel rate when the refresh rate is switched. Because Plane watermark registers and Link M/N registers have different double buffer update points, software will have to program both at the beginning of vactive. Alternatively, when switching from a low to high refresh rate, program the WM values first and when switching from a high to low refresh rate program the Link M/N values first. This second approach can result in a one frame delay in the refresh rate switch.
 - If plane 90 or 270 rotation is enabled, use the rotated width and height in pixel rate calculations.
 - II. If TRANS_CONF Interlaced Mode == PF-ID, adjusted pipe pixel rate = adjusted pipe pixel rate * 2
 - III. If pipe scaling enabled, adjusted pipe pixel rate = adjusted pipe pixel rate * pipe down scale amount
 - See the Scaling section to find the down scale amount
 - B. Program WM_LINETIME Line Time = roundup[8 * pipe horizontal total pixels / adjusted pipe pixel rate MHz]
3. For each enabled plane (run each time pipe or plane configuration changes)
 - A. Calculate adjusted plane pixel rate
 - I. Adjusted plane pixel rate = adjusted pipe pixel rate
 - II. If plane scaling enabled, adjusted plane pixel rate = adjusted plane pixel rate * plane down scale amount
 - See the Scaling section to find the down scale amount
 - B. For each valid memory latency level

Plane Bytes per pixel	Minimum Scanlines for Y Tile	
	0/180 Rotation	90/270 Rotation
1	4	16
2	4	8
4	4	4
8	4	N/A

Plane Memory format	DBuf block size
8 bits per pixel surface format + Yf tiling	256
All other tiling and surface formats	512

- I. Calculate method 1
 - Method 1 = (memory latency microseconds * adjusted plane pixel rate MHz * plane source bytes per pixel / DBuf block size) + 1
- II. Calculate method 2
 - plane bytes per line = plane source width pixels * plane source bytes per pixel
 - Calculate plane blocks per line
 - If plane memory format is Linear
 - plane blocks per line = ceiling[plane bytes per line / DBuf block size] + 1
 - Else If plane memory format is Y tile
 - plane blocks per line = ceiling[(Minimum Scanlines for Y tile * plane bytes per line / DBuf block size) + 1] / Minimum Scanlines for Y tile
 - Else
 - plane blocks per line = ceiling[plane bytes per line / DBuf block size] + 1
 - Method 2 = ceiling[(memory latency microseconds * adjusted plane pixel rate MHz) / Pipe horizontal total number of pixels] * plane blocks per line
- III. Calculate Y tile minimum
 - Y tile minimum = Minimum Scanlines for Y tile * plane blocks per line
- IV. Select the watermark result
 - line time microseconds = pipe horizontal total pixels / adjusted plane pixel rate MHz
 - If plane memory format is X tile or linear
 - If (((plane source bytes per pixel * pipe horizontal total number of pixels) / DBuf block size) < 1) AND ((plane bytes per line / DBuf block size) < 1) // Special case for unrealistically small horizontal total
 - Selected Result Blocks = Method 2
 - Else If ('plane buffer allocation' is known and (plane buffer allocation / plane blocks per line) >= 1)
 - Selected Result Blocks = Method 2
 - Else If (memory latency microseconds >= line time microseconds)
 - Selected Result Blocks = Method 2
 - Else
 - Selected Result Blocks = Method 1
 - Else // Y tile

- Selected Result Blocks = maximum[Method 2, Y tile minimum]
- If Pipe YUV420 Bypass // PIPE_MISC YUV420 Enable == Enable and YUV420 Mode == Bypass
 - Selected Result Blocks = maximum[Selected Result Blocks, plane blocks per line] // Minimum is 1 line when using pipe YUV420 bypass

V. Convert result to blocks and lines

- Result Blocks = ceiling[Selected Result Blocks] + 1
- Result Lines = ceiling[Selected Result Blocks / plane blocks per line]
- If Y Tiling
 - If 'Result Lines' is multiple of 'Minimum Scanlines for Y tile'
 - Extra Lines = Minimum Scanlines for Y tile
 - Else
 - Extra Lines = (Minimum Scanlines for Y tile * 2) - (Result Lines % Minimum Scanlines for Y tile)
 - Minimum Display Buffer allocation Needed = ceiling[(Result Lines + Extra Lines) * plane blocks per line]
 - Else
 - Minimum Display Buffer allocation Needed = ceiling[Result Blocks + (Result Blocks * 0.1)]

VI. Compare against the maximum

- If (Result Blocks >= plane buffer allocation), maximum exceeded for this latency level
- Maximum Lines: 31
- If (Result Lines > Maximum Lines), maximum exceeded for this latency level
- If (Minimum Display Buffer allocation Needed >= plane buffer allocation), maximum exceeded for this latency level
- or YUV 420 Planar formats, perform the above check for both Y and UV planes.

4. For each transition watermark

- This includes <PLANE,CUR>_WM_TRANS, which is calculated relative to memory latency level 0.
- A. Calculate transition offset
 - Transition Offset Blocks = Transition minimum + Transition amount
 - See Transition Watermark section for transition minimum and transition amount
- B. Calculate transition Y tile minimum
 - Transition Y tile minimum = 2 * relative memory latency level Y tiled minimum
- C. Select the watermark result
 - If plane memory format is X tile or linear

- Result Blocks = Relative memory latency level Selected Result Blocks + Transition Offset Blocks
 - Else // Y tile
 - Result Blocks = maximum[Relative memory latency level Selected Result Blocks, Transition Y tile minimum] + Transition Offset Blocks
- D. Convert result to blocks
- Result Blocks = ceiling[Result Blocks] + 1
- E. Compare against the maximum
- If (Result Blocks >= plane buffer allocation), maximum exceeded for transition watermark
 - For YUV 420 Planar formats, perform the above check for both Y and UV planes.
5. Program watermark registers
- A. For each latency level
- If memory latency for this level is invalid, or the maximum was exceeded for this level or any previous level, program PLANE_WM_<latency level> Enable = 0
 - **If watermark latency level 0 exceeds the maximum, the plane must not be enabled.**
 - Else program PLANE_WM_<latency level> Enable = 1, Lines = Result Lines, Blocks = Result Blocks
- B. For transition watermarks
- If the maximum was exceeded for the transition watermark, program the transition watermark register Enable = 0
 - Else program the transition watermark register Enable = 1, Blocks = Transition Result Blocks
 - The transition watermark Lines value is ignored by hardware
- C. Write the plane surface base address register to trigger update of the watermarks and other plane double buffered registers. This should be done only after all plane configuration is configured to match the new watermark values.

Transition Watermark

The transition watermark is used for Isochronous Priority Control (IPC). When IPC is enabled, plane read requests are sent at high priority until filling above the transition watermark, then the requests are sent at lower priority until dropping below the level 0 watermark. The lower priority requests allow other memory clients to have better memory access. If the transition watermark is not enabled, the plane behaves as if the transition watermark was programmed to the top of the plane buffer allocation. When IPC is disabled, all plane read requests are sent at high priority. It is allowed for one of a pair of planes supporting YUV420 together to have transition watermark disabled while the other plane has it enabled.

The transition watermark is programmed as a tunable amount above the relative memory latency level watermark. Tuning to higher values will tend to cause longer periods of high priority reads followed by



longer periods of lower priority reads. Tuning to lower values will tend to cause shorter periods of high and lower priority reads. The exact behavior depends on the memory bandwidth, display bandwidth, and other memory traffic in the system.

The transition watermark has a minimum value to ensure the demote does not happen before enough data has been read to meet the relative memory latency level watermark requirements.

Transition Minimum: 4 Blocks

IPC Enable Register Field
ARB_CTL2 Enable IPC

Scaling

A scaler (pipe or plane scaler) is down scaling when it is enabled and the scaler input size is greater than the scaler output size.

Down scaling effectively increases the pixel rate. Up scaling does not reduce the pixel rate.

For plane scaling, the scaler input size is the plane size and the output size is the scaler window size.

For pipe scaling, the scaler input size is the pipe source size and the output size is the scaler window size.

Horizontal down scale amount = maximum[1, Horizontal source size / Horizontal destination size]

Vertical down scale amount = maximum[1, Vertical source size / Vertical destination size]

Total down scale amount = Horizontal down scale amount * Vertical down scale amount

System Agent Geyserville (SAGV) Interaction with Watermarks

SAGV dynamically adjusts the system agent voltage and clock frequencies depending on power and performance requirements. The display engine access to system memory is blocked outside of display engine, without a PM handshake, during the adjustment time.

To compensate for the block, display engine must increase the latency for watermark level 0 by the SAGV block time. If there is not enough data buffer to support that increase, then software must disable SAGV. Other watermark levels do not need increase because after a package C exit SAGV will not run until after display has refilled above watermark level 0.

SAGV defaults to enabled.

See the Memory Values section to find the SAGV block time.

The latency input to the watermark calculation for each level must be greater than or equal to the lower level. The latency increase to level 0 for SAGV requires the upper levels to be adjusted to meet that requirement. Use $\text{MIN}(\text{latency for this level, latency for next lower level})$ to correct the latency.

Requirement before plane enabling or configuration change: Calculate watermark level 0 with level 0 latency + SAGV block time. If the result can be supported (does not exceed maximum), then the plane can tolerate SAGV, so proceed with watermark calculations for level 0 using latency increased by SAGV block time.

If the plane cannot tolerate SAGV, proceed with watermark calculations for all levels using the normal latency

values.

Disable SAGV (sequence below) if any enabled plane cannot tolerate SAGV, or any transcoder is interlaced. Else, enable SAGV.

The expectation is that SAGV can be tolerated in most non-interlaced configurations and will only need to be disabled for some high resolution, multiple display scenarios.

If one plane was already enabled and able to tolerate SAGV, then a second plane becomes enabled and cannot tolerate SAGV, causing SAGV to disable, software can recalculate and reprogram the watermarks for the first plane without the SAGV block time. That reprogramming is preferred to give more optimal watermark results, but not required.

To disable SAGV, follow the SAGV section SAGV Point Selection Runtime flow to mask off all but the one QGV point that supplies the highest bandwidth for display.

To enable SAGV, follow the SAGV section SAGV Point Selection Runtime flow to unmask all of the QGV points that supply enough bandwidth for the display configuration.

Examples

Example pixel rate adjustments:

Pixel rate for screen resolution is 130 MHz. No interlacing. Pipe scale 1920x1080 pipe source size to 1714x1120 scaler window size. Plane scale 1920x1080 plane size to 800x600 scaler window size.

Pipe horizontal down scale amount = $\text{maximum}[1, 1920 / 1714] = 1.12$

Pipe vertical down scale amount = $\text{maximum}[1, 1080 / 1120] = 1$ // **Max condition was hit**

Pipe total down scale amount = $1.12 * 1 = 1.12$

Adjusted pipe pixel rate = 130 MHz * 1.12 = 145.6 MHz

Plane horizontal down scale amount = $\text{maximum}[1, 1920 / 800] = 2.4$

Plane vertical down scale amount = $\text{maximum}[1, 1080 / 600] = 1.8$

Plane total down scale amount = $2.4 * 1.8 = 4.32$

Adjusted plane pixel rate = 145.6 MHz * 4.32 = 628.99 MHz

Example method, block, and line calculations:

Plane source 4 Bpp, Plane X tile, Plane source width 1920 pixels, Horizontal total 2200 pixels, Adjusted plane pixel rate 148.5 MHz, memory latency 7.5 us

Method 1 = $148.5 \text{ MHz} * 4 \text{ Bpp} * 7.5 \text{ us} / 512 = 8.7 \text{ blocks}$

Plane bytes per line = $1920 \text{ pixels} * 4 \text{ Bpp} = 7680 \text{ Bytes/line}$

Plane blocks per lines = $\text{ceiling}[7680 / 512] = 15 \text{ blocks}$

Method 2 = $\text{ceiling}[(7.5 \text{ us} * 148.5 \text{ MHz}) / 2200 \text{ pixels}] * 15 \text{ blocks} = 15 \text{ blocks}$



Y tile minimum = 4 * 15 blocks = 60 blocks

Result Blocks = minimum[8.7 blocks, 15 blocks] = 8.7 blocks // X tile so does not use Y tile minimum

Result Blocks = ceiling[8.7 blocks] + 1 block = 10 blocks

Result Lines = ceiling[8.7 blocks / 15] = 1 lines

Memory Values

Retrieve Memory Latency Data

1. Ensure any previous GT Driver Mailbox transaction is complete.
2. Write GT Driver Mailbox Data0=0x0000_0000 (first set of latency values) and GT Driver Mailbox Data1=0x0000_0000
3. Write GT Driver Mailbox Interface Run/Busy=1, Address Control=All 0s, Command/Error Code=06h
4. Poll GT Driver Mailbox Interface for Run/Busy indication=0b and Command/Error Code=00h (success)
 - Timeout after 100 us and do not enable display planes.
5. Read GT Driver Mailbox Data0 for the first set of memory latency values
6. Write GT Driver Mailbox Data0=0x0000_0001 (second set of latency values) and GT Driver Mailbox Data1=0x0000_0000
7. Write GT Driver Mailbox Interface Run/Busy=1, Address Control=All 0s, Command/Error Code=06h
8. Poll GT Driver Mailbox Interface for Run/Busy indication=0b and Command/Error Code=00h (success)
 - Timeout after 100 us and do not enable display planes.
9. Read GT Driver Mailbox Data0 for the second set of memory latency values

Memory Latency Data Definition

First Set:

Data0 Bit	Name	Description
31:24	Level 3	Number of microseconds for level 3.
23:16	Level 2	Number of microseconds for level 2.
15:8	Level 1	Number of microseconds for level 1.
7:0	Level 0	Number of microseconds for level 0.

Second Set:

Data0 Bit	Name	Description
31:24	Level 7	Number of microseconds for level 7.
23:16	Level 6	Number of microseconds for level 6.
15:8	Level 5	Number of microseconds for level 5.

Data0 Bit	Name	Description
7:0	Level 4	Number of microseconds for level 4.

If level 1 or any higher level has a value of 0x00, that level and any higher levels are unused and invalid, so the associated watermark registers must not be enabled. Level 0 is always valid.

It is allowed to have the same value in adjacent levels.

Programming Note	
Context:	Display Watermark Programming
<p>The mailbox response data may not account for memory read latency. If the mailbox response data for level 0 is 0us, add 3 microseconds to the result for each valid level.</p> <p>The previous requirement was 2us, but high density DIMMs have increased refresh times that requires a 1us increase.</p>	

SAGV Block Time

The block time is retrieved from the GT driver pcode mailbox.

1. Ensure any previous GT Driver Mailbox transaction is complete.
2. Write GT Driver Mailbox Data0=0x0000_0000 (first set of latency values) and GT Driver Mailbox Data1=0x0000_0000
3. Write GT Driver Mailbox Interface Run/Busy=1, Address Control=All 0s, Command/Error Code=23h
4. Poll GT Driver Mailbox Interface for Run/Busy indication=0b and Command/Error Code=00h (success)
 - Timeout after 100 us and do not enable display planes.
5. Read GT Driver Mailbox Data0 for the block time in microseconds