Intel® Open Source HD Graphics

Programmer's Reference Manual

For the 2016 Intel Atom™ Processors, Celeron™ Processors, and Pentium™ Processors based on the "Apollo Lake" Platform (Broxton Graphics)

Volume 7: Display

May 2017, Revision 1.0

Creative Commons License

You are free to Share - to copy, distribute, display, and perform the work under the following conditions:

- **Attribution.** You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
- **No Derivative Works.** You may not alter, transform, or build upon this work.

Notices and Disclaimers

Table of Contents

Display Audio Codec Verbs

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.

Node ID 00h Root Node Verbs

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

F00h - Get 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	2809h	**Device ID (DID):** Indicates the 16-bit Device ID values used to identify the codec to the PnP subsystem. It is 280Ah for BXT
15:00	2809	**Device ID (DID):** Indicates the 16-bit Device ID values used to identify the codec to the PnP subsystem.

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

North Display Engine Registers

MIPI

MIPI Configurations

Configuration	Supported
Isolated MIPI-A (single display using single link)	Yes
Isolated MIPI-C	No
Isolated MIPI-A & Isolated MIPI-C (dual independent display using dual MIPI)	No
Dual Link Mode with MIPI-A as primary (single display using dual links)	Yes
Dual Link Mode with MIPI-C as primary	No

MIPI DSI Registers

Note
These registers must all be written only as full 32 bit Dwords. Byte or word writes are not supported.
MIPI DSI clocks MUST be running in order to read any of the MIPI registers.

Dual Link Programming	
Dual Link Operation	**Bit/Register**
MIPI dual link enable	Bit 0 of MIPI_PORT_CTL
MIPI dual link mode	Bit 26 of MIPI_PORT_CTL
Z-inversion count	Bit [13:10] of MIPI_CTRL

Address	Name
6B000h	**MIPIA_DEVICE_READY_REG**
6B004h	**MIPIA_INTR_STAT_REG**
6B008h	**MIPIA_INTR_EN_REG**

Address	Name
6B00Ch	MIPIA_DSI_FUNC_PRG_REG
6B010h	MIPIA_HS_TX_TIMEOUT_REG
6B014h	MIPIA_LP_RX_TIMEOUT_REG
6B018h	MIPIA_TURN_AROUND_TIMEOUT_REG
6B01Ch	MIPIA_DEVICE_RESET_TIMER
6B020h	MIPIA_DPI_RESOLUTION_REG
6B024h	MIPIA_DBI_FIFO_THRTL_REG
6B028h	MIPIA_HORIZ_SYNC_PADDING_COUNT
6B02Ch	MIPIA_HORIZ_BACK_PORCH_COUNT
6B030h	MIPIA_HORIZ_FRONT_PORCH_COUNT
6B034h	MIPIA_HORIZ_ACTIVE_AREA_COUNT
6B038h	MIPIA_VERT_SYNC_PADDING_COUNT
6B03Ch	MIPIA_VERT_BACK_PORCH_COUNT
6B040h	MIPIA_VERT_FRONT_PORCH_COUNT
6B044h	MIPIA_HIGH_LOW_SWITCH_COUNT
6B048h	MIPIA_DPI_CTRL_REG
6B04Ch	MIPIA_DPI_DATA_REGISTER
6B050h	MIPIA_INIT_COUNT_REGISTER
6B054h	MIPIA_MAX_RETURN_PKT_SIZE_REGISTER
6B058h	MIPIA_VIDEO_MODE_FORMAT_REGISTER
6B05Ch	MIPIA_EOT_DISABLE_REGISTER
6B060h	MIPIA_LP_BYTECLK_REGISTER

Address	Name
6B064h	MIPIA_LP_GEN_DATA_REGISTER
6B068h	MIPIA_HS_GEN_DATA_REGISTER
6B06Ch	MIPIA_LP_GEN_CTRL_REGISTER
6B070h	MIPIA_HS_GEN_CTRL_REGISTER
6B074h	MIPIA_GEN_FIFO_STAT_REGISTER
6B078h	MIPIA_HS_LS_DBI_ENABLE_REG
6B07Ch	MIPI_HS_READ_TRANSFER_COUNT
6B080h	MIPIA_DPHY_PARAM_REG
6B084h	MIPIA_DBI_BW_CTRL_REG
6B088h	MIPIA_CLK_LANE_SWITCHING_TIME_CNT
6B08Ch	MIPIA_STOP_STATE_STALL
6B090h	MIPIA_INTR_STAT_REG_1
6B094h	MIPIA_INTR_EN_REG_1
6B098h	MIPIA_CLK_LANE_TIMING
6B09Ch	MIPIA_PLL_LOCK_COUNT
6B0A0h	MIPIA_DATA_LANE_POLARITY
6B0A4h	MIPIA_TLPX_TIME_COUNT
6B0C0h	MIPIA_PORT_CTRL
6B0C4h	MIPIA_STATUS
6B0C8h	MIPIA_AUTOPWG
6B0CCh	MIPIA_WR_DATA 0
6B0D0h	MIPIA_WR_DATA 1

Address	Name
6B0D4h	MIPIA_WR_DATA 2
6B0D8h	MIPIA_WR_DATA 3
6B0DCh	MIPIA_WR_DATA 4
6B0E0h	MIPIA_WR_DATA 5
6B0E4h	MIPIA_WR_DATA 6
6B0E8h	MIPIA_WR_DATA 7
6B0ECh	MIPIA_WR_COMMAND
6B0F0h	MIPIA_WR_DATA_CTRL
6B0F4h	MIPIA_EN_DLY_CNTR
6B0F8h	MIPIA_TRANS_HACTIVE
6B0FCh	MIPIA_TRANS_VACTIVE
6B100h	MIPIA_TRANS_VTOTAL
6B104h	MIPIA_CTRL
6B108h	MIPIA_TE_CTR
6B118h	MIPIA_RD_DATA_RETURN 0
6B11Ch	MIPIA_RD_DATA_RETURN 1
6B120h	MIPIA_RD_DATA_RETURN 2
6B124h	MIPIA_RD_DATA_RETURN 3
6B128h	MIPIA_RD_DATA_RETURN 4
6B12Ch	MIPIA_RD_DATA_RETURN 5
6B130h	MIPIA_RD_DATA_RETURN 6
6B134h	MIPIA_RD_DATA_RETURN 7

Address	Name
6B138h	MIPIA_RD_DATA_VALID
6B13Ch	MIPIA_CRC_CTL
6B140h	MIPIA_CRC_RESULT
6B144h	MIPIA Dual Link Buffer Control
6B800h	MIPIC_DEVICE_READY_REG
6B804h	MIPIC_INTR_STAT_REG
6B808h	MIPIC_INTR_EN_REG
6B80Ch	MIPIC_DSI_FUNC_PRG_REG
6B810h	MIPIC_HS_TX_TIMEOUT_REG
6B814h	MIPIC_LP_RX_TIMEOUT_REG
6B818h	MIPIC_TURN_AROUND_TIMEOUT_REG
6B81Ch	MIPIC_DEVICE_RESET_TIMER
6B820h	MIPIC_DPI_RESOLUTION_REG
6B824h	MIPIC_DBI_FIFO_THRTL_REG
6B828h	MIPIC_HORIZ_SYNC_PADDING_COUNT
6B82Ch	MIPIC_HORIZ_BACK_PORCH_COUNT
6B830h	MIPIC_HORIZ_FRONT_PORCH_COUNT
6B834h	MIPIC_HORIZ_ACTIVE_AREA_COUNT
6B838h	MIPIC_VERT_SYNC_PADDING_COUNT
6B83Ch	MIPIC_VERT_BACK_PORCH_COUNT
6B840h	MIPIC_VERT_FRONT_PORCH_COUNT
6B844h	MIPIC_HIGH_LOW_SWITCH_COUNT

Address	Name
6B848h	MIPIC_DPI_CTRL_REG
6B84Ch	MIPIC_DPI_DATA_REGISTER
6B850h	MIPIC_INIT_COUNT_REGISTER
6B854h	MIPIC_MAX_RETURN_PKT_SIZE_REGISTER
6B858h	MIPIC_VIDEO_MODE_FORMAT_REGISTER
6B85Ch	MIPIC_EOT_DISABLE_REGISTER
6B860h	MIPIC_LP_BYTECLK_REGISTER
6B864h	MIPIC_LP_GEN_DATA_REGISTER
6B868h	MIPIC_HS_GEN_DATA_REGISTER
6B86Ch	MIPIC_LP_GEN_CTRL_REGISTER
6B870h	MIPIC_HS_GEN_CTRL_REGISTER
6B874h	MIPIC_GEN_FIFO_STAT_REGISTER
6B878h	MIPIC_HS_LS_DBI_ENABLE_REG
6B87Ch	MIPI_HS_READ_TRANSFER_COUNT
6B880h	MIPIC_DPHY_PARAM_REG
6B884h	MIPIC_DBI_BW_CTRL_REG
6B888h	MIPIC_CLK_LANE_SWITCHING_TIME_CNT
6B88Ch	MIPIC_STOP_STATE_STALL
6B890h	MIPIC_INTR_STAT_REG_1
6B894h	MIPIC_INTR_EN_REG_1
6B898h	MIPIC_CLK_LANE_TIMING
6B89Ch	MIPIC_PLL_LOCK_COUNT

Address	Name
6B8A0h	MIPIC_DATA_LANE_POLARITY
6B8A4h	MIPIC_TLPX_TIME_COUNT
6B8C0h	MIPIC_PORT_CTRL
6B8C4h	MIPIC_STATUS
6B8C8h	MIPIC_AUTOPWG
6B8CCh	MIPIC_WR_DATA 0
6B8D0h	MIPIC_WR_DATA 1
6B8D4h	MIPIC_WR_DATA 2
6B8D8h	MIPIC_WR_DATA 3
6B8DCh	MIPIC_WR_DATA 4
6B8E0h	MIPIC_WR_DATA 5
6B8E4h	MIPIC_WR_DATA 6
6B8E8h	MIPIC_WR_DATA 7
6B8ECh	MIPIC_WR_COMMAND
6B8F0h	MIPIC_WR_DATA_CTRL
6B8F4h	MIPIC_EN_DLY_CNTR
6B8F8h	MIPIC_TRANS_HACTIVE
6B8FCh	MIPIC_TRANS_VACTIVE
6B900h	MIPIC_TRANS_VTOTAL
6B904h	MIPIC_CTRL
6B908h	MIPIC_TE_CTR
6B918h	MIPIC_RD_DATA_RETURN 0

Address	Name
6B91Ch	MIPIC_RD_DATA_RETURN 1
6B920h	MIPIC_RD_DATA_RETURN 2
6B924h	MIPIC_RD_DATA_RETURN 3
6B928h	MIPIC_RD_DATA_RETURN 4
6B92Ch	MIPIC_RD_DATA_RETURN 5
6B930h	MIPIC_RD_DATA_RETURN 6
6B934h	MIPIC_RD_DATA_RETURN 7
6B938h	MIPIC_RD_DATA_VALID
6B93Ch	MIPIC_CRC_CTL
6B940h	MIPIC_CRC_RESULT
6B944h	MIPIC Dual Link Buffer Control

DSC

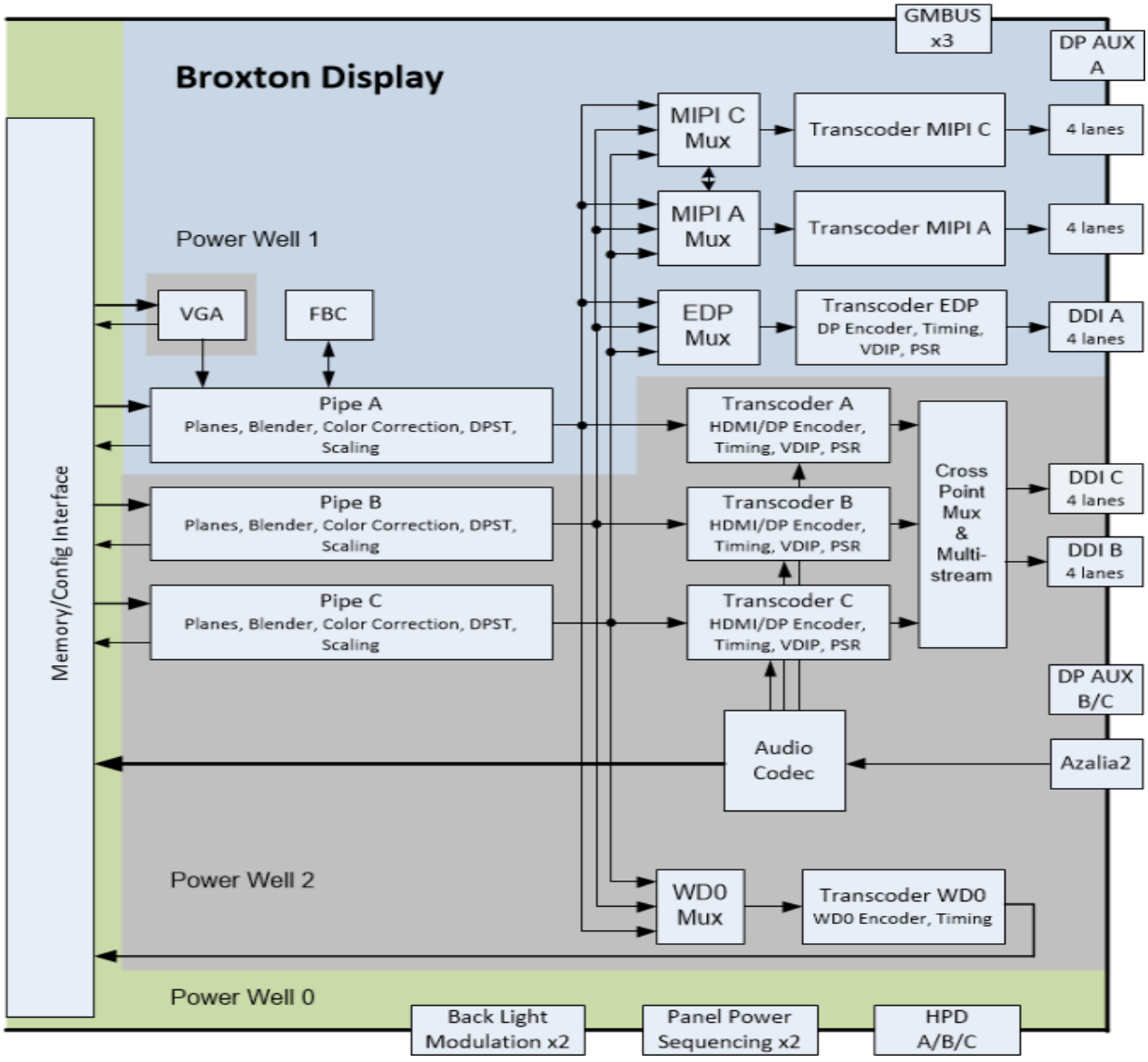
Note
These registers must all be written only as full 32 bit Dwords. Byte or word writes are not supported.

Address	Name
6B200h	DSCA_PICTURE_PARAMETER_SET_0
6B204h	DSCA_PICTURE_PARAMETER_SET_1
6B208h	DSCA_PICTURE_PARAMETER_SET_2
6B20Ch	DSCA_PICTURE_PARAMETER_SET_3
6B210h	DSCA_PICTURE_PARAMETER_SET_4
6B214h	DSCA_PICTURE_PARAMETER_SET_5
6B218h	DSCA_PICTURE_PARAMETER_SET_6
6B21Ch	DSCA_PICTURE_PARAMETER_SET_7
6B220h	DSCA_PICTURE_PARAMETER_SET_8
6B224h	DSCA_PICTURE_PARAMETER_SET_9
6B228h	DSCA_PICTURE_PARAMETER_SET_10
6B22Ch	DSCA_PICTURE_PARAMETER_SET_11
6B230h	DSCA_RC_BUF_THRESH_0
6B238h	DSCA_RC_BUF_THRESH_1
6B240h	DSCA_RC_RANGE_PARAMETERS_0
6B248h	DSCA_RC_RANGE_PARAMETERS_1
6B250h	DSCA_RC_RANGE_PARAMETERS_2
6B258h	DSCA_RC_RANGE_PARAMETERS_3
6B260h	DSCA_PICTURE_PARAMETER_SET_12

Address	Name
6B264h	DSCA_PICTURE_PARAMETER_SET_13
6B268h	DSCA_PICTURE_PARAMETER_SET_14
6B26Ch	DSCA_PICTURE_PARAMETER_SET_15
6B270h	DSCA_PICTURE_PARAMETER_SET_16
6B280h	Reserved
6B284h	DSC_CRC_CTL_A
6B288h	DSC_CRC_RES_A
6B28Ch	Reserved
6B290h	Reserved
6B294h	Reserved
6B298h	Reserved
6B29Ch	Reserved
6B2A0h	Reserved
6B2A4h	Reserved
6B2A8h	Reserved
6BA00h	DSCC_PICTURE_PARAMETER_SET_0
6BA04h	DSCC_PICTURE_PARAMETER_SET_1
6BA08h	DSCC_PICTURE_PARAMETER_SET_2
6BA0Ch	DSCC_PICTURE_PARAMETER_SET_3
6BA10h	DSCC_PICTURE_PARAMETER_SET_4
6BA14h	DSCC_PICTURE_PARAMETER_SET_5
6BA18h	DSCC_PICTURE_PARAMETER_SET_6
6BA1Ch	DSCC_PICTURE_PARAMETER_SET_7
6BA20h	DSCC_PICTURE_PARAMETER_SET_8
6BA24h	DSCC_PICTURE_PARAMETER_SET_9

Address	Name
6BA28h	DSCC_PICTURE_PARAMETER_SET_10
6BA2Ch	DSCC_PICTURE_PARAMETER_SET_11
6BA30h	DSCC_RC_BUF_THRESH_0
6BA38h	DSCC_RC_BUF_THRESH_1
6BA40h	DSCC_RC_RANGE_PARAMETERS_0
6BA48h	DSCC_RC_RANGE_PARAMETERS_1
6BA50h	DSCC_RC_RANGE_PARAMETERS_2
6BA58h	DSCC_RC_RANGE_PARAMETERS_3
6BA60h	DSCC_PICTURE_PARAMETER_SET_12
6BA64h	DSCC_PICTURE_PARAMETER_SET_13
6BA68h	DSCC_PICTURE_PARAMETER_SET_14
6BA6Ch	DSCC_PICTURE_PARAMETER_SET_15
6BA70h	DSCC_PICTURE_PARAMETER_SET_16
6BA80h	Reserved
6BA84h	DSC_CRC_CTL_C
6BA88h	DSC_CRC_RES_C
6BA8Ch	Reserved
6BA90h	Reserved
6BA94h	Reserved
6BA98h	Reserved
6BA9Ch	Reserved
6BAA0h	Reserved
6BAA4h	Reserved
6BAA8h	Reserved

Broxton Display Connections

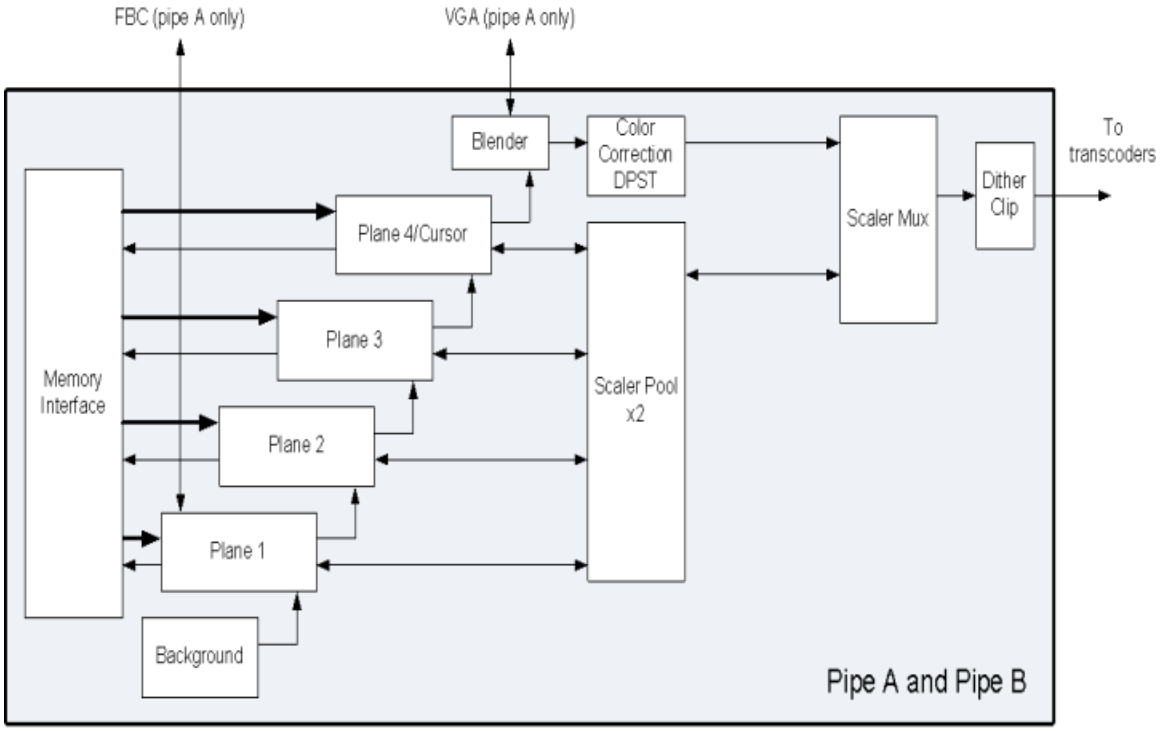


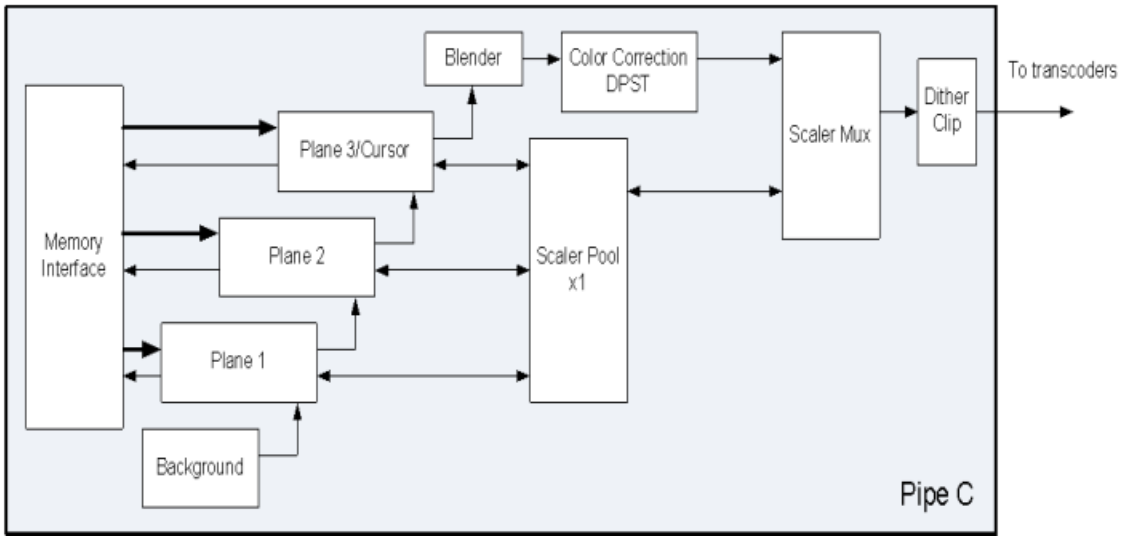
The front end of the display contains the pipes. There are three instances which are referred to as Pipe A, Pipe B, and Pipe C.

The pipes connect to the transcoders. There are seven instances which are referred to as Transcoder A, Transcoder B, Transcoder C, Transcoder EDP, Transcoder WD0, Transcoder MIPIA, and Transcoder MIPIC.

The transcoders connect to the DDIs. There are three instances which are referred to as DDI A, DDI B, and DDI C.

Display Pipes





The display pipes contain the planes, blending, color correction, DPST, scaling, dithering, and clipping.

Pipe A and Pipe B have four planes and a cursor. Each plane can be used as a sprite, primary, or overlay.

Pipe C has three planes and a cursor. Each plane can be used as a sprite, primary, or overlay.

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

The plane blending follows a fixed Z-order. Plane 1 is the bottom most plane and higher numbered planes stack on top of it.

The cursor and top most plane are mutually exclusive and cannot be both enabled at the same time.

Pipe A and Pipe B each have two pipe scalers. Pipe C has one pipe scaler.

Each pipe scaler can be assigned to scale an individual plane or scale the blended output.

Display Transcoders

The display transcoders contain the timing generators and port encoders.

Transcoders A, B, and C also contain Audio/Video mixers, Video Data Island Packet mixers, and Panel Self Refresh controllers.

Transcoder EDP also contains a Video Data Island Packet mixer and Panel Self Refresh controllers. It does not support HDMI, DVI, or Audio.

Transcoder WD0 only supports display capture and write back to memory.

Transcoders MIPIA and MIPIC only supports MIPI DSI.

Audio

The Azalia interface provides data to the audio codec.

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

The audio codec connects to memory write back for wireless audio.

DDIs

The DDIs contain the DisplayPort transport control and other port logic to interface to the DDI physical pins.

DDI A, DDI B, and DDI C support lane reversal where the internal lane to package lane mapping is swapped.

DDI A does not support DisplayPort multistream.

DisplayPort A, B, and C Lane Mapping:

Package Pin	Non-Reversed	Reversed
DDI 3	Main Link Lane 3	Main Link Lane 0
DDI 2	Main Link Lane 2	Main Link Lane 1
DDI 1	Main Link Lane 1	Main Link Lane 2
DDI 0	Main Link Lane 0	Main Link Lane 3

HDMI/DVI TMDS Lane Mapping:

Package Pin	Non-Reversed	Reversed
DDI 3	TMDS Clock	TMDS Data2
DDI 2	TMDS Data0	TMDS Data1

Package Pin	Non-Reversed	Reversed
DDI 1	TMDS Data1	TMDS Data0
DDI 0	TMDS Data2	TMDS Clock

DDI Equivalent Names:

DDI Name	Equivalent Names
DDI A	DDI-A, DDIA, Port A, eDP
DDI B	DDI-B, DDIB, Port B
DDI C	DDI-C, DDIC, Port C

Broxton 0 and Broxton 1 Board Mapping:

DDI Name in Display Engine	Display PHY Name	Die Bump Name	Board Connection
DDI A main link	Port 0 of the single port PHY	DDI-2	eDP
DDI B main link	Port 0 of the dual port PHY	DDI-0	DP/HDMI
DDI C main link	Port 1 of the dual port PHY	Not connected (DDI-1)	Not connected

Broxton P Board Mapping:

DDI Name in Display Engine	Display PHY Name	Die Bump Name	Board Connection
DDI A main link	Port 0 of the single port PHY	EDP	eDP
DDI B main link	Port 0 of the dual port PHY	DDI-0	DP/HDMI
DDI C main link	Port 1 of the dual port PHY	DDI-1	DP/HDMI

Pipe to Transcoder to DDI Mappings

Twin modes are not supported.

Any pipe can drive any single DDI.

With DisplayPort multistream it is possible to have multiple pipes driving a single DDI. DDI B and DDI C support multistream. DDI A does not support multistream.

Any pipe can connect to any transcoder, but not more than one simultaneously.

Transcoder A is tied to Pipe A.

Transcoder B is tied to Pipe B.

Transcoder C is tied to Pipe C.

Transcoders EDP, WD0, MIPIA, and MIPIC can connect to Pipe A, Pipe B, or Pipe C, but only one at a time.

Transcoders A, B, and C can connect to DDI B or DDI C, but only one at a time.

Transcoder EDP can connect only to DDI A.

Transcoder WD0 does not connect to any DDI. Transcoder WD0 output only goes to memory write back.

Transcoders MIPIA and MIPIC does not connect to any DDI. Transcoders MIPIA and MIPIC output only goes to MIPI.

DDI A can connect only to Transcoder EDP. DDI A does not support DisplayPort multistream.

DDI B can connect to Transcoder A, B, or C, individually or simultaneously if DisplayPort multistream is used.

DDI C can connect to Transcoder A, B, or C, individually or simultaneously if DisplayPort multistream is used.

Mode Set

Broxton Sequences to Initialize Display

These sequences are used to initialize the display engine before any display engine functions can be enabled.

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. Disable PCH Reset Handshake
    a. Clear NDE_RSTWRN_OPT RST PCH Handshake En to 0b.
2. Enable Power Well 1 (PG1)
    a. Poll for FUSE_STATUS Fuse PG0 Distribution Status == 1b.
        - Timeout and fail after 5 µs.
    b. Set PWR_WELL_CTL Power Well 1 Request to 1b.
    c. Poll for PWR_WELL_CTL Power Well 1 State == 1b.
        - Timeout and fail after 10 µs.
    d. Poll for FUSE_STATUS Fuse PG1 Distribution Status == 1b.
        - Timeout and fail after 5 µs.
3. Enable CD clock following the Sequences for Changing CD Clock Frequency
4. Enable DBUF
    a. Set DBUF_CTL DBUF Power Request to 1b.
    b. Poll for DBUF_CTL DBUF Power State == 1b.
        - Timeout and fail after 10 µs.
5. If DDIA/EDP, DDIB, or DDIC will be used, follow DDI Buffer DDI PHY Initialization Sequence to initialize DDI PHY
6. If MIPI will be used, follow Mode Set Sequences for MIPI to initialize MIPI PHY

7.

Workaround
If MIPI will not be used, power down DSI regulator to save power. a. Set P_CR_GT_DISP_PWRON_0_2_0_GTTMMADR MIPIO_RST_CTRL to 0x1 to bring MIPI IO out of reset b. Write 0x160020 = 0x00000001 (DSI_CFG: STAP_SELECT 0x1) c. Write 0x160054 = 0x00000001 (DSI_TXCNTRL: HS_IO_CONTROL_SELECT 0x1) d. Set P_CR_GT_DISP_PWRON_0_2_0_GTTMMADR MIPIO_RST_CTRL to 0x0 to put MIPI IO back into reset

Un-initialize Sequence

1. Disable all display engine functions using the full mode set disable sequence on all pipes, transcoders, ports, planes, and power well 2 (PG2).
2. Clear both PHY_CTL_FAMILY_EDP and PHY_CTL_FAMILY_DDI Common Reset to 0b (Disable)
3. Clear P_CR_GT_DISP_PWRON_0_2_0_GTTMMADR to 0x0
4. Disable DBUF
    a. Clear DBUF_CTL DBUF Power Request to 0b.
    b. Poll for DBUF_CTL DBUF Power State == 0b.
        - Timeout and fail after 10 µs.
5. Disable CD clock following the Sequences for Changing CD Clock Frequency
6. Disable Power Well 1 (PG1)
    a. Clear PWR_WELL_CTL Power Well 1 Request to 0b.
    b. Wait for 10us. Do not poll for the power well to disable. Other clients may be keeping it enabled.

Sequences for MIPI

This topic describes the sequences for MIPI enable, disable, commands, and DBI frames.

Programming Note	
Context:	DBI Mode
(BXT:*:A) In DBI mode there must always be at least one plane enabled on the pipe attached to MIPI.	

Enable Sequence

Display must already be initialized.

1. **If using pipe B or pipe C - Enable Power Well 2**
    a. Enable PWR_WELL_CTL Power Well 2 Request

b. Wait for PWR_WELL_CTL Power Well 2 State == Enabled, timeout after 20 us
c. Wait for FUSE_STATUS FUSE PG2 Distribution Status == Done, timeout after 1 us

2. **If using hardware panel power control - Enable Panel Power**
   a. Enable panel power sequencing
   b. Wait for panel power sequencing to reach the enabled state
3. **If DSI PLL not already enabled - configure and enable MIPI clocks**
   a. Configure dividers in MIPI_CLOCK_CTL
   b. Follow MIPI DSI PLL Enabling Sequence.
4. **Exit Low Power Mode**
   a. Set P_CR_GT_DISP_PWRON_0_2_0_GTTMMADR MIPIO_RST_CTRL to 0x1 to bring MIPI IO out of reset and start RCOMP
   b. BXT A0: Set DSI_RCOMP_CFG1 mipA_dphy_defeature_en and mipC_dphy_defeature_en to 0x1 and configure mip<A,C>_lane_count to the number of lanes that will be used
   c.

Workaround
Power up DSI regulator i. Write 0x160020 = 0x00000001 (DSI_CFG: STAP_SELECT 0x1) ii. Write 0x160054 = 0x00000000 (DSI_TXCNTRL: HS_IO_CONTROL_SELECT 0x0)

   d. If currently in Low Power mode, follow the Exit Low Power Mode sequence
   e. Enable transparent latch - Set MIPI_PORT_CTRL LPOUTPUT_HOLD to 0x1
5. **If needed - Send Manufacturer DCS Commands to Initialize DSI Controller**
   a. Set MIPI_DEVICE_READY_REG DEVICE_READY to 0x1
   b. Follow Command Sequence for Generic/Manufacturer Writes/Reads
6. **Enable Planes and Pipe Scaler**
   a. Configure and enable planes (VGA or hires). This can be done later if desired.
   b. If VGA, clear VGA I/O register SR01 bit 5
   c. Enable pipe scaler if needed (must be enabled for VGA)
   d. Configure pipe settings
7. **Prepare Port**
   a. Clear MIPI_DEVICE_READY_REG DEVICE_READY to 0x0
   b. Configure MIPIA_DPHY_PARAM_REG and MIPIC_DPHY_PARAM_REG (both must be configured regardless of single or dual link mode in order to allow ULPS to be entered later)
   c. Configure MIPI_TRANS_HACTIVE and MIPI_TRANS_VACTIVE
   d. If DPI, configure MIPI_TRANS_VTOTAL, MIPI_DPI_RESOLUTION_REG, back porch, front porch, and sync padding registers
   e. Configure other timing parameters, timeouts, switch count, byteclk

f. Configure MIPIA_INIT_COUNT_REGISTER and MIPIC_INIT_COUNT_REGISTER (both must be configured regardless of single or dual link mode in order to allow ULPS to be entered later)
g. If DBI, configure MIPI_HS_LS_DBI_ENABLE_REG, MIPI_DBI_FIFO_THRTL_REG, and MIPI_DBI_BW_CTRL_REG
h. Configure MIPI_DSI_FUNC_PRG_REG format
i. Configure MIPI_CTRL Pipe Select
j. Configure MIPI_EOT_DISABLE_REGISTER CLOCKSTOP, depending on panel type.
k. BXT B0 onwards: Set MIPI_EOT_DISABLE_REGISTER DPHY_DEFEATURE_EN to 0x1.
l. If DPI, set MIPI_EOT_DISABLE_REGISTER DEFEATURE_DPI_FIFO_CTR to 0x1.
m. If DPI, set MIPI_VIDEO_MODE_FORMAT_REGISTER IP_TG_CONFIG to 0x1.
n. If DBI dual link, configure MIPI_DLBUFFER_CTRL
o. If DBI using tearing, configure MIPI_PORT_CTRL EFFECT
p. If DBI using tearing and a tear delay is needed, configure MIPI_TE_CTR and set MIPI_PORT_CTRL DELAY to 0x1
q. Set MIPI_DEVICE_READY_REG DEVICE_READY to 0x1
r. If DPI, set MIPI_DPI_CTRL_REG TURN_ON to 0x1
s. If DPI, wait for the SPL_PKT_SENT_INTERRUPT. Timeout after 1ms.
t. Configure MIPI_MAX_RETURN_PKT_SIZE_REGISTER

8. **Send Display On MCS Commands**
   a. If DBI using tearing, send the set_tear_on or set_tear_scanline command
   b. If DBI using backlight control in the panel, send the backlight enable command
   c. Send Display On Commands, depending on the panel type
9. **Enable Timing Generator**
   a. If DPI, set MIPI_PORT_CTRL EN to 0x1 to start the timing generator
10. **Enable Backlight**
    a. If DPI, wait for data on to backlight enable delay
    b. If DPI, enable panel backlight

Interrupt Usage

MIPI interrupt events are enabled by setting bits in MIPI_INTR_EN_REG and MIPI_INTR_EN_REG_1. When an enabled interrupt event occurs, it sets a sticky bit in MIPI_INTR_STAT_REG or MIPI_INTR_STAT_REG_1. The sticky bits are ORd together to create the consolidated interrupts for MIPI A and MIPI C in the DE Port Interrupt, which propagates to the internal graphics interrupt. To detect a MIPI event, software can either poll on the MIPI status bit or use the internal graphics interrupt. The sticky bit should be cleared by writing a 1 to it.

Command Sequence for DBI Writes/Reads

1. Poll for MIPI_WR_DATA_CTRL COMMAND_VALID = 0x0 to ensure no command is being processed.
2. Configure MIPI_WR_COMMAND with the command byte to be sent to the panel.
3. Configure MIPI_WR_DATA registers with the data bytes to be sent to the panel.
4. Configure MIPI_WR_DATA_CTL WR_BYTE_LEGTH and set COMMAND_VALID to 0x1 to execute the write command, hardware will deassert the bit when done.
5. If doing a read, poll for MIPI_RD_DATA_VALID READ_DATA_VALID == count of number of registers expected to be read.
6. If doing a read, get read data from MIPI_RD_DATA_RETURN registers.

Command Sequence for Generic/Manufacturer Writes/Reads

1. If long, write data bytes to MIPI_<HS,LP>_GEN_DATA_REG. If more than 4 bytes, continue to write data to the same register where it will be placed into a FIFO.
2. If short, configure MIPI_<HS,LP>_GEN_CTRL_REG WORD_COUNT with the 0-2 parameters.
3. If long, configure MIPI_<HS,LP>_GEN_CTRL_REG WORD_COUNT with the data byte count.
4. Configure MIPI_<HS,LP>_GEN_CTRL_REG DATA_TYPE
5. If doing a read, wait for the GEN_READ_DATA_AVAIL interrupt. Timeout after 1ms.
6. If doing a read, get read data from MIPI_<HS,LP>_GEN_DATA_REGISTERS. If more than 4 bytes, continue to read the same register until all data is received.

DBI Frame Sequence

MIPI Enable Sequence must be completed first.

Repeat this sequence for each frame to be sent.

1. Poll for MIPI_WR_DATA_CTRL COMMAND_VALID == 0x0 to ensure no command is being processed.
2. Configure MIPI_WR_COMMAND with the command to be sent to the panel.
3. Set MIPI_WR_DATA_CTL COMMAND_VALID to 0x1 to execute the write command. Hardware will deassert the bit when done.
4. If using tearing command, set MIPI_DEVICE_READY_REG BUS_POSSESION to 0x1 to give bus possession to the panel. If using the tearing pin, the bus possesion does not have to be given.
5. If using tearing command, wait until the start of vertical blank, then clear MIPI_DEVICE_READY_REG BUS_POSSESION to 0x0.

Disable Sequence

1. **Disable Backlight**
    a. Disable panel backlight

2. **If DPI, Send DPI Shutdown Packet**
    a. Set MIPI_DPI_CTRL_REG SHUT_DOWN to 0x1
    b. Wait for SPL_PKT_SENT_INTERRUPT. Timeout after 1ms.
3. **Disable Planes**
    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 pipe scaler
4. **Disable Port**
    a. If DBI, poll for MIPI_WR_DATA_CTRL COMMAND_VALID == 0x0 to ensure command is done
    b. If DBI, poll for MIPI_GEN_FIFO_STAT_REGISTER DPI_FIFO_EMPTY and DBI_FIFO_EMPTY == 0x1 to ensure data is done
        - If dual link mode, only poll on MIPI C FIFO empties.
    c. If DPI, clear MIPI_PORT_CTRL EN to 0x0
5. **Send Manufacturer DCS Commands**
    a. If DBI using tearing, send set_tear_off command
    b. Send Display Off Commands, depending on the panel type
        - Note: If LP commands need to be sent as part of the Display Off commands, and MIPI is NOT running in clock stopping mode (MIPI_EOT_DISABLE_REGISTER CLOCKSTOP==0), then clock stopping mode needs to be enabled before the LP commands can be sent.
            i. Clear MIPI_DEVICE_READY_REG DEVICE_READY to 0x0
            ii. Set MIPI_EOT_DISABLE_REGISTER CLOCKSTOP to 0x1
            iii. Set MIPI_DEVICE_READY_REG DEVICE_READY to 0x1
            iv. Send LP commands as needed
    c. Poll for MIPI_GEN_FIFO_STAT_REGISTER HS_CTRL_FIFO_EMPTY, HS_DATA_FIFO_EMPTY, LP_CTRL_FIFO_EMPTY, and LP_DATA_FIFO_EMPTY == 0x1 to ensure all commands have been sent
6. **If using hardware panel power control - Disable Panel Power**
    a. Disable panel power sequencing
    b. Wait for power down delay and power cycle delay
7. **If all MIPI ports disabled and low power mode requested - Enter Low Power Mode**
    a. Follow the Enter Low Power Mode sequence
    b.

Workaround
Power down DSI regulator to save power.

Workaround
a. Write 0x160020 = 0x00000001 (DSI_CFG: STAP_SELECT 0x1) b. Write 0x160054 = 0x00000001 (DSI_TXCNTRL: HS_IO_CONTROL_SELECT 0x1)

c. Clear P_CR_GT_DISP_PWRON_0_2_0_GTTMMADR MIPIO_RST_CTRL to 0x0 to put MIPI IO into reset and stop RCOMP

d. Follow the MIPI DSI PLL Disabling Sequence

Enter Low Power Mode

1. Set MIPIA_DEVICE_READY_REG and MIPIC_DEVICE_READY_REG DEVICE_READY to 0x1
2. Enable transparent latch - Set MIPIA_PORT_CTRL LPOUTPUT_HOLD to 0x1
3. Wait for 10us
4. Enter ULPS - Set MIPIA_DEVICE_READY_REG and MIPIC_DEVICE_READY_REG DEVICE_READY to 0x1 and ULPS_STATE to 0x2
5. Poll for MIPIA_PORT_CTL AFE_LATCHOUT == 0x0
6. Disable transparent latch - Set MIPIA_PORT_CTRL LPOUTPUT_HOLD to 0x0

- Device ready must be kept at 0x1 while in low power mode

Exit Low Power Mode

1. Wait for at least 2.5ms while in the ULPS state
2. Enter ULPS - Set MIPIA_DEVICE_READY_REG and MIPIC_DEVICE_READY_REG DEVICE_READY to 0x1 and ULPS_STATE to 0x2
3. Wait for 2us for ULPS to complete
4. Enable transparent latch - Set MIPIA_PORT_CTRL LPOUTPUT_HOLD to 0x1
5. Exit ULPS - Set MIPIA_DEVICE_READY_REG and MIPIC_DEVICE_READY_REG DEVICE_READY to 0x1 and ULPS_STATE to 0x1
6. Wait for 1ms
7. Enter normal mode - Set MIPIA_DEVICE_READY_REG and MIPIC_DEVICE_READY_REG DEVICE_READY to 0x1 and ULPS_STATE to 0x0

Overall Flow

1. Initialization Sequence (ULPS exited by default, latch out disabled)
2. Enable PLL with dividers setup for any future use for both MIPI ports (can't change dividers after PLL is enabled)
3. Exit Low Power Mode if not already exited from boot

4. Enable and disable MIPI ports as needed (enable sequence and disable sequence, while keeping PLL enabled and not entering Low Power Mode)
5. If all MIPI ports disabled and low power requested, Enter Low Power Mode and Disable PLL
6. If MIPI needs to be enabled again, go to step 2. Else, display can be un-initialized or configured for DC9.

Sequences for DisplayPort

This topic describes how to enable and disable DisplayPort.

Enable Sequence

Display must already be initialized

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

1. **If not PipeA+DDIA - Enable Power Well 2**
   a. Enable PWR_WELL_CTL Power Well 2 Request
   b. Wait for PWR_WELL_CTL Power Well 2 State = Enabled, timeout after 20 us
   c. Wait for FUSE_STATUS FUSE PG2 Distribution Status = Done, timeout after 1 us
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. **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
5. **Enable and Train DisplayPort**
   a. Configure and enable DP_TP_CTL with link training pattern 1 selected
   b. Configure voltage swing and related IO settings. Refer to DDI Buffer section.
   c. Configure and enable DDI_BUF_CTL
   d. Follow DisplayPort specification training sequence (see notes for failure handling)
   e. 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)
   f. Set DP_TP_CTL link training to Normal.
6. **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
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. Enable panel fitter if needed (must be enabled for VGA)
f. Configure transcoder timings, M/N/TU/VC payload size, and other pipe and transcoder settings
g. Configure and enable TRANS_DDI_FUNC_CTL
h. If DisplayPort multistream - Enable pipe VC payload allocation in TRANS_DDI_FUNC_CTL
i. If DisplayPort multistream - Wait for ACT sent status in DP_TP_STATUS and receiver DPCD (timeout after >410us)
j. Configure and enable TRANS_CONF
k. 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. If PLL frequency needs to change, Follow Disable Sequence for DisplayPort to Disable PLL, then follow Enable Sequence for DisplayPort to Enable PLL, using the new frequency settings
3. Follow 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

See TRANS_DDI_FUNC_CTL Port Sync Mode Enable for restrictions.

1. Follow the enable sequence for the DisplayPort slave, but skip the step that sets DP_TP_CTL link training to Normal (stay in Idle Pattern).
    - Set TRANS_DDI_FUNC_CTL Port Sync Mode Master Select and Port Sync Mode Enable when configuring and enabling TRANS_DDI_FUNC_CTL.

2. Follow the enable sequence for the DisplayPort master, but skip the step that sets DP_TP_CTL link training to Normal (stay in Idle Pattern).
3. Set DisplayPort slave DP_TP_CTL link training to Normal.
4. Wait 200 uS.
5. Set DisplayPort master 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.

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. **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. Wait for off status in TRANS_CONF, timeout after two frame times
    e. If DisplayPort multistream - use AUX to program receiver VC Payload ID table to delete stream
    f. If done with this VC payload
        i. Disable VC payload allocation in TRANS_DDI_FUNC_CTL
        ii. Wait for ACT sent status in DP_TP_STATUS and receiver DPCD
    g. Disable TRANS_DDI_FUNC_CTL with DDI_Select set to None
    h. Disable panel fitter
    i. 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. Disable DDI_BUF_CTL
    b. Disable DP_TP_CTL (do not set port to idle when disabling)
    c. BXT: Wait 16 us for buffers to return to idle

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. 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. If no required resource is in the power well - Disable PWR_WELL_CTL Power Well 2 Request

Disabling DisplayPort Sync Mode

1. Follow the disable sequence for the DisplayPort slave.
2. Follow the disable sequence for the DisplayPort master.

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**

a. Enable PWR_WELL_CTL Power Well 2 Request b. Wait for PWR_WELL_CTL Power Well 2 State = Enabled, timeout after 20 us c. Wait for FUSE_STATUS FUSE PG2 Distribution Status = Done, timeout after 1 us

2. **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
3. **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
4. **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

5. **Enable Port**
   a. Configure voltage swing and related IO settings. Refer to the DDI Buffer section.
   b. Configure and enable DDI_BUF_CTL

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
   g. Configure Transcoder Clock Select to direct no clock to the transcoder
2. **Disable Port**
   a. Disable DDI_BUF_CTL
   b. BXT: Wait 16 us for buffers to return to idle
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 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. **Disable Power Wells**

a. If no required resource is in the power well - Disable PWR_WELL_CTL Power Well 2 Request

Sequences for Display C5 and C6

Display C5 (DC5) is a power saving state where hardware dynamically disables power well 1 and the CDCLK PLL and saves the associated registers.

DC5 can be entered when software allows it, power well 2 is disabled, and hardware detects that all pipes are disabled or pipe A is enabled with PSR active.

Display C6 (DC6) is a deeper power saving state where hardware dynamically disables power well 0 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 cannot be used if the backlight is being driven from the display utility pin.

Core CPUs support DC5 and DC6.

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

Atom CPUs support DC5 and not DC6. They also support DC9, which has a separate sequence.

The context save and restore program is reset on DC9, cold boot, warm reset, PCI function level reset, and hibernate/suspend.

The MIPI DSI related registers are not saved and restored by hardware.

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 have power well 2 disabled, following the appropriate mode set disable sequences for any ports using power well 2. This can be done earlier if desired.
3. Set display register 0x45520 bit 1 to 1b. It does 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
Do not switch between "Enable up to DC5" and "Enable up to DC6" without moving to "Disable" and reloading the CSR program in between. Disable DC5/DC6 during mode set and re-enable after the mode set programming is completed. 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.	

Programming Note	
Context:	Display C5 and C6
For atom CPUs, in step 3 also set 0x45520 bit 0 to 1b. It does not need to be cleared at any time.	

Sequence to Disallow DC5 and DC6

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

DMC Firmware Package

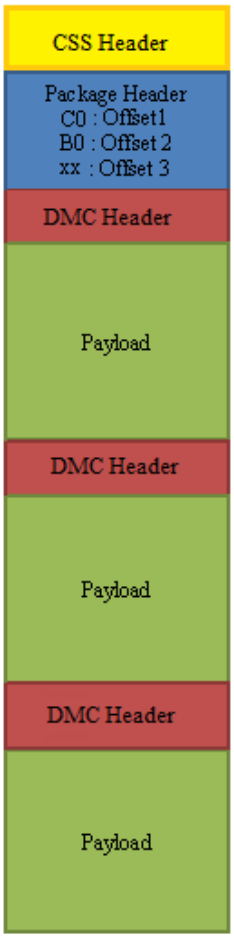
Display Micro-Controller firmware package includes all the firmwares that are required for different steppings of the product. The stepping dependent firmwares are all packaged and released as a single binary package. The package contains the CSS header, followed by the package header and the actual 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 1

CSS Header	1.0
Package Header	1
DMC Header	1

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

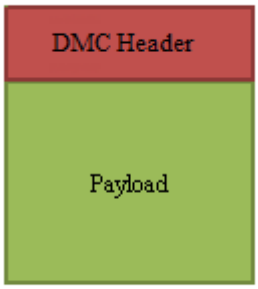
```
typedef struct _PackageHeader {
      uint8_t      headerLen;                           // DMC package header length
in dwords
      uint8_t      headerVer;                           // 0x01
```

```
      uint8_t      reserved[10];                    // Reserved
      uint32_t     numEntries;                      // Number of valid entries in
the FWInfo array below
      struct _FWInfo_ {
            uint16_t     reserved1;                 // Reserved
            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
            uint32_t     reserved2;                 // Reserved
      } FWInfo[20];
} 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]>
typedef struct _DMCHeader {
     uint32_t    reserved;              // 0x40403E3E
     uint8_t     headerLen;             // DMC binary header length in bytes
     uint8_t     headerVer;             // 0x01
     uint16_t    dmccVer;               // Reserved
     uint32_t    project;               // Major, Minor
```

```
    uint32_t    fwSize;                 // Firmware program size (excluding header)
    in dwords
    uint32_t    fwVersion;              // Major Minor version
    uint32_t    mmioCount;              // Number of valid MMIO cycles present in
    the MMIO address and data arrays below.
    uint32_t    mmioaddr[8];            // MMIO address
    uint32_t    mmiodata[8];            // MMIO data
    unint8_t    dfile[32];              // Reserved
    uint32_t    reserved1[2];           // Reserved
} DMCHeader;
```

Sequences for Display C9

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

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.

Note
Display C9 is supported on Broxton.

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 well 2 (PG2).
3. Disable and mask all graphics interrupts in display.
    - If hotplug detection is needed during DC9, keep hotplug detection and its associated interrupts enabled and unmasked, and additionally enable and unmask the PCU Interrupt for DDI S0ix HPD. See Hotplug Detection During DC9.
4. Save state of MMIO display registers.
    - The exact registers depend on software policy.
    - Hardware will save and restore the PCI Config, DGunit registers, and hotplug related 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.
4. Enable and unmask graphics interrupts as needed.

Hotplug Detection During DC9

Hotplug Detection (HPD) during DC9 requires special handling since the display engine hotplug detection logic is powered off.

If enabled by a BIOS configured register (DISPLAY_HPD_CTL at offset 1094h from base pmc_dev.BAR_HIGH,BAR), PMC will take over HPD when display engine is powered off, and detect a hotplug rising edge, falling edge, or any edge. Driver should determine whether HPD during DC9 is desired through BIOS VBT or some similar mechanism for OEM selection.

During DC9 hardware will dynamically transition between display engine powered on and off, so hotplug may be detected by PMC or display engine, and hardware will not reliably differentiate between short and long pulse hotplug events. If any kind of hotplug is detected, driver should use the HPD live state (Port Interrupt DDI Hotplug ISR) together with an AUX channel or GMBUS query to the receiver to determine if it was a connect, disconnect, or HPD_IRQ.

1. Display driver follows the Sequence to Allow DC9 with display engine hotplug detection logic enabled and the hotplug and DDI DC9 HPD interrupts enabled and unmasked.
2. When display is powered down, PMC wakes the system if it detects the selected edge type on the hotplug line.
3. The power controller unit forwards a DDI DC9 HPD interrupt to the display engine PCU Interrupt.
4. If the DDI DC9 HPD interrupt is enabled and unmasked, it triggers a graphics device interrupt. If it is disabled or masked, no interrupt happens and the system can re-enter DC9.
5. Display driver services the interrupt and determines it it was a connect, disconnect, or HPD_IRQ.
6. The display driver and operating system interpret the result and decide whether to disallow DC9 and enable a display output or do nothing.

Gen9 Display 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 Memory Read Bandwidth, and Maximum Watermark.

Maximum Pipe Pixel Rate

The display resolution must fit within the maximum pixel rate output from the pipe. Make sure that the display pipe is able to feed pixels at a rate required to support the desired resolution.

For each enabled plane on the pipe {

If plane scaling enabled {

Horizontal down scale amount = Maximum[1, plane horizontal size / scaler horizontal window size]

Vertical down scale amount = Maximum[1, plane vertical size / scaler vertical window size]

Plane down scale amount = Horizontal down scale amount * Vertical down scale amount

Plane Ratio = 1 / Plane down scale amount

}

Else {

Plane Ratio = 1

}

If plane source pixel format is 64 bits per pixel {

Plane Ratio = Plane Ratio * 8/9

}

}

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

If pipe scaling is enabled {

Horizontal down scale amount = Maximum[1, pipe horizontal source size / scaler horizontal window size]

Vertical down scale amount = Maximum[1, pipe vertical source size / scaler vertical window size]

Note: The progressive fetch - interlace display mode is equivalent to a 2.0 vertical down scale

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

Pipe Ratio = Pipe Ratio / Pipe down scale amount

}

Pipe maximum pixel rate = CDCLK frequency * Pipe Ratio // See the display clocks section for the supported CDCLK frequencies.

Maximum Port Link Rate

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

Port Type	Maximum Link Bit Rate
eDP/DP	HBR2 5.4 GHz
HDMI	3 GHz
DVI	1.65 GHz
MIPI DSI	1.066 GHz

Maximum Memory Read Bandwidth

The display resolution must not exceed the available system memory bandwidth, considering factors like thermal throttling and bandwidth available for other memory clients.

For each pipe {

For each plane enabled on the pipe { // cursor can be ignored

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

Total display bandwidth MB/s = Total display bandwidth + Plane bandwidth

}

}

Raw system memory bandwidth = (# of memory channels * memory frequency * 8 bytes)

If Total display bandwidth > system memory bandwidth available for display {Bandwidth is exceeded}

Programming Note	
Context:	Gen9 Display Resolution Support
Do not use more than 60% of raw system memory bandwidth for display.	

Maximum Watermark

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

Display Resolution Capabilities

These are examples of common resolutions that meet all the resolution restrictions for up to 3 simultaneous displays, 4 primary or sprite planes with 32bpp pixel format, and 1 cursor, with no panel fitter down scaling.

Attribute	CD 624 MHz	CD 576 MHz	CD 384 MHz	CD 288 MHz	CD 144 MHz
eDP/DP x4 single stream	4096x2304 60Hz 24Bpp[4]	4096x2160 60Hz 30Bpp[4]	2880x1620 60Hz 30Bpp[2]	4096x2160 30Hz 30Bpp[2] 2560x1600 60Hz 30Bpp[1]	1920x1080 60Hz 30Bpp[1]
HDMI	Same as entry to the right	Same as entry to the right	4Kx2K 24-30Hz 24Bpp[2]	1080p 59.94-60Hz 24Bpp[1]	720p 59.94-60Hz 24Bpp[1]
DVI	Same as entry to the right	Same as entry to the right	Same as entry to the right	1920x1200 60Hz 24Bpp[1]	1920x1080 60Hz 24Bpp[1]
MIPI DSI single link 1:1 compression	Same as entry to the right	Same as entry to the right	Same as entry to the right	1920x1200 60Hz 24Bpp[1]	1920x1080 60Hz 24Bpp[1]
MIPI DSI single link 2:1 compression	Same as entry to the right	Same as entry to the right	2880x1620 60Hz 24Bpp[2]	2560x1600 60Hz 24Bpp[1]	Same as entry above
MIPI DSI single link 3:1 compression	Same as entry to the right	3200x2000 60Hz 24Bpp[3]	Same as entry above	Same as entry above	Same as entry above
MIPI DSI dual link 1:1 compression	Same as entry to the right	Same as entry to the right	2880x1620 60Hz 24Bpp[2]	2560x1600 60Hz 24Bpp[1]	1920x1080 60Hz 24Bpp[1]
MIPI DSI dual link 2:1 compression	4096x2304 60Hz 24Bpp[4]	4096x2160 60Hz 24Bpp[4]	Same as entry above	Same as entry above	Same as entry above
MIPI DSI dual link 3:1 compression	Same as entry above	Same as entry above	Same as entry above	Same as entry above	Same as entry above

Each entry is showing the highest common resolutions at that CD clock frequency step. Lower resolutions are also supported. Higher, less common, resolutions can also work, but need to be calculated individually.

eDP, DP, DVI, and MIPI DSI are calculated using CVT 1.2 RB1 blanking and pixel rate.

HDMI is calculated using HDMI specification blanking and pixel rate.

Bpp is referring to the port output bits per pixel.

[1]Requires at least single channel DDR3 1333 for 3 simultaneous displays

[2]Requires at least single channel DDR3 1600 for 3 simultaneous displays

[3]Requires at least dual channel DDR3 1333 for 3 simultaneous displays

[4]Requires at least dual channel DDR3 1600 for 3 simultaneous displays

Examples

Example pipe pixel rate:

Plane 1 enabled at 32bpp, plane 2 enabled at 16bpp, pipe scaling enabled and down scale amount 1.12, and CDCLK 450 MHz:

Plane 1 ratio = 1

Plane 2 ratio = 1

Pipe ratio = Minimum[1, 1] = 1

Pipe ratio = 1/1.12 = 0.89

Pipe maximum pixel rate = 450 MHz * 0.89 = 400.5 MHz

Example pipe pixel rate:

Plane 1 enabled at 64bpp and plane down scale amount 1.25, plane 2 enabled at 32bpp, no panel fitting enabled, and CDCLK 540 MHz:

Plane 1 ratio = 1/1.25 * 8/9 = 0.71

Plane 2 ratio = 1

Pipe ratio = Minimum[1, 0.71] = 0.71

Pipe maximum pixel rate = 540 MHz * 0.71 = 383.4 MHz

Example memory bandwidth:

System memory bandwidth available for display = 4000 MB/s

Pipe A - Plane 1 enabled at 32bpp, plane 2 enabled at 16bpp, scaling disabled, pixel rate 148.5 MHz

Pipe B - Plane 1 enabled at 32bpp, scaling disabled, pixel rate 148.5 MHz

Pipe C - Plane 1 enabled at 32bpp, scaling disabled, pixel rate 148.5 MHz

Pipe A - Plane 1 bandwidth = 148.5 * 4 bytes = 594 MB/s

Pipe A - Plane 2 bandwidth = 148.5 * 2 bytes = 297 MB/s

Pipe B - Plane 1 bandwidth = 148.5 * 4 bytes = 594 MB/s

Pipe C - Plane 1 bandwidth = 148.5 * 4 bytes = 594 MB/s

Total display bandwidth = 594 + 297 + 594 + 594 = 2079 MB/s

System memory bandwidth available for display not exceeded

Example memory bandwidth:

System memory bandwidth available for display = 4000 MB/s

Pipe A - Plane 1 enabled at 32bpp, plane 2 plane enabled at 32bpp, pipe scaling enabled and down scale amount 1.12, pixel rate 414.5 MHz

Pipe B - Plane 1 enabled at 32bpp, scaling disabled, pixel rate 414.5 MHz

Pipe C - Plane 1 enabled at 32bpp, scaling disabled, pixel rate 414.5 MHz

Pipe A - Plane 1 bandwidth = 414.5 * 4 bytes * 1.12 = 1863 MB/s

Pipe A - Plane 2 bandwidth = 414.5 * 4 bytes * 1.12 = 1863 MB/s

Pipe B - Plane 1 bandwidth = 414.5 * 4 bytes = 1658 MB/s

Pipe C - Plane 1 bandwidth = 414.5 * 4 bytes = 1658 MB/s

Total display bandwidth = 1863 + 1863 + 1658 + 1658 = 7042 MB/s

System memory bandwidth available for display **exceeded**

Clocks

Broxton Clocks

Registers

CDCLK_CTL

DE_PLL_CTL

DE_PLL_ENABLE

MIPI_CLOCK_CTL

DSI_PLL_CTL

DSI_PLL_ENABLE

PORT_PLL_PCS_0

PORT_PCS_DW12

PORT_PLL_EBB_0

PORT_PLL_EBB_4

PORT_PLL_0

PORT_PLL_1

PORT_PLL_2

PORT_PLL_3

PORT_PLL_6

PORT_PLL_8

PORT_PLL_9

PORT_PLL_10

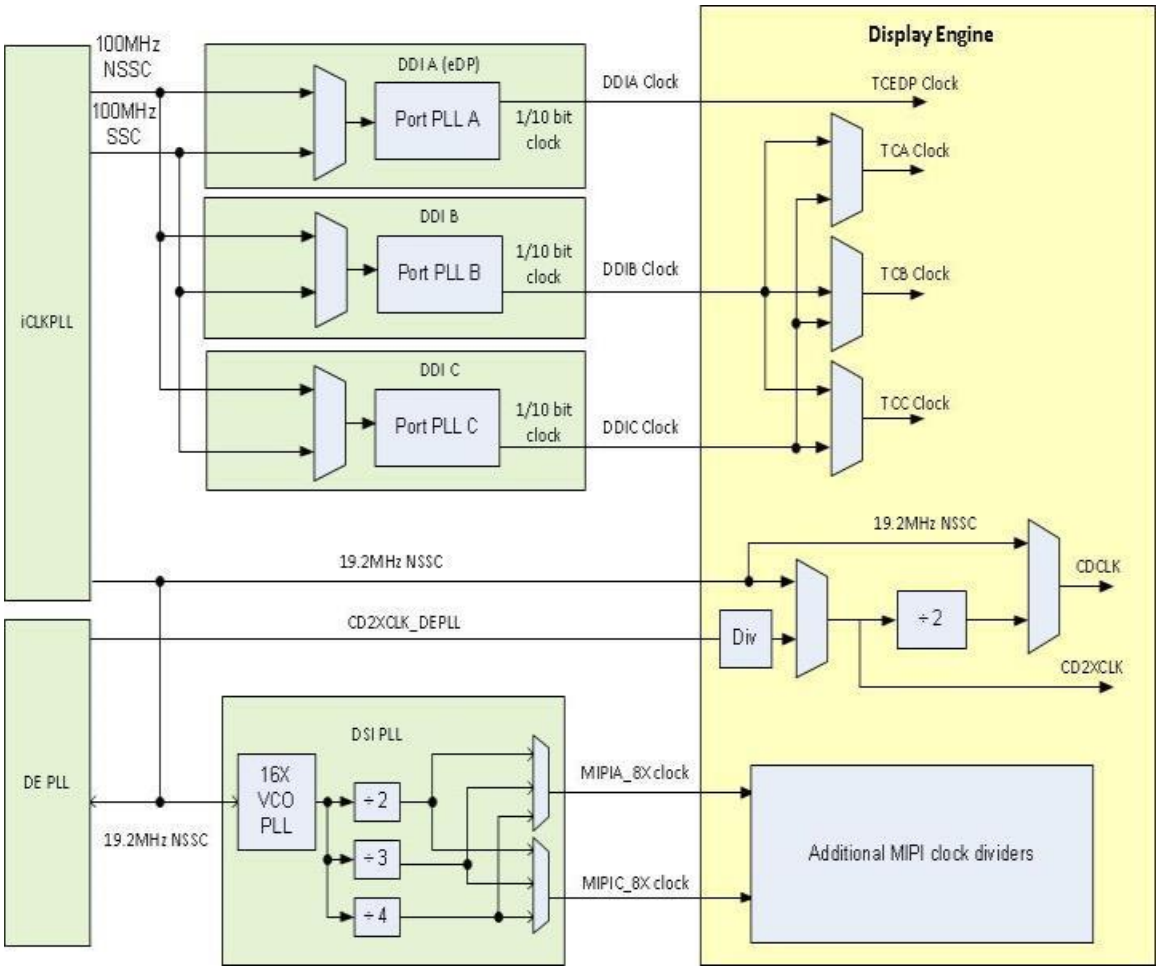
PORT_PLL_ENABLE

TRANS_CLK_SEL

TIMESTAMP_CTR

Overview of Display Clock Paths

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



Display Engine Clock Reference

There are three display engine clock references.

	100 MHz Non-SSC (NSSC) Reference	100 MHz SSC Reference	19.2 MHz Non-SSC Reference
Usage	Non-spread spectrum reference for the port DDI PLLs	Spread spectrum reference for the port DDI PLLs	Reference for the DE PLL and DSI PLL
Frequency	100 MHz	100 MHz	19.2 MHz
Default after reset	Enabled	Enabled	Enabled
Programming	Not programmable by display software.	Not programmable by display software.	Not programmable by display software.

Display Engine PLLs

	DE PLL	PORT PLL A, PORT PLL B, PORT PLL C	DSI PLL
Usage	Source for CD clock	Sources for DDI clocks	Source for MIPI clocks
Input	19.2 MHz Non-SSC reference	100 MHz SSC and Non-SSC references	19.2 MHz Non-SSC reference
Frequency	Programmable 1152, 1248 MHz	Programmable eDP link rates: 1.62, 2.16, 2.43, 2.7, 3.24, 4.32, 5.4 GHz SSC and Non-SSC DP link rates: 1.62, 2.7, 5.4 GHz SSC and Non-SSC HDMI/DVI link rates: 0.2 to 3.0 GHz Non-SSC* *2.233 - 2.4 GHz not supported	Programmable 8X frequency: 0.288 to 1.06 GHz Non-SSC
Default after reset	Disabled	Disabled	Disabled
Programming	Enable and frequency must be programmed by software when enabling and disabling any display. May be automatically enabled and disabled by hardware for some power states.	Enable and frequency must be programmed by software when enabling and disabling a HDMI/DVI or DisplayPort display. May be automatically enabled and disabled by hardware for some power states.	Enable and frequency must be programmed by software when enabling and disabling a MIPI display. May be automatically enabled and disabled by hardware for some power states.

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.

	DDI clocks A, B, C
Usage	DDI ports I/O bit clock and symbol/TMDS clock, source for transcoder clocks
Input	The associated Port PLL
Frequency	Port PLL frequency
Default after reset	Disabled
Programming	Not programmable

Transcoder Clocks

There is one Transcoder clock tied to each display transcoder, except Transcoder WD which uses only CD clock.

	Transcoder clock EDP	Transcoder clocks A, B, C	Transcoder clocks MIPI1 (MIPI A), MIPI2 (MIPI C)
Usage	Transcoder EDP symbol clock	Transcoders A, B, and C symbol/TMDS clocks	Transcoder MIPI clocks
Input	Always uses DDI A clock	Programmable selection between DDI clocks B, and C.	Always uses associated MIPI DSI output clock
Frequency	PLL output frequency divided by 5	PLL output frequency divided by 5	Multiple options from additional dividers in display engine
Default after reset	Connected to DDI A Clock	Disabled	Connected to DSI Clock
Programming	Not programmable.	Mapping of DDI to Transcoder must be programmed by software when enabling and disabling a display. Programming is done through the TRANS_CLK_SEL registers.	Dividers must be programmed by software when enabling and disabling a display. Programming is done through the MIPI_CLOCK_CTLregisters.

CD Clock

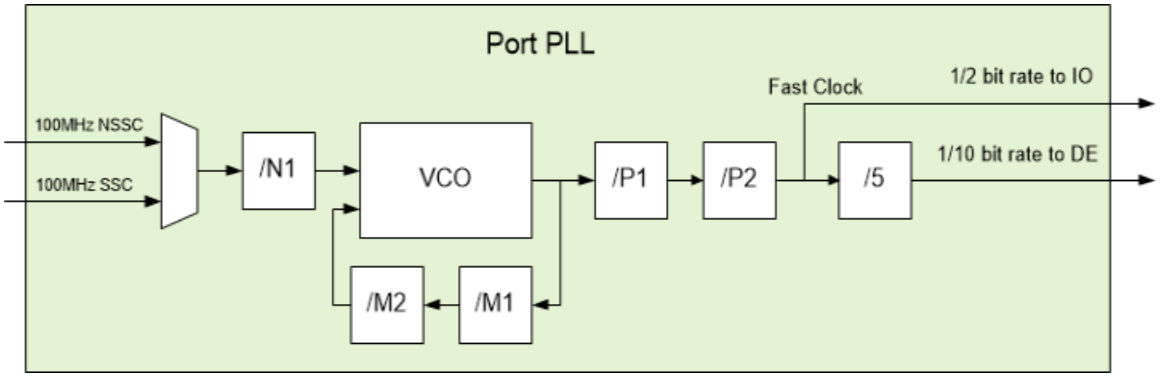
CD clock refers to the Core Display clock which includes the Core Display 1X Clock (CD clock, CDclk, cdclk, CDCLK) and the Core Display 2X Clock (CD2X clock, cd2xclk, CD2XCLK).

	CD clock
Usage	Clocking for most display engine functions
Input	DE PLL output (with two frequency options) or iCLKPLL 19.2 MHz reference. **iCLKPLL 19.2 MHz is automatically selected by hardware when Power Well #1 is disabled or DE PLL is not locked.**
Frequency	iCLKPLL (DE PLL disabled) - 19.2 MHz CD, 19.2 MHz CD2X DE PLL with output 1152 MHz • 144 MHz CD, 288 MHz CD2X • 288 MHz CD, 576 MHz CD2X • 384 MHz CD, 768 MHz CD2X • 576 MHz CD, 1152 MHz CD2X DE PLL with output 1248 MHz • 624 MHz CD, 1248 MHz CD2X
Default after reset	Running on the19.2 MHz Non-SSC reference
Programming	CD clock frequency must be programmed by software when enabling and disabling a display. Programming is done through the CDCLK_CTL and DE_PLL_CTL registers. The CD clock frequency may be changed on the fly in certain cases. Follow the Sequences for Changing CD Clock Frequency..

Recommended CD Clock Frequency Selection

In order to save power, software should use the lowest CD clock frequency necessary to support the current display configuration. Software may use a higher CD clock frequency if desired to reduce the cases where an already enabled display has to be disabled and re-enabled to change CD clock frequency when a second or third display is enable or disabled.

Port PLL Formula for Divider Values



eDP link rates: 1.62, 2.16, 2.43, 2.7, 3.24, 4.32, 5.4 GHz SSC and Non-SSC

DP link rates: 1.62, 2.7, 5.4 GHz SSC and Non-SSC

HDMI/DVI link rates: 0.2 to 3.0 GHz Non-SSC

Desired Output = Port bit rate in MHz (DisplayPort HBR2 is 5400 MHz)

Fast Clock = Desired Output / 2

VCO = Fast Clock * P1 * P2

P1 must be 2, 3, or 4

P2 must be 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 18, or 20 (1 to 20, even numbers when greater than 10)

VCO must be in the range 4800 to 6700

M2 = VCO / (M1 * Reference Clock / N)

M1 must be 2

N must be 1

Reference Clock must be 100 MHz

With the fixed values for M1, N, and reference clock, the formulas reduce to the following:

M2 = Desired Output * P1 * P2 / 400

VCO = M2 * 200

Actual Output = M2 * 400 / (P1 * P2) // Actual differs from desired due to limited M2 fractional precision

Note: If the desired output frequency cannot be achieved with the valid values of P1, P2, and M2, use a different screen resolution with a different output frequency.

Port PLL Algorithm to Find Divider Values

1. Try all valid values of P1 and P2
2. Calculate M2 for the P1, P2, and Desired Output
3. Adjust M2 for 22 bit fractional precision
4. Calculate VCO for the adjusted M2
5. Find if VCO is valid
6. Find the actual output frequency from the adjusted M2 and calculate the error PPM
7. Program PLL registers with the P1, P2, and M2 that had the lowest error and highest value of P1 * P2.

```
For P1 = (4,3,2)
        For P2 = (20,18,16,14,12,10,9,8,7,6,5,4,3,2,1)
                M2 = (Desired Output * P1 * P2) / 400
                M2 = ROUNDUP(2^22 * M2) / 2^22
                VCO = M2 * 200
                If (VCO>=4800) AND (VCO<6700)
                        Actual Output = M2 * 400 / (P1 * P2)
                        Error = Actual Output – Desired Output
                        PPM = 1,000,000 * Error / Desired Output
                        If PPM < 100 And (P1 * P2) > (Best P1 * Best P2)
                                Best PPM = 0
                                Best M2 = M2
                                Best P1 = P1
                                Best P2 = P2
                        End If
                        If PPM < (Best PPM – 10)
                                Best PPM = PPM
                                Best M2 = M2
                                Best P1 = P1
                                Best P2 = P2
                        End If
```

End If

Next P2

Next P1

Best M2 Integer = INT(Best M2)

Best M2 Fraction = ROUNDUP(2^22 * (Best M2 – Best M2 Integer))

Port PLL Example Divider Values

DisplayPort HBR2

Desired Output = 5400

Resulting P1=2, P2=1, and M2=27

Actual Output = 5400

Error PPM = 0

Program P1=2, P2=1, M2 Integer=27, M2 Fraction=0, and M2 Fraction Enable=0

HDMI 297 MHz

Desired Output = 2970

Resulting P1=4, P2=1, and M2=29.7

Actual Output = 2970.00000476837158203125

Error PPM = 0.0016

Program P1=4, P2=1, M2 Integer=29, M2 Fraction=2936013, and M2 Fraction Enable=1

PORT_PLL_6, PORT_PLL_8, and PORT_PLL_10 Values

These registers are programmed based on the frequency band.

The values in the table are decimal.

PLL VCO in GHz	i_prop_coeff	i_gainctrl	i_int_coeff	i_tdctargetcnt	i_dcoampovrden_h	i_dcoamp
6.2 ≤ VCO ≤ 6.7	4	3	9	8	1	15
5.4 < VCO < 6.2	5	3	11	9	1	15
5.4 = VCO	3	1	8	9	1	15
4.8 ≤ VCO < 5.4	5	3	11	9	1	15

Lane Stagger Values

Port symbol rate (bit rate/10) in MHz	Lane Stagger (hex)
270 < symbol rate ≤ 540	0x18
135 < symbol rate ≤ 270	0x0D
67 < symbol rate ≤ 135	0x07
33 < symbol rate ≤ 67	0x04
25 ≤ symbol rate ≤ 33	0x02

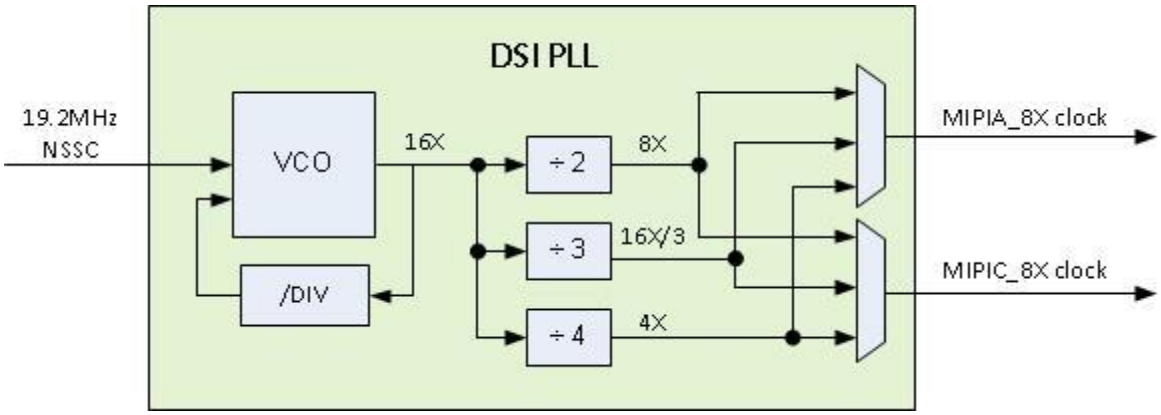
Port PLL Enabling Sequence

Step	Register	Programming	Comments
1	PORT_PLL_ENABLE	Write reference select	Select SSC or Non-SSC
2	PORT_PLL_EBB_4	Write 0 to o_dtdclkpen_h	Disable 10 bit clock to display engine
3	PORT_PLL_EBB_0	Write P1 to o_dtp1divsel Write P2 to o_dtp2divsel	
4	PORT_PLL_0	Write M2 integer portion to i_fbdivratio	INT(M2)
5	PORT_PLL_1	Write N to i_ndivratio	N is always 1
6	PORT_PLL_2	Write M2 fraction portion to i_fracdiv	ROUNDUP(2^22 * (M2 - INT(M2)))
7	PORT_PLL_3	Write M2 fraction enable to i_fracnen_h	M2 fraction enable = 1 if M2 fraction != 0
8	PORT_PLL_6	Write proportional coefficient to i_prop_coeff Write integral coefficient to i_int_coeff Write gain control to i_gainctrl	Values depend on frequency band
9	PORT_PLL_8	Write calibration value to i_tdctargetcnt	Values depend on frequency band
10	PORT_PLL_9	Write 5 to i_lockthresh	
11	PORT_PLL_10	Write DCO amplitude override value to i_dcoamp Write DCO amplitude override enable to i_dcoampovrden_h	Values depend on frequency band
12	PORT_PLL_EBB_4	Write 1 to o_dtafcrecal	Recalibrate with new settings
13	PORT_PLL_EBB_4	Write 1 to o_dtdclkpen_h	Enable 10 bit clock to display engine
14	PORT_PLL_ENABLE	Write 1 to PLL enable	Enable PLL
15	PORT_PLL_ENABLE	Poll for PLL lock = 1	Poll for lock
16	PORT_PCS_DW12 (use GRP instance)	Write lane stagger to reg_lanestagger_strap Write 1 to reg_lanestagger_strp_ovrd	Values depend on frequency band

Port PLL Disabling Sequence

1. Write 0 to PORT_PLL_ENABLE PLL enable
2. Poll PORT_PLL_ENABLE for PLL lock = 0
3. Timeout and fail if not unlocked after 200 us

MIPI DSI PLL



The MIPI DSI PLL block outputs the MIPI A 8X clock and MIPI C 8X clock.

A programmable divider controls the PLL 16X clock frequency, which goes through three other dividers in parallel to generate the outputs.

The DSI A clock and DSI C clock come from the 16X clock after being divided by 2 (8X), 3 (16X/3), or 4 (4X).

Example Ratios and Corresponding Frequencies (MHz)

Ratio	8X Clk	16X/3 Clk	4X Clk
34	326.4	217.6	163.2
35	336	224	168
40	384	256	192
45	432	288	216
50	480	320	240
55	528	352	264
57	547.2	364.8	273.6
60	576	384	288
63	604.8	403.2	302.4
65	624	416	312
70	672	448	336
75	720	480	360
80	768	512	384
85	816	544	408

90	864	576	432
95	912	608	456
100	960	640	480
110	1056	704	528
111	1065.5	710.4	532.8
125	1200	800	600

MIPI DSI PLL Enabling Sequence

1. Write DSI_PLL_CTL with the PLL ratio and output divider selections for DSI A and DSI C

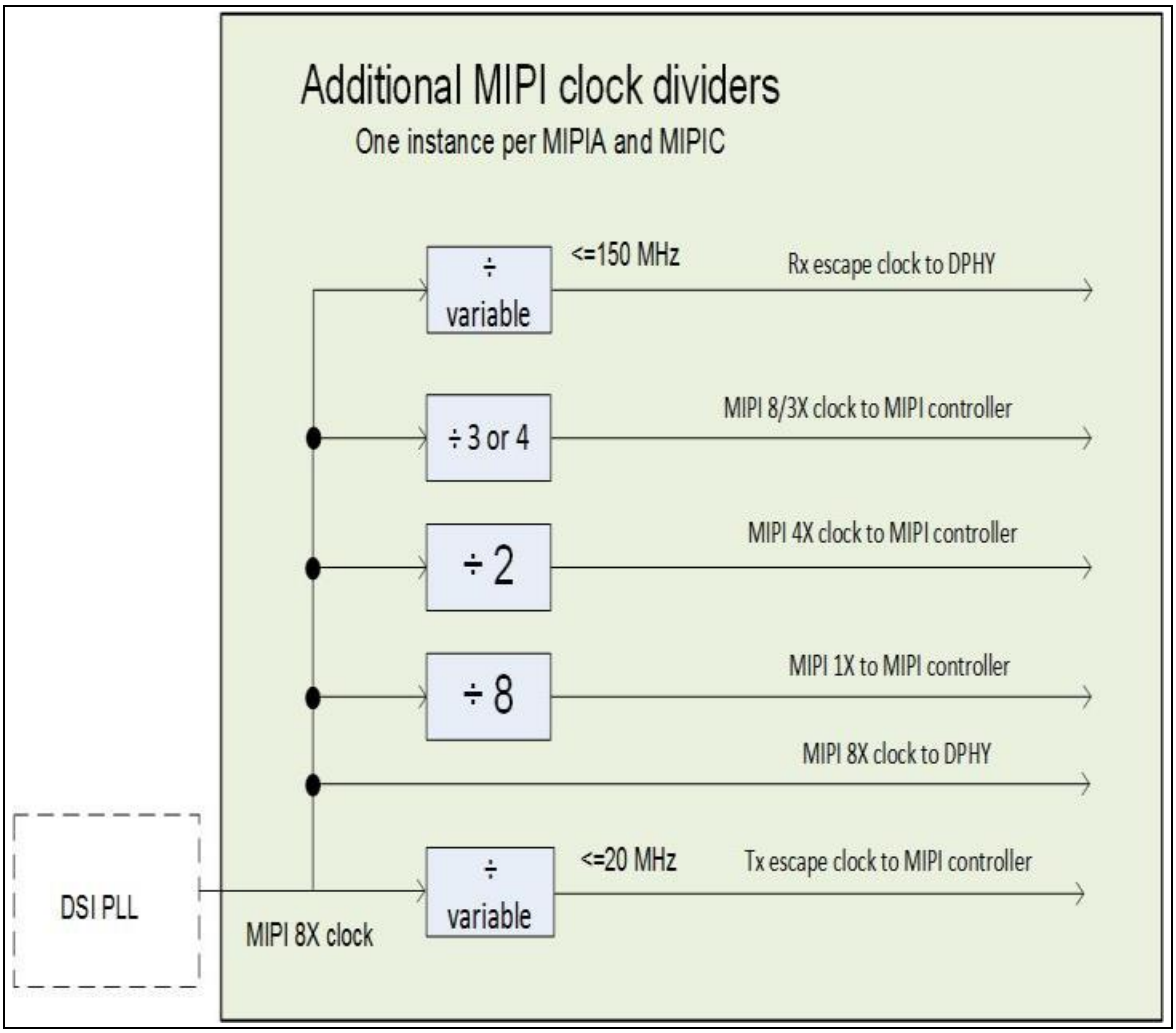
- Both output dividers must be programmed to valid values any time the PLL is enabled, even if only one output is used.
- The dividers cannot be changed while the PLL is enabled.

2. Write 1 to DSI_PLL_ENABLE PLL enable
3. Poll DSI_PLL_ENABLE for PLL lock = 1
4. Timeout and fail if not locked after 200 us

MIPI DSI PLL Disabling Sequence

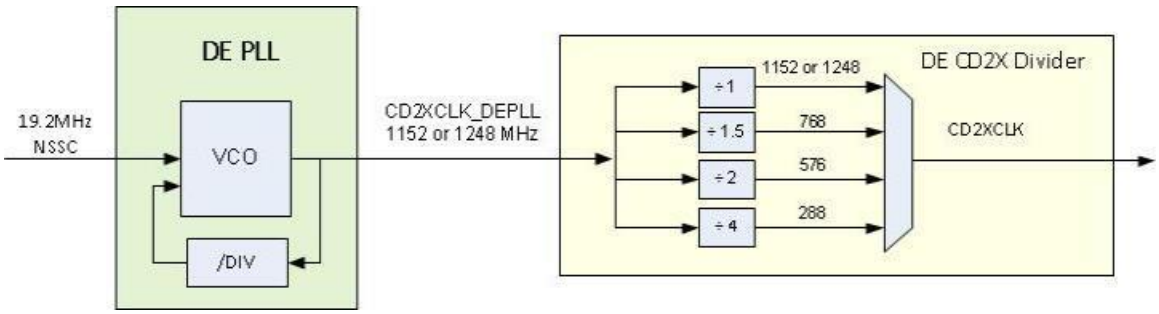
1. Write 0 to DSI_PLL_ENABLE PLL enable
2. Poll DSI_PLL_ENABLE for PLL lock = 0
3. Timeout and fail if not unlocked after 200 us

MIPI Additional Clock Dividers



The additional MIPI clock dividers further divide the MIPI PLL 8X clock to create the different clocks for the controller and I/O PHY. The dividers are programmed through the MIPI_CLOCK_CTL register and must follow the restrictions there.

DE PLL



The DE PLL outputs the CD2X DE PLL clock which is the source for CD clock.

A programmable divider, controlled by the DE_PLL_CTL register, inside the PLL controls the PLL frequency. The DE PLL CD2X Divider outside the PLL, controlled by the CDCLK_CTL register, provides additional dividing to create the CD2X clock used by the display engine.

Supported Ratios and Corresponding Frequencies (MHz)

DE PLL Ratio	DE CD2X Frequency	DE CD2X Divide	CD2XCLK	CDCLK
65	1248	1	1248	624
60	1152	1	1152	576
60	1152	1.5	768	384
60	1152	2	576	288
60	1152	4	288	144

DE PLL Sequences

- Follow the Broxton Sequences for Changing CD Clock Frequency

Broxton Sequences for Changing CD Clock Frequency

Restrictions
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.
Frequency changes required display to be disabled, except if only the DE CD2X Divider is changed.

Sequence for Changing CD Clock Frequency

1. Unless changing only the DE PLL 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. Inform power controller of upcoming frequency change
    a. Ensure any previous GT Driver Mailbox transaction is complete.
    b. Write GT Driver Mailbox Data Low = 0x80000000.
    c. Write GT Driver Mailbox Interface = 0x80000017.
    d. Poll GT Driver Mailbox Interface for Run/Busy indication cleared (bit 31 = 0).
    e. Timeout after 150 us. Do not change CD clock frequency if there is a timeout.
3. Enable or change the frequency of CD clock
    a. If enabling DE PLL
        i. Write DE_PLL_CTL with the PLL ratio
        ii. Set DE_PLL_ENABLE PLL Enable
        iii. Poll DE_PLL_ENABLE for PLL lock
        iv. Timeout and fail if not locked after 200 us
        v. Write CDCLK_CTL with the DE PLL CD2X Divider selection, CD Frequency Decimal value, and SSA Precharge Enable to match the desired CD clock frequency
    b. If disabling DE PLL
        i. Clear DE_PLL_ENABLE PLL Enable
        ii. Poll DE_PLL_ENABLE for PLL unlocked
        iii. Timeout and fail if not unlocked after 200 us
    c. If changing the DE PLL frequency
        i. Follow steps above for disabling DE PLL
        ii. Follow steps above for enabling DE PLL, using the new PLL ratio
    d. If changing only the DE PLL CD2X Divider

i. Write CDCLK_CTL with the DE CD2X Pipe selectoin, DE PLL CD2X Divider selection, CD Frequency Decimal value, and SSA Precharge Enable to match the desired CD clock frequency
ii. If pipe is enabled, wait for start of vertical blank for change to take effect

4. Inform power controller of the selected frequency
   - Use 19.2 MHz for the frequency if DE PLL is disabled

   a. Write GT Driver Mailbox Data Low with Ceiling[CDCLK frequency / 25].
      - For CD clock 19.2 MHz (DE PLL disabled), program 1 decimal.
      - For CD clock 79.2 MHz, program 4 decimal.
      - For CD clock 144 MHz, program 6 decimal.
      - For CD clock 288 MHz, program 12 decimal.
      - For CD clock 384 MHz, program 16 decimal.
      - For CD clock 576 MHz, program 24 decimal.
      - For CD clock 624 MHz, program 25 decimal.
      - For CD clock 156 or 158.4 MHz, program 7 decimal.
      - For CD clock 312 or 316.8 MHz, program 13 decimal.

   b. Write GT Driver Mailbox Interface = 0x80000017.
      - There is no need to wait for the Run/Busy indication to be cleared before continuing with display programming.

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 79.2 MHz, program M=15 and N=44 (decimal).
      - For CD clock 158.4 MHz, program M=15 and N=88 (decimal).
      - For CD clock 316.8 MHz, program M=15 and N=176 (decimal).
      - For CD clock 144 MHz, program M=3 and N=16 (decimal).
      - For CD clock 288 MHz, program M=3 and N=32 (decimal).
      - For CD clock 384 MHz, program M=9 and N=128 (decimal).
      - For CD clock 576 MHz, program M=3 and N=64 (decimal).
      - For CD clock 624 MHz, program M=9 and N=208 (decimal).
      - For CD clock 156 MHz, program M=9 and N=52 (decimal).
      - For CD clock 312 MHz, program M=9 and N=104 (decimal).

Resets

NDE_RSTWRN_OPT

The north and south display engines are reset by PCI Function Level Resets (FLR) and the chip level resets.

An FLR for Bus:Device:Function 0:2:0 resets the north and south display engines and audio codec and most of the related MMIO, PCI, and I/O configuration registers.

Display configuration registers that are reset by both the chip level reset and by FLR are marked as using the "soft" reset in the programming specification.

Display configuration registers that are reset only by the chip level reset and *not* by FLR are marked as using the "global" reset in the programming specification.

The south display engine runs panel power down sequencing (if configured to do so) before resetting.

This additional programming is needed to bring the display engine to the cold boot state after the FLR for Bus:Function:Device 0:2:0 completes. This is optional if software can tolerate display not being in the cold boot state after FLR.

1. Disable PWR_WELL_CTL Power Well 2 Request
2. Clear both PHY_CTL_FAMILY_EDP and PHY_CTL_FAMILY_DDI Common Reset to 0b (Disable)
3. Clear P_CR_GT_DISP_PWRON_0_2_0_GTTMMADR to 0x0
4. Disable PWR_WELL_CTL Power Well 1 Request

DGunit Registers (DGR)

Address Space	Address	Name
MMIO: 0/2/0	03000h	IOMMU_DEFEATURE_CAPECAPDIS
MMIO: 0/2/0	03004h	IOMMU_DEFEATURE_TLBDIS
MMIO: 0/2/0	03008h	IOMMU_DEFEATURE_PWSWTRDIS
MMIO: 0/2/0	0300Ch	IOMMU_DEFEATURE_MISCDIS
MMIO: 0/2/0	03010h	IOMMU_DEFEATURE_PWRDNOVRD
MMIO: 0/2/0	030B0h	GT_RELOAD_FLUSH
MMIO: 0/2/0	030C0h	GT_FLUSH_BCLD_ACK
MMIO: 0/2/0	04CD8h	PRMRR_BASE_LSB
MMIO: 0/2/0	04CDCh	PRMRR_BASE_MSB
MMIO: 0/2/0	04CE0h	PRMRR_MASK_LSB
MMIO: 0/2/0	04CE4h	PRMRR_MASK_MSB
MMIO: 0/2/0	06C88h	MCHBAR_LSB
MMIO: 0/2/0	06C8Ch	MCHBAR_MSB
MMIO: 0/2/0	100000h - 1000F8h	FENCE 0-31 LSB
MMIO: 0/2/0	100004h - 1000FCh	FENCE 0-31 MSB
MMIO: 0/2/0	100100h	DPFC_CONTROL_SA
MMIO: 0/2/0	100104h	DPFC_CPU_FENCE_OFFSET
MMIO: 0/2/0	101000h	TILECTL
MMIO: 0/2/0	101008h	GFX_FLSH_CNT
MMIO: 0/2/0	10102Ch	DISPLAY_DEADLINE_CONTROL
MMIO: 0/2/0	108000h	MTOLUD

Address Space	Address	Name
MMIO: 0/2/0	108040h	MMIO Mirror of GMCH Graphics Control Register (MGGC)
MMIO: 0/2/0	108080h	MTOUUD_LSB
MMIO: 0/2/0	108084h	MTOUUD_MSB
MMIO: 0/2/0	1080C0h	MBDSM
MMIO: 0/2/0	108100h	MBGSM
MMIO: 0/2/0	108140h	Base of DMA Protected Range
MMIO: 0/2/0	108180h	MPMEN
MMIO: 0/2/0	1081C0h	MPLMBASE
MMIO: 0/2/0	108200h	MPLMLIMIT
MMIO: 0/2/0	108240h	MPHMBASE_LSB
MMIO: 0/2/0	108244h	MPHMBASE_MSB
MMIO: 0/2/0	108280h	MPHMLIMIT_LSB
MMIO: 0/2/0	108284h	MPHMLIMIT_MSB
MMIO: 0/2/0	108300h	MGCMD
MMIO: 0/2/0	108340h	MEMRR_BASE_LSB
MMIO: 0/2/0	108344h	MEMRR_BASE_MSB
MMIO: 0/2/0	108380h	MEMRR_MASK_LSB
MMIO: 0/2/0	108384h	MEMRR_MASK_MSB
MMIO: 0/2/0	110000h	SVM_DEV_MODE_CNFG
MMIO: 0/2/0	120004h	GTACK
MMIO: 0/2/0	120800h	MAILBOX0
MMIO: 0/2/0	120804h	MAILBOX1

Address Space	Address	Name
MMIO: 0/2/0	120808h	MAILBOX2
MMIO: 0/2/0	12080Ch	MAILBOX3
MMIO: 0/2/0	124810h	FLT_RPT0
MMIO: 0/2/0	124814h	FLT_RPT1
MMIO: 0/2/0	124818h	FLT_RPT2
MMIO: 0/2/0	12481Ch	FLT_RPT3
MMIO: 0/2/0	124820h	PPRO
MMIO: 0/2/0	124824h	PPPR
MMIO: 0/2/0	124830h	RTADDR_LSB
MMIO: 0/2/0	124834h	RTADDR_MSB

Fuses and Straps

DFSERROR

Hot Plug Detection

Broxton Hot Plug Detection

HOTPLUG_CTL

HPD_PULSE_CNT

HPD_FILTER_CNT

For hot plug detection during Display C9 (DC9), see Broxton Sequences for Display C9.

Broxton 0 and Broxton 1 Board Mapping for A Stepping

HPD Pin Name in Display Engine	Die Bump Name	Board Connection
DDI A HPD	DDI0_HPD	DP/HDMI HPD
DDI B HPD	DDI1_HPD	Not connected
DDI C HPD	DDI2_HPD	EDP HPD

The board connections do not match the internal assignments, so software must remap how it assigns the HPDs. Software needs to activate the DDI A HPD logic and interrupts in order to check the external panel connection, and enable the DDI C HPD logic and interrupts in order to check the eDP panel connection. PSR on eDP will not automatically exit when there is a short pulse HPD_IRQ from the eDP panel. Software should set the PSR_MASK register Mask Hotplug field to prevent false PSR exits casued by the external panel.

Broxton 0 and Broxton 1 Board Mapping for E Stepping and Later

HPD Pin Name in Display Engine	Die Bump Name	Board Connection
DDI A HPD	DDI2_HPD	EDP HPD
DDI B HPD	DDI0_HPD	DP/HDMI HPD
DDI C HPD	DDI1_HPD	Not connected

Broxton P Board Mapping

HPD Pin Name in Display Engine	Die Bump Name	Board Connection
DDI A HPD	DDI2_HPD	EDP HPD
DDI B HPD	DDI0_HPD	DP/HDMI HPD
DDI C HPD	DDI1_HPD	DP/HDMI HPD

Backlight

This section refers to the CPU display backlight control. 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.

Description
PWM clock frequency = 19.2 MHz PWM increment = 1 PWM frequency divider maximum = 2^32 PWM output frequency range with PWM clock frequency 19.2 MHz and 100 brightness levels = 0.0004 to 192,000 Hz PWM output frequency range with PWM clock frequency 19.2 MHz and 256 brightness levels = 0.0004 to 75,000 Hz

Backlight Enabling Sequence

Description
1. Set frequency and duty cycle in BLC_PWM_FREQ Frequency and BLC_PWM_DUTY Duty Cycle. 2. Enable PWM output and set polarity in BLC_PWM_CTL PWM Enable and PWM Polarity. Polarity can be programmed earlier if needed ... 3. Change duty cycle as needed in BLC_PWM_DUTY Duty Cycle.

If needed, granularity and polarity can be programmed earlier than shown.

Backlight Registers

Description
BLC_PWM_CTL **BLC_PWM_FREQ** **BLC_PWM_DUTY** Backlight 2 BLC_PWM_*_2 uses the display utility pin for output. To use backlight 2, enable the utility pin with mode = PWM.

Panel Power

PP_STATUS

PP_CONTROL

PP_ON_DELAYS

PP_OFF_DELAYS

GMBUS and GPIO

Registers

GPIO_CTL - GPIO Control

GMBUS0 - GMBUS Clock/Port Select

GMBUS1 - GMBUS Command/Status

GMBUS2 - GMBUS Status

GMBUS3 - GMBUS Data Buffer

GMBUS4 - GMBUS Interrupt Mask

GMBUS5 - GMBUS 2 Byte Index

Pin Usage

These GPIO pins allow the support of simple query and control functions such as DDC interface protocols. The GMBUS controller can be used to run the interface protocol, or the GPIO pins can be manually programmed for a "bit banging" interface.

The following table describes the expected GPIO pin to register mapping. OEMs have the ability to remap these functions onto other pins as long as the hardware limitations are observed. The GPIO pins may also be muxed with other functions such that they are only available when the other function is not being used.

Port #	Name	Pull up/down	Description
1	DDIB CTLDATA	No (Weak pull down on reset)	DDC for DDI port B.
	DDIB CTLCLK	No	
2	DDIC CTLDATA	No (Weak pull down on reset)	DDC for DDI port C.
	DDIC CTLCLK	No	
3	MISC CTLDATA	No (Weak pull down on reset)	DDC for Miscellaneous usage.
	MISC CTLCLK	No	

Broxton 0 and Broxton 1 Board Mapping

DDI Name in Display Engine	Die Bump Name	Board Connection
DDIB CTLDATA	DDI0_DDC_SDA	DP/HDMI DDC Data
DDIB CTLCLK	DDI0_DDC_SDA	DP/HDMI DDC Clock

DDI Name in Display Engine	Die Bump Name	Board Connection
DDIC CTLDATA	DDI1_DDC_SDA	Not connected
DDIC CTLCLK	DDI1_DDC_SCL	Not connected
MISC CTLDATA	DBI_SDA	MIPI external LVDS/CRT Converter DDC Data
MISC CTLCLK	DBI_SCL	MIPI external LVDS/CRT Converter DDC Clock

DDI C is not connected on the current die, but may be connected in the future.

GMBUS Controller Programming Interface

The GMBUS (Graphic Management Bus) is used to access/control devices connected to the GPIO pins.

Basic features:

1. I²C compatible.
2. Bus clock frequency of 50 KHz or 100 KHz.
3. Attaches to any of the GPIO pin pairs.
4. 7-bit or 10-bit Slave Address and 8-bit or 16-bit index.
5. Double buffered data register and a 9 bit counter support 0 byte to 256 byte transfers.
6. Supports stalls generated by the slave device pulling down the clock line (Slave Stall), or delaying the slave acknowledge response.
7. Status register indicates error conditions, data buffer busy, time out, and data complete acknowledgement.
8. Detects and reports time out conditions for a stall from a slave device, or a delayed or missing slave acknowledge.
9. Interrupts may optionally be generated.
10. Does not directly support segment pointer addressing as defined by the Enhanced Display Data Channel standard.

Segment pointer addressing as defined by the Enhanced Display Data Channel standard:

1. Use bit bashing (manual GPIO programming) to complete segment pointer write **without terminating in a stop or wait cycle**.
2. Terminate bit bashing phase with both I²C lines pulled high by tri-stating the data line before the clock line. Follow EDDC requirement for response received from slave device.
3. Initiate GMBUS cycle as required to transfer EDID following normal procedure.

Program 0x4653C[14] = 0x1 when doing back to back GMBUS transactions. The value can be safely left at 0x1 when GMBUS is not being used.

This will disable GMBUS unit level clock gating and only rely on partition level clock gating.

DC States

DC_STATE_EN

Plane Assignments and Capabilities

Broxton Plane Capabilities:

	Pipe A	Pipe B	Pipe C
Plane 4	Plane or Cursor	Plane or Cursor	Not supported
Plane 3	Plane	Plane	Plane or Cursor
Plane 2	Plane + Render Decompression, NV12	Plane + Render Decompression, NV12	Plane
Plane 1	Plane + Render Decompression, Frame Buffer compression, NV12	Plane + Render Decompression, NV12	Plane

Broxton 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
N/A	Plane 12

Display Buffer Programming

This topic describes display buffer allocation and shows a basic allocation method for single and multi-pipe modes. The display driver can choose to use more advanced allocation techniques as desired.

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

Display Buffer Size	Total Display Buffer Blocks	Fixed Bypass Path Allocation in Blocks	Blocks Available for Driver Programming
256 KB	512	4	0 - 507

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.

Planar YUV 420 Surfaces:

For YUV 420 Planar formats (NV12, P0xx), buffer allocation is done for Y and UV surfaces separately. Treat Y and UV surface as 2 separate planes. Also,the plane height and plane width for the UV plane should be halved.

UV Plane Height = Plane Height/2

UV Plane Width = Plane Width/2

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 * Plane source width * Plane Bpp)/512] * 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

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

Allocate a fixed number of blocks to cursors and then allocate the remaining blocks among planes, based on each plane's data rate.

BlocksAvailable = 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).

CursorBufAlloc = 32

BlocksAvailable = BlocksAvailable – 32

2. Check for minimum buffer requirement

For each enabled plane

If Y tiled

MinScanLines = Look up minimum scanlines needed from the table

*PlaneMinAlloc = Ceiling [(4*Plane width*Bpp)/512] * MinScanLines/4 + 3*

Else

PlaneMinAlloc = 8

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:

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.

NumPipes = Total number of display pipes in the hardware

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.

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 acheive consistent latency levels supported in all planes across all pipes.

- Allocate fixed number of blocks for cursor (for example 32 blocks).
- For each enabled plane in all pipes, 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 remaing 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 prevent package power saving states from enabling.

Supported usages:

- Up to 3 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 render compression
- Any screen resolution
- Downscaling less than or equal to 12.5%

Allocate 160 blocks per pipe.

Pipe A: 0-159, Pipe B: 160-319, Pipe C: 320-479

PLANE_BUF_CFG_<plane number>_A = 0x009F0000

PLANE_BUF_CFG_<plane number>_B = 0x013F00A0

PLANE_BUF_CFG_<plane number>_C = 0x01DF0140

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.

VGA_CONTROL

VGA Enabling Sequence:

1. Program GTdriver mailbox data low = 0x000082BC (Override valid, latency reduced).
2. Program GT driver mailbox interface = 0x8000001D (WRITE_LTR_OVERRIDE).
3. Poll for GT driver mailbox interface = 0x00000000 (done and success). Timeout after 1ms and do not enable VGA.
4. Enable VGA in VGA_CONTROL
5. Clear VGA I/O register SR01 bit 5

VGA Disabling Sequence

1. Set VGA I/O register SR01 bit 5 for screen off
2. Wait for 100 us
3. Disable VGA in VGA_CONTROL
4. Program GT driver mailbox data low = 0x00000000 (Override invalid).
5. Program GT driver mailbox interface = 0x8000001D (WRITE_LTR_OVERRIDE).

Poll for GTdriver mailbox interface = 0x00000000 (done and success). Timeout after 1ms and continue.

Cursor Plane

Planes
PLANE_SURFLIVE
CUR_SURFLIVE

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.

DDI Buffer

Registers

DDI_BUF_CTL

PHY_CTL_DDI

PHY_CTL_FAMILY

PORT_CL1CM_DW0

PORT_CL1CM_DW9

PORT_CL1CM_DW10

PORT_CL1CM_DW28

PORT_CL1CM_DW30

PORT_REF_DW3

PORT_REF_DW6

PORT_REF_DW8

PORT_TX_DW2

PORT_TX_DW3

PORT_TX_DW4

PORT_TX_DW14

PORT_PCS_DW10

PORT_PCS_DW12

P_CR_GT_DISP_PWRON_0_2_0_GTTMMADR

PORT_CL and PORT_REF address entire PHYs.

PORT_PCS address lane pairs (0+1, 2+3) per DDI (A, B, C).

PORT_TX address individual lanes (0, 1, 2, 3) per DDI (A, B, C), or the entire group of lanes within a DDI (A, B, C).

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.

Broxton

DDIB and DDIC share a dual channel PHY, so PHY based programming will impact both DDIs at the same time.

DDIA/EDP uses a single channel PHY, so it is the only port in that PHY.

DDI PHY Initialization Sequence

The DDI PHY must be initialized before any DDI mode set or using the Aux channel.

Only the DDIA PHY is connected to a Rcomp resistor. DDIB and DDIC are dependent on the Rcomp value from DDIA, so DDIA must be enabled first. If DDIA won't be used, it can be disabled after Rcomp is copied.

1. If any DDI will be used, enable DDIA first since it supplies Rcomp
    a. Write 1b to power request CH1_PWRREQ1P0_SUS (DDIA/eDP IO Power On) in P_CR_GT_DISP_PWRON_0_2_0_GTTMMADR
    b. Poll for PORT_CL1CM_DW0_A iphypwrgood == 1b and bit 7 (reserved) == 0b. Reserved bit is checked to ensure register is in an accessible state (recommended poll time interval = 100 us).
    c. If motherboard DDI RCOMP resistor is 100 Ohms, set PORT_REF_DW8_A fcomprefsel = 1b, else leave at default 0b for 400 Ohms.
    d. Program PLL Rcomp code offset
        i. Write 0xE4 to iref0rcoffset in PORT_CL1CM_DW9_A
        ii. Write 0xE4 to iref1rcoffset in PORT_CL1CM_DW10_A
    e. Program power gating
        i. Write 11b to sus_clk_config and 1b to ocl1powerdownen and oldo_dynpwrdownen in PORT_CL1CM_DW28_A
    f. Set PHY_CTL_FAMILY_EDP Common Reset = 1b (Enable)
2. If DDIB or DDIC will be used enable both B and C since they share the same PHY
    a. Write 1b to power request CH0_PWRREQ1P0_SUS (DDIB/DDIC IO Power On) in P_CR_GT_DISP_PWRON_0_2_0_GTTMMADR
    b. Poll for PORT_CL1CM_DW0_BC iphypwrgood == 1b and bit 7 (reserved) == 0b. Reserved bit is checked to ensure register is in an accessible state.

c. If motherboard DDI RCOMP resistor is 100 Ohms, set PORT_REF_DW8_BC fcomprefsel = 1b, else leave at default 0b for 400 Ohms.
d. Program PLL Rcomp code offset
    i. Write 0xE4 to iref0rcoffset in PORT_CL1CM_DW9_B and PORT_CL1CM_DW9_C
    ii. Write 0xE4 to iref1rcoffset in PORT_CL1CM_DW10_B and PORT_CL1CM_DW10_C
e. Program power gating
    i. Write 11b to sus_clk_config and 1b to ocl1powerdownen and oldo_dynpwrdownen in PORT_CL1CM_DW28_BC
    ii. Write 1b to oldo_dynpwrdownen in PORT_CL2CM_DW6 [BXT]
f. Copy Rcomp from single channel PHY (EDP A) to dual channel PHY (DDI BC)
    i. Poll for PORT_REF_DW3_A grc_done == 1b
    ii. Read and save grccode from PORT_REF_DW6_A
    iii. Write the saved grccode to ogrccode_fast, ogrccode_slow, and ogrccode_nom in PORT_REF_DW6_BC
    iv. Write 1b to grcdis and grc_rdy_ovrd in PORT_REF_DW8_BC
g. Set PHY_CTL_FAMILY_DDI Common Reset = 1b (Enable)

3. If DDIA/EDP won't be used it can now be disabled
    1. Clear PHY_CTL_FAMILY_EDP Common Reset to 0b (disable)
    2. Clear P_CR_GT_DISP_PWRON_0_2_0_GTTMMADR CH1_PWRREQ1P0_SUS (DDIA/eDP IO Power On) to 0b (power off)

DDI PHY Un-Initialization Sequence

- Disable the PHY to save power if it will not be used for awhile
- DDIA/EDP
    1. Clear PHY_CTL_FAMILY_EDP Common Reset to 0b (disable)
    2. Clear P_CR_GT_DISP_PWRON_0_2_0_GTTMMADR CH1_PWRREQ1P0_SUS (DDIA/eDP IO Power On) to 0b (power off)
    3. If the PHY is needed again, follow the initialization sequence. Rcomp does not need to be copied if DDIB+DDIC have not been disabled.
- DDIB+DDIC
    1. Clear PHY_CTL_FAMILY_DDI Common Reset to 0b (disable)
    2. Clear P_CR_GT_DISP_PWRON_0_2_0_GTTMMADR CH0_PWRREQ1P0_SUS (DDIB/DDIC IO Power On) to 0b (power off)

If the PHY is needed again, follow the initialization sequence, including the DDIA enable and Rcomp copying.

Latency Optimization Settings

The latency optimization setting affects the alignment between output of different data lanes. It must be programmed during the DDI mode set, before enabling DDI_BUF_CTL. The value depends on the number of lanes enabled and lane reversal.

The table below gives the value to program in PORT_TX_DW14_LN<0,1,2,3>_<DDI> latency_optim for each lane for a given lane configuration on that DDI. Group access cannot be used here since each lane can have a unique value, and any later writes to PORT_TX_DW14 must not use group access so that they don't overwrite the inidivual lane values.

Lane configuration / latency_optim setting	X1 - Reversed and non-reversed	X2 - reversed and non-reversed	X4 - Reversed and non-reversed
PORT_TX_DW14_LN0_<DDI>	0	1	1
PORT_TX_DW14_LN1_<DDI>	0	0	0
PORT_TX_DW14_LN2_<DDI>	0	1	1
PORT_TX_DW14_LN3_<DDI>	0	0	1

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.

1. Clear calc init
    - Write 0 to PORT_PCS_DW10_GRP_<DDI Letter> reg_tx1swingcalcinit and reg_tx2swingcalcinit
2. Program swing and deemphasis
    - Write swing value to PORT_TX_DW2_GRP_<DDI Letter> omargin000
    - Write scale value to PORT_TX_DW2_GRP_<DDI Letter> ouniqtranscale
    - Write scale enable to PORT_TX_DW3_GRP_<DDI Letter> oscaledcompmethod
    - Write deepmphasis value to PORT_TX_DW4_GRP_<DDI Letter> ow2tapdeemph9p5
3. Set calc init to trigger update
    - Write 1 to PORT_PCS_DW10_GRP_<DDI Letter> reg_tx1swingcalcinit and reg_tx2swingcalcinit

Voltage swing programming for DisplayPort							
Voltage Swing Level[1]	**Pre-emphasis Level**[1]	**Non-Transition mV diff p-p**	**Transition mV diff p-p**	**Pre-emphasis dB**	**Scale Enable and Value**	**Deemphasis decimal**	**Swing decimal**
0	0	400	400	0	0, Don't Care	128	52
0	1	400	600	3.5	0, Don't Care	85	78
0	2	400	800	6	0, Don't Care	64	104
0	3	400	1200	9.5	0, Don't Care	43	154
1	0	600	600	0	0, Don't Care	128	77
1	1	600	900	3.5	0, Don't Care	85	116
1	2	600	1200	6	0, Don't Care	64	154
2	0	800	800	0	0, Don't Care	128	102

Voltage swing programming for DisplayPort							
2	1	800	1200	3.5	0, Don't Care	85	154
3	0	1200	1200	0	1, 0x9A	128	154

[1]The voltage swing level and pre-emphasis level values follow the naming used in the DisplayPort standard.

Voltage swing programming for embedded DisplayPorts that support low voltage swings							
Voltage Swing Level[1]	**Pre-emphasis Level**[1]	**Non-Transition mV diff p-p**	**Transition mV diff p-p**	**Pre-emphasis dB**	**Scale Enable and Value**	**Deemphasis decimal**	**Swing decimal**
0	0	200	200	0	0, Don't Care	128	26
0	1	200	240	1.5	0, Don't Care	112	38
0	2	200	320	4	0, Don't Care	96	48
0	3	200	400	6	0, Don't Care	69	54
1	0	250	250	0	0, Don't Care	128	32
1	1	250	300	1.5	0, Don't Care	104	48
1	2	250	400	4	0, Don't Care	85	54
2	0	300	300	0	0, Don't Care	128	43
2	1	300	360	1.5	0, Don't Care	101	54
3	0	350	350	0	0, Don't Care	128	48

[1]The voltage swing level and pre-emphasis level values follow the naming used in the DisplayPort standard.

Broxton

Voltage swing programming for HDMI and DVI						
Non-Transition mV diff p-p	**Transition mV diff p-p**	**Pre-emphasis dB**	**Scale Enable and Value**	**Deemphasis decimal**	**Swing decimal**	**Recommended Usage**
400	400	0	0, Don't Care	128	52	Recommended for HDMI Active Level Shifter
400	600	3.5	0, Don't Care	85	52	
400	900	6	0, Don't Care	64	52	
400	1200	9.5	0, Don't Care	43	42	
600	600	0	0, Don't Care	128	77	
600	900	3.5	0, Don't Care	85	77	
600	1200	6	0, Don't Care	64	77	
800	800	0	0, Don't Care	128	102	
800	1200	3.5	0, Don't Care	85	102	Recommended for HDMI Cost Reduced Level Shifter and DisplayPort Dual Mode
1200	1200	0	1, 0x9A	128	154	
1000	1000	0	1, 0x9A	128	133	Frequencies less than or equal to 165 MHz
850	1200	3	1, 0x9A	96 for lanes 0/1/2 *128 for lane 3	160	Frequencies greater than 165 MHz *Use group register offset to program deemphasis 96 to all lanes, then use the LN3 register offset to program deemphasis 128 for lane 3