



MIPI 2.5G DSI RX Controller Core User Guide

UG-CORE-MIPI-2-5G-DSI-RX-v1.0
March 2026
www.efinixinc.com



Contents

Introduction.....	3
Features.....	3
Device Support.....	3
Resource Utilization and Performance.....	4
Release Notes.....	4
Functional Description.....	5
Ports.....	6
Clocking.....	12
Register Definition.....	13
Video Mode Configuration.....	17
Command Packet Data Types.....	17
Sync Event Packet Data Type.....	18
Video Mode Pixel Encoding.....	18
MIPI Video Data DATA[63:0] Formats.....	19
Pixel Clock Calculation.....	20
Video Timing Parameters.....	21
Reset Sequence and Initialization.....	22
IP Manager.....	23
Customizing the MIPI 2.5G DSI RX Controller.....	24
MIPI 2.5G DSI RX Controller Example Design.....	25
Revision History.....	25

Introduction

The MIPI DSI specifies the physical link between the chip and display in devices such as smartphones, tablets, AR/VR headsets, and connected cars⁽¹⁾. It defines a serial bus and a communication protocol between the host (the source of the image data) and the destination (e.g., display peripherals). The MIPI 2.5G DSI RX Controller core implements the MIPI DSI interface in the FPGA and allows you to configure the related parameters.

Features

- Supports 1,2, and 4 lanes
- Supports continuous or discontinuous clock mode
- HS mode byte clock frequency from 5 MHz to 156.25 MHz (80 Mbps to 2,500 Mbps data rate)⁽²⁾
- 16-bit HS mode data width
- Includes AXI4-Lite interface for register access
- Error correction code (ECC) verification for packet headers
- Cyclic redundancy check (CRC) verification for data bytes
- Supports non-burst with sync pulses, non-burst with sync events, and burst mode
- Supports end of transmission packet
- Supports command transmission in HS or LP mode
- Supports initial auto-skew calibration
- Supports PPI interface

Device Support

Table 1: MIPI 2.5G DSI RX Controller Core Device Support

FPGA Family	Supported Device
Trion	-
Titanium	Refer to the Titanium Selector Guide
Topaz	Refer to the Topaz Selector Guide

⁽¹⁾ Source: MIPI Alliance.

⁽²⁾ The maximum data rate of IP depends on the devices. Refer to the respective device data sheet for more accurate information.

Resource Utilization and Performance



Note: The resources and performance values provided are based on some of the supported FPGAs. These values are just guidance and may change depending on the device resource utilization, design congestion, and user design.

Table 2: Titanium Resource Utilization and Performance

MIPI 2.5G DSI RX Controller with 4 data lanes.

FPGA	Flip-Flop	LUT	RAM	DSP	f_{MAX} (MHz) ⁽³⁾					Efinity® Version ⁽⁴⁾
					clk_esc	axi_clk	clk_byte_HS	phy_clk_byte_HS	clk_pixel	
Ti180 J484 C4	1,417	2,286	30	0	493	274	324	270	242	2025.2.288.3.8

Release Notes

You can refer to the IP Core Release Notes for more information about the IP core changes. The IP Core Release Notes are available on the [Efinity Downloads](#) page under each Efinity software release version.



Note: You must be logged in to the Support Center to view the IP Core Release Notes.

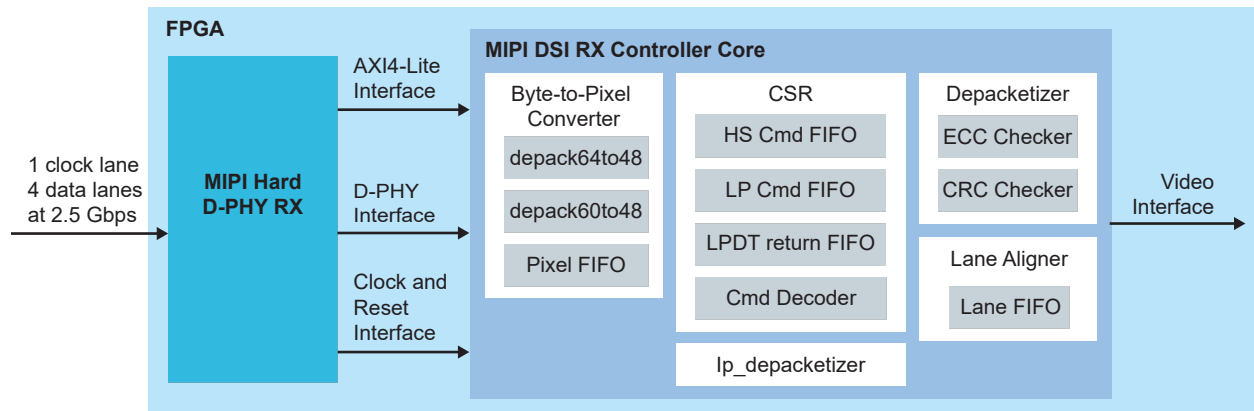
⁽³⁾ Using default parameter settings.

⁽⁴⁾ Using System Verilog.

Functional Description

The MIPI 2.5G DSI RX Controller core includes control status registers, an ECC checker, a CRC checker, a depacketizer, a byte-to-pixel converter, and a lane aligner. Additionally, the MIPI 2.5G DSI RX Controller core has a video interface that operates in the pixel clock domain, an AXI4-lite interface, a clock and reset interface, and a PPI interface which runs in the MIPI byte clock domain. The DSI RX controller communicates with the MIPI hard D-PHY in the core device through the PPI interface.

Figure 1: MIPI 2.5G DSI RX Controller System Block Diagram



Ports

Table 3: Clock and Reset Ports

Port	Direction	Description
clk	Input	IP core clock consumed by controller logic. Fixed at 100 Mhz.
reset_n	Input	IP core reset signal.
clk_byte_HS	Input	MIPI RX 8-bit parallel data clock signal. MIPI data rate / 8-bit data width. This clock must run at 2 times the phy_clk_byte_HS clock frequency. $clk_byte_HS = 2 * phy_clk_byte_HS$
reset_byte_HS_n	Input	MIPI RX 8-bit parallel data reset signal.
phy_clk_byte_HS	Input	MIPI RX 16-bit parallel data clock signal. Clock frequency = MIPI data rate / 16-bit data width. Supplied by MIPI hard D-PHY. $clk_byte_HS = 2 * phy_clk_byte_HS$
TxCkEsc	Input	Escape mode transmit clock. This clock is typically 20 MHz or lower.
clk_pixel	Input	Pixel clock.
reset_pixel_n	Input	Pixel reset signal.
axi_clk	Input	AXI4-Lite interface clock.
axi_reset_n	Input	AXI4-Lite interface reset.

Table 4: PHY Protocol interface (PPI)

Port	Direction	Description
RxUlpsClkNot	Input	Received ULPS on clock lane. This bus is synchronized to axi_clk in the controller and stored in the CSR. This active-low signal is asserted to indicate that the clock lane module has entered the ULPS due to the detection of a request to enter the ULPS.
RxUlpsActiveClkNot	Input	ULPS (not) active. This bus is synchronized to axi_clk in the controller and stored in the CSR. This active-low signal is asserted to indicate that the lane is in ULPS.
RxCkEsc [NUM_DATA_LANE-1:0]	Input	Escape mode receive clock. This escape clock is not used in the controller in the current core version. It is supplied by the MIPI hard D-PHY. This signal is used to transfer received data to the protocol during escape mode.
RxErEsc [NUM_DATA_LANE-1:0]	Input	Escape entry error. This bus is synchronized to axi_clk in the controller and stored in the CSR. If an unrecognized escape entry command is received in LP mode, this active-high signal is asserted and remains asserted until the next transaction starts, allowing the protocol to process the error.
RxErControl [NUM_DATA_LANE-1:0]	Input	Control error. This bus is synchronized to axi_clk in the controller and stored in the CSR. This active-high signal is asserted when an incorrect line state sequence is detected in LP mode. Once asserted, this signal remains asserted until the next transaction starts, so that the protocol can properly process the error.

Port	Direction	Description
RxErrSotSyncHS [NUM_DATA_LANE-1:0]	Input	Start of transmission synchronization error. This bus is be synchronized to axi_clk in the controller and stored in the CSR. If the HS SoT leader sequence is corrupted in a way that proper synchronization cannot be expected, this active-high signal is asserted for one cycle of phy_clk_byte_HS. When ErrSotSyncHS is asserted, RxSyncHS, ErrSotHS, and RxValidHS is not asserted.
RxUlpEsc [NUM_DATA_LANE-1:0]	Input	Escape ULPS. This bus is synchronized to axi_clk in the controller and stored in the CSR. This active-high signal is asserted to indicate that the lane module has entered the ULPS, due to the detection of a received ULPS command.
RxUlpActiveNot [NUM_DATA_LANE-1:0]	Input	ULPS (not) active. This bus is synchronized to axi_clk in the controller and stored in the CSR. This active-low signal is asserted to indicate that the lane is in ULPS.
RxSkewCalHS [NUM_DATA_LANE-1:0]	Input	HS receive skew calibration. This bus is clocked by clk_byte_HS. This optional active-high signal indicates that the high speed deskew burst is received.
RxStopState [NUM_DATA_LANE-1:0]	Input	Data lane in stop state. This asynchronous active-high signal indicates that the MIPI D-PHY data lane is currently in stop state.
RxSyncHS [NUM_DATA_LANE-1:0]	Input	Receiver synchronization observed. This bus is clocked by clk_byte_HS. This active-high signal indicates that the data lane is in appropriate synchronization event.
RxDataHSn [HS_DATA_WIDTH-1:0]	Input	HS data received by MIPI D-PHY data lane. This bus is clocked by clk_byte_HS. $n = \text{lane } 0 \text{ to } 3$
RxValidHSn [HS_DATA_WIDTH/8-1:0]	Input	This active-high signal indicates that the MIPI D-PHY data lane is driving data to the protocol layer on the RxDataHS output. This bus is clocked by clk_byte_HS. RxValidHSn[0]: RxDataHS[7:0] contains valid data received from the channel. RxValidHSn[1]: RxDataHS[15:8] contains valid data received from the channel. $n = \text{lane } 0 \text{ to } 3$
TxCkEsc	Input	Escape mode transmit clock. This clock is directly used to generate escape sequences. The period of this clock determines the symbol time for the low power signals in escape mode. Typically, it is 20 Mhz or below. If data turnaround feature is not needed, this clock can be tied to zero.

Port	Direction	Description
TxUlpsActiveNot	Input	<p>ULP state (not) active.</p> <p>This active low signal is asserted to indicate that the Lane is in the ULP state.</p> <p>At the beginning of the ULP state, UlpsActiveNot is asserted together with RxUlpsEsc, or RxClkUlpsNot for a clock lane. At the end of the ULP state, this signal becomes inactive to indicate that the Mark-1 state has been observed. Later, after a period of time, Twakeup, the RxUlpsEsc (or RxClkUlpsNot) signal is deasserted.</p>
TxUlpsEsc	Output	<p>Escape mode transmit ultra-low power state.</p> <p>This active high signal is asserted with TxRequestEsc to cause the lane module to enter low-power data transmission mode. The lane module remains in this mode until TxRequestEsc is de-asserted. TxUlpsEsc and all bits of TxTriggerEsc are low when TxLpdtEsc is asserted. If the application doesn't need DSI function, this signal is connected to 1'd0.</p>
TxUlpsExit	Output	<p>Transmit ULPS exit sequence.</p> <p>This active high signal is asserted when ULP state is active and the protocol is ready to leave ULP state. The PHY leaves ULP state and begins driving Mark-1 after TxUlpsExit is asserted. The PHY later drives the Stop state (LP-11) when TxRequestEsc is de-asserted. TxUlpsExit is synchronous to TxClkEsc. This signal is ignored when the Lane is not in the ULP State. If the application doesn't need DSI function, this signal is connected to 1'd0.</p>
TxRequestEsc	Output	<p>Escape mode transmit request.</p> <p>This active high signal, asserted together with exactly one of TxLpdtEsc, TxUlpsEsc, or one bit of TxTriggerEsc, is used to request entry into escape mode. Once in escape mode, the lane stays in escape mode until TxRequestEsc is deasserted. TxRequestEsc is only asserted by the protocol when TxRequestHS is low. If the application does not need a DSI function, then this signal is connected to 1'd0.</p>
TxTriggerEsc [3:0]	Output	<p>Escape mode transmit trigger 3-0.</p> <p>One of these active high signals is asserted with TxRequestEsc to cause the associated Trigger to be sent across the lane interconnect.</p>
TxLpdtEsc	Output	<p>Escape mode transmit low power data.</p> <p>This active high signal is asserted with TxRequestEsc to cause the lane module to enter low-power data transmission mode. The lane module remains in this mode until TxRequestEsc is deasserted. TxUlpsEsc and all the bits of TxTriggerEsc are low when TxLpdtEsc is asserted.</p>
TxDataEsc[7:0]	Output	<p>Escape mode transmit data bus.</p> <p>This is an 8-bit escape mode data to be transmitted in low-power data transmission mode. The signal connected to TxDataEsc[0] is transmitted first. Data is captured on rising edges of TxClkEsc.</p>
TxValidEsc	Output	<p>Escape mode transmit data valid.</p> <p>This active high signal indicates that the protocol is driving valid data on TxDataEsc to be transmitted.</p>

Port	Direction	Description
TxReadyEsc	Input	Escape mode transmit ready. This active high signal indicates that TxDataEsc is accepted by the lane module to be serially transmitted.
TxStopState	Input	Lane is in stop state. This active high signal indicates that the lane module, is currently in stop state. This signal is asynchronous to any clock in the PPI interface.

Table 5: Video Interface

All signals are clocked by `clk_pixel`.

Port	Direction	Description
hsync	Output	Active-low horizontal sync.
vsync	Output	Active-low vertical sync.
pixel_data [63:0]	Output	Video data. The actual width of this port is dependent on pixel type. See Video Mode Pixel Encoding on page 18.
pixel_data_valid	Output	Active-high pixel data enable.
pixel_vc [1:0]	Output	Virtual channel signal of packet received by DSI RX controller.
pixel_format [5:0]	Output	Pixel data format of packet received by DSI RX controller.

Table 6: Sideband Interface

All signals are clocked by `clk_byte_hs` except for `irq`, which is clocked by `axi_clk`.

Port	Direction	Description
video_format[5:0]	Input	Set to related video format data type for cycle-accurate pixel interface generation when converting the decoded packets (in byte clock domain) to pixel interface (in pixel clock domain). E.g., set to 6'h3E if using RGB888 format. Fixed to zero if there is no concern for inaccuracy of pixel interface timing (especially on HSA, and HBP), then the pixel interface (<code>hsync</code> , <code>vsync</code> , <code>pixel_data</code> , <code>pixel_data_valid</code>) will be decoded based on byte clock domain.
vc[1:0]	Output	2-bit virtual channel signal.
word_count[15:0]	Output	Byte count of the long packet received by DSI RX controller.
datatype[5:0]	Output	Decoded data type of packet received by DSI RX controller.
irq	Output	Interrupt signal for interrupt status register.

Table 7: AXI4-Lite Interface

All signals are clocked by `axi_clk`.

Port	Direction	Description
<code>axi_awaddr [6:0]</code>	Input	AXI4-Lite write address bus.
<code>axi_awvalid</code>	Input	AXI4-Lite write address valid strobe.
<code>axi_awready</code>	Output	AXI4-Lite write address ready signal.
<code>axi_wdata [31:0]</code>	Input	AXI4-Lite write data.
<code>axi_wvalid</code>	Input	AXI4-Lite write data valid strobe.
<code>axi_wready</code>	Output	AXI4-Lite write-ready signal.
<code>axi_bvalid</code>	Output	AXI4-Lite write response valid strobe.
<code>axi_bready</code>	Input	AXI4-Lite write response ready signal.
<code>axi_araddr [6:0]</code>	Input	AXI4-Lite read address bus.
<code>axi_arvalid</code>	Input	AXI4-Lite read address valid strobe.
<code>axi_arready</code>	Output	AXI4-Lite read address ready signal.
<code>axi_rdata [31:0]</code>	Output	AXI4-Lite read data.
<code>axi_rvalid</code>	Output	AXI4-Lite read data valid strobe.
<code>axi_rready</code>	Input	AXI4-Lite read data ready signal.

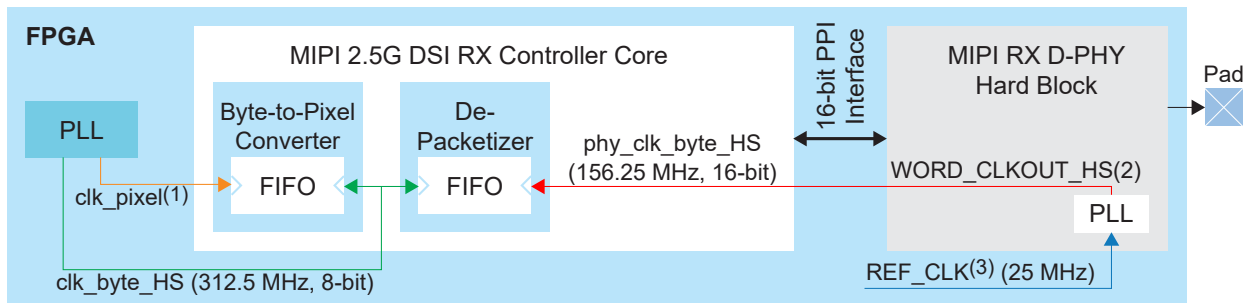
Table 8: Debug Interface

Port	Direction	Description
mipi_debug_out[31:0]	Output	<p>Debug port. Present if the parameter MIPI_DSI_RX_DEBUG is Enabled. The following is a list of internal signals that can be monitored.</p> <ul style="list-style-type: none"> [0] = pixel_fifo_full [1] = pixel_fifo_empty [2] = receive_error [3] = init_done (initialization completed, after tINIT_NS expired) [4] = hs_cmdfifo_full [5] = lp_cmdfifo_full [6] = lp_dcs_rfifo_full [7] = hs_cmdfifo_empty [8] = lp_cmdfifo_empty [9] = lp_dcs_rfifo_empty [10] = lp_cmd_in_progress [11] = hs_crc_error [12] = hs_ecc_1bit_error [13] = hs_ecc_2bit_error [14] = lp_crc_error [15] = lp_ecc_1bit_error [16] = lp_ecc_2bit_error [17] = RxErrSotSynchHS_0 [18] = RxErrControl_0 [19] = RxErrEsc_0 [20] = RxStopState_0 [21] = RxSkewCalHS_0 [22] = RxUlpsActiveNot_0 [23] = RxUlpsEsc_0 [27:24] = RxTriggerEsc [31:28] = reserved
mipi_debug_in[31:0]	Input	<p>Debug ports. Present if the parameter MIPI_DSI_RX_DEBUG is Enabled. Currently, no function has been implemented. Synchronized all input bits to zero.</p> <ul style="list-style-type: none"> [31:0] = reserved

Clocking

The following diagram illustrates the example of clock settings for a 2.5 Gbps DSI RX implementation.

Figure 2: 2.5 Gbps DSI RX Clocking Example



- (1) Refer to Pixel Clock Calculation section for the `clk_pixel` value.
- (2) **HS Receive Byte/Word Clock Pin Name** setting in the Interface Designer.
- (3) **Reference Clock** setting in the Interface Designer. You can select the source to be from either the core, GPIO, or PLL.

Register Definition

Table 9: Control Status Registers

Word Offset	Bits	Name	R/W	Width (bits)
0x00	16:0	Interrupt status register.	Bit[10:4],[1:0] = RO Bit[16:11],[3:2] = R/W1C	17
0x04	16:0	Interrupt enable register.	R/W	17
0x08	6:0	RX DPHY data lane 0 status.	RO	7
0x0C	6:0	RX DPHY data lane 1 status.	RO	7
0x10	6:0	RX DPHY data lane 2 status.	RO	7
0x14	6:0	RX DPHY data lane 3 status.	RO	7
0x18 - 0x24	N/A	Reserved.	N/A	N/A
0x28	1:0	RX DPHY clock lane status.	RO	2
0x2C	31:0	Received high speed write command/payload for short/longpacket (hs_cmdfifo_data).	RO	32
0x30	7:0	Received low power write command/payload for short/longpacket (lp_cmdfifo_data).	RO	8
0x34	31:0	LPDTread command data return. Write into this register to transmit the data return from peripheral (DSI RX) to host (DSI TX) (lp_dcs_rfifo_data).	WO	32
0x38	23:0	ESC mode LPDT command.	WO	24
0x2C - 0x40	N/A	Reserved.	N/A	N/A
0x44	15:0	Horizontal sync active (HSA) in pixel count. Only write to this register when it is sync pulse mode. minimum = 2	R/W	16
0x48	15:0	Horizontal black porch (HBP) in pixel count. For burst event mode, factor in the HSA value into the HBP value (not in used as per current implementation).	R/W	16
0x4C	15:0	Horizontal front porch (HFP) in byte (not in used as per current implementation).	R/W	16
0x50	7:0	Vertical sync active (VSA) in line. The minimum number of lines is 1.	R/W	8
0x54	7:0	Vertical black porch (VBP) in line. The minimum number of lines is 1 (not in used as per current implementation).	R/W	8
0x58	7:0	Vertical front porch (VFP) in line. The minimum number of lines is 2 (not in used as per current implementation).	R/W	8

Table 10: Interrupt Status Register Definition (0x00)

Bit	Name	Description
0	pixel_fifo_full	Pixel FIFO full. Pixel FIFO in the byte-to-pixel converter module is full.

Bit	Name	Description
1	pixel_fifo_empty	Pixel FIFO empty. Pixel FIFO in the byte-to-pixel converter module is empty.
2	undersize_pkt_error	Undersize packet error. The incoming MIPI HS data byte is lesser than the word count value.
3	receive_error	Initialization error MIPI HS data is received before t_{Init} is completed.
4	hs_cmdfifo_full	HS command fifo is full.
5	lp_cmdfifo_full	LP command fifo is full.
6	lp_dcs_rfifo_full	LP DCS read data fifo is full.
7	hs_cmdfifo_empty	HS command fifo is empty.
8	lp_cmdfifo_empty	LP command fifo is empty.
9	lp_dcs_rfifo_empty	LP DCS read data fifo is empty.
10	lp_cmd_in_progress	LP command transmission in LP lane is in progress.
11	hs_crc_error	HS packet receives CRC error indicator.
12	hs_ecc_1bit_error	HS packet receives ECC 1-bit error indicator.
13	hs_ecc_2bit_error	HS packet receives ECC 2-bit error indicator.
14	lp_crc_error	LP packet receives CRC error indicator.
15	lp_ecc_1bit_error	LP packet receives ECC 1-bit error indicator.
16	lp_ecc_2bit_error	LP packet receives ECC 2-bit error indicator.

Table 11: Interrupt Enable Register Definition (0x04)

Each enabled interrupt status bit is aggregated to the irq output port as an indicator. By default, all interrupt-enabled registers are set to 1'b0 (disabled).

Bit	Name	Description
0	pixel_fifo_full	Enables interrupt generation for pixel_fifo_full status register.
1	pixel_fifo_empty	Enables interrupt generation for pixel_fifo_empty status register.
2	undersize_pkt_error	Enables interrupt generation for undersize_pkt_error status register.
3	receive_error	Enables interrupt generation for receive_error status register.
4	hs_cmdfifo_full	Enables interrupt generation for hs_cmdfifo_full status register.
5	lp_cmdfifo_full	Enables interrupt generation for lp_cmdfifo_full status register.
6	lp_dcs_rfifo_full	Enables interrupt generation for lp_dcs_rfifo_full status register.
7	hs_cmdfifo_empty	Enables interrupt generation for hs_cmdfifo_empty status register.
8	lp_cmdfifo_empty	Enables interrupt generation for lp_cmdfifo_empty status register.
9	lp_dcs_rfifo_empty	Enables interrupt generation for lp_dcs_rfifo_empty status register.
10	lp_cmd_in_progress	Enables interrupt generation for LP command transmission in progress.
11	hs_crc_error	Enables interrupt generation for CRC error during hs packet reception.
12	hs_ecc_1bit_error	Enables interrupt generation for ECC 1-bit error during hs packet reception.
13	hs_ecc_2bit_error	Enables interrupt generation for ECC 2-bit error during hs packet reception.
14	lp_crc_error	Enables interrupt generation for CRC error during lp packet reception.
15	lp_ecc_1bit_error	Enables interrupt generation for ECC 1-bit error during lp packet reception.
16	lp_ecc_2bit_error	Enables interrupt generation for ECC 2-bit error during lp packet reception.

Table 12: D-PHYStatus for Data Lanes Register Definition (0x08 - 0x24)

Bit	Name	Description
0	RxErrSotSyncHS	Start-of-transmission (SoT) synchronization error. The core asserts this signal high for one cycle of phy_clk_byte_HS if the HS SoT leader sequence is corrupted in a way that proper synchronization cannot be expected.
1	RxErrControl	Control error. The core asserts this signal high when an incorrect line state sequence is detected in LP modes. Once asserted, this signal remains asserted until the next transaction starts, so that the protocol can properly process the error.
2	RxErrEsc	Escape entry error. The core asserts this signal high if an unrecognized escape entry command is received in LP mode. Once asserted, this signal remains asserted until the next transaction starts, so that the protocol can properly process the error.
3	RxStopState	Lane is in stop state.
4	Reserved	Reserved.
5	RxUlpsActiveNot	Ultra low power state (ULPS) (not) active. The core deasserts this signal low to indicate that the data lane is in ULP state.
6	RxUlpsEsc	Escape ULPS (receive) mode. The core asserts this signal high to indicate that the lane module has entered the ULPS because of the detection of a received ULPS command. The lane module remains in this mode with RxUlpsEsc asserted until a stop state is detected on the interconnect lane.
7	Reserved	Reserved.

Table 13: D-PHYStatus for Clock Lane Register Definition (0x28)

Bit	Name	Description
0	RxUlpsActiveClkNot	ULPS (not) active. The core asserts this signal high to indicate that the clock lane is in ULPS.
1	RxUlpsClkNot	Receive ULPS on clock lane. The core deasserts this signal low to indicate that the clock lane module has entered the ULPS because of the detection of a request to enter the ULPS. The lane module remains in this mode with RxUlpsClkNot asserted until a stop state is detected on the interconnect lane.

Video Mode Configuration

The MIPI 2.5G DSI RX Controller core supports the following video modes:

- Non-burst with sync pulses
- Non-burst with sync events
- Burst mode

Command Packet Data Types

The following table describes the supported command packet data types. The command packets are sent through the command register. In LP mode, the command packets are sent through lane 0. Use the command to send non-video packets to the DSI RX Controller.

Table 14: Command Packet Data Types

Type	Data Type	Packet Size	Transfer Mode
0x2	Color mode off command	Short	LP/HS
0x12	Color mode on command	Short	LP/HS
0x22	Shutdown peripheral command	Short	LP/HS
0x32	Turn on peripheral command	Short	LP/HS
0x3	Generic short write, no parameter	Short	LP/HS
0x13	Generic short write, 1 parameter	Short	LP/HS
0x23	Generic short write, 2 parameters	Short	LP/HS
0x4	Generic short read, no parameter	Short	LP/HS
0x14	Generic short read, 1 parameter	Short	LP/HS
0x24	Generic short read, 2 parameters	Short	LP/HS
0x5	DCS short write, no parameter	Short	LP/HS
0x15	DCS short write, 1 parameter	Short	LP/HS
0x6	DCS read	Short	LP/HS
0x37	Set max return packet size	Short	LP/HS
0x29	Generic long write	Long	LP/HS
0x39	DCS long write	Long	LP/HS

Sync Event Packet Data Type

Sync events are short packets and can accurately represent events like the start and end of sync pulses.

Table 15: Sync Events

Data type	Description	Packet Size
0x1	V sync start	Short
0x11	V sync end	Short
0x21	H sync start	Short
0x31	H sync end	Short

Video Mode Pixel Encoding

Table 16: Video Mode Pixel Encoding

TYPE[5:0]	Data Type	Bits per Pixel	Pixels per Pixel Clock	Bytes	Pack Bits	Packet Size	Transfer Mode
0xC	20-bit YCbCr	24	2	6	48	Long	HS
0x1C	24-bit YCbCr	24	2	6	48	Long	HS
0x2C	16-bit YCbCr	16	4	8	64	Long	HS
0x3D	12-bit YCbCr	12	4	6	48	Long	HS
0xE	RGB565	16	4	8	64	Long	HS
0x2E	RGB666 (24-bit)	24	2	6	48	Long	HS
0x3E	RGB888	24	2	6	48	Long	HS
0x0D	RGB101010	30	2	60/8	60	Long	HS

MIPI Video Data DATA[63:0] Formats

The format depends on the data type. New data arrives on every pixel clock.

Table 17: Loosely Packed Pixel Stream, 20-bit YCbCr (2-pixels per clock)

63	48	47							0
0		Pixel 1 and Pixel 2							
N/A	[47:38] Y	[37:36] 2'b00	[35:26] Cr	[25:24] 2'b00	[23:14] Y	[13:12] 2'b00	[11:2] Cb	[1:0] 2'b0	

Table 18: Packed Pixel Stream, 24-bit YCbCr (2-pixels per clock)

63	48	47					0
0		Pixel 1 and Pixel 2					
N/A	[47:36] Y	[35:24] Cr	[23:12] Y	[11:0] Cb			

Table 19: Packed Pixel Stream, 16-bit YCbCr (4-pixels per clock)

63	32			31				0
Pixel 3 and Pixel 4				Pixel 1 and Pixel 2				
[63:56] Y	[55:48] Cr	[47:40] Y	[39:32] Cb	[31:24] Y	[23:16] Cr	[15:8] Y	[7:0] Cb	

Table 20: Packed Pixel Stream, 12-bit YCbCr (4-pixels per clock)

63	48	47	24		23			0
Odd Lines								
0		Pixel 3 and Pixel 4			Pixel 1 and Pixel 2			
N/A	[47:40] Y	[39:32] Y	[31:24] Cb	[23:16] Y	[15:8] Y	[7:0] Cb		
Even Lines								
0		Pixel 3 and Pixel 4			Pixel 1 and Pixel 2			
N/A	[47:40] Y	[39:32] Y	[31:24] Cr	[23:16] Y	[15:8] Y	[7:0] Cb		

Table 21: RGB565 (4-pixels per clock)

63			48			47			32			31			16			15			0		
Pixel 4			Pixel 3			Pixel 2			Pixel 1														
[63:59]	[58:53]	[52:48]	[47:43]	[42:37]	[36:32]	[31:27]	[26:21]	[20:16]	[15:11]	[10:5]	[4:0]												
Blue	Green	Red	Blue	Green	Red	Blue	Green	Red	Blue	Green	Red												

Table 22: RGB666 (2-pixels per clock)

63			48			47			24			23			0		
0			Pixel 2						Pixel 1								
N/A			[47:42]	[41:40]	[39:34]	[33:32]	[31:26]	[25:24]	[23:18]	[17:16]	[15:10]	[9:8]	[7:2]	[1:0]			
			Blue	2'b00	Green	2'b00	Red	2'b00	Blue	2'b00	Green	2'b00	Red	2'b00			

Table 23: RGB888 (2-pixels per clock)

63			48			47			24			23			0		
0			Pixel 2						Pixel 1								
N/A			[47:40]	[39:32]	[31:24]	[23:16]	[15:8]	[7:0]									
			Blue	Green	Red	Blue	Green	Red									

Table 24: RGB101010 (2-pixels per clock)

63			60			59			30			29			0		
0			Pixel 2						Pixel 1								
N/A			[59:50]	[49:40]	[39:30]	[29:20]	[19:10]	[9:0]									
			Blue	Green	Red	Blue	Green	Red									

Pixel Clock Calculation

The following formula calculates the pixel clock frequency that you need to drive the pixel clock input port, `clk_pixel`.

$$\text{PIX_CLK_MHZ} \geq (\text{DATARATE_MBPS} * \text{NUM_DATA_LANE}) / \text{PACK_BIT},$$

where:

- `PIX_CLK_MHZ` is the pixel clock in MHz
- `DATARATE_MBPS` is the MIPI data rate in Mbps
- `NUM_DATA_LANE` is the number of data lanes
- `PACK_BIT` is the pixel data bits per pixel clock from **Video Mode Pixel Encoding** on page 18.

A FIFO for data transfer is available from the byte clock domain to the pixel clock domain. The pixel clock frequency must be applied approximately equal to the presented formula (or within 100% to 150% of the calculated value). If the pixel clock frequency is too low, a FIFO overflow occurs (overflow happens when the reading clock is slower than the writing clock). Contrastingly, if the pixel clock frequency is too high, a FIFO underflow occurs (which causes the `pixel_data_valid` to toggle within a one-pixel line).

Video Timing Parameters

The following waveforms show the video interface signals relationship.

Figure 3: Video Timing Waveform (Horizontal)

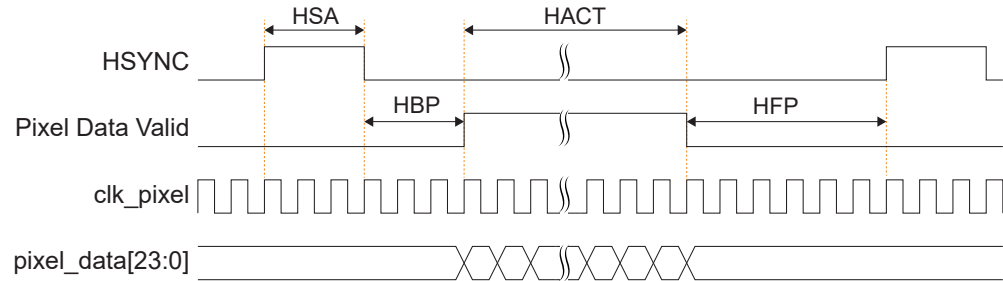


Figure 4: Video Timing Waveform (Vertical)

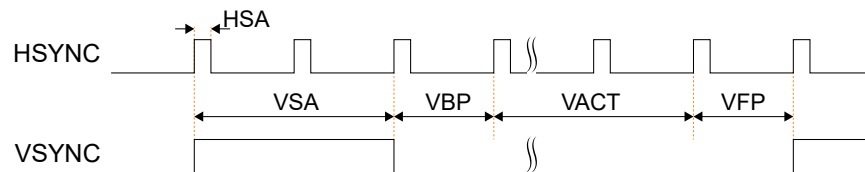


Table 25: MIPI Video Timing Parameters

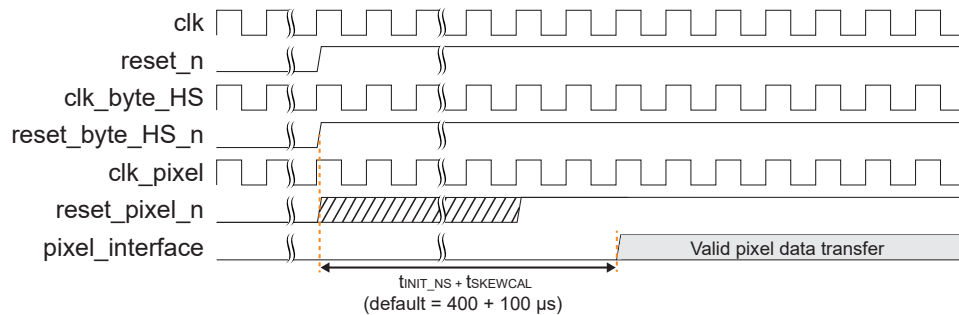
Parameter	Definition
tLine	Total horizontal line period.
HACT	Total number of actual pixels per line.
VACT	Total number of actual pixels horizontal line per frame.
HSA	HSYNC pulse width.
HBP	Horizontal back porch.
HFP	Horizontal front porch.
VSA	VSYNC pulse width.
VBP	Vertical back porch.
VFP	Vertical front porch.
Pixel clock	Video stream pixel clock frequency in MHz.
MIPI speed	DSI RX MIPI speed in Mbps.
No. data lane	Number of MIPI data lane.

Reset Sequence and Initialization

During the initial power-up state, an initialization time, t_{INIT_NS} , of $400\ \mu s$ is the minimum requirement for the MIPI D-PHY receiver to function properly before an LP/HS data transfer ($400\ \mu s$ of initialization time is required when Efinix MIPI hard D-PHY transmitter is used, you can adjust this timing depending on the other transmitter's specifications). Another $100\ \mu s$ of initial skew calibration time is needed (if the initial skew calibration feature is enabled) after the `INIT` time expires. Typically, `reset_pixel_n` can be deasserted at any time after the other reset domains are deasserted, but it should be done before the end of the initialization and skew calibration processes. The output of a valid pixel data can be decoded from the PPI interface after the initialization process and skew calibration processes are completed. For any reset assertion event during user mode, you must follow the reinitialization sequence for the reset deassertion procedure.

The following figure describes the reset sequencing and initialization requirements.

Figure 5: Reset Sequence and Initialization



IP Manager

The Efinity® IP Manager is an interactive wizard that helps you customize and generate Efinity® IP cores. The IP Manager performs validation checks on the parameters you set to ensure that your selections are valid. When you generate the IP core, you can optionally generate an example design targeting an Efinity development board and/or a testbench. This wizard is helpful when you use several IP cores, multiple instances of an IP core with different parameters, or the same IP core across different projects.



Note: Not all Efinity IP cores include an example design or a testbench.

Generating the MIPI 2.5G DSI RX Controller Core with the IP Manager

The following steps explain how to customize an IP core with the IP Configuration wizard.

1. Open the IP Catalog.
2. Choose **MIPI > MIPI 2.5G DSI RX Controller** core and click **Next**. The **IP Configuration** wizard opens.
3. Enter the module name in the **Module Name** box.



Note: You cannot generate the core without a module name.

4. Customize the IP core using the options shown in the wizard. For detailed information on the options, refer to the *Customizing the MIPI 2.5G DSI RX Controller* section.
5. (Optional) In the **Deliverables** tab, specify whether to generate an IP core example design targeting an Efinity® development board and/or testbench. These options are turned on by default.
6. (Optional) In the **Summary** tab, review your selections.
7. Click **Generate** to generate the IP core and other selected deliverables.
8. In the **Review configuration generation** dialog box, click **Generate**. The Console in the **Summary** tab shows the generation status.



Note: You can disable the **Review configuration generation** dialog box by turning off the **Show Confirmation Box** option in the wizard.

9. When generation finishes, the wizard displays the **Generation Success** dialog box. Click **OK** to close the wizard.

The wizard adds the IP to your project and displays it under **IP** in the Project pane.

Generated Files

The IP Manager generates these files and directories:

- **<module name>_define.svh**—Contains the customized parameters.
- **<module name>_tpl.sv**—Verilog HDL instantiation template.
- **<module name>_tpl.vhd**—VHDL instantiation template.
- **<module name>.sv**—IP source code.
- **settings.json**—Configuration file.
- **<kit name>_devkit**—Has generated RTL, example design, and Efinity® project targeting a specific development board.
- **Testbench**—Contains generated RTL files for a specific simulator.

Customizing the MIPI 2.5G DSI RX Controller

Table 26: MIPI 2.5G DSI RX Controller Core Parameter

Name	Option	Description
tINIT (ns)	Values according to MIPI D-PHY specifications	Initialization time for the MIPI DSI RX controller in ns. Default: 400,000
Data Lanes	1, 2, 4	Number of data lanes. Default: 4
IP Core Clock Frequency	40 - 100	IP core clock frequency in MHz. Default: 100
Pack Type 60	Enable, Disable	Turn on pack 60-bit datatype, for example, RGB101010. Default: Disable
Pack Type 48	Enable, Disable	Turn on pack 48-bit datatype, for example, 20-bit YCbCr, 24-bit YCbCr, 12-bit YCbCr, RGB666 (24-bit), or RGB888. Default: Enable
Pack Type 64	Enable, Disable	Turn on pack 64-bit datatype, for example, 16-bit YCbCr, or RGB565. Default: Disable
Number of Asynchronous Register Stages	2 - 8	Cross clock domain control signal synchronization stage. Default: 2
PPI Interface Data Width	16	HS mode data width. Default: 16 (Fixed)
Video Transmission Packet Sequences	0, 1, 2	Select video mode: 0: Non-Burst Mode with Sync Pulses 1: Non-Burst Mode with Sync event (default) 2: Burst Mode
Pixel Data FIFO Depth	256 - 8192	FIFO depth size to store the pixel packet data (need to be set to a power of 2 value). Minimum FIFO depth required > horizontal_pixel (HACT) x bits_per_pixel / 64 Default: 2048
High speed Command Write Data FIFO Depth	8 - 2048	HS command write data FIFO depth. Default: 64
Low Power Write Data FIFO Depth	8 - 2048	LP command write data FIFO depth. Default: 64
Low Power Read Data FIFO Depth	8 - 2048	Bus turnaround read data depth. Default: 64
MIPI_DSI_RX_DEBUG	Enable, Disable	Enable debug ports for internal signal observation and monitoring. Default: Disable

MIPI 2.5G DSI RX Controller Example Design

You can choose to generate the example design when generating the core in the IP Manager Configuration window. Compile the example design project and download the **.hex** or **.bit** file to your board.

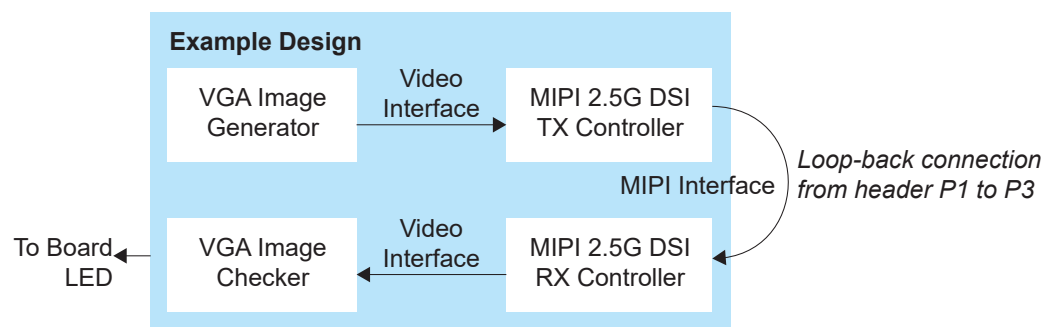


Important: Efinix tested the example design generated with the default parameter options only.

The example design targets the Titanium Ti180 J484 Development Board. The design instantiates both MIPI 2.5G DSI TX and RX Controller cores. This design requires a QSE header-compatible cable.

The example design generates an image and sends the VGA data to the MIPI 2.5G DSI TX Controller core. The data is then sent through a hardware loopback on the board using a 4-lane MIPI interface to the MIPI 2.5G DSI RX Controller core. The VGA checker compares the data received with the one created by the VGA generator, and outputs the results using the Titanium Ti180 J484 Development Board LEDs.

Figure 6: MIPI 2.5G DSI RX Controller Core Example Design



After programming the bitstream file into the development board, the onboard LED:

- LED2—Blinks continuously indicating there are traffic being sent over to MIPI IP.
- LED3—Turns on to indicate that the hsync signal matches TX and RX.
- LED4—Turns on to indicate that the vsync signal matches TX and RX.
- LED5—Turns on to indicate that the pixel data signal matches TX and RX.
- LED6— turns on to indicate that the pixel data valid signal matches TX and RX.



Note: You can use the **Titanium MIPI Utility-v<version>.xism** to check if your design will work. Enter the related design information, then verify whether your selections pass various tests. You can download the Titanium MIPI utility from the Design Support page in the [Support Center](#).

Revision History

Table 27: Revision History

Date	Document Version	IP Version	Description
March 2026	1.0	1.0	Initial release.