



# MIPI 2.5G DSI TX Controller Core User Guide

---

UG-CORE-MIPI-2-5G-DSI-TX-v1.0  
March 2026  
[www.efinixinc.com](http://www.efinixinc.com)



# Contents

<b>Introduction</b> .....	<b>3</b>
<b>Features</b> .....	<b>3</b>
<b>Device Support</b> .....	<b>3</b>
<b>Resource Utilization and Performance</b> .....	<b>4</b>
<b>Release Notes</b> .....	<b>4</b>
<b>Functional Description</b> .....	<b>5</b>
Ports.....	6
Clocking.....	10
Register Definition.....	11
Video Mode Configuration.....	14
Command Packet Data Types.....	14
Sync Event Packet Data Type.....	15
Video Mode Pixel Encoding.....	15
MIPI Video Data DATA[63:0] Formats.....	16
Pixel Clock Calculation.....	17
Video Timing Parameters.....	18
Reset Sequence and Initialization.....	19
<b>IP Manager</b> .....	<b>20</b>
<b>Customizing the MIPI 2.5G DSI TX Controller</b> .....	<b>21</b>
<b>MIPI 2.5G DSI TX Controller Example Design</b> .....	<b>23</b>
Loopback.....	23
Colorbar Display.....	24
<b>Revision History</b> .....	<b>25</b>

# 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<sup>(1)</sup>. 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 TX 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)<sup>(2)</sup>
- 16-bit HS mode data width
- Includes AXI4-Lite interface for register access
- Error correction code (ECC) generation for packet headers
- Cyclic redundancy check (CRC) generation 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 TX Controller Core Device Support*

FPGA Family	Supported Device
Trion	-
Titanium	Refer to the <a href="#">Titanium Selector Guide</a>
Topaz	Refer to the <a href="#">Topaz Selector Guide</a>

<sup>(1)</sup> Source: MIPI Alliance.

<sup>(2)</sup> 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 TX Controller with 4 data lanes.

FPGA	Flip-Flop	LUT	RAM	DSP	$f_{MAX}$ (MHz) <sup>(3)</sup>					Efinity® Version <sup>(4)</sup>
					clk_esc	axi_clk	clk_byte_HS	phy_clk_byte_HS	clk_pixel	
Ti180 J484 C4	1,527	2,919	28	0	156	275	336	217	259	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.

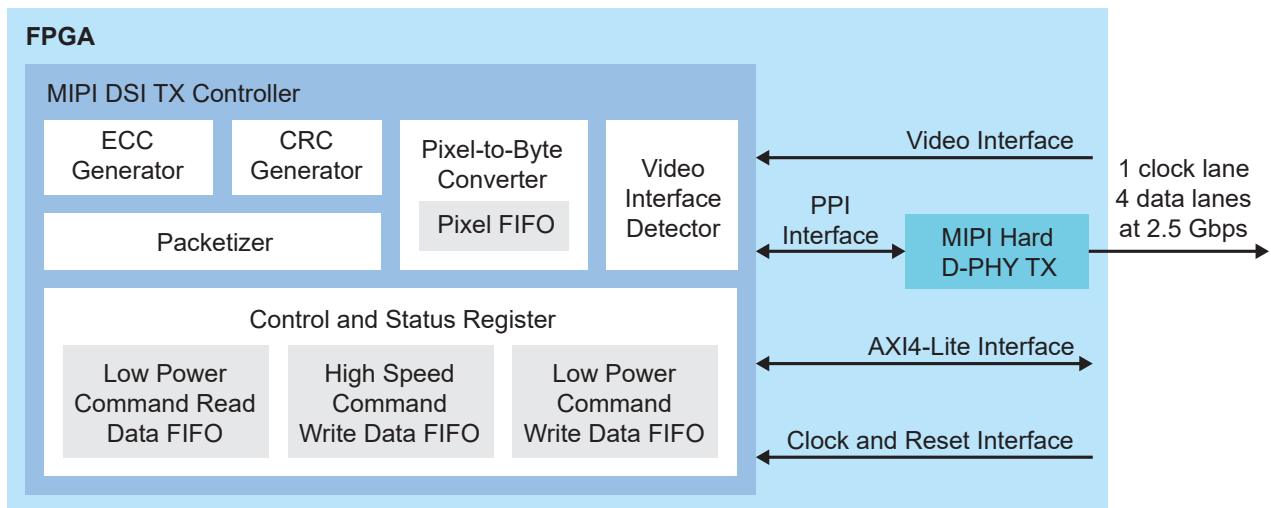
<sup>(3)</sup> Using default parameter settings.

<sup>(4)</sup> Using System Verilog.

# Functional Description

The MIPI 2.5G DSI TX Controller core consists of a control status registers, ECC generator, CRC generator, packetizer, pixel-to-byte converter, and video interface detector. The core has a video interface which is running at pixel clock domain, AXI4-lite interface, clock and reset interface, and PPI interface which is running at MIPI byte clock domain. The DSI TX controller communicates with the MIPI hard D-PHY in the core device through the PPI interface.

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



## Ports

**Table 3: Clock and Reset Ports**

Port	Direction	Description
clk_esc	Input	Transmit escape mode clock (~20 MHz).
reset_esc_n	Input	Transmit escape mode reset signal.
clk_byte_HS	Input	MIPI TX 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 TX 8-bit parallel data reset signal.
phy_clk_byte_HS	Input	MIPI TX 16-bit parallel data clock signal. MIPI data rate / 16-bit data width. Supplied by MIPI hard D-PHY. clk_byte_HS = 2 * phy_clk_byte_HS
RxClkEsc	Input	Receive escape mode clock. Supplied by MIPI hard D-PHY.
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
TxUlpsClk	Output	Transmit ULPS on MIPI hard D-PHY clock lane. This active-high signal is asserted to trigger a clock lane to enter the ULPS. It is clocked with clk_esc.
TxUlpsExitClk	Output	Transmit the ULPS exit sequence on MIPI hard D-PHY clock lane. This active-high signal is asserted when ULPS is active, and the protocol is ready to leave ULPS. It is clocked with clk_esc.
TxUlpsActiveClkNot	Input	ULPS (not) active. This active-low signal is asserted to indicate that the MIPI hard D-PHY clock lane is in ULPS. It is clocked with clk_esc.
TxUlpsEsc [NUM_DATA_LANE-1:0]	Output	Transmit ULPS escape mode. This active-high signal is asserted with TxRequestEsc to cause the MIPI hard D-PHY data lane to enter ULPS. It is clocked with clk_esc.
TxUlpsExit [NUM_DATA_LANE-1:0]	Output	Transmit the ULPS exit sequence. This active-high signal is asserted when ULPS is active, and the protocol is ready to leave ULPS on the MIPI hard D-PHY data lane. It is clocked with clk_esc.
TxUlpsActiveNot [NUM_DATA_LANE-1:0]	Input	ULPS (not) active. This active-low signal is asserted to indicate that the MIPI hard D-PHY data lane is in the ULPS. It is clocked with clk_esc.
TxRequestEsc [NUM_DATA_LANE-1:0]	Output	Transmit escape mode request. This active-high signal is used to request escape sequences on the MIPI hard D-PHY data lane. It is clocked with clk_esc.

Port	Direction	Description
TxStopStateD [NUM_DATA_LANE-1:0]	Input	Data lane in stop state. This is an asynchronous active-high signal from the MIPI hard D-PHY that indicates that the data lane is in a stop state.
TxStopStateC	Input	Clock lane is in a stop state. This is an asynchronous active-high signal from the MIPI hard D-PHY that indicates that the clock lane is in a stop state.
TxSkewCalHS [NUM_DATA_LANE-1:0]	Output	HS transmit skew calibration. This is an optional signal to initiate the periodic deskew burst at the transmitter. A low-to-high transition triggers the PHY to transmit a skew calibration pattern. A high-to-low transition triggers the MIPI hard D-PHY to end the transmission of a skew calibration pattern and initiate an end-of-transmission sequence.
TxReadyHS [NUM_DATA_LANE-1:0]	Input	HS transmit ready. This active-high signal indicates that the lane module has accepted TxDataHS for serial transmission.
TxRequestHS [NUM_DATA_LANE-1:0]	Output	A request for HS transmit valid data. A low-to-high transition triggers the lane module to initiate a start-of-transmission sequence. A high-to-low transition triggers the lane module to initiate an end-of-transmission sequence. This active-high signal indicates that the protocol is driving valid data that is ready to be transmitted on TxDataHS.
TxRequestHSc	Output	A request for HS transmit valid data. A low-to-high transition triggers the lane module to initiate a start-of-transmission sequence. A high-to-low transition triggers the lane module to initiate an end-of-transmission sequence. This active-high signal triggers the lane module to begin transmitting a HS clock.
TxDataHS <sub>n</sub> [HS_DATA_WIDTH-1:0]	Output	HS data to be transmitted for MIPI hard D-PHY data lane. <i>n</i> = Lane 0 to 3
TxReqValidHS <sub>n</sub> [1:0]	Output	High-speed transmission of valid word data. When the High-Speed Transmit Data width is greater than 8 bits, it is necessary to indicate which 8-bit segments contain valid transmit data to be able to transmit any number of words. <i>n</i> = Lane 0 to 3

**Table 5: Video Interface**

All signals are clocked by `clk_pixel` and `reset_pixel_n`.

Port	Direction	Description
hsync	Input	Active-low horizontal sync.
vsync	Input	Active-low vertical sync.
datatype [5:0]	Input	Data type of the HS packet. Sampled at hsync rising edge.
pixel_data [63:0]	Input	Video data. Sampled when <code>pixel_data_valid</code> is high. The actual width is dependent on pixel type. See <a href="#">Video Mode Pixel Encoding</a> on page 15.
pixel_data_valid	Input	Active-high pixel data enable. Once the TX valid signal goes high, the MIPI TX interface expects to receive pixel data every clock cycle until the entire line is sent. Additionally, the TX valid signal must remain high for the entire line to be sent.

Port	Direction	Description
haddr [15:0]	Input	Number of horizontal pixels is 16-bit. Sampled at hsync rising edge.
vc [1:0]	Input	2-bit virtual channel signal.

Table 6: Conduit Interface

Port	Direction	Description
TurnRequest_dbg	Input	User control turnaround request. This active high signal indicates that the protocol needs to initiate a bi-directional data lane turnaround, allowing the other side to begin transmissions. TurnRequest is valid on the rising edge of clk. TurnRequest is only meaningful for a bidirectional data lane module that is currently the transmitter ( <b>Direction = 0</b> ). If the bi-directional data lane module is in receive mode ( <b>Direction = 1</b> ), this signal is ignored. A low-to-high transition on TurnRequest can only happen when Stopstate is asserted.
TurnRequest_done	Output	Indicates that the RX D-PHY acknowledges the bus turnaround or timeout. If this signal is high together with turnaround timeout, it indicates that there is no acknowledgement from the RX on the turnaround request.
irq	Output	Interrupt signal for the interrupt status register.
Direction	Input	This signal indicates the current direction of lane 0 interconnect. When <b>Direction = 0</b> , lane 0 is in transmitting mode (0 = Output). When <b>Direction = 1</b> , lane 0 is in receiving mode (1 = Input).

Table 7: AXI4-Lite Interface

All signals are clocked with axi\_clk.

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

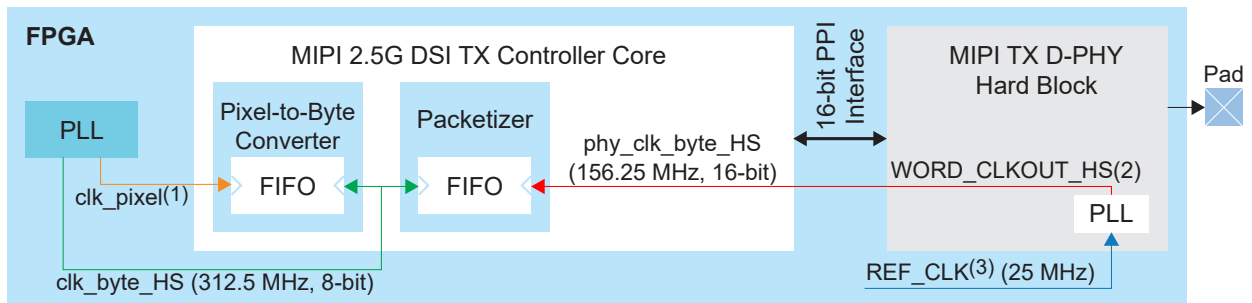
Table 8: Debug Interface

Port	Direction	Description
mipi_debug_out[31:0]	Output	<p>Debug port. Present if the parameter <b>MIPI_DSI_TX_DEBUG</b> is <b>Enabled</b>. The following is a list of internal signals that can be monitored.</p> <ul style="list-style-type: none"> <li>[0] = fifo_full</li> <li>[1] = fifo_empty</li> <li>[2] = non_support_longpkt</li> <li>[3] = turnaround_timeout</li> <li>[4] = ready_to_xmit (indicator for initialization done, prior to skewcal)</li> <li>[5] = init_skewcal_done (indicator for skewcal process completion)</li> <li>[6] = lp_dcs_rfifo_full</li> <li>[7] = lp_dcs_rfifo_empty</li> <li>[8] = lp_dcs_wfifo_full</li> <li>[9] = lp_dcs_wfifo_empty</li> <li>[10] = hs_dcs_wfifo_full</li> <li>[11] = hs_dcs_wfifo_empty</li> <li>[12] = lp_cmd_in_progress</li> <li>[13] = hs_cmd_in_progress</li> <li>[14] = TxStopState_0</li> <li>[15] = TxStopState_1</li> <li>[16] = TxStopState_2</li> <li>[17] = TxStopState_3</li> <li>[31:18] = Reserved</li> </ul>
mipi_debug_in[31:0]	Input	<p>Debug ports. Present if the parameter <b>MIPI_DSI_TX_DEBUG</b> is <b>Enabled</b>. Currently, no function has been implemented. Synchronize all input bits to zero.</p> <ul style="list-style-type: none"> <li>[31:0] = reserved</li> </ul>

## Clocking

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

Figure 2: 2.5 Gbps DSI TX Clocking Example



- (1) Refer to Pixel Clock Calculation section for the `clk_pixel` value.
- (2) **HS Transmit 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	4:0	Interrupt status register.	Bit[1:0],[4] - RO Bit[3:2] - R/W1C	5
0x04	4:0	Interrupt enable register.	R/W	5
0x08	7:0	PHY stop state status.	RO	8
0x0C	0	TxUlpsActiveClkNot.	RO	9
	8:1	TxUlpsActiveNot_7: TxUlpsActiveNot_0.		
0x10	7:0	Skew calibration high speed.	R/W	8
0x14	0	UlpsClk.	R/W	13
	8:1	UlpsEsc[7:0].		
	12:9	TxTriggerEsc.		
0x18	0	Reserved.	R/W	4
	1	Reserved.		
	2	Reserved.		
	3	video_stream_en. 1: Turn on HS video stream on the MIPI lane 0: Turn off HS video stream on the MIPI lane		
0x1C	HS Command Queue. Ensure that the bit <b>11 of the status register</b> is low before issuing the next command.		R/W	24
	7:0	datatype.		
	15:8	Parameter 1.		
	23:16	Parameter 2.		
0x20	LP Command Queue. Ensure that the bit <b>10 of the status register</b> is low before issuing the next command.		R/W	24
	7:0	datatype.		
	15:8	Parameter 1.		
	23:16	Parameter 2.		
0x24	19:0	Status register.	RO	20
0x28	31:0	Low power command write long data FIFO. You must write the LP write data to this FIFO before issuing the LP command packet to register 0x20. Store only one complete write packet data in the FIFO at a time.	WO	32

Word Offset	Bits	Name	R/W	Width (bits)
0x2C	31:0	High speed command write long data FIFO. You must write the HS write data to this FIFO before issuing the HS command packet to register 0x1C. Store only one complete write packet data in the FIFO. <sup>(5)</sup>	WO	32
0x30	7:0	Low power command read long data FIFO. The bus turnaround read data is pushed into this FIFO. Store only one complete read packet data in the FIFO at a time. The MIPI 2.5G DSI TX Controller stores all the return read data into the read FIFO and does not check whether the return read data matches maximum return packet size (MRPS).	RO	8
0x34, 0x38, 0x3C	Reserved.			
0x40	31:0	Total H line word count in byte.	R/W	32
0x44	15:0	Horizontal sync active (HSA) in byte. Only write to this register when it is sync pulse mode.	R/W	16
0x48	15:0	Horizontal black porch (HBP) in byte. For burst event mode, factor in HSA value into the HBP value.	R/W	16
0x4C	15:0	Horizontal front porch (HFP) in byte.	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.	R/W	8
0x58	7:0	Vertical front porch (VFP) in line. The minimum number of lines is 2.	R/W	8
0x5C	15:0	Vertical active (VACT) in line. The minimum number of lines is 1.	R/W	16

<sup>(5)</sup> The word count for a HS write long command data has to be larger or equal than the number of MIPI data lane (NUM\_DATA\_LANE).

**Table 10: 0x24 Status Register Definition**

Bit	Description
0	lp_dcs_rfifo_full. LP command read data FIFO full.
1	lp_dcs_rfifo_empty. LP command read data FIFO empty.
2	lp_dcs_wfifo_full. LP command write data FIFO full.
3	lp_dcs_wfifo_empty. LP command write data FIFO empty.
4	hs_dcs_wfifo_full. HS command write data FIFO full.
5	hs_dcs_wfifo_empty. HS command write data FIFO empty.
6	Reserved.
7	Reserved.
8	Reserved.
9	Reserved.
10	lp_cmd_in_progress. LP command transmission in LP lane is in progress.   <b>Note:</b> After writing LP cmd (addr: 0x20), lp_cmd_in_progress will not flag high immediately. You need to wait for at least 10 cycles of axi_clk prior to polling this indicator.
11	hs_cmd_in_progress. HS command transmission in HS lane is in progress.   <b>Note:</b> After writing HS cmd (addr: 0x1c), hs_cmd_in_progress will not flag high immediately. You need to wait for at least 10 cycles of axi_clk prior to polling this indicator.

**Table 11: 0x00 Interrupt Status Register Definition**

Bit	Description
0	Pixel FIFO full.
1	Pixel FIFO empty.
2	Unsupported video data type.
3	Turnaround timeout.

## Video Mode Configuration

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

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

The following table describes the configuration for each video mode based on blanking or low-power interval (BLLP) mode setting.

**Table 12: Video Mode Settings**

Mode	HSA	HBP	HFP	BLLP
Non-burst with Sync Pulse	HS Blank Packet	HS Blank Packet	HS Blank Packet	HS Blank Packet
Non-burst with Sync Event	HS Blank Packet	HS Blank Packet	HS Blank Packet	HS Blank Packet
Burst	HS Blank Packet	HS Blank Packet	LP-11	LP-11

## 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 display peripherals.

**Table 13: 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 14: 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 15: 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 16: Loosely Packed Pixel Stream, 20-bit YCbCr (2-pixels per clock)**

63	48	47					0
0	Pixel 1 and Pixel 2						
N/A	[47:44] 4'h0	[43:34] Y	[33:24] Cr	[23:20] 4'h0	[19:10] Y	[9:0] Cb	

**Table 17: 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 18: 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 19: 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 20: 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 21: RGB666 (2-pixels per clock)**

63		48			47			24			23			0		
0		Pixel 2			Pixel 1											
N/A		[47:42]	[41:36]	[35:30]	[29:24]	[23:18]	[17:12]	[11:6]	[5:0]							
		6'h0	Blue	Green	Red	6'h0	Blue	Green	Red							

**Table 22: 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 23: 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`.

$$\text{PIX\_CLK\_MHZ} < (\text{DATARATE\_MBPS} * \text{NUM\_DATA\_LANE}) / \text{PACK\_BIT},$$

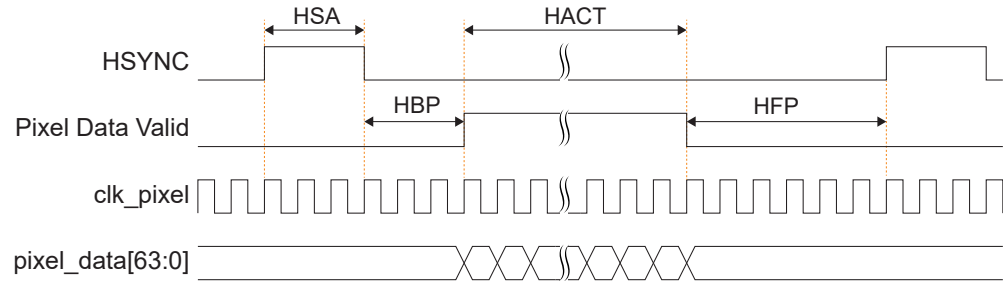
where:

- `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 15.

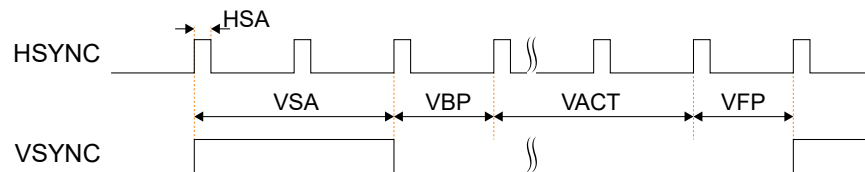
## Video Timing Parameters

The following waveforms show the video interface signals relationship.

**Figure 3: Video Timing Waveform (Horizontal)**



**Figure 4: Video Timing Waveform (Vertical)**



**Table 24: MIPI Video Timing Parameters**

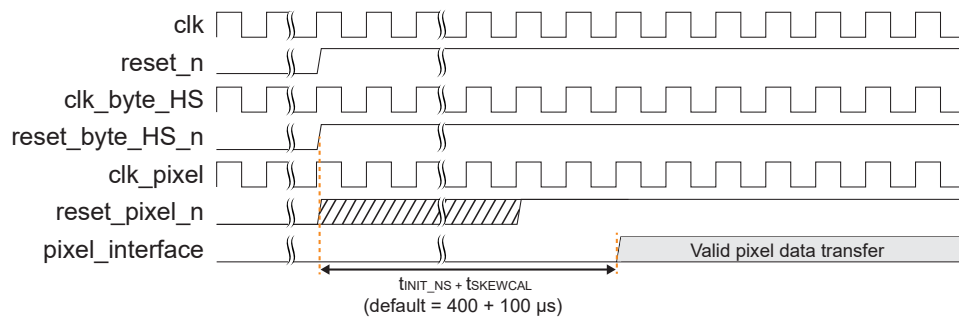
Parameter	Definition
tLine	Total horizontal line period.
HACT	Total number of actual pixels per line. To ensure no flaws in the pixel-to-byte conversion, apply the following formula, and the result must be a rounded number: $HACT \times \text{bits\_per\_pixel} / 64$
VACT	Total number of actual pixels 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 TX 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 transmitter to function properly before an LP/HS data transfer. Another  $100\ \mu s$  of initial skew calibration time is needed (if the initial skew calibration feature is enabled) after the `INIT` time expires. You must ensure that no traffic is sent to the pixel interface before  $t_{INIT\_NS}$  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 process. A valid pixel data can be decoded from the PPI interface after the initialization process is 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 TX 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 TX 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 TX 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 TX Controller

**Table 25: MIPI 2.5G DSI TX Controller Core Parameter**

Name	Option	Description
tINIT	Values according to MIPI D-PHY specifications	Initialization time for the MIPI DSI TX controller in ns. Default: 100,000
Initial tSKEWCAL	Values according to MIPI D-PHY specifications	Initial skew calibration time for the MIPI DSI TX controller in ns. Default: 100,000 Skew Calibration is carried out after initialization period.
Data Lanes	1, 2, 4	Number of data lanes. Default: 4
MIPI Parallel Clock Frequency	5 - 156	MIPI parallel clock (phy_clk_byte_HS) frequency in MHz to support data rate from 80 Mbps to 2,500 Mbps. Default: 156
DPHY Clock Mode	Continuous, Discontinuous	D-PHY clock mode. Default: Continuous
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
Maximum Horizontal Resolution	Values according to video display	Maximum horizontal pixel resolution. Default: 1080
MIPI_DSI_TX_DEBUG	Enable, Disable	Enable debug ports for internal signal observation and monitoring. Default: Disable
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

Name	Option	Description
High Speed Write Data FIFO DEPTH	8 - 2048 <sup>(6)</sup>	HS command writes data FIFO depth. Default: 64
Low Power Write Data FIFO DEPTH	8 - 2048 <sup>(6)</sup>	LP command writes data FIFO depth. Default: 64
Low Power Read Data FIFO DEPTH	8 - 2048 <sup>(6)</sup>	Bus turnaround reads data depth. Default: 64
Pixel Data FIFO DEPTH	256 - 8192 <sup>(6)</sup>	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
HS Data FIFO Depth	8 - 2048 <sup>(6)</sup>	PHY protocol interface high-speed data FIFO depth (for packetizer). Default: 512
Enable End Of Transmission Packet	Enable, Disable	Enables or disables the end-of-transmission packet. 1'b1: Enable 1'b0: Disable (Default)
Enable Init Skew Calibration	Enable, Disable	Enables or disables the initial skew calibration sequence. For data rate 1.5 Gbps and below, set this to disable. For data rate above 1.5 Gbps, set this to enable.
PPI Interface Data Width	16	HS mode data width. Default: 16 (Fixed)

<sup>(6)</sup> 2<sup>n</sup>, where n can be from 3 to 11.

# MIPI 2.5G DSI TX 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.



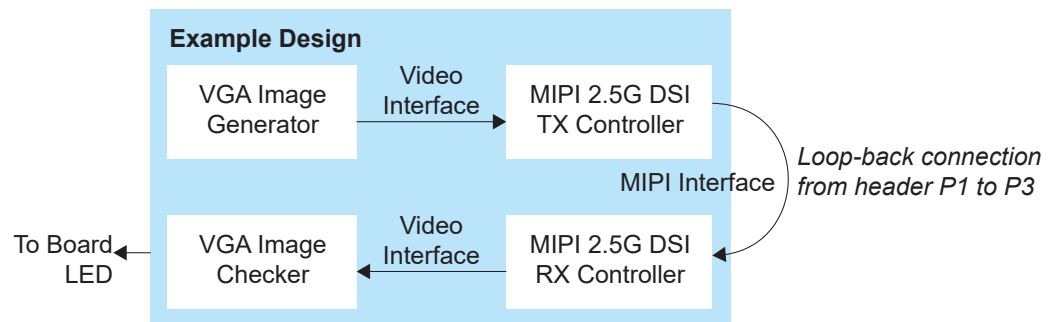
**Important:** Efinix tested the example design generated with the default parameter options only.

## Loopback

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 image checker through 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 image generator, and outputs the results using the Titanium Ti180 J484 Development Board LEDs.

*Figure 6: MIPI 2.5G DSI TX 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.



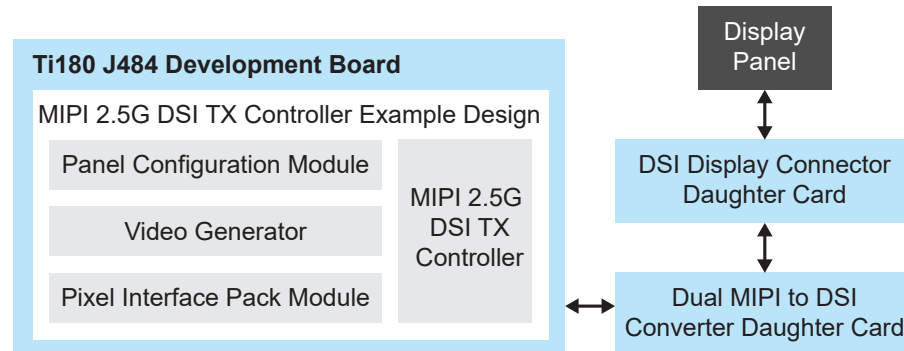
**Note:** You can use the **Titanium MIPI Utility-v<version>.xslm** to check if your design will work. Enter the related design information, then verify whether your selections pass various tests. You can download the Titanium MIPI utility from the Design Support page in the [Support Center](#).

## Colorbar Display

The example design targets the Titanium Ti180 J484 Development Board. This design generates a video stream and sends the video data to a display panel through the MIPI 2.5G DSI TX Controller. Apart from the Titanium Ti180 J484 Development Board, the example design requires the following hardware:

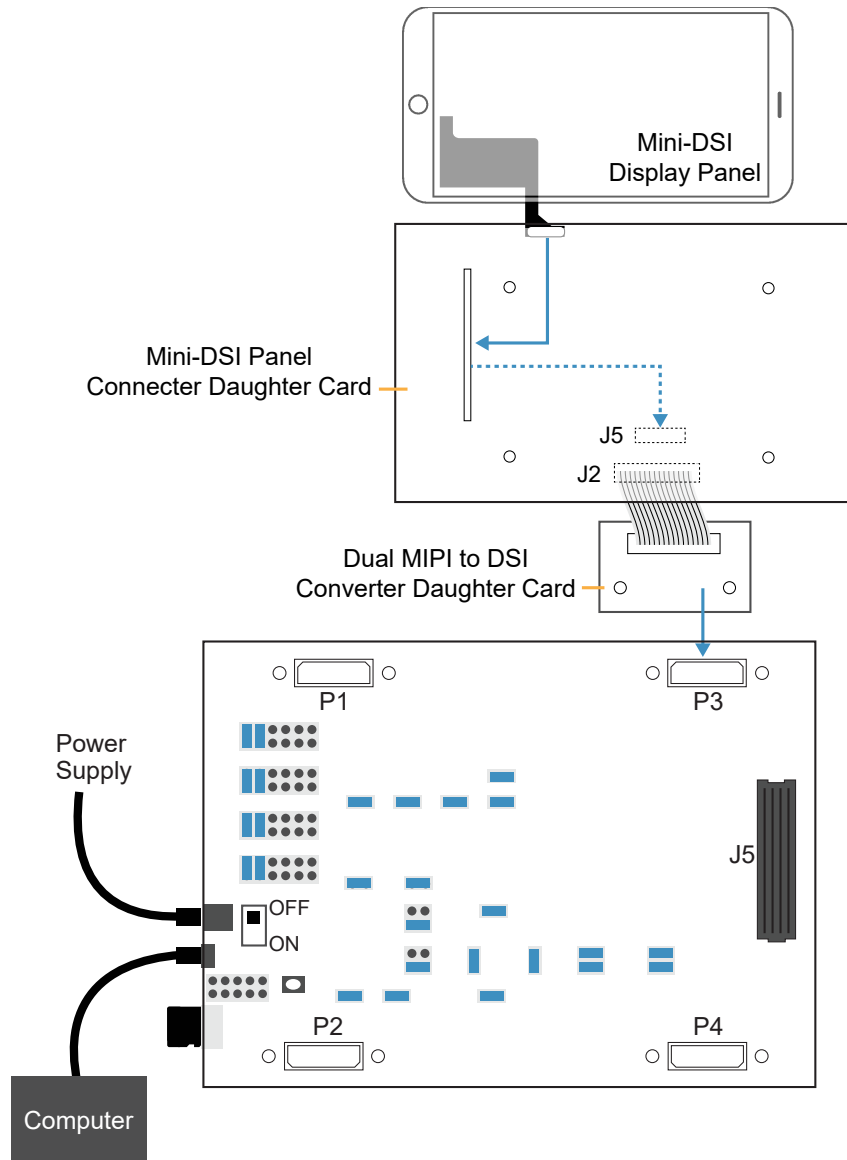
- Dual MIPI to DSI Converter Daughter Card
- Mini-DSI Panel Connector Daughter Card
- Mini-DSI Panel display

*Figure 7: MIPI 2.5G DSI TX Controller Core Example Design*



After power-up, program the example design bit file into the FPGA device and press the reset button (SW3), then you should be able to see a video (colorbar) displayed on the panel module.

Figure 8: MIPI 2.5G DSI TX Controller Example Design Design Set Up



**Note:** Ensure that R171 (default open) on Titanium Ti180 J484 Development Board is soldered with zero-ohm resistor. This is associated to the display panel reset signal.

## Revision History

Table 26: Revision History

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