



MIPI DSI RX Controller Core User Guide

UG-CORE-MIPI-DSI-RX-v1.8
June 2025
www.efinixinc.com



Contents

Introduction.....	3
Features.....	3
Device Support.....	3
Resource Utilization and Performance.....	4
Release Notes.....	4
Functional Description.....	5
Ports.....	6
Register Definition.....	10
Video Mode Configuration.....	13
Command Packet Data Types.....	13
Sync Event Packet Data Type.....	14
Video Mode Pixel Encoding.....	14
MIPI Video Data DATA[63:0] Formats.....	15
Pixel Clock Calculation.....	16
Video Timing Parameters.....	17
Reset Sequence and Initialization.....	18
IP Manager.....	19
Customizing the MIPI DSI RX Controller.....	20
MIPI DSI RX Controller Example Design.....	23
MIPI DSI RX Controller Testbench.....	25
Revision History.....	26

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, for example, display peripherals. The MIPI DSI RX Controller core implements the MIPI DSI interface in the FPGA and allows you to configure the related parameters.

Use the IP Manager to select IP, customize it, and generate files. The MIPI DSI RX Controller core has an interactive wizard to help you set parameters. The wizard also has options to create a testbench and/or example design targeting an Efinix[®] development board.

Features

- Supports 1, 2, and 4 lanes
- Supports continuous or discontinuous clock mode
- IP core clock frequency at 100 MHz
- 8-bit HS mode data width
- HS mode byte clock frequency from 10 MHz up to 187 MHz (from 80 Mbps up to 1,500 Mbps data rate)⁽²⁾
- 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 command transmission in HS or LP mode
- Supports all Titanium and Topaz FPGAs

Device Support

Table 1: MIPI DSI RX Controller Core Device Support

FPGA Family	Supported Device
Trion	-
Titanium and Topaz	All

⁽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 can change depending on the device resource utilization, design congestion, and user design.

Table 2: Titanium Resource Utilization and Performance

MIPI DSI RX Controller with 4 data lanes.

FPGA	Logic Elements (Logic, Adders, Flipflops,etc.)	Memory Blocks	DSP Blocks	f_{MAX} (MHz) ⁽³⁾				Efinity® Version ⁽⁴⁾
				clk	axi_clk	clk_byte_HS	clk_pixel	
Ti60 F225 C4	5,737 / 60,800 (9.44%)	35/256 (13.67%)	0/160 (0%)	205	252	227	261	2023.2.307.4.14

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.

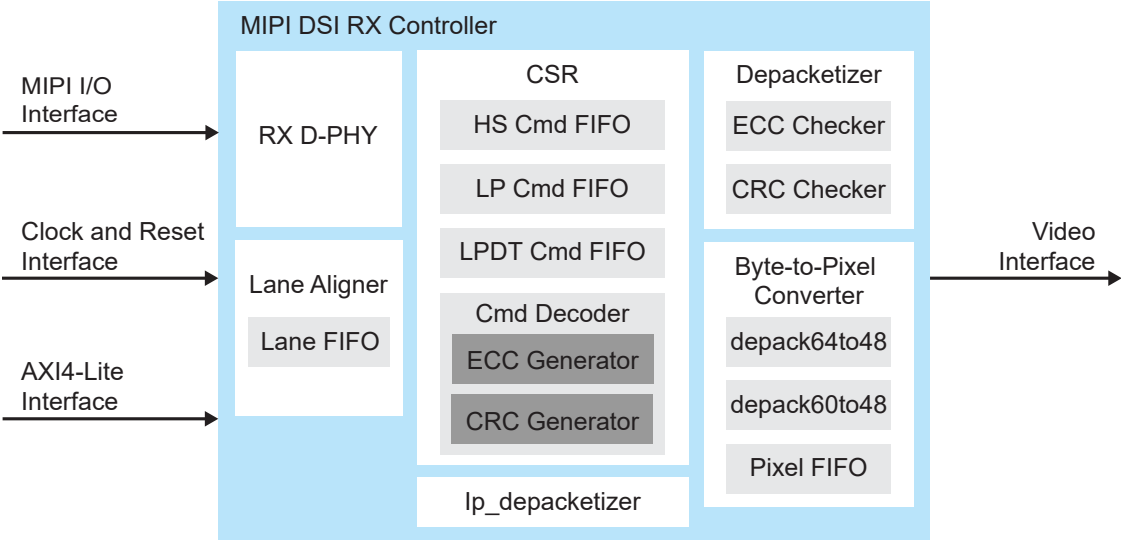
⁽³⁾ Using default parameter settings.

⁽⁴⁾ Using System Verilog.

Functional Description

The MIPI DSI RX Controller consists of a RX D-PHY block, control status registers, ECC and CRC checkers, packetizer, pixel-to-byte converter, and video interface detector. The core has a video, AXI4-lite, MIPI RX I/O, and clock and reset interfaces.

Figure 1: MIPI 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. 100 MHz.
reset_n	Input	IP core reset signal.
clk_byte_HS	Input	MIPI RX parallel clock. This clock is a HS mode reception clock.
reset_byte_HS_n	Input	MIPI RX parallel clock reset signal.
clk_pixel	Input	Pixel clock.
reset_pixel_n	Input	Pixel clock reset signal.
axi_clk	Input	AXI4-Lite interface clock. Recommended to use the same clock as the IP core clock for the LP command to function.
axi_reset_n	Input	AXI4-Lite interface reset.

Table 4: MIPI RX I/O interface

Port	Direction	Description
Rx_LP_CLK_P	Input	LP mode RX clock single-ended P signal.
Rx_LP_CLK_N	Input	LP mode RX clock single-ended N signal.
Rx_HS_enable_C	Output	Signal to enable HS mode clock lane.
LVDS_termen_C	Output	Signal to enable HS mode clock lane termination.
Rx_LP_D_P [NUM_DATA_LANE-1:0]	Input	LP mode RX data single-ended P signal.
Rx_LP_D_N [NUM_DATA_LANE-1:0]	Input	LP mode RX data single-ended N signal.
Rx_HS_D_0[7:0]	Input	HS mode differential lane data bus.
Rx_HS_D_1[7:0]	Input	HS mode differential lane data bus.
Rx_HS_D_2[7:0]	Input	HS mode differential lane data bus.
Rx_HS_D_3[7:0]	Input	HS mode differential lane data bus.
Rx_HS_enable_D[NUM_DATA_LANE-1:0]	Output	Signal to enable HS mode data lane.
LVDS_termen_D[NUM_DATA_LANE-1:0]	Output	Signal to enable HS mode data lane termination.
fifo_rd_enable[NUM_DATA_LANE-1:0]	Output	RX HS mode data lane FIFO read enable signal.
fifo_rd_empty[NUM_DATA_LANE-1:0]	Input	RX HS mode data lane FIFO empty signal.
DLY_enable_D[NUM_DATA_LANE-1:0]	Output	Enable dynamic delay for Rx data lane. (Not in use for current release.)
DLY_inc_D[NUM_DATA_LANE-1:0]	Output	Increment dynamic delay for Rx data lane. (Not in use for current release.)
u_dly_enable_D[NUM_DATA_LANE-1:0]	Input	Controls the RX data lane dynamic delay. Used together with u_dly_inc_D. Refer to u_dly_inc_D for usage examples. Available when ENABLE_USER_DESKEWCAL = 1. (Not in use for current release.)

Port	Direction	Description
u_dly_inc_D[NUM_DATA_LANE-1:0]	Input	Controls the RX data lane dynamic delay. Example: u_dly_inc_D = 1 and u_dly_enable_D = 1, the delay step increases every clock cycle. u_dly_inc_D = 0 and u_dly_enable_D = 1, the delay step decreases every clock cycle. u_dly_enable = 0, the delay value stays put. Available when ENABLE_USER_DESKEWCAL = 1. (Not in use for current release.)
Tx_LP_D_P	Output	LP mode TX data single-ended P signal.
Tx_LP_D_P_OE	Output	Output enable for LP mode TX data single-ended P signal.
Tx_LP_D_N	Output	LP mode TX data single-ended N signal.
Tx_LP_D_N_OE	Output	Output enable for LP mode TX data single-ended N signal.

Table 5: Video Interface

All signals are clocked by clk_pixel.

Port	Direction	Description
hsync	Output	Active-low horizontal sync signal.
vsync	Output	Active-low vertical sync signal.
pixel_data [63:0]	Output	Video Data. The actual data width of this port is dependent on pixel type. See Video Mode Pixel Encoding on page 14.
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: Side band Interface

All signals are clocked by clk_byte_HS except irq which is clocked by axi_clk.

Port	Direction	Description
video_format	Input	Set to related video format datatype 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. Tie to zeros if there is no concern for inaccuracy of pixel interface timing (especially on HSA, and HBP), then the pixel interface (hsync, vsync, pixel_data, pixel_data_valid) 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
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

All signals are clocked with axi_clk and axi_reset_n.

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 the 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 [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] = RxErrSotSyncHS_0 [18] = RxErrControl_0 [19] = RxErrEsc_0 [20] = RxStopState_0 [21] = RxSkewCalHS_0 [22] = RxUtpsActiveNot_0 [23] = RxUtpsEsc_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.</p> <p>Currently no function has been implemented. Tie all input bits to zero.</p> <ul style="list-style-type: none"> [31:0] = reserved

Register Definition

Table 9: Control Status Registers

Word Offset	Bits	Name	R/W	Width (bits)
0x00	16:0	Interrupt status register.	R/W1C	17
0x04	16:0	Interrupt enable register.	R/W	17
0x08	6:0	Rx DPHY Data lane0 status.	RO	7
0x0C	6:0	Rx DPHY Data lane1 status.	RO	7
0x10	6:0	Rx DPHY Data lane2 status.	RO	7
0x14	6:0	Rx DPHY Data lane3 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/long packet (hs_cmdfifo_data).	RO	32
0x30	7:0	Received low power write command/payload for short/long packet (lp_cmdfifo_data).	RO	8
0x34	31:0	LPDT read 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 HSA value into the HBP value (not in use as per current implementation).	R/W	16
0x4C	15:0	Horizontal front porch (HFP) in byte (not in use 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 use 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 use 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.
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 wordcount 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 received CRC error indicator.
12	hs_ecc_1bit_error	HS packet received ECC 1-bit error indicator.
13	hs_ecc_2bit_error	HS packet received ECC 2-bit error indicator.
14	lp_crc_error	LP packet received CRC error indicator.
15	lp_ecc_1bit_error	LP packet received ECC 1-bit error indicator.
16	lp_ecc_2bit_error	LP packet received 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 enable registers are set to 1'b0 (disabled).

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

Bit	Name	Description
14	lp_crc_error	Enable interrupt generation for CRC error during lp packet reception.
15	lp_ecc_1bit_error	Enable interrupt generation for ECC 1-bit error during lp packet reception.
16	lp_ecc_2bit_error	Enable interrupt generation for ECC 2-bit error during lp packet reception.

Table 12: D-PHY Status 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 RxWordClkHS 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 and ALP 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	EscapeULPS (Receive) mode. The core asserts this signal high to indicate that the lane module has entered the ULPS, due to 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-PHY Status 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	ReceiveULPS on Clock Lane. The core deasserts this signal low to indicate that the clock lane module has entered the ULPS due to 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 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 parameters	Short	LP/HS
0x13	Generic short write, 1 parameters	Short	LP/HS
0x23	Generic short write, 2 parameters	Short	LP/HS
0x4	Generic short read, no parameters	Short	LP/HS
0x14	Generic short read, 1 parameters	Short	LP/HS
0x24	Generic short read, 2 parameters	Short	LP/HS
0x5	DCS short write, no parameters	Short	LP/HS
0x15	DCS short write, 1 parameters	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	[47:44] 4'h0	[43:34] Y	[33:24] Cr	[23:20] 4'h0	[19:10] Y	[9:0] Cb	

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]	[41:36]	[35:30]	[29:24]	[23:18]	[17:12]	[11:6]	[5:0]							
			6'h0	Blue	Green	Red	6'h0	Blue	Green	Red							

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 14.

There is a FIFO for data transfer from the byte clock domain to the pixel clock domain. The pixel clock frequency is applied approximately equal to the formula given above (or, within $\pm 100\%$ to 150% of the calculated value). If the pixel clock frequency is too low, a FIFO overflow occurs (overflow happens when the read clock is slower than the write clock). If the pixel clock frequency is too high, a FIFO underflow occurs (which causes the `pixel_data_valid` to be in a toggling fashion within a one-pixel line).

Video Timing Parameters

The following waveforms show the video interface signals relationship.

Figure 2: Video Timing Waveform (Horizontal)

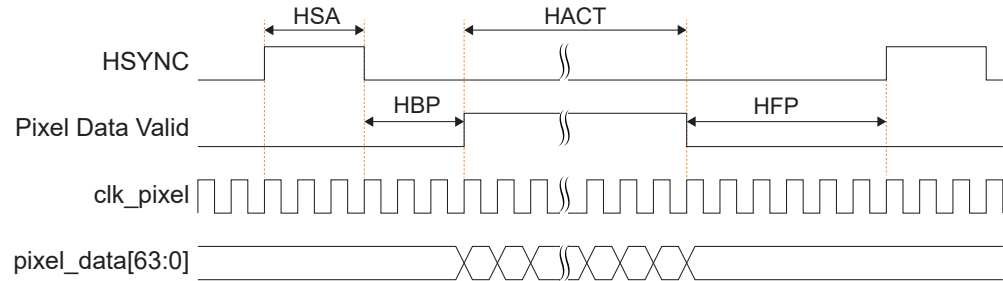


Figure 3: Video Timing Waveform (Vertical)

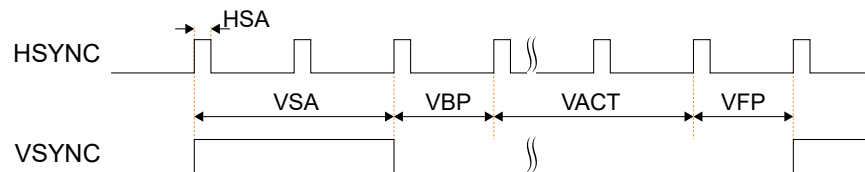


Table 25: Video Timing Parameter Definitions

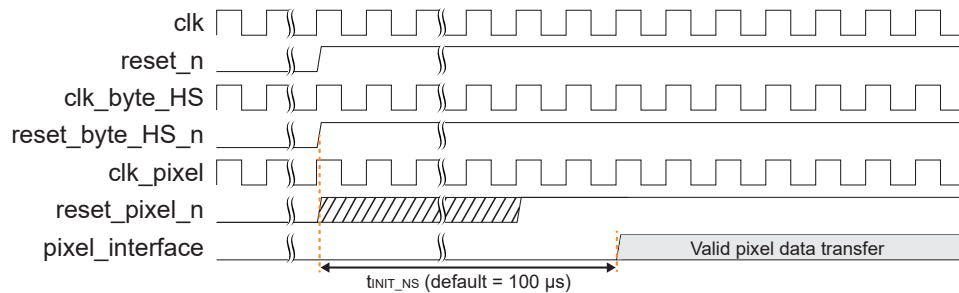
MIPI Video Timing Parameters	Definition
HACT	Total number of pixel per line
VACT	Total number of line per frame
HSA	HSYNC pulse width
HBP	Horizontal back porch
HFP	Horizontal front porch
VSA	VSYNC pulse width
VBP	Vertical back porch
VFP	Vertical front porch
Pixel Clock	Video stream pixel clock frequency 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 $100\ \mu s$ is the minimum requirement for the MIPI D-PHY transmitter to function properly before an LP/HS data transfer. You must ensure that no traffic is sent to the pixel interface before the expiration of t_{INIT_NS} . For a valid data transfer to take place, there should be at least 2-pixel clocks after the `reset_pixel_n` is out from reset. For any reset assertion event during user mode, you must follow the reinitialization sequence for reset deassertion procedure.

The following figure describes the reset sequencing and initialization requirements.

Figure 4: 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 in situations in which you use several IP cores, multiple instances of an IP core with different parameters, or the same IP core for different projects.



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

Generating the MIPI 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 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 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.vh**—Contains the customized parameters.
- **<module name>_tpl.v**—Verilog HDL instantiation template.
- **<module name>_tpl.vhd**—VHDL instantiation template.
- **<module name>.v**—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 and testbench files.

Customizing the MIPI DSI RX Controller

The core has parameters so you can customize its function. You set the parameters in the **General** tab of the core's IP Configuration window.

Table 26: MIPI DSI RX Controller Core Parameter

Name	Option	Description
t _{LPX_NS}	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. The t _{LPX_NS} ratio between host processor and peripheral must not exceed 3:2. The host processor is responsible for controlling its own clock frequency to match the peripheral. The host processor LP clock frequency must be in the range of 67% to 150% of peripheral LP clock frequency. Default: 50
t _{INIT_NS}	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. Default: 100000
t _{LP_EXIT_NS}	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. Default: 100
Bus turnaround timeout period (ns)	Values according to MIPI D-PHY specifications.	Bus turnaround timeout period in ns. Default: 100000
t _{D_TERM_EN_NS}	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. Default: 35
t _{CLK_TERM_EN_NS}	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. Default: 38
t _{HS_PREPARE_ZERO_NS}	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns (value before adding UI). This parameter describes the expected timing (combination of t _{HS_PREPARE} and t _{HS_ZERO}) driven by transmitter from TX. Default: 145
t _{HS_SETTLE_NS}	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns (value before adding UI). Default: 85
t _{HS_PREPARE_NS}	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns (value before adding UI). This parameter dedicates to the transmitter timing from RX to TX during the BTA function. Default: 40
t _{WAKEUP_NS}	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. This parameter dedicates to the transmitter timing from RX to TX during BTA function. Default: 1000

Name	Option	Description
t _{HS_EXIT_NS}	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. This parameter dedicates to the transmitter timing from RX to TX during BTA function. Default: 100
t _{HS_ZERO_NS}	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns (value before adding UI). This parameter dedicates to the transmitter timing from RX to TX during BTA function. Default: 105
t _{HS_TRAIL_NS}	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns (value before adding UI). This parameter dedicates to the transmitter timing from RX to TX during BTA function. Default: 60
MIPI Parallel Clock Frequency	10 - 187	MIPI parallel clock frequency in MHz. Round down the frequency value to an integer if there is any floating point from your calculation. Default: 125
IP Core Clock Frequency	40 - 100	IP core clock frequency in MHz. Default: 100
Data Lanes	1, 2, 4	Number of data lanes. Default: 4
Enable user setting on RX data I/O lane dynamic	0,1	Allows user to control the RX data I/O lane dynamic delay. Default: 0 (Not applicable anymore, always set it to 0.)
DPHY Clock Mode	Continuous, Discontinuous	DPHY clock mode. Default: Continuous
Enable bidirectional DPHY	Enable, Disable	To instantiate a unidirectional or bidirectional soft D-PHY. Default: Enable
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
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
Pixel Data FIFO Depth	256 - 8192	FIFO depth size to store the pixel packet data (set to 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 (set to power of 2 value). Default: 64
Low-power write data FIFO depth	8 - 2048	LP command write data FIFO depth (set to power of 2 value). Default: 64
Low-power read data FIFO depth	8 - 2048	Bus turnaround read data depth (set to power of 2 value). Default: 64
MIPI_DSI_RX_DEBUG	Enable, Disable	Enable debug ports for internal signal observation and monitoring. Default: Disable

MIPI 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.

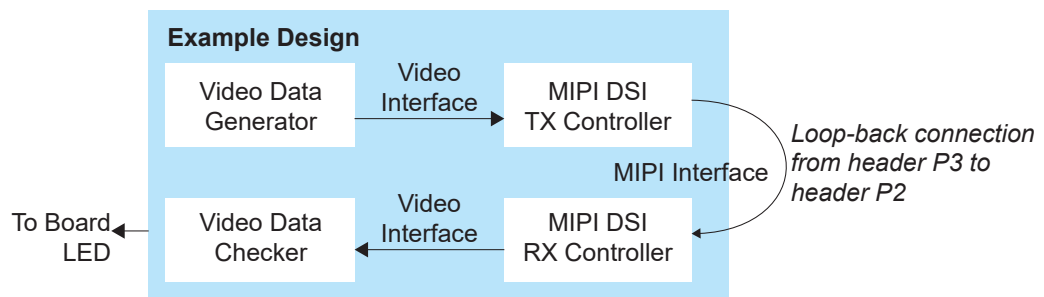


Important: Efinix tested the example design generated with the default parameter options only.

The example design targets the Titanium Ti60 F225 Development Board. The design instantiates both MIPI DSI TX and RX Controller cores. This design requires a QTE header-compatible cable.

The design generates video frame data and sends the video data to the MIPI DSI TX Controller. The data is then sent through a hardware loopback on the board using a 4-lane MIPI interface to the MIPI DSI RX Controller. There is video data checker compares the data received with the one created by the video data generator, and outputs the results using the board LEDs.

Figure 5: MIPI DSI RX Controller Core Example Design



After power-up and device programming is done, the following response can be observed:

- LED D17 B—Blinks continuously indicating there are traffic being sent over to MIPI IP.
- LED D16 R—Turns on to indicate that the hsync signal matches TX and RX.
- LED D16 G—Turns on to indicate that the vsync signal matches TX and RX.
- LED D16 B—Turns on to indicate that the pixel data signal matches TX and RX.
- LED D17 R—Turns on to indicate that the pixel data valid signal matches TX and RX.

The RX clock to RX data skew varies in different board, hence there is a possibility where the RX clock might not be able to capture the RX data correctly. In this case, both the LEDs do not turn on. You have to try increase the **Static Mode Delay Setting** of `mipi_dphy_rx_data` in the Interface Designer.



Note: You can use the **Titanium MIPI Utility-v<version>.xism** to check if your own 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.

Table 27: Titanium Resource Utilization and Performance

MIPI DSI RX Controller with 4 data lanes.

FPGA	Logic Elements (Logic, Adders, Flipflops,etc.)	Memory Blocks	DSP Blocks	f _{MAX} (MHz) ⁽⁵⁾				Efinity® Version ⁽⁶⁾
				clk	axi_clk	clk_byte_HS	clk_pixel	
Ti60 F225 C4	10,308 / 60800 (16.95%)	59/256 (23.05%)	0/160 (0%)	205	298	209	184	2024.2.294.4.12

⁽⁵⁾ Using default parameter settings.⁽⁶⁾ Using SystemVerilog.

MIPI DSI RX Controller Testbench

You can choose to generate the testbench when generating the core in the IP Manager Configuration window.



Note: You must include all `.v` files generated in the `/testbench` directory in your simulation.



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

Efinix provides a simulation script for you to run the testbench quickly using the Modelsim software. To run the Modelsim testbench script, run `vsim -do modelsim.do` in a terminal application. You must have Modelsim installed on your computer to use this script.

The IP Manager generates different encrypted source code for you to simulate with different simulators.

Table 28: Testbench Files

Directory/File	Note
../Testbench	Contains the example design and testbench files.
../Testbench/aldec	Contains the generated encrypted source code to simulate with the Aldec simulator.
../Testbench/modelsim	Contains the generated encrypted source code to simulate with the Modelsim simulator.
../Testbench/ncsim	Contains the generated encrypted source code to simulate with the NCSIM simulator.
../Testbench/synopsys	Contains the generated encrypted source code to simulate with the VCS simulator.

The simulation testbench simulates the example design. The design generates a video pattern to the MIPI DSI TX controller and loopback to MIPI DSI RX Controller. The output pixel interface from the DSI RX controller is checked by a checker in the testbench.

After running the simulation successfully, the test prints the following message:

```
# Mipi test: begin @ 0
# Mipi test: reset released @ 1000
# Mipi test; pll locked @ 1000
# Mipi test: simulation done @ 2001001000
# Mipi test: ~~~~~
# Mipi test: RESULT ..... TEST PASSED ^ ^
# Mipi test: ~~~~~
```

Revision History

Table 29: Revision History

Date	Document Version	IP Version	Description
June 2025	1.8	5.8	Add support for trigger escape command. (SIP-953) Added [27:24] = RxTriggerEsc in mipi_debug_out[31:0] of Debug Interface table.
May 2025	1.7	5.7	Added Reset Sequence and Initialization. (DOC-2485) Update default parameter value. (SIP-910) Example design I/O bank update (HVIO 3.3 V). (SIP-907) Updated example design. Updated Customizing the MIPI DSI RX Controller.
March 2025	1.6	5.6	RTL fix for HSIO RX HS ENABLE timing. (SIP-842) RTL fix for tHS_TRAIL improvement. (SIP-849)
January 2025	1.5	5.5	Example design update to align with MIPI Utility change (DOC-1783). (SIP-792)
December 2024	1.4	5.4	Updated mapping in Table 17: Loosely Packed Pixel Stream, 20-bit YCbCr (2-pixels per clock) on page 15 and Table 22: RGB666 (2-pixels per clock) on page 16. (DOC-2277)
December 2024	1.3	5.4	Added debug ports for internal signal observation and monitoring in Ports and Customizing the MIPI DSI RX Controller. (SIP-580) Removed Pack Type 36 parameter from MIPI DSI RX Controller Core Parameter table. (SIP-803) Added Testbench support.
November 2024	1.2	5.3	Added Topaz in Features and Device Support. (DOC-2102) Added IP Version in Revision History. (DOC-2185) Soft DPHY 1.5Gbps performance improvement. (SIP-614)
September 2024	1.1	-	Updated HSA value to 2. (SIP-526) Updated Table 26: MIPI DSI RX Controller Core Parameter on page 20, Table 3: Clock and Reset Ports on page 6, and Table 4: MIPI RX I/O interface on page 6. Updated Resource Utilization and Performance, IP Manager, and Example Design. Updated pixel_data[63:0] in Figure 2: Video Timing Waveform (Horizontal) on page 17.
February 2024	1.0	-	Initial release.