MIPI CSI-2 RX Controller Core User Guide

UG-CORE-MIPI-CSI2-RX-v3.2
July 2025
www.efinixinc.com

Contents

Introduction

The MIPI CSI-2 interface, which defines a simple, high-speed protocol, is the most widely used camera interface for mobile[1]. Adding a MIPI interface to an FPGA creates a powerful bridge to transmit or receive high-speed video data easily to/from an application processor. The MIPI CSI-2 RX Controller core allows you to perform complex video and image processing as a part of a complete system solution.

Use the IP Manager to select IP, customize it, and generate files. The MIPI CSI-2 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

- Configurable data lanes: 1, 2, 4, or 8
- High-speed (HS) mode and Low-power (LP) mode
- Arbitrary number of payload data bytes
- HS mode byte clock frequency from 10 MHz up to 187 MHz (from 80 Mbps up to 1,500 Mbps data rate)[2]
- Continuous HS mode byte clock and discontinuous HS mode byte clock
- 8-bit HS mode data width
- Pixel format:
  - RAW: RAW6, RAW7, RAW8, RAW10, RAW12, RAW14, RAW16, RAW20, RAW24, RAW28
  - RGB: RGB444, RGB555, RGB565, RGB888
  - YUV: YUV420 8-bit (legacy), YUV420 8-bit, YUV420 10-bit, YUV420 8-bit (CSPS), YUV420 10-bit (CSPS), YUV422 8-bit, YUV422 10-bit
- User defined 8-bit data types
- Generic 8-bit long packet
- Null, blank, and embedded 8-bit non-image data
- PPI interface
- Generic frame mode and accurate frame mode
- Supports end of transmission error, start of transmission sync error, control error & LP escape error
- Supports control status register (CSR) for status and error assertion accessed through AXI4-Lite interface

(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.

Device Support

Table 1: MIPI CSI-2 RX Controller Core Device Support

FPGA Family	Supported Device
Trion	-
Titanium and Topaz	All

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

FPGA	Logic and Adders	Flip-flops	Memory Blocks	DSP48 Blocks	f_{MAX} (MHz)[3]				Efinity® Version[4]
					clk	axi_clk	clk_byte_HS	clk_pixel	
Ti60 F225 C4	3,678	1,503	11	0	415	453	359	377	2021.2

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.

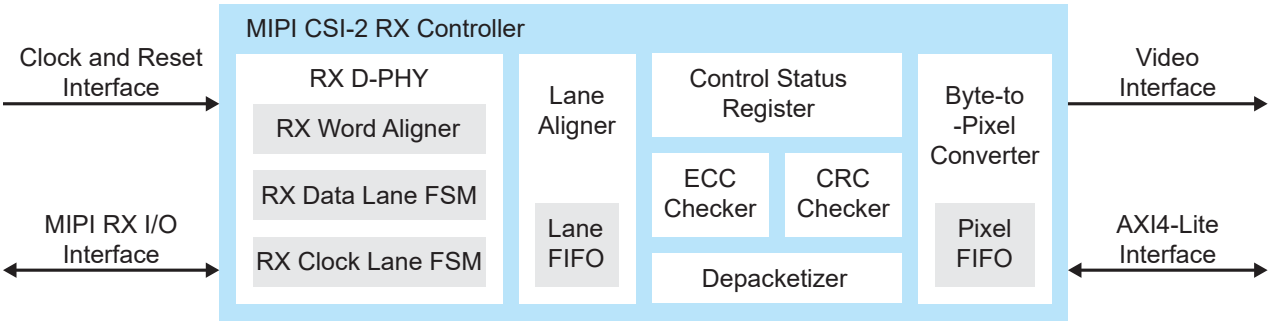
Note: You must be logged in to the Support Center to view the IP Core Release Notes.

[3] Using default parameter settings.
[4] Using Verilog HDL.

Functional Description

The MIPI CSI-2 RX Controller consists of a RX D-PHY block, lane aligner, control status registers, ECC and CRC checkers, depacketizer, and byte-to pixel converter. The core has a camera, AXI4-lite, MIPI RX I/O, and clock and reset interfaces.

Figure 1: MIPI CSI-2 RX Controller System Block Diagram



Ports

Table 3: Clock and Reset Ports

Port	Direction	Description
clk	Input	IP core clock consumed by controller logics. 100 MHz.
reset_n	Input	IP core reset signal.
clk_byte_HS	Input	MIPI RX parallel clock. This is a HS transmission 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.
axi_reset_n	Input	AXI4-Lite interface active low reset.

Note: Refer to the Interfaces User Guide in the Support Center for serial or parallel clock requirements.

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_*n* [7:0]	Input	HS mode differential lane data bus. *n* = lane 0 to 7
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	Reserved port.
DLY_inc_D [NUM_DATA_LANE-1:0]	Output	Reserved port.
u_dly_enable_D [NUM_DATA_LANE-1:0]	Input	Reserved port, Tie the signal to zero.
u_dly_inc_D [NUM_DATA_LANE-1:0]	Input	Reserved port. Tie the signal to zero.

Table 5: AXI4-Lite Interface

Interface to access **Table 10: Control Status Registers** on page 10.

All signals are clocked with axi_clk and axi_reset_n.

Port	Direction	Description
axi_awaddr [15:0]	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 [15:0]	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 6: Video Interface

All signals are clocked with clk_pixel and reset_pixel_n. The hsync_vc and vsync_vc are level signals and not pulse signals. See **Video Timing Parameters**.

Port	Direction	Description
hsync_vc*x*	Output	Active-high horizontal sync for virtual channel. *x* = virtual lane 0 to 15
vsync_vc*x*	Output	Active-high vertical sync for virtual channel. *x* = virtual lane 0 to 15
pixel_data [63:0]	Output	Video Data. The actual data width of this port is dependent on pixel type. Refer to the pixel encoding table.
pixel_data_valid	Output	Active-high pixel data enable.
pixel_per_clk [15:0]	Output	Number of pixel per pixel clock. This signal only valid when pixel_data_valid flag is high.

Table 7: Sideband Interface

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

Port	Direction	Description
vc [1:0]	Output	2-bit virtual channel signal decoded from the packet header [7:6].
vcx [1:0]	Output	2-bit virtual channel signal decoded from the packet header [25:24].
word_count [15:0]	Output	Byte count of the long packet received by CSI2 RX controller.
datatype [5:0]	Output	Decoded data type of incoming packet received by CSI2 RX controller.
shortpkt_data_field [15:0]	Output	16-bit short packet data field for short packet.
irq	Output	Interrupt signal for Interrupt Status Register.

Table 8: Debug Interface

All signals are clocked with axi_clk and axi_reset_n.

Port	Direction	Description
mipi_debug_out[31:0]	Output	Debug port. Present if the parameter **MIPI_CSI2_RX_DEBUG** is **Enabled**. The following is the list of internal signals that can be monitored. [0] = pixel_fifo_full [1] = pixel_fifo_empty [2] = crc_error [3] = ecc_1bit_error [4] = ecc_2bit_error [5] = undersize_pkt_error [6] = line_vc0_error [7] = line_vc1_error [8] = line_vc2_error [9] = line_vc3_error [10] = frame_vc0_error [11] = frame_vc1_error [12] = frame_vc2_error [13] = frame_vc3_error [14] = receive_error [15] = init_done [16] = RxErrSotSyncHS_0 [17] = RxErrControl_0 [18] = RxErrEsc_0 [19] = RxStopState_0 [20] = RxSkewCalHS_0 [21] = RxUlpsActiveNot_0 [22] = RxUlpsEsc_0 [31:23] = reserved
mipi_debug_in[31:0]	Input	Debug ports. Present if the parameter **MIPI_CSI2_RX_DEBUG** is **Enabled**. Currently no function has been implemented. Tie all input bits to zero. [31:0] = reserved

Table 9: Pixel Sideband Interface

All signals are clocked by clk_pixel and reset_pixel_n. These output ports are present when `MIPI_CSI2_RX_PIXEL_SIDEBAND` switch is set to `ENABLE`.

Port	Direction	Description
pixel_line_num[15:0]	Output	Line number decoded from the incoming short packet. This signal is valid or changes only when the hsync signal asserts from low to high.
pixel_frame_num[15:0]	Output	Frame number decoded from the incoming short packet. This signal is valid or changes when the vsync signal asserts from low to high.
pixel_datatype[5:0]	Output	Decoded data type of incoming packet received by the RX controller. This signal is valid or changes when the pixel_data_valid signal asserts from low to high.
pixel_wordcount[15:0]	Output	Decoded wordcount of incoming long packet received by the RX controller. This signal is valid or changes when the pixel_data_valid signal asserts from low to high.
pixel_vc[1:0]	Output	2-bit virtual channel signal decoded from the packet header [7:6]. This signal is valid or changes when the vsync/hsync/pixel_data_valid signal asserts from low to high.
pixel_vcx[1:0]	Output	2-bit virtual channel signal decoded from the packet header [25:24]. This signal is valid or changes when the vsync/hsync/pixel_data_valid signal asserts from low to high.

Pixel Clock Calculation

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

PIX_CLK_MHZ >= (DATARATE_MBPS * NUM_DATA_LANE) / 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 Pixel Encoding on page 13

There is a FIFO for data transfer from the byte clock domain to the pixel clock domain. You are recommended to apply the pixel clock frequency approximately equal to the formula given. 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 manner within a horizontal line). The maximum supported frequency for the pixel clock is 300 MHz.

Control Status Registers

Table 10: Control Status Registers

Word Address Offset	Name	R/W	Width (bits)
0x00	Interrupt Status Register	Bit[1:0] - R Bit[14:2] - R/W1C[5]	15
0x04	Interrupt Enable Register	R/W	15
0x08	D-PHY status for lane 0	R	8
0x0C	D-PHY status for lane 1	R	8
0x10	D-PHY status for lane 2	R	8
0x14	D-PHY status for lane 3	R	8
0x18	D-PHY status for lane 4	R	8
0x1C	D-PHY status for lane 5	R	8
0x20	D-PHY status for lane 6	R	8
0x24	D-PHY status for lane 7	R	8
0x28	D-PHY status for clock lane	R	2

[5] Read register. Write 1 to clear the register.

Table 11: Interrupt Status Register Definition (0x00)

Bit	Name	Description
0	Pixel FIFO full	Pixel FIFO in the byte-to-pixel converter module is full.
1	Pixel FIFO empty	Pixel FIFO in the byte-to-pixel converter module is empty.
2	CRC error	CRC error indicator.
3	Ecc 1 bit error	ECC with 1 bit error indicator.
4	Ecc 2 bit error	ECC with 2 bit error indicator.
5	Undersize packet error	The incoming MIPI HS data byte is lesser than the wordcount value.
6	VC0 Line number synchronization error	Line number synchronization error for virtual channel 0.
7	VC1 Line number synchronization error	Line number synchronization error for virtual channel 1.
8	VC2 Line number synchronization error	Line number synchronization error for virtual channel 2.
9	VC3 Line number synchronization error	Line number synchronization error for virtual channel 3.
10	VC0 Frame number synchronization error	Frame number synchronization error for virtual channel 0.
11	VC1 Frame number synchronization error	Frame number synchronization error for virtual channel 1.
12	VC2 Frame number synchronization error	Frame number synchronization error for virtual channel 2.
13	VC3 Frame number synchronization error	Frame number synchronization error for virtual channel 3.
14	Initialization error	MIPI HS data is received before tInit is completed.

Table 12: Interrupt Enable Register Definition (0x04)

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

Bit	Name	Description
0	Pixel FIFO full full interrupt enable	Enable interrupt generation for Pixel FIFO full status bit.
1	Pixel FIFO empty full interrupt enable	Enable interrupt generation for Pixel FIFO empty status bit.
2	CRC error full interrupt enable	Enable interrupt generation for CRC error status bit.
3	ECC 1 bit error full interrupt enable	Enable interrupt generation for ECC 1 bit error status bit.
4	ECC 2 bit error full interrupt enable	Enable interrupt generation for ECC 2 bit error status bit.
5	Undersize packet error full interrupt enable	Enable interrupt generation for Undersize packet error status bit.
6	VC0 Line number synchronization error full interrupt enable	Enable interrupt generation for VC0 Line number synchronization error status bit.
7	VC1 Line number synchronization error full interrupt enable	Enable interrupt generation for VC1 Line number synchronization error status bit.
8	VC2 Line number synchronization error full interrupt enable	Enable interrupt generation for VC2 Line number synchronization error status bit.
9	VC3 Line number synchronization error full interrupt enable	Enable interrupt generation for VC3 Line number synchronization error status bit.
10	VC0 Frame number synchronization error full interrupt enable	Enable interrupt generation for VC0 Frame number synchronization error status bit.
11	VC1 Frame number synchronization error full interrupt enable	Enable interrupt generation for VC1 Frame number synchronization error status bit.

Bit	Name	Description
12	VC2 Frame number synchronization error full interrupt enable	Enable interrupt generation for VC2 Frame number synchronization error status bit.
13	VC3 Frame number synchronization error full interrupt enable	Enable interrupt generation for VC3 Frame number synchronization error status bit.
14	Initialization error full interrupt enable	Enable interrupt generation for Initialization error status bit.

Table 13: 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	Escape ULPS (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 lane interconnect.
7	Reserved	Reserved

Table 14: 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	Receive ULPS 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 lane interconnect.

Pixel Encoding

Table 15: Pixel Encoding

TYPE[5:0]	Data Type	Pixels per Clock	Bits per Pixel	Pixel Data Bits per Pixel Clock
0x20	RGB444	4	12	48
0x21	RGB555	4	15	60
0x22	RGB565	4	16	64
0x24	RGB888	2	24	48
0x28	RAW6	8	6	48
0x29	RAW7	8	7	56
0x2A	RAW8	8	8	64
0x2B	RAW10	4	10	40
0x2C	RAW12	4	12	48
0x2D	RAW14	4	14	56
0x2E	RAW16	4	16	64
0x2F	RAW20	2	20	40
0x27	RAW24	2	24	48
0x26	RAW28	2	28	56
0x18	YUV420 8 bit	Odd line: 8 Even line: 4	Odd line: 8 Even line: 8, 24	Odd line: 64 Even line: 64
0x19	YUV420 10 bit	Odd line: 4 Even line: 2	Odd line: 10 Even line: 10, 30	Odd line: 40 Even line: 40
0x1A	Legacy YUV420 8 bit	4	8, 16	48
0x1C	YUV420 8 bit (CSPS)	Odd line: 8 Even line: 4	Odd line: 8 Even line: 8, 24	Odd line: 64 Even line: 64
0x1D	YUV420 10 bit (CSPS)	Odd line: 4 Even line: 2	Odd line: 10 Even line: 10, 30	Odd line: 40 Even line: 40
0x1E	YUV422 8 bit	4	8, 24	64
0x1F	YUV422 10 bit	2	10, 30	40
0x30 - 37	User defined 8 bit	8	8	64
0x13 – 0x16	Generic 8-bit long packet	8	8	64
0x12	Embedded 8-bit non image data	8	8	64

MIPI RX Video Data DATA[63:0] Formats

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

Table 16: RAW6 (8 Pixels per Clock)

63 48	47 42	41 36	35 30	29 24	23 18	17 12	11 6	5 0
0	Pixel 8	Pixel 7	Pixel 6	Pixel 5	Pixel 4	Pixel 3	Pixel 2	Pixel 1

Table 17: RAW7 (8 Pixels per Clock)

63 56	55 49	48 42	41 35	34 28	27 21	20 14	13 7	6 0
0	Pixel 8	Pixel 7	Pixel 6	Pixel 5	Pixel 4	Pixel 3	Pixel 2	Pixel 1

Table 18: RAW8 (8 Pixels per Clock)

63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0
Pixel 8	Pixel 7	Pixel 6	Pixel 5	Pixel 4	Pixel 3	Pixel 2	Pixel 1

Table 19: RAW10 (4 Pixels per Clock)

63 40	39 30	29 20	19 10	9 0
0	Pixel 4	Pixel 3	Pixel 2	Pixel 1

Table 20: RAW12 (4 Pixels per Clock)

63 48	47 36	35 24	23 12	11 0
0	Pixel 4	Pixel 3	Pixel 2	Pixel 1

Table 21: RAW14 (4 Pixels per Clock)

63 56	55 42	41 28	27 14	13 0
0	Pixel 4	Pixel 3	Pixel 2	Pixel 1

Table 22: RAW16 (4 Pixels per Clock)

63 48	47 32	31 16	15 0
Pixel 4	Pixel 3	Pixel 2	Pixel 1

Table 23: RAW20 (2 Pixels per Clock)

63 40	39 20	19 0
0	Pixel 2	Pixel 1

Table 24: RAW24 (2 Pixels per Clock)

63 48	47 24	23 0
0	Pixel 2	Pixel 1

Table 25: RAW28 (2 Pixels per Clock)

63 56	55 28	27 0
0	Pixel 2	Pixel 1

Table 26: RGB444 (4 Pixels per Clock)

63 48	47 36			35 24			23 12			11 0		
0	Pixel 4			Pixel 3			Pixel 2			Pixel 1		
–	[47:44] Red	[43:40] Green	[39:36] Blue	[35:32] Red	[31:28] Green	[27:24] Blue	[23:20] Red	[19:16] Green	[15:12] Blue	[11:8] Red	[7:4] Green	[3:0] Blue

Table 27: RGB555 (4 Pixels per Clock)

63 60	59 45			44 30			29 15			14 0		
0	Pixel 4			Pixel 3			Pixel 2			Pixel 1		
–	[59:55] Red	[54:50] Green	[49:45] Blue	[44:40] Red	[39:35] Green	[34:30] Blue	[29:25] Red	[24:20] Green	[19:15] Blue	[14:10] Red	[9:5] Green	[4:0] Blue

Table 28: RGB565 (4 Pixels per Clock)

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

Table 29: RGB888 (2 Pixels per Clock)

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

Table 30: YUV420 8 bit Odd Line (8 Pixels per Clock), Even Line (4 Pixels per Clock)

63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0
Odd Lines							
Pixel 8 Y8	Pixel 7 Y7	Pixel 6 Y6	Pixel 5 Y5	Pixel 4 Y4	Pixel 3 Y3	Pixel 2 Y2	Pixel 1 Y1
Even Lines							
Pixel 4	Pixel 3			Pixel 2	Pixel 1		
Y4	V3	Y3	U3	Y2	V1	Y1	U1

Table 31: Legacy YUV420 8 bit (4 Pixels per Clock)

63 48	47 40	39 32	31 24	23 16	15 8	7 0
0	Pixel 4	Pixel 3		Pixel 2	Pixel 1	
Odd Lines	Y4	Y3	U3	Y2	Y1	U1
Even Lines	Y4	Y3	V3	Y2	Y1	V1

Table 32: YUV420 10 bit Odd Line (4 Pixels per Clock), Even Line (2 Pixels per Clock)

63 40	39 30	29 20	19 10	9 0
Odd Lines				
0	Pixel 4 Y4	Pixel 3 Y3	Pixel 2 Y2	Pixel 1 Y1
Even Lines				
0	Pixel 1	Pixel 2	Pixel 1	
	Y2	V1	Y1	U1

Table 33: YUV422 8 bit (4 Pixels per Clock)

63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0
Pixel 4	Pixel 3			Pixel 2	Pixel 1		
Y4	V3	Y3	U3	Y2	V1	Y1	U1

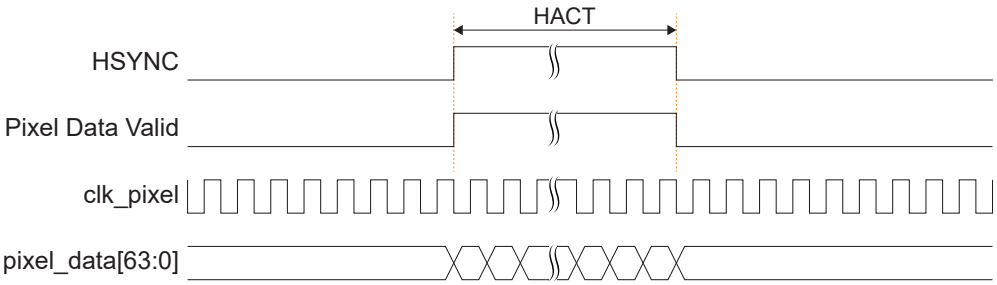
Table 34: YUV422 10 bit (2 Pixels per Clock)

63 40	39 30	29 20	19 10	9 0
0	Pixel 1	Pixel 2	Pixel 1	
	Y2	V1	Y1	U1

Video Timing Parameters

The following waveforms show the video interface signals relationship.

Figure 2: Video Timing Waveform (Horizontal) - Example of Generic Frame Mode



Note: In **Generic** mode, there are no incoming byte packets for Line Start and Line End. Thus, a decoded hsync output is unable to carry any information about the origin HFP and HBP of a horizontal line. Instead, the hsync output follows the valid signal assertion and deassertion of a valid pixel data type (except for embedded 8-bit data type, which is non-image data; therefore, the hsync is not asserted according to the pixel data valid).

Figure 3: Video Timing Waveform (Vertical) - Example of Generic Frame Mode

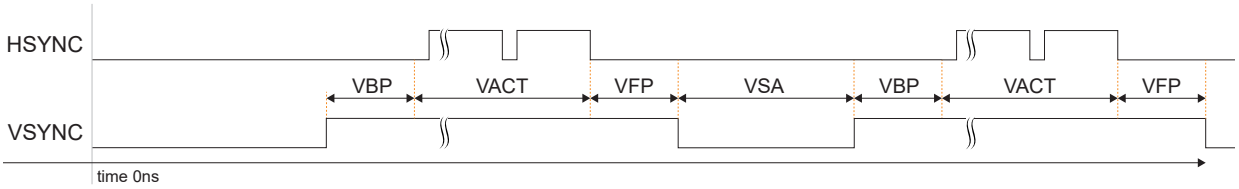
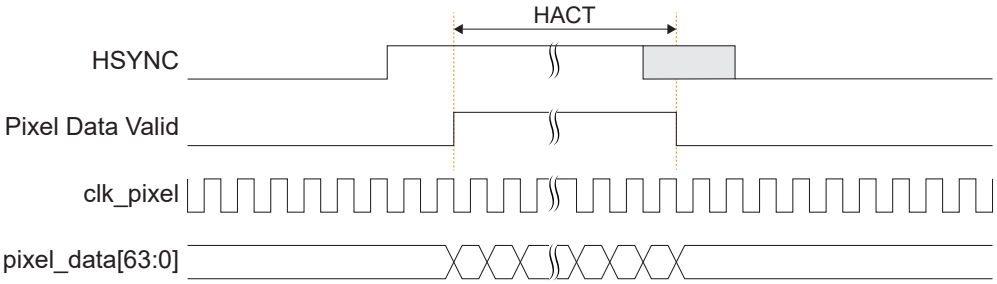


Figure 4: Video Timing Waveform (Horizontal) – Example of Accurate Frame Mode



Note: Since there is no precise pixel interface timing information from the received packets (byte clk domain), the pixel interface signal may not be accurately represented by the waveform, e.g., the falling edge of hsync may be coming earlier or later than pixel data valid (shaded area), depending on the byte clk, pixel clk, and data type for byte2pixel conversion. If the falling edge of hsync is critical, you need to extend the original HFP period in TX.

Figure 5: Video Timing Waveform (Vertical) - Example of Accurate Frame Mode

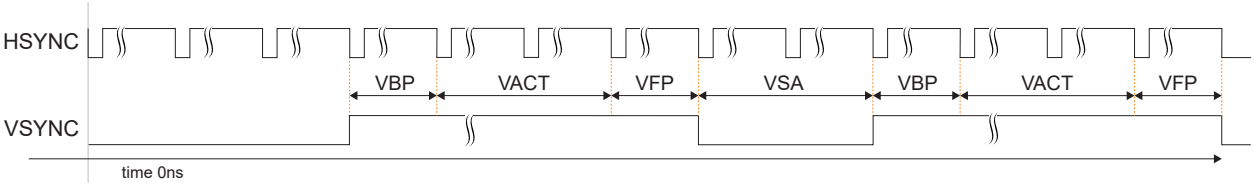


Table 35: Video Timing Parameter Definitions

MIPI Video Timing Parameters	Definition	Min	Max	Unit
HACT	Total number of pixel per line	16	8,192	Pixel
VACT	Total number of line per frame	1	8,192	Line
HSA	HSYNC pulse width	1	4,096	Pixel
HBP	Horizontal back porch	1	4,096	Pixel
HFP	Horizontal front porch	1	4,096	Pixel
VSA	VSYNC pulse width	1	8,192	Line
VBP	Vertical back porch	1	8,192	Line
VFP	Vertical front porch	1	8,192	Line
Pixel Clock	Video stream pixel clock frequency in MHz	(6)	(6)	MHz
MIPI Speed	CSI-2 RX MIPI speed in Mbps	80	1,500	Mbps
No. data lane	Number of MIPI data lane	1	8	Lane

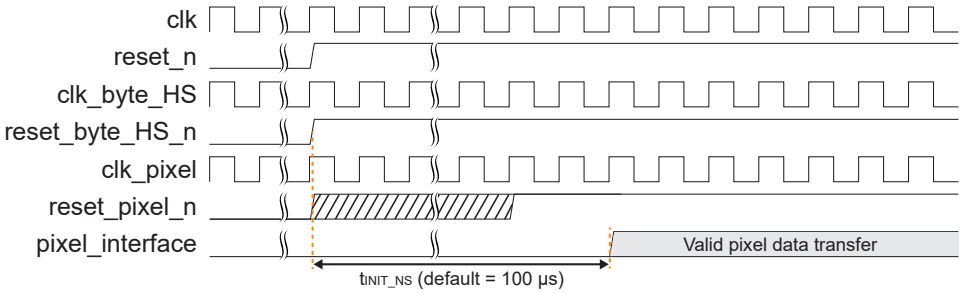
Table 35: Video Timing Parameter Definitions on page 18 describes the support range for video timing parameters. For a more precise check on timing compliance, you can refer to the **Titanium MIPI Utility**.

Reset Sequence and Initialization

During the initial power-up state, an initialization time, t_{INIT_NS} of 100 μs is the minimum requirement for the MIPI D-PHY receiver to function properly before an LP/HS data transfer (100 μs of initialization time is required when Efinix MIPI soft DPHY transmitter is used, you can adjust this timing depending on other transmitter's specifications). Typically, `reset_pixel_n` can be deasserted at any time after deassertion of other reset domains, but before the end of initialization process. The output of 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 reset deassertion procedure.

The following figure describes the reset sequencing and initialization requirements.

Figure 6: Reset Sequence and Initialization



(6) Refer to **Pixel Clock Calculation** on page 10.

IP Manager

The Efinity® IP Manager is an interactive wizard that helps you customize and generate Efinix® 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 Efinix 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 Efinix IP cores include an example design or a testbench.

Generating the MIPI CSI-2 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 CSI-2 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 CSI-2 RX Controller* section.
5. (Optional) In the **Deliverables** tab, specify whether to generate an IP core example design targeting an Efinix® 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>_tmpl.v**—Verilog HDL instantiation template.
- **<module name>_tmpl.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 CSI-2 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 36: MIPI CSI-2 RX Controller Core Parameter

Name	Option	Description
Data Lanes	1, 2, 4, 8	Number of data lanes. Default: 4
MIPI Parallel Clock Frequency	10 – 187	MIPI parallel clock (clk_byte_HS) frequency in MHz to support data rate of 80 Mbps to 1500 Mbps. Default: 187
IP Core Clock Frequency	40 - 100	IP core clock frequency in MHz Default: 100
D-PHY Clock Mode	Continuous, Discontinuous	To enable discontinuous or continuous HS mode clock. Default: Continuous
Pixel Data FIFO Depth	256 - 8192	FIFO depth size that stores the pixel packet data (set to power of 2 value). Minimum FIFO depth required > horizontal_pixel (HACT) x bits_per_pixel / 64 Default: 1024
Image Frame Mode	GENERIC, ACCURATE	Select the image frame mode: Generic mode: Frame format without accurate synchronization timing via Line Start and Line End. Accurate mode: Frame format with accurate synchronization timing via Line Start and Line End. Default: Generic
Enable Pipeline State for RXStopState Signal	8 – 15	To enable pipeline stage for RXStopState signal. The pipeline registers are clocked with HS mode byte clock. Compensates the MIPI HSIO deserializer, read FIFO and data synchronizer latency in designs with low MIPI data rate. Default: 8
t_{LPX} (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. Default: 50
t_{INIT} (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. Default: 100000
$t_{CLK\_TERM\_EN}$ (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns. Default: 38
$t_{D\_TERM\_EN}$ (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns (value before adding UI). Default: 35
t_{HS_SETTLE} (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns (value before adding UI). Default: 85

Name	Option	Description
$t_{HS\_PREPARE\_ZERO}$ (ns)	Values according to MIPI D-PHY specifications.	Soft D-PHY timing parameter in ns (value before adding UI). This parameter includes the tHS_ZERO parameter. Default: 145
Pack Type 40	Enable, Disable	Enables the controller to pack RAW10, RAW20, YUV_420_10, and YUV_422_10 data type.(7) Default: Enable
Pack Type 48	Enable, Disable	Enables the controller to pack RAW6, RAW12, RAW24, RGB888, and YUV_420_8_legacy data type.(7) Default: Enable
Pack Type 56	Enable, Disable	Enables the controller to pack RAW7, RAW14, and RAW28.(7) Default: Enable
Pack Type 64	Enable, Disable	Enables the controller to pack RAW8, RAW16, RGB444, RGB565, RGB555, YUV_422_8, YUV_420_8, generic long packet, user define 8-bit, and embedded 8-bit non image packet.(7) Default: Enable
Enable Extra Bits on Virtual Channel	Enable, Disable	Enables 16 virtual channel support. Default: Disable
MIPI_CSI2_RX_DEBUG	Enable, Disable	Enables debug ports for internal signal observation and monitoring. Default: Disable
MIPI_CSI2_RX_PIXEL_SIDEBAND	Enable, Disable	Enabels more sideband ports under pixel interface. Refer to portlist for more details. Default: Disable

(7) Only enable the pack type that you are using to save logic resources.

MIPI CSI-2 RX Controller Example Design

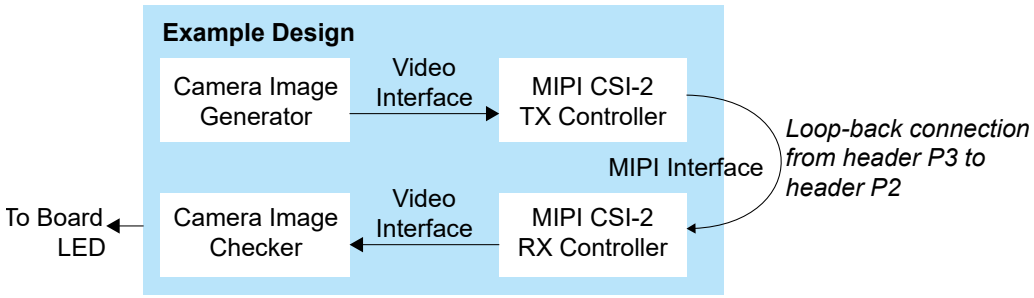
You can choose to generate the example design when generating the core in the IP Manager Configuration window. Compile the example design project and download the **.hex** or **.bit** file to your board.

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 CSI-2 TX and RX Controller cores. This design requires a QTE header-compatible cable.

The design generates an image and sends the image data to the camera image checker through the MIPI CSI-2 TX Controller. The data is then sent through a hardware loopback on the board using a 4-lane MIPI interface to the MIPI CSI-2 RX Controller. The camera image checker compares the data received with the one created by the image generator, and outputs the results using the board LEDs.

Figure 7: MIPI CSI-2 RX Controller Core Example Design



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

- LED0_B—Blinks continuously indicating there are traffic being sent over to MIPI IP.
- LED0_G—Turns on to indicate that the hsync signal matches TX and RX.
- LED1_B—Turns on to indicate that the vsync signal matches TX and RX.
- LED1_G—Turns on to indicate that the pixel data signal matches TX and RX.
- LED1_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>.xlsm** 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 37: Example Design Implementation

FPGA	Logic Elements (Logic, Adders, Flipflops, etc.)	Memory Blocks	DSP Blocks	f_{MAX} (MHz)[8]				Efinity® Version[9]
				clk1	clk2	clk3	clk4	
Ti60 F225 C4	6,329/60,800 (10.4%)	44/256 (17.2%)	0/160 (0%)	231	298	233	265	2024.2.294.4.12

- clk1—mipi_clk
- clk2—mipi_dphy_rx_clk_CLKOUT
- clk3—clk_pixel
- clk4—mipi_dphy_tx_SLOWCLK

[8] Using default parameter settings.
[9] Using Verilog HDL.

MIPI CSI-2 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 38: Testbench Files

Directory/File	Note
../Testbench/ modelsim.do	Modelsim testbench script.
../Testbench/ efx_<ip>_modelsim.sv	Encrypted source code for simulating loop-back example design in Modelsim testbench.
../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.

Tip: The default testbench is Modelsim which complies with CSI-2 TX IP loopback to CSI-2 RX IP. To simulate with other simulators, you can get the encrypted source code from **../Testbench/<simulation_tool>/** folder by generating the targeted CSI-2 IP and replacing it with **efx_csi2_tx_modelsim.sv** or **efx_csi2_rx_modelsim.sv**.

The simulation testbench simulates the example design. The design instantiates both MIPI CSI-2 TX and RX Controller cores. The loopback connection on the MIPI interface is done in the testbench file. This design generates an image and sends the image data to the camera image checker through the MIPI CSI-2 TX Controller and MIPI CSI-2 RX Controller. The camera image checker compares the data received with the one created by the image generator. After running the simulation successfully, the test prints the following message:

```
Correct RX data AA, received
Correct RX data AA, received
```

Revision History

Table 39: Revision History

Date	Document Version	IP Version	Description
July 2025	3.2	5.13	Corrected on Interrupt Status Register in Control Status Registers R/W access attribute table. (DOC-2609) Added more details on the Video Timing Parameter Definition table in the Video Timing Parameters topic.
June 2025	3.1	5.12	Added upper bound for pixel clk frequency. (SIP-952) Add ports to get line and frame number. (SIP-943) Update reset sequence descriptions. (DOC-2561) Added table Pixel Sideband Interface in Ports. Added Note, Video Timing Waveform (Horizontal) - Example for Accurate Frame Mode,and Video Timing Waveform (Vertical) - Example for Accurate Frame Mode in Video Timing Parameters. Updated Customizing the MIPI CSI-2 RX Controller, Pixel Clock Calculation, and Reset Sequence and Initialization. Corrected description of static delay adjustment in example design.
May 2025	3.0	5.11	Updated example design. (SIP-891) Example Design IO bank update (HVIO 3.3V). (SIP-907) Update default parameter value. (SIP-910) Updated Customizing the MIPI CSI-2 RX Controller. Updated observation after downloading bitstream and Example Design Implementation table in Example Design. Updated Video Interface table. Added Sideband Interface Ports table. (DOC-2382) Updated Reset Sequence and Initialization topic. (DOC-2485)
March 2025	2.9	5.10	Removed unsupported dynamic delay control feature. (SIP-841) RTL fix for HSIO RX HS ENABLE timing. (SIP-842).

Date	Document Version	IP Version	Description
January 2025	2.8	5.9	Updated Figure Video Timing Waveform (Horizontal), Video Timing Waveform (Vertical). (SIP-823) Example design update to align with MIPI Utility change (DOC-1783). (SIP-792)
December 2024	2.7	5.8	Added debug ports for internal signal observation and monitoring in Ports and Customizing the MIPI CSI-2 RX Controller. (SIP-580)
November 2024	2.6	5.7	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) Fix byte2pixel conversion issue when ENABLE_VCX = 1. (SIP-759)
September 2024	2.5	–	Added 8 lanes support in Features and **Table 36: MIPI CSI-2 RX Controller Core Parameter** on page 20. (SIP-677) Updated pixel data[63:0] in **Figure 2: Video Timing Waveform (Horizontal) - Example of Generic Frame Mode** on page 17. Removed data type RGB666 from **Table 15: Pixel Encoding** on page 13. (DOC-2068)
July 2024	2.4	–	Fixed error in **Table 5: AXI4-Lite Interface** on page 7. (DOC-2004)
June 2024	2.3	–	Update Pixel FIFO depth requirement in table MIPI CSI-2 RX Controller Core Parameter. (SIP-570) Revised supported lane number. Removed 8 from Features and MIPI CSI-2 RX Controller Core Parameter. (SIP-578) Added Reset Sequence and Initialization sub-section.
March 2024	2.2	–	Added important note in Testbench regarding using default parameters options only. (DOC-1781) Added testbench file for Modelsim and Aldec simulation model support. (DOC-1782)
October 2023	2.1	–	Updated MIPI video data format tables to include RGB information. (DOC-1474)
July 2023	2.0	–	Added more description for Accurate and Generic image frame modes. (DOC-1343)

Date	Document Version	IP Version	Description
June 2023	1.9	–	Added Device Support and release notes sections. (DOC-1234) Updated supported data rate. (DOC-1217) Updated port descriptions. Added RAW16, RAW20, RAW24, and RAW28 format support. Updated MIPI Parallel Clock Frequency, IP Core Clock Frequency, Pixel Data FIFO Depth Size, Pack Type40, Pack Type48, Pack Type56, Pack Type64 parameters. Improved Interrupt Enable Register Definition descriptions. Editorial changes.
April 2023	1.8	–	Updated tHS_PREPARE_ZERO (ns) description to indicate that includes tHS_ZERO. (DOC-1186)
February 2023	1.7	–	Added note about the resource and performance values in the resource and utilization table are for guidance only.
August 2022	1.6	–	Updated Control Status Register note. (DOC-898)
August 2022	1.5	–	Added video parameters waveform, and port clock domains. (DOC-819)
January 2022	1.4	–	Improved description about CSR is accessed through AXI4-Lite interface. (DOC-690) Corrected interrupt status register width and improved D-PHY stop state status description. (DOC-697) Updated resource utilization table. (DOC-700)
December 2021	1.3	–	Added simulation testbench. Added new IP manager parameters. Added new ports.
November 2021	1.2	–	Added support for 8 data lanes. (DOC-604)
October 2021	1.1	–	Added note to state that the f_{MAX} in Resource Utilization and Performance, and Example Design Implementation tables were based on default parameter settings. Updated design example target board to production Titanium Ti60 F225 Development Board and updated Resource Utilization and Performance, and Example Design Implementation tables. (DOC-553)
June 2021	1.0	–	Initial release.