Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.



EZ-USB® CX3 Technical Reference Manual

(Supplement to the EZ-USB FX3 Technical Reference Manual)

Doc. No. 001-91492 Rev. *B

1.1 Introduction

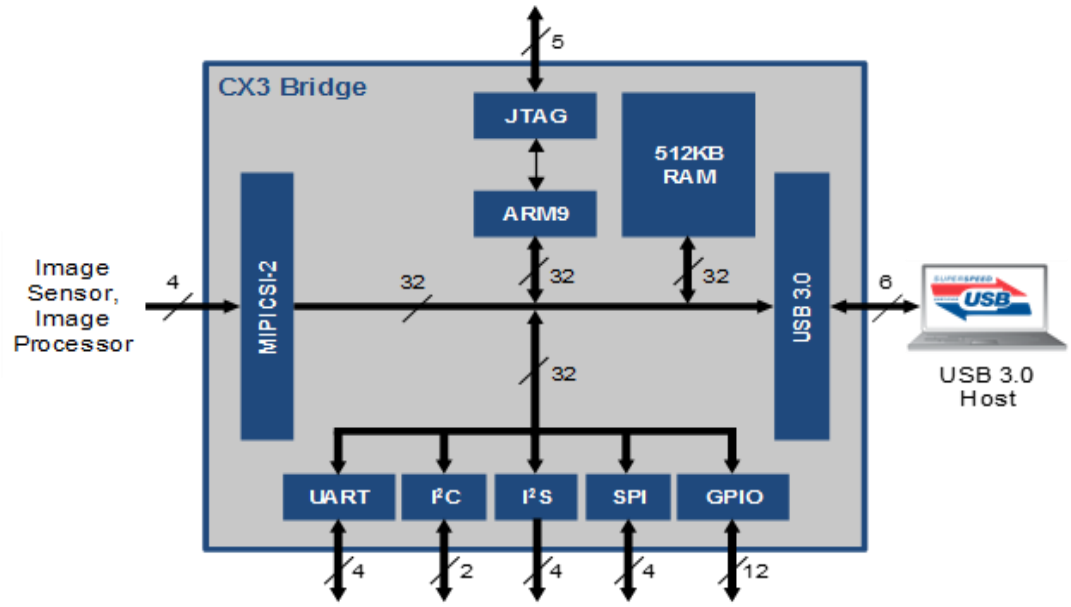
Cypress EZ-USB® CX3 is a USB 3.0 camera controller that enables developers to add USB 3.0 connectivity to any image sensors conforming to the Mobile Industry Processor Interface (MIPI) Camera Serial Interface Type 2 (CSI-2) standard. It serves as a MIPI CSI-2-to-USB Bridge.

EZ-USB CX3 is a variant of the EZ-USB FX3 device that features an integrated MIPI CSI-2 receiver mated to the general programming interface II (GPIF II). CX3 provides the ability to add SuperSpeed USB connectivity to image sensors supporting the MIPI CSI-2 interface.

CX3 conforms to the MIPI CSI-2 specification (version 1.01) and supports up to four data lanes with speed up to 1 gigabits per second (Gbps) per lane for a total bandwidth of 2.4 Gbps.

CX3 is ideally suited for high-definition or high-speed image capturing applications and is capable of streaming uncompressed video up to 1080p at 30 fps or 720p at 60 fps. CX3 supports a wide variety of image formats including RAW8/10/12/14, YUV422, RGB888/666/565, and user-defined 8-bit.

Figure 1: EZ-USB® CX3 Device



Based on the proven EZ-USB FX3 platform, CX3 includes an ARM9™ CPU and 512 KB SRAM that provide 200 MIPS of computational power. CX3 supports multiple peripheral interfaces such as I^2C, SPI, and UART, which can be programmed to support Autofocus, Pan, Tilt, and Zoom (PTZ), or other camera control functions.

CX3 uses the same application development tools as FX3. The FX3 Software Development Kit provides support for CX3 along with application examples for accelerating time to market. CX3 complies with the USB 3.0 v1.0 specification and therefore is backward compatible with USB 2.0.

This technical reference manual (TRM) is a supplement to the *EZ-USB FX3 TRM*, and provides details about the added CX3 MIPI CSI-2 receiver function block. Sections 1.5 through 1.9 of this TRM supplement describe the MIPI CSI-2 receiver functional block, including the fixed-function GPIF II state machine. Section 1.10 provides the register details for the MIPI CSI-2 receiver interface. Detailed descriptions of the existing FX3

functional blocks, such as the CPU Subsystem, Memory, Global Controller, DMA, USB, and low-bandwidth (serial and GPIO) peripherals are available in the *EZ-USB FX3 TRM*. Technical terms used in this TRM are defined in the Glossary.

Figure 2 is the block diagram of FX3. Certain functional blocks are not included in CX3; these are shown in red. The GPIF II, shown in blue, is included in CX3, but only with functionality specific to the camera interface.

Figure 2: FX3 Block Diagram

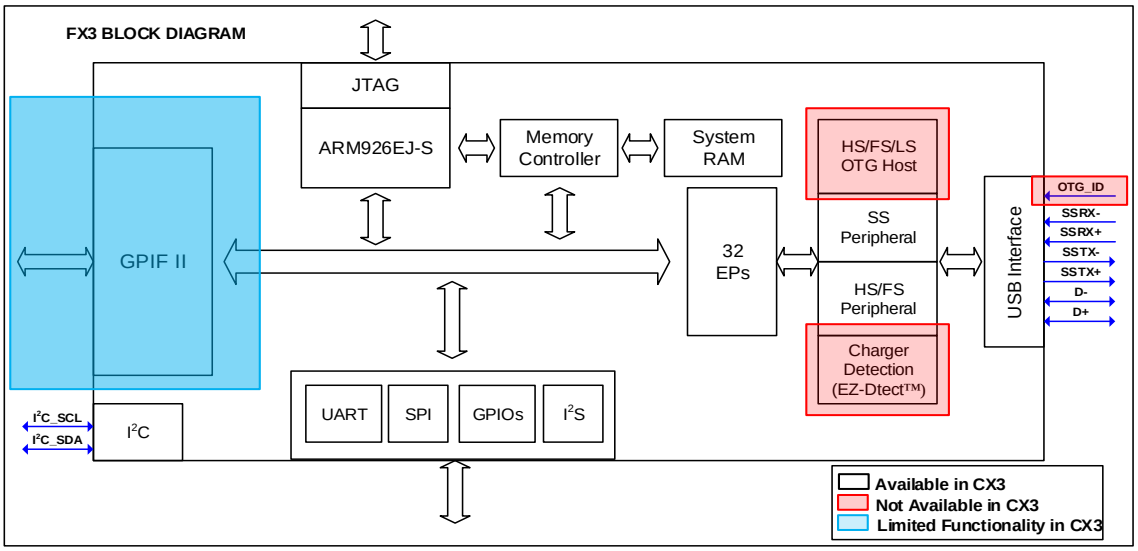
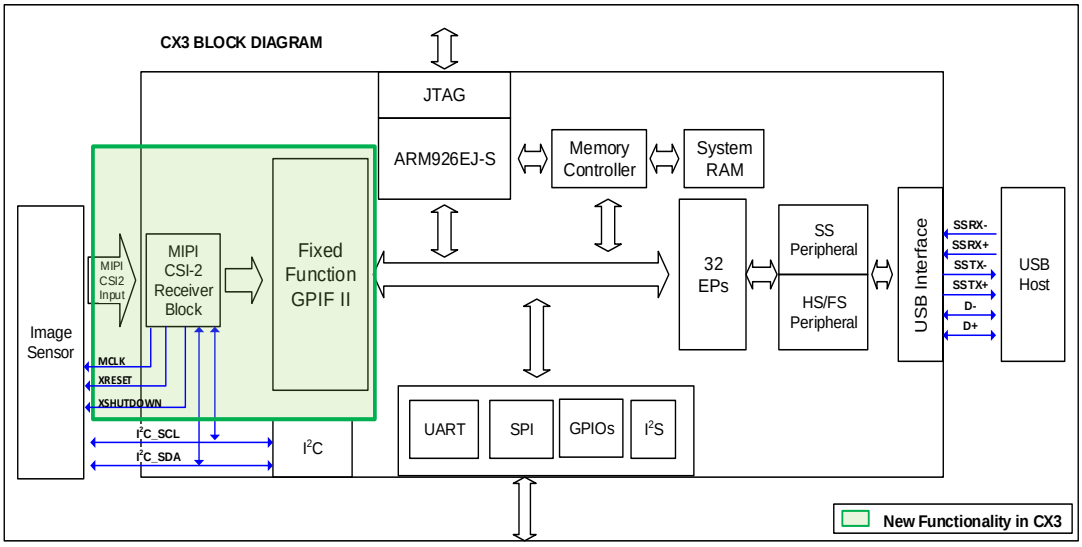


Figure 3 shows the CX3 block diagram, including the added MIPI CSI-2 functional block and the fixed-function GPIF II interface available in CX3, highlighted in green.

Figure 3: CX3 Block Diagram



The differences between CX3 and FX3 devices are listed below.

Table 1: Differences Between FX3 and CX3 Devices

Feature	Supported in FX3	Supported in CX3
P-Port Support		
Sync ADMux implemented with GPIF II	Yes	No
Async SRAM implemented with GPIF II	Yes	No
Async ADMux implemented with GPIF II	Yes	No
MMC Slave	Yes	No
GPIF II	Yes	Fixed-function GPIF II state machine implemented to interface with the CX3 MIPI CSI-2 receiver block.
Low-Bandwidth Peripherals		
I^2S Master (Transmitter only)	Yes	Yes
SPI Master	Yes	Yes
UART	Yes	Yes
I^2C Master Controller	Yes	Yes
U-Port Support		
USB 3.0 Peripheral	Yes	Yes
USB 2.0 Peripheral	Yes	Yes
32 Physical endpoints	Yes	Yes
Charger Detection 1.1 Support (EZ-Dtect™)	Yes	No
Accessory Charger Adaptor (ACA) Support	Yes	No
Integrated Hi-Speed USB Switch	Yes	No
Carkit UART Pass-through mode	Yes	No
USB OTG (Hi-Speed, Full-Speed, Low-Speed host or peripheral)	Yes	No
Clock input frequencies:	19.2 MHz, 26 MHz, 38.4 MHz, 52 MHz	19.2 MHz
Crystal input: 19.2 MHz	Yes	No
JTAG Support for debug only (Boundary Scan not supported)	Yes	Yes
VBUS, VBAT signals	Separate VBUS, VBAT signals for 5-V and 3.3-V operation	VBUS, VBAT combined to a single signal VUSB. Supports 3.3-V and 5-V operation.
GPIOs	42	12
Clock output for Image sensor	No	Yes
MIPI CSI-2 input	No	Yes

1.2 CX3 Features

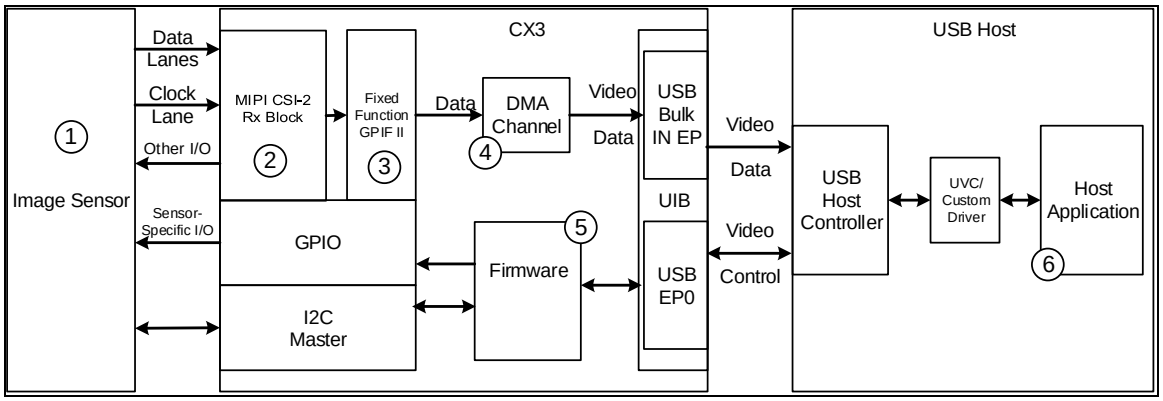
CX3 supports the following features:

- USB 3.0 and USB 2.0 peripheral controller compliant with the USB 3.0 specification 1.0
- MIPI CSI-2 RX Interface
  - MIPI CSI-2 conformant (Version 1.01 Revision 0.04 –April 2, 2009)
  - Supports up to four data lanes; each lane supports up to 1 Gbps
  - Camera Control Interface (over I^2C) support for image sensor configuration
- Supports the following video data formats:
  - RAW 8/10/12/14
  - YUV 422 (8/10bit)
  - RGB 888 / 666 / 565
  - User-defined 8-bit
- Twelve GPIOs for controlling camera-related functions (for example, lighting, sync in, sync out, etc.)
- Support for I^2C, SPI, I^2S outputs, and UART interfaces, identical to FX3
- JTAG interface for debugging

1.3 Block Diagram

Figure 4 is a detailed block diagram of a typical system using CX3 to transfer data from an image sensor to a USB Host.

Figure 4: System Block Diagram



The main sub-blocks of the block diagram are numbered, and the tasks executed by each sub-block are described below:

1. The MIPI CSI-2-based image sensor connects to CX3 and is configured using the Camera Control Interface (I^2C) bus.
2. The CX3 MIPI CSI-2 receiver block reads the data from the image sensor, de-serializes it, merges lanes, de-packetizes it, and then sends it as a parallel input to the fixed function GPIF II block.
3. The GPIF II block and its fixed-function state machine make the image sensor data available to the USB interface using a DMA channel.
4. The DMA channel moves the image data from the GPIF II block to the USB Interface Block (UIB).
5. The CX3 firmware initializes the CX3 hardware blocks, configures the image sensor and MIPI CSI-2 controller, controls the USB interface, and services all USB protocol requests. The CX3 firmware can be customized to add class-specific headers to the video stream data before it is committed to the USB interface.
6. A host application such as an image signal processor or a video stream player provides control requests to configure the video stream and sensor, and processes and renders the video stream on the host PC.

1.4 MIPI CSI-2 Block Configuration APIs

The MIPI CSI-2 Rx-block configuration APIs allow user applications to initialize, configure, and perform power management for the camera interface. These APIs communicate with the camera over the I^2C bus.

For modularity, the MIPI CSI-2 configuration APIs are compiled into a separate API library (*cyu3mipicsi.a*). This library is linked with the application only if its functionality is required.

The main APIs provided in the API library are as follows:

Table 2: CX3-Specific APIs in the EZ-USB FX3 SDK

API Name	API Description
CyU3PMipicsiInit()	API to initialize the MIPI CSI-2 block.
CyU3PMipicsiDeInit()	API to de-initialize the MIPI CSI-2 block.
CyU3PMipicsiSetIntfParams()	API to configure the clocks and interface settings on the MIPI CSI-2 block.
CyU3PMipicsiQueryIntfParams()	API to query settings from the MIPI CSI-2 block.
CyU3PMipicsiSleep()	API to place the MIPI CSI-2 block in the low-power sleep mode.
CyU3PMipicsiWakeup()	API to wake the MIPI CSI-2 block from the low-power sleep mode to active mode.
CyU3PCx3DeviceReset()	API to reset the MIPI CSI-2 block.
CyU3PCx3DeviceReset()	API to perform a warm or cold reset on the CX3 device.
CyU3PMipicsiSetSensorControl()	API to drive the MIPI CSI-2 XRESET and XSHUTDOWN signals to the image sensor. XRESET is a CX3 output signal that can be used to reset the image sensor. XSHUTDOWN is a CX3 output signal that can be used to control image sensor power modes.
CyU3PMipicsiGpifLoad()	API to load the fixed-function GPIF II waveform and configure the fixed-function GPIF II bus widths and DMA buffer size.

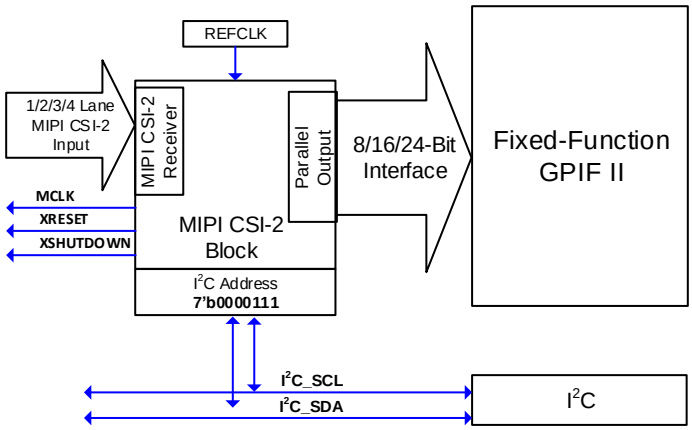
More information on the APIs is available in Section 1.11

Detailed API documentation is available in the *MIPI CSI-2 and Fixed-Function GPIF Interface for CX3* section of the *EZ-USB FX3 SDK Firmware API Guide*, available as part of the EZ USB FX3 SDK.

1.5 MIPI CSI-2 Block

The CX3 device has an integrated MIPI CSI-2 block, which is hard-wired to the GPIF II interface on one side and provides a MIPI CSI-2 interface on the other side to interface to an image sensor that supports MIPI CSI-2. The block supports up to four MIPI CSI-2 data lanes, and is capable of speeds up to 1 Gbps per lane. The MIPI CSI-2 receiver is connected to a fixed-function GPIF II controller via an 8-, 16-, or 24-bit data bus, which can be clocked up to 100 MHz. The maximum bandwidth that can be achieved is 2.4Gbps (i.e. 24 * 100 Mbps). The MIPI CSI-2 block is configured over I^2C and is available on the CX3 I^2C bus at the 7-bit I^2C slave address 7'b0000111.

Figure 5. CX3 MIPI CSI-2 Block



1.6 CX3 MIPI CSI-2 Stream Formats

The MIPI CSI-2 Rx block on CX3 natively supports the following stream formats and output modes:

Table 3: CX3 MIPI CSI-2 Stream Formats

Format and Mode	CX3 Firmware Stream Format Name	Format Description, Pixel Depth	CSI-2 Data Typea	GPIF II Bus Width	Output Stream
RAW8	CY_U3P_CSI_DF_RAW8	RAW format, 8 bits per pixel	0x2A	8-bit	RAW [7:0]
RAW10	CY_U3P_CSI_DF_RAW10	RAW format, 10 bits per pixel	0x2B	16-bit	6'b0, RAW [9:0]
RAW12	CY_U3P_CSI_DF_RAW12	RAW format, 12 bits per pixel	0x2C	16-bit	4'b0, RAW [11:0]
RAW14	CY_U3P_CSI_DF_RAW14	RAW format, 14 bits per pixel	0x2D	16-bit	2'b0,RAW[13:0]
RGB888	CY_U3P_CSI_DF_RGB888	RGB 888 format, 24 bits per pixel	0x24	24-bit	R[7:0], G[7:0], B[7:0]
RGB666 Mode 0	CY_U3P_CSI_DF_RGB666_0	RGB 666 format, 24 bits per pixel	0x23	24-bit	2'b0, R[5:0], 2'b0, G[5:0], 2'b0, B[5:0]
RGB666 Mode 1	CY_U3P_CSI_DF_RGB666_1	RGB 666 format, 24 bits per pixel	0x23	24 bit	6'b0, R[5:0], G[5:0], B[5:0]
RGB565 Mode 0	CY_U3P_CSI_DF_RGB565_0	RGB 565 format, 24 bits per pixel	0x22	24-bit	2'b0, R[4:0], 3'b0, G[5:0], 2'b0, B[4:0], 1'b0
RGB565 Mode 1	CY_U3P_CSI_DF_RGB565_1	RGB 565 format, 24 bits per pixel	0x22	24-bit	3'b0, R[4:0], 2'b0, G[5:0], 3'b0,B[4:0]
RGB565 Mode 2	CY_U3P_CSI_DF_RGB565_2	RGB 565 format, 16 bits per pixel	0x22	16-bit	R[4:0], G[5:0], B[4:0]
YUV422 8-bit Mode 0	CY_U3P_CSI_DF_YUV422_8_0	YUV422 format 16 bits per pixel	0x1E	8-bit	P[7:0] Data Order: U1,Y1,V1,Y2,U3, Y3,..

[a] MIPI CSI-2 defined Data Type code. Please refer to the MIPI CSI-2 specification for details.

Format and Mode	CX3 Firmware Stream Format Name	Format Description, Pixel Depth	CSI-2 Data Typea	GPIF II Bus Width	Output Stream
YUV422 8-bit Mode 1	CY_U3P_CSI_DF_YUV422_8_1	YUV422 format 16 bits per pixel	0x1E	16-bit	P[15:0] Data Order: {U1,Y1},{V1,Y2},{ U3, Y3},{V3,Y4}...
YUV422 8-bit Mode 2	CY_U3P_CSI_DF_YUV422_8_2	YUV422 format 16 bits per pixel	0x1E	16-bit	P[15:0] Data Order: {Y1,U1},{Y2,V1},{ Y3, U3},{Y4,V3}....
YUV422 10-bit	CY_U3P_CSI_DF_YUV422_10	YUV422 format 20 bits per pixel	0x1F	16-bit	6'b0,P[9:0] Data Order: U1,Y1,V1,Y2,U3, Y3, V3,Y4.

[a] MIPI CSI-2 defined Data Type code. Please refer to the MIPI CSI-2 specification for details.

If the GPIF II bus width selected is larger than the width of the output stream, for example if a 24-bit GPIF II bus width is used for the CY_U3P_CSI_DF_YUV422_8_1 type, the upper bits on the GPIF II are padded with 0s.

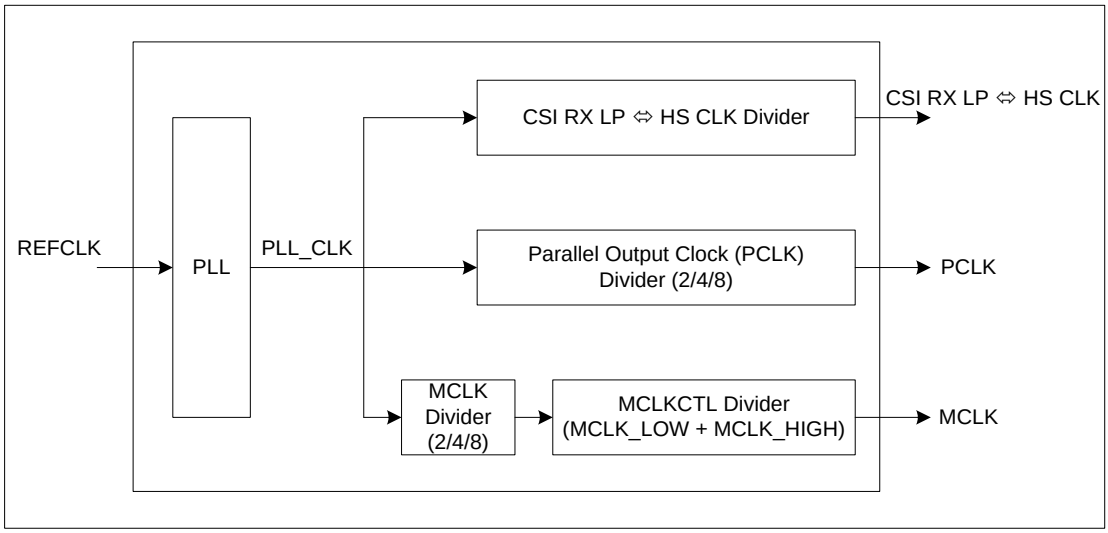
For stream formats not listed in Table 3, such as MJPEG or custom format streams, 8 bit wide streams can be transported on the RAW8 stream format, and 24 bit wide streams can be transported by treating the stream as an RGB888 format stream. GPIF bus widths and buffers should be set up appropriately to match the stream output. Section 1.11.12 has more details on setting up the GPIF bus-width and DMA buffers.

Packing more than one pixel per PCLK is possible. For example, selecting the “MIPI CSI input -Data format" configured as "RAW8" and the “MIPI interface configuration – data format” as a 24bit format (RGB888) will output three pixel data per PCLK. Similarly, two 10 bit or 12bit pixels can be packed and output per PCLK using the 24 bit output format.

1.7 MIPI CSI-2 Block Clocks

Figure 6 shows the CX3 clocks on the MIPI CSI-2 block. The interface takes a reference clock as its input and generates the required clocking using a PLL followed by multiple clock dividers. In the CX3 application firmware, clock configuration parameters are a part of the *CyU3PMipicsiCfg_t* structure, which is passed to the *CyU3PMipicsiSetIntfParams()* API to configure the CSI-2 block. Details of the structure and configuration API can be found in the EZ-USB FX3 SDK Firmware API Guide.

Figure 6. CX3 MIPI CSI-2 Block Clocks



A brief description of each of the clocks is provided in the following sections.

1.7.1 Reference Clock (REFCLK)

This is the reference clock input provided to the MIPI CSI-2 block. This input clock should be between 6 and 40 MHz.

1.7.2 PLL Clock (PLL_CLK)

The PLL_CLK is the primary clock of the MIPI CSI-2 block. The value for PLL clock should be between 62.5 MHz and 1 GHz. All other internal and output clocks are derived from this clock.

The PLL clock frequency is generated from the input reference clock using the following equation:

PLL_CLK = REFCLK * [(PLL_FBD + 1) / (PLL_PRD + 1)] / (2^ PLL_FRS)

Where

PLL_FBD is the feedback divider whose range is between 0 and 0x1FF.

PLL_PRD is the input divider whose range is between 0 and 0x0F.

PLL_FRS is the frequency range selection parameter that takes the following values:

‘0’ if the PLL clock is between 500 MHz and 1 GHz.

‘1’ if the PLL clock is between 250 MHz and 500 MHz.

‘2’ if the PLL clock is between 125 MHz and 250 MHz.

‘3’ if the PLL clock is between 62.5 MHz and 125 MHz.

Sample computations for the PLL clock frequency are provided for a REFCLK value of 19.2 MHz in the following table:

Table 4: PLL Clock Frequency Calculation example

REFCLK in MHz	PLL_PRD	PLL_FBD	PLL_FRS	PLL_CLK in MHz
19.2	1	69	0	672
19.2	1	69	2	168
19.2	3	201	1	484.8
19.2	2	97	3	78.4
19.2	2	125	1	403.2

1.7.3 CSI RX LP ←→ HS Clock

This clock is used for detecting the CSI Link Low Power (LP) ←→ High Speed (HS) transition. It is generated by dividing the PLL_CLK by a value of 2, 4, or 8. Details regarding the CSI Link LP ←→ HS transitions can be found in the MIPI CSI-2 specification documentation available from the MIPI alliance (http://www.mipi.org/specifications/camera-interface).

The maximum value for this clock is 125 MHz.

This clock frequency is calculated automatically by the “MIPI Receiver configuration tool”.

1.7.4 Output Parallel Clock (PCLK)

This clock is the PCLK output, which drives the fixed-function GPIF II interface on CX3. It is generated by dividing the PLL_CLK by a value of 2, 4, or 8.

The maximum value for this clock is 100 MHz.

This clock frequency is calculated automatically by the “MIPI Receiver configuration tool”.

1.7.5 Image Sensor Reference Clock (MCLK)

The MCLK is an optional clock output that can be used as the input reference clock for the image sensor. It is sourced from the PLL_CLK by first dividing down using the mClkRefDiv (2/4/8) and then dividing further using the MCLKCTL divider. The MCLKCTL divider specifies the HIGH time and LOW time counted by the divided down PLL_CLK.

The upper eight bits define the HIGH time count (1-255) and the lower eight bits define the LOW time count (1-255).

MCLK is computed using the following equation:

MCLK = (PLL_CLK/mClkRefDiv) / [(HighByte (mClkCtl) + 1) + (LowByte (mClkCtl) + 1)]

For example:

If the PLL_CLK is 672 MHz, to generate MCLK of 24 MHz, set mClkRefDiv to 4 and mClkCtl to 0x0203.

Thus:

MCLK = (672 / 4) / ((2 + 1) + (3 + 1))

= 168 / 7 = 24 MHz

MCLK is the only output when both HighByte (MClkCtl) and LowByte (MclkCtl) are non-zero.

This clock frequency is calculated automatically by the “MIPI Receiver configuration tool”.

1.8 CX3 MIPI CSI-2 Block Power Modes

The MIPI CSI-2 block on CX3 supports the following power modes:

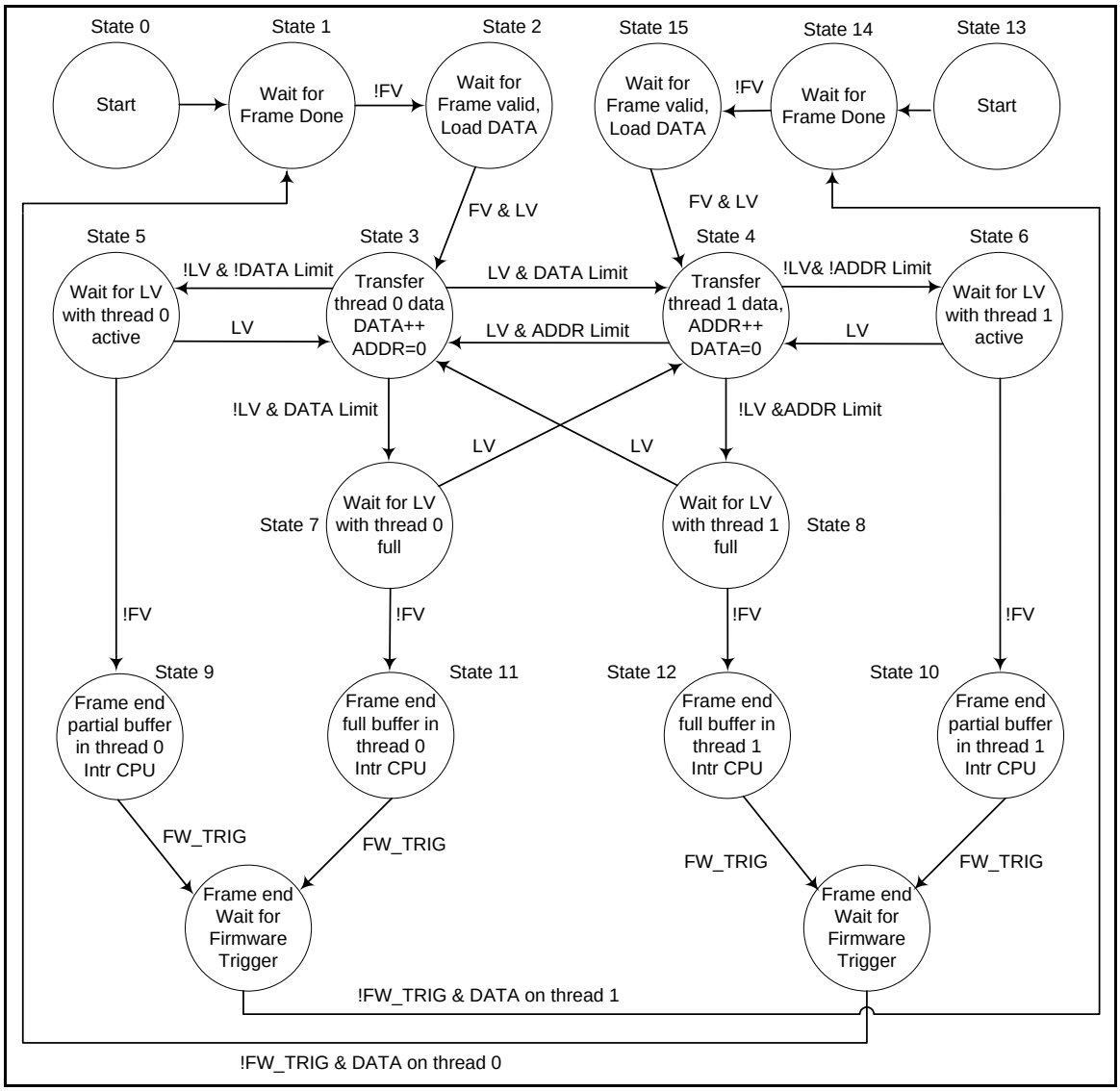
- **Active**: In this mode, the MIPI CSI-2 receiver block will be active and transfer data from the image sensor to the GPIF II interface.
- **Low-Power Sleep**: In this mode, all data transfer clocks on the MIPI CSI-2 block are stopped and there will be no data transfer. All register settings are retained in this state.
- **Soft Reset**: This state puts the MIPI CSI-2 block into reset mode where all clocks on the interface are stopped. The MIPI CSI-2 block configuration register states are retained across a soft reset.
- **Hard Reset**: In this state the MIPI CSI-2 controller and all clocks are stopped. The MIPI CSI-2 block configuration registers are reset to their default states by a hard reset. Hard reset is automatically executed during MIPI CSI-2 receiver initialization.

1.9 Fixed-Function GPIF II Interface on CX3

The fixed-function GPIF II state machine of CX3 is shown in Figure 7. This state machine makes the parallel data provided by the MIPI CSI-2 receiver available for transfer over two sockets, which can be connected to a manual, Many-to-One DMA channel.

The functionality of this state machine is similar to the GPIF II state machine described in *Application Note AN75779 - How to Implement an Image Sensor Interface with EZ-USB® FX3™ in a USB Video Class (UVC) Framework*.

Figure 7. CX3 GPIFII State Machine



The FX3 SDK (Software Development Kit) provides the *CyU3PMipicsiGpifLoad()* API to load this fixed-function state machine into the CX3 GPIF II block. This API allows selection of the bus width (8, 16, or 24 bits) to match the output from the parallel interface on the MIPI CSI-2 block, and also allows setting of the DMA buffer size to be used for the GPIF II interface. The buffer size calculation is explained in *CyU3PMipicsiGpifLoad() API documentation in section 1.11.12*

1.10 MIPI CSI-2 Block Registers

The MIPI CSI-2 block exposes a set of registers used to configure the block over I²C. The block is available at the 7-bit I²C slave address **7'b0000111** (Read address **0x0F**; Write address **0x0E**). The block supports 100 kHz and 400 kHz I²C operation.

Note: No other I²C device with the same address as the MIPI CSI-2 block should be connected to the I²C bus of the CX3 device.

The registers are 16-bit aligned, and data transfers are performed Most Significant Bit (MSB) first.

Figure 8 and Figure 9 show typical I²C Read and Write transfers to the MIPI CSI-2 block.

Figure 8. Write Transfer Sequence

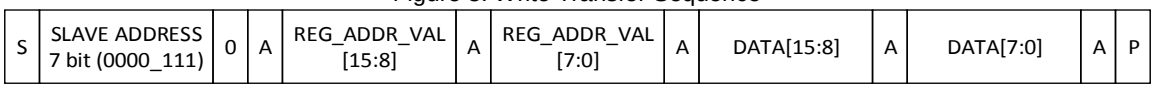


Figure 9. Read Transfer Sequence

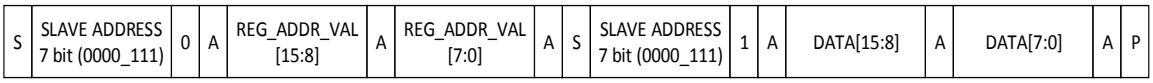


Table 5 provides a list of the configuration registers included in the CX3 MIPI CSI-2 block.

Table 5: MIPI CSI-2 Block Configuration Registers

Register Address	Register Name	Description
0x0002	CX3_SYSTEM_CTRL	System Control Register
0x0004	CX3_CONFIG_CTRL	Configuration Control Register
0x0006	CX3_FIFO_CTRL	FIFO Control register
0x0008	CX3_DATA_FMT	Data Format Control Register
0x000C	CX3_MCLK_CTRL	MCLK Control Register
0x0010	CX3_CSI_ SENSOR_SIG_EN	Register to Enable MIPI CSI-2 signals (XRESET and XSHUTDOWN)
0X0014	CX3_CSI_SENSOR_SIG_VAL	Register to Control MIPI CSI-2 signals (XRESET and XSHUTDOWN)
0x0016	CX3_PLL_CTRL0	PLL Clock Control Register 0
0x0018	CX3_PLL_CTRL1	PLL Clock Control Register 1
0x0020	CX3_CLK_CTRL	Clock Control Register
0x0022	CX3_BYTE_COUNT	Byte Count Register
0x0060	CX3_PHY_TIME_DELAY	Register to set MIPI THS-Settle timer settings.

The following sections provide detailed descriptions of each of the CX3 MIPI CSI-2 configuration registers.

Note: RESERVED Register Bits should not be changed. When writing to a register with RESERVED bits, you should ensure that the values are not changed during a Write (preferably by doing a Read before the Write and maintaining the values for the RESERVED bits during the Write).

1.10.1 CX3_SYSTEM_CTRL (Register Address: 0x0002)

This register controls the Sleep and Software Reset functionality for the MIPI CSI-2 block.

This register is used by the *CyU3PMipicsiSleep()*, *CyU3PMipicsiWakeup()*, and *CyU3PMipicsiReset()* APIs.

BIT	15	14	13	12	11	10	9	8
NAME	RESERVED							
BIT	7	6	5	4	3	2	1	0
NAME	RESERVED						SLEEP	RESET

Register Field	Bit	Description
RESERVED	[15:2]	RESERVED. Firmware must preserve their settings by reading them, changing non-reserved bits, and re-writing them.
SLEEP	[1]	Sleep Control: Places the MIPI CSI-2 block into low-power sleep mode. 0: Normal Operation 1: Sleep Mode
RESET	[0]	Software Reset: Forces software reset of the MIPI CSI-2 block. Does not clear the configuration registers. 0: Normal Operation 1: Reset Operation

1.10.2 CX3_CONFIG_CTRL (Register Address: 0x0004)

This register controls number of data lanes and the output data mode for the MIPI CSI-2 block.

This register is set by the *CyU3PMipicsiSetIntfParams()* API and queried using the *CyU3PMipicsiQueryIntfParams()* API.

BIT	15	14	13	12	11	10	9	8
NAME	RESERVED						DATA MODE	
BIT	7	6	5	4	3	2	1	0
NAME	RESERVED	OUTEN	RESERVED				DATA LANES	

Register Field	Bit	Description
RESERVED	[15:10]	RESERVED. Firmware must preserve their settings by reading them, changing non-reserved bits, and re-writing them.
DATA MODE	[9:8]	Data Mode Selection: Sets the data mode for the data format selected using the DATA FORMAT bits of the CX3_DATA_FMT register. The combination of DATA MODE and the DATA FORMAT is used to select the output stream as defined in Table 3. The DATA FORMAT setting determines the output data format, while the DATA MODE setting determines the data packing and byte ordering for the DATA TYPE as defined in Table 3. 2'b00: Mode 0 2'b01: Mode 1 2'b10: Mode 2 2'b11: Reserved
RESERVED	[7]	RESERVED. Firmware must preserve their settings by reading them, changing non-reserved bits, and re-writing them.
OUTEN	[6]	Enable Parallel Output: Enables parallel output from MIPI CSI-2 receiver block to fixed-function GPIF II. 0: Disable Parallel Output. 1: Enable Parallel Output
RESERVED	[5:2]	RESERVED. Firmware must preserve their settings by reading them, changing non-reserved bits, and re-writing them.
DATA LANES	[1:0]	MIPI CSI-2 Data Lane Selection: Selects the number of data lanes to be used. 2'b00: 1 Data Lane 2'b01: 2 Data Lanes 2'b10: 3 Data Lanes 2'b11: 4 Data Lanes

1.10.3 CX3_FIFO_CTRL (Register Address: 0x0006)

This register determines the FIFO trigger level for the initiation of parallel data output from the parallel output buffer of the MIPI CSI-2 block. The MIPI CSI-2 block waits for the parallel output buffer to reach the level specified by this register before transferring data to GPIF II interface.

This register is set by the *CyU3PMipicsiSetIntfParams()* API and queried using the *CyU3PMipicsiQueryIntfParams()* API.

BIT	15	14	13	12	11	10	9	8
NAME	RESERVED							FIFO LEVEL[8]
BIT	7	6	5	4	3	2	1	0
NAME	FIFO LEVEL [7:0]							

Register Field	Bit	Description
RESERVED	[15:9]	RESERVED. Firmware must preserve their settings by reading them, changing non-reserved bits, and re-writing them.
FIFO LEVEL	[8:0]	FIFO Level: Determines the FIFO Write trigger level. The MIPI CSI-2 block starts parallel output to the GPIF II only when the output buffer reaches this level. Range: 0x000-0x1FF

1.10.4 CX3_DATA_FMT (Register Address: 0x0008)

This register controls the output data format for the MIPI CSI-2 block.

This register is set by the *CyU3PMipicsiSetIntfParams()* API and queried using the *CyU3PMipicsiQueryIntfParams()* API.

BIT	15	14	13	12	11	10	9	8
NAME	RESERVED							
BIT	7	6	5	4	3	2	1	0
NAME	DATA FORMAT				RESERVED			DATA FMT ENABLE

Register Field	Bit	Description
RESERVED	[15:8]	RESERVED. Firmware must preserve their settings by reading them, changing non-reserved bits, and re-writing them.
DATA FORMAT	[7:4]	Data Format Selection: Selects the output data format. 4'b0000: RAW8 4'b0001: RAW10 4'b0010: RAW12 4'b0011: RGB888 4'b0100: RGB666 4'b0101: RGB565 4'b0110: YUV422-8-bit 4'b0111: RESERVED 4'b1000: RAW14 4'b1001: YUV422-10-bit 4'b1010-1111: RESERVED This field, along with the DATA MODE field from the CX3_CONFIG_CTRL register, is used to select the output stream format defined in Table 3. For stream formats not listed in Table 3, such as MJPEG or custom format streams, 8 bit wide streams can be transported on the RAW8 stream format, and 24 bit wide streams can be transported by treating the stream as an RGB888 format stream.
RESERVED	[3:1]	RESERVED. Firmware must preserve their settings by reading them, changing non-reserved bits, and re-writing them.
DATA FMT ENABLE	[0]	Data Format Enable: 0: Not supported. 1: Normal operation.

1.10.5 CX3_MCLK_CTRL (Register Address: 0x000C)

This register configures the MCLK divider and controls the Image Sensor Reference Clock (MCLK) output from the MIPI CSI-2 block. Detailed information on calculating the MCLK value is provided in Section 1.7.5 .

This register is set by the *CyU3PMipicsiSetIntfParams()* API and queried using the *CyU3PMipicsiQueryIntfParams()* API.

BIT	15	14	13	12	11	10	9	8
NAME	MCLK HIGH							
BIT	7	6	5	4	3	2	1	0
NAME	MCLK LOW							

Register field	Bit	Description
MCLK HIGH	[15:8]	MCLK HIGH Time Count: HIGH time count for the MCLK divider. See Section 1.7.5 for details on this field. MCLK is enabled only if MCLK HIGH and MCLK LOW are non-zero.
MCLK LOW	[7:0]	MCLK LOW Time Count: LOW Time count for MCLK divider. See Section 1.7.5 for details on this field. MCLK Divider = (MCLK HIGH +1) + (MCLK LOW+1)

1.10.6 CX3_CSI_SENSOR_SIG_EN (Register Address: 0x0010)

This register enables or disables the output of the XSHUTDOWN and XRESET signals from the MIPI CSI-2 block.

This register is set by the *CyU3PMipicsiInit()* and *CyU3PMipicsiReset()* APIs.

BIT	15	14	13	12	11	10	9	8
NAME	RESERVED							
BIT	7	6	5	4	3	2	1	0
NAME	RESERVED					XSHUTDOWN ENABLE	XRESET ENABLE	RESERVED

Register field	Bit	Description
RESERVED	[15:3]	RESERVED. Firmware must preserve their settings by reading them, changing non-reserved bits, and re-writing them.
XSHUTDOWN ENABLE	[2]	Enable MIPI CSI-2 XSHUTDOWN Signal: 0: Enables the output for the MIPI CSI-2 XSHUTDOWN signal. 1: Disables output for the MIPI CSI-2 XSHUTDOWN signal. The signal drive value is determined by CX3_CSI_SENSOR_SIG_VAL (Register Address: 0x0014) [2].
XRESET ENABLE	[1]	Enable MIPI CSI-2 XRESET Signal: 0: Enables the output for the MIPI CSI-2 XRESET signal. 1: Disables the output for the MIPI CSI-2 XRESET signal. The signal drive value is determined by CX3_CSI_SENSOR_SIG_VAL (Register Address: 0x0014) [1].
RESERVED	[0]	RESERVED. Firmware must preserve their settings by reading them, changing non-reserved bits, and re-writing them.

1.10.7 CX3_CSI_SENSOR_SIG_VAL (Register Address: 0x0014)

This register configures the drive value for the XSHUTDOWN and XRESET signals for the MIPI CSI-2 block.

This register is set using the *CyU3PMipicsiSetSensorControl()* API.

BIT	15	14	13	12	11	10	9	8
NAME	RESERVED							
BIT	7	6	5	4	3	2	1	0
NAME	RESERVED					XSHUTDOWN OUTPUT	XRESET OUTPUT	RESERVED

Register Field	Bit	Description
RESERVED	[15:3]	RESERVED. Firmware must preserve their settings by reading them, changing non-reserved bits, and re-writing them.
XSHUTDOWN OUTPUT	[2]	Drive MIPI CSI-2 XSHUTDOWN Signal: 0: Drive XSHUTDOWN LOW. 1: Drive XSHUTDOWN HIGH. The signal is only driven when CX3_CSI_SENSOR_SIG_EN (Register Address: 0x0010) [2] is 0.
XRESET OUTPUT	[1]	Drive MIPI CSI-2 XRESET Signal: 0: Drive XRESET LOW. 1: Drive XRESET HIGH. The signal is only driven when CX3_CSI_SENSOR_SIG_EN (Register Address: 0x0010) [1] is 0.
RESERVED	[0]	RESERVED. Firmware must preserve their settings by reading them, changing non-reserved bits, and re-writing them.

1.10.8 CX3_PLL_CTRL0 (Register Address: 0x0016)

This register configures the PLL clock on the MIPI CSI-2 block. Detailed description of how the PLL Clock is generated based on the values from CX3_PLL_CTRL0 (Register Address: 0x0016) and CX3_PLL_CTRL1 (Register Address: 0x0018) is provided in Section 1.7.2

This register is set by the *CyU3PMipicsiSetIntfParams()* API and queried using the *CyU3PMipicsiQueryIntfParams()* API.

BIT	15	14	13	12	11	10	9	8
NAME	PLL PRD				RESERVED			PLL FBD [8]
BIT	7	6	5	4	3	2	1	0
NAME	PLL FBD [7:0]							

Register Field	Bit	Description
PLL PRD	[15:12]	Input Divider: Input divider for PLL generation. See Section 1.7.2 for details on this field. Range: 0x0 - 0xF.
RESERVED	[11:9]	RESERVED. Firmware must preserve their settings by reading them, changing non-reserved bits, and re-writing them.
PLL FBD	[8:0]	Feedback Divider: Feedback divider for PLL generation. See Section 1.7.2 for details on this field. Range: 0x000 - 0x1FF.

1.10.9 CX3_PLL_CTRL1 (Register Address: 0x0018)

This register configures the PLL clock on the MIPI CSI-2 block. Detailed description of how the PLL Clock is generated based on the values from CX3_PLL_CTRL0 (Register Address: 0x0016) and CX3_PLL_CTRL1 (Register Address: 0x0018) is provided in Section 1.7.2 .

This register is set by the *CyU3PMipicsiSetIntfParams()* API and queried using the *CyU3PMipicsiQueryIntfParams()* API.

BIT	15	14	13	12	11	10	9	8
NAME	RESERVED				PLL FRS		RESERVED	
BIT	7	6	5	4	3	2	1	0
NAME	RESERVED			CLOCK ENABLE	RESERVED		PLL ENABLE	

Register Field	Bit	Description
RESERVED	[15:12]	RESERVED. Firmware must preserve their settings by reading them, changing non-reserved bits, and re-writing them.
PLL FRS	[11:10]	Frequency Range Selection: Determines the PLL frequency range. See Section 1.7.2 for details on this field. 2'b00: 500-1000 MHz PLL frequency 2'b01: 250-500 MHz PLL frequency 2'b10: 125-250 MHz PLL frequency 2'b11: 62.5-250 MHz PLL frequency
RESERVED	[9:5]	RESERVED. Firmware must preserve their settings by reading them, changing non-reserved bits, and re-writing them.
CLOCK ENABLE	[4]	Clock Enable: 0: Clocks on the MIPI CSI-2 block are switched off. 1: Clocks on the MIPI CSI-2 block are enabled (Normal Operation).
RESERVED	[3:2]	RESERVED. Firmware must preserve their settings by reading them, changing non-reserved bits, and re-writing them.
PLL ENABLE	[1:0]	Enable PLL Clock: The setting to enable the PLL clock. 2'b00: PLL clock disabled 2'b11: PLL clock enabled (Normal Operation) 2b'01- 2b'10: RESERVED

1.10.10 CX3_CLK_CTRL (Register Address: 0x0020)

This register configures the interface clock dividers for the MIPI CSI-2 block.

This register is set by the *CyU3PMipicsiSetIntfParams()* API and queried using the *CyU3PMipicsiQueryIntfParams()* API.

BIT	15	14	13	12	11	10	9	8
NAME	RESERVED							
BIT	7	6	5	4	3	2	1	0
NAME	RESERVED		CSI RX CLK DIV		MCLK REF DIV		PAR OUT CLK DIV	

Register Field	Bit	Description
RESERVED	[15:6]	RESERVED. Firmware must preserve their settings by reading them, changing non-reserved bits, and re-writing them.
CSI RX CLK DIV	[5:4]	Clock Divider for CSI RX LP←→ HS Transition Clock: Divides down from the PLL clock to generate this clock. See Section 1.7.3 for details. 2'b00: PLL CLOCK/8 2'b01: PLL CLOCK/4 2'b10: PLL CLOCK/2 2'b00: RESERVED This frequency must be between 66-125 MHz.
MCLK REF DIV	[3:2]	Clock Divider for MCLK Reference Clock: Divides down from the PLL clock to generate the MCLK reference clock. See Section 1.7.5 for details. 2'b00: PLL CLOCK/8 2'b01: PLL CLOCK/4 2'b10: PLL CLOCK/2 2'b00: RESERVED This frequency must be less than 125 MHz.
PAR OUT CLK DIV	[1:0]	Clock Divider for Parallel Output Clock (PCLK): Divides down from the PLL clock to generate the parallel output clock (which drives the GPIF II interface). See Section 1.7.4 for details. 2'b00: PLL CLOCK/8 2'b01: PLL CLOCK/4 2'b10: PLL CLOCK/2 2'b00: RESERVED This frequency cannot be greater than 100 MHz.

1.10.11 CX3_BYTE_COUNT (Register Address: 0x0022)

This register configures the byte count per active line for the MIPI CSI-2 block.

This register is set by the *CyU3PMipicsiSetIntfParams()* API and queried using the *CyU3PMipicsiQueryIntfParams()* API.

BIT	15	14	13	12	11	10	9	8
NAME	BYTE COUNT [15:8]							
TYPE	R/W							
BIT	7	6	5	4	3	2	1	0
NAME	BYTE COUNT [7:0]							
TYPE	R/W							

Register Field	Bit	Description
BYTE COUNT	[15:0]	Total number of bytes per Line: Number of bytes per line of input. This is computed as: Number of Horizontal Pixels (Active Pixels) x Number of Bytes per Pixel. For example, for a 1920x1080 RGB888 stream this value is computed as follows: Active pixels per line: 1920 Bytes per Pixel: 3 BYTE COUNT = 1920x3 = 5760 = 0x1680 BYTE COUNT [15:8] = 0x16 BYTE COUNT [7:0] = 0x80

1.10.12 CX3_PHY_TIME_DELAY (Register Address: 0x0060)

This register configures delay parameters for the MIPI CSI-2 Receiver PHY on the CX3 MIPI CSI-2 block. The settings depend on the CSI_RX_CLK defined in 1.7.3

This register is set by the *CyU3PMipicsiSetIntfParams()* API and queried using the *CyU3PMipicsiQueryIntfParams()* API.

BIT	15	14	13	12	11	10	9	8
NAME	TC TERM	RESERVED						
BIT	7	6	5	4	3	2	1	0
NAME	TD TERM	THS SETTLE						

Register Field	Bit	Description
TC TERM	[15]	TC TERM Selection: Set to 1 for normal operation, 0 is not supported.
RESERVED	[14:8]	RESERVED. Firmware must preserve their settings by reading them, changing non-reserved bits, and re-writing them.
TD TERM	[7]	TD TERM selection: 0: Data Lane HS termination occurs after 2 x CSI_Rx_Clk or 3 x CSI_Rx_Clk time when LP to HS transition. 1: Data Lane HS termination is set immediately when LP to HS transition (preferred).
THS SETTLE	[6:0]	THS SETTLE Timer: Timer to control delay between LP to HS transition. Range: 0x00 - 0x7F. Delay = (THS SETTLE + 1) x CSI_RX_CLK

1.11 CX3 MIPI CSI-2 APIs

This section details the APIs exposed by the EZ USB FX3 SDK to configure and utilize the MIPI CSI-2 block on the CX3. Additional details on the APIs and the enumerations and structure types used by them can be found in the EZ-USB FX3/FX3S/CX3 SDK Firmware API Guide available as part of the EZ USB FX3 SDK.

The MIPI CSI-2 block configuration APIs are available through the cyu3mipicsi.a library file and the function declarations, enumerations and structure types are available through the header file *cyu3mipicsi.h*.

EZ USB FX3 SDK 1.3 provided a BETA level support for the CX3 part. Full support for the CX3 part is available as part of the EZ-USB FX3 SDK starting from the EZ-USB FX3 SDK 1.3.1 release.

1.11.1 CyU3PMipicsiInit()

CyU3PReturnStatus_t CyU3PMipicsiInit (void)

This function initializes MIPI CSI-2 block on the CX3 device and is expected to be called prior to any other calls to the MIPI CSI-2 block. The CX3 GPIO Block, the PIB block and the I^2C blocks should have been initialized prior to calling this function.

This call leaves the MIPI CSI-2 block in a low-power mode and *CyU3PMipicsiWakeup()* should be called to enable the clocks on the MIPI CSI-2 block.

This function internally power cycles the MIPI CSI-2 block.

It also enables the output of XRESET and XSHUTDOWN signals by setting **XSHUTDOWN ENABLE** and **XRESET ENABLE** bits of the CX3_CSI_SENSOR_SIG_EN register to 0.

This function leaves the CX3_CSI_SENSOR_SIG_VAL register in its default state (0x0000) thereby setting the MIPI XRESET and XSHUTDOWN signals to Drive LOW. If either of these signals, needs to be in Drive HIGH state for sensor operation, it needs to be explicitly set to Drive HIGH state using *CyU3PMipicsiSetSensorControl()* API after calling this function.

On successful execution this function returns CY_U3P_SUCCESS value to the calling function.

1.11.2 CyU3PMipicsiDeInit()

CyU3PReturnStatus_t CyU3PMipicsiDeInit (void)

This function de-initializes MIPI-CSI interface block on the CX3 device and is expected to be called prior to calling *CyU3PSysEnterStandbyMode()*. This function should be called before the I^2C block or GPIO blocks are de-initialized.

The MIPI XRESET and XSHUTDOWN signals shall not be driven by the CX3 after this call.

On successful execution this function returns CY_U3P_SUCCESS value to the calling function.

1.11.3 CyU3PMipicsiReset()

CyU3PReturnStatus_t CyU3PMipicsiReset (CyU3PMipicsiReset_t resetType)

This function is used to reset the MIPI-CSI block on the CX3. The MIPI-CSI block provides two reset modes – **Hard reset**, which power-cycles the entire block, and **Soft reset**, which does not reset the I^2C communication channel used by the block. The reset mode is selected via the parameter of type *CyU3PMipicsiReset_t* passed to this function.

Soft Reset cannot be called on the interface until the block has been initialized. A Hard reset however, can be called on the interface at any point in time.

The *CyU3PMipicsiInit()* call internally performs a Hard reset on the interface before initializing it.

Hard reset enables the output of XRESET and XSHUTDOWN signals by setting **XSHUTDOWN ENABLE** and **XRESET ENABLE** bits of the CX3_CSI_SENSOR_SIG_EN register to 0. This function leaves the

CX3_CSI_SENSOR_SIG_VAL register in its default state (0x0000) thereby setting the MIPI XRESET and XSHUTDOWN signals as Drive 0. If either of these signals, needs to be in Drive 1 state for sensor operation, it needs to be explicitly set to Drive 1 state using *CyU3PMipicsiSetSensorControl()* after calling *CyU3PMipicsiReset(CY_U3P_CSI_HARD_RST)*.

A Soft reset does not change the state of the XRESET and XSHUTDOWN signals. A soft reset sets the **RESET** bit of the CX3_SYSTEM_CTRL register to 1, waits for 5 timer ticks on the CX3 and then clears the **RESET** bit to 0 to resume normal operation.

On successful execution this function returns CY_U3P_SUCCESS value to the calling function.

1.11.4 CyU3PCx3DeviceReset()

void CyU3PCx3DeviceReset (CyBool_t isWarmReset,

CyBool_t sensorResetHigh)

This function is similar to the *CyU3PDeviceReset ()* API used for FX3/FX3S devices. It uses the *isWarmReset* parameter to determine if the device is going in for a warm reset or a cold reset.

In addition to resetting the CX3 device, this function also drives the MIPI XRESET signal to reset the Image Sensor before the CX3 device is reset, by internally calling CyU3PMipicsiSetSensorControl() and setting the *CY_U3P_CSI_IO_XRES* signal to high or low based on the value provided in *sensorResetHigh*.

This function does not return.

1.11.5 CyU3PMipicsiSleep()

CyU3PReturnStatus_t CyU3PMipicsiSleep (void)

This function is used to disable the PLL clocks on the MIPI CSI-2 block and place it in low-power sleep. No data transfers from the Image Sensor to the CX3 will occur while the block is in low power sleep mode.

This API sets the **SLEEP** bit of the CX3_SYSTEM_CTRL register of the MIPI CSI-2 block to 1.

On successfully placing the block to sleep this function returns CY_U3P_SUCCESS.

1.11.6 CyU3PMipicsiWakeup()

CyU3PReturnStatus_t CyU3PMipicsiWakeup (void)

This function is used enable the clocks on the MIPI CSI-2 block to take it from Low power sleep to Active.

This API clears the **SLEEP** bit of the CX3_SYSTEM_CTRL register of the MIPI CSI-2 block to 0.

On successful execution this function returns CY_U3P_SUCCESS.

1.11.7 CyU3PMipicsiSetSensorControl()

CyU3PReturnStatus_t CyU3PMipicsiSetSensorControl (CyU3PMipicsiSensorIo_t io,

CyBool_t value)

This function is used to drive the XRES and XSHUTDOWN signals from the CX3 to Image sensor. The function allows for the signals to be driven high or low.

The function can drive either one of the two signals or both signals (both driven to the same value) simultaneously. To set both signals to the same value use the mask *(CY_U3P_CSI_IO_XRES | CY_U3P_CSI_IO_XSHUTDOWN)* as value for *io*. To set the two signals to separate values multiple calls to this function are required (once for each signal).

This API sets the appropriate bits of the CX3_CSI_SENSOR_SIG_VAL to 1 or 0 to drive the corresponding MIPI signal to the sensor high or low.

To drive the XRESET signal high or low, the **XRESET OUTPUT** bit is set to 1 or cleared to 0. Similarly, to drive the XSHUTDOWN signal high or low the **XSHUTDOWN OUTPUT** bit is set to 1 or cleared to 0.

On successful execution this function returns CY_U3P_SUCCESS.

1.11.8 CyU3PMipicsiCheckBlockActive()

CyBool_t CyU3PMipicsiCheckBlockActive (void)

This function is used to check if the MIPI CSI-2 block is Active or in low power sleep.

It returns a CyTrue if the block is active and CyFalse if it is not active.

This function only checks for a flag which is set when the *CyU3PMipicsiWakeup()* is called and does not check the actual interface registers. In case you are modifying the configuration registers directly, please check the actual value of the SLEEP bit of the CX3_SYSTEM_CTRL register rather than using this API.

1.11.9 CyU3PMipicsiSetIntfParams()

CyU3PReturnStatus_t CyU3PMipicsiSetIntfParams (CyU3PMipicsiCfg_t * csiCfg,

CyBool_t wakeOnConfigure)

This function is used to configure the MIPI CSI-2 block parameters over the I^2C interface.

The function takes in an object of the type *CyU3PMipicsiCfg_t* and configures the MIPI CSI-2 block. The function powers off the interface clocks by calling *CyU3PMipicsiSleep()* before making any changes to the interface configuration registers.

This function sets the following MIPI CSI-2 block registers with values from the *CyU3PMipicsiCfg_t* structure passed to it as shown in Table 6.

Table 6: Parameters Used for Configuring MIPI CSI-2 Block Registers

CX3 MIPI CSI-2 Block Register	CyU3PMipicsiCfg_t Parameter(s) Used for Configuration
CX3_PLL_CTRL0	*pllPrd* and *pllFbd* parameters
CX3_PLL_CTRL1	*pllFrs* parameter
CX3_CLK_CTRL	*csiRxClkDiv*, *parClkDiv* and *mClkRefDiv* parameters
CX3_MCLK_CTRL	*mClkCtl* parameter,
CX3_BYTE_COUNT	Computed from *hResolution* and *dataFormat* parameters
CX3_PHY_TIME_DELAY	*thsSettle* parameter available starting with FX3 SDK 1.3.3
CX3_DATA_FMT	*dataFormat* parameter
CX3_CONFIG_CTRL	*dataFormat* and *numDataLanes* parameters

Parameter *wakeOnConfigure* is used to turn the clocks ON immediately after the configuration has been completed or to leave the clocks powered down. If the clocks are left powered down, *CyU3PMipicsiWakeup()* should be called to start the clocks.

On successful execution this function returns CY_U3P_SUCCESS.

A CX3 configuration generation tool has been provided as part of the EZ-USB Suite (Eclipse based IDE) provided with the EZ-USB SDK. The tool can be used to generate the *CyU3PMipicsiCfg_t* structure element used to configure the MIPI CSI-2 block based on image sensor inputs and stream parameters.

Detailed usage instructions for the configuration tool are available as part of the EZ-USB Suite help menus and as part of the Cypress EZ-USB FX3 Quick Start Guide (Getting Started with FX3 SDK.pdf) available in the doc folder of the EZ-USB FX3 SDK installation path.

1.11.10 CyU3PMipicsiQueryIntfParams()

CyU3PReturnStatus_t CyU3PMipicsiQueryIntfParams (CyU3PMipicsiCfg_t * csiCfg)

This function is used to read back the MIPI-CSI interface parameters from the block. The parameters read back are provided to the calling function via the pointer of type *CyU3PMipicsiCfg_t* passed in from the calling function. The function reads the registers which are written to by *CyU3PMipicsiSetIntfParams()* the *CyU3PMipicsiCfg_t* structure object pointed to by *csiCfg* should be initialized prior to being passed to this API.

1.11.11 CyU3PMipicsiGetErrors()

CyU3PReturnStatus_t CyU3PMipicsiGetErrors (CyBool_t clrErrCnts,

CyU3PMipicsiErrorCounts_t * errorCounts)

This function is used to get a count of CSI-2 protocol and physical layer errors from the MIPI CSI-2 block.

The function takes a parameter which determines whether or not the error counts on the interface are cleared.

The error counts for each error type are retrieved via a pointer of type *CyU3PMipicsiErrorCounts_t* passed to this function. The error count values for each type can reach a maximum count of 0xFF. The count values will continue to report the existing error value on each call unless the function explicitly clears the counts using clrErrCnts.

The *errorCounts* object should be initialized prior to being passed to this function.

1.11.12 CyU3PMipicsiGpifLoad()

CyU3PReturnStatus_t CyU3PMipicsiGpifLoad (CyU3PMipicsiBusWidth_t busWidth,

uint32 t bufferSize)

As described in Section 1.9 , the CX3 has a fixed function GPIF interface designed for Image sensor data acquisition from the MIPI CSI-2 block.

This function allows selection of the GPIF data bus-width and configuration of the size of the DMA buffer provided for GPIF transfers. The PIB block should have been initialized prior to calling this function.

The DMA buffer size needs to be a multiple of the bus-width. For example, if the GPIF bus width has been set up to 24 bit width, then the DMA buffer needs to be a multiple of 24 bits (i.e. 3 bytes).

Additionally, the DMA buffer size needs to be a multiple of 16 bytes to satisfy the requirements of the CyU3PDmaChannelCreate API.

The bus width and the DMA buffer size being passed to this API should be based on the width of the data format selected for data transfer.

For example, an RGB 888 data type needs the bus-width to be set to 24 bits, whereas YUV 422 requires that the bus width be set to 16 bits and RAW 8 format needs the bus width to be set to 8 bits.

If the bus-width configured using this API is smaller than that required for the data format selected, the upper data bits being transferred from the MIPI CSI-2 block will be lost. If the bus-width configured by this API is larger than what is required for the data format selected, the unused upper bits will be set to 0 in the data received.

1.12 Additional References

Additional information on the CX3 including Datasheets and Application notes can be found at http://www.cypress.com/cx3/.

Additional information on the FX3 including Datasheets, Application notes and the EZ-USB FX3 Technical Reference Manual (TRM) can be found at http://www.cypress.com/fx3/.

The EZ-USB FX3 SDK software download and documentation provided as part of the SDK (Programmers Manual, Firmware API Guide, and Quick Start Guide etc.) can be found at http://www.cypress.com/?rID=57990

Details on the MIPI CSI-2 Specification can be found on the MIPI alliance website at http://www.mipi.org/specifications/camera-interface

1.13 Glossary

API	Application Programming Interface. A series of software routines that comprise an interface between an application and lower-level services and functions (for example interfaces to configure and control the FX3/CX3 device operation). The EZ-USB FX3 SDK provides a set of libraries which provide APIs to configure and control the operation of the FX3 device.
Bayer Filter	A color filter array for arranging RGB color filters on a square grid of photo sensors. The filter pattern is 50% green, 25% red and 25% blue.
CPU subsystem	The EZ-USB FX3/CX3 devices have an embedded 32-bit ARM926EJ-S core which delivers processing capability of 200 MIPS. This ARM core is coupled with Instruction and Data Caches, Tightly Coupled Memories (TCM), and a PL192 Vectored Interrupt Controller (VIC). More details can be found in the FX3 CPU Subsystem chapter of the *EZ-USB FX3 TRM*.
Data Lanes	Unidirectional MIPI CSI-2 differential serial interface used for data transfer. Each lane comprises two signals: Data+ and Data-. CX3 supports up to four Data lanes, each capable of speeds up to 1 Gigabits per second.
DMA	The Direct Memory Access (DMA) subsystem performs high-bandwidth data transfers between memories and peripherals without CPU intervention. The FX3/CX3 architecture includes a DMA fabric that is used to steer data between various peripheral interfaces and system memory. Details regarding the DMA subsystem can be found in the FX3 DMA Subsystem chapter of the *EZ-USB FX3 TRM*.
Global Controller	FX3/CX3's Global Controller (GCTL) unit contains a Clock Generation Block, Multifunction I/Os, a Power Control Block, and a Reset Block. The Clock Generation block provides clocks to all device peripherals. More details can be found in the FX3 Global Controller (GCTL) chapter of the *EZ-USB FX3 TRM*.
GPIF II	The CX3 GPIF II interface is a fixed-function implementation of the General Programmable Interface available in the FX3. The CX3 GPIF II interface is used to transfer data from the MIPI CSI-2 receiver block. Details of the fixed-function GPIF II state machine on the CX3 are available in Section 1.9 of this TRM supplement. More details can be found in the General Programmable Interface II (GPIF II) chapter of the *EZ-USB FX3 TRM*.
I^2C	The I^2C-bus protocol, created by Philips Semiconductor, stands for 'Inter-Integrated Circuit bus' and allows data communication between I^2C capable devices using two wires. It sends information serially using a data (SDA) line and a clock (SCL) signal. FX3/CX3 can function as a master I^2C controller supporting 100 kHz, 400 kHz, and 1 MHz operation. More details can be found in the I^2C Interface section of the Low Bandwidth Peripherals chapter of the *EZ-USB FX3 TRM*.
JTAG	Joint Test Action Group (JTAG) is the common name for the IEEE 1149.1 Standard Test Access Port and Boundary-Scan Architecture. FX3/CX3's JTAG interface has a standard 5-pin interface to connect to a JTAG debugger to debug firmware using the on-chip debug circuitry. Industry-standard debugging tools for the ARM926EJ-S core can be used for debugging the FX3/CX3 devices.
Low-Bandwidth Peripherals	FX3/CX3's serial peripherals I^2C, SPI, I^2S and UART including GPIO form the Low-Bandwidth Peripherals (LBP). More details can be found in the Low-Bandwidth Peripherals chapter of the *EZ-USB FX3 TRM*.
MCLK	The MCLK is an optional clock output from the CX3, which can be used as the input reference clock for the image sensor. More details are available in Section 1.7.5 of this TRM supplement.
Memory subsystem	The CX3 memory subsystem comprises system RAM that serves as program and data memory, SRAM controller, and an Advanced High-performance Bus (AHB)-based interconnect that allows the ARM CPU and the hardware blocks to access these memories. The Memory Mapped I/O (MMIO) interconnect provides access to registers in various peripheral blocks. More details can be found in the Memory and System Interconnect chapter of the *EZ-USB FX3 TRM*.
MIPI CSI-2	Mobile Industry Processor Interface (MIPI) Alliance Standard for Camera Serial Interface 2 (CSI-2) is a serial camera interface defined by the MIPI Alliance. It defines standard data transmission and control interfaces between the camera transmitter and receiver. The data transmission interface is a unidirectional differential serial interface with data and clock signals. More details on the MIPI CSI-2 specification can be obtained from the MIPI Alliance website http://www.mipi.org/specifications/camera-interface

MJPEG Format	Motion-JPEG or MJPEG is a video format in which each video frame is separately compressed using the JPEG still-image compression algorithm. A sequence of such frames represents the source video.
Pixel	A pixel is the smallest controllable element of a picture represented on the screen. Each pixel is a sample of an original image; more samples typically provide more accurate representations of the original. Typically, the number of pixels defines the resolution of the image sensor.
RAW Format	RAW format data provides direct, unprocessed output from an image sensor. The CX3 MIPI CSI-2 receiver interface supports RAW8, RAW10, RAW12 and RAW14 formats, where the numeric value signifies the number of bits of data per image sensor pixel. A color filter array (such as the Bayer Filter) is typically used to obtain RGB color information from the RAW format output.
RGB Format	A color format which defines a color space in terms of the Red, Green, and Blue (R, G, B) components. The CX3 MIPI CSI-2 receiver interface supports RGB888, RGB666 and RGB565 streams, where the numeric value signifies the number of bits per color component in the stream (for example, RGB 565 has five bits of Red data, six bits of Green data and five bits of Blue data).
Reserved Bits	RESERVED Register Bits are not meant to be changed by the user. When writing to a register with RESERVED bits, you should ensure that the values are not changed during a Write (preferably by doing a Read before the Write and maintaining the values for the RESERVED bits during the Write).
SDK (Software Development Kit)	The EZ-USB FX3 SDK (Software Development Kit) is a full-featured firmware solution that allows customers to leverage the features of the FX3/FX3S/CX3 devices and build a custom design in a short time. The SDK provides a firmware framework which is made up of a set of drivers and convenience APIs for the various hardware blocks on these devices. The drivers abstract the complexity of the FX3 device architecture and provide easy-to-use APIs for customers to configure and control the FX3 device operation.
Socket	A DMA socket is an FX3 device construct that maps to one end of a data path into or out of the device. More details can be found in the FX3 DMA Subsystem chapter of the *EZ-USB FX3 TRM*.
USB	Details regarding the USB subsystem on the FX3/CX3 are available in the USB chapter in the EZ-USB FX3 TRM. The CX3 device can only operate as a USB peripheral device and does not support the USB Host, OTG, and Charger features supported by FX3.
YUV Format	A color format that defines the color space in terms of luminance (Y) and chrominance (UV) components. The CX3 MIPI CSI-2 receiver interface supports YUV422 8- and 10-bit color formats.
XRESET	The CX3 output signal that can be used to reset the image sensor.
XSHUTDOWN	The CX3 output signal that can be used to control image sensor power modes.

Revision History

Document Revision History

Document Title: EZ-USB® CX3 Technical Reference Manual Document Number: 001-91492			
Revision	**ECN No.**	**Origin of Change**	**Description of Change**
**	4307893	VOM	New Specification
*A	5975621	SAVJ	Updated Introduction section
*B	6181664	SAVJ	Updated Figure 7, CX3 GPIF II state machine Updated template

PSoC is a registered trademark of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2014-2018. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC (“Cypress”). This document, including any software or firmware included or referenced in this document (“Software”), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress’s patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.
TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress does not assume any liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. In addition, the products described in these materials may contain design defects or errors known as errata which may cause the product to deviate from published specifications. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage (“Unintended Uses”). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.
Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.