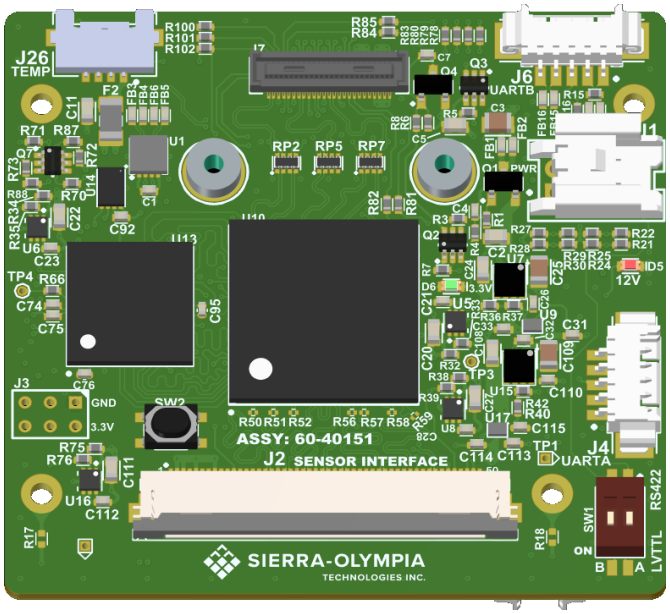
SIERRA-OLYMPIA
TECHNOLOGIES INC.



SOTI MIPI-USB ADAPTER

10-80217
REVISION A

603 Portway Avenue, Suite 100 | Hood River, OR 97031 | 855.222.1801 | sierraolympia.com

Table of Contents

Reference Documents

Document No.	Description
20-70106	Electrical ICD, Ventus Micro 225, MUAD
20-70108	MIPI-USB Adapter Mezzanine Header Electrical ICD
DTER2VE3004	SCD Sparrow Communication Protocol Commands

Revision History

Version	ECO	Description	Date
1 (draft)	-	Draft Release. Based on doc# 10-80023. I2C register mapping changed significantly to support multiple UART ports / pass-throughs as well as 32-bit pixbus.	10/21/2025
2 (draft)	-	Table 3-1 was missing i2c reg 0x00, Revision. Consequently, all listed register addresses were offset by 1 count. Descriptions starting page 6 were correct in Rev1, only table 3-1 was in error.	11/10/2025
A	1940	As of release date, USB FW was not implemented. This document is limited to describing MIPI functionality. Removed remaining references to unsupported sensor.	02/20/2026

List of Figures

List of Tables

1 BACKGROUND

Sierra-Olympia Technologies, Inc. Ventus cameras using the MIPI-USB board utilizes a basic i2c register set for controlling the MIPI and communications core. In addition to control registers, the firmware interface provides an i2c ↔ uart bridge which allows read/write operations to the camera core or alternately providing a direct LVCMOS UART pass-through when needing to program camera firmware directly. At present the USB (FX3) portion of the device is under development, so while the block diagram shows its existence, the RevA document release is wholly focused on the MIPI aspects. In time additional details will be added for the USB/UVC functionality.

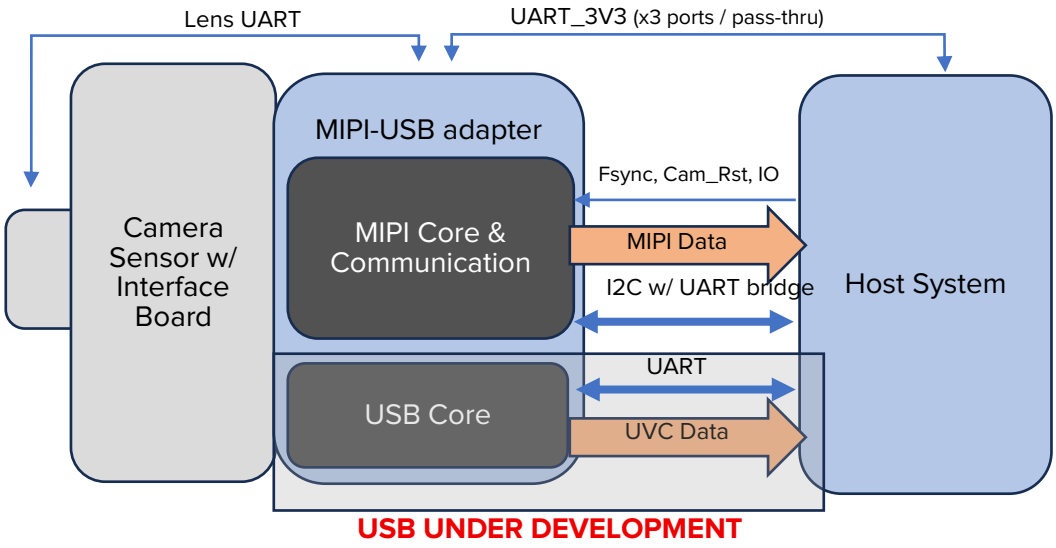


Figure 1-1 Adapter Block Diagram

2 COMMUNICATION

2.1 I2C Protocol

The primary communication connection to the MIPI backend electrical interface complies with standard i2c signaling. Low read addresses (0x00-0x03) are read-only and do not auto-increment the embedded register address with repeated read events similar to the write-only address 0x04. Register addresses 0x05 and higher addresses are read/write and support auto address incrementing. All configuration bytes may be written during initialization with a single, multi-byte write operation starting with address 0x05. Repeated start bits are supported. Writing individual bytes will also work but requires more i2c bus traffic for initialization. All register settings are volatile and must be initialized on every power up (subject to change in future revisions). System defaults to 640x512, 2-lanes, mipi_clk = 2x camera pclk, pass-through UART (921600 baud), cam_pwr/mipi_en active, mipi_DT=0x2E (also the power up default even if connected to a 1280x1024 resolution sensor).

Table 2-1 I2C Slave Parameters

SCL clk rate	7-bit Slave address
100-400 kHz (1MHz possible but not tested)	0x33

Figure 2-1 I2C Addressing

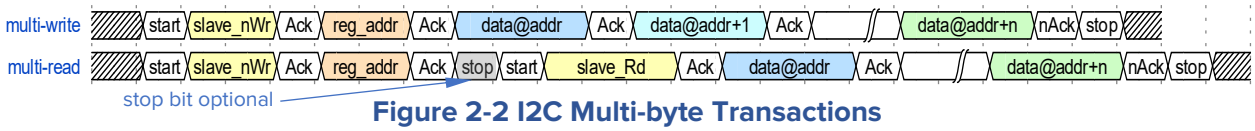


Figure 2-2 I2C Multi-byte Transactions

2.2 I2C↔UART Bridge

Direct communication with the attached sensor and/or zoom lens is possible with an i2c↔UART bridge but comms are multiplexed to one device at a time. Alternately the system may simultaneously use the i2c bridge for one device and use a pass-through UART for the other device. Bridge communication is performed using dedicated i2c read (0x03) and i2c write addresses (0x04). Since camera specific commands typically have a binary packet structure with checksum, it is recommended to send mulit-byte write operations instead of individual i2c byte reads/writes. Register *0x02, Rd_Bytes* indicates the number of bytes in the UART_Rd fifo (0x03). Performing a single multibyte read of Rd_Bytes from register 0x03 may then be used to fetch only valid reply data or by reading one byte at a time. Reading the camera replies is optional and may be disabled by asserting the ***uart_rd_clr*** control bit.

Figure 2-3 i2c↔UART bridge read/write operations

Table 2-2 lists example Linux terminal command formatting (useful for system debug). Examples also illustrate command sequences coded within an application. See Sparrow SW ICD(s) for more detail.

Table 2-2 Example Sensor Commands

Cmd Description	Bash strings
Close shutter	*i2ctransfer -f -y 9 w3@0x52 0x17 0x01 0x00*
Execute 1pt cal [1]	*i2ctransfer -f -y 3 w7@0x33 0x04 0xAA 0x02 0x84 0x00 0x00 0xD0*
Enable Fsync pass-thru [1]	*i2ctransfer -f -y 9 w2@0x33 0x06 0x4F (power all peripherals, enable fsync pass-thru)*
Enable Camera Fsync_In [1]	*i2ctransfer -f -y 3 w7@0x33 0x04 (TBD, waiting on supplier feedback)*

2.3 External UART Connections

Two of the three user accessible UART pass-through ports can be configured for simultaneous sensor or lens communications. Ports are selectively muxed so one is connected to the sensor and another to the lens port. UART Rx (input) pins have weak pull-ups so external pull-ups are not required if one or both connections are unused / floating even if active based on MUX selection. See electrical ICD for details. The system is not required to use the pass-through serial ports. When user configures i2c-bridge communication with the sensor/lens, the unconnected device will be connected to the external pass-through port of the user's choice.

[1] Note: i2c-UART bridge must be enabled for these commands.

3 REGISTER ADDRESS DESCRIPTION

3.1 Control Register Summary

Table 3-1 I2C Register Address Summary

Reg(hex)	Name	type	7	6	5	4	3	2	1	0	default
0x00	Revsion	rd {wr}	Revision[7:0]								Note 1
0x01	Status_Reg	rd	shutter_evt	meas_stat	mipi_u_flow	mipi_o_flow	wr_fifo_full	wr_done	rd_fifo_full	rd_rdy	
0x02	Rd_Bytes	rd	UART bytes available in Rx read fifo if *Rd_Bytes* < 256. Reports as 255 when: 256 <= *bytes_avail* <= 1024								Note 2
0x03	UART_Rd	rd	UART Rx readback register. i2c reg-address not auto incremented (supports multi-byte reads)								Note 3
0x04	UART_Wr	wr	UART Tx write register. i2c reg-address not auto incremented (supports multi-byte writes)								0x00
0x05	Mode_Reg	rd/wr	mipi_num_lanes		mipi_clk_sel	pixel_mode	frm_cnt_en	mode_sel[2:0]			0xB6
0x06	Ctrl_Reg	rd/wr	fsync_sel		fsync_dir	fsync_en	mipi_en	lens_en	cool_pwr_en	snsr_en	0x0F
0x07	xres_low	rd/wr	horizontal resolution, low byte								0x80
0x08	xres_high	rd/wr	horizontal resolution, high byte								0x02
0x09	yres_low	rd/wr	vertical resolution, low byte								0x00
0x0A	yres_high	rd/wr	vertical resolution, high byte								0x02
0x0B	pixel_format	rd/wr	test_pat_en	test_pat_ctrl	mipi data type (DT[5:0])						0x2E
0x0C	Ctrl_Reg2	rd/wr	uart_mux_sel					shutter_nrst	cool_nStby	fx3_data_en	0x16
0x0D	Ctrl_Reg3	rd/wr	cont_clk_en	baudL_sel			uart_rd_clr	baudS_sel			0x12
0x0E	Rsvd/user	rd/wr	user_byte (resets to 0 on power cycle or hard reset)								0x00

Once per power up, reset, or pwr_en 0→1 transition, video input resolution is measured and set in these registers. If xres > 2048 or yres < 240, defaults are set. mipi_clk_sel set to 0 if yres > 576, 1 otherwise.

7-bit slave address: 0x33

nominal scl frequency: 400 kHz (no minimum frequency, maximum frequency not tested).

Note 1: Revision register is read repeatedly for version# and system info string. Write any value to reset string pointer to major revision (0th byte). It is recommended to read three bytes, then read one byte at a time while checking for null string termination to determine if the entire string has been read. See section 3.2.0 for more information.

Note 2: UART bridge read FIFO is 1024 bytes deep. If FIFO is not read between UART_Wr commands overflow bytes will be ignored. FIFO will remain empty if the *uart_rd_clr* bit is asserted high. See sections 3.2.2 - 3.2.3 for more information.

Note 3: UART bridge write FIFO is 1024 bytes deep. With default baud_sel = 0 (921600), UART module is faster than the i2c bus transactions so the FIFO will never get full. When baud_sel = 1 (57600), it would be possible to overflow the write FIFO but it is not likely considering a typical camera core command is 8 bytes or less. See section 3.3.1 for more information.

3.2 Read Only Registers

3.2.0 0x00, Revision (rd/rst)

Returns the version of the Sierra-Olympia firmware MIPI FW. Repeated read operations return successive bytes of the firmware version (binary) and information string (ASCII characters). The first three reads return the binary version bytes, {Major Rev}, {Minor Rev}, {Build Num}, some of which may report as 0x00 and are not intended to be a null string terminator. Subsequent read operations return FW system information string (ASCII characters) and a null character (0x00) terminator. Continuing to read will start over with the {Major Rev} byte. Write any value to reg address 0x00 to reset the version readback pointer to {Major Rev} regardless of where the internal string counter was pointing. Version string is typically around 32 bytes but will be less than 64 bytes in length.

Bits							
7	6	5	4	3	2	1	0
Revision[7:0], revision readout register. Address does not auto increment. Supports multi-byte reads.							

Default Value	n/a

Example return string:

0x01 0x03 0x05 0x48 0x45 0x58 0x41 0x5f 0x4d 0x49 0x50 0x49 0x20 0x32 0x30 0x32 0x34 0x2d 0x30 0x39 0x2d 0x30 0x33 0x00

Version: *1.3.5*

Sys Info: *HEXA_MIPI 2024-09-03*

3.2.1 0x01, Status (rd)

Reports status information of MIPI backend firmware. Register writes are ignored. Reading this address does not auto-increment internal register address.

Bits							
7	6	5	4	3	2	1	0
shutter_evt	tbd	mipi_u_flow	mipi_o_flow	wr_fifo_full	wr_done	rd_fifo_full	rd_rdy

Default Value	n/a

Bit Definitions

Bit #	Field Name	Values	Description
0	rd_rdy		data available in UART read FIFO
		0	read FIFO empty
		1	read FIFO non-empty
1	rd_fifo_full		UART read FIFO status
		0	read FIFO not full
		1	read FIFO full, subsequent readback bytes will be ignored.
2	wr_done		reports that all bytes written to addr 0x04 have been sent to cam. UART
		0	UART write FIFO not empty
		1	UART write FIFO empty

Bit #	Field Name	Values	Description
3	wr_fifo_full		UART write_fifo status
		0	UART write FIFO not full
		1	UART write FIFO full
4	mipi_o_flow		MIPI Tx core, data FIFO overflow. Indicates selected mipi_clk is too slow
		0	status OK (expected value by design)
		1	overflow occurred
5	mipi_u_flow		MIPI Tx core, data FIFO underflow. Indicates selected mipi_clk is too fast
		0	status OK (expected value by design)
		1	overflow occurred
6	-		
7	shutter_evt		indicates shutter event in progress. optionally poll after sending 1pt cmd
		0	shutter idle
		1	shutter active due to manually or automatically initiated 1pt cal event

3.2.2 0x02, Rd_Bytes (rd)

Reports camera UART response bytes following camera UART commands. Read FIFO is 1024 bytes and this register will report a value of 255 if the number of bytes in FIFO is >254. Recommend UART_Rd register readback using exactly the reported value of bytes to minimize i2c traffic. Value will remain zero if uart_rd_clr bit is asserted in register 0x06, ctrl_Reg.

Bits							
7	6	5	4	3	2	1	0
UART bytes available in Rx read fifo if Rd_Bytes < 256. Reports as 255 when: 256 <= bytes_avail <= 1024							

Default Value	n/a

3.2.3 0x03, UART_Rd (rd)

Dedicated byte wide UART read FIFO register address. Repeated reads at this address will readout reply bytes received from the camera core. Each read will decrement the Rd_Bytes value by 1 if non-zero. Reported value is not defined when Rd_Bytes register reports zero.

Bits							
7	6	5	4	3	2	1	0
UART Rx read byte register. i2c reg-address not auto incremented (supports multi-byte reads @ 0x03)							

Default Value	n/a

3.3 UART Bridge Write Register

3.3.1 0x04, UART_Wr (wr)

Dedicated byte wide UART write FIFO register address. All bytes written to this address are repeated to the camera core's internal UART port. FIFO depth is 1024 but due to the relative clock rates, it is unlikely there will be more than a few bytes stored even with the lower baud rate selection.

Bits							
7	6	5	4	3	2	1	0
UART Tx write byte register. i2c reg-address not auto incremented (supports multi-byte writes @ 0x04)							

Default Value	n/a

3.4 Read/Write Registers

Writing to any rd/wr register skips one video frame to reset and resynchronize the MIPI core at the beginning of the next available frame following settings changes.

3.4.1 0x05, mode_Reg (rd/wr)

Bit mode control register. Controls bit shuffling of the input pixel data bus, optional frame counter enable, MIPI bit depth per-pixel, MIPI clock multiplier select, and number of MIPI lanes. Always powers up with *default value* and user initialization is required when using different settings. "pixel_mode" applies to both Y16 and YUV camera data (see definition below).

Bits							
7	6	5	4	3	2	1	0
mipi_num_lanes		mipi_clk_sel	pixel_mode	frame_cnt_en	mode_sel		

Default register Value (POR)	0xb6

Table 3-2 Possible Values for Reg 0x05 [4]

	8-bit	16-bit	16-bit, byte-swap
1 lane	0x41	0x54	0x56
2 lanes	~~0x81~~	0x94	0x96
2 lanes	0xA1	0xB4	0xB6
4 lanes	~~0xC1~~	0xF4	0xF6

Note 4: Achievable MIPI clock rates are dependent on sensor's video timing. Each FW build will have two selectable mipi_clk ratios / bandwidths tailored to the sensor and selectable in any lane configuration. MIPI bandwidth must be slightly higher than video feed to prevent buffer overflow but not so large that it causes buffer underflow. For example a 640x512 resolution sensor with pclk < 40MHz only supports 1 or 2-lane operation at the lower bandwidth setting. 640x512 sensors with pclk > 40MHz and large vertical blanking work in either low or high bandwidth settings.

Bit Definitions

Bit(s)	Field Name	Values	**Description** (mode_sel bit mapping uses compact Verilog notation)
2:0	mode_sel		Pixel Bus bit ordering mode select
		0	14-bit pass-through, Output[15:0] = { 2'h0, Input[13:0] } (reserved)
		1	8-bit mode. Output={ 8'h0, Input[13:6] }
		2	8-bit to 16-bit byte packing from LH Superframe (reserved)
		3	8-bit to 16-bit byte packing from RH Superframe (reserved)
		4	16-bit pass-through. Output [15:0] = Input[15:0]
		5	Debug. Output[15:0] = { Input[13:6], 8'h0 } . Subject to change.
		6	16-bit byte-swap. Output[15:0] = { Input[7:0], Input[15:8] }
		7	16-bit pass-through from RH Superframe (reserve)
3	frame_cnt_en		Replace pixels (0,0), (0,1) with 32-bit frame counter which increments once per frame since power-up or hard reset. Not valid in 8-bit mode. { Pix(0,0), Pix(0,1) } = { 16'frame_cnt[31:16], 16'frame_cnt[15:0] }
		0	disabled. Input data passed through to output on pixels (0,0),(0,1).
		1	enabled. Output data for pixels (0,0),(0,1) replaced with frame count.
4	pixel_mode		MIPI byte packing mode select (affects byte count per MIPI row).
		0	8-bit per-pixel MIPI stream. Sensor in 8-bit mode only (half the MIPI bandwidth required vs. 16-bit mode, e.g. 1280x1024 in 1 MIPI lane).
		1	16-bit per-pixel MIPI stream. Sensor in YUV or Y16 mode.
5	mipi_clk_sel		MIPI clock multiplier vs. input pclk. (Note 5)
		0	mipi_clk = 4x pclk (160 / 320MHz, physical clk, not devicetree setting).
		1	mipi_clk = 2x pclk (80 / 160MHz, physical clk, not devicetree setting).
7:6	mipi_num_ lanes		Number of MIPI lanes used for output data
		0	Reserved (decodes to 2-lanes)
		1	1-lane
		2	2-lanes
		3	4-lanes

Note 5: MIPI_clk output frequency depends on sensor frame rate setting. Pixel clock / Camera Link clock is 40MHz @ 30Hz frame rate and 80MHz for 60Hz. mipi_clk_sel sets the internal PLL multiplier to 4x or 2x pixel clock for running the MIPI serializer module which also scales with pixel clock at input.

3.4.2 0x06, ctrl_Reg (rd/wr)

Controls sensor power, MIPI core enable, fsync, UART options. Metadata enable was not implemented at the time of writing, but the development option is reserved should there is sufficient customer demand.

Bits							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
fsync_sel		fsync_dir	fsync_en	mipi_en	lens_en	cool_en	snsr_en

Default register Value (POR)	0xCF

Bit Definitions

Bit(s)	Field Name	Values	Description
0	snsr_en	-	enable/disable camera electronics power
		0	disabled
		1	enabled
1	cool_en	-	enable/disable cooler power
		0	disabled
		1	enabled
2	lens_en	-	enable MIPI-board fsync pass-through. does not set "mode" of camera core.
		0	disabled. FPGA 3.3V in/out pin and camera 1.8V in/out pin set highZ.
		1	enable pass through pins. set direction matching camera core setting.
3	mipi_en	-	enable/disable MIPI Tx output.
		0	disabled
		1	enabled
4	fsync_en		clear UART read FIFO
		0	de-asserted. FIFO will be filled with UART replies from camera core.
		1	asserted. FIFO will always report empty (ignore camera UART replies).
5	fsync_dir	-	set fsync direction, only input supported as of document RevA
		0	input
		1	reserved for future sensors which support fsync master output.
7:6	fsync_sel	-	fsync pin select. bits ignored when fsync_en = 0.
		0	fsync input from gpio0. see electrical ICD for availability on PCA variant
		1	fsync input from gpio1. see electrical ICD for availability on PCA variant
		2	fsync input from gpio2. see electrical ICD for availability on PCA variant
		3	fsync input from external connector (default). May also be configured as master mode fsync output (fsync_dir=1) with appropriate camera core settings.

Note 6: 3.3V UART present on MIPI backend PCA, J1. UART2 connection present on MIPI backend PCA, J2. UART2 levels determined by PCA variant (see pinout in electrical ICD). Both UART connections are internally muxed to the same UART logic so only one Rx input may be used at a time while the other remains at bus idle (logic high) state. OK to leave one or both Rx inputs floating because both 3.3V and 1.8V Rx inputs have internal weak pull-ups. Both Tx outputs will repeat camera reply data if uart_sel bit is logic zero.

3.4.3 0x07, xres_low (rd/wr)

Low byte of 16-bit, horizontal (width) resolution register. Resolution specified to determine byte count when generating MIPI row data packets. Video input resolution measured/set on pwr_en 0→1 change.

Bits							
7	6	5	4	3	2	1	0
horizontal resolution, low byte							

Default register Value (POR)	0x80

3.4.4 0x08, xres_high (rd/wr)

High byte of 16-bit horizontal resolution register. Video input resolution measured/set on pwr_en 0➔1 change (also on power up for build configurations which enable camera by default).

Bits							
7	6	5	4	3	2	1	0
horizontal resolution, high byte							

Default register Value (POR)	0x02

3.4.5 0x09, yres_low (rd/wr)

Low byte of 16-bit, vertical (height) resolution register. Resolution specified to determine byte count when generating MIPI row data packets. Video input resolution measured/set on pwr_en 0➔1 change.

Bits							
7	6	5	4	3	2	1	0
vertical resolution, low byte							

Default register Value (POR)	0x00

3.4.6 0x0A, yres_high (rd/wr)

High byte of 16-bit, vertical (height) resolution register. Video input resolution measured/set on pwr_en 0➔1 change (also on power up for build configurations which enable camera by default).

Bits							
7	6	5	4	3	2	1	0
vertical resolution, high byte							

Default register Value (POR)	0x02

3.4.7 0x0B, pixel_format (rd/wr)

MIPI DT (data type) encoded at beginning of each MIPI packet on the physical interface and test pattern control bits. Test pattern requires presence of a camera pixel clock to serve as the timing base for generating the test pattern. Output resolution of the test pattern will match the x_res, y_res registers and does not have to match the camera resolution. May be used to simulate other camera resolutions for the purpose of testing MIPI clk/lane settings which have sufficient bandwidth for a given resolution/frame rate.

Bits							
7	6	5	4	3	2	1	0
test_pat_en	test_pat_ctrl	mipi_DT					

Default register Value (POR)	0x2E

Bit Definitions

Bit(s)	Field Name	Values	Description
5:0	mipi_DT	-	MIPI packet data type. Tx core transmits whatever is in this register. Input format is not analyzed. DT value should be set to match expectations of host system vs. the camera core's output format. Specifically, the MIPI_tx core does not distinguish between YUV and Y16 video data which are both 16-bits per pixel.
		0x12	Embedded 8-bit non-Image. not supported at time of writing.
		0x1E	YUV, 16-bits per pixel {Chroma, Luma}. Typically YUV422.
		0x2A	RAW8. 8-bit per pixel grayscale.
		0x2E	RAW16. 16-bit per pixel grayscale. Byte swap pixel data using *mode_sel*.
6	test_pat_ctl	-	Test pattern 60/30Hz select. Generated as 16-bit pattern but also valid for 8-bit MIPI *pixel_mode*.
		0	60 Hz test pattern. Only valid for lane settings which allow MIPI_tx to finish within 1/60 sec. If frame transmission is longer, then every other frame will be skipped (e.g. 1280x1024, 2-lanes only outputs 30Hz).
		1	30Hz test pattern.
7	test_pat_en	-	Enable test pattern (clock base matches attached camera core).
		0	disable. transmit camera video
		1	enable. transmit synthetic diagonal test pattern (scrolling). Ignores byte swap attribute.

3.4.8 0x0C, ctrl_Reg2 (rd/wr)

Cooler power enable, cooler nStandby, shutter nReset, and UART mux select.

Bits							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
uart_mux_sel					shtr_nrst	cool_nStby	fx3_data_en

Default register Value (POR)	0x16

Bit Definitions

Bit(s)	Field Name	Values	Description
0	fx3_data_en	-	Enable 32-bit pixel bus output to fx3 (UVC video processor)
		0	disabled. video data not available to fx3. (when UVC not in use)
		1	enable 32-bit pixbus driving two, 16-bit pixels per pclk to fx3 for conversion to UVC. Bandwidth required is dependent on paired sensor and FW build.
1	cool_nStby	-	active low, standby pin (if cooler includes a standby input pin)
		0	cooler in reduced power state.
		1	cooler in normal operation mode if powered.
2	shtr_nrst		shutter nRst pin control
		0	shutter in reset
		1	shutter enabled / powered

Bit(s)	Field Name	Values	Description		
7:3	uart_mux_sel	Reg val	Sensor port	Lens port	UART mux selection. Sensor/Lens will be connected to two different destination ports. Other Ports will be set highZ with weak pull-ups. portA, portB, i=i2c, M=mezzanine, f= reserved for fx3 (UVC video IC).
		0x00	i	B	
		0x01	i	A	
		0x02	i	M	
		0x03	i	f	
		0x04	A	f	
		0x05	B	f	
		0x06	M	f	
		0x07	**A**	**B**	
		0x08	B	A	
		0x09	A	M	
		0x0A	B	M	
		0x0B	M	B	
		0x0C	f	B	
		0x0D	f	A	
		0x0E	f	M	
		0x0F	M	A	
		0x10	A	i	
		0x11	B	i	
		0x12	M	i	
		0x13	f	i	

3.4.9 0x0D, ctrl_Reg3 (rd/wr)

Cooler power enable, cooler nStandby, shutter nreset, and UART mux select.

Bits							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**

Default register Value (POR)	0x12

Bit Definitions

Bit(s)	Field Name	Values	Description
2:0	baudS_sel	-	i2c uart bridge, baud rate selection for lens port communication
		0x0	9600
		0x1	57600
		0x2	**115200**
		0x3	460800
		0x4	921600
		x	up to three more rates possible w/ FW updates.

Bit(s)	Field Name	Values	Description
3	uart_rd_clr		Clear UART read FIFO
		0	de-asserted. FIFO will be filled with UART replies from camera core.
		1	asserted. FIFO will always report empty (ignore camera UART replies).
6:4	baudL_sel	-	i2c uart bridge, baud rate selection for lens port communication.
		0x0	9600
		0x1	**57600**
		0x2	115200
		0x3	460800
		0x4	921600
		x	up to three more rates possible w/ FW updates.
7	cont_clk_en	-	continuous clock enable. Primarily used for debug, not recommended.
		0	use pseudo discontinuous clock, mipi_clk enters LP once after each frame.
		1	once enabled, keep mipi_clk in HS mode even between frames.

4 APPENDIX A, EXAMPLE CAMERA COMMANDS (BASH SHELL)

The following examples may be entered in the Linux bash shell for setting various camera modes through the i2c / UART bridge interfaces. The left most number, "9" used in the following examples specifies as-mapped i2cbus #. The bus number is specific to host hardware/device tree mapping. Ensure register address 0x06, ctrl_Reg **(rd/wr)** is set to enable the i2c↔UART bridge. Dest. (destination) column indicates whether the command sets a register in the MIPI FPGA or is forwarded through the UART bridge to camera. Cmd in 1st row of

Table 4-2 enables UART bridge but disables UART readback so camera reply data is ignored. Properly formed camera command packets take effect immediately and reading camera UART reply data is optional.

All commands in the following tables which address i2c reg address 0x04 are directed to the camera or lens . Camera/Lens specific commands are identified with "Snsr"/"Lens" in the "Dest." (destination) column. See Sensor / Lens ICDs for more information about respective protocols.
Color coding: **slave_addr**, **reg_addr**, **uart_payload**. All examples use i2cbus 9, change as needed.

Table 4-1 I2C Bus and Slave Device Address for Examples

i2c bus # on host:	9	example i2c host bus number
i2c slave addr:	0x33	fixed camera slave device address

Table 4-2 General Configuration Commands

Description	Dest.	bash command line string:
i2c 2-lane init:	FPGA	i2ctransfer -f -y 9 w9@**0x33** **0x05** 0xB4 0xCF 0x80 0x02 0x00 0x02 0x2E 0x3E 0x12
2L init w/ byte swap:	FPGA	i2ctransfer -f -y 9 w9@**0x33** **0x05** 0xB6 0xCF 0x80 0x02 0x00 0x02 0x2E 0x3E 0x12
open shutter	SHTR	i2cset -f -y 0x03 **0x52** **0x17** 0x0000 w *(word write)* i2ctransfer -f -y 9 w3@**0x52** **0x17** 0x00 0x00 *(little endian byte order)*
close shutter	SHTR	i2cset -f -y 0x03 0x52 **0x17** 0x0001 w *(word write)* i2ctransfer -f -y 9 w3@**0x52** **0x17** 0x01 0x00 *(little endian byte order)*
snsr execute 1pt	Snsr	i2ctransfer -y -f 3 w7@**0x33** **0x04** 0xAA 0x02 0x84 0x00 0x00 0xD0
snsr portA, lens portB	FPGA	i2ctransfer -f -y 9 w2@**0x33** 0x0C 0x3A
snsr i2c, lens portB	FPGA	i2ctransfer -f -y 9 w2@**0x33** 0x0C 0x06
snsr portA, lens i2c	FPGA	i2ctransfer -f -y 9 w2@**0x33** 0x0C 0x86
		note: other reg 0x06 values allow sensor / lens connectivity to UART port on mezzanine (M).
rd snsr FPA temp	Snsr	i2ctransfer -f -y 9 w2@**0x33** **0x0C** 0x06 **&&** i2ctransfer -y -f 3 w7@**0x33** **0x4** 0xAA 0x21 0x80 0x00 0x00 0xB5 **&&** ret_hex_str=$(i2ctransfer -f -y 9 w1@**0x33** **0x03** r0x0C) ret_hex_str*[0:0xC], temperature is bytes [10:7], little endian (LSB first at byte 7)*
Lens ASCII commands:		Customer cmds will likely use U1/2 protocol. set lens to i2c bridge with uart read fifo enabled, then:
Lens focus→Infinity	Lens	i2ctransfer -f -y 9 w5@**0x33** **0x04** 0x24 0x46 0x49 0x5E
Lens to Wide	Lens	i2ctransfer -f -y 9 w7@**0x33** **0x04** 0x24 0x4D 0x46 0x20 0x31 0x5E
Lens to Mid	Lens	i2ctransfer -f -y 9 w7@**0x33** **0x04** 0x24 0x4D 0x46 0x20 0x32 0x5E
Lens to Narrow	Lens	i2ctransfer -f -y 9 w7@**0x33** **0x04** 0x24 0x4D 0x46 0x20 0x33 0x5E
read 0x00, 32bytes	FPGA	i2ctransfer -f -y 9 w1@0x33 0x00 r32
get Rd_bytes	FPGA	i2cget -f -y 9 0x33 0x02 b
read bridge, 32bytes	FPGA	i2ctransfer -f -y 9 w1@0x33 0x03 r32

Table 4-3 Fsync Configuration (early Gen2 phase1, Sensor FW v1.1.15 only)

Description	Dest.	bash command line string:
setting fsync mode		presumes sensor NVM set for CL fsync input. set sensor to i2c bridge with uart read fifo enabled, then:
snsr OpCmd Tint=15ms, int fsync	Snsr	i2ctransfer -f -y 9 w34@**0x33 0x04** 0xAA 0x02 0x80 0x1B 0x00 0x98 0x3A 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x01 0xA0 0x00 0x00 0x00 0xB0 0x35 0x82 0x00 0x06 0x01 0x02 0x10 0x00 0x00 0x08 0x00 0x00 0xBE
snsr OpCmd Tint=15ms, ext fsync	Snsr	i2ctransfer -f -y 9 w34@**0x33 0x04** 0xAA 0x02 0x80 0x1B 0x00 0x98 0x3A 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x01 0xA0 0x00 0x00 0x00 0xB0 0x35 0x82 0x00 0x02 0x01 0x02 0x10 0x00 0x00 0x08 0x00 0x00 0xC2
set external fsync in	FPGA	i2ctransfer -f -y 9 w2@**0x33 0x06** 0xD7 *(Rev1 PCAs bypass FPGA routing to sensor, use J23)*

- Sensor supplier has deprecate opcode 0x8002 in newer FW releases. Most controls are available as separate, single function commands. External sync mode can only be enabled in the latest sensor FW release via "configuration" which sets parameters in flash which gets loaded at boot time only.

Table 4-4 Fsync Configuration (Gen2 phase2, Sensor FW v1.1.20 and higher)

Description	Dest.	bash command line string:
setting fsync mode		Sensor NVM parameters for CL fsync input select and external sync must be set using sensor tookit "Configurator" using direct UART connection. OpCode 0x8002 has been removed. Dynamic enable/disable not available. The 3-pin header (sensor interface board) or the FPGA pass-through port(s) may be used if configured over i2c.
Read Sensor FW ver	Snsr	i2ctransfer -f -y 9 w2@**0x33 0x0C** 0x06 **&&** i2ctransfer -y -f 3 w7@**0x33 0x4** 0xAA 0x11 0xF0 0x00 0x00 0x55 **&&** ret_str=$(i2ctransfer -f -y 9 w1@**0x33 0x03** r0x11) ret_hex_str[0:17], version is bytes [5:7] in hex*: ret_str[5].ret_str[6].ret_str[7]*
Set frame rate 30Hz	Snsr	i2ctransfer -y -f 3 w8@**0x33 0x04** 0xAA 0x06 0x80 0x01 0x00 0x01 0xCE

- Frame rate must be set (0x8006) to one of the discrete modes which most closely matches an external sync input. OpCode 0x8006 affects the FPA scan duration which limits the valid range of integration times. The externally applied sync input period must be no greater than the frame rate selected by 0x8006. On external sync, the internal scan timing will complete roughly 2µs before the period set by 0x8006. For example, with a selected rate of 30Hz (33.3333ms period) , the externally applied frame rate cannot exceed 30.0018Hz (33.3313ms period); if the external sync exceeds 30.002Hz, no video output will be present.
- Sensor Read Serials (0xF401) uses ASCII encoding for the reply payload bytes. Refer to sensor's Protocol document for location of fields within the 261 byte reply. Reply is mostly null zero data because each field reserves space for 64 characters.
- Video output size may only be set between 640x512 (default) and 640x480 using sensor toolkit configurator. After the removal of OpCode 0x8002, there is no longer a user control to set an arbitrary video crop window.
- Functionality such as resolution, frame rate, and image flip controls are replaced with dedicated opcodes. Some are new to the phase2 sensor FW and are not backward compatible with the phase1 FW release.

5 APPENDIX B, T_SETTLE PARAMETER

In practice, setting the parameter `cil_settletime = "0"` (auto) in a Linux devicetree should be sufficient. If $T_{HS\text{-}SETTLE}$ phy timing parameter must be manually adjusted, Figure 5-1 illustrates the MIPI phy's LP to HS transition and sync byte for the SOF data packet. Note that the bandwidth for the o-scope capture was only 200MHz so the "fast" clk setting is missing some detail (aliased).

Note unit interval, UI = 1/(2*mipi_clk_freq). See Note 5 in section 3.4.1

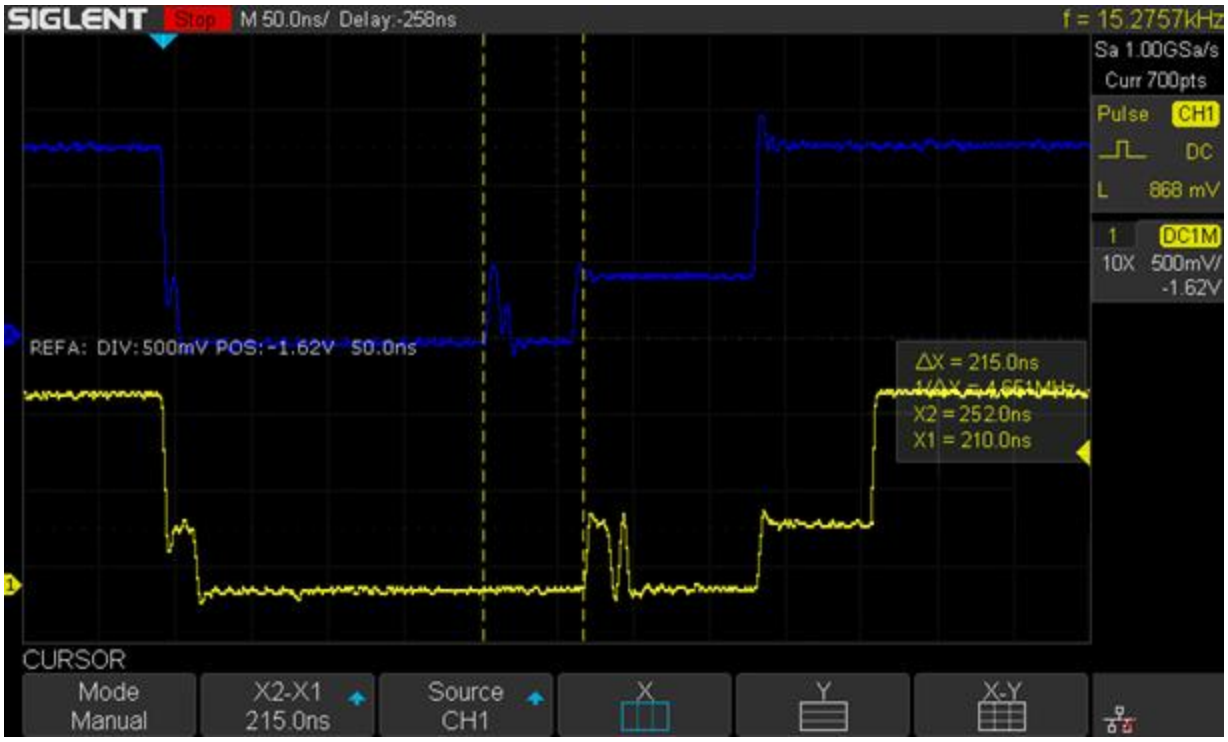


Figure 5-1 Tsettle for mipi_clk_sel settings

	Selected CSI clk rate	CSI_CLK rate	Approximate Tsettle time	inter-frame clk LP11 duration
Blue trace:	mipi_clk_sel = 0	mipi_clk = 189Hz	Tsettle ≈ 300ns	≈ 250us
Yellow trace:	mipi_clk_sel = 1	mipi_clk = 94.5Hz	Tsettle ≈ 300ns	≈ 500us