1/3.2-Inch 8Mp CMOS Digital Image Sensor

AR0832 Data Sheet

For the latest data sheet, refer to Aptina's Web site: www.aptina.com

Features

- Low dark current
- Simple two-wire serial interface
- Auto black level calibration
- Support for external mechanical shutter
- Support for external LED or Xenon flash
- Programmable controls: gain, horizontal and vertical blanking, auto black level offset correction, frame size/rate, exposure, left–right and top–bottom image reversal, window size, and panning
- Data interfaces: parallel or single/dual lanes serial mobile industry processor interface (MIPI)
- On-die phase-locked loop (PLL) oscillator
- Bayer pattern down-size scaler
- Superior low-light performance
- Integrated position and color-based shading correction
- 11.5Kb one-time programmable memory (OTPM) for storing shading correction coefficients of three light sources and module information
- Extended Flash duration that is up to start of frame readout
- Internal VCM driver and access to internal analog-to-digital converter (ADC)

Applications

- Cellular phones
- Digital still cameras
- PC cameras
- PDAs

Table 1: Key Performance Parameters

Parameter	Value
Optical format	1/3.2-inch (4:3)
Active imager size	4.57mm x 3.43mm: 5.71mm diagonal
Active pixels	3264H x 2448V
Pixel size	1.4 x 1.4µm
Chief ray angle	25.0°
Color filter array	RGB Bayer pattern
Shutter type	Electronic rolling shutter (ERS) with global reset release (GRR)
Input clock frequency	6–27 MHz

Table 1: Key Performance Parameters (continued)

Parameter		Value
Maximum data rate	Parallel	100 Mpps at 100 MHz PIXCLK
	MIPI	800 Mbps per lane
Frame rate	Full resolution (3264 x2448) 10-bits:	• Parallel: 11 fps • 1-lane MIPI: 11 fps • 2-lane MIPI: 16 fps
	Full resolution (16:9; 3264 x 1836) 10-bits	• Parallel: 15 fps • 1-lane MIPI: 14 fps • 2-lane MIPI: 21 fps
	Video 1080p (1920x1080)	2-lane MIPI: 30 fps with 2640x1486 (80% FOV, direct cropping and external scaling)
	Video 720p (1280x720)	2-lane MIPI: 60 fps (with 3264x1836 FOV by skip2 mode) or 30 fps (with 3264x1836 FOV by bin2 mode)
	VGA (640x480)	2-lane MIPI: 60 fps (with 3264x2448 FOV by skip4 mode) or 30 fps (with 3264x2448 FOV by skip2 bin2 mode)
ADC resolution		10-bit, on-die
Responsivity		0.82V/lux.s
Dynamic range		66.5dB
SNR_{MAX}		36dB
Supply voltage	V_{AA}, V_{AA_PIX}	2.65 – 3.1 V (2.8 typical)
	V_{DD_TX}, V_{DD_REGIN} (internal regulator)	1.7 – 1.95 V (1.8 typical)
	V_{DD_IO}	1.8V or 2.8V
	V_{DD}, V_{DD_PLL}	1.08 – 1.32 V (1.2 typical)
	V_{PP}	6.0 – 7.0 V (6.5 typical)
Power Consumption	Full resolution (excluding I/O power)	Parallel: 395 mW at 55°C (TYP)
		MIPI: 405 mW at 55°C (TYP)
	Hardware standby/ shutdown (no state retention by using XSHUTDOWN pin)	30µA
Package		Bare die
Operating temperature		–30°C to +70°C (at junction)

Table 2: MIPI 2 Lane Speed and FOV

Modes	Resolution	Mode	Use on Chip Scaler	FOV	Max FPS	MIPI Data Rate
Full Resolution	3264x2448	Full Mode	No	100%	16	800 Mbps/lane
VGA (640x480)	3264x2448	Bin2 + Skip2 + Scaling	Yes	100%	30	300 Mbps/lane
	3264x2448	Skip4 + Scaling	Yes	100%	60	312 Mbps/lane
1080p (1920x1080)	3264x1836	Full Mode + Scaling	No	100% @ 16:9	21.5	800 Mbps/lane
	2640x1846	Full Mode + Scaling	No	80% @ 16:9	30	800 Mbps/lane
720p (1280x720)	3264x1836	Bin2 + Scaling	Yes	100% @ 16:9	30	325 Mbps/lane
	3264x1836	Skip2 + Scaling	Yes	100% @ 16:9	60	800 Mbps/lane

Ordering Information

Table 3: Available Part Numbers

Part Number	Description
AR0832MBSC00SUD20-E	Bare die

Table of Contents

Features . . . 1
Applications . . . 1
Ordering Information . . . 2
General Description . . . 9
Functional Overview . . . 9
  Pixel Array . . . 10
Operating Modes . . . 11
Typical Connections . . . 12
Signal Descriptions . . . 14
Output Data Format . . . 15
  Serial Pixel Data Interface . . . 15
  Parallel Pixel Data Interface . . . 15
  Output Data Timing (Parallel Pixel Data Interface) . . . 16
Two-Wire Serial Register Interface . . . 17
  Protocol . . . 17
    Start Condition . . . 17
    Stop Condition . . . 17
    Data Transfer . . . 18
    Slave Address/Data Direction Byte . . . 18
    Message Byte . . . 18
    Acknowledge Bit . . . 18
    No-Acknowledge Bit . . . 18
  Typical Sequence . . . 18
  Single READ from Random Location . . . 19
  Single READ from Current Location . . . 19
  Sequential READ, Start from Random Location . . . 19
  Sequential READ, Start from Current Location . . . 20
  Single WRITE to Random Location . . . 20
  Sequential WRITE, Start at Random Location . . . 21
Registers . . . 22
  Register Notation . . . 22
    Register Aliases . . . 22
    Bit Fields . . . 22
    Bit Field Aliases . . . 22
    Byte Ordering . . . 23
    Address Alignment . . . 23
    Bit Representation . . . 23
    Data Format . . . 23
  Register Behavior . . . 23
    Double-Buffered Registers . . . 23
    Using grouped_parameter_hold . . . 24
    Bad Frames . . . 24
    Changes to Integration Time . . . 24
    Changes to Gain Settings . . . 25
    Embedded Data . . . 25
Reading the Sensor Revision Number . . . 25
Programming Restrictions . . . 26
    Output Size Restrictions . . . 27
    Effect of Scaler on Legal Range of Output Sizes . . . 27
    Output Data Timing . . . 29
    Changing Registers While Streaming . . . 29

Programming Restrictions when Using Global Reset 30
Control of the Signal Interface 30
Power-Up State 30
MIPI Serial Pixel Data Interface 30
Parallel Pixel Data Interface 31
Output Enable Control 31
Configuration of the Pixel Data Interface 31
System States 32
Power-On Reset Sequence 34
Soft Reset Sequence 34
Signal State During Reset 35
General Purpose Inputs 35
Streaming/Standby Control 36
Trigger Control 36
Clocking 37
PLL Clocking 39
Influence of ccp_data_format 39
Influence of ccp2_signalling_mode 39
Clock Control 39
Features 40
Shading Correction (SC) 40
Bayer Resampler 41
One-Time Programmable Memory (OTPM) 42
Programming and Verifying the OTPM 42
Reading the OTPM 43
Image Acquisition Modes 44
Window Control 44
Pixel Border 44
Readout Modes 45
Horizontal Mirror 45
Vertical Flip 45
Subsampling 45
Programming Restrictions when Subsampling 48
Binning 49
Programming Restrictions When Binning 52
Summing Mode 53
Scaler 54
Frame Rate Control 54
Minimum Row Time 55
Minimum Frame Time 55
Integration Time 56
Fine Integration Time Limits 56
fine_correction 57
Flash Timing Control 57
Global Reset 59
Overview of Global Reset Sequence 59
Entering and Leaving the Global Reset Sequence 60
Programmable Settings 60
Control of the Electromechanical Shutter 61
Using FLASH with Global Reset 62
External Control of Integration Time 63
Retriggering the Global Reset Sequence 64
Using Global Reset with SMIA Data Path 64

Global Reset and Soft Standby 65
Analog Gain 65
Using Per-Color or Global Gain Control. 65
SMIA Gain Model 66
Aptina Gain Model 66
Gain Code Mapping 67
Sensor Core Digital Data Path 68
Test Patterns 68
Effect of Data Path Processing on Test Patterns 69
Solid Color Test Pattern 69
100% Color Bars Test Pattern 69
Fade-to-gray Color Bars Test Pattern 70
PN9 Link Integrity Pattern 71
Walking 1s 72
Test Cursors 72
Digital Gain 74
Pedestal 74
Digital Data Path 75
Embedded Data Format and Control 75
Timing Specifications 76
Power-Up Sequence Using Internal Regulator 76
Power-Up Sequence Using External Regulator 77
Power-Down Sequence Using Internal Regulator 78
Power-Down Sequence Without Internal Regulator 79
Hard Standby and Hard Reset 80
Soft Standby and Soft Reset 80
Soft Standby 80
Soft Reset 81
Internal VCM Driver 81
User -Accessible Internal ADC 82
Multimaster Serial Interface 82
Spectral Characteristics 83
Read the Sensor CRA 84
Electrical Characteristics 85
Two-Wire Serial Register Interface 85
EXTCLK 87
Parallel Pixel Data Interface 88
Serial Pixel Data Interface 91
Control Interface 91
Operating Voltages 92
Absolute Maximum Ratings 94
SMIA and MIPI Specification Reference 94
Revision History 95

List of Figures

Figure 1: Block Diagram 9
Figure 2: Pixel Color Pattern Detail (Top Right Corner) 10
Figure 3: Parallel/MIPI Typical Connections (Using Internal Regulator for VDD and Sensor-Connected PLL) 12
Figure 4: Parallel/MIPI Typical Connections (Not Using Internal Regulator) 13
Figure 5: Spatial Illustration of Image Readout 15
Figure 6: Pixel Data Timing Example 16
Figure 7: Row Timing and FV/LV Signals 16
Figure 8: Single READ from Random Location 19
Figure 9: Single READ from Current Location 19
Figure 10: Sequential READ, Start from Random Location 20
Figure 11: Sequential READ, Start from Current Location 20
Figure 12: Single WRITE to Random Location 20
Figure 13: Sequential WRITE, Start at Random Location 21
Figure 14: Effect of Limiter on the Data Path 28
Figure 15: Timing of Data Path 29
Figure 16: AR0832 System States 32
Figure 17: AR0832 Profile 1/2 Clocking Structure 37
Figure 18: Bayer Resampling 41
Figure 19: Results of Resampling 41
Figure 20: Illustration of Resampling Operation 41
Figure 21: Effect of horizontal_mirror on Readout Order 45
Figure 22: Effect of vertical_flip on Readout Order 45
Figure 23: Effect of x_odd_inc = 3 on Readout Sequence 46
Figure 24: Effect of x_odd_inc = 7 on Readout Sequence 46
Figure 25: Pixel Readout (No Subsampling) 46
Figure 26: Pixel Readout (x_odd_inc = 3, y_odd_inc = 3) 47
Figure 27: Pixel Readout (x_odd_inc = 7, y_odd_inc = 7) 47
Figure 28: Pixel Readout (x_odd_inc = 3, y_odd_inc = 1, x_bin = 1) 49
Figure 29: Pixel Readout (x_odd_inc = 3, y_odd_inc = 3, xy_bin = 1) 50
Figure 30: Pixel Readout (x_odd_inc = 7, y_odd_inc = 7, xy_bin = 1) 50
Figure 31: Pixel Binning and Summing 53
Figure 32: Xenon Flash Enabled 57
Figure 33: LED Flash Enabled 58
Figure 34: Overview of Global Reset Sequence 59
Figure 35: Entering and Leaving a Global Reset Sequence 60
Figure 36: Controlling the Reset and Integration Phases of the Global Reset Sequence 60
Figure 37: Control of the Electromechanical Shutter 61
Figure 38: Controlling the SHUTTER Output 62
Figure 39: Using FLASH with Global Reset 62
Figure 40: Extending FLASH Duration in Global Reset (Reference Readout Start) 63
Figure 41: Global Reset Bulb 64
Figure 42: Entering Soft Standby During a Global Reset Sequence 65
Figure 43: 100 Percent Color Bars Test Pattern 70
Figure 44: Fade-to-Gray Color Bars Test Pattern 71
Figure 45: Walking 1s 10-bit Pattern 72
Figure 46: Walking 1s 8-bit Pattern 72
Figure 47: Test Cursor Behavior with image_orientation 74
Figure 48: Data Path 75
Figure 49: Power-Up Sequence 76
Figure 50: Power-Up Sequence without Internal Regulator 77
Figure 51: Power-Down Sequence 78
Figure 52: Power-Down Sequence Without Internal Regulator 79
Figure 53: Hard Standby and Hard Reset 80
Figure 54: Soft Standby and Soft Reset 81
Figure 55: VCM Driver Typical Diagram 81
Figure 56: Quantum Efficiency 83

Figure 57: Chief Ray Angle 83
Figure 58: Two-Wire Serial Bus Timing Parameters. 85
Figure 59: Parallel Data Output Timing Diagram 86
Figure 60: Fall Slew Rates (Cap Load = 25pF) 89
Figure 61: Rise Slew Rates (Cap Load = 25pF) 90

List of Tables

Table 1: Key Performance Parameters . . . 1
Table 2: MIPI 2 Lane Speed and FOV . . . 2
Table 3: Available Part Numbers . . . 2
Table 4: Signal Descriptions . . . 14
Table 5: Row Timing . . . 16
Table 6: Address Space Regions . . . 22
Table 7: Data Formats . . . 23
Table 8: Definitions for Programming Rules . . . 25
Table 9: Programming Rules . . . 26
Table 10: Output Enable Control . . . 31
Table 11: Configuration of the Pixel Data Interface . . . 31
Table 12: RESET_BAR and PLL in System States . . . 33
Table 13: Signal State During Reset . . . 35
Table 14: Streaming/STANDBY . . . 36
Table 15: Trigger Control . . . 36
Table 16: Default Settings and Range of Values for Divider/Multiplier Registers . . . 37
Table 17: Row Address Sequencing During Subsampling . . . 48
Table 18: Column Address Sequencing During Binning . . . 50
Table 19: Row Address Sequencing During Binning . . . 51
Table 20: Readout Modes . . . 52
Table 21: Readout Modes . . . 53
Table 22: Minimum Row Time and Blanking Numbers . . . 55
Table 23: Minimum Frame Time and Blanking Numbers . . . 55
Table 24: fine_integration_time Limits . . . 56
Table 25: fine_correction Values . . . 57
Table 26: Recommended Gain Settings . . . 66
Table 27: Test Patterns . . . 68
Table 28: Power-Up Sequence . . . 76
Table 29: Power-Up Timing . . . 78
Table 30: Power-Down Sequence . . . 79
Table 31: Power-Down Timing Without Internal Regulator . . . 80
Table 32: VCM Driver Typical . . . 82
Table 33: CRA Value . . . 84
Table 34: Two-Wire Serial Register Interface Electrical Characteristics . . . 85
Table 35: Two-Wire Serial Interface Timing Specifications . . . 86
Table 36: Electrical Characteristics (EXTCLK) . . . 87
Table 37: Electrical Characteristics (Parallel Pixel Data Interface) . . . 88
Table 38: Electrical Characteristics (Serial MIPI Pixel Data Interface) . . . 91
Table 39: DC Electrical Characteristics (Control Interface) . . . 91
Table 40: DC Electrical Definitions and Characteristics (Using Internal Regulator) . . . 92
Table 41: DC Electrical Definitions and Characteristics (Using Exernal Regulator) . . . 92
Table 42: Absolute Maximum Values . . . 94
Table 43: Absolute Max Voltages . . . 94

General Description

The Aptina AR0832 is a 1/3.2-inch CMOS active-pixel digital image sensor with a pixel array of 3264H x 2448V (3280H x 2464V including border pixels). It incorporates sophisticated on-chip camera functions such as mirroring, column and row skip modes, and snapshot mode. It is programmable through a simple two-wire serial interface and has very low power consumption.

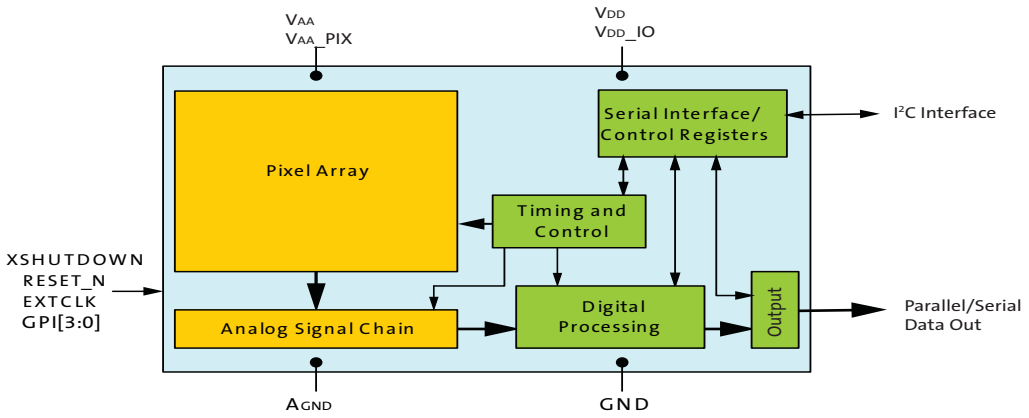
The AR0832 digital image sensor features Aptina's breakthrough low-noise CMOS imaging technology that achieves near-CCD image quality (based on signal-to-noise ratio and low-light sensitivity) while maintaining the inherent size, cost, and integration advantages of CMOS.

The AR0832 sensor can generate full resolution image at up to 15 frames per second (fps). An on-chip analog-to-digital converter (ADC) generates a 10-bit value for each pixel.

Functional Overview

The AR0832 is a progressive-scan sensor that generates a stream of pixel data at a constant frame rate. It uses an on-chip, phase-locked loop (PLL) to generate all internal clocks from a single master input clock running between 6 and 27 MHz. The maximum pixel rate is 100 Mp/s, corresponding to a pixel clock rate of 100 MHz. A block diagram of the sensor is shown in Figure 1.

Figure 1: Block Diagram



The core of the sensor is an 8Mp active-pixel array. The timing and control circuitry sequences through the rows of the array, resetting and then reading each row in turn. In the time interval between resetting a row and reading that row, the pixels in the row integrate incident light. The exposure is controlled by varying the time interval between reset and readout. Once a row has been read, the data from the columns is sequenced through an analog signal chain (providing offset correction and gain), and then through an ADC. The output from the ADC is a 10-bit value for each pixel in the array. The ADC output passes through a digital processing signal chain (which provides further data path corrections and applies digital gain).

The pixel array contains optically active and light-shielded (“dark”) pixels. The dark pixels are used to provide data for on-chip offset-correction algorithms (“black level” control).

The sensor contains a set of control and status registers that can be used to control many aspects of the sensor behavior including the frame size, exposure, and gain setting. These registers can be accessed through a two-wire serial interface.

The output from the sensor is a Bayer pattern; alternate rows are a sequence of either green and red pixels or blue and green pixels. The offset and gain stages of the analog signal chain provide per-color control of the pixel data.

The control registers, timing and control, and digital processing functions shown in Figure 1 on page 9 are partitioned into three logical parts:

- A sensor core that provides array control and data path corrections. The output of the sensor core is a 10-bit parallel pixel data stream qualified by an output data clock (PIXCLK), together with LINE_VALID (LV) and FRAME_VALID (FV) signals.
- A digital shading correction block to compensate for color/brightness shading introduced by the lens or chief ray angle (CRA) curve mismatch.
- Additional functionality is provided. This includes a horizontal and vertical image scaler, a limiter, a data compressor, an output FIFO, and a serializer.

The output FIFO is present to prevent data bursts by keeping the data rate continuous.

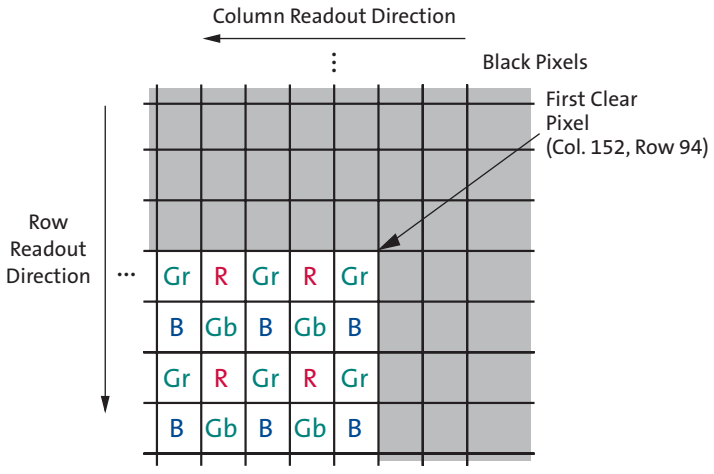
Programmable slew rates are also available to reduce the effect of electromagnetic interference from the output interface.

A flash output signal is provided to allow an external xenon or LED light source to synchronize with the sensor exposure time. Additional I/O signals support the provision of an external mechanical shutter.

Pixel Array

The sensor core uses a Bayer color pattern, as shown in Figure 2. The even-numbered rows contain green and red pixels; odd-numbered rows contain blue and green pixels. Even-numbered columns contain green and blue pixels; odd-numbered columns contain red and green pixels.

Figure 2: Pixel Color Pattern Detail (Top Right Corner)



Operating Modes

By default, the AR0832 powers up with the serial pixel data interface enabled. The sensor operates in serial MIPI mode. This mode is preconfigured at the factory.

The AR0832 also supports parallel configuration. Typical configurations are shown in Figure on page 12 and in Figure 4 on page 13. These operating modes are described in "Control of the Signal Interface" on page 30.

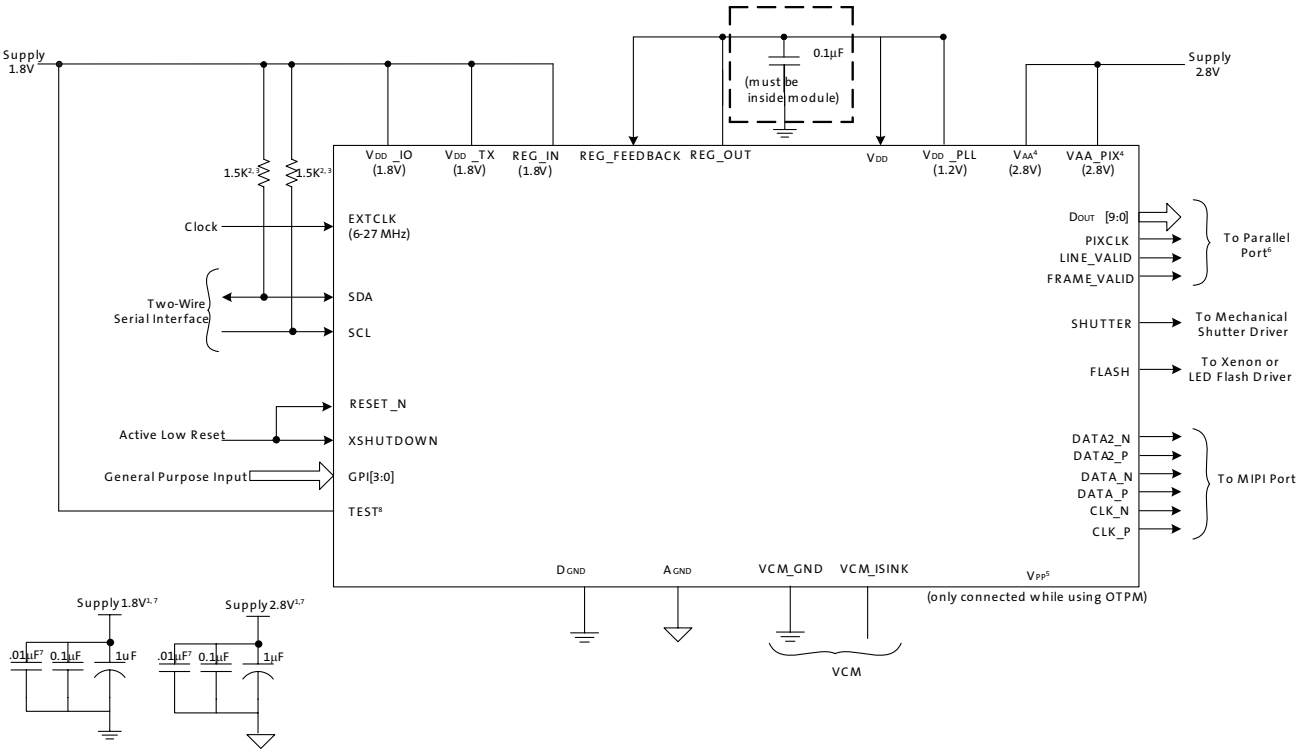
For low-noise operation, the AR0832 requires separate power supplies for analog and digital. Incoming digital and analog ground conductors can be tied together next to the die. Both power supply rails should be decoupled from ground using capacitors as close as possible to the die.

Caution **Aptina does not recommend the use of inductance filters on the power supplies or output signals.**

Typical Connections

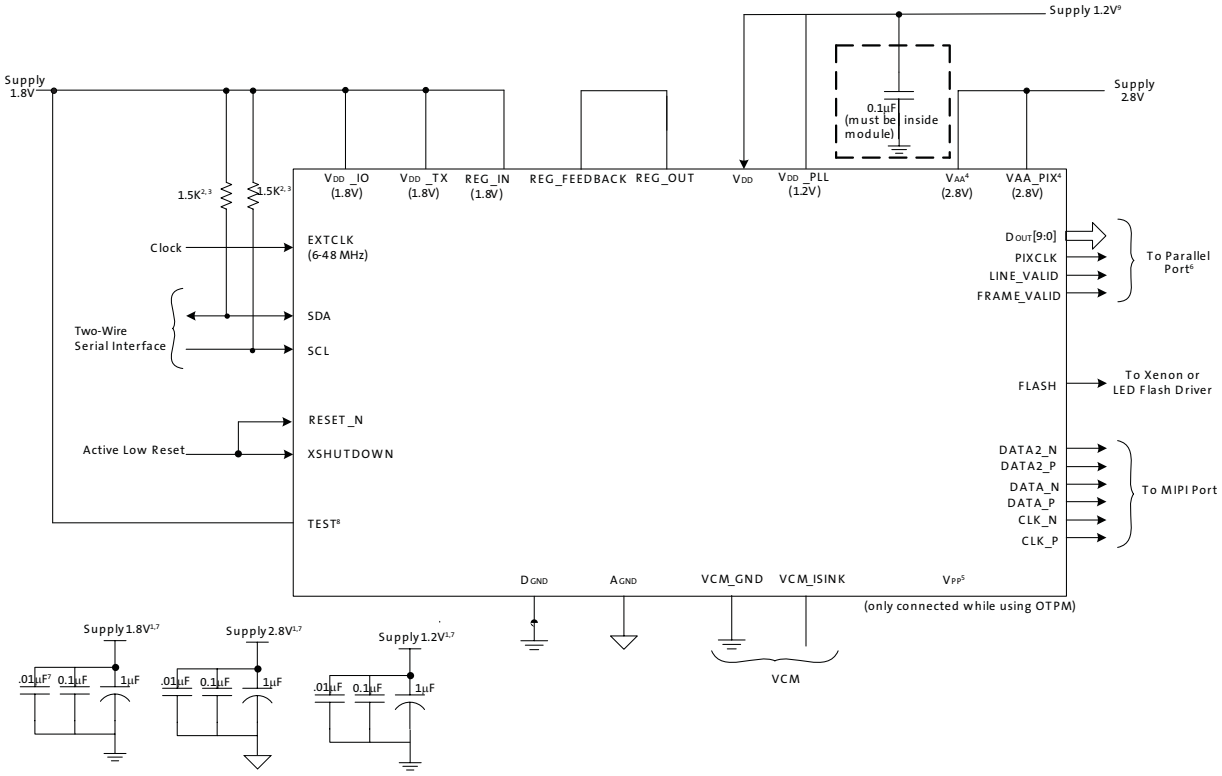
Figure 3 and Figure 4 show the typical AR0832 connections.

Figure 3: Parallel/MIPI Typical Connections (Using Internal Regulator for VDD and Sensor-Connected PLL)



Notes:
1. All power supplies must be adequately decoupled.
2. Aptina recommends using a resistor value of 1.5kΩ, but a greater value can be used for slower two-wire speed.
3. This pull-up resistor is not required if the controller drives a valid logic level on SCL at all times.
4. VAA and VAA_PIX must be tied together.
5. VPP, 6-7V (typically 6.2V), is used for programming OTPM. This pad is left unconnected during normal operation.
6. The parallel interface output pads can be left unconnected if the serial output interface is used.
7. Aptina recommends that 0.01µF, 0.1µF, and 1µF decoupling capacitors for each power supply are mounted as close as possible to the pad. Actual values and results may vary depending on layout and design considerations.
8. For MIPI configuration, TEST must be tied to VDD_IO. For CCP2, TEST must be tied to GND.
9. ATEST1_TOP, ATEST2_TOP, ATEST1_BTM, and ATEST2_BTM should be floating.
10. The serial interface output pads can be left unconnected if the parallel output interface is used.
11. GPI can be used to distinguish module maker. If not used, it should be grounded.
12. When using external VCM, VCM_ISINK must be floating and VCM_GND must be connected to DGND.

Figure 4: Parallel/MIPI Typical Connections (Not Using Internal Regulator)



Notes:
1. All power supplies must be adequately decoupled.
2. Aptina recommends using a resistor value of 1.5kΩ, but a greater value can be used for slower two-wire speed.
3. This pull-up resistor is not required if the controller drives a valid logic level on SCLK at all times.
4. VAA and VAA_PIX must be tied together.
5. VPP, 6-7V (typically 6.2V), is used for programming OTPM. This pad is left unconnected during normal operation.
6. The parallel interface output pads can be left unconnected if the serial output interface is used.
7. Aptina recommends that 0.01µF, 0.1µF, and 1µF decoupling capacitors for each power supply are mounted as close as possible to the pad. Actual values and results may vary depending on layout and design considerations.
8. For MIPI configuration, TEST must be tied to VDD_IO. For CCP2, TEST must be tied to GND.
9. ATEST1_TOP, ATEST2_TOP, ATEST1_BTM, and ATEST2_BTM should be floating.
10. The serial interface output pads can be left unconnected if the parallel output interface is used.
11. GPI can be used to distinguish module maker. If not used, it should be grounded.
12. When using external VCM, VCM_ISINK must be floating and VCM_GND must be connected to DGND.

Signal Descriptions

Table 4 provides signal descriptions for AR0832 die. For pad location and aperture information, refer to the AR0832 die data sheet.

Table 4: Signal Descriptions

Pad Name	Pad Type	Description
EXTCLK	Input	Master clock input, 6–27 MHz.
RESET_BAR	Input	Asynchronous active LOW reset. When asserted, data output stops and all internal registers are restored to their factory default settings. Can be tied high when XSHUTDOWN is used as the reset signal with the internal POR.
XSHUTDOWN	Input	Asynchronous active LOW reset. When asserted, data output stops and all internal registers are restored to their factory default settings. This pin will turn off the digital power domain and is the lowest power state of the sensor.
SCLK	Input	Serial clock for access to control and status registers.
GPI[3:0]	Input	General purpose inputs. After reset, these pads are powered-down by default; this means that it is not necessary to bond to these pads. Any of these pads can be configured to provide hardware control of the standby, output enable, SADDR select, and shutter trigger functions. Aptina recommends that unused GPI pins be tied to DGND, but can also be left floating.
TEST	Input	This pin is used to select the power-up state of the serial interface (CCP2 or MIPI). Wire to digital GND for CCP2 operation, or to digital VDD for MIPI operation. (Note: For MIPI configuration, TEST must be tied to VDD_IO. For CCP2, TEST must be tied to GND.)
SDATA	I/O	Serial data from reads and writes to control and status registers.
DATA_P	Output	Differential MIPI (sub-LVDS) serial data (positive).
DATA_N	Output	Differential MIPI (sub-LVDS) serial data (negative).
DATA2_P	Output	Differential MIPI (sub-LVDS) serial data 2nd lane (positive). Can be left floating when using 1-lane MIPI serial interface.
DATA2_N	Output	Differential MIPI (sub-LVDS) serial data 2nd lane (negative). Can be left floating when using 1-lane MIPI serial interface.
CLK_P	Output	Differential MIPI (sub-LVDS) serial clock/strobe (positive).
CLK_N	Output	Differential MIPI (sub-LVDS) serial clock/strobe (negative).
LINE_VALID	Output	LINE_VALID (LV) output. Qualified by PIXCLK.
FRAME_VALID	Output	FRAME_VALID (FV) output. Qualified by PIXCLK.
DOUT[9:0]	Output	Parallel pixel data output. Qualified by PIXCLK.
PIXCLK	Output	Pixel clock. Used to qualify the LV, FV, and DOUT[9:0] outputs.
FLASH	Output	Flash output. Synchronization pulse for external light source. Can be left floating if not used.
SHUTTER	Output	Control for external mechanical shutter. Can be left floating if not used.
VPP	Supply	Power supply used to program one-time programmable (OTP) memory. Disconnect pad when programming or when feature is not used.
VDD_TX	Supply	Digital PHY power supply. Digital power supply for the serial interface.
VAA	Supply	Analog power supply (2.8V).
VAA_PIX	Supply	Analog power supply for the pixel array(2.8V).
AGND	Supply	Analog ground.
VDD	Supply	1.2V digital power supply inputs.
VDD_IO	Supply	I/O power supply (1.8V).
DGND	Supply	Common ground for digital and I/O.
VDD_PLL	I/O	1.2V PLL power supply (connect to external capacitor).
VCM_ISINK	I/O	Connected to VCM actuator. 100 mA max, 3.3V max
VCM_GND	I/O	Connected to VCM actuator
REG_OUT	I/O	1.2V regulator output node.

Table 4: Signal Descriptions (continued)

Pad Name	Pad Type	Description
REG_IN	I/O	When using internal regulator, it needs to be connected to external 1.8V.
REG_FEEDBACK	I/O	When using internal regulator, it needs to be connected to REG_OUT. this pad is receiving the 1.2 feedback from REG_OUT.

Output Data Format

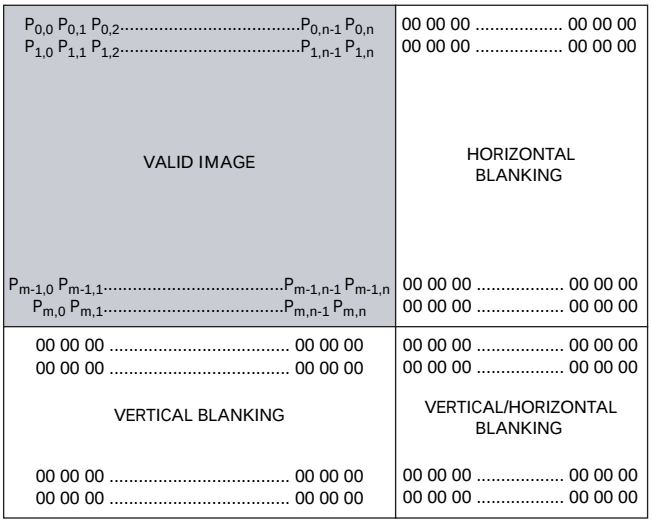
Serial Pixel Data Interface

The AR0832 serial pixel data interface implements data/clock and data/strobe signaling in accordance with the MIPI specifications. The RAW10/RAW8 image data format is supported.

Parallel Pixel Data Interface

AR0832 image data is read out in a progressive scan. Valid image data is surrounded by horizontal blanking and vertical blanking, as shown in Figure 5. The amount of horizontal blanking and vertical blanking is programmable; LV is HIGH during the shaded region of the figure. FV timing is described in the “Output Data Timing (Parallel Pixel Data Interface)”.

Figure 5: Spatial Illustration of Image Readout



Output Data Timing (Parallel Pixel Data Interface)

AR0832 output data is synchronized with the PIXCLK output. When LV is HIGH, one pixel value is output on the 10-bit DOUT output every PIXCLK period. The pixel clock frequency can be determined based on the sensor's master input clock and internal PLL configuration. The rising edges on the PIXCLK signal occurs one-half of a pixel clock period after transitions on LV, FV, and DOUT (see Figure 6 on page 16). This allows PIXCLK to be used as a clock to sample the data. PIXCLK is continuously enabled, even during the blanking period. The AR0832 can be programmed to delay the PIXCLK edge relative to the DOUT transitions. This can be achieved by programming the corresponding bits in the row_speed register. The parameters P, A, and Q in Figure 7 on page 16 are defined in Table 5 on page 16.

Figure 6: Pixel Data Timing Example

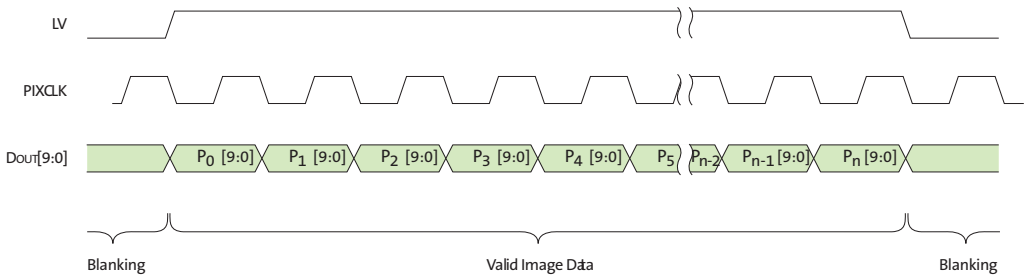


Figure 7: Row Timing and FV/LV Signals

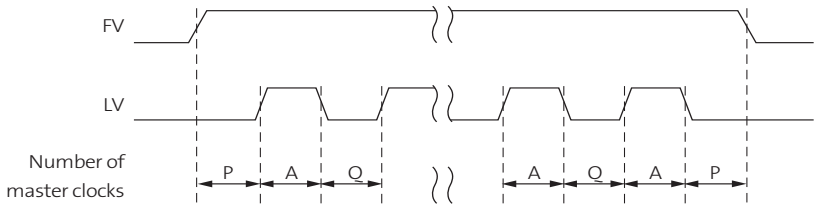


Table 5: Row Timing

Parameter	Name	Equation	Default Timing
PIXCLK_PERIOD	Internal pixel clock period	R0x3016–7[2:0] / vt_pix_clk_freq_mhz	1 pixel clock = 5ns Note: vt_pix_clk_freq_mhz is default 200MHz.)
OP_PIXCLK_PERIOD	Output pixel clock period	R0x3016–7[10:8] / op_pix_clk_freq_mhz	10ns
S	Skip (subsampling) factor	For x_odd_inc = y_odd_inc = 3, S = 2. For x_odd_inc = y_odd_inc = 7, S = 4. otherwise, S = 1	1
A	Active data time	(x_addr_end – x_addr_start + x_odd_inc) * OP_PIXCLK_PERIOD/S	6 OP_PIXCLK_PERIOD =60ns
P	Frame start/end blanking	6 * OP_PIXCLK_PERIOD	6 output pixel clocks = 60ns
Q	Horizontal blanking	(line_length_pck * PIXCLK_PERIOD – A)	162 pixel clocks = 0.81µs
A + Q	Row time	line_length_pck * PIXCLK_PERIOD	6690 pixel clocks = 33.45µs
N	Number of rows	(y_addr_end - y_addr_start + y_odd_inc)/S	2448 rows
V	Vertical blanking	((frame_length_lines - N) * (A+Q)) + Q – (2*P)	4.78ms

Table 5: Row Timing

Parameter	Name	Equation	Default Timing
T	Frame valid time	(N * (A + Q)) – Q + (2*P)	81.89ms
F	Total frame time	line_length_pck * frame_length_lines * PIXCLK_PERIOD	17333790 pixel clocks = 86.67ms Note: line_length_pck is register 0x300C-D

The sensor timing is shown in terms of internal and output pixel clock periods (see Figure 6). The default settings for the on-chip PLL generate a 100 MHz output pixel clock (op_pix_clk) given a 25 MHz input clock to the AR0832. Equations for calculating the frame rate are given in "Frame Rate Control" on page 54.

Two-Wire Serial Register Interface

The two-wire serial interface bus enables read/write access to control and status registers within the AR0832. This interface is designed to be compatible with the electrical characteristics and transfer protocols of the I^2C specification.

The interface protocol uses a master/slave model in which a master controls one or more slave devices. The sensor acts as a slave device. The master generates a clock (SCLK) that is an input to the sensor and is used to synchronize transfers. Data is transferred between the master and the slave on a bidirectional signal (SDATA). SDATA is pulled up to VDD off-chip by a 1.5kΩ resistor. Either the slave or master device can drive SDATA LOW—the interface protocol determines which device is allowed to drive SDATA at any given time.

The protocols described in the two-wire serial interface specification allow the slave device to drive SCLK LOW; the AR0832 uses SCLK as an input only and therefore never drives it LOW.

Protocol

Data transfers on the two-wire serial interface bus are performed by a sequence of low-level protocol elements:

1. a (repeated) start condition
2. a slave address/data direction byte
3. an (a no) acknowledge bit
4. a message byte
5. a stop condition

The bus is idle when both SCLK and SDATA are HIGH. Control of the bus is initiated with a start condition, and the bus is released with a stop condition. Only the master can generate the start and stop conditions.

Start Condition

A start condition is defined as a HIGH-to-LOW transition on SDATA while SCLK is HIGH. At the end of a transfer, the master can generate a start condition without previously generating a stop condition; this is known as a "repeated start" or "restart" condition.

Stop Condition

A stop condition is defined as a LOW-to-HIGH transition on SDATA while SCLK is HIGH.

Data Transfer

Data is transferred serially, 8 bits at a time, with the MSB transmitted first. Each byte of data is followed by an acknowledge bit or a no-acknowledge bit. This data transfer mechanism is used for the slave address/data direction byte and for message bytes.

One data bit is transferred during each SCLK clock period. SDATA can change when SCLK is LOW and must be stable while SCLK is HIGH.

Slave Address/Data Direction Byte

Bits [7:1] of this byte represent the device slave address and bit [0] indicates the data transfer direction. A "0" in bit [0] indicates a WRITE, and a "1" indicates a READ. The default slave addresses used by the AR0832 for the MIPI configured sensor are 0x6C (write address) and 0x6D (read address) in accordance with the MIPI specification. Alternate slave addresses of 0x6E(write address) and 0x6F(read address) can be selected by enabling and asserting the SADDR signal through the GPI pad.

An alternate slave address can also be programmed through R0x31FC.

Message Byte

Message bytes are used for sending register addresses and register write data to the slave device and for retrieving register read data.

Acknowledge Bit

Each 8-bit data transfer is followed by an acknowledge bit or a no-acknowledge bit in the SCLK clock period following the data transfer. The transmitter (which is the master when writing, or the slave when reading) releases SDATA. The receiver indicates an acknowledge bit by driving SDATA LOW. As for data transfers, SDATA can change when SCLK is LOW and must be stable while SCLK is HIGH.

No-Acknowledge Bit

The no-acknowledge bit is generated when the receiver does not drive SDATA LOW during the SCLK clock period following a data transfer. A no-acknowledge bit is used to terminate a read sequence.

Typical Sequence

A typical READ or WRITE sequence begins by the master generating a start condition on the bus. After the start condition, the master sends the 8-bit slave address/data direction byte. The last bit indicates whether the request is for a read or a write, where a "0" indicates a write and a "1" indicates a read. If the address matches the address of the slave device, the slave device acknowledges receipt of the address by generating an acknowledge bit on the bus.

If the request was a WRITE, the master then transfers the 16-bit register address to which the WRITE should take place. This transfer takes place as two 8-bit sequences and the slave sends an acknowledge bit after each sequence to indicate that the byte has been received. The master then transfers the data as an 8-bit sequence; the slave sends an acknowledge bit at the end of the sequence. The master stops writing by generating a (re)start or stop condition.

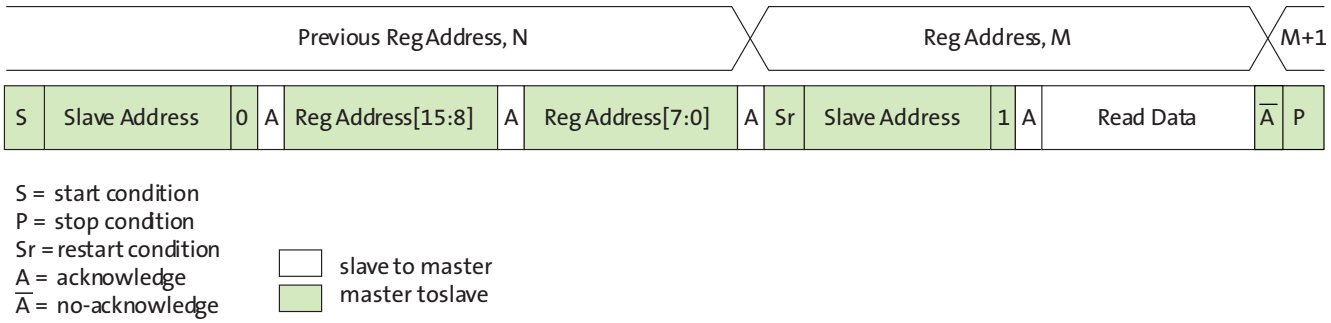
If the request was a READ, the master sends the 8-bit write slave address/data direction byte and 16-bit register address, the same way as with a WRITE request. The master then generates a (re)start condition and the 8-bit read slave address/data direction byte, and clocks out the register data, eight bits at a time. The master generates an acknowledge

bit after each 8-bit transfer. The slave's internal register address is automatically incremented after every 8 bits are transferred. The data transfer is stopped when the master sends a no-acknowledge bit.

Single READ from Random Location

This sequence (Figure 8 on page 19) starts with a dummy WRITE to the 16-bit address that is to be used for the READ. The master terminates the WRITE by generating a restart condition. The master then sends the 8-bit read slave address/data direction byte and clocks out one byte of register data. The master terminates the READ by generating a no-acknowledge bit followed by a stop condition. Figure 8 shows how the internal register address maintained by the AR0832 is loaded and incremented as the sequence proceeds.

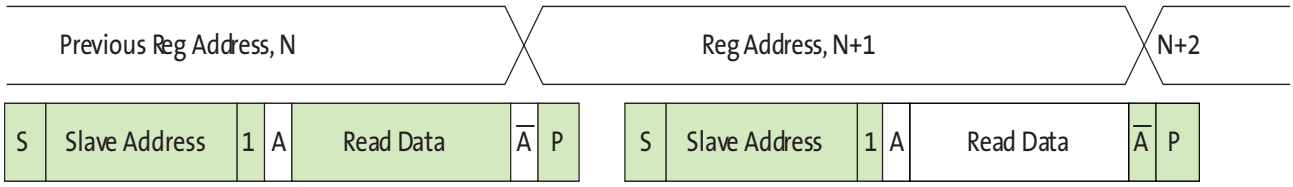
Figure 8: Single READ from Random Location



Single READ from Current Location

This sequence (Figure 9) performs a read using the current value of the AR0832 internal register address. The master terminates the READ by generating a no-acknowledge bit followed by a stop condition. The figure shows two independent READ sequences.

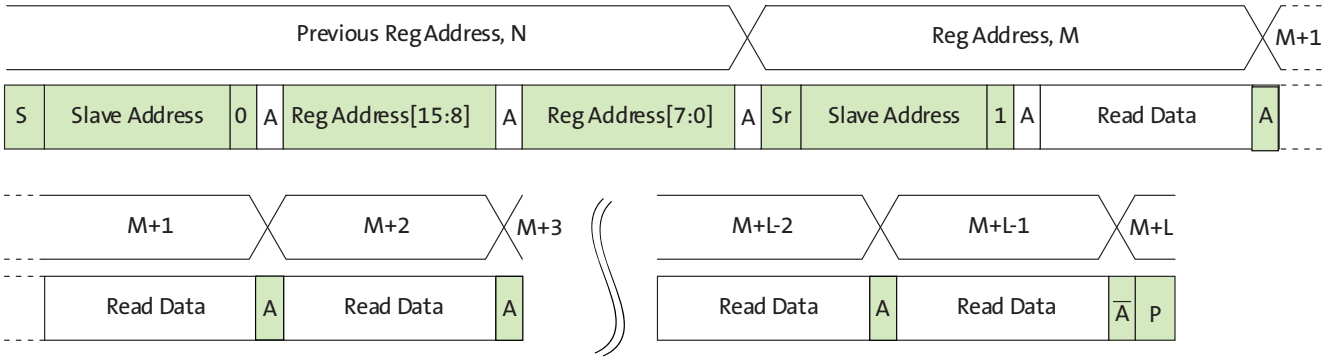
Figure 9: Single READ from Current Location



Sequential READ, Start from Random Location

This sequence (Figure 10) starts in the same way as the single READ from random location (Figure 8). Instead of generating a no-acknowledge bit after the first byte of data has been transferred, the master generates an acknowledge bit and continues to perform byte READs until "L" bytes have been read.

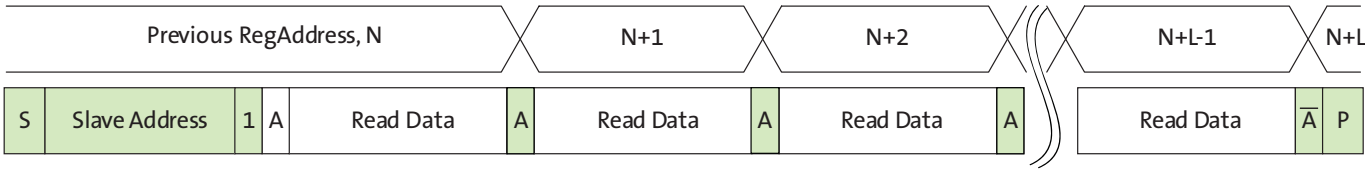
Figure 10: Sequential READ, Start from Random Location



Sequential READ, Start from Current Location

This sequence (Figure 11) starts in the same way as the single READ from current location (Figure 9 on page 19). Instead of generating a no-acknowledge bit after the first byte of data has been transferred, the master generates an acknowledge bit and continues to perform byte READs until "L" bytes have been read.

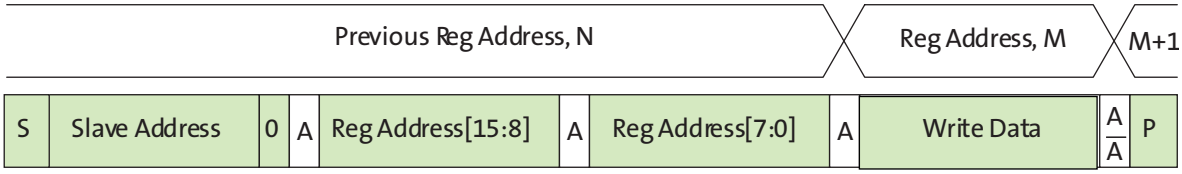
Figure 11: Sequential READ, Start from Current Location



Single WRITE to Random Location

This sequence (Figure 12) begins with the master generating a start condition. The slave address/data direction byte signals a WRITE and is followed by the HIGH then LOW bytes of the register address that is to be written. The master follows this with the byte of write data. The WRITE is terminated by the master generating a stop condition.

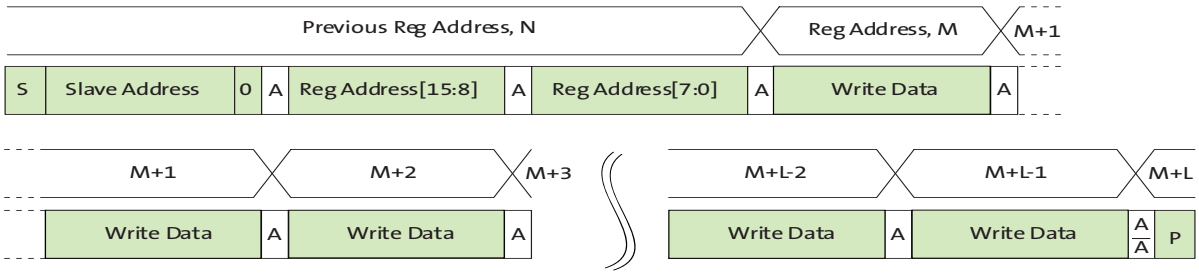
Figure 12: Single WRITE to Random Location



Sequential WRITE, Start at Random Location

This sequence (Figure 13) starts in the same way as the single WRITE to random location (Figure 12). Instead of generating a no-acknowledge bit after the first byte of data has been transferred, the master generates an acknowledge bit and continues to perform byte WRITEs until "L" bytes have been written. The WRITE is terminated by the master generating a stop condition.

Figure 13: Sequential WRITE, Start at Random Location



Registers

The AR0832 provides a 16-bit register address space accessed through a serial interface (“Two-Wire Serial Register Interface” on page 17). Each register location is 8 or 16 bits in size.

The address space is divided into the five major regions shown in Table 6. The remainder of this section describes these registers in detail.

Table 6: Address Space Regions

Address Range	Description
0x0000–0x0FFF	Configuration registers (read-only and read-write dynamic registers)
0x1000–0x1FFF	Parameter limit registers (read-only static registers)
0x2000–0x2FFF	Image statistics registers (none currently defined)
0x3000–0x3FFF	Manufacturer-specific registers (read-only and read-write dynamic registers)
0x4000–0xFFFF	Reserved (undefined)

Register Notation

The underlying mechanism for reading and writing registers provides byte write capability. However, it is convenient to consider some registers as multiple adjacent bytes. The AR0832 uses 8-bit, 16-bit, and 32-bit registers, all implemented as 1 or more bytes at naturally aligned, contiguous locations in the address space.

In this document, registers are described either by address or by name. When registers are described by address, the size of the registers is explicit. For example, R0x3024 is an 8-bit register at address 0x3024, and R0x3000–1 is a 16-bit register at address 0x3000–0x3001. When registers are described by name, the size of the register is implicit. It is necessary to refer to the register table to determine that model_id is a 16-bit register.

Register Aliases

A consequence of the internal architecture of the AR0832 is that some registers are decoded at multiple addresses. Some registers in “configuration space” are also decoded in “manufacturer-specific space.” To provide unique names for all registers, the name of the register within manufacturer-specific register space has a trailing underscore. For example, R0x0000–1 is model_id, and R0x3000–1 is model_id_. The effect of reading or writing a register through any of its aliases is identical.

Bit Fields

Some registers provide control of several different pieces of related functionality, and this makes it necessary to refer to bit fields within registers. As an example of the notation used for this, the least significant 4 bits of the model_id register are referred to as model_id[3:0] or R0x0000–1[3:0].

Bit Field Aliases

In addition to the register aliases described above, some register fields are aliased in multiple places. For example, R0x0100 (mode_select) has only one operational bit, R0x0100[0]. This bit is aliased to R0x301A–B[2]. The effect of reading or writing a bit field through any of its aliases is identical.

Byte Ordering

Registers that occupy more than one byte of address space are shown with the lowest address in the highest-order byte lane to match the byte-ordering on the data bus. For example, the model_id register is R0x0000–1. In the register table the default value is shown as 0x4B00. This means that a read from address 0x0000 would return 0x4B, and a read from address 0x0001 would return 0x00. When reading this register as two 8-bit transfers on the serial interface, the 0x4B will appear on the serial interface first, followed by the 0x00.

Address Alignment

All register addresses are aligned naturally. Registers that occupy 2 bytes of address space are aligned to even 16-bit addresses, and registers that occupy 4 bytes of address space are aligned to 16-bit addresses that are an integer multiple of 4.

Bit Representation

For clarity, 32-bit hex numbers are shown with an underscore between the upper and lower 16 bits. For example: 0x3000_01AB.

Data Format

Most registers represent an unsigned binary value or set of bit fields. For all other register formats, the format is stated explicitly at the start of the register description. The notation for these formats is shown in Table 7.

Table 7: Data Formats

Name	Description
FIX16	Signed fixed-point, 16-bit number: two's complement number, 8 fractional bits. Examples: 0x0100 = 1.0, 0x8000 = –128, 0xFFFF = –0.0039065
UFIX16	Unsigned fixed-point, 16-bit number: 8.8 format. Examples: 0x0100 = 1.0, 0x280 = 2.5
FLP32	Signed floating-point, 32-bit number: IEEE 754 format. Example: 0x4280_0000 = 64.0

Register Behavior

Registers vary from "read-only," "read/write," and "read, write-1-to-clear."

Double-Buffered Registers

Some sensor settings cannot be changed during frame readout. For example, changing R0x0344–5 (x_addr_start) partway through frame readout would result in inconsistent row lengths within a frame. To avoid this, the AR0832 double-buffers many registers by implementing a "pending" and a "live" version. Reads and writes access the pending register. The live register controls the sensor operation.

The value in the pending register is transferred to a live register at a fixed point in the frame timing, called frame start. Frame start is defined as the point at which the first dark row is read out internally to the sensor. In the register tables the "Frame Sync'd" column shows which registers or register fields are double-buffered in this way.

Using grouped_parameter_hold

Register grouped_parameter_hold (R0x0104) can be used to inhibit transfers from the pending to the live registers. When the AR0832 is in streaming mode, this register should be written to "1" before making changes to any group of registers where a set of changes is required to take effect simultaneously. When this register is written to "0," all transfers from pending to live registers take place on the next frame start.

An example of the consequences of failing to set this bit follows:

An external auto exposure algorithm might want to change both gain and integration time between two frames. If the next frame starts between these operations, it will have the new gain, but not the new integration time, which would return a frame with the wrong brightness that might lead to a feedback loop with the AE algorithm resulting in flickering.

Bad Frames

A bad frame is a frame where all rows do not have the same integration time or where offsets to the pixel values have changed during the frame.

Many changes to the sensor register settings can cause a bad frame. For example, when line_length_pck (R0x0342–3) is changed, the new register value does not affect sensor behavior until the next frame start. However, the frame that would be read out at that frame start will have been integrated using the old row width, so reading it out using the new row width would result in a frame with an incorrect integration time.

By default, bad frames are masked. If the masked bad frame option is enabled, both LV and FV are inhibited for these frames so that the vertical blanking time between frames is extended by the frame time.

In the register tables, the "Bad Frame" column shows where changing a register or register field will cause a bad frame. This notation is used:

N—No. Changing the register value will not produce a bad frame.
Y—Yes. Changing the register value might produce a bad frame.
YM—Yes; but the bad frame will be masked out when mask_corrupted_frames (R0x0105) is set to "1."

Changes to Integration Time

If the integration time is changed while FV is asserted for frame *n*, the first frame output using the new integration time is frame *(n + 2)*. The sequence is as follows:

1. During frame *n*, the new integration time is held in the pending register.
2. At the start of frame (*n + 1*), the new integration time is transferred to the live register. Integration for each row of frame (*n + 1*) has been completed using the old integration time.
3. The earliest time that a row can start integrating using the new integration time is immediately after that row has been read for frame (*n + 1*). The actual time that rows start integrating using the new integration time is dependent upon the new value of the integration time.
4. When frame (*n + 2*) is read out, it will have been integrated using the new integration time.

If the integration time is changed on successive frames, each value written will be applied for a single frame; the latency between writing a value and it affecting the frame readout remains at two frames.

Changes to Gain Settings

Usually, when the gain settings are changed, the gain is updated on the next frame start. When the integration time and the gain are changed at the same time, the gain update is held off by one frame so that the first frame output with the new integration time also has the new gain applied. In this case, a new gain should not be set during the extra frame delay. There is an option to turn off the extra frame delay by setting R0x301A–B[14].

Embedded Data

The current values of implemented registers in the address range 0x0000–0x0FFF can be generated as part of the pixel data. This embedded data is enabled by default when the serial pixel data interface is enabled.

The current value of a register is the value that was used for the image data in that frame. In general, this is the live value of the register. The exceptions are:

- The integration time is delayed by one further frame, so that the value corresponds to the integration time used for the image data in the frame. See “Changes to Integration Time” on page 24.
- The PLL timing registers are not double-buffered because the result of changing them in streaming mode is undefined. Therefore, the pending and live values for these registers are equivalent.

For further details, see “Embedded Data Format and Control” on page 75.

Reading the Sensor Revision Number

Follow the steps below to obtain the revision number of the image sensor:

1. Set the register bit field R0x301A[5] = 1.
2. Read the register bit fields R0x31FE[3:0].
3. Convert the binary number to decimal to obtain customer revision. For example, binary value “0010” = sensor revision 2.

Table 8: Definitions for Programming Rules

Name	Definition
xskip	xskip = 1 if x_odd_inc = 1 xskip = 2 if x_odd_inc = 3 xskip = 4 if x_odd_inc = 7
yskip	yskip = 1 if y_odd_inc = 1 yskip = 2 if y_odd_inc = 3 yskip = 4 if y_odd_inc = 7

Programming Restrictions

Table 9 shows a list of programming rules that must be adhered to for correct operation of the AR0832. It is recommended that these rules are encoded into the device driver stack—either implicitly or explicitly.

Table 9: Programming Rules

Parameter	Minimum Value	Maximum Value
coarse_integration_time	8 rows	frame_length_lines - coarse_integration_time_max_margin
fine_integration_time	fine_integration_time_min	line_length_pck - fine_integration_time_max_margin
digital_gain_*	digital_gain_min	digital_gain_max
digital_gain_* is an integer multiple of digital_gain_step_size		
frame_length_lines	min_frame_length_lines	max_frame_length_lines
line_length_pck	min_line_length_pck	max_line_length_pck
	((x_addr_end - x_addr_start + x_odd_inc) /xskip) + min_line_blanking_pck	
	line_length_pck ≥ (x_output_size + constant) * "vt_pix_clk period" / "op_pix_clk period Note: Constant is 0x20 for parallel and 0x78 for MIPI.	
frame_length_lines	min_frame_length_lines	
	((y_addr_end - y_addr_start + y_odd_inc)/yskip) + min_frame_blanking_lines	
x_addr_start (must be an even number)	x_addr_min	x_addr_max
x_addr_end (must be an odd number)	x_addr_start	x_addr_max
(x_addr_end – x_addr_start + x_odd_inc)	Must be multiple of 8 for skip1, 16 for skip2, 32 for skip4	must be positive
y_addr_start (must be an even number)	y_addr_min	y_addr_max
y_addr_end (must be an odd number)	y_addr_start	y_addr_max
(y_addr_end – y_addr_start + y_odd_inc)	Must be multiple of 8 for skip1, 16 for skip2, 32 for skip4	must be positive
x_even_inc (must be an even number)	min_even_inc	max_even_inc
y_even_inc (must be an even number)	min_even_inc	max_even_inc
x_odd_inc (must be an odd number)	min_odd_inc	max_odd_inc
y_odd_inc (must be an odd number)	min_odd_inc	max_odd_inc
scale_m	scaler_m_min	scaler_m_max
scale_n	scaler_n_min	scaler_n_max

Table 9: Programming Rules (continued)

Parameter	Minimum Value	Maximum Value
x_output_size (must be even number – this is enforced in hardware)	320	3280
y_output_size (must be even number – this is enforced in hardware)	2	frame_length_lines

Output Size Restrictions

The design specification imposes the restriction that an output line (the gap between CCP2 start and stop codes) is a multiple of 32 bits in length. This imposes an additional restriction on the legal values of x_output_size:

- When ccp_data_format[7:0] = 10 (RAW10/RAW8 data), x_output_size must be a multiple of 16 (x_output_size[3:0] = 0).

This restriction only applies when the serial pixel data path is in use. It can be met by rounding up x_output_size to an appropriate multiple. Any extra pixels in the output image as a result of this rounding contain undefined pixel data but are guaranteed not to cause false synchronization on the serial data stream.

When the parallel pixel data path is in use, the only restriction on x_output_size is that it must be even (x_output_size[0] = 0), and this restriction is enforced in hardware.

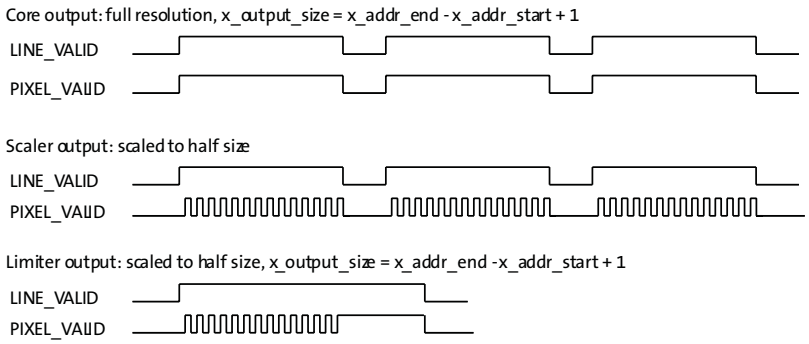
When the serial pixel data path is in use, there is an additional restriction that x_output_size must be small enough such that the output row time (set by x_output_size, the framing and CRC overhead of 12 bytes and the output clock rate) must be less than the row time of the video array (set by line_length_pck and the video timing clock rate).

Effect of Scaler on Legal Range of Output Sizes

When the scaler is enabled, it is necessary to adjust the values of x_output_size and y_output_size to match the image size generated by the scaler. The AR0832 will operate incorrectly if the x_output_size and y_output_size are significantly larger than the output image.

To understand the reason for this, consider the situation where the sensor is operating at full resolution and the scaler is enabled with a scaling factor of 32 (half the number of pixels in each direction). This situation is shown in Figure 14 on page 28.

Figure 14: Effect of Limiter on the Data Path



In Figure 14, three different stages in the data path (see "Digital Data Path" on page 75) are shown. The first stage is the output of the sensor core. The core is running at full resolution and x_output_size is set to match the active array size. The LV signal is asserted once per row and remains asserted for *N* pixel times. The PIXEL_VALID signal toggles with the same timing as LV, indicating that all pixels in the row are valid.

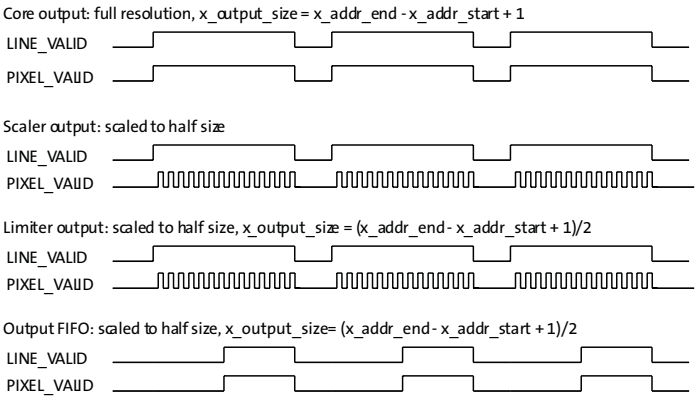
The second stage is the output of the scaler, when the scaler is set to reduce the image size by one-half in each dimension. The effect of the scaler is to combine groups of pixels. Therefore, the row time remains the same, but only half the pixels out of the scaler are valid. This is signalled by transitions in PIXEL_VALID. Overall, PIXEL_VALID is asserted for (*N*/2) pixel times per row.

The third stage is the output of the limiter when the x_output_size is still set to match the active array size. Because the scaler has reduced the amount of valid pixel data without reducing the row time, the limiter attempts to pad the row with (*N*/2) additional pixels. If this has the effect of extending LV across the whole of the horizontal blanking time, the AR0832 will cease to generate output frames.

A correct configuration is shown in Figure 15 on page 29, in addition to showing the x_output_size reduced to match the output size of the scaler. In this configuration, the output of the limiter does not extend LV.

Figure 15 on page 29 also shows the effect of the output FIFO, which forms the final stage in the data path. The output FIFO merges the intermittent pixel data back into a contiguous stream. Although not shown in this example, the output FIFO is also capable of operating with an output clock that is at a different frequency from its input clock.

Figure 15: Timing of Data Path



Output Data Timing

The output FIFO acts as a boundary between two clock domains. Data is written to the FIFO in the VT (video timing) clock domain. Data is read out of the FIFO in the OP (output) clock domain.

When the scaler is disabled, the data rate in the VT clock domain is constant and uniform during the active period of each pixel array row readout. When the scaler is enabled, the data rate in the VT clock domain becomes intermittent, corresponding to the data reduction performed by the scaler.

A key constraint when configuring the clock for the output FIFO is that the frame rate out of the FIFO must exactly match the frame rate into the FIFO. When the scaler is disabled, this constraint can be met by imposing the rule that the row time on the serial data stream must be greater than or equal to the row time at the pixel array. The row time on the serial data stream is calculated from the x_output_size and the data_format (8, 10, or 12 bits per pixel), and must include the time taken in the serial data stream for start of frame/row, end of row/frame and checksum symbols.

Caution **If this constraint is not met, the FIFO will either underrun or overrun. FIFO underrun or overrun is a fatal error condition that is signalled through the data path_status register (R0x306A).**

Changing Registers While Streaming

The following registers should only be reprogrammed while the sensor is in software standby:

- ccp_channel_identifier
- ccp_data_format
- ccp_signaling_mode
- vt_pix_clk_div
- vt_sys_clk_div
- pre_pll_clk_div
- pll_multiplier
- op_pix_clk_div
- op_sys_clk_div
- scale_m

Programming Restrictions when Using Global Reset

Interactions between the registers that control the global reset imposes some programming restrictions on the way in which they are used; these are discussed in "Global Reset" on page 59.

Control of the Signal Interface

This section describes the operation of the signal interface in all functional modes.

Power-Up State

The AR0832 sensor can provide up to two separate interfaces for pixel data: the MIPI serial interface and a parallel data interface.

MIPI Serial Pixel Data Interface

The serial pixel data interface uses the following output-only signal pairs:

- DATAP
- DATAN
- DATA2P
- DATA2N
- CLKP
- CLKN

The signal pairs use both single-ended and differential signaling, in accordance with the MIPI specification. The serial pixel data interface is enabled by default at power up and after reset.

The DATAP, DATAN, DATA2P, DATA2N, CLKP, and CLKN pads are set to the Ultra Low Power State (ULPS) if the SMIA serial disable bit is asserted (R0x301A-B[12]=1) or when the sensor is in the hardware standby or soft standby system states.

When the serial pixel data interface is used, the LINE_VALID, FRAME_VALID, PIXCLK and DOUT[9:0] signals (if present) can be left unconnected.

The ccp_data_format (R0x0112-3) register can be programmed to the following data format setting:

- 0x0A0A – Sensor supports RAW10 uncompressed data format. This mode is supported by discarding all but the upper 10 bits of a pixel value.
- 0x0808 – Sensor supports RAW8 uncompressed data format. This mode is supported by discarding all but the upper 8 bits of a pixel value.
- 0x0A08 – Sensor supports RAW8 data format in which an adaptive compression algorithm is used to perform 10-bit to 8-bit compression on the upper 10 bits of each pixel value

The serial_format register (R0x31AE–F) register controls which serial interface is in use when the serial interface is enabled (reset_register[12] = 0). The following serial formats are supported:

- 0x0101 – Sensor supports single-lane CCP2 operation
- 0x0201 – Sensor supports single-lane MIPI operation
- 0x0202 – Sensor supports dual-lane MIPI operation

Parallel Pixel Data Interface

The parallel pixel data interface uses these output-only signals:

- FV
- LV
- PIXCLK
- DOUT[9:0]

The parallel pixel data interface is disabled by default at power up and after reset. It can be enabled by programming R0x301A–B. Table 11 on page 31 shows the recommended settings.

When the parallel pixel data interface is in use, the serial data output signals (DATAP, DATAN, DATA2P, DATA2N, CLKP, and CLKN) can be left unconnected. Set reset_register[12] to disable the serializer while in parallel output mode.

To use the parallel interface, the VDD_TX pad must be tied to a 1.8V supply (VDD recommended). For MIPI sensor, the VDD_IO supply can be set at 1.8V (nominal).

Output Enable Control

When the parallel pixel data interface is enabled, its signals can be switched asynchronously between the driven and High-Z under pin or register control, as shown in Table 10. Selection of a pin to use for the OE_N function is described in "General Purpose Inputs" on page 35.

Table 10: Output Enable Control

OE_N Pin	Drive Signals R0x301A–B[6]	Description
Disabled	0	Interface High-Z
Disabled	1	Interface driven
1	0	Interface High-Z
X	1	Interface driven
0	X	Interface driven

Configuration of the Pixel Data Interface

Fields in R0x301A–B are used to configure the operation of the pixel data interface. The supported combinations are shown in Table 11.

Table 11: Configuration of the Pixel Data Interface

Serializer Disable R0x301 A–B[12]	Parallel Enable R0x301A–B[7]	Standby End-of-Frame R0x301A–B[4]	Description
0	0	1	Power up default. Serial pixel data interface and its clocks are enabled. Transitions to soft standby are synchronized to the end of frames on the serial pixel data interface.
1	1	0	Parallel pixel data interface, sensor core data output. Serial pixel data interface and its clocks disabled to save power. Transitions to soft standby are synchronized to the end of the current row readout on the parallel pixel data interface.

Table 11: Configuration of the Pixel Data Interface

Serializer Disable R0x301 A–B[12]	Parallel Enable R0x301A–B[7]	Standby End-of-Frame R0x301A–B[4]	Description
1	1	1	Parallel pixel data interface, sensor core data output. Serial pixel data interface and its clocks disabled to save power. Transitions to soft standby are synchronized to the end of frames in the parallel pixel data interface.

System States

The system states of the AR0832 are represented as a state diagram in Figure 16 on page 32 and described in subsequent sections. The effect of RESET_BAR on the system state and the configuration of the PLL in the different states are shown in Table 12 on page 33.

The sensor's operation is broken down into three separate states: hardware standby, software standby, and streaming. The transition between these states might take a certain amount of clock cycles as outlined in Table 12 on page 33.

Figure 16: AR0832 System States

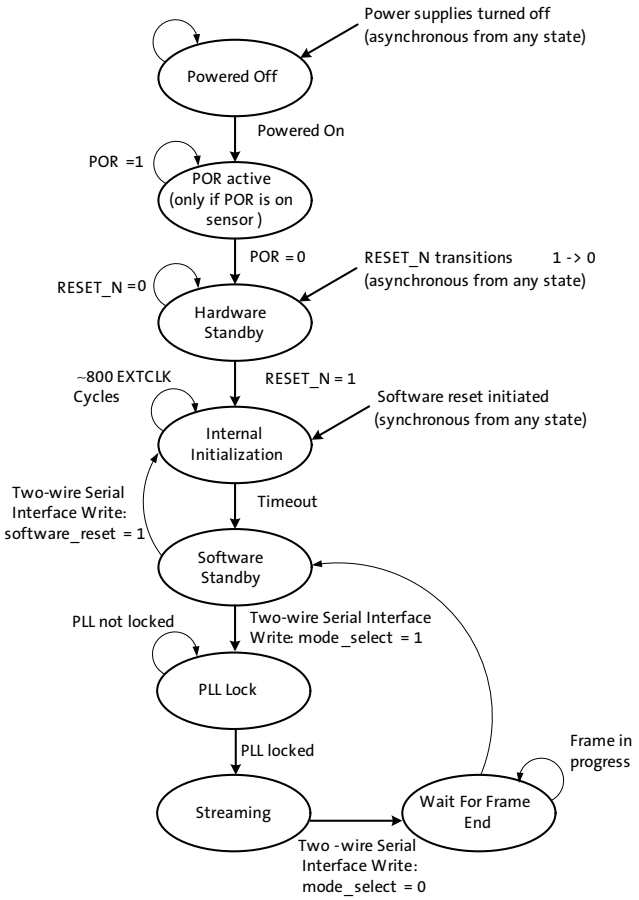


Table 12: RESET_BAR and PLL in System States

State	RESET_BAR	PLL
Powered off	x	VCO powered down
POR active	x	
Hardware standby	0	
Internal initialization	1	
Software standby		
PLL Lock		VCO powering up and locking, PLL output bypassed
Streaming		VCO running, PLL output active
Wait for frame end		

Power-On Reset Sequence

When power is applied to the AR0832, it enters a low-power hardware standby state. Exit from this state is controlled by the later of two events:

- The negation of the RESET_BAR input.
- The negation of the XSHUTDOWN input.
- A timeout of the internal power-on reset circuit.

It is possible to hold RESET_BAR permanently negated and rely upon the internal power-on reset circuit. The RESET_BAR signal is functionally equivalent to the SMIA-specified XSHUTDOWN signal.

While XSHUTDOWN is asserted the sensor is in its lowest-powered state with the internal digital power turned off.

When RESET_BAR is asserted it asynchronously resets the sensor, truncating any frame that is in progress.

While RESET_BAR is asserted (or the internal power-on reset circuit is active), the AR0832 is in its lowest-powered, powered-up state, the internal PLL is disabled, the CCP2 serializer is disabled, internal clocks are gated off, and the MIPI serial interface is in ULPS state.

While XSHUTDOWN is asserted, the sensor is in its lowest-powered state with the internal digital power turned off.

When the sensor leaves the hardware standby state, it performs an internal initialization sequence that takes a minimum of 2400 EXTCLK cycles. After this, it enters a low-power software standby state. While the initialization sequence is in progress, the AR0832 will not respond to read transactions on its two-wire serial interface. Therefore, a method to determine when the initialization sequence has completed is to poll a sensor register; for example, R0x0000. While the initialization sequence is in progress, the sensor will not respond to its device address and READs from the sensor will result in a NACK on the two-wire serial interface bus. When the sequence has completed, READs will return the operational value for the register (0x2800 if R0x0000 is read).

Also note that the two-wire serial interface can be disabled after the sensor is in the software standby state. This happens if there are OTPM records of the type 0x1n that are being auto-uploaded to registers. So the customer must be prepared for a NACK on the two-wire serial interface until the OTPM auto-read end bit is set.

When the sensor leaves software standby mode and enables the VCO, an internal delay will keep the PLL disconnected for up to 1ms so that the PLL can lock. The VCO lock time is 0.2ms (typical), 1ms (maximum).

Soft Reset Sequence

The AR0832 can be reset under software control by writing “1” to software_reset (R0x0103). A software reset asynchronously resets the sensor, truncating any frame that is in progress. The sensor starts the internal initialization sequence, while the PLL and analog blocks are turned off. At this point, the behavior is exactly the same as for the power-on reset sequence.

Signal State During Reset

Table 13 shows the state of the signal interface during hardware standby (RESET_BAR asserted) and the default state during software standby (after exit from hardware standby and before any registers within the sensor have been changed from their default power-up values).

Table 13: Signal State During Reset

Pad Name	Pad Type	Hardware Standby	Software Standby
EXTCLK	Input	Enabled. Must be driven to a valid logic level.	
RESET_BAR	Input	Enabled. Must be driven to a valid logic level.	
XSHUTDOWN	Input	Enabled. Must be driven to a valid logic level.	
LINE_VALID	Output	High-Z. Can be left disconnected/floating.	
FRAME_VALID	Output		
DOUT[9:0]	Output		
PIXCLK	Output		
SCLK	Input	Enabled. Must be pulled up or driven to a valid logic level.	
SDATA	I/O	Enabled as an input. Must be pulled up or driven to a valid logic level.	
FLASH	Output	High-Z.	Logic 0.
SHUTTER	Output	High-Z.	Logic 0.
DATA0_P	Output	CCP2: High Z MIPI: Ultra Low-Power State (ULPS), represented as an LP-00 state on the wire (both wires at 0V).	
DATA0_N	Output		
DATA1_P	Output		
DATA1_N	Output		
CLK_P	Output		
CLK_N	Output		
GPI[3:0]	Input	Powered down. Can be left disconnected/floating.	
TEST	Input	Must be driven to 1 for MIPI, 0 for CCP.	

General Purpose Inputs

The AR0832 provides four general purpose inputs, GPI[3:0]. After reset, the input pads associated with these signals are powered down by default, allowing the pads to be left disconnected/floating.

The general purpose inputs are enabled by setting reset_register[8] (R0x301A–B). Once enabled, all four inputs must be driven to valid logic levels by external signals. The state of the general purpose inputs can be read through gpi_status[3:0] (R0x3026).

In addition, each of the following functions can be associated with none, one, or more of the general purpose inputs so that the function can be directly controlled by a hardware input:

- Output enable (see “Output Enable Control” on page 31)
- Trigger (see the sections below)
- Standby functions

The gpi_status register is used to associate a function with a general purpose input.

Streaming/Standby Control

The AR0832 can be switched between its soft standby and streaming states under pin or register control, as shown in Table 14. Selection of a pin to use for the STANDBY function is described in "General Purpose Inputs" on page 35. The state diagram for transitions between soft standby and streaming states is shown in Figure 16 on page 32.

Table 14: Streaming/STANDBY

STANDBY	Streaming R0x301A–B[2]	Description
Disabled	0	Soft standby
Disabled	1	Streaming
X	0	Soft standby
0	1	Streaming
1	X	Soft standby

Trigger Control

When the global reset feature is in use, the trigger for the sequence can be initiated either under pin or register control, as shown in Table 15. Selection of a pin to use for the TRIGGER function is described in "General Purpose Inputs" on page 35.

Table 15: Trigger Control

Trigger	Global Trigger R0x3160–1[0]	Description
Disabled	0	Idle
Disabled	1	Trigger
0	0	Idle
X	1	Trigger
1	X	Trigger

Clocking

The AR0832 contains a PLL for timing generation and control. The PLL contains a prescaler to divide the input clock applied on EXTCLK, a VCO to multiply the prescaler output, and a set of dividers to generate the output clocks.

Both SMIA profile 0 and profile 1/2 clock schemes are supported. Sensor profile level represents an increasing level of data rate reduction for video applications, for example, viewfinder in full resolution. The clocking scheme can be selected by setting R0x306E–F[7] to 0 for profile 0 or to 1 for profile 1/ 2.

Figure 17 shows the different clocks and the names of the registers that contain or are used to control their values. Table 16 shows the default setting for each divider/multipler control register and the range of legal values for each divider/multiplier control register.

Figure 17: AR0832 Profile 1/2 Clocking Structure

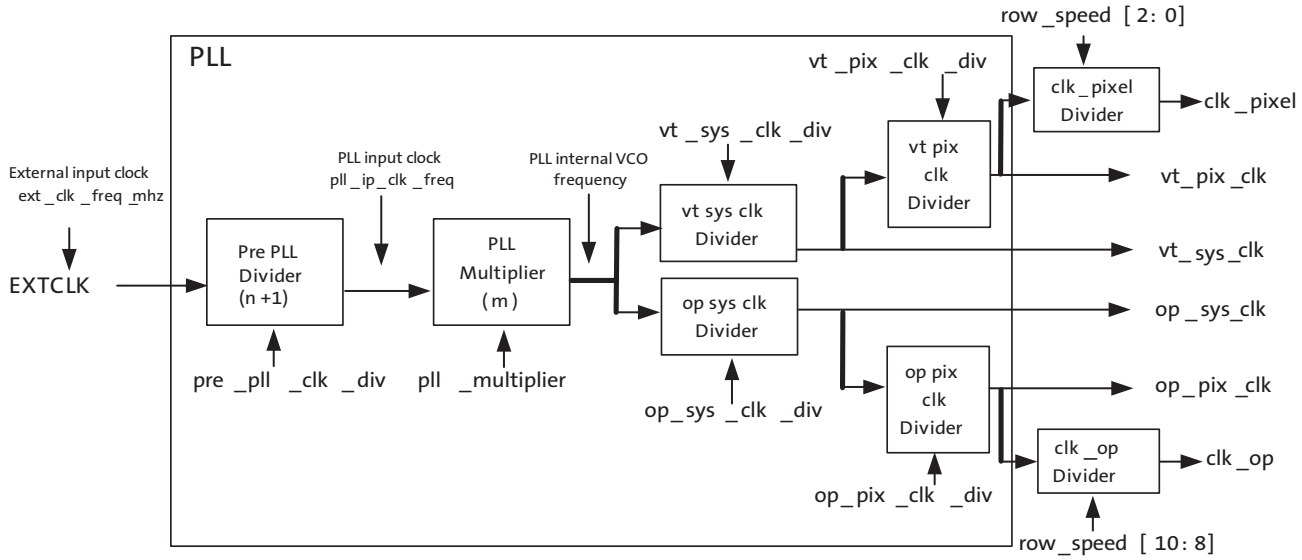


Table 16: Default Settings and Range of Values for Divider/Multiplier Registers

Name	Default	Range
ext_clk_freq_mhz	25	6 – 27 MHz
pll_ip_clk_freq_mhz		4 – 24 MHz
PLL internal VCO frequency (pll_op_clk_freq_mhz)		384 – 1000 MHz
vt_sys_clk_div	1	1, 2, 4, 6, 8, 10, 12, 14, 16
vt_pix_clk_div	5	4 – 16
row_speed	1	1, 2, 4
clk_pixel		4.8 – 200 MHz
op_sys_clk		24 – 800 MHz
clk_op		2.4 – 100 MHz
op_pix_clk_div	10	8, 10
op_sys_clk_div	1	1, 2, 4, 6, 8, 10, 12, 14, 16

Table 16: Default Settings and Range of Values for Divider/Multiplier Registers (continued)

Name	Default	Range
pll_multiplier	80	even values: 32 – 384 odd values: 17 – 191
pre_pll_clk_div	2	1 – 64 Note: A value of 1 must be used with even multiplier values

The parameter limit register space contains registers that declare the minimum and maximum allowable values for:

- The frequency allowable on each clock
- The divisors that are used to control each clock

These factors determine what are valid values, or combinations of valid values, for the divider/multiplier control registers:

- The minimum/maximum frequency limits for the associated clock must be met pll_ip_clk_freq must be in the range 4–24 MHz. Higher frequencies are preferred. PLL internal VCO frequency must be in the range 384–1000 MHz.
- The minimum/maximum value for the divider/multiplier must be met. Range for m: 32 –192. (In addition odd values between 17–31 and even values between 194–384 are accepted.)
- clk_op must never run faster than the clk_pixel to ensure that the output data stream is contiguous.
- Given the maximum programmed line length, the minimum blanking time, the maximum image width, the available PLL divisor/multiplier values, and the requirement that the output line time (including the necessary blanking) must be output in a time equal to or less than the time defined by line_length_pck.

Although the PLL VCO input frequency range is advertised as 6–27 MHz, superior performance is obtained by keeping the VCO input frequency as high as possible.

The usage of the output clocks is shown below:

- clk_pixel (vt_pix_clk / row_speed[2:0]) is used by the sensor core to readout and control the timing of the pixel array. The sensor core produces one 10-bit pixel each vt_pix_clk period. The line length (line_length_pck) and fine integration time (fine_integration_time) are controlled in increments of the clk_pixel period.
- clk_op (op_pix_clk / row_speed[10:8]) is used to load parallel pixel data from the output FIFO (see Figure 48 on page 75) to the serializer. The output FIFO generates one pixel each op_pix_clk period. The pixel is either 8-bit or 10-bit depending upon the output data format, controlled by R0x0112–3 (ccp_data_format).
- op_sys_clk is used to generate the serial data stream on the output. The relationship between this clock frequency and the op_pix_clk frequency is dependent upon the output data format.

In Profile 1/2, the output clock frequencies can be calculated as:

$$clk\_pix\_freq\_mhz = \frac{ext\_clk\_freq\_mhz * pll\_multiplier}{pre\_pll\_clk\_div*vt\_sys\_clk\_div *vt\_pix\_clk\_div*row\_speed[2:0]} \quad \text{(EQ 1)}$$

$$clk\_op\_freq\_mhz = \frac{ext\_clk\_freq\_mhz * pll\_multiplier}{pre\_pll\_clk\_div*op\_sys\_clk\_div*op\_pix\_clk\_div*row\_speed[10:8]} \quad \text{(EQ 2)}$$

$$op\_sys\_clk\_freq\_mhz = \frac{ext\_clk\_freq\_mhz * pll\_multiplier}{pre\_pll\_clk\_div * op\_sys\_clk\_div} \quad \text{(EQ 3)}$$

Note: In Profile 0, RAW10 data format is required. As a result, op_pix_clk_div must be set to 10. Also, due to the inherent design of the AR0832 sensor, vt_pix_clk_div must be set to 5 for profile 0 mode.)

PLL Clocking

The PLL divisors should be programmed while the AR0832 is in the software standby state. After programming the divisors, it is necessary to wait for the VCO lock time before enabling the PLL. The PLL is enabled by entering the streaming state.

An external timer will need to delay the entrance of the streaming mode by 1 millisecond so that the PLL can lock.

The effect of programming the PLL divisors while the AR0832 is in the streaming state is undefined.

Influence of ccp_data_format

R0x0112–3 (ccp_data_format) controls whether the pixel data interface will generate 10 or 8 bits per pixel.

When the pixel data interface is generating 10 bits per pixel, op_pix_clk_div must be programmed with the value 10.

Influence of ccp2_signalling_mode

R0x0111 (ccp2_signalling_mode) controls whether the serial pixel data interface uses data/strobe signalling or data/clock signalling.

When data/clock signalling is selected, the pll_multiplier supports both odd and even values.

When data/strobe signalling is selected, the pll_multiplier only supports even values; the least significant bit of the programmed value is ignored and treated as “0.”

This behavior is a result of the implementation of the CCP serializer and the PLL. When the serializer is using data/strobe signalling, it uses both edges of the op_sys_clk, and therefore that clock runs at one half of the bit rate. All of the programmed divisors are set up to make this behavior invisible. For example, when the divisors are programmed to generate a PLL output of 640 MHz, the actual PLL output is 320MHz, but both edges are used.

When the serializer is using data/clock signalling, it uses a single edge on the op_sys_clk, and therefore that clock runs at the bit rate.

To disguise this behavior from the programmer, the actual PLL multiplier is right-shifted by one bit relative to the programmed value when ccp2_signalling_mode selects data/strobe signalling.

Clock Control

The AR0832 uses an aggressive clock-gating methodology to reduce power consumption. The clocked logic is divided into a number of separate domains, each of which is only clocked when required.

When the AR0832 enters a low-power state, almost all of the internal clocks are stopped. The only exception is that a small amount of logic is clocked so that the two-wire serial interface continues to respond to read and write requests.

Features

Shading Correction (SC)

Lenses tend to produce images whose brightness is significantly attenuated near the edges. There are also other factors causing fixed pattern signal gradients in images captured by image sensors. The cumulative result of all these factors is known as image shading. The AR0832 has an embedded shading correction module that can be programmed to counter the shading effects on each individual Red, GreenB, GreenR, and Blue color signal.

The Correction Function

Color-dependent solutions are calibrated using the sensor, lens system and an image of an evenly illuminated, featureless gray calibration field. From the resulting image, register values for the color correction function (coefficients) can be derived.

The correction functions can then be applied to each pixel value to equalize the response across the image as follows:

$$Pcorrected(row, col) = Psensor(row,col) * f(row,col) \quad \text{(EQ 4)}$$

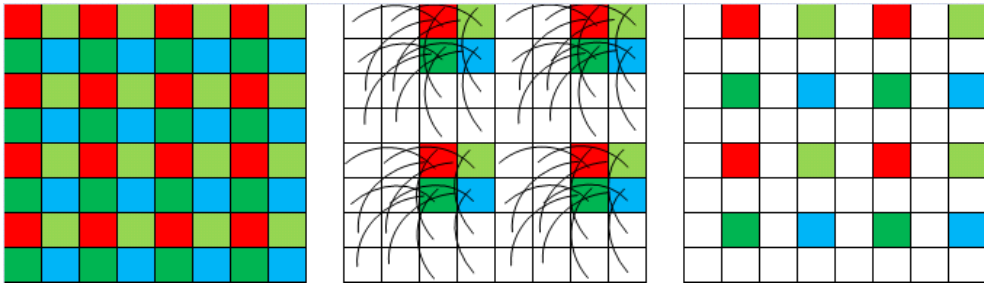
where *P* are the pixel values and *f* is the color dependent correction functions for each color channel.

Each function includes a set of color-dependent coefficients defined by registers R0x3600–3726. The function's origin is the center point of the function used in the calculation of the coefficients. Using an origin near the central point of symmetry of the sensor response provides the best results. The center point of the function is determined by ORIGIN_C (R0x3782) and ORIGIN_R (R0x3784) and can be used to counter an offset in the system lens from the center of the sensor array.

Bayer Resampler

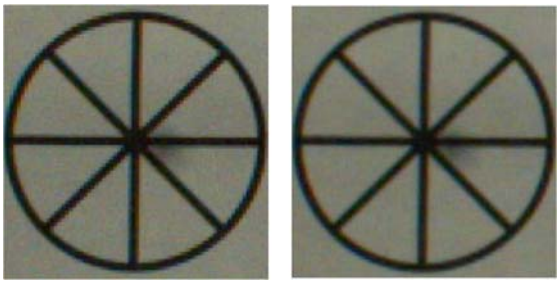
The imaging artifacts found from a 2x2 binning or summing will show image artifacts from aliasing. These can be corrected by resampling the sampled pixels in order to filter these artifacts. Figure 18 shows the pixel location resulting from 2x2 summing or binning located in the middle and the resulting pixel locations after the Bayer re-sampling function has been applied.

Figure 18: Bayer Resampling



The improvements from using the Bayer resampling feature can be seen in Figure 19. In this example, image edges seen on a diagonal have smoother edges when the Bayer re-sampling feature is applied. This feature is only designed to be used with modes configured with 2x2 binning or summing. The feature will not remove aliasing artifacts that are caused skipping pixels.

Figure 19: Results of Resampling



To enable the Bayer resampling feature:

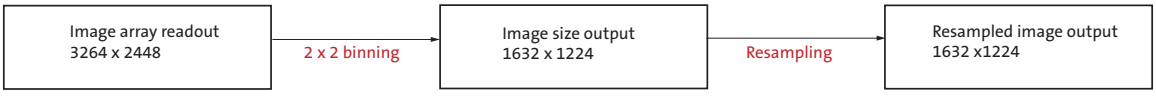
1. Set bit field = 0x306E, 0x0030, 0x3// Enable the on-chip scalar.

To disable the Bayer resampling feature:

1. Set bit field = 0x306E, 0x0030, 0x0// Disable the on-chip scalar.

Note: The image readout (rows and columns) has to have two extra rows and two extra columns when using the resample feature.

Figure 20: Illustration of Resampling Operation



One-Time Programmable Memory (OTPM)

The AR0832 features 11.5Kb of one-time programmable memory (OTPM) for storing shading correction coefficients, individual module, and sensor specific information. Roughly 5Kb (100regs x 16 bits x 3 sets = 4800 bits) are available to store three sets of illuminate dependent shading coefficients (R0x3600–27, R0x3640–67, R0x3680–A7, R0x36C0–E7, and R0x3700–27). The user may program which set to be used. Additional bits are used by the error detection and correction scheme. OTPM can be accessed through two-wire serial interface. The AR0832 uses the auto mode for fast OTPM programming and read operations.

During the programming process, a dedicated pin for high voltage needs to be provided to perform the anti-fusing operation. This voltage (VPP) would need to be 6–7V. The completion of the programming process will be communicated by a register through the two-wire serial interface.

Because this programming pin needs to sustain a higher voltage than other input/output pins, having a dedicated high voltage pin (VPP) minimizes the design risk. If the module manufacturing process can probe the sensor at the die or PCB level (that is, supply all the power rails, clocks, two-wire serial interface signals), then this dedicated high voltage pin does not need to be assigned to the module connector pinout. However, if the VPP pin needs to be bonded out as a pin on the module, the trace for VPP needs to carry a maximum of 1mA – for programming only. This pin should be left floating once the module is integrated to a design. If the VPP pin does not need to be bonded out as a pin on the module, it should be left floating inside the module.

The programming of the OTPM requires the sensor to be fully powered and remain in software standby with its clock input applied. The information will be programmed through the use of the two-wire serial interface, and once the data is written to an internal register, the programming host machine will apply a high voltage to the programming pin, and send a program command to initiate the anti-fusing process. After the sensor has finished programming the OTPM, a status bit will be set to indicate the end of the programming cycle, and the host machine can poll the setting of the status bit through the two-wire serial interface. Only one programming cycle for the 16-bit word can be performed.

Reading the OTPM data requires the sensor to be fully powered and operational with its clock input applied. The data can be read through a register from the two-wire serial interface.

Programming and Verifying the OTPM

The procedure for programming and verifying the AR0832 OTPM follows:

1. Apply power to all the power rails of the sensor (VDD, VDD_TX, VDD_IO, VAA, VAA_PIX, and VDD_PLL). VAA must be set to 2.8V during the programming process. VPP must be initially being floating. All other supplies must be at their nominal voltage.
2. Provide a 12-MHz EXTCLK clock input.
3. Set R0x301A = 0x10D8, to put sensor in the soft standby mode.
4. Set R0x3064[9] =1.
5. Set R0x3054[8]=1
6. Write data into all the OTPM data registers: R0x3800–R0x39FE.

7. Set OTPM start address register R0x3050[15:8] = 0. The OTPM start address is initially set to "0" while programming the array with the first batch of data (for the first time). When programming the second batch of data, the start address should be set to 128 (considering that all the previous 0–127 locations are already written by the data registers 0–255), otherwise it should be set accordingly.
8. Set R0x3054[9] = 0 to ensure that the error checking and correction is enabled.
9. The length Register (R0x304C [7:0]) should be set accordingly depending on the number of OTPM data registers that are filled in.
10. Set R0x3052 = 0x2504 (OTPM_CONFIG)
11. Ramp up VPP to 6.5V.
12. Set the otpm_control_auto_wr_start bit in the otpm_manual_control register R0x304A[0] = 1, to initiate the auto program sequence. The sensor will now program the data into the OTPM starting with the location specified by the start address.
13. Poll OTPM_Control_Auto_WR_end (R0x304A [1]) to determine when the sensor is finished programming the word.
14. Verify that the otpm_control_auto_wr_success(0x304A[2]) bit is set.
15. Repeat steps from 12 through 17 two more times.
16. Remove the high voltage (VPP) and float VPP pin.
17. Power down the sensor.

Reading the OTPM

1. 1. Apply power to all the power rails of the sensor (VDD, VDD_TX,VDD_IO, VAA, VAA_PIX, and VDD_PLL) at their nominal voltage.
2. Set EXTCLK to normal operating frequency.
3. Perform the proper reset sequence to the sensor.
4. Set OTPM_CONFIG register R0x3052 = 0x2704.
5. Set R0x3054[8] = 1.
6. Program R0x3050[15:8] with the appropriate value to specify the start address.
7. Program R0x304C [7:0] with the appropriate value to specify to specify the length (number of data registers to be read back, starting from the specified start address).
8. Initiate the auto read sequence by setting the otpm_control_auto_read_start bit (R0x304A[4]).
9. Poll the otpm_control_auto_rd_end bit (R0x304A[5]) to determine when the sensor is finished reading the word(s). When this bit becomes "1", the otpm_control_suto_rd_success bit (R0x304A[6]) will indicate whether the memory was read successfully or not.
10. Data can now be read back from the otpm_data registers (R0x3800–R0x39FE).

Image Acquisition Modes

The AR0832 supports two image acquisition modes:

1. Electronic rolling shutter (ERS) mode.
   This is the normal mode of operation. When the AR0832 is streaming, it generates frames at a fixed rate, and each frame is integrated (exposed) using the ERS. When the ERS is in use, timing and control logic within the sensor sequences through the rows of the array, resetting and then reading each row in turn. In the time interval between resetting a row and subsequently reading that row, the pixels in the row integrate incident light. The integration (exposure) time is controlled by varying the time between row reset and row readout. For each row in a frame, the time between row reset and row readout is fixed, leading to a uniform integration time across the frame. When the integration time is changed (by using the two-wire serial interface to change register settings), the timing and control logic controls the transition from old to new integration time in such a way that the stream of output frames from the AR0832 switches cleanly from the old integration time to the new while only generating frames with uniform integration. See “Changes to Integration Time” on page 24.
2. Global reset mode.
   This mode can be used to acquire a single image at the current resolution. In this mode, the end point of the pixel integration time is controlled by an external electromechanical shutter, and the AR0832 provides control signals to interface to that shutter. The operation of this mode is described in detail in “Global Reset” on page 59.

The benefit for the use of an external electromechanical shutter is that it eliminates the visual artifacts associated with ERS operation. Visual artifacts arise in ERS operation, particularly at low frame rates, because an ERS image effectively integrates each row of the pixel array at a different point in time.

Window Control

The sequencing of the pixel array is controlled by the x_addr_start, y_addr_start, x_addr_end, and y_addr_end registers. For both parallel and serial interfaces, the output image size is controlled by the x_output_size and y_output_size registers.

Pixel Border

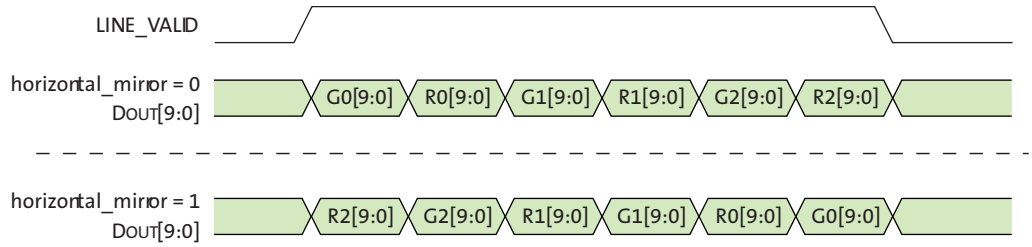
The default settings of the sensor provide a 3264H x 2448V image. A border of up to 8 pixels (4 in binning) on each edge can be enabled by reprogramming the x_addr_start, y_addr_start, x_addr_end, y_addr_end, x_output_size, and y_output_size registers accordingly.

Readout Modes

Horizontal Mirror

When the horizontal_mirror bit is set in the image_orientation register, the order of pixel readout within a row is reversed, so that readout starts from x_addr_end and ends at x_addr_start. Figure 21 on page 45 shows a sequence of 6 pixels being read out with horizontal_mirror = 0 and horizontal_mirror = 1. Changing horizontal_mirror causes the Bayer order of the output image to change; the new Bayer order is reflected in the value of the pixel_order register.

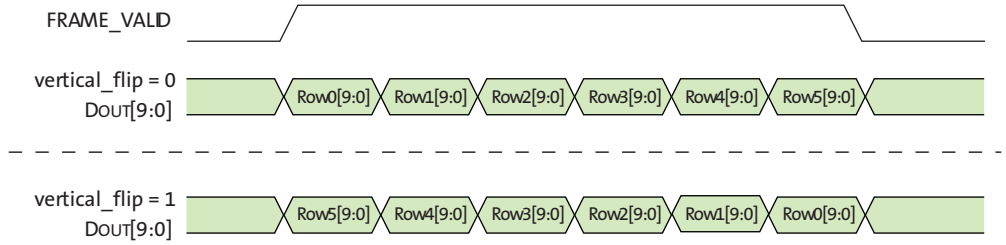
Figure 21: Effect of horizontal_mirror on Readout Order



Vertical Flip

When the vertical_flip bit is set in the image_orientation register, the order in which pixel rows are read out is reversed, so that row readout starts from y_addr_end and ends at y_addr_start. Figure 22 shows a sequence of 6 rows being read out with vertical_flip = 0 and vertical_flip = 1. Changing vertical_flip causes the Bayer order of the output image to change; the new Bayer order is reflected in the value of the pixel_order register.

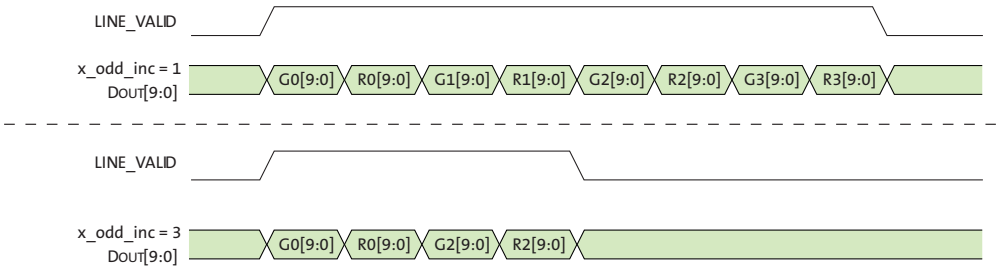
Figure 22: Effect of vertical_flip on Readout Order



Subsampling

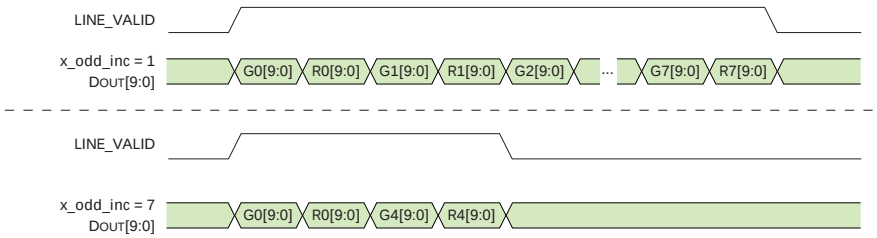
The AR0832 supports subsampling. Subsampling reduces the amount of data processed by the analog signal chain in the AR0832 thereby allowing the frame rate to be increased. Subsampling is enabled by setting x_odd_inc and/or y_odd_inc. Values of 1, 3, and 7 can be supported. Setting both of these variables to 3 reduces the amount of row and column data processed and is equivalent to the 2 x 2 skipping readout mode provided by the AR0832. Setting x_odd_inc = 3 and y_odd_inc = 3 results in a quarter reduction in output image size. Figure 23 on page 46 shows a sequence of 8 columns being read out with x_odd_inc = 3 and y_odd_inc = 1.

Figure 23: Effect of x_odd_inc = 3 on Readout Sequence



A 1/16 reduction in resolution is achieved by setting both x_odd_inc and y_odd_inc to 7. This is equivalent to 4 x 4 skipping readout mode provided by the AR0832. Figure 24 shows a sequence of 16 columns being read out with x_odd_inc = 7 and y_odd_inc = 1.

Figure 24: Effect of x_odd_inc = 7 on Readout Sequence



The effect of the different subsampling settings on the pixel array readout is shown in Figure 25 through Figure 27 on page 47.

Figure 25: Pixel Readout (No Subsampling)

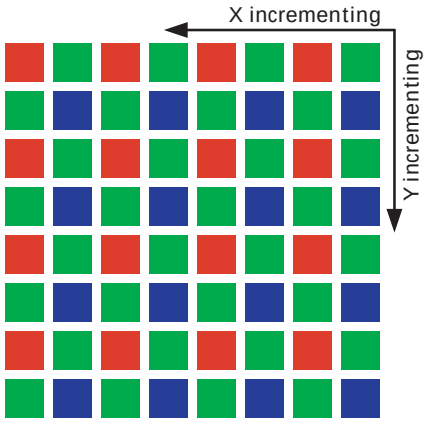


Figure 26: Pixel Readout (x_odd_inc = 3, y_odd_inc = 3)

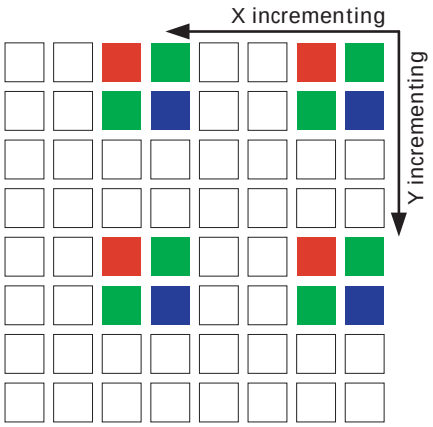
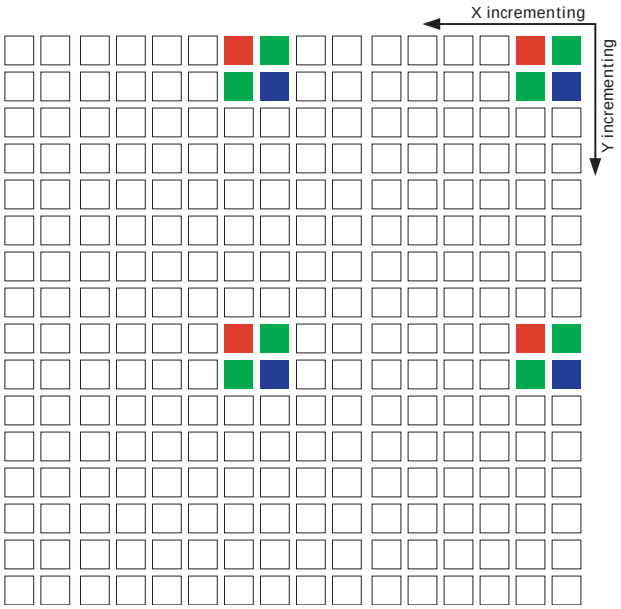


Figure 27: Pixel Readout (x_odd_inc = 7, y_odd_inc = 7)



Programming Restrictions when Subsampling

When subsampling is enabled as a viewfinder mode and the sensor is switched back and forth between full resolution and subsampling, Aptina recommends that line_length_pck be kept constant between the two modes. This allows the same integration times to be used in each mode.

When subsampling is enabled, it may be necessary to adjust the x_addr_end, x_addr_start and y_addr_end settings: the values for these registers are required to correspond with rows/columns that form part of the subsampling sequence. The adjustment should be made in accordance with these rules:

x_skip_factor = (x_odd_inc + 1) / 2
y_skip_factor = (y_odd_inc + 1) / 2

- x_addr_start should be a multiple of x_skip_factor * 4
- (x_addr_end - x_addr_start + x_odd_inc) should be a multiple of x_skip_factor * 4
- (y_addr_end - y_addr_start + y_odd_inc) should be a multiple of y_skip_factor * 4

The number of columns/rows read out with subsampling can be found from the equation below:

- columns/rows = (addr_end - addr_start + odd_inc) / skip_factor

Table 17 shows the row or column address sequencing for normal and subsampled readout. In the 2X skip case, there are two possible subsampling sequences (because the subsampling sequence only reads half of the pixels) depending upon the alignment of the start address. Similarly, there will be four possible subsampling sequences in the 4X skip case (though only the first two are shown in Table 17).

Table 17: Row Address Sequencing During Subsampling

odd_inc = 1 (Normal)	odd_inc = 3 (2X Skip)	odd_inc = 7 (4X Skip)
start = 0	start = 0	start = 0
0	0	0
1	1	1
2		
3		
4	4	
5	5	
6		
7		
8	8	8
9	9	9
10		
11		
12	12	
13	13	
14		
15		

Binning

The AR0832 supports 2 x 1 (column binning, also called x-binning) and 2 x 2 analog binning (row/column binning, also called xy-binning). Binning has many of the same characteristics as skipping, but because it gathers image data from all pixels in the active window (rather than a subset of them), it achieves superior image quality and avoids the aliasing artifacts that can be a characteristic side effect of skipping.

Binning is enabled by selecting the appropriate subsampling settings (odd_inc = 3 and y_odd_inc = 1 for x-binning, or the combination of x_odd_inc = 3 and y_odd_inc = 3 for xy-binning) and setting the appropriate binning bit in read_mode (R0x3040[10] = 1 for XY_Bin_Enable or R0x3040[11] = 1 for only X_Bin_Enable). As with skipping, x_addr_end and y_addr_end may require adjustment when binning is enabled. It is the first of the two columns/rows binned together that should be the end column/row in binning, so the requirements to the end address are exactly the same as in skipping mode. The effect of the different binning is shown in Figure 28 and Figure 29.

Binning can also be enabled when the 4X subsampling mode is enabled (x_odd_inc = 7 and y_odd_inc = 1 for x-binning, x_odd_inc = 7 and y_odd_inc = 7 for 4X xy-binning). In this mode, however, not all pixels will be used so this is not a 4X binning implementation. An implementation providing a combination of skip2 and bin2 is used to achieve 4X subsampling with better image quality. The effect of this subsampling mode is shown in Figure 30 on page 50.

Figure 28: Pixel Readout (x_odd_inc = 3, y_odd_inc = 1, x_bin = 1)

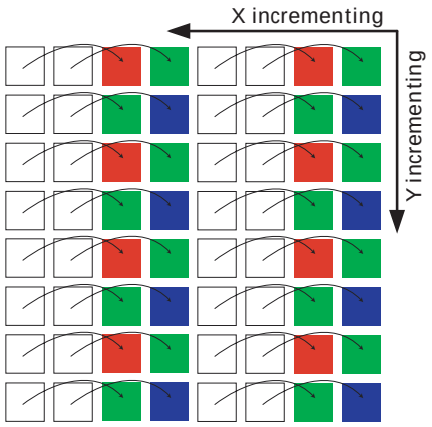


Figure 29: Pixel Readout (x_odd_inc = 3, y_odd_inc = 3, xy_bin = 1)

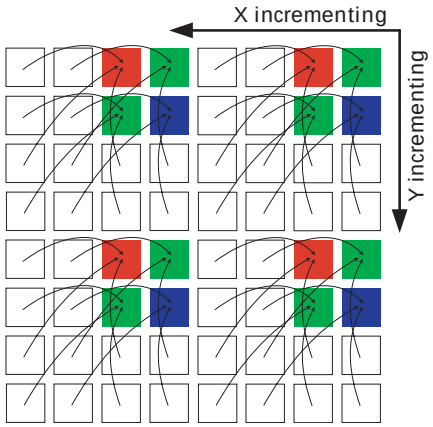
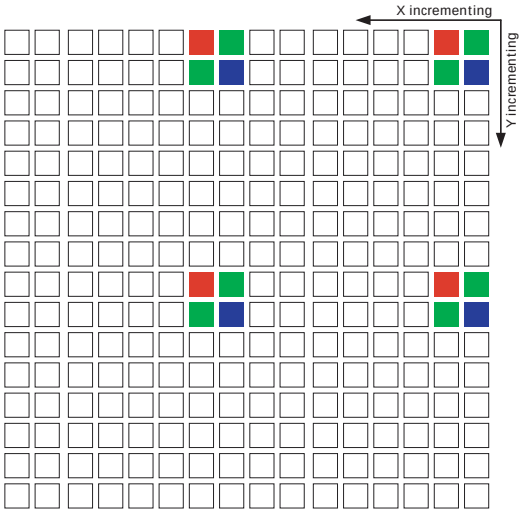


Figure 30: Pixel Readout (x_odd_inc = 7, y_odd_inc = 7, xy_bin = 1)



Binning address sequencing is a bit more complicated than during subsampling only, because of the implementation of the binning itself.

For a given column *n*, there is only one other column, n_bin, that can be binned with, because of physical limitations in the column readout circuitry. The possible address sequences are shown in Table 18.

Table 18: Column Address Sequencing During Binning

odd_inc = 1 (Normal)	odd_inc = 3 (2X Bin)	odd_inc = 7 (2X Skip + 2XBin)
x_addr_start = 0	x_addr_start = 0	x_addr_start = 0
0	0/2	0/4
1	1/3	1/5
2		
3		
4	4/6	
5	5/7	

Table 18: Column Address Sequencing During Binning (continued)

odd_inc = 1 (Normal)	odd_inc = 3 (2X Bin)	odd_inc = 7 (2X Skip + 2XBin)
x_addr_start = 0	x_addr_start = 0	x_addr_start = 0
6		
7		
8	8/10	8/10
9	9/11	9/13
10		
11		
12	12/14	
13	13/15	
14		
15		

There are no physical limitations on what can be binned together in the row direction. A given row *n* will always be binned with row n+2 in 2X subsampling mode and with row n+4 in 4X subsampling mode. Therefore, which rows get binned together depends upon the alignment of y_addr_start. The possible sequences are shown in Table 19 on page 51.

Table 19: Row Address Sequencing During Binning

odd_inc = 1 (Normal)	odd_inc = 3 (2X Bin)	odd_inc = 7 (2X Skip + 2X Bin)
x_addr_start = 0	x_addr_start = 0	x_addr_start = 0
0	0/2	0/4
1	1/3	1/5
2		
3		
4	4/6	
5	5/7	
6		
7		
8	8/10	8/12
9	9/11	9/13
10		
11		
12	12/14	
13	13/15	
14		
15		

Programming Restrictions When Binning

Binning requires different sequencing of the pixel array and imposes different timing limits on the operation of the sensor. In particular, xy-binning requires two read operations from the pixel array for each line of output data, which has the effect of increasing the minimum line blanking time. The SMIA specification cannot accommodate this variation because its parameter limit registers are defined as being static.

As a result, when xy-binning is enabled, some of the programming limits declared in the parameter limit registers are no longer valid. In addition, the default values for some of the manufacturer-specific registers need to be reprogrammed. See section "Minimum Frame Time" on page 55, section "Minimum Row Time" on page 55, and section "Fine Integration Time Limits" on page 56.

Table 20: Readout Modes

Readout Modes	x_odd_inc, y_odd_inc	xy_bin
2x skip	3	0
2x bin	3	1
4x skip	7	0
2x skip + 2x bin	7	1

Summing Mode

Summing can be enabled with binning. The register x_odd_inc must be set to 3 to enable summing. Unlike binning mode where the values of adjacent same color pixels are averaged together, summing adds the pixel values together resulting in better sensor sensitivity. Summing is supposed to provide two times the sensitivity compared to the binning only mode.

Figure 31: Pixel Binning and Summing

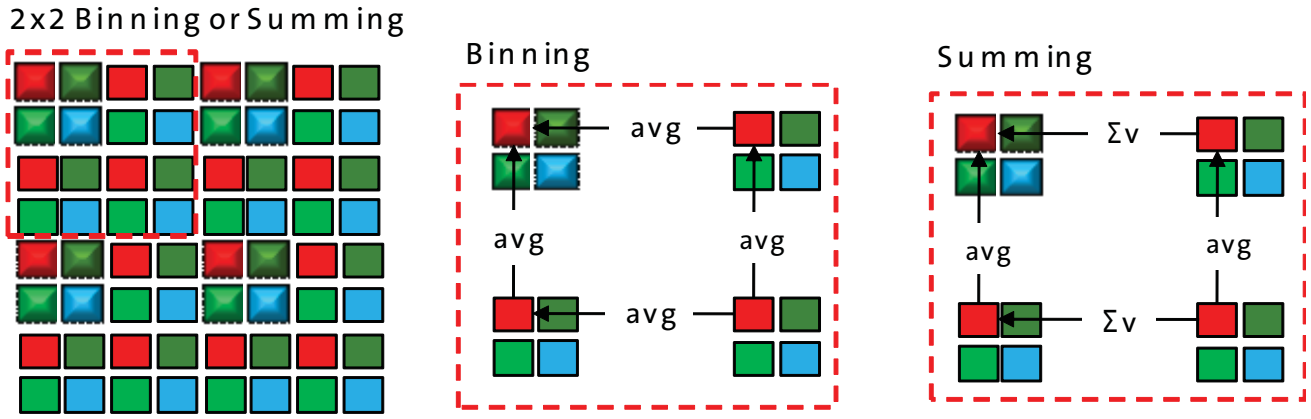


Table 21: Readout Modes

Readout Modes	x_odd_inc, y_odd_inc	xy_bin	x_bin
2x skip	3	0	0
2x bin	3	1	0
4x skip	7	0	0
2x skip + 2x bin	7	1	0
4y skip; 2xskip + 2x bin	7	0	1

Scaler

Scaling is a “zoom out” operation to reduce the size of the output image while covering the same extent as the original image. Each scaled output pixel is calculated by taking a weighted average of a group input pixels which is composed of neighboring pixels. The input and output of the scaler is in Bayer format.

When compared to skipping, scaling is advantageous because it uses all pixel values to calculate the output image which helps avoid aliasing. Also, it is also more convenient than binning because the scale factor varies smoothly and the user is not limited to certain ratios of size reduction.

The AR0832 sensor is capable of horizontal scaling and full (horizontal and vertical) scaling.

The scaling factor, programmable in 1/16th steps, is used for horizontal and vertical scalers.

The scale factor is determined by:

- n, which is fixed at 16
- m, which is adjustable with register R0x0404
- Legal values for m are 16 through 256, giving the user the ability to scale from 1:1 (m=16) to 1:8 (m=256).

Frame Rate Control

The formulas for calculating the frame rate of the AR0832 are shown below.

The line length is programmed directly in pixel clock periods through register line_length_pck. For a specific window size, the minimum line length can be found from in Equation 5:

$$minimum\ line\_length\_pck = \left(\frac{x\_addr\_end - x\_addr\_start + 1}{subsampling\ factor} + min\_line\_blanking\_pck\right) \quad \text{(EQ 5)}$$

Note that line_length_pck also needs to meet the minimum line length requirement set in register min_line_length_pck. The row time can either be limited by the time it takes to sample and reset the pixel array for each row, or by the time it takes to sample and read out a row. Values for min_line_blanking_pck are provided in “Minimum Row Time” on page 55.

The frame length is programmed directly in number of lines in the register frame_line_length. For a specific window size, the minimum frame length can be found in Equation 6:

$$minimum\ frame\_length\_lines = \left(\frac{y\_addr\_end - y\_addr\_start + 1}{subsampling\ factor} + min\_frame\_blanking\_lines\right) \quad \text{(EQ 6)}$$

The frame rate can be calculated from these variables and the pixel clock speed as shown in Equation 7:

$$frame\ rate = \frac{vt\_pixel\_clock\_mhz \times 1 \times 10^6}{line\_length\_pck\_x\ frame\_length\_lines} \quad \text{(EQ 7)}$$

If coarse_integration_time is set larger than frame_length_lines the frame size will be expanded to coarse_integration_time + 1.

Minimum Row Time

The minimum row time and blanking values with default register settings are shown in Table 22.

Table 22: Minimum Row Time and Blanking Numbers

	No Row Binning			Row Binning		
row_speed[2:0]	1	2	4	1	2	4
min_line_blanking_pck	0x0588	0x0364	0x0252	0x09B4	0x0566	0x0340
min_line_length_pck	0x0750	0x04A8	0x0398	0x0E80	0x0740	0x0480

In addition, enough time must be given to the output FIFO so it can output all data at the set frequency within one row time.

There are therefore three checks that must all be met when programming line_length_pck:

- line_length_pck ≥ min_line_length_pck in Table 22.
- line_length_pck ≥ (x_addr_end - x_addr_start + x_odd_inc)/((1+x_odd_inc)/2) + min_line_blanking_pck in Table 22.
- The row time must allow the FIFO to output all data during each row. That is, line_length_pck ≥ (x_output_size * 2 + 0x005E) * “vt_pix_clk period” / “op_pix_clk period”

Minimum Frame Time

The minimum number of rows in the image is 2, so min_frame_length_lines will always equal (min_frame_blanking_lines + 2).

Table 23: Minimum Frame Time and Blanking Numbers

	No Row Binning	Row Binning
min_frame_blanking_lines	0x008F	0x008F
min_frame_length_lines	0x0A1F	0x0557

Integration Time

The integration (exposure) time of the AR0832 is controlled by the fine_integration_time and coarse_integration_time registers.

The limits for the fine integration time are defined by:

$$fine\_integration\_time\_min \leq fine\_integration\_time \leq (line\_length\_pck - fine\_integration\_time\_max\_margin) \quad \textit{(EQ 8)}$$

The limits for the coarse integration time are defined by:

$$coarse\_integration\_time\_min \leq coarse\_integration\_time \quad \textit{(EQ 9)}$$

The actual integration time is given by:

$$integration\_time = \frac{((coarse\_integration\_time * line\_length\_pck) + fine\_integration\_time)}{(vt\_pix\_clk\_freq\_mhz * 10^6)} \quad \text{(EQ 10)}$$

It is required that:

$$coarse\_integration\_time \leq (frame\_length\_lines - coarse\_integration\_time\_max\_margin) \quad \text{(EQ 11)}$$

If this limit is broken, the frame time will automatically be extended to (coarse_integration_time + coarse_integartion_time_max_margin) to accommodate the larger integration time.

In binning mode, frame_length_lines should be set larger than coarse_integration_time by at least 3 to avoid column imbalance artifact.

Fine Integration Time Limits

The limits for the fine_integration_time can be found from fine_integration_time_min and fine_integration_time_max_margin. Values for different mode combinations are shown in Table 24.

Table 24: fine_integration_time Limits

	No Row Binning			Row Binning		
row_speed[2:0]	1	2	4	1	2	4
fine_integration_time_min	0x03F6	0x01FC	0x00FE	0x0846	0x0424	0x0212
fine_integration_time_max_margin	0x0356	0x01AC	0x00D6	0x0636	0x031C	0x018E

fine_correction

For the fine_integration_time limits, the fine_correction constant will change with the pixel clock speed and binning mode. It is necessary to fine_correction (R0x3010) when binning is enabled or the pixel clock divider (row_speed[2:0]) is used. The corresponding fine_correction values are shown in Table 25.

Table 25: fine_correction Values

	No Row Binning			Row Binning		
row_speed[2:0]	1	2	4	1	2	4
fine_correction	0x0078	0x0036	0x0015	0x0130	0x0092	0x0043

Flash Timing Control

The AR0832 supports both xenon and LED flash timing through the FLASH output signal. The timing of the FLASH signal with the default settings is shown in Figure 32 (xenon) and Figure 33 on page 58 (LED). The flash and flash_count registers allow the timing of the flash to be changed. The flash can be programmed to fire only once, delayed by a few frames when asserted, and (for xenon flash) the flash duration can be programmed.

Enabling the LED flash will cause one bad frame, where several of the rows only have the flash on for part of their integration time. This can be avoided either by first enabling mask bad frames (R0x301A[9] = 1) before the enabling the flash or by forcing a restart (R0x301A[1] = 1) immediately after enabling the flash; the first bad frame will then be masked out, as shown in Figure 33. Read-only bit flash[14] is set during frames that are correctly integrated; the state of this bit is shown in Figures 32 and Figure 33.

Figure 32: Xenon Flash Enabled

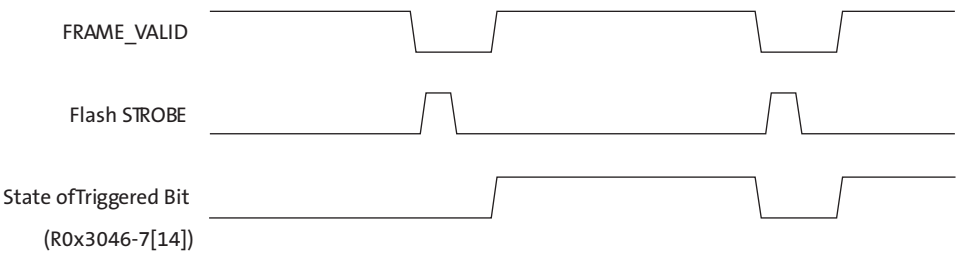
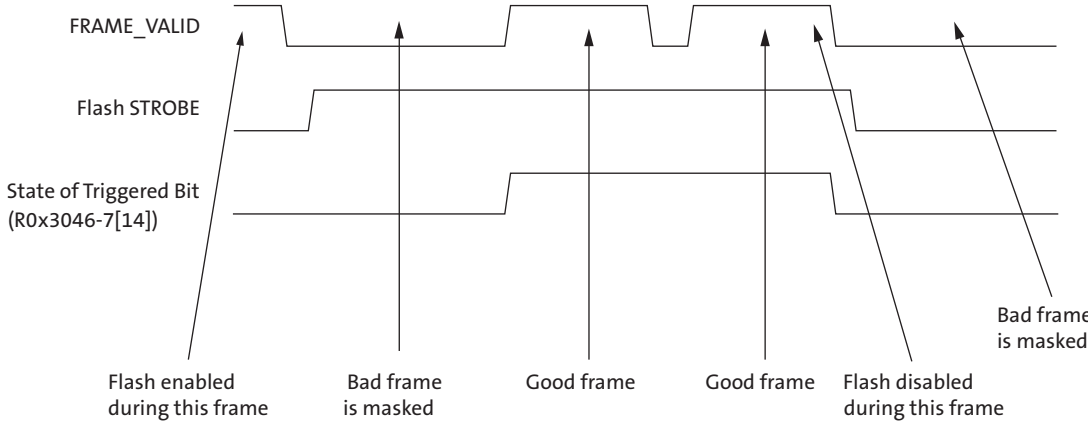


Figure 33: LED Flash Enabled



Note: An option to invert the flash output signal through R0x3046[7] is also available.

Global Reset

Global reset mode allows the integration time of the AR0832 to be controlled by an external electromechanical shutter. Global reset mode is generally used in conjunction with ERS mode. The ERS mode is used to provide viewfinder information, the sensor is switched into global reset mode to capture a single frame, and the sensor is then returned to ERS mode to restore viewfinder operation.

Global reset mode is designed for use in conjunction with the parallel pixel data interface. The SMIA specification does not define a global reset mode and only provides for operation in ERS mode. The AR0832 does support the use of global reset mode in conjunction with the SMIA data path, but there are additional restrictions on its use.

Overview of Global Reset Sequence

The basic elements of the global reset sequence are:

1. By default, the sensor operates in ERS mode and the SHUTTER output signal is LOW. The electromechanical shutter must be open to allow light to fall on the pixel array. Integration time is controlled by the coarse_integration_time and fine_integration_time registers.
2. A global reset sequence is triggered.
3. All of the rows of the pixel array are placed in reset.
4. All of the rows of the pixel array are taken out of reset simultaneously. All rows start to integrate incident light. The electromechanical shutter may be open or closed at this time.
5. If the electromechanical shutter has been closed, it is opened.
6. After the desired integration time (controlled internally or externally to the AR0832), the electromechanical shutter is closed.
7. A single output frame is generated by the sensor with the usual LV, FV, PIXCLK, and DOUT timing. As soon as the output frame has completed (FV negates), the electromechanical shutter may be opened again.
8. The sensor automatically resumes operation in ERS mode.

This sequence is shown in Figure 34. The following sections expand to show how the timing of this sequence is controlled.

Figure 34: Overview of Global Reset Sequence

Entering and Leaving the Global Reset Sequence

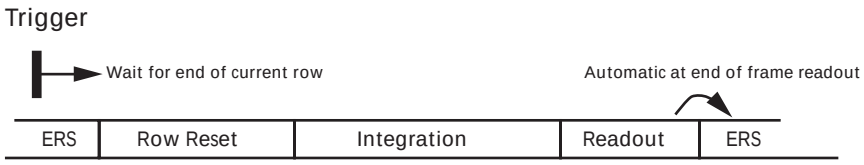
A global reset sequence can be triggered by a register write to R0x3160 global_seq_trigger[0] (global trigger, to transition this bit from a 0 to a 1) or by a rising edge on a suitably-configured GPI input (see "Trigger Control" on page 36).

When a global reset sequence is triggered, the sensor waits for the end of the current row. When LV negates for that row, FV is negated 6 PIXCLK periods later, potentially truncating the frame that was in progress.

The global reset sequence completes with a frame readout. At the end of this readout phase, the sensor automatically resumes operation in ERS mode. The first frame integrated with ERS will be generated after a delay of approximately ((13 + coarse_integration_time) * line_length_pck). This sequence is shown in Figure 35.

While operating in ERS mode, double-buffered registers ("Double-Buffered Registers" on page 23) are updated at the start of each frame in the usual way. During the global reset sequence, double-buffered registers are updated just before the start of the readout phase.

Figure 35: Entering and Leaving a Global Reset Sequence



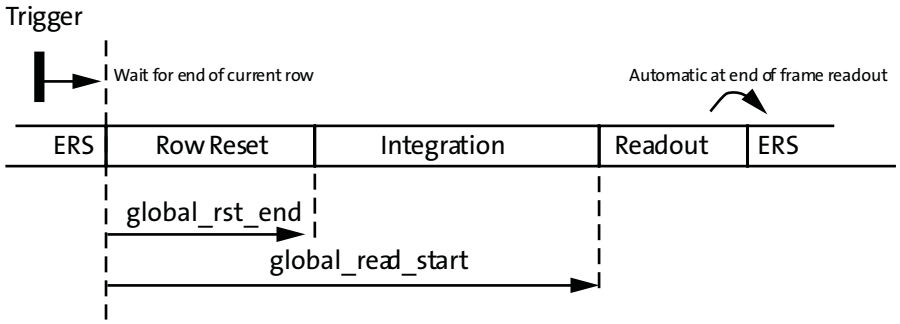
Programmable Settings

The registers global_rst_end and global_read_start allow the duration of the row reset phase and the integration phase to be controlled, as shown in Figure 36. The duration of the readout phase is determined by the active image size.

The recommended setting for global_rst_end is 0x00C8 (for example, 512 µs total reset time) with default vt_pix_clk. This allows sufficient time for all rows of the pixel array to be set to the correct reset voltage level. The row reset phase takes a finite amount of time due to the capacitance of the pixel array and the capability of the internal voltage booster circuit that is used to generate the reset voltage level.

As soon as the global_rst_end count has expired, all rows in the pixel array are taken out of reset simultaneously and the pixel array begins to integrate incident light.

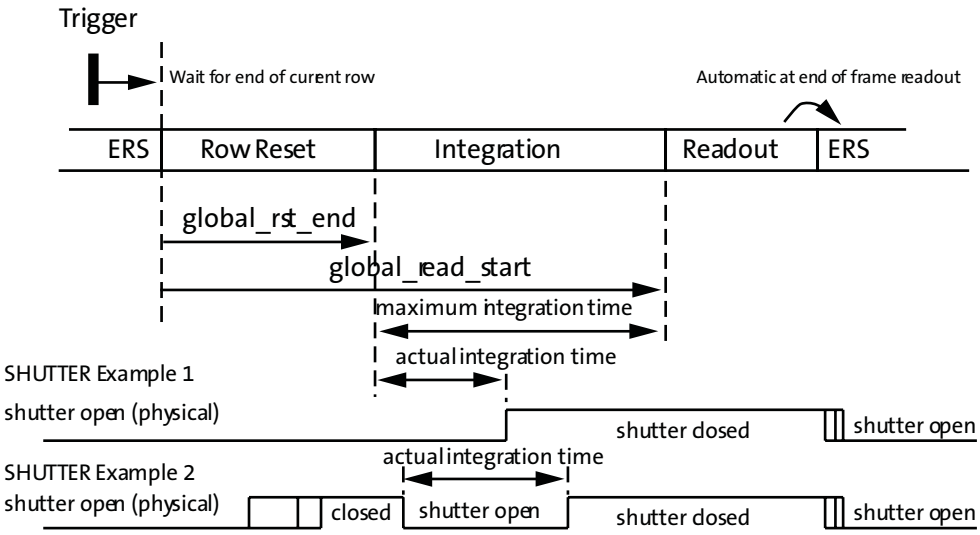
Figure 36: Controlling the Reset and Integration Phases of the Global Reset Sequence



Control of the Electromechanical Shutter

Figure 37 on page 61 shows two different ways in which a shutter can be controlled during the global reset sequence. In both cases, the maximum integration time is set by the difference between global_read_start and global_rst_end. In shutter example 1, the shutter is open during the initial ERS sequence and during the row reset phase. The shutter closes during the integration phase. The pixel array is integrating incident light from the start of the integration phase to the point at which the shutter closes. Finally, the shutter opens again after the end of the readout phase. In shutter example 2, the shutter is open during the initial ERS sequence and closes sometime during the row reset phase. The shutter both opens and closes during the integration phase. The pixel array is integrating incident light for the part of the integration phase during which the shutter is open. As for the previous example, the shutter opens again after the end of the readout phase.

Figure 37: Control of the Electromechanical Shutter



It is essential that the shutter remains closed during the entire row readout phase (that is, until FV has negated for the frame readout); otherwise, some rows of data will be corrupted (over-integrated).

It is essential that the shutter closes before the end of the integration phase. If the row readout phase is allowed to start before the shutter closes, each row in turn will be integrated for one row-time longer than the previous row.

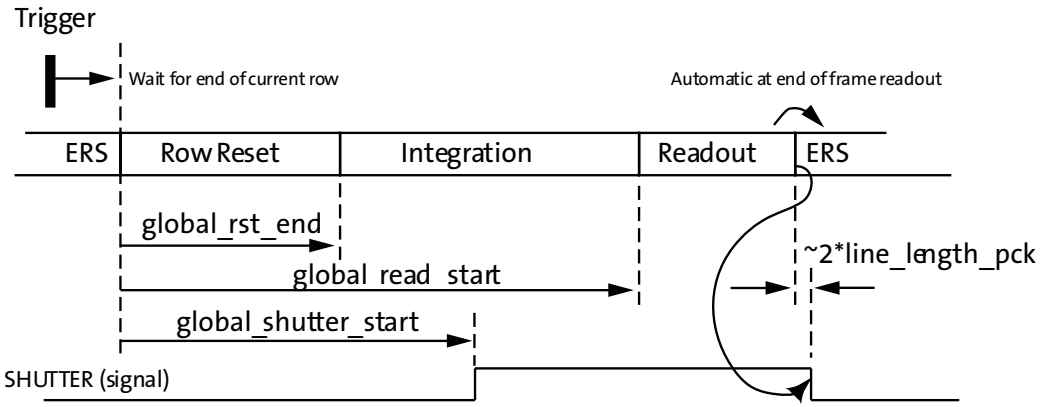
After FV negates to signal the completion of the readout phase, there is a time delay of approximately (10 * line_length_pck) before the sensor starts to integrate light-sensitive rows for the next ERS frame. It is essential that the shutter be opened at some point in this time window; otherwise, the first ERS frame will not be uniformly integrated.

The AR0832 provides a SHUTTER output signal to control (or help the host system control) the electromechanical shutter. The timing of the SHUTTER output is shown in Figure 38 on page 62. SHUTTER is negated by default. The point at which it asserts is controlled by the programming of global_shutter_start. At the end of the global reset readout phase, SHUTTER negates approximately (2 * line_length_pck) after the negation of FV.

This programming restriction must be met for correct operation:

- global_read_start > global_shutter_start.

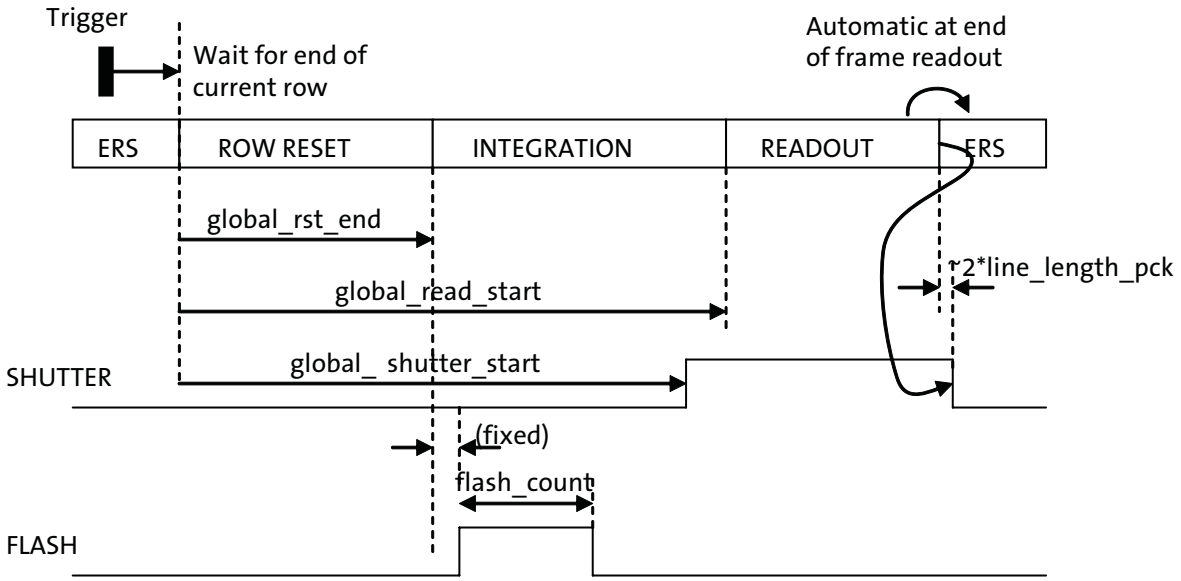
Figure 38: Controlling the SHUTTER Output



Using FLASH with Global Reset

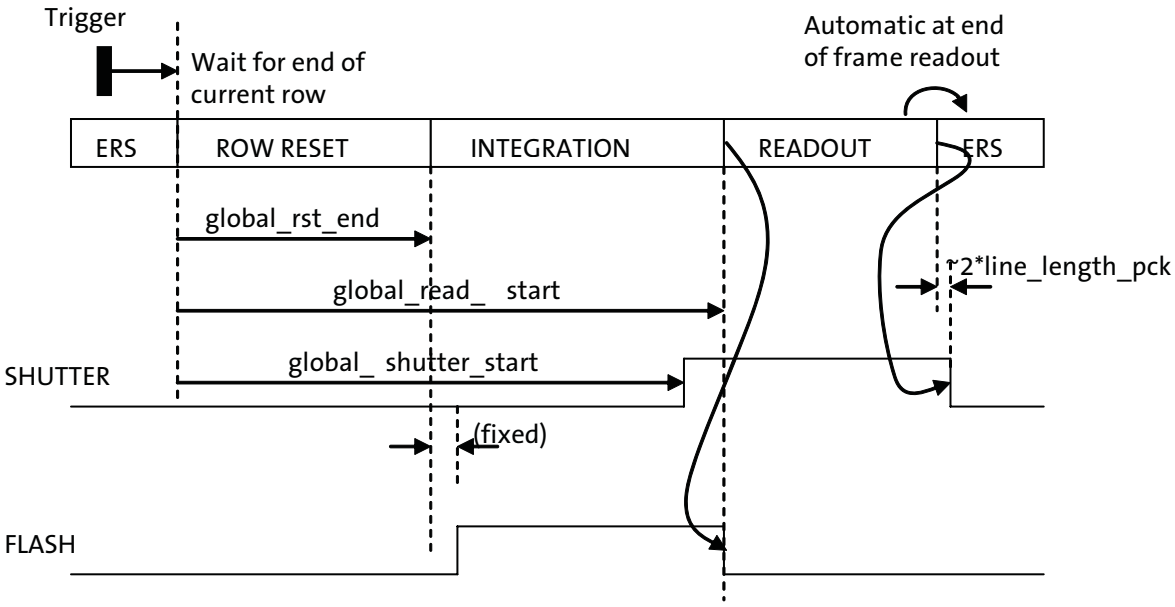
If R0x3160 global_seq_trigger[2] = 1 (global flash enabled) when a global reset sequence is triggered, the FLASH output signal will be pulsed during the integration phase of the global reset sequence. The FLASH output will assert a fixed number of cycles after the start of the integration phase and will remain asserted for a time that is controlled by the value of the flash_count register. When flash_count is programmed for value N, (where N is 0–0x3FE) the resulting flash duration is given by N * 512 * (1/vt_pix_clk_freq_mhz), as shown in Figure 39.

Figure 39: Using FLASH with Global Reset



When the flash_count = 0x3FF, the flash signal will be maximized and goes LOW when readout starts, as shown in Figure 40 on page 63. This would be preferred if the latency in closing the shutter is longer than the latency for turning off the flash. This guarantees that the flash stays on while the shutter is open.

Figure 40: Extending FLASH Duration in Global Reset (Reference Readout Start)



External Control of Integration Time

If global_seq_trigger[1] = 1 (global bulb enabled) when a global reset sequence is triggered, the end of the integration phase is controlled by the level of trigger (global_seq_trigger[0] or the associated GPI input). This allows the integration time to be controlled directly by an input to the sensor.

This operation corresponds to the shutter "B" setting on a traditional camera, where "B" originally stood for "Bulb" (the shutter setting used for synchronization with a magnesium foil flash bulb) and was later considered to stand for "Brief" (an exposure that was longer than the shutter could automatically accommodate).

When the trigger is de-asserted to end integration, the integration phase is extended by a further time given by *global_read_start – global_shutter_start*. Usually this means that global_read_start should be set to *global_shutter_start + 1*.

The operation of this mode is shown in Figure 41 on page 64. The figure shows the global reset sequence being triggered by the GPI2 input, but it could be triggered by any of the GPI inputs or by the setting and subsequence clearing of the global_seq_trigger[0] under software control.

The integration time of the GRR sequence is defined as:

$$Integration\ Time = \frac{global\_scale \times [global\_read\_start - global\_shutter\_start - global\_rst\_end]}{vt\_pix\_clk\_freq\_mhz} \quad \text{(EQ 12)}$$

Where:

$$global\_read\_start = (2^{16} \times global\_read\_start2[7:0] + global\_read\_start1[15:0]) \quad \text{(EQ 13)}$$

$$global\_shutter\_start = (2^{16} \times global\_shutter\_start2[7:0] + global\_shutter\_start1[15:0]) \quad \text{(EQ 14)}$$

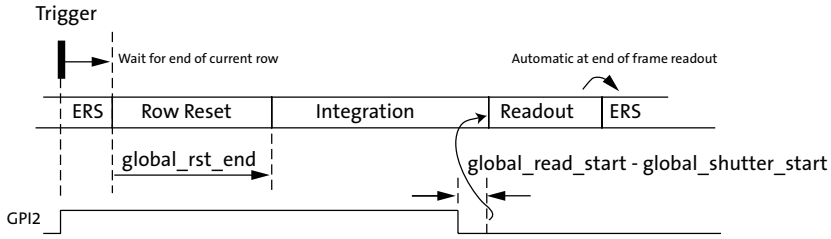
The integration equation allows for 24-bit precision when calculating both the shutter and readout of the image. The global_rst_end has only 16-bit as the array reset function and requires a short amount of time.

The integration time can also be scaled using global_scale. The variable can be set to 0–512, 1–2048, 2–128, and 3–32.

These programming restrictions must be met for correct operation of bulb exposures:

- global_read_start > global_shutter_start
- global_shutter_start > global_rst_end
- global_shutter_start must be smaller than the exposure time (that is, this counter must expire before the trigger is de-asserted)

Figure 41: Global Reset Bulb



Retriggering the Global Reset Sequence

The trigger for the global reset sequence is edge-sensitive; the global reset sequence cannot be retriggered until the global trigger bit (in the R0x3160 global_seq_trigger register) has been returned to "0," and the GPI (if any) associated with the trigger function has been negated.

The earliest time that the global reset sequence can be retriggered is the point at which the SHUTTER output negates; this occurs approximately (2 * line_length_pck) after the negation of FV for the global reset readout phase.

Using Global Reset with SMIA Data Path

When a global reset sequence is triggered, it usually results in the frame in progress being truncated (at the end of the current output line). The SMIA data path limiter function (see Figure 48 on page 75) attempts to extend (pad) all frames to the programmed value of y_output_size. If this padding is still in progress when the global reset readout phase starts, the SMIA data path will not detect the start of the frame correctly. Therefore, to use global reset with the serial data path, this timing scenario must be avoided. One possible way of doing this would be to synchronize (under software control) the assertion of trigger to an end-of-frame marker on the CCP serial data stream.

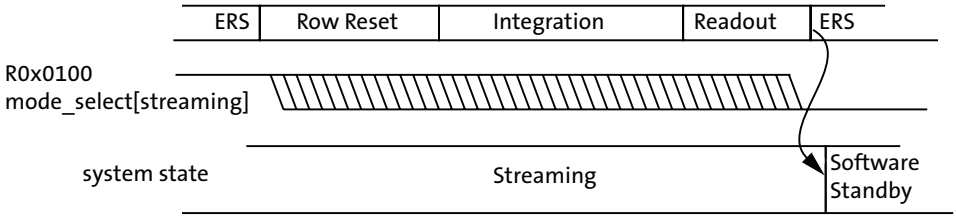
At the end of the readout phase of the global reset sequence, the sensor automatically resumes operation in ERS mode.

The frame that is read out of the sensor during the global reset readout phase has exactly the same format as any other frame out of the serial pixel data interface, including the addition of two lines of embedded data. The values of the coarse_integration_time and fine_integration_time registers within the embedded data match the programmed values of those registers and do *not* reflect the integration time used during the global reset sequence.

Global Reset and Soft Standby

If the R0x0100 mode_select[stream] bit is cleared while a global reset sequence is in progress, the AR0832 will remain in streaming state until the global reset sequence (including frame readout) has completed, as shown in Figure 42 on page 65.

Figure 42: Entering Soft Standby During a Global Reset Sequence



Analog Gain

The AR0832 provides two mechanisms for setting the analog gain. The first uses the SMIA gain model; the second uses the traditional Aptina gain model.The following sections describe both models, the mapping between the models, and the operation of the per-color and global gain control.

Using Per-Color or Global Gain Control

The read-only analogue_gain_capability register returns a value of "1," indicating that the AR0832 provides per-color gain control. However, the AR0832 also provides the option of global gain control. Per-color and global gain control can be used interchangeably. A WRITE to a global gain register is aliased as a write of the same data to the four associated color-dependent gain registers. A READ from a global gain register is aliased to a read of the associated greenB/greenR gain register.

The read/write gain_mode register required by SMIA has no defined function. In the AR0832, this register has no side-effects on the operation of the gain; per-color and global gain control can be used interchangeably regardless of the state of the gain_mode register.

SMIA Gain Model

The SMIA gain model uses these registers to set the analog gain:

- analogue_gain_code_global
- analogue_gain_code_greenR
- analogue_gain_code_red
- analogue_gain_code_blue
- analogue_gain_code_greenB

The SMIA gain model requires a uniform step size between all gain settings. The analog gain is given by:

$$gain = \frac{analogue\_gain\_m0 \times analogue\_gain\_code}{analogue\_gain\_c1} = \frac{analogue\_gain\_code\_<color>}{8} \quad \text{(EQ 15)}$$

Aptina Gain Model

The Aptina gain model uses these registers to set the analog gain:

- global_gain
- green1_gain
- red_gain
- blue_gain
- green2_gain

The AR0832 uses 11 bits analog gain control. The total analog gain is given by:

$$(Total\ gain) = (8 / (8\text{-}gain<10.8>)) \times (gain<7> + 1) \times gain<6.0> \times \frac{1}{64} \quad \text{(EQ 16)}$$

As a result, the step size varies depending upon which range the gain is in. Many of the possible gain settings can be achieved in different ways. However, the recommended gain settings are listed in Table 26, which will result lower noise.

Table 26: Recommended Gain Settings

Desired Gain	Recommended Gain Register Setting
1x – 1.984375x	0x040 – 0x07F
2x – 3.96875x	0x440 – 0x47F
4x – 7.9375x	0x640 – 0x67F
8x – 15.875x	0x6C0 – 0x6FF

Gain Code Mapping

The Aptina gain model maps directly to the underlying structure of the gain stages in the analog signal chain. When the SMIA gain model is used, gain codes are translated into equivalent settings in the Aptina gain model.

When the SMIA gain model is in use and values have been written to the analogue_gain_code_<color> registers, the associated value in the Aptina gain model can be read from the associated <color>_gain register. In cases where there is more than one possible mapping, follow the recommended gain settings to achieve the lowest noise.

The result of this is that the two gain models can be used interchangeably, but once gains have been written gains through one set of registers, those gains should be read back through the same set of registers.

Sensor Core Digital Data Path

Test Patterns

The AR0832 supports a number of test patterns to facilitate system debug. Test patterns are enabled using test_pattern_mode (R0x0600–1). The test patterns are listed in Table 27.

Table 27: Test Patterns

test_pattern_mode	Description
0	Normal operation: no test pattern
1	Solid color
2	100% color bars
3	Fade-to-gray color bars
4	PN9 link integrity pattern (only on sensors with serial interface)
256	Walking 1s (10-bits)
257	Walking 1s (8-bits)

Test patterns 0–3 replace pixel data in the output image (the embedded data rows are still present). Test pattern 4 replaces all data in the output image (the embedded data rows are omitted and test pattern data replaces the pixel data).

For all of the test patterns, the AR0832 registers must be set appropriately to control the frame rate and output timing. This includes:

- All clock divisors
- x_addr_start
- x_addr_end
- y_addr_start
- y_addr_end
- frame_length_lines
- line_length_pck
- x_output_size
- y_output_size

Effect of Data Path Processing on Test Patterns

Test patterns are introduced early in the pixel data path. As a result, they can be affected by pixel processing that occurs within the data path. This includes:

- Black pedestal adjustment
- Lens and color shading correction

These effects can be eliminated by the following register settings:

- R0x3044–5[10] = 0
- R0x30CA–B[0] = 1
- R0x30D4–5[15] = 0
- R0x31E0–1[0] = 0
- R0x301A–B[3] = 0 (enable writes to data pedestal)
- R0x301E–F = 0x0000 (set data pedestal to “0”)
- R0x3780[15] = 0 (turn off lens/color shading correction)

Solid Color Test Pattern

In this mode, all pixel data is replaced by fixed Bayer pattern test data. The intensity of each pixel is set by its associated test data register (test_data_red, test_data_greenR, test_data_blue, test_data_greenB).

100% Color Bars Test Pattern

In this test pattern, shown in Figure 43 on page 70, all pixel data is replaced by a Bayer version of an 8-color, color-bar chart (white, yellow, cyan, green, magenta, red, blue, black). Each bar is 1/8 of the width of the pixel array (3264/8 = 408 pixels). The pattern repeats after 8 * 408 = 3264 pixels.

Each color component of each bar is set to either 0 (fully off) or 0x3FF (fully on for 10-bit data).

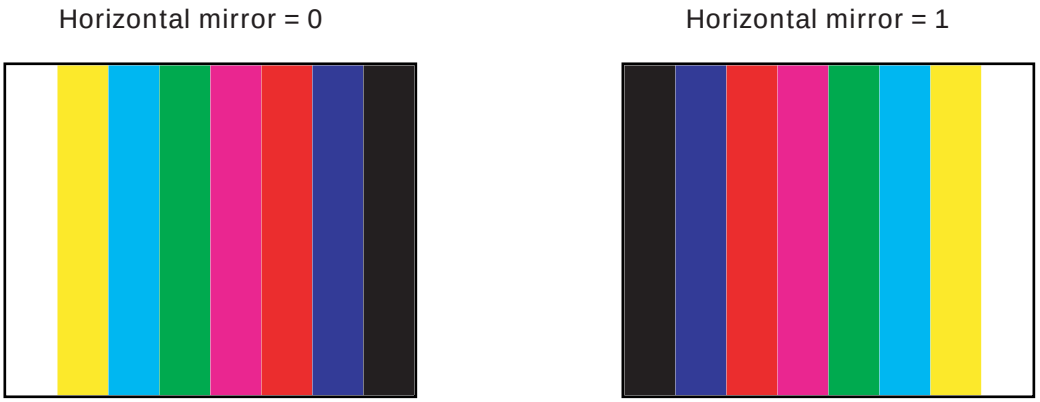
The pattern occupies the full height of the output image.

The image size is set by x_addr_start, x_addr_end, y_addr_start, y_addr_end and may be affected by the setting of x_output_size, y_output_size. The color-bar pattern is disconnected from the addressing of the pixel array, and will therefore always start on the first visible pixel, regardless of the value of x_addr_start. The number of colors that are visible in the output is dependent upon x_addr_end - x_addr_start and the setting of x_output_size: the width of each color bar is fixed at 408 pixels.

The effect of setting horizontal_mirror in conjunction with this test pattern is that the order in which the colors are generated is reversed: the black bar appears at the left side of the output image. Any pattern repeat occurs at the right side of the output image regardless of the setting of horizontal_mirror. The state of vertical_flip has no effect on this test pattern.

The effect of subsampling, binning and scaling of this test pattern is undefined.

Figure 43: 100 Percent Color Bars Test Pattern



Fade-to-gray Color Bars Test Pattern

In this test pattern, shown in Figure 44 on page 71, all pixel data is replaced by a Bayer version of an 8-color, color-bar chart (white, yellow, cyan, green, magenta, red, blue, black). Each bar is 1/8 of the width of the pixel array (3264/8 = 408 pixels). The test pattern repeats after 3264 pixels.

Each color bar fades vertically from zero or full intensity at the top of the image to 50 percent intensity (mid-gray) on the last TBD row of the pattern. Each color bar is divided into a left and a right half, in which the left half fades smoothly and the right half fades in quantized steps.

The speed at which each color fades is dependent on the sensor's data width and the height of the pixel array. We want half of the data range (from 100 or 0 to 50 percent) difference between the top and bottom of the pattern. Because of the Bayer pattern, each state must be held for two rows.

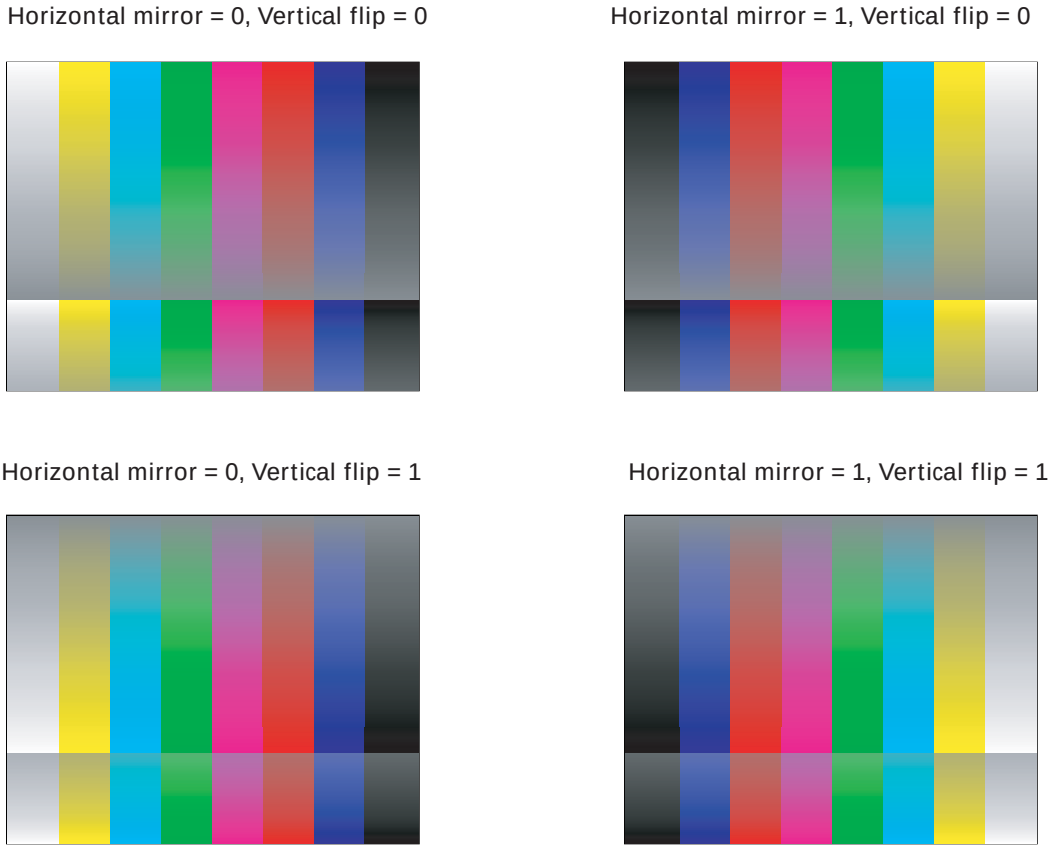
The rate-of-fade of the Bayer pattern is set so that there is at least one full pattern within a full-sized image for the sensor. Factors that affect this are the resolution of the ADC (10-bit) and the image height. For example, the AR0832 fades the pixels by 2 LSB for each two rows. With 10-bit data, the pattern is 2048 pixels high and repeats after that, if the window is higher.

The image size is set by x_addr_start, x_addr_end, y_addr_start, y_addr_end and may be affected by the setting of x_output_size, y_output_size. The color-bar pattern starts at the first column in the image, regardless of the value of x_addr_start. The number of colors that are visible in the output is dependent upon x_addr_end – x_addr_start and the setting of x_output_size: the width of each color bar is fixed at 408 pixels.

The effect of setting horizontal_mirror or vertical_flip in conjunction with this test pattern is that the order in which the colors are generated is reversed: the black bar appears at the left side of the output image. Any pattern repeat occurs at the right side of the output image regardless of the setting of horizontal_mirror.

The effect of subsampling, binning, and scaling of this test pattern is undefined.

Figure 44: Fade-to-Gray Color Bars Test Pattern



PN9 Link Integrity Pattern

The PN9 link integrity pattern is intended to allow testing of a CCP2 serial pixel data interface. Unlike the other test patterns, the position of this test pattern at the end of the data path means that it is not affected by other data path corrections.

This test pattern provides a 512-bit pseudo-random test sequence to test the integrity of the serial pixel data output stream. The polynomial x9 + x5 + 1 is used. The polynomial is initialized to 0x1FF at the start of each frame.

When this test pattern is enabled:

- The embedded data rows are disabled and the value of frame_format_decriptor_1 changes from 0x1002 to 0x1000 to indicate that no rows of embedded data are present.
- The whole output frame, bounded by the limits programmed in x_output_size and y_output_size, is filled with data from the PN9 sequence.
- The output data format is (effectively) forced into RAW10 mode regardless of the state of the ccp_data_format register.

Before enabling this test pattern the clock divisors must be configured for RAW10 operation (op_pix_clk_div = 10).

This polynomial generates this sequence of 10-bit values: 0x1FF, 0x378, 0x1A1, 0x336, 0x385... . On the parallel pixel data output, these values are presented 10 bits per PIXCLK. On the serial pixel data output, these values are streamed out sequentially without performing the RAW10 packing to bytes that normally occurs on this interface.

Walking 1s

When selected, a walking 1s pattern will be sent through the digital pipeline. The first value in each row is 0. Each value will be valid for 2 pixels.

Figure 45: Walking 1s 10-bit Pattern

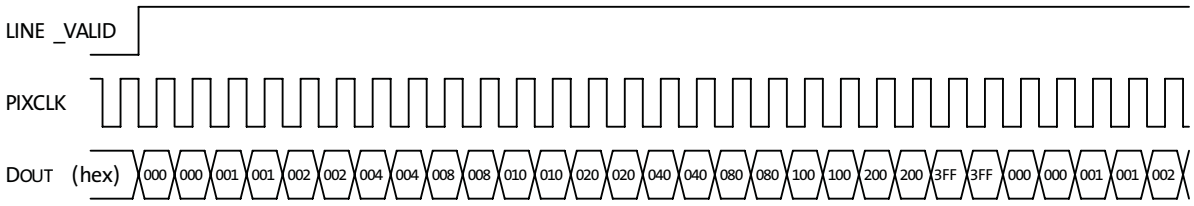
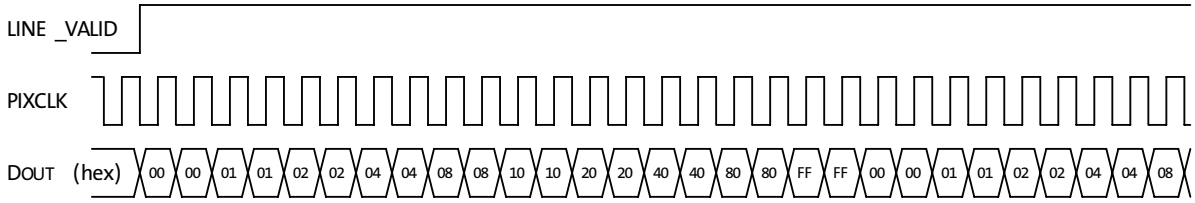


Figure 46: Walking 1s 8-bit Pattern



The walking 1s pattern was implemented to facilitate assembly testing of modules with a parallel interface. The false synchronization code removal imposed by CCP2 serial interfaces mean that this test pattern cannot be carried across a CCP2 link in an unmodified form.

The walking 1 test pattern is not active during the blanking periods, hence the output would reset to a value of 0x0. When the active period starts again, the pattern would restart from the beginning. The behavior of this test pattern is the same between full resolution and subsampling mode. RAW10 and RAW8 walking 1 modes are enabled by different test pattern codes.

The walking 1 test pattern is not active during the blanking periods, hence the output would reset to a value of 0x0. When the active period starts again, the pattern would restart from the beginning. The behavior of this test pattern is the same between full resolution and subsampling modes. RAW10 and RAW8 walking 1 modes are enabled by different test pattern codes.

Test Cursors

The AR0832 supports one horizontal and one vertical cursor, allowing a crosshair to be superimposed on the image or on test patterns 1–3. The position and width of each cursor are programmable in registers 0x31E8–0x31EE. Both even and odd cursor positions and widths are supported.

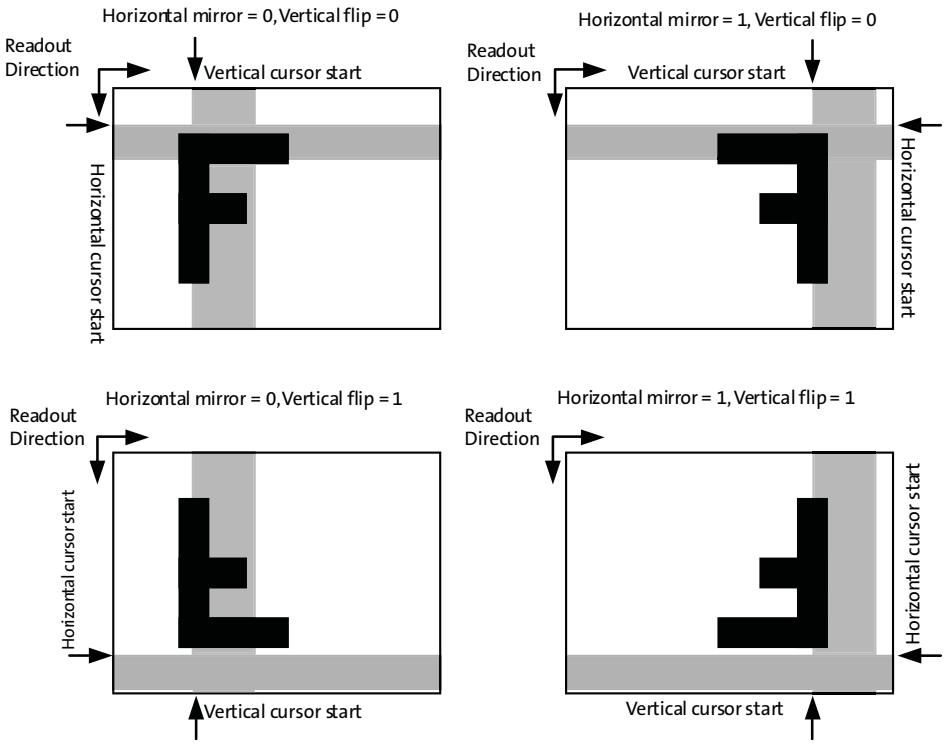
Each cursor can be inhibited by setting its width to 0. The programmed cursor position corresponds to the x and y addresses of the pixel array. For example, setting horizontal_cursor_position to the same value as y_addr_start would result in a horizontal cursor being drawn starting on the first row of the image. The cursors are opaque (they replace data from the imaged scene or test pattern). The color of each cursor is set by the values of the Bayer components in the test_data_red, test_data_greenR, test_data_blue and test_data_greenB registers. As a consequence, the cursors are the same color as test pattern 1 and are therefore invisible when test pattern 1 is selected.

When vertical_cursor_position = 0x0fff, the vertical cursor operates in an automatic mode in which its position advances every frame. In this mode the cursor starts at the column associated with x_addr_start = 0 and advances by a step-size of 8 columns each frame, until it reaches the column associated with x_addr_start = 2040, after which it wraps (256 steps). The width and color of the cursor in this automatic mode are controlled in the usual way.

The effect of enabling the test cursors when the image_orientation register is non-zero is not defined by the design specification. The behavior of the AR0832 is shown in Figure 47 on page 74 and the test cursors are shown as translucent, for clarity. In practice, they are opaque (they overlay the imaged scene). The manner in which the test cursors are affected by the value of image_orientation can be understood from these implementation details:

- The test cursors are inserted last in the data path, the cursor is applied with out any sensor corrections.
- The drawing of a cursor starts when the pixel array row or column address is within the address range of cursor start to cursor start + width.
- The cursor is independent of image orientation.

Figure 47: Test Cursor Behavior with image_orientation



Digital Gain

Integer digital gains in the range 1–7 can be programmed.

Pedestal

This block adds the value from R0x0008–9 or (data_pedestal_) to the incoming pixel value.

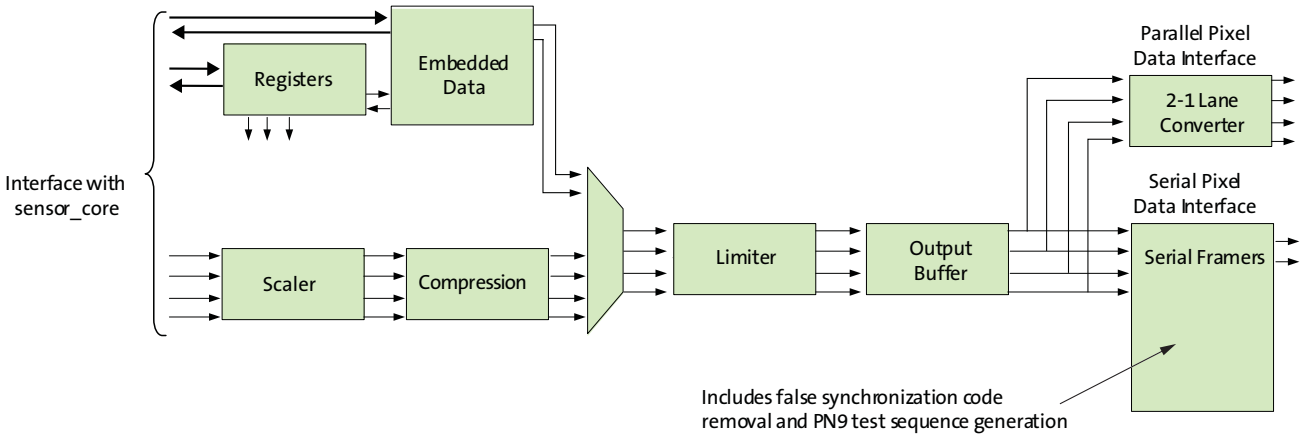
The data_pedestal register is read-only by default but can be made read/write by clearing the lock_reg bit in R0x301A–B.

The only way to disable the effect of the pedestal is to set it to 0.

Digital Data Path

The digital data path after the sensor core is shown in Figure 48.

Figure 48: Data Path



Embedded Data Format and Control

When the serial pixel data path is selected, the first two rows of the output image contain register values that are appropriate for the image. The 10-bit format places the data byte in bits [11:4] and sets bits [3:0] to a constant value of 0101. Some register values are dynamic and may change from frame to frame. Additional information on the format of the embedded data can be located in the SMIA specification.

Timing Specifications

Power-Up Sequence Using Internal Regulator

The recommended power-up sequence for the AR0832 is shown in Figure 49. The available power supplies—VDD_IO, VDD_TX, VDD_PLL, VAA, VAA_PIX—can be turned on at the same time or have the separation specified below.

1. Turn on VDD_IO power supply.
2. After 1–500ms, turn on VDD_TX power supply.
3. After 1–500ms, turn on VAA/VAA_PIX power supplies.
4. After the last power supply is stable, enable EXTCLK.
5. Assert RESET_BAR for at least 1ms.
6. Wait 2400 EXTCLKs for internal initialization into software standby.
7. Configure PLL, output, and image settings to desired values
8. Set mode_select = 1 (R0x0100).

Figure 49: Power-Up Sequence

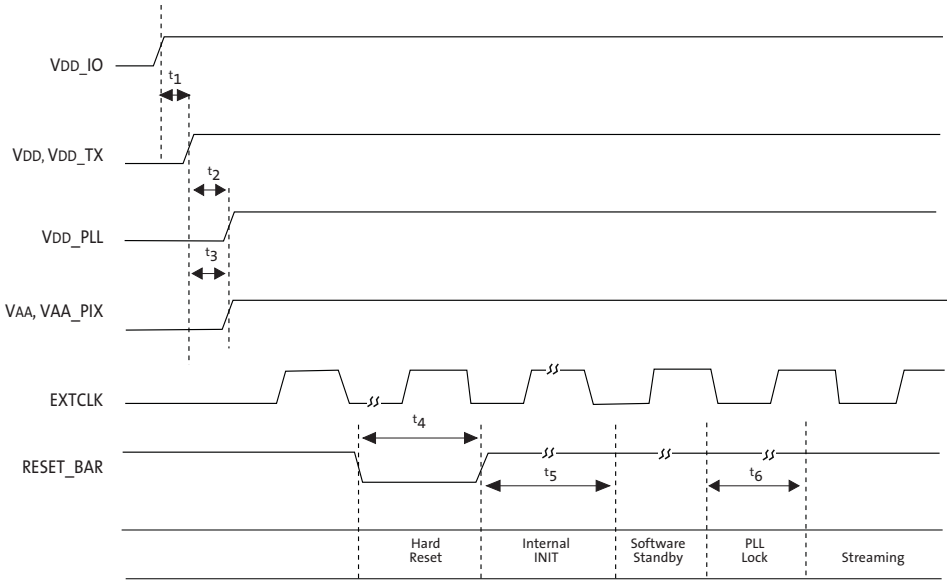


Table 28: Power-Up Sequence

Definition	Symbol	Min	Typ	Max	Unit
VDD_IO to VDD_TX time	t_1	–	–	500	ms
VDD/VDD_TX to VDD_PLL time	t_2	–	–	500	ms
VDD_TX to VAA/VAA_PIX time	t_3	–	–	500	ms
Active hard reset	t_4	–	–	500	ms
Internal initialization	t_5	2400	–	–	EXTCLKs
PLL lock time	t_6	–	–	500	ms

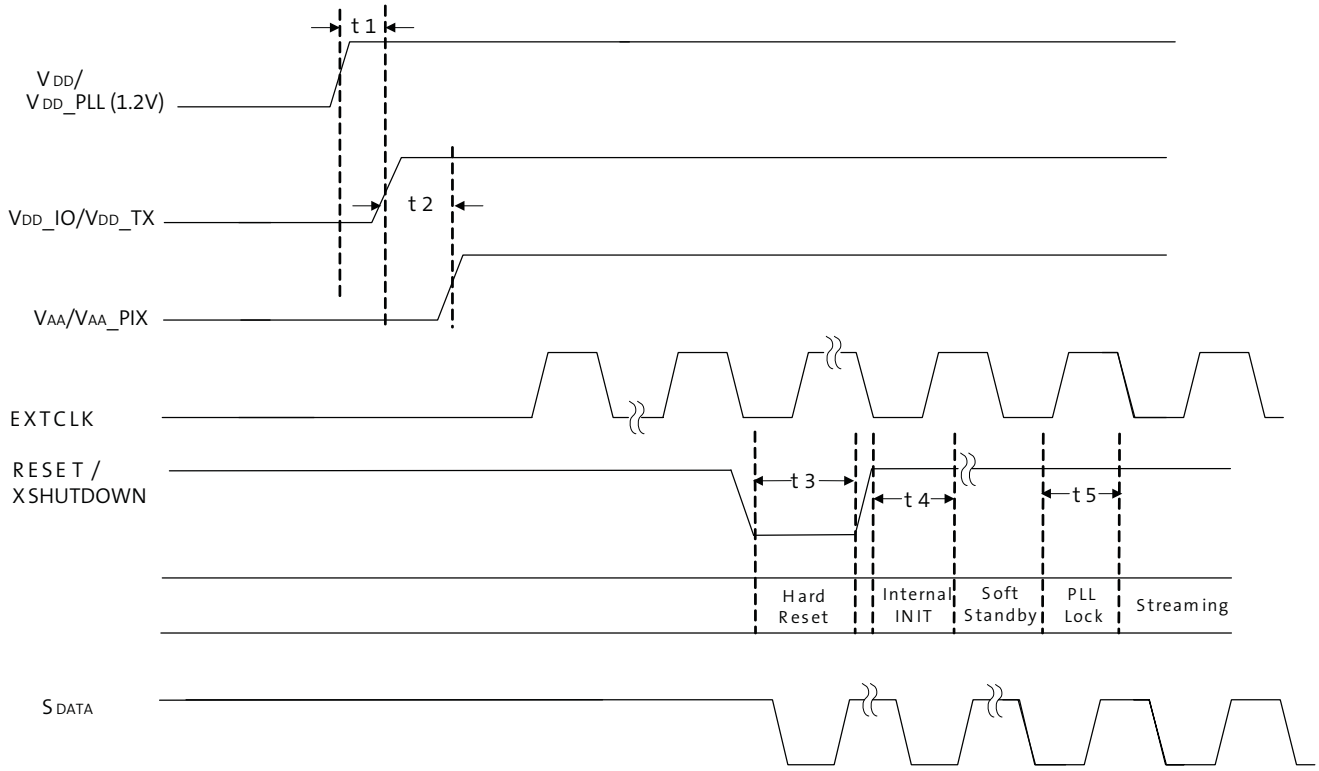
Note: Digital supplies must be turned on before analog supplies.

Power-Up Sequence Using External Regulator

The recommended power-up sequence for the AR0832 is shown in Figure 1. The available power supplies – VDD/VDD_PLL, VDD_IO, VAA/ VAA_PIX – can be turned on at the same time or have the separation specified below.

1. Turn on VDD/VDD_PLL power supply.
2. After 1-500ms, turn on VDD_IO power supply.
3. After 1-500ms, turn on VAA/VAA_PIX power supplies.
4. After the last power supply is stable, enable EXTCLK.
5. Assert RESET_BAR for at least 1ms.
6. Wait 2400 EXTCLKs for internal initialization into software standby.
7. Configure PLL, output, and image settings to desired values
8. Set mode_select = 1 (R0x0100).

Figure 50: Power-Up Sequence without Internal Regulator



Note: There is no SDATA low issue when AR0832 is powered up in non-regulated mode using the above sequence.

The reliability of non-regulator mode with new power up sequence is same as the one with regulator mode (The reliability includes the sensor life time, image quality, operation and so on).

Table 29: Power-Up Timing

Definition	Symbol	Min	Typ	Max	Unit
VDD/VDD_PLL to VDD_IO	T1	1	-	500	ms
VDD_IO to VAA/VAA_PIX	T2	1	-	500	ms
Active hard Reset	T3	1	-	500	ms
Internal Initialization	T4	2400	-	-	EXTCLKs
PLL Lock Time	T5	1	-	500	ms

Power-Down Sequence Using Internal Regulator

The recommended power-down sequence for the AR0832 is shown in Figure 51. The available power supplies—VDD_IO, VDD_TX0, VAA, VAA_PIX—can be turned off at the same time or have the separation specified below.

1. Disable streaming if output is active by setting mode_select = 0 (R0x0100).
2. The soft standby state is reached after the current row or frame, depending on configuration, has ended.
3. Assert hard reset by setting RESET_BAR to a logic "0."
4. Turn off the VAA/VAA_PIX power supplies.
5. After 1–500ms, turn off VDD_TX0 power supply.
6. After 1–500ms, turn off VDD_IO power supply.

Figure 51: Power-Down Sequence

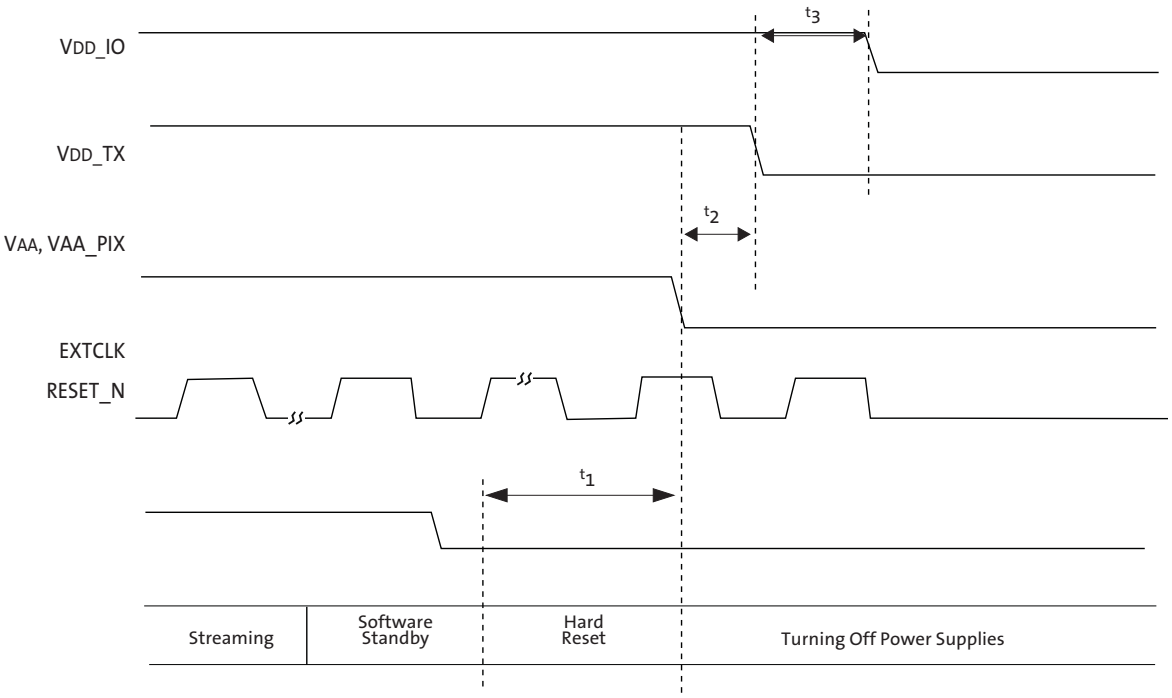


Table 30: Power-Down Sequence

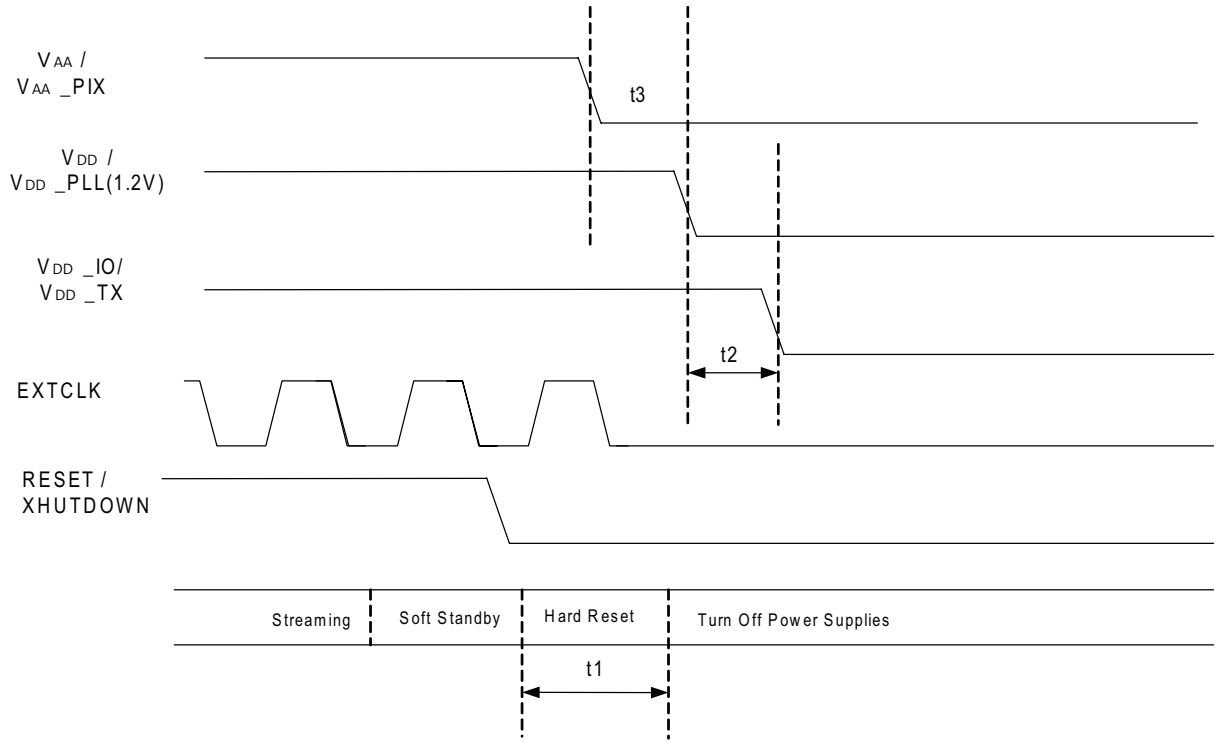
Definition	Symbol	Min	Typ	Max	Unit
Hard reset	t_1	0	–	500	ms
VAA/VAA_PIX to VDD_TX time	t_2	0	–	500	ms
VDD_TX to VDD_IO time	t_3	–0	–	500	ms

Power-Down Sequence Without Internal Regulator

The recommended power-down sequence for the AR0832 is shown in Figure 52. The available power supplies—VDD_IO/VDD_TX, VDD/VDD_PLL, VAA, VAA_PIX—can be turned off at the same time or have the separation specified below.

1. Disable streaming if output is active by setting mode_select = 0 (R0x0100).
2. The soft standby state is reached after the current row or frame, depending on configuration, has ended.
3. Assert hard reset by setting RESET_BAR to a logic "0."
4. Turn off the VAA/VAA_PIX power supplies.
5. After 1-500ms, turn off VDD_IO/VDD_TX power supply.
6. After 1-500ms, turn off VDD/VDD_PLL power supply.

Figure 52: Power-Down Sequence Without Internal Regulator



Note: There is no SDATA low issue when AR0832 is powered down in non-regulated mode using the above sequence.

The reliability of non-regulator mode with new power down sequence is same as the one with regulator mode (The reliability includes the sensor life time, image quality, operation and so on).

Table 31: Power-Down Timing Without Internal Regulator

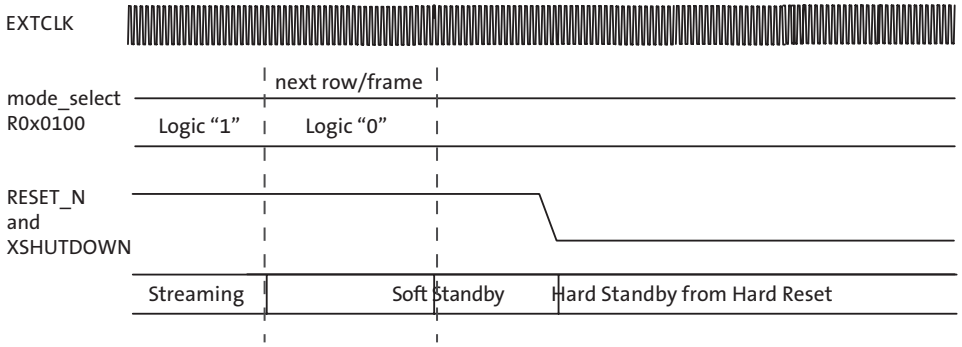
Definition	Symbol	Min	Typ	Max	Unit
Hard Reset	t1	1	-	500	ms
VDD/VDD_PLL to VDD_IO/VDD_TX	t2	1	-	500	ms
VAA/VAA_PIX to VDD/VDD_PLL (1.2V)	t3	1	-	500	ms

Hard Standby and Hard Reset

The hard standby state is reached by the assertion of the RESET_BAR and XSHUTDOWN pads (hard reset). Register values are not retained by this action, and will be returned to their default values once hard reset is completed. The minimum power consumption is achieved by the hard standby state. The details of the sequence are described below and shown in Figure 53.

1. Disable streaming if output is active by setting mode_select = 0 (R0x0100).
2. The soft standby state is reached after the current row or frame, depending on configuration, has ended.
3. Assert RESET_BAR and XSHUTDOWN (active LOW) to reset the sensor.
4. The sensor remains in hard standby state if RESET_BAR and XSHUTDOWN remain in the logic “0” state.

Figure 53: Hard Standby and Hard Reset



Soft Standby and Soft Reset

The AR0832 can reduce power consumption by switching to the soft standby state when the output is not needed. Register values are retained in the soft standby state. Once this state is reached, soft reset can be enabled optionally to return all register values back to the default. The details of the sequence are described below and shown in Figure 54 on page 81.

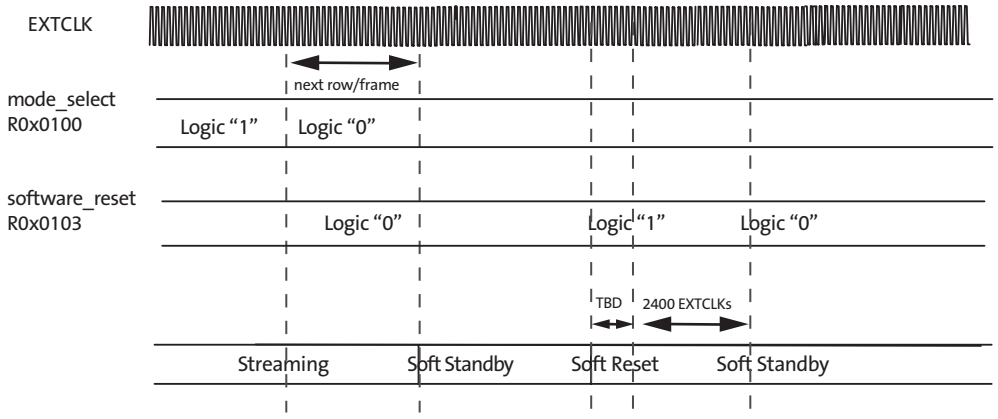
Soft Standby

1. Disable streaming if output is active by setting mode_select = 0 (R0x0100).
2. The soft standby state is reached after the current row or frame, depending on configuration, has ended.

Soft Reset

1. Follow the soft standby sequence list above.
2. Set software_reset = 1 (R0x0103) to start the internal initialization sequence.
3. After 2400 EXTCLKs, the internal initialization sequence is completed and the current state returns to soft standby automatically. All registers, including software_reset, returns to their default values.

Figure 54: Soft Standby and Soft Reset



Internal VCM Driver

The AR0832 utilizes an internal Voice Coil Motor (VCM) driver. The VCM functions are register-controlled through the serial interface.

There are two output ports, VCM_ISINK and GNDIO_VCM, which would connect directly to the AF actuator.

Take precautions in the design of the power supply routing to provide a low impedance path for the ground return. Appropriate filtering would also be required on the actuator supply. Typical values would be a 0.1µF and 10µF in parallel.

Figure 55: VCM Driver Typical Diagram

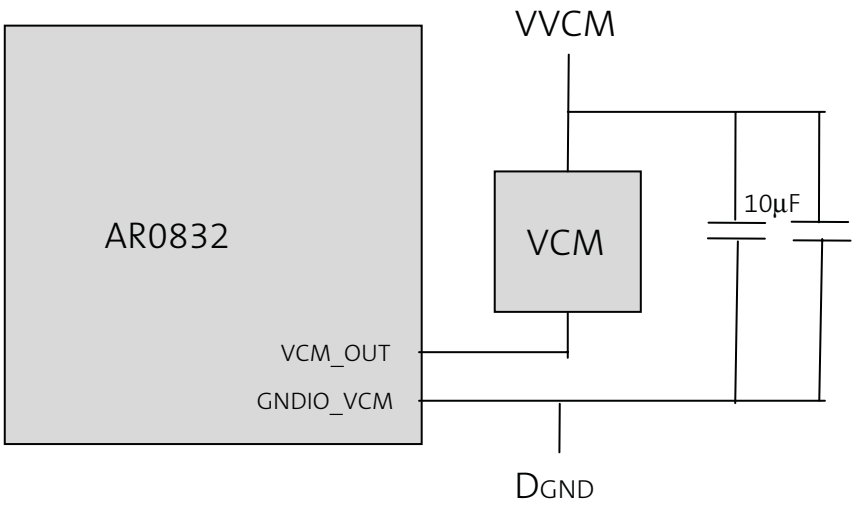


Table 32: VCM Driver Typical

Characteristic	Parameter	Minimum	Typical	Maximum	Units
VCM_OUT	Voltage at VCM current sink	2.5	2.8	3.3	V
WVCM	Voltage at VCM actuator	2.5	2.8	3.3	V
INL	Relative accuracy	–	±1.5	± 4	LSB
RES	Resolution	–	8	–	bits
DNL	Differential nonlinearity	–1	–	+1	LSB
IVCM	Output current	5	–	100	mA
	Slew rate	–	.3	–	mA/µs

User -Accessible Internal ADC

The AR0832 provides access to an internal 12-bit ADC for customer use. The ADC provides sampling, correction, and filtering.

One application for the internal ADC is to use as an interface to AF drivers that require analog feedback support. The access to the internal ADC is through the ATEST0/1 pins and the data is accessible through the serial interface.

Multimaster Serial Interface

The AR0832 provides, in addition to the standard serial interface (SDATA, SCLK), another two-wire serial interface for customer use. These could be used for any number of uses to control external components such as auto focus, mechanical shutter, and flash drivers.

These are the S_SCLK and S_DATA pins.

Spectral Characteristics

Figure 56: Quantum Efficiency

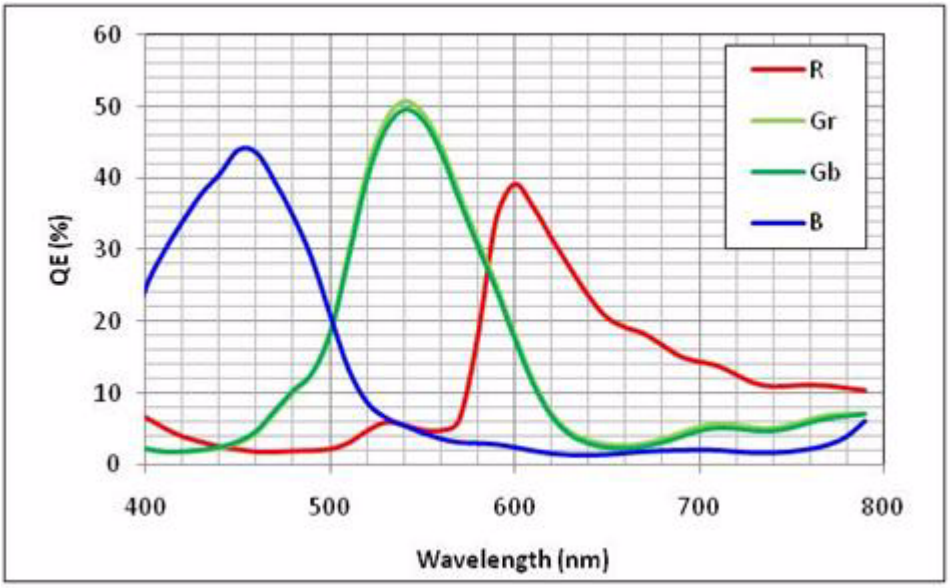
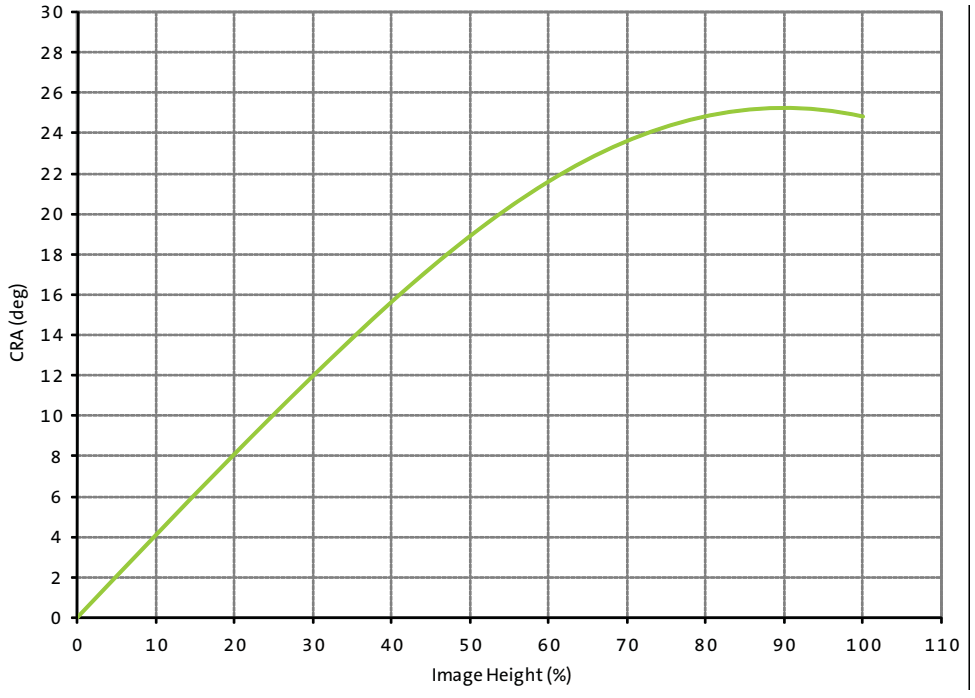


Figure 57: Chief Ray Angle

CRA vs. Image Height Plot	Image Height (%)	Image Height (mm)	CRA (°)
	0	0	0
	5	0.143	2.07
	10	0.286	4.09
	15	0.428	6.10
	20	0.571	8.09
	25	0.714	10.05
	30	0.857	11.98
	35	1.000	13.84
	40	1.142	15.63
	45	1.285	17.33
	50	1.428	18.90
	55	1.571	20.33
	60	1.714	21.60
	65	1.856	22.69
	70	1.999	23.60
	75	2.142	24.32
	80	2.285	24.83
	85	2.428	25.14
	90	2.570	25.24
	95	2.713	25.13
	100	2.856	24.81

Read the Sensor CRA

Follow the steps below to obtain the CRA value of the image sensor:

1. Set the register bit field R0x301A[5] = 1.
2. Read the register bit fields R0x31FE [6:4].
3. Determine the CRA value according to Table 33.

Table 33: CRA Value

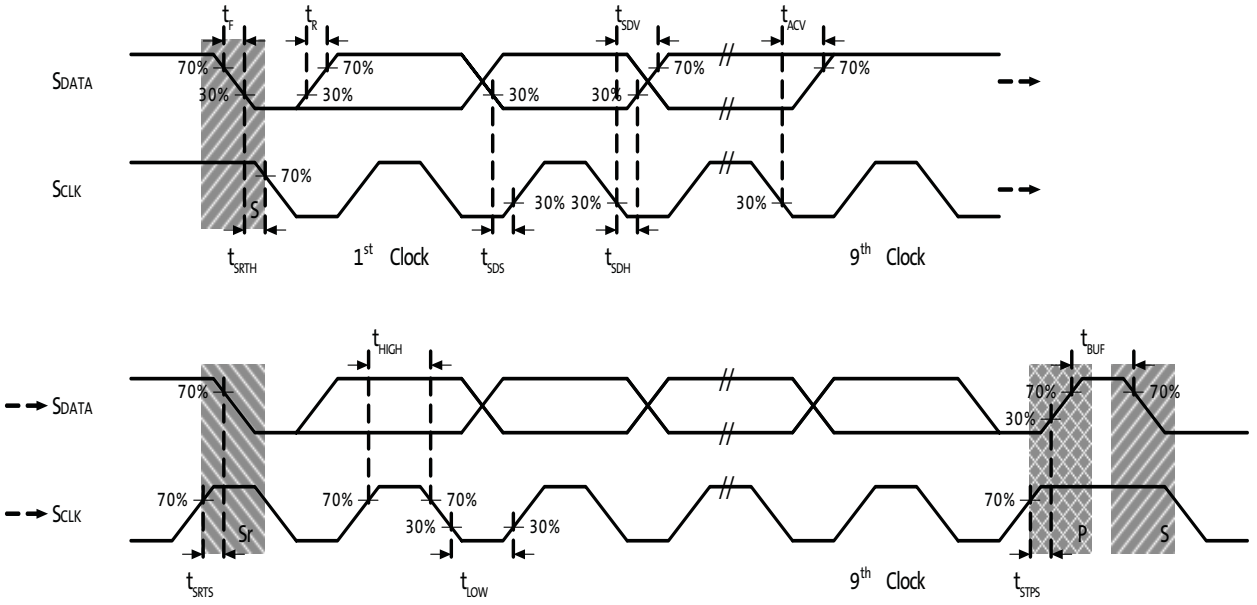
Binary Value of R0x31FE[6:4]	CRA Value
000	25.0

Electrical Characteristics

Two-Wire Serial Register Interface

The electrical characteristics of the two-wire serial register interface (SCLK, SDATA) are shown in Figure 58 and Table 34. The SCLK and SDATA signals feature fail-safe input protection, Schmitt trigger input, and suppression of input pulses of less than 50ns.

Figure 58: Two-Wire Serial Bus Timing Parameters



Note: Read sequence: For an 8-bit READ, read waveforms start after WRITE command and register address are issued.

Table 34: Two-Wire Serial Register Interface Electrical Characteristics

^{f}EXTCLK = 25 MHz; V_{AA} = 2.8V; V_{AA_PIX} = 2.8V; V_{DD_IO} = 1.8V; V_{DD} (digital core) = 1.2V; V_{DD_PLL} = 1.2V; V_{DD_TX} = 1.8V; Output load = 68.5pF; T_J = 55°C

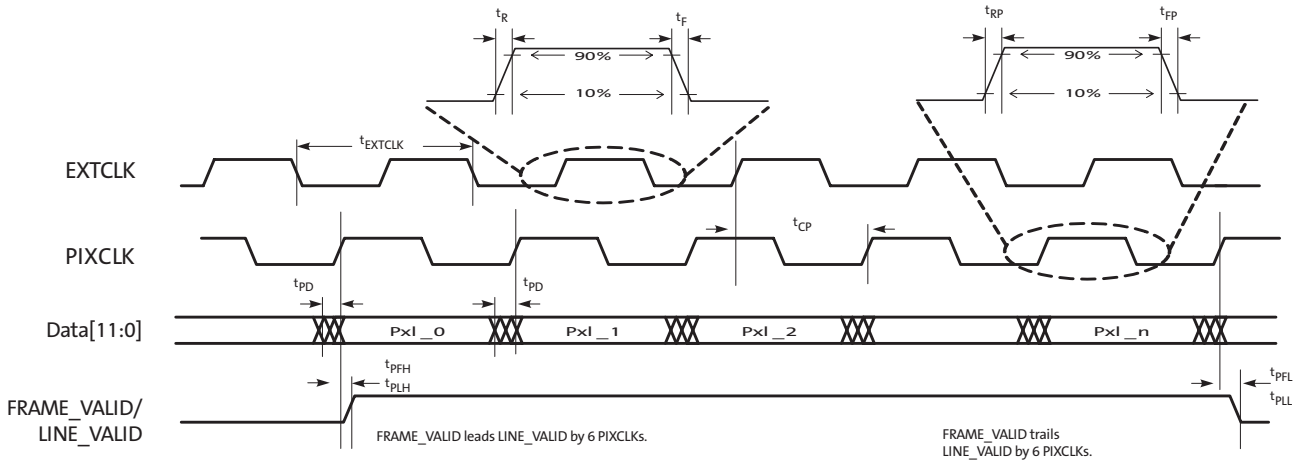
Symbol	Parameter	Condition	Min	Typ	Max	Unit
V_{IL}	Input LOW voltage		-0.5	–	0.3 x V_{DD_IO}	V
V_{IH}	Input HIGH voltage		0.7 x V_{DD_IO}	–	V_{DD_IO} + 0.5	V
I_{IN}	Input leakage current	No pull up resistor; V_{IN} = V_{DD_IO} or D_{GND}	10	–	14	µA
V_{OL}	Output LOW voltage	At specified 2mA	0.11	–	0.3	V
I_{OL}	Output LOW current	At specified V_{OL} 0.1V	–	–	6	mA
C_{IN}	Input pad capacitance		–	–	6	pF
C_{LOAD}	Load capacitance		–	–	N/A	pF

Table 35: Two-Wire Serial Interface Timing Specifications
V_{DD} = 1.8V; V_{AA} =2.8V; Environment temperature = 55°C

Symbol	Definition	Min	Max	Unit
f_{SCLK}	SCLK Frequency	0	400	KHz
t_{HIGH}	SCLK High Period	0.6	-	µs
t_{LOW}	SCLK Low Period	1.3	-	µs
t_{SRTS}	START Setup Time	0.6	-	µs
t_{SRTH}	START Hold Time	0.6	-	µs
t_{SDS}	Data Setup Time	100	-	ns
t_{SDH}	Data Hold Time	0	0.81	µs
t_{SDV}	Data Valid Time	-	0.9	µs
t_{ACV}	Data Valid Acknowledge Time	-	0.9	µs
tS_{TPS}	STOP Setup Time	0.6	-	µs
tB_{UF}	Bus Free Time between STOP and START	1.3	-	µs
t_R	SCLK and SDATA Rise Time	-	300	ns
t_F	SCLK and SDATA Fall Time	-	300	ns

Note: Max t_{SDH} could be 0.9us but must be less than max of t_{SDV} and t_{ACV} by a transition time

Figure 59: Parallel Data Output Timing Diagram



Note: PLL disabled for ^tCP.

EXTCLK

The electrical characteristics of the EXTCLK input are shown in Table 36. The EXTCLK input supports an AC-coupled sine-wave input clock or a DC-coupled square-wave input clock.

If EXTCLK is AC-coupled to the AR0832 and the clock is stopped, the EXTCLK input to the AR0832 must be driven to ground or to V_{DD_IO}. Failure to do this will result in excessive current consumption within the EXTCLK input receiver.

Table 36: Electrical Characteristics (EXTCLK)
f_{EXTCLK} = 25 MHz; V_{AA} = 2.8V; V_{AA_PIX} = 2.8V; V_{DD_IO} = 1.8V; V_{DD} (digital core) = 1.2V; V_{DD_PLL} = 1.2V; Output load = 68.5pF; T_J = 55°C

Symbol	Parameter	Condition	Min	Typ	Max	Unit
f_{EXTCLK}	Input clock frequency	PLL enabled	6	–	27	MHz
t_R	Input clock rise slew rate	C_{LOAD}<20pF	–	2.883	–	ns
t_F	Input clock fall slew rate	C_{LOAD}<20pF	–	2.687	–	ns
V_{IN_AC}	Input clock minimum voltage swing (AC coupled)	–	0.5	–	–	V (p-p)
V_{IN_DC}	Input clock maximum voltage swing (DC coupled)	–	–	–	V_{DD_IO} + 0.5	V
$f_{CLKMAX(AC)}$	Input clock signalling frequency (low amplitude)	V_{IN} = V_{IN_AC} (MIN)	–	–	25	MHz
$f_{CLKMAX((dc)}$	Input clock signalling frequency (full amplitude)	V_{IN} = V_{DD_IO}	–	–	48	MHz
	Clock duty cycle	–	45	50	55	%
t_{JITTER}	Input clock jitter	cycle-to-cycle	–	545	600	ps
t_{LOCK}	PLL VCO lock time	–	–	0.2	1	ms
C_{IN}	Input pad capacitance	–	–	6	–	pF
I_{IH}	Input HIGH leakage current	–	0	–	10	µA
V_{IH}	Input HIGH voltage		0.65 x V_{DD_IO}		V_{DD_IO} + 0.5V	V
V_{IL}	Input LOW voltage		–0.5		0.3 x V_{DD_IO}	V

Parallel Pixel Data Interface

The electrical characteristics of the parallel pixel data interface (FV, LV, DOUT[11:0], PIXCLK, SHUTTER, and FLASH outputs) are shown in Table 37.

Table 37: Electrical Characteristics (Parallel Pixel Data Interface)
MCLK = 25 MHz, Slew rate= 4, Ambient temperature; VAA =2.8V, VAA_PIX = 2.8V; VDD (digital core) = 1.2V; VDD_PLL =1.2V; Output load <20pF

Symbol	Parameter	Condition	Min	Typ	Max	Unit
	Output pin slew (rising)	Default slew rate register settings, C_{LOAD} =25pF, 100 MHz PIXCLK	0.49	1.39	–	V/ns
	Output pin slew (falling)	Default slew rate register settings, C_{LOAD} =25pF, 100 MHz PIXCLK	0.57	1.52	–	V/ns
t_{PD}	PIXCLK to data valid	100 MHz PIXCLK	4.7	5.284	5.8	ns
t_{PIXCLK}	PIXCLK frequency	Default	6	48	100	MHz
t_{PFH}	PIXCLK to FV HIGH	100 MHz PIXCLK	4.5	5.05	5.55	ns
t_{PLH}	PIXCLK to LV HIGH	100 MHz PIXCLK	4.5	5	5.5	ns
t_{PFL}	PIXCLK to FV LOW	100 MHz PIXCLK	4.1	4.52	4.97	ns
t_{PLL}	PIXCLK to LV LOW	100 MHz PIXCLK	4.35	4.84	5.32	ns

Figure 60: Fall Slew Rates (Cap Load = 25pF)

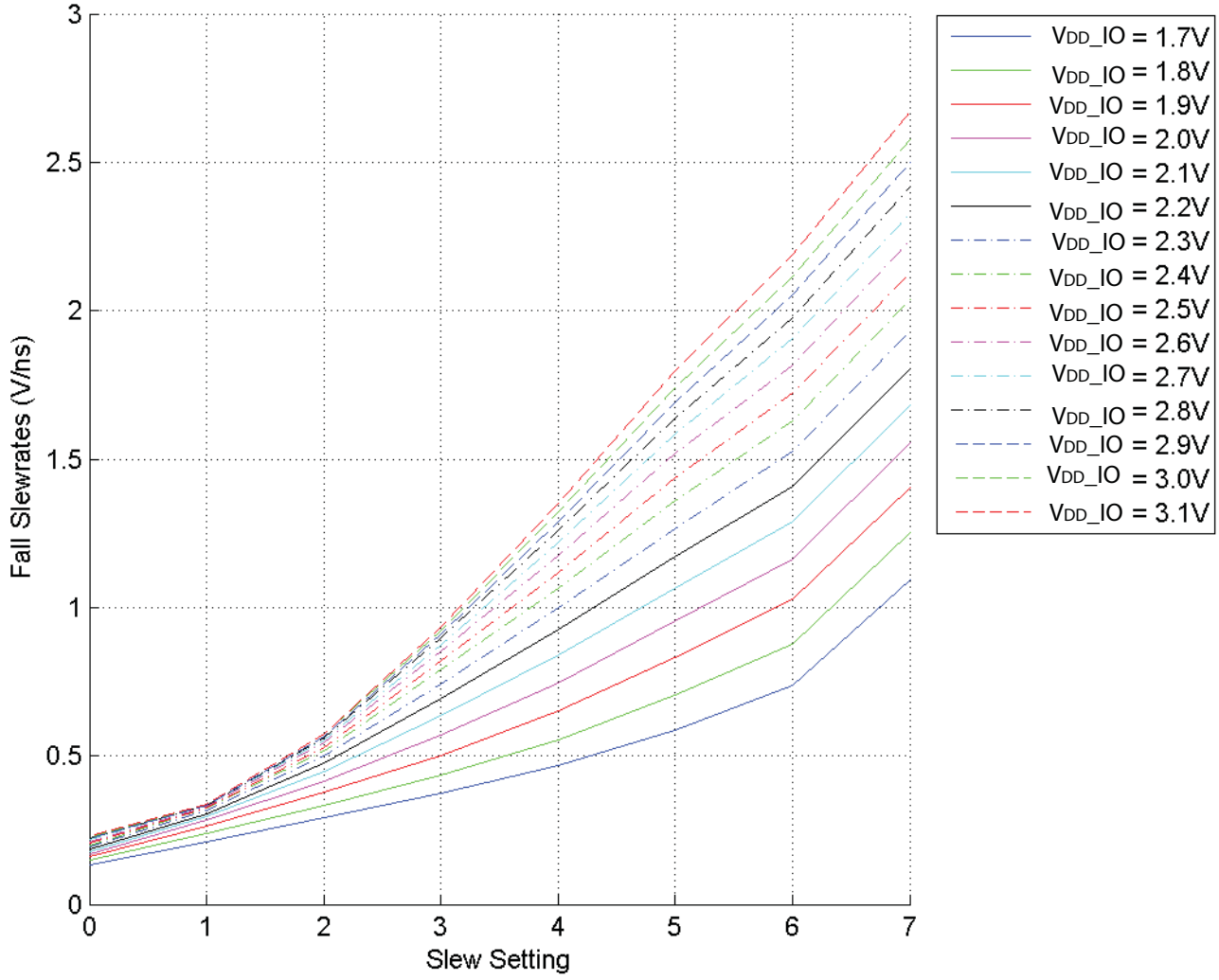
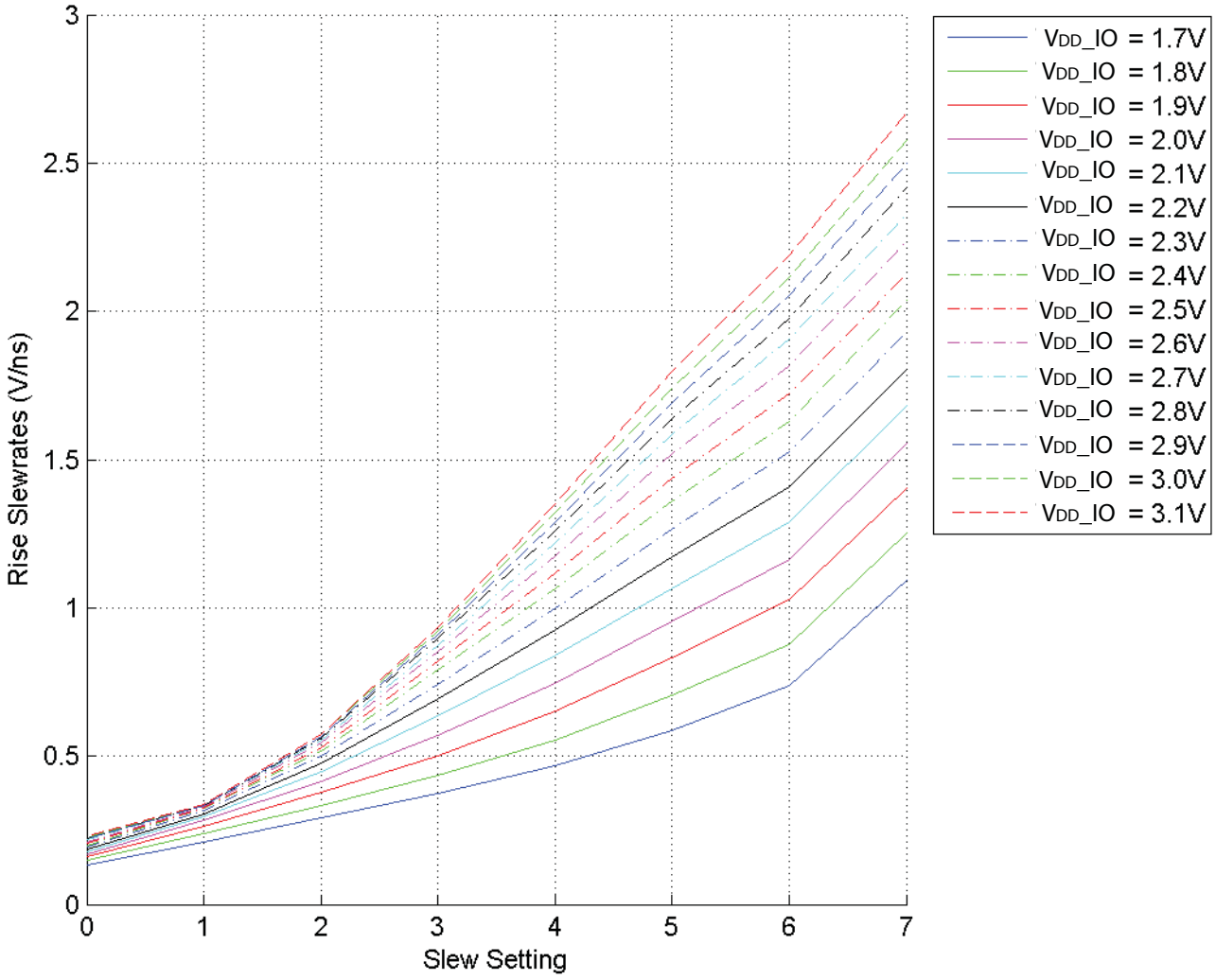


Figure 61: Rise Slew Rates (Cap Load = 25pF)



Serial Pixel Data Interface

The electrical characteristics of the serial pixel data interface (CLK_P, CLK_N,DATA0_P, DATA1_P, DATA0_N, and DATA1_N) are shown in Table 38.

To operate the serial pixel data interface within the electrical limits of the CSI-2 and CCP2 specifications, V_{DD_IO} (I/O digital voltage) is restricted to operate in the range 1.7–1.9V.

Table 38: Electrical Characteristics (Serial MIPI Pixel Data Interface)

Symbol	Parameter	Min	Typ	Max	Unit
V_{OD}	High speed transmit differential voltage	140	–	270	mV
V_{CMTX}	High speed transmit static common-mode voltage	150	–	250	mV
ΔV_{OD}	V_{OD} mismatch when output is Differential-1 or Differential-0		<10		mV
Z_{OS}	Single ended output impedance	40	–	62.5	Ω
ΔZ_{OS}	Single ended output impedance mismatch		<14		%
$\Delta V_{CMTX(L,F)}$	Common-level variation between 50–450 MHz		<25		mV
^{t}R	Rise time (20–80%)	150	–	321	ps
^{t}F	Fall time (20–80%)	150	–	321	ps
V_{OL}	Output LOW level		<50		mV
V_{OH}	Output HIGH level	1.08	–	1.3	V
Z_{OLP}	Output impedance of low power parameter	110	–	–	Ω
T_{RLP}	15–85% rise time	–	–	25	ns
T_{FLP}	15–85% fall time	–	–	25	ns
Δv/Δdtsr	Slew rate (C_{LOAD} = 20–70pF)	30	–	–	mV/ns

Control Interface

The electrical characteristics of the control interface (RESET_BAR, TEST, GPI0, GPI1, GPI2, and GPI3) are shown in Table 39.

Table 39: DC Electrical Characteristics (Control Interface)

^{f}EXTCLK = 25 MHz; V_{AA} = 2.8V; V_{AA_PIX} = 2.8V; V_{DD_IO} = 1.8V; V_{DD} (DIGITAL CORE) = 1.2V; V_{DD_PLL} = 1.2V; Output load = 68.5pF; T_J = 55°C

Symbol	Parameter	Condition	Min	Typ	Max	Unit
V_{IH}	Input HIGH voltage		0.65 x V_{DD_IO}	–	V_{DD_IO} + 0.5	V
V_{IL}	Input LOW voltage		–0.5	–	V_{DD_IO} x 0.3	V
I_{IN}	Input leakage current	No pull-up resistor; V_{IN} = V_{DD_IO} or D_{GND}	–	–	10	µA
C_{IN}	Input pad capacitance		–	6	–	pF

Operating Voltages

V_{AA} and V_{AA_PIX} must be at the same potential for correct operation of the AR0832.

Table 40: DC Electrical Definitions and Characteristics (Using Internal Regulator)

f_{EXTCLK} = 25 MHz; V_{AA} = 2.8V; V_{AA_PIX} = 2.8V; V_{DD_IO} = 1.8V; V_{DD} (DIGITAL CORE) = 1.2V; V_{DD_PLL} = 1.2V; Output load = 68.5pF; T_J = 55°C; Mode = Full Resolution (3264x2488); Frame rate = 15 fps

Symbol	Parameter	Condition	Min	Typ	Max	Unit
V_{DD_TX}	PHY digital voltage		1.7	1.8	1.95	V
V_{DD_IO}	I/O digital voltage		1.7	1.8	1.95	V
V_{AA}	Analog voltage		2.65	2.8	3.1	V
V_{AA_PIX}	Pixel supply voltage		2.65	2.8	3.1	V
I_{AA}	Analog current		95	100	105	mA
I_{AA_PIX}	Pixel supply current		5	6	7	mA
I_{DD_TX}	PHY digital operating current		0	0	0	mA
I_{DD_IO}	I/O digital current		0.03	0.0425	0.05	mA
I_{DD} (V_{DD} + PLL)	Digital current		60	65	70	mA
	Hard Standby (No clock)	Analog		10	15	µA
		IO (V_{DD_IO})		15	25	µA
		Digital (V_{DD})		3.5	15	µA
	Hard Standby (With clock)	Analog		10	15	µA
		IO (V_{DD_IO})		15	25	µA
		Digital (V_{DD})		3.5	15	µA
	Soft Standby (No clock)	Analog		10	15	µA
		IO (V_{DD_IO})		15	25	µA
		Digital (V_{DD})		1.5	8	mA
	Soft Standby (With clock)	Analog		10	25	µA
		IO (V_{DD_IO})		15	25	µA
		Digital (V_{DD})		1.5	8	mA

Table 41: DC Electrical Definitions and Characteristics (Using Exernal Regulator)

f_{EXTCLK} = 25 MHz; V_{AA} = 2.8V; V_{AA_PIX} = 2.8V; V_{DD_IO} = 1.8V; V_{DD} (DIGITAL CORE) = 1.2V; V_{DD_PLL} = 1.2V; Output load = 68.5pF; T_J = 55°C; Mode = Full Resolution (3264x2488); Frame rate = 15 fps

Symbol	Parameter	Condition	Min	Typ	Max	Unit
V_{DD_Tx}	PHY digital voltage		1.7	1.8	1.95	V
V_{DD_IO}	I/O digital voltage		1.7	1.8	1.95	V
V_{AA}	Analog voltage		2.65	2.8	3.1	V
V_{AA_PIX}	Pixel supply voltage		2.65	2.8	3.1	V
I_{AA}	Analog current		95	100	105	mA
I_{AA_PIX}	Pixel supply current		5	6	7	mA
I_{DD_Tx}	PHY digital operating current		0	0	0	mA
I_{DD_IO}	I/O digital current		0.03	0.0425	0.05	mA

Table 41: DC Electrical Definitions and Characteristics (Using Exernal Regulator)

f_{EXTCLK} = 25 MHz; V_{AA} = 2.8V; V_{AA_PIX} = 2.8V; V_{DD_IO} = 1.8V; V_{DD} (DIGITAL CORE) = 1.2V; V_{DD_PLL} = 1.2V; Output load = 68.5pF; T_J = 55°C; Mode = Full Resolution (3264x2488); Frame rate = 15 fps

Symbol	Parameter	Condition	Min	Typ	Max	Unit
$I_{DD}(V_{DD}+PLL)$	Digital current		60	65	70	mA
	Hard Standby (No clock)	Analog		15	25	µA
		IO (V_{DD_IO})		15	25	µA
		Digital (V_{DD})		2	8	mA
	Hard Standby (With clock)	Analog		15	25	µA
		IO (V_{DD_IO})		15	25	µA
		Digital (V_{DD})		2.1	8	mA
	Soft Standby (No clock)	Analog		10	15	µA
		IO (V_{DD_IO})		15	25	µA
		Digital (V_{DD})		1.5	8	mA
	Soft Standby (With clock)	Analog		10	25	µA
		IO (V_{DD_IO})		15	25	µA
		Digital (V_{DD})		1.5	8	mA

Note: The termination on the VDDIO power supply is 100 Ω and this value is used to get the above current numbers.

Absolute Maximum Ratings

Table 42: Absolute Maximum Values

f_{EXTCLK} = 25 MHz; V_{AA} = 3.1V; V_{AA_PIX} = 3.1V; V_{DD_IO} = 1.95V; V_{DD} (DIGITAL CORE) = 1.2V; V_{DD_PLL} = 1.2V; Output load = 68.5pF; T_J = 70°C; Mode = Full Resolution (3264x2488); Frame rate = 15 fps

Symbol	Parameter	Condition	Min	Typ	Max	Unit
V_{DD1V8_MAX}	Core digital voltage		1.7	1.8	1.95	V
$V_{DD\_IO\_MAX}$	I/O digital voltage	V_{DD_IO} = 1.8V	1.7	1.8	1.95	V
V_{AA}	Analog voltage		2.7	2.8	3.1	V
V_{AA_PIX}	Pixel supply voltage		2.7	2.8	3.1	V
I_{DD} (V_{DD} + PLL)	Digital operating current	Worst case current	–	–	95	mA
I_{DD_IO}	I/O digital operating current	Worst case current (MIPI)	–	–	1	mA
I_{AA}	Analog operating current	Worst case current	–	–	128	mA
I_{AA_PIX}	Pixel supply current	Worst case current	–	–	12	mA
T_{OP}	Operating temperature	Measure at junction	-30	–	70	°C
T_{STG}	Storage temperature		-40	–	85	°C

Caution **Stresses greater than those listed in Table 43 may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect reliability. This is a stress rating only, and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.**

Table 43: Absolute Max Voltages

Symbol	Parameter	Condition	Min	Max	Unit
V_{DD1V8_MAX}	Core digital voltage		-0.3	2.1	V
$V_{DD\_IO\_MAX}$	I/O digital voltage		-0.3	3.5	V
V_{AA}	Analog voltage		-0.3	3.5	V
V_{AA_PIX}	Pixel supply voltage		-0.3	3.5	V

SMIA and MIPI Specification Reference

The sensor design and this documentation is based on the following reference documents:

1. SMIA Specifications:
   - Functional Specification:
     SMIA 1.0 Part 1: Functional Specification (Version 1.0 dated 30 June 2004)
     SMIA 1.0 Part 1: Functional Specification ECR0001 (Version 1.0 dated 11 Feb 2005)
   - Electrical Specification:
     SMIA 1.0 Part 2: CCP2 Specification (Version 1.0 dated 30 June 2004)
     SMIA 1.0 Part 2: CCP2 Specification ECR0001 (Version 1.0 dated 11 Feb 2005)
2. MIPI Specifications:
   - MIPI Alliance Standard for CSI-2 version 1.0
   - MIPI Alliance Standard for D-PHY version 1.0

Revision History

Rev. J 10/25/11

- Added IO Conditions to Table 40, “DC Electrical Definitions and Characteristics (Using Internal Regulator),” on page 92 and Table 41, “DC Electrical Definitions and Characteristics (Using Exernal Regulator),” on page 92
- Changed Hard Standby Digital Typ and Max values in Table 41, “DC Electrical Definitions and Characteristics (Using Exernal Regulator),” on page 92

Rev. I 10/20/11

- Added Table 41, “DC Electrical Definitions and Characteristics (Using Exernal Regulator),” on page 92

Rev. H 9/27/11

- Updated Figure 50: “Power-Up Sequence without Internal Regulator,” on page 77
- Added “Power-Down Sequence Using Internal Regulator” on page 78
- Updated Table 31, “Power-Down Timing Without Internal Regulator,” on page 80
- Updated Figure 58: “Two-Wire Serial Bus Timing Parameters,” on page 85
- Updated Table 35, “Two-Wire Serial Interface Timing Specifications,” on page 86
- Updated Table 36, “Electrical Characteristics (EXTCLK),” on page 87
- Updated Table 39, “DC Electrical Characteristics (Control Interface),” on page 91
- Updated Table 40, “DC Electrical Definitions and Characteristics (Using Internal Regulator),” on page 92

Rev. G 8/23/11

- Added “Power-Up Sequence Using External Regulator” on page 77

Rev. F 6/6/11

- Updated to Production
- Updated Table 27, “Test Patterns,” on page 68

Rev. E 4/13/11

- Updated Table 40, “DC Electrical Definitions and Characteristics (Using Internal Regulator),” on page 92

Rev. D 3/30/11

- Updated Table 1, “Key Performance Parameters,” on page 1
- Updated Table 40, “DC Electrical Definitions and Characteristics (Using Internal Regulator),” on page 92

Rev. C 3/1/11

- Updated Table 1, “Key Performance Parameters,” on page 1
- Updated Table 38, “Electrical Characteristics (Serial MIPI Pixel Data Interface),” on page 91
- Updated “SMIA and MIPI Specification Reference” on page 94

10 Eunos Road 8 13-40, Singapore Post Center, Singapore 408600 prodmktg@aptina.com www.aptina.com

Rev. B 1/28/11

- Updated Table 1, “Key Performance Parameters,” on page 1
- Updated OTPM in “Features” on page 1 and in “One-Time Programmable Memory (OTPM)” on page 42
- Updated Figure 3: “Parallel/MIPI Typical Connections (Using Internal Regulator for Vdd and Sensor-Connected PLL),” on page 12
- Updated Figure 4: “Parallel/MIPI Typical Connections (Not Using Internal Regulator),” on page 13
- Updated “Signal Descriptions” on page 14
- Updated “Serial Pixel Data Interface” on page 15
- Updated “Output Size Restrictions” on page 27
- Updated “Power-On Reset Sequence” on page 34
- Updated Table 16, “Default Settings and Range of Values for Divider/Multiplier Registers,” on page 37
- Updated Figure 20: “Illustration of Resampling Operation,” on page 41
- Updated Figure 55: “VCM Driver Typical Diagram,” on page 81
- Updated Table 36, “Electrical Characteristics (EXTCLK),” on page 87
- Updated Table 37, “Electrical Characteristics (Parallel Pixel Data Interface),” on page 88
- Updated Table 40, “DC Electrical Definitions and Characteristics (Using Internal Regulator),” on page 92
- Changed RESET_N to RESET_BAR globally.
- Changed PDF number in footer to reflect change in file location

Rev. A, Preliminary 12/1/10

- Initial release