1/3.2-Inch 12Mp CMOS Digital Image Sensor

AR1230 Data Sheet, Rev. B

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

Features

- 12Mp CMOS sensor with advanced 1.1um pixel BSI technology
- Data interfaces: two- and four-lane serial mobile industry processor interface (MIPI)
- Bit-depth compression available for MIPI Interface: 10-8 and 10-6 to enable lower bandwidth receivers for full frame rate applications
- 3D synchronization controls to enable stereo video capture
- Interlaced multi-exposure readout enabling High Dynamic Range (HDR) still and video applications
- 6Kb one-time programmable memory (OTPM) for storing shading correction coefficients and module information
- On-chip 10-bit VCM driver
- 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
- On-die phase-locked loop (PLL) oscillator
- Bayer pattern down-size scaler
- Superior low-light performance
- Low dark current
- Simple two-wire serial interface
- On-chip lens shading correction
- On-chip dynamic defect correction
- Support for external mechanical shutter
- Support for external LED or Xenon Flash
- Extended Flash duration that is up to start of frame readout

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 pixels		4000H x 3000V
Pixel size		1.1µm Back Side Illuminated (BSI)
Chief ray angle (CRA)		29°
Die size		6.80mm x 6.42mm (43.65mm2)
Input clock frequency		6 - 48 MHz
Interface		4-lane MIPI (2-lane supported); Max data rate: 750Mbps/lane
Subsampling modes		X - 2x, 4x
		Y - 2x, 4x
ADC resolution		10 bits, on-die
Analog gain		1x, 2x, 3x, 4x, 6x, 8x
High quality Bayer Scalar		Adjustable scaling up to 1/6x scaling
Temperature sensor		10-bit, single instance on chip, controlled by two-wire serial I/F
Compression		DPCM: 10-8-10, 10-6-10
VCM AF driver		10-bit resolution
3D support		Frame rate and exposure synchronization
Supply voltage	V_{AA}, V_{AA_PIX}	2.6 - 3.1V (2.8V nominal)
	V_{DD_IO}	1.7 - 1.95V (1.8V nominal)
	V_{DD}	1.14 - 1.3V (1.2V nominal)
	V_{DD_SLVS}	1.14 - 1.3V (1.2V nominal)
	V_{PP}	6.0 - 7.0V (6.5V nominal)
Power consumption		415mW at 12Mp 20fps
Responsivity		0.88 V/lux-sec
SNRMAX		33.9 dB
Dynamic Range		63.7 dB
Operating Temperature		-30°C to +70°C

Table 2: MIPI 4-lane Output Modes and Power Consumption

Modes	Resolution	Mode	HFOV	FPS	Power Consumption [mW]
4:3 Snapshot Mode					
Full Resolution 4:3	4000 x 3000	12Mp Full Mode	100%	20	415
VGA (LP/HS)	640 x 480	Subsampling + Scaling	100%	90	377
QVGA (HS)	320 x 240	Subsampling + Scaling	100%	90	377
16:9 Video Mode 30 FPS					
Full Resolution 16:9	4000 x 2250	9M Full Mode	100%	30	438
1080p (HQ)	1920 x 1080	Crop +Scaling	96%	30	421
720p (HQ)	1280 x 720	Scaling	100%	30	423
16:9 Video Mode 60 FPS					
1080p (LP/HS)	1920 x 1080	Crop + Subsampling	96%	60	396
720p (LP/HS)	1280 x 720	Crop + Subsampling + Scaling	96%	60	392

Note: HQ: High Quality Mode
LP: Low Power Mode
HS: High Speed Mode

Ordering Information

Table 3: Available Part Numbers

Part Number	Description
AR1230MBSC29SMD20	Bare die

Table of Contents

List of Figures

List of Tables

General Description

The Aptina AR1230 is a 1/3.2-inch BSI (back side illuminated) CMOS active-pixel digital image sensor with a pixel array of 4000H x 3000V (4016H x 3016V 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.

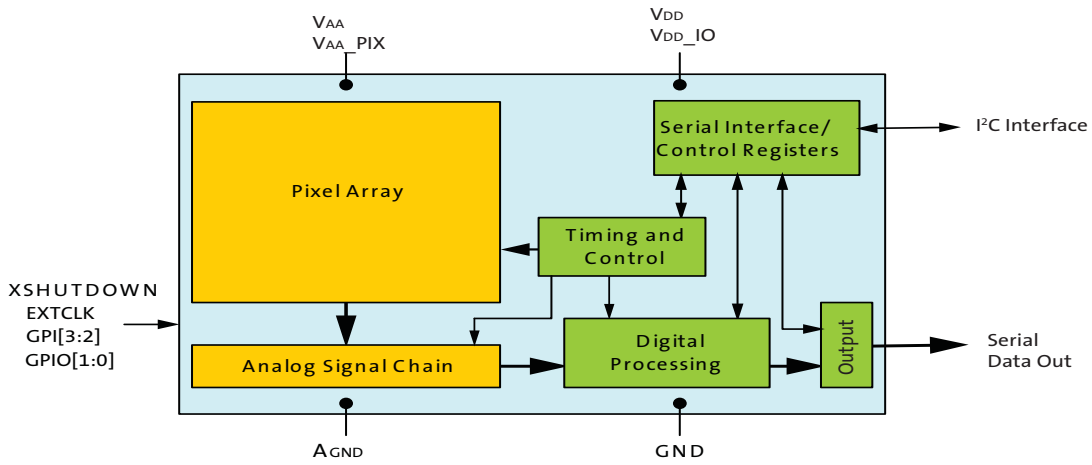
The AR1230 digital image sensor features Aptina's breakthrough low-noise CMOS imaging technology that achieves near-CCD image quality (based on signal-to-noise ratio and low-light sensitivity) while maintaining the inherent size, cost, and integration advantages of CMOS.

The AR1230 sensor can generate full resolution image at up to 20 frames per second (fps). An on-chip analog-to-digital converter (ADC) generates a 10-bit value for each pixel.

Functional Overview

The AR1230 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 48 MHz. The maximum data rate is 750 Mbps per lane. A block diagram of the sensor is shown in Figure 1.

Figure 1: Block Diagram



The core of the sensor is a 12Mp 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 1 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 serial pixel data stream qualified by a 4-lane MIPI output clock.
- 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.

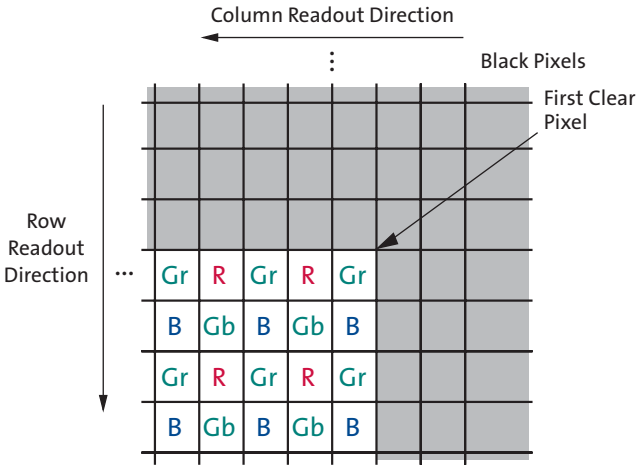
Programmable slew rates are also available to reduce the effect of electromagnetic interference from the output interface.

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

The AR1230 supports MIPI serial output which can be configured in 2- and 4-lanes. There is no parallel data output port. Typical configurations are shown in Figure 3 on page 4. These operating modes are described in “Control of the Signal Interface” on page 27.

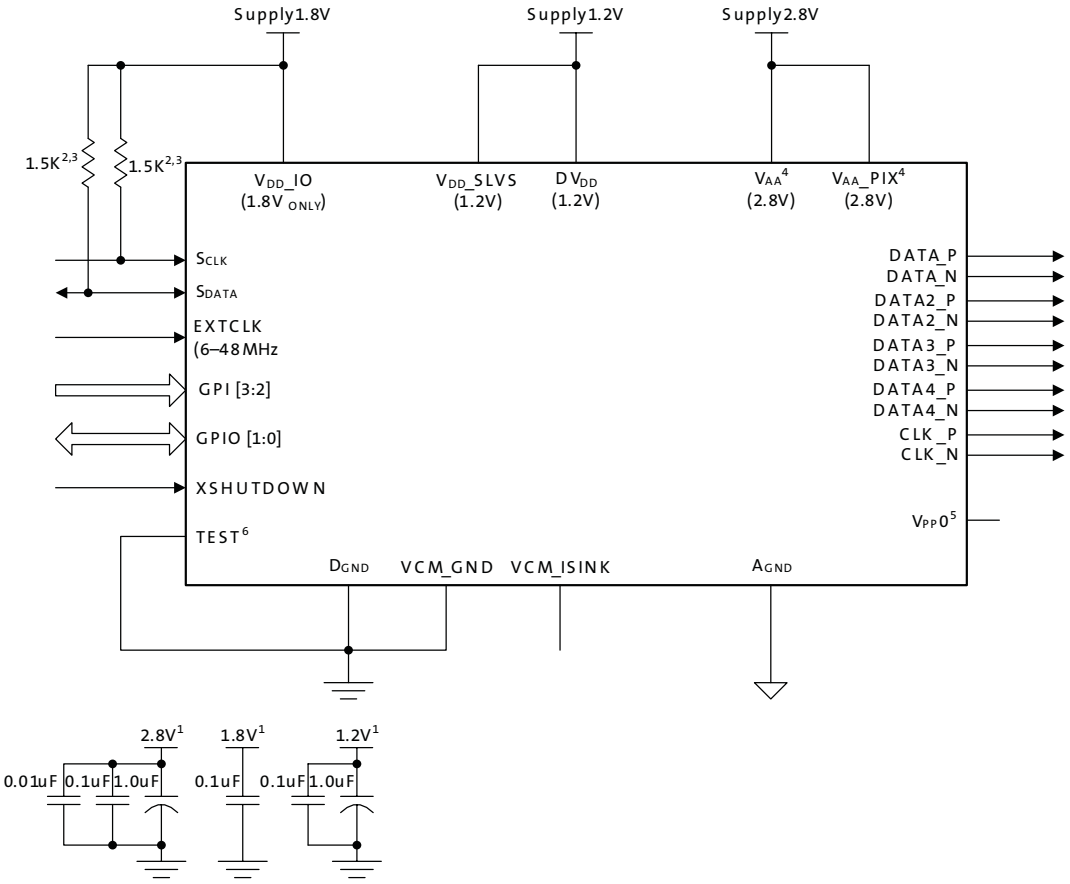
For low-noise operation, the AR1230 requires separate power supplies for analog and digital. Incoming digital and analog ground conductors can be tied together next to the die. Both power supply rails should be decoupled from ground using capacitors as close as possible to the die.

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

Typical Connections

Figure 3 shows the typical AR1230 connections.

Figure 3: MIPI Typical Connections



1. All power supplies must be adequately decoupled. The order of preference is as follows: 2.8V supply- 1.0µF, 0.1µF and then 0.01µF; 1.2V supply - 1.0µF and 0.1µF; 1.8V supply - 0.1µF.
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 drivers a valid logic level on SCLK at all times.
4. VAA and VAA_PIX must be tied together.
5. VPP0, 6-7V, is sued for programming OTPM. This pad is left unconnected during normal operation.
6. TEST pin must be tied to DGND.

Signal Descriptions

Table 4 provides signal descriptions for AR1230 die. For pad location and aperture information, refer to the AR1230 die data sheet.

Table 1: Signal Descriptions

Pad Name	Pad Type	Description
EXTCLK	Input	Master clock input, 6–48 MHz.
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[3:2]	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. These pads can be configured to provide hardware control of: GPI[2]: SADDR, Trigger signal for Slave mode and Standby GPI[3]: 3D daisy chain communication input, all options in GPI[2]Aptina recommends that unused GPI pins be tied to DGND, but can also be left floating.
GPIO[1:0]	Input/ Output	General purpose inputs and outputs. After reset, these pads are not powered-down since its default use is as output. These pads can be configured to provide hardware control of: GPIO[0]: Flash output (default), all input options in GPI[2] GPIO[1]: Shutter output (default), 3-D daisy chain communication output and all input options in GPI[2] Aptina recommends that unused GPIO pins be tied to DGND, but can also be left floating.
TEST	Input	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 1st lane (positive).
DATA_N	Output	Differential MIPI (sub-LVDS) serial data 1st lane (negative).
DATA2_P	Output	Differential MIPI (sub-LVDS) serial data 2nd lane (positive).
DATA2_N	Output	Differential MIPI (sub-LVDS) serial data 2nd lane (negative).
DATA3_P	Output	Differential MIPI (sub-LVDS) serial data 3rd lane (positive).
DATA3_N	Output	Differential MIPI (sub-LVDS) serial data 3rd lane (negative).
DATA4_P	Output	Differential MIPI (sub-LVDS) serial data 4th lane (positive).
DATA4_N	Output	Differential MIPI (sub-LVDS) serial data 4th lane (negative).
CLK_P	Output	Differential MIPI (sub-LVDS) serial clock/strobe (positive).
CLK_N	Output	Differential MIPI (sub-LVDS) serial clock/strobe (negative).
VPP0	Supply	Power supply used to program one-time programmable (OTP) memory. Disconnect pad when programming or when feature is not used.
VDD_SLVS	Supply	Digital PHY power supply. Digital power supply for the serial interface (1.2V).
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.
VCM_ISINK	I/O	Connected to VCM actuator. Can be left floating if not used.
VCM_GND	I/O	Connected to DGND. Tie to DGND if not used.

Output Data Format

Serial Pixel Data Interface

The AR1230 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.

Two-Wire Serial Register Interface

The two-wire serial interface bus enables read/write access to control and status registers within the AR1230.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 AR1230 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 AR1230 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.

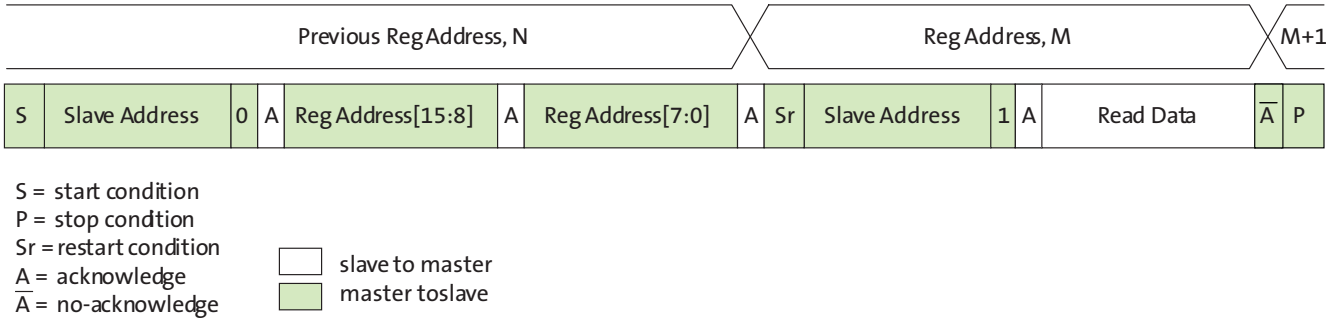
If the request was a WRITE, the master then transfers the 16-bit register address to which the WRITE should take place. This transfer takes place as two 8-bit sequences and the slave sends an acknowledge bit after each sequence to indicate that the byte has been received. The master then transfers the 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 4 on page 9) 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 4 shows how the internal register address maintained by the AR1230 is loaded and incremented as the sequence proceeds.

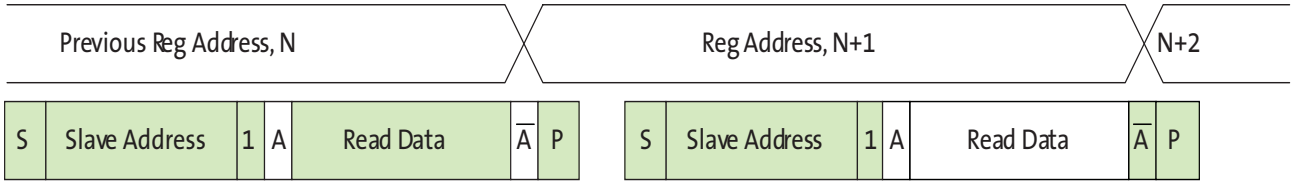
Figure 4: Single READ from Random Location



Single READ from Current Location

This sequence (Figure 5) performs a read using the current value of the AR1230 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.

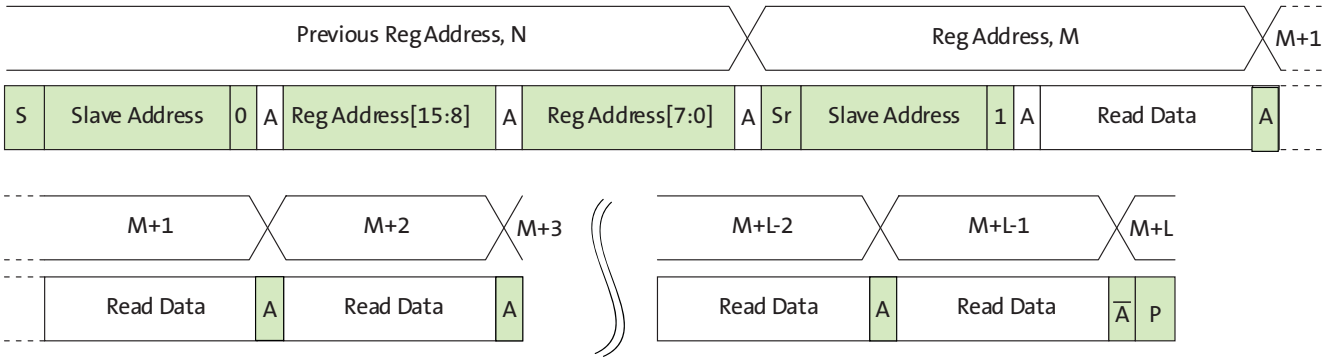
Figure 5: Single READ from Current Location



Sequential READ, Start from Random Location

This sequence (Figure 6) starts in the same way as the single READ from random location (Figure 4). 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.

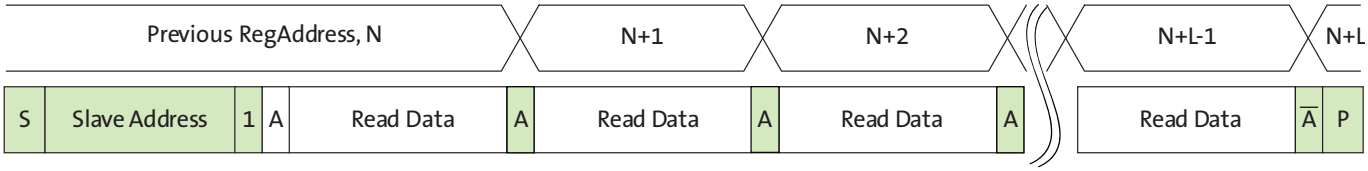
Figure 6: Sequential READ, Start from Random Location



Sequential READ, Start from Current Location

This sequence (Figure 7) starts in the same way as the single READ from current location (Figure 5 on page 9). 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.

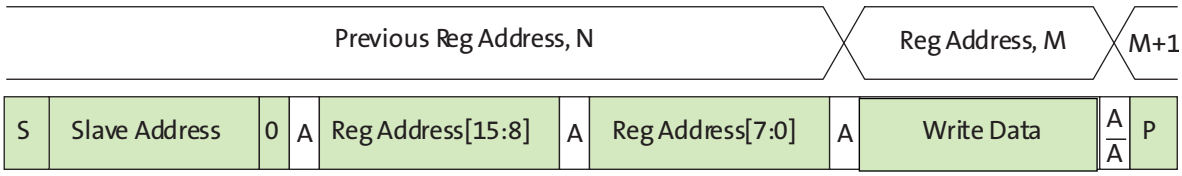
Figure 7: Sequential READ, Start from Current Location



Single WRITE to Random Location

This sequence (Figure 8) 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.

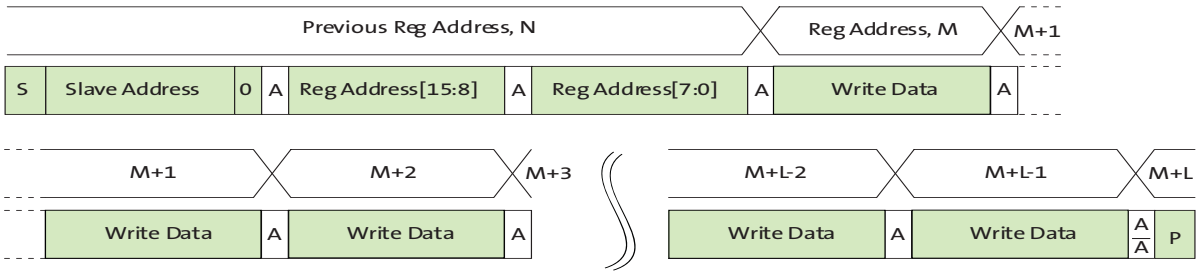
Figure 8: Single WRITE to Random Location



Sequential WRITE, Start at Random Location

This sequence (Figure 9) starts in the same way as the single WRITE to 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 WRITEs until "L" bytes have been written. The WRITE is terminated by the master generating a stop condition.

Figure 9: Sequential WRITE, Start at Random Location



Registers

The AR1230 provides a 16-bit register address space accessed through a serial interface (“Two-Wire Serial Register Interface” on page 7). Each register location is 8 or 16 bits in size.

The address space is divided into the five major regions shown in Table 5. The remainder of this section describes these registers in detail.

Table 2: 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 AR1230 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 AR1230 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 0x0054. This means that a read from address 0x0054 would return 0x00, and a read from address 0x0001 would return 0x00. When reading this register as two 8-bit transfers on the serial interface, the 0x00 will appear on the serial interface first, followed by the 0x54.

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 6.

Table 3: 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 AR1230 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 AR1230 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 14.
- 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.

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 4: 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 8 shows a list of programming rules that must be adhered to for correct operation of the AR1230. It is recommended that these rules are encoded into the device driver stack—either implicitly or explicitly.

Table 1: 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 1: Programming Rules (continued)

Parameter	Minimum Value	Maximum Value
x_output_size (must be even number – this is enforced in hardware)	320	3016
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.

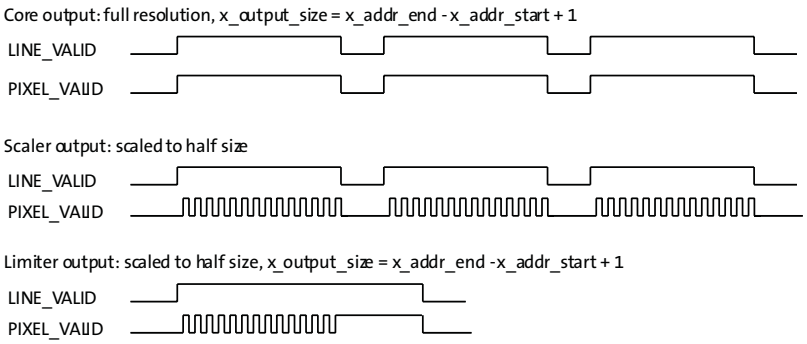
When the serial pixel data path is in use, there is an additional restriction that x_output_size must be small enough such that the output row time (set by x_output_size, the framing and CRC overhead of 12 bytes and the output clock rate) must be less than the row time of the video array (set by line_length_pck 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 AR1230 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 10 on page 3.

Figure 1: Effect of Limiter on the Data Path



In Figure 10, three different stages in the data path (see “Timing Specifications” on page 37) 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.

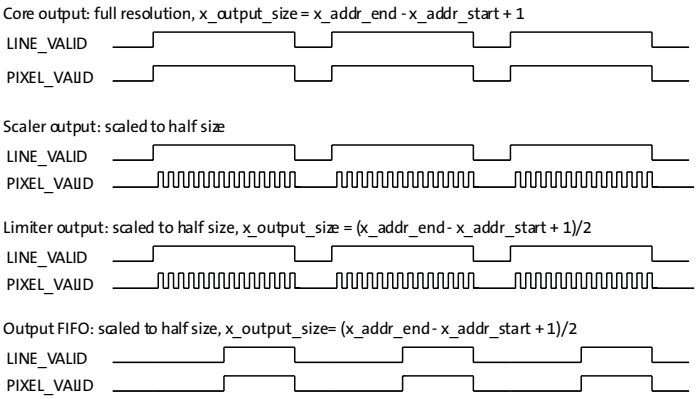
The second stage is the output of the scaler, when the scaler is set to reduce the image size by one-half in each dimension. The effect of the scaler is to combine groups of pixels. Therefore, the row time remains the same, but only half the pixels out of the scaler are valid. This is signalled by transitions 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 AR1230 will cease to generate output frames.

A correct configuration is shown in Figure 11 on page 4, 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 11 on page 4 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 2: 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

Control of the Signal Interface

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

The AR1230 sensor provides a MIPI serial interface for pixel data.

MIPI Serial Pixel Data Interface

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

- DATA_P
- DATA_N
- DATA2_P
- DATA2_N
- DATA3_P
- DATA3_N
- DATA4_P
- DATA4_N
- CLK_P
- CLK_N

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 DATA_P, DATA_N, DATA2_P, DATA2_N, CLK_P, and CLK_N 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.

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:

- 0x0201 – Sensor supports 2-lane MIPI operation
- 0x0202 – Sensor supports 4-lane MIPI operation

System States

The system states of the AR1230 are represented as a state diagram in Figure 12 and described in subsequent sections. The effect of XSHUTDOWN on the system state and the configuration of the PLL in the different states are shown in Table 9 on page 7.

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 9 on page 7.

Figure 3: AR1230 System States

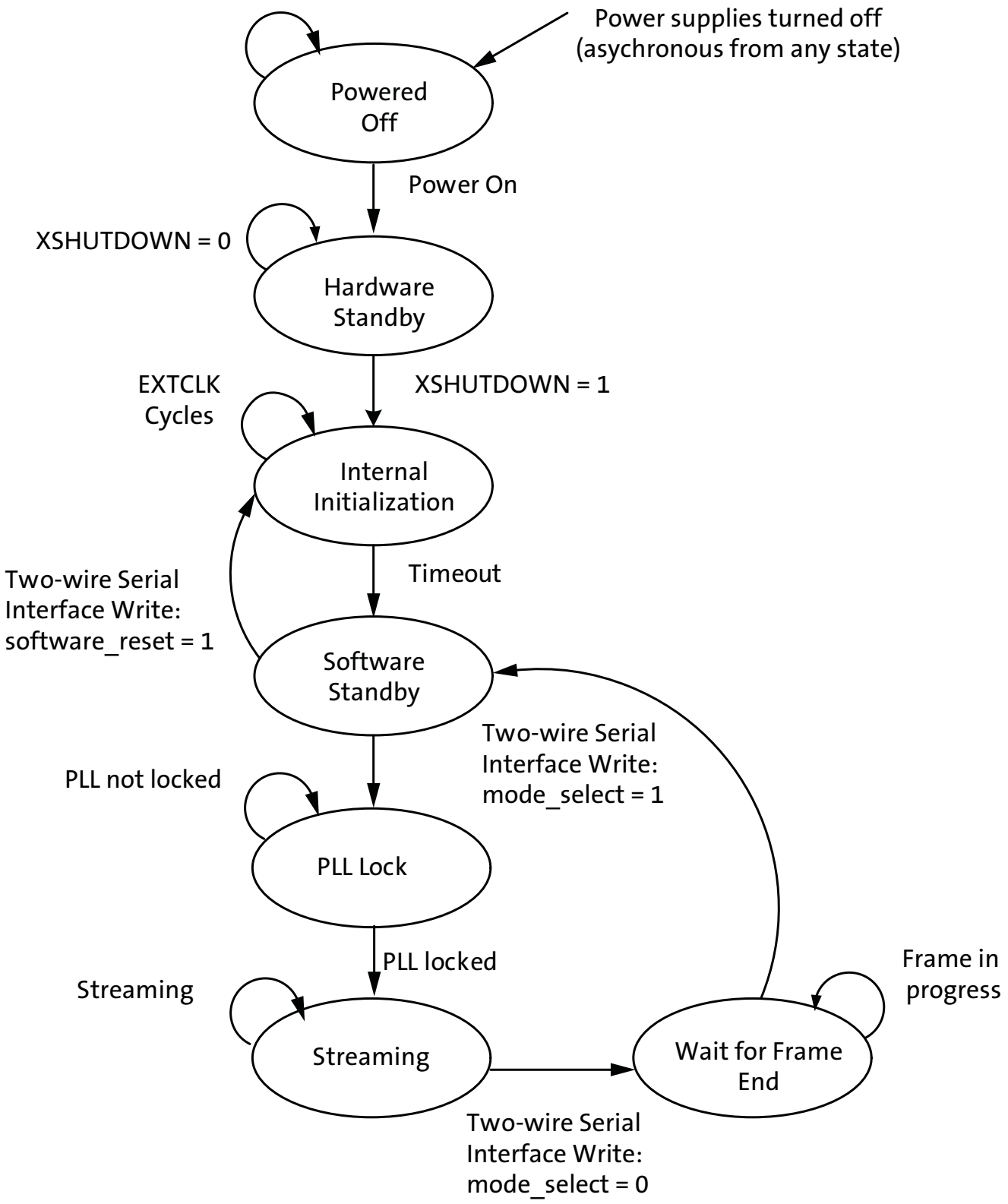


Table 2: XSHUTDOWN and PLL in System States

State	XSHUTDOWN	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		

Soft Reset Sequence

The AR1230 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 10 shows the state of the signal interface during hardware standby (XSHUTDOWN 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 3: Signal State During Reset

Pad Name	Pad Type	Hardware Standby	Software Standby
EXTCLK	Input	Enabled. Must be driven to a valid logic level.	
XSHUTDOWN	Input	Enabled. Must be driven to a valid logic level.	
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.	
GPI[1:0]	I/O	High-Z.	Logic 0.
DATA_P	Output	MIPI: Ultra Low-Power State (ULPS), represented as an LP-00 state on the wire (both wires at 0V).	
DATA_N	Output		
DATA2_P	Output		
DATA2_N	Output		
DATA3_P	Output		
DATA3_N	Output		
DATA4_P	Output		
DATA4_N	Output		
CLK_P	Output		
CLK_N	Output		
GPI[3:2]	Input	Powered down. Can be left disconnected/floating.	
TEST	Input	Must be driven to 0 for normal operation.	

General Purpose Input and Output

The AR1230 provides four general purpose input and output pads, as listed inTable 11. GPIO[1:0] are defined as bidirectional (input and output), GPI[3:2] input only.

After power on reset, GPIO[0] is assigned as an output of Flash signal and GPIO[1] is assigned as an output of Shutter signal. Other two GPI are powered down if not used.

Table 4: General Purpose Input and Output Pad Functions

PIN Names	Functions
GPIO[0]	General Input and one Output a. (Default Output) Flash b. (Input) All options in GPI2
GPIO[1]	General Input and two Output functions a. (Default Output) Shutter b. (Output) 3-D daisy chain communication output c. (Input) all options in GPI2
GPI[2]	General Input a. SADDR, second I^2C device address b. Trigger signal for Slave Mode c. Standby
GPI[3]	General Input a. 3-D daisy chain communication input b. All options in GPI2

Streaming/Standby Control

The AR1230 can be switched between its soft standby and streaming states under pin or register control, as shown in Table 12. Selection of a pin to use for the STANDBY function is described in "General Purpose Input and Output" on page 8. The state diagram for transitions between soft standby and streaming states is shown in Figure 12 on page 6.

Table 5: 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

Clocking

The AR1230 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 13 shows the different clocks and the names of the registers that contain or are used to control their values. Table 13 shows the default setting for each divider/multipler control register and the range of legal values for each divider/multiplier control register.

Figure 4: AR1230 Profile 1/2 Clocking Structure

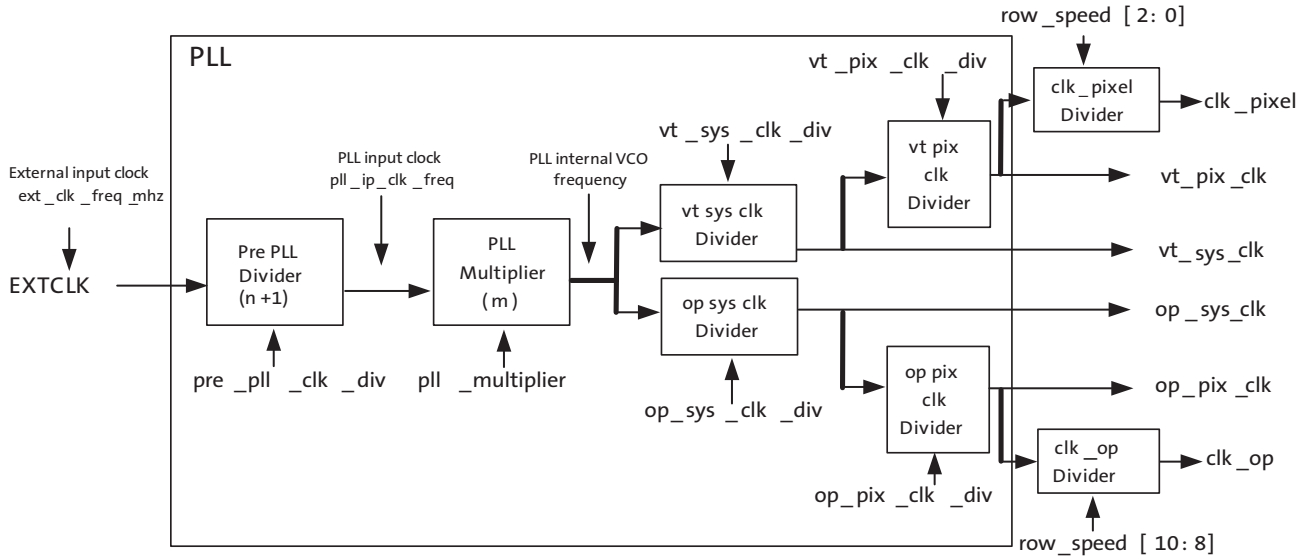


Table 6: Default Settings and Range of Values for Divider/Multiplier Registers

Name	Default	Range
ext_clk_freq_mhz	25	6 – 48MHz
pll_ip_clk_freq_mhz		4 – 24 MHz
PLL internal VCO frequency (pll_op_clk_freq_mhz)		384 – 750 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 – 750 MHz
clk_op		24 – 750 MHz
op_pix_clk_div	10	6, 8, 10
op_sys_clk_div	1	1, 2, 4, 6, 8, 10, 12, 14, 16

Table 6: 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– 750 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 6–48 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 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 AR1230 sensor, vt_pix_clk_div must be set to 5 for profile 0 mode.)

PLL Clocking

The PLL divisors should be programmed while the AR1230 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 AR1230 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 AR1230 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 AR1230 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 AR1230 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:

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

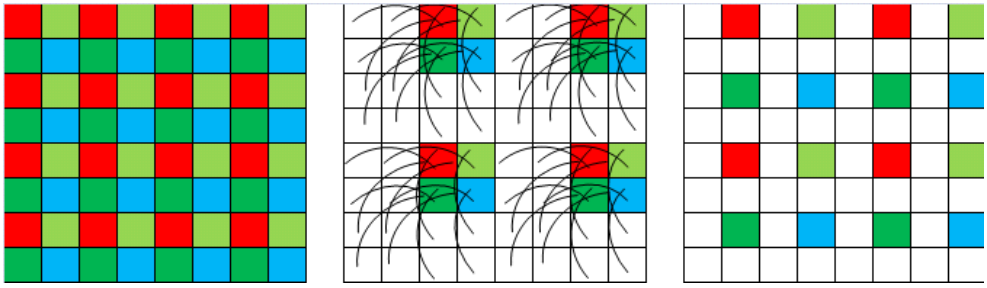
where *P* are the pixel values and *f* is the color dependent correction functions for each color channel.

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

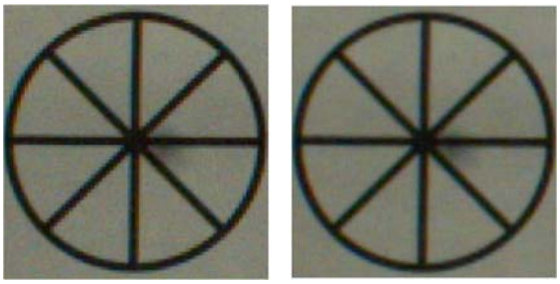
The imaging artifacts found from a 2x2 subsampling will show image artifacts from aliasing. These can be corrected by resampling the sampled pixels in order to filter these artifacts. Figure 14 shows the pixel location resulting from 2x2 subsampling located in the middle and the resulting pixel locations after the Bayer re-sampling function has been applied.

Figure 5: Bayer Resampling



The improvements from using the Bayer resampling feature can be seen in Figure 15. 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 6: Results of Resampling



To enable the Bayer resampling feature:

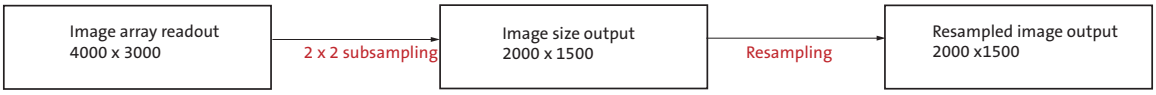
1. Set R0x0400 = 0x0002 // Enable the on-chip scalar.
2. Set R0x306E to 0x90B0 // Configure the on-chip scalar to resample Bayer data.

To disable the Bayer resampling feature:

1. Set R0x0400 = 0x0000 // Enable the on-chip scalar.
2. Set R0x306E to 0x9080 // Configure the on-chip scalar to resample Bayer data.

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

Figure 7: Illustration of Resampling Operation



One-Time Programmable Memory (OTPM)

The AR1230 features 6Kb of one-time programmable memory (OTPM) for storing shading correction coefficients, individual module, and sensor specific information. 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 AR1230 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 AR1230 OTPM follows:

1. Apply power to all the power rails of the sensor (VDD_IO, VDD_SLVS, DVDD, VAA, and VAA_PIX). VAA must be set to 2.8V during the programming process. VPP must initially be floating. All other supplies must be at their nominal voltage.
2. Provide a 12-MHz EXTCLK clock input.
3. Set R0x301A = 0x18, to put the sensor in the soft standby mode.
4. Set R0x3130 = 0xFF01 (timing configuration).
5. Set R0x304C[15:8] = Record type (E.G. 0x30).
6. Set R0x304C[7:0] = Length of the record which is the number of OTPM data registers that are filled in.
7. Set R0x3054[9] = 0 to ensure that error checking and correction is enabled.
8. Write data into all the OTPM data registers: R0x3800-R0x39FE.
9. Ramp up Vpp to 6.5V.
10. 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.

11. Poll OTPM_Control_Auto_WR_end (R0x304A [1]) to determine when the sensor is finished programming the word.
12. Verify that the otpm_control_auto_wr_success(0x304A[2]) bit is set.
13. If the above bits are not set to 1, then examine otpm_status register R0x304E[9] to verify if the OTPM memory is full and 0x304E[10] to verify if OTPM memory is insufficient.
14. Remove the high voltage (VPP) and float VPP pin.

Reading the OTPM

1. 1. Apply power to all the power rails of the sensor (VDD_IO, VDD_SLVS, DVDD, VAA, and VAA_PIX) at their nominal voltage.
2. Set EXTCLK to normal operating frequency.
3. Perform the proper reset sequence to the sensor.
4. Set R0x3134 = 0xCD95 (timing configuration).
5. Set R0x304C[15:8] = Record Type (e.g., 0x30).
6. Set R0x304C[7:0] = Length of the record, which is the number of data registers to be read back. This could be set to 0 during OTPM auto read if length is unknown.
7. Set R0x3054 = 0x0400.
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 Mode

The AR1230 supports an image acquisition mode, the electronic rolling shutter (ERS) mode. This is the normal mode of operation. When the AR1230 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 AR1230 switches cleanly from the old integration time to the new while only generating frames with uniform integration. See “Changes to Integration Time” on page 21.

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. The output image size is controlled by the x_output_size and y_output_size registers.

Pixel Border

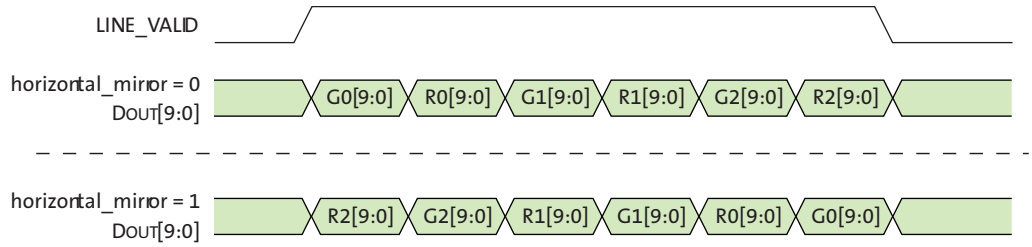
The default settings of the sensor provide a4000H x 3000V image. A border of up to 8 pixels (4 in subsampling) on each edge can be enabled by reprogramming the x_addr_start, y_addr_start, x_addr_end, y_addr_end, x_output_size, and y_output_size registers accordingly.

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 17 on page 18 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.

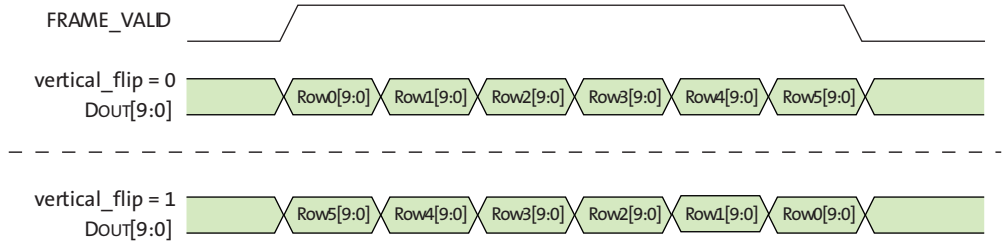
Figure 8: Effect of horizontal_mirror on Readout Order



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 18 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.

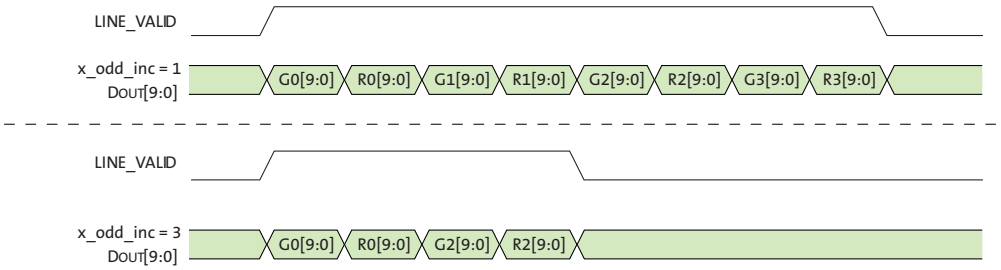
Figure 9: Effect of vertical_flip on Readout Order



Subsampling

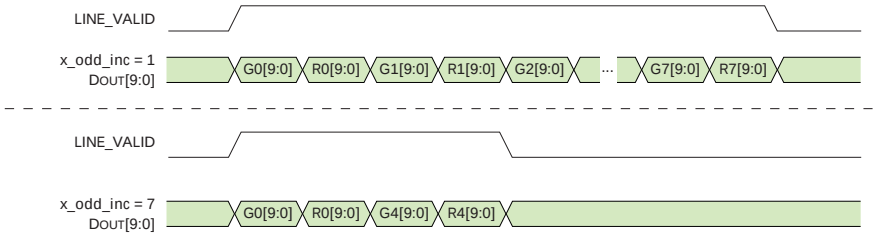
The AR1230 supports subsampling. Subsampling reduces the amount of data processed by the analog signal chain in the AR1230 thereby allowing the frame rate to be increased. Subsampling is enabled by setting x_odd_inc and/or y_odd_inc. Values of 1, 3, and 7 can be supported. Setting both of these variables 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 AR1230. Setting x_odd_inc = 3 and y_odd_inc = 3 results in a quarter reduction in output image size. Figure 19 on page 19 shows a sequence of 8 columns being read out with x_odd_inc = 3 and y_odd_inc = 1.

Figure 10: 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 AR1230. Figure 20 shows a sequence of 16 columns being read out with x_odd_inc = 7 and y_odd_inc = 1.

Figure 11: 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 21 through Figure 23 on page 20.

Figure 12: Pixel Readout (No Subsampling)

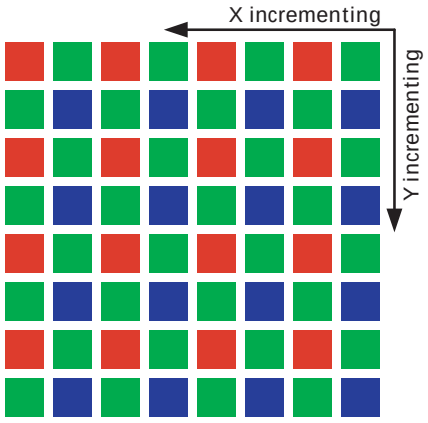


Figure 13: Pixel Readout (x_odd_inc = 3, y_odd_inc = 3)

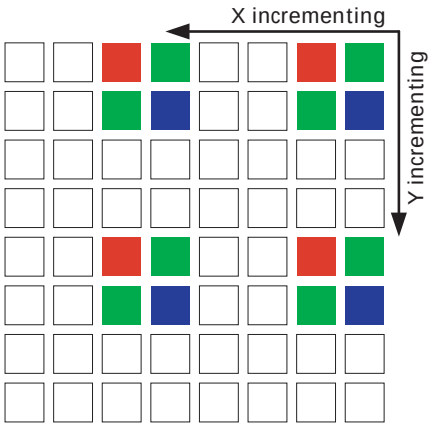
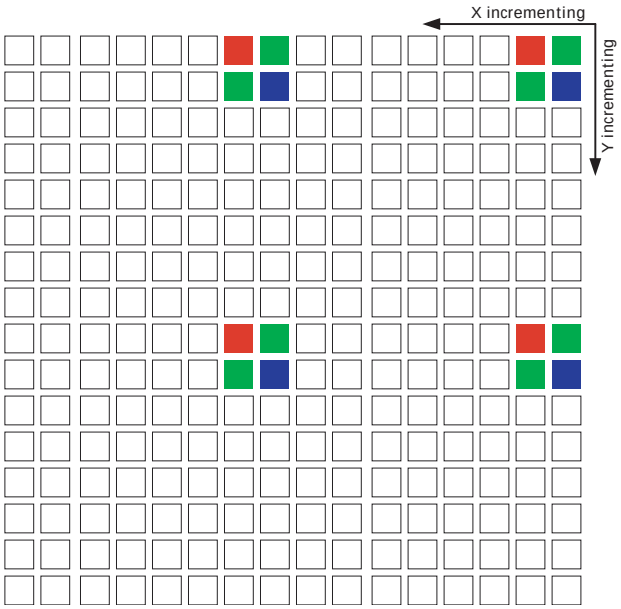


Figure 14: Pixel Readout (x_odd_inc = 7, y_odd_inc = 7)



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 14 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 14).

Table 7: 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		

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.

The AR1230 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 AR1230 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 23.

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 15.

Table 8: Minimum Row Time and Blanking Numbers

	No Subsampling		Subsampling	
row_speed[2:0]	1	2	1	2
min_line_blanking_pck	0x0200	0x0200	0x01CC	0x01CC
min_line_length_pck	0x1110	0x0918	0x1110	0x0918

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 15.
- line_length_pck ≥ (x_addr_end - x_addr_start + x_odd_inc)/((1+x_odd_inc)/2) + min_line_blanking_pck in Table 15.
- 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 9: Minimum Frame Time and Blanking Numbers

	No Subsampling	Subsampling
min_frame_blanking_lines	0x005C	0x0054

Integration Time

The integration (exposure) time of the AR1230 is controlled by the fine_integration_time and coarse_integration_time registers.

The limits for the fine integration time are defined by:

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

The limits for the coarse integration time are defined by:

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

The actual integration time is given by:

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

It is required that:

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

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

In subsampling 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_max_margin. Values for different mode combinations are shown in Table 17.

Table 10: fine_integration_time Limits

	No Row Binning		Row Binning	
row_speed[2:0]	1	2	1	2
fine_integration_time_max_margin	0x016A	0x01AC	0x016A	0x00B6

Flash Timing Control

The AR1230 supports both xenon and LED flash timing through GPIO[0] (default output). The timing of the FLASH signal with the default settings is shown in Figure 24 (xenon) and Figure 25 on page 25 (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 25. Read-only bit flash[14] is set during frames that are correctly integrated; the state of this bit is shown in Figures 24 and Figure 25.

Figure 15: Xenon Flash Enabled

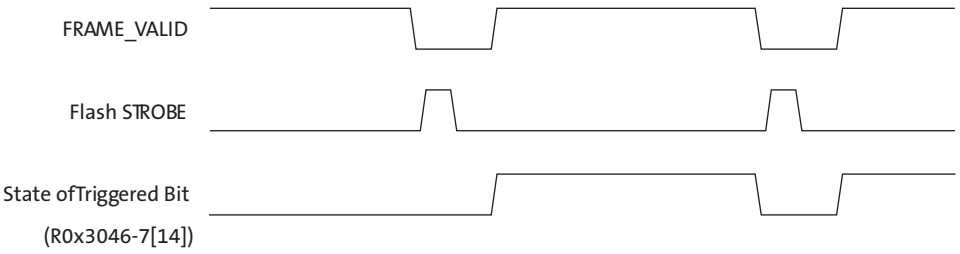
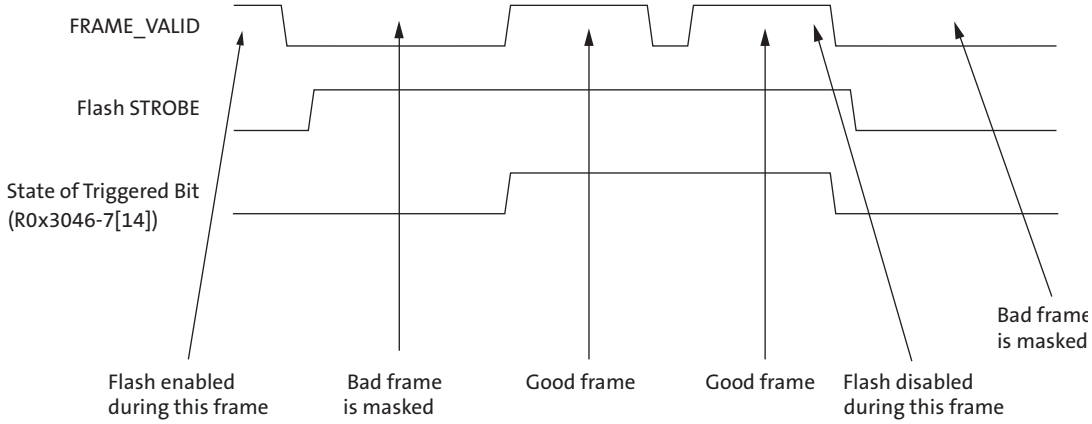


Figure 16: LED Flash Enabled



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

Gain

AR1230 supports both analog and digital gain.

Analog gain

Analog gain is provided by colamp and ADC reference scaling (there is no ASC gain due to column parallel nature of architecture). Only global (not per-color) coarse gain can be set by analog gain. Global gain register (R0x305E) sets the analog gain. Bits [1:0] set the colamp gain while bits [4:2] are reserved for ADC gain. While the 2-bit colamp gain provides up to 4x analog gain, only LSB (bit [2]) of ADC gain bits is utilized to support 2x ADC gain. Table 18 is the recommended gain setting:

Table 11: Recommended Analog Gain Setting

ADC Gain Codes (R0x305E[4:2])			Colamp Gain Codes (R0x305E[1:0])		ADC Gain*	Colamp Gain	Total Gain
0	0	0	0	0	1	1	1
0	0	0	0	1	1	2	2
0	0	0	1	0	1	3	3
0	0	0	1	1	1	4	4
0	0	1	1	0	2	3	6
0	0	1	1	1	2	4	8

Note: *R0x3EE6[15:12] = 2 needs to be set (at streaming off) for 2x ADC Gain.

Digital Gain

Digital gain provides both per-color and fine (sub 1x) gain. The analog and digital gains are multiplicative to give the total gain. Digital gain is set by setting bits R0x305E[15:5] to set global gain or by individually setting digital color gain R0x3056-C[15:5] where these 11 bits are designed in 4p7 format i.e. 4 MSB provide gain up to 15x in step of 1x while 7 LSB provide sub-1x gain with a step size of 1/128. This sub-1x gain provides the fine gain control for the sensor.

Total Gain

Max. total gain required by design spec is 8x (analog) and 16x (digital) with min. step size of 1/8. The total gain equation can be formulated as:

$$Total\ gain = (1 + dec(R0x305E[1:0])) \times (1 + R0x305E[2]) \times \frac{dec(R0x305E[15:5])}{128} \quad \text{(EQ 12)}$$

where X is 6, 8, A, C, for Gr, B, R and Gb, respectively

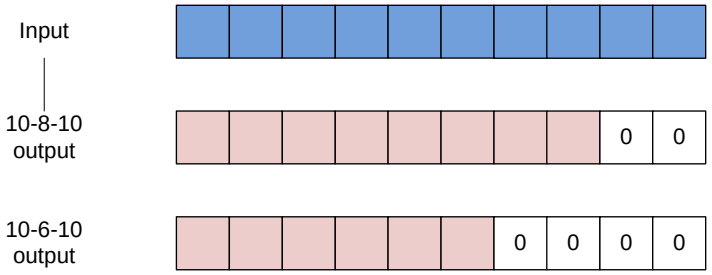
Note: Aptina recommends using the registers mentioned above for gain settings. Avoid R0x3028 to R0x3038 unless their mapping to above registers is well understood and taken into account.

New Features

Bit-depth Compression

AR1230 supports SMIA DPCM with 10-8-10 and 10-6-10 compression.

The output compressed data is MSB-aligned. When compression is enabled, '0's are padded to the LSB side.



The registers listed in Table 19 are related to compression.

Table 12: Compression Registers

Address	Name	Register	Description
0x0112	ccp_data_format	RW	[3:0]: The bit-width of the compressed pixel data [11:8]: The bit-width of the uncompressed data
0x31AE	serial format	RW	[9:8]: serial interface type 2: MIPI 3: Reserved [2:0]: serial data lanes

3D Support

This function uses GPI (as an input) and GPIO1 (as an output). The input is sampled on rising CLK (EXTCLK) and the output changes on rising CLK (EXTCLK). In order to use this function, GPIO1 must be enabled as an output.

The CHAIN_CONTROL register should only be written when the sensor is not in streaming mode; the effect of writing to this register when the sensor is in streaming mode is UNDEFINED.

When chain_enable=1 and master=0, transitions on the SADDR input are propagated synchronously to the GPIO1 output.

When chain_enable=1 and master=1, GPIO1 generates 'events' under certain circumstances. An event is a 'start' bit followed by a 4-bit code.

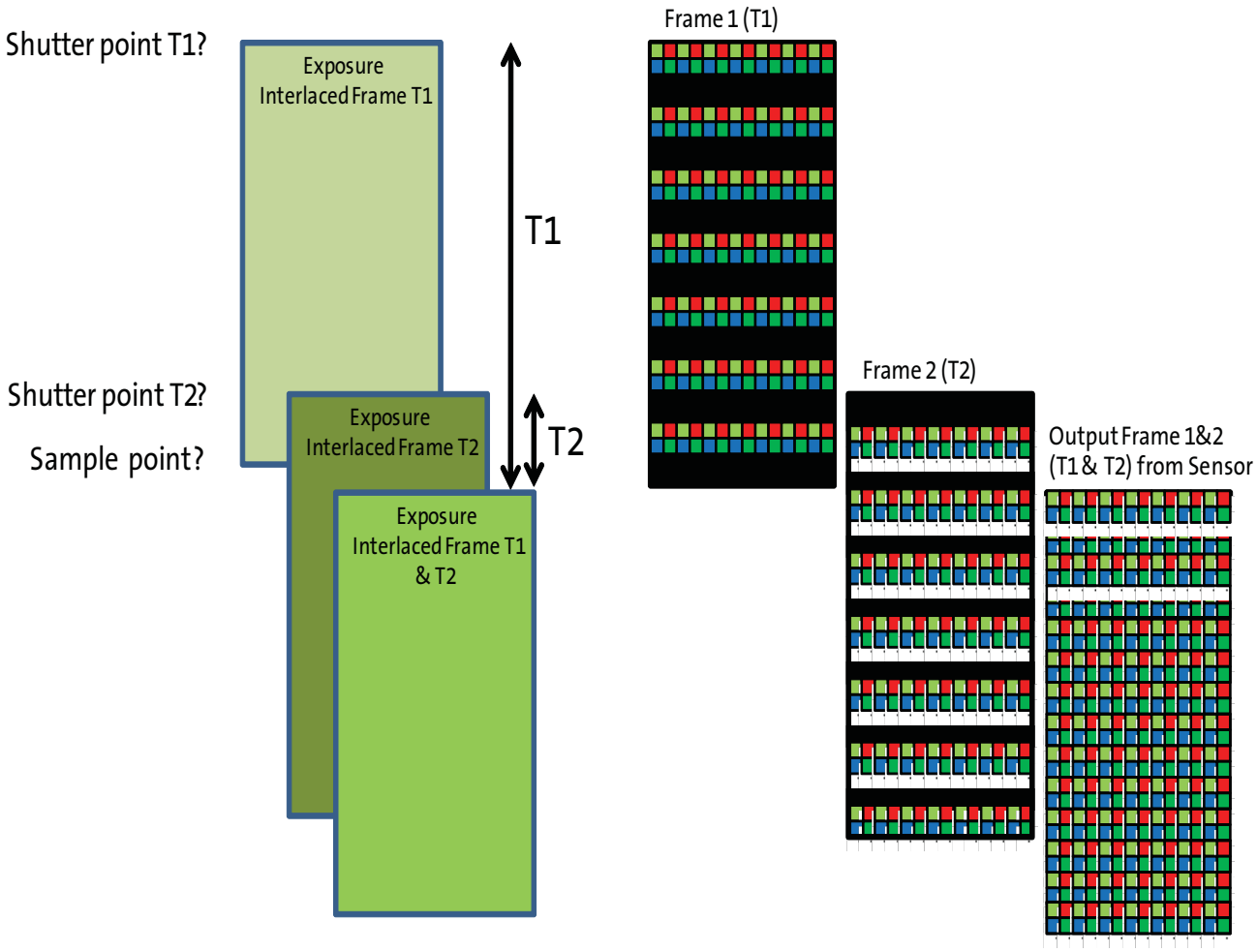
When chain_enable=1 and sync_enable=1 and master=0, and the sensor core enters streaming mode (bit 2 in reset_register is changed to 1), the actual entry to streaming mode is delayed until the sensor receives an event.

Interlaced HDR Readout

The sensor enables HDR by outputting frames where even and odd row pairs within a single frame are captured at different integration times. This output is then matched with an algorithm designed to reconstruct this output into an HDR still image or video.

The sensor HDR is controlled by two shutter pointers (Shutter pointer1, Shutter pointer2) that control the integration of the odd (Shutter pointer1) and even (Shutter pointer 2) row pairs.

Figure 17: HDR Integration Time



Integration Time for Interlaced HDR Readout

Tint1 (integration time 1) and Tint2 (integration time 2)

The limits for the coarse integration time are defined by:

$$coarse\_integration\_time\_min \le coarse\_integration\_time \le (frame\_length\_lines - coarse\_integration\_time\_max\_margin) \quad \text{(EQ 13)}$$

$$coarse\_integration\_time2\_min \le coarse\_integration\_time2 \le (frame\_length\_lines - coarse\_integration\_time2\_max\_margin) \quad \text{(EQ 14)}$$

The actual integration time is given by:

$$integration\_time = \frac{(coarse\_integration\_time \times line\_length\_pck)}{vt\_pix\_clk\_freq\_mhz \times 10^6} \quad \textit{(EQ 15)}$$

$$integration\_time2 = \frac{(coarse\_integration\_time2 \times line\_length\_pck)}{vt\_pix\_clk\_freq\_mhz \times 10^6} \quad \text{(EQ 16)}$$

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

The ratio between even and odd rows is typically adjusted to 1x, 2x, 4x, and 8x.

Sensor Core Digital Data Path

Test Patterns

The AR1230 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 20.

Table 13: 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 (12-bits) (Note: 12-bit is not supported in AR1230)
257	Walking 1s (10-bits)
258	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 AR1230 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 27 on page 32, 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).

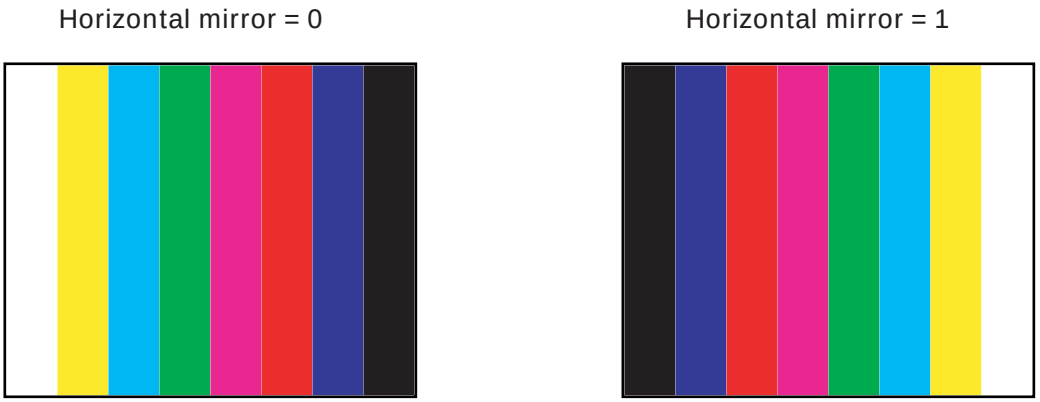
The pattern occupies the full height of the output image.

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 and scaling of this test pattern is undefined.

Figure 18: 100 Percent Color Bars Test Pattern



Fade-to-gray Color Bars Test Pattern

In this test pattern, shown in Figure 28 on page 33, 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.

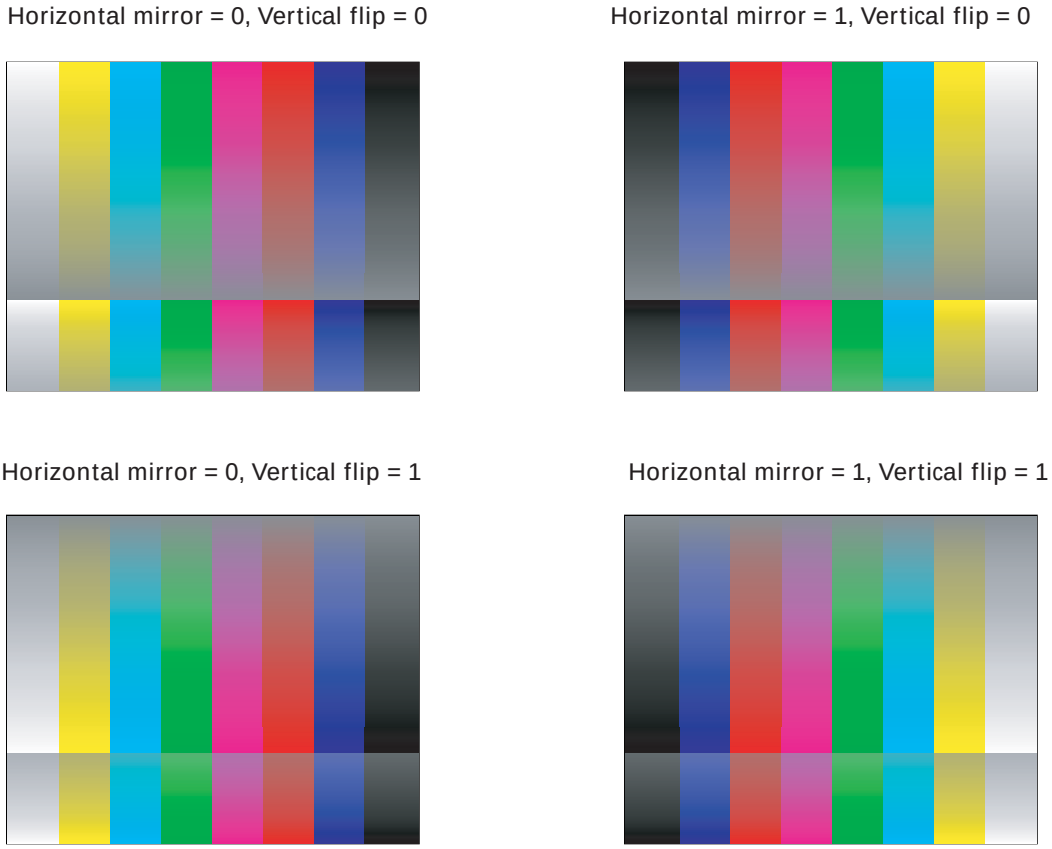
The rate-of-fade of the Bayer pattern is set so that there is at least one full pattern within a full-sized image for the sensor. Factors that affect this are the resolution of the ADC (10-bit) and the image height. For example, the AR1230 fades the pixels by 2 LSB for each two rows. With 10-bit data, 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 and scaling of this test pattern is undefined.

Figure 19: 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 20: Walking 1s 10-bit Pattern

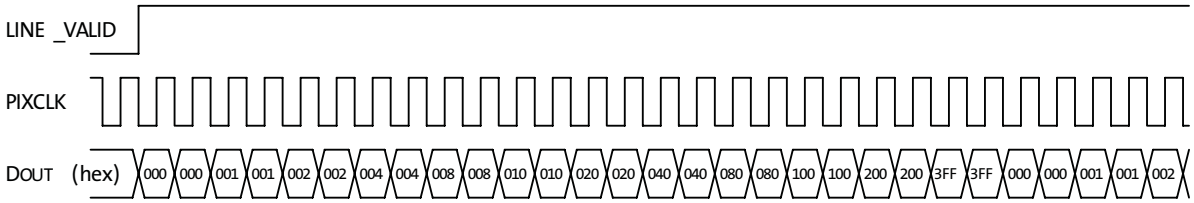
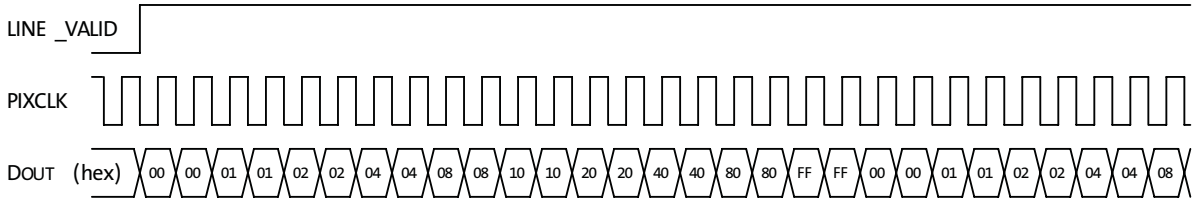


Figure 21: Walking 1s 8-bit Pattern



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 AR1230 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.

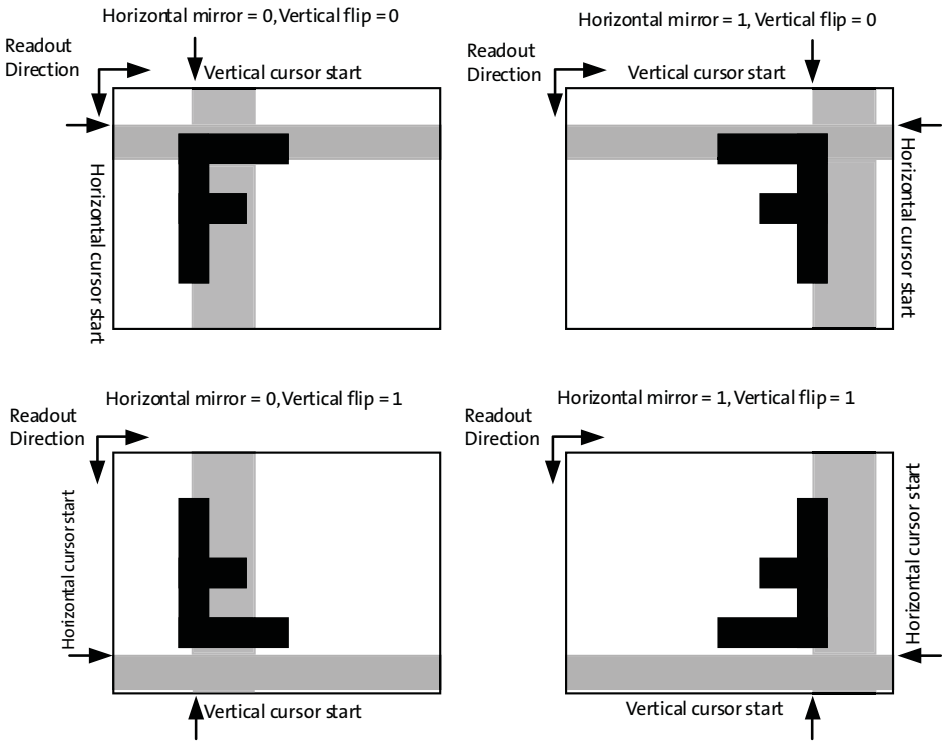
Each cursor can be inhibited by setting its width to 0. The programmed cursor position corresponds to the x and y addresses of the pixel array. For example, setting horizontal_cursor_position to the same value as y_addr_start would result in a horizontal cursor being drawn starting on the first row of 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 AR1230 is shown in Figure 31 on page 36 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 22: 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.

Timing Specifications

Power-Up Sequence

The recommended power-up sequence for the AR1230 is shown in Figure 32.

1. Set XSHUTDOWN LOW
2. Power up VDD_IO (1.8V-only)
3. After 1-500ms, power up VDD, VDD_SLVS (1.2V) and VAA, VAA_PIX (2.8V) (any order).
4. After 1-500ms, set XSHUTDOWN HIGH
5. Apply EXTCLK (can be applied anytime).
6. HOST configuration through I2C (PLL, output, etc).
7. Set mode_select = 1 bit.
8. PLL internally enables and locks.
9. AR1230 enters streaming mode.

Figure 23: Power-Up Sequence

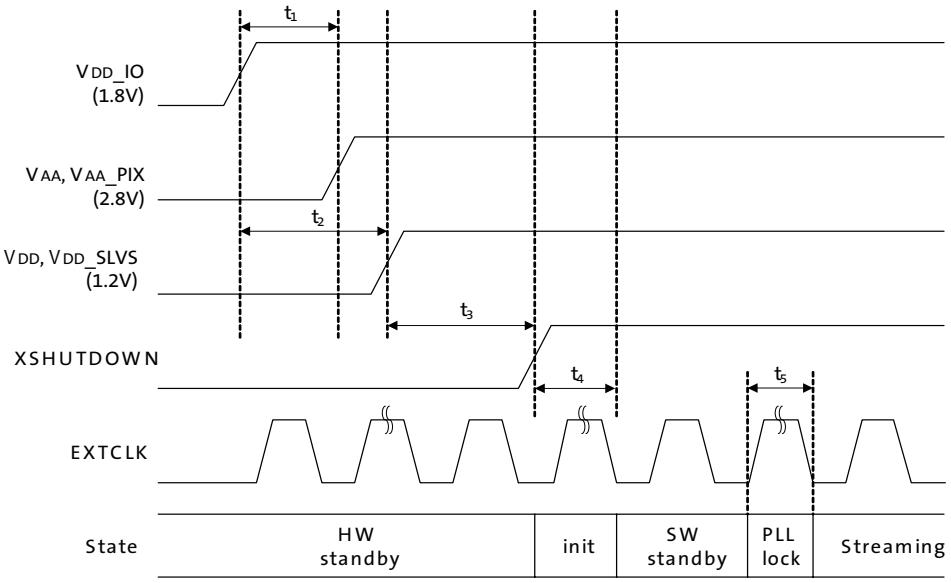


Table 14: Power-Up Sequence

Definition	Symbol	Min	Typ	Max	Unit
VDD_IO to VAA/VAA_PIX	t_1	1	–	500	ms
VDD_IO TO VDD/VDD_SLVS	t_2	1	–	500	ms
VDD, VDD_SLVS to XSHUTDOWN HIGH	t_3	1	–	500	ms
Internal initialization prior to the first I^2C transaction	t_4	2400	–	–	EXTCLKs
PLL lock time	t_5	0.25	–	500	ms

Power-Down Sequence

The recommended power-down sequence for the AR1230 is shown in Figure 33. The available power supplies—VDD_IO, VDD, VDD_SLVS, VAA, VAA_PIX—can be turned off at the same time or have the separation specified below.

1. Set software standby mode (mode_select = 0) register.
2. Wait till the end of the current frame.
3. Set XSHUTDOWN LOW.
4. Power off VDD, VDD_SLVS (1.2V) and VAA, VAA_PIX (2.8V) (any order).
5. Power off VDD_IO (1.8V).

Figure 24: Power-Down Sequence

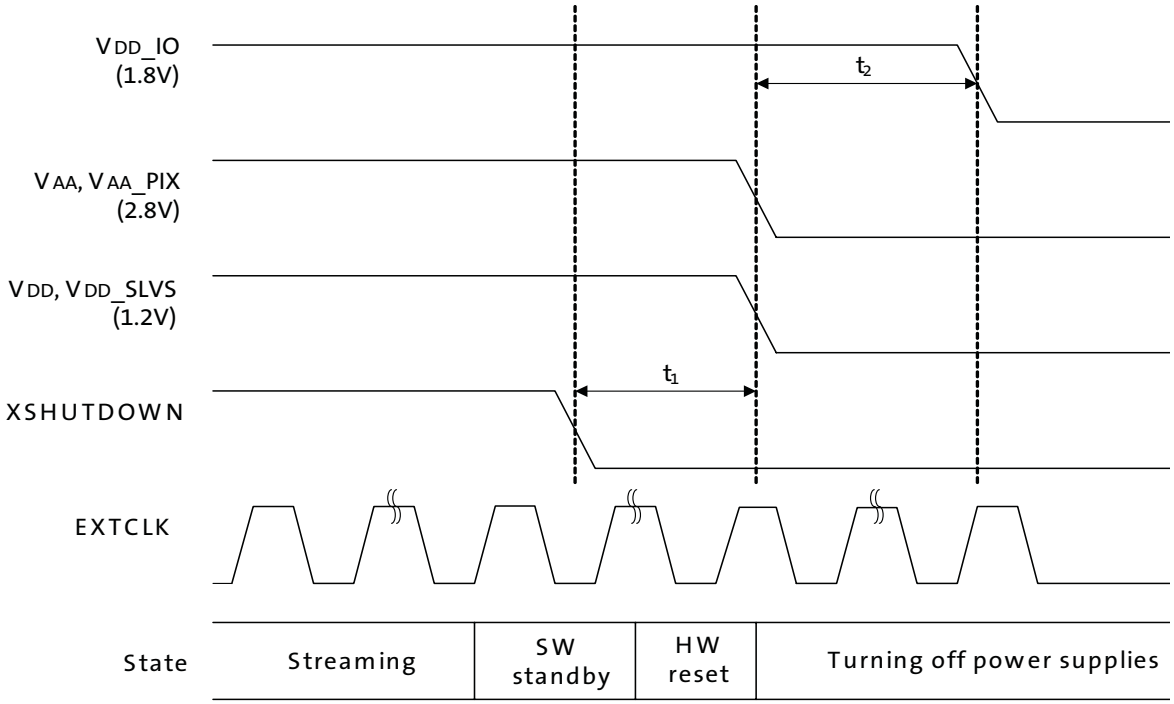


Table 15: Power-Down Sequence

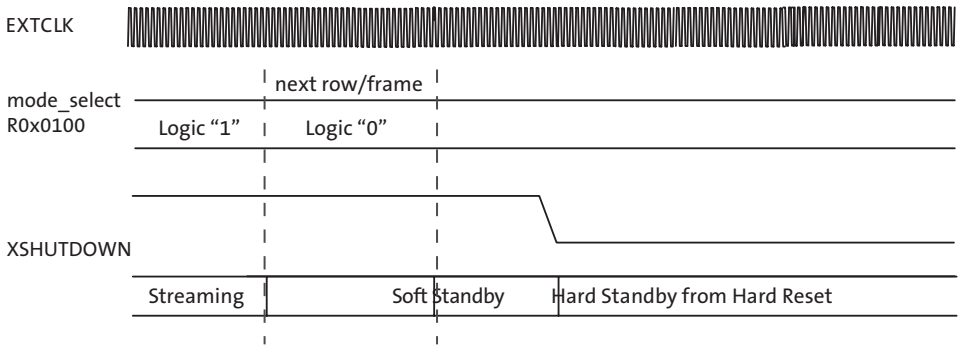
Definition	Symbol	Min	Typ	Max	Unit
Hard reset	t_1	0	–	500	ms
VAA/VAA_PIX, VDD, VDD_SLVS	t_2	0	–	500	ms

Hard Standby and Hard Reset

The hard standby state is reached by the assertion of the 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 34.

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 XSHUTDOWN (active LOW) to reset the sensor.
4. The sensor remains in hard standby state if XSHUTDOWN remain in the logic “0” state.

Figure 25: Hard Standby and Hard Reset



Soft Standby and Soft Reset

The AR1230 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 35 on page 40.

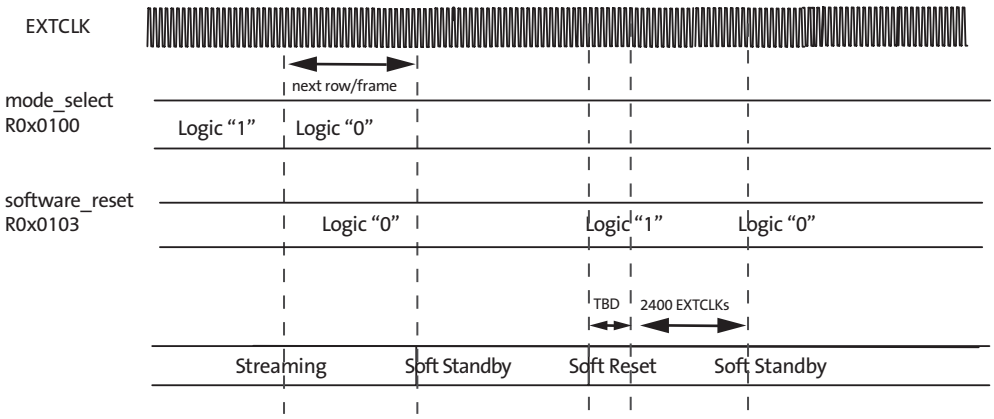
Soft Standby

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 26: Soft Standby and Soft Reset



Internal VCM Driver

The AR1230 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 27: VCM Driver Typical Diagram

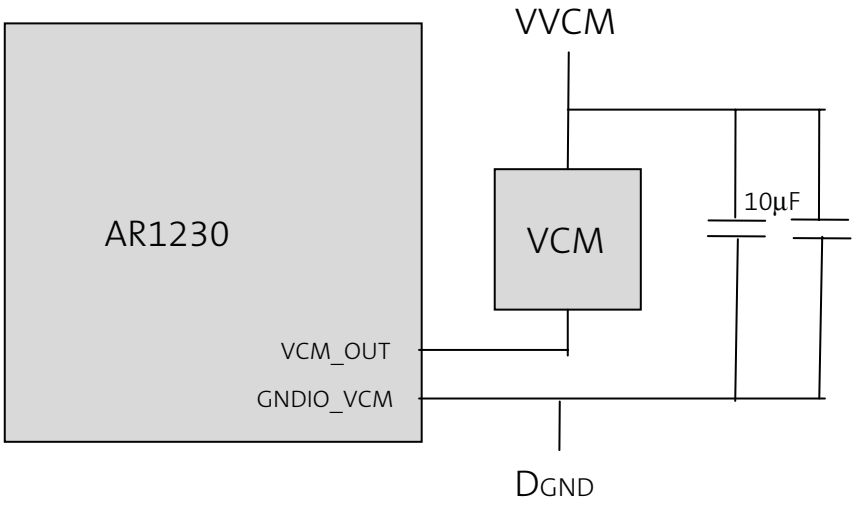


Table 16: VCM Driver Typical

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

Table 16: VCM Driver Typical

Characteristic	Parameter	Minimum	Typical	Maximum	Units
INL	Relative accuracy	–	±1.5	± 4	LSB
RES	Resolution	–	10	–	bits
DNL	Differential nonlinearity	–1	–	+1	LSB
IVCM	Output current	5	–	100	mA
	Slew rate	–	.3	–	mA/µs

Spectral Characteristics

Figure 28: Quantum Efficiency

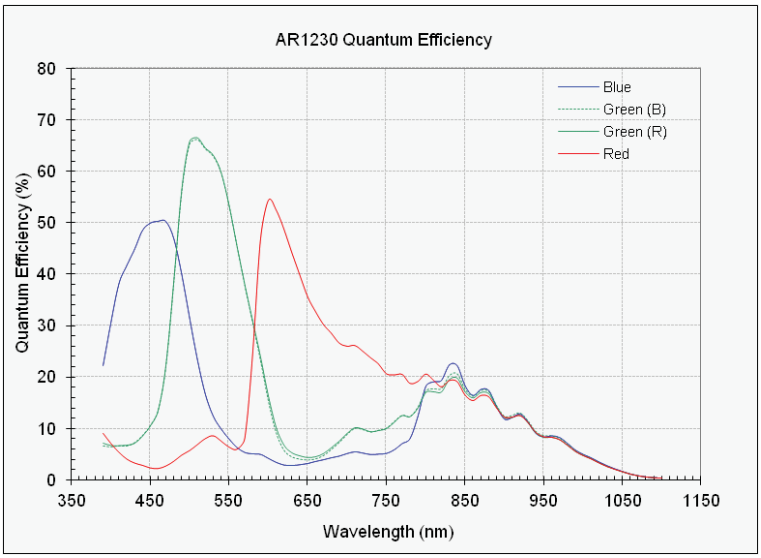
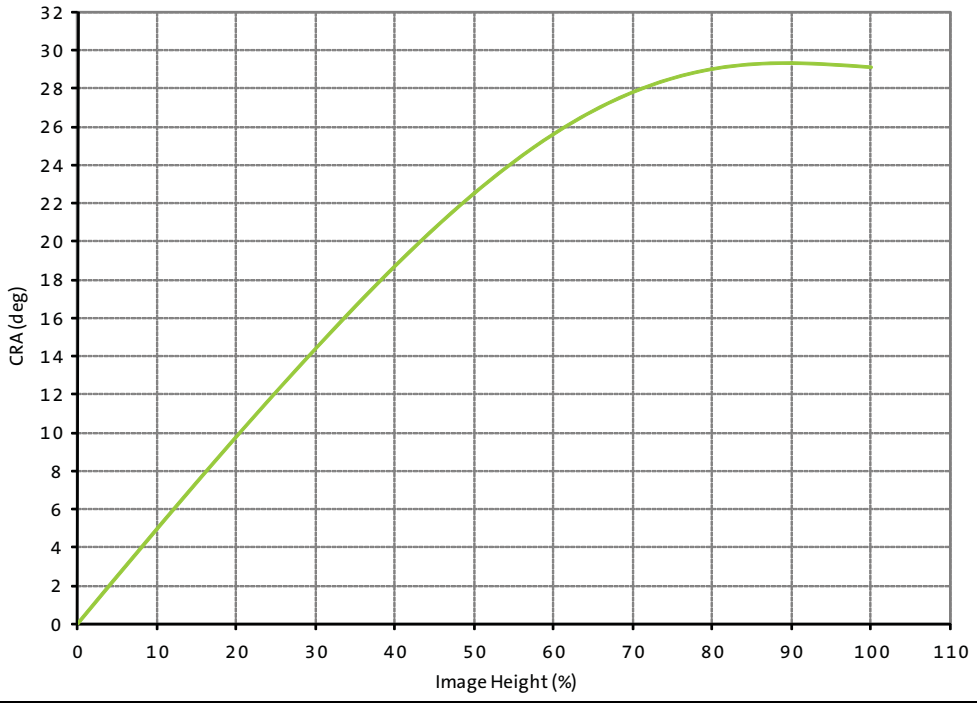


Figure 29: Chief Ray Angle Image Height



CRA vs. Image Height Plot	Image Height (%)	Image Height (mm)	CRA (°)
	0	0	0
	5	0.138	2.50
	10	0.275	4.97
	15	0.413	7.39
	20	0.550	9.78
	25	0.688	12.12
	30	0.825	14.40
	35	0.963	16.61
	40	1.100	18.71
	45	1.238	20.70
	50	1.375	22.52
	55	1.513	24.17
	60	1.650	25.62
	65	1.788	26.84
	70	1.925	27.81
	75	2.063	28.54
	80	2.200	29.02
	85	2.338	29.27
	90	2.475	29.332
	95	2.613	29.25
	100	2.750	29.11

Table 17: Chief Ray Angle

Pixel Size	Array Size (pixels)		Array Size (mm)	
1.1µm	X_{max}	4000	X_{max}	4.400
	Y_{max}	3000	Y_{max}	3.300
			Diagonal	5.500

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 25.

Table 18: CRA Value

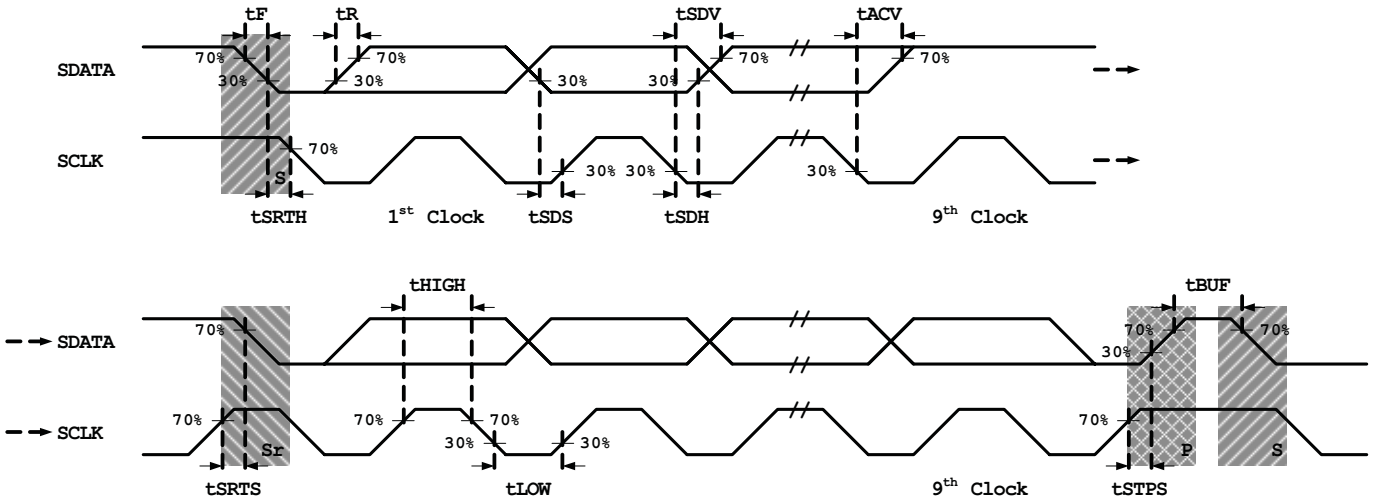
Binary Value of R0x31FE[6:4]	CRA Value
000	29.0

Electrical Characteristics

Two-Wire Serial Register Interface

The electrical characteristics of the two-wire serial register interface (SCLK, SDATA) are shown in Figure 39 and Table 26. Table 27 shows the timing specification for the two-wire serial interface.

Figure 30: 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 19: Two-Wire Serial Register Interface Electrical Characteristics

^{f}EXTCLK = 25 MHz; VAA = 2.8V; VAA_PIX = 2.8V; VDD_IO = 1.8V; VDD (digital core) = 1.2V; VDD_PLL = 1.2V; VDD_TX = 1.8V; Output load = 68.5pF; TJ = 55°C

Symbol	Parameter	Condition	Min	Typ	Max	Unit
VIL	Input LOW voltage		-0.5	–	0.3 x VDD_IO	V
VIH	Input HIGH voltage		0.7 x VDD_IO	–	VDD_IO + 0.5	V
IIN	Input leakage current	No pull up resistor; VIN = VDD_IO or DGND	10	–	14	µA
VOL	Output LOW voltage	At specified 2mA	0.11	–	0.3	V
IOL	Output LOW current	At specified VOL 0.1V	–	–	6	mA
CIN	Input pad capacitance		–	–	6	pF
CLOAD	Load capacitance		–	–	N/A	pF

Table 20: Two-Wire Serial Interface Timing Specifications

VDD = 1.7-1.9V; VAA = 2.4 -3.1V; Environment temperature = -30°C to 50°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

Table 20: Two-Wire Serial Interface Timing Specifications (continued)
V_{DD} = 1.7-1.9V; V_{AA} = 2.4 -3.1V; Environment temperature = -30°C to 50°C

Symbol	Definition	Min	Max	Unit
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	Note	µs
t_{SDV}	Data Valid Time	–	0.9	µs
t_{ACV}	Data Valid Acknowledge Time	–	0.9	µs
t_{STPS}	STOP Setup Time	0.6	–	µs
t_{BUF}	Bus Free Time between STOP and START	1.3	–	µs
tr	SCLK and SDATA Rise Time	–	300	ns
tf	SCLK and SDATA Fall Time	20	300	ns

EXTCLK

The electrical characteristics of the EXTCLK input are shown in Table 28. 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 AR1230 and the clock is stopped, the EXTCLK input to the AR1230 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 21: 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}_SLVS = 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	–	48	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
	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		–0.5		0.3 x V_{DD}_IO	V
V_{IL}	Input LOW voltage		–0.5		0.3 x V_{DD}_IO	V

Serial Pixel Data Interface

The electrical characteristics of the serial pixel data interface are shown in Table 29.

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 22: 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 (XSHUTDOWN, TEST, GPI0, GPI1, GPI2, and GPI3) are shown in Table 30.

Table 23: 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_SLVS} = 1.2V; Output load = 68.5pF; T_J = 55°C

Symbol	Parameter	Condition	Min	Typ	Max	Unit
V_{IH}	Input HIGH voltage		0.7 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 AR1230.

Table 24: DC Electrical Definitions and 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_SLVS} = 1.2V; Output load = 68.5pF; T_J = 55°C; Mode = Full Resolution (4000x3000); Frame rate = 20 fps

Symbol	Parameter	Condition	Min	Typ	Max	Unit
V_{AA}	Analog voltage		2.6	2.8	3.1	V
V_{AA_PIX}	Pixel supply voltage		2.6	2.8	3.1	V
V_{DD_IO}	I/O digital voltage		1.7	1.8	1.95	V
V_{DD}	Digital voltage		1.14	1.2	1.3	V
V_{DD_SLVS}	PHY digital voltage		1.14	1.2	1.3	V
V_{PP0}	OTPM Programming voltage		6	6.5	7	V
Operating Current						
I_{DD_IO}	I/O digital current	12Mp full mode (4000x3000), 20fps	–	0.07	0.3	mA
I_{DD}/I_{DD_SLVS}	PHY/digital current		–	140	185	
I_{AA}/I_{AA_PIX}	Analog/Pixel current		–	88	120	
Power Consumption			–	415	613	mW
I_{DD_IO}	I/O digital current	VGA-LP/HS (640x480), subsampling +scaling, 90fps	–	0.07	0.3	mA
I_{DD}/I_{DD_SLVS}	PHY/digital current		–	88	120	
I_{AA}/I_{AA_PIX}	Analog/Pixel current		–	97	130	
Power Consumption			–	377	560	mW
I_{DD_IO}	I/O digital current	QVGA-HS (640x480), subsampling + scaling, 90fps	–	0.7	0.3	mA
I_{DD}/I_{DD_SLVS}	PHY/digital current		–	150	195	
I_{AA}/I_{AA_PIX}	Analog/Pixel current		–	92	125	
Power Consumption				377	560	mW
I_{DD_IO}	I/O digital current	9M full mode (4000x2250), 30fps	–	0.07	0.3	mA
I_{DD}/I_{DD_SLVS}	PHY/digital current		–	150	195	
I_{AA}/I_{AA_PIX}	Analog/Pixel current		–	92	125	
Power Consumption			–	438	642	mW
I_{DD_IO}	I/O digital current	1080p-HQ (1920x1080), crop + scaling, 30fps	–	0.07	0.3	mA
I_{DD}/I_{DD_SLVS}	PHY/digital current		–	131	175	
I_{AA}/I_{AA_PIX}	Analog/Pixel current		–	94	125	
Power Consumption			–	421	616	mW
I_{DD_IO}	I/O digital current	720p-HQ (1280x720), scaling, 30fps	–	0.07	0.3	mA
I_{DD}/I_{DD_SLVS}	PHY/digital current		–	131	170	
I_{AA}/I_{AA_PIX}	Analog/Pixel current		–	95	125	
Power Consumption				423	609	mW
I_{DD_IO}	I/O digital current	1080p-LP/HS (1920x1080), crop+ subsampling, 60fps	–	0.07	0.3	mA
I_{DD}/I_{DD_SLVS}	PHY/digital current		–	108	145	
I_{AA}/I_{AA_PIX}	Analog/Pixel current		–	95	125	
Power Consumption			–	396	577	mW
I_{DD_IO}	I/O digital current	720p-LP/HS (1280x720), crop+ subsampling +scaling, 60fps	–	0.07	0.3	mA
I_{DD}/I_{DD_SLVS}	PHY/digital current		–	105	140	
I_{AA}/I_{AA_PIX}	Analog/Pixel current		–	95	125	
Power Consumption			–	392	570	mW

Table 24: DC Electrical Definitions and 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_SLVS} = 1.2V; Output load = 68.5pF; T_J = 55°C; Mode = Full Resolution (4000x3000); Frame rate = 20 fps

Symbol	Parameter	Condition	Min	Typ	Max	Unit
STANDBY Current						
I_{DD_IO}	I/O digital current	STANDBY current when asserting XSHUTDOWN signal (EXTCLK on)		<15		µA
I_{DD}/I_{DD_SLVS}	PHY/digital current					
I_{AA}/I_{AA_PIX}	Analog/Pixel current					
I_{DD_IO}	I/O digital current	STANDBY current when asserting XSHUTDOWN signal (EXTCLK off)		<15		µA
I_{DD}/I_{DD_SLVS}	PHY/digital current					
I_{AA}/I_{AA_PIX}	Analog/Pixel current					
I_{DD_IO}	I/O digital current	STANDBY current when asserting R0x0100 = 1 (EXTCLK on)		<20		mA
I_{DD}/I_{DD_SLVS}	PHY/digital current					
I_{AA}/I_{AA_PIX}	Analog/Pixel current					
I_{DD_IO}	I/O digital current	STANDBY current when asserting R0x0100 = 1 (EXTCLK off)		<20		mA
I_{DD}/I_{DD_SLVS}	PHY/digital current					
I_{AA}/I_{AA_PIX}	Analog/Pixel current					

Absolute Maximum Ratings

Caution **Stresses greater than those listed in Table 32 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 25: Absolute Max Voltages

Symbol	Parameter	Min	Max	Unit
VDD_MAX	Core digital voltage	-0.3	1.5	V
VDD_IO_MAX	I/O digital voltage	-0.3	2.1	V
VDD_SLVS_MAX	PHY digital voltage	-0.3	1.5	V
VAA_MAX	Analog voltage	-0.3	3.5	V
VAA_PIX_MAX	Pixel supply voltage	-0.3	3.5	V
T_OP	Operating temperature (at junction)	-30	70	°C
T_ST	Storage temperature	-40	85	°C

MIPI Specification Reference

The sensor design and this documentation is based on the following MIPI specifications:

- MIPI Alliance Standard for CSI-2 version 1.0
- MIPI Alliance Standard for D-PHY version 1.0

Revision History

Rev. B 4/4/13

- Updated to Production
- Applied updated Aptina template
- Updated “Features” on page 13
- Updated “Key Performance Parameters” on page 1

Rev. A, Advance 10/23/12

- Initial release

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