1/2.3-Inch 9Mp CMOS Digital Image Sensor

MT9N001

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

Features

- DigitalClarity® CMOS imaging technology
- Simple two-wire serial interface
- Auto black level calibration
- Support for external mechanical shutter
- Support for external LED or xenon flash
- High frame rate preview mode with arbitrary down-size scaling from maximum resolution
- Programmable controls: gain, horizontal and vertical blanking, auto black level offset correction, frame size/rate, exposure, left–right and top–bottom image reversal, window size, and panning
- Data interfaces: parallel- or CCP2-compliant sub-low-voltage differential signalling (sub-LVDS) or one/two lane serial mobile industry processor interface (MIPI)
- On-die phase-locked loop (PLL) oscillator
- Bayer pattern down-size scaler
- Integrated position-based color and lens shading correction
- One-time programmable (OTP) memory for storing module information

Applications

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

General Description

The Aptina Imaging MT9N001 is a 1/2.3-inch CMOS active-pixel digital image sensor with a pixel array of 3488H x 2616V including border pixels. It incorporates sophisticated on-chip camera functions such as windowing, 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.

Table 1: Key Performance Parameters

Parameter		Value
Optical format		1/2.3-inch (4:3)
Active imager size		6.104mm(H) x 4.578mm(V) 7.63mm diagonal
Active pixels		3488H x 2616V
Pixel size		1.75 x 1.75µm
Chief ray angle		0°
Color filter array		RGB Bayer pattern
Shutter type		Electronic rolling shutter (ERS) with global reset release (GRR)
Input clock frequency		6–48 MHz
Maximum data rate	Parallel	96 Mp/s at 96 MHz PIXCLK
	CCP2	640 Mbps
	MIPI (two-lane)	1.536 Gbps
Frame rate	Full resolution	Programmable up to 13.2 fps serial, 9.7 fps parallel
	VGA	30 fps VGA binning with 8Mp field-of-view
ADC resolution		12-bit, on-die
Responsivity		0.44 V/lux-sec (550nm)
Dynamic range		65dB
SNR_{MAX}		35dB
Supply voltage	I/O Digital	1.7–1.9V (1.8V nominal) or 2.4–3.1V (2.8V nominal)
	Digital	1.7–1.9V (1.8V nominal)
	Analog	2.6–3.1V (2.8V nominal)
Power Consumption	Full resolution	500mW
	Preview	210mW low power VGA
	Standby	500µW (typical, EXTCLK disabled)
Package		48-pin iLCC (10mm x 10mm) Bare die
Operating temperature		–30°C to +70°C (at junction)

Ordering Information

Table 2: Available Part Numbers

Part Number	Description
MT9N001D00STCC2BBC1	Bare die
MT9N001I12STC	48-pin iLCC

Table of Contents

List of Figures

List of Tables

General Description

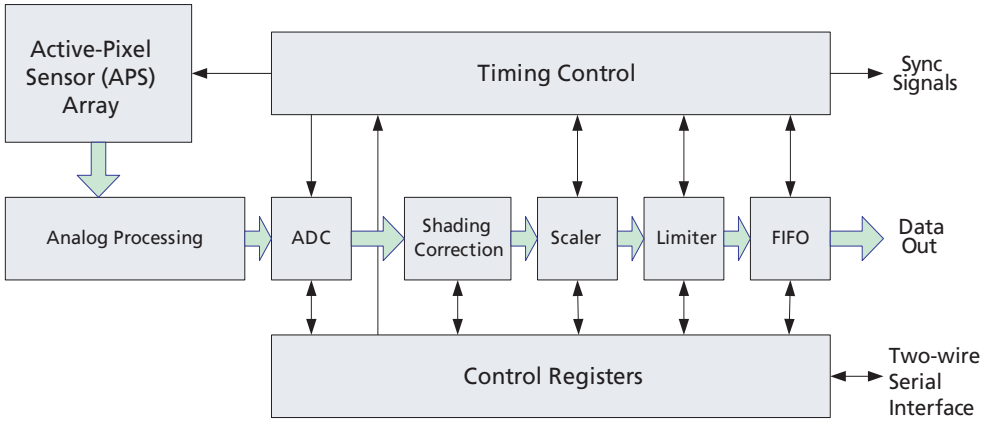
The MT9N001 digital image sensor features DigitalClarity—Aptina's breakthrough low-noise CMOS imaging technology that achieves near-CCD image quality (based on signal-to-noise ratio and low-light sensitivity) while maintaining the inherent size, cost, and integration advantages of CMOS.

When operated in its default mode, the sensor generates a full resolution image at 13.2 frames per second (fps). An on-chip analog-to-digital converter (ADC) generates a 12-bit value for each pixel.

Functional Overview

The MT9N001 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 output pixel rate is 96 Mp/s, corresponding to a pixel clock rate of 96 MHz. A block diagram of the sensor is shown in Figure 1.

Figure 1: Block Diagram



The core of the sensor is a 9Mp 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 12-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 7 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 12-bit parallel pixel data stream qualified by an output data clock (PIXCLK), together with LINE_VALID (LV) and FRAME_VALID (FV) signals.
- A digital shading correction block to compensate for color/brightness shading introduced by the lens or chief ray angle (CRA) curve mismatch.
- Additional functionality is provided. This includes a horizontal and vertical image scaler, a limiter, a data compressor, an output FIFO, and a serializer.

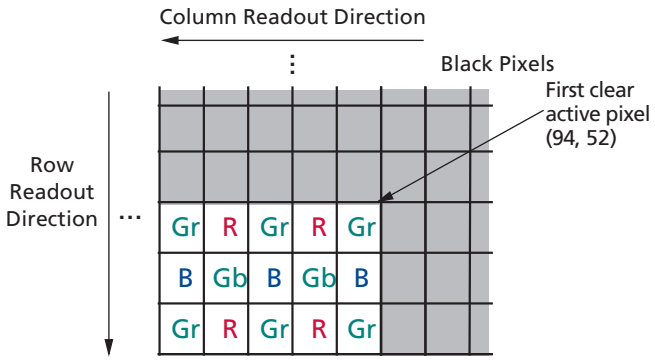
The output FIFO is present to prevent data bursts by keeping the data rate continuous. Programmable slew rates are also available to reduce the effect of electromagnetic interference from the output interface.

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

Pixel Array

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

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



Operating Modes

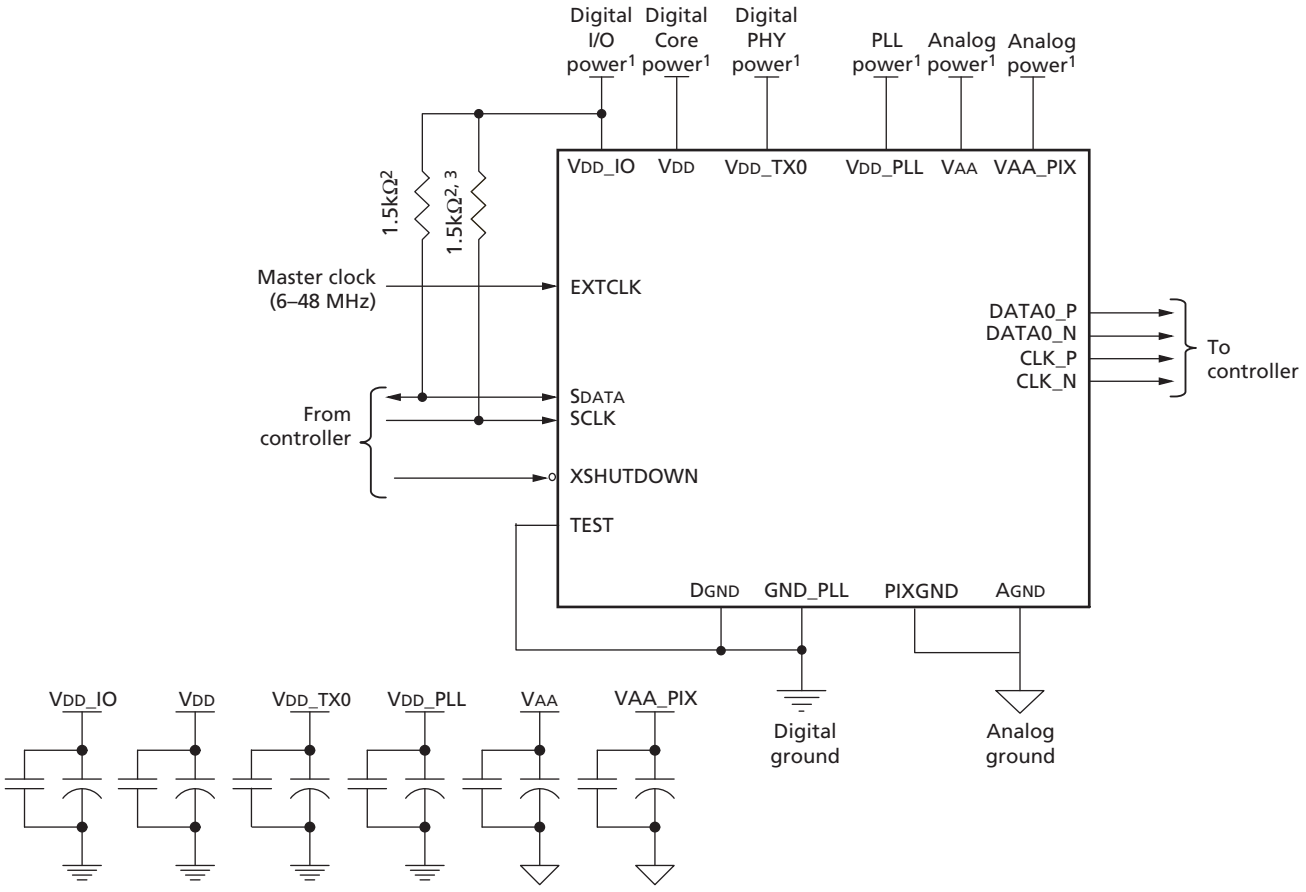
By default, the MT9N001 powers up with the serial pixel data interface enabled. The sensor can operate either in serial CPP2 or serial MIPI mode, and this mode is preconfigured at the factory. In both cases, the sensor has an SMIA-compatible register interface while the I^2C device address is compliant with SMIA or MIPI requirements as appropriate. The reset level on the TEST pin must be tied in a way that is compatible with the configured serial interface of the sensor, for instance TEST = 0 for CCP2 and TEST = 1 for MIPI.

The MT9N001 also supports parallel data out in both CCP2 and MIPI configuration. Typical configurations are shown in Figure 4 on page 11, Figure 5 on page 12, Figure 3 on page 10, and Figure 7 on page 15. These operating modes are described in "Programming Restrictions" on page 69.

For low-noise operation, the MT9N001 requires separate power supplies for analog and digital. Incoming digital and analog ground conductions should be placed in such a way that coupling between the two are minimized. Both power supply rails should also be routed in such a way that noise coupling between the two supplies and ground is minimized.

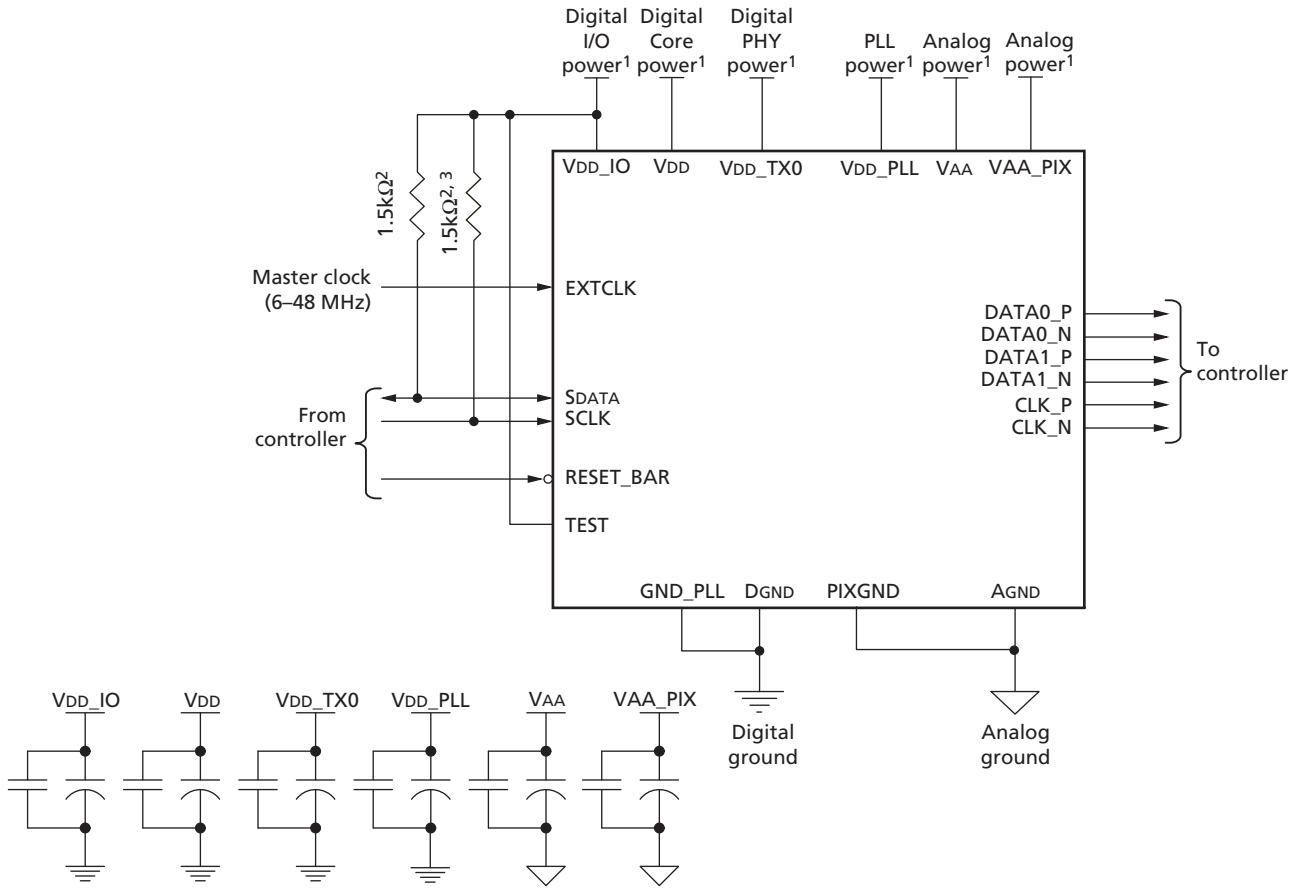
Caution **Aptina does not recommend the use of inductance filters on the power supplies or output signals.**

Figure 3: Typical Configuration: Serial CCP2 Pixel Data Interface



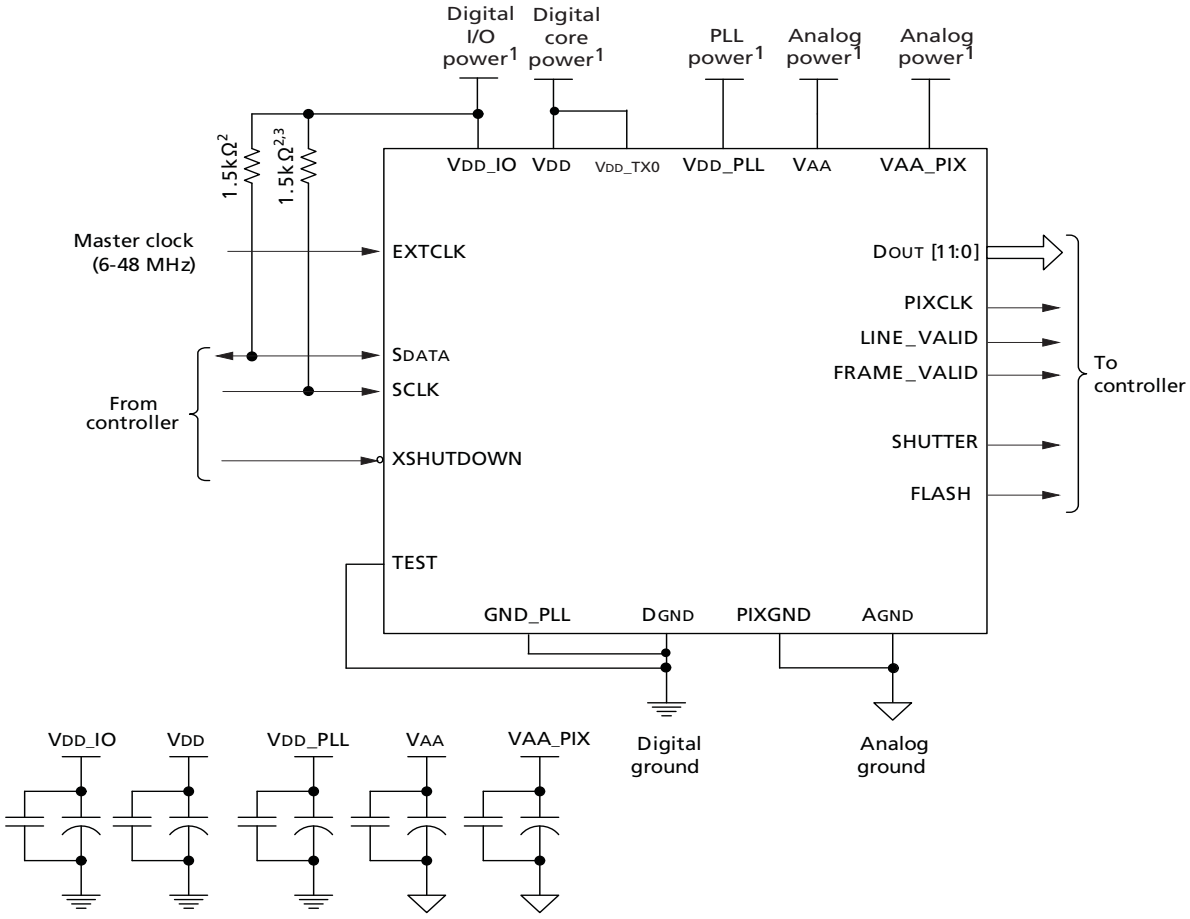
Notes:
1. All power supplies should be adequately decoupled.
2. Aptina recommends a resistor value of 1.5kΩ, but it may be greater for slower two-wire speed.
3. This pull-up resistor is not required if the controller drives a valid logic level on SCLK at all times.
4. VPP, which can be used during the module manufacturing process, is not shown in Figure 3. This pad is left unconnected during normal operation.
5. The parallel interface output pads can be left unconnected if the serial output interface is used.
6. Aptina recommends that 0.1µF and 10µF decoupling capacitors for each power supply are mounted as close as possible to the pad. Actual values and results may vary depending on layout and design considerations.
7. Aptina recommends that VDD_TX0 be tied to VDD.
8. Aptina recommends that analog power planes are placed in a manner such that coupling with the digital power planes is minimized.

Figure 4: Typical Configuration: Serial Two-Lane MIPI Pixel Data Interface



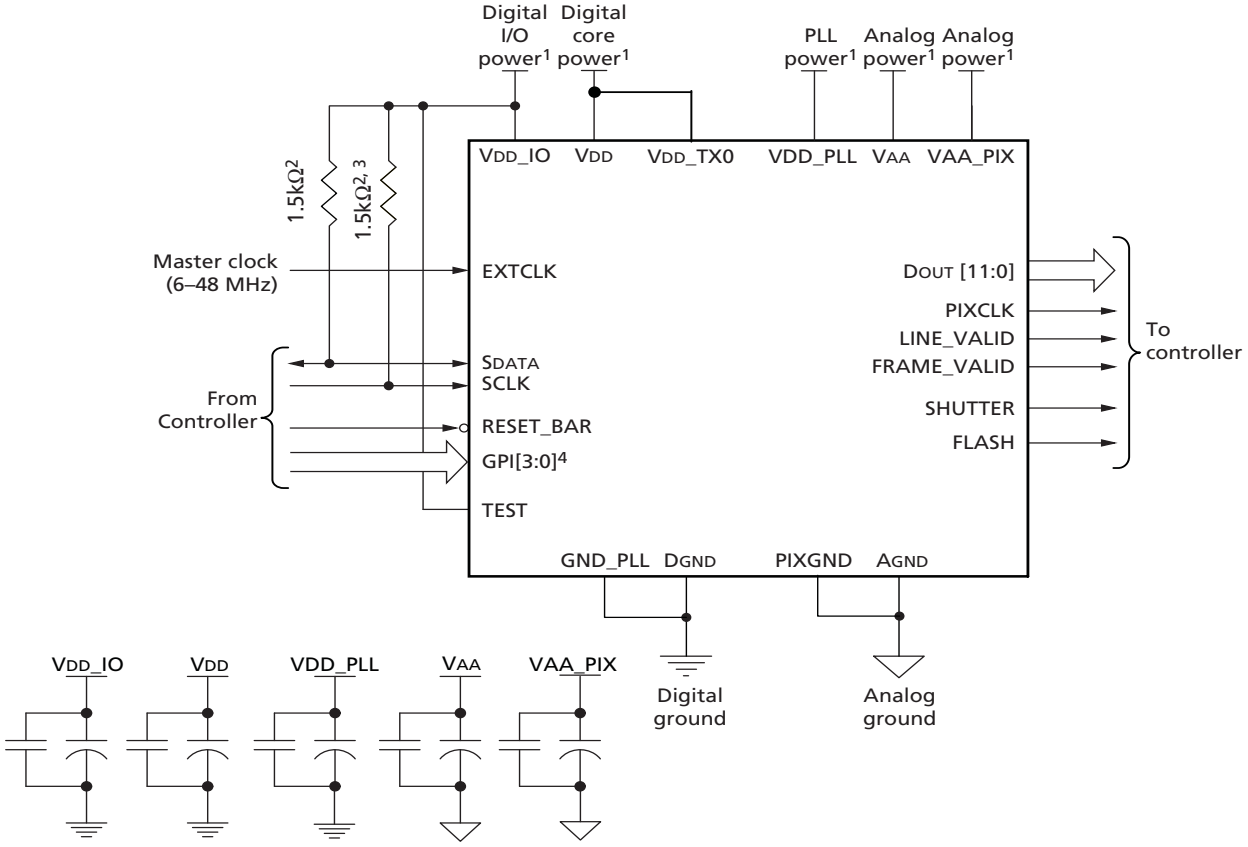
Notes:
1. All power supplies should be adequately decoupled.
2. Aptina recommends a resistor value of 1.5kΩ, but it may be greater for slower two-wire speed.
3. This pull-up resistor is not required if the controller drives a valid logic level on SCLK at all times.
4. VPP, which can be used during the module manufacturing process, is not shown in Figure 4. This pad is left unconnected during normal operation.
5. The parallel interface output pads can be left unconnected if the serial output interface is used.
6. Aptina recommends that 0.1µF and 10µF decoupling capacitors for each power supply are mounted as close as possible to the pad. Actual values and results may vary depending on layout and design considerations.
7. Aptina recommends that VDD_TX0 be tied to VDD.
8. Aptina recommends that analog power planes are placed in a manner such that coupling with the digital power planes is minimized.

Figure 5: Typical Configuration: CCP2 Parallel Pixel Data Interface



Notes:
1. All power supplies should be adequately decoupled.
2. Aptina recommends a resistor value of 1.5kΩ, but it may be greater for slower two-wire speed.
3. This pull-up resistor is not required if the controller drives a valid logic level on SCLK at all times.
4. The GPI pins can be statically pulled HIGH or LOW to be used as module IDs, or they can be programmed to perform special functions (TRIGGER, OE_N, SADDR, STANDBY) to be dynamically controlled.
5. VPP, which can be used during the module manufacturing process, is not shown in Figure 5. This pad is left unconnected during normal operation.
6. The serial interface output pads can be left unconnected if the parallel output interface is used.
7. Aptina recommends that 0.1µF and 10µF decoupling capacitors for each power supply are mounted as close as possible to the pad. Actual values and results may vary depending on layout and design considerations.
8. Aptina recommends that analog power planes are placed in a manner such that coupling with the digital power planes is minimized.
9. Aptina recommends that VDD_TX0 is tied to VDD when the sensor is using the parallel interface.

Figure 6: Typical Configuration: Parallel MIPI Pixel Data Interface (MIPI)



Notes:
1. All power supplies should be adequately decoupled.
2. Aptina recommends a resistor value of 1.5kΩ, but it may be greater for slower two-wire speed.
3. This pull-up resistor is not required if the controller drives a valid logic level on SCLK at all times.
4. The GPI pins can be statically pulled HIGH or LOW to be used as module IDs, or they can be programmed to perform special functions (TRIGGER, OE_N, SADDR, STANDBY) to be dynamically controlled.
5. VPP, which can be used during the module manufacturing process, is not shown in Figure 7. This pad is left unconnected during normal operation.
6. The serial interface output pads can be left unconnected if the parallel output interface is used.
7. Aptina recommends that 0.1µF and 10µF decoupling capacitors for each power supply are mounted as close as possible to the pad. Actual values and results may vary depending on layout and design considerations.
8. Aptina recommends that analog power planes are placed in a manner such that coupling with the digital power planes is minimized.
9. Aptina recommends that VDD_TX0 is tied to VDD when the sensor is using the parallel interface.

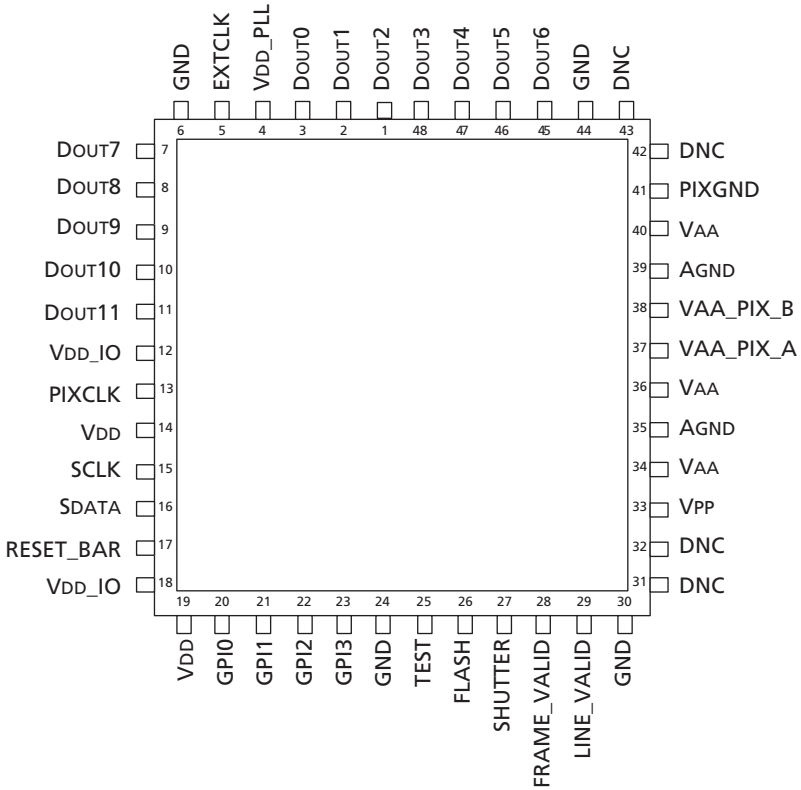
Signal Descriptions

Table 3 provides signal descriptions for MT9N001 die. For pad location and aperture information, refer to the MT9N001 die data sheet.

Table 3: Signal Descriptions

Pad Name	Pad Type	Description
EXTCLK	Input	Master clock input, 6–48 MHz.
RESET_BAR (XSHUTDOWN)	Input	Asynchronous active LOW reset. When asserted, data output stops and all internal registers are restored to their factory default settings.
SCLK	Input	Serial clock for access to control and status registers.
GPI[3:0]	Input	General purpose inputs. After reset, these pads are powered-down by default; this means that it is not necessary to bond to these pads. Any of these pads can be configured to provide hardware control of the standby, output enable, SADDR select, and shutter trigger functions. Can be left floating if not used.
TEST	Input	Enable manufacturing test modes. Connect to DGND for normal operation of the CCP2 configured sensor, or connect to VDD_IO power for the MIPI-configured sensor.
SDATA	I/O	Serial data from READs and WRITEs to control and status registers.
DATA0_P	Output	Differential CCP2/MIPI (sub-LVDS) serial data (positive).
DATA0_N	Output	Differential CCP2/MIPI (sub-LVDS) serial data (negative).
DATA1_P	Output	Differential MIPI (sub-LVDS) serial data 2nd lane (positive). Can be left floating when using one-lane MIPI or CCP2 serial interface.
DATA1_N	Output	Differential MIPI (sub-LVDS) serial data 2nd lane (negative). Can be left floating when using one-lane MIPI or CCP2 serial interface.
CLK_P	Output	Differential CCP2/MIPI (sub-LVDS) serial clock/strobe (positive).
CLK_N	Output	Differential CCP2/MIPI (sub-LVDS) serial clock/strobe (negative).
LINE_VALID	Output	LINE_VALID (LV) output. Qualified by PIXCLK.
FRAME_VALID	Output	FRAME_VALID (FV) output. Qualified by PIXCLK.
DOUT[11:0]	Output	Parallel pixel data output. Qualified by PIXCLK.
PIXCLK	Output	Pixel clock. Used to qualify the LV, FV, and DOUT[11:0] outputs.
FLASH	Output	Flash output. Synchronization pulse for external light source. Can be left floating if not used.
SHUTTER	Output	Control for external mechanical shutter. Can be left floating if not used.
VPP	Supply	Power supply used to program one-time programmable (OTP) memory. Disconnect pad when not programming or when feature is not used.
VDD_TX0	Supply	PHY power supply. Digital power supply for the serial data interface. Aptina recommends that VDD_TX0 is tied to VDD when the sensor is used in parallel mode.
VAA	Supply	Analog power supply.
VAA_PIX	Supply	Analog power supply for the pixel array.
AGND	Supply	Analog ground.
VDD	Supply	Digital power supply.
VDD_IO	Supply	I/O power supply.
DGND	Supply	Common ground for digital and I/O.
VDD_PLL	Supply	PLL power supply.
GND_PLL	Supply	PLL ground.
PIXGND	Supply	Pixel ground.

Figure 7: 48-Pin iLCC Package Pinout Diagram



Output Data Format

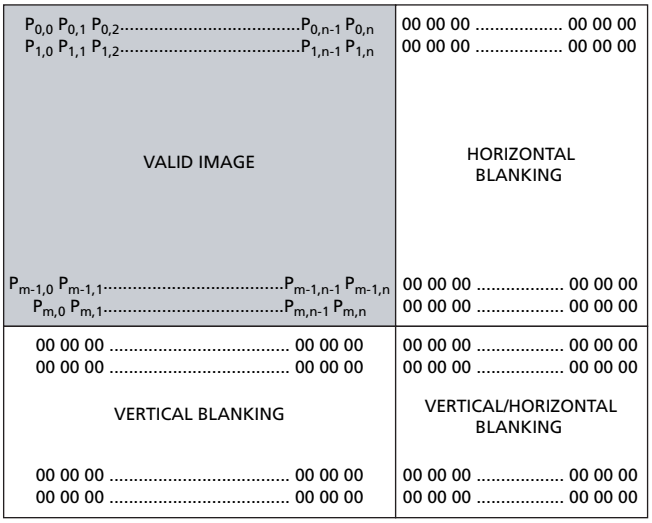
Serial Pixel Data Interface

The MT9N001 supports RAW8, RAW10, and RAW12 image data formats.

Parallel Pixel Data Interface

MT9N001 image data is read out in a progressive scan. Valid image data is surrounded by horizontal blanking and vertical blanking, as shown in Figure 8. The amount of horizontal blanking and vertical blanking is programmable; LV is HIGH during the shaded region of the figure. FV timing is described in the "Output Data Timing (Parallel Pixel Data Interface)".

Figure 8: Spatial Illustration of Image Readout



Output Data Timing (Parallel Pixel Data Interface)

MT9N001 output data is synchronized with the PIXCLK output. When LV is HIGH, one pixel value is output on the 12-bit DOUT output every PIXCLK period. The pixel clock frequency can be determined based on the sensor's master input clock and internal PLL configuration. The rising edges on the PIXCLK signal occurs one-half of a pixel clock period after transitions on LV, FV, and DOUT (see Figure 9). This allows PIXCLK to be used as a clock to sample the data. PIXCLK is continuously enabled, even during the blanking period. The MT9N001 can be programmed to delay the PIXCLK edge relative to the DOUT transitions. This can be achieved by programming the corresponding bits in the row_speed register. The parameters P, A, and Q in Figure 10 are defined in Table 4 on page 18.

Figure 9: Pixel Data Timing Example

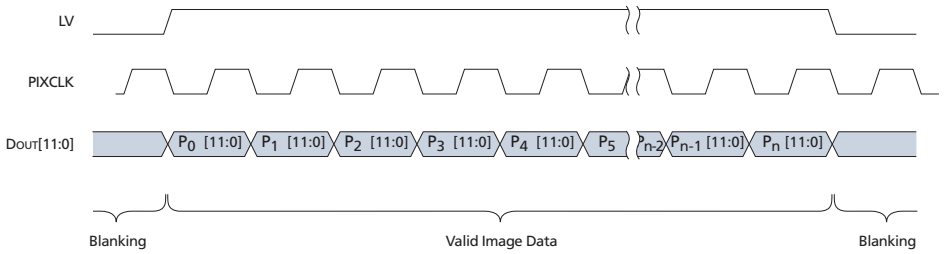
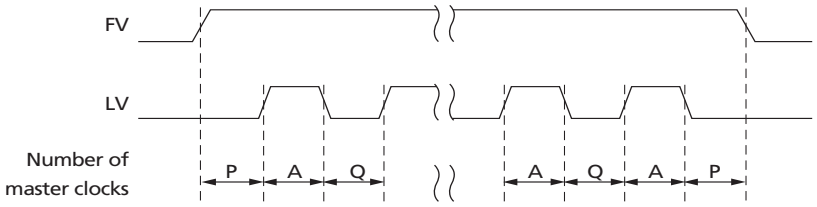


Figure 10: Row Timing and FV/LV Signals



The sensor timing (shown in Table 4) is shown in terms of pixel clock and master clock cycles (see Figure 9 on page 17). The default settings for the on-chip PLL generate 96–192 MHz output pixel clock (op_pix_clk) given a 24 MHz input clock to the MT9N001. Equations for calculating the frame rate are given in "Frame Rate Control" on page 100.

Table 4: Row Timing

Parameter	Name	Equation	Default Timing
PIXCLK_PERIOD	Pixel clock period	R0x3016–7[2:0] / vt_pix_clk_freq_mhz (vt_pix_clk set to 192 MHz)	1 pixel clock = 5.2ns
S	Skip (subsampling) factor	For x_odd_inc = y_odd_inc = 3, S = 2. For x_odd_inc = y_odd_inc = 7, S = 4. otherwise, S = 1 For y_odd_inc = 3, S = 2 For y_odd_inc = 7, S = 4 For y_odd_inc = 15, S = 8 For y_odd_inc = 31, S = 16 For y_odd_inc = 63, S = 32	1
A	Active data time	(x_addr_end – x_addr_start + x_odd_inc) * PIXCLK_PERIOD/S	3488 pixel clocks = 18.166µs
P	Frame start/end blanking	6 * PIXCLK_PERIOD	6 pixel clocks = 31.25ns
Q	Horizontal blanking	(line_length_pck – A) * PIXCLK_PERIOD	1156 pixel clocks = 6.02µs
A + Q	Row time	line_length_pck * PIXCLK_PERIOD	7006 pixel clocks = 36.49µs
N	Number of rows	(y_addr_end - y_addr_start + y_odd_inc) / S	2616 rows
V	Vertical blanking	((frame_length_lines - N) * (A + Q)) + Q – (2 * P)	596654 pixel clocks = 3.11ms
T	Frame valid time	(N * (A + Q)) - Q + (2 * P)	18326552 pixel clocks = 95.45ms
F	Total frame time	line_length_pck * frame_length_lines * PIXCLK_PERIOD	1891240 pixel clocks = 98.50ms

Two-Wire Serial Register Interface

The two-wire serial interface bus enables read/write access to control and status registers within the MT9N001. This interface is designed to be compatible with the electrical characteristics and transfer protocols of the I^2C specification.

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

The protocols described in the two-wire serial interface specification allow the slave device to drive SCLK LOW; the MT9N001 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 MT9N001 for the MIPI configured sensor are 0x6C (write address) and 0x6D (read address) in accordance with the MIPI specification. Alter-

nate slave addresses of 0x6E(write address) and 0x6F(read address) can be selected by enabling and asserting the SADDR signal through the GPI pad. But for the CCP2 configured sensor, the default slave addresses used are 0x20 (write address) and 0x21 (read address) in accordance with the SMIA specification. Also, alternate slave addresses of 0x30 (write address) and 0x31 (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.

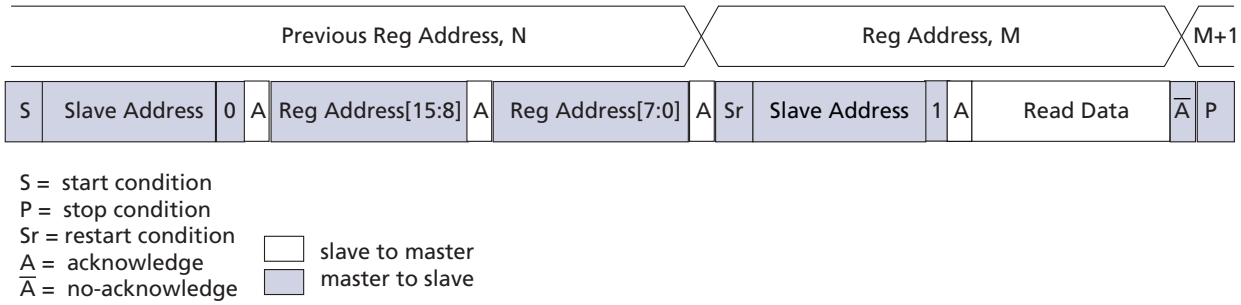
If the request was a WRITE, the master then transfers the 16-bit register address to which the WRITE should take place. This transfer takes place as two 8-bit sequences and the slave sends an acknowledge bit after each sequence to indicate that the byte has been received. The master then transfers the 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, 8 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 11 on page 21) 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 11 shows how the internal register address maintained by the MT9N001 is loaded and incremented as the sequence proceeds.

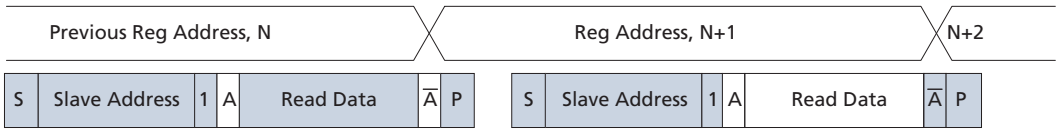
Figure 11: Single READ from Random Location



Single READ from Current Location

This sequence (Figure 12) performs a read using the current value of the MT9N001 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.

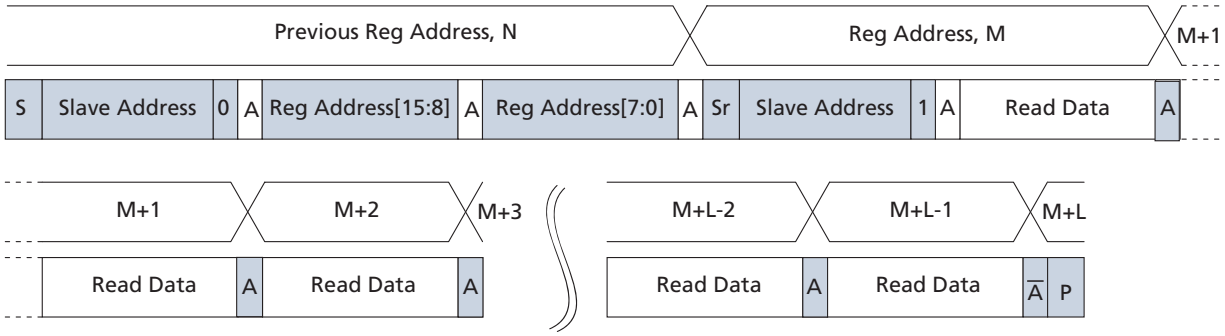
Figure 12: Single READ from Current Location



Sequential READ, Start from Random Location

This sequence (Figure 13) starts in the same way as the single READ from random location (Figure 11). 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.

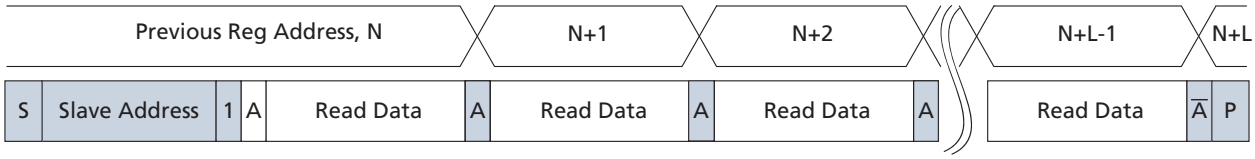
Figure 13: Sequential READ, Start from Random Location



Sequential READ, Start from Current Location

This sequence (Figure 14) starts in the same way as the single READ from current location (Figure 12 on page 21). 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.

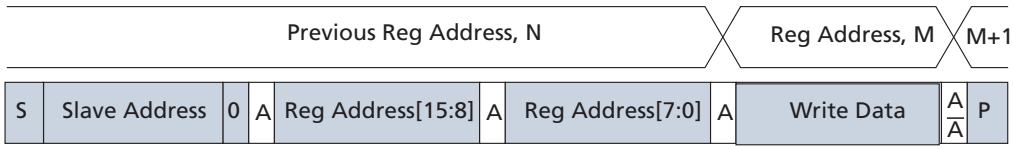
Figure 14: Sequential READ, Start from Current Location



Single WRITE to Random Location

This sequence (Figure 15) 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.

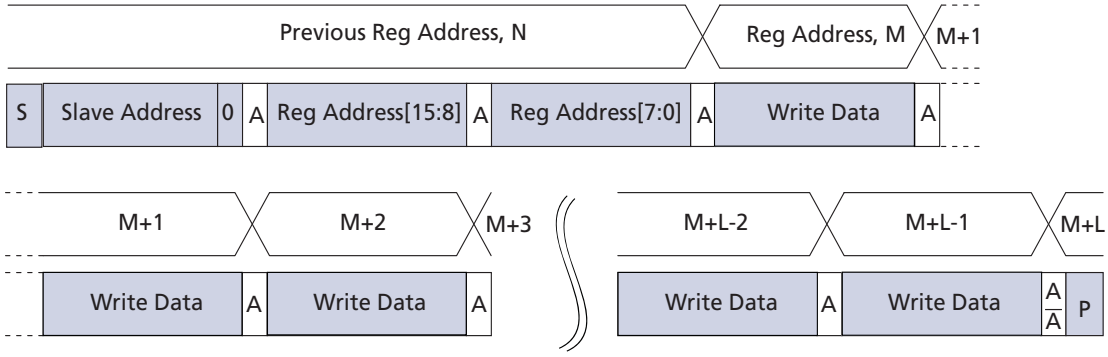
Figure 15: Single WRITE to Random Location



Sequential WRITE, Start at Random Location

This sequence (Figure 16) starts in the same way as the single WRITE to random location (Figure 15). 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 16: Sequential WRITE, Start at Random Location



Registers

The MT9N001 provides a 16-bit register address space accessed through a serial interface ("Two-Wire Serial Register Interface" on page 19). 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 5: 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 MT9N001 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, 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 MT9N001 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) only has 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 1 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 0x2600. This means that a READ from address 0x0000 would return 0x26, and a READ from address 0x0001 would return 0x00. When reading this register as two 8-bit transfers on the serial interface, the 0x26 will appear on the serial interface first, followed by the 0x00.

Address Alignment

All register addresses are aligned naturally. Registers that occupy two bytes of address space are aligned to even 16-bit addresses, and registers that occupy four 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 6: 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 MT9N001 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 "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 MT9N001 is in streaming mode, write "1" to this register before making changes to any group of registers where a set of changes is required to take effect simultaneously. When this register is set 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 not 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 reset_register[14] bit.

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 26.
- The PLL timing registers are not double-buffered because the result of changing them in streaming mode is undefined. Therefore, the pending and live values for these registers are equivalent.

For further details, see "Embedded Data Format and Control" on page 115.

Register List and Default Values

Table 7 through Table 9 on page 32 lists sensor registers and their default values.

Table 7: Register List and Default—SMIA Configuration

1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Name	Data Format (Binary)	Default Value Dec (Hex)
R0(R0x0000)	model_id	dddd dddd dddd dddd	11009 (0x2B01)
R2(R0x0002)	revision_number	0000 0000 dddd dddd	0 (0x0000)
R3(R0x0003)	manufacturer_id	0000 0000 ???? ????	6 (0x0006)
R4(R0x0004)	smia_version	0000 0000 ???? ????	10 (0x000A)
R5(R0x0005)	frame_count	0000 0000 ???? ????	255 (0x00FF)
R6(R0x0006)	pixel_order	0000 0000 0000 00??	0 (0x0000)
R8(R0x0008)	data_pedestal	0000 dddd dddd dddd	168 (0x00A8)
R64(R0x0040)	frame_format_model_type	0000 0000 ???? ????	1 (0x0001)
R65(R0x0041)	frame_format_model_subtype	0000 0000 ???? ????	18 (0x0012)
R66(R0x0042)	frame_format_descriptor_0	???? ???? ???? ????	23968 (0x5DA0)
R68(R0x0044)	frame_format_descriptor_1	???? ???? ???? ????	4098 (0x1002)
R70(R0x0046)	frame_format_descriptor_2	???? ???? ???? ????	23096 (0x5A38)
R72(R0x0048)	frame_format_descriptor_3	???? ???? ???? ????	0 (0x0000)
R74(R0x004A)	frame_format_descriptor_4	???? ???? ???? ????	0 (0x0000)
R76(R0x004C)	frame_format_descriptor_5	???? ???? ???? ????	0 (0x0000)
R78(R0x004E)	frame_format_descriptor_6	???? ???? ???? ????	0 (0x0000)

Table 7: Register List and Default—SMIA Configuration (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Name	Data Format (Binary)	Default Value Dec (Hex)
R80(R0x0050)	frame_format_descriptor_7	???? ???? ???? ????	0 (0x0000)
R82(R0x0052)	frame_format_descriptor_8	???? ???? ???? ????	0 (0x0000)
R84(R0x0054)	frame_format_descriptor_9	???? ???? ???? ????	0 (0x0000)
R86(R0x0056)	frame_format_descriptor_10	???? ???? ???? ????	0 (0x0000)
R88(R0x0058)	frame_format_descriptor_11	???? ???? ???? ????	0 (0x0000)
R90(R0x005A)	frame_format_descriptor_12	???? ???? ???? ????	0 (0x0000)
R92(R0x005C)	frame_format_descriptor_13	???? ???? ???? ????	0 (0x0000)
R94(R0x005E)	frame_format_descriptor_14	???? ???? ???? ????	0 (0x0000)
R128(R0x0080)	analogue_gain_capability	???? ???? ???? ????	1 (0x0001)
R132(R0x0084)	analogue_gain_code_min	???? ???? ???? ????	8 (0x0008)
R134(R0x0086)	analogue_gain_code_max	???? ???? ???? ????	191 (0x00BF)
R136(R0x0088)	analogue_gain_code_step	???? ???? ???? ????	1 (0x0001)
R138(R0x008A)	analogue_gain_type	???? ???? ???? ????	0 (0x0000)
R140(R0x008C)	analogue_gain_m0	???? ???? ???? ????	1 (0x0001)
R142(R0x008E)	analogue_gain_c0	???? ???? ???? ????	0 (0x0000)
R144(R0x0090)	analogue_gain_m1	???? ???? ???? ????	0 (0x0000)
R146(R0x0092)	analogue_gain_c1	???? ???? ???? ????	8 (0x0008)
R192(R0x00C0)	data_format_model_type	0000 0000 ???? ????	1 (0x0001)
R193(R0x00C1)	data_format_model_subtype	0000 0000 ???? ????	5 (0x0005)
R194(R0x00C2)	data_format_descriptor_0	???? ???? ???? ????	2570 (0x0A0A)
R196(R0x00C4)	data_format_descriptor_1	???? ???? ???? ????	2056 (0x0808)
R198(R0x00C6)	data_format_descriptor_2	???? ???? ???? ????	2568 (0x0A08)
R200(R0x00C8)	data_format_descriptor_3	???? ???? ???? ????	3084 (0x0C0C)
R202(R0x00CA)	data_format_descriptor_4	???? ???? ???? ????	3080 (0x0C08)
R204(R0x00CC)	data_format_descriptor_5	???? ???? ???? ????	0 (0x0000)
R206(R0x00CE)	data_format_descriptor_6	???? ???? ???? ????	0 (0x0000)
R256(R0x0100)	mode_select	0000 0000 0000 000d	0 (0x0000)
R257(R0x0101)	image_orientation	0000 0000 0000 00dd	0 (0x0000)
R259(R0x0103)	software_reset	0000 0000 0000 000d	0 (0x0000)
R260(R0x0104)	grouped_parameter_hold	0000 0000 0000 000d	0 (0x0000)
R261(R0x0105)	mask_corrupted_frames	0000 0000 0000 000d	0 (0x0000)
R272(R0x0110)	ccp2_channel_identifier	0000 0000 0000 0ddd	0 (0x0000)
R273(R0x0111)	ccp2_signalling_mode	0000 0000 0000 000d	1 (0x0001)
R274(R0x0112)	ccp_data_format	dddd dddd dddd dddd	2056 (0x0808)
R288(R0x0120)	gain_mode	0000 0000 0000 000d	0 (0x0000)
R512(R0x0200)	fine_integration_time	dddd dddd dddd ddd0	1386 (0x056A)
R514(R0x0202)	coarse_integration_time	dddd dddd dddd dddd	16 (0x0010)
R516(R0x0204)	analogue_gain_code_global	0000 0000 dddd dddd	8 (0x0008)
R518(R0x0206)	analogue_gain_code_greenR	0000 0000 dddd dddd	8 (0x0008)

Table 7: Register List and Default—SMIA Configuration (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Name	Data Format (Binary)	Default Value Dec (Hex)
R520(R0x0208)	analogue_gain_code_red	0000 0000 dddd dddd	8 (0x0008)
R522(R0x020A)	analogue_gain_code_blue	0000 0000 dddd dddd	8 (0x0008)
R524(R0x020C)	analogue_gain_code_greenB	0000 0000 dddd dddd	8 (0x0008)
R526(R0x020E)	digital_gain_greenR	0000 0ddd 0000 0000	256 (0x0100)
R528(R0x0210)	digital_gain_red	0000 0ddd 0000 0000	256 (0x0100)
R530(R0x0212)	digital_gain_blue	0000 0ddd 0000 0000	256 (0x0100)
R532(R0x0214)	digital_gain_greenB	0000 0ddd 0000 0000	256 (0x0100)
R768(R0x0300)	vt_pix_clk_div	0000 0000 0000 dddd	4 (0x0004)
R770(R0x0302)	vt_sys_clk_div	0000 0000 000d dddd	1 (0x0001)
R772(R0x0304)	pre_pll_clk_div	0000 0000 00dd dddd	2 (0x0002)
R774(R0x0306)	pll_multiplier	0000 0000 dddd dddd	64 (0x0040)
R776(R0x0308)	op_pix_clk_div	0000 0000 000d dddd	8 (0x0008)
R778(R0x030A)	op_sys_clk_div	0000 0000 000d dddd	1 (0x0001)
R832(R0x0340)	frame_length_lines	dddd dddd dddd dddd	2759 (0x0AC7)
R834(R0x0342)	line_length_pck	dddd dddd dddd ddd0	7324 (0x1C9C)
R836(R0x0344)	x_addr_start	0000 dddd dddd dddd	0 (0x0000)
R838(R0x0346)	y_addr_start	0000 dddd dddd dddd	8 (0x0008)
R840(R0x0348)	x_addr_end	0000 dddd dddd dddd	3487 (0x0D9F)
R842(R0x034A)	y_addr_end	0000 dddd dddd dddd	2623 (0x0A3F)
R844(R0x034C)	x_output_size	0000 dddd dddd ddd0	3488 (0x0DA0)
R846(R0x034E)	y_output_size	0000 dddd dddd ddd0	2616 (0x0A38)
R896(R0x0380)	x_even_inc	0000 0000 0000 000?	1 (0x0001)
R898(R0x0382)	x_odd_inc	0000 0000 0000 0ddd	1 (0x0001)
R900(R0x0384)	y_even_inc	0000 0000 0000 000?	1 (0x0001)
R902(R0x0386)	y_odd_inc	0000 0000 00dd dddd	1 (0x0001)
R1024(R0x0400)	scaling_mode	0000 0000 0000 00dd	0 (0x0000)
R1026(R0x0402)	spatial_sampling	0000 0000 0000 000d	0 (0x0000)
R1028(R0x0404)	scale_m	0000 0000 dddd dddd	16 (0x0010)
R1030(R0x0406)	scale_n	0000 0000 ???? ????	16 (0x0010)
R1280(R0x0500)	compression_mode	0000 0000 0000 000?	1 (0x0001)
R1536(R0x0600)	test_pattern_mode	0000 000d 0000 0ddd	0 (0x0000)
R1538(R0x0602)	test_data_red	0000 dddd dddd dddd	0 (0x0000)
R1540(R0x0604)	test_data_greenR	0000 dddd dddd dddd	0 (0x0000)
R1542(R0x0606)	test_data_blue	0000 dddd dddd dddd	0 (0x0000)
R1544(R0x0608)	test_data_greenB	0000 dddd dddd dddd	0 (0x0000)
R1546(R0x060A)	horizontal_cursor_width	0000 dddd dddd dddd	0 (0x0000)
R1548(R0x060C)	horizontal_cursor_position	0000 dddd dddd dddd	0 (0x0000)
R1550(R0x060E)	vertical_cursor_width	0000 dddd dddd dddd	0 (0x0000)
R1552(R0x0610)	vertical_cursor_position	0000 dddd dddd dddd	0 (0x0000)

Table 8: Register List and Default Values—SMIA Parameter Limits

1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Name	Data Format (Binary)	Default Value Dec (Hex)
R4096(R0x1000)	integration_time_capability	0000 0000 0000 000?	1 (0x0001)
R4100(R0x1004)	coarse_integration_time_min	dddd dddd dddd dddd	0 (0x0000)
R4102(R0x1006)	coarse_integration_time_max_margin	dddd dddd dddd dddd	1 (0x0001)
R4104(R0x1008)	fine_integration_time_min	dddd dddd dddd dddd	1386 (0x056A)
R4106(R0x100A)	fine_integration_time_max_margin	dddd dddd dddd dddd	934 (0x03A6)
R4224(R0x1080)	digital_gain_capability	0000 0000 0000 000?	1 (0x0001)
R4228(R0x1084)	digital_gain_min	???? ???? ???? ????	256 (0x0100)
R4230(R0x1086)	digital_gain_max	???? ???? ???? ????	1792 (0x0700)
R4232(R0x1088)	digital_gain_step_size	???? ???? ???? ????	256 (0x0100)
R4352(R0x1100)	min_ext_clk_freq_mhz_1	???? ???? ???? ????	16384 (0x4000)
R4354(R0x1102)	min_ext_clk_freq_mhz_2	???? ???? ???? ????	0 (0x0000)
R4356(R0x1104)	max_ext_clk_freq_mhz_1	???? ???? ???? ????	17024 (0x4280)
R4358(R0x1106)	max_ext_clk_freq_mhz_2	???? ???? ???? ????	0 (0x0000)
R4360(R0x1108)	min_pre_pll_clk_div	???? ???? ???? ????	1 (0x0001)
R4362(R0x110A)	max_pre_pll_clk_div	???? ???? ???? ????	64 (0x0040)
R4364(R0x110C)	min_pll_ip_freq_mhz_1	???? ???? ???? ????	16512 (0x4080)
R4366(R0x110E)	min_pll_ip_freq_mhz_2	???? ???? ???? ????	0 (0x0000)
R4368(R0x1110)	max_pll_ip_freq_mhz_1	???? ???? ???? ????	16832 (0x41C0)
R4370(R0x1112)	max_pll_ip_freq_mhz_2	???? ???? ???? ????	0 (0x0000)
R4372(R0x1114)	min_pll_multiplier	???? ???? ???? ????	32 (0x0020)
R4374(R0x1116)	max_pll_multiplier	???? ???? ???? ????	384 (0x0180)
R4376(R0x1118)	min_pll_op_freq_mhz_1	???? ???? ???? ????	17344 (0x43C0)
R4378(R0x111A)	min_pll_op_freq_mhz_2	???? ???? ???? ????	0 (0x0000)
R4380(R0x111C)	max_pll_op_freq_mhz_1	???? ???? ???? ????	17472 (0x4440)
R4382(R0x111E)	max_pll_op_freq_mhz_2	???? ???? ???? ????	0 (0x0000)
R4384(R0x1120)	min_vt_sys_clk_div	???? ???? ???? ????	1 (0x0001)
R4386(R0x1122)	max_vt_sys_clk_div	???? ???? ???? ????	1 (0x0001)
R4388(R0x1124)	min_vt_sys_clk_freq_mhz_1	???? ???? ???? ????	16832 (0x41C0)
R4390(R0x1126)	min_vt_sys_clk_freq_mhz_2	???? ???? ???? ????	0 (0x0000)
R4392(R0x1128)	max_vt_sys_clk_freq_mhz_1	???? ???? ???? ????	17472 (0x4440)
R4394(R0x112A)	max_vt_sys_clk_freq_mhz_2	???? ???? ???? ????	0 (0x0000)
R4396(R0x112C)	min_vt_pix_clk_freq_mhz_1	???? ???? ???? ????	16409 (0x4019)
R4398(R0x112E)	min_vt_pix_clk_freq_mhz_2	???? ???? ???? ????	39322 (0x999A)
R4400(R0x1130)	max_vt_pix_clk_freq_mhz_1	???? ???? ???? ????	17216 (0x4340)
R4402(R0x1132)	max_vt_pix_clk_freq_mhz_2	???? ???? ???? ????	0 (0x0000)
R4404(R0x1134)	min_vt_pix_clk_div	???? ???? ???? ????	4 (0x0004)
R4406(R0x1136)	max_vt_pix_clk_div	???? ???? ???? ????	6 (0x0006)
R4416(R0x1140)	min_frame_length_lines	dddd dddd dddd dddd	145 (0x0091)
R4418(R0x1142)	max_frame_length_lines	dddd dddd dddd dddd	65535 (0xFFFF)

Table 8: Register List and Default Values—SMIA Parameter Limits (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Name	Data Format (Binary)	Default Value Dec (Hex)
R4420(R0x1144)	min_line_length_pck	dddd dddd dddd dddd	2320 (0x0910)
R4422(R0x1146)	max_line_length_pck	dddd dddd dddd dddd	65534 (0xFFFE)
R4424(R0x1148)	min_line_blanking_pck	dddd dddd dddd dddd	1764 (0x06E4)
R4426(R0x114A)	min_frame_blanking_lines	dddd dddd dddd dddd	143 (0x008F)
R4448(R0x1160)	min_op_sys_clk_div	???? ???? ???? ????	1 (0x0001)
R4450(R0x1162)	max_op_sys_clk_div	???? ???? ???? ????	1 (0x0001)
R4452(R0x1164)	min_op_sys_clk_freq_mhz_1	???? ???? ???? ????	16793 (0x4199)
R4454(R0x1166)	min_op_sys_clk_freq_mhz_2	???? ???? ???? ????	39322 (0x999A)
R4456(R0x1168)	max_op_sys_clk_freq_mhz_1	???? ???? ???? ????	17472 (0x4440)
R4458(R0x116A)	max_op_sys_clk_freq_mhz_2	???? ???? ???? ????	0 (0x0000)
R4460(R0x116C)	min_op_pix_clk_div	???? ???? ???? ????	8 (0x0008)
R4462(R0x116E)	max_op_pix_clk_div	???? ???? ???? ????	12 (0x000C)
R4464(R0x1170)	min_op_pix_clk_freq_mhz_1	???? ???? ???? ????	16409 (0x4019)
R4466(R0x1172)	min_op_pix_clk_freq_mhz_2	???? ???? ???? ????	39322 (0x999A)
R4468(R0x1174)	max_op_pix_clk_freq_mhz_1	???? ???? ???? ????	17216 (0x4340)
R4470(R0x1176)	max_op_pix_clk_freq_mhz_2	???? ???? ???? ????	0 (0x0000)
R4480(R0x1180)	x_addr_min	???? ???? ???? ????	0 (0x0000)
R4482(R0x1182)	y_addr_min	???? ???? ???? ????	0 (0x0000)
R4484(R0x1184)	x_addr_max	???? ???? ???? ????	3503 (0x0DAF)
R4486(R0x1186)	y_addr_max	???? ???? ???? ????	2631 (0x0A47)
R4544(R0x11C0)	min_even_inc	???? ???? ???? ????	1 (0x0001)
R4546(R0x11C2)	max_even_inc	???? ???? ???? ????	1 (0x0001)
R4548(R0x11C4)	min_odd_inc	???? ???? ???? ????	1 (0x0001)
R4550(R0x11C6)	max_odd_inc	???? ???? ???? ????	3 (0x0003)
R4608(R0x1200)	scaling_capability	0000 0000 0000 00??	2 (0x0002)
R4612(R0x1204)	scaler_m_min	???? ???? ???? ????	16 (0x0010)
R4614(R0x1206)	scaler_m_max	???? ???? ???? ????	128 (0x0080)
R4616(R0x1208)	scaler_n_min	???? ???? ???? ????	16 (0x0010)
R4618(R0x120A)	scaler_n_max	???? ???? ???? ????	16 (0x0010)
R4864(R0x1300)	compression_capability	0000 0000 0000 000?	1 (0x0001)
R5120(R0x1400)	matrix_element_redinred	dddd dddd dddd dddd	578 (0x0242)
R5122(R0x1402)	matrix_element_greeninred	dddd dddd dddd dddd	65280 (0xFF00)
R5124(R0x1404)	matrix_element_blueinred	dddd dddd dddd dddd	65470 (0xFFBE)
R5126(R0x1406)	matrix_element_redingreen	dddd dddd dddd dddd	65460 (0xFFB4)
R5128(R0x1408)	matrix_element_greeningreen	dddd dddd dddd dddd	512 (0x0200)
R5130(R0x140A)	matrix_element_blueingreen	dddd dddd dddd dddd	65357 (0xFF4D)
R5132(R0x140C)	matrix_element_redinblue	dddd dddd dddd dddd	65521 (0xFFF1)
R5134(R0x140E)	matrix_element_greeninblue	dddd dddd dddd dddd	65332 (0xFF34)
R5136(R0x1410)	matrix_element_blueinblue	dddd dddd dddd dddd	476 (0x01DC)

Table 9: Register List and Default Values—Manufacturer-Specific

1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Name	Data Format (Binary)	Default Value Dec (Hex)
R12288(R0x3000)	model_id_	dddd dddd dddd dddd	11009 (0x2B01)
R12290(R0x3002)	y_addr_start_	0000 dddd dddd dddd	8 (0x0008)
R12292(R0x3004)	x_addr_start_	0000 dddd dddd dddd	0 (0x0000)
R12294(R0x3006)	y_addr_end_	0000 dddd dddd dddd	2623 (0x0A3F)
R12296(R0x3008)	x_addr_end_	0000 dddd dddd dddd	3487 (0x0D9F)
R12298(R0x300A)	frame_length_lines_	dddd dddd dddd dddd	2759 (0x0AC7)
R12300(R0x300C)	line_length_pck_	dddd dddd dddd ddd0	7324 (0x1C9C)
R12304(R0x3010)	fine_correction	0ddd dddd dddd dddd	256 (0x0100)
R12306(R0x3012)	coarse_integration_time_	dddd dddd dddd dddd	16 (0x0010)
R12308(R0x3014)	fine_integration_time_	dddd dddd dddd ddd0	1386 (0x056A)
R12310(R0x3016)	row_speed	0000 0ddd 0ddd 0ddd	273 (0x0111)
R12312(R0x3018)	extra_delay	dddd dddd dddd ddd0	0 (0x0000)
R12314(R0x301A)	reset_register	dd0d 0ddd dddd dddd	88 (0x0058)
R12316(R0x301C)	mode_select_	0000 0000 0000 000d	0 (0x0000)
R12317(R0x301D)	image_orientation_	0000 0000 0000 00dd	0 (0x0000)
R12318(R0x301E)	data_pedestal_	0000 dddd dddd dddd	168 (0x00A8)
R12321(R0x3021)	software_reset_	0000 0000 0000 000d	0 (0x0000)
R12322(R0x3022)	grouped_parameter_hold_	0000 0000 0000 000d	0 (0x0000)
R12323(R0x3023)	mask_corrupted_frames_	0000 0000 0000 000d	0 (0x0000)
R12324(R0x3024)	pixel_order_	0000 0000 0000 00??	0 (0x0000)
R12326(R0x3026)	gpi_status	dddd dddd dddd ????	65535 (0xFFFF)
R12328(R0x3028)	analogue_gain_code_global_	0000 0000 dddd dddd	8 (0x0008)
R12330(R0x302A)	analogue_gain_code_greenR_	0000 0000 dddd dddd	8 (0x0008)
R12332(R0x302C)	analogue_gain_code_red_	0000 0000 dddd dddd	8 (0x0008)
R12334(R0x302E)	analogue_gain_code_blue_	0000 0000 dddd dddd	8 (0x0008)
R12336(R0x3030)	analogue_gain_code_greenB	0000 0000 dddd dddd	8 (0x0008)
R12338(R0x3032)	digital_gain_greenR_	0000 0ddd 0000 0000	256 (0x0100)
R12340(R0x3034)	digital_gain_red_	0000 0ddd 0000 0000	256 (0x0100)
R12342(R0x3036)	digital_gain_blue_	0000 0ddd 0000 0000	256 (0x0100)
R12344(R0x3038)	digital_gain_greenB_	0000 0ddd 0000 0000	256 (0x0100)
R12346(R0x303A)	smia_version_	0000 0000 ???? ????	10 (0x000A)
R12347(R0x303B)	frame_count_	0000 0000 ???? ????	255 (0x00FF)
R12348(R0x303C)	frame_status	0000 0000 0000 00??	0 (0x0000)
R12352(R0x3040)	read_mode	dddd dddd dddd dddd	65 (0x0041)
R12358(R0x3046)	flash	??dd dddd d000 0000	1536 (0x0600)
R12360(R0x3048)	flash_count	0000 00dd dddd dddd	8 (0x0008)
R12374(R0x3056)	greenR_gain	0ddd 00dd dddd dddd	4128 (0x1020)
R12376(R0x3058)	blue_gain	0ddd 00dd dddd dddd	4128 (0x1020)
R12378(R0x305A)	red_gain	0ddd 00dd dddd dddd	4128 (0x1020)

Table 9: Register List and Default Values—Manufacturer-Specific (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Name	Data Format (Binary)	Default Value Dec (Hex)
R12380(R0x305C)	greenB_gain	0ddd 00dd dddd dddd	4128 (0x1020)
R12382(R0x305E)	global_gain	0ddd 00dd dddd dddd	4128 (0x1020)
R12394(R0x306A)	datapath_status	0000 0000 00?d dddd	0 (0x0000)
R12398(R0x306E)	datapath_select	dddd dd00 ?00d 00dd	36864 (0x9000)
R12400(R0x3070)	test_pattern_mode_	0000 000d 0000 0ddd	0 (0x0000)
R12402(R0x3072)	test_data_red_	0000 dddd dddd dddd	0 (0x0000)
R12404(R0x3074)	test_data_greenR_	0000 dddd dddd dddd	0 (0x0000)
R12406(R0x3076)	test_data_blue_	0000 dddd dddd dddd	0 (0x0000)
R12408(R0x3078)	test_data_greenB_	0000 dddd dddd dddd	0 (0x0000)
R12410(R0x307A)	test_raw_mode	0000 0000 0000 00dd	0 (0x0000)
R12448(R0x30A0)	x_even_inc_	0000 0000 0000 000?	1 (0x0001)
R12450(R0x30A2)	x_odd_inc_	0000 0000 0000 0ddd	1 (0x0001)
R12452(R0x30A4)	y_even_inc_	0000 0000 0000 000?	1 (0x0001)
R12454(R0x30A6)	y_odd_inc_	0000 0000 00dd dddd	1 (0x0001)
R12456(R0x30A8)	calib_greenR_asc1	0000 000d dddd dddd	0 (0x0000)
R12458(R0x30AA)	calib_blue_asc1	0000 000d dddd dddd	0 (0x0000)
R12460(R0x30AC)	calib_red_asc1	0000 000d dddd dddd	0 (0x0000)
R12462(R0x30AE)	calib_greenB_asc1	0000 000d dddd dddd	0 (0x0000)
R12470(R0x30B6)	dark_greenR_average	???? ???? ???? ????	0 (0x0000)
R12472(R0x30B8)	dark_blue_average	???? ???? ???? ????	0 (0x0000)
R12474(R0x30BA)	dark_red_average	???? ???? ???? ????	0 (0x0000)
R12476(R0x30BC)	dark_greenB_average	???? ???? ???? ????	0 (0x0000)
R12480(R0x30C0)	calib_control	000d 00dd dddd dddd	32 (0x0020)
R12482(R0x30C2)	calib_greenR	0000 000d dddd dddd	0 (0x0000)
R12484(R0x30C4)	calib_blue	0000 000d dddd dddd	0 (0x0000)
R12486(R0x30C6)	calib_red	0000 000d dddd dddd	0 (0x0000)
R12488(R0x30C8)	calib_greenB	0000 000d dddd dddd	0 (0x0000)
R12640(R0x3160)	global_seq_trigger	0000 00?? 0000 0ddd	0 (0x0000)
R12642(R0x3162)	global_rst_end	dddd dddd dddd dddd	152 (0x0098)
R12644(R0x3164)	global_shutter_start	dddd dddd dddd dddd	159 (0x009F)
R12646(R0x3166)	global_read_start	dddd dddd dddd dddd	160 (0x00A0)
R12704(R0x31A0)	serial_format_descriptor_0	???? ???? ???? ????	257 (0x0101)
R12706(R0x31A2)	serial_format_descriptor_1	???? ???? ???? ????	513 (0x0201)
R12708(R0x31A4)	serial_format_descriptor_2	???? ???? ???? ????	514 (0x0202)
R12710(R0x31A6)	serial_format_descriptor_3	???? ???? ???? ????	0 (0x0000)
R12712(R0x31A8)	serial_format_descriptor_4	???? ???? ???? ????	0 (0x0000)
R12714(R0x31AA)	serial_format_descriptor_5	???? ???? ???? ????	0 (0x0000)
R12716(R0x31AC)	serial_format_descriptor_6	???? ???? ???? ????	0 (0x0000)
R12718(R0x31AE)	serial_format	0000 00dd 0000 00dd	1 (0x0001)

Table 9: Register List and Default Values—Manufacturer-Specific (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Name	Data Format (Binary)	Default Value Dec (Hex)
R12720(R0x31B0)	frame_preamble	0000 0000 dddd dddd	91 (0x005B)
R12722(R0x31B2)	line_preamble	0000 0000 dddd dddd	48 (0x0030)
R12724(R0x31B4)	mipi_timing_0	???? dddd dddd dddd	3415 (0x0D57)
R12726(R0x31B6)	mipi_timing_1	??dd dddd ??dd dddd	2832 (0x0B10)
R12728(R0x31B8)	mipi_timing_2	??dd dddd ??dd dddd	269 (0x010D)
R12730(R0x31BA)	mipi_timing_3	??dd dddd ?ddd dddd	1293 (0x050D)
R12732(R0x31BC)	mipi_timing_4	???? ???? ?ddd dddd	11 (0x000B)
R12776(R0x31E8)	horizontal_cursor_position_	0000 dddd dddd dddd	0 (0x0000)
R12778(R0x31EA)	vertical_cursor_position_	0000 dddd dddd dddd	0 (0x0000)
R12780(R0x31EC)	horizontal_cursor_width_	0000 dddd dddd dddd	0 (0x0000)
R12782(R0x31EE)	vertical_cursor_width_	0000 dddd dddd dddd	0 (0x0000)
R12786(R0x31F2)	i2c_ids_mipi_default	dddd dddd dddd dddd	28268 (0x6E6C)
R12796(R0x31FC)	i2c_ids	dddd dddd dddd dddd	12320 (0x3020)
R13824(R0x3600)	p_gr_p0q0	dddd dddd dddd dddd	0 (0x0000)
R13826(R0x3602)	p_gr_p0q1	dddd dddd dddd dddd	0 (0x0000)
R13828(R0x3604)	p_gr_p0q2	dddd dddd dddd dddd	0 (0x0000)
R13830(R0x3606)	p_gr_p0q3	dddd dddd dddd dddd	0 (0x0000)
R13832(R0x3608)	p_gr_p0q4	dddd dddd dddd dddd	0 (0x0000)
R13834(R0x360A)	p_rd_p0q0	dddd dddd dddd dddd	0 (0x0000)
R13836(R0x360C)	p_rd_p0q1	dddd dddd dddd dddd	0 (0x0000)
R13838(R0x360E)	p_rd_p0q2	dddd dddd dddd dddd	0 (0x0000)
R13840(R0x3610)	p_rd_p0q3	dddd dddd dddd dddd	0 (0x0000)
R13842(R0x3612)	p_rd_p0q4	dddd dddd dddd dddd	0 (0x0000)
R13844(R0x3614)	p_bl_p0q0	dddd dddd dddd dddd	0 (0x0000)
R13846(R0x3616)	p_bl_p0q1	dddd dddd dddd dddd	0 (0x0000)
R13848(R0x3618)	p_bl_p0q2	dddd dddd dddd dddd	0 (0x0000)
R13850(R0x361A)	p_bl_p0q3	dddd dddd dddd dddd	0 (0x0000)
R13852(R0x361C)	p_bl_p0q4	dddd dddd dddd dddd	0 (0x0000)
R13854(R0x361E)	p_gb_p0q0	dddd dddd dddd dddd	0 (0x0000)
R13856(R0x3620)	p_gb_p0q1	dddd dddd dddd dddd	0 (0x0000)
R13858(R0x3622)	p_gb_p0q2	dddd dddd dddd dddd	0 (0x0000)
R13860(R0x3624)	p_gb_p0q3	dddd dddd dddd dddd	0 (0x0000)
R13862(R0x3626)	p_gb_p0q4	dddd dddd dddd dddd	0 (0x0000)
R13888(R0x3640)	p_gr_p1q0	dddd dddd dddd dddd	0 (0x0000)
R13890(R0x3642)	p_gr_p1q1	dddd dddd dddd dddd	0 (0x0000)
R13892(R0x3644)	p_gr_p1q2	dddd dddd dddd dddd	0 (0x0000)
R13894(R0x3646)	p_gr_p1q3	dddd dddd dddd dddd	0 (0x0000)
R13896(R0x3648)	p_gr_p1q4	dddd dddd dddd dddd	0 (0x0000)
R13898(R0x364A)	p_rd_p1q0	dddd dddd dddd dddd	0 (0x0000)

Table 9: Register List and Default Values—Manufacturer-Specific (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Name	Data Format (Binary)	Default Value Dec (Hex)
R13900(R0x364C)	p_rd_p1q1	dddd dddd dddd dddd	0 (0x0000)
R13902(R0x364E)	p_rd_p1q2	dddd dddd dddd dddd	0 (0x0000)
R13904(R0x3650)	p_rd_p1q3	dddd dddd dddd dddd	0 (0x0000)
R13906(R0x3652)	p_rd_p1q4	dddd dddd dddd dddd	0 (0x0000)
R13908(R0x3654)	p_bl_p1q0	dddd dddd dddd dddd	0 (0x0000)
R13910(R0x3656)	p_bl_p1q1	dddd dddd dddd dddd	0 (0x0000)
R13912(R0x3658)	p_bl_p1q2	dddd dddd dddd dddd	0 (0x0000)
R13914(R0x365A)	p_bl_p1q3	dddd dddd dddd dddd	0 (0x0000)
R13916(R0x365C)	p_bl_p1q4	dddd dddd dddd dddd	0 (0x0000)
R13918(R0x365E)	p_gb_p1q0	dddd dddd dddd dddd	0 (0x0000)
R13920(R0x3660)	p_gb_p1q1	dddd dddd dddd dddd	0 (0x0000)
R13922(R0x3662)	p_gb_p1q2	dddd dddd dddd dddd	0 (0x0000)
R13924(R0x3664)	p_gb_p1q3	dddd dddd dddd dddd	0 (0x0000)
R13926(R0x3666)	p_gb_p1q4	dddd dddd dddd dddd	0 (0x0000)
R13952(R0x3680)	p_gr_p2q0	dddd dddd dddd dddd	0 (0x0000)
R13954(R0x3682)	p_gr_p2q1	dddd dddd dddd dddd	0 (0x0000)
R13956(R0x3684)	p_gr_p2q2	dddd dddd dddd dddd	0 (0x0000)
R13958(R0x3686)	p_gr_p2q3	dddd dddd dddd dddd	0 (0x0000)
R13960(R0x3688)	p_gr_p2q4	dddd dddd dddd dddd	0 (0x0000)
R13962(R0x368A)	p_rd_p2q0	dddd dddd dddd dddd	0 (0x0000)
R13964(R0x368C)	p_rd_p2q1	dddd dddd dddd dddd	0 (0x0000)
R13966(R0x368E)	p_rd_p2q2	dddd dddd dddd dddd	0 (0x0000)
R13968(R0x3690)	p_rd_p2q3	dddd dddd dddd dddd	0 (0x0000)
R13970(R0x3692)	p_rd_p2q4	dddd dddd dddd dddd	0 (0x0000)
R13972(R0x3694)	p_bl_p2q0	dddd dddd dddd dddd	0 (0x0000)
R13974(R0x3696)	p_bl_p2q1	dddd dddd dddd dddd	0 (0x0000)
R13976(R0x3698)	p_bl_p2q2	dddd dddd dddd dddd	0 (0x0000)
R13978(R0x369A)	p_bl_p2q3	dddd dddd dddd dddd	0 (0x0000)
R13980(R0x369C)	p_bl_p2q4	dddd dddd dddd dddd	0 (0x0000)
R13982(R0x369E)	p_gb_p2q0	dddd dddd dddd dddd	0 (0x0000)
R13984(R0x36A0)	p_gb_p2q1	dddd dddd dddd dddd	0 (0x0000)
R13986(R0x36A2)	p_gb_p2q2	dddd dddd dddd dddd	0 (0x0000)
R13988(R0x36A4)	p_gb_p2q3	dddd dddd dddd dddd	0 (0x0000)
R13990(R0x36A6)	p_gb_p2q4	dddd dddd dddd dddd	0 (0x0000)
R14016(R0x36C0)	p_gr_p3q0	dddd dddd dddd dddd	0 (0x0000)
R14018(R0x36C2)	p_gr_p3q1	dddd dddd dddd dddd	0 (0x0000)
R14020(R0x36C4)	p_gr_p3q2	dddd dddd dddd dddd	0 (0x0000)
R14022(R0x36C6)	p_gr_p3q3	dddd dddd dddd dddd	0 (0x0000)
R14024(R0x36C8)	p_gr_p3q4	dddd dddd dddd dddd	0 (0x0000)

Table 9: Register List and Default Values—Manufacturer-Specific (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Name	Data Format (Binary)	Default Value Dec (Hex)
R14026(R0x36CA)	p_rd_p3q0	dddd dddd dddd dddd	0 (0x0000)
R14028(R0x36CC)	p_rd_p3q1	dddd dddd dddd dddd	0 (0x0000)
R14030(R0x36CE)	p_rd_p3q2	dddd dddd dddd dddd	0 (0x0000)
R14032(R0x36D0)	p_rd_p3q3	dddd dddd dddd dddd	0 (0x0000)
R14034(R0x36D2)	p_rd_p3q4	dddd dddd dddd dddd	0 (0x0000)
R14036(R0x36D4)	p_bl_p3q0	dddd dddd dddd dddd	0 (0x0000)
R14038(R0x36D6)	p_bl_p3q1	dddd dddd dddd dddd	0 (0x0000)
R14040(R0x36D8)	p_bl_p3q2	dddd dddd dddd dddd	0 (0x0000)
R14042(R0x36DA)	p_bl_p3q3	dddd dddd dddd dddd	0 (0x0000)
R14044(R0x36DC)	p_bl_p3q4	dddd dddd dddd dddd	0 (0x0000)
R14046(R0x36DE)	p_gb_p3q0	dddd dddd dddd dddd	0 (0x0000)
R14048(R0x36E0)	p_gb_p3q1	dddd dddd dddd dddd	0 (0x0000)
R14050(R0x36E2)	p_gb_p3q2	dddd dddd dddd dddd	0 (0x0000)
R14052(R0x36E4)	p_gb_p3q3	dddd dddd dddd dddd	0 (0x0000)
R14054(R0x36E6)	p_gb_p3q4	dddd dddd dddd dddd	0 (0x0000)
R14080(R0x3700)	p_gr_p4q0	dddd dddd dddd dddd	0 (0x0000)
R14082(R0x3702)	p_gr_p4q1	dddd dddd dddd dddd	0 (0x0000)
R14084(R0x3704)	p_gr_p4q2	dddd dddd dddd dddd	0 (0x0000)
R14086(R0x3706)	p_gr_p4q3	dddd dddd dddd dddd	0 (0x0000)
R14088(R0x3708)	p_gr_p4q4	dddd dddd dddd dddd	0 (0x0000)
R14090(R0x370A)	p_rd_p4q0	dddd dddd dddd dddd	0 (0x0000)
R14092(R0x370C)	p_rd_p4q1	dddd dddd dddd dddd	0 (0x0000)
R14094(R0x370E)	p_rd_p4q2	dddd dddd dddd dddd	0 (0x0000)
R14096(R0x3710)	p_rd_p4q3	dddd dddd dddd dddd	0 (0x0000)
R14098(R0x3712)	p_rd_p4q4	dddd dddd dddd dddd	0 (0x0000)
R14100(R0x3714)	p_bl_p4q0	dddd dddd dddd dddd	0 (0x0000)
R14102(R0x3716)	p_bl_p4q1	dddd dddd dddd dddd	0 (0x0000)
R14104(R0x3718)	p_bl_p4q2	dddd dddd dddd dddd	0 (0x0000)
R14106(R0x371A)	p_bl_p4q3	dddd dddd dddd dddd	0 (0x0000)
R14108(R0x371C)	p_bl_p4q4	dddd dddd dddd dddd	0 (0x0000)
R14110(R0x371E)	p_gb_p4q0	dddd dddd dddd dddd	0 (0x0000)
R14112(R0x3720)	p_gb_p4q1	dddd dddd dddd dddd	0 (0x0000)
R14114(R0x3722)	p_gb_p4q2	dddd dddd dddd dddd	0 (0x0000)
R14116(R0x3724)	p_gb_p4q3	dddd dddd dddd dddd	0 (0x0000)
R14118(R0x3726)	p_gb_p4q4	dddd dddd dddd dddd	0 (0x0000)
R14208(R0x3780)	sc_enable	d000 0000 0000 0000	0 (0x0000)
R14210(R0x3782)	origin_c	0000 dddd dddd dddd	0 (0x0000)
R14212(R0x3784)	origin_r	0000 dddd dddd dddd	0 (0x0000)
R14336(R0x3800)	otpm_data_0	dddd dddd dddd dddd	0 (0x0000)

Table 9: Register List and Default Values—Manufacturer-Specific (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Name	Data Format (Binary)	Default Value Dec (Hex)
R14338(R0x3802)	otpm_data_1	dddd dddd dddd dddd	0 (0x0000)

Register Descriptions

Table 10 through Table 12 on page 47 show sensor register descriptions.

Table 10: Register Description—SMIA Configuration

1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
0 R0x0000	15:0	0x2B01	model_id (R/W)	N	N
	This register is an alias of R0x3000–1. Read-only. Can be made read/write by clearing R0x301A–B[3].				
2 R0x0002	15:0	0x0000	revision_number (R/W)	N	N
	Aptina Imaging assigned revision number. Read-only. Can be made read/write by clearing R0x301A–B[3].				
3 R0x0003	15:0	0x0006	manufacturer_id (RO)	N	N
	Manufacturer ID assigned to Aptina Imaging. Read-only. Can be made read/write by clearing R0x301A–B[3].				
4 R0x0004	15:0	0x000A	smia_version (RO)	N	N
	This register is an alias of R0x303A. Read-only.				
5 R0x0005	15:0	0x00FF	frame_count (RO)	Y	N
	This register is an alias of R0x303B. Read-only.				
6 R0x0006	15:0	0x0000	pixel_order (RO)	N	N
	This register is an alias of R0x3024. Read-only.				
8 R0x0008	15:0	0x00A8	data_pedestal (R/W)	N	Y
	This register is an alias of R0x301E–F. Read-only. Can be made read/write by clearing R0x301A–B[3].				
64 R0x0040	15:0	0x0001	frame_format_model_type (RO)	N	N
	Type 1. 2-byte Generic Frame Format Description. Read-only.				
65 R0x0041	15:0	0x0012	frame_format_model_subtype (RO)	N	N
	Number of descriptors: 1 X (column) descriptor and two Y (row) descriptors. Read-only.				
66 R0x0042	15:0	0x5DA0	frame_format_descriptor_0 (RO)	Y	N
	X descriptor: Bits[11:0] of this register reflect the current value of x_output_size[11:0]. Upper 4 bits is the pixel code; 5 = Visible Pixel Data. Read-only, dynamic.				
68 R0x0044	15:0	0x1002	frame_format_descriptor_1 (RO)	Y	N
	Y descriptor: In normal operation, returns 0x1002 to indicate that two rows of embedded data are present in the output image. If embedded data is disabled (by selecting the PN9 test pattern using R0x3070-1) this register will return 0x1000. Read-only.				
70 R0x0046	15:0	0x5A38	frame_format_descriptor_2 (RO)	Y	N
	Y descriptor: Bits[11:0] of this register reflect the current value of y_output_size[11:0]. Upper four bits is the pixel code; 5 = Visible Pixel Data. Read-only, dynamic.				
72 R0x0048	15:0	0x0000	frame_format_descriptor_3 (RO)	N	N
	Read-only.				
74 R0x004A	15:0	0x0000	frame_format_descriptor_4 (RO)	N	N
	Read-only.				
76 R0x004C	15:0	0x0000	frame_format_descriptor_5 (RO)	N	N
	Read-only.				
78 R0x004E	15:0	0x0000	frame_format_descriptor_6 (RO)	N	N
	Read-only.				
80 R0x0050	15:0	0x0000	frame_format_descriptor_7 (RO)	N	N
	Read-only.				
82 R0x0052	15:0	0x0000	frame_format_descriptor_8 (RO)	N	N
	Read-only.				

Table 10: Register Description—SMIA Configuration (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
84 R0x0054	15:0	0x0000	frame_format_descriptor_9 (RO)	N	N
	Read-only.				
86 R0x0056	15:0	0x0000	frame_format_descriptor_10 (RO)	N	N
	Read-only.				
88 R0x0058	15:0	0x0000	frame_format_descriptor_11 (RO)	N	N
	Read-only.				
90 R0x005A	15:0	0x0000	frame_format_descriptor_12 (RO)	N	N
	Read-only.				
92 R0x005C	15:0	0x0000	frame_format_descriptor_13 (RO)	N	N
	Read-only.				
94 R0x005E	15:0	0x0000	frame_format_descriptor_14 (RO)	N	N
	Read-only.				
128 R0x0080	15:0	0x0001	analogue_gain_capability (RO)	N	N
	Indicates the provision of separate (per-color) analog gain control. The sensor supports both global and separate (per-color) analog gain control. Read-only.				
132 R0x0084	15:0	0x0008	analogue_gain_code_min (RO)	N	N
	Minimum gain code. Read-only.				
134 R0x0086	15:0	0x00BF	analogue_gain_code_max (RO)	N	N
	Maximum gain code. Read-only.				
136 R0x0088	15:0	0x0001	analogue_gain_code_step (RO)	N	N
	Gain code step size. Read-only.				
138 R0x008A	15:0	0x0000	analogue_gain_type (RO)	N	N
	Indicates support for analog gain coding type 0 (baseline SMIA). Read-only.				
140 R0x008C	15:0	0x0001	analogue_gain_m0 (RO)	N	N
	Constants for the gain equation. Read-only.				
142 R0x008E	15:0	0x0000	analogue_gain_c0 (RO)	N	N
	Constants for the gain equation. Read-only.				
144 R0x0090	15:0	0x0000	analogue_gain_m1 (RO)	N	N
	Constants for the gain equation. Read-only.				
146 R0x0092	15:0	0x0008	analogue_gain_c1 (RO)	N	N
	Constants for the gain equation. Read-only.				
192 R0x00C0	15:0	0x0001	data_format_model_type (RO)	N	N
	Indicates the use of 2-byte data format. Read-only.				
193 R0x00C1	15:0	0x0005	data_format_model_subtype (RO)	N	N
	Indicates the provision of 3 data format descriptors. Read-only.				
194 R0x00C2	15:0	0x0A0A	data_format_descriptor_0 (RO)	N	N
	Indicates support for RAW10, uncompressed data format. Read-only.				
196 R0x00C4	15:0	0x0808	data_format_descriptor_1 (RO)	N	N
	Indicates support for RAW8 data format in which the two LSB of each 10-bit pixel data value are discarded. Read-only.				
198 R0x00C6	15:0	0x0A08	data_format_descriptor_2 (RO)	N	N
	Indicates support for RAW8 data format in which each 10-bit pixel data value is compressed to an 8-bit value. Read-only.				
200 R0x00C8	15:0	0x0C0C	data_format_descriptor_3 (RO)	N	N
	Read-only.				

Table 10: Register Description—SMIA Configuration (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
202 R0x00CA	15:0	0x0C08	data_format_descriptor_4 (RO)	N	N
	Read-only.				
204 R0x00CC	15:0	0x0000	data_format_descriptor_5 (RO)	N	N
	Read-only.				
206 R0x00CE	15:0	0x0000	data_format_descriptor_6 (RO)	N	N
	Read-only.				
256 R0x0100	15:0	0x0000	mode_select (R/W)	Y	N
	This register field is an alias of R0x301A[2].				
257 R0x0101	15:0	0x0000	image_orientation (R/W)		
	15:2	X	Reserved		
	1	0x0000	image_orientation_vertical_flip This register field is an alias of R0x3040–1[1].	Y	YM
	0	0x0000	image_orientation_horizontal_mirror This register field is an alias of R0x3040–1[0].	Y	YM
259 R0x0103	15:0	0x0000	software_reset (R/W)	N	Y
	This register field is an alias of R0x301A–B[0].				
260 R0x0104	15:0	0x0000	grouped_parameter_hold (R/W)	N	N
	This register field is an alias of R0x301A–B[15].				
261 R0x0105	15:0	0x0000	mask_corrupted_frames (R/W)	N	Y
	This register field is an alias of R0x301A–B[9].				
272 R0x0110	15:0	0x0000	ccp2_channel_identifier (R/W)	Y	N
	When the CCP2 serial pixel data interface is in use, this three-bit field supplies the DMA channel identifier that will be used within the CCP2 embedded synchronization codes. When the MIPI serial pixel data interface is in use, the low two bits of this three-bit field supply the Virtual Channel (VC) identifier in the Data Identifier (DI) byte which forms part of the short and long packet headers.				
273 R0x0111	15:0	0x0001	ccp2_signalling_mode (R/W)	Y	N
	0 = Use Data/Clock signaling on the CCP2 serial interface. 1 = Use Data/Strobe signaling on the CCP2 serial interface.				
274 R0x0112	15:0	0x0808	ccp_data_format (R/W)	Y	N
	[7:0] = The bit-width of the compressed pixel data [15:8] = The bit-width of the uncompressed pixel data The value in this register must match one of the valid data_format_descriptor registers (R0x00C2–R0x00C7).				
288 R0x0120	15:0	0x0000	gain_mode (R/W)	N	N
	This read/write bit has no function.				
512 R0x0200	15:0	0x056A	fine_integration_time (R/W)	Y	N
	Integration time programmed in units of PCK. This register is an alias of R0x3014–5.				
514 R0x0202	15:0	0x0010	coarse_integration_time (R/W)	Y	N
	Integration time programmed in units of line_length_pck. This register is an alias of R0x3012–3.				
516 R0x0204	15:0	0x0008	analogue_gain_code_global (R/W)	Y	N
	This register is an alias of R0x3028–9.				
518 R0x0206	15:0	0x0008	analogue_gain_code_greenR (R/W)	Y	N
	This register is an alias of R0x302A–B.				
520 R0x0208	15:0	0x0008	analogue_gain_code_red (R/W)	Y	N
	This register is an alias of R0x302C–D.				

Table 10: Register Description—SMIA Configuration (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
522 R0x020A	15:0	0x0008	analogue_gain_code_blue (R/W)	Y	N
	This register is an alias of R0x302E–F.				
524 R0x020C	15:0	0x0008	analogue_gain_code_greenB (R/W)	Y	N
	This register is an alias of R0x3030–1.				
526 R0x020E	15:0	0x0100	digital_gain_greenR (R/W)	Y	N
	This register is an alias of R0x3032–3.				
528 R0x0210	15:0	0x0100	digital_gain_red (R/W)	Y	N
	This register is an alias of R0x3034–5.				
530 R0x0212	15:0	0x0100	digital_gain_blue (R/W)	Y	N
	This register is an alias of R0x3036–7.				
532 R0x0214	15:0	0x0100	digital_gain_greenB (R/W)	Y	N
	This register is an alias of R0x3038–9.				
768 R0x0300	15:0	0x0004	vt_pix_clk_div (R/W)	N	Y
	Clock divisor applied to video timing system clock to generate video timing pixel clock.				
770 R0x0302	15:0	0x0001	vt_sys_clk_div (R/W)	N	N
	Clock divisor applied to PLL output clock to generate video timing system clock. Read-only.				
772 R0x0304	15:0	0x0002	pre_pll_clk_div (R/W)	N	Y
	Clock divisor applied to EXTCLK to generate PLL input clock.				
774 R0x0306	15:0	0x0040	pll_multiplier (R/W)	N	Y
	Clock multiplier applied to PLL input clock.				
776 R0x0308	15:0	0x0008	op_pix_clk_div (R/W)	N	Y
	Clock divisor applied to the output system clock to generate the output pixel clock.				
778 R0x030A	15:0	0x0001	op_sys_clk_div (R/W)	N	Y
	Clock divisor applied to PLL output clock to generate output system clock. Read-only.				
832 R0x0340	15:0	0x0AC7	frame_length_lines (R/W)	Y	YM
	This register is an alias of R0x300A–B.				
834 R0x0342	15:0	0x1C9C	line_length_pck (R/W)	Y	YM
	This register is an alias of R0x300C–D.				
836 R0x0344	15:0	0x0000	x_addr_start (R/W)	Y	N
	This register is an alias of R0x3004–5.				
838 R0x0346	15:0	0x0008	y_addr_start (R/W)	Y	YM
	This register is an alias of R0x3002–5.				
840 R0x0348	15:0	0x0D9F	x_addr_end (R/W)	Y	N
	This register is an alias of R0x3008–9.				
842 R0x034A	15:0	0x0A3F	y_addr_end (R/W)	Y	YM
	This register is an alias of R0x3006–7.				
844 R0x034C	15:0	0x0DA0	x_output_size (R/W)	Y	N
	Set X output size of displayed image. Bit[0] is read-only “0.” The default value of this register is set to be consistent with the default values of x_addr_end and x_addr_start.				
846 R0x034E	15:0	0x0A38	y_output_size (R/W)	Y	N
	Set Y output size of the displayed image. Bit[0] is read-only “0.” The default value of this registers set to be consistent with the default values of y_addr_end and y_addr_start. The output image will have two additional rows containing embedded data, in accordance with the frame format descriptors.				
896 R0x0380	15:0	0x0001	x_even_inc (RO)	N	N
	Read-only. The fixed value of 1 constrains subsampling operation to use adjacent pixels of a pixel quad.				

Table 10: Register Description—SMIA Configuration (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
898 R0x0382	15:0	0x0001	x_odd_inc (R/W)	Y	YM
	This register field is an alias of R0x3040–1[7:5].				
900 R0x0384	15:0	0x0001	y_even_inc (RO)	N	N
	Read-only. The fixed value of 1 constrains subsampling operation to use adjacent pixels of a pixel quad.				
902 R0x0386	15:0	0x0001	y_odd_inc (R/W)	Y	YM
	This register field is an alias of R0x3040–1[4:2].				
1024 R0x0400	15:0	0x0000	scaling_mode (R/W)	Y	N
	0 = Disable scaler 1 = Enable horizontal scaling 2 = Enable horizontal and vertical scaling 3 = Reserved				
1026 R0x0402	15:0	0x0000	spatial_sampling (R/W)	Y	N
	0 = Bayer sampling 1 = Co-sited sampling				
1028 R0x0404	15:0	0x0010	scale_m (R/W)	Y	N
	Scale factor M.				
1030 R0x0406	15:0	0x0010	scale_n (RO)	N	N
	Scale factor N. Read-only.				
1280 R0x0500	15:0	0x0001	compression_mode (RO)	N	Y
	0x0001 = 10-bit to 8-bit and 12-bit to 8-bit compression uses the DPCM/PCM Simple Predictor algorithm. Read-only. This register controls the algorithm that is to be used for compression. The sensor only supports a single algorithm and therefore this register is read-only. This register does not control whether data compression is enabled; that is controlled by the ccp_data_format register (R0x0012–3).				
1536 R0x0600	15:0	0x0000	test_pattern_mode (R/W)	N	Y
	This register is an alias of R0x3070–1.				
1538 R0x0602	15:0	0x0000	test_data_red (R/W)	N	Y
	This register is an alias of R0x3072–3.				
1540 R0x0604	15:0	0x0000	test_data_greenR (R/W)	N	Y
	This register is an alias of R0x3074–5.				
1542 R0x0606	15:0	0x0000	test_data_blue (R/W)	N	Y
	This register is an alias of R0x3076–7.				
1544 R0x0608	15:0	0x0000	test_data_greenB (R/W)	N	Y
	This register is an alias of R0x3078–8.				
1546 R0x060A	15:0	0x0000	horizontal_cursor_width (R/W)	N	N
	This register is an alias of R0x31EC–D.				
1548 R0x060C	15:0	0x0000	horizontal_cursor_position (R/W)	N	N
	This register is an alias of R0x31E8–9.				
1550 R0x060E	15:0	0x0000	vertical_cursor_width (R/W)	N	N
	This register is an alias of R0x31EE–F.				
1552 R0x0610	15:0	0x0000	vertical_cursor_position (R/W)	N	N
	This register is an alias of R0x31EA–B.				

Table 11: Register Description—SMIA Parameter Limits
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
4096 R0x1000	15:0	0x0001	integration_time_capability (RO)	N	N
	Indicates the provision of coarse and fine integration time control. Read-only. Can be made read/write by clearing R0x301A–B[3].				
4100 R0x1004	15:0	0x0000	coarse_integration_time_min (R/W)	N	N
	The minimum coarse integration time. Read-only. Can be made read/write by clearing R0x301A–B[3].				
4102 R0x1006	15:0	0x0001	coarse_integration_time_max_margin (R/W)	N	N
	The maximum coarse integration time is (frame_length_lines - coarse_integration_time_max_margin). Read-only. Can be made read/write by clearing R0x301A–B[3]. In the sensor, this limit can be broken. The result will be a graceful degradation of frame rate.				
4104 R0x1008	15:0	0x056A	fine_integration_time_min (R/W)	N	N
	The minimum fine integration time. Read-only. Can be made read/write by clearing R0x301A–B[3].				
4106 R0x100A	15:0	0x03A6	fine_integration_time_max_margin (R/W)	N	N
	The minimum fine integration time is (line_length_pck - fine_integration_time_max_margin). Read-only. Can be made read/write by clearing R0x301A–B[3].				
4224 R0x1080	15:0	0x0001	digital_gain_capability (RO)	N	N
	Indicates the provision of separate (per-color) digital gain control. Read-only.				
4228 R0x1084	15:0	0x0100	digital_gain_min (RO)	N	N
	UFIX16. Minimum value of digital gain is 1.0. Read-only.				
4230 R0x1086	15:0	0x0700	digital_gain_max (RO)	N	N
	UFIX16. Maximum value of digital gain is 7.0. Read-only.				
4232 R0x1088	15:0	0x0100	digital_gain_step_size (RO)	N	N
	UFIX16. Step size for digital gain is 1.0. Read-only.				
4352 R0x1100	15:0	0x4000	min_ext_clk_freq_mhz_1 (RO)	N	N
	FLP32. Minimum external clock frequency into PLL is 2.0 MHz. Read-only.				
4354 R0x1102	15:0	0x0000	min_ext_clk_freq_mhz_2 (RO)	N	N
	FLP32. Minimum external clock frequency into PLL is 2.0 MHz. Read-only.				
4356 R0x1104	15:0	0x4280	max_ext_clk_freq_mhz_1 (RO)	N	N
	FLP32. Maximum external clock frequency into PLL is 64.0 MHz. Read-only.				
4358 R0x1106	15:0	0x0000	max_ext_clk_freq_mhz_2 (RO)	N	N
	FLP32. Maximum external clock frequency into PLL is 64.0 MHz. Read-only.				
4360 R0x1108	15:0	0x0001	min_pre_pll_clk_div (RO)	N	N
	Minimum clock divisor applied to PLL input clock. Read-only.				
4362 R0x110A	15:0	0x0040	max_pre_pll_clk_div (RO)	N	N
	Maximum clock divisor applied to PLL input clock. Read-only.				
4364 R0x110C	15:0	0x4080	min_pll_ip_freq_mhz_1 (RO)	N	N
	FLP32. Minimum clock frequency into the PFD of the PLL is 4.0 MHz. Read-only.				
4366 R0x110E	15:0	0x0000	min_pll_ip_freq_mhz_2 (RO)	N	N
	FLP32. Minimum clock frequency into the PFD of the PLL is 4.0 MHz. Read-only.				
4368 R0x1110	15:0	0x41C0	max_pll_ip_freq_mhz_1 (RO)	N	N
	FLP32. Maximum clock frequency into the PFD of the PLL is 24 MHz. Read-only.				
4370 R0x1112	15:0	0x0000	max_pll_ip_freq_mhz_2 (RO)	N	N
	FLP32. Maximum clock frequency into the PFD of the PLL is 24 MHz. Read-only.				

Table 11: Register Description—SMIA Parameter Limits (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
4372 R0x1114	15:0	0x0020	min_pll_multiplier (RO)	N	N
	Minimum multiplier applied by PLL. Read-only.				
4374 R0x1116	15:0	0x0180	max_pll_multiplier (RO)	N	N
	Maximum multiplier applied by PLL. Read-only.				
4376 R0x1118	15:0	0x43C0	min_pll_op_freq_mhz_1 (RO)	N	N
	FLP32. Minimum output frequency supported by the PLL is 384.0 MHz. Read-only.				
4378 R0x111A	15:0	0x0000	min_pll_op_freq_mhz_2 (RO)	N	N
	FLP32. Minimum output frequency supported by the PLL is 384.0 MHz. Read-only.				
4380 R0x111C	15:0	0x4440	max_pll_op_freq_mhz_1 (RO)	N	N
	FLP32. Maximum output frequency supported by the PLL is 768.0 MHz. Read-only.				
4382 R0x111E	15:0	0x0000	max_pll_op_freq_mhz_2 (RO)	N	N
	FLP32. Maximum output frequency supported by the PLL is 768.0 MHz. Read-only.				
4384 R0x1120	15:0	0x0001	min_vt_sys_clk_div (RO)	N	N
	The video timing sys_clk has a fixed divisor. Read-only.				
4386 R0x1122	15:0	0x0001	max_vt_sys_clk_div (RO)	N	N
	The video timing sys_clk has a fixed divisor. Read-only.				
4388 R0x1124	15:0	0x41C0	min_vt_sys_clk_freq_mhz_1 (RO)	N	N
	FLP32. Minimum frequency for the video timing sys_clk is 24.0 MHz.				
4390 R0x1126	15:0	0x0000	min_vt_sys_clk_freq_mhz_2 (RO)	N	N
	FLP32. Minimum frequency for the video timing sys_clk is 24.0 MHz.				
4392 R0x1128	15:0	0x4440	max_vt_sys_clk_freq_mhz_1 (RO)	N	N
	Maximum frequency for the video timing sys_clk is 768.0 MHz. Read-only.				
4394 R0x112A	15:0	0x0000	max_vt_sys_clk_freq_mhz_2 (RO)	N	N
	Maximum frequency for the video timing sys_clk is 768.0 MHz. Read-only.				
4396 R0x112C	15:0	0x4019	min_vt_pix_clk_freq_mhz_1 (RO)	N	N
	FLP32. Minimum frequency for video timing pix_clk is 2.4 MHz. Read-only.				
4398 R0x112E	15:0	0x999A	min_vt_pix_clk_freq_mhz_2 (RO)	N	N
	FLP32. Minimum frequency for video timing pix_clk is 2.4 MHz. Read-only.				
4400 R0x1130	15:0	0x4340	max_vt_pix_clk_freq_mhz_1 (RO)	N	N
	FLP32. Maximum frequency for video timing pix_clk is 96.0 MHz. Read-only.				
4402 R0x1132	15:0	0x0000	max_vt_pix_clk_freq_mhz_2 (RO)	N	N
	FLP32. Maximum frequency for video timing pix_clk is 96.0 MHz. Read-only.				
4404 R0x1134	15:0	0x0004	min_vt_pix_clk_div (RO)	N	N
	Minimum divisor for the video timing pix_clk. Read-only.				
4406 R0x1136	15:0	0x0006	max_vt_pix_clk_div (RO)	N	N
	Maximum divisor for the video timing pix_clk. Read-only.				
4416 R0x1140	15:0	0x0091	min_frame_length_lines (R/W)	N	N
	Minimum frame length. Read-only. Can be made read/write by clearing R0x301A–B[3].				
4418 R0x1142	15:0	0xFFFF	max_frame_length_lines (R/W)	N	N
	Maximum frame length. The maximum frame length is only constrained by the size of the read/write field in the frame_length_lines register (16 bits). Read-only. Can be made read/write by clearing R0x301A–B[3].				
4420 R0x1144	15:0	0x0910	min_line_length_pck (R/W)	N	N
	Minimum line length. Read-only. Can be made read/write by clearing R0x301A–B[3].				

Table 11: Register Description—SMIA Parameter Limits (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
4422 R0x1146	15:0	0xFFFE	max_line_length_pck (R/W)	N	N
	Maximum line length. The maximum line length is only constrained by the size of the read/write field in the line_length_pck register (16 bits). Read-only. Can be made read/write by clearing R0x301A–B[3].				
4424 R0x1148	15:0	0x06E4	min_line_blanking_pck (R/W)	N	N
	Minimum line blanking time. Read-only. Can be made read/write by clearing R0x301A–B[3].				
4426 R0x114A	15:0	0x008F	min_frame_blanking_lines (R/W)	N	N
	Minimum frame blanking time. Read-only. Can be made read/write by clearing R0x301A–B[3].				
4448 R0x1160	15:0	0x0001	min_op_sys_clk_div (RO)	N	N
	Minimum divisor for the output sys_clk. Read-only.				
4450 R0x1162	15:0	0x0001	max_op_sys_clk_div (RO)	N	N
	Maximum divisor for the output sys_clk. Read-only.				
4452 R0x1164	15:0	0x4199	min_op_sys_clk_freq_mhz_1 (RO)	N	N
	FLP32. Minimum frequency for output sys_clk is 19.2 MHz. Read-only.				
4454 R0x1166	15:0	0x999A	min_op_sys_clk_freq_mhz_2 (RO)	N	N
	FLP32. Minimum frequency for output sys_clk is 19.2 MHz. Read-only.				
4456 R0x1168	15:0	0x4440	max_op_sys_clk_freq_mhz_1 (RO)	N	N
	FLP32. Maximum frequency for output sys_clk is 768.0 MHz. Read-only.				
4458 R0x116A	15:0	0x0000	max_op_sys_clk_freq_mhz_2 (RO)	N	N
	FLP32. Maximum frequency for output sys_clk is 768.0 MHz. Read-only.				
4460 R0x116C	15:0	0x0008	min_op_pix_clk_div (RO)	N	N
	Minimum divisor for output pix_clk. Read-only.				
4462 R0x116E	15:0	0x000C	max_op_pix_clk_div (RO)	N	N
	Maximum divisor for output pix_clk. Read-only.				
4464 R0x1170	15:0	0x4019	min_op_pix_clk_freq_mhz_1 (RO)	N	N
	FLP32. Minimum frequency for output pix_clk is 2.4 MHz. Read-only.				
4466 R0x1172	15:0	0x999A	min_op_pix_clk_freq_mhz_2 (RO)	N	N
	FLP32. Minimum frequency for output pix_clk is 2.4 MHz. Read-only.				
4468 R0x1174	15:0	0x4340	max_op_pix_clk_freq_mhz_1 (RO)	N	N
	FLP32. Maximum frequency for output pix_clk is 96.0 MHz. Read-only.				
4470 R0x1176	15:0	0x0000	max_op_pix_clk_freq_mhz_2 (RO)	N	N
	FLP32. Maximum frequency for output pix_clk is 96.0 MHz. Read-only.				
4480 R0x1180	15:0	0x0000	x_addr_min (RO)	N	N
	Minimum value for x_addr_start, x_addr_end. Read-only.				
4482 R0x1182	15:0	0x0000	y_addr_min (RO)	N	N
	Minimum value for y_addr_start, y_addr_end. Read-only.				
4484 R0x1184	15:0	0x0DAF	x_addr_max (RO)	N	N
	Maximum value for x_addr_start, x_addr_end. Read-only.				
4486 R0x1186	15:0	0x0A47	y_addr_max (RO)	N	N
	Maximum value for y_addr_start, y_addr_end. Read-only.				
4544 R0x11C0	15:0	0x0001	min_even_inc (RO)	N	N
	Minimum value for increment of even X/Y addresses when subsampling is enabled. Read-only.				
4546 R0x11C2	15:0	0x0001	max_even_inc (RO)	N	N
	Maximum value for increment of even X/Y addresses when subsampling is enabled. Read-only.				
4548 R0x11C4	15:0	0x0001	min_odd_inc (RO)	N	N
	Minimum value for increment of odd X/Y addresses when subsampling is enabled. Read-only.				

Table 11: Register Description—SMIA Parameter Limits (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
4550 R0x11C6	15:0	0x0003	max_odd_inc (RO)	N	N
	Maximum value for increment of odd X/Y addresses when subsampling is enabled. Read-only. This set of 4 registers declares the capability for the subsampling mode that was called "skip2" on earlier Aptina Imaging sensors. Note that "skip4" is also supported, since values of 1, 3, and 7 are supported.				
4608 R0x1200	15:0	0x0002	scaling_capability (RO)	N	N
	Indicates the provision of a full (horizontal and vertical) scaler. Read-only.				
4612 R0x1204	15:0	0x0010	scaler_m_min (RO)	N	N
	Indicates the minimum M value for the scaler. Read-only.				
4614 R0x1206	15:0	0x0080	scaler_m_max (RO)	N	N
	Indicates the maximum M value for the scaler. Read-only.				
4616 R0x1208	15:0	0x0010	scaler_n_min (RO)	N	N
	Indicates the minimum N value for the scaler. Read-only.				
4618 R0x120A	15:0	0x0010	scaler_n_max (RO)	N	N
	Indicates the maximum N value for the scaler. Read-only.				
4864 R0x1300	15:0	0x0001	compression_capability (RO)	N	N
	Indicates the capability for performing 10-bit to 8-bit pixel data compression. Read-only.				
5120 R0x1400	15:0	0x0242	matrix_element_redinred (R/W)	N	N
	Read-only. Can be made read/write by clearing R0x301A–B[3].				
5122 R0x1402	15:0	0xFF00	matrix_element_greeninred (R/W)	N	N
	Color-correction matrix. Read-only. Can be made read/write by clearing R0x301A–B[3].				
5124 R0x1404	15:0	0xFFBE	matrix_element_blueinred (R/W)	N	N
	Color-correction matrix. Read-only. Can be made read/write by clearing R0x301A–B[3].				
5126 R0x1406	15:0	0xFFB4	matrix_element_redingreen (R/W)	N	N
	Color-correction matrix. Read-only. Can be made read/write by clearing R0x301A–B[3].				
5128 R0x1408	15:0	0x0200	matrix_element_greeningreen (R/W)	N	N
	Color-correction matrix. Read-only. Can be made read/write by clearing R0x301A–B[3].				
5130 R0x140A	15:0	0xFF4D	matrix_element_blueingreen (R/W)	N	N
	Color-correction matrix. Read-only. Can be made read/write by clearing R0x301A–B[3].				
5132 R0x140C	15:0	0xFFF1	matrix_element_redinblue (R/W)	N	N
	Color-correction matrix. Read-only. Can be made read/write by clearing R0x301A–B[3].				
5134 R0x140E	15:0	0xFF34	matrix_element_greeninblue (R/W)	N	N
	Color-correction matrix. Read-only. Can be made read/write by clearing R0x301A–B[3].				
5136 R0x1410	15:0	0x01DC	matrix_element_blueinblue (R/W)	N	N
	Color-correction matrix. Read-only. Can be made read/write by clearing R0x301A–B[3].				

Table 12: Register Description—Manufacturer-Specific

1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
12288 R0x3000	15:0	0x2B01	model_id_ (R/W)	N	N
	Model ID. Read-only. Can be made read/write by clearing R0x301A–B[3].				
12290 R0x3002	15:0	0x0008	y_addr_start_ (R/W)	Y	YM
	The first row of visible pixels to be read out (not counting any dark rows that may be read). To move the image window, set this register to the starting Y value.				
12292 R0x3004	15:0	0x0000	x_addr_start_ (R/W)	Y	N
	The first column of visible pixels to be read out (not counting any dark columns that may be read). To move the image window, set this register to the starting X value.				
12294 R0x3006	15:0	0x0A3F	y_addr_end_ (R/W)	Y	YM
	The last row of visible pixels to be read out.				
12296 R0x3008	15:0	0x0D9F	x_addr_end_ (R/W)	Y	N
	The last column of visible pixels to be read out.				
12298 R0x300A	15:0	0x0AC7	frame_length_lines_ (R/W)	Y	YM
	The number of complete lines (rows) in the output frame. This includes visible lines and vertical blanking lines.				
12300 R0x300C	15:0	0x1C9C	line_length_pck_ (R/W)	Y	YM
	The number of pixel clock periods in one line (row) time. This includes visible pixels and horizontal blanking time.				
12304 R0x3010	15:0	0x0100	fine_correction (R/W)	N	Y
	Fine integration time correction factor. This is an offset that is applied to the programmed value of fine_integration_time such that the actual integration time matches the integration time equation. This register should not be modified under normal operation, but must be modified when binning is enabled or the internal pixel clock divider (pc_speed[2:0]) is used.				
12306 R0x3012	15:0	0x0010	coarse_integration_time_ (R/W)	Y	N
	Integration time specified in multiples of line_length_pck_.				
12308 R0x3014	15:0	0x056A	fine_integration_time_ (R/W)	Y	N
	Integration time specified as a number of pixel clocks.				
12310 R0x3016	15:0	0x0111	row_speed (R/W)		
	15:11	X	Reserved		
	10:8	0x0001	row_speed_opclk_speed Slows down the output pixel clock frequency relative to the system clock frequency. A programmed value of N gives an output pixel clock period of N system clocks. Only values 1, 2, and 4 are supported. A value of "0" is illegal: it causes the clock to stop.	N	N
	7	X	Reserved		
	6:4	0x0001	row_speed_opclk_delay Number of half-system-clock-cycle increments to delay the rising edge of PIXCLK relative to transitions on FV, LV, and DOUT.	N	N
	3	X	Reserved		
12310 R0x3016	2:0	0x0001	row_speed_pixclk_speed Slows down the internal pixel clock frequency relative to the system clock frequency. A programmed value of N gives a pixel clock period of N system clocks. Only values 1, 2, and 4 are supported. A value of "0" is illegal: it causes the clock to stop.	Y	YM

Table 12: Register Description—Manufacturer-Specific (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
12312 R0x3018	15:0	0x0000	extra_delay (R/W)	Y	N
	Extra blanking inserted between frames. A programmed value of N increases the vertical blanking time by N pixel clock periods. Can be used to get a more exact frame rate. May affect the integration times of parts of the image when the integration time is less than 1 frame.				
12314 R0x301A	15:0	0x0058	reset_register (R/W)		
	15	0x0000	reset_register_grouped_parameter_hold 0 = Update of many of the registers is synchronized to frame start. 1 = Inhibit register updates; register changes will remain pending until this bit is returned to "0." When this bit is returned to "0," all pending register updates will be made on the next frame start.	N	N
	14	0x0000	reset_register_gain_insert When set, the gain values will always take affect the following frame independent of the integration time. When not set, gain will not update if an integration time change is in progress.	N	Y
	13	X	Reserved		
	12	0x0000	reset_register_smia_serializer_dis This bit disables the SMIA high speed serializer and differential output buffers.	N	N
	11	X	Reserved		
	10	0x0000	reset_register_restart_bad 1 = a restart is forced any time a bad frame is detected. This can shorten the delay when waiting for a good frame, since the delay for masking out a bad frame will be the integration time rather than the full-frame time.	N	N
	9	0x0000	reset_register_mask_bad 0 = The sensor will produce bad (corrupted) frames as a result of some register changes. 1 = Bad (corrupted) frames are masked within the sensor by extending the vertical blanking time for the duration of the bad frame.	N	N
	8	0x0000	reset_register_gpi_en 0 = the primary input buffers associated with the GPI0, GPI1, GPI2, and GPI3 inputs are powered down and the GPI cannot be used. 1 = the input buffers are enabled and can be read through R0x3026–7.	N	N

Table 12: Register Description—Manufacturer-Specific (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
12314 R0x301A	7	0x0000	reset_register_parallel_en 0 = The parallel data interface (DOUT[9:0], LV, FV, and PIXCLK) is disabled and the outputs are placed in a High-Z. 1 = The parallel data interface is enabled. The output signals can be switched between a driven and a high-impedance state using output enable control.	N	N
	6	0x0001	reset_register_drive_pins 0 = The parallel data interface (DOUT[9:0], LV, FV, and PIXCLK) may enter a High-Z (depending upon the configuration of R0x3026). 1 = The parallel data interface is driven. This bit is “don't care” unless bit[7] = 1.	N	N
	5	0x0000	Reserved		
	4	0x0001	reset_register_stdby_eof 0 = Transition to standby is synchronized to the end of a sensor row readout (held off until LV has fallen). 1 = Transition to standby is synchronized to the end of a frame.	N	Y
	3	0x0001	reset_register_lock_reg Many SMIA registers that are specified as read-only are actually implemented as read/write registers. Clearing this bit allows such registers to be written.	N	N
	2	0x0000	reset_register_stream Setting this bit places the sensor in streaming mode. Clearing this bit places the sensor in a low power mode. The result of clearing this bit depends upon the operating mode of the sensor. Entry and exit from streaming mode can also be controlled from the signal interface.	Y	N
	1	0x0000	reset_register_restart This bit always reads as “0.” Setting this bit causes the sensor to truncate the current frame at the end of the current row and start resetting (integrating) the first row. The delay before the first valid frame is read out is equal to the integration time.	N	Y
	0	0x0000	reset_register_reset This bit always reads as “0.” Setting this bit initiates a reset sequence; the frame being generated will be truncated.	N	Y
12316 R0x301C	15:0	0x0000	mode_select_ (R/W)	Y	N
	This bit is an alias of R0x301A-B[2].				
12317 R0x301D	15:0	0x0000	image_orientation_ (R/W)		
	15:2	X	Reserved		
	1	0x0000	image_orientation_vert_flip This bit is an alias of R0x3040[1].	Y	YM
	0	0x0000	image_orientation_horiz_mirror This bit is an alias of R0x3040[0].	Y	YM
12318 R0x301E	15:0	0x00A8	data_pedestal_ (R/W)	N	Y
	Constant offset that is added to the ADC output for all visible pixels in order to set the black level to a value greater than 0. Read-only. Can be made read/write by clearing R0x301A–B[3].				
12321 R0x3021	15:0	0x0000	software_reset_ (R/W)	N	Y
	This bit is an alias of R0x301A–B[0].				
12322 R0x3022	15:0	0x0000	grouped_parameter_hold_ (R/W)	N	N
	This bit is an alias of R0x301A–B[15].				

Table 12: Register Description—Manufacturer-Specific (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
12323 R0x3023	15:0	0x0000	mask_corrupted_frames_ (R/W)	N	N
	This bit is an alias of R0x301A–B[9].				
12324 R0x3024	15:0	0x0000	pixel_order_ (RO)	N	N
	00 = First row is GreenR/Red, first pixel is GreenR 01 = First row is GreenR/Red, first pixel is Red 02 = First row is Blue/GreenB, first pixel is Blue 03 = First row is Blue/GreenB, first pixel is GreenB The value in this register changes as a function of R0x3040[1:0].				
12326 R0x3026	15:0	0xFFFF	gpi_status (R/W)		
	15:13	0x0007	gpi_status_standby_pin_select Associate the standby function with an active-high input pin 0 = associate with GPI0 1 = associate with GPI1 2 = associate with GPI2 3 = associate with GPI3 4–6 = Reserved 7 = standby function cannot be controlled by any pin Must be set to 7 if reset[8] = 0.	N	N
	12:10	0x0007	gpi_status_oe_n_pin_select Associate the output-enable function with an active-low input pin 0 = associate with GPI0 1 = associate with GPI1 2 = associate with GPI2 3 = associate with GPI3 4–6 = Reserved 7 = output-enable function is not controlled by any pin Must be set to 7 if reset[8] = 0.	N	N

Table 12: Register Description—Manufacturer-Specific (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
12326 R0x3026	9:7	0x0007	gpi_status_trigger_pin_select Associate the trigger function with an active-high input pin 0 = associate with GPI0 1 = associate with GPI1 2 = associate with GPI2 3 = associate with GPI3 4–6 = Reserved 7 = trigger function is not controlled by any pin Must be set to 7 if R0x301A–B[8] = 0.	N	N
	6:4	0x0007	gpi_status_saddr_pin_select Associate the SADDR function with an active HIGH input pin 0 = associate with GPI0 1 = associate with GPI1 2 = associate with GPI2 3 = associate with GPI3 4–6 = Reserved 7 = SADDR function is not controlled by any pin Must be set to 7 if R0x301A–B[8] = 0.	N	N
	3	RO	gpi_status_gpi3 Read-only. Return the current state of the GPI3 input pin. Invalid if R0x301A–B[8] = 0.	N	N
	2	RO	gpi_status_gpi2 Read-only. Return the current state of the GPI2 input pin. Invalid if R0x301A–B[8]=0.	N	N
	1	RO	gpi_status_gpi1 Read-only. Return the current state of the GPI1 input pin. Invalid if R0x301A–B[8]=0.	N	N
	0	RO	gpi_status_gpi0 Read-only. Return the current state of the GPI0 input pin. Invalid if R0x301A–B[8]=0.	N	N
12328 R0x3028	15:0	0x0008	analogue_gain_code_global_ (R/W)	Y	N
	Writing a gain code to this register is equivalent to writing that code to each of the four color-specific gain code registers. Reading from this register returns the value most recently written to the analogue_gain_code_greenR register.				
12330 R0x302A	15:0	0x0008	analogue_gain_code_greenR_ (R/W)	Y	N
	The gain code written to this register sets the gain for green pixels on red/green rows of the pixel array.				
12332 R0x302C	15:0	0x0008	analogue_gain_code_red_ (R/W)	Y	N
	The gain code written to this register sets the gain for red pixels.				
12334 R0x302E	15:0	0x0008	analogue_gain_code_blue_ (R/W)	Y	N
	The gain code written to this register sets the gain for blue pixels.				
12336 R0x3030	15:0	0x0008	analogue_gain_code_greenB_ (R/W)	Y	N
	The gain code written to this register sets the gain for green pixels on blue/green rows of the pixel array.				
12338 R0x3032	15:0	0x0100	digital_gain_greenR_ (R/W)	Y	N
	Digital gain applied to green pixels on red/green rows of the pixel array. The value is an unsigned 8.8 fixed-point format. Bits [10:8] are significant and are an alias of R0x3056[11:9].				
12340 R0x3034	15:0	0x0100	digital_gain_red_ (R/W)	Y	N
	Digital gain applied to red pixels of the pixel array. The value is an unsigned 8.8 fixed-point format. Bits [10:8] are significant and are an alias of R0x305A[11:9].				

Table 12: Register Description—Manufacturer-Specific (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
12342 R0x3036	15:0	0x0100	digital_gain_blue_ (R/W)	Y	N
	Digital gain applied to blue pixels of the pixel array. The value is an unsigned 8.8 fixed-point format. Bits [10:8] are significant and are an alias of R0x3058[11:9].				
12344 R0x3038	15:0	0x0100	digital_gain_greenB_ (R/W)	Y	N
	Digital gain applied to green pixels on blue/green rows of the pixel array. The value is an unsigned 8.8 fixed-point format. Bits [10:8] are significant and are an alias of R0x305C[11:9].				
12346 R0x303A	15:0	0x000A	smia_version_ (RO)	N	N
	Return the value 10 to indicate an implementation of revision 1.0 of the SMIA specification. Read-only.				
12347 R0x303B	15:0	0x00FF	frame_count_ (RO)	Y	N
	In the soft standby state this counter is set to 0xFF. In streaming state this counter increments by 1 (modulo 255) at the start of each frame. The counter is incremented for both good frames and bad (corrupted) frames; its behavior is not affected by the state of R0x301A–B[9] (mask_corrupted_frames). After entry to the streaming state, the first frame will show a frame count of 0x01 in its embedded data. Read-only.				
12348 R0x303C	15:0	0x0000	frame_status (RO)		
	15:2	X	Reserved		
	1	RO	frame_status_standby This bit tells you whether the sensor is in standby state. Can be polled after standby is entered to see when the real low-power state is entered; which can happen at the end of row or frame depending on bit 0x301A[4].	N	N
	0	RO	frame_status_framesync Set on register write and reset on frame synchronization. Acts as debug flag to verify that register writes completed before last frame synchronization.	N	N
12352 R0x3040	15:0	0x0041	read_mode (R/W)		
	15	0x0000	read_mode_vert_flip 0 = Normal readout 1 = Readout is flipped (mirrored) vertically so that the row specified by y_addr_end_ is read out of the sensor first. Setting this bit will change the Bayer pixel order (see R0x3024).	Y	YM
	14	0x0000	read_mode_horiz_mirror 0 = Normal readout 1 = Readout is mirrored horizontally so that the column specified by x_addr_end_ is read out of the sensor first. Setting this bit will change the Bayer pixel order (see R0x3024).	Y	YM
	13	0x0000	READ_MODE_Y_SUM_EN Enable Y summing mode for binning.	Y	YM
	12	0x0000	Reserved		

Table 12: Register Description—Manufacturer-Specific (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
12352 R0x3040	11	0x0000	read_mode_x_bin_en Enable analogue binning in X (column) direction. When set, x_odd_inc must be set to 3 and y_odd_inc must be set to 1, along with other register changes.	Y	N
	10	0x0000	read_mode_xy_bin_en Enable analogue binning in X and Y (column and row) directions. When set, x_odd_inc and y_odd_inc must be set to 3, along with other register changes.	Y	N
	9	0x0000	read_mode_low_power Enables low power mode. This will automatically half the pixel clock speed. Cannot be used when pc_speed[2:0] = 4.	Y	YM
	8:6	0x0001	read_mode_x_odd_inc Increment applied to odd addresses in X (column) direction. 1= Normal readout 3 = Read out alternate pixel pairs to halve the amount of horizontal data in a frame. 7 = Read out 1 of 4 pixel pairs to reduce the amount of horizontal data in a frame by 4.	Y	YM
	5:0	0x0001	read_mode_y_odd_inc Increment applied to odd addresses in Y (row) direction. 1= Normal readout 3 = Read out alternate pixel pairs to halve the amount of vertical data in a frame. 7 = Read out 1 of 4 pixel pairs to reduce the amount of vertical data in a frame by 4. 15= Read out 1 out of 8 pixel pairs to reduce the amount of vertical data in a frame by 8. 31 = Read out 1 out of 16 pixel pairs to reduce the amount of vertical data in a frame by 16. 63 = Read out 1 out of 32 pixel pairs to reduce the amount of vertical data in a frame by 32.	Y	YM
12358 R0x3046	15:0	0x0600	flash (R/W)		
	15	RO	flash_strobe Reflects the current state of the FLASH output signal. Read-only.	N	N
	14	RO	flash_triggered Indicates that the FLASH output signal was asserted for the current frame. Read-only.	N	N
	13	0x0000	flash_xenon_flash Enable xenon flash. When set, the FLASH output signal will assert for the programmed period (bits [7:0]) during vertical blanking. This is achieved by keeping the integration time equal to one frame, and the pulse width less than the vertical blanking time.	Y	N

Table 12: Register Description—Manufacturer-Specific (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
12358 R0x3046	12:11	0x0000	flash_frame_delay Flash pulse delay measured in frames.	N	N
	10	0x0001	flash_end_of_reset 0 = In xenon mode, the flash is triggered after a frame readout. 1 = In xenon mode, the flash is triggered after resetting a frame.	N	N
	9	0x0001	flash_every_frame 0 = Flash should be enabled for one frame only. 1 = Flash should be enabled every frame.	N	N
	8	0x0000	flash_led_flash Enable LED flash. When set, the FLASH output signal will assert prior to the start of the resetting of a frame and will remain asserted until the end of the frame readout.	Y	Y
	7	0x0000	flash_invert_flash Invert flash output signal. When set, the FLASH output signal will be active LOW.	N	N
	6:0	X	Reserved		
12360 R0x3048	15:0	0x0008	flash_count (R/W)	N	N
	Length of flash pulse when Xenon flash is enabled. The value specifies the length in units of 256 x PIXCLK cycle increments (by default, PIXCLK = system_clock). When the xenon count is set to its maximum value (0x3FF), the flash pulse will automatically be truncated prior to the readout of the first row, giving the longest pulse possible.				
12374 R0x3056	15:0	0x1020	greenR_gain (R/W)		
	15	X	Reserved		
	14:12	0x0001	greenR_gain_digital_gain GreenR Digital Gain. Legal values 1–7.	Y	N
	11:10	X	Reserved		
	9:7	0x0000	greenR_gain_analog_gain Analog gain = {X * (bit [7] + 1) * initial gain}, where: X = 1 if bit[9:8] = 00 X = 2 if bit[9:8] = 01 X = 3 if bit[9:8] = 10 X = 3 if bit[9:8] = 11	Y	N
	6:0	0x0020	greenR_gain_initial_gain Initial gain = bits [6:0] * 1/32.	Y	N
12376 R0x3058	15:0	0x1020	blue_gain (R/W)		
	15	X	Reserved		
	14:12	0x0001	blue_gain_digital_gain Blue Digital Gain. Legal values 1–7.	Y	N
	11:10	X	Reserved		
	9:7	0x0000	blue_gain_analog_gain Analog gain = {X * (bit [7] + 1) * initial gain}, where: X = 1 if bit[9:8] = 00 X = 2 if bit[9:8] = 01 X = 3 if bit[9:8] = 10 X = 3 if bit[9:8] = 11	Y	N
	6:0	0x0020	blue_gain_initial_gain Initial gain = bits [6:0] * 1/32.	Y	N

Table 12: Register Description—Manufacturer-Specific (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
12378 R0x305A	15:0	0x1020	red_gain (R/W)		
	15	X	Reserved		
	14:12	0x0001	red_gain_digital_gain Red Digital Gain. Legal values 1–7.	Y	N
	11:10	X	Reserved		
	9:7	0x0000	red_gain_analog_gain Analog gain = {X * (bit [7] + 1) * initial gain}, where: X = 1 if bit[9:8] = 00 X = 2 if bit[9:8] = 01 X = 3 if bit[9:8] = 10 X = 3 if bit[9:8] = 11	Y	N
	6:0	0x0020	red_gain_initial_gain Initial gain = bits [6:0] * 1/32.	Y	N
12380 R0x305C	15:0	0x1020	greenB_gain (R/W)		
	15	X	Reserved		
	14:12	0x0001	greenB_gain_digital_gain GreenB Digital Gain. Legal values 1–7.	Y	N
	11:10	X	Reserved		
	9:7	0x0000	greenB_gain_analog_gain Analog gain = {X * (bit [7] + 1) * initial gain}, where: X = 1 if bit[9:8] = 00 X = 2 if bit[9:8] = 01 X = 3 if bit[9:8] = 10 X = 3 if bit[9:8] = 11	Y	N
	6:0	0x0020	greenB_gain_initial_gain Initial gain = bits [6:0] * 1/32.	Y	N
12382 R0x305E	15:0	0x1020	global_gain (R/W)	Y	N
	Writing a gain to this register is equivalent to writing that code to each of the four color-specific gain registers. Reading from this register returns the value most recently written to the greenR_gain register.				
12394 R0x306A	15:0	0x0000	datapath_status (RO)		
	15:6	X	Reserved		
	5	RO	Reserved		

Table 12: Register Description—Manufacturer-Specific (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
12394 R0x306A	4	0x0000	datapath_status_line_length_mismatch A fatal error occurred because the line length of the pixel data that the MIPI serializer expected to transmit did not match the line length set by X_OUTPUT_SIZE. The most likely reason for this error is that the clocking is configured incorrectly or that some register values that should remain static were changed whilst the sensor was in its streaming system state.	N	N
	3	0x0000	datapath_status_frameover A fatal error occurred because a new frame started before the current frame had completed. The usual reason for this is that the Y_OUTPUT_SIZE has been set too large so that the sensor output is still padding the frame with rows of undefined pixel data when the next frame starts. An alternative reason is that clock ratio between the VT clock domain and the OP clock domain does not allow sufficient time for the OP domain to complete a frame before the next frame starts. The first step in avoiding this error is to set Y_OUTPUT_SIZE so that it matches the number of rows generated from the pixel array (Y_ADDR_END - Y_ADDR_START + 1, taking into account any sub-sampling/binning and any scaling). If the error remains, the next step is to increase FRAME_LENGTH_LINES.	N	N
12394 R0x306A	2	0x0000	datapath_status_lineover A fatal error occurred because the sensor output was unable to generate a full line of pixel data in the time budget provided by the setting of LINE_LENGTH_PCK and the vt_pix_clk period. The usual reason for this is that the X_OUTPUT_SIZE has been set too large so that the sensor output is still padding the row with undefined pixel data when the next row starts. An alternative reason is that clock ratio between the VT clock domain and the OP clock domain does not allow sufficient time for the OP domain to complete a row before the next row starts. The first step in avoiding this error is to set X_OUTPUT_SIZE so that it matches the number of pixels generated from the pixel array (X_ADDR_END - X_ADDR_START + 1, taking into account any sub-sampling/binning and any scaling). If the error remains, the next step is to increase LINE_LENGTH_PCK. This can be caused by an over-large setting of X_OUTPUT_SIZE, where the sensor output is still padding the line with undefined data when the next line begins. The first step in solving this If the ODP clock rate and x_output_size do not allow an output line to be generated within the time allowed by line_length_pck, the “line time exceeded” error will be flagged. The general solution to this error is first to reduce x_output_size to match the size of the frame being generated by the sensor_core and then (if necessary) increase line_length_pck to allow time for the output line. Once this bit is set, you must clear the condition that caused the error then write a “1” to this bit position to clear it.	N	N

Table 12: Register Description—Manufacturer-Specific (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
12394 R0x306A	1	0x0000	datapath_status_fifo_overflow A fatal error occurred because the output FIFO overflowed. The FIFO is sized to accommodate a full-length line from the pixel array, so this error can only occur when X_OUTPUT_SIZE is unnecessarily large, or when the ratio between the VT and OP clock domains has been set incorrectly. The first step in avoiding this error is to set X_OUTPUT_SIZE so that it matches the number of rows generated from the pixel array (X_ADDR_END - X_ADDR_START + 1, taking into account any sub-sampling/binning and any scaling).	N	N
	0	0x0000	datapath_status_fifo_underflow This fatal error condition is flagged if the output FIFO detects a data underflow.	N	N
	This register flags fatal error conditions associated with incorrect configuration of the sensor. Once any bit in this register has been set, behavior of the sensor is undefined and a reset may be required to restore correct operation. All bits are cleared automatically on the transition from the software standby system state to the streaming system state.				
12398 R0x306E	15:0	0x9000	datapath_select (R/W)		
	15:13	0x0004	datapath_select_slew_rate_ctrl_parallel Selects the slew (edge) rate for the DOUT[9:0], SHUTTER, FV, LV, and FLASH outputs. Only affects SHUTTER and FLASH outputs when parallel data output is disabled. The value 7 results in the fastest edge rates on these signals. Slowing down the edge rate can reduce ringing and electromagnetic emissions.	N	N
	12:10	0x0004	datapath_select_slew_rate_ctrl_pixclk Selects the slew (edge) rate for the PIXCLK output. Has no effect when parallel data output is disabled. The value 7 results in the fastest edge rates on this signal. Slowing down the edge rate can reduce ringing and electromagnetic emissions.	N	N
	9:8	X	Reserved		
	7	RO	datapath_select_profile SMIA Profile Mode: 0 = Profile 0 1 = Profile 1/2 (this bit is read-only)	N	N
	6:5	X	Reserved		
	4	0x0000	datapath_select_true_bayer Enables true Bayer scaling mode.	N	N
	3:2	X	Reserved		
12398 R0x306E	1:0	0x0000	datapath_select_special_line_valid 00 = Normal behavior of LV 01 = LV is driven continuously (continue generating LV during vertical blanking) 10 = LV is driven continuously as LV XOR FV.	N	N

Table 12: Register Description—Manufacturer-Specific (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
12400 R0x3070	15:0	0x0000	test_pattern_mode_ (R/W)	N	Y
	0 = Normal operation: Generate output data from pixel array 1 = Solid color test pattern. 2 = 100% color bar test pattern 3 = Fade to grey color bar test pattern 4 = PN9 Link integrity test pattern 256 = Marching 1s test pattern (10 bit) 257 = Marching 1s test pattern (8 bit) other = Reserved.				
12402 R0x3072	15:0	0x0000	test_data_red_ (R/W)	N	Y
	The value for red pixels in the Bayer data used for the solid color test pattern and the test cursors.				
12404 R0x3074	15:0	0x0000	test_data_greenR_ (R/W)	N	Y
	The value for green pixels in red/green rows of the Bayer data used for the solid color test pattern and the test cursors.				
12406 R0x3076	15:0	0x0000	test_data_blue_ (R/W)	N	Y
	The value for blue pixels in the Bayer data used for the solid color test pattern and the test cursors.				
12408 R0x3078	15:0	0x0000	test_data_greenB_ (R/W)	N	Y
	The value for green pixels in blue/green rows of the Bayer data used for the solid color test pattern and the test cursors.				
12410 R0x307A	15:0	0x0000	test_raw_mode (R/W)		
	15:2	X	Reserved		
	1	0x0000	Reserved		
	0	0x0000	Reserved		
12448 R0x30A0	15:0	0x0001	x_even_inc_ (RO)	N	N
	Read-only.				
12450 R0x30A2	15:0	0x0001	x_odd_inc_ (R/W)	Y	YM
	This register field is an alias of R0x3040[7:5]				
12452 R0x30A4	15:0	0x0001	y_even_inc_ (RO)	N	N
	Read-only.				
12454 R0x30A6	15:0	0x0001	y_odd_inc_ (R/W)	Y	YM
	This register field is an alias of R0x3040[4:2]				

Table 12: Register Description—Manufacturer-Specific (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
12640 R0x3160	15:0	0x0000	global_seq_trigger (R/W)		
	15:10	X	Reserved		
	9	RO	global_seq_trigger_grst_rd Read-Only. Global reset read sequence indicator.	N	N
	8	RO	global_seq_trigger_grst_seq Read-only. Global reset sequence indicator.	N	N
	7:3	X	Reserved		
	2	0x0000	global_seq_trigger_global_flash 0 = When a Global Reset sequence is triggered, the FLASH output will remain de-asserted. 1 = When a Global Reset sequence is triggered, the FLASH output will pulse during the integration phase.	N	Y
	1	0x0000	global_seq_trigger_global_bulb 0 = Shutter open is triggered from bit[0] and shutter close is timed from the trigger point. 1 = Shutter open and close are triggered from bit[0]. This corresponds to the shutter "B" setting on a traditional camera, where "B" originally stood for "Bulb" (the shutter setting used for synchronization with a magnesium foil flash bulb) and was later considered to stand for "Brief" (an exposure that was longer than the shutter could automatically accommodate).	N	Y
	0	0x0000	global_seq_trigger_global_trigger When bit[1] = 0, a 0-to-1 transition of this bit initiates (triggers) a global reset sequence. When bit[1] = 1, a 0-to-1 transition of this bit initiates a global reset sequence, and leaves the shutter open; a 1-to-0 transition of this bit closes the shutter. These operations can also be controlled from the signal interface by enabling one of the GPI[3:0] signals as a trigger input.	N	Y
12642 R0x3162	15:0	0x0098	global_rst_end (R/W)	N	N
	Controls the duration of the global reset row reset phase. A value of N gives a duration of N * 512 / vt_pix_clk_freq_mhz.				
12644 R0x3164	15:0	0x009F	global_shutter_start (R/W)	N	N
	Controls the delay before the assertion of the SHUTTER output during a global reset sequence. A value of N gives an assertion time of N * 512 / vt_pix_clk_freq_mhz timed from the end of row that was in progress when the global reset sequence was triggered.				
12646 R0x3166	15:0	0x00A0	global_read_start (R/W)	N	N
	Controls the delay before the start of the global reset readout phase (equivalent to the end of global reset integration phase). A value of N gives a delay of N * 512 / vt_pix_clk_freq_mhz. The integration time is given by (global_read_start - global_rst_end) * 512 / vt_pix_clk_freq_mhz.				
12704 R0x31A0	15:0	0x0101	serial_format_descriptor_0 (RO)	N	N
	The value of this descriptor indicates that a single-lane CCP2 interface is available on this device.				
12706 R0x31A2	15:0	0x0201	serial_format_descriptor_1 (RO)	N	N
	The value of this descriptor indicates that a one-lane MIPI interface is available on this device.				
12708 R0x31A4	15:0	0x0202	serial_format_descriptor_2 (RO)	N	N
	The value of this descriptor indicates that a two-lane MIPI interface is available on this device.				
12710 R0x31A6	15:0	0x0000	serial_format_descriptor_3 (RO)	N	N
	The value of this descriptor indicates that it is unused.				

Table 12: Register Description—Manufacturer-Specific (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
12712 R0x31A8	15:0	0x0000	serial_format_descriptor_4 (RO)	N	N
	The value of this descriptor indicates that it is unused.				
12714 R0x31AA	15:0	0x0000	serial_format_descriptor_5 (RO)	N	N
	The value of this descriptor indicates that it is unused.				
12716 R0x31AC	15:0	0x0000	serial_format_descriptor_6 (RO)	N	N
	The value of this descriptor indicates that it is unused.				
12718 R0x31AE	15:0	0x0001	serial_format (R/W)	Y	N
	When the serial interface is enabled (reset_register[12] = 0), this register controls which serial interface is in use. Any non-zero serial_format_descriptor value is a legal value for this register. The upper byte of this register (interface type) is read-only. The lower byte is read/write.				
12720 R0x31B0	15:0	0x005B	frame_preamble (R/W)	N	N
	This timing value, expressed in op_pix_clk periods, must be large enough to allow the MIPI to wake up and start-of-frame short packet to be transmitted prior to the start of a frame of pixel data. The default value should be correct for most applications. Too small a value will result in an INSUFFICIENT_FRAME_PREAMBLE error being flagged in the DATAPATH_STATUS register.				
12722 R0x31B2	15:0	0x0030	line_preamble (R/W)	N	N
	This timing value, expressed in op_pix_clk periods, must be large enough to allow the MIPI long packet header to be transmitted prior to the start of a line of pixel data. The default value should be correct for most applications. Too small a value will result in an INSUFFICIENT_LINE_PREAMBLE error being flagged in the DATAPATH_STATUS register.				
12724 R0x31B4	15:0	0x0D57	mipi_timing_0 (R/W)		
	15:12	RO	mipi_timing_0_reserved Reserved. Read as “0.”	N	N
	11:8	0x000D	mipi_timing_0_t_hs_zero Time, in op_pix_clk periods, to drive HS-0 before the sync sequence.	N	N
	7:4	0x0005	mipi_timing_0_t_hs_trail Time, in op_pix_clk periods, to drive flipped differential state after last payload data bit of an HS transmission burst.	N	N
	3:0	0x0007	mipi_timing_0_t_clk_trail Time, in op_pix_clk periods, to drive HS differential state after last payload clock bit of an HS transmission burst.	N	N
12726 R0x31B6	15:0	0x0B10	mipi_timing_1 (R/W)		
	15:14	RO	mipi_timing_1_reserved_1 Reserved. Read as “0.”	N	N
	13:8	0x000B	mipi_timing_1_t_hs_exit Time, in op_pix_clk periods, to drive LP-11 after HS burst	N	N
	7:6	RO	mipi_timing_1_reserved_0 Reserved. Read as “0.”	N	N
	5:0	0x0010	mipi_timing_1_t_clk_zero Minimum time, in op_pix_clk periods, to drive HS-0 on clock lane prior to starting clock.	N	N

Table 12: Register Description—Manufacturer-Specific (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
12728 R0x31B8	15:0	0x010D	mipi_timing_2 (R/W)		
	15:14	RO	mipi_timing_2_reserved_1 Reserved. Read as "0."	N	N
	13:8	0x0001	mipi_timing_2_t_clk_pre Time, in op_pix_clk periods, to drive the HS clock before any data lane might start up.	N	N
	7:6	RO	mipi_timing_2_reserved_0 Reserved. Read as "0."	N	N
	5:0	0x000D	mipi_timing_2_t_clk_post Time, in op_pix_clk periods, to drive the HS clock after the data lane has gone into low-power mode.	N	N
12730 R0x31BA	15:0	0x050D	mipi_timing_3 (R/W)		
	15:14	RO	mipi_timing_3_reserved_1 Reserved. Read as "0."	N	N
	13:8	0x0005	mipi_timing_3_t_lpx Time, in op_pix_clk periods, of any low-power state period.	N	N
	7	RO	mipi_timing_3_reserved_0 Reserved. Read as "0."	N	N
	6:0	0x000D	mipi_timing_3_t_wake_up Time to recover from ultra low-power mode (ULPM). ULPM is exited by applying a mark state for (8192) * T_WAKE_UP * op_pix_clk.	N	N
12732 R0x31BC	15:0	0x000B	mipi_timing_4 (R/W)		
	15:7	RO	mipi_timing_4_reserved Reserved. Read as "0."	N	N
	6:0	0x000B	mipi_timing_4_t_init Initialization time when first entering stop state (LP-11) after power up or reset. LP-11 is transmitted for a minimum of (1024) * T_INIT * op_pix_clk.	N	N
12776 R0x31E8	15:0	0x0000	horizontal_cursor_position_ (R/W)	N	N
	Specify the start row for the test cursor.				
12778 R0x31EA	15:0	0x0000	vertical_cursor_position_ (R/W)	N	N
	Specify the start column for the test cursor.				
12780 R0x31EC	15:0	0x0000	horizontal_cursor_width_ (R/W)	N	N
	Specify the width, in rows, of the horizontal test cursor. A width of 0 disables the cursor.				
12782 R0x31EE	15:0	0x0000	vertical_cursor_width_ (R/W)	N	N
	Specify the width, in columns, of the vertical test cursor. A width of 0 disables the cursor.				
12786 R0x31F2	15:0	0x6E6C	i2c_ids_mipi_default (R/W)	N	N
	Programmable two-wire serial interface slave addresses for MIPI operation.				
12796 R0x31FC	15:0	0x3020	i2c_ids (R/W)	N	N
	I2C addresses.				
13824 R0x3600	15:0	0x0000	p_gr_p0q0 (R/W)	N	N
	P0 coefficient for Q0 for Gr. P_GR_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gr pixels.				
13826 R0x3602	15:0	0x0000	p_gr_p0q1 (R/W)	N	N
	P0 coefficient for Q1 for Gr. P_GR_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gr pixels.				

Table 12: Register Description—Manufacturer-Specific (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
13828 R0x3604	15:0	0x0000	p_gr_p0q2 (R/W)	N	N
	P0 coefficient for Q2 for Gr. P_GR_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gr pixels.				
13830 R0x3606	15:0	0x0000	p_gr_p0q3 (R/W)	N	N
	P0 coefficient for Q3 for Gr. P_GR_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gr pixels.				
13832 R0x3608	15:0	0x0000	p_gr_p0q4 (R/W)	N	N
	P0 coefficient for Q4 for Gr. P_GR_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gr pixels.				
13834 R0x360A	15:0	0x0000	p_rd_p0q0 (R/W)	N	N
	P0 coefficient for Q0 for Rd. P_RD_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Rd pixels.				
13836 R0x360C	15:0	0x0000	p_rd_p0q1 (R/W)	N	N
	P0 coefficient for Q1 for Rd. P_RD_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Rd pixels.				
13838 R0x360E	15:0	0x0000	p_rd_p0q2 (R/W)	N	N
	P0 coefficient for Q2 for Rd. P_RD_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Rd pixels.				
13840 R0x3610	15:0	0x0000	p_rd_p0q3 (R/W)	N	N
	P0 coefficient for Q3 for Rd. P_RD_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Rd pixels.				
13842 R0x3612	15:0	0x0000	p_rd_p0q4 (R/W)	N	N
	P0 coefficient for Q4 for Rd. P_RD_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Rd pixels.				
13844 R0x3614	15:0	0x0000	p_bl_p0q0 (R/W)	N	N
	P0 coefficient for Q0 for Bl. P_BL_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Bl pixels.				
13846 R0x3616	15:0	0x0000	p_bl_p0q1 (R/W)	N	N
	P0 coefficient for Q1 for Bl. P_BL_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Bl pixels.				
13848 R0x3618	15:0	0x0000	p_bl_p0q2 (R/W)	N	N
	P0 coefficient for Q2 for Bl. P_BL_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Bl pixels.				
13850 R0x361A	15:0	0x0000	p_bl_p0q3 (R/W)	N	N
	P0 coefficient for Q3 for Bl. P_BL_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Bl pixels.				
13852 R0x361C	15:0	0x0000	p_bl_p0q4 (R/W)	N	N
	P0 coefficient for Q4 for Bl. P_BL_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Bl pixels.				
13854 R0x361E	15:0	0x0000	p_gb_p0q0 (R/W)	N	N
	P0 coefficient for Q0 for Gb. P_GB_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gb pixels.				
13856 R0x3620	15:0	0x0000	p_gb_p0q1 (R/W)	N	N
	P0 coefficient for Q1 for Gb. P_GB_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gb pixels.				

Table 12: Register Description—Manufacturer-Specific (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
13858 R0x3622	15:0	0x0000	p_gb_p0q2 (R/W)	N	N
	P0 coefficient for Q2 for Gb. P_GB_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gb pixels.				
13860 R0x3624	15:0	0x0000	p_gb_p0q3 (R/W)	N	N
	P0 coefficient for Q3 for Gb. P_GB_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gb pixels.				
13862 R0x3626	15:0	0x0000	p_gb_p0q4 (R/W)	N	N
	P0 coefficient for Q4 for Gb. P_GB_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gb pixels.				
13888 R0x3640	15:0	0x0000	p_gr_p1q0 (R/W)	N	N
	P1 coefficient for Q0 for Gr. P_GR_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gr pixels.				
13890 R0x3642	15:0	0x0000	p_gr_p1q1 (R/W)	N	N
	P1 coefficient for Q1 for Gr. P_GR_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gr pixels.				
13892 R0x3644	15:0	0x0000	p_gr_p1q2 (R/W)	N	N
	P1 coefficient for Q2 for Gr. P_GR_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gr pixels.				
13894 R0x3646	15:0	0x0000	p_gr_p1q3 (R/W)	N	N
	P1 coefficient for Q3 for Gr. P_GR_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gr pixels.				
13896 R0x3648	15:0	0x0000	p_gr_p1q4 (R/W)	N	N
	P1 coefficient for Q4 for Gr. P_GR_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gr pixels.				
13898 R0x364A	15:0	0x0000	p_rd_p1q0 (R/W)	N	N
	P1 coefficient for Q0 for Rd. P_RD_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Rd pixels.				
13900 R0x364C	15:0	0x0000	p_rd_p1q1 (R/W)	N	N
	P1 coefficient for Q1 for Rd. P_RD_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Rd pixels.				
13902 R0x364E	15:0	0x0000	p_rd_p1q2 (R/W)	N	N
	P1 coefficient for Q2 for Rd. P_RD_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Rd pixels.				
13904 R0x3650	15:0	0x0000	p_rd_p1q3 (R/W)	N	N
	P1 coefficient for Q3 for Rd. P_RD_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Rd pixels.				
13906 R0x3652	15:0	0x0000	p_rd_p1q4 (R/W)	N	N
	P1 coefficient for Q4 for Rd. P_RD_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Rd pixels.				
13908 R0x3654	15:0	0x0000	p_bl_p1q0 (R/W)	N	N
	P1 coefficient for Q0 for Bl. P_BL_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Bl pixels.				
13910 R0x3656	15:0	0x0000	p_bl_p1q1 (R/W)	N	N
	P1 coefficient for Q1 for Bl. P_BL_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Bl pixels.				

Table 12: Register Description—Manufacturer-Specific (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
13912 R0x3658	15:0	0x0000	p_bl_p1q2 (R/W)	N	N
	P1 coefficient for Q2 for Bl. P_BL_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Bl pixels.				
13914 R0x365A	15:0	0x0000	p_bl_p1q3 (R/W)	N	N
	P1 coefficient for Q3 for Bl. P_BL_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Bl pixels.				
13916 R0x365C	15:0	0x0000	p_bl_p1q4 (R/W)	N	N
	P1 coefficient for Q4 for Bl. P_BL_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Bl pixels.				
13918 R0x365E	15:0	0x0000	p_gb_p1q0 (R/W)	N	N
	P1 coefficient for Q0 for Gb. P_GB_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gb pixels.				
13920 R0x3660	15:0	0x0000	p_gb_p1q1 (R/W)	N	N
	P1 coefficient for Q1 for Gb. P_GB_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gb pixels.				
13922 R0x3662	15:0	0x0000	p_gb_p1q2 (R/W)	N	N
	P1 coefficient for Q2 for Gb. P_GB_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gb pixels.				
13924 R0x3664	15:0	0x0000	p_gb_p1q3 (R/W)	N	N
	P1 coefficient for Q3 for Gb. P_GB_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gb pixels.				
13926 R0x3666	15:0	0x0000	p_gb_p1q4 (R/W)	N	N
	P1 coefficient for Q4 for Gb. P_GB_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gb pixels.				
13952 R0x3680	15:0	0x0000	p_gr_p2q0 (R/W)	N	N
	P2 coefficient for Q0 for Gr. P_GR_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gr pixels.				
13954 R0x3682	15:0	0x0000	p_gr_p2q1 (R/W)	N	N
	P2 coefficient for Q1 for Gr. P_GR_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gr pixels.				
13956 R0x3684	15:0	0x0000	p_gr_p2q2 (R/W)	N	N
	P2 coefficient for Q2 for Gr. P_GR_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gr pixels.				
13958 R0x3686	15:0	0x0000	p_gr_p2q3 (R/W)	N	N
	P2 coefficient for Q3 for Gr. P_GR_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gr pixels.				
13960 R0x3688	15:0	0x0000	p_gr_p2q4 (R/W)	N	N
	P2 coefficient for Q4 for Gr. P_GR_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gr pixels.				
13962 R0x368A	15:0	0x0000	p_rd_p2q0 (R/W)	N	N
	P2 coefficient for Q0 for Rd. P_RD_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Rd pixels.				
13964 R0x368C	15:0	0x0000	p_rd_p2q1 (R/W)	N	N
	P2 coefficient for Q1 for Rd. P_RD_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Rd pixels.				

Table 12: Register Description—Manufacturer-Specific (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
13966 R0x368E	15:0	0x0000	p_rd_p2q2 (R/W)	N	N
	P2 coefficient for Q2 for Rd. P_RD_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Rd pixels.				
13968 R0x3690	15:0	0x0000	p_rd_p2q3 (R/W)	N	N
	P2 coefficient for Q3 for Rd. P_RD_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Rd pixels.				
13970 R0x3692	15:0	0x0000	p_rd_p2q4 (R/W)	N	N
	P2 coefficient for Q4 for Rd. P_RD_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Rd pixels.				
13972 R0x3694	15:0	0x0000	p_bl_p2q0 (R/W)	N	N
	P2 coefficient for Q0 for Bl. P_BL_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Bl pixels.				
13974 R0x3696	15:0	0x0000	p_bl_p2q1 (R/W)	N	N
	P2 coefficient for Q1 for Bl. P_BL_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Bl pixels.				
13976 R0x3698	15:0	0x0000	p_bl_p2q2 (R/W)	N	N
	P2 coefficient for Q2 for Bl. P_BL_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Bl pixels.				
13978 R0x369A	15:0	0x0000	p_bl_p2q3 (R/W)	N	N
	P2 coefficient for Q3 for Bl. P_BL_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Bl pixels.				
13980 R0x369C	15:0	0x0000	p_bl_p2q4 (R/W)	N	N
	P2 coefficient for Q4 for Bl. P_BL_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Bl pixels.				
13982 R0x369E	15:0	0x0000	p_gb_p2q0 (R/W)	N	N
	P2 coefficient for Q0 for Gb. P_GB_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gb pixels.				
13984 R0x36A0	15:0	0x0000	p_gb_p2q1 (R/W)	N	N
	P2 coefficient for Q1 for Gb. P_GB_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gb pixels.				
13986 R0x36A2	15:0	0x0000	p_gb_p2q2 (R/W)	N	N
	P2 coefficient for Q2 for Gb. P_GB_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gb pixels.				
13988 R0x36A4	15:0	0x0000	p_gb_p2q3 (R/W)	N	N
	P2 coefficient for Q3 for Gb. P_GB_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gb pixels.				
13990 R0x36A6	15:0	0x0000	p_gb_p2q4 (R/W)	N	N
	P2 coefficient for Q4 for Gb. P_GB_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gb pixels.				
14016 R0x36C0	15:0	0x0000	p_gr_p3q0 (R/W)	N	N
	P3 coefficient for Q0 for Gr. P_GR_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gr pixels.				
14018 R0x36C2	15:0	0x0000	p_gr_p3q1 (R/W)	N	N
	P3 coefficient for Q1 for Gr. P_GR_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gr pixels.				

Table 12: Register Description—Manufacturer-Specific (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
14020 R0x36C4	15:0	0x0000	p_gr_p3q2 (R/W)	N	N
	P3 coefficient for Q2 for Gr. P_GR_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gr pixels.				
14022 R0x36C6	15:0	0x0000	p_gr_p3q3 (R/W)	N	N
	P3 coefficient for Q3 for Gr. P_GR_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gr pixels.				
14024 R0x36C8	15:0	0x0000	p_gr_p3q4 (R/W)	N	N
	P3 coefficient for Q4 for Gr. P_GR_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gr pixels.				
14026 R0x36CA	15:0	0x0000	p_rd_p3q0 (R/W)	N	N
	P3 coefficient for Q0 for Rd. P_RD_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Rd pixels.				
14028 R0x36CC	15:0	0x0000	p_rd_p3q1 (R/W)	N	N
	P3 coefficient for Q1 for Rd. P_RD_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Rd pixels.				
14030 R0x36CE	15:0	0x0000	p_rd_p3q2 (R/W)	N	N
	P3 coefficient for Q2 for Rd. P_RD_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Rd pixels.				
14032 R0x36D0	15:0	0x0000	p_rd_p3q3 (R/W)	N	N
	P3 coefficient for Q3 for Rd. P_RD_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Rd pixels.				
14034 R0x36D2	15:0	0x0000	p_rd_p3q4 (R/W)	N	N
	P3 coefficient for Q4 for Rd. P_RD_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Rd pixels.				
14036 R0x36D4	15:0	0x0000	p_bl_p3q0 (R/W)	N	N
	P3 coefficient for Q0 for Bl. P_BL_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Bl pixels.				
14038 R0x36D6	15:0	0x0000	p_bl_p3q1 (R/W)	N	N
	P3 coefficient for Q1 for Bl. P_BL_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Bl pixels.				
14040 R0x36D8	15:0	0x0000	p_bl_p3q2 (R/W)	N	N
	P3 coefficient for Q2 for Bl. P_BL_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Bl pixels.				
14042 R0x36DA	15:0	0x0000	p_bl_p3q3 (R/W)	N	N
	P3 coefficient for Q3 for Bl. P_BL_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Bl pixels.				
14044 R0x36DC	15:0	0x0000	p_bl_p3q4 (R/W)	N	N
	P3 coefficient for Q4 for Bl. P_BL_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Bl pixels.				
14046 R0x36DE	15:0	0x0000	p_gb_p3q0 (R/W)	N	N
	P3 coefficient for Q0 for Gb. P_GB_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gb pixels.				
14048 R0x36E0	15:0	0x0000	p_gb_p3q1 (R/W)	N	N
	P3 coefficient for Q1 for Gb. P_GB_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gb pixels.				

Table 12: Register Description—Manufacturer-Specific (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
14050 R0x36E2	15:0	0x0000	p_gb_p3q2 (R/W)	N	N
	P3 coefficient for Q2 for Gb. P_GB_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gb pixels.				
14052 R0x36E4	15:0	0x0000	p_gb_p3q3 (R/W)	N	N
	P3 coefficient for Q3 for Gb. P_GB_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gb pixels.				
14054 R0x36E6	15:0	0x0000	p_gb_p3q4 (R/W)	N	N
	P3 coefficient for Q4 for Gb. P_GB_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gb pixels.				
14080 R0x3700	15:0	0x0000	p_gr_p4q0 (R/W)	N	N
	P4 coefficient for Q0 for Gr. P_GR_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gr pixels.				
14082 R0x3702	15:0	0x0000	p_gr_p4q1 (R/W)	N	N
	P4 coefficient for Q1 for Gr. P_GR_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gr pixels.				
14084 R0x3704	15:0	0x0000	p_gr_p4q2 (R/W)	N	N
	P4 coefficient for Q2 for Gr. P_GR_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gr pixels.				
14086 R0x3706	15:0	0x0000	p_gr_p4q3 (R/W)	N	N
	P4 coefficient for Q3 for Gr. P_GR_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gr pixels.				
14088 R0x3708	15:0	0x0000	p_gr_p4q4 (R/W)	N	N
	P4 coefficient for Q4 for Gr. P_GR_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gr pixels.				
14090 R0x370A	15:0	0x0000	p_rd_p4q0 (R/W)	N	N
	P4 coefficient for Q0 for Rd. P_RD_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Rd pixels.				
14092 R0x370C	15:0	0x0000	p_rd_p4q1 (R/W)	N	N
	P4 coefficient for Q1 for Rd. P_RD_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Rd pixels.				
14094 R0x370E	15:0	0x0000	p_rd_p4q2 (R/W)	N	N
	P4 coefficient for Q2 for Rd. P_RD_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Rd pixels.				
14096 R0x3710	15:0	0x0000	p_rd_p4q3 (R/W)	N	N
	P4 coefficient for Q3 for Rd. P_RD_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Rd pixels.				
14098 R0x3712	15:0	0x0000	p_rd_p4q4 (R/W)	N	N
	P4 coefficient for Q4 for Rd. P_RD_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Rd pixels.				
14100 R0x3714	15:0	0x0000	p_bl_p4q0 (R/W)	N	N
	P4 coefficient for Q0 for Bl. P_BL_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Bl pixels.				
14102 R0x3716	15:0	0x0000	p_bl_p4q1 (R/W)	N	N
	P4 coefficient for Q1 for Bl. P_BL_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Bl pixels.				

Table 12: Register Description—Manufacturer-Specific (Continued)
1 = read-only, always 1; 0 = read-only, always 0; d = programmable; ? = read-only, dynamic

Register Dec (Hex)	Bits	Default	Name	Frame Sync'd	Bad Frame
14104 R0x3718	15:0	0x0000	p_bl_p4q2 (R/W)	N	N
	P4 coefficient for Q2 for Bl. P_BL_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Bl pixels.				
14106 R0x371A	15:0	0x0000	p_bl_p4q3 (R/W)	N	N
	P4 coefficient for Q3 for Bl. P_BL_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Bl pixels.				
14108 R0x371C	15:0	0x0000	p_bl_p4q4 (R/W)	N	N
	P4 coefficient for Q4 for Bl. P_BL_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Bl pixels.				
14110 R0x371E	15:0	0x0000	p_gb_p4q0 (R/W)	N	N
	P4 coefficient for Q0 for Gb. P_GB_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gb pixels.				
14112 R0x3720	15:0	0x0000	p_gb_p4q1 (R/W)	N	N
	P4 coefficient for Q1 for Gb. P_GB_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gb pixels.				
14114 R0x3722	15:0	0x0000	p_gb_p4q2 (R/W)	N	N
	P4 coefficient for Q2 for Gb. P_GB_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gb pixels.				
14116 R0x3724	15:0	0x0000	p_gb_p4q3 (R/W)	N	N
	P4 coefficient for Q3 for Gb. P_GB_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gb pixels.				
14118 R0x3726	15:0	0x0000	p_gb_p4q4 (R/W)	N	N
	P4 coefficient for Q4 for Gb. P_GB_PpQq registers are read successively when the row (Q) coefficients are calculated during the horizontal blanking period before a row containing Gb pixels.				
14208 R0x3780	15:0	0x0000	sc_enable (R/W)		
	15	0x0000	sc_en Turn on shading correction.	N	N
	14:0	X	Reserved		
	When SC_ENABLE bit is set shading correction will generate function and correct stream of pixels. When not set shading correction will bypass data.				
14210 R0x3782	15:0	0x0000	origin_c (R/W)	N	N
	Origin of shading correction function: applied as offset to X (col) coordinate of pixel.				
14212 R0x3784	15:0	0x0000	origin_r (R/W)	N	N
	Origin of shading correction function: applied as offset to Y (row) coordinate of pixel.				
14336 R0x3800	15:0	0x0000	otpm_data_0 (R/W)	N	N
	OTPM_DATA_0				
14338 R0x3802	15:0	0x0000	otpm_data_1 (R/W)	N	N
	OTPM_DATA_1				

Programming Restrictions

Table 14 shows a list of programming rules that must be adhered to for correct operation of the MT9N001. It is recommended that these rules are encoded into the device driver stack—either implicitly or explicitly.

Table 13: 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; yskip = 8 if y_odd_inc = 15; yskip = 16 if y_odd_inc = 31; yskip = 32 if y_odd_inc = 63

Table 14: Programming Rules

Parameter	Minimum Value	Maximum Value
coarse_integration_time	coarse_integration_time_min	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	
frame_length_lines	((y_addr_end - y_addr_start + y_odd_inc)/yskip) + min_frame_blanking_lines	
x_addr_start	x_addr_min	x_addr_max
x_addr_end	x_addr_start	x_addr_max
(x_addr_end - x_addr_start+ x_odd_inc)	must be positive	must be positive
x_addr_start[0]	0	0
x_addr_end[0]	1	1
y_addr_start	y_addr_min	y_addr_max
y_addr_end	y_addr_start	y_addr_max
(y_addr_end - y_addr_start + y_odd_inc)/	must be positive	must be positive
y_addr_start[0]	0	0
y_addr_end[0]	1	1
x_even_inc	min_even_inc	max_even_inc
x_even_inc[0]	1	1
y_even_inc	min_even_inc	max_even_inc
y_even_inc[0]	1	1
x_odd_inc	min_odd_inc	max_odd_inc
x_odd_inc[0]	1	1
y_odd_inc	min_odd_inc	max_odd_inc
y_odd_inc[0]	1	1

Table 14: Programming Rules (Continued)

Parameter	Minimum Value	Maximum Value
scale_m	scaler_m_min	scaler_m_max
scale_n	scaler_n_min	scaler_n_max
x_output_size	256	2608
x_output_size[0]	0 (this is enforced in hardware: bit[0] is read-only)	0
y_output_size	2	frame_length_lines
y_output_size[0]	0 (this is enforced in hardware: bit[0] is read-only)	0
With subsampling, start and end pixels must be addressed (impact on x/y start/end addresses, function of image orientation bits)		

Output Size Restrictions

The design specification imposes the restriction that an output line shall be 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] = 8 (RAW8 data), x_output_size must be a multiple of 4 (x_output_size[1:0] = 0).
- When ccp_data_format[7:0] = 10 (RAW10 data), x_output_size must be a multiple of 16 (x_output_size[3:0] = 0).
- When ccp_data_format[7:0] = 12 (RAW12 data), x_output_size must be a multiple of 8 (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.

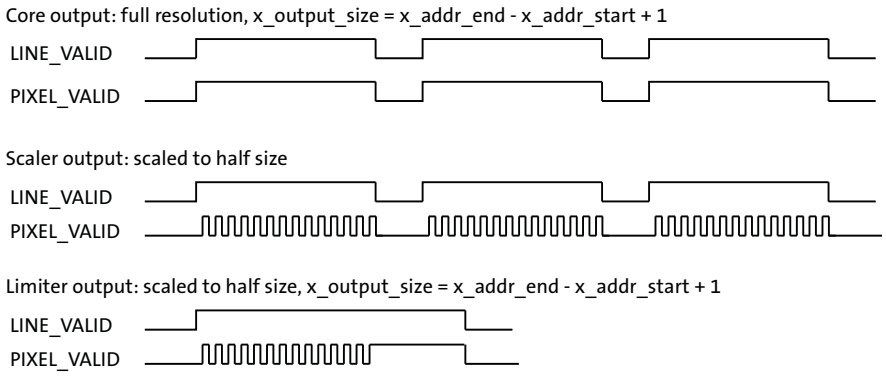
When the parallel pixel data path is in use, the only restriction on x_output_size is that it must be even (x_output_size[0] = 0), and this restriction is enforced in hardware.

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 MT9N001 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 17 on page 71.

Figure 17: Effect of Limiter on the Data Path



In Figure 17, three different stages in the data path (see "Digital Data Path" on page 115) 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.

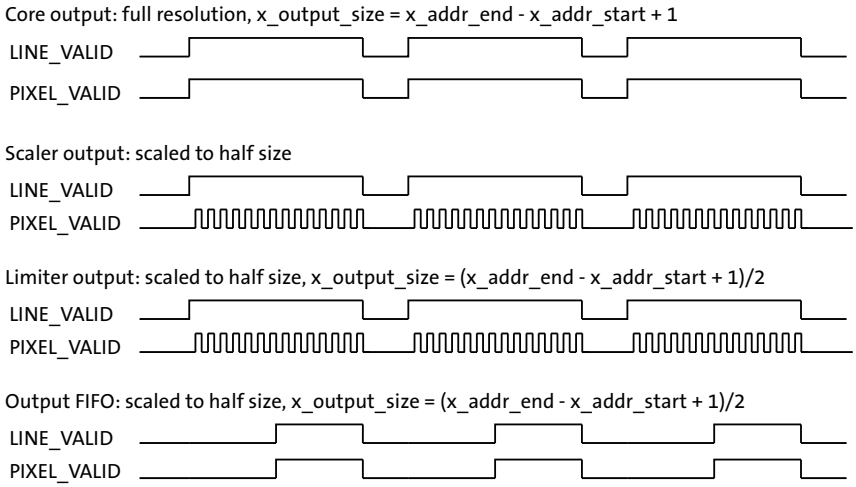
The second stage is the output of the scaler, when the scaler is set to reduce the image size by one-half in each dimension. The effect of the scaler is to combine groups of pixels. Therefore, the row time remains the same, but only half the pixels out of the scaler are valid. This is signalled by transitions 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 MT9N001 will cease to generate output frames.

A correct configuration is shown in Figure 18 on page 72, 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 18 on page 72 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 18: 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

Programming Restrictions when Using Global Reset

Interactions between the registers that control the global reset imposes some programming restrictions on the way in which they are used; these are discussed in "Global Reset" on page 105.

Control of the Signal Interface

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

Serial Register Interface

The serial register interface uses these signals:

- SCLK
- SDATA
- SADDR (through the GPI pad)

SCLK is an input-only signal and must always be driven to a valid logic level for correct operation; if the driving device can place this signal in High-Z, an external pull-up resistor should be connected on this signal.

SDATA is a bidirectional signal. An external pull-up resistor should be connected on this signal.

SADDR is a signal, which can be optionally enabled and controlled by a GPI pad, to select an alternate slave address. These slave addresses can also be programmed through R0x31FC.

This interface is described in detail in "Two-Wire Serial Register Interface" on page 19.

Default Power-Up State

The MT9N001 provides interfaces for pixel data through the CCP2 high speed serial interface described by the SMIA specification or the MIPI serial interface and a parallel data interface.

At power-up and after a hard or soft reset, the reset state of the MT9N001 is to enable the SMIA CCP2 high speed serial interface for a CCP2-configured sensor, and CSI-2 high speed serial interface for a MIPI-configured sensor.

The CCP2 and MIPI serial interfaces share pins, and only one can be enabled at a time. This is done at the factory.

Serial Pixel Data Interface

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

- DATA0_P
- DATA0_N
- DATA1_P
- DATA1_N
- CLK_P
- CLK_N

The signal pairs are driven differentially using sub-LVDS switching levels. The serial pixel data interface is enabled by default at power up and after reset. DATA0_P and DATA0_N are the data pair for the CCP2 or one-lane MIPI serial interface, while DATA1_P and DATA1_N are the second data pair for two-lane serial MIPI.

The DATA0_P, DATA0_N, DATA1_P, DATA1_N, CLK_P, and CLK_N pads are turned off if the SMIA serial disable bit is asserted (R0x301A–B[12] = 1) or when the sensor is in the soft standby state.

In data/clock mode, the clock remains HIGH when no data is being transmitted. In data/strobe mode before frame start, clock is LOW and data is HIGH.

When the serial pixel data interface is used, the LV, FV, PIXCLK, and DOUT[11:0] signals can be left unconnected.

R0x0112-3 (ccp_data_format) The following data formats are setting supported:

- 0x0A0A – sensor supports RAW10 uncompressed data format. A sensor with a 12-bit ADC can support this mode by discarding all but the upper 10 bits of a pixel value.
- 0x0C0C – sensor supports RAW12 uncompressed data format
- 0x0808 – sensor supports RAW8 uncompressed data format. A sensor with a 12-bit or 10-bit ADC can support this mode 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.
- 0x0C08 – sensor supports RAW8 data format in which an adaptive compression algorithm is used to perform 12-bit to 8-bit compression on the upper 12 bits of each pixel value.

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

- 0x0101 – sensor supports one-lane CCP2 operation.
- 0x0201 – sensor supports one-lane MIPI operation.
- 0x0202 – sensor supports two-lane MIPI operation.

Parallel Pixel Data Interface

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

- FV
- LV
- PIXCLK
- DOUT[11:0]

The parallel pixel data interface is disabled by default at power up and after reset. It can be enabled by programming R0x301A. Table 16 shows the recommended settings.

When the parallel pixel data interface is in use, the DATA0_P, DATA0_N, DATA1_P, DATA1_N, CLK_P, and CLK_N signals can be left unconnected. Set reset_register[12] to disable the serializer while in parallel output mode.

Output Enable Control

When the parallel pixel data interface is enabled, its signals can be switched asynchronously between the driven and High-Z under pin or register control, as shown in Table 15. Selection of a pin to use for the OE_N function is described in "General Purpose Inputs" on page 80.

Table 15: Output Enable Control

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

Configuration of the Pixel Data Interface

Fields in R0x301A are used to configure the operation of the pixel data interface. The supported combinations are shown in Table 16.

Table 16: Configuration of the Pixel Data Interface

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

System States

The system states of the MT9N001 are represented as a state diagram in Figure 19 and described in subsequent sections. The effect of RESET_BAR on the system state and the configuration of the PLL in the different states are shown in Table 17 on page 78.

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 17.

Figure 19: MT9N001 System States

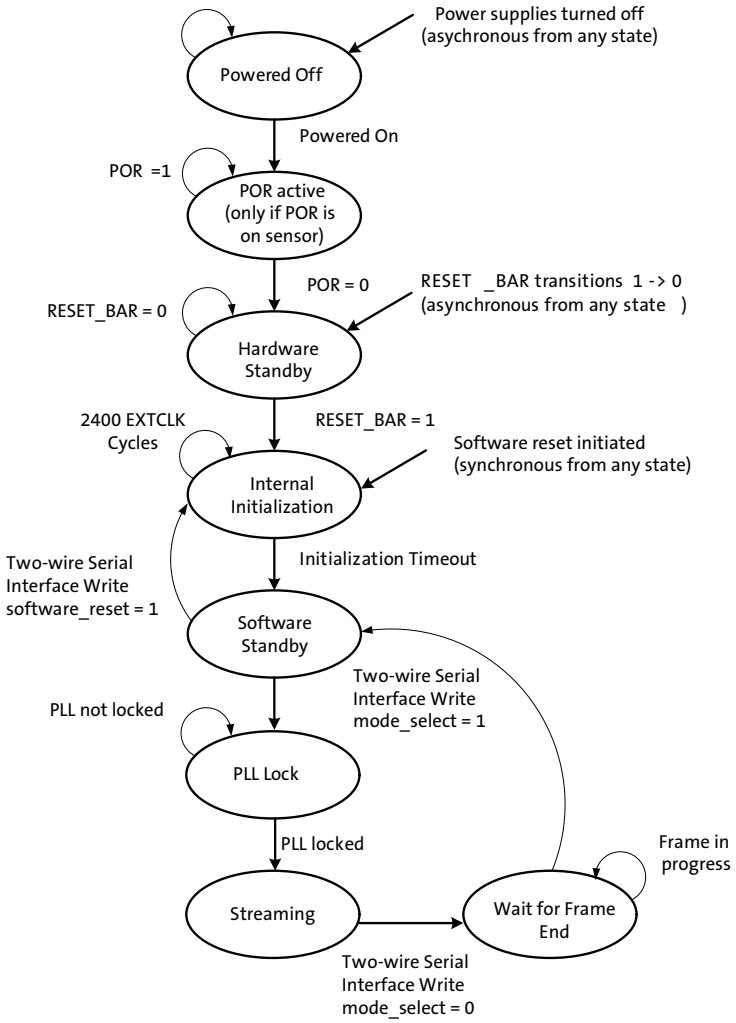


Table 17: RESET_BAR and PLL in System States

State	EXTCLKs	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		

Notes: 1. VCO = voltage-controlled oscillator.

Power-On Reset Sequence

When power is applied to the MT9N001, it enters a low-power hardware standby state. Exit from this state is controlled by the later of two events:

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

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

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

While RESET_BAR is asserted (or the internal power-on reset circuit is active) the MT9N001 is in its lowest-powered, powered-up state; the internal PLL is disabled, the CCP2 serializer is disabled and internal clocks are gated off.

When the sensor leaves the hardware standby state it performs an internal initialization sequence that takes 2400 EXTCLK cycles. After this, it enters a low-power software standby state. While the initialization sequence is in progress, the MT9N001 will not respond to READ transactions on its two-wire serial interface. Therefore, a method to determine when the initialization sequence has completed is to poll a sensor register; for example, R0x0000. While the initialization sequence is in progress, the sensor will not respond to its device address and READs from the sensor will result in a NACK on the two-wire serial interface bus. When the sequence has completed, READs will return the operational value for the register (0x2800 if R0x0000 is read).

When the sensor leaves software standby mode and enables the VCO, an internal delay will keep the PLL disconnected for up to 1ms so that the PLL can lock. The VCO lock time is 200µs(typical), 1ms (maximum).

Soft Reset Sequence

The MT9N001 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 18 shows the state of the signal interface during hardware standby (RESET_BAR asserted) and the default state during software standby. After exit from hardware standby and before any registers within the sensor have been changed from their default power-up values.

Table 18: Signal State During Reset

Pad Name	Pad Type	Hardware Standby	Software Standby
EXTCLK	Input	Enabled. Must be driven to a valid logic level.	
RESET_BAR (XSHUTDOWN)	Input	Enabled. Must be driven to a valid logic level.	
LINE_VALID	Output	High-Z. Can be left disconnected/floating.	
FRAME_VALID	Output		
DOUT[11:0]	Output		
PIXCLK	Output		
SCLK	Input	Enabled. Must be pulled up or driven to a valid logic level.	
SDATA	I/O	Enabled as an input. Must be pulled up or driven to a valid logic level.	
FLASH	Output	High-Z.	Logic 0.
SHUTTER	Output	High-Z.	Logic 0.
DATA0_P	Output	CCP2: High Z MIPI: Ultra Low-Power State (ULPS), represented as an LP-00 state on the input (both wires at 0V).	
DATA0_N	Output		
DATA1_P	Output		
DATA1_N	Output		
CLK_P	Output		
CLK_N	Output		
GPI[3:0]	Input	Powered down. Can be left disconnected/floating.	
TEST	Input	Enabled. Must be driven to a logic 0 for a serial CCP2-configured sensor, or 1 for a serial MIPI-configured sensor.	

General Purpose Inputs

The MT9N001 provides four general purpose inputs. After reset, the input pads associated with these signals are powered down by default, allowing the pads to be left disconnected/floating.

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

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

- Output enable (see "Output Enable Control" on page 76)
- Trigger (see the sections below)
- Standby functions
- SADDR selection (see "Serial Register Interface" on page 74)

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

Streaming/Standby Control

The MT9N001 can be switched between its soft standby and streaming states under pin or register control, as shown in Table 19. Selection of a pin to use for the STANDBY function is described in "General Purpose Inputs" on page 80. The state diagram for transitions between soft standby and streaming states is shown in Figure 19 on page 77.

Table 19: Streaming/STANDBY

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

Trigger Control

When the global reset feature is in use, the trigger for the sequence can be initiated either under pin or register control, as shown in Table 20. Selection of a pin to use for the TRIGGER function is described in "General Purpose Inputs" on page 80.

Table 20: Trigger Control

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

PLL

The sensor 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. The clocking structure is shown in Figure 20.

Figure 20: Clocking Structure

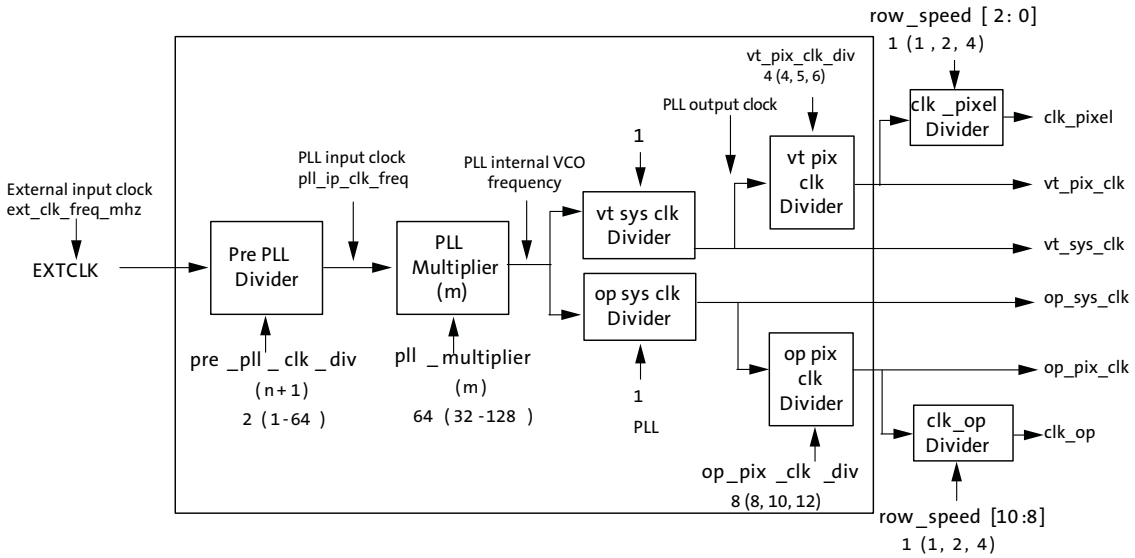


Figure 20 shows the different clocks and the register names. It also shows the default setting for each divider/multiplier control register, and the range of legal values for each divider/multiplier control register. The vt and op sys clk Divider is hardwired in the design.

From the diagram, the clock frequencies can be calculated as follows:

Internal pixel clock used to readout the pixel array:

(EQ 1)

$$clk\_pixel\_freq\_mhz = \frac{ext\_clk\_freq\_mhz \times pll\_multiplier}{pre\_pll\_clk\_div \times vt\_pix\_clk\_div \times row\_speed\ [2:0]} = \frac{24\ MHz \times 64}{2x4x1} = 192\ MHz$$

External pixel clock used to output the data:

(EQ 2)

$$clk\_op\_freq\_mhz = \frac{ext\_clk\_freq\_mhz \times pll\_multiplier}{pre\_pll\_clk\_div \times op\_pix\_clk\_div \times row\_speed\ [10:8]} = \frac{24\ MHz \times 64}{2x8x1} = 96\ MHz$$

Internal master clock:

(EQ 3)

$$vt\_pix\_clk\_freq\_mhz/2$$

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.

The following 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 2–24 MHz. Higher frequencies are preferred.
  PLL internal VCO frequency must be in the range 384–768 MHz.

- The minimum/maximum value for the divider/multiplier must be met.

  Range for m: 32–128.
  Range for n: 0–63. Range for (n + 1): 1–64.

- The op_pix_clk must never run faster than the vt_pix_clk 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 is used by the sensor core to control the timing of the pixel array. The sensor core produces one 12-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 is used to load parallel pixel data from the output FIFO. The output FIFO generates one pixel each op_pix_clk period.

Programming the PLL Divisors

The PLL divisors should be programmed while the MT9N001 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 MT9N001 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 12, 10, or 8 bits per pixel.

When the pixel data interface is generating 8 bits per-pixel, op_pix_clk_div must be programmed with the value 8. When the pixel data interface is generating 10 bits per pixel, op_pix_clk_div must be programmed with the value 10.

Clock Control

The MT9N001 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 MT9N001 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

Scaler

The MT9N001 sensor includes scaling capabilities. This allows the user to generate full field-of-view, low resolution images. Scaling is advantageous because it uses all pixel values to calculate the output image which helps to avoid aliasing. It is also more convenient than binning because the scale factor varies smoothly and the user is not limited to certain ratios of size resolution.

The scaling factor is programmable in 1/16 steps.

$$ScaleFactor = \frac{scale\_n}{scale\_m} = \frac{16}{scale\_m} \quad \text{(EQ 4)}$$

scale_n is fixed at 16.
scale_m is adjustable with R0x0404 (scale_m)
Legal values for m are 16 through 128. The user has the ability to scale from 1:1 (*scale_m* = 16) to 1:8 (*scale_m* = 128).

Shading Correction (SC)

Lenses tend to produce images whose brightness is significantly attenuated near the edges. There are also other factors causing color plane nonuniformity in images captured by image sensors. The cumulative result of all these factors is known as image shading. The MT9N001 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:

$$Pcorrected(row, col) = Psensor(row,col) * f(row,col) \quad \text{(EQ 5)}$$

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.

One-Time Programmable (OTP) Memory

The MT9N001 has a two-byte OTP memory that can be utilized during module manufacturing to store specific information about the module. This feature provides system integrators and module manufacturers the ability to label and distinguish various module types based on lens, IR-cut filter, or other properties.

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 8.5V ±3%. Instantaneous VPP cannot exceed 9V at any time. 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 is needed 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 OTP memory 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 OTP memory, 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 OTP memory 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.

The steps below describe the process to program and verify the programmed data in the OTP memory:

1. Apply power to all the power rails of the sensor (VDD, VDD_IO, VAA, VAA_PIX, VDD_PLL, and VDD_TX0).
   1a. Set VAA to 3.1V during OTP memory programming phase.
   1b. VPP needs to be floated during this phase.
   1c. Other supplies at nominal.
2. Provide 24 MHz EXTCLK clock input. The PLL settings are discussed at the end of the document.
3. Perform the proper reset sequence to the sensor.
4. Place the sensor in soft standby (sensor default state upon power-up) or ensure the streaming is turned OFF when the part is in active mode.
5. VPP ramps to 8.5V in preparation to program. Power supply (VPP) slew rate should be slower than 1V/µs.
6. Set R0x3052 to the value 0x045C.
7. Set R0x3054 to the value 0XEA99.
8. Write the 16-bit word data by programming R0x304C.

9. Initiate the OTP memory programming process by setting R0x304A[0] to the value 0x0001.
10. Check R0x304A [2] = 1, until bit is set to "1" to check for program completion.
11. Repeat steps 9 and 10 two more times.
12. Remove high voltage and float VPP pin.
13. Power down the sensor.
14. Apply nominal power to all the power rails of the sensor VDD, VDD_IO, VAA, VAA_PIX and VDD_PLL). VPP must be floated.
15. Set EXTCLK to normal or customer-defined operating frequency.
16. Perform the proper reset sequence to the sensor.
17. Initiate the OTP memory reading process by setting R0x304A[4] to the value 0x0010.
18. Poll the register bit R0x304A[6] until bit set to "1" to check for read completion.
19. Read the 16-bit word data from the R0x304E.

Figure 21: Sequence for Programming the MT9N001

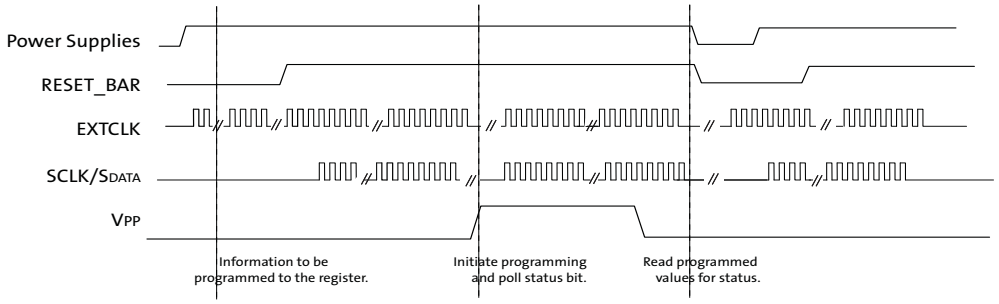


Image Acquisition Modes

The MT9N001 supports two image acquisition modes:

1. Electronic rolling shutter (ERS) mode
   This is the normal mode of operation. When the MT9N001 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 MT9N001 switches cleanly from the old integration time to the new while only generating frames with uniform integration. See "Changes to Integration Time" on page 26.
2. Global reset mode
   This mode can be used to acquire a single image at the current resolution. In this mode, the end point of the pixel integration time is controlled by an external electromechanical shutter, and the MT9N001 provides control signals to interface to that shutter. The operation of this mode is described in detail in "Global Reset" on page 105.

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

Window Control

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

Pixel Border

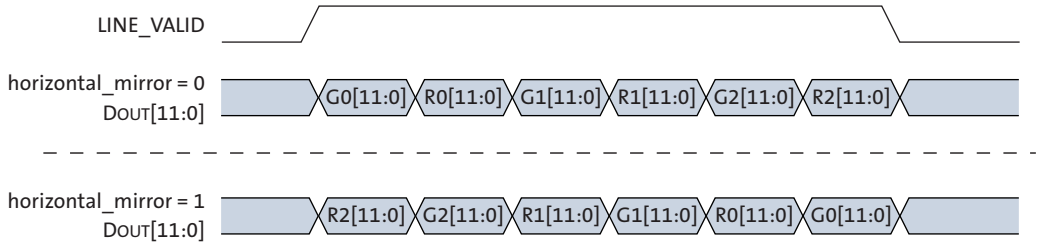
The default settings of the sensor provide a 3488H x 2616V image. A border of up to 8 pixels (4 in binning) on each edge can be enabled by reprogramming the x_addr_start, y_addr_start, x_addr_end, y_addr_end, x_output_size, and y_output_size registers accordingly.

Readout Modes

Horizontal Mirror

When the horizontal_mirror bit is set in the image_orientation register, the order of pixel readout within a row is reversed, so that readout starts from x_addr_end and ends at x_addr_start. Figure 22 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.

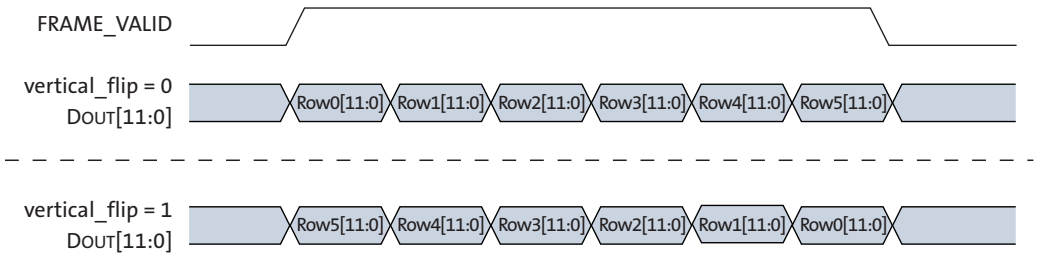
Figure 22: Effect of horizontal_mirror on Readout Order



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 23 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.

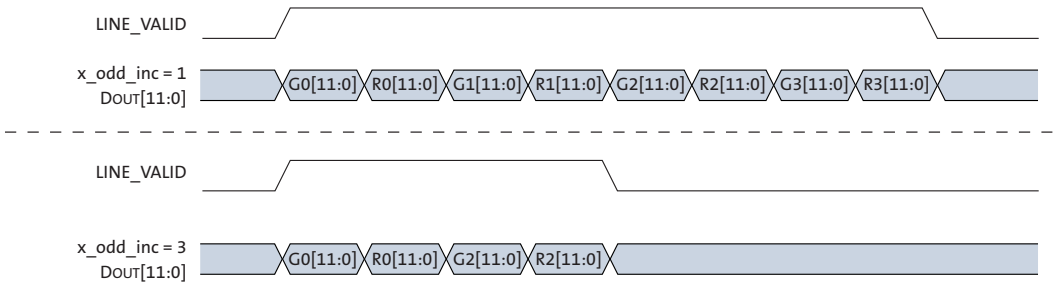
Figure 23: Effect of vertical_flip on Readout Order



Subsampling

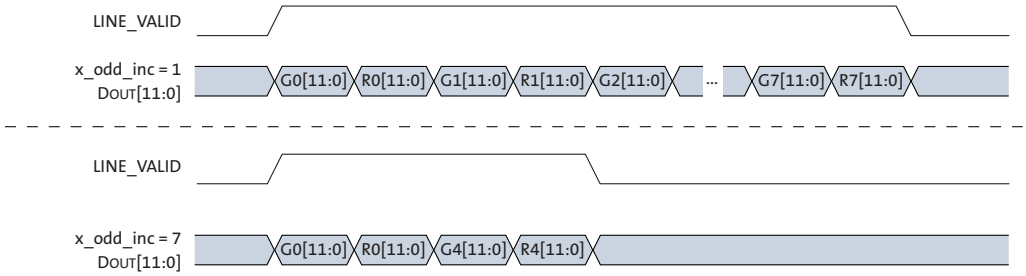
The MT9N001 supports subsampling. Subsampling reduces the amount of data processed by the analog signal chain in the MT9N001, thereby allowing the frame rate to be increased. Subsampling is enabled by setting x_odd_inc and/or y_odd_inc. Values of 1, 3, and 7 horizontal and 1, 3, 7, 15, 31, and 63 vertical 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 MT9N001. Figure 24 shows a sequence of 8 columns being read out with x_odd_inc = 3 and y_odd_inc = 1.

Figure 24: 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 MT9N001. Figure 25 shows a sequence of 16 columns being read out with x_odd_inc = 7 and y_odd_inc = 1.

Figure 25: 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 26 on page 90, Figure 27 on page 90, and Figure 28 on page 91.

Figure 26: Pixel Readout (No Subsampling)

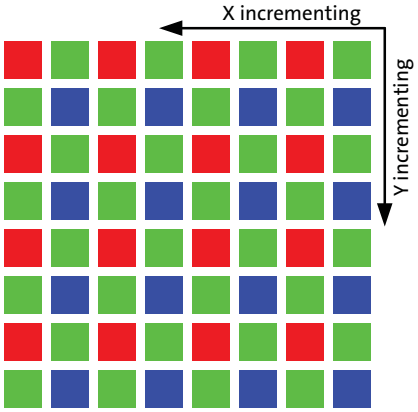


Figure 27: Pixel Readout (x_odd_inc = 3, y_odd_inc = 3)

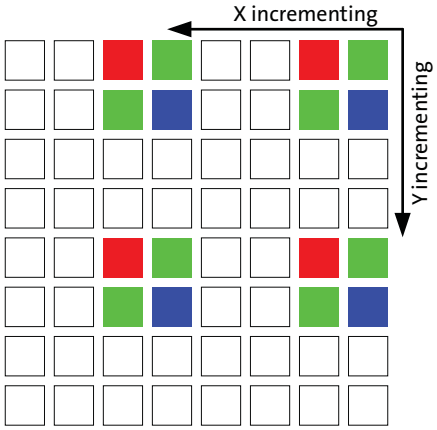


Figure 28: Pixel Readout (x_odd_inc = 7, y_odd_inc = 7)

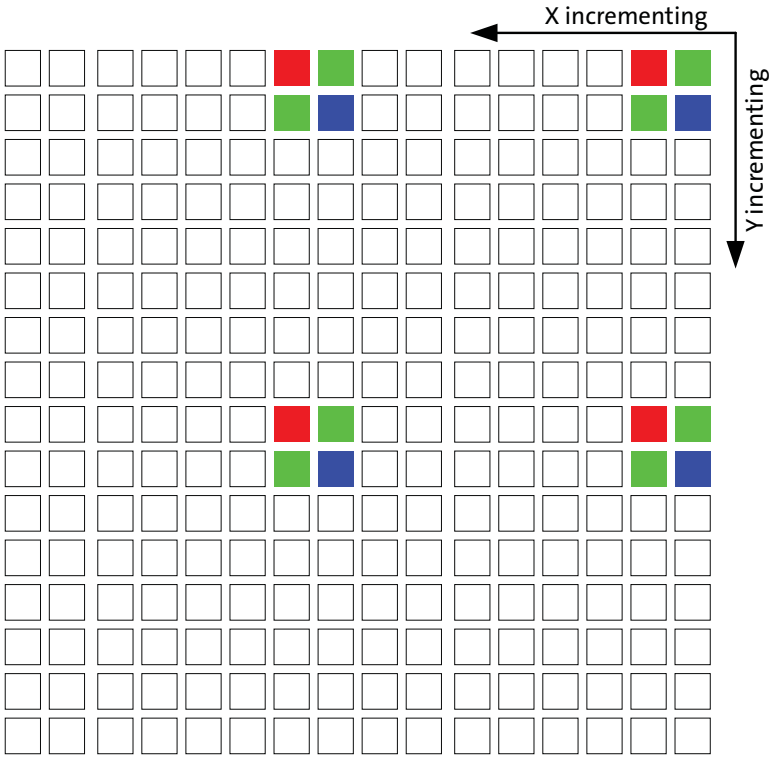


Figure 29: Pixel Readout (x_odd_inc = 7, y_odd_inc = 15)

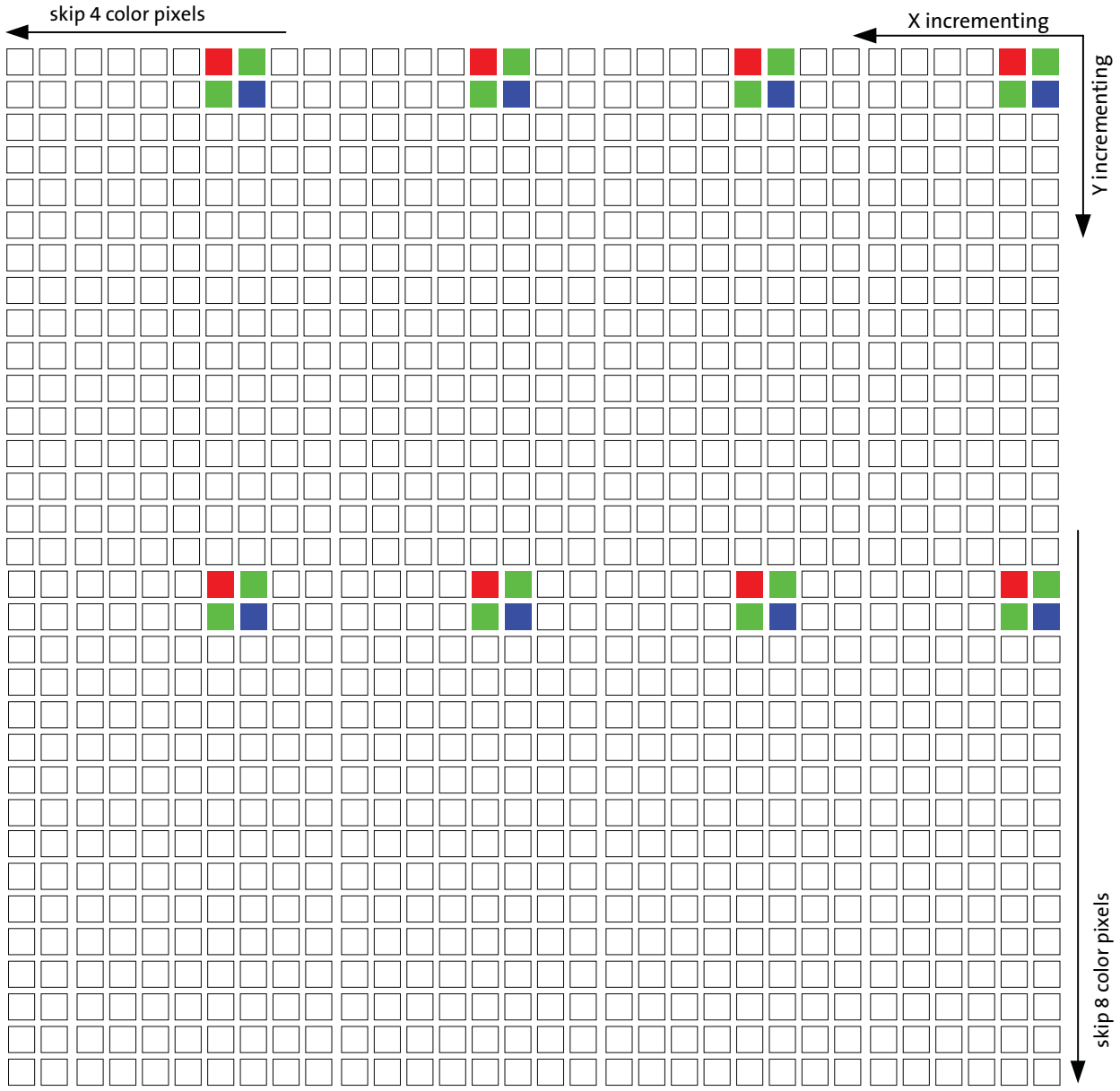


Figure 30: Pixel Readout (x_odd_inc = 7, y_odd_inc = 31)

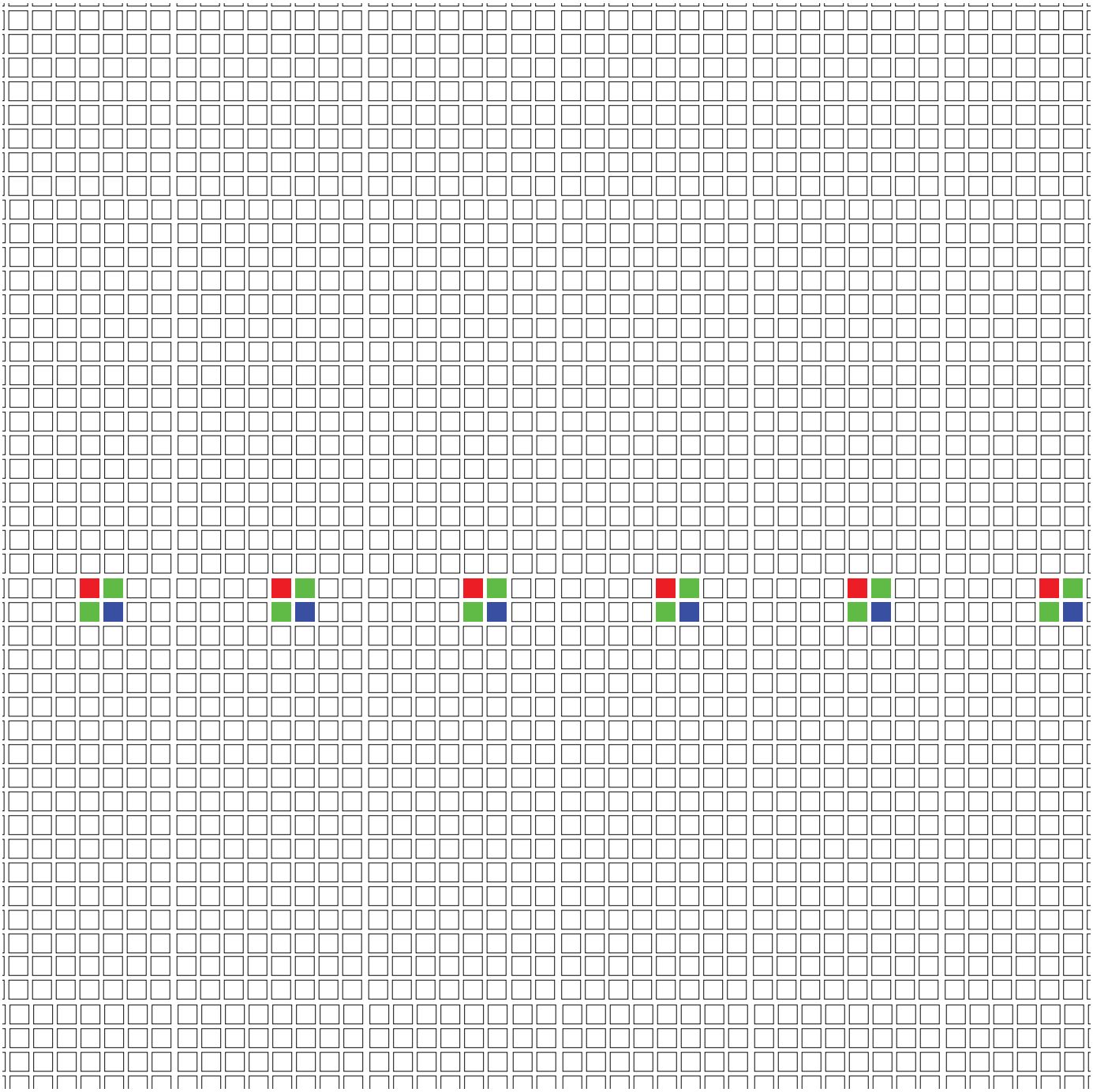
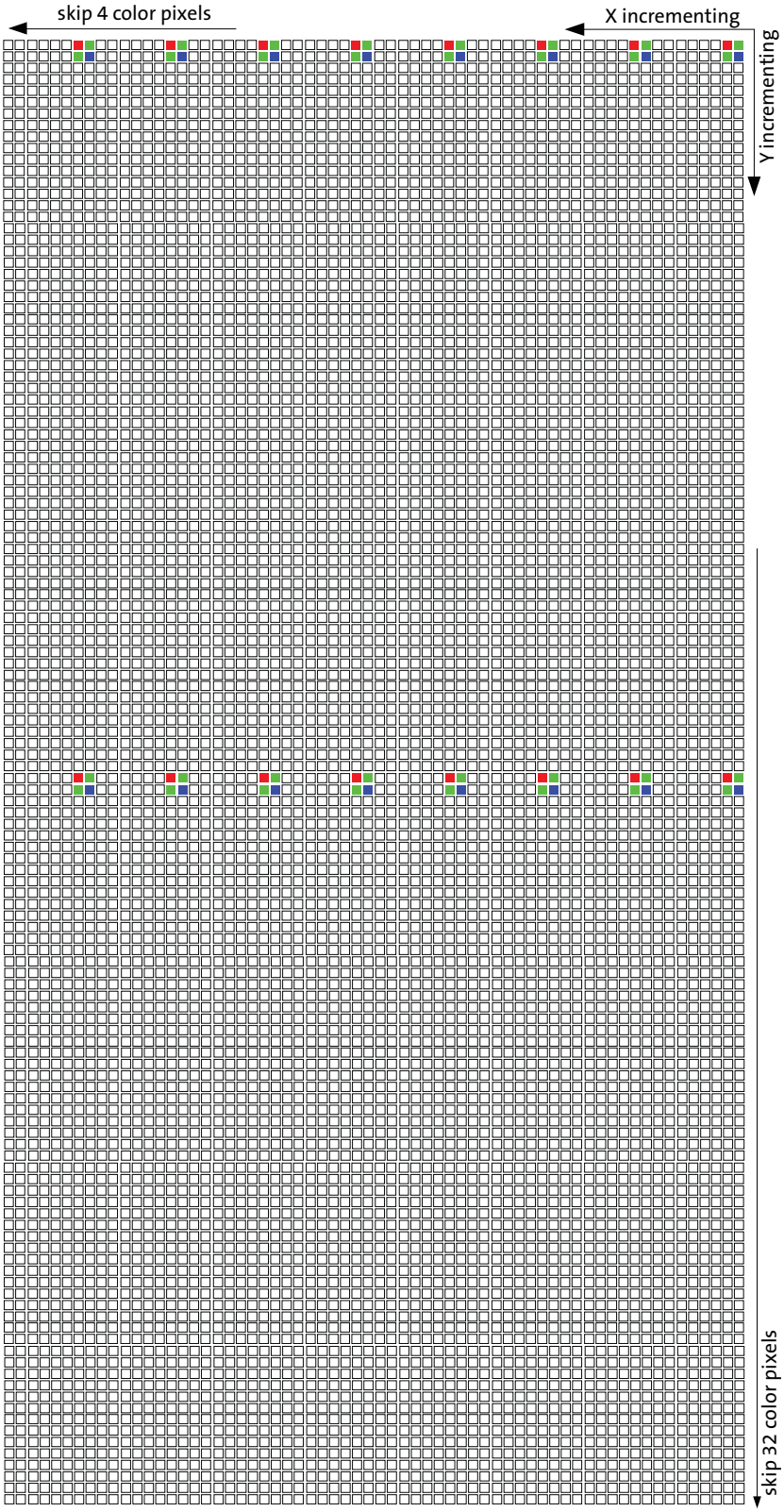


Figure 31: Pixel Readout (x_odd_inc = 7, y_odd_inc = 63)



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

- for 2X skip, x_addr_start is multiple of 2 * 4 = 8
- for 4X skip, x_addr_start is multiple of 4 * 4 = 16

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 (EQ 6)

Example:

The sensor is set up to give out a full resolution 3488 x 2616 image:

[full resolution starting address with (8,8)]

```
REG=0x0104, 1          //GROUPED_PARAMETER_HOLD
REG=0x0382, 1          //X_ODD_INC
REG=0x0386, 1          //Y_ODD_INC
REG=0x0344, 8          //X_ADDR_START
REG=0x0346, 8          //Y_ADDR_START
REG=0x0348, 3495       //X_ADDR_END
REG=0x034A, 2623       //Y_ADDR_END
REG=0x034C, 3488       //X_OUTPUT_SIZE
REG=0x034E, 2616       //Y_OUTPUT_SIZE
REG=0x0104, 0          //GROUPED_PARAMETER_HOLD
```

To halve the resolution in each direction (1744 x 1308), the registers need to be reprogrammed as follows:

[2 x 2 skipping starting address with (8,8)]

```
REG=0x0104, 1          //GROUPED_PARAMETER_HOLD
REG=0x0382, 3          //X_ODD_INC
REG=0x0386, 3          //Y_ODD_INC
REG=0x0344, 8          //X_ADDR_START
REG=0x0346, 8          //Y_ADDR_START
REG=0x0348, 3493       //X_ADDR_END
REG=0x034A, 2621       //Y_ADDR_END
REG=0x034C, 1744       //X_OUTPUT_SIZE
REG=0x034E, 1308       //Y_OUTPUT_SIZE
REG=0x0104, 0          //GROUPED_PARAMETER_HOLD
```

To quarter the resolution in each direction (872 x 654) the registers need to be reprogrammed as follows:

[4 x 4 skipping starting address with (16,16)]

```
REG=0x0104, 1                   //GROUPED_PARAMETER_HOLD
REG=0x0382, 7                   //X_ODD_INC
REG=0x0386, 7                   //Y_ODD_INC
REG=0x0344, 16                  //X_ADDR_START
REG=0x0346, 16                  //Y_ADDR_START
REG=0x0348, 3497                //X_ADDR_END
REG=0x034A, 2625                //Y_ADDR_END
REG=0x034C, 872                 //X_OUTPUT_SIZE
REG=0x034E, 654                 //Y_OUTPUT_SIZE
REG=0x0104, 0                   //GROUPED_PARAMETER_HOLD
```

Table 21 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 21).

Table 21: Row Address Sequencing During Subsampling

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

Binning

The MT9N001 supports 2 x 1 and 2 x 2 analog binning (column binning, also called x-binning, and row/column binning, also called xy-binning). Binning has many of the same characteristics as subsampling, but because it gathers image data from all pixels in the active window (rather than a subset of them), it achieves superior image quality and avoids the aliasing artifacts that can be a characteristic side effect of subsampling.

Binning is enabled by selecting the appropriate subsampling settings (odd_inc = 3 and y_odd_inc = 1 for x-binning, x_odd_inc = 3 and y_odd_inc = 3 for xy-binning) and setting the appropriate binning bit in read_mode (R0x3040–1). As with subsampling, x_addr_end and y_addr_end may require adjustment when binning is enabled. It is the first of the two columns/rows binned together that should be the end column/row in binning, so the requirements to the end address are exactly the same as in non-binning subsampling mode. The effect of the different subsampling settings is shown in Figure 32 and Figure 33 on page 98.

Binning can also be enabled when the 4X subsampling mode is enabled (x_odd_inc = 7 and y_odd_inc = 1 for x-binning, x_odd_inc = 7 and y_odd_inc = 7 for xy-binning). In this mode, however, not all pixels will be used so this is not a 4X binning implementation. An implementation providing a combination of skip2 and bin2 is used to achieve 4X subsampling with better image quality. The effect of this subsampling mode is shown in Figure 34 on page 98.

Figure 32: Pixel Readout (x_odd_inc = 3, y_odd_inc = 1, x_bin = 1)

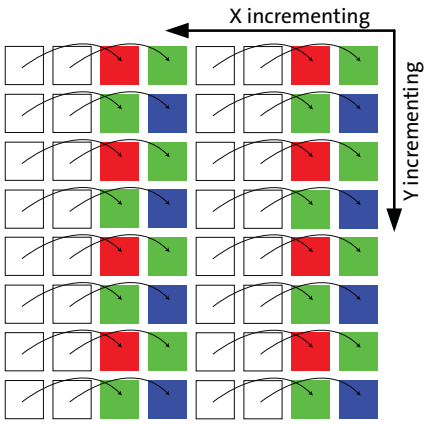


Figure 33: Pixel Readout (x_odd_inc = 3, y_odd_inc = 3, xy_bin = 1)

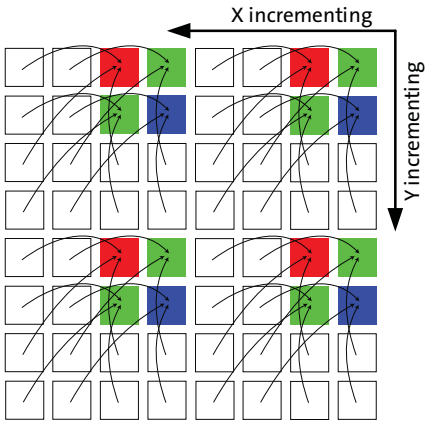
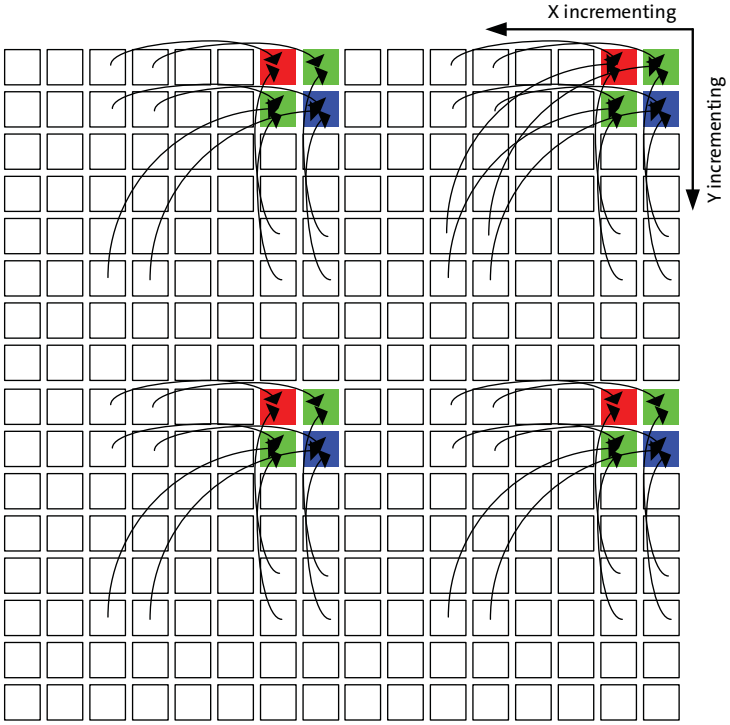


Figure 34: Pixel Readout (x_odd_inc = 7, y_odd_inc = 7, xy_bin = 1)



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

For a given column *n*, there is only one other column, n_bin, that it can be binned with, because of physical limitations in the column readout circuitry. The possible address sequences are shown in Table 22.

Summing can also be implemented in binning mode where pixel values are added together as opposed to being averaged in regular binning mode. Vertical summing can be enabled with the MT9N001. The Aptina-recommended combination is to enable Y summing using register 0x3040[13], and X binning using register 0x3040[11].

Table 22: Column Address Sequencing During Binning

odd_inc = 1—Normal	odd_inc = 3, 2X Bin
x_addr_start = 0	x_addr_start = 0
0	0/2
1	1/3
2	
3	
4	4/6
5	5/7
6	
7	
8	8/10
9	9/11
10	
11	
12	12/14
13	13/15
14	
15	

There are no physical limitations on what can be binned together in the row direction. A given row *n* will always be binned with row n + 2 in 2X subsampling mode and with row n + 4 in 4X subsampling mode. Therefore, which rows get binned together depends upon the alignment of y_addr_start. The possible sequences are shown in Table 23.

Table 23: Row Address Sequencing During Binning

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

Programming Restrictions when Binning

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

As a result, when xy-binning is enabled, some of the programming limits declared in the parameter limit registers are no longer valid. In addition, the default values for some of the manufacturer-specific registers need to be reprogrammed. See section "Minimum Frame Time" on page 101, section "Minimum Row Time" on page 101, and section "Fine Integration Time Limits" on page 102.

Table 24: Readout Modes

Readout Modes	x_odd_inc, y_odd_inc	xy_bin
2X skip	3	0
2X bin	3	1
4X skip	7	0
2X skip and 2X bin	7	1

Frame Rate Control

The formulas for calculating the frame rate of the MT9N001 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 7:

$$minimum\ line\_length\_pck = \left(\frac{x\_addr\_end - x\_addr\_start + 1}{subsampling\ factor} + min\_line\_blanking\_pck\right) \quad \text{(EQ 7)}$$

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 101.

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 is shown in Equation 8:

$$minimum\ frame\_length\_lines = \left(\frac{y\_addr\_end - y\_addr\_start + 1}{subsampling\ factor} + min\_frame\_blanking\_lines\right) \quad \text{(EQ 8)}$$

The frame rate can be calculated from these variables and the pixel clock speed as shown in Equation 9:

$$frame\ rate = \frac{vt\_pixel\_clock\_mhz \times 1 \times 10^6}{line\_length\_pck\_x\ frame\_length\_lines} \quad \text{(EQ 9)}$$

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 25.

Table 25: Minimum Row Time and Blanking Numbers

Register	No Row Binning			Row Binning		
row_speed[2:0]	1	2	4	1	2	4
min_line_blanking_pck	0x06E4	0x0442	0x02F1	0x0C48	0x06CC	0x040E
min_line_length_pck	0x0910	0x0588	0x0438	0x01200	0x0900	0x0550

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:

1. line_length_pck≥ min_line_length_pck in Table 25.
2. line_length_pck ≥ (x_addr_end - x_addr_start + x_odd_inc)/((1+x_odd_inc)/2) + min_line_blanking_pck in Table 25.
3. The row time must allow the FIFO to output all data during each row 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 26: Minimum Frame Time and Blanking Numbers

Register	
min_frame_blanking_lines	0x008F
min_frame_length_lines	0x0091

Integration Time

The integration (exposure) time of the MT9N001 is controlled by the fine_integration_time and coarse_integration_time registers.

The limits for the fine integration time are defined by:

$$fine\_integration\_time\_min \leq fine\_integration\_time \leq (line\_length\_pck - fine\_integration\_time\_max\_margin) \quad \text{(EQ 10)}$$

The limits for the coarse integration time are defined by:

$$coarse\_integration\_time\_min \leq coarse\_integration\_time \quad \text{(EQ 11)}$$

The actual integration time is given by:

$$integration\_time = \frac{((coarse\_integration\_time * line\_length\_pck) + fine\_integration\_time)}{(vt\_pix\_clk\_freq\_mhz * 10^6)} \quad \text{(EQ 12)}$$

It is required that:

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

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

Fine Integration Time Limits

The limits for the fine_integration_time can be found from fine_integration_time_min and fine_integration_time_max_margin. Values for different mode combinations are shown in Table 27.

Table 27: fine_integration_time Limits

Register	No Row Binning			Row Binning		
row_speed[2:0]	1	2	4	1	2	4
fine_integration_time_min	0x056A	0x0C26	0x00C2	0x0B1A	0x059E	0x0178
fine_integration_time_max_margin	0x03A6	0x01C2	0x0182	0x06E6	0x0362	0x0308

fine_correction

For the fine_integration_time limits, the fine_correction constant will change with the pixel clock speed and binning mode. These values are shown in Table 28.

Table 28: fine_correction Values

Register	No Row Binning			Row Binning		
row_speed[2:0]	1	2	4	1	2	4
fine_correction	0x0100	0x007A	0x0037	0x0238	0x0116	0x0085

Low Power Mode

The MT9N001 sensor supports a low power mode, which can be entered by programming register bit read_mode[9]. Setting this bit will do the following:

- Double the value of pc_speed[2:0] internally. This means halving the internal pixel clock frequency.
- Lower currents in the analog domain. This can be done by setting a low power bit in the static control register. The current will be halved where appropriate in the analog domain.

Note that enabling the low power mode will not put the sensor in subsampling mode. This will have to be programmed separately as described earlier in this document. Low power is independent of the readout mode and can also be enabled in full resolution mode. Because the pixel clock speed is halved, the frame rates that can be achieved with low power mode are lower than in full power mode.

Because only internal pixel clock speeds of 1, 2, and 4 are supported, low power mode combined with pc_speed[2:0] = 4 is an illegal combination.

Any limitations related to changing the internal pixel clock speed will also apply to low power mode, because it automatically changes the pixel clock speed. Therefore, the limiter registers need to be reprogrammed to match the new internal pixel clock frequency.

Flash Control

The MT9N001 supports both xenon and LED flash through the FLASH output signal. The timing of the FLASH signal with the default settings is shown in Figure 35, and in Figure 36 and Figure 37 on page 104. 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 (write reset_register[9] = 1) before the enabling the flash or by forcing a restart (write reset_register[1] = 1) immediately after enabling the flash; the first bad frame will then be masked out, as shown in Figure 37 on page 104. Read-only bit flash[14] is set during frames that are correctly integrated; the state of this bit is shown in Figures 35, 36, and Figure 37.

Figure 35: Xenon Flash Enabled

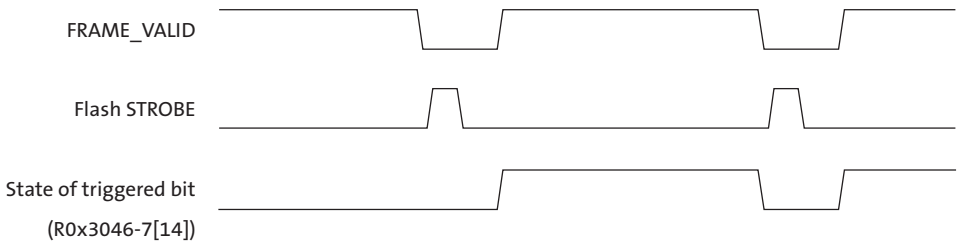
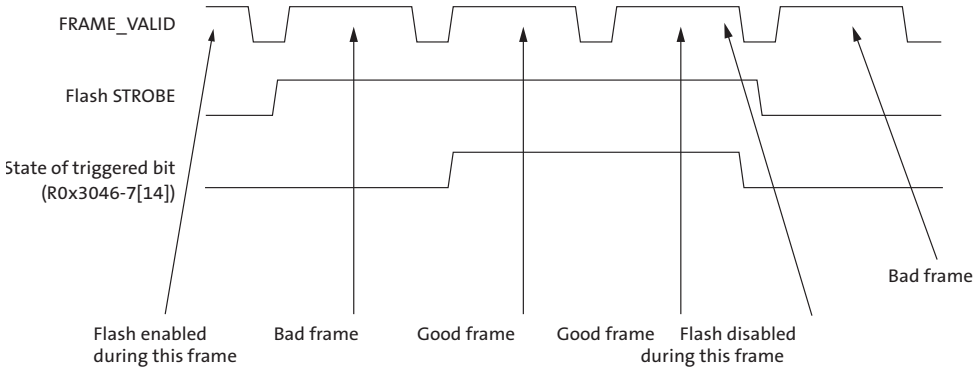
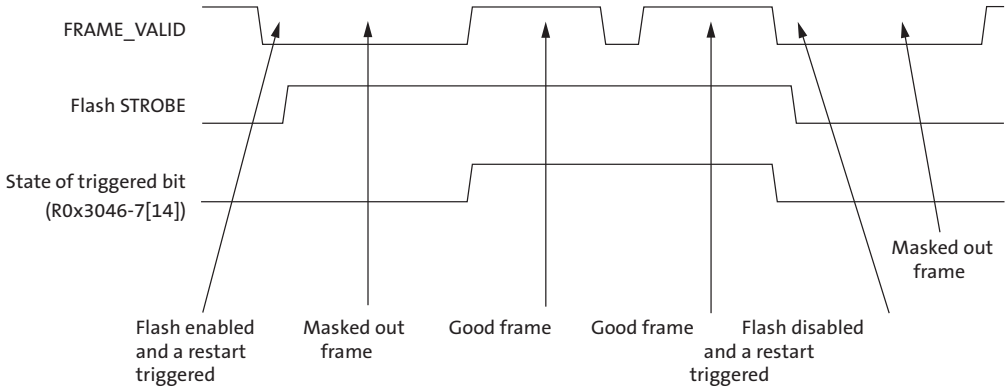


Figure 36: LED Flash Enabled



Notes:
1. Integration time = number of rows in a frame.
2. Bad frames will be masked during LED flash operation when mask bad frames bit field is set (R0x301A[9] = 1).
3. An option to invert the flash output signal through R0x3046[7] is also available.

Figure 37: LED Flash Enabled Following Forced Restart



Global Reset

Global reset mode allows the integration time of the MT9N001 to be controlled by an external electromechanical shutter. Global reset mode is generally used in conjunction with ERS mode. The ERS mode is used to provide viewfinder information, the sensor is switched into global reset mode to capture a single frame, and the sensor is then returned to ERS mode to restore viewfinder operation.

Global reset mode is designed for use in conjunction with the parallel pixel data interface. The SMIA specification does not define a global reset mode and only provides for operation in ERS mode. The MT9N001 does support the use of global reset mode in conjunction with the SMIA data path, but there are additional restrictions on its use.

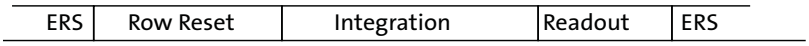
Overview of Global Reset Sequence

The basic elements of the global reset sequence are:

1. By default, the sensor operates in ERS mode and the SHUTTER output signal is LOW. The electromechanical shutter must be open to allow light to fall on the pixel array. Integration time is controlled by the coarse_integration_time and fine_integration_time registers.
2. A global reset sequence is triggered.
3. All of the rows of the pixel array are placed in reset.
4. All of the rows of the pixel array are taken out of reset simultaneously. All rows start to integrate incident light. The electromechanical shutter may be open or closed at this time.
5. If the electromechanical shutter has been closed, it is opened.
6. After the desired integration time (controlled internally or externally to the MT9N001), the electromechanical shutter is closed.
7. A single output frame is generated by the sensor with the usual LV, FV, PIXCLK, and DOUT timing. As soon as the output frame has completed (FV de-asserts), the electromechanical shutter may be opened again.
8. The sensor automatically resumes operation in ERS mode.

This sequence is shown in Figure 38. The following sections expand to show how the timing of this sequence is controlled.

Figure 38: Overview of Global Reset Sequence



Entering and Leaving the Global Reset Sequence

A global reset sequence can be triggered by a register write to global_seq_trigger[0] (global trigger, to transition this bit from a 0 to a 1) or by a rising edge on a suitably-configured GPI input (see "Trigger Control" on page 80).

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

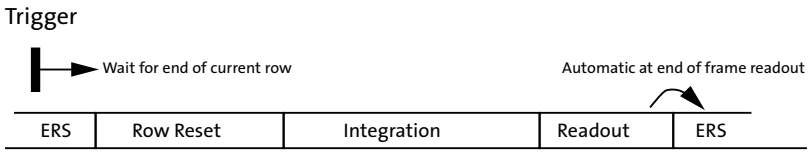
The global reset sequence completes with a frame readout. At the end of this readout phase, the sensor automatically resumes operation in ERS mode. The first frame integrated with ERS will be generated after a delay of approximately:

*((13 + coarse_integration_time) * line_length_pck).*

This sequence is shown in Figure 39.

While operating in ERS mode, double-buffered registers ("Double-Buffered Registers" on page 25) are updated at the start of each frame in the usual way. During the global reset sequence, double-buffered registers are updated just before the start of the readout phase.

Figure 39: Entering and Leaving a Global Reset Sequence



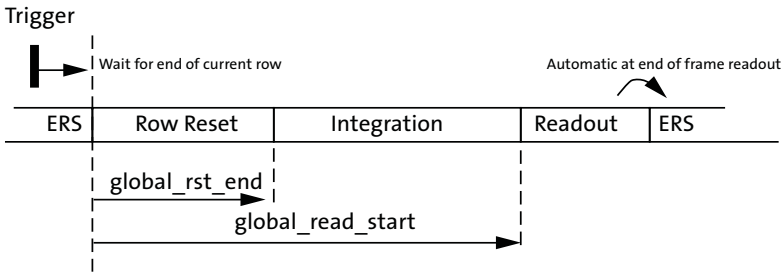
Programmable Settings

The registers global_rst_end and global_read_start allow the duration of the row reset phase and the integration phase to be controlled, as shown in Figure 40. The duration of the readout phase is determined by the active image size.

The recommended setting for global_rst_end is 0x074C. This allows sufficient time for all rows of the pixel array to be set to the correct reset voltage level. The row reset phase takes a finite amount of time due to the capacitance of the pixel array and the capability of the internal voltage booster circuit that is used to generate the reset voltage level.

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

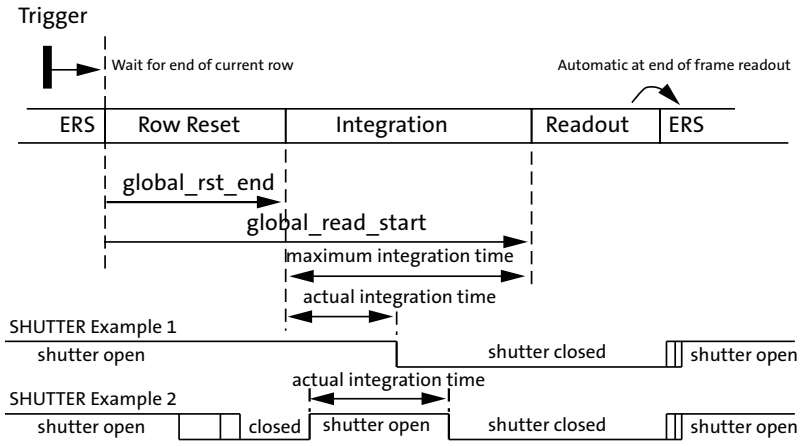
Figure 40: Controlling the Reset and Integration Phases of the Global Reset Sequence



Control of the Electromechanical Shutter

Figure 41 shows two different ways in which a shutter can be controlled during the global reset sequence. In both cases, the maximum integration time is set by the difference between global_read_start and global_rst_end. In shutter example 1, the shutter is open during the initial ERS sequence and during the row reset phase. The shutter closes during the integration phase. The pixel array is integrating incident light from the start of the integration phase to the point at which the shutter closes. Finally, the shutter opens again after the end of the readout phase. In shutter example 2, the shutter is open during the initial ERS sequence and closes sometime during the row reset phase. The shutter both opens and closes during the integration phase. The pixel array is integrating incident light for the part of the integration phase during which the shutter is open. As for the previous example, the shutter opens again after the end of the readout phase.

Figure 41: Control of the Electromechanical Shutter



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

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

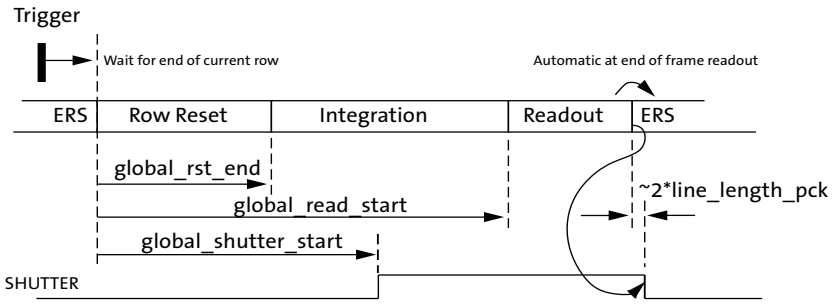
After FV de-asserts to signal the completion of the readout phase, there is a time delay of approximately *10 * line_length_pck* before the sensor starts to integrate light-sensitive rows for the next ERS frame. It is essential that the shutter be opened at some point in this time window; otherwise,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

The MT9N001 provides a SHUTTER output signal to control (or help the host system control) the electromechanical shutter. The timing of the SHUTTER output is shown in Figure 42 on page 108. SHUTTER is de-asserted by default. The point at which it asserts is controlled by the programming of global_shutter_start. At the end of the global reset readout phase, SHUTTER de-asserts approximately *2 * line_length_pck* after the de-assertion of FV.

This programming restriction must be met for correct operation:

- global_read_start > global_shutter_start.

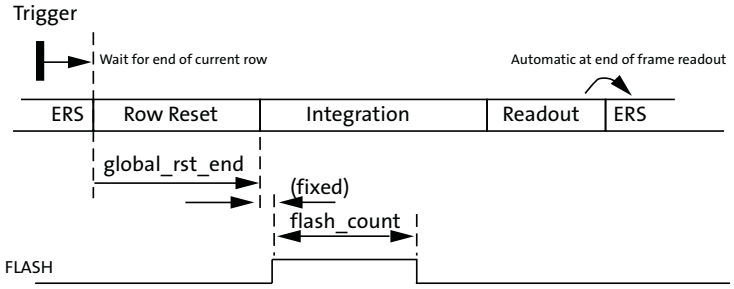
Figure 42: Controlling the SHUTTER Output



Using FLASH with Global Reset

If global_seq_trigger[2] = 1 (global flash enabled) when a global reset sequence is triggered, the FLASH output signal will be pulsed during the integration phase of the global reset sequence. The FLASH output will assert a fixed number of cycles after the start of the integration phase and will remain asserted for a time that is controlled by the value of the flash_count register, as shown in Figure 43.

Figure 43: Using FLASH with Global Reset



External Control of Integration Time

If global_seq_trigger[1] = 1 (global bulb enabled) when a global reset sequence is triggered, the end of the integration phase is controlled by the level of trigger (global_seq_trigger[0] or the associated GPI input). This allows the integration time to be controlled directly by an input to the sensor and allows integration times that are longer than can be accommodated by the programming limits 174msec by the global_read_start register.

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

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

The operation of this mode is shown in Figure 44 on page 109. The figure shows the global reset sequence being triggered by the GPI2 input, but it could be triggered by any of the GPI inputs or by the setting and subsequence clearing of the global_seq_trigger[0] under software control.

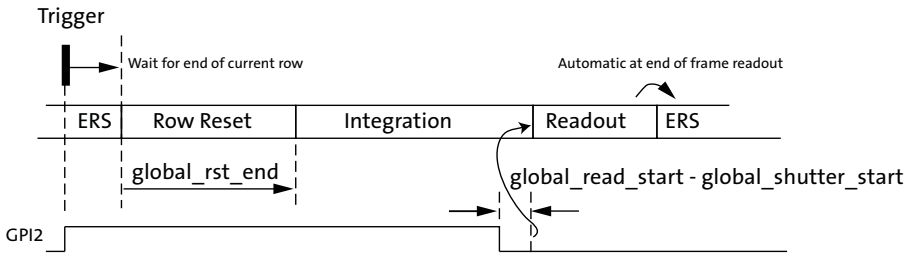
The integration time of the GRR sequence is defined as:

$$Integration = \frac{(global\_read\_start - global\_rst\_end)\ x\ 512}{vt\_pix\_clk\_freq\_mhz} \quad \text{(EQ 14)}$$

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

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

Figure 44: Global Reset Bulb



Retriggering the Global Reset Sequence

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

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

Using Global Reset with SMIA Data Path

When a global reset sequence is triggered, it usually results in the frame in progress being truncated (at the end of the current output line). The SMIA data path limiter function (see Figure 47 on page 115) attempts to extend (pad) all frames to the programmed value of y_output_size. If this padding is still in progress when the global reset readout phase starts, the SMIA data path will not detect the start of the frame correctly. Therefore, to use global reset with the serial data path, this timing scenario must be avoided. One possible way of doing this would be to synchronize (under software control) the assertion of trigger to an end-of-frame marker on the serial data stream.

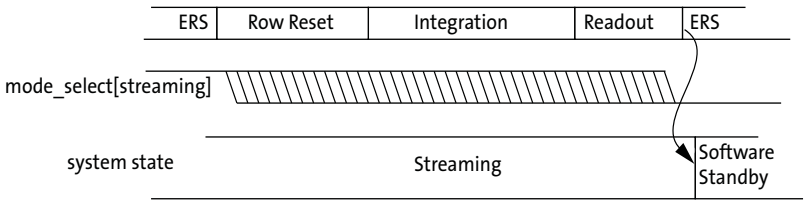
At the end of the readout phase of the global reset sequence, the sensor automatically resumes operation in ERS mode.

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

Global Reset and Soft Standby

If the mode_select[stream] bit is cleared while a global reset sequence is in progress, the MT9N001 will remain in streaming state until the global reset sequence (including frame readout) has completed, as shown in Figure 45.

Figure 45: Entering Soft Standby During a Global Reset Sequence



Analog Gain

The MT9N001 provides two mechanisms for setting the analog gain. The first uses the SMIA gain model. The second uses the traditional Aptina Imaging gain model. The following sections describe both models, the mapping between the models, and the operation of the per-color and global gain control.

Using Per-color or Global Gain Control

The read-only analogue_gain_capability register returns a value of "1," indicating that the MT9N001 provides per-color gain control. However, the MT9N001 also provides the option of global gain control. Per-color and global gain control can be used interchangeably. A WRITE to a global gain register is aliased as a WRITE of the same data to the four associated color-dependent gain registers. A READ from a global gain register is aliased to a READ of the associated greenB/greenR gain register.

The read/write gain_mode register required by SMIA has no defined function. In the MT9N001, this register has no side effects on the operation of the gain; per-color and global gain control can be used interchangeably regardless of the state of the gain_mode register.

SMIA Gain Model

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

- analogue_gain_code_global
- analogue_gain_code_greenR
- analogue_gain_code_red
- analogue_gain_code_blue
- analogue_gain_code_greenB

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

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

Aptina Imaging Gain Model

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

- global_gain
- greenR_gain
- red_gain
- blue_gain
- greenB_gain

This provides a 7-bit, a 3X, and a 2X gain state. As a result, the step size varies depending upon if the 2X and 3X gain stages are enabled. The analog gain is given by:

(EQ 16)

$$gain = (2*<color>\_gain[9]+<color>\_gain[8]-<color>\_gain[9]*<color>\_gain[8]+1)\ x\ (<color>\_gain[7]+1)x\frac{<color>\_gain[6:0]}{32}$$

As a result of the 2X and 3X gain stages, many of the possible gain settings can be achieved in many different ways. The recommended gain sequence is shown in Table 29.

Table 29: Recommended Gain Stages

Desired Gain	Recommended Gain Register Setting
1–1.96875	0x1020–0x103F
2–7.938	0x10A0–0x10FF
8–15.875	0x11C0–0x11FF
15.9735–23.8125	0x12D5–0x12FF

Gain Code Mapping

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

When the SMIA gain model is in use and values have been written to the analogue_gain_code_<color> registers, the associated value in the Aptina Imaging gain model can be read from the associated <color>_gain register. In cases where there is more than one possible mapping, the 2X and 3X gain stages are enabled to provide the mapping with the lowest noise.

When the Aptina Imaging gain model is in use and values have been written to the gain_<color> undefined. The reason for this is that many of the gain codes available in the Aptina Imaging gain model have no corresponding value in the SMIA gain model.

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

Sensor Core Digital Data Path

Test Patterns

The MT9N001 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 30.

Table 30: 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-bit value)
257	Walking 1s (10-bit value)
258	Walking 1s (8-bit value)

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 MT9N001 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

Test Cursors

The MT9N001 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 R0x31E8–R0x31EE. Both even and odd cursor positions and widths are supported.

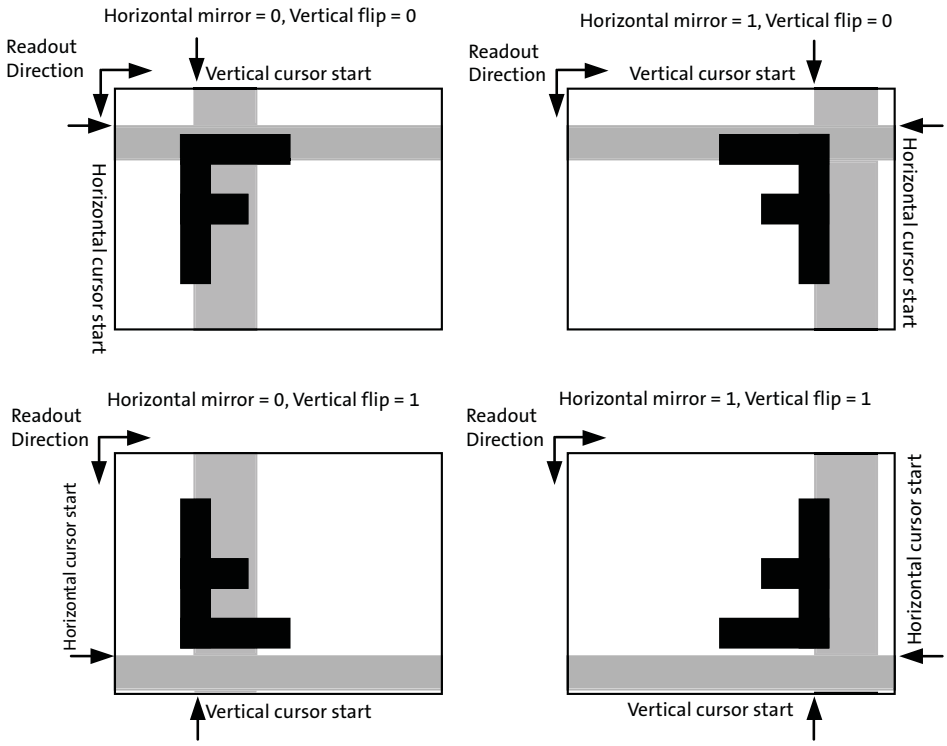
Each cursor can be inhibited by setting its width to “0.” The programmed cursor position corresponds to the x and y addresses of the pixel array. For example, setting horizontal_cursor_position to the same value as y_addr_start would result in a horizontal cursor being drawn starting on the first row 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 MT9N001 is shown in Figure 46 on page 114 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.

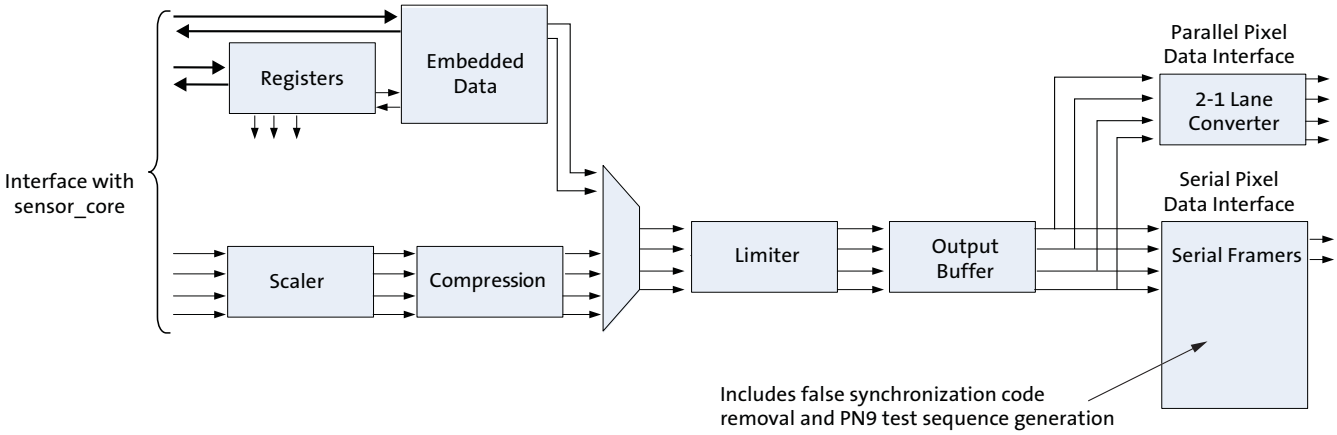
Figure 46: Test Cursor Behavior with image_orientation



Digital Data Path

The digital data path after the sensor core is shown in Figure 47.

Figure 47: Data Path



Embedded Data Format and Control

When the serial pixel data path is selected, the first two rows of the output image contain register values that are appropriate for the image. The 12-bit format places the data byte in bits [11:4] and sets bits [3:0] to a constant value of 0101. Some register values are dynamic and may change from frame to frame. Additional information on the format of the embedded data can be located in the SMIA functional specification.

Timing Specifications

Power-Up Sequence

The recommended power-up sequence for the MT9N001 is shown in Figure 48. The available power supplies (VDD_IO, VDD, VDD_TX0, VDD_PLL, VAA, VAA_PIX) can be turned on at the same time or have the separation specified below.

1. Turn on VDD_IO power supply.
2. After 0–500ms, turn on VDD and VDD_TX0 power supply.
3. After 0–500ms, turn on VDD_PLL and VAA/VAA_PIX power supplies.
4. After the last power supply is stable, enable EXTCLK.
5. Assert RESET_BAR for at least 1ms.
6. Wait 2400 EXTCLKs for internal initialization into software standby.
7. Configure PLL, output, and image settings to desired values.
8. Set mode_select = 1 (R0x0100).
9. Wait 1ms for the PLL to lock before streaming state is reached.

Figure 48: Power-Up Sequence

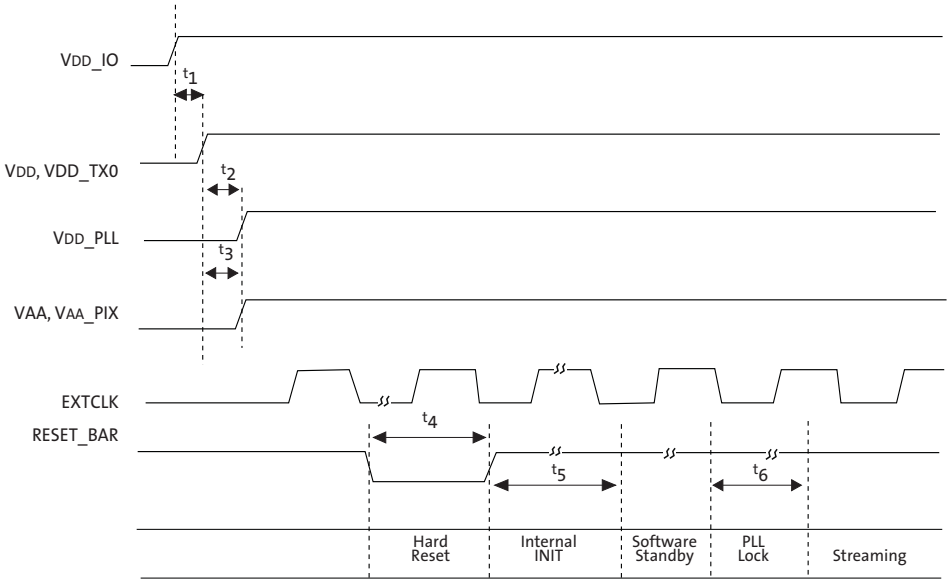


Table 31: Power-Up Sequence

Definition	Symbol	Min	Typ	Max	Unit
VDD_IO to VDD time	t_1	0	–	500	ms
VDD to VDD_PLL time	t_2	0	–	500	ms
VDD to VAA/VAA_PIX time	t_3	0	–	500	ms
Active hard reset	t_4	1	–	–	ms
Internal initialization	t_5	2400	–	–	EXTCLKs
PLL lock time	t_6	1	–	–	ms

Power-Down Sequence

The recommended power-down sequence for the MT9N001 is shown in Figure 49. The available power supplies (VDD_IO, VDD, VDD_TX0, VDD_PLL, VAA, VAA_PIX) can be turned off at the same time or have the separation specified below.

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

Figure 49: Power-Down Sequence

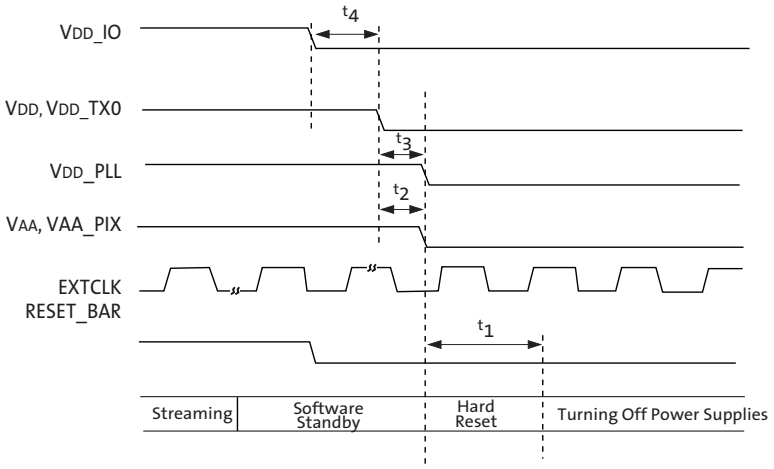


Table 32: Power-Down Sequence

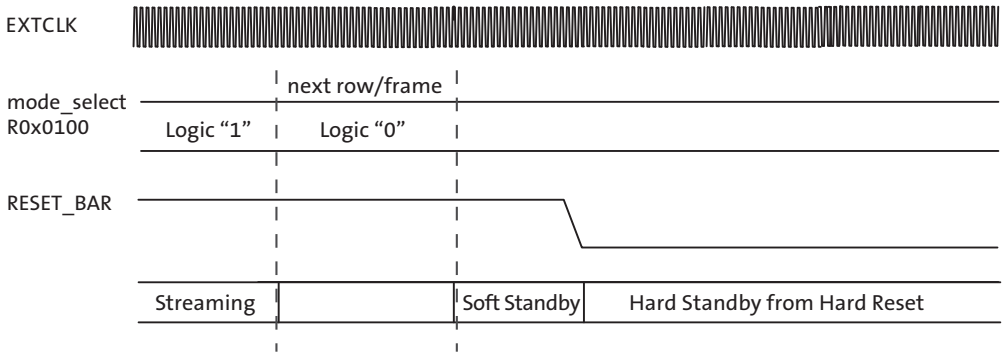
Definition	Symbol	Min	Typ	Max	Unit
Hard reset	t_1	1	–	–	ms
VDD/VAA/VAA_PIX to VDD time	t_2	0	–	500	ms
VDD_PLL to VDD time	t_3	0	–	500	ms
VDD to VDD_IO time	t_4	0	–	500	ms

Hard Standby and Hard Reset

The hard standby state is reached by the assertion of the RESET_BAR pad (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 50.

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

Figure 50: Hard Standby and Hard Reset



Soft Standby and Soft Reset

The MT9N001 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 51.

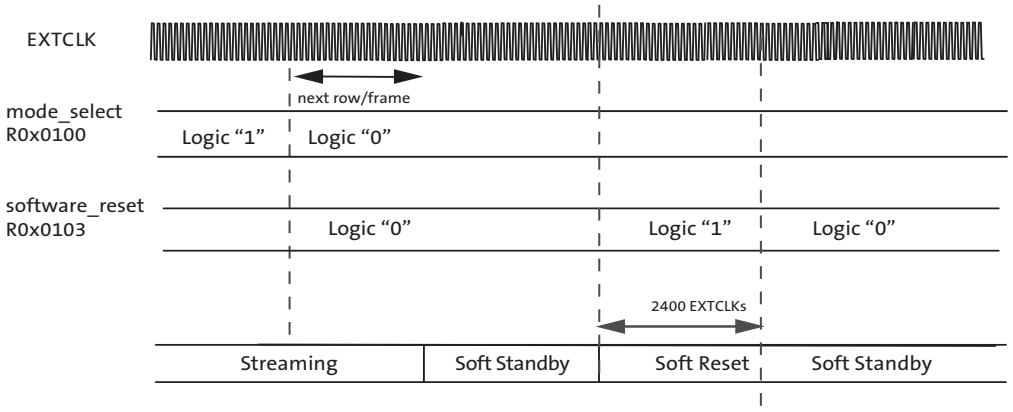
Soft Standby

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

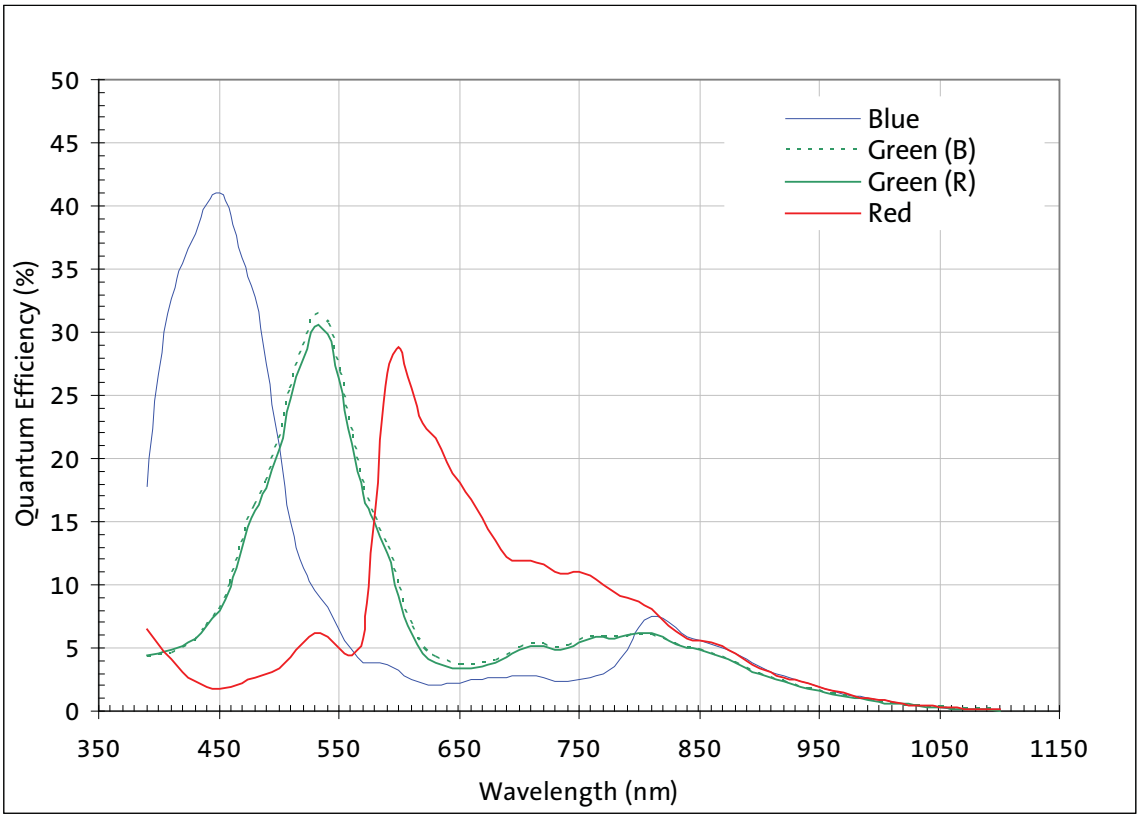
1. Follow the soft standby sequence listed above.
2. Set software_reset = 1 (R0x0103) to start the internal initialization sequence.
3. After 2400 EXTCLKs, the internal initialization sequence is completed and the current state returns to soft standby automatically. All registers, including software_reset,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

Figure 51: Soft Standby and Soft Reset



Spectral Characteristics

Figure 52: Quantum Efficiency



Electrical Characteristics

Table 33: DC Electrical Definitions and Characteristics

[f]EXTCLK = 24 MHz; VDD = 1.8V; VDD_IO = 1.8V; VAA = 2.8V; VAA_PIX = 2.8V; VDD_PLL = 2.8V; Output load = 68.5pF

Definition	Condition	Symbol	Min	Typ	Max	Unit
TJ = 25°C						
Core digital voltage		VDD	1.7	1.8	1.9	V
I/O digital voltage	Parallel pixel data interface	VDD_IO	1.7	1.8	1.9	V
Analog voltage		VAA	2.6	2.8	3.1	V
Pixel supply voltage		VAA_PIX	2.6	2.8	3.1	V
PLL supply voltage		VDD_PLL	2.4	2.8	3.1	V
Digital operating current	Streaming, full resolution		75	80	85	mA
I/O digital operating current	Streaming, full resolution		50	60	70	mA
Analog operating current	Streaming, full resolution		135	160	185	mA
Pixel supply current	Streaming, full resolution		2	4	5	mA
PLL supply current	Streaming, full resolution		14	16	18	mA
Hard standby (clock on)	Analog and pixel		100	125	150	µA
	Digital and PLL		50	85	120	µA
Soft standby (clock on)	Analog and pixel		100	125	150	µA
	Digital and PLL		50	85	120	µA
TJ = 60°C						
Core digital voltage		VDD	1.7	1.8	1.9	V
I/O digital voltage	Parallel pixel data interface	VDD_IO	1.7	1.8	1.9	V
Analog voltage		VAA	2.6	1.8	1.9	V
Pixel supply voltage		VAA_PIX	2.6	2.8	3.1	V
PLL supply voltage		VDD_PLL	2.4	2.8	3.1	V
Digital operating current	Streaming, full resolution		70	75	80	mA
I/O digital operating current	Streaming, full resolution		45	55	65	mA
Analog operating current	Streaming, full resolution		130	155	180	mA
Pixel supply current	Streaming, full resolution		2	4	5	mA
PLL supply current	Streaming, full resolution		14	16	18	mA
Hard standby (clock on)	Analog and pixel		90	100	110	µA
	Digital and PLL		85	120	155	µA
Soft standby (clock on)	Analog and pixel		90	100	110	µA
	Digital and PLL		85	120	155	µA

Table 34: I/O Parameters

^{f}EXTCLK = 24 MHz; V_{DD} = 1.8V; V_{DD_IO} = 1.8V; V_{AA} = 2.8V; V_{AA_PIX} = 2.8V; V_{DD_PLL} = 2.8V; Lighting conditions = 0 lux

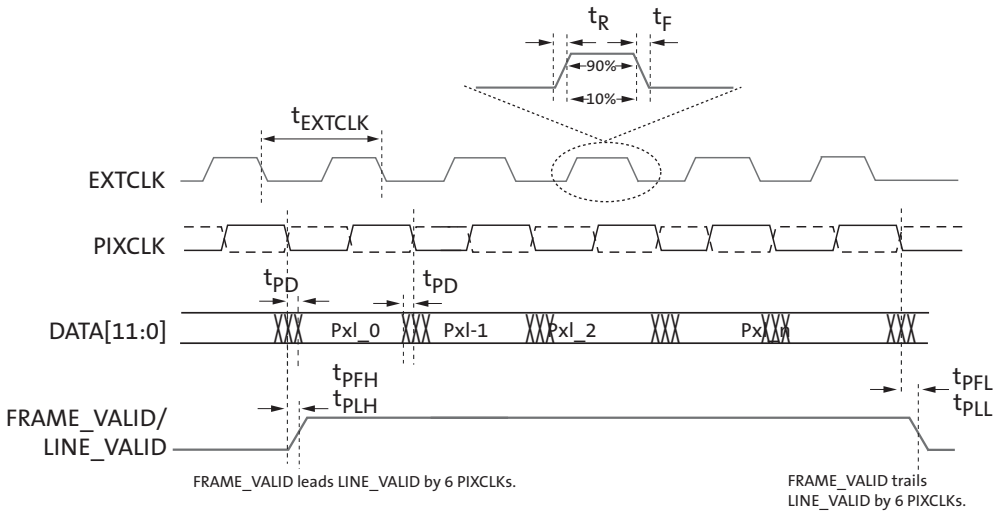
Symbols	Definition	Conditions	Min	Max	Units
V_{IH}	Input HIGH voltage	V_{DD_IO} = 1.8V	1.4	V_{DDIO} + 0.3	V
V_{IH}	Input HIGH voltage	V_{DD_IO} = 2.8V	2.4	V_{DDIO} + 0.3	V
V_{IL}	Input LOW voltage	V_{DD_IO} = 1.8V	GND - 0.3	0.4	V
V_{IL}	Input LOW voltage	V_{DD_IO} = 2.8V	GND - 0.3	0.8	V
I_{IN}	Input leakage current	No pull-up resistor; V_{IN} = V_{DD} or D_{GND}	-20	20	µA
V_{OH}	Output HIGH voltage	At specified I_{OH}	V_{DDIO} - 0.4V	–	V
V_{OL}	Output LOW voltage	At specified I_{OL}	–	0.4	V
I_{OH}	Output HIGH current	At specified V_{OH}	–	-12	mA
I_{OL}	Output LOW current	At specified V_{OL}	–	9	mA
I_{OZ}	Tri-state output leakage current		–	10	µA

Table 35: I/O Timing

^{f}EXTCLK = 24 MHz; V_{DD} = 1.8V; V_{DD_IO} = 1.8V; V_{AA} = 2.8V; V_{AA_PIX} = 2.8V; V_{DD_PLL} = 2.8V

Symbol	Definition	Conditions	Min	Typ	Max	Units
^{f}EXTCLK	Input clock frequency	PLL enabled	6	24	48	MHz
^{t}EXTCLK	Input clock period	PLL enabled	166	41	20	ns
^{t}R	Input clock rise time		0.1	–	1	V/ns
^{t}F	Input clock fall time		0.1	–	1	V/ns
	Clock duty cycle		45	50	55	%
^{t}JITTER	Input clock jitter		–	–	0.7	ns
Output pin slew	Fastest	C_{LOAD} = 15pF	–	0.7	–	V/ns
^{f}PIXCLK	PIXCLK frequency	Default	–	96	–	MHz
^{t}PD	PIXCLK to data valid	Default	–	–	3	ns
^{t}PFH	PIXCLK to FRAME_VALID HIGH	Default	–	–	3	ns
^{t}PLH	PIXCLK to LINE_VALID HIGH	Default	–	–	3	ns
^{t}PFL	PIXCLK to FRAME_VALID LOW	Default	–	–	3	ns
^{t}PLL	PIXCLK to LINE_VALID LOW	Default	–	–	3	ns

Figure 53: I/O Timing Diagram



Caution **Stresses greater than those listed in Table 36 may cause permanent damage to the device. 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.**

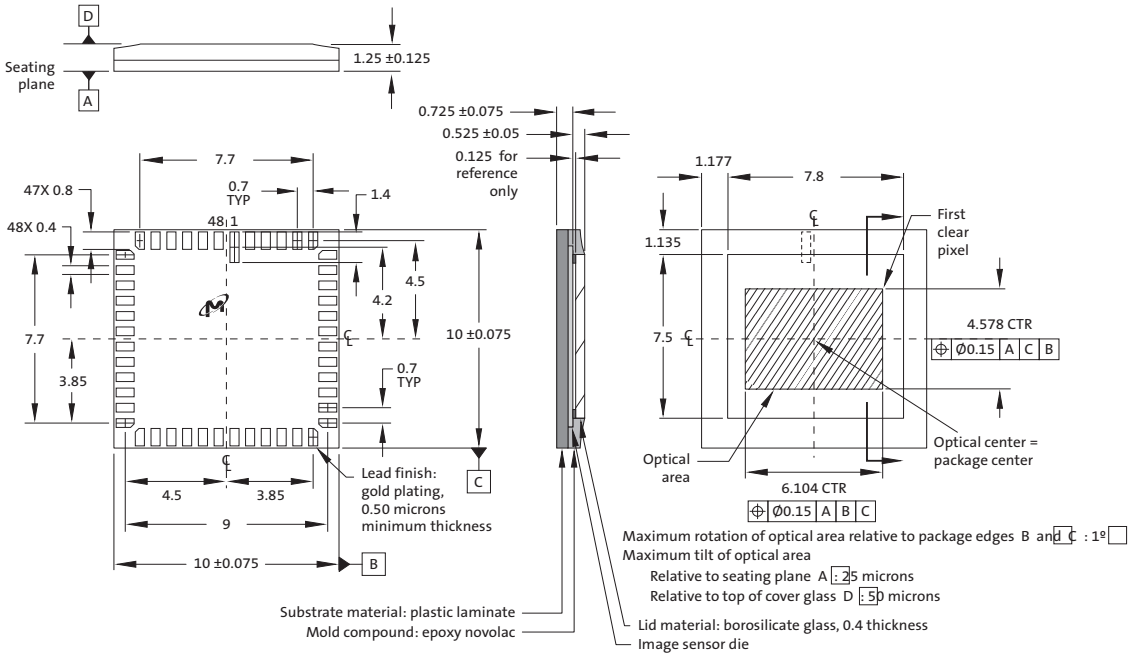
Table 36: Absolute Maximum Ratings

Symbol	Definition	Condition	Min	Max	Unit
V_{DD}_MAX	Core digital voltage		–0.3	1.9	V
V_{DD}_IO_MAX	I/O digital voltage		–0.3	3.1	V
VAA_MAX	Analog voltage		–0.3	3.5	V
VAA_PIX	Pixel supply voltage		–0.3	3.5	V
V_{DD}_PLL	PLL supply voltage		–0.3	3.5	V
I_{DD}	Digital operating current		–	90	mA
I_{DD}_IO	I/O digital operating current		–	100	mA
IAA_MAX	Analog operating current		–	225	mA
IAA_PIX	Pixel supply current		– 6	25	mA
I_{DD}_PLL	PLL supply current		–	25	mA
^{t}OP	Operating temperature	Measure at junction	–30	70	°C
^{t}ST	Storage temperature		–40	85	°C

Notes:
1. Exposure to absolute maximum rating conditions for extended periods may affect reliability.
2. To keep dark current and shot noise artifacts from impacting image quality, care should be taken to keep TOP at a minimum.

Package Dimensions

Figure 54: 48-Pin iLCC Package Outline Drawing



Notes: 1. All dimensions in millimeters.

SMIA and MIPI Specification Reference

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

1. SMIA Specifications:
   - Functional Specification:
     SMIA 1.0 Part 1: Functional Specification (Version 1.0 dated 30 June 2004)
     SMIA 1.0 Part 1: Functional Specification ECR0001 (Version 1.0 dated 11 Feb 2005)
   - Electrical Specification:
     SMIA 1.0 Part 2: CCP2 Specification (Version 1.0 dated 30 June 2004)
     SMIA 1.0 Part 2: CCP2 Specification ECR0001 (Version 1.0 dated 11 Feb 2005)
2. MIPI Specifications:
   - MIPI Alliance Standard for CSI-2 version 1.0
   - MIPI Alliance Standard for D-PHY version 0.81

Revision History

Rev. E, Production 6/17/2008

- Changed maximum ^{t}Jitter from 0.3 to 0.7 in Table 35 on page 122
- Added Table 36 on page 123

Rev. D, Production 5/5/2008

- Updated Figure 51: “Soft Standby and Soft Reset,” on page 119
- Deleted Figure 53: “Two-Wire Serial Bus Timing Parameters”
- Updated title of Table 35, “I/O Timing,” on page 122 to add parameters
- Replaced Table 34 on page 122 “Two-Wire Serial Interface Electrical Characteristics” with Table 34, “I/O Parameters,” on page 122

Rev. C, Production 4/10/2008

- Update "Features" on page 1
- Update "Applications" on page 1
- Update Table 1, “Key Performance Parameters,” on page 1
- Update "Operating Modes" on page 9
- Update Figure 5: “Typical Configuration: CCP2 Parallel Pixel Data Interface,” on page 12
- Update Figure 7: “48-Pin iLCC Package Pinout Diagram,” on page 15
- Update Table 3, “Signal Descriptions,” on page 14
- Update Figure 7: “48-Pin iLCC Package Pinout Diagram,” on page 15
- Update Table 12, “Register Description—Manufacturer-Specific,” on page 47
- Add "Scaler" on page 84
- Update "Binning" on page 97
- Update "Power-Down Sequence" on page 117
- Update "Spectral Characteristics" on page 120
- Update "Quantum Efficiency" on page 120
- Update "Electrical Characteristics" on page 121

Rev. B, Preliminary 10/07

- Update "Features" on page 1
- Update Table 1, “Key Performance Parameters,” on page 1
- Update "Operating Modes" on page 9
- Update Figure 3: “Typical Configuration: Serial CCP2 Pixel Data Interface,” on page 10
- Update Figure 4: “Typical Configuration: Serial Two-Lane MIPI Pixel Data Interface,” on page 11
- Update Figure 5: “Typical Configuration: CCP2 Parallel Pixel Data Interface,” on page 12
- Update Figure 7: “48-Pin iLCC Package Pinout Diagram,” on page 15
- Add Table 7, “Register List and Default—SMIA Configuration,” on page 27
- Add Table 8, “Register List and Default Values—SMIA Parameter Limits,” on page 30
- Add Table 9, “Register List and Default Values—Manufacturer-Specific,” on page 32
- Add See “Register Descriptions” on page 38
- Add Table 10, “Register Description—SMIA Configuration,” on page 38
- Add Table 11, “Register Description—SMIA Parameter Limits,” on page 43
- Add Table 12, “Register Description—Manufacturer-Specific,” on page 47
- Add "Programming Restrictions" on page 69

- Update Table 13, “Definitions for Programming Rules,” on page 69
- Update "Programming Restrictions when Subsampling" on page 95
- Update Table 25, “Minimum Row Time and Blanking Numbers,” on page 101
- Update Table 27, “fine_integration_time Limits,” on page 102
- Update Table 28, “fine_correction Values,” on page 102
- Update "Low Power Mode" on page 103
- Add "Package Dimensions" on page 124
- Add Figure 54: “48-Pin iLCC Package Outline Drawing,” on page 124

Rev. A, Advance 8/13/2007

- Initial release

3080 North 1st Street, San Jose, CA 95134, prodmktg@aptina.com www.aptina.com