1/3.2-Inch 3.1Mp CMOS Digital Image Sensor

MT9T012 Data Sheet

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

Features

- Superior low-light performance
- Low dark current
- 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, frame size/rate, exposure, left–right and top–bottom image reversal, window size and panning
- SMIA-compatible
- Data interfaces: parallel and sub-low-voltage differential signalling (sub-LVDS)
- On-chip PLL
- Bayer-pattern down-size scaler
- Four channel lens shading correction (LC) with independent corner correction

Applications

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

General Description

The Aptina® MT9T012 is a 1/3.2-inch QXGA-format CMOS active-pixel digital image sensor with a pixel array of 2056H x 1544V (2048H x 1536V 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.

When operated in its default mode, the sensor generates a QXGA image at 15 frames per second (fps). An on-chip analog-to-digital converter (ADC) generates a 10-bit value for each pixel.

Table 1: Key Performance Parameters

Parameter		Value
Optical format		1/3.2-inch QXGA (4:3)
Active imager size		4.52mm(H) x 3.40mm(V) 5.66mm diagonal
Active pixels		2056H x 1544V
Pixel size		2.2 x 2.2µm
Color filter array		RGB Bayer pattern
Chief ray angle		21.35° at 85% image height
Shutter type		Electronic rolling shutter (ERS)
Maximum data rate/master clock		64 Mp/s at 64 MHz PIXCLK
Frame rate	QXGA (2048 x 1536)	Programmable up to 15 fps
	VGA (640 x 480)	Programmable up to 30 fps
ADC resolution		10-bit, on-chip (61dB)
Responsivity		0.53 V/lux-sec
Dynamic range		59.5dB
SNR_{MAX}		37.7dB
Supply voltage	Analog	2.40–3.10V (2.50V or 2.80V nominal)
	Digital	1.70–1.90V (1.80V nominal)
	I/O	1.70–3.10V
Power consumption		205mW
Operating temperature		–30°C to +70°C
Packaging		Bare die

The MT9T012 digital image sensor features DigitalClarity—Aptina's breakthrough low-noise CMOS imaging technology that achieves CCD image quality (based on signal-to-noise ratio and low-light sensitivity) while maintaining the inherent size, cost, and integration advantages of CMOS.

Ordering Information

Table 2: Available Part Numbers

Part Number	Description
MT9T012D00STC C16CC1	Bare die

Table of Contents

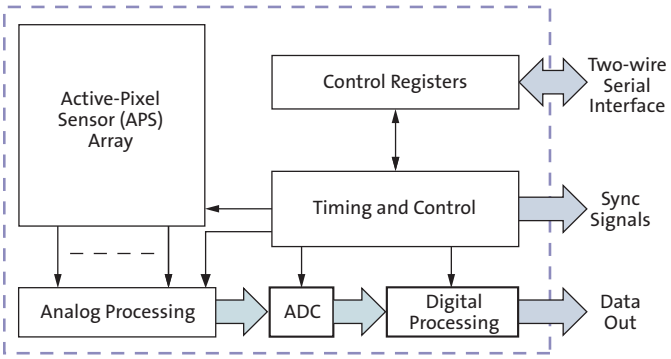
List of Figures

List of Tables

Functional Overview

The MT9T012 is a progressive-scan sensor that generates a stream of pixel data at a constant frame rate. It uses an on-chip, phase-locked loop (PLL) to generate all internal clocks from a single master input clock running between 6 and 27 MHz. The maximum pixel rate is 64 Mp/s, corresponding to a pixel clock rate of 64 MHz. Figure 1 shows a block diagram of the sensor.

Figure 1: Block Diagram



The core of the sensor is a 3Mp 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 analog-to-digital converter (ADC). The output from the ADC is a 10-bit value for each pixel in the array. The ADC output passes through a digital processing signal chain (which provides further data path corrections and applies digital gain).

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

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

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

The control registers, timing and control and digital processing functions shown in Figure 1 are partitioned into two logical parts:

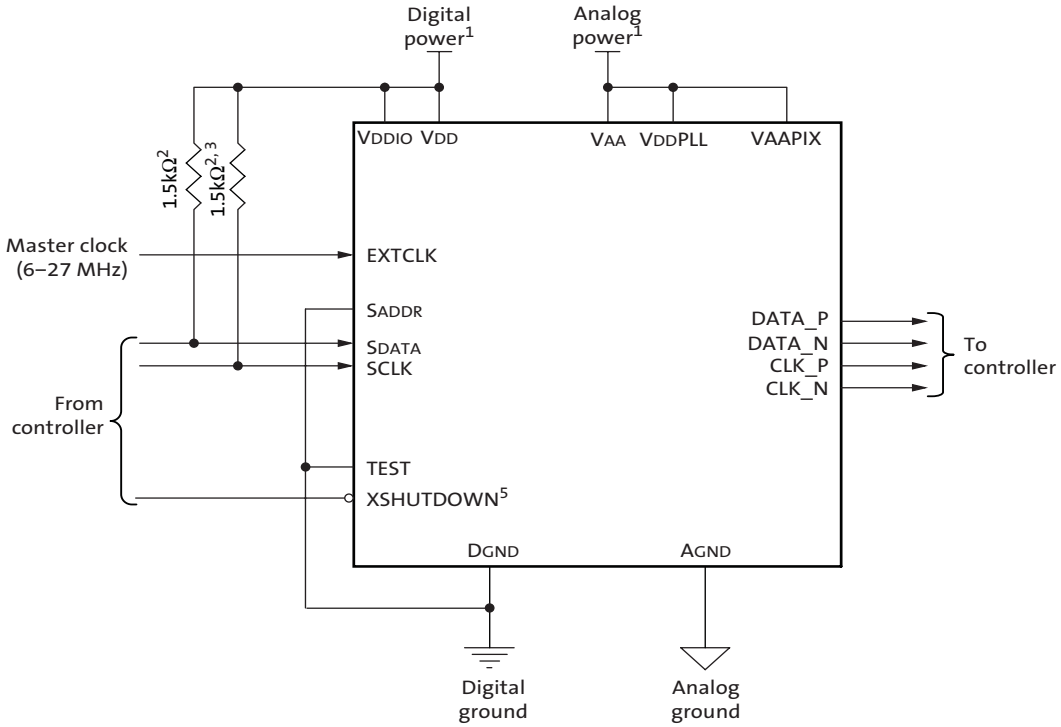
- A sensor core which provides array control and data path corrections. The output of the sensor core is a 10-bit parallel pixel data stream qualified by an output data clock (PIXCLK), together with LINE_VALID and FRAME_VALID signals.
- Additional functionality is required to support the SMIA standard. This includes a horizontal and vertical image scaler, a limiter, a data compressor, an output FIFO, and a serialiser.

A flash output strobe 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.

Operating Modes

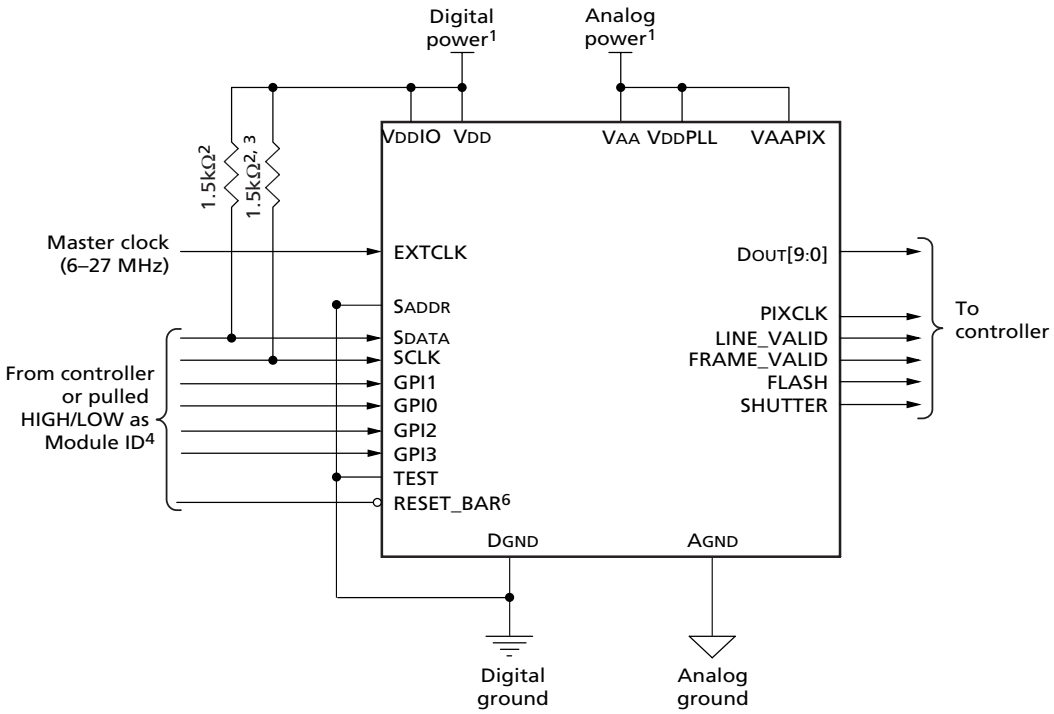
By default, the MT9T012 powers up as a SMIA-compatible sensor with the serial pixel data interface enabled. A typical configuration in this mode is shown in Figure 2. The MT9T012 can also be configured to operate with a parallel pixel data interface. A typical configuration in this mode is shown in Figure 3. These two operating modes are described in "Control of the Signal Interface" on page 24.

Figure 2: Typical Configuration: Serial Output Mode



Notes:
1. All power supplies should be adequately decoupled.
2. A resistor value of 1.5kΩ is recommended, but 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. VAA and VAAPIX must be tied together.
5. Also referred to as RESET_BAR.

Figure 3: Typical Configuration: Parallel Output Mode



Notes:
1. All power supplies should be adequately decoupled.
2. A resistor value of 1.5kΩ is recommended, but 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 either statically pulled HIGH/LOW to be used as module IDs, or can be programmed to perform special functions (TRIGGER, OE_N, STANDBY) to be dynamically controlled.
5. VAA and VAAPIX must be tied together.
6. Also referred to as XSHUTDOWN.

Signal Descriptions

Table 3 provides signal descriptions for MT9T012 die. For pad location and aperture information, refer to the MT9T012 die data sheet.

Table 3: Signal Descriptions

Pad Name	Pad Type	Description
EXTCLK	Input	Master clock input. PLL input clock. 6 –27 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.
SADDR	Input	Normally tied LOW so that the sensor responds to a two-wire serial interface device address of 0x20/0x21. When tied HIGH, the sensor responds to a two-wire serial interface device address of 0x30/0x31.
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 and shutter trigger functions. See "General Purpose Inputs" on page 28.
TEST	Input	Enable manufacturing test modes. Wire to digital GND for functional operation.
SDATA	I/O	Serial data for reads from and writes to control and status registers.
DATA_P	Output	Differential CCP (sub-LVDS) serial data (positive).
DATA_N	Output	Differential CCP (sub-LVDS) serial data (negative).
CLK_P	Output	Differential CCP (sub-LVDS) serial clock/strobe (positive).
CLK_N	Output	Differential CCP (sub-LVDS) serial clock/strobe (negative).
LINE_VALID	Output	LINE_VALID output. Qualified by PIXCLK.
FRAME_VALID	Output	FRAME_VALID output. Qualified by PIXCLK.
DOUT(9:0)	Output	Pixel data output. Qualified by PIXCLK.
PIXCLK	Output	Pixel clock. Used to qualify the LINE_VALID, FRAME_VALID and DOUT(9:0) outputs.
FLASH	Output	Flash output. Synchronization pulse for external light source.
SHUTTER	Output	Control for external mechanical shutter.
VAA1, VAA2, VAA3, VAA4	Supply	Analog power supply.
VAAPIX1, VAAPIX2, VAAPIX3	Supply	Analog power supply for the pixel array.
AGND1, AGND2, AGND3, AGND4	Supply	Analog ground.
VDD1, VDD2, VDD3, VDD4	Supply	Digital power supply.
VDDIO1, VDDIO2, VDDIO3, VDDIO4	Supply	I/O power supply.
DGND1, DGND2, DGND3, DGND4, DGND5	Supply	Common ground for digital and I/O.
VDDPLL	Supply	PLL power supply.

Output Data Format

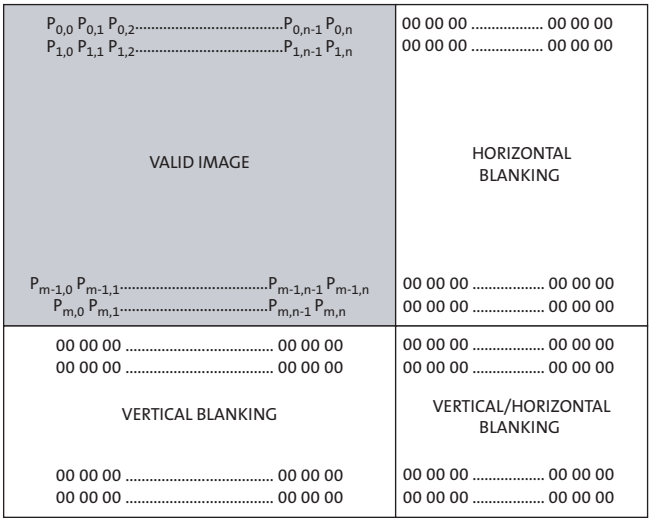
CCP Serial Pixel Data Interface

The MT9T012 serial pixel data interface implements data/clock and data/strobe signalling in accordance with the CCP2 specification. The RAW8 and RAW10 image data formats are supported.

Parallel Pixel Data Interface

MT9T012 image data is read out in a progressive scan. Valid image data is surrounded by horizontal blanking and vertical blanking, as shown in Figure 4. The amount of horizontal blanking and vertical blanking is programmable; LINE_VALID is HIGH during the shaded region of the figure. FRAME_VALID timing is described in the next section.

Figure 4: Spatial Illustration of Image Readout



Output Data Timing (Parallel Pixel Data Interface)

MT9T012 output data is synchronized with the PIXCLK output. When LINE_VALID is HIGH, one pixel value is output on the 10-bit DOUT output every PIXCLK period. By default, the pixel clock runs at the same frequency as the sensor's master input clock (vt_pix_clk_freq_mhz) and rising edges on the PIXCLK signal occur one-half of a pixel clock period after transitions on LINE_VALID, FRAME_VALID, and DOUT (see Figure 5 on page 12). This allows PIXCLK to be used as a clock to sample the data. PIXCLK is continuously enabled, even during the blanking period. The MT9T012 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 6 on page 12 are defined in Table 4 on page 12.

Figure 5: Pixel Data Timing Example

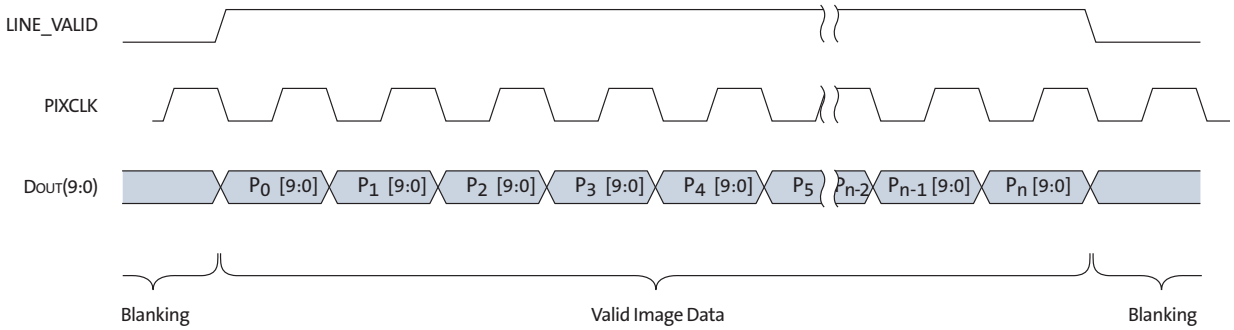


Figure 6: Row Timing and FRAME_VALID/LINE_VALID Signals

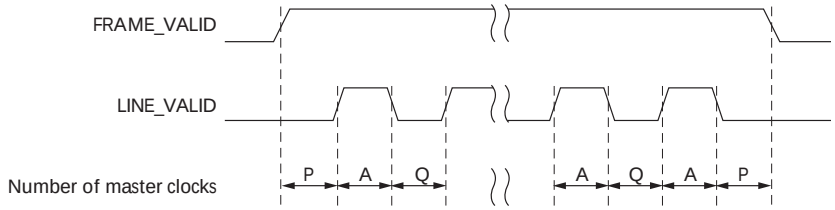


Table 4: Row Timing

Parameter	Name	Equation	Default Timing at 64 MHz
PIXCLK_PERIOD	Pixel clock period	R0x3016–7[2:0] / vt_pix_clk_freq_mhz	1 pixel clock = 15.625ns
S	Skip (subsampling) factor	For x_odd_inc = y_odd_inc = 3, S = 2 otherwise, S = 1	1
A	Active data time	(x_addr_end - x_addr_start + 1) * PIXCLK_PERIOD/S	2048 pixel clocks = 32.0µs
P	Frame start/end blanking	6 * PIXCLK_PERIOD	6 pixel clocks = 93.75ns
Q	Horizontal blanking	(line_length_pck - A) * PIXCLK_PERIOD	2732 - 2048 pixel clocks = 10.6875µs
A + Q	Row time	line_length_pck * PIXCLK_PERIOD	2732 pixel clocks = 42.6875µs
V	Vertical blanking	((frame_length_lines - N) * (A+Q)) + Q - (2*P)	44,384 pixel clocks = 693.5µs
N	Number of rows	(y_addr_end - y_addr_start + 1)/S	1536 rows
N * (A + Q)	Frame valid time	(N * (A + Q)) - Q + (2*P)	4,195,680 pixel clocks = 65.5575ms
F	Total frame time	line_length_pck * frame_length_lines * PIXCLK_PERIOD	4,240,064 pixel clocks = 66.251ms

The sensor timing (Table 4) is shown in terms of pixel clock and master clock cycles (see Figure 5). The default settings for the on-chip PLL generate a 64 MHz master input clock and pixel clock given a 16 MHz input clock to the MT9T012. Equations for calculating the frame rate are given in "Frame Rate Control" on page 47.

Two-Wire Serial Register Interface

The two-wire serial interface bus enables read/write access to control and status registers within the MT9T012. This interface is designed to be compatible with the SMIA 1.0 Part2: CCP2 Specification camera control interface (CCI) which uses the electrical characteristics and transfer protocols of the two-wire serial interface 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 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 MT9T012 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, as follows:

- a (repeated) start condition
- a slave address/data direction byte
- an (a no) acknowledge bit
- a message byte
- 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 an 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 MT9T012 are 0x20 (write address) and 0x21 (read address) in accordance with the SMIA specification. Alternate slave addresses of 0x30 (write address) and 0x31 (read address) can be selected by asserting the SADDR input signal.

Message Byte

Message bytes are used for sending register addresses and register write data to the slave device and for retrieving register read data. The protocol used is outside the scope of the two-wire serial interface specification and is defined as part of the SMIA CCI.

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.

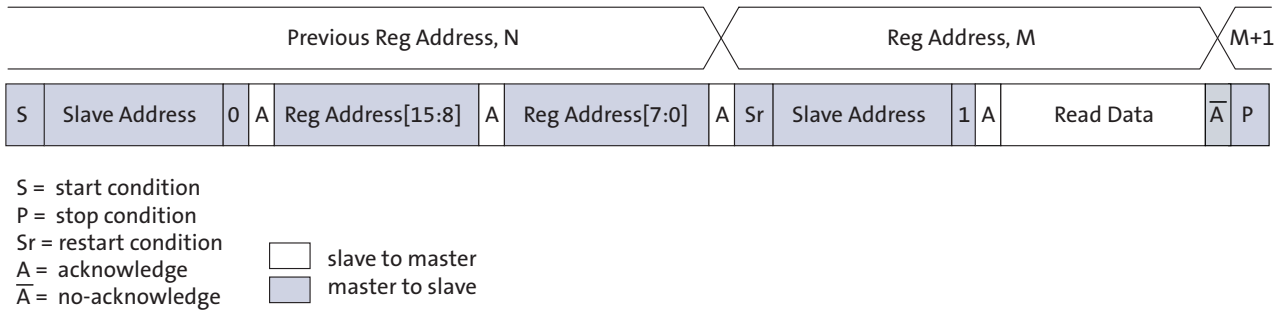
If the request was a write, the master then transfers the 16-bit register address to which the write should take place. This transfer takes place as two 8-bit sequences and the slave sends an acknowledge bit after each sequence to indicate that the byte has been received. The master then transfers the data as an 8-bit sequence; the slave sends an acknowledge bit at the end of the sequence. After 8 bits have been transferred, the slave's internal register address is incremented automatically, so that the next 8 bits are written to the next register address. 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 the 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 auto-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 (see Figure 7) 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 7 shows how the internal register address maintained by the MT9T012 is loaded and incremented as the sequence proceeds.

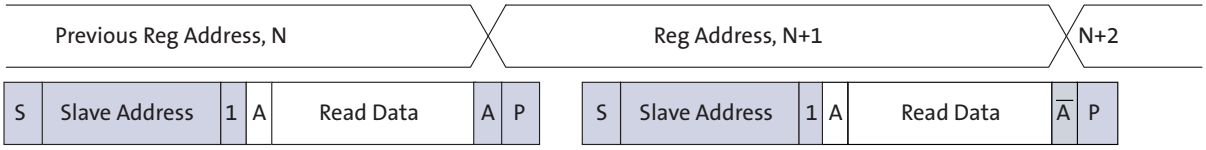
Figure 7: Single READ from Random Location



Single READ from Current Location

This sequence (Figure 8) performs a read using the current value of the MT9T012 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.

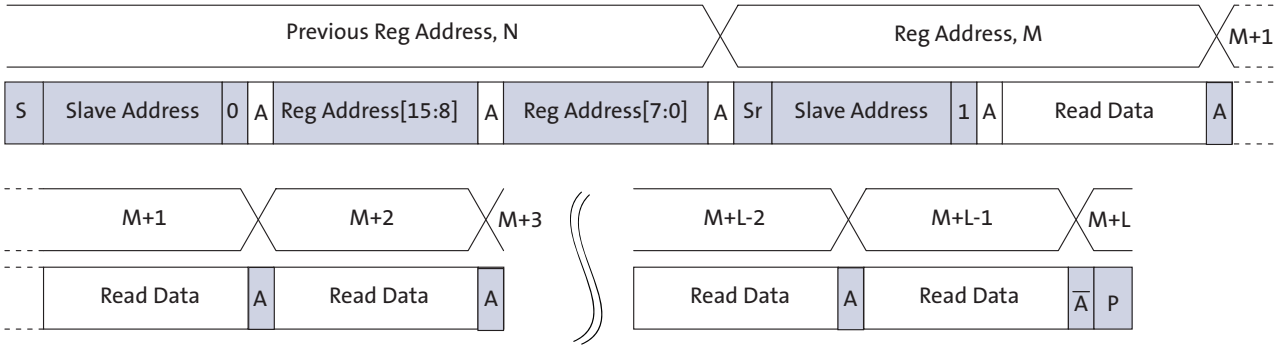
Figure 8: Single READ from Current Location



Sequential READ, Start from Random Location

This sequence (Figure 9 on page 16) starts in the same way as the single READ from random location (Figure 7). 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.

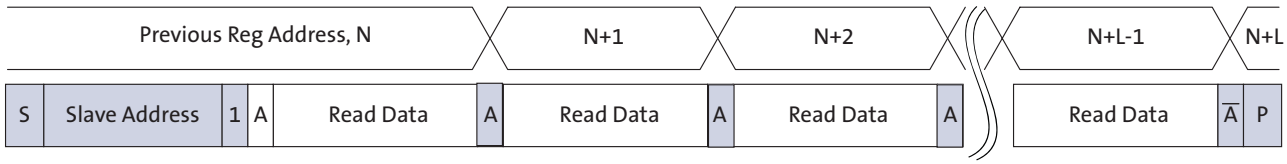
Figure 9: Sequential READ, Start from Random Location



Sequential READ Start from Current Location

This sequence (Figure 10) starts in the same way as the single READ from current location (Figure 8 on page 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 READs until *L* bytes have been read.

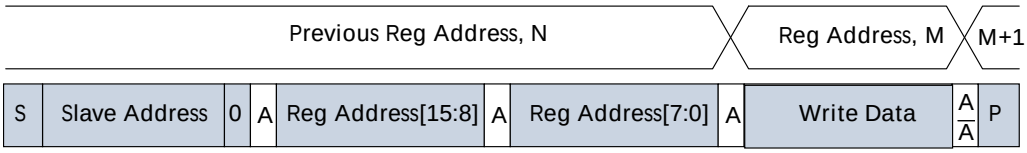
Figure 10: Sequential READ, Start from Current Location



Single WRITE to Random Location

This sequence (Figure 11) 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.

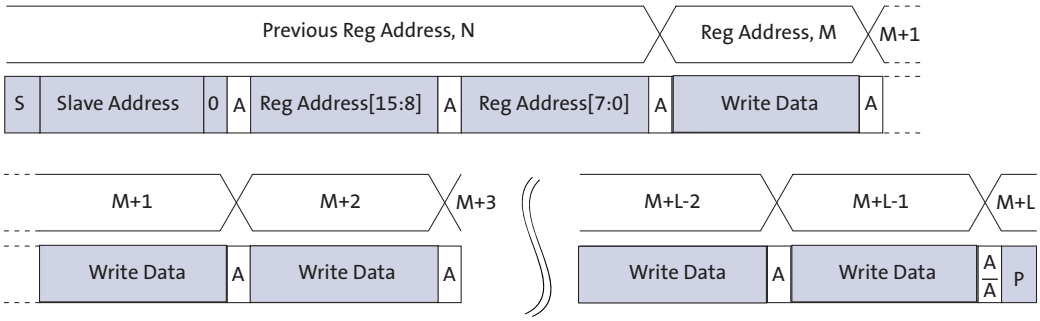
Figure 11: Single WRITE to Random Location



Sequential WRITE, Start at Random Location

This sequence (Figure 12) starts in the same way as the single WRITE to random location (Figure 11 on page 16). Instead of generating a stop condition after the first byte of data has been transferred, the master continues to perform byte WRITEs until *L* bytes have been written. The WRITE is terminated by the master generating a stop condition.

Figure 12: Sequential WRITE, Start at Random Location



Programming Restrictions

The SMIA specification imposes a number of programming restrictions. An implementation naturally imposes additional restrictions. Table 6 shows a list of programming rules that must be adhered to for correct operation of the MT9T012. It is recommended that these rules are encoded into the device driver stack: either implicitly or explicitly.

Table 5: Definitions for Programming Rules

Name	Definition
xskip	if (x_odd_inc == 1) xskip = 1 else xskip = 2
yskip	if (y_odd_inc == 1) yskip = 1 else yskip = 2

Table 6: Programming Rules

Parameter	Minimum Value	Maximum Value	Origin
coarse_integration_time	coarse_integration_time_min	frame_length_lines - coarse_integration_time_max_margin	SMIA
fine_integration_time	fine_integration_time_min	line_length_pck - fine_integration_time_max_margin	SMIA
digital_gain_*	digital_gain_min	digital_gain_max	SMIA
digital_gain_* is an integer multiple of digital_gain_step_size			SMIA
frame_length_lines	min_frame_length_lines	max_frame_length_lines	SMIA
line_length_pck	min_line_length_pck	max_line_length_pck	SMIA
line_length_pck	((x_addr_end - x_addr_start + 1)/ xskip) + min_line_blanking_pck		SMIA
frame_length_lines	((y_addr_end - y_addr_start + 1)/ yskip) + min_frame_blanking_lines		SMIA
x_addr_start	x_addr_min	x_addr_max	SMIA
x_addr_end	x_addr_start	x_addr_max	SMIA
(x_addr_end - x_addr_start+ 1)	must be positive	must be positive	SMIA
x_addr_start[0]	0	0	SMIA
x_addr_end[0]	1	1	SMIA
y_addr_start	y_addr_min	y_addr_max	SMIA
y_addr_end	y_addr_start	y_addr_max	SMIA
(y_addr_end - y_addr_start + 1)/	must be positive	must be positive	SMIA
y_addr_start[0]	0	0	SMIA
y_addr_end[0]	1	1	SMIA
x_even_inc	min_even_inc	max_even_inc	SMIA
x_even_inc[0]	1	1	SMIA
y_even_inc	min_even_inc	max_even_inc	SMIA
y_even_inc[0]	1	1	SMIA
x_odd_inc	min_odd_inc	max_odd_inc	SMIA
x_odd_inc[0]	1	1	SMIA
y_odd_inc	min_odd_inc	max_odd_inc	SMIA
y_odd_inc[0]	1	1	SMIA
scale_m	scaler_m_min	scaler_m_max	SMIA

Table 6: Programming Rules (Continued)

Parameter	Minimum Value	Maximum Value	Origin
scale_n	scaler_n_min	scaler_n_max	SMIA
x_output_size	256 (this is enforced in hardware: values lower than this are treated as 256)	2132	Minimum from SMIA FS Section 5.2.2.5 Maximum is a consequence of the output FIFO size on this implementation.
x_output_size[0]	0 (this is enforced in hardware: bit[0] is read-only)	0	SMIA FS Section 5.2.2.2
y_output_size	2	frame_length_lines (PRELIMINARY)	Minimum ensures 1 Bayer row-pair. Maximum avoids output frame being longer than pixel array frame.
y_output_size[0]	0 (this is enforced in hardware: bit[0] is read-only)	0	SMIA FS Section 5.2.2.2
with subsampling, start and end pixels must be addressed (impact on x/y start/end addresses, function of image orientation bits)			SMIA FS Errata See "Subsampling" on page 38.

Output Size Restrictions

The SMIA CCP 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 8 (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).

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 CCP2 data stream.

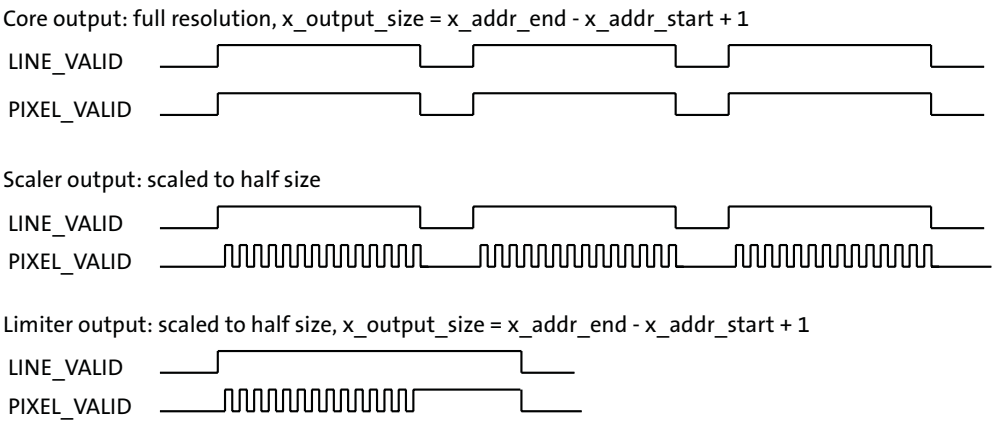
When the parallel pixel data path is in use, the only restriction on x_output_size is that it must be even (x_output_size[0] = 0) and this restriction is enforced in hardware.

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, the ccp_signalling_mode 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 MT9T012 will mis-operate 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 13.

Figure 13: Effect of Limiter on SMIA Data Path



In this figure, three different stages in the SMIA data path (See "SMIA Digital Data Path" on page 64) 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

LINE_VALID signal is asserts once per row and remains asserted for *N* pixel times. The PIXEL_VALID signal toggles with the same timing as LINE_VALID, indicating that all pixels in the row are valid.

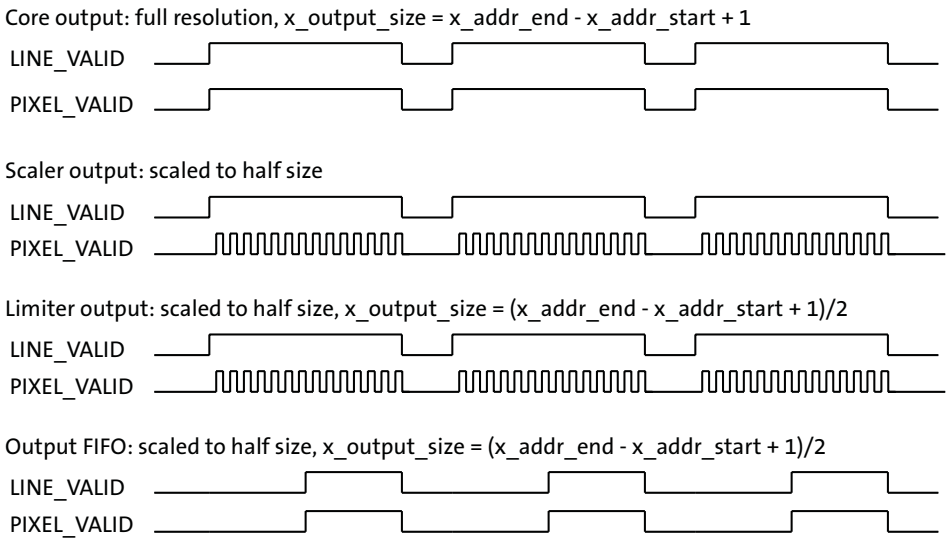
The second stage is the output of the scaler, when the scaler is set to reduce the image size by 1/2 in each dimension. The effect of the scaler is to combine groups of pixels. Therefore, the row time remains the same but only half the pixels out of the scaler are valid. This is signalled by transitions 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 LINE_VALID across the whole of the horizontal blanking time, as shown in the figure, the MT9T012 will cease to generate output frames.

A correct configuration is shown in Figure 14. This figure shows the x_output_size reduced to match the output size of the scaler. In this configuration, the output of the limiter does not extend LINE_VALID.

This figure also shows the effect of the output FIFO, which forms the final stage in the SMIA 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 14: Timing of SMIA Data Path



Effect of CCP Class on Legal Range of Output Sizes/Frame Rate

The pixel array readout rate is set by line_length_pck * frame_length_lines. With the default register values one frame time takes 2,732 * 1,552 = 4,240,064 pixel periods. This value includes vertical and horizontal blanking times, so that the full-size image 2,048 x 1,538 (1,536 lines of pixel data, 2 lines of embedded information) forms a subset of these pixels.

When the internal clock is running at 64 MHz, this frame time corresponds to 4,240,064/64e6 = 0.066251 second, giving rise to a frame rate of 15.09 fps.

Each pixel is 10 bits, by default. This data rate requires the serial interface to transmit 4,240,064 * 10 bits in 0.066251 second. That is, the serial data rate must be 10X the pixel rate - 640 Mb/s.

The SMIA CCP2 specification shows that class 0 (data/clock) runs up to 208Mb/s. Therefore, it is not possible to generate full resolution images at 15 fps using CCP class 0. Changing the ccp_data_format (to use 8 bits per pixel) reduces the bandwidth requirement, but is not enough to allows full-resolution operation.

The only way to get a full image out is to reduce the pixel clock rate until it is appropriate for the maximum CCP class 0 data rate. This requires the pixel rate to be reduced to 20.8 MHz. This has the side effect of reducing the frame rate. Repeating the calculation above, at 20.8 MHz internal clock, this corresponds to 4,240,064/20.8e6 = 0.2038 second, giving rise to a frame rate of 4.9 fps.

To use CCP class 0 with an internal clock of 64 MHz it is necessary to reduce the amount of output data. This can be achieved by changing x_output_size, y_output_size so that less data comes out per frame. A change to the output size can be done in conjunction with windowing the image from the sensor (by adjusting x_addr_start, x_addr_end, y_addr_start, y_addr_end) or by enabling the scaler.

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 CCP2 data stream must be greater than or equal to the row time at the pixel array. The row time on the CCP2 data stream is calculated from the x_output_size and the ccp_data_format (8 or 10 bits per pixel), and must include the time taken in the CCP2 data stream for start of frame/row, end of row/frame and checksum symbols.

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 (R0x306E–F).

Changing Registers While Streaming

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

- ccp2_channel_identifier
- ccp2_signalling_mode
- ccp_data_format
- scale_m
- vt_pix_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 50.

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 the following signals:

- SCLK
- SDATA
- SADDR

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 state, 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 in an input-only signal and must always be driven to a valid logic level for correct operation. In most applications this input will be hardwired to logic “0.”

This interface is described in detail in "Two-Wire Serial Register Interface" on page 13.

Default Power-up State

The MT9T012 provides two separate interfaces for pixel data, the CCP2 high-speed serial interface described by the SMIA specification, and a parallel data interface.

At power up and after a hard or soft reset, the reset state of the MT9T012 is to enable SMIA operation and the CCP2 high-speed serial interface.

Serial Pixel Data Interface

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

- DATA_P
- DATA_N
- CLK_P
- CLK_N

The signal pairs are driven differentially using sub-LVDS switching levels. This interface conforms to the SMIA 1.0 CCP2 requirements and supports both data/clock signalling and data/strobe signalling.

The serial pixel data interface is enabled by default at power up and after reset.

The DATA_P, DATA_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 unused, the LINE_VALID, FRAME_VALID, PIXCLK and DOUT[9:0] signals can be left unconnected.

Parallel Pixel Data Interface

The parallel pixel data interface uses the following output-only signals:

- FRAME_VALID

- LINE_VALID
- PIXCLK
- DOUT(9:0)

The parallel pixel data interface is disabled by default at power up and after reset. It can be enabled by programming R0x301A. Table 14 shows the recommended settings.

When the parallel pixel data interface is in use the DATA_P, DATA_N, CLK_P, CLK_N signals can be left unconnected.

Output Enable Control

When the parallel pixel data interface is enabled, its signals can be switched asynchronously between the driven and High-Z state, under pin or register control, as shown in Table 13. Selection of a pin to use for the OE_N function is described in "General Purpose Inputs" on page 28.

Table 13: 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 14.

Table 14: Configuration of the Pixel Data Interface

SMIA Disable R301A–B[12]	SMIA CLock Disable R301A–B[11]	Standby End-Of-Frame R301A–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	0	1	Parallel pixel data interface, pixel data processed by SMIA data path. Serial pixel data interface disabled to save power, but SMIA data path clocking enabled. Transitions to soft standby are synchronized to the end of frames on the serial pixel data interface to ensure correct synchronization with the SMIA data path.

System States

The system states of the MT9T012 are represented as a state diagram in Figure 15 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 15 on page 27.

Figure 15: MT9T012 System States

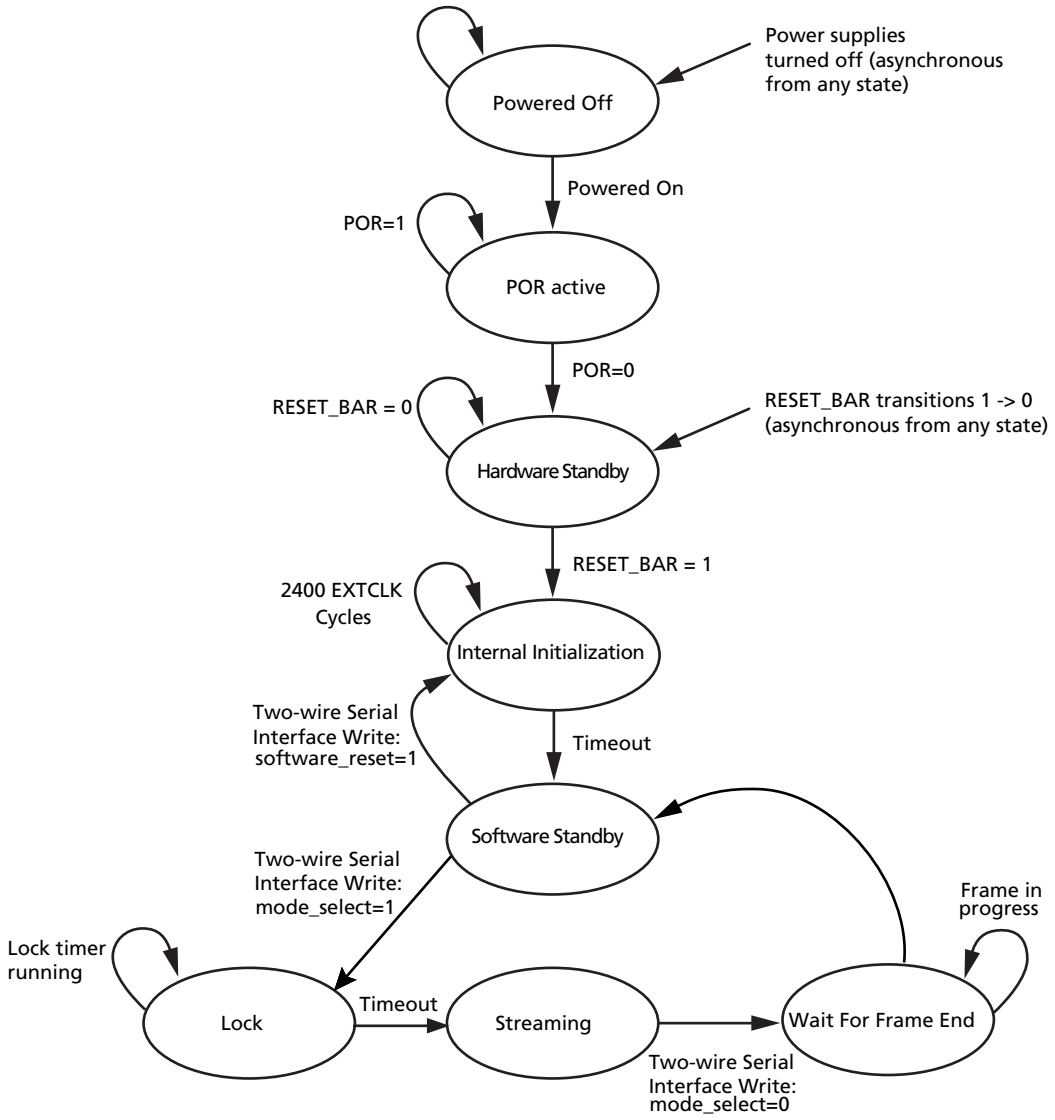


Table 15: RESET_BAR and PLL in System States

State	RESET_BAR	PLL
Powered off	x	VCO powered down
POR active	x	
Hardware standby	0	
Internal Initialization	1	VCO powered down, PLL clock outputs bypassed by EXTCLK
Software standby		
Lock		VCO running, PLL clock outputs active
Streaming		
Wait for frame end		

Power-On Reset Sequence

When power is applied to the MT9T012 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 time-out of the internal power-on reset circuit.

It is possible to hold RESET_BAR permanently negated and rely upon the internal power-on reset circuit.

The RESET_BAR signal is functionally equivalent to the SMIA-specified XSHUTDOWN signal.

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 MT9T012 is in its lowest-powered, powered-up state; the internal PLL is disabled, the CCP2 serialiser is disabled and internal clocks are gated off.

When the sensor leaves the hardware standby state it performs an internal initialization sequence that takes 2,400 EXTCLK cycles. After this time it enters a low-power software standby state. While the initialization sequence is in progress, the MT9T012 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 so 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 (0x16 if R0x0000 is read).

Soft Reset Sequence

The MT9T012 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 briefly enters the hardware standby state and then starts its internal initialization sequence. At this point, the behavior is exactly the same as for the power-on reset sequence.

Signal State During Reset

Table 16 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 16: 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(9:0)	Output		
PIXCLK	Output		
SCL	Input	Enabled. Must be pulled up or driven to a valid logic level.	
SDA	I/O	Enabled as an input. Must be pulled-up or driven to a valid logic level.	
SADDR	Input	Enabled. Must be driven to a valid logic level.	
FLASH	Output	High-Z.	Logic 0.
SHUTTER	Output	High-Z.	Logic 0.
DATA_P	Output	High-Z.	
DATA_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.	

General Purpose Inputs

The MT9T012 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]. 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].

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 25)
- Trigger (see the sections below)
- Standby functions (see the sections below)

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

Streaming/Standby Control

The MT9T012 can be switched between its soft standby and streaming states under pin or register control, as shown in Table 17. Selection of a pin to use for the STANDBY function is described in "General Purpose Inputs" above. The state diagram for transitions between soft standby and streaming states is shown in Figure 15 on page 26.

Table 17: 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 18. Selection of a pin to use for the TRIGGER function is described in "General Purpose Inputs" above.

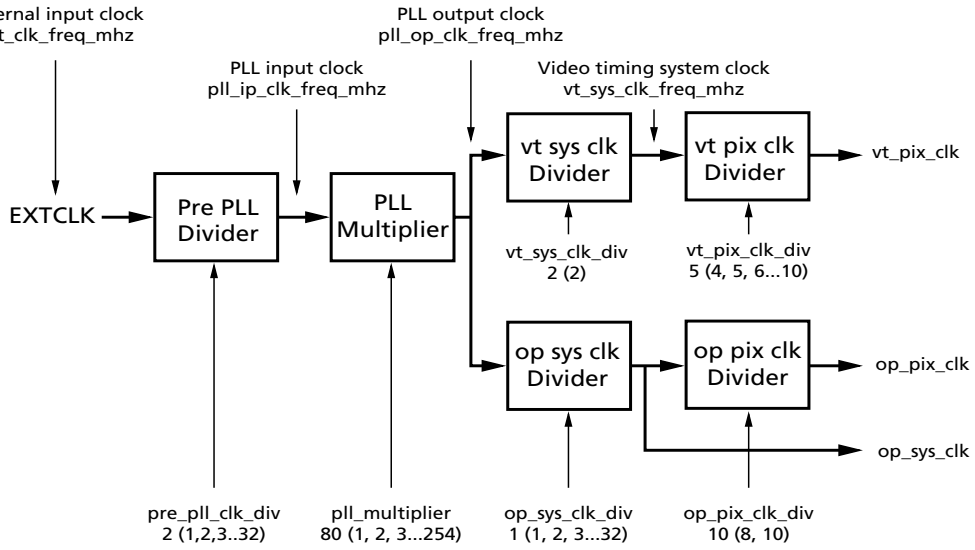
Table 18: Trigger Control

TRIGGER	Global Trigger R0x3060–1[0]	Description
Disabled	0	Idle
Disabled	1	Trigger
0	0	Idle
X	1	Trigger
1	X	Trigger

Clocking

The MT9T012 contains a phase-locked loop (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 16.

Figure 16: MT9T012 Clocking Structure



The figure shows the different clocks and the names of the registers that contain or are used to control their values. The figure shows the default setting for each divider/multiplier control register and the range of legal values for each divider/multiplier control register. From the diagram, the output clock frequencies can be calculated as follows:

$$\text{vt\_pix\_clk\_freq\_mhz} = \frac{\texttt{ext\_clk\_freq\_mhz} \times \texttt{pll\_multiplier}}{\texttt{pre\_pll\_plk\_div} \times \texttt{vt\_sys\_clk\_div} \times \texttt{vt\_pix\_clk\_div}}$$

$$\text{op\_pix\_clk\_freq\_mhz} = \frac{\texttt{ext\_clk\_freq\_mhz} \times \texttt{pll\_multiplier}}{\texttt{pre\_pll\_plk\_div} \times \texttt{op\_sys\_clk\_div} \times \texttt{op\_pix\_clk\_div}}$$

$$\text{op\_sys\_clk\_freq\_mhz} = \frac{\texttt{ext\_clk\_freq\_mhz} \times \texttt{pll\_multiplier}}{\texttt{pre\_pll\_plk\_div} \times \texttt{op\_sys\_clk\_div}}$$

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.
- The minimum/maximum value for the divider/multiplier must be met.
- The value of pll_multiplier should be a multiple of 2. Strictly speaking, this is only necessary for CCP2 Class 1/Class 2 operation, where the programmed value is right-shifted by 1 to produce the PLL multiplier input value.
- The op_pix_clk must never run faster than the vt_pix_clk to ensure that the CCP2 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, the valid combinations of the clock divisors are as shown in Table 19 and Table 20.

Table 19: Valid Divisor Combinations (10 bits per-pixel)

op_sys_clk_div	vt_pix_clk_div
1	4,5
2,4,6,8	4,5,6,7,8,9,10
10	5,6,7,8,9,10
12	6,7,8,9,10
14	7,8,9,10
16	8,9,10
18,20	9,10
22	10

Table 20: Valid Divisor Combinations (8 bits per-pixel)

op_sys_clk_div	vt_pix_clk_div
1	4
2	4,5,6,7,8
4,6,8,10	4,5,6,7,8,9,10
12	5,6,7,8,9,10
14,16	6,7,8,9,10
18	7,8,9,10
20	8,9,10
22,24	9,10
26	10

The PLL VCO input frequency range is 2.0 MHz –11.5 MHz.

The usage of the output clocks is shown below:

- vt_pix_clk is used by the sensor core to control the timing of the pixel array. The sensor core produces one 10-bit pixel each vt_pix_clk period. The line length (line_length_pck) and fine integration time (fine_integration_time) are controlled in increments of the vt_pix_clk period. When the MT9T012 is configured to generate

parallel pixel data output, and the parallel pixel data output MUX is in its default setting, the PIXCLK output will run at the same frequency as vt_pix_clk.

- op_pix_clk is used to load parallel pixel data from the output FIFO (see Figure 42 on page 64) to the CCP2 serialiser. The output FIFO generates one pixel each op_pix_clk period. The pixel is either 8-bit or 10-bit depending upon the output data format, controlled by R0x0112-3 (ccp_data_format).
- op_sys_clk is used to generate the serial data stream on the CCP2 output. The relationship between this clock frequency and the op_pix_clk frequency is dependent upon the output data format.

Programming the PLL Divisors

The PLL divisors should be programmed while the MT9T012 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.

The VCO lock time is 100µs (typical), 1ms (maximum).

The effect of programming the PLL divisors while the MT9T012 is in the streaming state is undefined.

PLL VCO Power-Down

In the MT9T012, the PLL VCO is automatically powered down when the sensor is in streaming mode. R0x301A–B[5] has no influence upon the operation of the PLL VCO. In previous revisions, the PLL VCO remained powered while the sensor was in streaming mode and could be manually powered down by setting R0x301A–B[5] = 1.

Influence of ccp_data_format

R0x0112-3 (ccp_data_format) controls whether the pixel data interface will generate 10 bits per-pixel or 8 bits per-pixel. The raw output of the sensor core is 10 bits per-pixel; the two 8-bit modes represent a compressed data mode and a mode in which the 2 LSBs of the 10-bit data are discarded.

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.

Influence of ccp2_signalling_mode

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

When data/clock signalling is selected, the pll_multiplier supports both odd and even values. When data/strobe signalling is selected, the pll_multiplier only supports even values; the LSB of the programmed value is ignored and treated as “0.”

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

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

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

Programming Examples

This section provides four programming examples. For each example, a table of register values is shown (for example, Table 21). The settings for the clock divisors show the "Programmed Value" and "Apparent Frequency." These values are consistent with Figure 16 and the associated equations for the different clocks. The table also shows the "Effective Value" and "Actual Frequency." These values are implementation details that reflect the internal operation of the clocks, they are consistent with the descriptions given in "Influence of ccp_data_format" on page 32.

1. Example 1:

- 10 bits per-pixel data
- CCP2 Class 1/Class 2 signalling
- Highest possible frame rate at maximum resolution

In order to meet the requirement for the highest possible frame rate, vt_pix_clk should run at 64 MHz. op_pix_clk needs to run at the same rate. For 10 bits per-pixel operation, op_sys_clk must run at 10x op_pix_clk. Therefore, op_sys_clk_div is set to 1 and op_pix_clk_div to 10, giving an overall divide-by-10 in the op clock domain. Since vt_sys_clk_div is fixed at 2, the same divide-by-10 is achieved in the vt clock domain by setting vt_pix_clk_div to 5. As a result, the PLL output frequency must be set to 64 * 10 = 640 MHz. There are various ways doing this, depending upon the frequency of EXTCLK.

The register settings for this example and the resultant clock frequencies are shown in Table 21. If the MT9T012 is programmed with these values (and all other registers are left at their default values), and is then put into streaming mode (mode_select = 1) it will stream frames at full resolution (2,048 x 1,536 pixels) through its CCP interface at 15.09 fps.

Table 21: Programming Example 1

Register	Programmed Value	Effective Value	Clock	Apparent Frequency	Actual Frequency
ccp_data_format	0x0A0A	10 bits per-pixel	–	–	–
ccp2_signalling_mode	1	Data/Strobe signalling	–	–	–
–	–	–	EXTCLK	16 MHz	16 MHz
pre_pll_clk_div	2	2	pll_ip_clk	8 MHz	8 MHz
pll_multiplier	80	40	pll_op_clk	640 MHz	320 MHz
vt_sys_clk_div	2	1	vt_sys_clk	320 MHz	320 MHz
vt_pix_clk_div	5	5	vt_pix_clk	64 MHz	64 MHz
op_sys_clk_div	1	1	op_sys_clk	640 MHz	320 MHz
op_pix_clk_div	10	5	op_pix_clk	64 MHz	64 MHz

2. Example 2:

- Highest possible frame rate
- 8 bits per-pixel data
- CCP2 Class 0 signalling

When operating with Class 0 signalling, there is insufficient bandwidth on the interface to get full resolution images out of the sensor at 15 fps (See "Influence of ccp_data_format" on page 32). In order to run the CCP interface at maximum speed in Class 0, the divisors are chosen to set op_sys_clk at 208 MHz. When choosing the vt divisors, there are two options, maximize image size, or maximize frame rate. This example maximizes the frame rate. Therefore, the vt divisors are chosen to give a vt_pix_clk of 2x the op_pix_clk. Since this produces twice as much data as the bandwidth of the CCP2 Class 0 interface can accommodate, the data per frame is reduced by setting x_output_size to 1,024. The register settings for this example and the resultant clock frequencies are shown in Table 22. If the MT9T012 is programmed with these values (and all other registers are left at their default values), and is then put into streaming mode (mode_select = 1) it will stream frames at half resolution (1,024 x 1,536 pixels) through its CCP interface at 11.97 fps.

.

Table 22: Programming Example 2

Register	Programmed Value	Effective Value	Clock	Apparent Frequency	Actual Frequency
ccp_data_format	0x0808	8 bits per-pixel	—	—	—
ccp2_signalling_mode	0	Data/Clock signalling	—	—	—
—	—	—	EXTCLK	16 MHz	16 MHz
pre_pll_clk_div	4	4	pll_ip_clk	4 MHz	4 MHz
pll_multiplier	52	52	pll_op_clk	208 MHz	208 MHz
vt_sys_clk_div	2	1	vt_sys_clk	104 MHz	208 MHz
vt_pix_clk_div	2	4	vt_pix_clk	52 MHz	52 MHz
op_sys_clk_div	1	1	op_sys_clk	208 MHz	208 MHz
op_pix_clk_div	8	8	op_pix_clk	26 MHz	26 MHz
x_output_size	0x400	—	—	—	—

3. Example 3:

- 8 bits per-pixel data
- CCP2 Class 1/ Class 2 signalling
- Highest possible frame rate at maximum resolution

In order to meet the requirement for the highest possible frame rate, vt_pix_clk should run at 64 MHz. op_pix_clk needs to run at the same rate. For 8-bit per pixel operation, op_sys_clk must run at 8x op_pix_clk. Therefore, op_sys_clk_div is set to 1 and op_pix_clk_div to 8, giving an overall divide-by-8 in the op clock domain. Since vt_sys_clk_div is fixed at 2, the same divide-by-8 is achieved in the vt clock domain by setting vt_pix_clk_div to 4. As a result, the PLL output frequency must be set to 64 * 8 = 512 MHz. There are various ways doing this, depending upon the frequency of EXTCLK.

The register settings for this example and the resultant clock frequencies are shown in Table 23. If the MT9T012 is programmed with these values (and all other registers are left at their default values), and is then put into streaming mode (mode_select = 1) it will stream frames at full resolution (2,048 x 1,536 pixels) through its CCP interface at 15.09 fps.

Table 23: Programming Example 3

Register	Programmed Value	Effective Value	Clock	Apparent Frequency	Actual Frequency
ccp_data_format	0x0808	8 bits per-pixel	—	—	—
ccp2_signalling_mode	1	Data/Strobe signalling	—	—	—
—	—	—	EXTCLK	16 MHz	16 MHz
pre_pll_clk_div	3	3	pll_ip_clk	5.33 MHz	5.33 MHz
pll_multiplier	96	48	pll_op_clk	512 MHz	256 MHz
vt_sys_clk_div	2	1	vt_sys_clk	256 MHz	256 MHz
vt_pix_clk_div	4	4	vt_pix_clk	64 MHz	64 MHz
op_sys_clk_div	1	2	op_sys_clk	512 MHz	256 MHz
op_pix_clk_div	8	4	op_pix_clk	64 MHz	64 MHz

4. Example 4:

- 8 bits per-pixel data
- CCP2 Class 0 signalling
- Full resolution image

When operating with Class 0 signalling, there is insufficient bandwidth on the interface to get full resolution images out of the sensor at 15 fps (see "Influence of ccp_data_format" on page 32). In order to run the CCP interface at maximum speed in Class 0, the divisors are chosen to set op_sys_clk at 208 MHz. When choosing the vt divisors, there are two options: maximize image size or maximize frame rate. This example maximizes the image size. Therefore, the vt divisors are chosen to give a vt_pix_clk that matches the op_pix_clk.

The register settings for this example and the resultant clock frequencies are shown in Table 24. If the MT9T012 is programmed with these values (and all other registers are left at their default values), and is then put into streaming mode (mode_select=1) it will stream frames at full resolution (2,048 x 1,536 pixels) through its CCP interface at 5.99 fps.

Table 24: Programming Example 4

Register	Programmed Value	Effective Value	Clock	Apparent Frequency	Actual Frequency
ccp_data_format	0x0808	8 bits per-pixel	–	–	–
ccp2_signalling_mode	0	Data/Clock signalling	–	–	–
–	–	–	EXTCLK	16 MHz	16 MHz
pre_pll_clk_div	4	4	pll_ip_clk	4 MHz	4 MHz
pll_multiplier	52	52	pll_op_clk	208 MHz	208 MHz
vt_sys_clk_div	2	1	vt_sys_clk	104 MHz	208 MHz
vt_pix_clk_div	4	4	vt_pix_clk	26 MHz	26 MHz
op_sys_clk_div	1	1	op_sys_clk	208 MHz	208 MHz
op_pix_clk_div	8	8	op_pix_clk	26 MHz	26 MHz

Clock Control

The MT9T012 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 MT9T012 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

Image Acquisition Modes

The MT9T012 supports two image acquisition modes:

1. Electronic rolling shutter (ERS) mode. This is the normal mode of operation. When the MT9T012 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 MT9T012 switches cleanly from the old integration time to the new while only generating frames with uniform integration.
2. Global reset mode. This mode can be used to acquire a single image at the current resolution. In this mode, the pixel integration time is controlled by an external electromechanical shutter, and the MT9T012 provides control signals to interface to that shutter. The operation of this mode is described in detail in "Global Reset" on page 50.

The use of an external electromechanical shutter increases cost and may reduce ruggedness of the end application. The motivation for the use of an external electromechanical shutter is that it eliminates the visual artefacts associated with ERS operation. Visual artefacts 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, so that its operation is somewhat akin to that of a photo-finish machine at a race track.

Window Control

The sequencing of the pixel array is controlled by the x_addr_start, y_addr_start, x_addr_end, y_addr_end registers. When the sensor core data path is enabled through the parallel pixel data interface, the whole image (controlled by the x_addr_start, y_addr_start, x_addr_end and y_addr_end registers) is output from the sensor. When the SMIA data path is enabled, 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 2048H x 1536V image. A border of up to 4 pixels on each edge can be enabled by reprogramming the x_addr_start, y_addr_start, x_addr_end and y_addr_end registers and then (if the SMIA data path is enabled) adjusting the x_output_size and y_output_size registers accordingly.

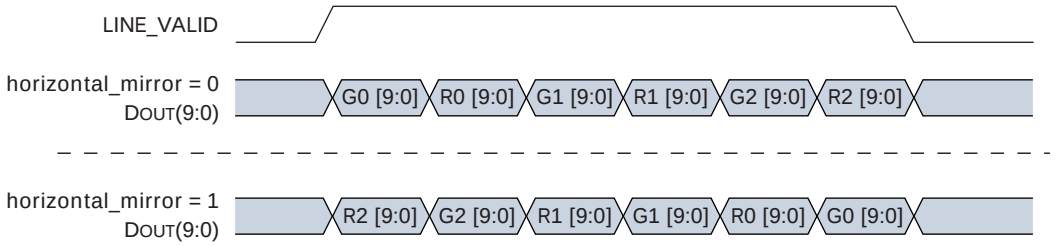
Readout Modes

Horizontal Mirror

When the horizontal_mirror bit is set in the image_orientation register, the order of pixel readout within a row is reversed, so that readout starts from x_addr_end and ends at x_addr_start. Figure 17 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.

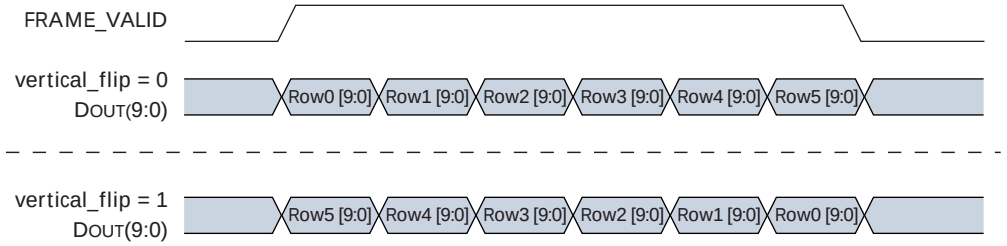
Figure 17: Effect of horizontal_mirror on Readout Order



Vertical Flip

When the vertical_flip bit is set in the image_orientation register, the order in which pixel rows are read out is reversed, so that row readout starts from y_addr_end and ends at y_addr_start. Figure 18 shows a sequence of 6 rows being read out with vertical_flip=0 and vertical_flip=1. Changing vertical_flip causes the bayer order of the output image to change; the new bayer order is reflected in the value of the pixel_order register.

Figure 18: Effect of vertical_flip on Readout Order



Subsampling

The MT9T012 supports subsampling. Subsampling reduces the amount of data processed by the analogue signal chain in the sensor and thereby allows the frame rate to be increased. Subsampling is enabled by setting x_odd_inc = 3 and/or y_odd_inc=3. This reduces the amount of row and column data processed and is equivalent to the skip2 readout mode provided by earlier Aptina Imaging sensors. Figure 19 shows a sequence of pixels being read out with x_odd_inc = 3 and y_odd_inc = 1. MT9T012 supports an additional subsampling value, y_odd_inc = 7. The effect of the different subsampling settings on the pixel array readout is shown in Figures 20 through 23.

Figure 19: Effect of x_odd_inc = 3 on Readout Sequence

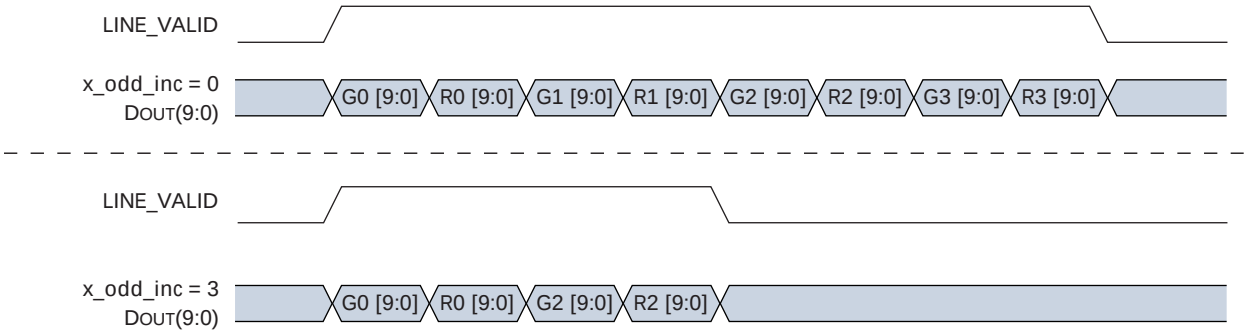


Figure 20: Pixel Readout (no subsampling)

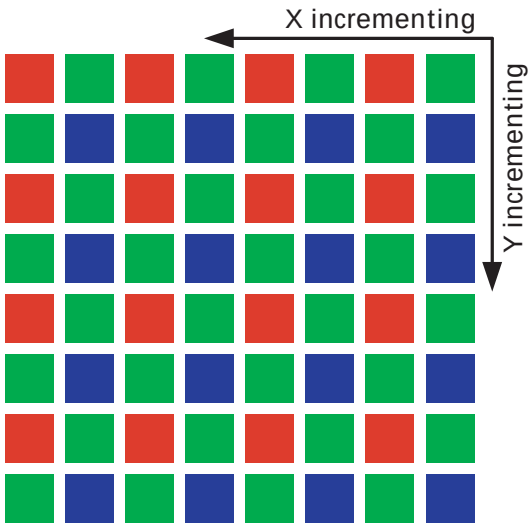


Figure 21: Pixel Readout (x_odd_inc = 3, y_odd_inc = 1)

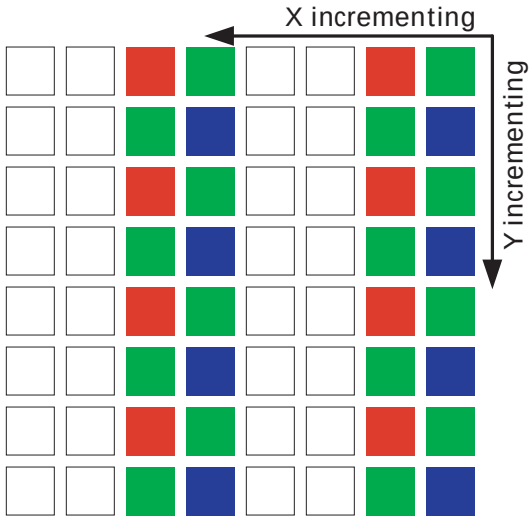


Figure 22: Pixel Readout (x_odd_inc = 1, y_odd_inc = 3)

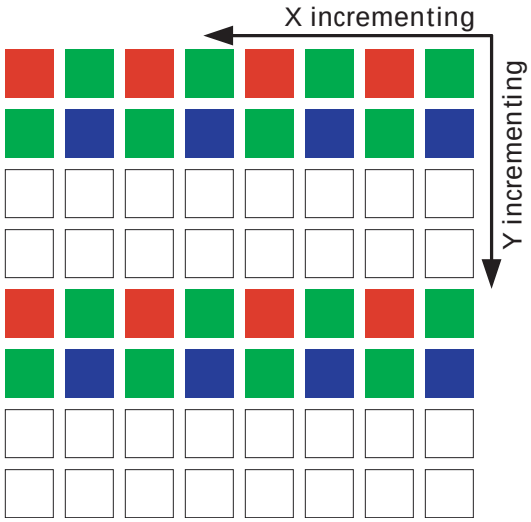
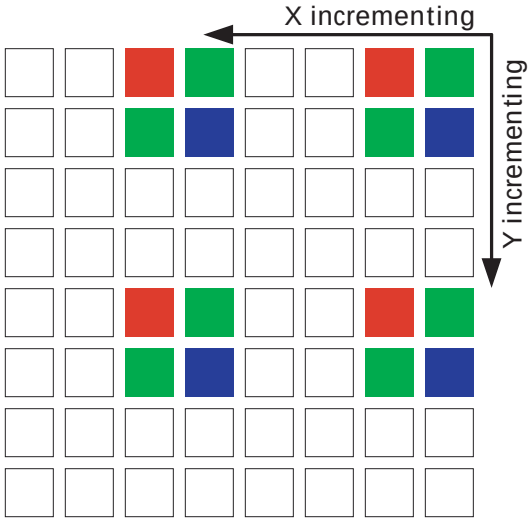


Figure 23: Pixel Readout (x_odd_inc = 3, y_odd_inc = 3)



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, it is recommended 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 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 the following rule:

remainder = (addr_end - addr_start + 1) AND 4

if (remainder == 0) addr_end = addr_end - 2

For a y_odd_inc = 7, the adjustment should be made in accordance with the following rule:

remainder = (y_addr_end - y_addr_start + 1) AND 7

if (remainder == 0) y_addr_end = y_addr_end - 6

if (remainder == 6) y_addr_end = y_addr_end - 4

if (remainder == 4) y_addr_end = y_addr_end - 2

Table 25 on page 41 shows the row address sequencing for normal and subsampled (with y_odd_inc = 3) readout. The same sequencing applies to column addresses for subsampled readout. There are two possible subsampling sequences (because the subsampling sequence only read half of the rows and columns) depending upon the alignment of the start address. The row address sequencing during binning is are also shown; the column address sequencing during binning is described in "Programming Restrictions when Binning" on page 42.

Table 25: Row Address Sequencing

Normal	Subsampled	Subsampled	Binned	Binned
0	0		0,2	
1	1		1,3	
2		2		2,4
3		3		3,5
4	4		4,6	
5	5		5,7	
6		6		6,8
7		7		7,9

Binning

The MT9T012 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 (x_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 for subsampling, x_addr_end and y_addr_end may require adjustment when binning is enabled.

The effect of the different subsampling settings is shown in Figure 24 and Figure 25 on page 42.

Figure 24: Pixel Readout (x_odd_inc = 3, y_odd_inc = 1, x_bin = 1)

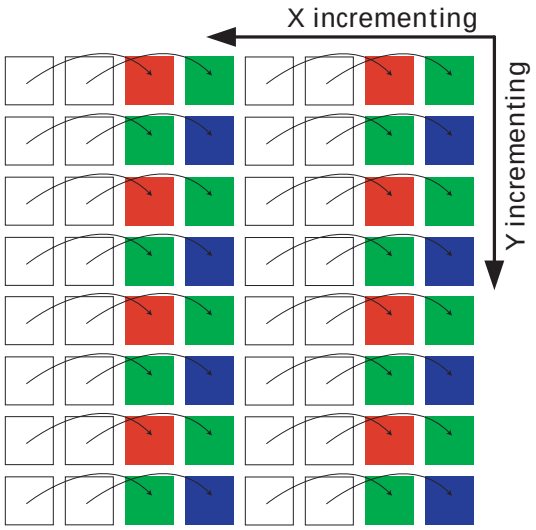
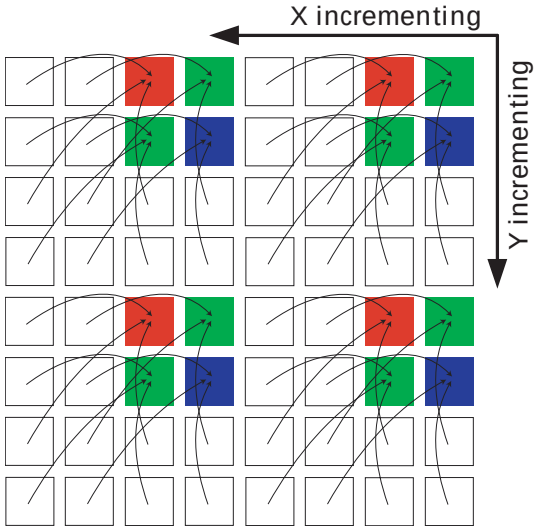


Figure 25: Pixel Readout (x_odd_inc = 3, y_odd_inc = 3, xy_bin = 1)



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, since 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. The recommended

settings are shown in Table 26. None of these adjustments are required for x-binning. The sensor must be taken out of streaming mode before switching between binned and non binned operation.

Table 26: Register Adjustments Required for Binning Mode

Register	Type	Default (Normal Readout)	Recommended Setting during Binning	Notes
min_line_blanking_pck	read-only	0x02AC	0x0468	Read-only register for control software; does not affect operation of sensor.
min_line_length_pck	read-only	0x03A8	0x05FD	Read-only register for control software; does not affect operation of sensor.
fine_integration_time_min	read-only	0x0204	0x03E5	Read-only register for control software; does not affect operation of sensor.
fine_integration_time_max_margin	read-only	0x0100	0x0215	Read-only register for control software; does not affect operation of sensor.
sample_time_pck	read/write	0x01EC	0x03CD	Affects operation of sensor
fine_correction	read/write	0x0100	0x0215	Affects operation of sensor
fine_integration_time	read/write	0x0204	0x03E5	Normal default is minimum value
x_addr_min	read-only	0	2	
x_addr_max	read-only	0x0807	0x0805	Keep binned column within the array
y_addr_min	read-only	0	0	Unchanged
y_addr_max	read-only	0x0607	0x0605	Keep binned row within the array
dark control	read-/write	0x0542	0x0541	

Since binning on the MT9T012 also requires subsampling to be enabled, the same restrictions apply to the setting of x_addr_end and y_addr_end (See "Programming Restrictions when Subsampling" on page 40).

When xy-binning is enabled and vertical-flip is enabled, the first row in the output image (which is generated from the row at the highest address in the pixel array) is binned incorrectly. It is binned with itself instead of with the adjacent same-color row. This results in the first row having a lower intensity than other rows in the image. This bug occurs for all settings of y_addr_start.

A given row *n* will always be binned with row *n*+2. Therefore, there are two candidate rows that a row can be binned with, depending upon the alignment of y_addr_start. The possible sequences are shown in Table 25 on page 41.

For a given column *n*, there is only one other column, *n_bin*, that is can be binned with:

remainder = (*n* + 2) AND 2;

n_bin = *n* + 2 - (2 * remainder);

Table 27 shows some examples of row and column binning pairs. Since row and column addresses less than 0 are illegal, this leads to the revised array limits shown in Table 26.

Table 27: Row and Column Binning Pairs

x_addr_start	Binned with
0	-2
2	4
4	2
6	8
8	6
10	12
12	10

y_addr_start	Binned with
0	2
2	4
4	6
6	8
8	10
10	12
12	14

Lens Shading Correction (LC)

Lenses tend to produce images whose brightness is significantly attenuated near the edges. There are also other factors causing fixed pattern signal gradients in images captured by image sensors. The cumulative result of all these factors is known as lens shading. The MT9T012 has an embedded lens shading correction module that can be programmed to counter the shading effect of a lens on each individual R, Gb, Gr, and B color signal.

The MT9T012 also includes 16 independent corner parameters, K, for each of the color channels. The K factors are independently adjustable at each corner and are not dependent on the vertical and horizontal spatial dimension.

LC can be enabled and disabled with R0x318A[15].

Lens Shading Correction Procedure

Lens correction is implemented by multiplying the value of incoming pixel data $S_i(x,y,c)$ by the value of the correction function Gain(x,y,c) which is calculated based upon the position and the color channel of the array sensor.

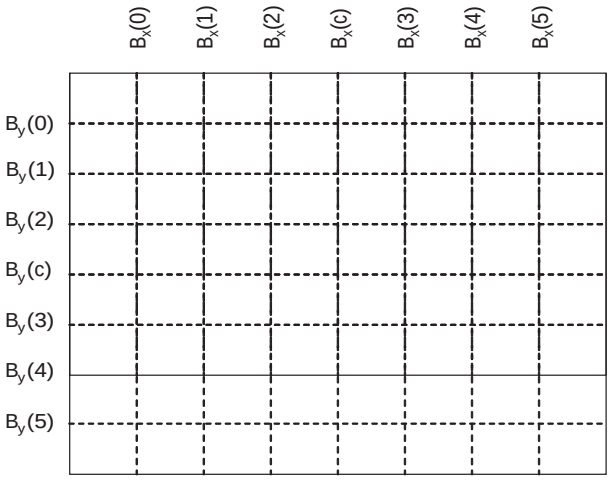
$$S_o(x,y,c) = S_i(x,y,c) * \mathrm{Gain}(x,y,c)$$

$$G(x,y,c) = G_0(c) + F_x(x,c) + F_y(y,c) + k(x,y,c) * (F_x(x,c) * 2\hat{}kpwrx - a) * (F_y(y,c) * 2\hat{}kpwry - a)$$

Lens Correction Zones

In order to increase the precision of the correction function, the array is divided into number of zones. Figure 26 shows the pixel array subdivided into eight horizontal and six vertical zones. The locations of the boundaries Bx By can be specified with R0x3602–R0x360C. This allows finer correction of complex shading responses. Each coordinate is stored as a byte, which represents the coordinate value divided by 32. There is no distinction for boundaries of different colors.

Figure 26: Definition of Zone Boundaries



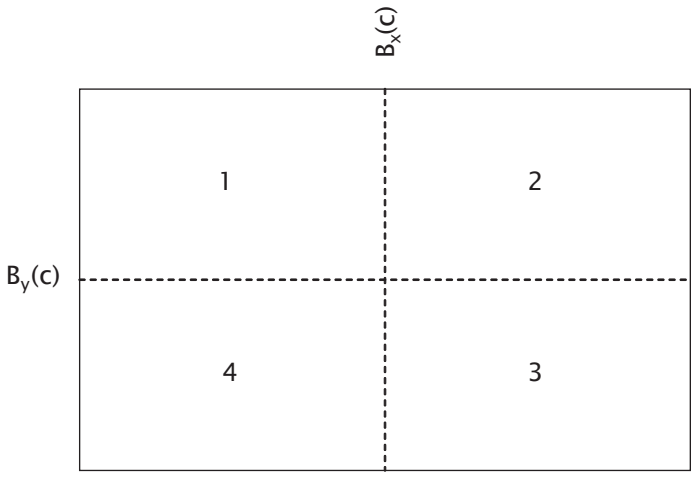
The coordinates of zone boundaries are referenced respectively to the lens center point C (position of the lens principal axis), where there coordinates of C are (Cx, Cy) are the center offset points specified at R0x360E.

The Correction Function

$$G(x,y,c) = G_0(c) + F_x(x,c) + F_y(y,c) + k(x,y,c) * (F_x(x,c) * 2\text{^}kpwrx - a) * (F_y(y,c) * 2\text{^}kpwry - a)$$

- F_x (x, c) F_y(y, c) are piece-wise quadratic (PWQ) functions. The sensor is divided into eight zones in the x and y directions as shown in Figure 26 on page 45. The function for each color is represented by 10 values–eight for the eight zones in the X and Y direction and two for the initial conditions used iteratively to calculate the function values across the entire pixel array. The F_x and F_y terms for each color channel are calculated iteratively by integrating Fx" and Fy".
- The initial values of $F_x(0,c)$, $F_y(0,c)$, $F_x'(0,c)$ and $F_y'(0,c)$ are specified for each color channel in registers R0x3610 through R0x3630. Values for F_x"(x,c) and F_y"(y,c) are specified in terms of zones. In each horizontal and vertical zone F_x"(x,c), F_y"(y,c) are set to a predefined value for every color channel in registers R0x3632 to R0x3670. This allows the user to set the second derivatives in one zone to a specific value, and to a different value in the next zone, thus producing a piece-wise quadratic term for every color channel.
- G_0 is the global offset constant, which offsets the maximum gain of 1 G_0 is set to "0" for lens shading auto-adjust function) and the value is specified in R0x3674.
- c represents the four channel index—R, G1, G2, B;
- k(x,y,c) is the corner parameter function for each color channel. The location of the corner (up left, up right, low left, and low right) as shown in Figure 27. K factors are independently adjustable at each corner for every color channel and are not dependent on the vertical and the horizontal spatial dimension. A total of 16 K-factors are available for tuning. Each K factor is effective on the corresponding quarter of the image as illustrated in Figure 27. Corner correction factor K is specified in R0x3676 through R0x3694.

Figure 27: Corner Quadrants



- Kpwrx and Kpwry are the corner exponents and introduce more correction gain at the corners. Kpwrx and Kpwry are defined by R0x3600 [12:11] and R0x3600 [14:13].

Corner Factors Switching

Special considerations apply when the corner factors k are switched. In a given color plane c the color factors must be switched at a location where

$$F_x''(x,c) = 0$$
$$F_y''(x,c) = 0$$

If this condition is not met, the corner cross-term value becomes discontinuous as the value of k changes abruptly. This artifact is highly undesirable and can be visible in the output image as colored quadrants. The location where F_x and F_y terms become zero corresponds to the peak of pixel response in the pixel array. The location of the peak can be different for each color plane.

The lens shading algorithm has two modes of switching the corner factors:

a. Automatic corner switching disabled
   When automatic corner switching is disabled the corner factors for all channels switch at the boundaries Bx(c) and By(c). To avoid the artifacts discussed above, the pedestal factor 'a' is subtracted and the corner factors are switched when Fx and Fy must be at or near to zero. The pedestal value is defined by R0x3696 [12:8]. Pedestal a should not be set excessively high as it will reduce the effect of the cross-factor term and can influence the effect of the orthogonal terms F_x and F_y.

Automatic corner switching is disabled by setting R0x3600[0] to zero.

b. Automatic corner switching enabled

When automatic corner switching is enabled the corner factors are switched automatically for each color and direction when Fy and Fx are equal to the gain minimum detection register (Fy = Fx = R0x3696 [7:0]*16). This mode does not require specifying coordinates or the corner factor negative pedestal.

Lens Shading Limitations

- Setting divisors ≥ 4 leads to a loss in correction precision and can introduce a mismatch when mirror and skip modes are enabled.
- Starting coordinates are rounded when skip modes are enabled which may introduce a slight shift in the starting point.

Frame Rate Control

The formula for calculating the frame rate of the MT9T012 are shown below:

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

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

$$\text{frame rate} = \frac{\text{vt\_pixel\_clock\_mhz} / 1\times10^{6}}{\text{line\_length\_pck} \times \text{frame\_length\_lines}}$$

The legal integration times range from zero to frame_length_lines. Therefore, the range of integration times is reduced as the frame rate is increased.

Frame Rates at Common Image Sizes

Table 28 shows the maximum frame rates that can be achieved at various common image sizes. The frame rates are shown both with subsampling disabled (full resolution) and with subsampling enabled (x_odd_inc = 3, y_odd_inc = 3). The frame rates assume a pixel rate of 64 megapixels/sec (vt_pix_clk = 64 MHz).

Table 28: Frame Rates

Image Size	Subsampling Disabled	Subsampling Enabled
QXGA (2,048 x 1,536)	15.10 fps	47.85 fps
UXGA (1,600 x 1,200)	23.06 fps	70.12 fps
SXVGA (1,280 x 960)	33.42 fps	97.65 fps
XGA (1,024 x 768)	47.85 fps	134.11 fps
SVGA (800 x 600)	70.12 fps	187.43 fps
VGA (640 x 480)	97.65 fps	249.98 fps

Integration Time

The integration (exposure) time of the MT9T012 is controlled by the fine_integration_time and coarse_integration_time registers.

The limits for the fine integration time are defined by:

$$\text{fine\_integration\_time\_min} \geq \text{fine\_integration\_time} \geq (\text{line\_length\_pck} - \text{fine\_integration\_time\_max\_margin})$$

The limits for the coarse integration time are defined by:

$$\text{coarse\_integration\_time\_min} \geq \text{coarse\_integration\_time} \geq (\text{frame\_length\_lines} - \text{coarse\_integration\_time\_max\_margin})$$

The actual integration time is given by:

$$\text{integration\_time} = \frac{((\text{coarse\_integration\_time} \times \text{line\_length\_pck}) + \text{fine\_integration\_time})}{\text{vt\_pix\_clk\_freq\_mhz} / 1\times10^{6}}$$

With default settings and a vt_pix_clk of 64 MHz, the maximum integration time is given by:

$$\text{integration\_time} = \frac{(((0x610 - 1) \times 0xAAC) + 0x204)}{64MHz / 1 \times 10^{6}} = 66.216ms$$

It is fundamental to the operation of an electronic rolling shutter (ERS) that it is not possible to set an integration time that is greater than the frame time. Unlike earlier Aptina Imaging parts, setting an integration time that is greater than the frame time does not affect the frame time; the behavior is undefined. On the MT9T012 it is necessary to reprogram the frame time (frame_length_lines) in order to make longer exposure times available. Note that long integration times increase the likelihood of image degradation due to increased accumulation of dark current.

Flash Strobe

The MT9T012 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 28, 29, and 30. The flash and flash_count registers allow the timing of the flash to be changed. The flash can be programmed to fire only once; be 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 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 30. Read-only bit flash[14] is set during frames that are correctly integrated; the state of this bit is shown in the figures below.

Figure 28: Xenon Flash Enabled

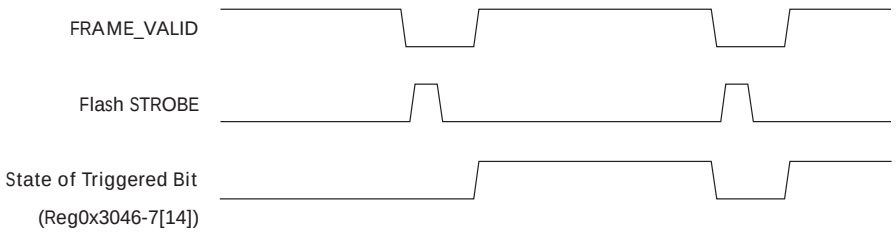
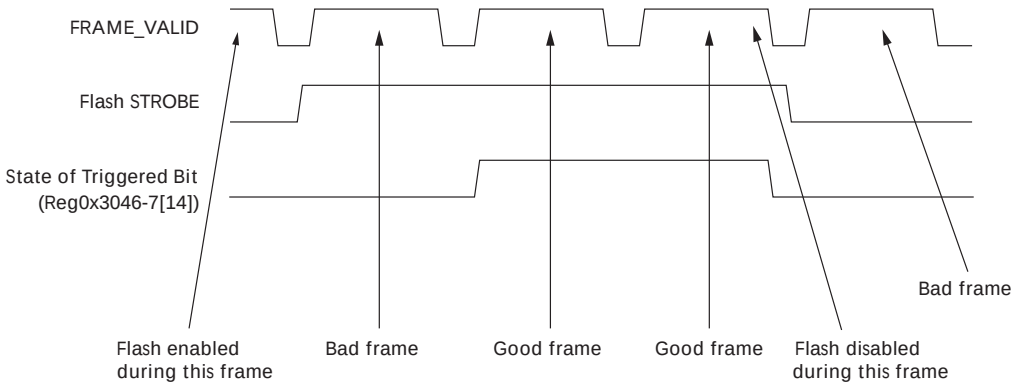
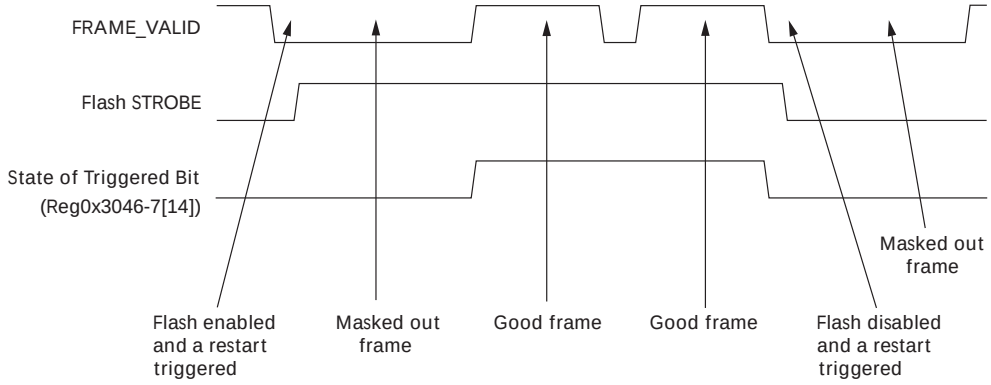


Figure 29: LED Flash Enabled



Note: Integration time = number of rows in a frame.

Figure 30: LED Flash Enabled Following Forced Restart



Global Reset

Global reset mode allows the integration time of the MT9T012 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 design for use in conjunction with the parallel pixel data interface. The SMIA specification only provides for operation in ERS mode. The MT9T012 does support the use of global reset mode in conjunction with the SMIA data path, but there are additional restrictions on its use.

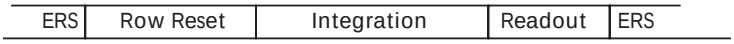
Overview of Global Reset Sequence

The basic elements of the global reset sequence are described below:

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 MT9T012), the electromechanical shutter is closed.
7. A single output frame is generated by the sensor, with the usual LINE_VALID, FRAME_VALID, PIXCLK, DOUT timing. As soon as the output frame has completed (FRAME_VALID negates) the electromechanical shutter may be opened again.
8. The sensor automatically resumes operation in ERS mode.

This sequence is shown in Figure 31. The following sections expand on this figure to show how the timing of this sequence is controlled.

Figure 31: Overview of Global Reset Sequence



Entering and Leaving the Global Reset Sequence

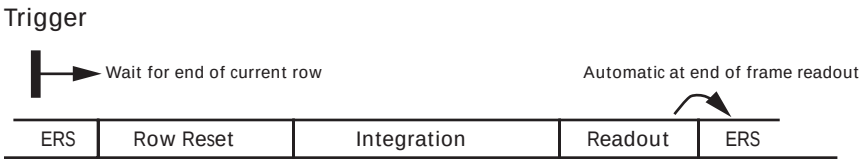
A global reset sequence can be triggered either by a register write to global_seq_trigger[0] (global trigger, to transition this bit from a 0 to a 1) or by a rising edge on a suitably-configured GPI input (See "Trigger Control" on page 29).

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

The global reset sequence completes with a frame readout. At the end of this readout phase, the sensor automatically resumes operation in ERS mode. The first frame integrated with ERS will be generated after a delay of approximately ((13 + coarse_integration_time) * line_length_pck). This sequence is shown in Figure 32.

While operating in ERS mode, double-buffered registers (See "Double-Buffered Registers" on page 2) 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 32: Entering and Leaving a Global Reset Sequence



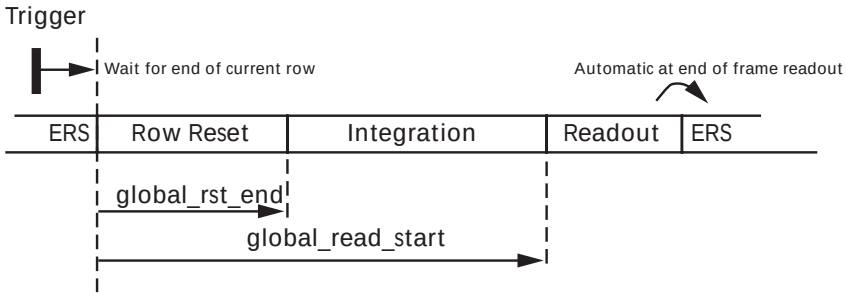
Programmable Settings

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 33. The duration of the readout phase is determined by the active image size.

The recommended setting for global_rst_end is 0xA0 (preliminary). This allows sufficient time for all rows of the pixel array to be set to the correct reset voltage level. The row reset phase takes a finite amount of time due to the capacitance of the pixel array and the capability of the internal voltage booster circuit that is used to generate the reset voltage level.

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

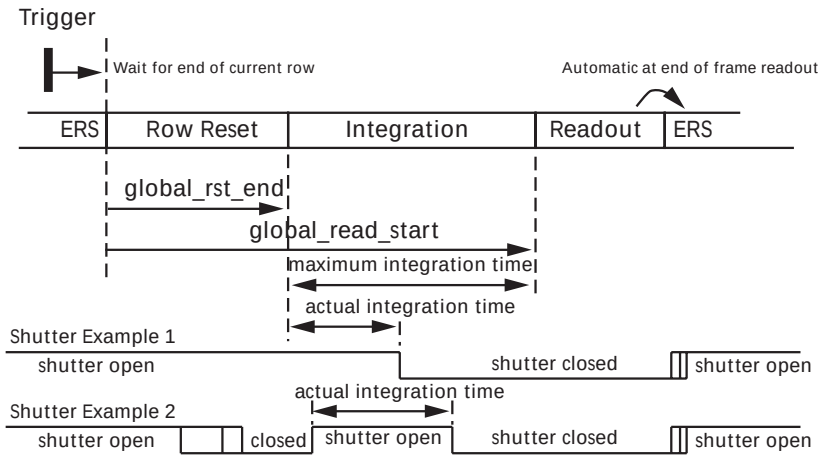
Figure 33: Controlling the Reset and Integration Phases of the Global Reset Sequence



Control of the Electromechanical Shutter

Figure 34 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 some time 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 34: Control of the Electromechanical Shutter



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

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

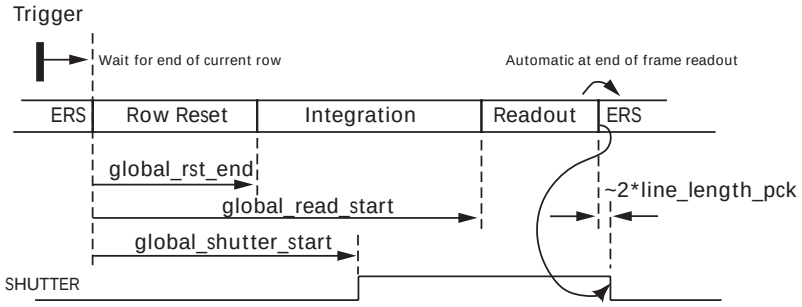
After FRAME_VALID negates to signal the completion of the readout phase, there is a time delay of approximately (10 * line_length_pck) before the sensor starts to integrate light-sensitive rows for the next ERS frame. It is essential that the shutter be opened at some point in this time window, otherwise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

The MT9T012 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 35. SHUTTER is negated by default. The point at which it asserts is controlled by the programming of global_shutter_start. At the end of the global reset readout phase, SHUTTER negates approximately (2 * line_length_pck) after the negation of FRAME_VALID.

The following programming restriction must be met for correct operation:

- global_read_start > global_shutter_start.

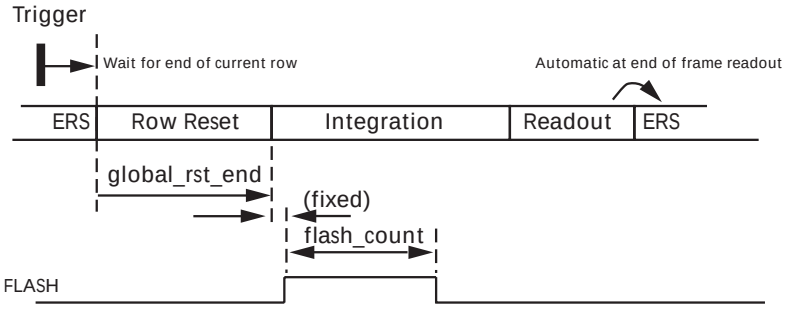
Figure 35: Controlling the SHUTTER Output



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. This is shown in Figure 36.

Figure 36: 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 of 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).

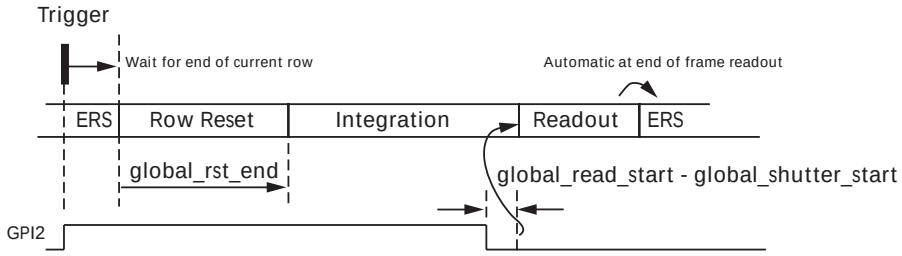
When the trigger is negated, to end integration, the integration phase is extended by a further time given by (global_read_start - global_shutter_start). Usually, this means that global_read_start should be set to (global_shutter_start + 1).

The operation of this mode is shown in Figure 37. 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 following 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 negated).

Figure 37: 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 negated.

The earliest time that the global reset sequence can be retriggered is the point at which the SHUTTER output negates; this occurs approximately (2 * line_length_pck) after the negation of FRAME_VALID 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 42, SMIA Data Path, on page 64) 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, in order to use global reset with the SMIA data path, this timing scenario must be avoided. Two possible ways of doing this are:

1. Synchronize (under software control) the assertion of trigger to an end-of-frame marker on the CCP serial data stream.
2. Alternatively, take the MT9T012 out of streaming mode and wait long enough to guarantee that the frame in progress has completed. Enter streaming mode and trigger the global reset sequence before(coarse_integration_time * line_length_pck) has elapsed (a write to trigger the global reset sequence under software control can occur immediately after the write to mode_select that puts the sensor back into streaming mode).

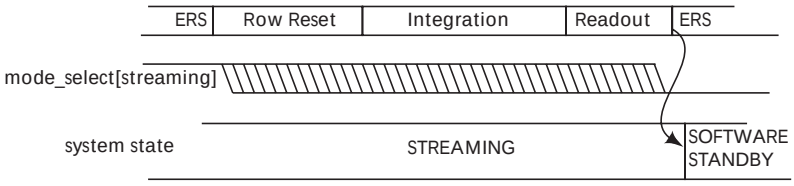
At the end of the readout phase of the global reset sequence, the sensor automatically resumes operation in ERS mode.

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 2 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 MT9T012 will remain in streaming state until the global reset sequence (including frame readout) has completed. This is shown in Figure 38.

Figure 38: Entering Soft Standby during a Global Reset Sequence



Analog Gain

The MT9T012 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, describe the mapping between the models and describe the operation of the per-color and global gain control. Use of high gains may result in reduced image quality due to the amplification of image defects/artifacts.

Using Per-color or Global Gain Control

The read-only analogue_gain_capability register returns a value of 1, indicating that the MT9T012 provides per-color gain control. However, the MT9T012 also provides the option of global gain control. Per-color and global gain control can be used interchange-

ably. 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 green1/greenR gain register.

The read/write gain_mode register required by SMIA has no defined function in the SMIA specification. In the MT9T012 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 the following 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:

$$gain = \frac{analogue\_gain\_m0 \times analogue\_gain\_code}{analogue\_gain\_c1} = \frac{analogue\_gain\_code\_<color>}{8}$$

Aptina Imaging Gain Model

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

- global_gain
- green1_gain
- red_gain
- blue_gain
- green2_gain

This gain model maps directly to the control settings applied to the gain stages of the analog signal chain. This provide a 7-bit gain stage and a 2X gain stage. As a result, the step size varies depending upon whether the 2X gain stage is enabled. The analog gain is given by:

$$gain = (<color>\_gain[7] + 1) \times \frac{<color>\_gain[6:0]}{16}$$

As a result of the 2X gain stage, many of the possible gain settings can be achieved in two different ways. For example, red_gain = 0x01A8 provides the same gain as red_gain = 0x0150. The first example uses the 2X gain stage and the second example does not. In all cases, the preferred setting is the setting that enables the 2X gain stage, since this will result in lower noise. The first uses the 2X stage and is, therefore, preferred. Using the 2X gain stage gives lower noise because it provides gain earlier in the signal chain.

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 gain stage is enabled, in order 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> registers, data read from the associated analogue_gain_code_<color> register is 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, having written gains through one set of registers, those gains should be read back through the same set of registers.

Sensor Core Digital Data Path

Test Patterns

The MT9T012 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 29.

Table 29: 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

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 MT9T012 registers must be set appropriately to control the frame rate and output timing. This includes:

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

Effect of Data Path Processing on Test Patterns

Test patterns 1–3 are introduced early in the pixel data path. As a result, they are affected by pixel processing that occurs within the data path. This includes:

- Noise cancellation
- Black pedestal adjustment

These effects can be eliminated by the following register settings:

- Reg0x3044–5[10] = 0
- Reg0x30CA–B[0] = 0
- Reg0x301A–B[3] = 0 (enable writes to data pedestal)
- Reg0x301E–F = 0x0000 (set data pedestal to “0”)
- Reg=0x0100, 0x0000
- Reg=0x0100, 0x0001
- Reg=0x3044, 0x0140
- Reg=0x301A, 0x0050
- Reg=0x301A, 0x18CC
- Reg=0x301E, 0x0000
- Reg=0x30C0, 0x0081

- Reg=0x30C2, 0x0000
- Reg=0x30C4, 0x0000
- Reg=0x30C6, 0x0000
- Reg=0x30C8, 0x0000
- Reg=0x30CA, 0x0001
- Reg=0x30CC, 0x0000
- Reg=0x30CE, 0x0000
- Reg=0x30D0, 0x0000
- Reg=0x30D2, 0x0000
- Reg=0x318A, 0x0000

Solid Color Test Pattern

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

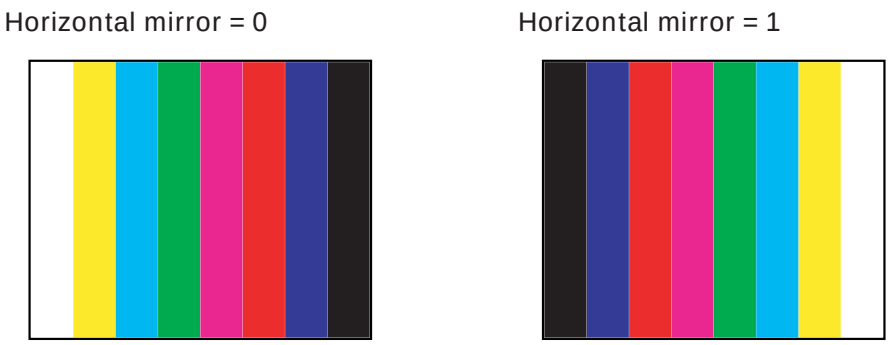
100% Color Bars Test Pattern

In this test pattern, shown in Figure 39, all pixel data is replaced by a Bayer version of an 8-color, color-bar chart (white, yellow, cyan, green, magenta, red, blue, black). Each bar is 256 pixels wide and occupies the full height of the output image. Each color component of each bar is set to either 0 (fully off) or 0x3FF (fully on for 10-bit data). The pattern repeats after 8 * 256 = 2048 pixels. The image size is set by x_addr_start, x_addr_end, y_addr_start, y_addr_end and may be affected by the setting of x_output_size, y_output_size. The color-bar pattern starts at the column identified by x_addr_start. The number of colors that are visible in the output is dependent upon x_addr_end - x_addr_start and the setting of x_output_size. The width of each color-bar is fixed at 256 pixels.

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

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

Figure 39: 100% Color Bars Test Pattern



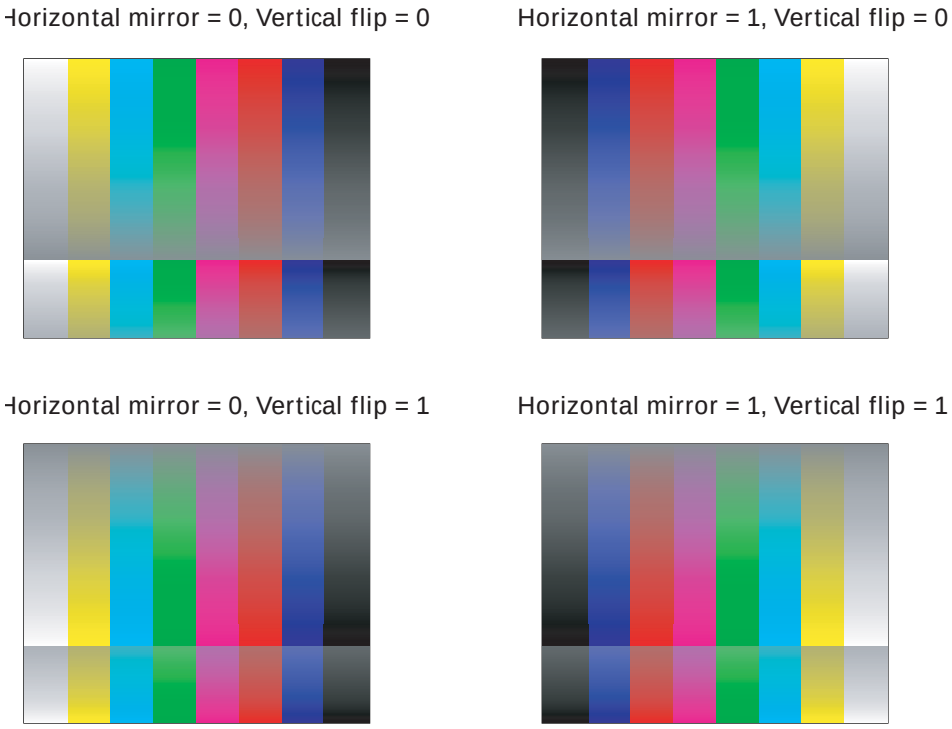
Fade-to-Gray Color Bars Test Pattern

In this test pattern, shown in Figure 40, all pixel data is replaced by a Bayer version of an 8-color, color-bar chart (white, yellow, cyan, green, magenta, red, blue, black). Each bar is 256 pixels wide and occupies 1024 rows of the output image. Each color bar fades vertically from full intensity at the top of the image to 50 percent intensity (mid-gray) on the 256th row. Each color bar is divided into a left and a right half, in which the left half fades smoothly and the right half fades in quantized steps every 8 pixels for a given color. Due to the Bayer pattern of the colors this means that the level changes every 16 rows. The pattern repeats horizontally after 8*256 = 2,048 pixels and vertically after 1,024 rows (Using 10-bit data, the fade-to-gray pattern goes from 100 to 50 percent or from 0 to 50 percent for each color component, so only half of the 2^{10} states of the 10-bit data are used. However, to get all of the gray levels, each state must be held for two rows, hence the vertical size of 2^{10} / 2 * 2 = 1024). The image size is set by x_addr_start, x_addr_end, y_addr_start, y_addr_end and may be affected by the setting of x_output_size, y_output_size. The color-bar pattern starts at the column identified by x_addr_start. The number of colors that are visible in the output is dependent upon x_addr_end - x_addr_start and the setting of x_output_size. The width of each color-bar is fixed at 256 pixels.

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

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

Figure 40: Fade-to-Gray Color Bars Test Pattern



PN9 Link Integrity Pattern

This test pattern provides a 512-bit pseudo-random test sequence to test the integrity of the serial pixel data output stream. The polynomial $x^9 + x^5 + 1$ is used. The polynomial is initialized to 0x1FF at the start of each frame.

When this test pattern is enabled:

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

Before enabling this test pattern the MT9T012 clock divisors must be configured for RAW10 operation (op_pix_clk_div = 10). This polynomial generates the following sequence of 10-bit values, 0x1FF, 0x378, 0x1A1, 0x336, 0x385... On the parallel pixel data output these values are presented 10-bits per PIXCLK. On the serial pixel data output these values are streamed out sequentially without performing the RAW10 packing to bytes that normally occurs on this interface.

Test Cursors

The MT9T012 supports one horizontal and one vertical cursor, allowing a "cross-hair" to be superimposed on the image, or on test patterns 1–3.

The position and width of each cursor is programmable. Only even cursor positions and even cursor widths are supported (this is a consequence of the internal architecture of the pixel array). Each cursor can be inhibited by setting its width to "0."

The programmed cursor position corresponds to an absolute row or column in 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.

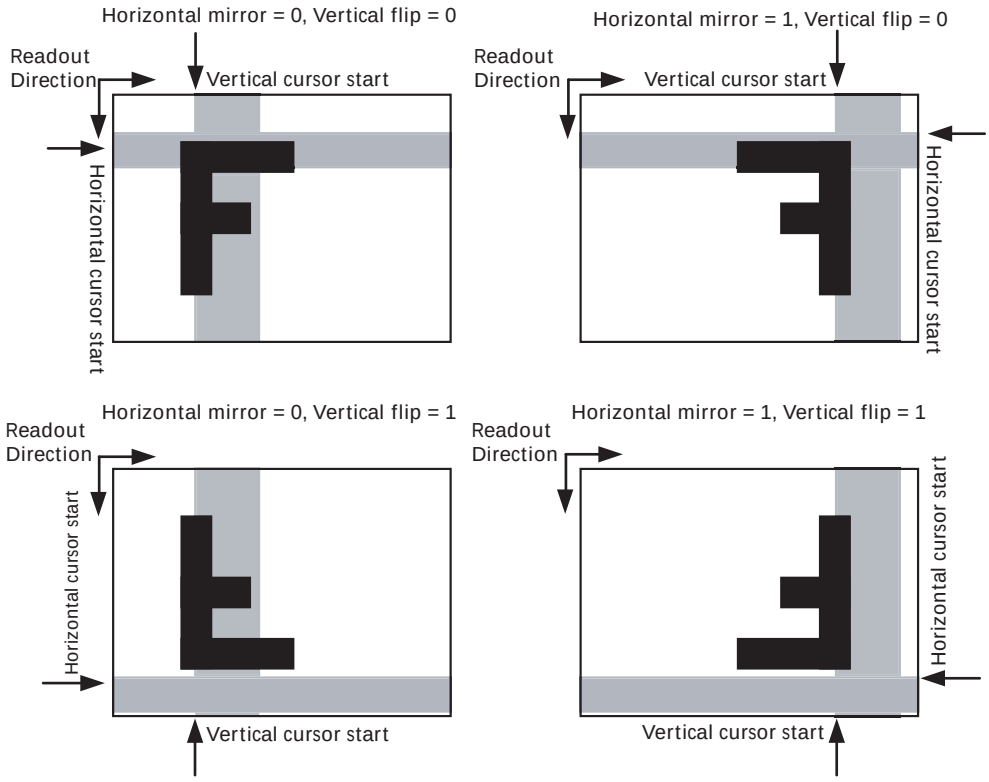
The cursors are opaque (they replace data from the imaged scene or test pattern). The color of each cursor is set by the values of the bayer components in the test_data_red, test_data_greenR, test_data_blue and test_data_greenB registers. As a consequence, the cursors are the same color as test pattern 1 and are therefore invisible when test pattern 1 is selected.

When vertical_cursor_position = 0x0FFF[1], 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 SMIA specification. The behaviour of the MT9T012 is shown in Figure 41 on page 63. In the figure, the test cursors are shown as translucent, for clarity. In practise, 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 the following implementation details:

- The test cursors are inserted early in the data path, so that they correlate to rows and to columns of the physical pixel array (rather than to x and to y coordinates of the output image).
- The drawing of a cursor starts when the pixel array row or column address matches the value of the associated cursor_position register. As a result, the cursor start position remains fixed, relative to the rows and columns of the pixel array, for all settings of image_orientation.
- The cursor generation continues until the appropriate cursor_width pixels have been drawn. The cursor width is generated from the start position and proceeds in the direction of pixel array readout. As a result, each cursor is reflected about an axis corresponding to its start position when the appropriate bit is set in the image_orientation register.

1. As Table 10 shows, vertical_cursor_position is implemented as a 12-bit register. Therefore, bits [15:12] are ignored on writes and return 0 on reads.

Figure 41: Test Cursor Behavior when image_orientation = 0



Digital Gain

Integer digital gains in the range 0–7 can be programmed. A digital gain of 0 sets all pixel values to 0 (the pixel data will simply represent the value applied by the pedestal block).

Note: The MT9T012 supports digital gains of 0–7. However, digital_gain_min advertises a minimum digital gain of 1.0 and digital_gain_max advertises a maximum digital gain of 4.0, which is an error. It should advertise a maximum digital gain of 7.0. Use of high gains, in particular the use of digital gain, may result in reduced image quality due to the amplification of image defects/artifacts.

Note that as gain is increased, image quality degrades due to the amplification of image defects.

Pedestal

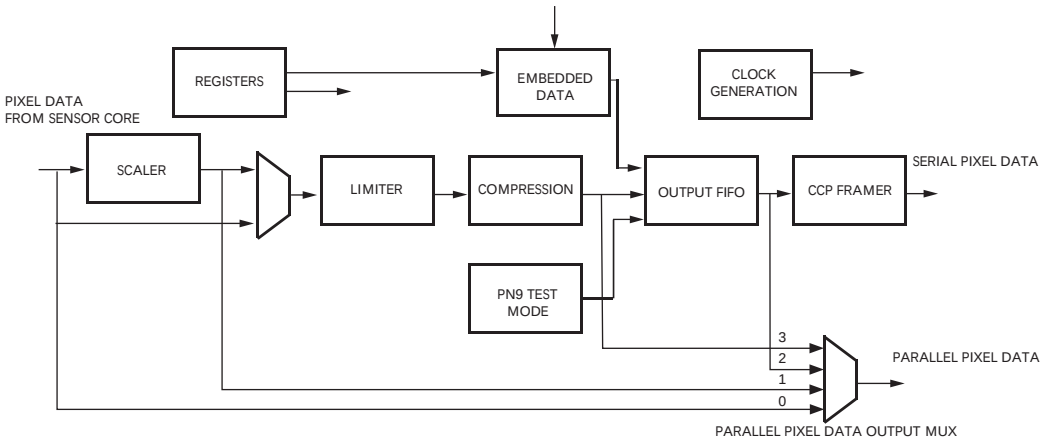
This block adds the value from R0x301E-F[10:0] (data_pedestal_) to the incoming pixel value. The data_pedestal register is read-only by default but can be made read/write by clearing the lock_reg bit in R0x301A–B.

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

SMIA Digital Data Path

The SMIA digital data path is shown in Figure 42.

Figure 42: SMIA 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. In this mode, the first two lines and the last line of data are not equally spaced. The format of this data is shown in Table 30. In the table, 8-bit (RAW8) and 10-bit (RAW10) versions of the data are shown. The 10-bit format places the data byte in bits [9:2] and sets bits [1:0] to a constant value of 01. Register values that are shown as "??" are dynamic and may change from frame to frame.

When the parallel pixel data path is selected and R0x306E-F[2:0]=2 (parallel pixel data output MUX selects FIFO data). The output image contains two rows of embedded data.

Table 30: Embedded Data

Row 0					Row 1			
Offset	10-bit	8-bit	Two-wire Serial Interface Addr	Comment	10-bit	8-bit	Two-wire Serial Interface Addr	Comment
0	0x029	0x0A		2-byte tagged data format (embedded data)	0x029	0x0A		Start of embedded data line
1	0x2a9	0xAA		CCI register index MSB	0x2a9	0xAA		CCI register index MSB
2	0x001	0x00		Address 00xx	0x005	0x01		Address 01xx
3	0x295	0xA5		CCI register index LSB	0x295	0xA5		CCI register index LSB
4	0x001	0x00		Address xx00	0x081	0x20		Address xx20
5	0x169	0x5A		auto increment	0x169	0x5A		auto increment
6	0x059	0x16	0000	model_id hi	0x001	0x00	120	gain mode
7	0x169	0x5A			0x2a9	0xAA		CCI register index MSB
8	0x001	0x00	0001	model_id lo	0x009	0x02		Address 02xx
9	0x169	0x5A			0x295	0xA5		CCI register index LSB
10	0x001	0x00	0002	revision_number	0x001	0x00		Address xx00

Table 30: Embedded Data (Continued)

Row 0					Row 1			
Offset	10-bit	8-bit	Two-wire Serial Interface Addr	Comment	10-bit	8-bit	Two-wire Serial Interface Addr	Comment
11	0x169	0x5A			0x169	0x5A		auto increment
12	0x019	0x06	0003	manufacturer_id	??	??	0200	fine_integration_time hi
13	0x169	0x5A			0x169	0x5A		
14	0x029	0x0A	0004	smia_version	??	??	0201	fine_integration_time lo
15	0x169	0x5A			0x169	0x5A		
16	??	??	0005	frame_count	??	??	0202	coarse_integration_time hi
17	0x169	0x5A			0x169	0x5A		
18	??	??	0006	pixel_order	??	??	0203	coarse_integration_time lo
19	0x2a9	0xAA		CCI register index MSB	0x169	0x5A		
20	0x001	0x00		Address 00xx	??	??	0204	analogue_gain_code_global hi
21	0x295	0xA5		CCI register index LSB	0x169	0x5A		
22	0x021	0x08		Address xx08	??	??	0205	analogue_gain_code_global lo
23	0x169	0x5A		auto increment	0x169	0x5A		
24	??	??	0008	data_pedestal_hi	??	??	0206	analogue_gain_code_greenR hi
25	0x169	0x5A			0x169	0x5A		
26	??	??	0009	data_pedestal lo	??	??	0207	analogue_gain_code_greenR lo
27	0x2a9	0xAA		CCI register index MSB	0x169	0x5A		
28	0x001	0x00		Address 00xx	??	??	0208	analogue_gain_code_red hi
29	0x295	0xA5		CCI register index LSB	0x169	0x5A		
30	0x101	0x40		Address xx40	??	??	0209	analogue_gain_code_red lo
31	0x169	0x5A		auto increment	0x169	0x5A		
32	0x005	0x01	0040	frame_format_model_type	??	??	020a	analogue_gain_code_blue hi
33	0x169	0x5A			0x169	0x5A		
34	0x049	0x12	0041	frame_format_model_subtype	??	??	020b	analogue_gain_code_blue lo
35	0x169	0x5A			0x169	0x5A		
36	??	??	0042	frame_format_descriptor_0 hi	??	??	020c	analogue_gain_code_greenB hi
37	0x169	0x5A			0x169	0x5A		
38	??	??	0043	frame_format_descriptor_0 lo	??	??	020d	analogue_gain_codegreenB lo
39	0x169	0x5A			0x169	0x5A		
40	??	??	0044	frame_format_descriptor_1 hi	??	??	020e	digital_gain_greenR hi
41	0x169	0x5A			0x169	0x5A		
42	??	??	0045	frame_format_descriptor_1 lo	??	??	020f	digital_gain_greenR lo

Table 30: Embedded Data (Continued)

Row 0					Row 1			
Offset	10-bit	8-bit	Two-wire Serial Interface Addr	Comment	10-bit	8-bit	Two-wire Serial Interface Addr	Comment
43	0x169	0x5A			0x169	0x5A		
44	??	??	0046	frame_format_descriptor_2 hi	??	??	0210	digital_gain_red hi
45	0x169	0x5A			0x169	0x5A		
46	??	??	0047	frame_format_descriptor_2 lo	??	??	0211	digital_gain_red lo
47	0x169	0x5A			0x169	0x5A		
48	0x001	0x00	0048	frame_format_descriptor_3 hi	??	??	0212	digital_gain_blue hi
49	0x169	0x5A			0x169	0x5A		
50	0x001	0x00	0049	frame_format_descriptor_3 lo	??	??	0213	digital_gain_blue lo
51	0x169	0x5A			0x169	0x5A		
52	0x001	0x00	004a	frame_format_descriptor_4 hi	??	??	0214	digital_gain_greenB hi
53	0x169	0x5A			0x169	0x5A		
54	0x001	0x00	004b	frame_format_descriptor_4 lo	??	??	0215	digital_gain_greenB lo
55	0x169	0x5A			0x2a9	0xAA		CCI register index MSB
56	0x001	0x00	004c	frame_format_descriptor_5 hi	0x00d	0x03		Address 03xx
57	0x169	0x5A			0x295	0xA5		CCI register index LSB
58	0x001	0x00	004d	frame_format_descriptor_5 lo	0x001	0x00		Address xx00
59	0x169	0x5A			0x169	0x5A		auto increment
60	0x001	0x00	004e	frame_format_descriptor_6 hi	??	??	0300	vt_pix_clk_div hi
61	0x169	0x5A			0x169	0x5A		
62	0x001	0x00	004f	frame_format_descriptor_6 lo	??	??	0301	vt_pix_clk_div lo
63	0x169	0x5A			0x169	0x5A		
64	0x001	0x00	0050	frame_format_descriptor_7 hi	??	??	0302	vt_sys_clk_div hi
65	0x169	0x5A			0x169	0x5A		
66	0x001	0x00	0051	frame_format_descriptor_7 lo	??	??	0303	vt_sys_clk_div lo
67	0x169	0x5A			0x169	0x5A		
68	0x001	0x00	0052	frame_format_descriptor_8 hi	??	??	0304	pre_pll_clk_div hi
69	0x169	0x5A			0x169	0x5A		
70	0x001	0x00	0053	frame_format_descriptor_8 lo	??	??	0305	pre_pll_clk_div lo
71	0x169	0x5A			0x169	0x5A		
72	0x001	0x00	0054	frame_format_descriptor_9 hi	??	??	0306	pll_multiplier_hi
73	0x169	0x5A			0x169	0x5A		
74	0x001	0x00	0055	frame_format_descriptor_9 lo	??	??	0307	pll_multiplier_lo
75	0x169	0x5A			0x169	0x5A		
76	0x001	0x00	0056	frame_format_descriptor_10 hi	??	??	0308	op_pix_clk_div hi
77	0x169	0x5A			0x169	0x5A		
78	0x001	0x00	0057	frame_format_descriptor_10 lo	??	??	0309	op_pix_clk_div lo
79	0x169	0x5A			0x169	0x5A		
80	0x001	0x00	0058	frame_format_descriptor_11 hi	??	??	030a	op_sys_clk_div hi
81	0x169	0x5A			0x169	0x5A		
82	0x001	0x00	0059	frame_format_descriptor_11 lo	??	??	030b	op_sys_clk_div lo
83	0x169	0x5A			0x2a9	0xAA		CCI register index MSB
84	0x001	0x00	005a	frame_format_descriptor_12 hi	0x00d	0x03		Address 03xx
85	0x169	0x5A			0x295	0x0A5		CCI register index LSB

Table 30: Embedded Data (Continued)

Row 0					Row 1			
Offset	10-bit	8-bit	Two-wire Serial Interface Addr	Comment	10-bit	8-bit	Two-wire Serial Interface Addr	Comment
86	0x001	0x00	005b	frame_format_descriptor_12 lo	0x101	0x40		Address xx40
87	0x169	0x5A			0x169	0x5A		auto increment
88	0x001	0x00	005c	frame_format_descriptor_13 hi	??	??	0340	frame_length_lines hi
89	0x169	0x5A			0x169	0x5A		
90	0x001	0x00	005d	frame_format_descriptor_13 lo	??	??	0341	frame_length_lines lo
91	0x169	0x5A			0x169	0x5A		
92	0x001	0x00	005e	frame_format_descriptor_14 hi	??	??	0342	line_length_pck hi
93	0x169	0x5A			0x169	0x5A		
94	0x001	0x00	005f	frame_format_descriptor_14 lo	??	??	0343	line_length_pck lo
95	0x2a9	0xAA		CCI register index MSB	0x169	0x5A		
96	0x001	0x00		Address 00xx	??	??	0344	x_addr_start hi
97	0x295	0xA5		CCI register index LSB	0x169	0x5A		
98	0x201	0x80		Address xx80	??	??	0345	x_addr_start lo
99	0x169	0x5A		auto increment	0x169	0x5A		
100	0x001	0x00	0080	analogue_gain_capability hi	??	??	0346	y_addr_start hi
101	0x169	0x5A			0x169	0x5A		
102	0x005	0x01	0081	analogue_gain_capability lo	??	??	0347	y_addr_start lo
103	0x2a9	0xAA		CCI register index MSB	0x169	0x5A		
104	0x001	0x00		Address 00xx	??	??	0348	x_addr_end hi
105	0x295	0xA5		CCI register index LSB	0x169	0x5A		
106	0x211	0x84		Address xx84	??	??	0349	x_addr_end lo
107	0x169	0x5A		auto increment	0x169	0x5A		
108	0x001	0x00	0084	analogue_gain_code_min hi	??	??	034a	y_addr_end hi
109	0x169	0x5A			0x169	0x5A		
110	0x021	0x08	0085	analogue_gain_code_min lo	??	??	034b	y_addr_end lo
111	0x169	0x5A			0x169	0x5A		
112	0x001	0x00	0086	analogue_gain_code_max hi	??	??	034c	x_output_size hi
113	0x169	0x5A			0x169	0x5A		
114	0x1fd	0x7F	0087	analogue_gain_code_max lo	??	??	034d	x_output_size lo
115	0x169	0x5A			0x169	0x5A		
116	0x001	0x00	0088	analogue_gain_code_step hi	??	??	034e	y_output_size hi
117	0x169	0x5A			0x169	0x5A		
118	0x005	0x01	0089	analogue_gain_code_step lo	??	??	034f	y_output_size lo
119	0x169	0x5A			0x2a9	0xAA		CCI register index MSB
120	0x001	0x00	008a	analogue_gain_type hi	0x00d	0x03		Address 03xx
121	0x169	0x5A			0x295	0xA5		CCI register index LSB
122	0x001	0x00	008b	analogue_gain_type lo	0x201	0x80		Address xx80
123	0x169	0x5A			0x169	0x5A		auto increment
124	0x001	0x00	008c	analogue_gain_m0 lo	??	??	0380	x_even_inc hi
125	0x169	0x5A			0x169	0x5A		
126	0x005	0x01	008d	analogue_gain_m0 lo	??	??	0381	x_even_inc lo
127	0x169	0x5A			0x169	0x5A		
128	0x001	0x00	008e	analogue_gain_c0 lo	??	??	0382	y_odd_inc hi

Table 30: Embedded Data (Continued)

Row 0					Row 1			
Offset	10-bit	8-bit	Two-wire Serial Interface Addr	Comment	10-bit	8-bit	Two-wire Serial Interface Addr	Comment
129	0x169	0x5A			0x169	0x5A		
130	0x001	0x00	008f	analogue_gain_c0 lo	??	??	0383	y_odd_inc lo
131	0x169	0x5A			0x169	0x5A		
132	0x001	0x00	0090	analogue_gain_m1 lo	??	??	0384	y_even_inc hi
133	0x169	0x5A			0x169	0x5A		
134	0x001	0x00	0091	analogue_gain_m1 lo	??	??	0385	y_even_inc lo
135	0x169	0x5A			0x169	0x5A		
136	0x001	0x00	0092	analogue_gain_c1 lo	??	??	0386	x_odd_inc hi
137	0x169	0x5A			0x169	0x5A		
138	0x021	0x08	0093	analogue_gain_c1 lo	??	??	0387	x_odd_inc lo
139	0x2a9	0xAA		CCI register index MSB	0x2a9	0xAA		CCI register index MSB
140	0x001	0x00		Address 00xx	0x011	0x04		Address 04xx
141	0x295	0xA5		CCI register index LSB	0x295	0xA5		CCI register index LSB
142	0x301	0xC0		Address xxC0	0x001	0x00		Address xx00
143	0x169	0x5A		auto increment	0x169	0x5A		auto increment
144	0x005	0x01	00C0	data_format_model_type	??	??	0400	scaling_mode hi
145	0x169	0x5A			0x169	0x5A		
146	0x00d	0x03	00c1	data_format_model_subtype	??	??	0401	scaling_mode lo
147	0x169	0x5A			0x169	0x5A		
148	0x029	0x0A	00c2	data_format_descriptor_0 hi	??	??	0402	spatial_sampling hi
149	0x169	0x5A			0x169	0x5A		
150	0x029	0x0A	00c3	data_format_descriptor_0 lo	??	??	0403	spatial_sampling lo
151	0x169	0x5A			0x169	0x5A		
152	0x021	0x08	00c4	data_format_descriptor_1 hi	??	??	0404	scale_m hi
153	0x169	0x5A			0x169	0x5A		
154	0x021	0x08	00c5	data_format_descriptor_1 lo	??	??	0405	scale_m lo
155	0x169	0x5A			0x169	0x5A		
156	0x029	0x0A	00c6	data_format_descriptor_2 hi	0x001	0x00	0406	scale_n hi
157	0x169	0x5A			0x169	0x5A		
158	0x021	0x08	00c7	data_format_descriptor_2 lo	0x041	0x10	0407	scale_n lo
159	0x169	0x5A			0x2a9	0xAA		CCI register index MSB
160	0x001	0x00	00c8	data_format_descriptor_3 hi	0x015	5		Address 05xx
161	0x169	0x5A			0x295	0xA5		CCI register index LSB
162	0x001	0x00	00c9	data_format_descriptor_3 lo	0x001	0x00		Address xx00
163	0x169	0x5A			0x169	0x5A		auto increment
164	0x001	0x00	00ca	data_format_descriptor_4 hi	0x001	0x00	0500	compression_mode hi
165	0x169	0x5A			0x169	0x5A		
166	0x001	0x00	00cb	data_format_descriptor_4 lo	0x005	0x01	0501	compression_mode lo
167	0x169	0x5A			0x2a9	0xAA		CCI register index MSB
168	0x001	0x00	00cc	data_format_descriptor_5 hi	0x019	0x06		Address 06xx
169	0x169	0x5A			0x295	0xA5		CCI register index LSB
170	0x001	0x00	00cd	data_format_descriptor_5 lo	0x001	0x00		Address xx00
171	0x169	0x5A			0x169	0x5A		auto increment

Table 30: Embedded Data (Continued)

Row 0					Row 1			
Offset	10-bit	8-bit	Two-wire Serial Interface Addr	Comment	10-bit	8-bit	Two-wire Serial Interface Addr	Comment
172	0x001	0x00	00ce	data_format_descriptor_6 hi	??	??	0600	test_pattern_mode hi
173	0x169	0x5A			0x169	0x5A		
174	0x001	0x00	00cf	data_format_descriptor_6 lo	??	??	0601	test_pattern_mode lo
175	0x2a9	0xAA		CCI register index MSB	0x169	0x5A		
176	0x005	0x01		Address 01xx	??	??	0602	test_data_red hi
177	0x295	0xA5		CCI register index LSB	0x169	0x5A		
178	0x001	0x00		Address xx00	??	??	0603	test_data_red lo
179	0x169	0x5A		auto increment	0x169	0x5A		
180	??	??	0100	mode_select	??	??	0604	test_data_greenR hi
181	0x169	??			0x169	0x5A		
182	??	0x00	0101	image_orientation	??	??	0605	test_data_greenR lo
183	0x2a9	0xAA		CCI register index MSB	0x169	0x5A		
184	0x005	0x01		Address 01xx	??	??	0606	test_data_blue hi
185	0x295	0xA5		CCI register index LSB	0x169	0x5A		
186	0x00d	0x03		Address xx03	??	??	0607	test_data_blue lo
187	0x169	0x5A		auto increment	0x169	0x5A		
188	0x001	0x00	0103	software_reset	??	??	0608	test_data_greenB hi
189	0x169	0x5A			0x169	0x5A		
190	??	??	0104	grouped_parameter_hold	??	??	0609	test_data_greenB lo
191	0x169	0x5A			0x169	0x5A		
192	??	??	0105	mask_corrupted_frames	??	??	060a	horizontal_cursor_width hi
193	0x2a9	0xAA		CCI register index MSB	0x169	0x5A		
194	0x005	0x01		Address 01xx	??	??	060b	horizontal_cursor_width lo
195	0x295	0xA5		CCI register index LSB	0x169	0x5A		
196	0x041	0x10		Address xx10	??	??	060c	horizontal_cursor_position hi
197	0x169	0x5A		auto increment	0x169	0x5A		
198	??	??	0110	CCP2_channel_identifier	??	??	060d	horizontal_cursor_position lo
199	0x169	0x5A			0x169	0x5A		
200	??	??	0111	CCP2_signalling_mode	??	??	060e	vertical_cursor_width hi
201	0x169	0x5A			0x169	0x5A		
202	??	??	0112	CCP_data_format hi	??	??	060f	vertical_cursor_width lo
203	0x169	0x5A			0x169	0x5A		
204	??	??	0113	CCP_data_format lo	??	??	0610	vertical_cursor_position hi
205	0x01d	0x07		Null Data	0x169	0x5A		
206	0x01d	0x07		Null Data - up to end-of-line	??	??	0611	vertical_cursor_position lo
207					0x01d	0x07		Null Data
208					0x01d	0x07		Null Data - up to end-of-line

Parallel Pixel Data Output MUX

By default, the output data MUX drives 10-bit data from the sensor core onto the parallel data interface. This MUX can also be used to drive data from different internal data paths of the SMIA. The output datapath selection is controlled by R0x306E-F. Figure 43 on page 70 illustrates differences in signaling schemes used to present the data.

This internal data path control is provided for debug only, not for functional operation.

When the scaler is enabled in the SMIA data path it has the effect of reducing the amount of pixel data. Therefore, for some pixel-times while LINE_VALID is asserted, no valid pixel is available. Internally to the sensor, there is a PIXEL_VALID signal to indicate when pixel data is valid. Usually, the output FIFO re-times the output data stream so that PIXEL_VALID is always asserted while LINE_VALID is asserted. When the parallel pixel data output MUX selects a data path that includes the scaler but excludes the output FIFO, a signalling mechanism is required to make the PIXEL_VALID information available outside the chip.

The signalling mechanism uses the redundant combination FRAME_VALID = 1, LINE_VALID = 1 (this is redundant because LINE_VALID = 1 implies that FRAME_VALID = 1) and uses this redundancy to modify the behavior of FRAME_VALID as follows:

FRAME_VALID'= FRAME_VALID ^ (LINE_VALID & !PIXEL_VALID).

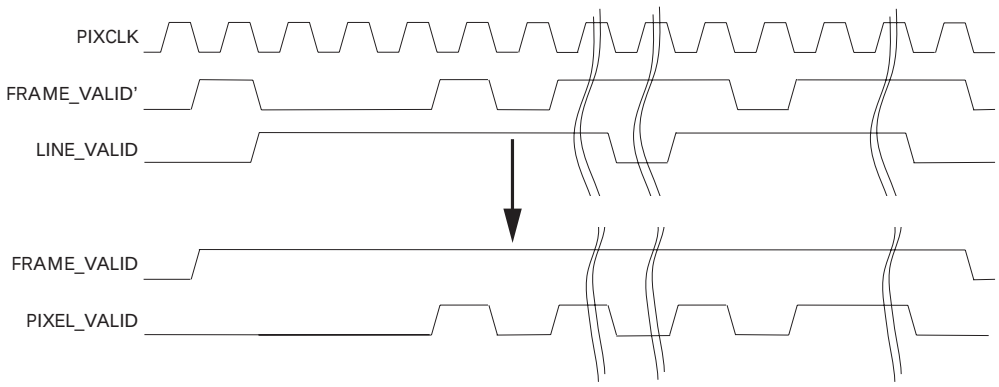
Externally to the sensor, the internal signals can be recovered trivially:

FRAME_VALID = FRAME_VALID' | LINE_VALID

PIXEL_VALID = FRAME_VALID' & LINE_VALID

When the scaler is disabled, pixel data is always valid and therefore FRAME_VALID behaves in the traditional way. The generation of PIXEL_VALID and the regeneration of FRAME_VALID are shown in Figure 43. PIXCLK and LINE_VALID are not affected.

Figure 43: Creating PIXEL_VALID and Recreating FRAME_VALID



The data formats on the parallel pixel data output bus are shown in Table 31.

Table 31: Data Formats on Parallel Pixel Data Output

Data Type	RAW8	RAW10
Pixel data	Data driven on bits [9:2]. Bits [1:0] are "Don't Care."	Data driven on bits [9:0].

Table 31: Data Formats on Parallel Pixel Data Output

Data Type	RAW8	RAW10
Embedded data	Data driven on bits [9:2]. Bits [1:0] = 01	Data driven on bits [9:2]. Bits [1:0] = 01
PN9 test pattern data	Undefined	Data driven on bits [9:0] See "PN9 Link Integrity Pattern" on page 61.

Spectral Characteristics

Figure 44: Quantum Efficiency

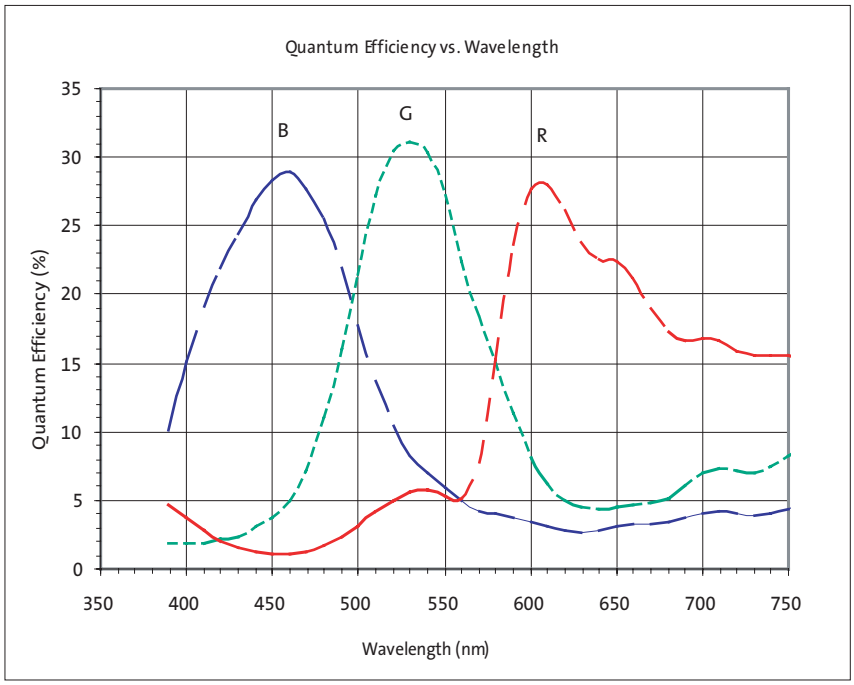


Figure 45: CRA vs. Image Height, Type A

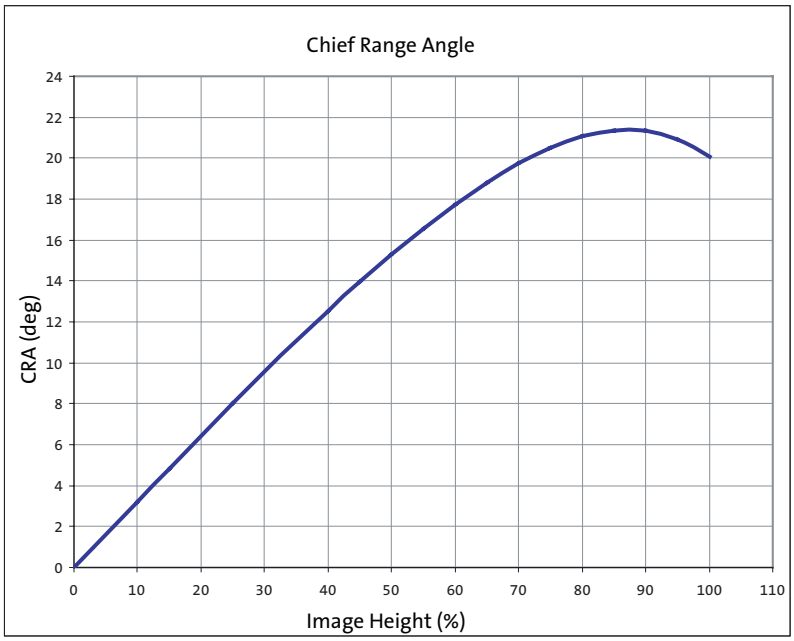
CRA vs. Image Height Plot	Image Height		CRA (Deg)
	(%)	(mm)	
	0	0	0
	5	0.141	1.57
	10	0.282	3.18
	15	0.422	4.81
	20	0.563	6.42
	25	0.704	8.02
	30	0.845	9.57
	35	0.986	11.09
	40	1.126	12.56
	45	1.267	13.96
	50	1.408	15.31
	55	1.549	16.57
	60	1.690	17.75
	65	1.830	18.81
	70	1.971	19.74
	75	2.112	20.50
	80	2.253	21.06
	85	2.394	21.36
	90	2.534	21.35
	95	2.675	20.94
	100	2.816	20.06

Figure 46: CRA vs. Image Height, Type B

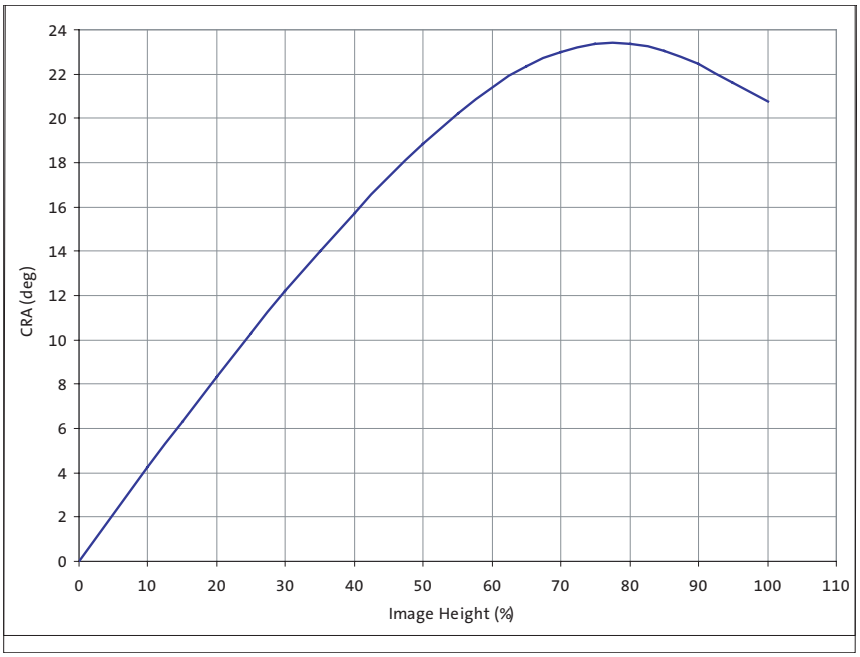
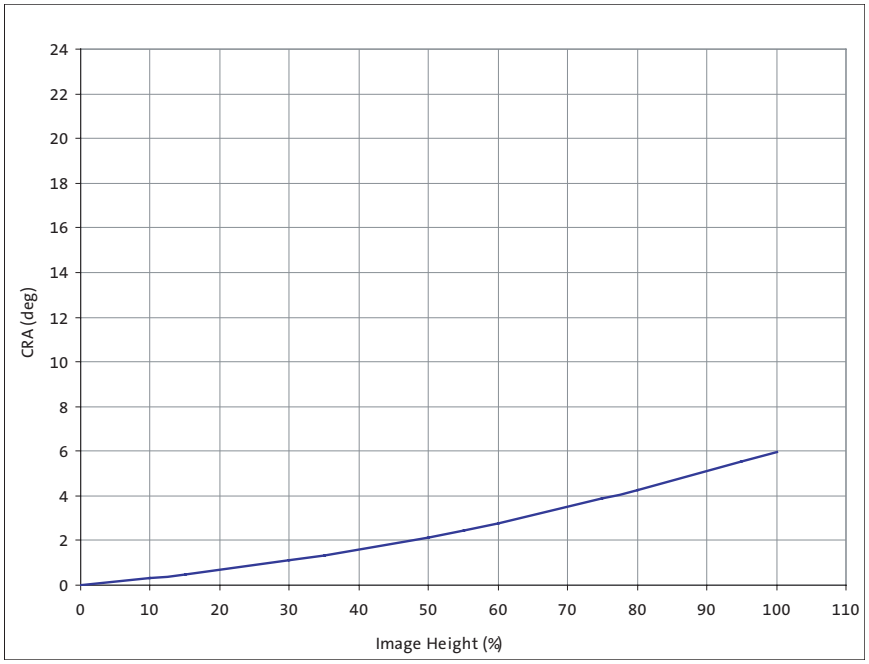
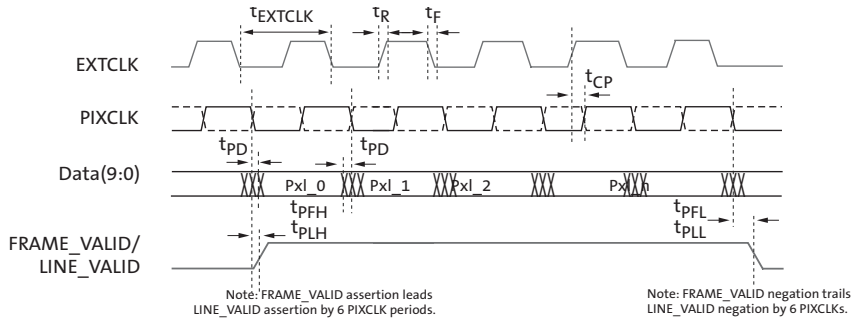
CRA vs. Image Height Plot	Image Height		CRA (Deg)
	(%)	(mm)	
	0	0	0
	5	0.141	2.11
	10	0.282	4.24
	15	0.422	6.33
	20	0.563	8.36
	25	0.704	10.32
	30	0.845	12.20
	35	0.986	14.01
	40	1.126	15.73
	45	1.267	17.36
	50	1.408	18.87
	55	1.549	20.23
	60	1.690	21.40
	65	1.830	22.34
	70	1.971	23.00
	75	2.112	23.35
	80	2.253	23.36
	85	2.394	23.05
	90	2.534	22.44
	95	2.675	21.63
	100	2.816	20.76

Figure 47: CRA vs. Image Height, Type C

CRA vs. Image Height Plot	Image Height (%)	Image Height (mm)	CRA (Deg)
	0	0	0
	5	0.141	0.16
	10	0.282	0.33
	15	0.422	0.50
	20	0.563	0.69
	25	0.704	0.89
	30	0.845	1.10
	35	0.986	1.33
	40	1.126	1.57
	45	1.267	1.84
	50	1.408	2.13
	55	1.549	2.44
	60	1.690	2.77
	65	1.830	3.12
	70	1.971	3.49
	75	2.112	3.87
	80	2.253	4.26
	85	2.394	4.67
	90	2.534	5.09
	95	2.675	5.52
	100	2.816	5.98

Electrical Specifications

Figure 48: Default Data Output Timing Diagram



EXTCLK

The electrical characteristics of the EXTCLK input are shown in Table 33. The EXTCLK input supports either an AC-coupled sine-wave input clock or a DC-coupled square-wave input clock.

If EXTCLK is AC-coupled to the MT9T012 and the clock is stopped, the EXTCLK input to the MT9T012 must be driven to ground or to VDDIO. Failure to do this will result in excessive current consumption within the EXTCLK input receiver.

Table 33: Electrical Characteristics (EXTCLK)

^{f}EXTCLK = 16 MHz; VDD = 1.8V; VDDIO 1.8V; VAA = 2.8V; VAAPIX = 2.8V; VDDPLL = 2.8V; Output load = 68.5pF; Ambient temperature

Definition	Conditions	Symbol	Min	Typ	Max	Units
Input clock frequency	PLL enabled	^{f}EXTCLK1	6	16	27	MHz
Input clock period	PLL enabled	^{t}EXTCLK1	166	62.5	37	ns
Input clock frequency	PLL disabled	^{f}EXTCLK2	6	–	64	MHz
Input clock period	PLL disabled	^{t}EXTCLK2	166	–	15.625	ns
Input clock rise slew rate		^{SR}ICR	0.03	–	1	V/ns
Input clock fall slew rate		^{SR}ICF	0.03	–	1	V/ns
Input clock minimum voltage swing (AC coupled)		VIN_AC	0.5	–	–	V (p-p)
Input clock maximum voltage (DC coupled)		VIN_DC	VDDIO	–	VDDIO + 0.5	V
Input clock signalling frequency (low amplitude)	VIN = VIN_AC (MIN)	^{f}CLKMAX(AC)	–	–	27	MHz
Input clock signalling frequency (full amplitude)	VIN = VDDIO	^{f}CLKMAX(DC)	–	–	64	MHz
Clock duty cycle			45	50	55	%
Input clock jitter		^{t}JITTER	–	–	300	ps
EXTCLK to PIXCLK propagation delay	PLL disabled	^{t}CP	–	15	–	ns
	PLL enabled		–	5	–	
PLL VCO Lock Time	PLL enabled	^{t}LOCK	–	0.1	1	ms
Input pad capacitance		CIN	–	2.5	–	pF

Table 33: Electrical Characteristics (EXTCLK) (Continued)

^{f}EXTCLK = 16 MHz; V_{DD} = 1.8V; V_{DDIO} 1.8V; V_{AA} = 2.8V; V_{AAPIX} = 2.8V; V_{DDPLL} = 2.8V; Output load = 68.5pF; Ambient temperature

Definition	Conditions	Symbol	Min	Typ	Max	Units
Input HIGH leakage current		I_{IH}	–	2.5	10	µA
Input LOW leakage current		I_{IL}	–	2.5	–10	µA
Input HIGH voltage		V_{IH}	0.7 x V_{DDIO}	–	V_{DDIO}+ 0.5	V
Input LOW voltage		V_{IL}	-0.5	–	0.3V x V_{DDIO}	V

Parallel Pixel Data Interface

The electrical characteristics of the Parallel pixel data interface (FRAME_VALID, LINE_VALID, D_{OUT}(9:0), PIXCLK, SHUTTER, FLASH outputs) are shown in Table 34.

Table 34: Electrical Characteristics (Parallel Pixel Data Interface)

^{f}EXTCLK = 16 MHz; V_{DD} = 1.8V; V_{DDIO} 1.8V; V_{AA} = 2.8V; V_{AAPIX} = 2.8V; V_{DDPLL} = 2.8V; Output Load = 68.5pF; Ambient temperature

Definition	Conditions	Symbol	Min	Typ	Max	Units
Output HIGH voltage	At specified I_{OH} 8mA	V_{OH}	V_{DDIO} - 0.5	–	–	V
Output LOW voltage	At specified I_{OL} 8mA	V_{OL}	–	–	0.4	V
Output HIGH current	At specified V_{OH} MIN, V_{DDIO} = 1.8V	I_{OH}	2.7	–	6.4	mA
Output HIGH current	At specified V_{OH} MAX, V_{DDIO} = 1.8V	I_{OH}	8.9	–	22.3	mA
Output LOW current	At specified V_{OL} 0.1V	I_{OL}	2.6	–	5.1	mA
Output LOW current	At specified V_{OL} 0.4V	I_{OL}	8.9	–	18.5	mA
Tri-state output leakage current		I_{OZ}	–	–	1	µA
Output pin slew	default, C_{LOAD} = 30pF		–	0.3818	–	V/ns
PIXCLK frequency	default	^{f}PIXCLK	–	64	64	MHz
PIXCLK to data valid	default	^{t}PD	–	–	3	ns
PIXCLK to FRAME_VALID HIGH	default	^{t}PFH	–	–	3	ns
PIXCLK to LINE_VALID HIGH	default	^{t}PLH	–	–	3	ns
PIXCLK to FRAME_VALID LOW	default	^{t}PFL	–	–	3	ns
PIXCLK to LINE_VALID LOW	default	^{t}PLL	–	–	3	ns
Input pad capacitance		C_{IN}	–	6.5	–	pF
Load capacitance		C_{LOAD}	–	–	N/A	pF

Two-Wire Serial Register Interface

The electrical characteristics of the two-wire serial register interface (SCLK, SDATA) are shown in Table 35. The SCLK and SDATA signals feature fail-safe input protection, schmitt trigger input, and suppression of input pulses of less than 50ns duration.

Table 35: Two-Wire Serial Register Interface Electrical Characteristics

^{f}EXTCLK = 16 MHz; V_{DD} = 1.8V; V_{DDIO} 1.8V; V_{AA} = 2.8V; V_{AAPIX} = 2.8V; V_{DDPLL} = 2.8V; Output load = 68.5pF; Ambient temperature

Definition	Conditions	Symbol	Min	Typ	Max	Units
Input HIGH voltage		V_{IH}	0.7 x V_{DDIO}	–	V_{DDIO} + 0.5	V
Input LOW voltage		V_{IL}	-0.5	–	0.3V x V_{DDIO}	V
Input leakage current	No pull-up resistor; V_{IN} = V_{DDIO} or D_{GND}	I_{IN}	–	10	–	µA
Output HIGH voltage	At specified I_{OH} 8mA	V_{OH}	V_{DDIO} - 0.5	–	–	V
Output LOW voltage	At specified I_{OL} 8mA	V_{OL}	–	–	0.4	V
Output HIGH current	At specified V_{OH} MAX, V_{DDIO} = 1.8V	I_{OH}	8.9	–	22.3	mA
Output LOW current	At specified V_{OL} 0.1V	I_{OL}	2.6	–	5.1	mA
Output LOW current	At specified V_{OL} 0.4V	I_{OL}	8.9	–	18.5	mA
Tri-state output leakage current		I_{OZ}	–	–	1	µA
Input pad capacitance		C_{IN}	–	–	6	pF
Load capacitance		C_{LOAD}	–	–	N/A	pF

Serial Pixel Data Interface

The electrical characteristics of the serial pixel data interface (CLK_P, CLK_N, DATA_P, DATA_N) are shown in Table 36.

In order to operate the serial pixel data interface within the electrical limits of the CCP2 specification, V_{DDIO} (I/O digital voltage) is restricted to operate in the range 1.7–1.9V.

Table 36: Electrical Characteristics (Serial Pixel Data Interface)

^{f}EXTCLK = 16 MHz; V_{DD} = 1.8V; V_{DDIO} 1.8V; V_{AA} = 2.8V; V_{AAPIX} = 2.8V; V_{DDPLL} = 2.8V; Output load = 5pF; Ambient temperature

Definition	Conditions	Symbol	Min	Typ	Max	Units
Operating frequency			1	–	320	MHz
Fixed common mode voltage		V_{CMF}	0.8	0.9	1	V
Differential voltage swing		Vod	100	155	200	mV
Drive current range			0.83	1.5	2	mA
Drive current variation			–	–	15	%
Output impedance			40	58	140	W
Output impedance mismatch			–	4	10	%
Clock duty cycle at 416 MHz			45	50	55	%
Rise time (20–80%)		Vod	300	330	400	ps
Fall time (20–80%)		Vod	280	300	470	ps
Differential skew			–	300	500	ps
Channel-to-channel skew			–	–	200	ps

Table 36: Electrical Characteristics (Serial Pixel Data Interface) (Continued)

[f]EXTCLK = 16 MHz; V_{DD} = 1.8V; V_{DDIO} 1.8V; V_{AA} = 2.8V; V_{AAPIX} = 2.8V; V_{DDPLL} = 2.8V; Output load = 5pF; Ambient temperature

Definition	Conditions	Symbol	Min	Typ	Max	Units
Maximum data rate Data/strobe mode Data/clock mode			–	–	 640 208	Mb/s
Power supply rejection ratio (PSRR) 0–100 MHz			30	–	–	dB
Power supply rejection ratio (PSRR) 100–1,000 MHz			10	–	–	dB

Control Interface

The electrical characteristics of the control interface (RESET_BAR, SADDR, TEST, GPI0, GPI1, GPI2, GPI3) are shown in Table 37.

Table 37: AC Electrical Characteristics (Control Interface)

[f]EXTCLK = 16 MHz; V_{DD} = 1.8V; V_{DDIO} 1.8V; V_{AA} = 2.8V; V_{AAPIX} = 2.8V; V_{DDPLL} = 2.8V; Output load = 68.5pF; Ambient temperature

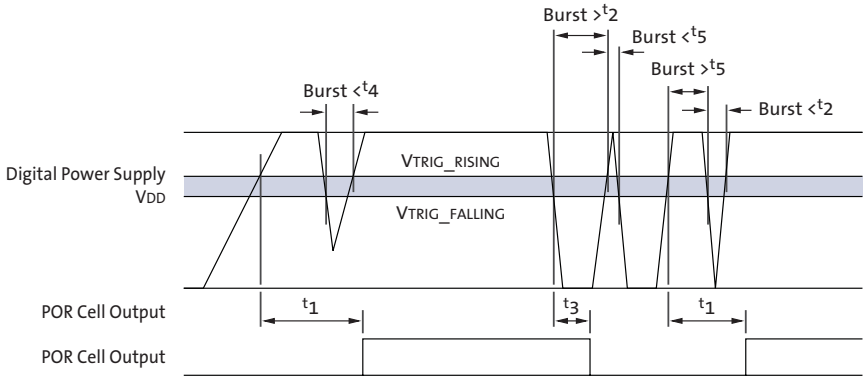
Definition	Conditions	Symbol	Min	Typ	Max	Units
Input HIGH voltage		V_{IH}	0.7 x V_{DDIO}	–	V_{DDIO} +0.5	V
Input LOW voltage		V_{IL}	-0.3	–	0.3 x V_{DDIO}	V
Input leakage current	No pull-up resistor; V_{IN} = V_{DDIO} or D_{GND}	I_{IN}	–	<10	–	µA
Input pad capacitance		C_{IN}	–	6.5	–	pF

Power-on Reset

Table 38: Power-On Reset Characteristics

Definition	Condition	Symbol	Min	Typ	Max	Units
V_{DD} rising, crossing V_{TRIG_RISING}; Internal reset being released		t_1	7	10	15	µs
V_{DD} falling, crossing $V_{TRIG\_FALLING}$; Internal reset active		t_2	–	0.5	1	µs
Minimum V_{DD} spike width below $V_{TRIG\_FALLING}$; considered to be a reset when POR cell output is HIGH		t_3	–	0.5	–	
Minimum V_{DD} spike width below $V_{TRIG\_FALLING}$; considered to be a reset when POR cell output is LOW		t_4	–	1	–	µs
Minimum V_{DD} spike width above V_{TRIG_RISING}; considered to be a stable supply when POR cell output is LOW	While the POR cell output is LOW, all V_{DD} spikes above V_{TRIG_RISING} less than t_5 must be ignored	t_5	–	50	–	ns
V_{DD} rising trigger voltage		V_{TRIG_RISING}	1.12	–	1.55	V
V_{DD} falling trigger voltage		$V_{TRIG\_FALLING}$	1.0	–	1.45	V

Figure 49: Internal Power-On Reset



Operating Voltages

VAA and VAAPIX must be at the same potential for correct operation of the MT9T012.

Table 39: DC Electrical Definitions and Characteristics

^{f}EXTCLKIN = 16 MHz; VDD = 1.8V; VDDIO 1.8V; VAA = 2.8V; VAAPIX = 2.8V; VDDPLL = 2.8V; Output Load = 68.5pF; Ambient temperature; 0 Lux on sensor.

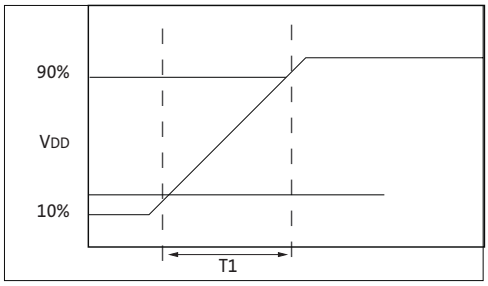
Definition	Conditions	Symbol	Min	Typ	Max	Units
Core digital voltage		VDD	1.7	1.8	1.9	V
I/O digital voltage	Parallel pixel data interface	VDDIO	1.7	1.8/2.8	3.1	V
I/O digital voltage	Serial pixel (CCP2) data interface	VDDIO	1.7	1.8	1.9	V
Analog voltage		VAA	2.4	2.8	3.1	V
Pixel supply voltage		VAAPIX	2.4	2.8	3.1	V
PLL supply voltage		VDDPLL	2.4	2.8	3.1	V
Digital operating current	Parallel pixel data interface, SMIA	IDD1	–	20	35	mA
I/O digital operating current	Parallel pixel data interface, SMIA	IDDQ1	–	15	20	mA
Analog operating current	Parallel pixel data interface, SMIA	IAA1	–	45	55	mA
Pixel supply current	Parallel pixel data interface, SMIA	IAAPIX1	–	1	1.5	mA
PLL supply current	Parallel pixel data interface, SMIA	IDDPLL1	–	5	7	mA
Digital operating current	Parallel pixel data interface, Non-SMIA	IDD2	–	9	35	mA
I/O digital operating current	Parallel pixel data interface, Non-SMIA	IDDQ2	–	15	20	mA
Analog operating current	Parallel pixel data interface, Non-SMIA	IAA2	–	45	55	mA
Pixel supply current	Parallel pixel data interface, Non-SMIA	IAAPIX2	–	1	1.5	mA
PLL supply current	Parallel pixel data interface, Non-SMIA	IDDPLL2	–	5	7	mA
Digital operating current	Serial pixel data interface, CCP 10 Class 1/2	IDD3	–	20	35	mA
I/O digital operating current	Serial pixel data interface, CCP 10 Class 1/2	IDDQ3	–	8	10	mA
Analog operating current	Serial pixel data interface, CCP 10 Class 1/2	IAA3	–	45	55	mA
Pixel supply current	Serial pixel data interface, CCP 10 Class 1/2	IAAPIX3	–	1	1.5	mA
PLL supply current	Serial pixel data interface, CCP 10 Class 1/2	IDDPLL3	–	5	7	mA

Table 39: DC Electrical Definitions and Characteristics (Continued)

[f]EXTCLKIN = 16 MHz; V_{DD} = 1.8V; V_{DDIO} 1.8V; V_{AA} = 2.8V; VAAPIX = 2.8V; V_{DDPLL} = 2.8V; Output Load = 68.5pF; Ambient temperature; 0 Lux on sensor.

Definition	Conditions	Symbol	Min	Typ	Max	Units
Hard standby (clock off)	Analog		0	–	10	µA
	Digital		20	–	150	µA
Hard standby (clock on (6 MHz)	Analog		10	–	50	µA
	Digital		200	–	300	µA
Soft standby (clock off)	Analog (VCO on)		500	–	700	µA
	Digital (VCO on)		200	–	400	µA
	Analog (VCO power down)		0	–	20	µA
	Digital (VCO power down)		50	–	250	µA
Soft standby (clock on 6 MHz))	Analog (VCO on)		0.6	–	2	mA
	Digital (VCO on)		0.25	–	1.2	mA
	Analog (VCO power down)		10	–	50	µA
	Digital (VCO power down)		150	–	600	µA
T1	VDD ramp time		50	–	500	µs

Figure 50: V_{DD} Ramp Time Waveform



Absolute Maximum Ratings

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

Table 40: Absolute Maximum Values

Definition	Conditions	Symbol	Min	Typ	Max	Units
Core digital voltage		V_{DD_MAX}	-0.3	–	2.2	V
I/O digital voltage		V_{DDIO_MAX}	-0.3	–	3.1	V
Analog voltage		V_{AA_MAX}	-0.3	–	3.1	V
Pixel supply voltage		V_{AAPIX_MAX}	-0.3	–	3.1	V
PLL supply voltage		V_{DDPLL_MAX}	-0.3	–	3.1	V
Input HIGH voltage		V_{IH_MAX}	0.7 x V_{DDIO}	–	V_{DDIO} + 0.5	V
Input LOW voltage		V_{IL_MAX}	–0.3	–	0.3 x V_{DDIO}	V
Digital operating current	Worst case current	I_{DD_MAX}	–	–	40	mA
I/O digital operating current	Worst case current	I_{DDQ_MAX}	–	–	25	mA
Analog operating current	Worst case current	I_{AA_MAX}	–	–	70	mA
Pixel supply current	Worst case current	I_{AAPIX_MAX}	–	–	3	mA

Table 40: Absolute Maximum Values (Continued)

Definition	Conditions	Symbol	Min	Typ	Max	Units
PLL supply current	Worst case current	I_{DDPLL_MAX}	–	–	8	mA
Operating temperature	Measure at Junction	T_{OP}[2]	–30	–	70	°C
Storage temperature		T_{STG}[1]	–40	–	125	°C

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

SMIA Specification Reference

The part itself and this documentation is based on the following SMIA reference documents:

- 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 ECR0002 (Version 1.0 dated 11-Feb-2005)

Revision History

Rev. N 2/12

- Updated trademarks
- Applied new Aptina template

Rev. M. 10/10

- Updated fonts of diagrams
- Restored micron symbol that got corrupted in last update
- Updated hyperlink on last page

Rev. L 2/10

- Updated to Aptina template

Rev. K 4/07

- Update VDDQ to VDDIO
- Update RESET# to RESET_BAR
- Update Table 7, “Register List and Default Values: SMIA Configuration,” on page 5
- Update Table 8, “Register List and Default Values: SMIA Parameter Limits,” on page 7
- Update Table 9, “Register List and Default Values: Manufacturer-Specific,” on page 9
- Update Table 10, “SMIA Configuration Register Descriptions,” on page 14
- Update Table 11, “SMIA Parameter Limits Register Descriptions,” on page 20
- Update Table 12, “Manufacturer-Specific Register Descriptions,” on page 25
- Update Table 26, “Register Adjustments Required for Binning Mode,” on page 43
- Add Figure 47: “CRA vs. Image Height, Type C,” on page 75

Rev. J. 11/06

- Update Figure 45: “CRA vs. Image Height, Type A,” on page 73

Rev. I. 10/06

- Update Table 9, “Register List and Default Values: Manufacturer Specific,” on page 26
- Update "Micron Imaging Gain Model" on page 103
- Update "Effect of Data Path Processing on Test Patterns" on page 58
- Remove Table 32, “Chief Ray Angle,” on page 116
- Replace Figure 45: “CRA vs. Image Height, Type A,” on page 73
- Add Figure 46: “CRA vs. Image Height, Type B,” on page 74
- Update Table 33, “Electrical Characteristics (EXTCLK),” on page 76
- Update Table 40, “Absolute Maximum Values,” on page 81

Rev. H 7/06

- Update Table 1, “Key Performance Parameters,” on page 1
- Update Table 4, “Row Timing,” on page 12
- Update "Programming Restrictions" on page 18
- Update "Changes to Gain Settings" on page 4
- Update Table 12, “Manufacturer Specific,” on page 43
- Remove Valid Data Signal Options and LINE_VALID Formats figure
- Update "Programming Examples" on page 33
- Update Table 23, “Programming Example 3,” on page 35
- Update Table 24, “Programming Example 4,” on page 35
- Update "Lens Correction Zones" on page 45

- Update Figure 26: “Definition of Zone Boundaries,” on page 45
- Update Figure 27: “Corner Quadrants,” on page 46
- Update "Corner Factors Switching" on page 47
- Update "Effect of Data Path Processing on Test Patterns" on page 58
- Update Table 33, “Electrical Characteristics (EXTCLK),” on page 76
- Update Table 34, “Electrical Characteristics (Parallel Pixel Data Interface),” on page 77
- Update Table 35, “Two-Wire Serial Register Interface Electrical Characteristics,” on page 78
- Update Table 36, “Electrical Characteristics (Serial Pixel Data Interface),” on page 78

Rev. G 4/06

- Update "Features" on page 1
- Update Table 2, “Available Part Numbers,” on page 1
- Update Table 7, “Register List and Default Values: SMIA Configuration,” on page 22
- Update Table 8, “Register List and Default Values: SMIA Parameter Limits,” on page 7
- Update Table 9, “Register List and Default Values: Manufacturer Specific,” on page 26
- Update Table 10, “SMIA Configuration Register Descriptions,” on page 32
- Update Table 11, “SMIA Parameter Limits Register Description,” on page 38
- Update Table 12, “Manufacturer Specific,” on page 43
- Update Figure 15: “MT9T012 System States,” on page 26
- Update Table 15, “RESET_BAR and PLL in System States,” on page 27
- Update "Influence of ccp2_signalling_mode" on page 32
- Update "PLL VCO Power-Down" on page 32
- Add "Lens Shading Correction (LC)" on page 45
- Add Figure 26: “Definition of Zone Boundaries,” on page 45
- Add Figure 27: “Corner Quadrants,” on page 46
- Update "Micron Imaging Gain Model" on page 103
- Update Table 30, “Embedded Data,” on page 64

Rev. F 3/06

- Update Figure 2 on page 8
- Update Figure 3 on page 9
- Update Table 7, “Register List and Default Value,” on page 22
- Update Table 30, “Embedded Data,” on page 64
- Update Table 15, “RESET_BAR and PLL in System States,” on page 27
- Update Figure 15: “MT9T012 System States,” on page 26
- Update "Micron Imaging Gain Model" on page 103
- Update "Influence of ccp2_signalling_mode" on page 32
- Update "PLL VCO Power-Down" on page 32
- Add "Lens Shading Correction (LC)" on page 45

Rev. E 11/05

- Update Table 1, “Key Performance Parameters,” on page 1
- Add Note 4 to Figure 2, Typical Configuration: Serial Output Mode, on page 8
- Add Note 5 to Figure 3, Typical Configuration: Parallel Output Mode, on page 9
- Update Table 8, “Configuration Registers Description,” on page 30 (bIT 0X0008-9 data_pedestal)
- Update "Changing Registers While Streaming" on page 23
- Update "Integration Time" on page 48

- Update "Analog Gain" on page 55
- Update "Micron Imaging Gain Model" on page 103
- Update "Digital Gain" on page 63
- Update "Embedded Data Format and Control" on page 64
- Update Table 30, “Embedded Data,” on page 64
- Add Figure 45, Chief Ray Angle (CRA), on page 115
- Add Table 32, “Chief Ray Angle,” on page 124
- Add Figure 50, Vdd Ramp Time Waveform, on page 81
- Update Table 39, “DC Electrical Definitions and Characteristics,” on page 80
- Update Table 40, “Absolute Maximum Values,” on page 81

Rev. D, Preliminary 9/05

- Remove ADVANCE disclaimer
- Update Table 1, “Key Performance Parameters,” on page 1
- Add Table 2, “Available Part Numbers,” on page 1
- Remove Pixel Data Format section
- Update Table 8, “Configuration Registers Description,” on page 30
- Update Table 9, “Parameter Limit Registers Description,” on page 37
- Update Table 10, “Manufacturer Specific Registers Description,” on page 42
- Update Table 6, “Programming Rules,” on page 18
- Remove Table 14, page 60
- Update "Programming Restrictions when Subsampling" on page 40
- Update "Programming Restrictions when Binning" on page 42
- Update "Fade-to-Gray Color Bars Test Pattern" on page 60
- Add Figure 44, Quantum Efficiency, on page 72
- Add Figure 49, Internal Power-On Reset, on page 80
- Add "Operating Voltages" on page 80
- Update Table 40, “Absolute Maximum Values,” on page 81

Rev. C, Advance 8/05

- Updated AC/DC electrical specifications in "Electrical Specifications" on page 76
- New material, reorganization of sections

Rev. B, Advance 6/05

- Updated AC/DC electrical specifications in "Electrical Specifications" on page 76
- New material, reorganization of sections

Rev. A, Advance 10/04

- Initial release

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