Aptina 1/4-Inch 2Mp CMOS Digital Image Sensor

MT9D012

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

Features

- DigitalClarity® CMOS imaging technology
- 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 CCP2 compliant sub-low-voltage differential signalling (sub-LVDS)
- On-chip phase-locked loop (PLL) oscillator
- Bayer-pattern down-size scaler
- Integrated color/lens shading correction
- Superior low-light performance

Applications

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

General Description

The Aptina Imaging MT9D012 is a 1/4-inch UXGA-format CMOS active-pixel digital image sensor with a pixel array of 1600H x 1200V (1616H x 1216V 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.

The MT9D012 digital image sensor features DigitalClarity—Aptina's breakthrough low-noise CMOS imaging technology that achieves near-CCD image quality (based on signal-to-noise ratio and low-light sensitivity) while maintaining the inherent size, cost, and integration advantages of CMOS.

Table 1: Key Performance Parameters

Parameter		Value
Optical format		1/4-inch UXGA (4:3)
Active imager size		3.55mm(H) x 2.68mm(V) 4.45mm diagonal
Active pixels		1616H x 1216V
Pixel size		2.2 x 2.2µm
Color filter array		RGB Bayer pattern
Shutter type		Electronic rolling shutter (ERS)
Maximum data rate/ master clock		64 Mp/s at 64 MHz PIXCLK
Frame rate	UXGA (1600 x 1200)	Programmable up to 23 fps
	VGA (640 x 480)	Programmable up to 60 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 (2.80V nominal)
	Digital	1.70–1.90V (1.80V nominal)
	I/O	1.70–1.90V (serial) 1.70–3.10V (parallel)
Power consumption		185mW, 23 fps (full resolution)
Operating temperature		–30°C to +70°C
Packaging		Bare die

When operated in its default mode, the sensor generates a UXGA image at 23 frames per second (fps). An on-chip analog-to-digital converter (ADC) generates a 10-bit value for each pixel.

Ordering Information

Table 2: Available Part Numbers

Part Number	Description
MT9D012D00STCZK	Bare die

Table of Contents

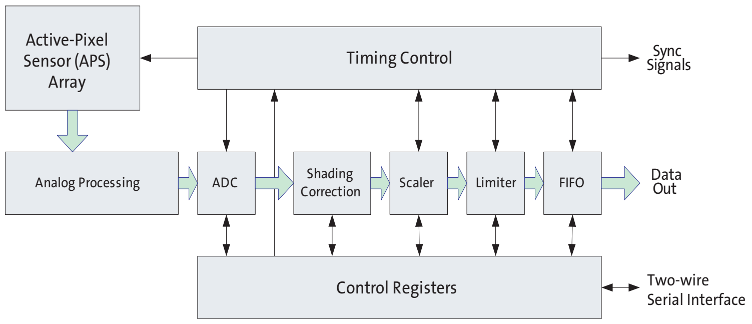
List of Figures

List of Tables

Functional Overview

The MT9D012 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. A block diagram of the sensor is shown in Figure 1.

Figure 1: Block Diagram



The core of the sensor is a 2Mp active-pixel array. The timing and control circuitry sequences through the rows of the array, resetting and then reading each row in turn. In the time interval between resetting a row and reading that row, the pixels in the row integrate incident light. The exposure is controlled by varying the time interval between reset and readout. Once a row has been read, the data from the columns is sequenced through an analog signal chain (providing offset correction and gain), and then through an ADC. The output from the ADC is a 10-bit value for each pixel in the array. The ADC output passes through a digital processing signal chain (that 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 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 three logical parts:

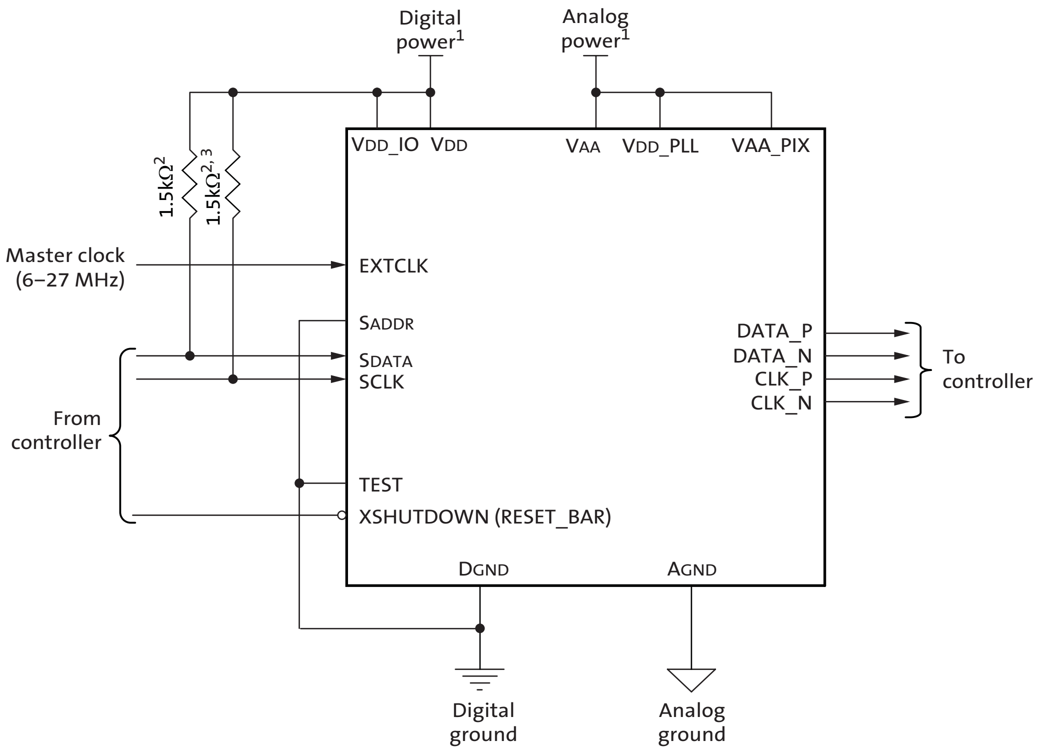
1. A sensor core that provides array control and data path corrections. The output of the sensor core is a 10-bit parallel pixel data stream qualified by an output data clock (PIXCLK), together with LINE_VALID (LV) and FRAME_VALID (FV) signals.
2. A digital shading correction block to compensate for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

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

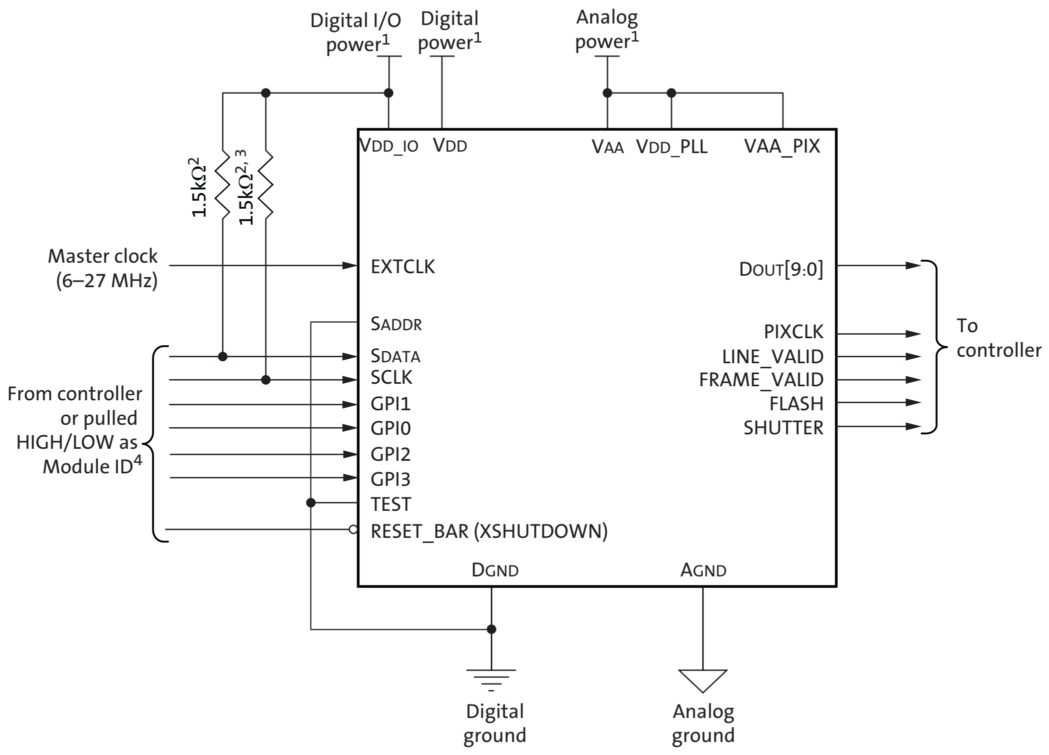
By default, the MT9D012 powers up as a SMIA-compatible sensor with the serial pixel data interface enabled. A typical configuration in this mode is shown in Figure 2. The MT9D012 can also be configured to operate with a parallel pixel data interface. A typical configuration in this mode is shown in Figure 3 on page 9. These two operating modes are described in "Control of the Signal Interface" on page 25.

Figure 2: Typical Configuration: Serial Pixel Data Interface



Notes:
1. All power supplies should be adequately decoupled.
2. Aptina recommends a resistor value of 1.5kΩ, but a greater value may be used for slower two-wire speed.
3. This pull-up resistor is not required if the controller drives a valid logic level on SCLK at all times.
4. The GPI pins must be statically pulled HIGH or LOW while using only the MT9D01200STCZK sensor. These pins can be used as module IDs, or they can be programmed to perform special functions (TRIGGER, OE_N, STANDBY) to be dynamically controlled.

Figure 3: Typical Configuration: Parallel Pixel Data Interface



Notes:
1. All power supplies should be adequately decoupled.
2. Aptina recommends a resistor value of 1.5kΩ, but a greater value may be used for slower two-wire speed.
3. This pull-up resistor is not required if the controller drives a valid logic level on SCLK at all times.
4. The GPI pins must be statically pulled HIGH or LOW while using only the MT9D01200STCZK sensor. These pins can be used as module IDs, or they can be programmed to perform special functions (TRIGGER, OE_N, STANDBY) to be dynamically controlled.

Camera Module Integrator Identification Procedure

Aptina has incorporated a method that enables the GPI to identify the camera module integrator, allowing for the loading of register settings optimized for specific camera module integrators.

The identification is obtained from four GPIs that are wire-bonded to a LOW or HIGH logic level. The sensor displays the GPI values through a register when configured as an input. The camera module integrators are responsible for implementing the correct identification code (as recommended by their end customers).

The identification code is accessible using a two-wire serial interface. Specific registers need to be set to enable a READ through the two-wire serial interface. Reading the results of a different register provides the binary response to the bias applied to the respective GPI.

The GPI pins can be pulled HIGH or LOW to be used as module IDs. Users will first bond these to the appropriate HIGH or LOW state, then enable R0x301A [bit 8] by setting its value to "1" (if not enabled by default), that will enable the GPI and the status read-back from R0x3026–7[3:0].

When the input buffer is enabled, it can be read through R0x3026–7 and R0x0002. When enabled, the user must bond these pins to a HIGH or LOW state to avoid excessive power consumption. Any of these pads can be configured to provide hardware control of the standby, output enable, and shutter trigger functions.

- MT9D01200STC: After reset, these pads are powered down by default; 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.
- MT9D01200STCZK: After reset, these pads are powered up by default; it is necessary to bond to these pads to a HIGH or LOW state.

Caution **Failure to do so will result in excessive power consumption.**

For the MT9D01200STCZ, it is not necessary to write to any registers to enable read-back of the GPI pins. The revision number bits shown in Table 3 apply to the MT9D01200STCZK sensor; its GPI pins must be statically pulled HIGH or LOW.

Table 3: Register 0x0002

Revision Number Bit	Maps To
7	GPI3
6	Silicon revision bit 3
5	Silicon revision bit 2
4	Silicon revision bit 1
3	Silicon revision bit 0
2	GPI2
1	GPI1
0	GPI0

- By default, the MT9D01200STCZK sensor responds as a SMIA Profile 0 sensor.
- MT9D01200STCZK maps the GPI pins (module ID_) and silicon revision into R0x0002.
- The MT9D01200STCZK capability registers show available functionality for Profile 0 behavior. The capability registers will be static, in accordance with the SMIA Functional Specification.

Signal Descriptions

Table 4 provides signal descriptions for MT9D012 die. For pad location and aperture information, refer to the MT9D012 die data sheet.

Table 4: Signal Descriptions

Signal Name	Signal Type	Description
EXTCLK	Input	Master clock input. PLL input clock. 6–27 MHz.
RESET_BAR (XSHUTDOWN)	Input	Asynchronous active LOW reset. When RESET_BAR is 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 SADDR is 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. For the MT9D01200STCZK, after reset, these pads are powered up (enabled—see R0x301A) by default; these pads must be bonded to a HIGH or LOW state. Any of these pads can be configured to provide hardware control of the standby, output enable, and shutter trigger functions. Failure to bond as required will result in excessive power consumption.
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 (LV) output. Qualified by PIXCLK.
FRAME_VALID	Output	FRAME_VALID (FV) output. Qualified by PIXCLK.
DOUT[9:0]	Output	Parallel pixel data output. Qualified by PIXCLK.
PIXCLK	Output	Pixel clock. Used to qualify the LV, FV, and DOUT(9:0) outputs.
FLASH	Output	Flash output. Synchronization pulse for external light source.
SHUTTER	Output	Control for external mechanical shutter.
VAA1, VAA2, VAA3, VAA4	Supply	Analog power supply.
VAA_PIX1, VAA_PIX2, VAA_PIX3	Supply	Analog power supply for the pixel array.
AGND1, AGND2, AGND3, AGND4	Supply	Analog ground.
VDD1, VDD2, VDD3, VDD4	Supply	Digital power supply.
VDD_IO1, VDD_IO2, VDD_IO3, VDD_IO4	Supply	I/O power supply.
DGND1, DGND2, DGND3, DGND4, DGND5	Supply	Common ground for digital and I/O.
VDD_PLL	Supply	PLL power supply.

Output Data Format

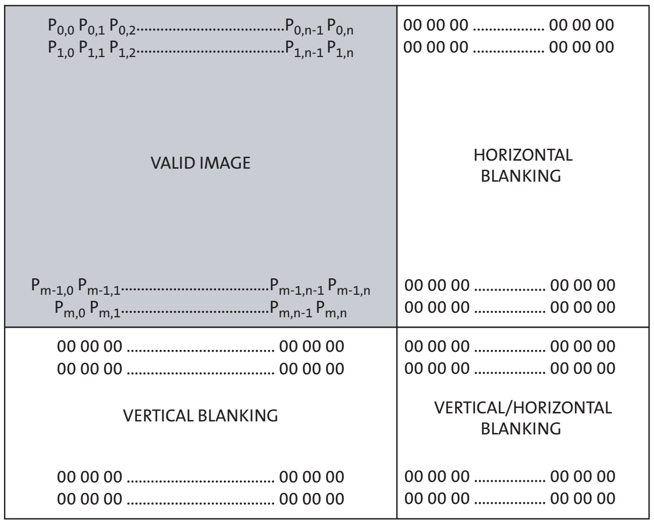
CCP Serial Pixel Data Interface

The MT9D012 serial pixel data interface implements data/clock and data/strobe signalling in accordance with the CCP2 specification. Only RAW10 is supported in profile 0 mode, that is the sensor default on the MT9D01200STCZK. RAW8 is supported in profile 1/2 mode, that can be enabled by setting R0x306E–F[7] = 1.

Parallel Pixel Data Interface

MT9D012 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; LV is HIGH during the shaded region of the figure. FV timing is described in the next section.

Figure 4: Spatial Illustration of Image Readout


$P_{0,0}$ $P_{0,1}$ $P_{0,2}$..................$P_{0,n-1}$ $P_{0,n}$
$P_{1,0}$ $P_{1,1}$ $P_{1,2}$..................$P_{1,n-1}$ $P_{1,n}$
VALID IMAGE
$P_{m-1,0}$ $P_{m-1,1}$..................$P_{m-1,n-1}$ $P_{m-1,n}$
$P_{m,0}$ $P_{m,1}$..................$P_{m,n-1}$ $P_{m,n}$
00 00 00 .... 00 00 00
HORIZONTAL BLANKING
VERTICAL BLANKING
VERTICAL/HORIZONTAL BLANKING

Output Data Timing (Parallel Pixel Data Interface)

MT9D012 output data is synchronized with the PIXCLK output. When LV is HIGH, one pixel value is output on the 10-bit DOUT output every PIXCLK period. 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 LV, FV, and DOUT (see Figure 5). This allows PIXCLK to be used as a clock to sample the data. PIXCLK is continuously enabled, even during the blanking period. The MT9D012 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.

Figure 5: Pixel Data Timing Example

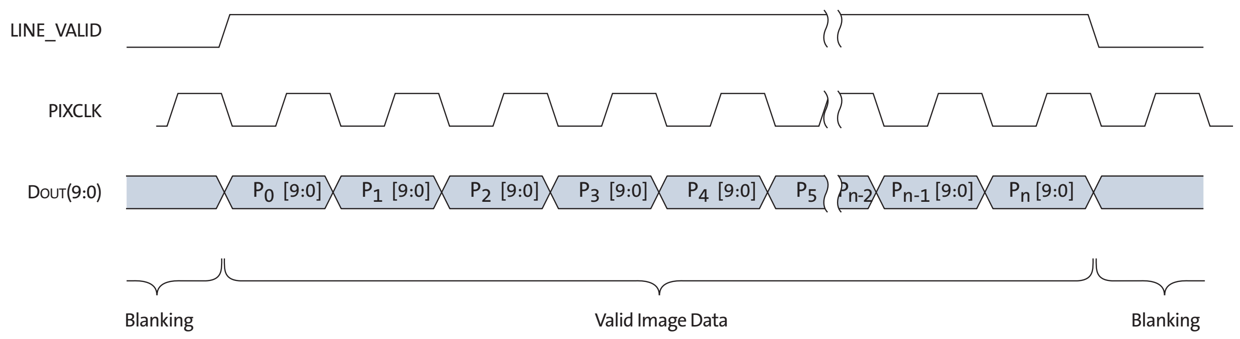
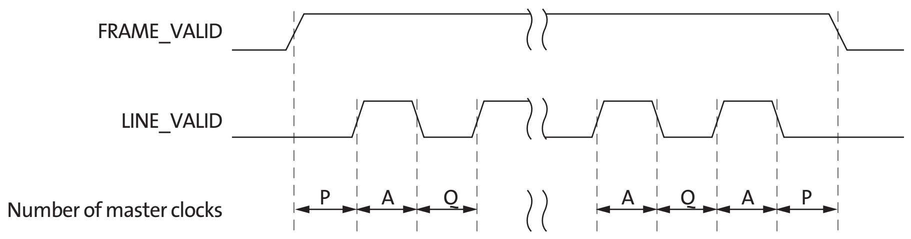


Figure 6: Row Timing and FV/LV Signals



The sensor timing (Table 5 on page 14) is shown in terms of pixel clock and master clock cycles (Figure 5 on page 13). The default settings for the on-chip PLL generate a 64 MHz pixel clock given a 16 MHz input clock to the MT9D012. Equations for calculating the frame rate are given in "Frame Rate Control" on page 47.

Table 5: 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	1600 pixel clocks = 25.00µs
P	Frame start/end blanking	6 * PIXCLK_PERIOD	6 pixel clocks = 93.75ns
Q	Horizontal blanking	(line_length_pck - A) * PIXCLK_PERIOD	2256 - 1600 pixel clocks = 10.250µs
A + Q	Row time	line_length_pck * PIXCLK_PERIOD	2256 pixel clocks = 35.25µs
V	Vertical blanking	([frame_length_lines - N] * [A+Q]) + Q - (2*P)	48020 pixel clocks = 750.3125µs
N	Number of rows	(y_addr_end – y_addr_start + 1) / S	1200 rows
T	Frame valid time	(N * (A + Q)) - Q + (2*P)	2706556 pixel clocks = 42.2899ms
F	Total frame time	line_length_pck * frame_length_lines * PIXCLK_PERIOD	2754576 pixel clocks = 43.04025ms

Two-Wire Serial Register Interface

The two-wire serial interface bus enables read/write access to control and status registers within the MT9D012. This interface is designed to be compatible with the *SMIA 1.0 Part2: CCP2 Specification* camera control interface (CCI) that 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 tied 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 MT9D012 uses SCLK as an input only and therefore never drives it LOW.

Protocol

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

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

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

Start Condition

A start condition is defined as a HIGH-to-LOW transition on SDATA while SCLK is HIGH. At the end of a transfer, the master can generate a start condition without previously generating a stop condition, and 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 MT9D012 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.

An alternate slave address can be programmed by means of R0x31FC.

Message Byte

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

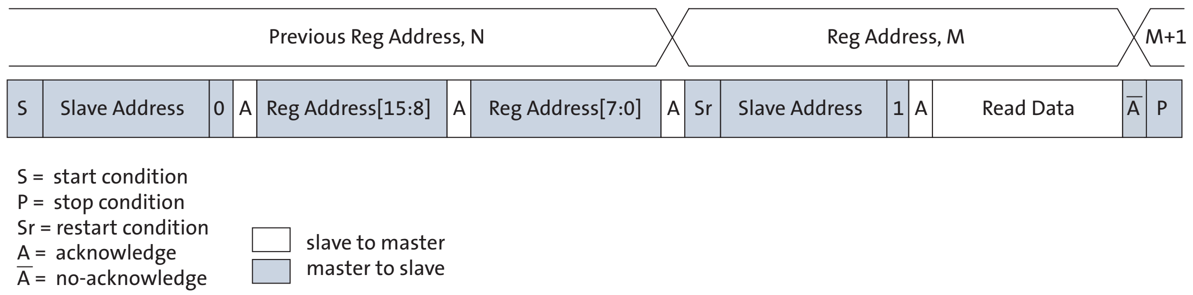
If the request was a WRITE, the master then transfers the 16-bit register address to which a write should take place. The transfer takes place as two 8-bit sequences and the slave sends an acknowledge bit after each sequence to indicate that the byte has been received. The master then transfers the data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

If the request was a READ, the master sends the 8-bit write slave address/data direction byte and 16-bit register address, just as in 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 automatically incremented after every 8 bits are transferred. The data transfer is stopped when the master sends a no-acknowledge bit.

Single READ from Random Location

This sequence (Figure 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 MT9D012 is loaded and incremented as the sequence proceeds.

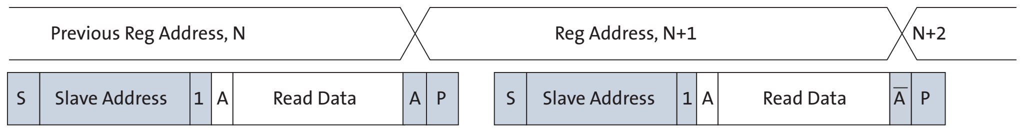
Figure 7: Single READ from Random Location



Single READ from Current Location

This sequence (Figure 8) performs a read using the current value of the MT9D012 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.

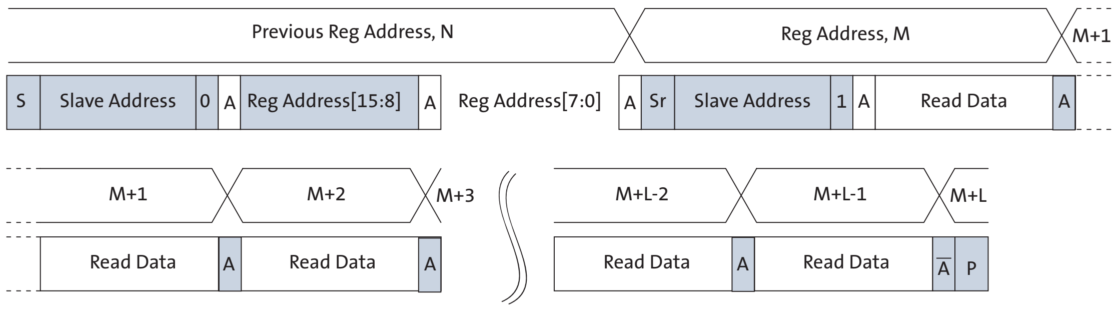
Figure 8: Single READ from Current Location



Sequential READ, Start from Random Location

This sequence (Figure 9) 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.

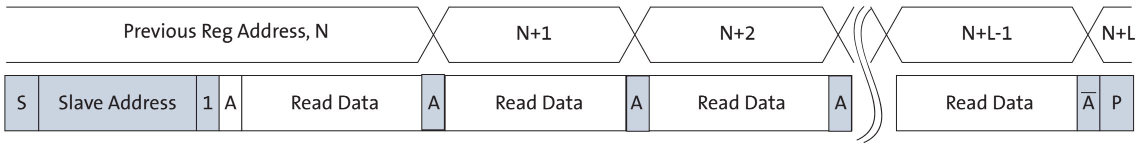
Figure 9: Sequential READ, Start from Random Location



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 17). 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.

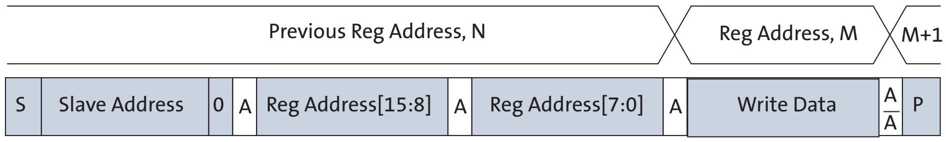
Figure 10: Sequential READ, Start from Current Location



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.

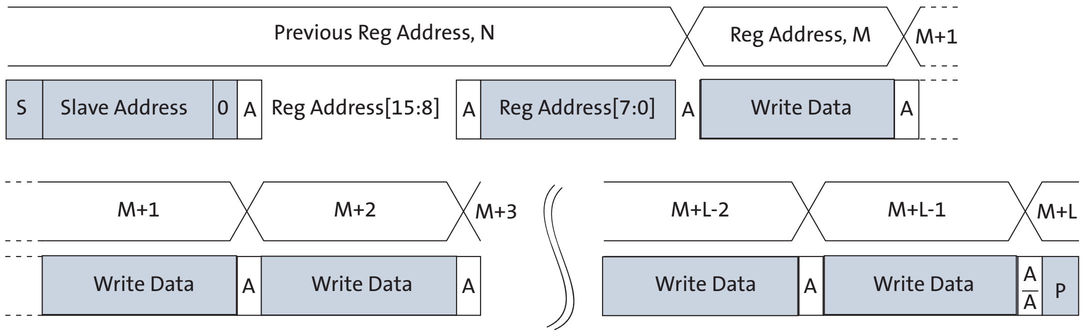
Figure 11: Single WRITE to Random Location



Sequential WRITE, Start at Random Location

This sequence (Figure 12 on page 18) starts in the same way as the single WRITE to random location (Figure 11). 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 MT9D012. It is recommended that these rules are encoded into the device driver stack—implicitly or explicitly.

Table 6: 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 7: 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 ((x_addr_end - x_addr_start + 1)/ xskip) + min_line_blanking_pck	max_line_length_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 7: 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)	1616+TBD	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)	1616	SMIA FS Section 5.2.2.2
y_output_size	2	frame_length_lines	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)	1216	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 41.

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 4 (x_output_size[1:0] = 0).
- When ccp_data_format[7:0] = 10 (RAW10 data), x_output_size must be a multiple of 16 (x_output_size[3:0] = 0).

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.

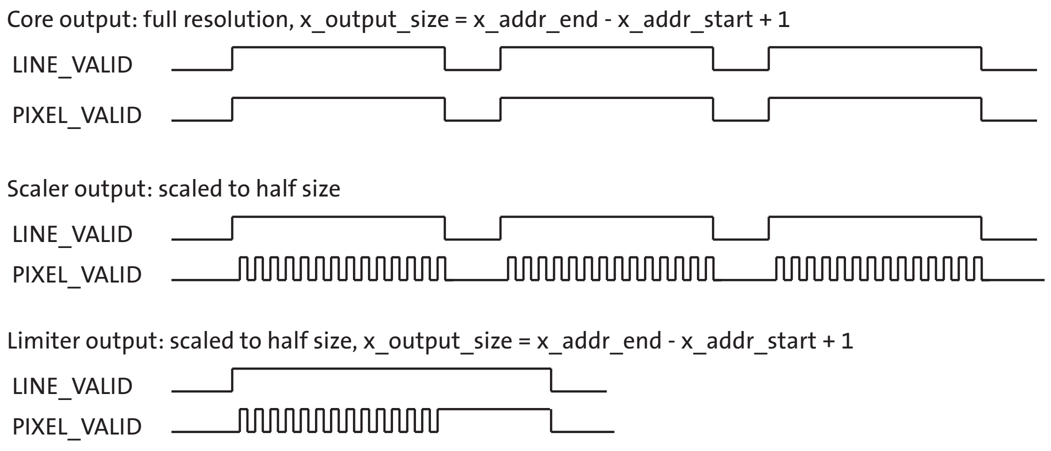
When the parallel pixel data path is in use, the only restriction on x_output_size is that it must be even (x_output_size[0] = 0), and this restriction is enforced in hardware.

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 MT9D012 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


Core output: full resolution, x_output_size = x_addr_end - x_addr_start + 1
LINE_VALID
PIXEL_VALID
Scaler output: scaled to half size
LINE_VALID
PIXEL_VALID
Limiter output: scaled to half size, x_output_size = x_addr_end - x_addr_start + 1
LINE_VALID
PIXEL_VALID

In Figure 13, three different stages in the SMIA data path (see "Digital Data Path" on page 62) are shown. The first stage is the output of the sensor core. The core is running at full resolution and x_output_size is set to match the active array size. The LV signal is asserted once per row and remains asserted for *N* pixel times. The PIXEL_VALID signal toggles with the same timing as LV, indicating that all pixels in the row are valid.

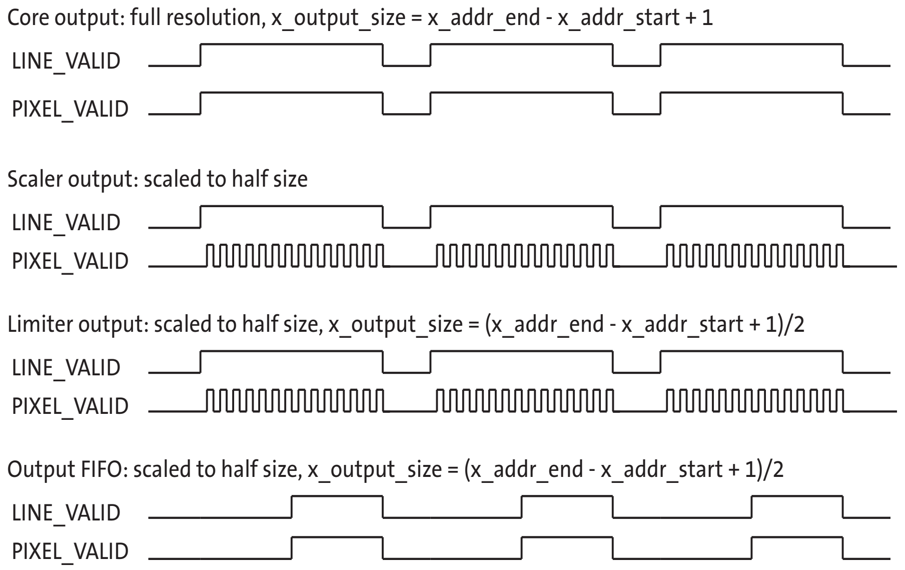
The second stage is the output of the scaler, when the scaler is set to reduce the image size by one-half in each dimension. The effect of the scaler is to combine groups of pixels. Therefore, the row time remains the same, but only half the pixels out of the scaler are valid. This is signalled by transitions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

The third stage is the output of the limiter when the x_output_size is still set to match the active array size. Because the scaler has reduced the amount of valid pixel data without reducing the row time, the limiter attempts to pad the row with $(N/2)$ additional pixels. If this has the effect of extending LV across the whole of the horizontal blanking time, as shown in Figure 13 on page 21, the MT9D012 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 LV.

Figure 14 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


Core output: full resolution, x_output_size = x_addr_end - x_addr_start + 1
LINE_VALID
PIXEL_VALID
Scaler output: scaled to half size
LINE_VALID
PIXEL_VALID
Limiter output: scaled to half size, x_output_size = (x_addr_end - x_addr_start + 1)/2
LINE_VALID
PIXEL_VALID
Output FIFO: scaled to half size, x_output_size = (x_addr_end - x_addr_start + 1)/2
LINE_VALID
PIXEL_VALID

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 2256 * 1221 = 2754576 pixel periods. This value includes vertical and horizontal blanking times so that the full-size image 1600 x 1202 (1200 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 2754576/64e6 = 43.04025ms, giving rise to a frame rate of 23.23 fps.

Each pixel is 10 bits, by default. This data rate requires the serial interface to transmit 2754576 * 10 bits in 43.04025ms. 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 208 Mb/s. Therefore, it is not possible to transmit full-resolution images at 23 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 allow 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 2,754,576/20.8e6 = 0.1324 second, giving rise to a frame rate of 7.55 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.

Caution **If this constraint is not met, the FIFO will 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 soft standby:

- ccp2_channel_identifier
- ccp2_signalling_mode
- ccp_data_format
- scale_m
- vt_pix_clk_div
- vt_sys_clk_div
- pre_pll_clk_div
- pll_multiplier
- op_pix_clk_div
- op_sys_clk_div
- Profile 0/1,2 selection

Programming Restrictions when Using Global Reset

Interactions between the registers that control the global reset impose some programming restrictions on the way that 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, 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 15.

Default Power-Up State

The MT9D012 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 MT9D012 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 (R0x301B[12] = 1) or when the sensor is in the soft standby state.

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

When the serial pixel data interface is used, the LV, FV, PIXCLK, and DOUT[9:0] signals can be left unconnected.

Parallel Pixel Data Interface

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

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

The parallel pixel data interface is disabled by default at power up and after reset. It can be enabled by programming R0x301A–B. Table 9 shows the recommended settings.

When the parallel pixel data interface is in use, the DATA_P, DATA_N, CLK_P, and 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 under pin or register control, as shown in Table 8. Selection of a pin to use for the OE_N function is described in "General Purpose Inputs" on page 29.

Table 8: Output Enable Control

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

Configuration of the Pixel Data Interface

Fields in R0x301A–B are used to configure the operation of the pixel data interface. The supported combinations are shown in Table 9.

Table 9: 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 MT9D012 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 10 on page 28.

The sensor's operation is broken down into three separate states: Hardware Standby, Soft Standby, and Streaming. The transition between these states might take a certain amount of clock cycles as outlined in Table 10.

Figure 15: MT9D012 System States

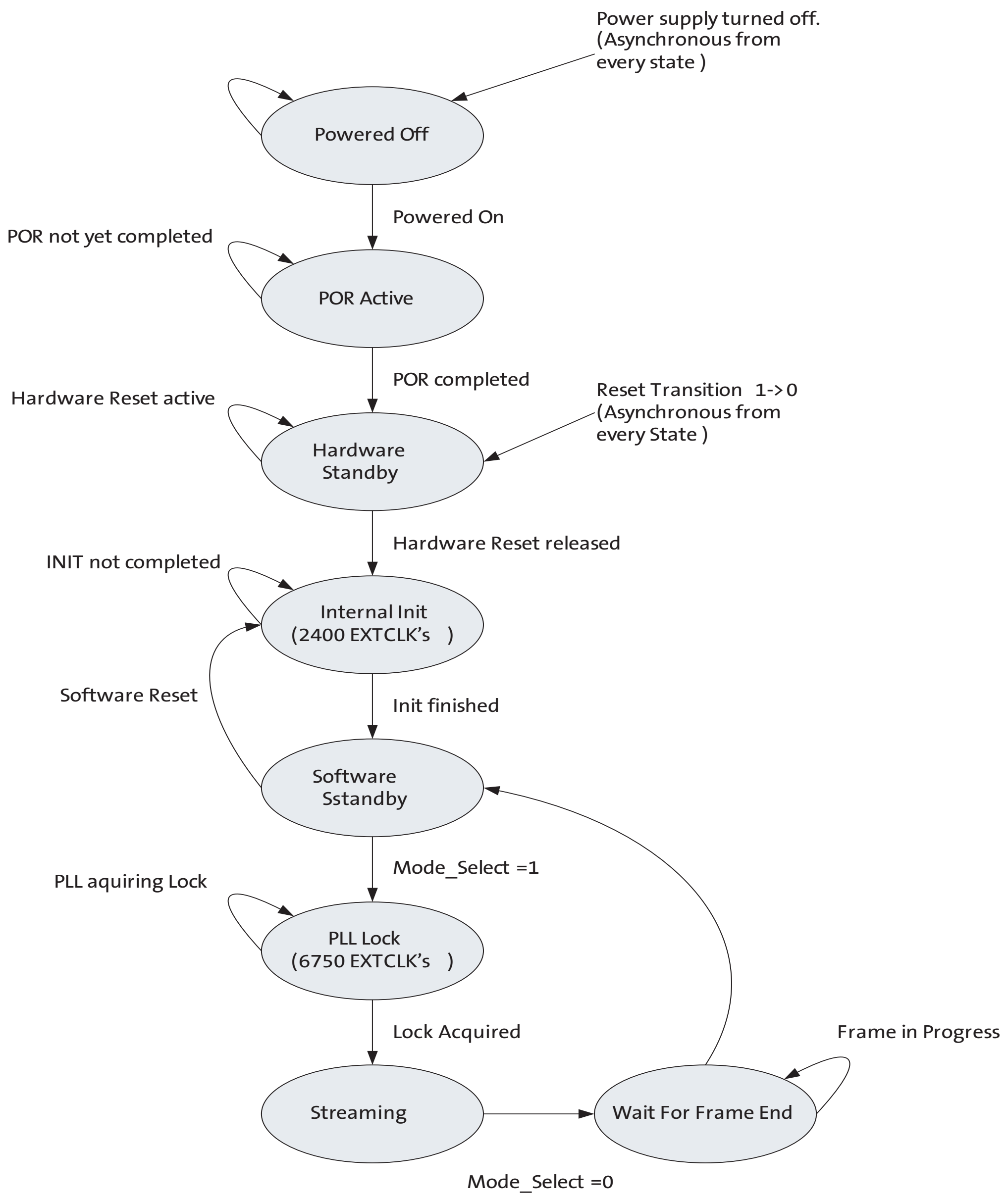


Table 10: PLL in System States

State	EXTCLKs	PLL
Powered off	2400	VCO powered down
POR Active		
Hardware standby		
Internal Initialization		
Soft standby		
PLL Lock	6750	VCO powering up and locking, PLL output bypassed
Streaming		VCO running, PLL output active
Wait for frame end		

Notes: 1. VCO = voltage-controlled oscillator.

Power-On Reset Sequence

When power is applied to the MT9D012, 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 MT9D012 is in its lowest-powered, powered-up state; the internal PLL is disabled, the CCP2 serializer is disabled and internal clocks are gated off.

When the sensor leaves the hardware standby state it performs an internal initialization sequence that takes 2400 EXTCLK cycles. Then it enters a low-power soft standby state. While the initialization sequence is in progress, the MT9D012 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, R0x0000–1. While the initialization sequence is in progress, the sensor will not respond to its device address and reads from the sensor will result in a NACK on the two-wire serial interface bus. When the sequence has completed, reads will return the operational value for the register (0x15 if R0x0000–1 is read).

When the sensor leaves soft standby mode and enables the VCO, an internal delay will keep the PLL disconnected for 6750 EXTCLKs so that the PLL can lock.

Soft Reset Sequence

The MT9D012 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 11 shows the state of the signal interface during hardware standby (RESET_BAR asserted) and the default state during soft standby (after exit from hardware standby and before any registers within the sensor have been changed from their default power-up values).

Table 11: 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.	
LV	Output	High-Z. Can be left disconnected/floating.	
FV	Output		
DOUT[9:0]	Output		
PIXCLK	Output		
SCL	Input	Enabled. Must be tied to or driven to a valid logic level.	
SDA	I/O	Enabled as an input. Must be tied to 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 up. Must be bonded to a HIGH or LOW state.	
TEST	Input	Enabled. Must be driven to a logic 0.	

General Purpose Inputs

The MT9D012 provides four general purpose inputs. After reset, the input pads associated with these signals are powered up for the MT9D01200STCZK. The pads must be left disconnected/floating if the GPIs are not enabled and must be bonded to a HIGh or LOW state if the GPIs are enabled.

For the MT9D01200STCZK, general purpose inputs are enabled by setting reset_register[8] (R0x301A). Once enabled, all four inputs must be driven to valid logic levels by external signals. The state of the general purpose inputs can be read through gpi_status[3:0] (R0x3026).

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

- Output enable (see "Output Enable Control" on page 26)
- Trigger (see the sections below)
- Standby functions (see the following sections)

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

Streaming/Standby Control

The MT9D012 can be switched between its soft standby and streaming states under pin or register control, as shown in Table 12. Selection of a pin to use for the STANDBY function is described in “General Purpose Inputs” on page 29. The state diagram for transitions between soft standby and streaming states is shown in Figure 15 on page 27.

Table 12: 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 under pin or register control, as shown in Table 13. Selection of a pin to use for the trigger function is described in “General Purpose Inputs” on page 29.

Table 13: 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 MT9D012 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.

Both SMIA profile 0 clock scheme and profile 1, 2 are supported. The clocking scheme can be selected by setting R0x306E–F[7] to 0 for profile 0 or to 1 for profile 1, 2.

Figure 16: MT9D012 SMIA Profile 1, 2 Clocking Structure

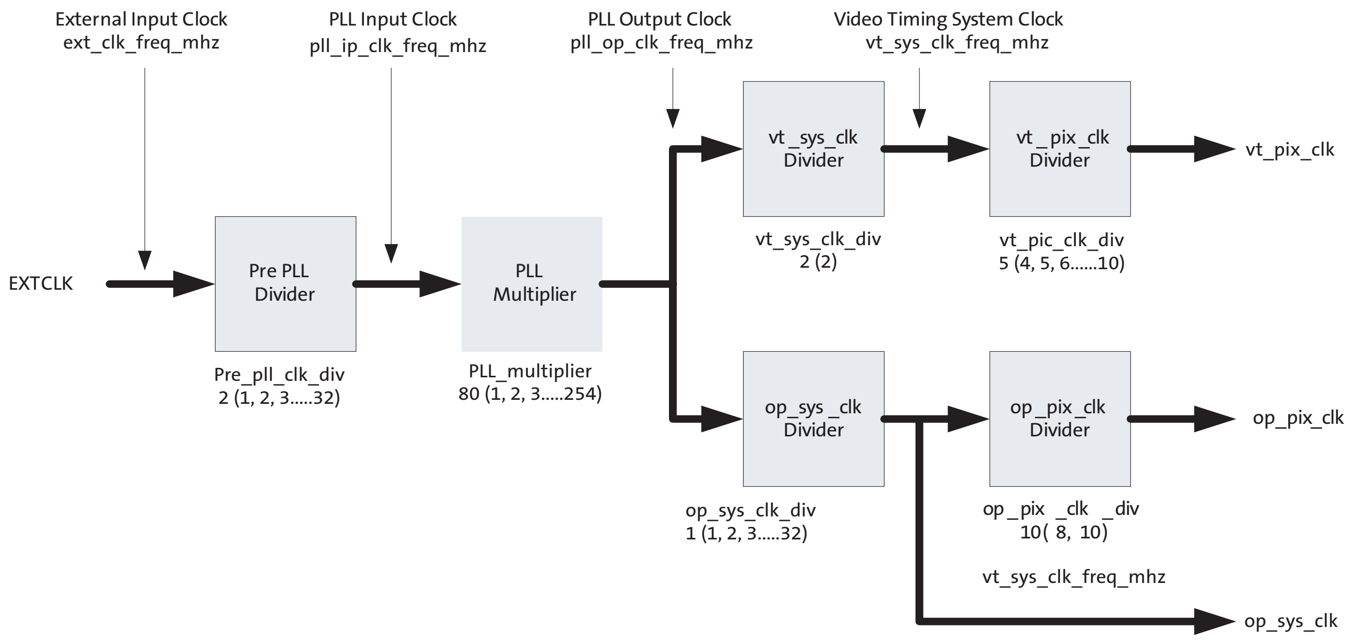


Figure 16 shows the different clocks and the names of the registers that contain or are used to control their values. The figure shows the default setting for each divider/multipler control register and the range of legal values for each divider/multiplier control register.

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.
- 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 14 and Table 15 on page 32.

Table 14: 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 15: 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

PLL input clock frequency range is 2.0–11.5 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 MT9D012 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 41 on page 62) to the CCP2 serializer. The output FIFO generates one pixel each op_pix_clk period. The pixel is 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.

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

$$vt\_pix\_clk\_freq\_mhz = \frac{ext\_clk\_freq\_mhz * pll\_multiplier}{pre\_pll\_clk\_div * vt\_sys\_clk\_div * vt\_pix\_clk\_div} \quad \text{(EQ 1)}$$

$$op\_pix\_clk\_freq\_mhz = \frac{ext\_clk\_freq\_mhz * pll\_multiplier}{pre\_pll\_clk\_div * op\_sys\_clk\_div * op\_pix\_clk\_div} \quad \text{(EQ 2)}$$

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

Figure 17: MT9D012 SMIA Profile 0 Clocking Structure

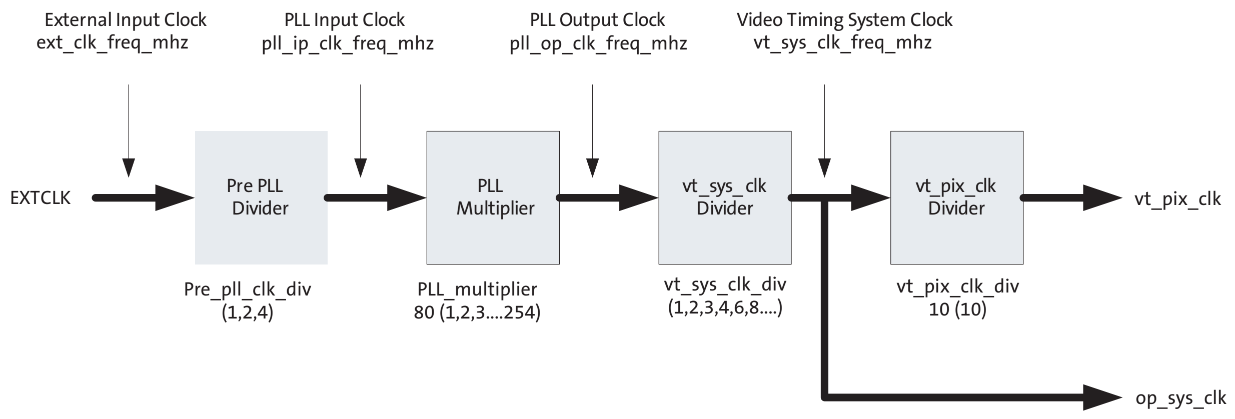


Figure 17 shows the different clocks and the names of the registers that contain or are used to control their values. The figure shows the default setting for each divider/multiplier control register and the range of legal values for each divider/multiplier control register.

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.
- 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.

PLL input clock frequency range is 2.0–11.5 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 MT9D012 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.
- vt_sys_clk is also used to generate the serial data stream on the CCP2 output.

In Profile 0 the output clock frequencies can be calculated as:

$$vt\_pix\_clk\_freq\_mhz = \frac{ext\_clk\_freq\_mhz*pll\_multiplier}{pre\_pll\_clk\_div*vt\_sys\_clk\_div*10} \quad \text{(EQ 4)}$$

$$op\_pix\_clk\_freq\_mhz = \frac{ext\_clk\_freq\_mhz*pll\_multiplier}{pre\_pll\_clk\_div*vt\_sys\_clk\_div*10} \quad \text{(EQ 5)}$$

$$op\_sys\_clk\_freq\_mhz = \frac{ext\_clk\_freq\_mhz*pll\_multiplier}{pre\_pll\_clk\_div*vt\_sys\_clk\_div} \quad \text{(EQ 6)}$$

Programming the PLL Divisors

The PLL divisors should be programmed while the MT9D012 is in the soft standby state. After programming the divisors, it is necessary to wait for the VCO lock time before enabling the PLL. The PLL is enabled by entering the STREAMING state.

An external timer will delay entering streaming mode by 6750 EXTCLKs so that the PLL can lock.

The effect of programming the PLL divisors while the MT9D012 is in the streaming state is undefined.

Influence of ccp_data_format

R0x0112-3 (ccp_data_format) controls whether the pixel data interface will generate 10 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 two least significant bits 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 least significant bit of the programmed value is ignored and treated as “0.”

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

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

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

Programming Examples

This section provides six programming examples. For each example, a table of register values is shown (for example, Table 16). The settings for the clock divisors show the “Programmed Value” and “Apparent Frequency.” These values are consistent with Figure 17 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 34.

Example 1:

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

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. Because 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 of doing this, depending upon the frequency of EXTCLK.

The register settings for this example and the resulting clock frequencies are shown in Table 16. If the MT9D012 is programmed with these values (and all other registers are left at their default values) and put into streaming mode (mode_select = 1) with the CCP interface enabled, then it will stream frames at full resolution (1600 x 1200 pixels) through its CCP interface at 23.23 fps.

Table 16: 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 (decimal)	40 (decimal)	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 (decimal)	5	op_pix_clk	64 MHz	64 MHz

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 23 fps (see “Influence of ccp_data_format” on page 34). To run the CCP interface at maximum speed in Class 0, the divisors are chosen to set op_sys_clk at 208 MHz. Vt_pix_clk is programmed to run as close to the maximum frequency (64 MHz) as possible—in this case, 52 MHz. Minimum link blanking is used to get the highest possible frame rate. Because 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 800. The register settings for this example and the resulting clock frequencies are shown in Table 17. If the MT9D012 is programmed with these values (and all other registers are left at their default values) and then put into streaming mode (mode_select = 1), it will stream frames at half resolution (800 x 1200 pixels) through its CCP interface at 18.88 fps.

Table 17: 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	0x320	–	–	–	–

Example 3:
- 8 bits per pixel data
- CCP2 Class 1/Class 2 signalling
- Highest possible frame rate at maximum resolution

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. Because 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 of doing this, depending upon the frequency of EXTCLK.

The register settings for this example and the resulting clock frequencies are shown in Table 18. If the MT9D012 is programmed with these values (and all other registers are left at their default values) and put into streaming mode (mode_select=1) with the CCP interface enabled, then it will stream frames at full resolution (1600 x 1200 pixels) through its CCP interface at 23.23 fps.

Table 18: 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

Table 18: Programming Example 3 (continued)

Register	Programmed Value	Effective Value	Clock	Apparent Frequency	Actual Frequency
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	1	op_sys_clk	512 MHz	256 MHz
op_pix_clk_div	8	4	op_pix_clk	64 MHz	64 MHz

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 23 fps (see "Influence of ccp_data_format" on page 34). To run the CCP interface at maximum speed in Class 0, the divisors are chosen to set op_sys_clk at 208 MHz. Vt_pix_clk is programmed to run as close to the maximum frequency (64 MHz) as possible—in this case, 52 MHz. Because op_pix_clk runs at half the frequency with full resolution frames, additional line blanking is required to get the same line rate in and out of the FIFO. A line length value of 3,234 will give a frame rate of 12.8 fps.

Table 19: 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	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

Example 5:

- 10 bits per pixel data
- Parallel output
- Highest possible frame rate at maximum resolution

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. Because 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 of doing this, depending upon the frequency of EXTCLK.

The register settings for this example and the resulting clock frequencies are shown in Table 20. If the MT9D012 is programmed with these values (and all other registers are left at their default values) and put into streaming mode (mode_select=1) with the parallel interface enabled, then it will stream frames at full resolution (1600 x 1200 pixels) through its parallel interface at 23.23 fps.

Table 20: Programming Example 5

Register	Programmed Value	Effective Value	Clock	Apparent Frequency	Actual Frequency
–	–	–	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

Example 6:
- 10 bits per pixel data
- Parallel output
- Highest possible frame rate at SVGA resolution (2X skip)

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. Because 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 of doing this, depending upon the frequency of EXTCLK.

The register settings for this example and the resulting clock frequencies are shown in Table 21. If the MT9D012 is programmed with these values (and all other registers are left at their default values) and put into streaming mode (mode_select=1) with the parallel interface enabled, then it will stream frames at half resolution (800 x 600 pixels) through its parallel interface at up to 70.78 fps.

Table 21: Programming Example 6

Register	Programmed Value	Effective Value	Clock	Apparent Frequency	Actual Frequency
–	–	–	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
x_odd_inc	3	–	–	–	–
y_odd_inc	3	–	–	–	–
x_addr_end	1605	–	–	–	–

Table 21: Programming Example 6 (continued)

Register	Programmed Value	Effective Value	Clock	Apparent Frequency	Actual Frequency
y_addr_end	1205	–	–	–	–
line_length_pck	1456	–	–	–	–
frame_lenght_lines	621	–	–	–	–

Clock Control

The MT9D012 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 MT9D012 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 MT9D012 supports two image acquisition modes:

1. Electronic rolling shutter (ERS) mode.
   This is the normal mode of operation. When the MT9D012 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 MT9D012 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 MT9D012 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 artifacts associated with ERS operation. Visual artifacts arise in ERS operation, particularly at low frame rates, because an ERS image effectively integrates each row of the pixel array at a different point in time 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, and 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

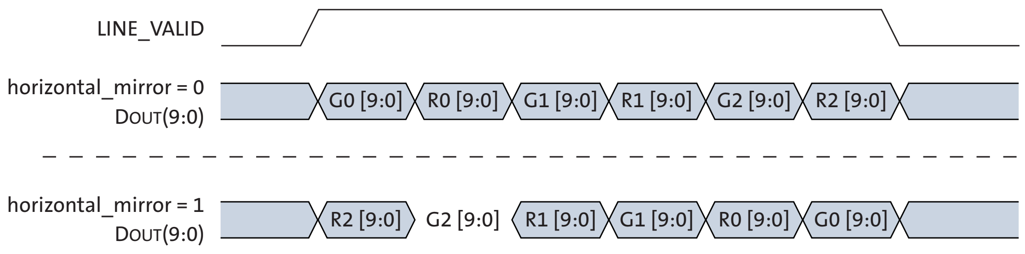
The default settings of the sensor provide a 1600H x 1200V image in parallel output mode. Two embedded data rows are added in serial mode (default operating mode) such that the default frame size is 1600H x 1202V. A border of up to 8 pixels (4 in binning) on each edge can be enabled by reprogramming the x_addr_start, y_addr_start, x_addr_end, 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 18 on page 41 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.

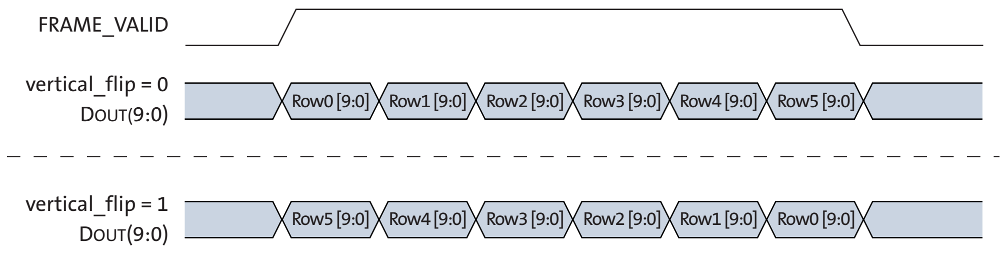
Figure 18: Effect of horizontal_mirror on Readout Order



Vertical Flip

When the vertical_flip bit is set in the image_orientation register, the order that pixel rows are read out is reversed, so that row readout starts from y_addr_end and ends at y_addr_start. Figure 19 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 19: Effect of vertical_flip on Readout Order



Subsampling

The MT9D012 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 will skip two rows/columns during readout and is equivalent to the skip2 readout mode provided by earlier Aptina Imaging sensors. Figure 20 shows a sequence of pixels being read out with x_odd_inc = 3 and y_odd_inc = 1. The effect of the different subsampling settings on the pixel array readout is shown in Figure 21 on page 42 through Figure 24 on page 44.

Figure 20: Effect of x_odd_inc = 3 on Readout Sequence

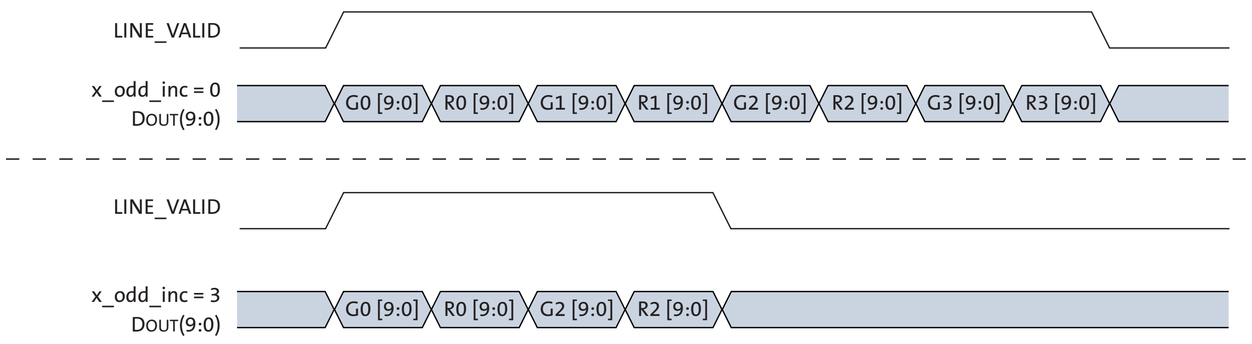


Figure 21: Pixel Readout (No Subsampling)

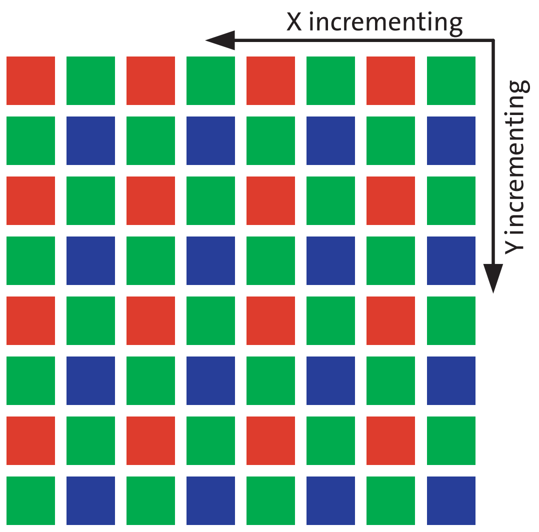


Figure 22: Pixel Readout (x_odd_inc = 3, y_odd_inc = 1)

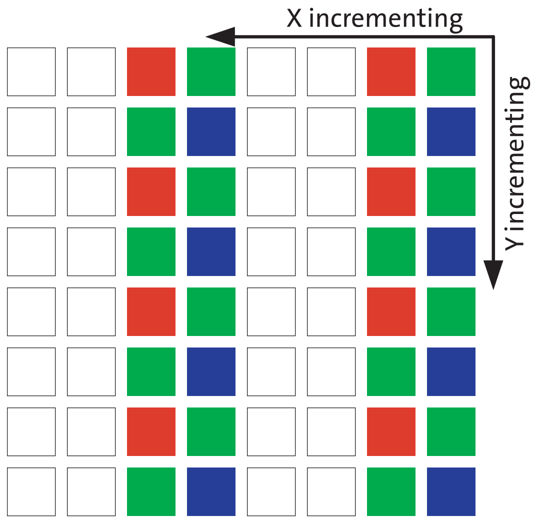


Figure 23: Pixel Readout (x_odd_inc = 1, y_odd_inc = 3)

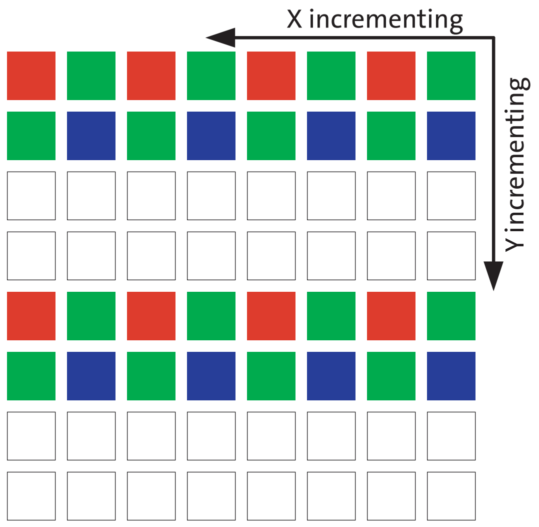
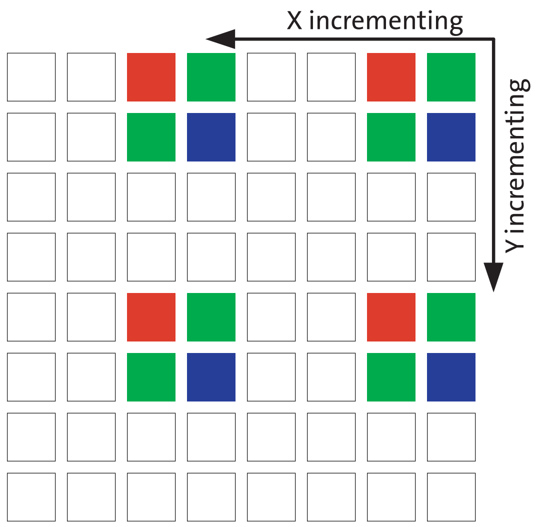


Figure 24: Pixel Readout (x_odd_inc = 3, y_odd_inc = 3)



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 0x0002;
if (remainder == 0) addr_end = addr_end - 2;
```

Table 22 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 also shown; the column address sequencing during binning is described in "Programming Restrictions when Binning" on page 46.

Table 22: 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 MT9D012 supports 2 x 1 and 2 x 2 analogue 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 25 and Figure 26.

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

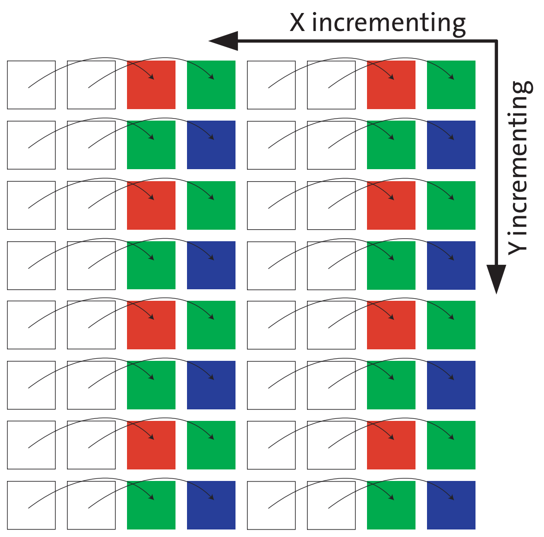
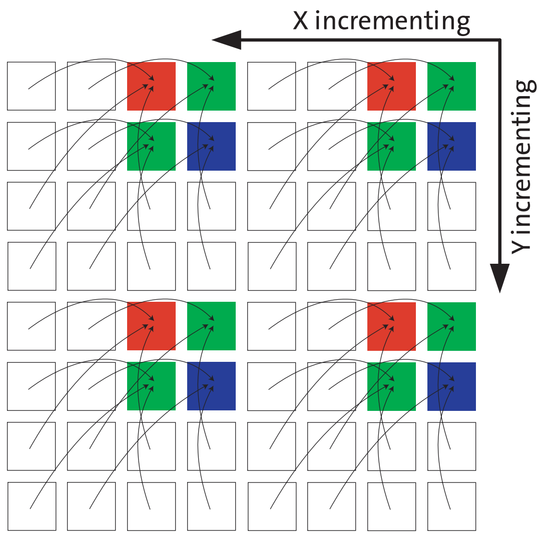


Figure 26: Pixel Readout (x_odd_inc = 3, y_odd_inc = 3, xy_bin = 1)



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 because its parameter limit registers are defined as being static.

As a result, when xy-binning is enabled, some of the programming limits declared in the parameter limit registers are no longer valid. In addition, the default values for some of the manufacturer-specific registers need to be reprogrammed. The recommended settings are shown in Table 23. 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 23: Register Adjustments Required for Binning Mode

Register	Type	Default (Normal Readout)	Recommended Setting During Binning	Notes
min_line_blanking_pck	read-only	0x0290	0x044D	Read-only register for control software; does not affect operation of sensor.
min_line_length_pck	read-only	0x0390	0x05FA	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	0x014D	0x03B5	Affects operation of sensor.
fine_correction	read/write	0x00DB	0x01A6	Affects operation of sensor.
fine_integration_time	read/write	0x0204	0x03E5	Normal default is minimum value.
x_addr_max	read-only	0x0807	0x0805	Keep binned columns within the array.
y_addr_max	read-only	0x0607	0x0605	Keep binned row within the array.

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 22 on page 44.

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

remainder = n AND 2;

$n\_bin = n + 2 - (2 * \text{remainder})$;

Table 24 on page 47 shows some examples of row and column binning pairs.

Table 24: Row and Column Binning Pairs

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

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

Frame Rate Control

The formula for calculating the frame rate of the MT9D012 is shown below:

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

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

$$frame\ rate\ [FPS] = \frac{(vt\_pixel\_clock\_mhz * 1\times10^{6})}{(line\_length\_pck * frame\_length\_lines)} \quad \text{(EQ 9)}$$

Frame Rates at Common Image Sizes

Table 25 shows the maximum frame rates that can be achieved at various common image sizes with a data rate limit of 640 Mb/s (CCP2). The frame rates assume a pixel rate of 64 Mp/s (vt_pix_clk = 64 MHz), a minimum line blanking of 656 pixels, and a minimum frame blanking of 21 lines.

Table 25: Frame Rates

Image Size	Maximum Frame Rate
UXGA (1600 x 1200)	23.23 fps
XVGA (1280 x 1024)	31.63 fps
XGA (1024 x 768)	48.28 fps
SVGA (800 x 600)	70.78 fps
VGA (640 x 480)	98.57 fps

Integration Time

The integration (exposure) time of the MT9D012 is controlled by the fine_integration_time and coarse_integration_time registers.

The limits for the fine integration time are defined by:

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

The limits for the coarse integration time are defined by:

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

If coarse_integration_time > (frame_length_lines – coarse_integration_time_max_margin), then the frame rate will be reduced and the frame rate will be given by replacing frame_length_lines in the frame rate formula (EQ 9) with (coarse_integration_time + course_integration_time_max_margin):

$$\text{frame rate (bound by integration time) [FPS]} = \frac{(vt\_pix\_clk\_mhz * 1 * 10^{6})}{(line\_length\_pck * (coarse\_integration\_time - coarse\_ integration\_time\_max\_margin))} \quad \text{(EQ 12)}$$

$$\frac{line\_length\_pck * coarse\_integration\_time}{(vt\_pix\_clk\_freq\_mhz * 1 * 10^{6})} \quad \text{(EQ 13)}$$

The actual integration time is given by:

$$integration\_time\ [sec] = \frac{((coarse\_integration\_time * line\_length\_pck) + fine\_integration\_time)}{(vt\_pix\_clk\_freq\_mhz*1*10^{6})} \quad \text{(EQ 14)}$$

With default settings and a vt_pix_clk of 64 MHz, the maximum integration time that can be achieved without reducing the frame rate is given by:

$$Maximum\ integration\ time\ [sec] = \frac{(((1221 - 1)*2256) + 2000)}{(64\ MHz*1*10^{6})} = 43.04\ ms \quad \text{(EQ 15)}$$

It is fundamental to the operation of an ERS that it is not possible to set an integration time that is greater than the frame time. Setting an integration time that is greater than the frame time therefore increases the frame time beyond frame_length_lines to make longer exposure times available.

Flash Control

The MT9D012 supports both xenon and LED flash through the FLASH output signal. The timing of the FLASH signal with the default settings is shown in Figures 27 through Figure 29 on page 49. The flash and flash_count registers allow the timing of the flash to be changed. The flash can be programmed to fire only once, delayed by a few frames when asserted, and (for xenon flash) the flash duration can be programmed.

Enabling the LED flash will cause one bad frame, where several of the rows only have the flash on for part of their integration time. This can be avoided 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 29. Read-only bit flash[14] is set during frames that are correctly integrated; the state of this bit is shown in Figure 27, Figure 28, and Figure 29.

Figure 27: Xenon Flash Enabled

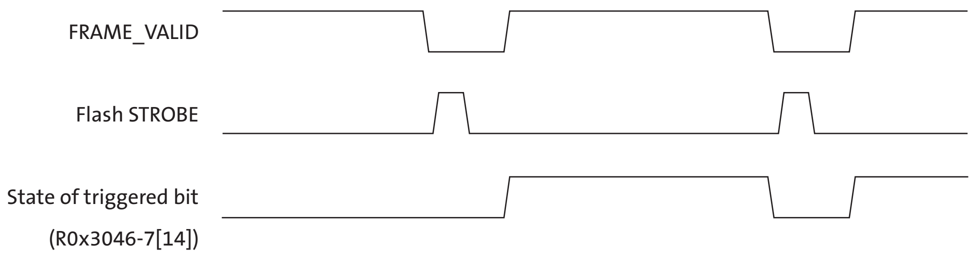
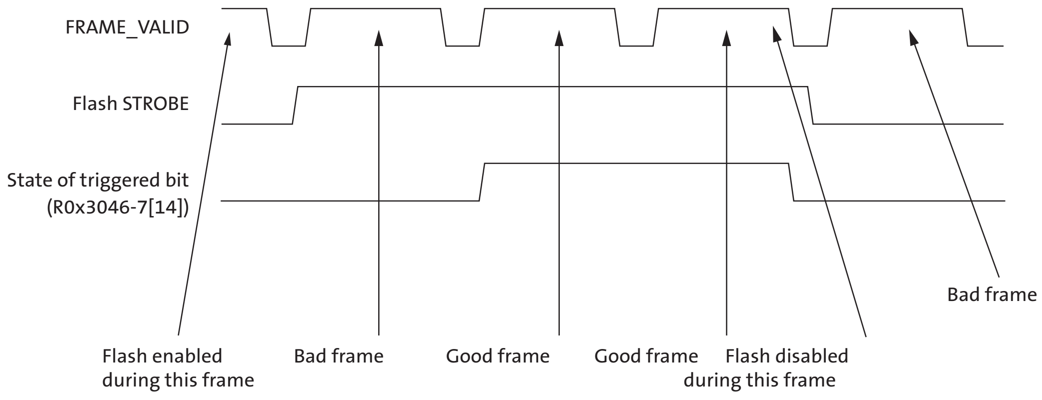
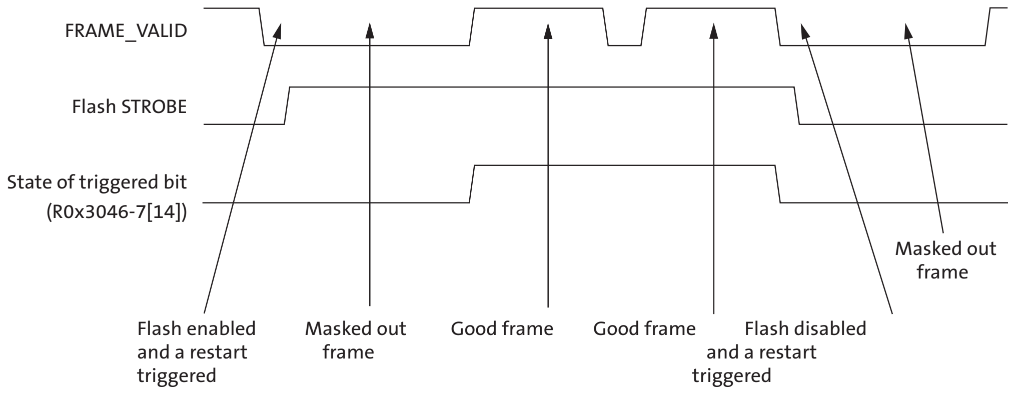


Figure 28: LED Flash Enabled



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

Figure 29: LED Flash Enabled Following Forced Restart



Global Reset

Global reset mode allows the integration time of the MT9D012 to be controlled by an external electromechanical shutter. Global reset mode is generally used in conjunction with ERS mode. The ERS mode is used to provide viewfinder information, the sensor is switched into global reset mode to capture a single frame, and the sensor is then returned to ERS mode to restore viewfinder operation.

Global reset mode is designed for use in conjunction with the parallel pixel data interface. The SMIA specification only provides for operation in ERS mode. The MT9D012 does support the use of global reset mode in conjunction with the SMIA data path, but there are additional restrictions on its use.

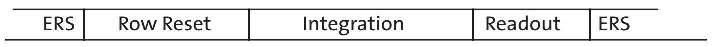
Overview of Global Reset Sequence

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 MT9D012), the electromechanical shutter is closed.
7. A single output frame is generated by the sensor with the usual LV, FV, PIXCLK, and DOUT timing. As soon as the output frame has completed (FV negates), the electromechanical shutter may be opened again.
8. The sensor automatically resumes operation in ERS mode.

This sequence is shown in Figure 30. The following sections expand on this figure to show how the timing of this sequence is controlled.

Figure 30: Overview of Global Reset Sequence



Entering and Leaving the Global Reset Sequence

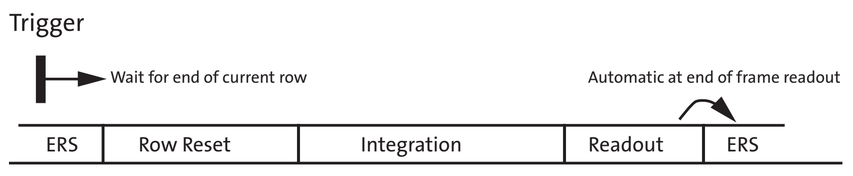
A global reset sequence can be triggered by a register write to global_seq_trigger[0] (global trigger, to transition this bit from a 0 to a 1) or by a rising edge on a suitably-configured GPI input (see "Trigger Control" on page 30).

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

The global reset sequence completes with a frame readout. At the end of this readout phase, the sensor automatically resumes operation in ERS mode. The first frame integrated with ERS will be generated after a delay of approximately ((min_frame_blanking + coarse_integration_time) * line_length_pck). This sequence is shown in Figure 31 on page 51.

While operating in ERS mode, double-buffered registers 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 31: Entering and Leaving a Global Reset Sequence


Trigger
Wait for end of current row
Automatic at end of frame readout
ERS
Row Reset
Integration
Readout
ERS

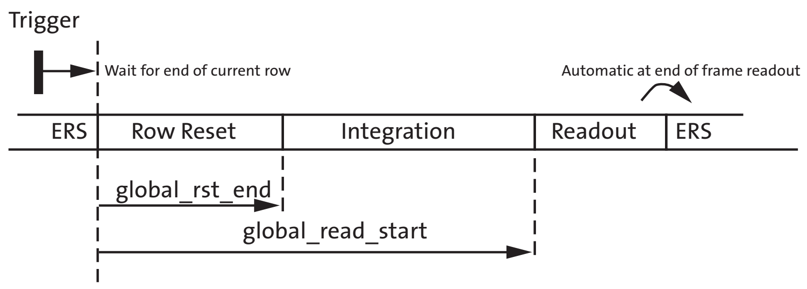
Programmable Settings

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 32. 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.

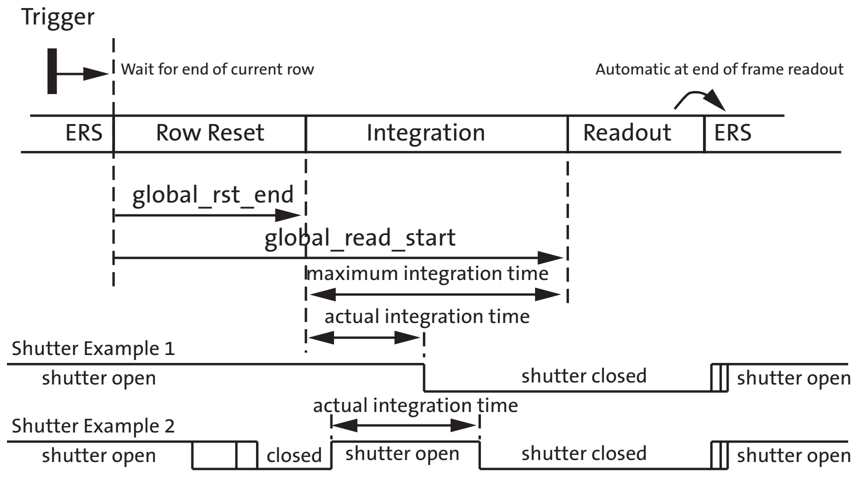
Figure 32: Controlling the Reset and Integration Phases of the Global Reset Sequence



Control of the Electromechanical Shutter

Figure 33 on page 52 shows two different ways that a shutter can be controlled during the global reset sequence. In both cases, the maximum integration time is set by the difference between global_read_start and global_rst_end. In shutter example 1, the shutter is open during the initial ERS sequence and during the row reset phase. The shutter closes during the integration phase. The pixel array is integrating incident light from the start of the integration phase to the point at which the shutter closes. Finally, the shutter opens again after the end of the readout phase. In shutter example 2, the shutter is open during the initial ERS sequence and closes sometime during the row reset phase. The shutter both opens and closes during the integration phase. The pixel array is integrating incident light for the part of the integration phase during which the shutter is open. As for the previous example, the shutter opens again after the end of the readout phase.

Figure 33: Control of the Electromechanical Shutter



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

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

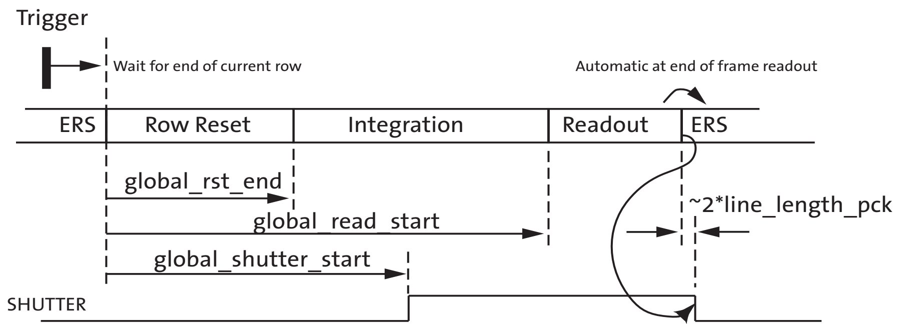
After FV negates to signal the completion of the readout phase, there is a time delay of approximately (10 * line_length_pck) before the sensor starts to integrate light-sensitive rows for the next ERS frame. It is essential that the shutter be opened at some point in this time window; otherwise, the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

The MT9D012 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 34. SHUTTER is negated by default. The point at which it asserts is controlled by the programming of global_shutter_start. At the end of the global reset readout phase, SHUTTER negates approximately (2 * line_length_pck) after the negation of FV.

The following programming restriction must be met for correct operation:

- global_read_start > global_shutter_start

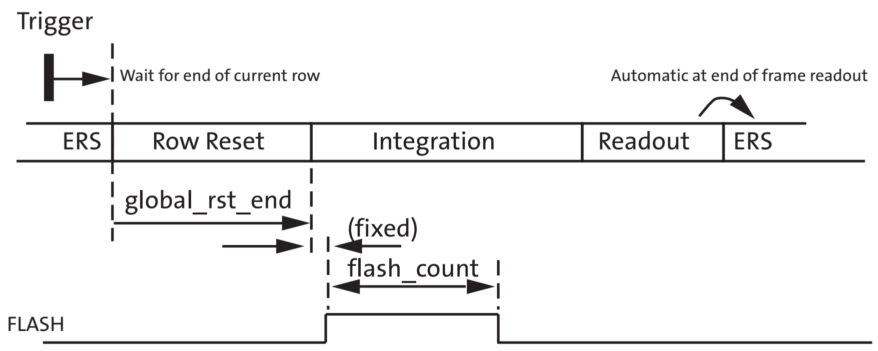
Figure 34: Controlling the SHUTTER Output



Using FLASH with Global Reset

If global_seq_trigger[2] = 1 (global flash enabled) when a global reset sequence is triggered, the FLASH output signal will be pulsed during the integration phase of the global reset sequence. The FLASH output will assert a fixed number of cycles after the start of the integration phase and will remain asserted for a time that is controlled by the value of the flash_count register, as shown in Figure 35.

Figure 35: 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).

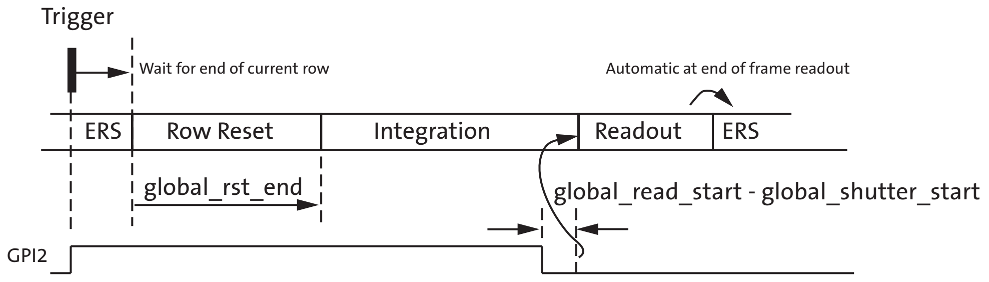
The following programming restrictions must be met for correct operation of 'Bulb' exposures:

- 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)

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 36. 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.

Figure 36: Global Reset Bulb


Trigger
Wait for end of current row
Automatic at end of frame readout
ERS
Row Reset
Integration
Readout
ERS
global_rst_end
global_read_start - global_shutter_start
GPI2

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 FV for the global reset readout phase.

Using Global Reset with SMIA Data Path

When a global reset sequence is triggered, it usually results in the frame in progress being truncated (at the end of the current output line). The SMIA data path limiter function (see Figure 41 on page 62) attempts to extend (pad) all frames to the programmed value of y_output_size. If this padding is still in progress when the global reset readout phase starts, the SMIA data path will not detect the start of the frame correctly. Therefore, to use global reset with the SMIA data path, this timing scenario must be avoided. One possible way of doing this would be to synchronize (under software control) the assertion of trigger to an end-of-frame marker on the CCP serial data stream.

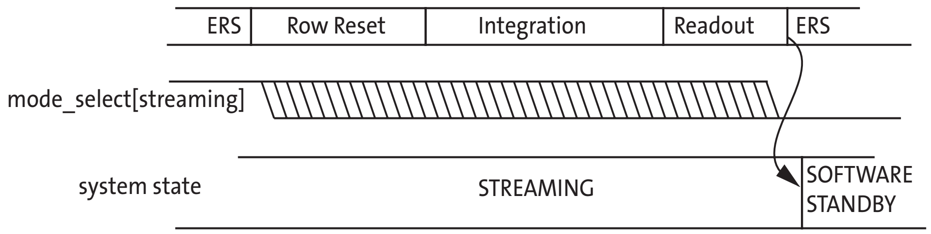
At the end of the readout phase of the global reset sequence, the sensor automatically resumes operation in ERS mode.

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

Global Reset and Soft Standby

If the mode_select[stream] bit is cleared while a global reset sequence is in progress, the MT9D012 will remain in streaming state until the global reset sequence (including frame readout) has completed, as shown in Figure 37.

Figure 37: Entering Soft Standby During a Global Reset Sequence



Analog Gain

The MT9D012 provides two mechanisms for setting the analog gain. The first uses the SMIA gain model. The second uses the traditional Aptina Imaging gain model. The following sections describe both models, the mapping between the models, and the operation of the per-color and global gain control.

Using Per-color or Global Gain Control

The read-only analogue_gain_capability register returns a value of "1," indicating that the MT9D012 provides per-color gain control. However, the MT9D012 also provides the option of global gain control. Per-color and global gain control can be used interchangeably. A write to a global gain register is aliased as a write of the same data to the four associated color-dependent gain registers. A read from a global gain register is aliased to a read of the associated green1/greenR gain register.

The read/write gain_mode register required by SMIA has no defined function in the SMIA specification. In the MT9D012 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:

$$gain = \frac{analogue\_gain\_m0 \times analogue\_gain\_code}{analogue\_gain\_c1} = \frac{analogue\_gain\_code\_<color>}{8} \qquad \text{(EQ 16)}$$

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 provides 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:

$$gain = (<color>\_gain[7] + 1) \times \frac{<color>\_gain[6:0]}{16} \qquad \text{(EQ 17)}$$

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 = 0x0190 provides the same gain as red_gain=0x0120. 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 because this will result in lower noise.

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 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 gains written through one set of registers must be read back through the same set of registers.

Sensor Core Digital Data Path

Test Patterns

The MT9D012 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 26.

Table 26: 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, various MT9D012 registers must be set appropriately to control the frame rate and output timing. These include:
- 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
- Lens/color shading correction

The complete list for disabling digital corrections is:
- R0x3044[10] = 0x0000
- R0x30CA[0] = 0x0001
- Rox30CC–R0x30D2 = 0x0000
- R0x30D4[15] = 0x0000
- R0x30C2–R0x30C8 = 0x0000
- R0x30C0[0] = 0x0001
- R0x301A[3] = 0x0000
- R0x301E = 0x0000
- R0x3180[15] = 0x0000
- R0x318A = 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).

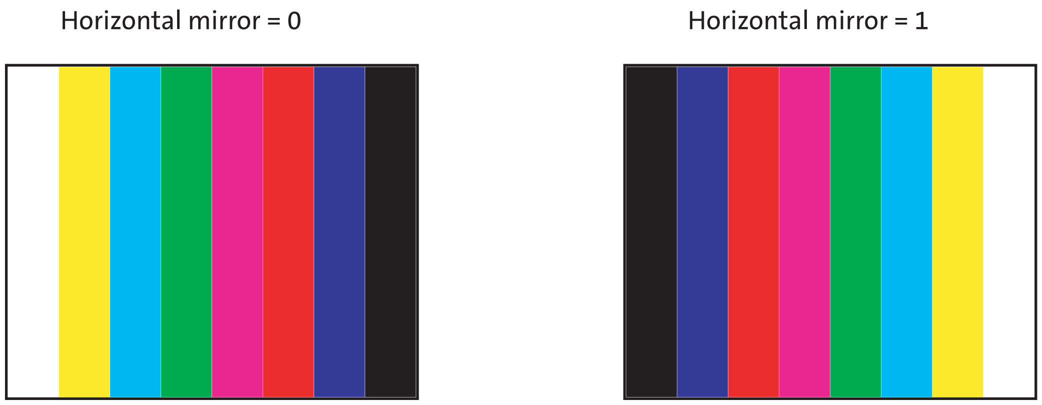
100 Percent Color Bars Test Pattern

In this test pattern, shown in Figure 38, all pixel data is replaced by a Bayer version of an 8-color, color-bar chart (white, yellow, cyan, green, magenta, red, blue, and black). Each bar is 200 pixels wide and occupies the full height of the output image. Each color component of each bar is set to "0" (fully off) or 0x3FF (fully on for 10-bit data). The pattern repeats after 8 * 200 = 1600 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 200 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 38: 100 Percent Color Bars Test Pattern



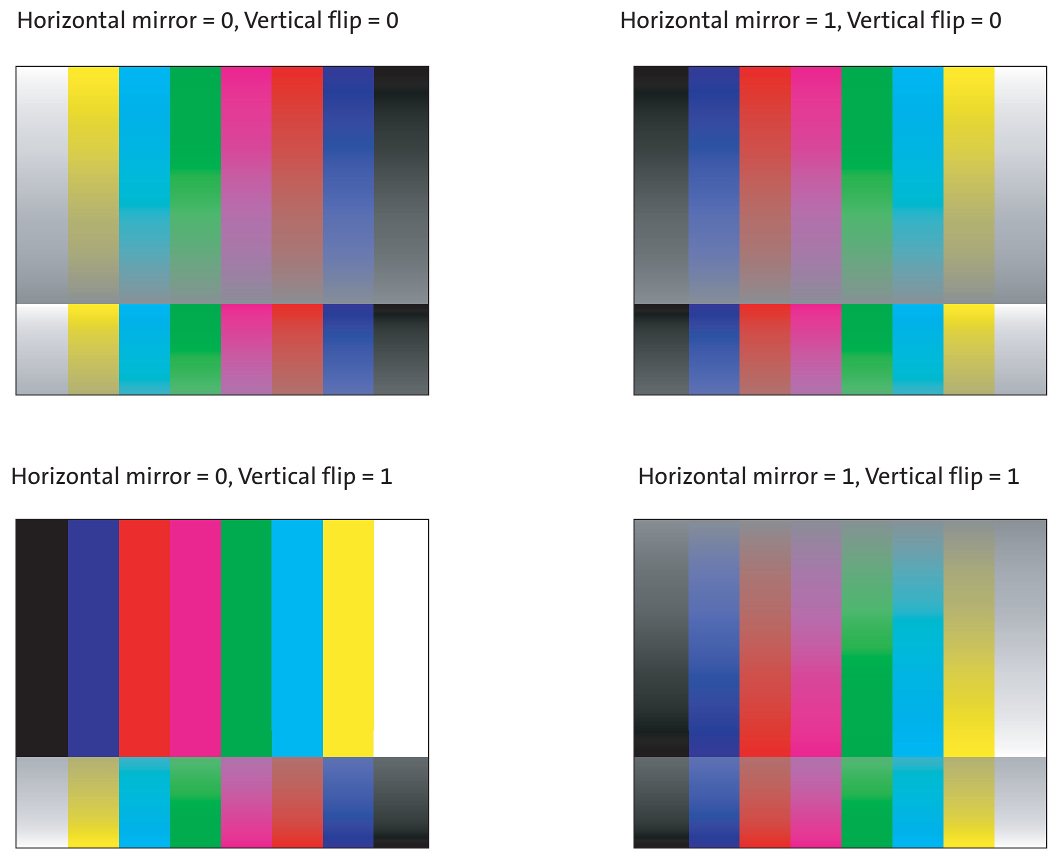
Fade-to-Gray Color Bars Test Pattern

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, and black). Each bar is 200 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 1024th 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 * 200 = 1600 pixels and vertically after 1024 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, and y_addr_end and may be affected by the setting of x_output_size and 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 200 pixels.

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

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

Figure 39: Fade-to-Gray Color Bars Test Pattern


Horizontal mirror = 0, Vertical flip = 0
Horizontal mirror = 1, Vertical flip = 0
Horizontal mirror = 0, Vertical flip = 1
Horizontal mirror = 1, Vertical flip = 1

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.

Test Cursors

The MT9D012 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 are 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.

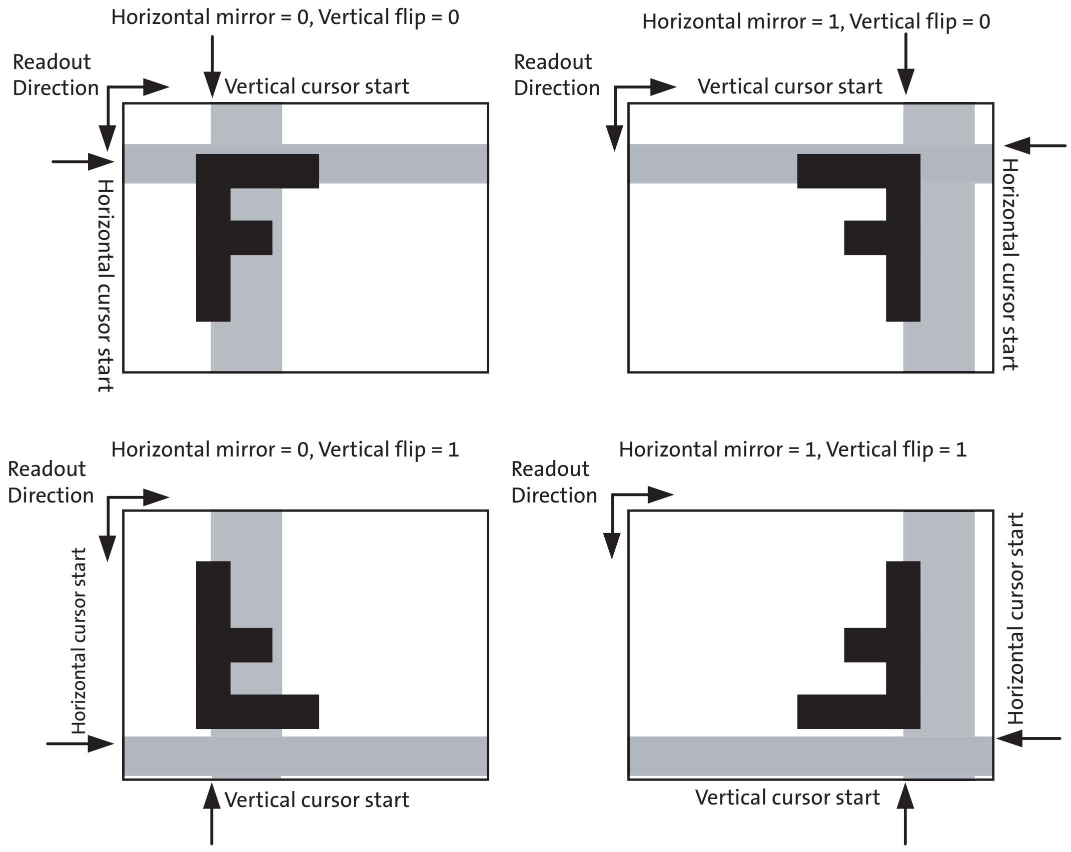
The cursors are opaque (they replace data from the imaged scene or test pattern). The color of each cursor is set by the values of the Bayer components in the test_data_red, test_data_greenR, test_data_blue, and test_data_greenB registers. As a consequence, the cursors are the same color as test pattern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

When vertical_cursor_position = 0x0FFF, 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). Note that the active pixel array is smaller than this, so in the last 56 steps the cursor will not be visible. 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 behavior of the MT9D012 is shown in Figure 40 on page 61. In the figure, the test cursors are shown as translucent for clarity. In practice, they are opaque (they overlay the imaged scene). The manner in which the test cursors are affected by the value of image_orientation can be understood from 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.

Figure 40: Test Cursor Behavior = image_orientation



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).

Pedestal

This block adds the value from R0x301E–F[10:0] (data_pedestal_) to the incoming pixel value.

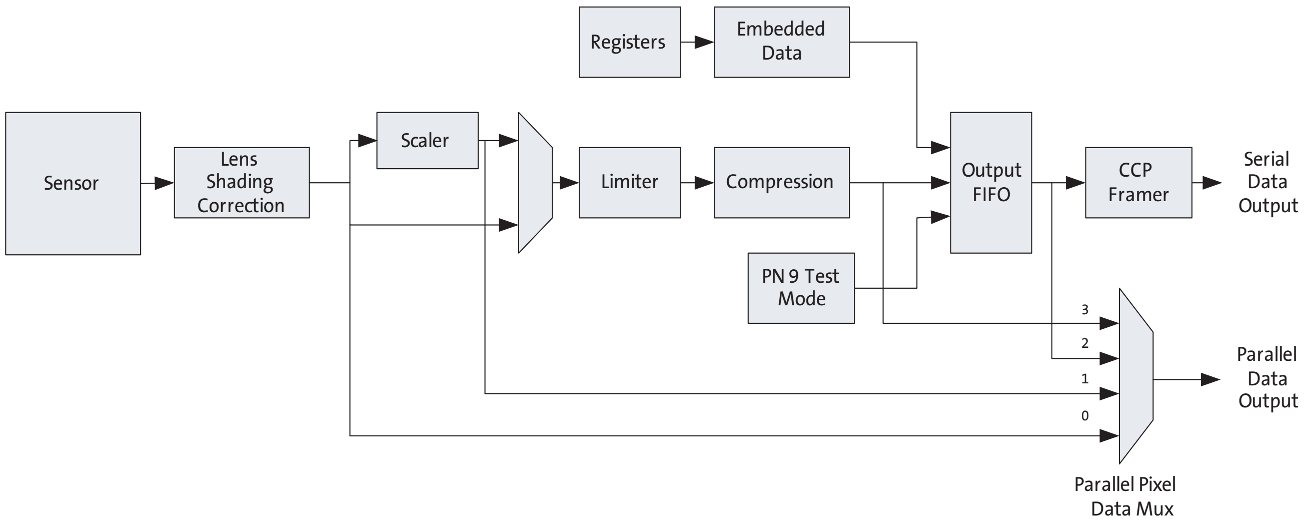
The data_pedestal register is read-only by default but can be made read/write by clearing the lock_reg bit in R0x301A–B.

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

Digital Data Path

The digital data path after the sensor core is shown in Figure 41.

Figure 41: Data Path



Digital Lens/Color Shading Correction

Lenses tend to produce images with the brightness significantly attenuated near the edges. Chromatic aberration in such lenses can cause color variation across the field of view. There are also other factors inside the pixels affected by the chief ray angle of the lens causing signal gradients in images captured by image sensors. The cumulative result of all these factors is known as lens/color shading. The MT9D012 has an embedded lens shading correction (LC) module that can be programmed to counter the shading effect of a lens and the pixels on each R, Gb, Gr, and B color signal that has been specifically optimized for Aptina's sensor. The LC module multiplies R, Gb, Gr, and B signals by a two-dimensional correction function F (x,y), whose profile in both x and y direction is a piecewise quadratic polynomial with coefficients independently programmable for each direction and color.

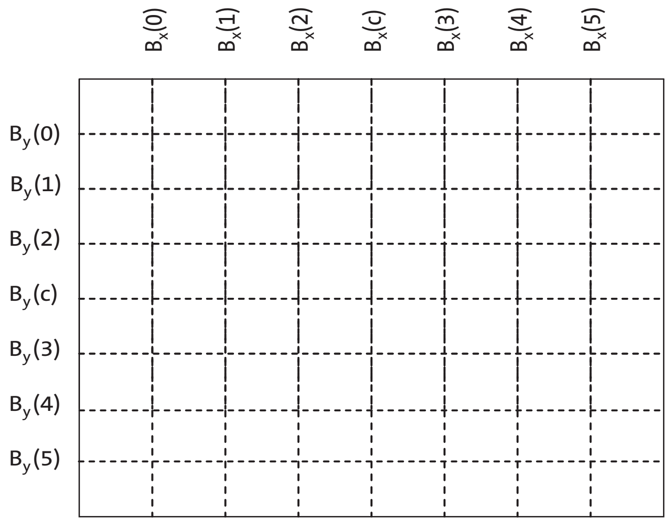
To increase the precision of the correction function, the image plane is divided into 8 zones in each dimension (shown in Figure 42 on page 63). Each boundary as well as C (Cx, Cy) coordinate is stored as a byte, which represents the coordinate value divided by 32. There are always three boundaries to the left (top) of the center and three to the right (bottom) of the center. These boundaries apply uniformly for each color channel. However, the correction functions are programmable independently for each color component.

The lens shading correction block can be enabled/disabled in register R0x318A–B.

There are no default values set in the lens shading block, all registers need to be initialized with valid values before enabling the lens color shading correction block.

Aptina provides a solution finder for the lens shading correction that can either work with RAW/BMP images or directly with a live sensor within its DevWare development environment.

Figure 42: Lens Correction Zones



Embedded Data Format and Control

When the serial pixel data path is selected, the first two rows of the output image contain register values that are appropriate for the image. The 10-bit format places the data byte in bits [9:2] and sets bits [1:0] to a constant value of 01.

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.

Parallel Pixel Data Output MUX

If in parallel mode, using the default value of R0x306E–F, 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 points in the SMIA. The output datapath selection is controlled by R0x306E–F. This internal data path control is provided for debug only, not for functional operation.

Figure 43 illustrates differences in signalling schemes used to present the data. 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 LV is asserted, no valid pixel data 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 LV 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 FV = 1, LV = 1 (this is redundant because LV = 1 implies that FV = 1) and uses this redundancy to modify the behavior of FV as follows:

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

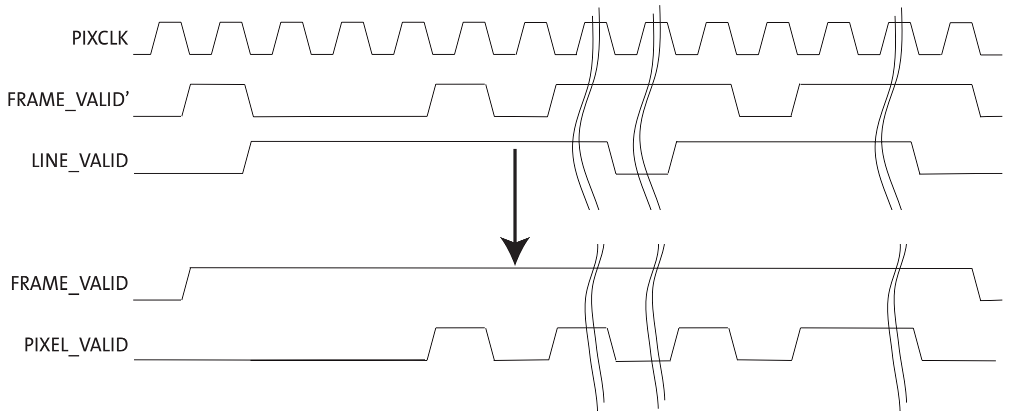
Externally to the sensor, the internal signals can be recovered trivially:

FRAME_VALID = FRAME_VALID' | LINE_VALID

PIXEL_VALID = FRAME_VALID' & LINE_VALID

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

Figure 43: Creating PIXEL_VALID and Re-creating FV



The data formats on the parallel pixel data output bus are shown in Table 27.

Table 27: 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].
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 59.

Timing Specifications

Power-Up Sequence

The recommended power-up sequence for the MT9D012 is shown in Figure 44. The available power supplies—VDD_IO, VDD, VDD_PLL, VAA, VAA_PIX—can be turned on at any point or have the sequence specified below for reducing current consumption during power-up:

1. Turn on VDD_IO power supply.
2. After 1–500ms, turn on VDD and power supply.
3. After 1–500ms, turn on VDD_PLL and VAA/VAA_PIX power supplies.
4. After the last power supply is stable, enable EXTCLK.
5. Assert RESET_BAR for at least 1ms.
6. Wait 2400 EXTCLKs for internal initialization into soft standby.
7. Configure PLL, output, and image settings to desired values.
8. Set mode_select = 1 (R0x0100).
9. Wait 6750 EXTCLKs for the PLL to lock before streaming state is reached (enforced in hardware).

Notes:
1. If VAA/VAA_PIX is powered- up prior to VDD, then a 100mA (worst case) static current can be drawn from VAA/VAA_PIX until all supplies are powered up.
2. If VDD_IO and VDD are not tied together in the application, VDD_IO must be powered up before or at the same time as VDD.

Figure 44: Power-Up Sequence

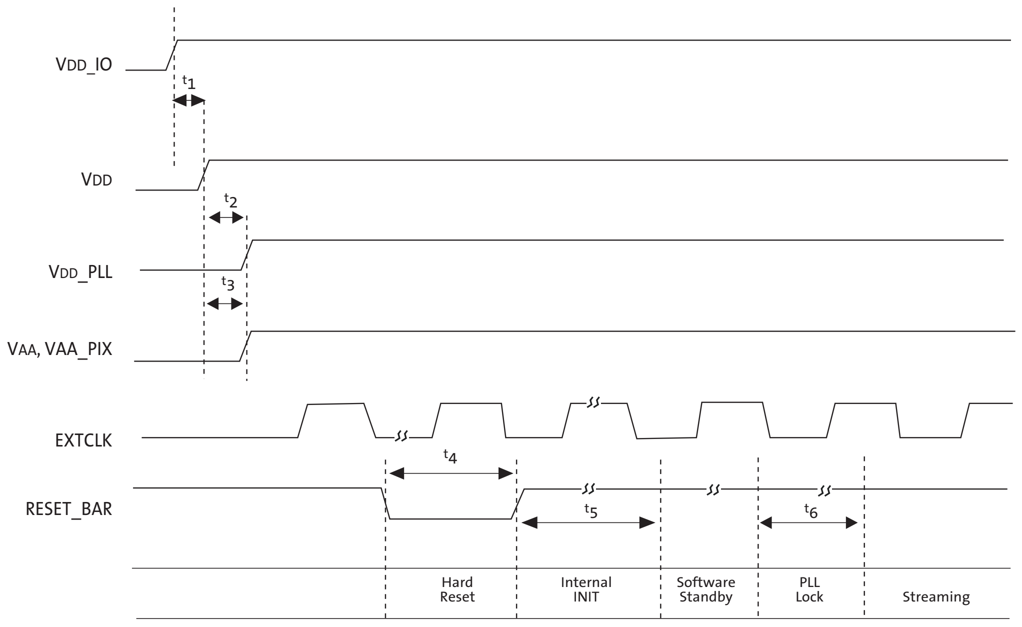


Table 28: **Power-Up Sequence**

Definition	Symbol	Min	Typ	Unit
VDD_IO to VDD time	$^t1$	0	–	ms
VDD to VDD_PLL time	$^t2$	0	–	ms
VDD to VAA/VAA_PIX time	$^t3$	0	–	ms
Active hard reset	$^t4$	1	–	ms
Internal initialization	$^t5$	2400	–	EXTCLKs
PLL lock time	$^t6$	6750	–	EXTCLKs

Power-Down Sequence

The recommended power-down sequence for the MT9D012 is shown in Figure 45. The available power supplies—VDD_IO, VDD, VDD_PLL, VAA, VAA_PIX—can be turned off at any point or have the sequence specified below for reducing current consumption during power-down:

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

Figure 45: **Power-Down Sequence**

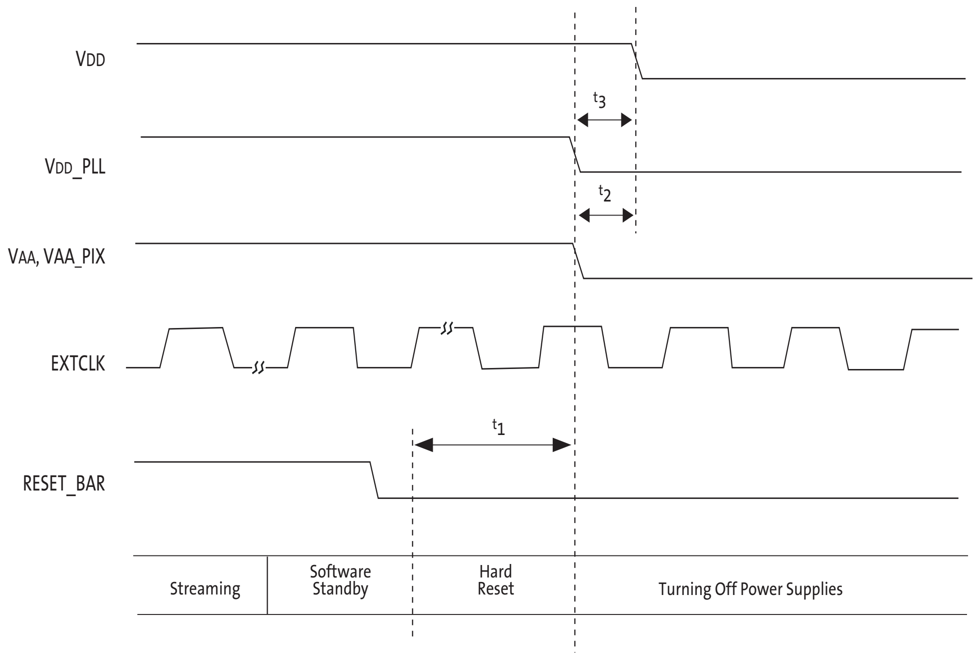


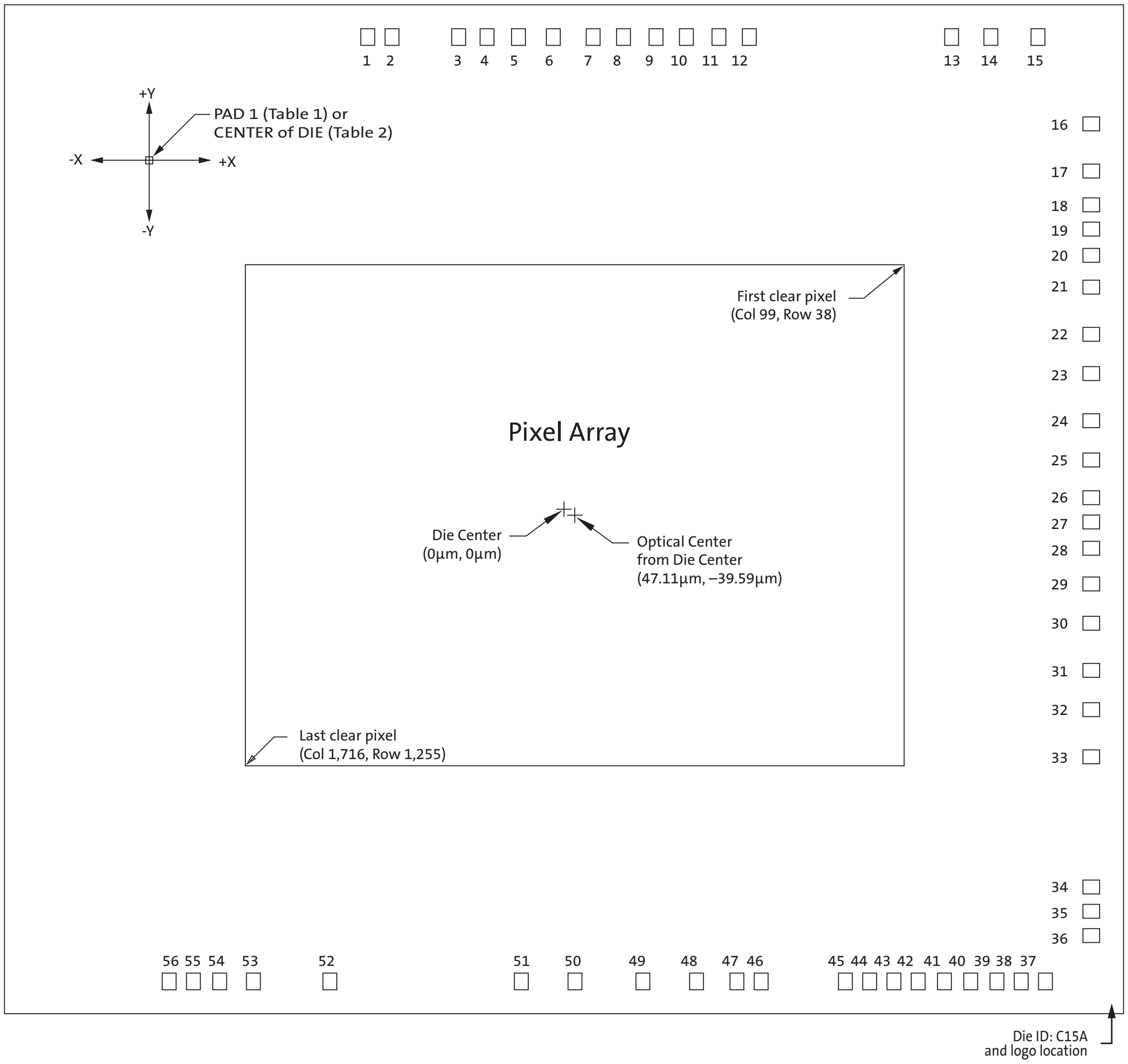
Table 29: Power-Down Sequence

Definition	Symbol	Min	Typ	Unit
Hard reset	${}^{t}1$	1	–	ms
VDD/VAA/VAA_PIX to VDD time	${}^{t}2$	0	–	ms
VDD_PLL to VDD time	${}^{t}3$	0	–	ms
VDD to VDD_IO time	${}^{t}4$	0	–	ms

Mechanical Specifications

Figure 46 shows the die outline, including the readout orientation, the optically-active area, and the offset of the optical center from the die center.

Figure 46: Die Outline



Spectral Characteristics

Figure 47: Quantum Efficiency

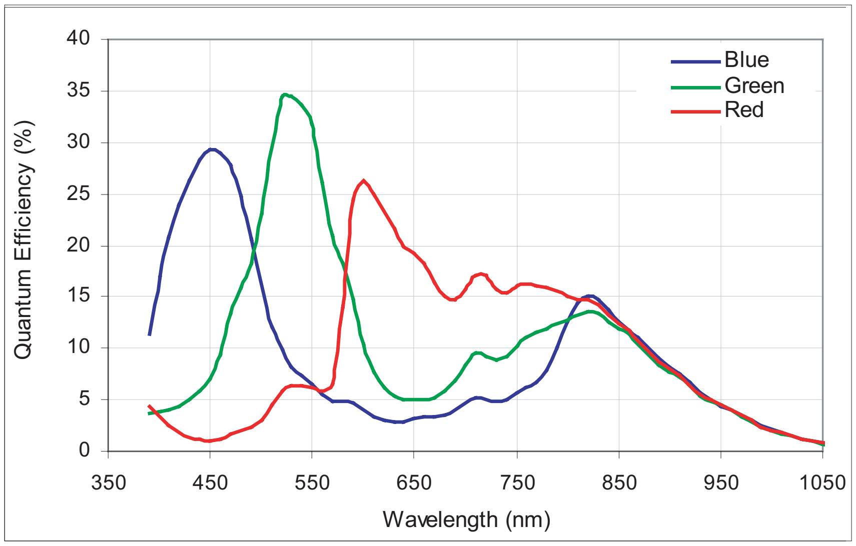
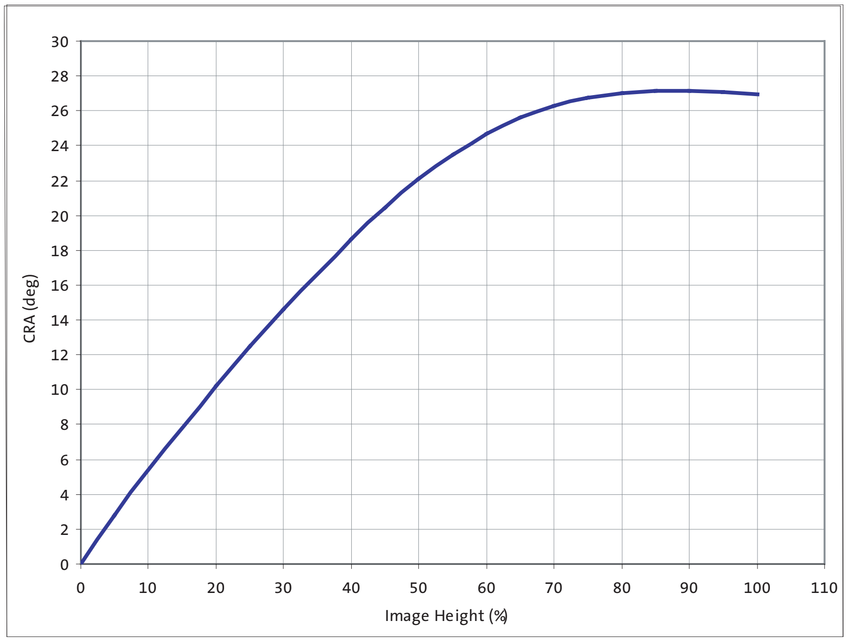
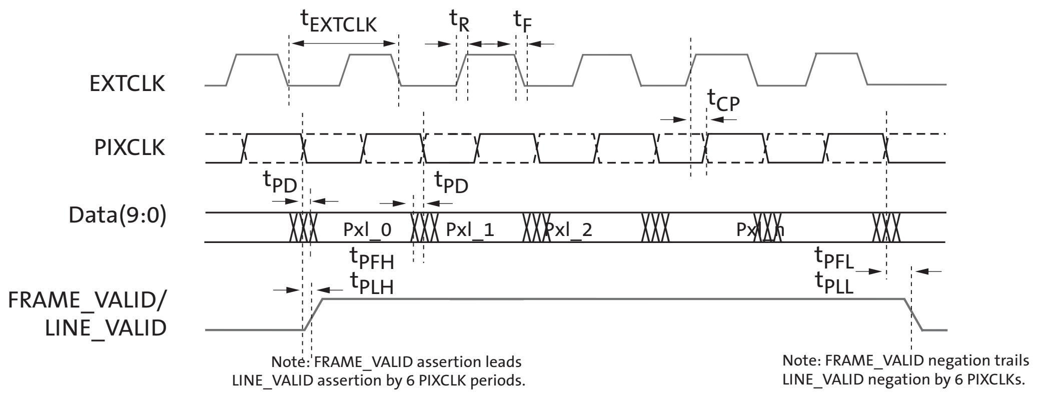


Figure 48: CRA vs. Image Height

CRA vs. Image Height Plot	Image Height (%)	Image Height (mm)	CRA (Deg)
	0	0	0
	5	0.110	2.79
	10	0.220	5.39
	15	0.330	7.85
	20	0.440	10.19
	25	0.550	12.45
	30	0.660	14.61
	35	0.770	16.68
	40	0.880	18.64
	45	0.990	20.45
	50	1.100	22.09
	55	1.210	23.51
	60	1.320	24.68
	65	1.430	25.61
	70	1.540	26.29
	75	1.650	26.75
	80	1.760	27.02
	85	1.870	27.14
	90	1.980	27.14
	95	2.090	27.06
	100	2.200	26.91

Electrical Specifications

Figure 49: Default Data Output Timing Diagram



EXTCLK

The electrical characteristics of the EXTCLK input are shown in Table 30. The EXTCLK input supports an AC-coupled sine-wave input clock or a DC-coupled square-wave input clock.

Table 30: Electrical Characteristics (EXTCLK)

Definition	Condition	Symbol	Min	Typ	Max	Unit
Input clock frequency		^{f}EXTCLK	6	16	27	MHz
Input clock period		^{t}EXTCLK	166.7	62.5	37	ns
Input clock amplitude (AC coupled sine wave)	External coupling cap value 50–100pF	VIN_AC	0.5	1.0	1.2	VPP
Input clock duty cycle			45	50	55	%
Input clock jitter		^{t}JITTER			300	ps
PLL VCO lock time		^{t}LOCK			6750[1]	EXTCLK cycles
Input pad capacitance				2.5		pF
Input HIGH leakage current	VIN = VDD_IO	IIH	–10	–	10	?A
Input LOW leakage current	VIN = DGND	IIL	–10	–	10	?A
Input HIGH voltage	Parallel pixel data interface	VIH	0.7 x VDD_IO	–	VDD_IO + 0.5	V
Input HIGH voltage	Serial pixel (CCP2) data interface	VIH	1.0	–	VAA + 0.3	V
Input LOW voltage (DC coupled)	At specified IIL	VIL	–0.5	–	0.3 x VDD_IO	V

Note: 6750 EXTCLK cycles or 1ms, whichever is smaller.

Parallel Pixel Data Interface

The electrical characteristics of the parallel pixel data interface (FV, LV, D_{OUT}(9:0), PIXCLK, SHUTTER, and FLASH outputs) are shown in Table 31.

Table 31: Electrical Characteristics (Parallel Pixel Data Interface)

Definition	Condition			Symbol	Min	Typ	Max	Unit	Note
Output HIGH voltage	At specified I_{OH}			V_{OH}	V_{DD_IO} – 0.4	–	–	V	
Output LOW voltage	At specified I_{OL}			V_{OL}	–	–	0.4	V	
Output HIGH current	Magnitude, at specified V_{OH}			I_{OH}	–	–	–7	mA	
Output LOW current	Magnitude, at specified V_{OL}			I_{OH}	–	–	7	mA	
Tri-state output leakage current	V_{IN} = V_{DD_IO} or GND			I_{OZ}	–10	–	10	?A	
Output pin slew	Programmable Slew = 7	V_{DD_IO} = 2.8V	C_{LOAD} = 30pF ±3pF	SR	0.68	–	190	V/ns	
			C_{LOAD} = 15pF ±3pF	SR	1.60	–	3.70	V/ns	
		V_{DD_IO} = 1.8V	C_{LOAD} = 30pF ±3pF	SR	0.26	–	0.50	V/ns	
			C_{LOAD} = 15pF ±3pF	SR	0.45	–	1.40	V/ns	
	Programmable Slew = 4	V_{DD_IO} = 2.8V	C_{LOAD} = 30pF ±3pF	SR	0.40	–	0.68	V/ns	
			C_{LOAD} = 15pF ±3pF	SR	0.59	–	0.77	V/ns	
		V_{DD_IO} = 1.8V	C_{LOAD} = 30pF ±3pF	SR	0.17	–	0.26	V/ns	
			C_{LOAD} = 15pF ±3pF	SR	0.21	–	0.37	V/ns	
	Programmable Slew = 1	V_{DD_IO} = 2.8V	C_{LOAD} = 30pF ±3pF	SR	0.26	–	0.34	V/ns	
			C_{LOAD} = 15pF ±3pF	SR	0.31	–	0.37	V/ns	
		V_{DD_IO} = 1.8V	C_{LOAD} = 30pF ±3pF	SR	0.12	–	0.16	V/ns	
			C_{LOAD} = 15pF ±3pF	SR	0.14	–	0.21	V/ns	
PIXCLK frequency	default			^{f}PIXCLK	6	–	64	MHz	
PIXCLK to data valid	default			^{t}PD	3.5	–	5	ns	1
PIXCLK to FV HIGH	default			^{t}PFH	–	2	5	ns	1
PIXCLK to LV HIGH	default			^{t}PLH	–	2	5	ns	1
PIXCLK to FV LOW	default			^{t}PFL	–	2	5	ns	1
PIXCLK to LV LOW	default			^{t}PLL	–	2	5	ns	1

Notes: 1. Valid for C_{LOAD} < 30pF on PIXCLK, D_{OUT}, LV, and FV. Loads must be matched as closely as possible.

Two-Wire Serial Register Interface

The electrical characteristics of the two-wire serial register interface (SCLK, SDATA) are shown in Table 32. The SCLK and SDATA signals feature fail-safe input protection, Schmitt trigger input, and suppression of input pulses of less than 50ns.

Table 32: Two-Wire Serial Register Interface Electrical Characteristics

Definition	Condition	Symbol	Min	Typ	Max	Unit
Input HIGH voltage		V_{IH}	0.7 x V_{DD_IO}	–	V_{DD_IO} + 0.5	V
Input LOW voltage		V_{IL}	–0.5	–	0.3 x V_{DD_IO}	V
Input leakage current	No pull-up resistor; V_{IN} = V_{DD_IO} or DGND	I_{IN}	–	–	10	?A
Output LOW voltage	At specified I_{OL}	V_{OL}	–	–	0.4	V
Output LOW current	At specified V_{OL}	I_{OL}	8.9	–	18.5	mA
Tri-state output leakage current		I_{OZ}	–	–	1	?A
Input pad capacitance		C_{IN}	–	–	6	pF
Load capacitance		C_{LOAD}	–	–	15	pF

Serial Pixel Data Interface

The electrical characteristics of the serial pixel data interface (CLK_P, CLK_N, DATA_P, and DATA_N) are shown in Table 33.

To operate the serial pixel data interface within the electrical limits of the CCP2 specification, VDD_IO (I/O digital voltage) is restricted to operate in the 1.7–1.9V range.

Table 33: Electrical Characteristics (Serial Pixel Data Interface)

Definition	Symbol	Min	Typ	Max	Unit
Operating frequency		1	–	320	MHz
Fixed common mode voltage	V_{CMF}	0.8	0.9	1	V
Differential voltage swing	V_{OD}	100	150	200	mV
Drive current range		0.83	1.5	2	mA
Drive current variation		–	–	15	%
Output impedance		40	–	140	W
Output impedance mismatch		–	–	10	%
Clock duty cycle at 416 MHz		45	50	55	%
Rise time (20–80%)	V_{OD}	300	–	400	ps
Fall time (20–80%)	V_{OD}	300	–	400	ps
Differential skew		–	–	500	ps
Channel-to-channel slew		–	–	200	ps
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) 100–1000 MHz		10	–	–	dB

Control Interface

The electrical characteristics of the control interface (RESET_BAR, SADDR, TEST, GPI0, GPI1, GPI2, and GPI3) are shown in Table 34.

Table 34: Electrical Characteristics (Control Interface)

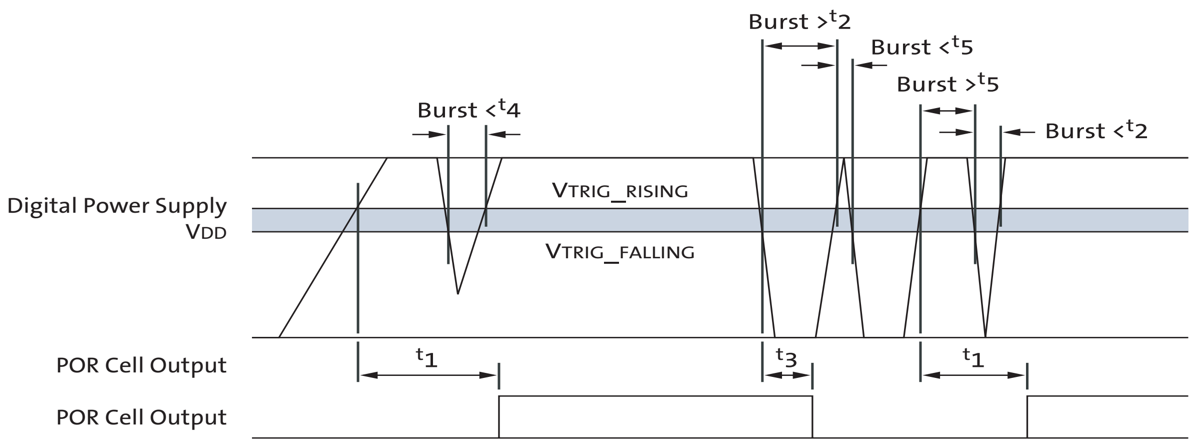
Definition	Condition	Symbol	Min	Typ	Max	Unit
Input HIGH voltage	Parallel pixel data interface	V_{IH}	0.7 x V_{DD_IO}	–	V_{DD_IO} + 0.5	V
Input HIGH voltage	Serial pixel (CCP2) data interface	V_{IH}	0.7 x V_{DD_IO}	–	V_{AA} + 0.3	V
Input LOW voltage		V_{IL}	–0.3	–	0.3 x V_{DD_IO}	V
Input leakage current	No pull-up resistor; V_{IN} = V_{DD_IO} or D_{GND}	I_{IN}	–	–	10	?A
Input pad capacitance		C_{IN}	–	6.5	–	pF

Power-On Reset

Table 35: Power-On Reset Characteristics

Definition	Condition	Symbol	Min	Typ	Max	Unit
VDD rising, crossing VTRIG_RISING; Internal reset being released		t_1	7	10	15	?s
VDD falling, crossing VTRIG_FALLING; Internal reset active		t_2	–	0.5	1	?s
Minimum VDD spike width below VTRIG_FALLING; considered to be a reset when POR cell output is HIGH		t_3	–	0.5	–	
Minimum VDD spike width below VTRIG_FALLING; considered to be a reset when POR cell output is LOW		t_4	–	1	–	?s
Minimum VDD spike width above VTRIG_RISING; considered to be a stable supply when POR cell output is LOW	While the POR cell output is LOW, all VDD spikes above VTRIG_RISING less than t_5 must be ignored	t_5	–	50	–	ns
VDD rising trigger voltage		VTRIG_RISING	1.12	–	1.55	V
VDD falling trigger voltage		VTRIG_FALLING	1.0	–	1.45	V

Figure 50: Internal Power-On Reset



Operating Voltages

VAA and VAA_PIX must be at the same potential for correct operation of the MT9D012.

Table 36: DC Electrical Definitions and Characteristics
Dark lighting conditions; CLOAD 15–30pF

Definition	Condition	Symbol	Min	Typ	Max	Unit
Core digital voltage		VDD	1.7	1.8	1.9	V
I/O digital voltage	Parallel pixel data interface	VDD_IO	1.7	1.8/2.8	3.1	V
I/O digital voltage	Serial pixel (CCP2) data interface	VDD_IO	1.7	1.8	1.9	V
Analog voltage		VAA	2.4	2.8	3.1	V
Pixel supply voltage		VAA_PIX	2.4	2.8	3.1	V
PLL supply voltage		VDD_PLL	2.4	2.8	3.1	V
Digital operating current	Streaming, full resolution, 23 fps, CCP Class 2, 640 Mb/s	IDD1	–	23	38	mA
I/O digital operating current	Streaming, full resolution, 23 fps, CCP Class 2, 640 Mb/s	IDD_IO	–	8	15	mA
Analog operating current	Streaming, full resolution, 23 fps, CCP Class 2, 640 Mb/s	IAA	–	41	60	mA
Pixel supply current	Streaming, full resolution, 23 fps, CCP Class 2, 640 Mb/s	IAA_PIX	–	0.7	5	mA
PLL supply current	Streaming, full resolution, 23 fps, CCP Class 2, 640 Mb/s	IDD_PLL	–	4.7	10	mA
Hard standby is specified at 25°C	Analog		–	0.2	5	?A
	Digital		–	7	10	?A
Soft standby (clock off)	Analog		–	0.2	50	?A
	Digital		–	65	120	?A
Soft standby (clock on (6 MHz))	Analog		–	3	50	?A
	Digital		–	185	500	?A

Absolute Maximum Ratings

Caution **Stresses greater than those listed in Table 37 may cause permanent damage to the device.**

Table 37: Absolute Maximum Values

Definition	Condition	Symbol	Min	Max	Unit
Core digital voltage		VDD_MAX	–0.3	2.2	V
I/O digital voltage		VDD_IO_MAX	–0.3	3.2	V
Analog voltage		VAA_MAX	–0.3	3.2	V
Pixel supply voltage		VAA_PIX_MAX	–0.3	3.2	V
PLL supply voltage		VDD_PLL_MAX	–0.3	3.2	V
Input HIGH voltage	Parallel pixel data interface	VIH_MAX	0.7 x VDD_IO	VDD_IO + 0.5	V
Input HIGH voltage	Serial pixel (CCP2) data interface	VIH_MAX	0.7 x VDD_IO	VAA + 0.3	V
Input LOW voltage		VIH_MAX	–0.3	0.3 x VDD_IO	V
Operating temperature	Measure at junction	TOP	–30	70	°C
Storage temperature		TSTG	–40	125	°C

Note: 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.

Table 38: MT9D012 Two-Wire Serial Interface Timing Specifications
VDD = 1.8V; VAA = 2.8V; VAA_PIX = 2.8V,;VDD_IO =1.8/2.8V; VDD_PLL = 2.8V; T_J = –30°C to +70°C, unless otherwise specified.

Symbol	Definition	Minimum	Typical	Maximum	Units
^{f}SCLK	Serial interface input clock frequency	0	100	400	KHz
^{t}ICH	SHIP clock period high	0.4 * SCLK	–	0.6 * SCLK	ns
^{t}ICL	SHIP clock period low	0.4 * SCLK	–	0.6 * SCLK	ns
^{t}ISS	Setup time for start condition	820			ns
^{t}IHS	Hold time for start condition	660			ns
^{t}ISD	Setup time for input data	300			ns
^{t}IHD	Hold time for input data	300			ns
^{t}OAA	Output data acknowledge time			350[1]	ns
				1,200[2]	
^{t}ODA	Output delay time	550[1]			ns
		1200[2]			
^{t}ISP	Setup time for stop conditon	600			ns
^{t}IHP	Hold time for stop condition	300			ns

Notes: 1. Master clock frequency of 50 MHz.
2. Master clock frequency of 6 MHz.

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. M. **5/25/10**

- Updated to non-confidential

Rev L. **5/3/10**

- Updated to Aptina template

Rev K, Production **7/24/2008**

- Updated default value of R0x0002 in Table 8, "SMIA Configuration," on page 1
- Updated description of R0x002 in Table 11, SMIA Configuration
- Updated description for R0x3160[2] in Table 13 on p. 54
- Added notes after "Power-Up Sequence" on page 66
- Removed Max column from Table 28, "Power-Up Sequence," on page 67 and from Table 29, "Power-Down Sequence," on page 68
- Added Table 38, "MT9D012 Two-Wire Serial Interface Timing Specifications," on page 78

Rev. J, Production **3/12/2008**

- Updated Figure 45: "Power-Down Sequence," on page 67

Rev. H **08/06/2007**

- Update Table 1, "Key Performance Parameters," on page 1 (power consumption)

Rev. G, Preliminary **07/11/2007**

- Update "General Description" on page 1
- Update Table 16, "Programming Example 1," on page 35
- Update "Frame Rates at Common Image Sizes" on page 47
- Update Table 25, "Frame Rates," on page 47
- Remove "CRA vs. Image Height, Type A," figure on page 111
- Update Table 30, "Electrical Characteristics (EXTCLK)," on page 72
- Update Table 31, "Electrical Characteristics (Parallel Pixel Data Interface)," on page 73
- Update Table 32, "Two-Wire Serial Register Interface Electrical Characteristics," on page 74
- Update Table 36, "DC Electrical Definitions and Characteristics," on page 77

Rev. F, Preliminary **05/21/2007**

- Update Table 1, "Key Performance Parameters," on page 1
- Update Table 2, "Available Part Numbers," on page 1
- Update Table 3, "Typical Configuration: Parallel Pixel Data Interface," on page 9 Notes
- Update "Camera Module Integrator Identification Procedure" on page 10
- Update Table 4, "Signal Descriptions," on page 11
- Update"CCP Serial Pixel Data Interface" on page 12
- Update "General Purpose Inputs" on page 29
- Update Table 37, "Absolute Maximum Values," on page 78

Rev. E, Preliminary **04/20/2007**

- Update global VDDQ to VDDIO; and RESET# to RESET_BAR
- Update Table 1, "Key Performance Parameters," on page 1
- Add "Camera Module Integrator Identification Procedure" on page 10
- Update Table 5, "Row Timing," on page 14

- Update "Two-Wire Serial Register Interface" on page 15
- Update "Typical Sequence" on page 16
- Update Figure 7: "Single READ from Random Location," on page 17
- Update Figure 8: "Single READ from Current Location," on page 17
- Update Figure 9: "Sequential READ, Start from Random Location," on page 17
- Update Figure 10: "Sequential READ, Start from Current Location," on page 18
- Update Figure 11: "Single WRITE to Random Location," on page 18
- Update Figure 12: "Sequential WRITE, Start at Random Location," on page 18
- Update
- Table 8, "SMIA Configuration," on page 1
- Table 9, "SMIA Parameter Limits," on page 3
- Table 10, "Manufacturer-Specific," on page 5
- Table 11, "SMIA Configuration," on page 9
- Table 12, "SMIA Parameter Limits," on page 14
- Table 13, "Manufacturer-Specific," on page 19
- Update "Output Size Restrictions" on page 21
- Update "Effect of CCP Class on Legal Range of Output Sizes/Frame Rate" on page 23
- Update "Programming Restrictions when Using Global Reset" on page 24
- Update "VCO = voltage-controlled oscillator." on page 28
- Update Table 11, "Signal State During Reset," on page 29
- Update "General Purpose Inputs" on page 29
- Update "Programming Examples" on page 35
- Update Table 16, "Programming Example 1," on page 35
- Update "Pixel Border" on page 40
- Update "Integration Time" on page 47
- Add "Effect of Data Path Processing on Test Patterns" on page 57
- Update "Digital Lens/Color Shading Correction" on page 62
- Update "Parallel Pixel Data Output MUX" on page 64
- Add "Timing Specifications" on page 66
- Add "Power-Up Sequence" on page 66
- Add Table 28, "Power-Up Sequence," on page 67
- Add Figure 44: "Power-Up Sequence," on page 66
- Add "Power-Down Sequence" on page 67
- Add Table 29, "Power-Down Sequence," on page 68
- Add Figure 45: "Power-Down Sequence," on page 67
- Update Figure 47: "Quantum Efficiency," on page 70
- Add Figure 48: "CRA vs. Image Height," on page 71
- Add Figure 48: "CRA vs. Image Height," on page 71
- Update Table 36, "DC Electrical Definitions and Characteristics," on page 77

Rev. D, Preliminary...**10/04/06**

- Update Table 1, "Key Performance Parameters," on page 1
- Update Table 37, "Absolute Maximum Values," on page 78

Rev. C, Preliminary...**9/27/06**

- Update Table 1, "Key Performance Parameters," on page 1
- Update Table 37, "Absolute Maximum Values," on page 78

Rev. B, Preliminary 7/06

- Update Table 1, "Key Performance Parameters," on page 1
- Update "Functional Overview" on page 7
- Update Figure 3: "Typical Configuration: Parallel Pixel Data Interface," on page 9
- Update Table 5, "Row Timing," on page 14
- Update "Registers" on page 19
- Update "Changes to Gain Settings" on page 22
- Add "Register Map" on page 1
- Add Table 8, "SMIA Configuration," on page 1
- Add Table 9, "SMIA Parameter Limits," on page 3
- Add Table 10, "Manufacturer-Specific," on page 5
- Update Table 13, "Manufacturer-Specific," on page 19
- Updated Table 13, "Manufacturer-Specific," on page 19
- Update "Configuration of the Pixel Data Interface" on page 26
- Update "Clocking" on page 31
- Add "Programming Examples" on page 35
- Update Table 18, "Programming Example 3," on page 36
- Update "Effect of Data Path Processing on Test Patterns" on page 57
- Update "Digital Lens/Color Shading Correction" on page 62
- Update Figure 42: "Lens Correction Zones," on page 63
- Update "Parallel Pixel Data Output MUX" on page 64
- Update Figure 46: "Die Outline," on page 69
- Update Table 30, "Electrical Characteristics (EXTCLK)," on page 72
- Update Table 31, "Electrical Characteristics (Parallel Pixel Data Interface)," on page 73
- Update Table 32, "Two-Wire Serial Register Interface Electrical Characteristics," on page 74
- Update Table 33, "Electrical Characteristics (Serial Pixel Data Interface)," on page 74
- Remove "Valid Data Signal Options" section and LINE_VALID Formats figure

Rev. A, Advance 2/06

- Initial release