1/2.5-Inch 5Mp CMOS Digital Image Sensor

MT9P014 Data Sheet

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

Features

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

Applications

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

General Description

The Aptina™ MT9P014 is a 1/2.5-inch CMOS active-pixel digital image sensor with a pixel array of 2592H x 1944V (2608H x 1960V including border pixels). It incorporates sophisticated on-chip camera functions such as windowing, mirroring, column and row skip modes, and snapshot mode. It is programmable through a simple two-wire serial interface and has very low power consumption.

Table 1: Key Performance Parameters

Parameter		Value
Optical format		1/2.5-inch (4:3)
Active imager size		5.70mm(H) x 4.28mm(V) 7.13mm diagonal
Active pixels		2592H x 1944V
Pixel size		2.2 x 2.2µm
Chief ray angle		25
Color filter array		RGB Bayer pattern
Shutter type		Electronic rolling shutter (ERS) with global reset release (GRR)
Input clock frequency		6–27 MHz
Maximum data rate	CCP2	650 Mbps
	MIPI	768 Mbps per lane
Frame rate	Full resolution	15 fps
	VGA	640H x 480V with 2X skip and 2X bin: 70 fps
	HD	720p60 and 1080p30
ADC resolution		12-bit
Responsivity		1.17 V/Lux-sec
Dynamic range		69dB
SNR_{MAX}		39dB
Supply voltage	I/O Digital	1.7–1.9V (1.8V nominal)
	Digital	1.7–1.9V (1.8V nominal)
	Analog	2.4–3.1V (2.8V nominal)
Power Consumption	Full resolution	332mW at 1.8V/2.8V, (T_J= ambient, CCP2 = 640 Mbps, 15 fps, RAW8)
	Standby	69.8µW @1.8V/2.8V, (T_J= ambient, Hard standby, clock off)
Package		Bare die
Operating temperature		–30°C to +70°C (at junction)

Ordering Information

Table 2: Available Part Numbers

Part Number	Description
MT9P014D00STCPC28CC1	Bare die (CCP)
MT9P014D00STCMC28CC1	Bare die (MIPI)

Table of Contents

List of Figures

List of Tables

General Description

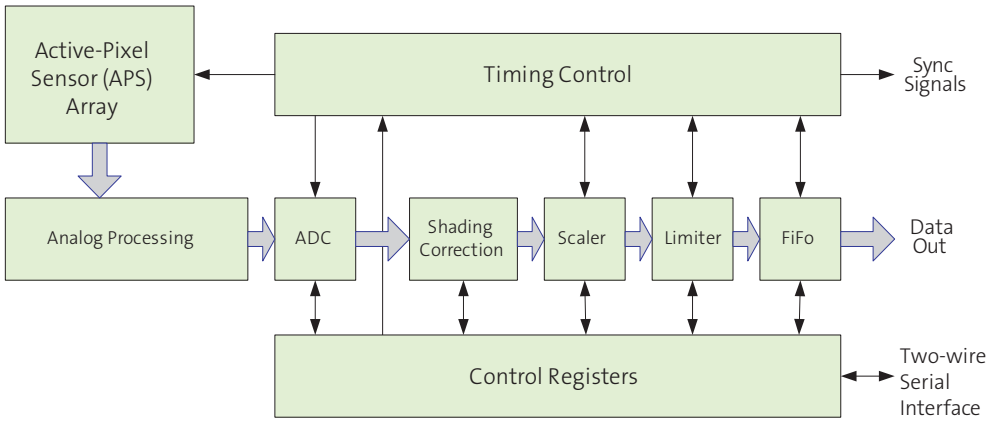
The MT9P014 digital image sensor features DigitalClarity—our breakthrough, low-noise CMOS imaging technology that achieves near-CCD image quality (based on signal-to-noise ratio and low-light sensitivity) while maintaining the inherent size, cost, and integration advantages of CMOS.

When operated in its default mode, the sensor generates a full resolution image at 15 frames per second (fps). An on-chip analog-to-digital converter (ADC) generates a 12-bit value for each pixel.

Functional Overview

The MT9P014 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. A block diagram of the sensor is shown in Figure 1.

Figure 1: Block Diagram



The core of the sensor is a 5Mp active-pixel array. In electronic shutter rolling (ERS) mode, the timing and control circuitry sequences through the rows of the array, resetting and then reading each row in turn. In the time interval between resetting a row and reading that row, the pixels in the row integrate incident light. The exposure is controlled by varying the time interval between reset and readout. Once a row has been read, the data from the columns is sequenced through an analog signal chain (providing offset correction and gain), and then through an ADC. The output from the ADC is a 12-bit value for each pixel in the array. The ADC output passes through a digital processing signal chain (which provides further data corrections and applies digital gain).

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

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

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

The control registers, timing and control, and digital processing functions shown in Figure 1 on page 7 are partitioned into three logical parts:

- A sensor core that provides array control and data path corrections. The output of the sensor core is a 12-bit pixel data.
- A digital shading correction block to compensate for color/brightness shading introduced by the lens or chief ray angle (CRA) curve mismatch.
- Additional functionality, including a horizontal and vertical image scaler, a limiter, a data compressor, an output FIFO, and a serializer.

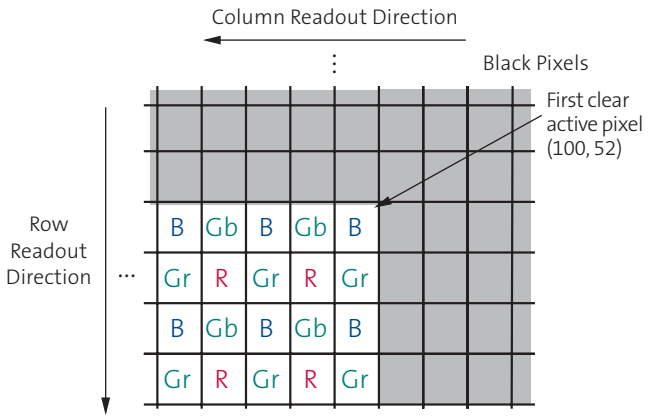
The output FIFO is present to prevent data bursts by keeping the data rate continuous.

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

Pixel Array

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

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



Operating Modes

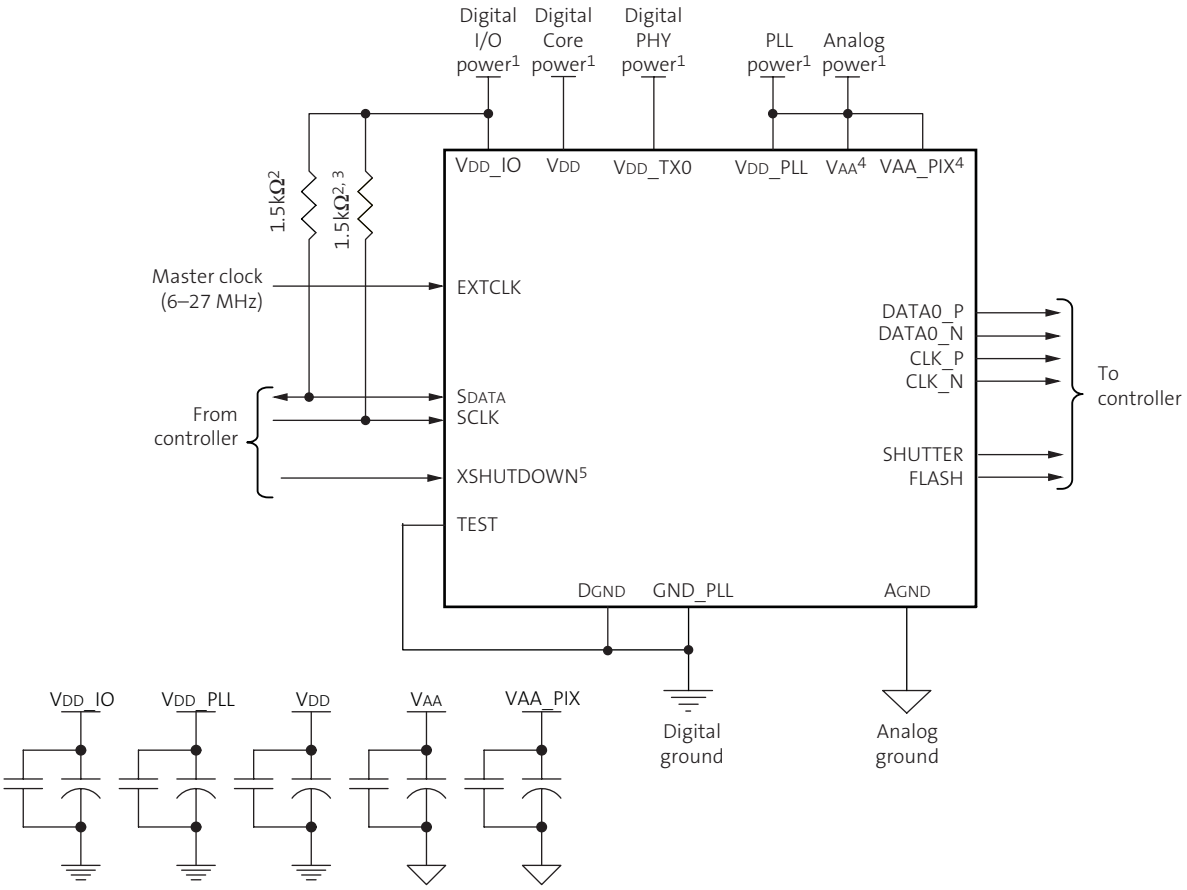
The MT9P014 can operate in either serial CPP2 or serial MIPI mode (preconfigured at the factory). In both cases, the sensor has an SMIA-compatible register interface while the I^2C device address is compliant with SMIA or MIPI requirements as appropriate. The reset level on the TEST pin must be tied in a way that is compatible with the configured serial interface of the sensor, for instance TEST = 0 for CCP2 and TEST = 1 for MIPI.

Typical configurations are shown in Figure 3 on page 10 and Figure 4 on page 11. These operating modes are described in "Control of the Signal Interface" on page 26.

For low-noise operation, the MT9P014 requires separate power supplies for analog and digital. Incoming digital and analog ground conductors can be tied together next to the die. Both power supply rails should be decoupled from ground using capacitors as close as possible to the die.

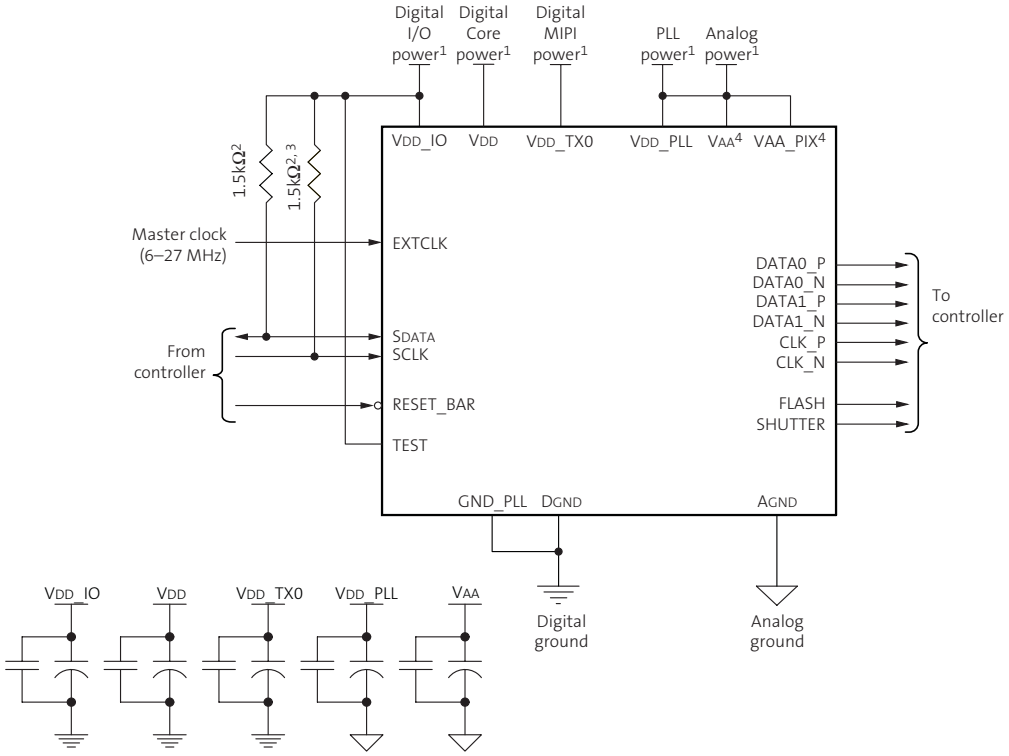
Caution **Aptina does not recommend the use of inductance filters on the power supplies or output signals.**

Figure 3: Typical Configuration: Serial CCP2 Pixel Data Interface



Notes:
1. All power supplies should be adequately decoupled.
2. Aptina recommends a resistor value of 1.5kΩ, but 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. VAA and VAA_PIX must be tied together.
5. Also referred to as RESET_BAR
6. VPP, which can be used during the module manufacturing process, is not shown in the figure above. This pad is left unconnected during normal operation.
7. Aptina recommends that 0.1µF and 1µF decoupling capacitors for each power supply are mounted as close as possible to the pad. Actual values and results may vary depending on layout and design considerations.
8. TEST must be tied to DGND.
9. Aptina recommends that GND_PLL be tied to DGND.
10. Aptina recommends that VDD_TX be tied to VDD.

Figure 4: Typical Configuration: Serial Dual-Lane MIPI 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. VAA and VAA_PIX must be tied together.
5. VPP, which can be used during the module manufacturing process, is not shown in Figure 4. This pad is left unconnected during normal operation.
6. Aptina recommends that 0.1µF and 1µF decoupling capacitors for each power supply are mounted as close as possible to the pad. Actual values and results may vary depending on layout and design considerations.
7. TEST must be tied to VDD_IO.
8. Aptina recommends that GND_PLL be tied to DGND.
9. Aptina recommends that VDD_TX be tied to VDD.

Signal Descriptions

Table 3 provides signal descriptions for MT9P014 die. For pad location and aperture information, refer to the MT9P014 die data sheet.

Table 3: Signal Descriptions

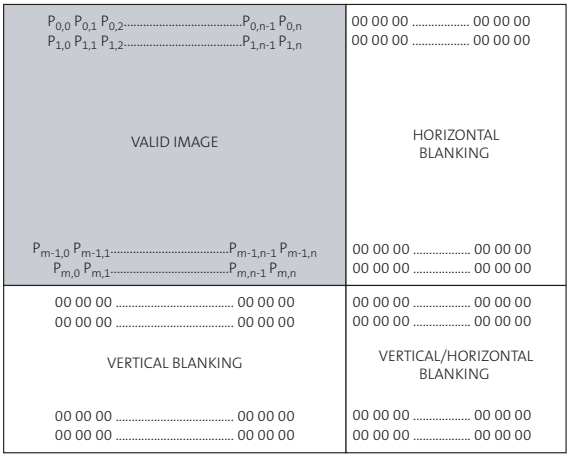
Pad Name	Pad Type	Description
EXTCLK	Input	Master clock input, 6–27 MHz.
RESET_BAR (XSHUTDOWN)	Input	Asynchronous active LOW reset. When asserted, data output stops and all internal registers are restored to their factory default settings.
SCLK	Input	Serial clock for access to control and status registers.
GPI[3:0]	Input	General purpose inputs. After reset, these pads are powered down by default; this means that it is not necessary to bond to these pads. Any of these pads can be configured to provide hardware control of the standby, output enable, SADDR select, and shutter trigger functions. Can be left floating if not used.
TEST	Input	Enable manufacturing test modes. Connect to DGND for normal operation of the CCP2 configured sensor, or connect to VDD_IO power for the MIPI configured sensor.
SDATA	I/O	Serial data from READS and WRITES to control and status registers.
DATA0_P	Output	Differential CCP2/MIPI (sub-LVDS) serial data (positive).
DATA0_N	Output	Differential CCP2/MIPI (sub-LVDS) serial data (negative).
DATA1_P	Output	Differential MIPI (sub-LVDS) serial data 2nd lane (positive). Can be left floating when using 1-lane MIPI or CCP2 serial interface.
DATA1_N	Output	Differential MIPI (sub-LVDS) serial data 2nd lane (negative). Can be left floating when using 1-lane MIPI or CCP2 serial interface.
CLK_P	Output	Differential CCP2/MIPI (sub-LVDS) serial clock/strobe (positive).
CLK_N	Output	Differential CCP2/MIPI (sub-LVDS) serial clock/strobe (negative).
FLASH	Output	Flash output. Synchronization pulse for external light source. Can be left floating if not used.
SHUTTER	Output	Control for external mechanical shutter. Can be left floating if not used.
VPP	Supply	Power supply used to program one-time programmable (OTP) memory. Disconnect pad when not programming or when feature is not used.
VDD_TX0	Supply	PHY power supply. Digital power supply for the serial interface.
VAA	Supply	Analog power supply.
VAA_PIX	Supply	Analog power supply for the pixel array.
AGND	Supply	Analog ground.
VDD	Supply	Digital power supply.
VDD_IO	Supply	I/O power supply.
DGND	Supply	Common ground for digital and I/O.
VDD_PLL	Supply	PLL power supply.
GND_PLL	Supply	PLL ground.

Output Data Format

Serial Pixel Data Interface

The MT9P014 serial pixel data interface implements data/clock and data/strobe signaling in accordance with the CCP2 and MIPI specifications. The RAW8, RAW10, and RAW12 image data formats are supported.

Figure 5: Spatial Illustration of Image Readout



The sensor timing (Table 4) is shown in terms of pixel clock and master clock cycles. The default settings for the on-chip PLL generate a 60 MHz output pixel clock (op_pix_clk) given a 24 MHz input clock to the MT9P014. Equations for calculating the frame rate are given in "Frame Rate Control" on page 46.

Table 4: Row Timing

Parameter	Name	Equation	Default Timing
PIXCLK_PERIOD	Pixel clock period	R0x3016–7[2:0] / vt_pix_clk_freq_mhz	1 pixel clock = 8.33ns
S	Skip (subsampling) factor	For x_odd_inc = y_odd_inc = 3, S = 2. For x_odd_inc = y_odd_inc = 7, S = 4. otherwise, S = 1	1
A	Active data time	(x_addr_end - x_addr_start + x_odd_inc) * PIXCLK_PERIOD/S	2592 pixel clocks = 21.60µs
P	Frame start/end blanking	6 * PIXCLK_PERIOD	6 pixel clocks = 50.00ns
Q	Horizontal blanking	(line_length_pck - A)	2780 pixel clocks = 23.17µs
A + Q	Row time	line_length_pck * PIXCLK_PERIOD	5372 pixel clocks = 44.77µs
N	Number of rows	(y_addr_end - y_addr_start + y_odd_inc) / S	1944 rows
V	Vertical blanking	((frame_length_lines - N) * (A+Q)) + Q - (2*P)	604432 pixel clocks = 5.04ms
T	Frame valid time	(N * (A + Q)) - Q + (2*P)	10440400 pixel clocks = 87.00ms
F	Total frame time	line_length_pck * frame_length_lines * PIXCLK_PERIOD	11044832 pixel clocks = 92.04ms

Two-Wire Serial Register Interface

The two-wire serial interface bus enables read/write access to the control and status registers within the MT9P014. This interface is designed to be compatible with the electrical characteristics and transfer protocols of the I^2C specification.

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

The protocols described in the two-wire serial interface specification allow the slave device to drive SCLK LOW; the MT9P014 uses SCLK as an input only and therefore never drives it LOW.

Protocol

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

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

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

Start Condition

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

Stop Condition

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

Data Transfer

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

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

Slave Address/Data Direction Byte

Bits [7:1] of this byte represent the device slave address and bit [0] indicates the data transfer direction. A “0” in bit [0] indicates a write, and a “1” indicates a read. The default slave addresses used by the MT9P014 for the MIPI configured sensor are 0x6C (write address) and 0x6D (read address) in accordance with the MIPI specification. Alternate slave addresses of 0x6E(write address) and 0x6F(read address) can be selected by

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

An alternate slave address can also be programmed through R0x31FC.

Message Byte

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

Acknowledge Bit

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

No-Acknowledge Bit

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

Typical Sequence

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

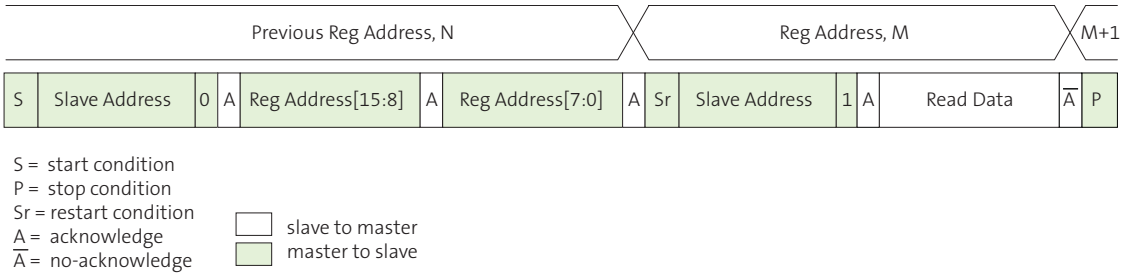
If the request was a WRITE, the master then transfers the 16-bit register address to which the write should take place. This transfer takes place as two 8-bit sequences and the slave sends an acknowledge bit after each sequence to indicate that the byte has been received. The master then transfers the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

If the request was a READ, the master sends the 8-bit write slave address/data direction byte and 16-bit register address, the same way as with a write request. The master then generates a (re)start condition and the 8-bit read slave address/data direction byte, and clocks out the register data, 8 bits at a time. The master generates an acknowledge bit after each 8-bit transfer. The slave’s internal register address is automatically incremented after every 8 bits are transferred. The data transfer is stopped when the master sends a no-acknowledge bit.

Single READ from Random Location

This sequence (Figure 6) 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 6 shows how the internal register address maintained by the MT9P014 is loaded and incremented as the sequence proceeds.

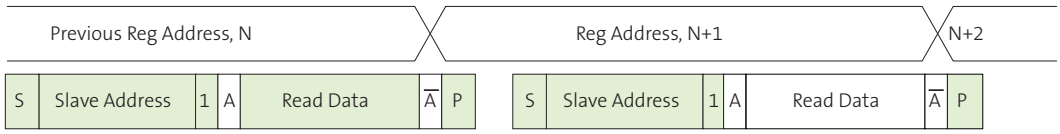
Figure 6: Single READ from Random Location



Single READ from Current Location

This sequence (Figure 7) performs a READ using the current value of the MT9P014 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.

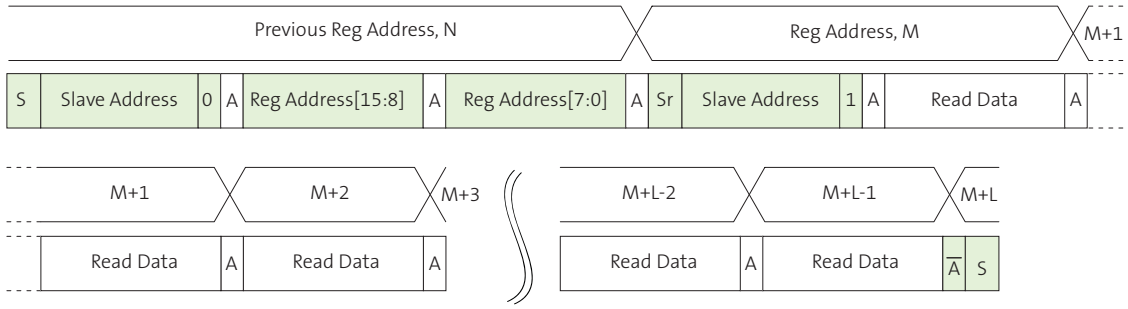
Figure 7: Single READ from Current Location



Sequential READ, Start from Random Location

This sequence (Figure 8 on page 17) starts in the same way as the single READ from random location (Figure 6). 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.

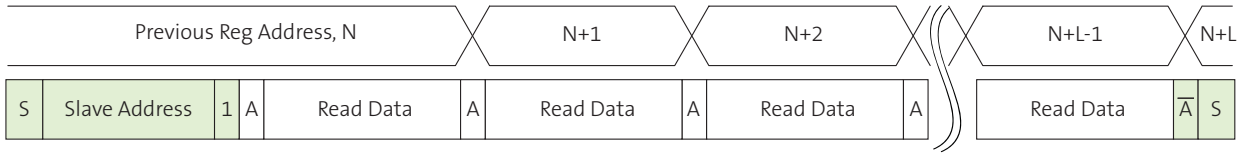
Figure 8: Sequential READ, Start from Random Location



Sequential READ, Start from Current Location

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

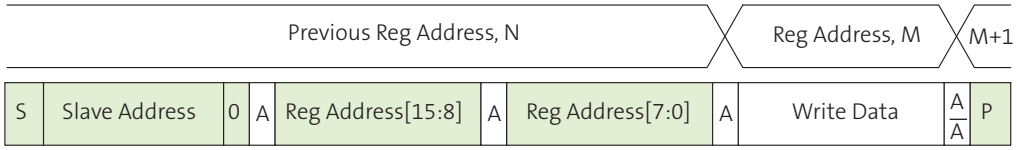
Figure 9: Sequential READ, Start from Current Location



Single WRITE to Random Location

This sequence (Figure 10) 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.

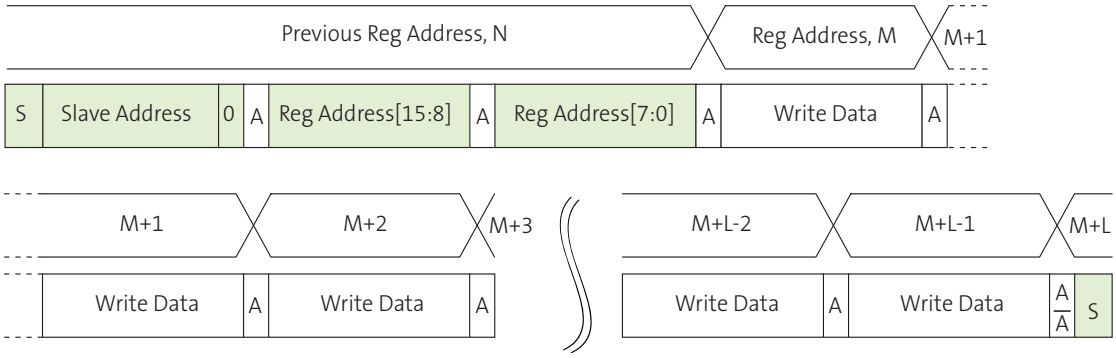
Figure 10: Single WRITE to Random Location



Sequential WRITE, Start at Random Location

This sequence (Figure 11) starts in the same way as the single WRITE to random location (Figure 10). Instead of generating a no-acknowledge bit after the first byte of data has been transferred, the master generates an acknowledge bit and continues to perform byte WRITEs until "L" bytes have been written. The WRITE is terminated by the master generating a stop condition.

Figure 11: Sequential WRITE, Start at Random Location



Registers

The MT9P014 provides a 16-bit register address space accessed through a serial interface ("Two-Wire Serial Register Interface" on page 14). Each register location is 8 or 16 bits in size. For detailed information on the registers, see the MT9P014 Register Reference.

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

Table 5: Address Space Regions

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

Register Notation

The underlying mechanism for reading and writing registers provides byte write capability. However, it is convenient to consider some registers as multiple adjacent bytes. The MT9P014 uses 8-bit, 16-bit, and 32-bit registers, all implemented as 1 or more bytes at naturally aligned, contiguous locations in the address space.

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

Register Aliases

A consequence of the internal architecture of the MT9P014 is that some registers are decoded at multiple addresses. Some registers in "configuration space" are also decoded in "manufacturer-specific space." To provide unique names for all registers, the name of the register within manufacturer-specific register space has a trailing underscore. For example, R0x0000–1 is model_id, and R0x3000–1 is model_id_. The effect of reading or writing a register through any of its aliases is identical.

Bit Fields

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

Bit Field Aliases

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

Byte Ordering

Registers that occupy more than one byte of address space are shown with the lowest address in the highest-order byte lane to match the byte-ordering on the data bus. For example, the model_id register is R0x0000–1. In the register table the default value is shown as 0x2600. This means that a read from address 0x0000 would return 0x26, and a read from address 0x0001 would return 0x00. When reading this register as two 8-bit transfers on the serial interface, the 0x26 will appear on the serial interface first, followed by the 0x00.

Address Alignment

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

Bit Representation

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

Data Format

Most registers represent an unsigned binary value or set of bit fields. For all other register formats, the format is stated explicitly at the start of the register description. The notation for these formats is shown in Table 6.

Table 6: Data Formats

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

Register Behavior

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

Double-Buffered Registers

Some sensor settings cannot be changed during frame readout. For example, changing R0x0344–5 (x_addr_start) partway through frame readout would result in inconsistent row lengths within a frame. To avoid this, the MT9P014 double-buffers many registers by implementing a "pending" and a "live" version. Reads and writes access the pending register. The live register controls the sensor operation.

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

Using grouped_parameter_hold

Register grouped_parameter_hold (R0x0104) can be used to inhibit transfers from the pending to the live registers. When the MT9P014 is in streaming mode, this register should be set to "1" before making changes to any group of registers where a set of changes is required to take effect simultaneously. When this register is set to "0," all transfers from pending to live registers take place on the next frame start.

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

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

Bad Frames

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

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

By default, bad frames are not masked.

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

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

Changes to Integration Time

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

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

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

Changes to Gain Settings

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

Embedded Data

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

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

- The integration time is delayed by one further frame, so that the value corresponds to the integration time used for the image data in the frame. See “Changes to Integration Time” on page 21.
- The PLL timing registers are not double-buffered because the result of changing them in streaming mode is undefined. Therefore, the pending and live values for these registers are equivalent.

For further details, see “Embedded Data Format and Control” on page 64.

Programming Restrictions

Table 8 shows a list of programming rules that must be adhered to for correct operation of the MT9P014. Aptina recommends that these rules be encoded into the device driver—either implicitly or explicitly.

Table 7: Definitions for Programming Rules

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

Table 8: Programming Rules

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

Table 8: Programming Rules (continued)

Parameter	Minimum Value	Maximum Value
x_output_size	256	2608
x_output_size[0]	0 (this is enforced in hardware: bit[0] is read-only)	0
y_output_size	2	frame_length_lines
y_output_size[0]	0 (this is enforced in hardware: bit[0] is read-only)	0

Output Size Restrictions

The design specification imposes the restriction that an output line shall be a multiple of 32 bits in length. This imposes an additional restriction on the legal values of x_output_size:

- When ccp_data_format[7:0] = 8 (RAW8 data), x_output_size must be a multiple of 4 (x_output_size[1:0] = 0).
- When ccp_data_format[7:0] = 10 (RAW10 data), x_output_size must be a multiple of 16 (x_output_size[3:0] = 0).
- When ccp_data_format[7:0] = 12 (RAW12 data), x_output_size must be a multiple of 8 (x_output_size[3:0] = 0).

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

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

Effect of Scaler on Legal Range of Output Sizes

When the scaler is enabled, it is necessary to adjust the values of x_output_size and y_output_size to match the image size generated by the scaler. The MT9P014 will operate incorrectly if the x_output_size and y_output_size are significantly larger than the output image. To understand the reason for this, consider the situation where the sensor is operating at full resolution and the scaler is enabled with a scaling factor of 32 (half the number of pixels in each direction).

Output Data Timing

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

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

A key constraint when configuring the clock for the output FIFO is that the frame rate out of the FIFO must exactly match the frame rate into the FIFO. When the scaler is disabled, this constraint can be met by imposing the rule that the row time on the serial

data stream must be greater than or equal to the row time at the pixel array. The row time on the serial data stream is calculated from the x_output_size and the ccp_data_format (8, 10, or 12 bits per pixel), and must include the time taken in the serial data stream for start of frame/row, end of row/frame and checksum symbols.

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

Changing Registers while Streaming

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

- ccp2_channel_identifier
- ccp_data_format
- ccp_signaling_mode
- vt_pix_clk_div
- vt_sys_clk_div
- pre_pll_clk_div
- pll_multiplier
- op_pix_clk_div
- op_sys_clk_div

Programming Restrictions when Using Global Reset

Interactions between the registers that control the global reset impose some programming restrictions on the way in which they are used; these are discussed in "Global Reset" on page 50.

Control of the Signal Interface

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

Serial Register Interface

The serial register interface uses these signals:

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

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

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

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

This interface is described in detail in "Two-Wire Serial Register Interface" on page 14.

Default Power-Up State

The MT9P014 provides interfaces for pixel data through the CCP2 high speed serial interface described by the SMIA specification or the MIPI serial interface.

At power up and after a hard or soft reset, the reset state of the MT9P014 is to enable the SMIA CCP2 high speed serial interface for a CCP2-configured sensor, and CSI-2 high speed serial interface for a MIPI-configured sensor.

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

Serial Pixel Data Interface

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

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

The signal pairs are driven differentially using sub-LVDS switching levels. This interface conforms to the MIPI 1.0 CSI-2 and SMIA 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. DATA0_P and DATA0_N are the data pair for the CCP2 or 1-lane MIPI serial interface, while DATA1_P and DATA1_N are the second data pair for 2-lane serial MIPI.

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

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

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

- 0x0A0A – sensor supports RAW10 uncompressed data format. A sensor with a 12-bit ADC can support this mode by discarding all but the upper 10 bits of a pixel value.
- 0x0C0C – sensor supports RAW12 uncompressed data format
- 0x0808 – sensor supports RAW8 uncompressed data format. A sensor with a 12-bit or 10-bit ADC can support this mode by discarding all but the upper 8 bits of a pixel value.
- 0x0A08 – sensor supports RAW8 data format in which an adaptive compression algorithm is used to perform 10-bit to 8-bit compression on the upper 10 bits of each pixel value.
- 0x0C08 – sensor supports RAW8 data format in which an adaptive compression algorithm is used to perform 12-bit to 8-bit compression on the upper 12 bits of each pixel value.

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

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

System States

The system states of the MT9P014 are represented as a state diagram in Figure 12 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 9 on page 29.

The sensor's operation is broken down into three separate states: hardware standby, software standby, and streaming. The transition between these states might take a certain amount of clock cycles as outlined in Table 9 on page 29.

Figure 12: MT9P014 System States

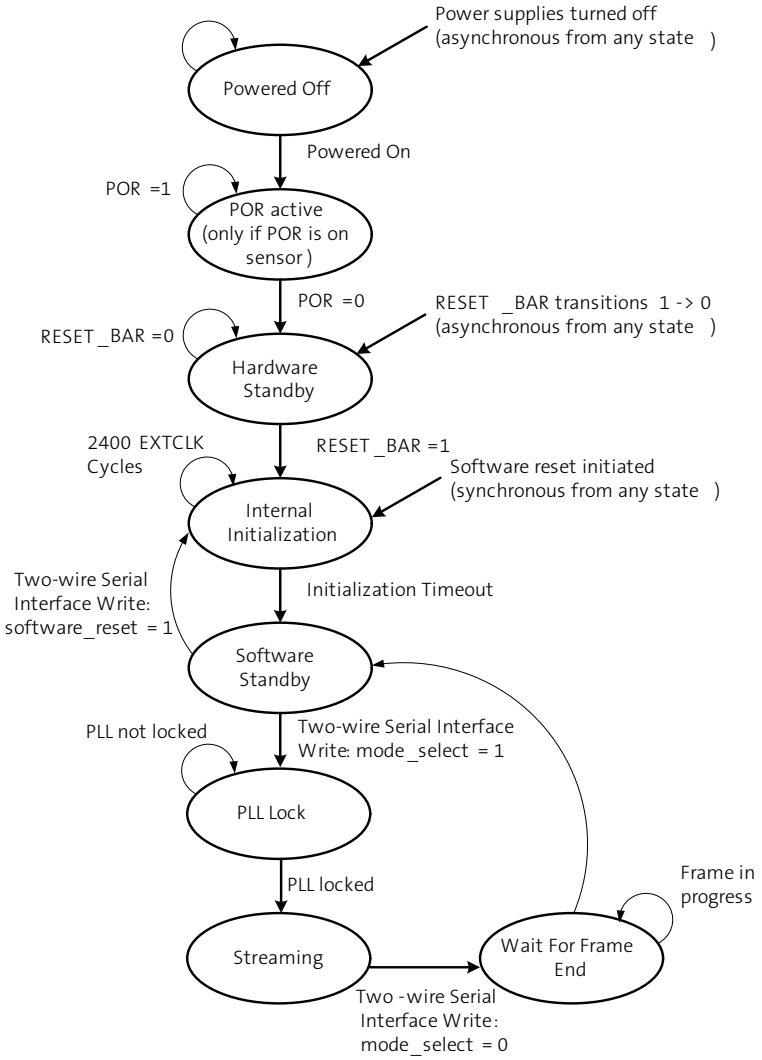


Table 9: RESET_BAR and PLL in System States

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

Notes: 1. VCO = voltage-controlled oscillator.

Power-On Reset Sequence

When power is applied to the MT9P014, it enters a low-power hardware standby state. Exit from this state is controlled by the later of two events:

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

It is possible to hold RESET_BAR permanently 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 MT9P014 is in its lowest-powered, powered-up state; the internal PLL is disabled, the CCP2 serializer is disabled and internal clocks are gated off.

When the sensor leaves the hardware standby state it performs an internal initialization sequence that takes 2400 EXTCLK cycles. After this, it enters a low-power software standby state. While the initialization sequence is in progress, the MT9P014 will not respond to READ transactions on its two-wire serial interface. Therefore, a method to determine when the initialization sequence has completed is to poll a sensor register; for example, R0x0000. While the initialization sequence is in progress, the sensor will not respond to its device address and READs from the sensor will result in a NACK on the two-wire serial interface bus. When the sequence has completed, READs will return the operational value for the register (0x2800 if R0x0000 is read).

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

Soft Reset Sequence

The MT9P014 can be reset under software control by writing “1” to software_reset (R0x0103). A software reset asynchronously resets the sensor, truncating any frame that is in progress. The sensor starts the internal initialization sequence, while the PLL and analog blocks are turned off. At this point, the behavior is exactly the same as for the power-on reset sequence.

Signal State During Reset

Table 10 shows the state of the signal interface during hardware standby (RESET_BAR asserted) and the default state during software standby. After exit from hardware standby and before any registers within the sensor have been changed from their default power-up values the sensor is still in standby mode.

Table 10: 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.	
SCLK	Input	Enabled. Must be pulled up or driven to a valid logic level.	
GPI[3:0]	Input	Powered down. Can be left disconnected/floating.	
TEST	Input	Enabled. Must be driven to a logic 0 for a serial CCP2-configured sensor, or 1 for a serial MIPI-configured sensor.	
SDATA	I/O	Enabled as an input. Must be pulled up or driven to a valid logic level.	
FLASH	Output	High-Z.	Logic 0.
SHUTTER	Output	High-Z.	Logic 0.
DATA0_P	Output	CCP2: High Z MIPI: Ultra Low-Power State (ULPS), represented as an LP-00 state on the output (both wires at 0V)	
DATA0_N	Output		
DATA1_P	Output		
DATA1_N	Output		
CLK_P	Output		
CLK_N	Output		

General Purpose Inputs

The MT9P014 provides four general purpose inputs. After reset, the input pads associated with these signals are powered down by default, allowing the pads to be left disconnected/floating.

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

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

- Trigger (see the sections below)
- Standby functions
- SADDR selection (see “Serial Register Interface” on page 26)

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

Streaming/Standby Control

The MT9P014 can be switched between its soft standby and streaming states under pin or register control, as shown in Table 11. The state diagram for transitions between soft standby and streaming states is shown in Figure 12 on page 28.

Table 11: Streaming/STANDBY

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

Trigger Control

When the global reset feature is in use, the trigger for the sequence can be initiated either under pin or register control, as shown in Table 12.

Table 12: Trigger Control

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

Clocking

The MT9P014 contains a PLL for timing generation and control. The PLL contains a prescaler to divide the input clock applied on EXTCLK, a VCO to multiply the prescaler output, and a set of dividers to generate the output clocks.

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

Figure 13: MT9P014 Profile 1/2 Clocking Structure

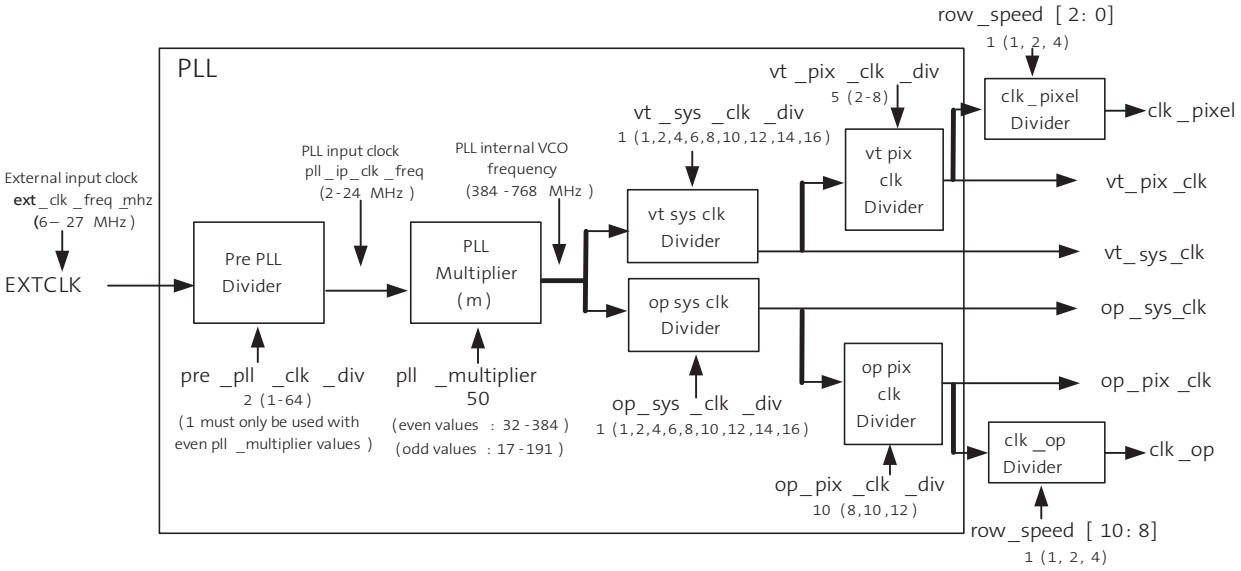


Figure 13 shows the different clocks and the names of the registers that contain or are used to control their values. Also shown is the default setting for each divider/multipler control register and the range of legal values for each divider/multiplier control register.

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

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

- The minimum/maximum frequency limits for the associated clock must be met pll_ip_clk_freq must be in the range 2–24 MHz. Higher frequencies are preferred. PLL internal VCO frequency must be in the range 384–768 MHz.
- The minimum/maximum value for the divider/multiplier must be met. Range for m: 32 –192. (In addition odd values between 17–31 and even values between 194–384 are accepted.) Range for n: 0-63. Range for (n+1): 1–64.
- clk_op must never run faster than the clk_pixel to ensure that the output data stream is contiguous.
- Given the maximum programmed line length, the minimum blanking time, the maximum image width, and the available PLL divisor/multiplier values, the require-

ment for the output line time (including the necessary blanking) is that it must be output in a time equal to or less than the time defined by line_length_pck.

Although the PLL VCO input frequency range is advertised as 2–24 MHz, superior performance is obtained by keeping the VCO input frequency as high as possible.

The usage of the output clocks is shown below:

- clk_pixel (vt_pix_clk / row_speed[2:0]) is used by the sensor core to readout and control the timing of the pixel array. The sensor core produces one 12-bit pixel each vt_pix_clk period. The line length (line_length_pck) and fine integration time (fine_integration_time) are controlled in increments of the vt_pix_clk period.
- clk_op (op_pix_clk / row_speed[10:8]) is used to load parallel pixel data from the output FIFO (see Figure 35 on page 64) to the serializer. The output FIFO generates one pixel each op_pix_clk period. The pixel is either 8-bit, 10-bit, or 12-bit depending upon the output data format, controlled by R0x0112–3 (ccp_data_format).
- op_sys_clk is used to generate the serial data stream on the output. The relationship between this clock frequency and the op_pix_clk frequency is dependent upon the output data format.

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

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

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

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

In Profile 0, RAW10 data format is required. As a result, op_pix_clk_div should be set to 10. Also, due to the inherent design of the MT9P012 sensor, vt_pix_clk_div should be set to 5 for profile 0 mode.

Programming the PLL Divisors

The PLL divisors should be programmed while the MT9P014 is in the software standby state. After programming the divisors, it is necessary to wait for the VCO lock time before enabling the PLL. The PLL is enabled by entering the streaming state.

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

The effect of programming the PLL divisors while the MT9P014 is in the streaming state is undefined.

Influence of ccp_data_format

R0x0112–3 (ccp_data_format) controls whether the pixel data interface will generate 12, 10, or 8 bits per pixel.

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

Clock Control

The MT9P014 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 MT9P014 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

Lens Shading Correction (LC)

Lenses tend to produce images whose brightness is significantly attenuated near the edges. There are also other factors causing fixed pattern signal gradients in images captured by image sensors. The cumulative result of all these factors is known as image shading. The MT9P014 has an embedded shading correction module that can be programmed to counter the shading effects on each individual Red, GreenB, GreenR, and Blue color signal.

The Correction Function

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

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

(EQ 4)

$$Pcorrected(row, col) = Psensor(row,col) * f(row,col)$$

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

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

Lens Shading Coefficient Read/Write Procedure

The MT9P014 LSC coefficient registers read back incorrectly only if the LSC is enabled. It also does not write correctly if the LSC is enabled.

Before reading or writing the LSC coefficients, disable the LSC by setting the POLY_SC_ENABLE bitfield to “0” as shown below:

BITFIELD = 0x3780, 0x8000, 0 //POLY_SC_ENABLE

If needed, enable the LSC by setting the POLY_SC_ENABLE bitfield to “1” as shown below:

BITFIELD = 0x3780, 0x8000, 1 //POLY_SC_ENABLE

One-Time Programmable (OTP) Memory

The MT9P014 has a two-byte OTP memory that can be used during module manufacturing to store specific information about the module. This feature allows system integrators and module manufacturers to label and distinguish various module types based on lens, IR-cut filter, or other properties.

During the programming process, a dedicated pin for high voltage needs to be provided to perform the anti-fusing operation. This voltage (VPP) would need to be 8.5V ±%3. Instantaneous VPP cannot exceed 9V at any time. The completion of the programming process will be communicated by a register through the two-wire serial interface.

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

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

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

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

1. Apply power to all the power rails of the sensor (VDD, VDD_IO, VAA, VAA_PIX, and VDD_PLL)
   1a. Set VAA to 3.1V during OTP memory programming phase.
   1b. VPP needs to be floated during this phase.
   1c. Other supplies at nominal.
2. Provide 24 MHz EXTCLK clock input. The PLL settings are discussed at the end of the document.
3. Perform the proper reset sequence to the sensor (see “Power-On Reset Sequence” on page 29).
4. Place the sensor in soft standby (sensor default state upon power-up) or ensure the streaming is turned OFF when the part is in active mode.
5. VPP ramps to 8.5V in preparation to program. Power supply (VPP) slew rate should be slower than 1V/µs.
6. Program R0x3052 to the value 0x045C.
7. Program R0x3054 to the value 0XEA99.
8. Write the 16-bit word data by programming R0x304C.
9. Initiate the OTP memory programming process by programming R0x304A[0] to the value 0x0001.
10. Check R0x304A [2] = 1, until the bit is set to “1” to check for program completion.
11. Repeat steps 9 and 10 two more times.
12. Remove high voltage and float VPP pin.
13. Power down the sensor.

14. Apply nominal power to all the power rails of the sensor VDD, VDD_IO, VAA, VAA_PIX and VDD_PLL). VPP must be floated.
15. Set EXTCLK to normal or customer defined operating frequency.
16. Perform the proper reset sequence to the sensor.
17. Initiate the OTP memory reading process by setting R0x304A[4] to the value 0x0010.
18. Poll the register bit R0x304A[6] until bit set to "1" to check for read completion.
19. Read the 16-bit word data from the R0x304E.

OTP Memory Write—recommended PLL settings:
R0x301A = 0x10C8
R0x0300 = 0x0008
R0x0302 = 0x0006
R0x0304 = 0x0002
R0x0306 = 0x0020
R0x0308 = 0x000A
R0x030A = 0x0006
R0x301A = 0x10CC

The above frequency settings yield an output frequency of op_pix_clk = 6.4 MHz with EXTCLK = 24 MHz.

Figure 14: Sequence Programming for the MT9P014

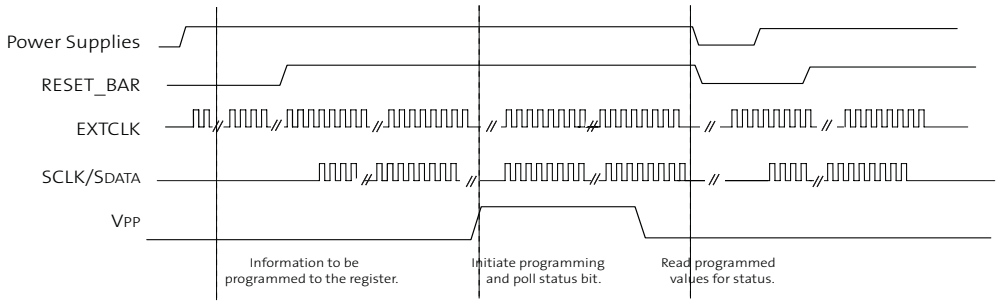


Image Acquisition Modes

The MT9P014 supports two image acquisition modes:

Electronic rolling shutter (ERS) mode. This is the normal mode of operation. When the MT9P014 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 MT9P014 switches cleanly from the old integration time to the new while only generating frames with uniform integration. (See "Changes to Integration Time" on page 21.)

Global reset mode. This mode can be used to acquire a single image at the current resolution. In this mode, the end point of the pixel integration time is controlled by an external electromechanical shutter, and the MT9P014 provides control signals to interface to that shutter. The operation of this mode is described in detail in "Global Reset" on page 50.

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

Window Control

The sequencing of the pixel array is controlled by the x_addr_start, y_addr_start, x_addr_end, and y_addr_end registers. 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 2592H x 1944V image. A border of up to 8 pixels (4 in binning) on each edge can be enabled by reprogramming the x_addr_start, y_addr_start, x_addr_end, y_addr_end, x_output_size, and y_output_size registers accordingly.

Readout Modes

Horizontal Mirror

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

Vertical Flip

When the vertical_flip bit is set in the image_orientation register, the order in which pixel rows are read out is reversed, so that row readout starts from y_addr_end and ends at y_addr_start. 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.

Subsampling

The MT9P014 supports subsampling. Subsampling reduces the amount of data processed by the analog signal chain in the MT9P014 thereby allowing the frame rate to be increased. Subsampling is enabled by setting x_odd_inc and/or y_odd_inc. Values of 1, 3, and 7 can be supported. Setting both of these variables to 3 reduces the amount of row and column data processed and is equivalent to the 2 x 2 skipping readout mode provided by the MT9P014.

A 1/16 reduction in resolution is achieved by setting both x_odd_inc and y_odd_inc to 7. This is equivalent to 4 x 4 skipping readout mode provided by the MT9P014.

The effect of the different subsampling settings on the pixel array readout is shown in Figure 15 through Figure 17 on page 40.

Figure 15: Pixel Readout (No Subsampling)

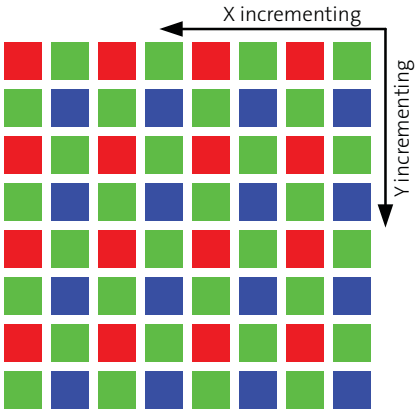


Figure 16: Pixel Readout (x_odd_inc = 3, y_odd_inc = 3)

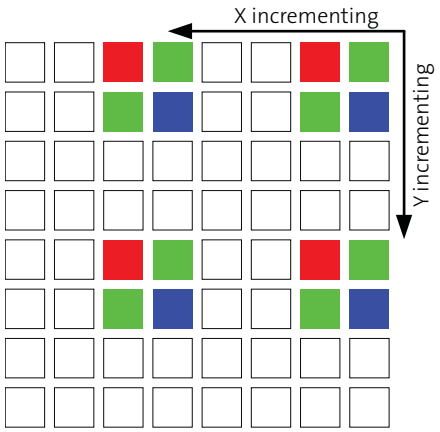
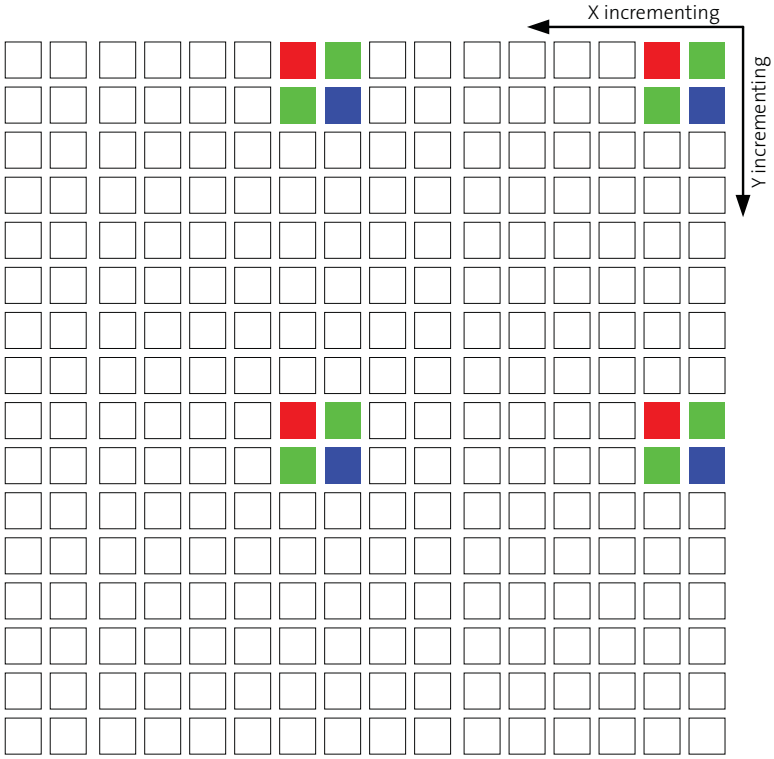


Figure 17: Pixel Readout (x_odd_inc = 7, y_odd_inc = 7)



Programming Restrictions when Subsampling

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

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

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

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

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

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

Example:

The sensor is set up to give out a full resolution 2592 x 1944 image:

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

```
REG=0x0104, 1          //GROUPED_PARAMETER_HOLD
REG=0x0382, 1          //X_ODD_INC
REG=0x0386, 1          //Y_ODD_INC
REG=0x0344, 8          //X_ADDR_START
REG=0x0346, 8          //Y_ADDR_START
REG=0x0348, 2599       //X_ADDR_END
REG=0x034A, 1951       //Y_ADDR_END
REG=0x034C, 2592       //X_OUTPUT_SIZE
REG=0x034E, 1944       //Y_OUTPUT_SIZE
REG=0x0104, 0          //GROUPED_PARAMETER_HOLD
```

To halve the resolution in each direction (1296 x 972), the registers need to be reprogrammed as follows:

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

```
REG=0x0104, 1          //GROUPED_PARAMETER_HOLD
REG=0x0382, 3          //X_ODD_INC
REG=0x0386, 3          //Y_ODD_INC
REG=0x0344, 8          //X_ADDR_START
REG=0x0346, 8          //Y_ADDR_START
REG=0x0348, 2597       //X_ADDR_END
REG=0x034A, 1949       //Y_ADDR_END
REG=0x034C, 1296       //X_OUTPUT_SIZE
REG=0x034E, 972        //Y_OUTPUT_SIZE
REG=0x0104, 0          //GROUPED_PARAMETER_HOLD
```

To quarter the resolution in each direction (648 x 486) the registers need to be reprogrammed as follows:

[4 x 4 skipping starting address with (8, 8)]

```
REG=0x0104, 1          //GROUPED_PARAMETER_HOLD
REG=0x0382, 7          //X_ODD_INC
REG=0x0386, 7          //Y_ODD_INC
REG=0x0344, 8          //X_ADDR_START
REG=0x0346, 8          //Y_ADDR_START
REG=0x0348, 2593       //X_ADDR_END
REG=0x034A, 1945       //Y_ADDR_END
REG=0x034C, 648        //X_OUTPUT_SIZE
REG=0x034E, 486        //Y_OUTPUT_SIZE
REG=0x0104, 0          //GROUPED_PARAMETER_HOLD
```

Table 13 shows the row or column address sequencing for normal and subsampled readout. In the 2X skip case, there are two possible subsampling sequences (because the subsampling sequence only read half of the pixels) depending upon the alignment of the start address. Similarly, there will be four possible subsampling sequences in the 4X skip case (though only the first two are shown in Table 13).

Table 13: Row Address Sequencing During Subsampling

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

Binning

The MT9P014 supports 2 x 1 and 2 x 2 analog binning (column binning, also called x-binning and row/column binning, also called xy-binning). Binning has many of the same characteristics as subsampling, but because it gathers image data from all pixels in the active window (rather than a subset of them), it achieves superior image quality and avoids the aliasing artifacts that can be a characteristic side effect of subsampling.

Binning is enabled by selecting the appropriate subsampling settings (odd_inc = 3 and y_odd_inc = 1 for x-binning, x_odd_inc = 3 and y_odd_inc = 3 for xy-binning) and setting the appropriate binning bit in read_mode (R0x3040–1). As with subsampling, x_addr_end and y_addr_end may require adjustment when binning is enabled. It is the first of the two columns/rows binned together that should be the end column/row in binning, so the requirements to the end address are exactly the same as in non-binning subsampling mode. The effect of the different subsampling settings is shown in Figure 18 and in Figure 19 on page 44.

Binning can also be enabled when the 4X subsampling mode is enabled (x_odd_inc = 7 and y_odd_inc = 1 for x-binning, x_odd_inc = 7 and y_odd_inc = 7 for xy-binning). In this mode, however, not all pixels will be used so this is not a 4X binning implementation. An implementation providing a combination of skip2 and bin2 is used to achieve 4X subsampling with better image quality. The effect of this subsampling mode is shown in Figure 20 on page 44.

Figure 18: Pixel Readout (x_odd_inc = 3, y_odd_inc = 1, x_bin = 1)

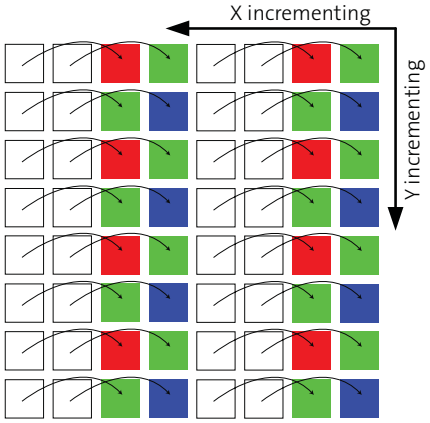


Figure 19: Pixel Readout (x_odd_inc = 3, y_odd_inc = 3, xy_bin = 1)

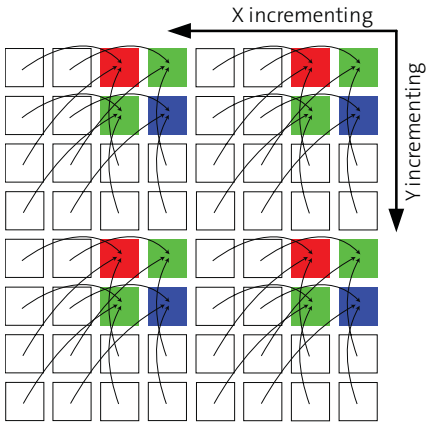
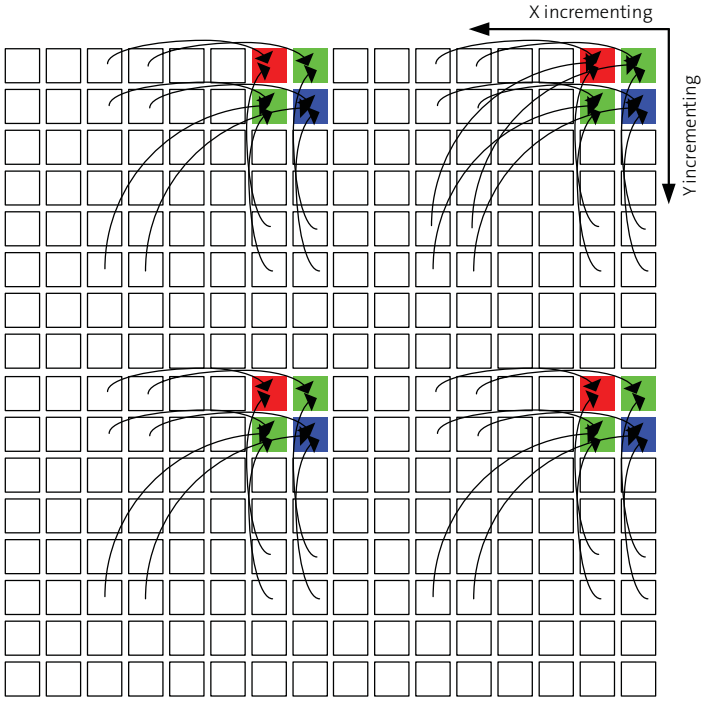


Figure 20: Pixel Readout (x_odd_inc = 7, y_odd_inc = 7, xy_bin = 1)



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

For a given column *n*, there is only one other column, n_bin, that it can be binned with, because of physical limitations in the column readout circuitry. The possible address sequences are shown in Table 14 on page 44.

Table 14: Column Address Sequencing During Binning

odd_inc = 1—Normal	odd_inc = 3, 2X Bin	odd_inc = 7, 2X Skip + 2X Bin
x_addr_start = 0	x_addr_start = 0	x_addr_start = 0
0	0/2	0/2

Table 14: Column Address Sequencing During Binning (continued)

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

There are no physical limitations on what can be binned together in the row direction. A given row *n* will always be binned with row n+2 in 2X subsampling mode and with row n+4 in 4X subsampling mode. Therefore, which rows get binned together depends upon the alignment of y_addr_start. The possible sequences are shown in Table 15.

Table 15: Row Address Sequencing During Binning

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

Programming Restrictions when Binning

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

As a result, when xy-binning is enabled, some of the programming limits declared in the parameter limit registers are no longer valid. In addition, the default values for some of the manufacturer-specific registers need to be reprogrammed. See sections "Minimum Frame Time" on page 47, "Minimum Row Time" on page 47, and "Fine Integration Time Limits" on page 48.

Table 16: Readout Modes

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

Frame Rate Control

The formulas for calculating the frame rate of the MT9P014 are shown below.

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

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

Note that line_length_pck also needs to meet the minimum line length requirement set in register min_line_length_pck. The row time can either be limited by the time it takes to sample and reset the pixel array for each row, or by the time it takes to sample and read out a row. Values for min_line_blanking_pck are provided in “Minimum Row Time” on page 47.

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

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

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

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

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

Minimum Row Time

The minimum row time and blanking values with default register settings are shown in Table 17.

Table 17: Minimum Row Time and Blanking Numbers

Register	No Row Binning			Row Binning		
row_speed[2:0]	1	2	4	1	2	4
min_line_blanking_pck	0x0478	0x02A2	0x01B7	0x0824	0x0468	0x028A
min_line_length_pck	0x0630	0x03E8	0x02FB	0x0C40	0x0620	0x03D0

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

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

1. line_length_pck ≥ min_line_length_pck in Table 17.
2. line_length_pck ≥ (x_addr_end – x_addr_start + x_odd_inc)/((1+x_odd_inc)/2) + min_line_blanking_pck in Table 17.
3. The row time must allow the FIFO to output all data during each row line_length_pck ≥ (x_output_size * 2 + 0x005E) * “vt_pix_clk period” / “op_pix_clk period”

Minimum Frame Time

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

Table 18: Minimum Frame Time and Blanking Numbers

Register	No Row Binning	Row Binning
min_frame_blanking_lines	0x004F	0x004F
min_frame_length_lines	0x0057	0x0055

Integration Time

The integration (exposure) time of the MT9P014 is controlled by the fine_integration_time and coarse_integration_time registers.

The limits for the fine integration time are defined by:

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

The limits for the coarse integration time are defined by:

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

The actual integration time is given by:

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

It is required that:

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

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

Fine Integration Time Limits

The limits for the fine_integration_time can be found from fine_integration_time_min and fine_integration_time_max_margin. Values for different mode combinations are shown in Table 19.

Table 19: fine_integration_time Limits

Register	No Row Binning			Row Binning		
row_speed[2:0]	1	2	4	1	2	4
fine_integration_time_min	0x03C2	0x01F2	0x008E	0x078A	0x03D6	0x0106
fine_integration_time_max_margin	0x026E	0x0126	0x00FE	0x04B6	0x024A	0x020A

fine_correction

For the fine_integration_time limits, the fine_correction constant will change with the pixel clock speed and binning mode. These values are shown in Table 20.

Table 20: fine_correction Values

Register	No Row Binning			Row Binning		
row_speed[2:0]	1	2	4	1	2	4
fine_correction	0x00A8	0x004E	0x0021	0x0168	0x00AE	0x0051

Low-Power Mode

The MT9P014 supports a low-power mode, which can be entered by programming register bit read_mode[9]. Setting this bit will:

- Double the value of pc_speed[2:0]. This means halving the vt_pix_clk frequency.
- Lower currents in the analog domain.

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

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

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

Flash Control

The MT9P014 supports both xenon and LED flash through the FLASH output signal. The timing of the FLASH signal with the default settings is shown in Figure 21, and in Figures 22 and Figure 23 on page 50. The flash and flash_count registers allow the timing of the flash to be changed. The flash can be programmed to fire only once, delayed by a few frames when asserted, and (for xenon flash) the flash duration can be programmed.

Enabling the LED flash will cause one bad frame, where several of the rows only have the flash on for part of their integration time. This can be avoided either by first enabling mask bad frames (write reset_register[9] = 1) before the enabling the flash or by forcing a restart (write reset_register[1] = 1) immediately after enabling the flash; the first bad frame will then be masked out, as shown in Figure 23. Read-only bit flash[14] is set during frames that are correctly integrated; the state of this bit is shown in Figure 21, and in Figures 22 and Figure 23 on page 50.

Figure 21: Xenon Flash Enabled

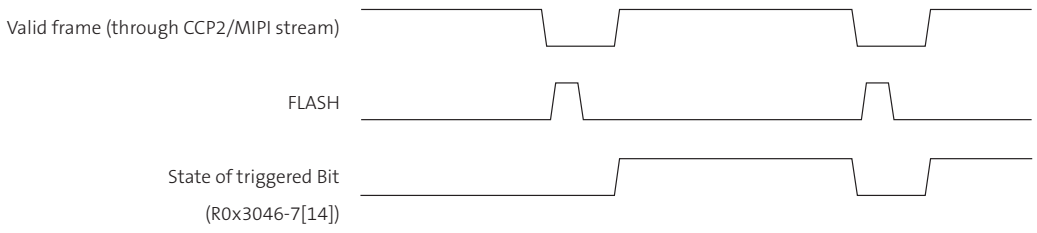
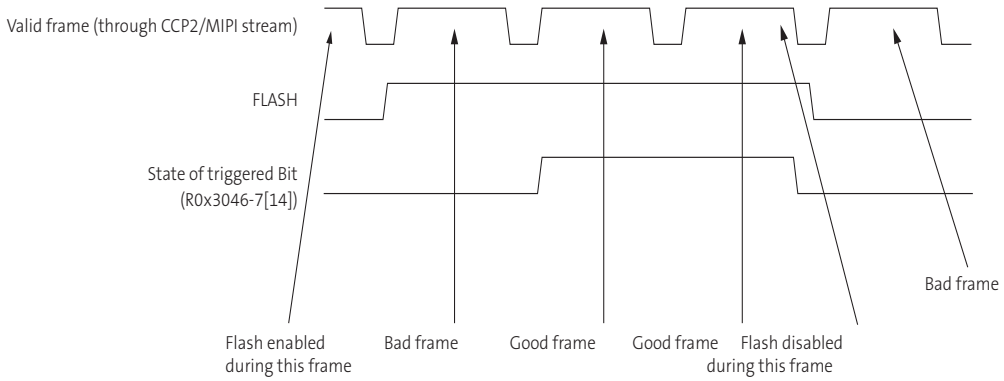
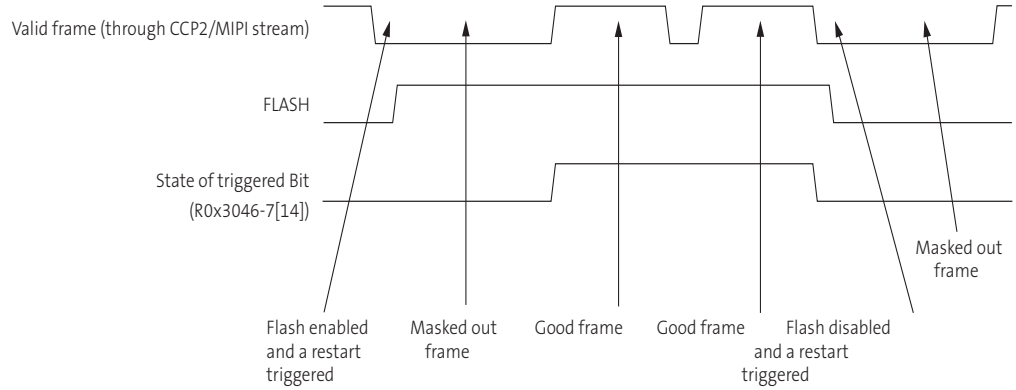


Figure 22: LED Flash Enabled



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

Figure 23: LED Flash Enabled Following Forced Restart



Global Reset

Global reset mode allows the integration time of the MT9P014 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.

The SMIA specification does not define a global reset mode and only provides for operation in ERS mode. The MT9P014 does support the use of global reset mode in conjunction with the SMIA data path, but there are additional restrictions on its use.

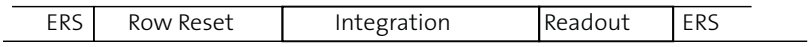
Overview of Global Reset Sequence

The basic elements of the global reset sequence are:

1. By default, the sensor operates in ERS mode and the SHUTTER output signal is LOW. The electromechanical shutter must be open to allow light to fall on the pixel array. Integration time is controlled by the coarse_integration_time and fine_integration_time registers.
2. A global reset sequence is triggered.
3. All of the rows of the pixel array are placed in reset.
4. All of the rows of the pixel array are taken out of reset simultaneously. All rows start to integrate incident light. The electromechanical shutter may be open or closed at this time.
5. If the electromechanical shutter has been closed, it is opened.
6. After the desired integration time (controlled internally or externally to the MT9P014), the electromechanical shutter is closed.
7. A single output frame is generated by the sensor. As soon as the output frame has completed, the electromechanical shutter may be opened again.
8. The sensor automatically resumes operation in ERS mode.

This sequence is shown in Figure 24. The following sections expand to show how the timing of this sequence is controlled.

Figure 24: Overview of Global Reset Sequence



Entering and Leaving the Global Reset Sequence

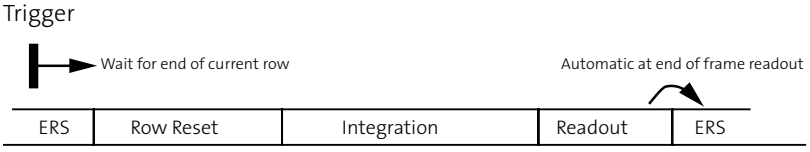
A global reset sequence can be triggered by a register write to global_seq_trigger[0] (global trigger, to transition this bit from a 0 to a 1) or by a rising edge on a suitably-configured GPI input (see “Trigger Control” on page 31).

When a global reset sequence is triggered, the sensor waits for the end of the current row. When internal line valid negates for that row, the frame becomes invalid 6 system_clock periods later, potentially truncating the frame that was in progress.

The global reset sequence completes with a frame readout. At the end of this readout phase, the sensor automatically resumes operation in ERS mode. The first frame integrated with ERS will be generated after a delay of approximately ((13 + coarse_integration_time) * line_length_pck). This sequence is shown in Figure 25 on page 51.

While operating in ERS mode, double-buffered registers (“Double-Buffered Registers” on page 20) 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 25: Entering and Leaving a Global Reset Sequence



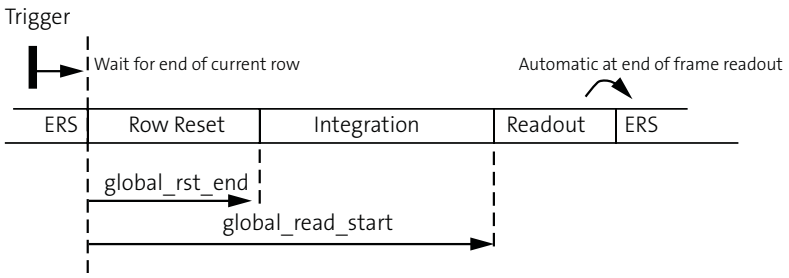
Programmable Settings

The registers global_rst_end and global_read_start allow the duration of the row reset phase and the integration phase to be controlled, as shown in Figure 26. The duration of the readout phase is determined by the active image size.

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

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

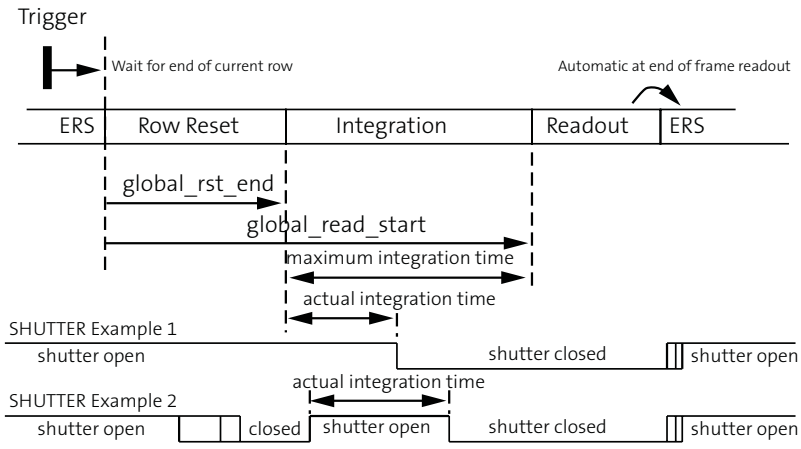
Figure 26: Controlling the Reset and Integration Phases of the Global Reset Sequence



Control of the Electromechanical Shutter

Figure 27 on page 53 shows two different ways in which a shutter can be controlled during the global reset sequence. In both cases, the maximum integration time is set by the difference between global_read_start and global_rst_end. In shutter example 1, the shutter is open during the initial ERS sequence and during the row reset phase. The shutter closes during the integration phase. The pixel array is integrating incident light from the start of the integration phase to the point at which the shutter closes. Finally, the shutter opens again after the end of the readout phase. In shutter example 2, the shutter is open during the initial ERS sequence and closes sometime during the row reset phase. The shutter both opens and closes during the integration phase. The pixel array is integrating incident light for the part of the integration phase during which the shutter is open. As for the previous example, the shutter opens again after the end of the readout phase.

Figure 27: Control of the Electromechanical Shutter



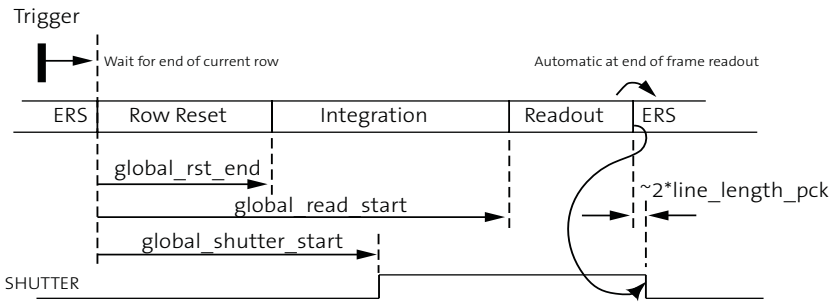
It is essential that the shutter remains closed during the entire row readout phase; otherwise, some rows of data will be corrupted (over-integrated).

After frame readout is completed, there is a time delay of approximately *10 x 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 MT9P014 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 28. 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 x line_length_pck* after frame readout has completed.

This programming restriction must be met for correct operation:
global_read_start > global_shutter_start

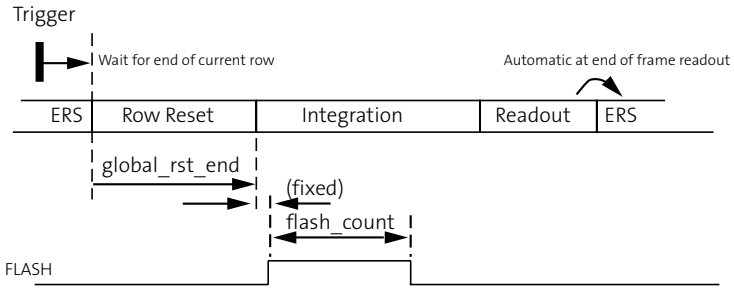
Figure 28: Controlling the SHUTTER Output



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 29.

Figure 29: 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.

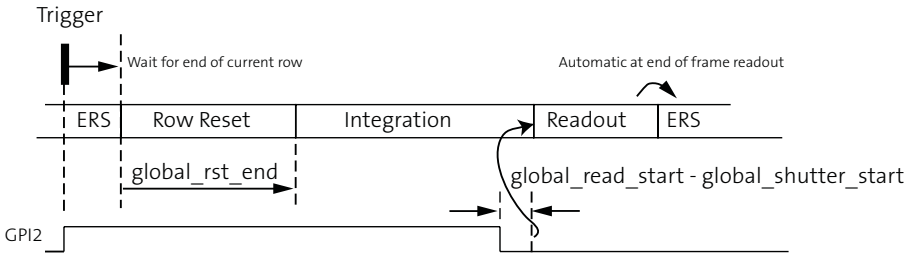
When the trigger is negated to end integration, the integration phase is extended by a further time given by *global_read_start – global_shutter_start*. Usually this means that global_read_start should be set to *global_shutter_start + 1*.

The operation of this mode is shown in Figure 30. 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.

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

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

Figure 30: Global Reset Bulb



Retriggering the Global Reset Sequence

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

The earliest time that the global reset sequence can be retriggered is the point at which the SHUTTER output negates.

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 35 on page 64) attempts to extend (pad) all frames to the programmed value of y_output_size. If this padding is still in progress when the global reset readout phase starts, the SMIA data path will not detect the start of the frame correctly. Therefore, to use global reset with the serial data path, this timing scenario must be avoided. One possible way of doing this would be to synchronize (under software control) the assertion of trigger to an end-of-frame marker on the CCP serial data stream.

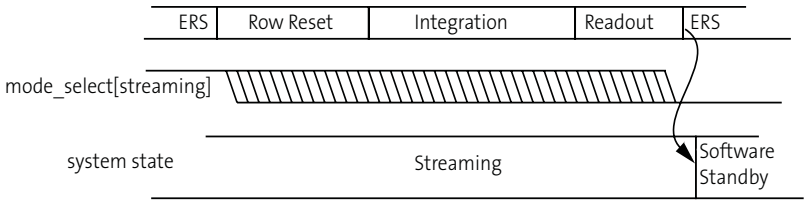
At the end of the readout phase of the global reset sequence, the sensor automatically resumes operation in ERS mode.

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

Global Reset and Soft Standby

If the mode_select[stream] bit is cleared while a global reset sequence is in progress, the MT9P014 will remain in streaming state until the global reset sequence (including frame readout) has completed, as shown in Figure 31.

Figure 31: Entering Soft Standby During a Global Reset Sequence



Analog Gain

The MT9P014 provides two mechanisms for setting the analog gain. The first uses the SMIA gain model; the second uses the traditional Aptina gain model.The following sections describe both models, the mapping between the models, and the operation of the per-color and global gain control.

Using Per-color or Global Gain Control

The read-only analogue_gain_capability register returns a value of "1," indicating that the MT9P014 provides per-color gain control. However, the MT9P014 also provides the option of global gain control. Per-color and global gain control can be used interchangeably. A write to a global gain register is aliased as a write of the same data to the four associated color-dependent gain registers. A read from a global gain register is aliased to a read of the associated greenB/greenR gain register.

The read/write gain_mode register required by SMIA has no defined function. In the MT9P014, this register has no side-effects on the operation of the gain; per-color and global gain control can be used interchangeably regardless of the state of the gain_mode register.

SMIA Gain Model

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

- analogue_gain_code_global
- analogue_gain_code_greenR
- analogue_gain_code_red
- analogue_gain_code_blue
- analogue_gain_code_greenB

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

$$gain = \frac{analogue\_gain\_m0 \times analogue\_gain\_code}{analogue\_gain\_c1} = \frac{analogue\_gain\_code\_<color>}{8} \quad \text{(EQ 12)}$$

Aptina Gain Model

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

- global_gain
- green1_gain
- red_gain
- blue_gain
- green2_gain

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

$$gain = (<color>\_gain[8] + 1) \times (<color>\_gain[7] + 1) \times \frac{<color>\_gain[6:0]}{32} \quad \text{(EQ 13)}$$

As a result of the 2X gain stages, many of the possible gain settings can be achieved in two different ways. For example, red_gain = 0x02A0 provides the same gain as red_gain = 0x0240 and red_gain = 0x0320. The first example uses the first 2X gain stage, the second example uses no 2X gain stage and the third example uses the second 2X gain stage. In all cases, the preferred setting is the setting that enables the first 2X gain stage and not the last 2X gain stage, because this will result in lower noise. The recommended sequence is shown in Table 21.

Table 21: Recommended Gain Settings

Desired Gain	Recommended Gain Register Setting
1–1.96875	0x0220–0x023F
2–7.9375	0x02A0–0x02FF
8–15.875	0x03C0–0x03FF

Note: Use analog gain ≤ 8 for improved image quality.

Gain Code Mapping

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

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

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

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

Sensor Core Digital Data Path

Test Patterns

The MT9P014 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 22.

Table 22: Test Patterns

test_pattern_mode	Description
0	Normal operation: no test pattern
1	Solid color
2	100% color bars
3	Fade-to-gray color bars
4	PN9 link integrity pattern (only on sensors with serial interface)
256	Walking 1s
257	Walking 1s (8-bit for 10-bit sensors, 10-bit for 12-bit sensors)
258	Walking 1s (8-bit for 12-bit sensors)

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

For all of the test patterns, the MT9P014 registers must be set appropriately to control the frame rate and output timing. This includes:

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

Effect of Data Path Processing on Test Patterns

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

- Noise cancellation
- Black pedestal adjustment
- Defect pixel correction/tagging
- Lens and color shading correction

When enabling the test patterns, all these and other related functions are automatically disabled, to prevent the test pattern from being affected.

If the test pattern is used specifically to study algorithms internal to the chip, this automatic disabling can be turned off by setting R0x307A–B[1] = 1.

Solid Color Test Pattern

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

100% Color Bars Test Pattern

In this test pattern, shown in Figure 32, all pixel data is replaced by a Bayer version of an 8-color, color-bar chart (white, yellow, cyan, green, magenta, red, blue, black). Each bar is 1/8 of the width of the pixel array (2592/8 = 324 pixels). The pattern repeats after 8 * 324 = 2592 pixels.

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

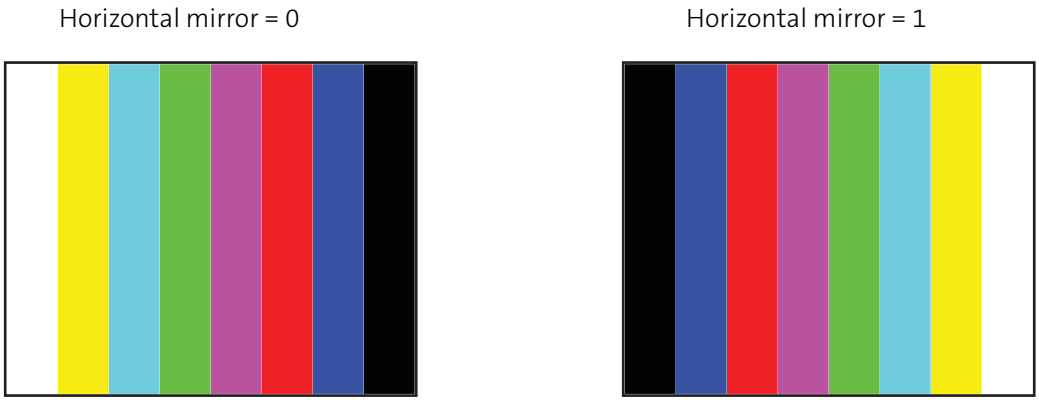
The pattern occupies the full height of the output image.

The image size is set by x_addr_start, x_addr_end, y_addr_start, y_addr_end and may be affected by the setting of x_output_size, y_output_size. The color-bar pattern is disconnected from the addressing of the pixel array, and will therefore always start on the first visible pixel, regardless of the value of x_addr_start. The number of colors that are visible in the output is dependent upon x_addr_end – x_addr_start and the setting of x_output_size: the width of each color bar is fixed at 324 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 32: 100 Percent Color Bars Test Pattern



Fade-to-gray Color Bars Test Pattern

In this test pattern, shown in Figure 32 on page 59, all pixel data is replaced by a Bayer version of an 8-color, color-bar chart (white, yellow, cyan, green, magenta, red, blue, black). Each bar is 1/8 of the width of the pixel array (2592/8 = 324 pixels). The test pattern repeats after 2592 pixels.

Each color bar fades vertically from zero or full intensity at the top of the image to 50 percent intensity (mid-gray) on the last (968th) row of the pattern. Each color bar is divided into a left and a right half, in which the left half fades smoothly and the right half fades in quantized steps.

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

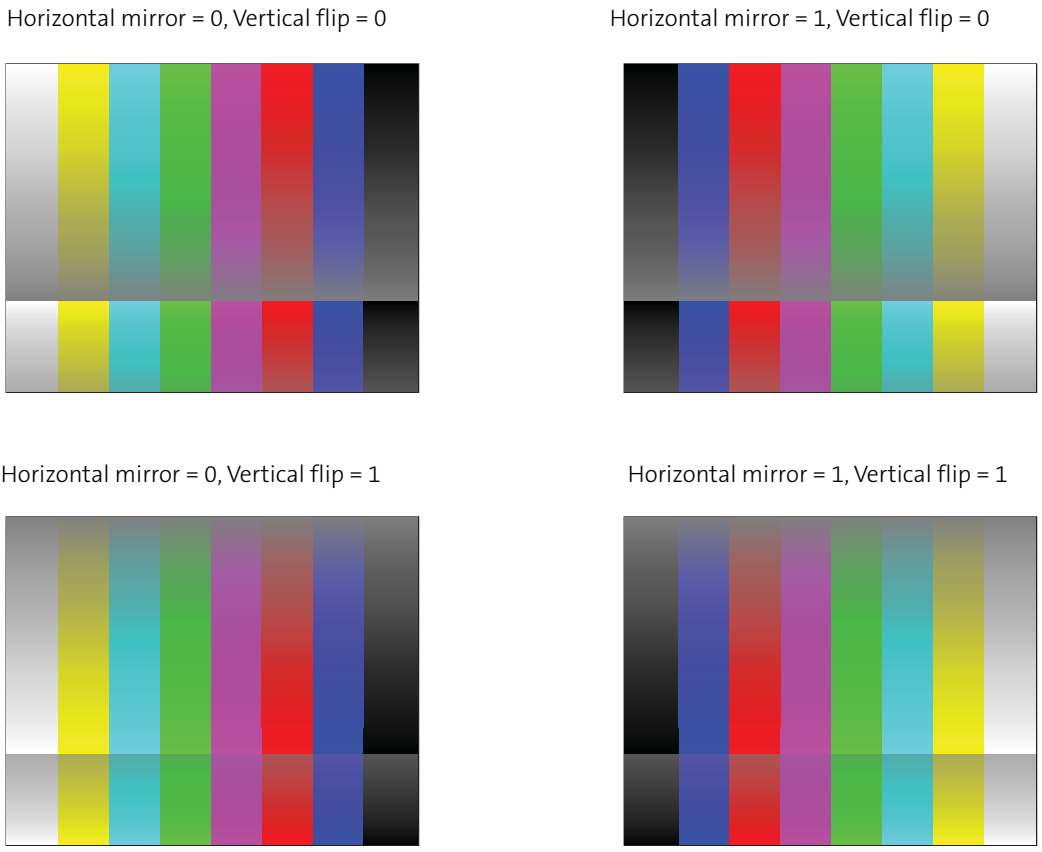
The rate-of-fade of the Bayer pattern is set so that there is at least one full pattern within a full-sized image for the sensor. Factors that affect this are the resolution of the ADC (10-bit or 12-bit) and the image height. For example, the MT9P014 fades the pixels by 2 LSBs for each two rows. With 12-bit data, the pattern is 2048 pixels high and repeats after that, if the window is higher.

The image size is set by x_addr_start, x_addr_end, y_addr_start, y_addr_end and may be affected by the setting of x_output_size, y_output_size. The color-bar pattern starts at the first column in the image, regardless of the value of x_addr_start. The number of colors that are visible in the output is dependent upon x_addr_end – x_addr_start and the setting of x_output_size: the width of each color bar is fixed at 324 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 33: Fade-to-Gray Color Bars Test Pattern



PN9 Link Integrity Pattern

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

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

When this test pattern is enabled:

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

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

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

Walking 1s

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

The false synchronization code removal imposed by CCP2 serial interfaces mean that this test pattern cannot be carried across a CCP2 link in an unmodified form.

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

Test Cursors

The MT9P014 supports one horizontal and one vertical cursor, allowing a crosshair to be superimposed on the image or on test patterns 1–3. The position and width of each cursor are programmable in R0x31E8–R0x31EE. Both even and odd cursor positions and widths are supported.

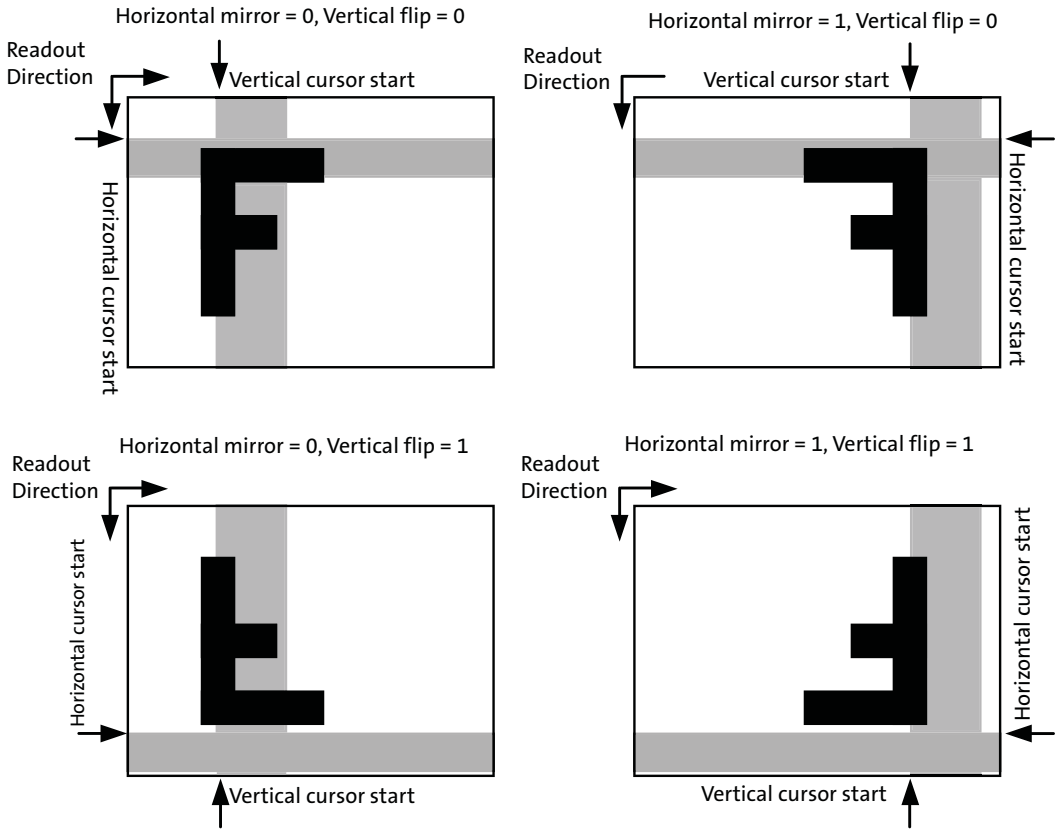
Each cursor can be inhibited by setting its width to "0." The programmed cursor position corresponds to the x and y addresses of the pixel array. For example, setting horizontal_cursor_position to the same value as y_addr_start would result in a horizontal cursor being drawn starting on the first row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

When vertical_cursor_position = 0x0FFF, the vertical cursor operates in an automatic mode in which its position advances every frame. In this mode the cursor starts at the column associated with x_addr_start = 0 and advances by a step-size of 8 columns each frame, until it reaches the column associated with x_addr_start = 2040, after which it wraps (256 steps). The width and color of the cursor in this automatic mode are controlled in the usual way.

The effect of enabling the test cursors when the image_orientation register is non-zero is not defined by the design specification. The behavior of the MT9P014 is shown in Figure 34 on page 63 and the test cursors are shown as translucent, for clarity. In practice, they are opaque (they overlay the imaged scene). The manner in which the test cursors are affected by the value of image_orientation can be understood from these implementation details:

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

Figure 34: Test Cursor Behavior with image_orientation



Digital Gain

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

Pedestal

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

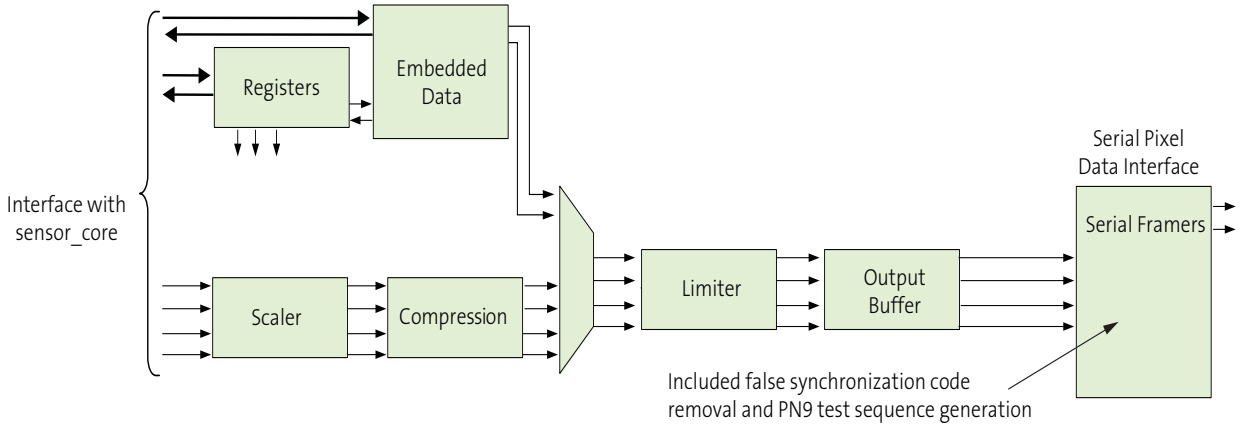
The data_pedestal register is read-only by default but can be made read/write by clearing the lock_reg bit in R0x301A–B.

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

Digital Data Path

The digital data path after the sensor core is shown in Figure 35.

Figure 35: Data Path



Embedded Data Format and Control

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

Timing Specifications

Power-Up Sequence

The recommended power-up sequence for the MT9P014 is shown in Figure 36. The available power supplies—VDD_IO, VDD, VDD_TX0, VDD_PLL, VAA, VAA_PIX—can be turned on at the same time or have the separation specified below.

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

Figure 36: Power-Up Sequence

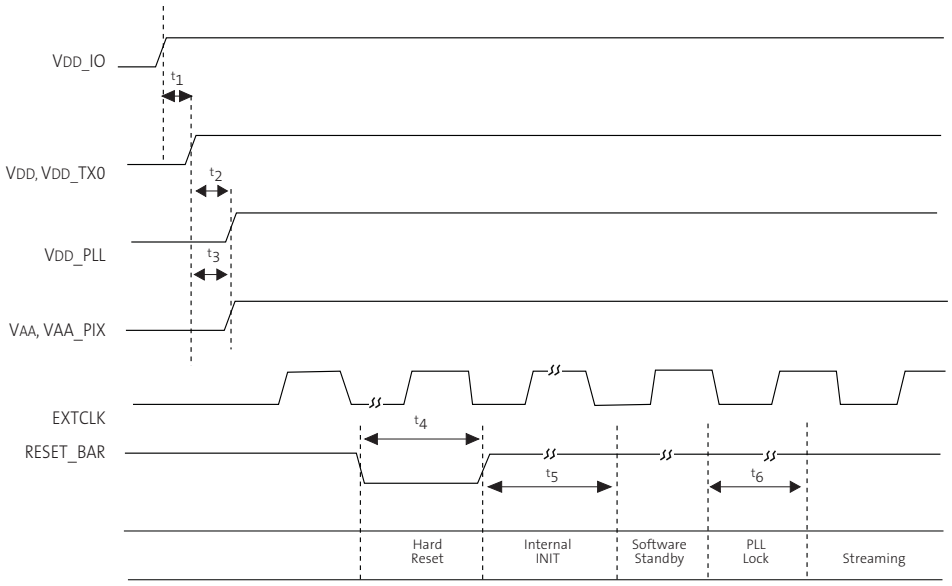


Table 23: Power-Up Sequence

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

Power-Down Sequence

The recommended power-down sequence for the MT9P014 is shown in Figure 37. The available power supplies—VDD_IO, VDD, VDD_TX0, VDD_PLL, VAA, VAA_PIX—can be turned off at the same time or have the separation specified below.

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

Figure 37: Power-Down Sequence

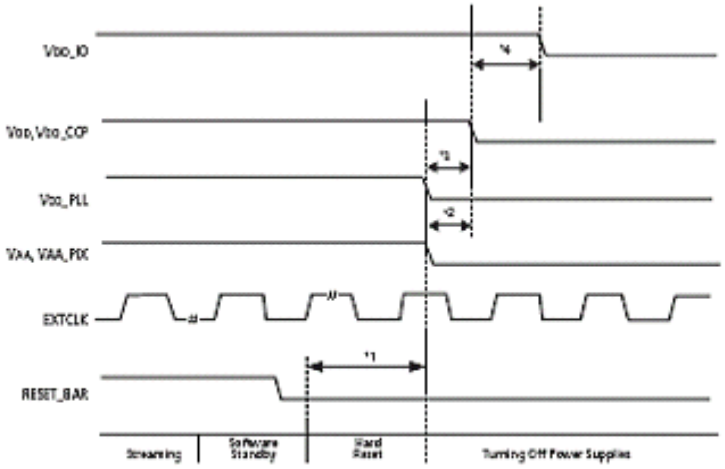


Table 24: Power-Down Sequence

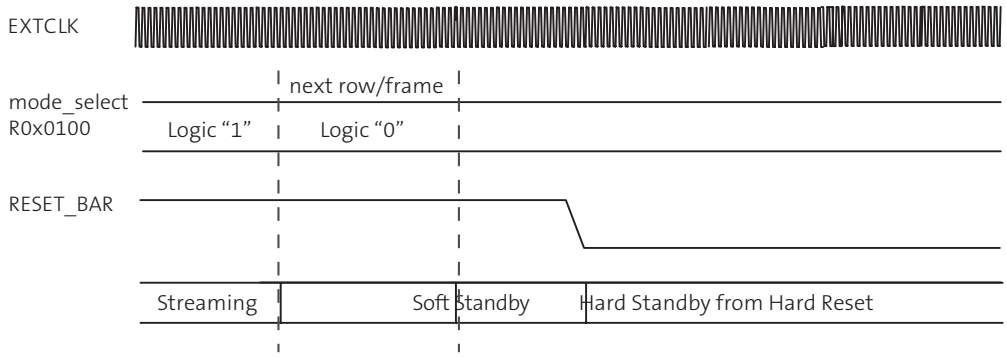
Definition	Symbol	Min	Typ	Max	Unit
Hard reset	t_1	1	–	–	ms
VDD/VAA/VAA_PIX to VDD time	t_2	0	–	–	ms
VDD_PLL to VDD time	t_3	0	–	–	ms
VDD to VDD_IO time	t_4	0	–	–	ms

Hard Standby and Hard Reset

The hard standby state is reached by the assertion of the RESET_BAR signal (hard reset). Register values are not retained by this action, and will be returned to their default values once hard reset is completed. The minimum power consumption is achieved by the hard standby state. The details of the sequence are described below and shown in Figure 38.

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

Figure 38: Hard Standby and Hard Reset



Soft Standby and Soft Reset

The MT9P014 can reduce power consumption by switching to the soft standby state when the output is not needed. Register values are retained in the soft standby state. Once this state is reached, soft reset can be enabled optionally to return all register values back to the default. The details of the sequence are described below and shown in Figure 39.

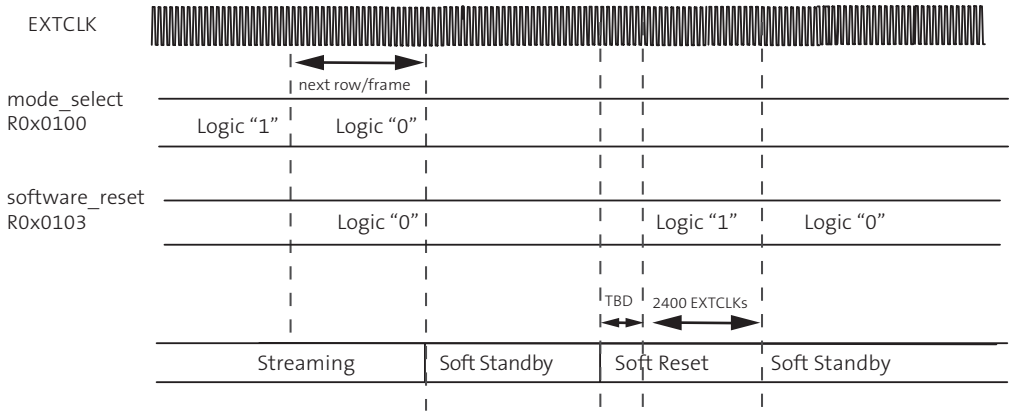
Soft Standby

1. If output is active, set mode_select = 0 (R0x0100) to disable streaming.
2. The soft standby state is reached after the current row or frame, depending on configuration, has ended.

Soft Reset

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

Figure 39: Soft Standby and Soft Reset

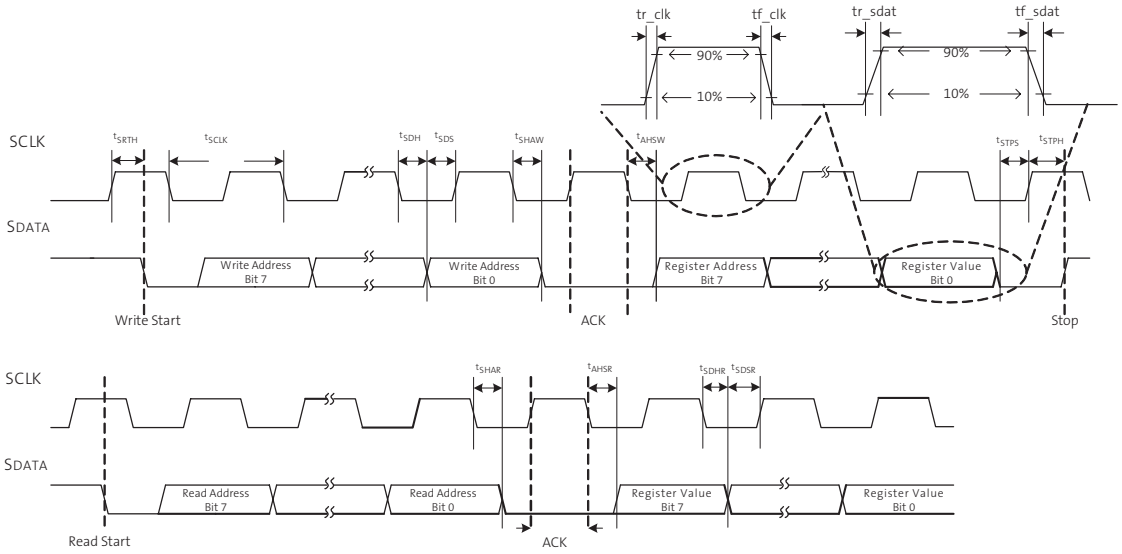


Electrical Specifications

Two-Wire Serial Register Interface

The electrical characteristics of the two-wire serial register interface (SCLK, SDATA) are shown in Figure 40 and Table 25. The SCLK and SDATA signals feature fail-safe input protection, Schmitt trigger input, and suppression of input pulses of less than 50ns.

Figure 40: Two-Wire Serial Bus Timing Parameters



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

Table 25: Two-Wire Serial Register Interface Electrical Characteristics

f_{EXTCLK} = 24 MHz; V_{DD} = 1.8V; V_{DD_IO} = 1.8V; V_{AA} = 2.8V; V_{AA_PIX} = 2.8V; V_{DD_PLL} = 2.8V; Output load = 68.5pF; Ambient temperature

Definition	Condition	Symbol	Min	Typ	Max	Unit
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 D_{GND}	I_{IN}			2	µA
Output LOW voltage	At specified 3mA	V_{OL}	0.11		0.275	V
Output LOW current	At specified V_{OL} = 0.1V	I_{OL}			3	mA
Input pad capacitance		C_{IN}			6	pF
Load capacitance		C_{LOAD}			N/A	pF

Table 26: Two-Wire Serial Register Interface Timing Specification

^{f}EXTCLK = 24 MHz; V_{DD} = 1.8V; V_{DD_IO} = 1.8V; V_{AA} = 2.8V; V_{AA_PIX} = 2.8V; V_{DD_PLL} = 2.8V; Output load = 68.5pF; Ambient temperature

Symbol	Definition	Condition	Min	Typ	Max	Unit
^{f}SCLK	Serial interface input clock	–	0		400	kHz
	SCLK duty cycle	VOD	45	50	60	%
tr	SCLK/SDATA rise time				300	µs
^{t}SRTS	Start setup time	Master write to slave	0.6			µs
^{t}SRTH	Start hold time	Master write to slave	0.3			µs
^{t}SDH	SDATA hold	Master write to slave	0.3		0.65	µs
^{t}SDS	SDATA setup	Master write to slave	0.3			µs
^{t}SHAW	SDATA hold to ACK	Master read to slave	0.15		0.65	µs
^{t}AHSW	ACK hold to SDATA	Master write to slave	0.15		0.70	µs
^{t}STPS	Stop setup time	Master write to slave	0.3			µs
^{t}STPH	Stop hold time	Master write to slave	0.6			µs
^{t}SHAR	SDATA hold to ACK	Master write to slave	0.3		1.65	µs
^{t}AHSR	ACK hold to SDATA	Master write to slave	0.3		0.70	µs
^{t}SDHR	SDATA hold	Master read from slave	0.3		0.65	µs
^{t}SDSR	SDATA setup	Master read from slave	0.3			µs

EXTCLK

The electrical characteristics of the EXTCLK input are shown in Table 27. The EXTCLK input supports an AC-coupled sine-wave input clock or a DC-coupled square-wave input clock.

Caution **If EXTCLK is AC-coupled to the MT9P014 and the clock is stopped, the EXTCLK input to the MT9P014 must be driven to ground or to V_{DD_IO}. Failure to do this will result in excessive current consumption within the EXTCLK input receiver.**

Table 27: Electrical Characteristics (EXTCLK)

^{f}EXTCLK = 24 MHz; V_{DD} = 1.8V; V_{DD_IO} = 1.8V; V_{AA} = 2.8V; V_{AA_PIX} = 2.8V; V_{DD_PLL} = 2.8V; Output load = 68.5pF; Ambient temperature

Definition	Condition	Symbol	Min	Typ	Max	Unit
Input clock frequency	PLL enabled	^{f}EXTCLK1	6		27	MHz
Input clock period	PLL enabled	^{t}EXTCLK1	37		166	ns
Input clock rise slew rate		^{t}R	1			V/ns
Input clock fall slew rate		^{t}F	1			V/ns
Input clock minimum voltage swing (AC coupled)		V_{IN_AC}	0.5			MHz
Input clock maximum voltage swing (DC coupled)		V_{IN_DC}			V_{DD_IO} + 0.5	V
Input clock signalling frequency (low amplitude)	V_{IN} = V_{IN_AC} (MIN)	^{f}CLKMAX(AC)			27	MHz
Input clock signalling frequency (full amplitude)	V_{IN} = V_{DD_IO}	^{f}CLKMAX(DC)			48	MHz
Clock duty cycle			45	50	55	%
Input clock jitter	cycle to cycle	^{t}JITTER			500	ps
PLL VCO lock time		^{t}LOCK		0.2	1	µs
Input pad capacitance		C_{IN}		3.5		pF
Input HIGH leakage current		I_{IH}	–10		10	µA
Input HIGH voltage		V_{IH}	V_{DD_IO} x 0.7		2.9	V
Input LOW voltage		V_{IL}	–0.5		0.3 x V_{DD_IO}	V

Serial Pixel Data Interface

The electrical characteristics of the serial pixel data interface (CLK_P, CLK_N, DATA0_P, DATA1_P, DATA0_N, and DATA1_N) are shown in Table 28 and Table 29.

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

Table 28: Electrical Characteristics (Serial CCP2 Pixel Data Interface)

Definition	Condition	Symbol	Min	Typ	Max	Unit
Operating frequency					650	MHz
Fixed common mode voltage		V_{CMF}	0.8		1.0	mV
Differential voltage swing		V_{OD}		137	200	mV
Drive current range			0.833	1.3	2	mA
Drive current variation				15	20	%
Output impedance			40	67	140	Ω
Output impedance mismatch				16.5	19	%
Clock Duty cycle at 416 MHz			45	50	55	%
Rise time (20–80)%					400	ps
Fall time (20–80)%					400	ps
Differential skew					50	ps
Channel to channel skew				100	150	ps
Maximum data rate Data/strobe mode Data/clock mode					 650 280	Mb/s
Power supply rejection ratio (0–100 MHz)					TBD	

Table 29: Electrical Characteristics (Serial MIPI Pixel Data Interface)

^{f}EXTCLK = 24 MHz; V_{DD} = 1.8V; V_{DD}_TX0 = 1.8V; V_{AA} = 2.8V; VAA_PIX = 2.8V; V_{DD}_PLL = 2.8V; Output Load = 68.5pF; Ambient temperature

Definition	Symbol	Min	Typ	Max	Unit
High speed transmit differential voltage	V_{OD}	140		270	mV
High speed transmit static common-mode voltage	V_{CMTX}	150		250	mV
V_{OD} mismatch when output is Differential-1 or Differential-0	dVOD			<14	mV
High speed output high voltage mismatch	dVCMTX			16	mV
Single ended output impedance	Z_{OS}	40		64	Ω
Single ended output impedance mismatch	dZOS			10	%
20–80% rise time	^{t}R			250	ps
20–80% fall time	^{t}F			250	ps
Output LOW level	V_{OL}			50	mV
Output HIGH level	V_{OH}			1.3	V
Output impedance of low power parameter	Z_{OLP}			110	Ω
15–85% rise time	T_{RLP}			25	ns
15–85% fall time	T_{FLP}			25	ns
Slew rate (C_{LOAD} = 5–20pF)	dv/dtsr			235	mV/ns
Slew rate (C_{LOAD} = 20–70pF)	dv/dstr			200	mV/ns

Control Interface

The electrical characteristics of the control interface (RESET_BAR, TEST, GPI0, GPI1, GPI2, and GPI3) are shown in Table 30.

Table 30: AC Electrical Characteristics (Control Interface)

^{f}EXTCLK = 24 MHz; V_{DD} = 1.8V; V_{DD}_IO = 1.8V; V_{AA} = 2.8V; VAA_PIX = 2.8V; V_{DD}_PLL = 2.8V; Output load = 68.5pF; Ambient temperature

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 D_{GND}	I_{IN}	–10		10	µA
Input pad capacitance		C_{IN}		6.5		pF

Operating Voltages

V_{AA} and V_{AA_PIX} must be at the same potential for correct operation of the MT9P014.

Table 31: DC Electrical Definitions and Characteristics

^{f}EXTCLK = 24 MHz; V_{DD} = 1.8V; V_{DD_IO} = 1.8V; V_{AA} = 2.8V; V_{AA_PIX} = 2.8V; V_{DD_PLL} = 2.8V; Output Load = 68.5pF; Ambient temperature

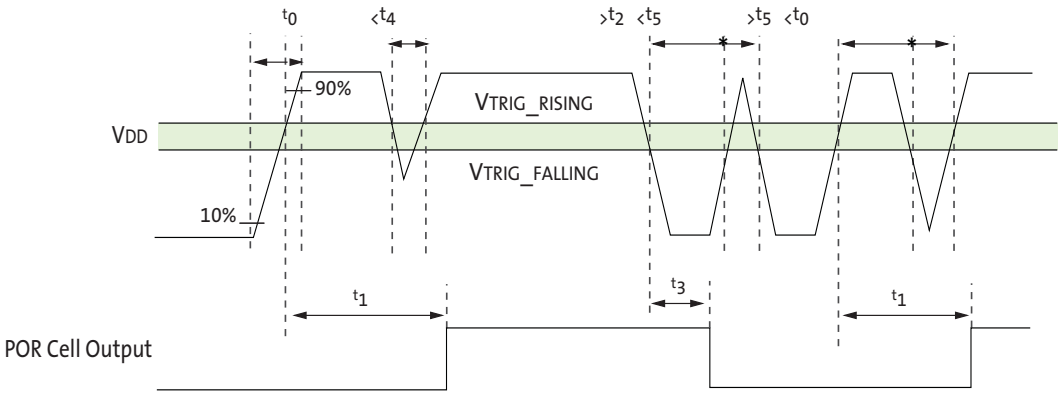
Definition	Condition	Symbol	Min	Typ	Max	Unit
Core digital voltage		V_{DD}	1.7	1.8	1.9	V
I/O digital voltage		V_{DD_IO}	1.7	1.8	1.9	V
Analog voltage		V_{AA}	2.4	2.8	3.1	V
Pixel supply voltage		V_{AA_PIX}	2.4	2.8	3.1	V
PLL supply voltage		V_{DD_PLL}	2.4	2.8	3.1	V
Digital operating current	Streaming, full resolution			47		mA
I/O digital operating current	Streaming, full resolution			0.011		mA
Analog operating current	Streaming, full resolution			48		mA
Pixel supply current	Streaming, full resolution			3.1		mA
PLL supply current	Streaming, full resolution			28		mA
Hard standby	(clock on)	Analog		14		µA
		Digital		18		µA
Hard standby	(clock off)	Analog		14		µA
		Digital		17		µA
Soft standby	(clock on)	Analog		47		µA
		Digital		1562		µA
Soft standby	(clock off)	Analog		14		µA
		Digital		19		µA

Power-On Reset

Table 32: 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	µs
VDD falling, crossing VTRIG_FALLING; Internal reset active		t_2		0.5	1	µs
Minimum VDD spike width below VTRIG_FALLING; considered to be a reset when POR cell output is HIGH		t_3		0.5		µs
Minimum VDD spike width below VTRIG_FALLING; considered to be a reset when POR cell output is LOW		t_4		1.0		µs
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	1.15	1.4	1.55	V
VDD falling trigger voltage		VTRIG_FALLING	1.00	1.25	1.45	V

Figure 41: Internal Power-On Reset



Absolute Maximum Ratings

Caution **Stresses greater than those listed in Table 33 may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect reliability.**

Table 33: Absolute Maximum Values

Definition	Condition	Symbol	Min	Typ	Max	Unit
Core digital voltage		V_{DD_MAX}	1.7	1.8	1.9	V
I/O digital voltage		$V_{DD\_IO\_MAX}$	1.7	1.8	1.9	V
Analog voltage		V_{AA}	2.4	2.8	3.1	V
Pixel supply voltage		VAA_PIX	2.4	2.8	3.1	V
PLL supply voltage		V_{DD_PLL}	2.4	2.8	3.1	V
Digital operating current	Worst case current	I_{DD}		–	55	mA
I/O digital operating current	Worst case current	I_{DD_IO}		–	0.016	mA
Analog operating current	Worst case current	I_{AA}		–	61	mA
Pixel supply current	Worst case current	I_{AA_PIX}		–	3.7	mA
PLL supply current	Worst case current	I_{DD_PLL}		–	32	mA
Operating temperature	Measure at junction	T_{OP}		–30	70	°C
Storage temperature		T_{STG}		–40	85	°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.

SMIA and MIPI Specification Reference

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

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

Revision History

Rev. E 3/16/12

- Updated trademarks
- Applied updated Aptina template

Rev D 1/21/10

- Moved register tables to a separate document, MT9P014 Register Reference

Rev C 6/3/09

- Changed the following parameters in Table 1, "Key Performance Parameters," on page 1:
  - Chief Ray Angle: from TBD to 25 deg.
  - ADC resolution: from TBD to 12-bit
  - Responsivity: from TBD to 1.7 V/Lux-sec
  - Dynamic range: from TBD to 69dB
  - SNR_{MAX}: from TBD to 39dB
  - Supply voltage (I/O Digital): deleted "or 2.4–3.1V (2.8V nominal)"
  - Power Consumption (Full resolution): changed332mW
  - Power Consumption (Standby): changed to 69.8µW
- Changed the default value of revision_number from 30(0x0020) to 48 (0x0030)
- Deleted Table 42 because it duplicated Table 37.
- Added section on "Lens Shading Coefficient Read/Write Procedure" on page 35.
- Deleted 500ms maximum for VDD_IO to VDD time, VDD to VDD_PLL time, and VDD to VAA/VAA_PIX time from Table 23 on page 65.
- Deleted 500ms maximum for VDD/VAA/VAA_PIX to VDD time, VDD_PLL to VDD time, and VDD to VDD_IO time from Table 24 on page 66.
- Updated typical value of Analog operating current to 57 in Table 31, "DC Electrical Definitions and Characteristics," on page 74
- Updated maximum value of Analog operating current to 61 in Table 33, "Absolute Maximum Values," on page 76
- Updated register tables

Rev. B 3/27/2008

- Updated Figure 2: "Pixel Color Pattern Detail (Top Right Corner)," on page 8 to show first clear pixel at (100,52)
- Updated EXTCLK from 6–64 MHz to 6–27 MHz throughout the document
- Updated register data from RegDB
- Updated from Table 1, "Register List and Default Values—SMIA Configuration," on page 24 through Table 9, "Register Description—Manufacturer-Specific," on page 53
- Updated Table 1, "Key Performance Parameters," on page 1, Table 28, "Electrical Characteristics (Serial CCP2 Pixel Data Interface)," on page 72, and Table 42, "Serial CCP2 Pixel Data Interface," on page 123 to call out 650 Mbps

Rev. A 07/17/2007

- Initial release

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