1/4-Inch 3.1Mp Digital Image Sensor

MT9T013 Data Sheet

Refer to the latest data sheet on Aptina's Web site at 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, frame size/rate, exposure, left–right and top–bottom image reversal, window size and panning
- SMIA-compatible
- Data interfaces: parallel and CCP2 compliant sub-low-voltage differential signalling (sub-LVDS)
- On-chip phase-locked loop (PLL) oscillator
- Bayer-pattern down-size scaler
- Integrated lens shading correction
- One-time programmable (OTP) memory for storing module information
- Superior low-light performance

Applications

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

General Description

The Aptina MT9T013 is a 1/4-inch QXGA-format CMOS active-pixel digital image sensor with a pixel array of 2048H x 1536V (2064H x 1552V 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		1/4-inch QXGA (4:3)
Active imager size		3.61mm(H) x 2.72mm(V) 4.52mm diagonal
Active pixels		2064H x 1552V
Pixel size		1.75 x 1.75µm
Color filter array		RGB Bayer pattern
Shutter type		Electronic rolling shutter (ERS)
Maximum data rate		72 Mpixels/s at parallel interface 650 Mbits/s at CCP2 interface
Frame rate	QXGA (2048 x 1536)	Programmable up to 15 fps
	XGA (1024 x 768)	Programmable up to 30 fps
ADC resolution		10-bit, on-chip (61dB)
Responsivity		0.57 V/lux-sec
Dynamic range		64.4dB
SNR_{MAX}		39.2dB
Supply voltage	Analog	2.40–3.10V (2.80V nominal)
	Digital	1.70–1.90V (1.80V nominal)
	I/O	1.70–1.90V or 2.40–3.10V
Power consumption		360mW at 15 fps, full resolution
Operating temperature		–30°C to +70°C
Packaging		Bare die

Ordering Information

Table 2: Available Part Numbers

Part Number	Description
MT9T013D00STCPC26AC1	Bare die

Table of Contents

Features . . . 1
Applications . . . 1
General Description . . . 1
Ordering Information . . . 1
General Description . . . 7
Functional Overview . . . 7
Pixel Array . . . 8
Operating Modes . . . 9
Signal Descriptions . . . 11
Output Data Format . . . 12
CCP Serial Pixel Data Interface . . . 12
Parallel Pixel Data Interface . . . 12
Parallel Pixel Data Interface Output Data Timing . . . 12
Two-Wire Serial Register Interface . . . 14
Protocol . . . 14
Start Condition . . . 14
Slave Address/Data Direction Byte . . . 14
Acknowledge Bit . . . 14
No-Acknowledge Bit . . . 15
Message Byte . . . 15
Stop Condition . . . 15
Data Transfer . . . 15
Typical Sequence . . . 15
Single READ from Random Location . . . 16
Single READ from Current Location . . . 16
Sequential READ, Start from Random Location . . . 17
Sequential READ, Start from Current Location . . . 17
Single Write to Random Location . . . 17
Sequential WRITE, Start at Random Location . . . 18
Registers . . . 19
Register Notation . . . 19
Register Aliases . . . 19
Bit Fields . . . 19
Bit Field Aliases . . . 20
Byte Ordering . . . 20
Address Alignment . . . 20
Bit Representation . . . 20
Data Format . . . 20
Register Behavior . . . 20
Double-Buffered Registers . . . 20
Using grouped_parameter_hold . . . 21
Bad Frames . . . 21
Changes to Integration Time . . . 21
Changes to Gain Settings . . . 22
Embedded Data . . . 22
Programming Restrictions . . . 23
Effect of Scaler on Legal Range of Output Sizes . . . 25
Effect of CCP Class on Legal Range of Output Sizes/Frame Rate . . . 26
Output Data Timing . . . 27
Programming Restrictions when Using Global Reset . . . 28
Control of the Signal Interface . . . 29

Serial Register Interface 29
Default Power-up State 29
Serial Pixel Data Interface 29
Parallel Pixel Data Interface 30
Output Enable Control 30
Configuration of the Pixel Data Interface 30
System States 31
Power-On Reset Sequence 32
Soft Reset Sequence 32
Signal State During Reset 33
General Purpose Inputs 33
Streaming/Standby Control 34
Trigger Control 34
Clocking 35
Programming the PLL Divisors 37
Influence of ccp_data_format 37
Influence of ccp2_signalling_mode 38
Clock Control 38
Features 39
Shading Correction (SC) 39
The Correction Function 39
One-Time Programmable (OTP) Memory 39
Image Acquisition Modes 41
Window Control 41
Pixel Border 41
Readout Modes 41
Horizontal Mirror 41
Vertical Flip 42
Subsampling 42
Binning 47
Frame Rate Control 50
Integration Time 50
Flash Control 51
Global Reset 52
Overview of Global Reset Sequence 52
Entering and Leaving the Global Reset Sequence 53
Programmable Settings 53
Control of the Electromechanical Shutter 54
Using FLASH with Global Reset 55
External Control of Integration Time 55
Retriggering the Global Reset Sequence 56
Global Reset and Soft Standby 57
Analog Gain 57
Using Per-color or Global Gain Control 57
SMIA Gain Model 58
Aptina Gain Model 58
Gain Code Mapping 58
Sensor Core Digital Data Path 60
Test Patterns 60
Effect of Data Path Processing on Test Patterns 60
Solid Color Test Pattern 61
100 Percent Color Bars Test Pattern 61
Fade-to-Gray Color Bars Test Pattern 61

Walking 1s Test Pattern 63
Test Cursors 63
Digital Gain 65
Pedestal 65
Digital Data Path 66
Embedded Data Format and Control 66
Timing Specifications 67
Power-Up Sequence 67
Power-Down Sequence 68
Hard Standby and Hard Reset 69
Soft Standby and Soft Reset 70
Soft Standby 70
Soft Reset 70
Spectral Characteristics 71
Electrical Specifications 73
EXTCLK 73
Parallel Pixel Data Interface 74
Two-Wire Serial Register Interface 76
Serial Pixel Data Interface 78
Control Interface 78
Power-On Reset 79
Operating Voltages 80
Absolute Maximum Ratings 81
SMIA Specification Reference 81
Revision History 82

List of Figures

Figure 1: Block Diagram . . . 7
Figure 2: Pixel Color Pattern Detail (Top Right Corner) . . . 8
Figure 3: Typical Configuration: Serial Pixel Data Interface . . . 9
Figure 4: Typical Configuration: Parallel Pixel Data Interface . . . 10
Figure 5: Spatial Illustration of Image Readout . . . 12
Figure 6: Pixel Data Timing Example . . . 13
Figure 7: Row Timing and FV/LV Signals . . . 13
Figure 8: Single READ from Random Location . . . 16
Figure 9: Single READ from Current Location . . . 16
Figure 10: Sequential READ, Start from Random Location . . . 17
Figure 11: Sequential READ, Start from Current Location . . . 17
Figure 12: Single WRITE to Random Location . . . 17
Figure 13: Sequential WRITE, Start at Random Location . . . 18
Figure 14: Effect of Limiter on SMIA Data Path . . . 25
Figure 15: Timing of SMIA Data Path . . . 26
Figure 16: MT9T013 System States . . . 31
Figure 17: MT9T013 SMIA Profile 1, 2 Clocking Structure . . . 35
Figure 18: MT9T013 SMIA Profile 0 Clocking Structure . . . 36
Figure 19: Sequence for Programming the Device . . . 40
Figure 20: Effect of horizontal_mirror on Readout Order . . . 42
Figure 21: Effect of vertical_flip on Readout Order . . . 42
Figure 22: Effect of x_odd_inc = 3 on Readout Sequence . . . 42
Figure 23: Effect of x_odd_inc = 7 on Readout Sequence . . . 43
Figure 24: Pixel Readout (No Subsampling) . . . 43
Figure 25: Pixel Readout (x_odd_inc = 3, y_odd_inc = 3) . . . 44
Figure 26: Pixel Readout (x_odd_inc = 7, y_odd_inc = 7) . . . 44
Figure 27: Pixel Readout (x_odd_inc = 3, y_odd_inc = 1, x_bin = 1) . . . 47
Figure 28: Pixel Readout (x_odd_inc = 3, y_odd_inc = 3, xy_bin = 1) . . . 48
Figure 29: Pixel Readout (x_odd_inc = 7, y_odd_inc = 7, xy_bin = 1) . . . 48
Figure 30: Xenon Flash Enabled . . . 51
Figure 31: LED Flash Enabled . . . 51
Figure 32: LED Flash Enabled Following Forced Restart . . . 52
Figure 33: Overview of Global Reset Sequence . . . 53
Figure 34: Entering and Leaving a Global Reset Sequence . . . 53
Figure 35: Controlling the Reset and Integration Phases of the Global Reset Sequence . . . 54
Figure 36: Control of the Electromechanical Shutter . . . 54
Figure 37: Controlling the SHUTTER Output . . . 55
Figure 38: Using FLASH with Global Reset . . . 55
Figure 39: Global Reset Bulb . . . 56
Figure 40: Entering Soft Standby During a Global Reset Sequence . . . 57
Figure 41: 100 Percent Color Bars Test Pattern . . . 61
Figure 42: Fade-to-Gray Color Bars Test Pattern . . . 62
Figure 43: Test Cursor Behavior when image_orientation . . . 65
Figure 44: Data Path . . . 66
Figure 45: Power-Up Sequence . . . 67
Figure 46: Power-Down Sequence . . . 68
Figure 47: Hard Standby and Hard Reset . . . 69
Figure 48: Soft Standby and Soft Reset . . . 70
Figure 49: Chief Ray Angle (CRA) (Z18 Type A) . . . 71
Figure 50: Chief Ray Angle (CRA) (Z19 Type B) . . . 72
Figure 51: Quantum Efficiency . . . 72
Figure 52: Default Data Output Timing Diagram . . . 73
Figure 53: Two-Wire Serial Bus Timing Parameters . . . 76
Figure 54: Internal Power-On Reset . . . 79

List of Tables

Table 1: Key Performance Parameters . . . 1
Table 2: Available Part Numbers . . . 1
Table 3: Signal Descriptions . . . 11
Table 4: Row Timing . . . 13
Table 5: Address Space Regions . . . 19
Table 6: Data Formats . . . 20
Table 7: Definitions for Programming Rules . . . 23
Table 8: Programming Rules . . . 23
Table 9: Output Enable Control . . . 30
Table 10: Configuration of the Pixel Data Interface . . . 30
Table 11: PLL in System States . . . 32
Table 12: Signal State During Reset . . . 33
Table 13: Streaming/STANDBY . . . 34
Table 14: Trigger Control . . . 34
Table 15: Row Address Sequencing . . . 46
Table 16: Register Adjustments Required for Binning Mode . . . 49
Table 17: Recommended Gain Settings . . . 58
Table 18: Test Patterns . . . 60
Table 19: Power-Up Sequence . . . 67
Table 20: Power-Down Sequence . . . 68
Table 21: Electrical Characteristics (EXTCLK) . . . 74
Table 22: Electrical Characteristics (Parallel Pixel Data Interface) . . . 74
Table 23: Two-Wire Serial Register Interface Electrical Characteristics . . . 76
Table 24: Two-Wire Serial Interface Timing Specifications . . . 77
Table 25: Electrical Characteristics (Serial Pixel Data Interface) . . . 78
Table 26: AC Electrical Characteristics (Control Interface) . . . 78
Table 27: Power-On Reset Characteristics . . . 79
Table 28: DC Electrical Definitions and Characteristics . . . 80
Table 29: Absolute Maximum Values . . . 81

General Description

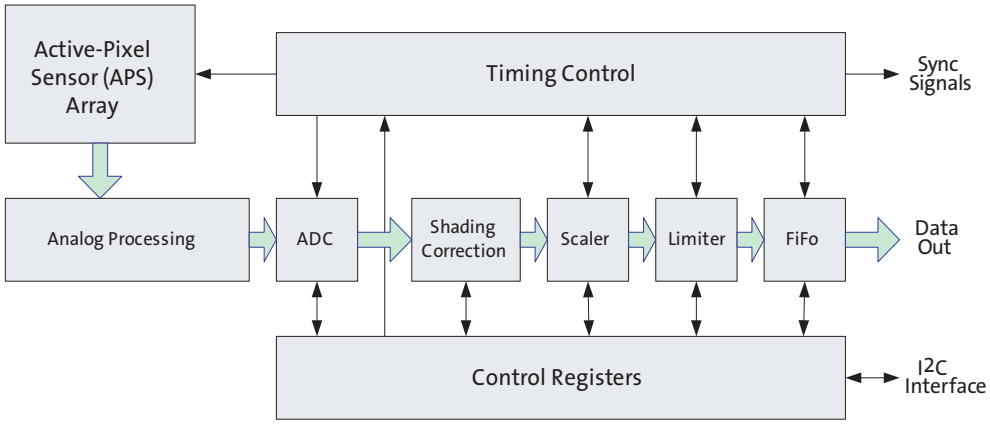
The MT9T013 digital image sensor features Aptina's breakthrough low-noise CMOS imaging technology that achieves near-CCD image quality (based on signal-to-noise ratio and low-light sensitivity) while maintaining the inherent size, cost, and integration advantages of CMOS.

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

Functional Overview

The MT9T013 is a progressive-scan sensor that generates a stream of pixel data at a constant frame rate. It uses an on-chip, phase-locked loop (PLL) to generate all internal clocks from a single master input clock running between 6 and 48 MHz. The maximum pixel rate is 72 Mb/s, corresponding to a pixel clock rate of 72 MHz. A block diagram of the sensor is shown in Figure 1.

Figure 1: Block Diagram



The core of the sensor is a 3.1Mp active-pixel array. The timing and control circuitry sequences through the rows of the array, resetting and then reading each row in turn. In the time interval between resetting a row and reading that row, the pixels in the row integrate incident light. The exposure is controlled by varying the time interval between reset and readout. Once a row has been read, the data from the columns is sequenced through an analog signal chain (providing offset correction and gain), and then through an ADC. The output from the ADC is a 10-bit value for each pixel in the array. The ADC output passes through a digital processing signal chain (which provides further data path corrections and applies digital gain).

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

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

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

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

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

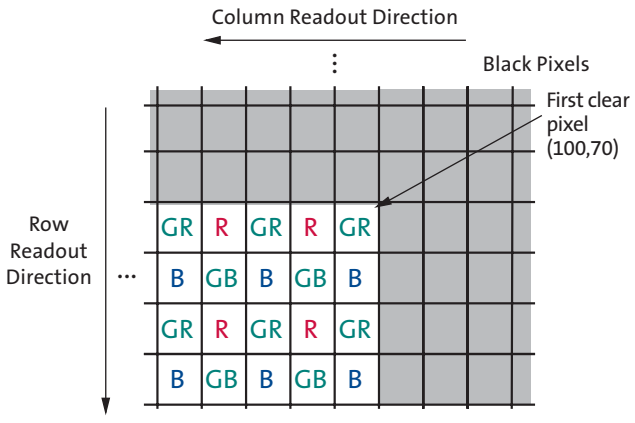
The output FIFO prevents data bursts by keeping the data rate continuous. Programmable slew rates are also available to reduce the effect of electromagnetic interference from the output interface.

A flash output strobe allows 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, shown in Figure 2. The even-numbered rows contain green and red pixels; odd-numbered rows contain blue and green pixels. Even-numbered columns contain green and blue pixels; odd-numbered columns contain red and green pixels.

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

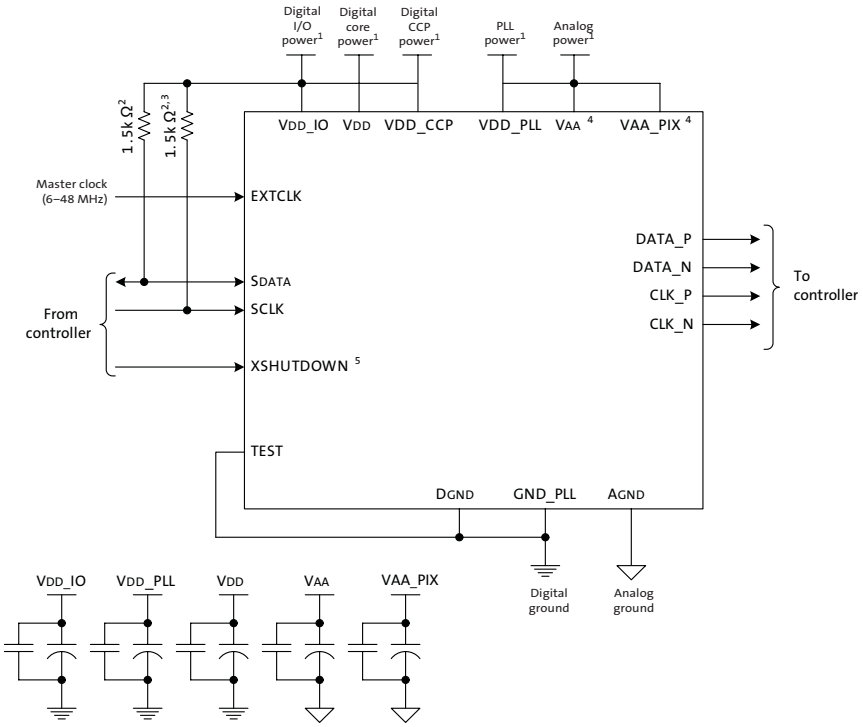


Operating Modes

By default, the MT9T013 powers up as a SMIA-compatible sensor with the serial pixel data interface enabled. A typical configuration in this mode is shown in Figure 3. The MT9T013 can also be configured to operate with a parallel pixel data interface. A typical configuration in this mode is shown in Figure 4. These two operating modes are described in "Control of the Signal Interface" on page 29.

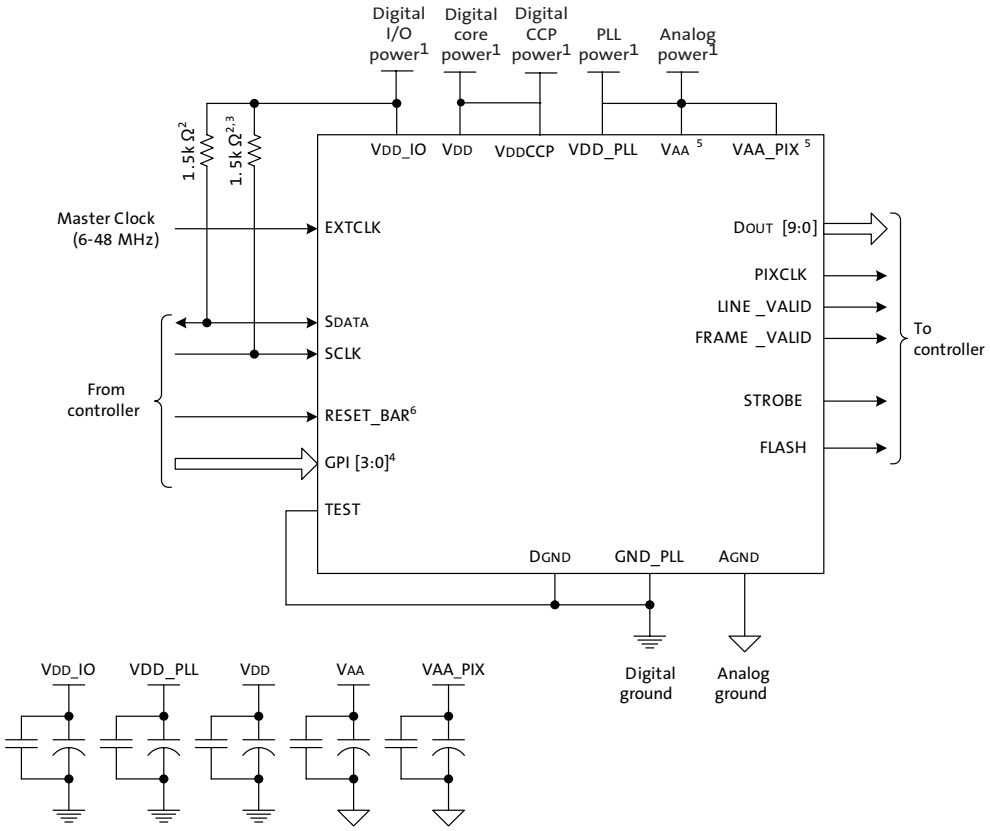
For low-noise operation, the MT9T013 requires separate power supplies for analog and digital. Incoming digital and analog ground conductors can be tied together next to the die. Both power supply rails should be decoupled to ground using capacitors as close as possible to the die. The use of inductance filters is not recommended on the power supplies or output signals.

Figure 3: Typical Configuration: Serial Pixel Data Interface



Notes:
1. All power supplies should be adequately decoupled.
2. Aptina recommends a resistor value of 1.5kΩ, but may be greater for slower two-wire speed.
3. This pull-up resistor is not required if the controller drives a valid logic level on SCLK at all times.
4. VAA and 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 Figure 3. This pad is left unconnected during normal operation.
7. The parallel interface output pads can be left unconnected if the serial output interface is used.
8. 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.
9. VDD_IO2, VDD_IO3, VDD2, DGND2, and DGND3 can be left unconnected for a serial pixel data interface.
10. TEST must be tied to DGND.
11. Aptina recommends that GND_PLL be tied to DGND.
12. Aptina recommends that VDD_CCP be tied to VDD.

Figure 4: Typical Configuration: Parallel Pixel Data Interface



Notes:
1. All power supplies should be adequately decoupled.
2. Aptina recommends a resistor value of 1.5kΩ, but may be greater for slower two-wire speed.
3. This pull-up resistor is not required if the controller drives a valid logic level on SCLK at all times.
4. The GPI pins can be statically pulled HIGH/LOW to be used as module IDs, or they can be programmed to perform special functions (TRIGGER, OE_N, SADDR, STANDBY) to be dynamically controlled.
5. VAA and VAA_PIX must be tied together.
6. Also referred to as XSHUTDOWN.
7. 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.
8. The serial interface output pads can be left unconnected if the parallel output interface is used.
9. 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.
10. TEST must be tied to DGND.
11. Aptina recommends that GND_PLL be tied to DGND.
12. Aptina recommends that VDD_CCP be tied to VDD.

Signal Descriptions

Table 3 provides signal descriptions for MT9T013 die. For pad location and aperture information, refer to the MT9T013 die data sheet.

Table 3: Signal Descriptions

Pad Name	Pad Type	Description
EXTCLK	Input	Disconnect pad after programming or when feature is not used. Master clock input. PLL input clock. 6–48 MHz.
RESET_BAR (XSHUTDOWN)	Input	Asynchronous active LOW reset. When asserted, data output stops and all internal registers are restored to their factory default settings.
SCLK	Input	Serial clock for access to control and status registers.
GPI[3:0]	Input	General purpose inputs. After reset, these pads are powered-down by default; this means that it is not necessary to bond to these pads. Any of these pads can be configured to provide hardware control of the standby, output enable, SADDR select, and shutter trigger functions.
TEST	Input	Enable manufacturing test modes. Wire to digital GND for functional operation.
SDATA	I/O	Serial data for reads from and writes to control and status registers.
DATA_P	Output	Differential CCP (sub-LVDS) serial data (positive).
DATA_N	Output	Differential CCP (sub-LVDS) serial data (negative).
CLK_P	Output	Differential CCP (sub-LVDS) serial clock/strobe (positive).
CLK_N	Output	Differential CCP (sub-LVDS) serial clock/strobe (negative).
LINE_VALID	Output	LINE_VALID (LV) output. Qualified by PIXCLK.
FRAME_VALID	Output	FRAME_VALID (FV) output. Qualified by PIXCLK.
DOUT[9:0]	Output	Parallel pixel data output. Qualified by PIXCLK.
PIXCLK	Output	Pixel clock. Used to qualify the LV, FV, and DOUT[9:0] outputs.
FLASH	Output	Flash output. Synchronization pulse for external light source.
SHUTTER	Output	Control for external mechanical shutter.
VPP	Supply	Power supply used to program one-time programmable (OTP) memory. Disconnect pad when programming or when feature is not used.
VDD_CCP	Supply	Digital power supply for the CCP interface. Also known as VDD_TX_0. Aptina recommends that VDD_CCP be tied to VDD.
VAA1, VAA2, VAA3, VAA4	Supply	Analog power supply.
VAA_PIX1, VAA_PIX2, VAA_PIX3	Supply	Analog power supply for the pixel array.
AGND1, AGND2, AGND3, AGND4, AGND5, AGND6, AGND7	Supply	Analog ground.
VDD1, VDD2, VDD3 VDD4	Supply	Digital power supply.
VDD_IO1, VDD_IO2, VDD_IO3, VDD_IO4, VDD_IO5, VDD_IO6	Supply	I/O power supply.
DGND1, DGND2, DGND3 DGND4, DGND5, DGND6	Supply	Common ground for digital and I/O.
VDD_PLL	Supply	PLL power supply.
DGNDPLL	Supply	PLL ground.

Output Data Format

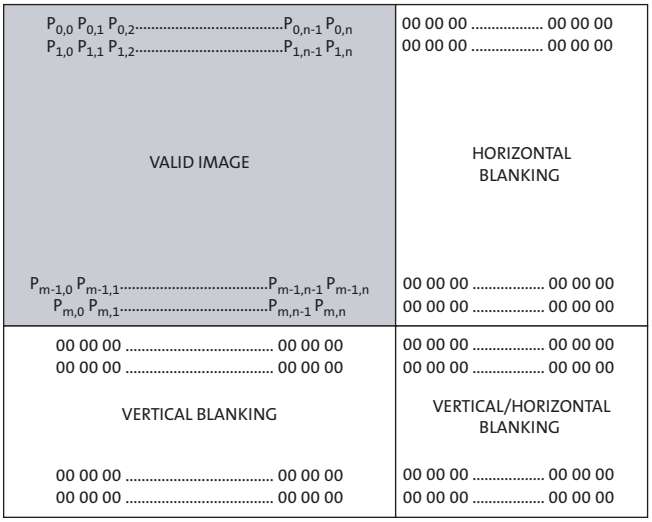
CCP Serial Pixel Data Interface

The MT9T013 serial pixel data interface implements data/clock and data/strobe signalling in accordance with the CCP2 specification. The RAW8 and RAW10 image data formats are supported.

Parallel Pixel Data Interface

MT9T013 image data is read out in a progressive scan. Valid image data is surrounded by horizontal blanking and vertical blanking, shown in Figure 5. The amount of horizontal blanking and vertical blanking is programmable; LV is HIGH during the shaded region of the figure. FV timing is described in the next section.

Figure 5: Spatial Illustration of Image Readout



Parallel Pixel Data Interface Output Data Timing

MT9T013 output data is synchronized with the PIXCLK output. When LV is HIGH, one pixel value is output on the 10-bit DOUT output every PIXCLK period. The pixel clock runs at the calculated frequency based on the sensor's master input clock and internal PLL configuration, and rising edges on the PIXCLK signal occur one-half of a pixel clock period after transitions on LV, FV, and DOUT (see Figure 6 on page 13). This allows PIXCLK to be used as a clock to sample the data. PIXCLK is continuously enabled, even during the blanking period. The MT9T013 can be programmed to delay the PIXCLK edge relative to the DOUT transitions. This can be achieved by programming the corresponding bits in the row_speed register.

Figure 6: Pixel Data Timing Example

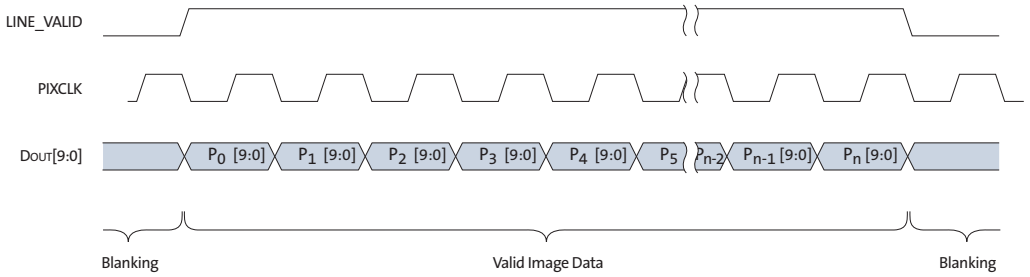


Figure 7: Row Timing and FV/LV Signals

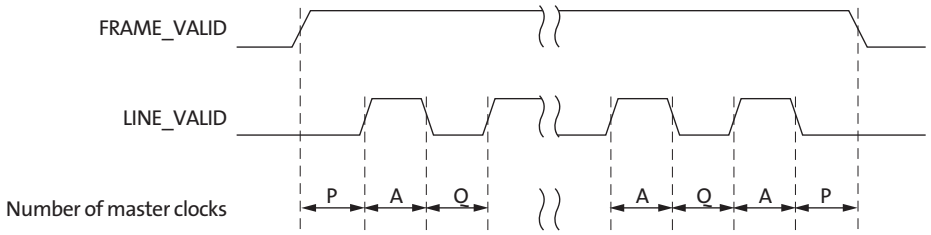


Table 4: Row Timing

Parameter	Name	Equation	Default Timing at 72 MHz
PIXCLK_PERIOD	Pixel clock period	R0x3016–7[2:0] / vt_pix_clk_freq_mhz	1 pixel clock = 13.889ns
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	2,048 pixel clocks = 28.44µs
P	Frame start/end blanking	6 * PIXCLK_PERIOD	6 pixel clocks = 83.33ns
Q	Horizontal blanking	(line_length_pck - A) * PIXCLK_PERIOD	12.36µs
A + Q	Row time	line_length_pck * PIXCLK_PERIOD	40.81µs
N	Number of rows	(y_addr_end - y_addr_start + y_odd_inc)/S	1,536 rows
V	Vertical blanking	([frame_length_lines - N] * [A+Q]) + Q - (2*P)	2.23ms
T	Frame valid time	(N * (A + Q)) - Q + (2*P)	62.67ms
F	Total frame time	line_length_pck * frame_length_lines * PIXCLK_PERIOD	66.15ms

The sensor timing (Table 4) is shown in terms of pixel clock and master clock cycles (see Figure 6). The default settings for the on-chip PLL generate a 72 MHz pixel clock given a 24 MHz input clock to the MT9T013. Equations for calculating the frame rate are given in "Frame Rate Control" on page 50.

Two-Wire Serial Register Interface

The two-wire serial interface bus enables read/write access to control and status registers within the sensor. This interface is designed to be compatible with the *SMIA 1.0 Part2: CCP2 Specification* camera control interface (CCI), which uses the electrical characteristics and transfer protocols of the 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 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 I^2C specification allow the slave device to drive SCLK protocols described in the I^2C specification allow the slave device to drive LOW; the MT9T013 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.

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 MT9T013 are 0x20 (write address) and 0x21 (read address) in accordance with the SMIA specification. Alternate slave addresses of 0x30 (write address) and 0x31 (read address) can be selected by enabling and asserting the SADDR signal through the GPI pad.

An alternate slave address can also be programmed by R0x31FC.

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.

Message Byte

Message bytes are used for sending register addresses and register write data to the slave device and for retrieving register read data. The protocol used is outside the scope of the I^2C specification and is defined as part of the SMIA CCI.

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.

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.

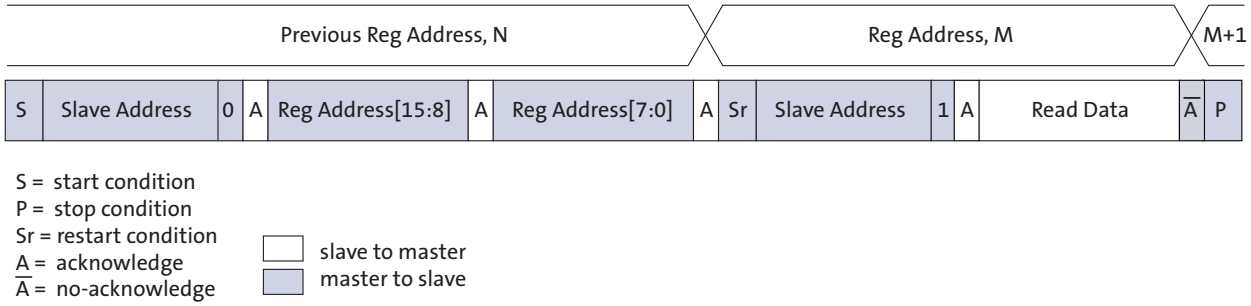
If the request was a WRITE, the master then transfers the 16-bit register address to which the write should take place. This transfer takes place as two 8-bit sequences and the slave sends an acknowledge bit after each sequence to indicate that the byte has been received. The master then transfers the 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 auto-incremented after every 8 bits are transferred. The data transfer is stopped when the master sends a no-acknowledge bit.

Single READ from Random Location

This sequence (Figure 8 on page 16) 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 8 shows how the internal register address maintained by the MT9T013 is loaded and incremented as the sequence proceeds.

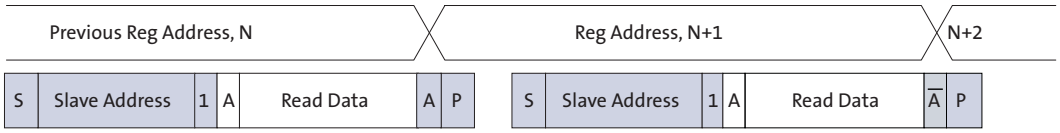
Figure 8: Single READ from Random Location



Single READ from Current Location

This sequence (Figure 9) performs a READ using the current value of the MT9T013 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.

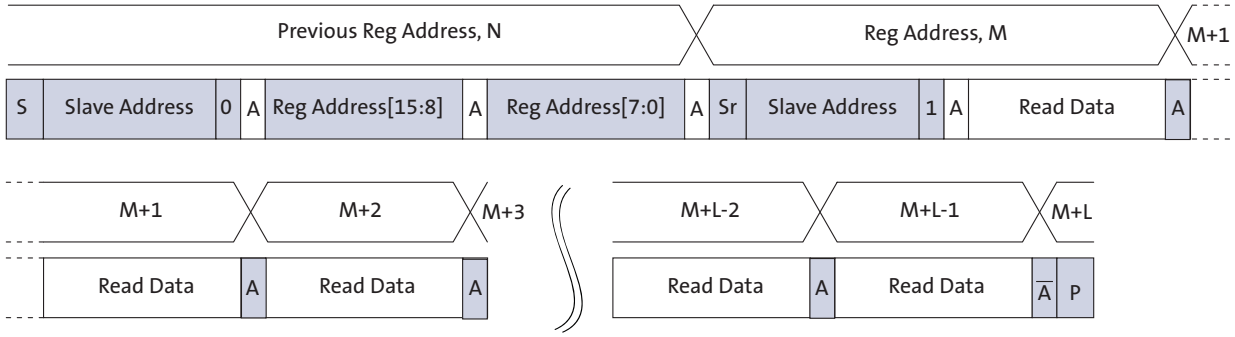
Figure 9: Single READ from Current Location



Sequential READ, Start from Random Location

This sequence (Figure 10) starts in the same way as the single READ from random location (Figure 8 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.

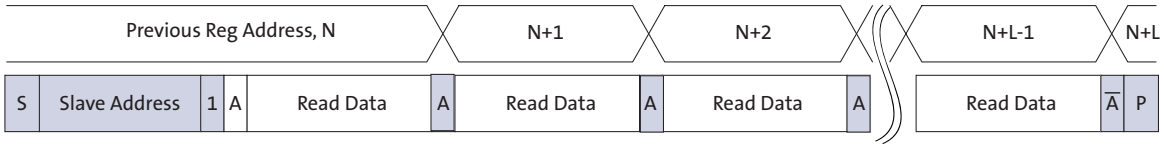
Figure 10: Sequential READ, Start from Random Location



Sequential READ, Start from Current Location

This sequence (Figure 11) starts in the same way as the single READ from current location (Figure 9 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.

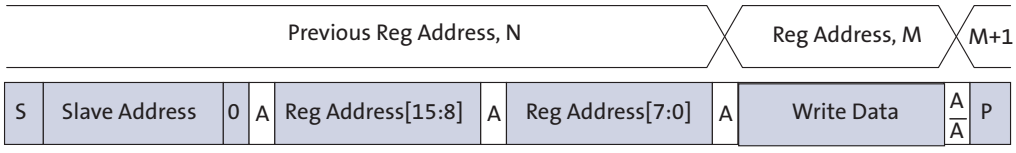
Figure 11: Sequential READ, Start from Current Location



Single Write to Random Location

This sequence (Figure 12) 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.

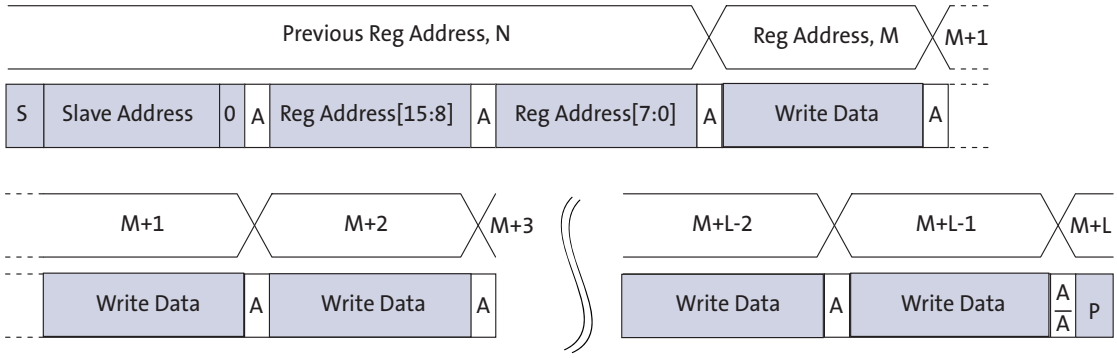
Figure 12: Single WRITE to Random Location



Sequential WRITE, Start at Random Location

This sequence (Figure 13) starts in the same way as the single WRITE to random location (Figure 12 on page 17). Instead of generating a no-acknowledge bit after the first byte of data has been transferred, the master generates an acknowledge bit and continues to perform byte WRITEs until "L" bytes have been written. The WRITE is terminated by the master generating a stop condition.

Figure 13: Sequential WRITE, Start at Random Location



Registers

The MT9T013 provides a 32-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.

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 1: 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 MT9T013 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, refer to the register table to determine the register size.

Register Aliases

A consequence of the internal architecture of the MT9T013 is that some registers are decoded at multiple addresses. Some registers in the configuration space are also decoded in the manufacturer-specific space. To provide unique names for all registers, the name of the register within the manufacturer-specific register space has a trailing underscore. For example, R0x0000–1 is model_id, and R0x3000–1 is model_id_ (see the register table for more examples). 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 SMIA 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 2: 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 MT9T013 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 MT9T013 is in streaming mode, this register should be written to "1" before making changes to any group of registers where a set of changes is required to take effect simultaneously. When this register is written to "0," all transfers from pending to live registers take place on the next frame start.

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

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

Bad Frames

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

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

By default, most bad frames are not masked. If the masked bad frame option is enabled, both LV and FV are inhibited for these frames so that the vertical blanking time between frames is extended by the frame time.

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

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

Changes to Integration Time

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

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

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

Changes to Gain Settings

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

Embedded Data

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

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

- The integration time is delayed by one further frame, so that the value corresponds to the integration time used for the image data in the frame. See “Changes to Integration Time” on page 3.
- 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 66.

Note: For the detailed register tables, see the MT9T013 Register Reference.

Programming Restrictions

The SMIA specification imposes a number of programming restrictions. An implementation naturally imposes additional restrictions. Table 8 shows a list of programming rules that must be adhered to for correct operation of the MT9T013. It is recommended that these rules are encoded into the device driver stack—either implicitly or explicitly.

Table 1: 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 2: Programming Rules

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

Table 2: Programming Rules (continued)

Parameter	Minimum Value	Maximum Value	Origin
y_odd_inc[0]	1	1	SMIA
scale_m	scaler_m_min	scaler_m_max	SMIA
scale_n	scaler_n_min	scaler_n_max	SMIA
x_output_size	256	2064	Minimum from SMIA FS Section 5.2.2.5. Maximum is a consequence of the output FIFO size on this implementation.
x_output_size[0]	0 (this is enforced in hardware: bit[0] is read-only)	0	SMIA FS Section 5.2.2.2.
y_output_size	2	frame_length_lines	Minimum ensures 1 Bayer row-pair. Maximum avoids output frame being longer than pixel array frame.
y_output_size[0]	0 (this is enforced in hardware: bit[0] is read-only)	0	SMIA FS Section 5.2.2.2
With subsampling, start and end pixels must be addressed (impact on x/y start/end addresses, function of image orientation bits)			SMIA FS Errata See "Subsampling" on page 20.

Output Size Restrictions

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

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

This restriction only applies when the serial pixel data path is in use. It can be met by rounding up x_output_size to an appropriate multiple. Any extra pixels in the output image as a result of this rounding contain undefined pixel data but are guaranteed not to cause false synchronization on the CCP2 data stream.

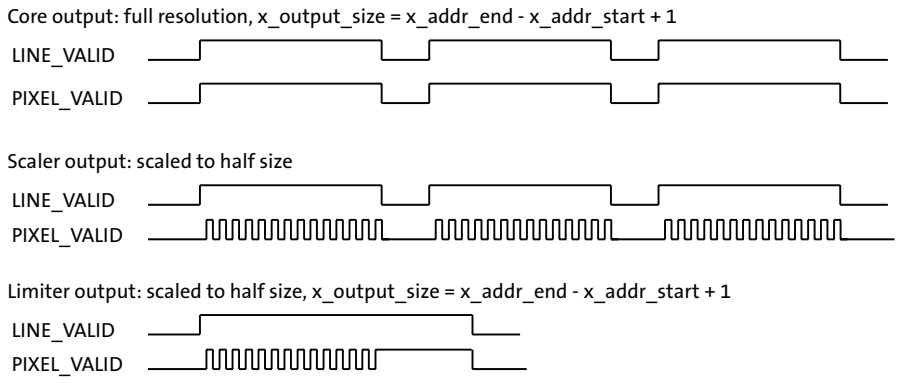
When the parallel pixel data path is in use, the only restriction on x_output_size is that it must be even (x_output_size[0] = 0), and this restriction is enforced in hardware.

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

Effect of Scaler on Legal Range of Output Sizes

When the scaler is enabled, it is necessary to adjust the values of x_output_size and y_output_size to match the image size generated by the scaler. The MT9T013 will not operate properly if the x_output_size and y_output_size are significantly larger than the output image. To understand the reason for this, consider the situation where the sensor is operating at full resolution and the scaler is enabled with a scaling factor of 32 (half the number of pixels in each direction). This situation is shown in Figure 14.

Figure 1: Effect of Limiter on SMIA Data Path



In this figure, three different stages in the SMIA data path (see “Digital Data Path” on page 44) are shown. The first stage is the output of the sensor core. The core is running at full resolution and x_output_size is set to match the active array size. The LV signal is asserted once per row and remains asserted for *N* pixel times. The PIXEL_VALID signal toggles with the same timing as LV, indicating that all pixels in the row are valid.

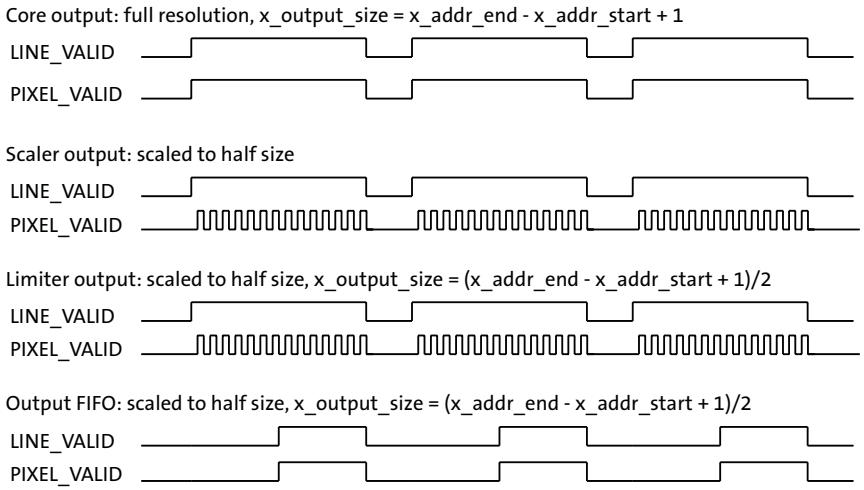
The second stage is the output of the scaler, when the scaler is set to reduce the image size by one-half in each dimension. The effect of the scaler is to combine groups of pixels. Therefore, the row time remains the same, but only half the pixels out of the scaler are valid. This is signalled by transitions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

The third stage is the output of the limiter when the x_output_size is still set to match the active array size. Because the scaler has reduced the amount of valid pixel data without reducing the row time, the limiter attempts to pad the row with $(n/2)$ additional pixels. If this has the effect of extending LV across the whole of the horizontal blanking time, the MT9T013 will cease to generate output frames.

A correct configuration is shown in Figure 15. This figure shows the x_output_size reduced to match the output size of the scaler. In this configuration, the output of the limiter does not extend LV.

This figure also shows the effect of the output FIFO, which forms the final stage in the SMIA data path. The output FIFO merges the intermittent pixel data back into a contiguous stream. Although not shown in this example, the output FIFO is also capable of operating with an output clock that is at a different frequency from its input clock.

Figure 2: Timing of SMIA Data Path



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

The pixel array readout rate is set by line_length_pck * frame_length_lines. With the default register values, one frame time takes 3,328 * 1,648 = 5,484,544 pixel periods. This value includes vertical and horizontal blanking times so that the full-size image 2,048 x 1,538 (1,536 lines of pixel data, 2 lines of embedded information) forms a subset of these pixels.

When the internal clock is running at 64 MHz, this frame time corresponds to 5,484,544/64e6 = 85.696ms, giving rise to a frame rate of 11.67 fps.

Each pixel is 10 bits, by default. As a result, the serial data rate is required to transmit faster than the pixel rate. However, the SMIA CCP2 class 2 specification has a maximum of 650 Mb/s, which cannot be exceeded.

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

The only way to get a full image out is to reduce the pixel clock rate until it is appropriate for the maximum CCP class 0 data rate. This requires the pixel rate to be reduced to 20.8 MHz. This has the side effect of reducing the frame rate. Repeating the calculation above, at 20.8 MHz internal clock, this corresponds to 5,484,544/20.8e6 = 0.2637 second, giving rise to a frame rate of 3.79 fps.

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

Output Data Timing

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

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

A key constraint when configuring the clock for the output FIFO is that the frame rate out of the FIFO must exactly match the frame rate into the FIFO. When the scaler is disabled, this constraint can be met by imposing the rule that the row time on the CCP2 data stream must be greater than or equal to the row time at the pixel array. The row time on the CCP2 data stream is calculated from the x_output_size and the ccp_data_format (8 or 10 bits per pixel), and must include the time taken in the CCP2 data stream for start of frame/row, end of row/frame and checksum symbols.

Caution **If this constraint is not met, the FIFO will 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
- ccp2_signalling_mode
- ccp_data_format
- scale_m
- vt_pix_clk_div
- vt_sys_clk_div
- pre_pll_clk_div
- pll_multiplier
- op_pix_clk_div
- op_sys_clk_div
- Profile 0/1, 2 selection

Programming Restrictions when Using Global Reset

Interactions between the registers that control the global reset imposes some programming restrictions on the way in which they are used; these are discussed in "Global Reset" on page 30.

Control of the Signal Interface

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

Serial Register Interface

The serial register interface uses the following signals:

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

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

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

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

This interface is described in detail in "Two-Wire Serial Register Interface" on page 76.

Default Power-up State

The MT9T013 provides interfaces for pixel data through the CCP2 high speed serial interface described by the SMIA specification, and a parallel data interface.

At power up and after a hard or soft reset, the reset state of the MT9T013 is to enable the SMIA CCP2 high speed serial interface.

Serial Pixel Data Interface

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

- DATA_P
- DATA_N
- CLK_P
- CLK_N

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

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

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

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

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

Parallel Pixel Data Interface

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

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

The parallel pixel data interface is disabled by default at power up and after reset. It can be enabled by programming R0x301A. Table 10 shows the recommended settings.

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

Output Enable Control

When the parallel pixel data interface is enabled, its signals can be switched asynchronously between the driven and High-Z under pin or register control, shown in Table 9. Selection of a pin to use for the OE_N function is described in "General Purpose Inputs" on page 11.

Table 3: Output Enable Control

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

Configuration of the Pixel Data Interface

Fields in R0x301A are used to configure the operation of the pixel data interface. The supported combinations are shown in Table 10.

Table 4: Configuration of the Pixel Data Interface

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

System States

The system states of the MT9T013 are represented as a state diagram in Figure 16 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 11.

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 11 on page 10.

Figure 3: MT9T013 System States

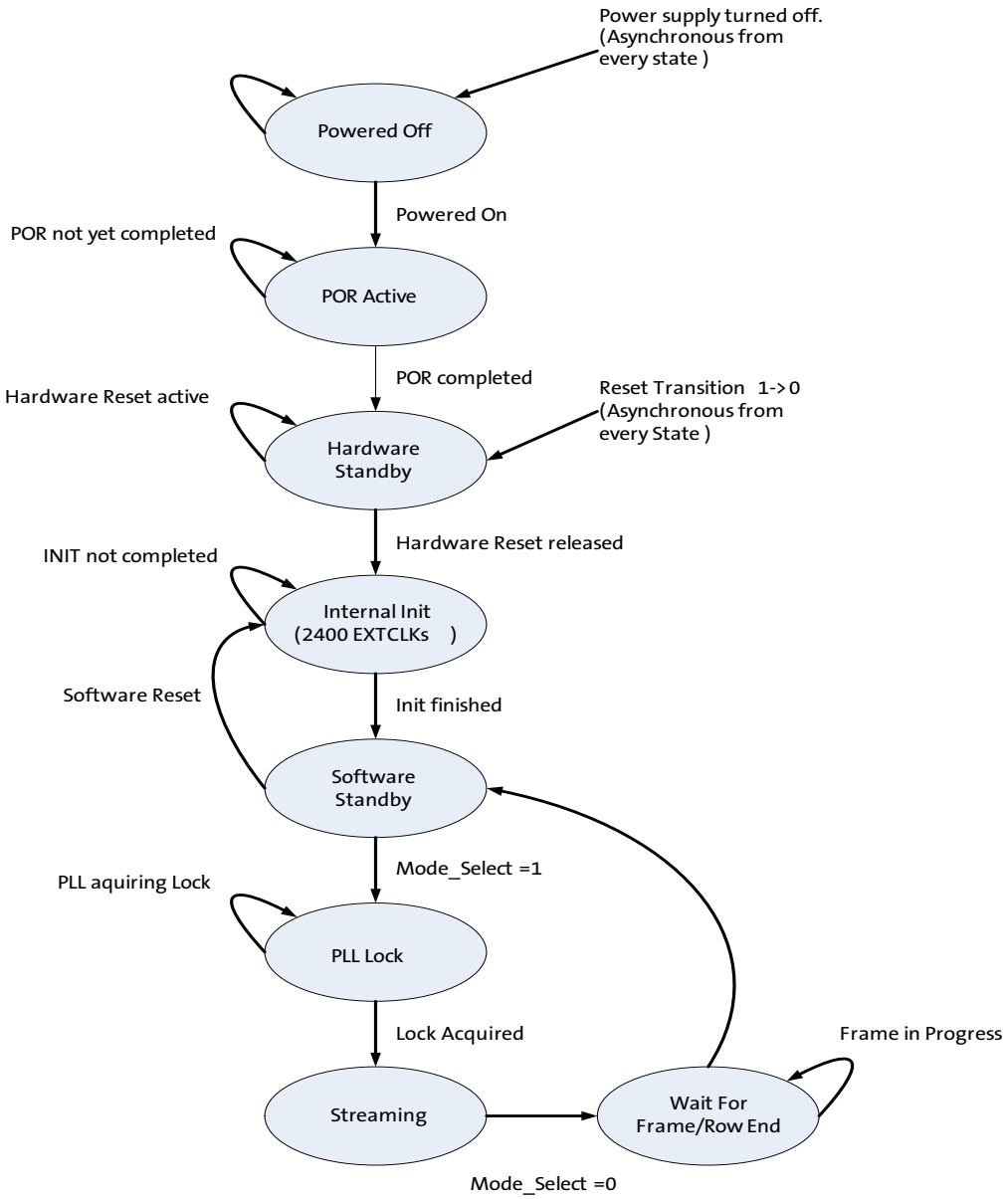


Table 5: PLL in System States

State	EXTCLKs	PLL
Powered off		
POR Active		
Hardware standby		
Internal Initialization	2400	VCO powered down
Software standby		
PLL Lock	1ms * EXTCLK_FREQ_MHz	VCO powering up and locking, PLL output bypassed
Streaming		VCO running, PLL output active
Wait for frame end		

Power-On Reset Sequence

When power is applied to the MT9T013, it enters a low-power hardware standby state. Exit from this state is controlled by the later of two events:

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

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

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

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

While RESET_BAR is asserted (or the internal power-on reset circuit is active) the MT9T013 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; it then enters a low-power software standby state. While the initialization sequence is in progress, the MT9T013 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 (0x26 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.

Soft Reset Sequence

The MT9T013 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 12 shows the state of the signal interface during hardware standby (RESET_BAR asserted) and the default state during software standby (after exit from hardware standby and before any registers within the sensor have been changed from their default power-up values).

Table 6: Signal State During Reset

Pad Name	Pad Type	Hardware Standby	Software Standby
EXTCLK	Input	Enabled. Must be driven to a valid logic level.	
RESET_BAR (XSHUTDOWN)	Input	Enabled. Must be driven to a valid logic level.	
LV	Output	High-Z. Can be left disconnected/floating.	
FV	Output		
DOUT[9:0]	Output		
PIXCLK	Output		
SCL	Input	Enabled. Must be pulled up or driven to a valid logic level.	
SDA	I/O	Enabled as an input. Must be pulled up or driven to a valid logic level.	
FLASH	Output	High-Z.	Logic 0.
SHUTTER	Output	High-Z.	Logic 0.
DATA_P	Output	High-Z.	
DATA_N	Output		
CLK_P	Output		
CLK_N	Output		
GPI[3:0]	Input	Powered down. Can be left disconnected/floating.	
TEST	Input	Enabled. Must be driven to a logic 0.	

General Purpose Inputs

The MT9T013 provides four general purpose inputs. After reset, the input pads associated with these signals are powered down by default, allowing the pads to be left disconnected/floating.

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

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

- Output enable (see “Output Enable Control” on page 8)
- Trigger (see the sections below)
- Standby functions (see the following sections)
- SADDR selection (see “Serial Register Interface” on page 7)

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

Streaming/Standby Control

The MT9T013 can be switched between its soft standby and streaming states under pin or register control, shown in Table 13. Selection of a pin to use for the STANDBY function is described in "General Purpose Inputs" on page 11. The state diagram for transitions between soft standby and streaming states is shown in Figure 16 on page 9.

Table 7: 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, shown in Table 14. Selection of a pin to use for the TRIGGER function is described in "General Purpose Inputs" on page 11.

Table 8: 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 MT9T013 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 clock scheme and profile 1, 2 are supported. The clocking scheme can be selected by either setting register 0x306E–F[7] to 0 for profile 0 or to 1 for profile 1, 2.

Figure 4: MT9T013 SMIA Profile 1, 2 Clocking Structure

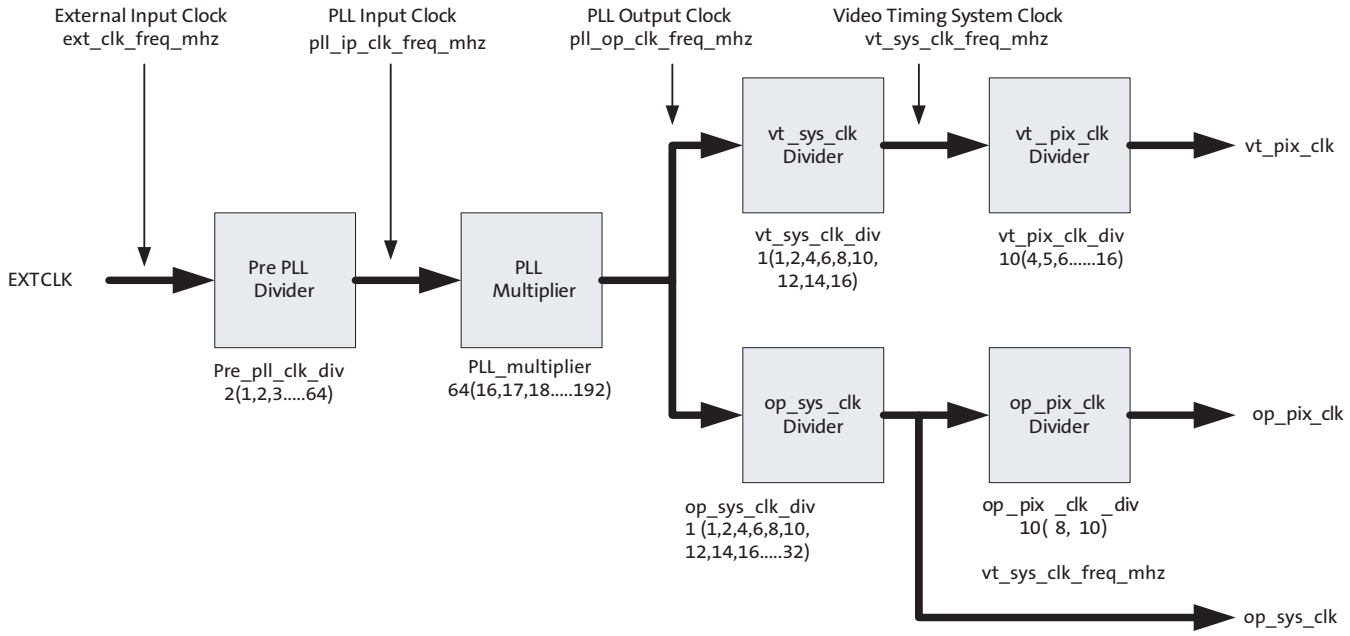


Figure 17 shows the different clocks and the names of the registers that contain or are used to control their values. The figure shows the default setting for each divider/multiplier control register and the range of legal values for each divider/multiplier control register.

The parameter limit register space contains registers that declare the minimum and maximum allowable values for:

- The frequency allowable on each clock
- The divisors that are used to control each clock

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

- The minimum/maximum frequency limits for the associated clock must be met.
- The minimum/maximum value for the divider/multiplier must be met.
- The value of pll_multiplier should be a multiple of 2.
- The op_pix_clk must never run faster than the vt_pix_clk to ensure that the CCP2 output data stream is contiguous.
- Given the maximum programmed line length, the minimum blanking time, the maximum image width, the available PLL divisor/multiplier values, and the requirement that the output line time (including the necessary blanking) must be output in a

time equal to or less than the time defined by line_length_pck, the valid combinations of the clock divisors.

PLL input clock frequency range, after the pre-PLL divider stage, is 6.0–48.0 MHz.

The usage of the output clocks is shown below:

- vt_pix_clk is used by the sensor core to control the timing of the pixel array. The sensor core produces one 10-bit pixel each vt_pix_clk period. The line length (line_length_pck) and fine integration time (fine_integration_time) are controlled in increments of the vt_pix_clk period.
- op_pix_clk is used to load parallel pixel data from the output FIFO (see Figure 44 on page 44) to the CCP2 serializer. The output FIFO generates one pixel each op_pix_clk period. The pixel is either 8-bit or 10-bit depending upon the output data format, controlled by R0x0112–3 (ccp_data_format).
- op_sys_clk is used to generate the serial data stream on the CCP2 output. The relationship between this clock frequency and the op_pix_clk frequency is dependent upon the output data format.

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

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

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

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

Figure 5: MT9T013 SMIA Profile 0 Clocking Structure

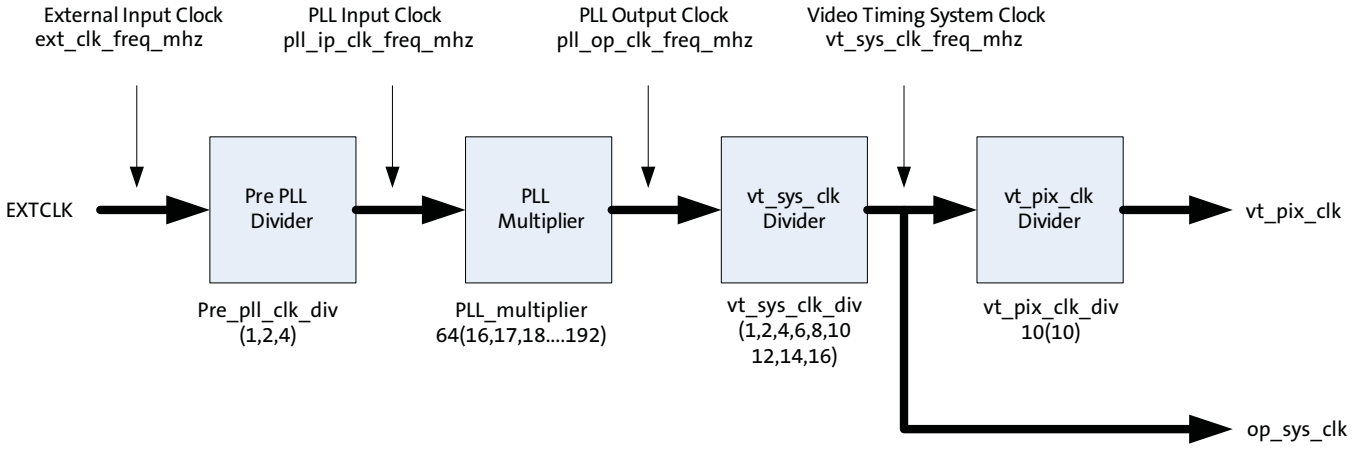


Figure 18 shows the different clocks and the names of the registers that contain or are used to control their values. Also shown is the default setting for each divider/multipler control register and the range of legal values for each divider/multiplier control register.

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

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

- The minimum/maximum frequency limits for the associated clock must be met.
- The minimum/maximum value for the divider/multiplier must be met.
- Given the maximum programmed line length, the minimum blanking time, the maximum image width, the available PLL divisor/multiplier values, and the requirement that the output line time (including the necessary blanking) must be output in a time equal to or less than the time defined by line_length_pck

PLL input clock frequency range is 2.0–64.0 MHz.

The usage of the output clocks is shown below:

- vt_pix_clk is used by the sensor core to control the timing of the pixel array. The sensor core produces one 10-bit pixel each vt_pix_clk period. The line length (line_length_pck) and fine integration time (fine_integration_time) are controlled in increments of the vt_pix_clk period.
- vt_sys_clk is also used to generate the serial data stream on the CCP2 output.

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

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

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

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

Programming the PLL Divisors

The PLL divisors should be programmed while the MT9T013 is in the software standby state. After programming the divisors, 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 MT9T013 is in the streaming state is undefined.

Influence of ccp_data_format

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

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

Influence of ccp2_signalling_mode

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

When data/clock signalling is selected, the pll_multiplier supports both odd and even values.

When data/strobe signalling is selected, the pll_multiplier only supports even values; the least significant bit of the programmed value is ignored and treated as "0."

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

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

Clock Control

The MT9T013 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 MT9T013 enters a low-power state, almost all of the internal clocks are stopped. The only exception is that a small amount of logic is clocked so that the two-wire serial interface continues to respond to read and write requests.

Features

Shading Correction (SC)

Lenses tend to produce images whose brightness is significantly attenuated near the edges. There are also other factors causing fixed pattern signal gradients in images captured by image sensors. The cumulative result of all these factors is known as image shading. The MT9T013 has an embedded shading correction module that can be programmed to counter the shading effects on each individual Red, GreenB, GreenR, and Blue color signal.

The Correction Function

Color-dependent solutions are calibrated using the sensor, lens system and an image of an evenly illuminated, featureless grey calibration field. From the resulting image the color correction functions can be derived.

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

$$Pcorrected(row, col) = Psensor(row,col) * f(row,col) \quad \text{(EQ 7)}$$

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

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

One-Time Programmable (OTP) Memory

The MT9T013 has a two-byte OTP memory that can be utilized during module manufacturing to store specific information about the module. This feature provides system integrators and module manufacturers the ability to label and distinguish various module types based on lens, IR-cut filter, or other properties.

During the programming process, a dedicated pin for high voltage needs to be provided to perform the programming operation. This voltage (VPP) would need to be 8.5V ± 5 percent. Completion of the programming process will be communicated by a register through the two-wire serial interface.

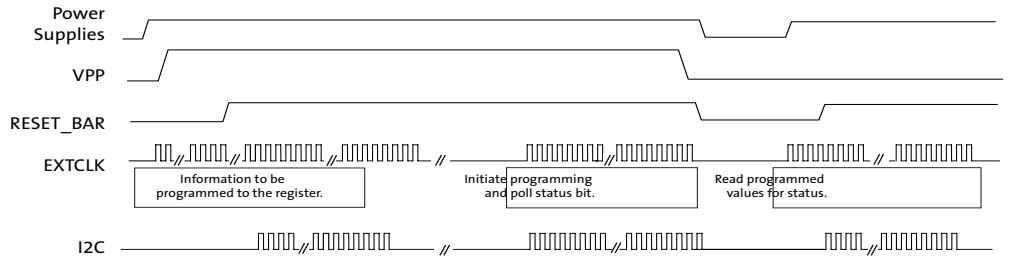
Because this programming pin needs to sustain a higher voltage than other input/output pins, having a dedicated high voltage pin (VPP) minimizes the design risk. If the module manufacturing process can probe the sensor at the die or PCB level (that is, supply all the power rails, clocks, two-wire serial interface signals), then this dedicated high voltage pin does not need to be assigned to the module connector pin-out. However, if VPP needs to be bonded out as a pin on the module, the trace for VPP needs to carry a maximum of 200µA. This pin should be left floating, but may optionally be connect to analog ground.

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 the programming 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.

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 steps of operational during the manufacturing process:

1. Apply power to all the power rails of the sensor.
2. Supply 8.5V to VPP pin.
3. Provide 12 MHz EXTCLK clock input.
4. Perform the proper reset sequence to the sensor.
5. Place the sensor in soft standby (sensor default state upon power-up).
6. Set burn duration to 16ms by programming R0x3054 = 0x50B6.
7. Write information to be programmed to a register R0x304C–D on the sensor through the two-wire serial interface.
8. Set the control bit to register R0x304A–B to initiate programming.
9. Poll register R0x304A[1] for the completion of programming.
10. Remove high voltage.
11. Set EXTCLK to normal operating frequency (24 MHz).
12. Perform the proper reset sequence to the sensor.
13. Set control bit to register 0x304A–B to initiate reading process.
14. Read data from register 0x304E–F.

Figure 6: Sequence for Programming the Device



Notes:
1. Timing specifications are NOT included in this sequence. Timing information will be present after characterization.
2. Two-wire serial interface bus is pulled HIGH when NOT in active state. This is NOT presented in the timing diagram.
3. Sensor powers up to software standby and should remain in software standby for the duration of the programming of the OTP memory.

Image Acquisition Modes

The MT9T013 supports two image acquisition modes:

- Electronic rolling shutter (ERS) mode.
  This is the normal mode of operation. When the MT9T013 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 MT9T013 switches cleanly from the old integration time to the new while only generating frames with uniform integration.
- Global reset mode.
  This mode can be used to acquire a single image at the current resolution. In this mode, the pixel integration time is controlled by an external electromechanical shutter, and the MT9T013 provides control signals to interface to that shutter. The operation of this mode is described in detail in "Global Reset" on page 30.

The use of an external electromechanical shutter increases cost and may reduce ruggedness of the end application. The motivation for the use of an external electromechanical shutter is that it eliminates the visual artifacts associated with ERS operation. Visual artifacts arise in ERS operation, particularly at low frame rates, because an ERS image effectively integrates each row of the pixel array at a different point in time so that its operation is somewhat akin to that of a photo-finish machine at a race track.

Window Control

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

Pixel Border

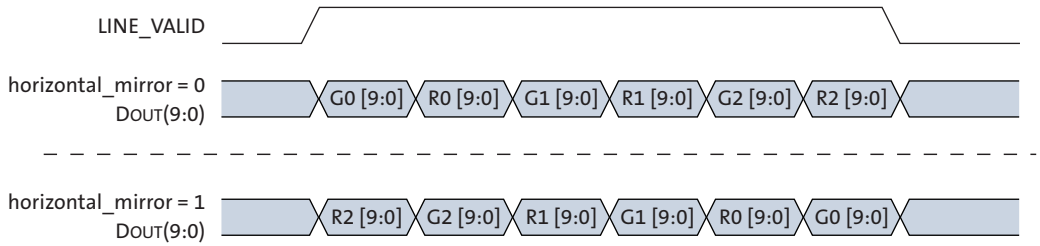
The default settings of the sensor provide a 2048H x 1536V image. A border of up to 8 pixels (4 in binning) on each edge can be enabled by reprogramming the x_addr_start, y_addr_start, x_addr_end, y_addr_end registers, x_output_size, and y_output_size registers accordingly.

Readout Modes

Horizontal Mirror

When the horizontal_mirror bit is set in the image_orientation register, the order of pixel readout within a row is reversed, so that readout starts from x_addr_end and ends at x_addr_start. Figure 20 on page 20 shows a sequence of 6 pixels being read out with horizontal_mirror = 0 and horizontal_mirror = 1. Changing horizontal_mirror causes the Bayer order of the output image to change; the new Bayer order is reflected in the value of the pixel_order register.

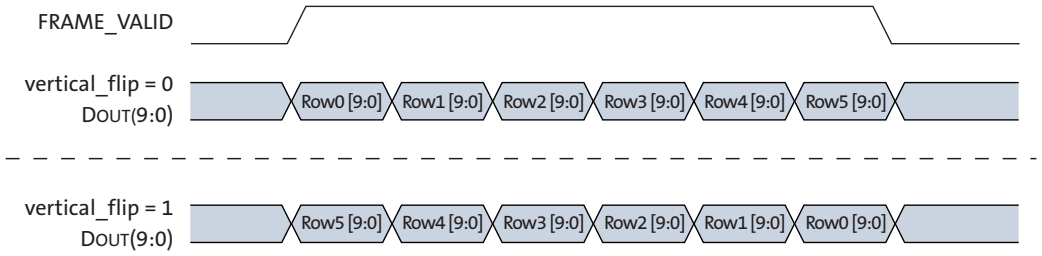
Figure 7: Effect of horizontal_mirror on Readout Order



Vertical Flip

When the vertical_flip bit is set in the image_orientation register, the order in which pixel rows are read out is reversed, so that row readout starts from y_addr_end and ends at y_addr_start. Figure 21 shows a sequence of 6 rows being read out with vertical_flip = 0 and vertical_flip = 1. Changing vertical_flip causes the Bayer order of the output image to change; the new Bayer order is reflected in the value of the pixel_order register.

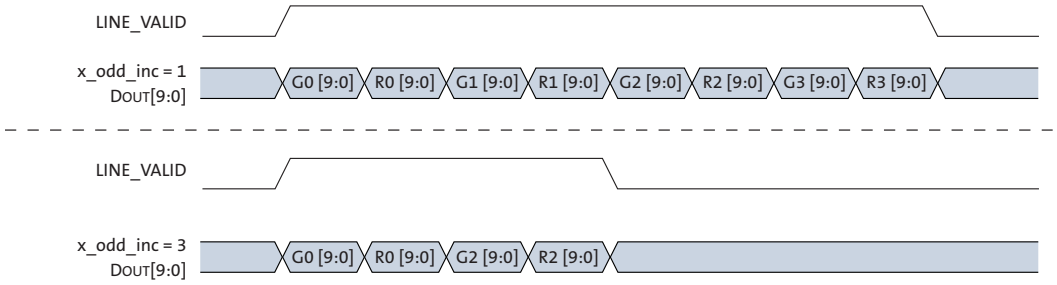
Figure 8: Effect of vertical_flip on Readout Order



Subsampling

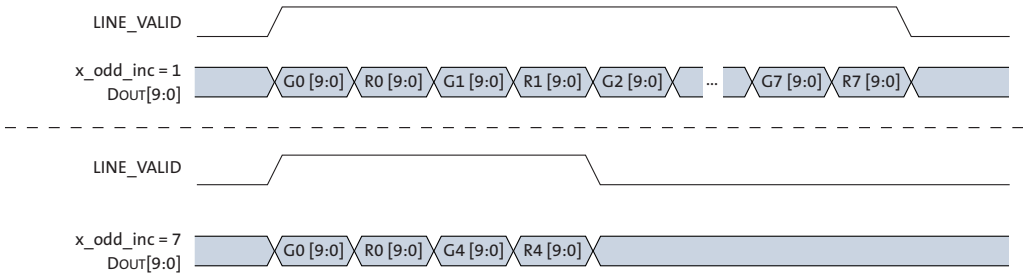
The MT9T013 supports subsampling. Subsampling reduces the amount of data processed by the analog signal chain in the MT9T013 thereby allowing the frame rate to be increased. Subsampling is enabled by setting x_odd_inc and/or y_odd_inc. Values of 1, 3, and 7 can be supported. Setting both of these variables 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 MT9T013. Figure 22 shows a sequence of 8 columns being read out with x_odd_inc = 3 and y_odd_inc = 1.

Figure 9: Effect of x_odd_inc = 3 on Readout Sequence



A 1/16 reduction in resolution is achieved by setting both x_odd_inc and y_odd_inc to 7. This is equivalent to 4 x 4 skipping readout mode provided by the MT9T013. Figure 23 shows a sequence of 16 columns being read out with x_odd_inc = 7 and y_odd_inc = 1.

Figure 10: Effect of x_odd_inc = 7 on Readout Sequence



The effect of the different subsampling settings on the pixel array readout is shown in Figure 24 through Figure 26 on page 22.

Figure 11: Pixel Readout (No Subsampling)

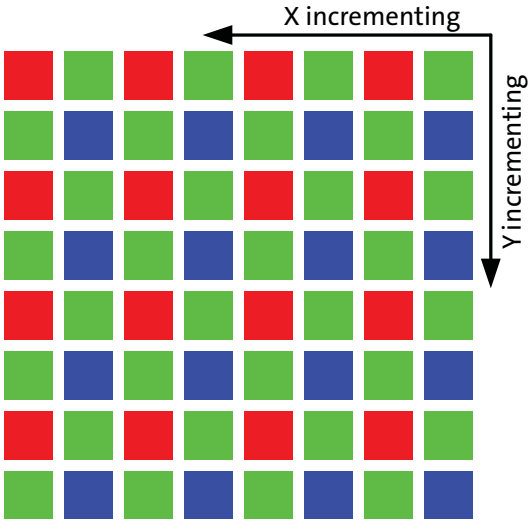


Figure 12: Pixel Readout (x_odd_inc = 3, y_odd_inc = 3)

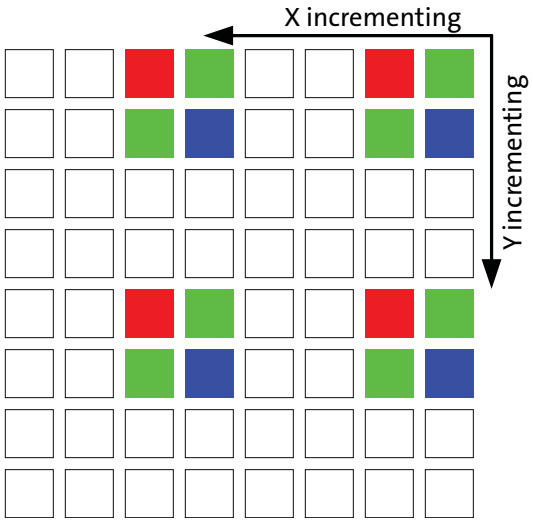
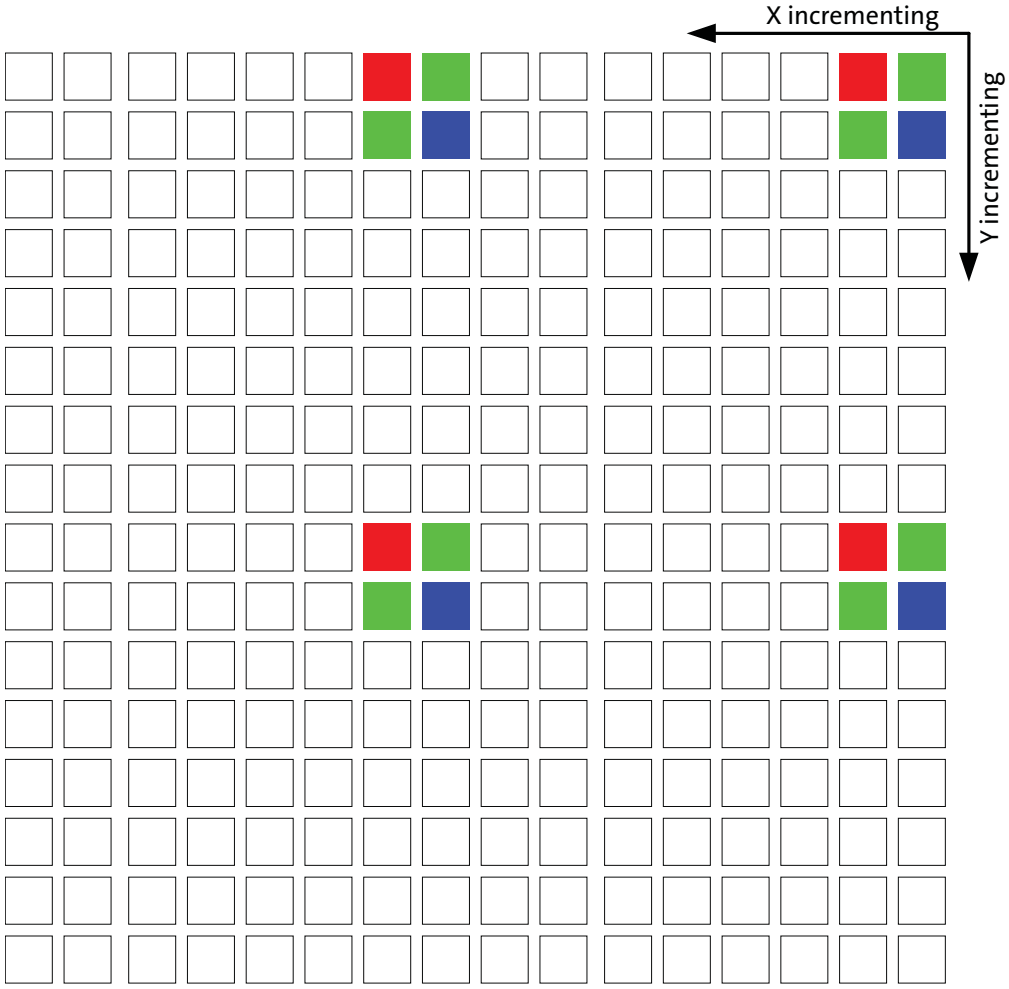


Figure 13: Pixel Readout (x_odd_inc = 7, y_odd_inc = 7)



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, it is recommended that line_length_pck be kept constant between the two modes. This allows the same integration times to be used in each mode.

When subsampling is enabled, it may be necessary to adjust the x_addr_end, 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 the following rules:

- x_addr_start must be a multiple of 2 for example 0, 4, 6, 8, and x_addr_start = 2 is not supported

When 2 x 2 skipping mode is enabled,

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

When 4 x 4 skipping mode is enabled,

- (x_addr_end - x_addr_start + x_odd_inc) should be a multiple of 8
- (y_addr_end - y_addr_start + y_odd_inc) should be a multiple of 8

The number of columns/rows read out with subsampling can be found from Equation 8 and Equation 9:

When 2 x 2 skipping mode is enabled

$$columns/rows = (addr\_end - addr\_start + odd\_inc) / 2 \quad \textit{(EQ 8)}$$

When 4 x 4 skipping mode is enabled

$$columns/rows = (addr\_end - addr\_start + odd\_inc) / 4 \quad \text{(EQ 9)}$$

Example:

To achieve 2048 x 1536 full resolution without skipping, the recommended register settings are:

```
[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, 2055                // X_ADDR_END
REG=0x034A, 1543                // Y_ADDR_END
REG=0x034C, 2048                // X_OUTPUT_SIZE
REG=0x034E, 1536                // Y_OUTPUT_SIZE
REG=0x0104, 0                   // GROUPED_PARAMETER_HOLD
```

To achieve a 1024 x 786 resolution with 2 x 2 skipping, the recommended register settings are:

```
[2x2 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, 2053                // X_ADDR_END
REG=0x034A, 1541                // Y_ADDR_END
REG=0x034C, 1024                // X_OUTPUT_SIZE
REG=0x034E, 768                 // Y_OUTPUT_SIZE
REG=0x0104, 0                   // GROUPED_PARAMETER_HOLD
```

To achieve a 512 x 384 resolution with 4 x 4 skipping, the recommended register settings are:

```
[4x4 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, 2049                // X_ADDR_END
REG=0x034A, 1537                // Y_ADDR_END
REG=0x034C, 512                 // X_OUTPUT_SIZE
REG=0x034E, 384                 // Y_OUTPUT_SIZE
REG=0x0104, 0                   // GROUPED_PARAMETER_HOLD
```

Table 15 shows the row address sequencing for normal and subsampled readout. The same sequencing applies to column addresses for subsampled readout. There are two possible subsampling sequences for the rows (because the subsampling sequence only read half of the rows) depending upon the alignment of the start address. The row address sequencing during binning is also shown.

Table 9: Row Address Sequencing

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

Table 9: Row Address Sequencing (continued)

odd_inc = 1	odd_inc = 3				odd_inc = 7			
Normal	Normal		Binned		Normal		Binned	
start = 0	start = 0	start = 2	start = 0	start = 2	start = 0	start = 2	start = 0	start = 2
12	12		12, 14					
13	13		13, 15					
14		14		14, 16				
15		15		15, 17				

Binning

The MT9T013 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 for subsampling, x_addr_end and y_addr_end may require adjustment when binning is enabled as described in "Programming Restrictions when Subsampling" on page 23. Note that 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 27 and in Figure 28 on page 26.

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 29 on page 26.

Figure 14: Pixel Readout (x_odd_inc = 3, y_odd_inc = 1, x_bin = 1)

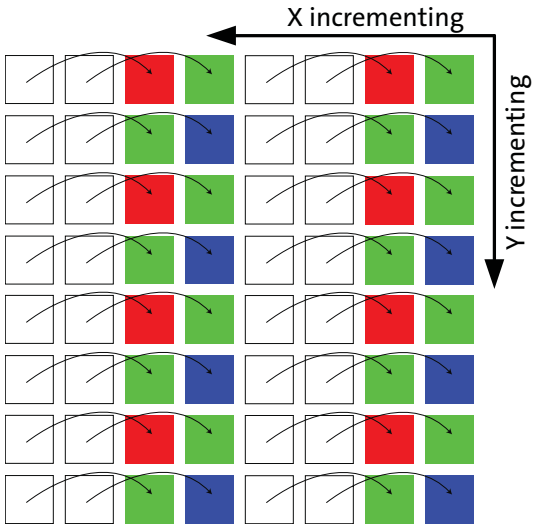


Figure 15: Pixel Readout (x_odd_inc = 3, y_odd_inc = 3, xy_bin = 1)

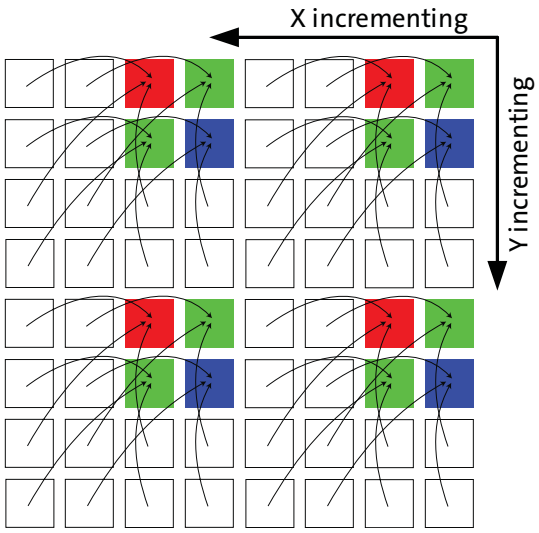
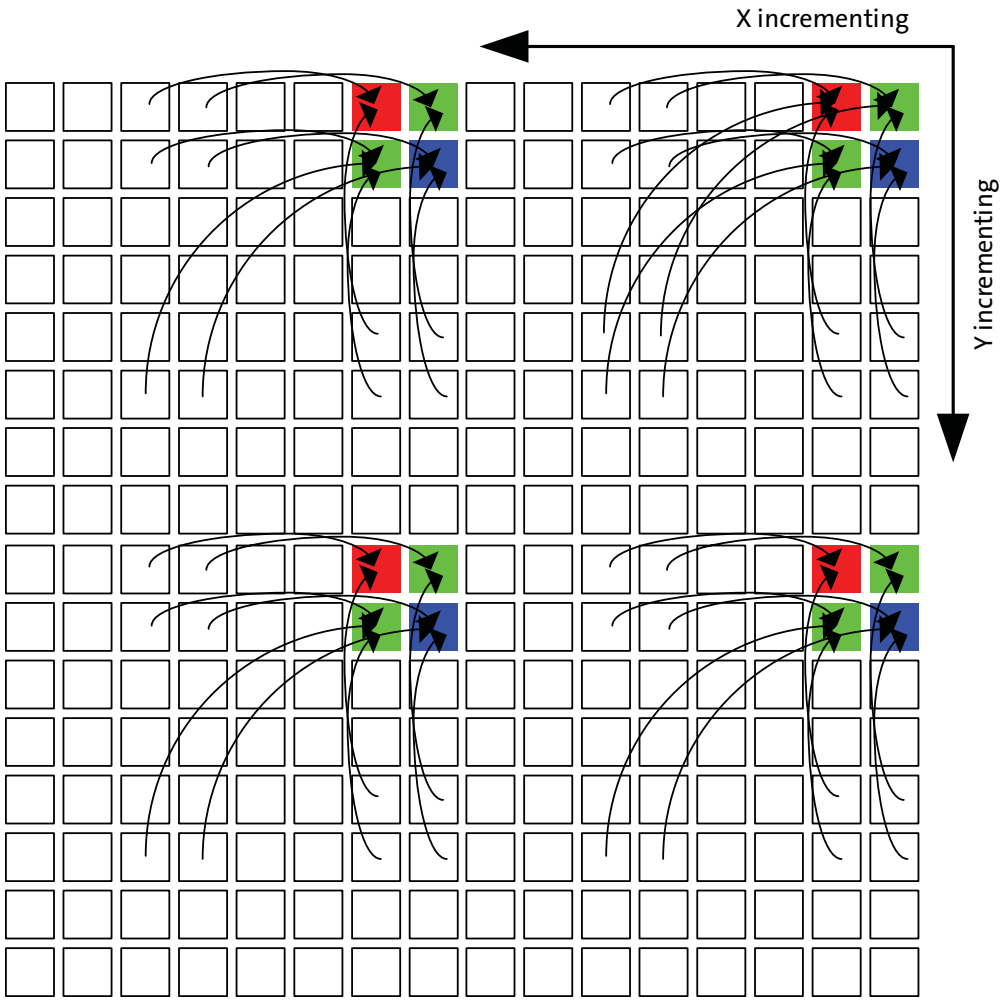


Figure 16: Pixel Readout (x_odd_inc = 7, y_odd_inc = 7, xy_bin = 1)



Programming Restrictions when Binning

Binning requires different sequencing of the pixel array and imposes different timing limits on the operation of the sensor. In particular, xy-binning requires two read operations from the pixel array for each line of output data, which has the effect of increasing the minimum line blanking time.

As a result, when xy-binning is enabled, some of the programming limits declared in the parameter limit registers are no longer valid. In addition, the default values for some of the manufacturer specific registers need to be reprogrammed. The recommended settings are shown in Table 16. None of these adjustments are required for x-binning.

Table 10: Register Adjustments Required for Binning Mode

Register	Type	Default (Normal Readout)	Recommended Setting During Binning	Notes
min_line_blanking_pck	read-only	0x037A	0x0638	Read-only register for control software; does not affect operation of sensor.
min_line_length_pck	read-only	0x0488	0x0900	Read-only register for control software; does not affect operation of sensor.
fine_integration_time_min	read-only	0x02DD	0x05B5	Read-only register for control software; does not affect operation of sensor.
fine_integration_time_max_margin	read-only	0x01AB	0x034B	Read-only register for control software; does not affect operation of sensor.
fine_correction	read/write	0x0128	0x0260	Affects operation of sensor.
fine_integration_time	read/write	0x02DD	0x05B5	Normal default is minimum value.

Because binning also requires subsampling to be enabled, the same restrictions apply to the setting of x_addr_end and y_addr_end.

A given row n will always be binned with row $n + 2$ for 2X subsampling mode and row $n + 4$ for 4X subsampling mode. Therefore, there are two candidate rows that a row can be binned with, depending upon the alignment of y_addr_start.

For a given column n, there is only one other column, n_bin, that it can be binned with. Because the x_addr_start is restricted to multiples of 2, a column n will also always be binned with column $n + 2$ for 2X subsampling mode and column $n + 4$ for 4X subsampling mode.

Frame Rate Control

The formulas for calculating the frame rate of the MT9T013 are shown below:

$$line\_length\_pck = \left(\frac{x\_addr\_end - x\_addr\_start + x\_odd\_inc}{subsampling\ factor} + min\_line\_blanking\_pck\right) \quad \text{(EQ 10)}$$

$$frame\_length\_lines = \left(\frac{y\_addr\_end - y\_addr\_start + y\_odd\_inc}{subsampling\ factor} + min\_frame\_blanking\_lines\right) \quad \text{(EQ 11)}$$

$$frame\ rate\ [FPS] = \frac{(vt\_pixel\_clock\_mhz * 1\times10^{6})}{(line\_length\_pck * frame\_length\_lines)} \quad \text{(EQ 12)}$$

Integration Time

The integration (exposure) time of the MT9T013 is controlled by the fine_integration_time and coarse_integration_time registers.

The limits for the fine integration time are defined by:

$$fine\_integration\_time\_min <= fine\_integration\_time <= (line\_length\_pck\text{-}fine\_integration\_time\_max\_margin) \quad \text{(EQ 13)}$$

The limits for the coarse integration time are defined by:

$$coarse\_integration\_time\_min <= coarse\_integration\_time \quad \text{(EQ 14)}$$

If coarse_integration_time > (frame_lenth_lines-coarse_integration_time_max_margin), then the frame rate will be reduced.

The actual integration time is given by:

$$integration\_time\ [sec] = \frac{((coarse\_integration\_time * line\_length\_pck) + fine\_integration\_time)}{(vt\_pix\_clk\_freq\_mhz * 1 * 10^{6})} \quad \text{(EQ 15)}$$

With a vt_pix_clk of 72 MHz, the maximum integration time that can be achieved without reducing the frame rate is given by:

$$Maximum\ integration\ time\ [sec] = \frac{(((0x670\text{-}1) * 0xD00) + 0x2DD)}{(72\ MHz * 1 * 10^{6})} = 76.13ms \quad \text{(EQ 16)}$$

Setting an integration time that is greater than the frame time increases the frame time beyond frame_length_lines to make longer exposure times available.

Flash Control

The MT9T013 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 30, 31, and Figure 32 on page 30. The flash and flash_count registers allow the timing of the flash to be changed. The flash can be programmed to fire only once, delayed by a few frames when asserted, and (for xenon flash) the flash duration can be programmed.

Enabling the LED flash will cause one bad frame, where several of the rows only have the flash on for part of their integration time. This can be avoided by forcing a restart (write reset_register[1] = 1) immediately after enabling the flash; the first bad frame will then be masked out as shown in Figure 32 on page 30. Read-only bit flash[14] is set during frames that are correctly integrated; the state of this bit is shown in the Figure 30 on page 29, Figure 31 on page 29, and Figure 32 on page 30.

Figure 17: Xenon Flash Enabled

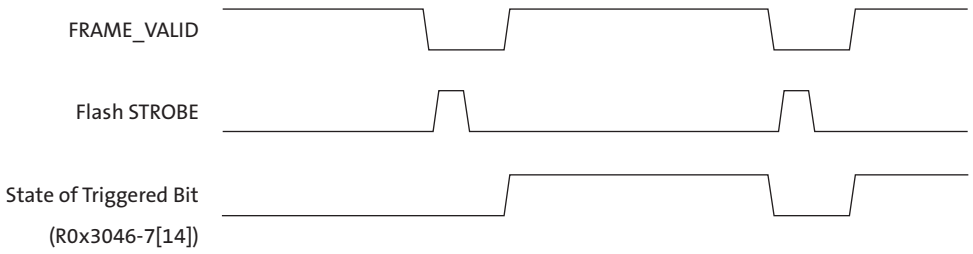
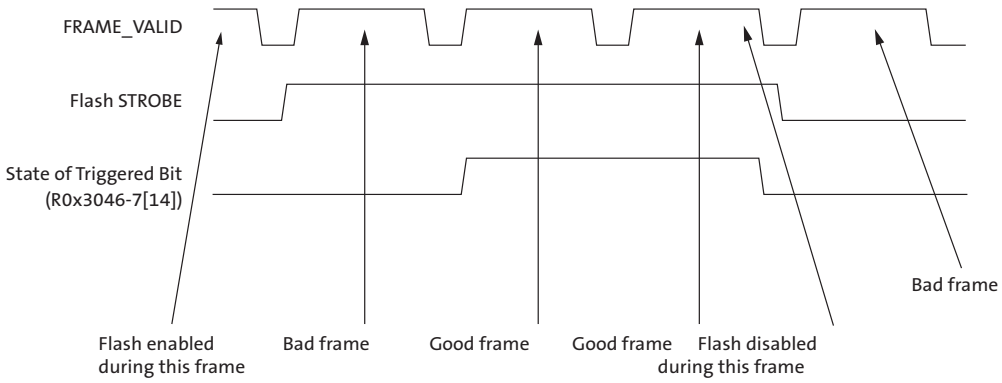
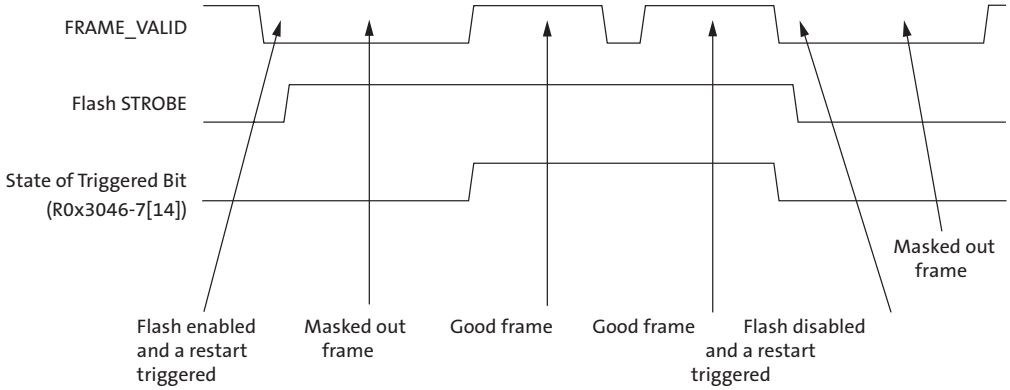


Figure 18: LED Flash Enabled



Notes: 1. Integration time = number of rows in a frame.
Bad frames will be masked when mask corrupted frames option is enabled.
An option to invert the flash output signal is also available.

Figure 19: LED Flash Enabled Following Forced Restart



Global Reset

Global reset mode allows the integration time of the MT9T013 to be controlled by an external electromechanical shutter. Global reset mode is generally used in conjunction with ERS mode. The ERS mode is used to provide viewfinder information, the sensor is switched into global reset mode to capture a single frame, and the sensor is then returned to ERS mode to restore viewfinder operation.

Global reset mode is designed for use in conjunction with the parallel pixel data interface. The SMIA specification only provides for operation in ERS mode. The MT9T013 does support the use of global reset mode in conjunction with the SMIA data path, but there are additional restrictions on its use.

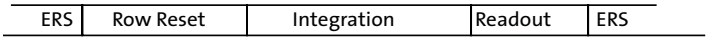
Overview of Global Reset Sequence

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

1. By default, the sensor operates in ERS mode and the SHUTTER output signal is LOW. The electromechanical shutter must be open to allow light to fall on the pixel array. Integration time is controlled by the coarse_integration_time and fine_integration_time registers.
2. A global reset sequence is triggered.
3. All of the rows of the pixel array are placed in reset.
4. All of the rows of the pixel array are taken out of reset simultaneously. All rows start to integrate incident light. The electromechanical shutter may be open or closed at this time.
5. If the electromechanical shutter has been closed, it is opened.
6. After the desired integration time (controlled internally or externally to the MT9T013), the electromechanical shutter is closed.
7. A single output frame is generated by the sensor with the usual LV, FV, PIXCLK, and DOUT timing. As soon as the output frame has completed (FV de-asserts), the electromechanical shutter may be opened again.
8. The sensor automatically resumes operation in ERS mode.

This sequence is shown in Figure 33. The following sections expand on this figure to show how the timing of this sequence is controlled.

Figure 20: Overview of Global Reset Sequence



Entering and Leaving the Global Reset Sequence

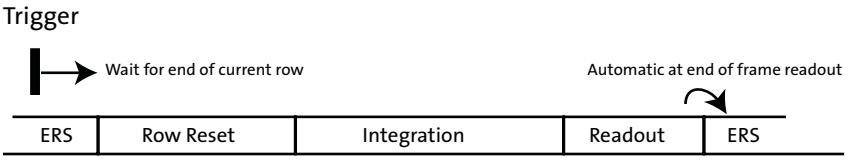
A global reset sequence can be triggered either by a register write to global_seq_trigger[0] (global trigger, to transition this bit from a 0 to a 1) or by a rising edge on a suitably-configured GPI input (see "Trigger Control" on page 12).

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

The global reset sequence completes with a frame readout. At the end of this readout phase, the sensor automatically resumes operation in ERS mode. The first frame integrated with ERS will be generated after a delay of approximately ((min_frame_blanking + coarse_integration_time) * line_length_pck). This sequence is shown in Figure 34.

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 21: Entering and Leaving a Global Reset Sequence



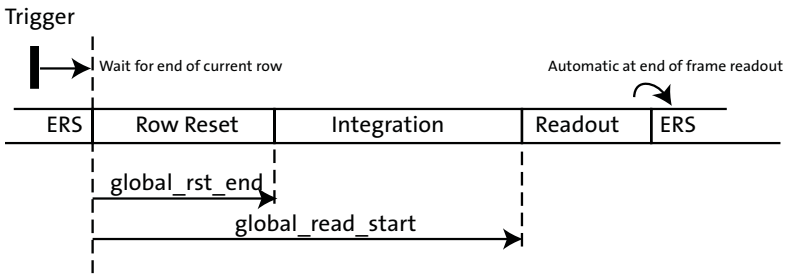
Programmable Settings

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

The recommended setting for global_rst_end is 0xA0. 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.

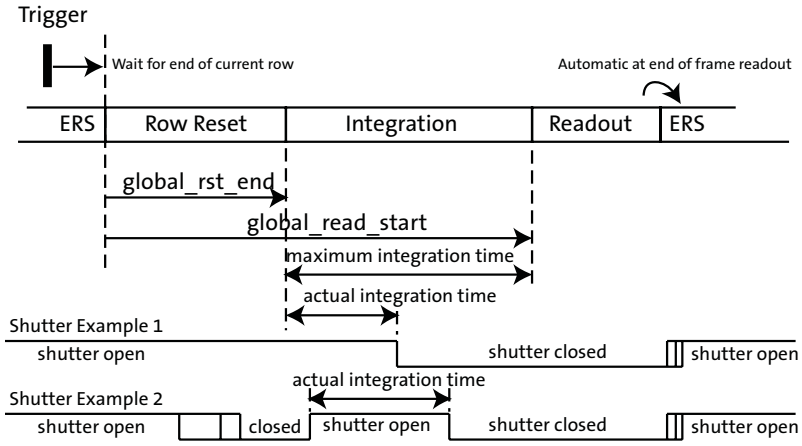
Figure 22: Controlling the Reset and Integration Phases of the Global Reset Sequence



Control of the Electromechanical Shutter

Figure 36 on page 32 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 23: Control of the Electromechanical Shutter



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

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

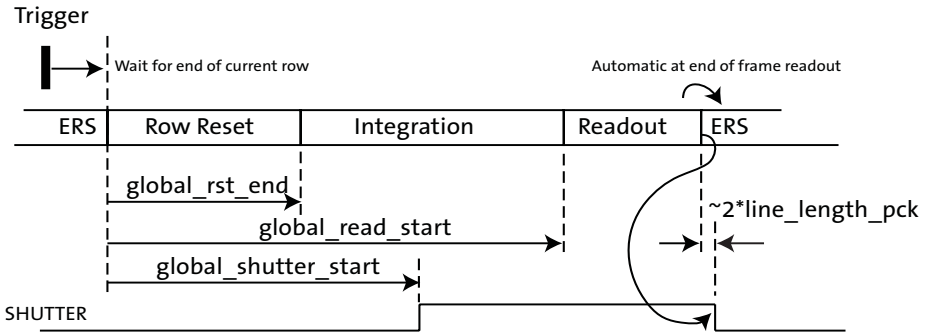
After FV de-asserts to signal the completion of the readout phase, there is a time delay of approximately (10 * line_length_pck) before the sensor starts to integrate light-sensitive rows for the next ERS frame. It is essential that the shutter be opened at some point in this time window; otherwise,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

The MT9T013 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 37. SHUTTER is de-asserted by default. The point at which it asserts is controlled by the programming of global_shutter_start. At the end of the global reset readout phase, SHUTTER de-asserts approximately (2 * line_length_pck) after the negation of FV.

The following programming restriction must be met for correct operation:

- global_read_start > global_shutter_start.

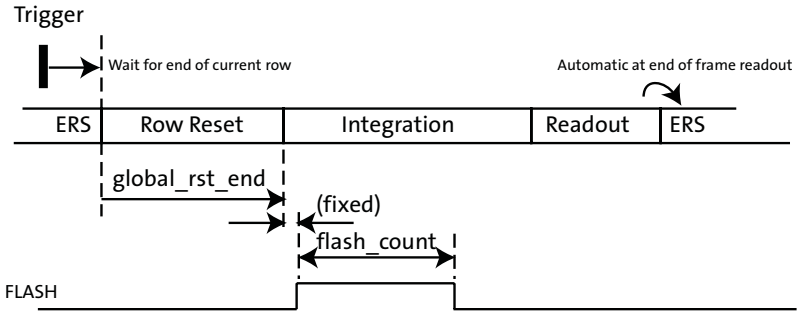
Figure 24: Controlling the SHUTTER Output



Using FLASH with Global Reset

If global_seq_trigger[2] = 1 (global flash enabled) when a global reset sequence is triggered, the FLASH output signal will be pulsed during the integration phase of the global reset sequence. The FLASH output will assert a fixed number of cycles after the start of the integration phase and will remain asserted for a time that is controlled by the value of the flash_count register. This is shown in Figure 38.

Figure 25: Using FLASH with Global Reset



External Control of Integration Time

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

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

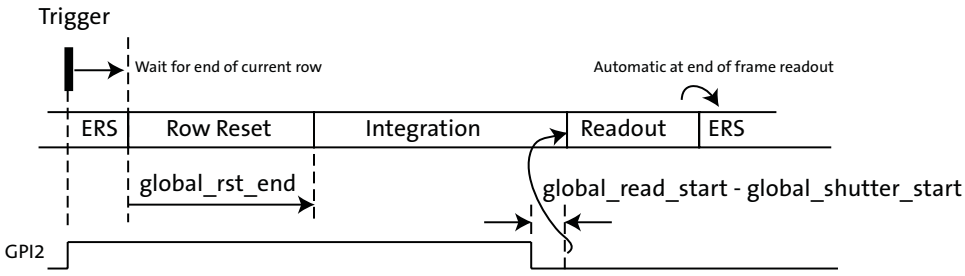
When the trigger is de-asserted to end integration, the integration phase is extended by a further time given by (global_read_start - global_shutter_start). Usually this means that global_read_start should be set to (global_shutter_start + 1).

The operation of this mode is shown in Figure 39 on page 34. The figure shows the global reset sequence being triggered by the GPI2 input, but it could be triggered by any of the GPI inputs or by the setting and subsequence clearing of the global_seq_trigger[0] under software control.

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

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

Figure 26: Global Reset Bulb



Retriggering the Global Reset Sequence

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

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

Using Global Reset with SMIA Data Path

When a global reset sequence is triggered, it usually results in the frame in progress being truncated (at the end of the current output line). The SMIA data path limiter function (see Figure 44 on page 44) attempts to extend (pad) all frames to the programmed value of y_output_size. If this padding is still in progress when the global reset readout phase starts, the SMIA data path will not detect the start of the frame correctly. Therefore, to use global reset with the SMIA data path, this timing scenario must be avoided. One possible way of doing this would be to synchronize (under software control) the assertion of trigger to an end-of-frame marker on the CCP serial data stream.

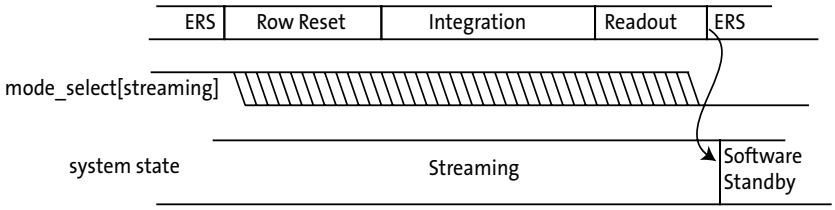
At the end of the readout phase of the global reset sequence, the sensor automatically resumes operation in ERS mode.

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

Global Reset and Soft Standby

If the mode_select[stream] bit is cleared while a global reset sequence is in progress, the MT9T013 will remain in streaming state until the global reset sequence (including frame readout) has completed, shown in Figure 40.

Figure 27: Entering Soft Standby During a Global Reset Sequence



Analog Gain

The MT9T013 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 MT9T013 provides per-color gain control. However, the MT9T013 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 SMIA specification. In the MT9T013 this register has no side-effects on the operation of the gain; per-color and global gain control can be used interchangeably regardless of the state of the gain_mode register.

SMIA Gain Model

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

- analogue_gain_code_global
- analogue_gain_code_greenR
- analogue_gain_code_red
- analogue_gain_code_blue
- analogue_gain_code_greenB

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

$$gain = \frac{analogue\_gain\_m0 \times analogue\_gain\_code}{analogue\_gain\_c1} = \frac{analogue\_gain\_code\_<color>}{8} \quad \text{(EQ 17)}$$

Aptina Gain Model

The Aptina gain model uses the following 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:

$$gain = (<color>\_gain[8] + 1) \times (<color>\_gain[7] + 1) \times \frac{<color>\_gain[6:0]}{32} \quad \text{(EQ 18)}$$

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 17.

Table 11: Recommended Gain Settings

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

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> registers, data read from the associated analogue_gain_code_<color> register is undefined. The reason for this is that many of the gain codes available in the Aptina 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 MT9T013 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 18.

Table 12: 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
256	Walking 1s (10-bit)
257	Walking 1s (8-bit)

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 MT9T013 registers must be set appropriately to control the frame rate and output timing. This includes:

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

Effect of Data Path Processing on Test Patterns

Test patterns 1–3 and 256 are introduced early in the pixel data path. As a result, they are affected by pixel processing that occurs within the data path. This includes:

- Noise cancellation
- Black pedestal adjustment
- Lens/color shading correction

These effects can be eliminated by the following register settings:

- R0x3044–5[10] = 0
- R0x30CA–B[0] = 1
- R0x30D4–5[15] = 0
- R0x31E0–1[0] = 0
- R0x3180–1[15] = 0
- R0x301A–B[3] = 0 (enable writes to data pedestal)
- R0x301E–F = 0x0000 (set data pedestal to “0”)
- R0x3780[15] = 0 (turn off lens/color shading correction)

Solid Color Test Pattern

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

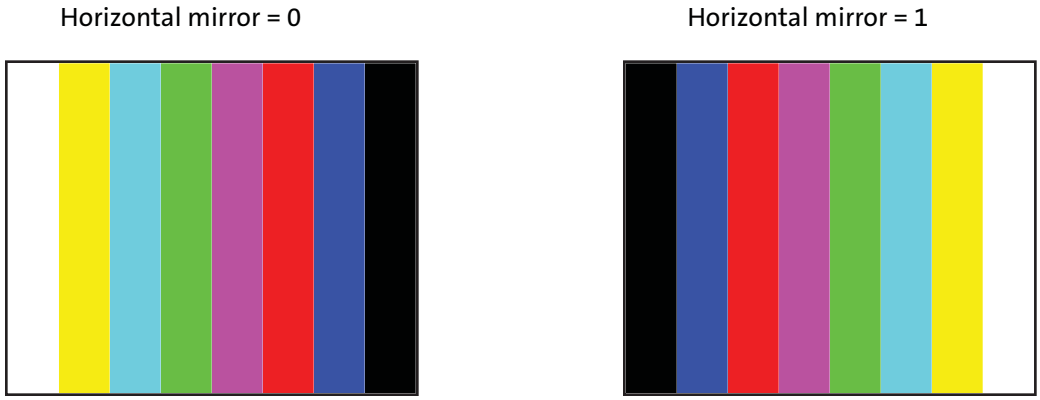
100 Percent Color Bars Test Pattern

In this test pattern, shown in Figure 41, all pixel data is replaced by a Bayer version of an 8-color, color-bar chart (white, yellow, cyan, green, magenta, red, blue, and black). Each bar is 256 pixels wide and occupies the full height of the output image. Each color component of each bar is set to either 0 (fully off) or 0x3FF (fully on for 10-bit data). The pattern repeats after 8 * 256 = 2048 pixels. The image size is set by x_addr_start, x_addr_end, y_addr_start, y_addr_end and may be affected by the setting of x_output_size, y_output_size. The color-bar pattern starts at the column identified by x_addr_start. The number of colors that are visible in the output is dependent upon x_addr_end - x_addr_start and the setting of x_output_size. The width of each color-bar is fixed at 256 pixels.

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

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

Figure 28: 100 Percent Color Bars Test Pattern



Fade-to-Gray Color Bars Test Pattern

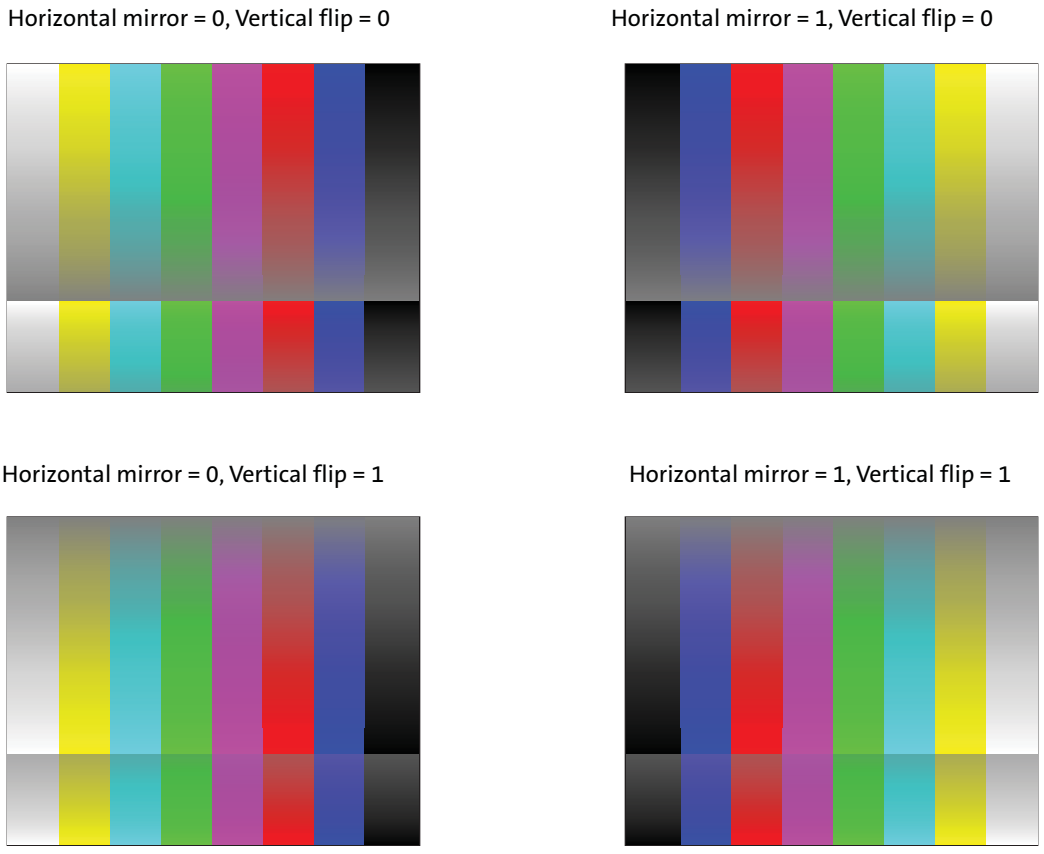
In this test pattern, shown in Figure 42 on page 40, all pixel data is replaced by a Bayer version of an 8-color, color-bar chart (white, yellow, cyan, green, magenta, red, blue, and black). Each bar is 256 pixels wide and occupies 1024 rows of the output image. Each color bar fades vertically from full intensity at the top of the image to 50 percent intensity (mid-gray) on the 1,024th row. Each color bar is divided into a left and a right half, in which the left half fades smoothly and the right half fades in quantized steps every 8 pixels for a given color. Due to the Bayer pattern of the colors this means that the level changes every 16 rows. The pattern repeats horizontally after 8 * 256 = 2,048 pixels and vertically after 1024 rows (Using 10-bit data, the fade-to-gray pattern goes from 100 to 50 percent or from 0 to 50 percent for each color component, so only half of the 2^{10} states of the 10-bit data are used. However, to get all of the gray levels, each state must be held for

two rows, hence the vertical size of 2^{10} / 2 * 2 = 1024). The image size is set by x_addr_start, x_addr_end, y_addr_start, and y_addr_end and may be affected by the setting of x_output_size and y_output_size. The color-bar pattern starts at the column identified by x_addr_start. The number of colors that are visible in the output is dependent upon x_addr_end - x_addr_start and the setting of x_output_size. The width of each color-bar is fixed at 256 pixels.

The effect of setting horizontal_mirror or vertical_flip in conjunction with this test pattern is that the order in which the colors are generated is reversed. The black bar appears at the left 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 29: Fade-to-Gray Color Bars Test Pattern



PN9 Link Integrity Pattern

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

When this test pattern is enabled:

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

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

Walking 1s Test Pattern

The main purpose of the walking 1s test pattern is to detect stuck-at bits at the parallel interface, DOUT[9:0]. During active data period, no more than one logic HIGH would appear at the parallel interface at any given time. Each value in the pattern would appear for two consecutive PIXCLKs. The resulting pattern would have the sequence of:

RAW10:
0x000, 0x000, 0x001, 0x001, 0x002, 0x002, 0x004, 0x004, 0x008, 0x008, 0x010, 0x010, …, 0x3FF, 0x3FF

RAW8:
0x00, 0x00, 0x01, 0x01, 0x02, 0x02, 0x04, 0x04, 0x08, 0x08, 0x10, 0x10, …, 0xFF, 0xFF

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 1s modes are enabled by different test pattern codes.

Test Cursors

The MT9T013 supports one horizontal and one vertical cursor, allowing a “cross hair” to be superimposed on the image or on test patterns 1–3.

The position and width of each cursor is programmable in registers 0x31E8–0x31EE. Only even cursor positions and even cursor widths are supported (this is a consequence of the internal architecture of the pixel array). Each cursor can be inhibited by setting its width to “0.”

The programmed cursor position corresponds to an absolute row or column in the pixel array. For example, setting horizontal_cursor_position to the same value as y_addr_start would result in a horizontal cursor being drawn starting on the first row of the image.

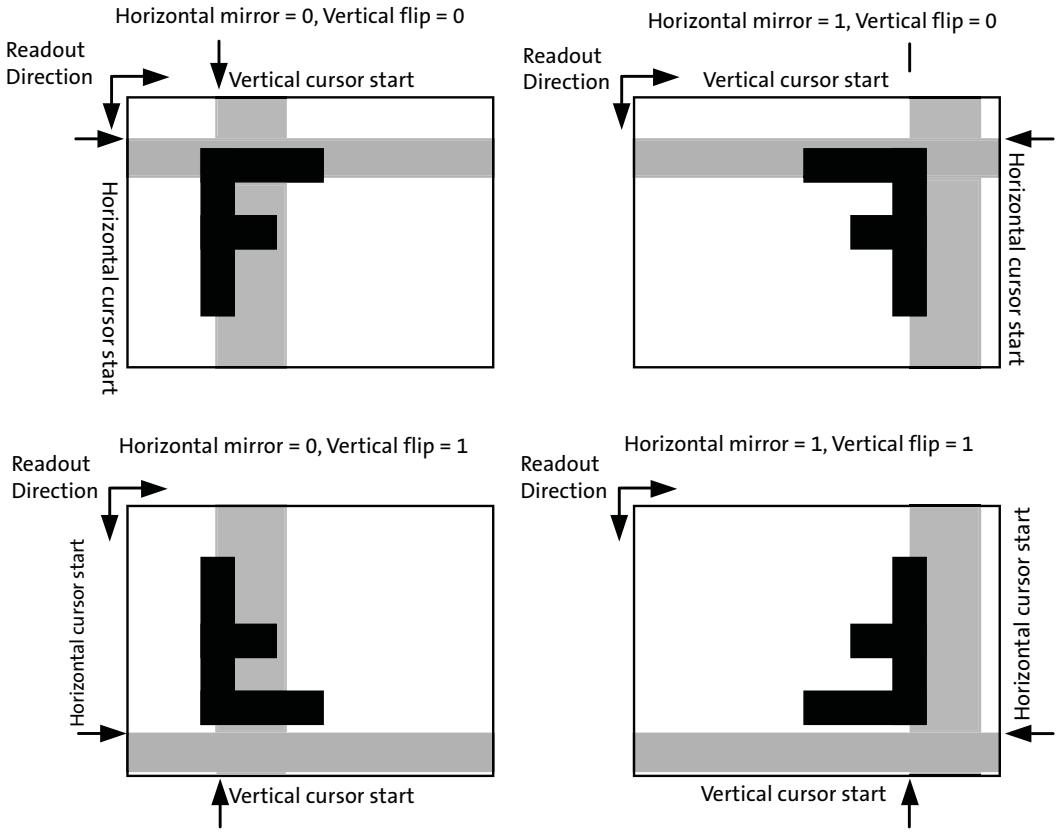
The cursors are opaque (they replace data from the imaged scene or test pattern). The color of each cursor is set by the values of the Bayer components in the test_data_red, test_data_greenR, test_data_blue, and test_data_greenB registers. As a consequence, the cursors are the same color as test pattern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

When vertical_cursor_position = 0x0FFF, the vertical cursor operates in an automatic mode in which its position advances every frame. In this mode the cursor starts at the column associated with x_addr_start=0 and advances by a step-size of 8 columns each frame until it reaches the column associated with x_addr_start = 2040, after which it wraps (256 steps). Note that the active pixel array is smaller than this, so in the last 56 steps the cursor will not be visible. The width and color of the cursor in this automatic mode are controlled in the usual way.

The effect of enabling the test cursors when the image_orientation register is non-zero is not defined by the SMIA specification. The behavior of the MT9T013 is shown in Figure 43 on page 43, where the test cursors are shown as translucent for clarity. In practice, they are opaque (they overlay the imaged scene). The manner in which the test cursors are affected by the value of image_orientation can be understood from the following implementation details:

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

Figure 30: Test Cursor Behavior when image_orientation



Digital Gain

Integer digital gains in the range 1–7 can be programmed. A digital gain of "0" sets all pixel values to "0" (the pixel data will simply represent the value applied by the pedestal block).

Pedestal

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

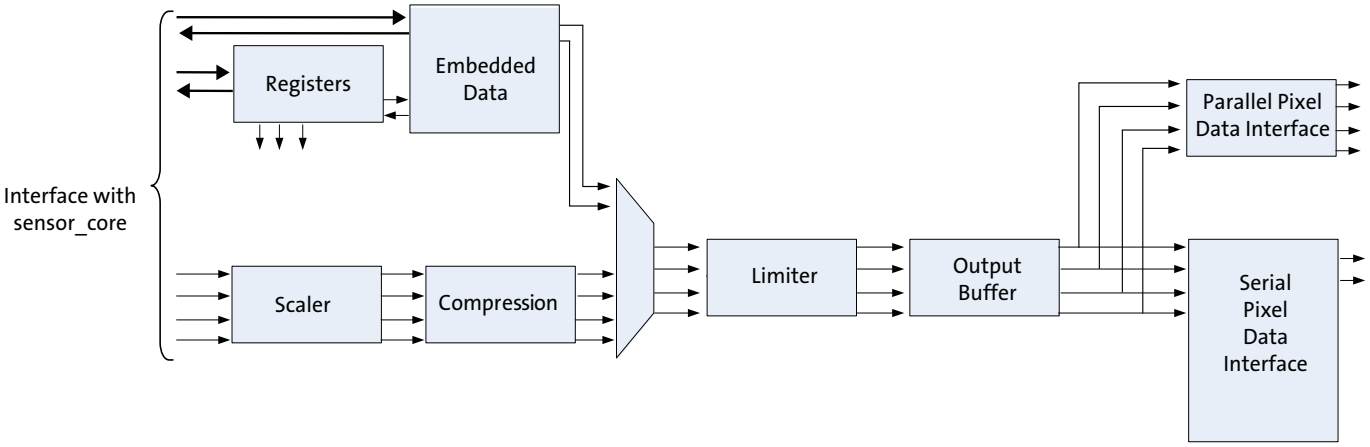
The data_pedestal register is read-only by default but can be made read/write by clearing the lock_reg bit in R0x301A–B.

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 44.

Figure 31: 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 10-bit format places the data byte in bits [9:2] and sets bits [1:0] to a constant value of 01. 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 specifications.

Timing Specifications

Power-Up Sequence

The recommended power-up sequence for the MT9T013 is shown in Figure 45. The available power supplies—VDD_IO, VDD, VDD_CCP, VDD_PLL, VAA, VAA_PIX—can be turned on at the same time or have the separation specified below.

1. Turn on the VDD_IO power supply.
2. After 1–500ms, turn on the VDD and the VDD_CCP power supplies.
3. After 1–500ms, turn on the VDD_PLL and the 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 32: Power-Up Sequence

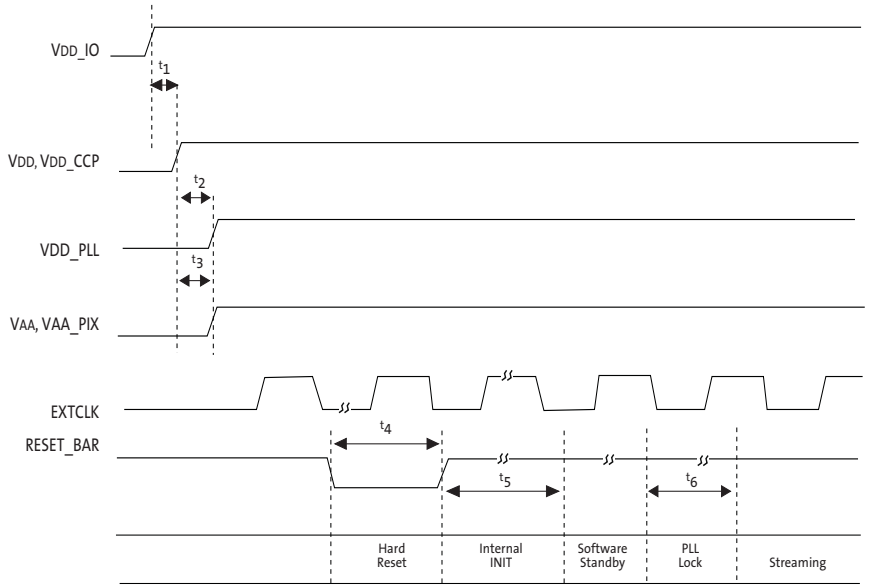


Table 13: Power-Up Sequence

Definition	Symbol	Min	Typ	Max	Unit
VDD_IO to VDD, VDD_CCP time	t_1	0	–	–	ms
VDD, VDD_CCP to VDD_PLL time	t_2	0	–	–	ms
VDD, VDD_CCP 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 MT9T013 is shown in Figure 46. The available power supplies—VDD_IO, VDD, VDD_CCP, 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 the VDD_PLL power supplies.
5. After 1–500ms, turn off the VDD and the VDD_CCP power supplies.
6. After 1–500ms, turn off the VDD_IO power supply.

Figure 33: Power-Down Sequence

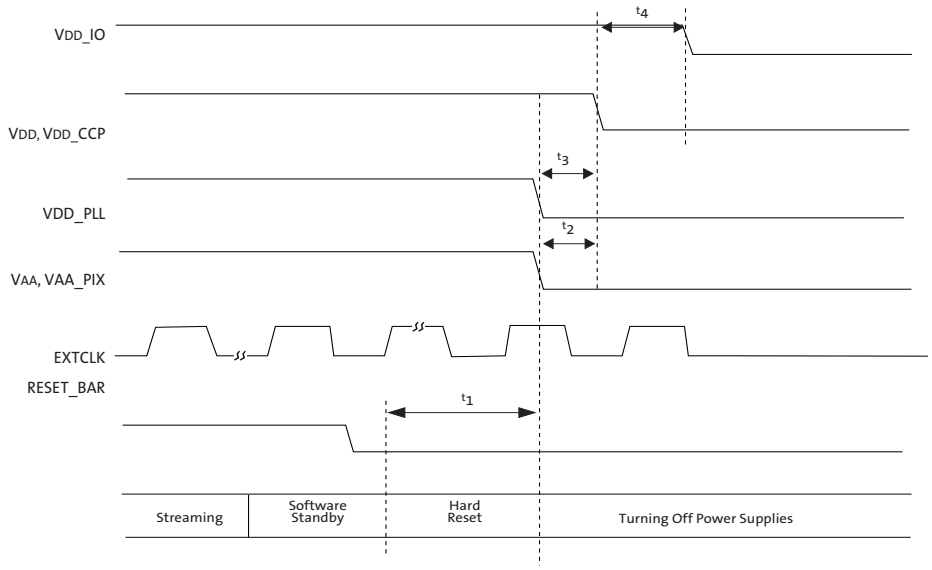


Table 14: Power-Down Sequence

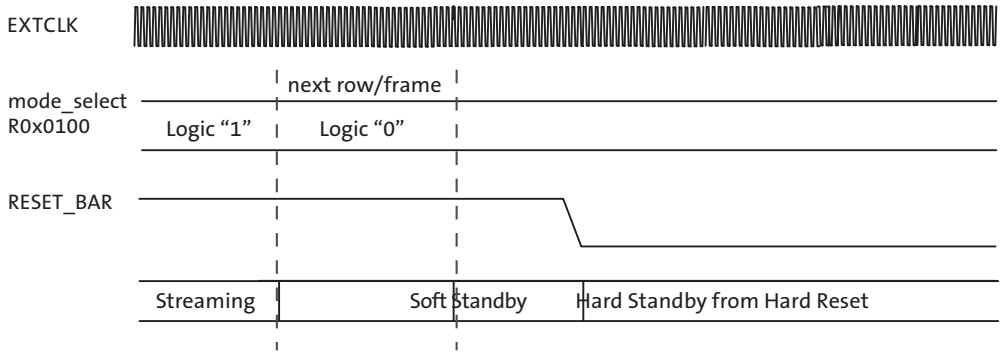
Definition	Symbol	Min	Typ	Max	Unit
Hard reset	t_1	1	–	–	ms
VAA, VAA_PIX to VDD, VDD_CCP time	t_2	0	–	–	ms
VDD_PLL to VDD, VDD_CCP time	t_3	0	–	–	ms
VDD, VDD_CCP to VDD_IO time	t_4	0	–	–	ms

Hard Standby and Hard Reset

The hard standby state is reached by the assertion of the RESET_BAR pad (hard reset). Register values are not retained by this action, and will be returned to their default values once hard reset is completed. The minimum power consumption is achieved by the hard standby state. The details of the sequence is described below and shown in Figure 47.

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 34: Hard Standby and Hard Reset



Soft Standby and Soft Reset

The MT9T013 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 48.

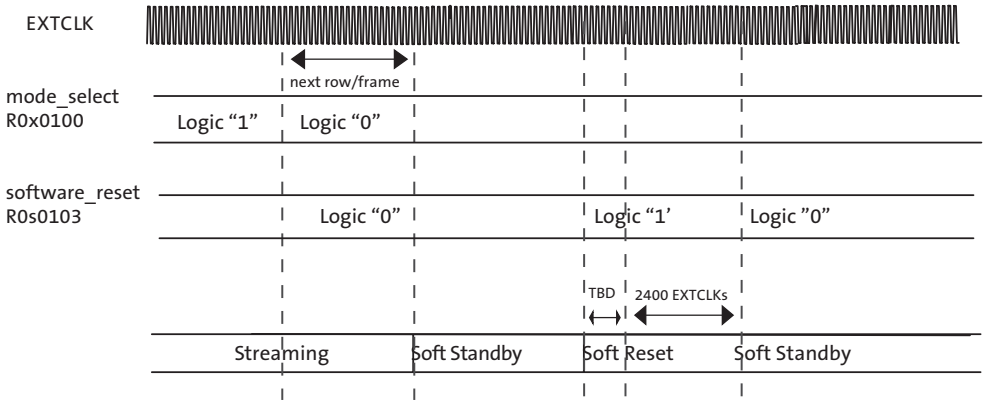
Soft Standby

1. Disable streaming if output is active by setting mode_select = 0 (R0x0100).
2. The soft standby state is reached after the current row or frame, depending on configuration, has ended.

Soft Reset

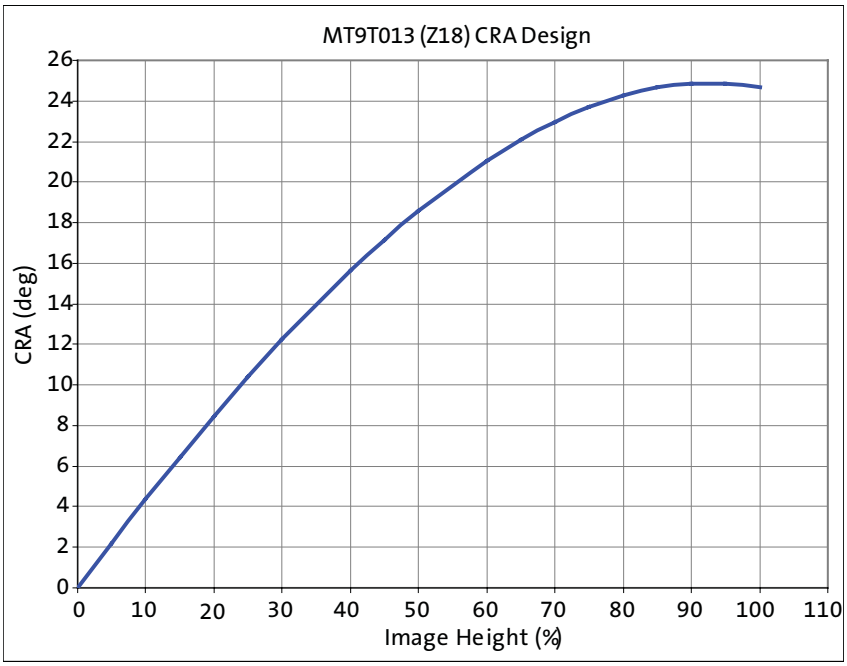
3. Follow the soft standby sequence list above.
4. Set software_reset = 1 (R0x0103) to start the internal initialization sequence.
5. After 2400 EXTCLKs, the internal initialization sequence is completed and the current state returns to soft standby automatically. All registers, including software_reset,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

Figure 35: Soft Standby and Soft Reset



Spectral Characteristics

Figure 1: Chief Ray Angle (CRA) (Z18 Type A)



CRA vs. Image Height Plot, Z18 (Type A)	Image Height (%)	Image Height (mm)	CRA (Deg)
	0	0	0
	5	0.112	2.21
	10	0.224	4.37
	15	0.336	6.46
	20	0.448	8.47
	25	0.560	10.41
	30	0.672	12.25
	35	0.784	13.99
	40	0.896	15.63
	45	1.008	17.16
	50	1.120	18.58
	55	1.232	19.87
	60	1.344	21.04
	65	1.456	22.07
	70	1.568	22.96
	75	1.680	23.70
	80	1.792	24.27
	85	1.904	24.66
	90	2.016	24.87
	95	2.128	24.87
	100	2.240	24.65

Figure 2: Chief Ray Angle (CRA) (Z19 Type B)

CRA vs. Image Height Plot, Z19 (Type B)

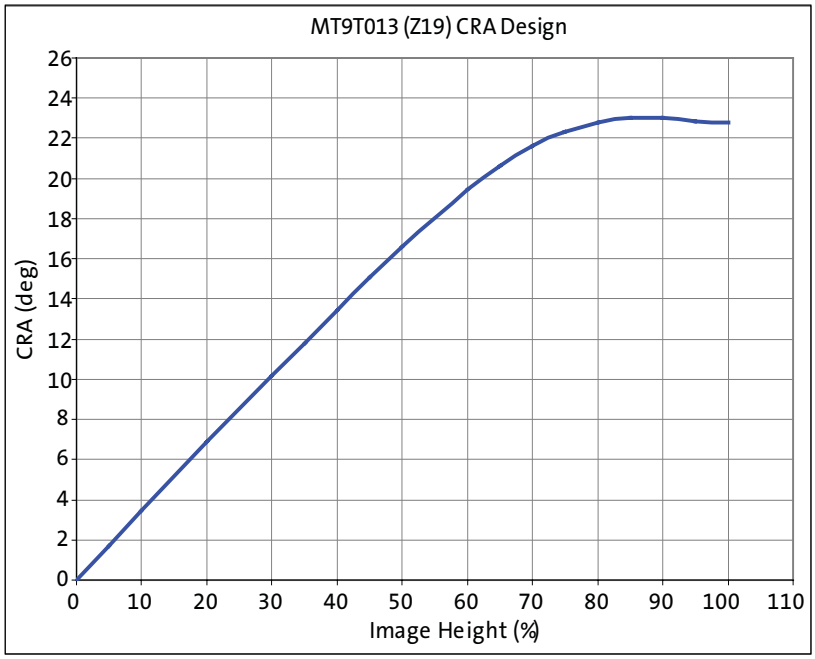
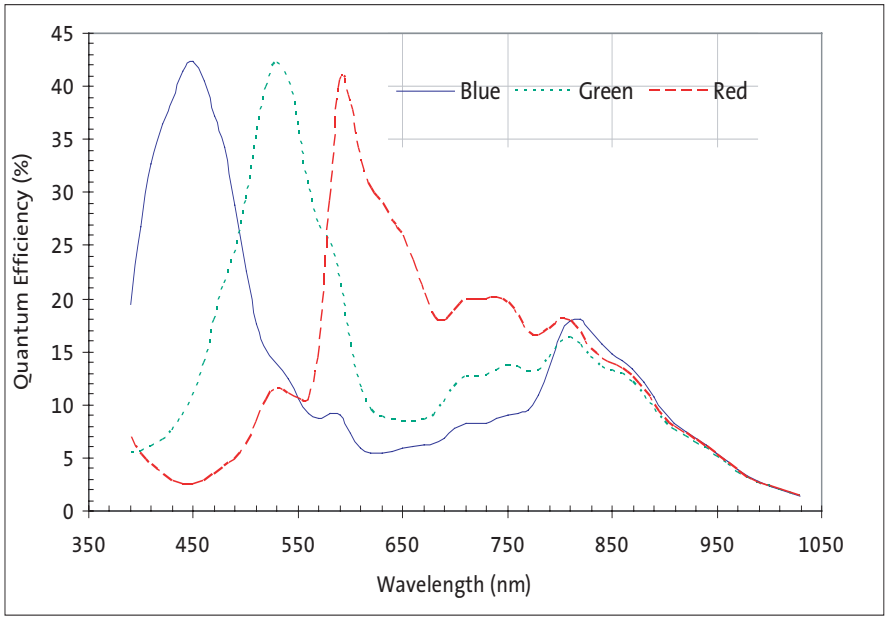


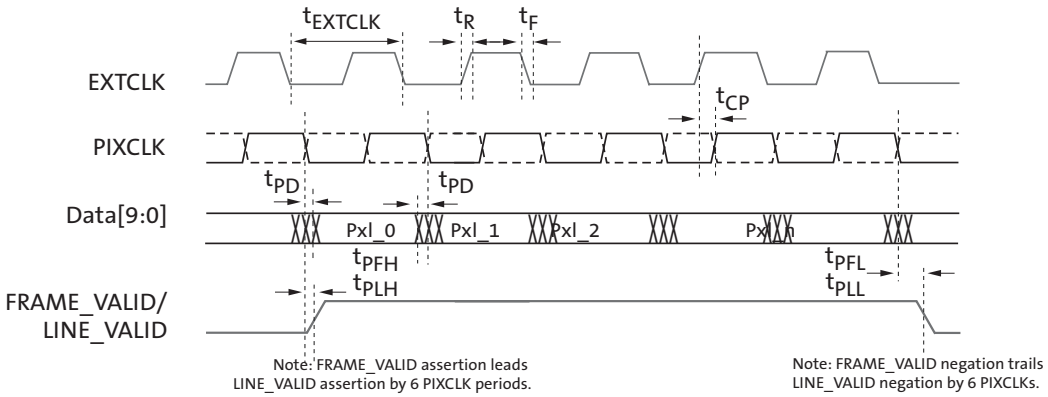
Image Height (%)	Image Height (mm)	CRA (Deg)
0	0	0
5	0.112	1.71
10	0.224	3.45
15	0.336	5.17
20	0.448	6.87
25	0.560	8.54
30	0.672	10.19
35	0.784	11.83
40	0.896	13.45
45	1.008	15.05
50	1.120	16.60
55	1.232	18.08
60	1.344	19.44
65	1.456	20.63
70	1.568	21.61
75	1.680	22.35
80	1.792	22.81
85	1.904	23.01
90	2.016	22.99
95	2.128	22.85
100	2.240	22.76

Figure 3: Quantum Efficiency



Electrical Specifications

Figure 4: Default Data Output Timing Diagram



EXTCLK

The electrical characteristics of the EXTCLK input are shown in Table 21 on page 4. The EXTCLK input supports either an AC-coupled sine-wave input clock or a DC-coupled square-wave input clock.

If EXTCLK is AC-coupled to the MT9T013 and the clock is stopped, the EXTCLK input to the MT9T013 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 1: 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; Junction temperature = 70°C

Definition	Condition	Symbol	Min	Typ	Max	Unit
Input clock frequency		^{f}EXTCLK1	6		48	MHz
Input clock period		^{t}EXTCLK1	21		167	ns
Input clock rise slew rate			1			V/ns
Input clock fall slew rate			1			V/ns
Input clock minimum voltage swing (AC coupled)		V_{IN_AC}	0.5			V (p-p)
Input clock maximum voltage (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	ms
Input pad capacitance		C_{IN}		3.5		pF
Input leakage current		I_{IN}	–10		10	µA
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

Parallel Pixel Data Interface

The electrical characteristics of the parallel pixel data interface (FV, LV, D_{OUT}[9:0], PIXCLK, SHUTTER, and FLASH outputs) are shown in Table 22.

Table 2: Electrical Characteristics (Parallel Pixel Data Interface)

^{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; Junction temperature = 70°C

Definition	Condition	Symbol	Min	Typ	Max	Unit
Output HIGH voltage	At specified I_{OH} 8mA	V_{OH}	V_{DD_IO} - 0.4			V
Output LOW voltage	At specified I_{OL} 8mA	V_{OL}			0.4	V
Output HIGH current	At specified V_{OH}, V_{DD_IO} = 1.8V	I_{OH}			20	mA
Output LOW current	At specified V_{OL} 0.1V	I_{OL}			-15	mA
Output LOW current	At specified V_{OL} 0.4V	I_{OL}			-25	mA
Tri-state output leakage current		I_{OZ}	–10		10	µA
Output pin slew (rising)	Default slew rate register settings, C_{LOAD} = 35pF, 64 MHz PIXCLK			0.29		V/ns
Output pin slew (falling)	Default slew rate register settings, C_{LOAD} = 35pF, 64 MHz PIXCLK			0.4		V/ns
PIXCLK frequency	default	^{f}PIXCLK		72	72	MHz
Propagation delay	Rising edge to rising edge	^{t}CP			14	ns
PIXCLK to data valid	72 MHz PIXCLK frequency	^{t}PD		3	7	ns
PIXCLK to FV HIGH	72 MHz PIXCLK frequency	^{t}FH		3	7	ns
PIXCLK to LV HIGH	72 MHz PIXCLK frequency	^{t}PLH		3	7	ns
PIXCLK to FV LOW	72 MHz PIXCLK frequency	^{t}PFL		3	7	ns

Table 2: Electrical Characteristics (Parallel Pixel Data Interface) (continued)

^{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; Junction temperature = 70°C

Definition	Condition	Symbol	Min	Typ	Max	Unit
PIXCLK to LV LOW	72 MHz PIXCLK frequency	^{t}PLL		3	7	ns

Two-Wire Serial Register Interface

The electrical characteristics of the two-wire serial register interface (SCLK, SDATA) are shown in Table 23. The SCLK and SDATA signals feature fail-safe input protection, Schmitt trigger input, and suppression of input pulses of less than 50ns duration.

Table 3: 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; Junction temperature = 70°C

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		2	µA
Output LOW voltage	At specified I_{OL} 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

Figure 5: Two-Wire Serial Bus Timing Parameters

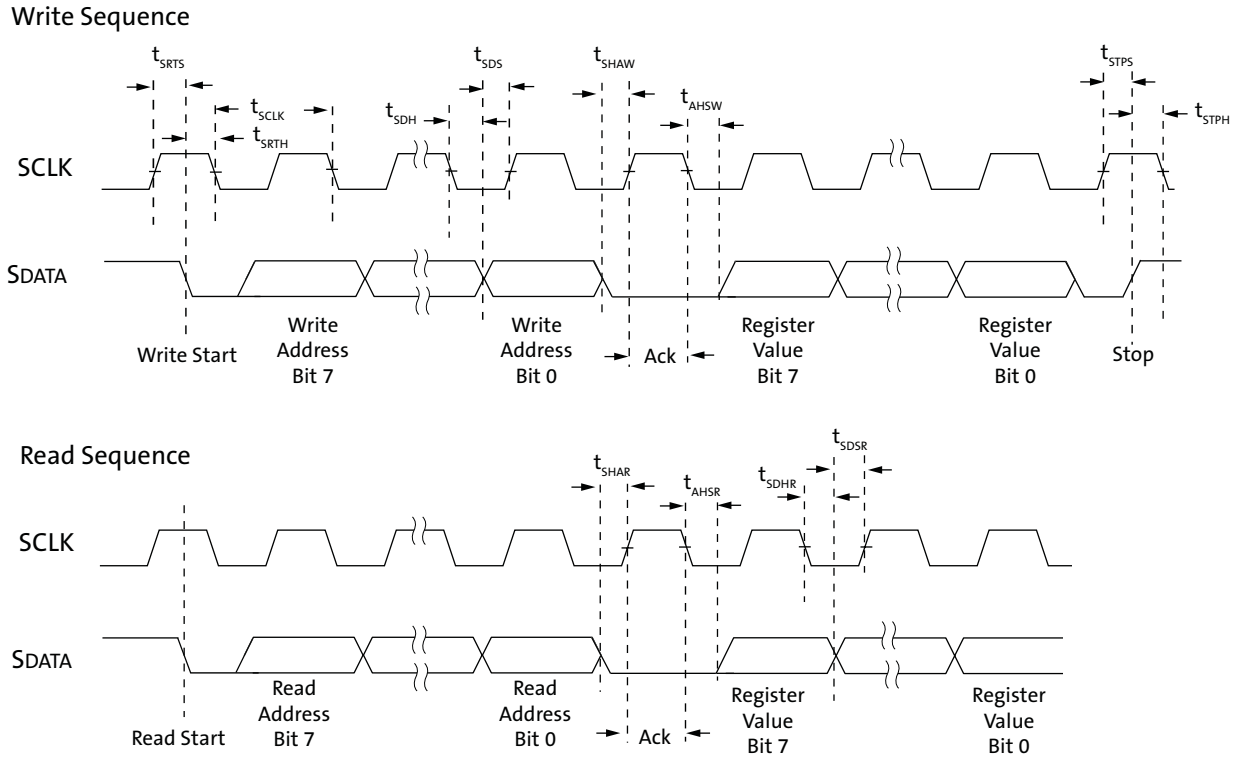


Table 4: Two-Wire Serial Interface Timing Specifications

Symbol	Definition	Condition	Min	Typ	Max	Unit
^{f}SCLK	Serial Interface Input clock frequency			100	400	KHz
^{t}SCLK	Serial Interface Input clock period		2.5	10		us
	SCLK duty cycle		45	50	50	%
tr	SCLK /SDATA rise time				300	ns
^{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 from Slave	0.15		0.65	µs
^{t}AHSW	ACK hold to SDATA	Master read from Slave	0.15		0.65	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		0.65	µs
^{t}AHSR	ACK hold to SDATA	Master write to Slave	0.3		0.65	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

Serial Pixel Data Interface

The electrical characteristics of the serial pixel data interface (CLK_P, CLK_N, DATA_P, and DATA_N) are shown in Table 25.

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

Table 5: Electrical Characteristics (Serial Pixel Data Interface)

^{f}EXTCLK = 24 MHz; VDD = 1.8V; VDD_IO = 1.8V; VAA = 2.8V; VAA_PIX = 2.8V; VDD_PLL = 2.8V; Output load = 56pF; Ambient temperature

Definition	Symbol	Min	Typ	Max	Unit
Operating frequency				650	MHz
Fixed common mode voltage	VCMF		0.85		V
Differential voltage swing	VOD		145		mV
Drive current range		TBD	TBD	TBD	mA
Drive current variation		TBD	TBD	TBD	%
Output impedance			70		W
Output impedance mismatch			3		%
Clock duty cycle at 416 MHz			52		%
Rise time (20–80%)	VOD		200		ps
Fall time (20–80%)	VOD		205		ps
Differential skew			27		ps
Channel-to-channel slew				100	ps
Maximum data rate Data/strobe mode Data/clock mode				 640 208	Mb/s
Power supply rejection ratio (PSRR) 0–100 MHz					dB
Power supply rejection ratio (PSRR) 100–1,000 MHz					dB

Control Interface

The electrical characteristics of the control interface (RESET_BAR, TEST, GPI0, GPI1, GPI2, and GPI3) are shown in Table 26.

Table 6: AC Electrical Characteristics (Control Interface)

^{f}EXTCLK = 24 MHz; VDD = 1.8V; VDD_IO = 1.8V; VAA = 2.8V; VAA_PIX = 2.8V; VDD_PLL = 2.8V; Output load = 68.5pF; Junction temperature = 70°C

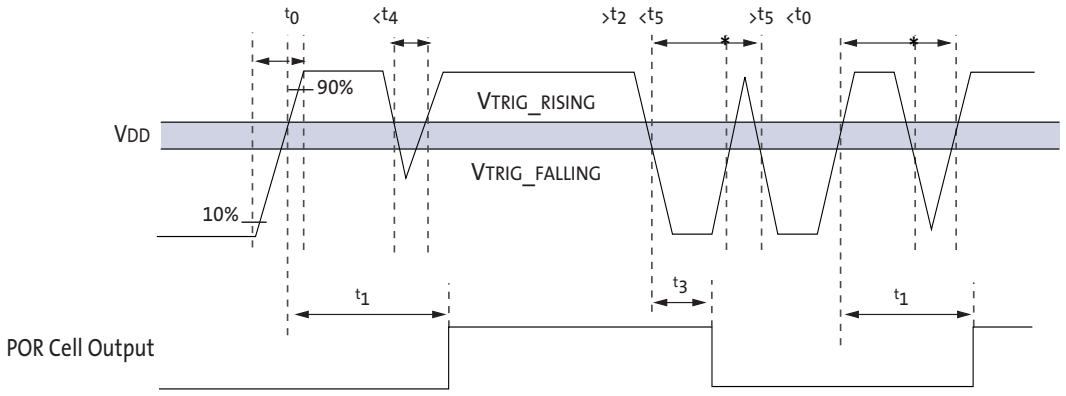
Definition	Condition	Symbol	Min	Typ	Max	Unit
Input HIGH voltage		VIH	0.7 x VDD_IO		VDD_IO + 0.5	V
Input LOW voltage		VIL	-0.5		0.3 x VDD_IO	V
Input leakage current	No pull-up resistor; VIN = VDD_IO or DGND	IIN	-10		10	µA
Input pad capacitance		CIN		6.5		pF

Power-On Reset

Table 7: Power-On Reset Characteristics

Definition	Condition	Symbol	Min	Typ	Max	Unit
VDD rising, crossing VTRIG_RISING; Internal reset being released		t_1		10	15	µs
VDD falling, crossing VTRIG_FALLING; Internal reset active		t_2		0.5	1	µs
VDD spike width below VTRIG_FALLING; considered to be a reset when POR cell output is HIGH		t_3		0.5		
VDD spike width below VTRIG_FALLING; considered to be a reset when POR cell output is LOW		t_4		1		µs
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	1.25	1.45	V

Figure 6: Internal Power-On Reset



Operating Voltages

V_{AA} and VAA_PIX must be at the same potential for correct operation of the MT9T013.

Table 8: DC Electrical Definitions and Characteristics

^{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; Junction temperature = 70°C

Definition	Condition	Symbol	Min	Typ	Max	Unit
Core digital voltage		V_{DD}	1.7	1.8	1.9	V
I/O digital voltage	Parallel pixel data interface	V_{DD}_IO	1.7	1.8	1.9	V
			2.4	2.8	3.1	V
I/O digital voltage	Serial pixel (CCP2) data interface	V_{DD}_CCP	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	Streaming, full resolution	I_{DD}1		28		mA
I/O digital operating current	Streaming, full resolution	I_{DD}_IO		12		mA
Analog operating current	Streaming, full resolution	I_{AA}		82		mA
Pixel supply current	Streaming, full resolution	I_{AA}_PIX		2		mA
PLL supply current	Streaming, full resolution	I_{DD}_PLL		19		mA
Hard standby (clock off)	Analog, 3.1V				15	µA
	Digital, 1.9V				115	µA
Hard standby (clock on)	Analog, 3.1V				50	µA
	Digital, 1.9V				2100	µA
Soft standby (clock off)	Analog, 3.1V				15	µA
	Digital, 1.9V				115	µA
Soft standby (clock on)	Analog, 3.1V				50	µA
	Digital, 1.9V				1530	µA

Absolute Maximum Ratings

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

Table 9: Absolute Maximum Values

Definition	Condition	Symbol	Min	Typ	Max	Unit
Core digital voltage		V_{DD_MAX}	–	–	2.4	V
I/O digital voltage		$V_{DD\_IO\_MAX}$	–	–	4	V
Analog voltage		V_{AA_MAX}	–	–	4	V
Pixel supply voltage		$V_{AA\_PIX\_MAX}$	–	–	4	V
PLL supply voltage		$V_{DD\_PLL\_MAX}$	–	–	4	V
Digital operating current	Worst case current	I_{DD_MAX}	–	–	60	mA
I/O digital operating current	Worst case current	$I_{DD\_IO\_MAX}$	–	–	75	mA
Analog operating current	Worst case current	I_{AA_MAX}	–	–	140	mA
Pixel supply current	Worst case current	$I_{AA\_PIX\_MAX}$	–	–	5	mA
PLL supply current	Worst case current	$I_{DD\_PLL\_MAX}$	–	–	40	mA
Operating temperature	Measure at junction	T_{OP}	-30	–	70	°C
Storage temperature		T_{STG}	-40	–	85	°C

Notes: 1. This is a stress rating only, and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

SMIA Specification Reference

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

–Functional Specification:

SMIA 1.0 Part 1: Functional Specification (Version 1.0 dated 30-June-2004)

SMIA 1.0 Part 1: Functional Specification ECR0001 (Version 1.0 dated 11-Feb-2005)

–Electrical Specification

SMIA 1.0 Part 2: CCP2 Specification (Version 1.0 dated 30-June-2004)

SMIA 1.0 Part 2: CCP2 Specification ECR0002 (Version 1.0 dated 11-Feb-2005)

Revision History

Rev. J 3/16/12

- Updated trademarks

Rev. I 1/21/10

- Updated to Aptina template
- Moved register tables to a separate document (MT9T013 Register Reference)

Rev. H 6/12/08

- Upated values in Table 34, “DC Electrical Definitions and Characteristics,” on page 119
  - Hard standby (clock off) to 15
  - Hard standby (clock on) to 50
  - Hard standby (clock off) to 15
- Updated values in Table 30, “Two-Wire Serial Interface Timing Specifications,” on page 116
  - ^{f}SCLK from 100 MIN to 100 TYP and 400 MAX
  - ^{t}SCLK from 2.5 TYP to 2.5 MIN and 10 TYP

Rev. G 04/15/08

- Updated “Power-Up Sequence” on page 67
- Updated Figure 45: “Power-Up Sequence,” on page 67
- Updated "Power-Down Sequence" on page 68
- Updated Figure 46: “Power-Down Sequence,” on page 68
- Updated to Aptina template

Rev. F 01/04/08

- Update Figure 4: “Typical Configuration: Parallel Pixel Data Interface,” on page 10
- Update Table 22, “Electrical Characteristics (Parallel Pixel Data Interface),” on page 4
- Update Figure 53: “Two-Wire Serial Bus Timing Parameters,” on page 6

Rev. E 10/18/07

- Update "Register Notation" on page 19
- Update Table 12, “Register Description—Manufacturer-Specific,” on page 44
- Update Figure 16: “MT9T013 System States,” on page 31 (remove Note)
- Update Table 11, “PLL in System States,” on page 32
- Update "Power-On Reset Sequence" on page 32
- Update "Programming Restrictions when Subsampling" on page 45
- Update "Integration Time" on page 50
- Update"Power-Up Sequence" on page 67
- Update "Soft Reset" on page 70
- Update Table 28, “DC Electrical Definitions and Characteristics,” on page 10

Rev. D 06/11/07

- Update to Production data sheet
- Update Table 1, “Key Performance Parameters,” on page 1
- Update Figure 19: “Sequence for Programming the Device,” on page 40
- Update Table 21, “Electrical Characteristics (EXTCLK),” on page 4
- Update Table 22, “Electrical Characteristics (Parallel Pixel Data Interface),” on page 4

- Update Table 23, “Two-Wire Serial Register Interface Electrical Characteristics,” on page 6
- Add Table 24, “Two-Wire Serial Interface Timing Specifications,” on page 7
- Add Figure 53: “Two-Wire Serial Bus Timing Parameters,” on page 6
- Update Table 25, “Electrical Characteristics (Serial Pixel Data Interface),” on page 8
- Update Table 28, “DC Electrical Definitions and Characteristics,” on page 10

Rev. C 05/01/07

- Update "Features" on page 1
- Update Table 1, “Key Performance Parameters,” on page 1
- Update Table 3, “Signal Descriptions,” on page 11
- Update Figure 10: “Sequential READ, Start from Random Location,” on page 17
- Update Figure 11: “Sequential READ, Start from Current Location,” on page 17
- Update Figure 12: “Single WRITE to Random Location,” on page 17
- Update Figure 13: “Sequential WRITE, Start at Random Location,” on page 18
- Update Table 9, “Register List and Default Values—Manufacturer-Specific,” on page 26
- Update Table 21, “Electrical Characteristics (EXTCLK),” on page 4
- Update Table 22, “Electrical Characteristics (Parallel Pixel Data Interface),” on page 4
- Update Table 23, “Two-Wire Serial Register Interface Electrical Characteristics,” on page 6
- Update Table 25, “Electrical Characteristics (Serial Pixel Data Interface),” on page 8
- Update Table 26, “AC Electrical Characteristics (Control Interface),” on page 8
- Update Table 27, “Power-On Reset Characteristics,” on page 9
- Update Table 28, “DC Electrical Definitions and Characteristics,” on page 10
- Update Table 29, “Absolute Maximum Values,” on page 11

Rev. B 02/20/07

- Update "Features" on page 1
- Update Table 1, “Key Performance Parameters,” on page 1
- Update "Functional Overview" on page 7
- Update Figure 2: “Pixel Color Pattern Detail (Top Right Corner),” on page 8
- Update Figure 3: “Typical Configuration: Serial Pixel Data Interface,” on page 9
- Update Figure 4: “Typical Configuration: Parallel Pixel Data Interface,” on page 10
- Update Table 3, “Signal Descriptions,” on page 11
- Update "Parallel Pixel Data Interface Output Data Timing" on page 12
- Update "Registers" on page 19
- Update Table 7, “Register List and Default Values—SMIA Configuration,” on page 23
- Update Table 8, “Register List and Default Values—SMIA Parameter Limits,” on page 25
- Update Table 9, “Register List and Default Values—Manufacturer-Specific,” on page 27
- Update Table 10, “Register List and Default Values—SMIA Configuration,” on page 33
- Update Table 11, “Register List and Default Values—SMIA Parameter Limits,” on page 35
- Update Table 12, “Register List and Default Values—Manufacturer-Specific,” on page 37
- Update Table 10, “Register Description—SMIA Configuration,” on page 35
- Update Table 11, “Register Description—SMIA Parameter Limits,” on page 40

- Update Table 12, “Register Description—Manufacturer-Specific,” on page 44
- Update Figure 17: “MT9T013 SMIA Profile 1, 2 Clocking Structure,” on page 35
- Update “Trigger Control,” on page 34
- Update "One-Time Programmable (OTP) Memory" on page 39
- Update Table 16, “Register Adjustments Required for Binning Mode,” on page 49
- Update "Integration Time" on page 50
- Update Equation 16 on page 50
- Update Table 17, “Recommended Gain Settings,” on page 58
- UpdateFigure 44: “Data Path,” on page 66
- Update "Power-Up Sequence" on page 67
- Update Figure 45: “Power-Up Sequence,” on page 67
- Update Figure 49: “Chief Ray Angle (CRA) (Z18 Type A),” on page 1
- Add Figure 50: “Chief Ray Angle (CRA) (Z19 Type B),” on page 2
- Update "Power-Down Sequence" on page 68
- Update Figure 46: “Power-Down Sequence,” on page 68
- Update Figure 51: “Quantum Efficiency,” on page 2

Rev. A 11/06

- Initial release

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