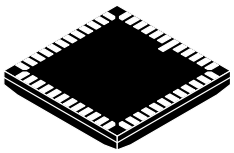
# 1/2.3-Inch 10 Mp CMOS Digital Image Sensor

# MT9J003



ILCC48 10x10
CASE 847AK

## ORDERING INFORMATION

See detailed ordering and shipping information on page 3 of this data sheet.

### General Description

The **onsemi** MT9J003 is a 1/2.3−inch CMOS active−pixel digital imaging sensor with an active pixel array of 3856 (H) x 2764 (V) including border pixels. It can support 10 Mp (3664 (H) x 2748 (V)) digital still images and a 1080 p (3840 (H) x 2160 (V)) digital video mode. It incorporates sophisticated on−chip camera functions such as windowing, mirroring, column and row skip modes, and snapshot mode. It is programmable through a simple two−wire serial interface and has very low power consumption.

The MT9J003 digital image sensor features **onsemi**'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 4:3 still−mode, the sensor generates a full resolution image at 15 frames per second (fps) using the HiSPi serial interface. An on−chip analog−to−digital converter (ADC) generates a 12−bit value for each pixel.

### Features

- 1080p Digital Video Mode
- Simple Two−wire Serial Interface
- Auto Black Level Calibration
- Support for External Mechanical Shutter
- Support for External LED or Xenon Flash
- High Frame Rate Preview Mode with Arbitrary Down−size Scaling from Maximum Resolution
- Programmable Controls: Gain, Horizontal and Vertical Blanking, Auto Black Level Offset Correction, Frame Size/rate, Exposure, Left−right and Top−bottom Image Reversal, Window Size, and Panning
- Data Interfaces: Parallel or Four−lane Serial High−speed Pixel Interface (HiSPi) Differential Signaling (Sub−LVDS)
- On−die Phase−locked Loop (PLL) Oscillator
- Bayer Pattern Downsize Scaler
- Integrated Position−based Color and Lens Shading Correction
- One−time Programmable Memory (OTPM) for Storing Module Information

### Applications

- Digital Video Cameras
- Digital Still Cameras

© Semiconductor Components Industries, LLC, 2009

**Table 1. KEY PARAMETERS**

| Parameter | | Value |
|---|---|---|
| Optical Format | | 1/2.3−inch (4:3) |
| Active Imager Size | | 6.440 mm (H) x 4.616 mm (V), 7.923 mm Diagonal (Entire Sensor)<br>6.119 mm (H) x 4.589 mm (V), 7.649 mm Diagonal (Still Mode)<br>6.413 mm (H) x 3.607 mm (V), 7.358 mm Diagonal (Video Mode) |
| Active Pixels | | 3856 (H) x 2764 (V) (Entire Sensor)<br>3664 (H) x 2748 (V) (4:3, Still Mode)<br>3840 (H) x 2160 (V) (16:9, Video Mode) |
| Pixel Size | | 1.67 x 1.67 μm |
| Chief Ray Angle | | 0°, 13.4° |
| Color Filter Array | | RGB Bayer Pattern |
| Shutter Type | | Electronic Rolling Shutter (ERS) with Global Reset Release (GRR) |
| Maximum Data Rate | | 96 Mp/s |
| Maximum Master Clock | | 60 MHz |
| Input Clock Frequency | | 6−48 MHz |
| Maximum Data Rate | Parallel | 80 Mp/s at 80 MHz PIXCLK |
| | HiSPi (4−lane) | 2.8 Gbps |
| Frame Rate | Still Mode, 4:3 (3664 (H) x 2748 (V) | Programmable up to 15 fps Serial I/F, 7.5 fps Parallel I/F |
| | Preview Mode<br>VGA | 30 fps with Binning<br>60 fps with Skip2bin2 |
| | 1080p Mode<br>(1920 H x 1080 V) | 60 fps Using HiSPi I/F<br>30 fps Using Parallel I/F |
| ADC Resolution | | 12−bit, On−die |
| Responsivity | | 0.31 V/lux−sec (550 nm) |
| Dynamic Range | | 65.2 dB |
| $SNR_{MAX}$ | | 34 dB |
| Supply Voltage | I/O Digital | 1.7−1.9 (V) (1.8 (V) Nominal)<br>or 2.4−3.1 (V) (2.8 (V) Nominal) |
| | Digital | 1.7−1.9 (V) (1.8 (V) Nominal) |
| | Analog | 2.4−3.1 (V) (2.8 (V) Nominal) |
| | SLVS I/O | 0.4−0.8 (V) (0.4 or 0.8 (V) Nominal) |
| Power Consumption | Still Mode at 15 fps w/ Serial I/F | 638 mW |
| | Still Mode at 7.5 fps w/ Parallel I/F | 388 mW |
| | Preview | 250 mW Low Power VGA |
| | Standby | 500 μW (Typical, EXTCLK Disabled) |
| Power Consumption | | TBD |
| Package | | 48−pin iLCC (10 mm x 10 mm) Bare Die,<br>48pin Tiny PLCC (12 mm x 12 mm) |
| Operating Temperature | | −30°C to +70°C (at Junction) |

## ORDERING INFORMATION

**Table 2. AVAILABLE PART NUMBERS**

| Part Number | Product Description | Orderable Product Attribute Description† |
|---|---|---|
| MT9J003D00STMUC2CBC1−200 | 10 MP 1” CIS | Die Sales, 200 μm Thickness |
| MT9J003I12STCU−DP | 10 MP 1/2.3” CIS | Dry Pack with Protective Film |
| MT9J003I12STCU−DR | 10 MP 1/2.3” CIS | Dry Pack without Protective Film |
| MT9J003I12STCV2−DP | 10 MP 1/2.3” CIS | Dry Pack with Protective Film |
| MT9J003I12STCV2−TP | 10 MP 1/2.3” CIS | Tape & Reel with Protective Film |
| MT9J003I12STMU−DP | 10 MP 1/2.3” CIS | Dry Pack with Protective Film |

†For information on tape and reel specifications, including part orientation and tape sizes, please refer to our Tape and Reel Packaging Specification Brochure, BRD8011/D.

## FUNCTIONAL OVERVIEW

The MT9J003 is a progressive−scan sensor that generates a stream of pixel data at a constant frame rate. It uses an on−chip, phase−locked loop (PLL) to generate all internal clocks from a single master input clock running between 6 and 48 MHz. The maximum output pixel rate is 80 Mp/s, corresponding to a pixel clock rate of 80 MHz. A block diagram of the sensor is shown in Figure 1.

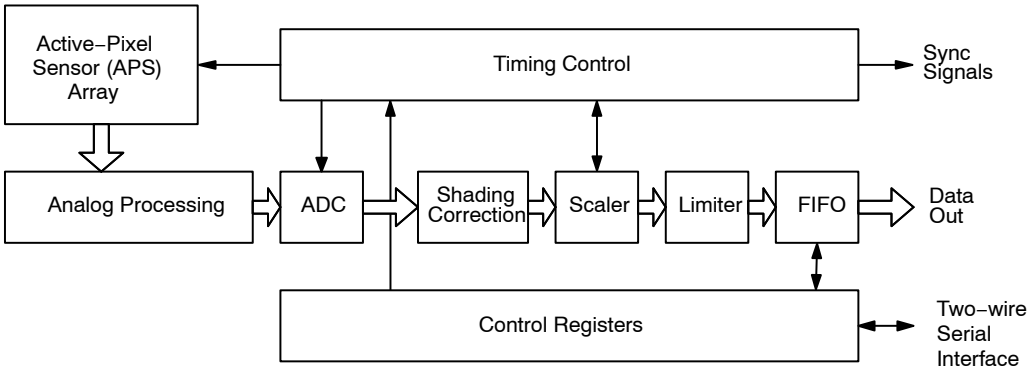



**Figure 1. Block Diagram**

The core of the sensor is a 10 Mp active−pixel array. The timing and control circuitry sequences through the rows of the array, resetting and then reading each row in turn. In the time interval between resetting a row and reading that row, the pixels in the row integrate incident light. The exposure is controlled by varying the time interval between reset and readout. Once a row has been read, the data from the columns is sequenced through an analog signal chain (providing offset correction and gain), and then through an ADC. The output from the ADC is a 12−bit value for each pixel in the array. The ADC output passes through a digital processing signal chain (which provides further data path corrections and applies digital gain).

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

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

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

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

- A sensor core that provides array control and data path corrections. The output of the sensor core is a 12−bit parallel pixel data stream qualified by an output data clock (PIXCLK), together with LINE_VALID (LV) and FRAME_VALID (FV) signals or a 4−lane serial high−speed pixel interface (HiSPi).
- A digital shading correction block to compensate for color/brightness shading introduced by the lens or chief ray angle (CRA) curve mismatch.
- Additional functionality is provided. This includes a horizontal and vertical image scaler, a limiter, a data compressor, an output FIFO, and a serializer.

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

are also available to reduce the effect of electromagnetic interference from the output interface.

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

### Pixel Array

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

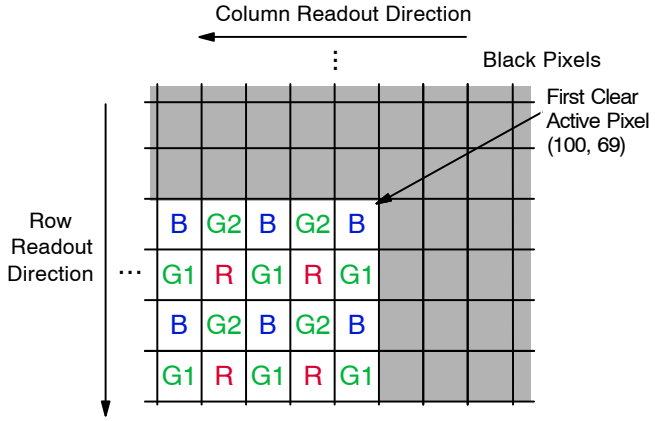


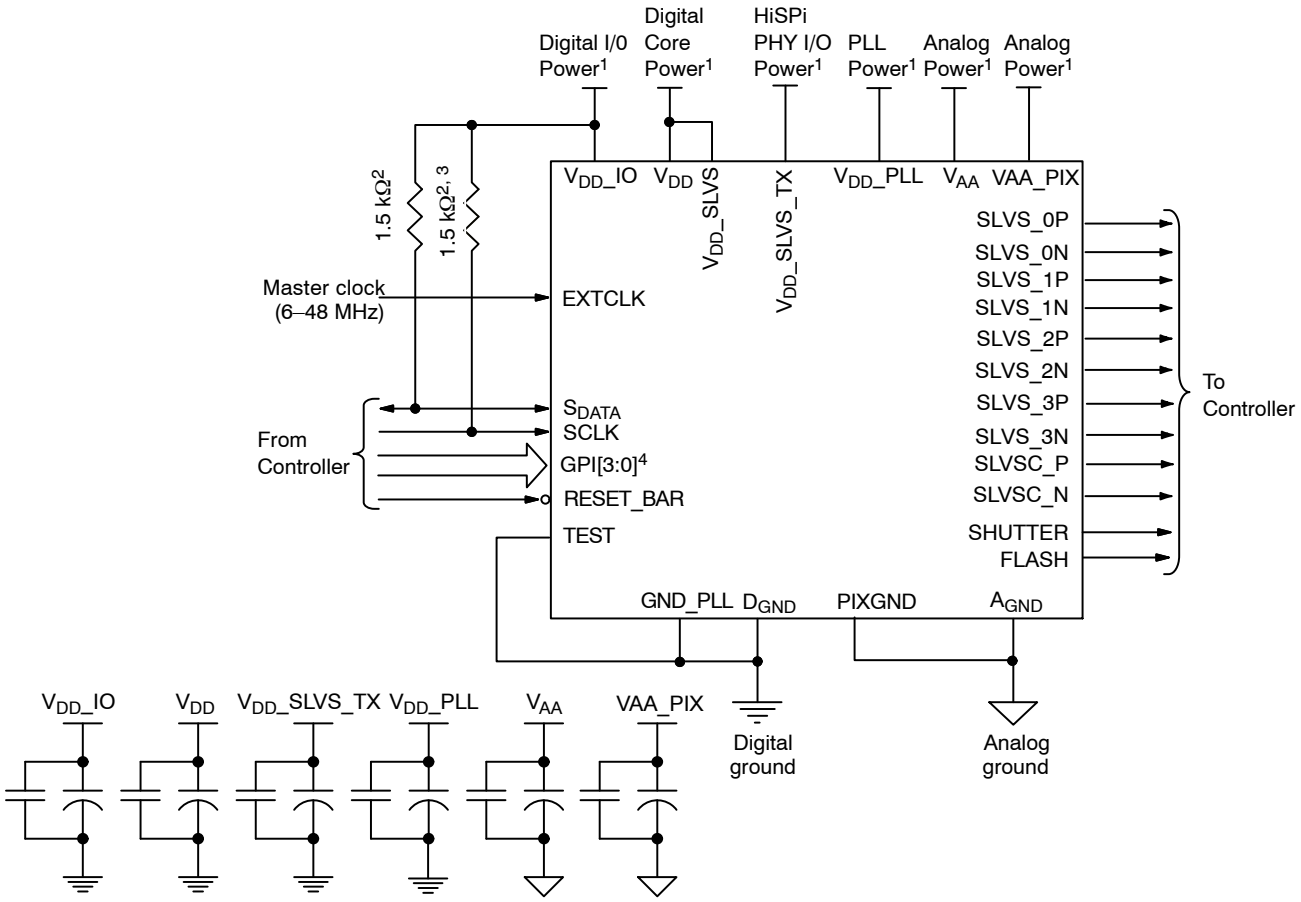

**Figure 2. Block Diagram**

## OPERATING MODES

By default, the MT9J003 powers up with the serial pixel data interface enabled. The sensor can operate in serial HiSPi or parallel mode

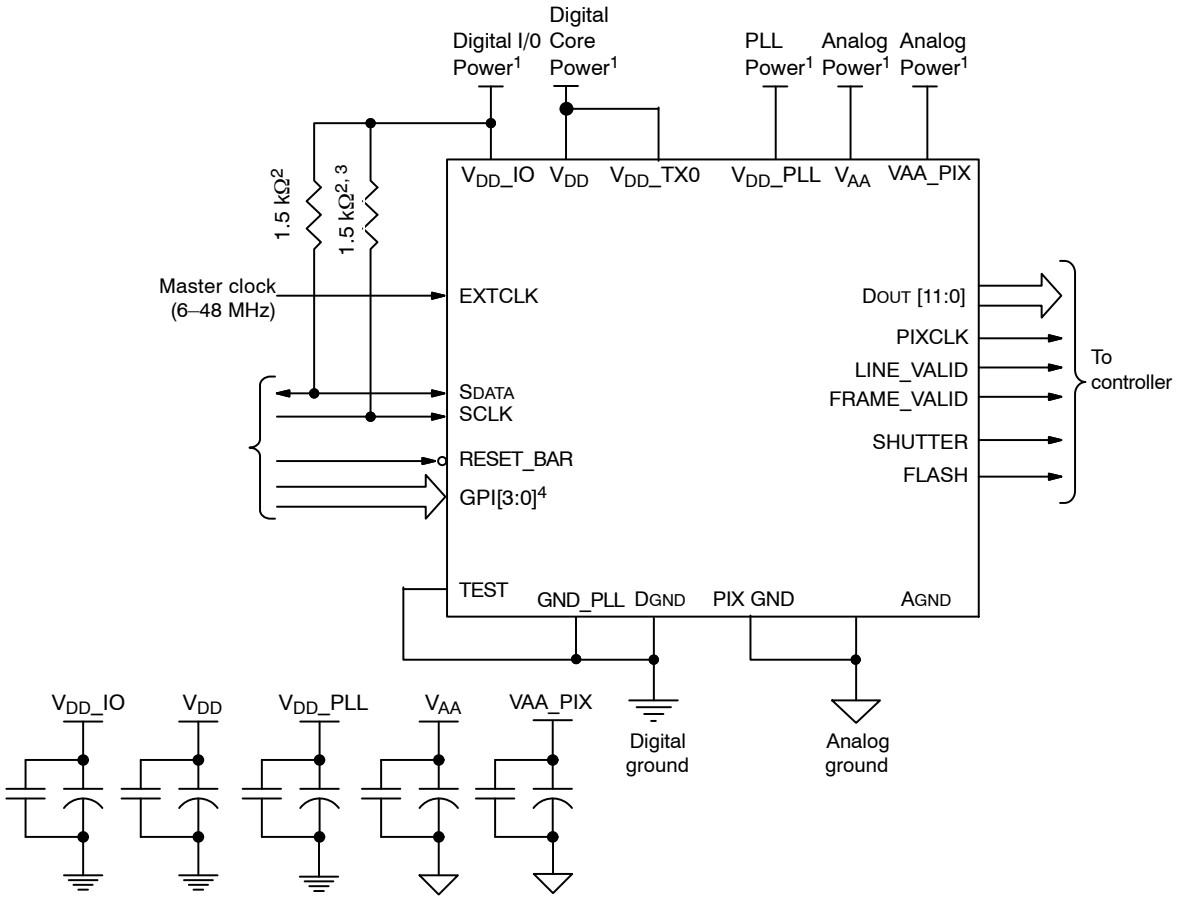
For low−noise operation, the MT9J003 requires separate power supplies for analog and digital power. Incoming digital and analog ground conductors should be placed in such a way that coupling between the two are minimized. Both power supply rails should also be routed in such a way that noise coupling between the two supplies and ground is minimized.

**CAUTION:** **onsemi** does not recommend the use of inductance filters on the power supplies or output signals.




1. All power supplies should be adequately decoupled.
2. **onsemi** recommends a resistor value of 1.5 kΩ, but it may be greater for slower two−wire speed.
3. This pull−up resistor is not required if the controller drives a valid logic level on SCLK at all times.
4. The GPI pins can be statically pulled HIGH or LOW to be used as module IDs, or they can be programmed to perform special functions (TRIGGER, OE_N, SADDR, STANDBY) to be dynamically controlled.
5. VPP, which can be used during the module manufacturing process, is not shown in Figure 3. This pad is left unconnected during normal operation.
6. The parallel interface output pads can be left unconnected if the serial output interface is used.
7. **onsemi** recommends that 0.1 μF and 10 μF 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. Check the MT9J003 demo head-board schematics for circuit recommendations
8. **onsemi** recommends that analog power planes are placed in a manner such that coupling with the digital power planes is minimized.
9. The signal path between the HiSPi serial transmitter and receiver should be adequately designed to minimize any trans−impedance mismatch and/or reflections on the data path.

**Figure 3. Typical Configuration: Serial Four−Lane HiSPi Interface**




1. All power supplies should be adequately decoupled.
2. **onsemi** recommends a resistor value of 1.5 kΩ, but it may be greater for slower two−wire speed.
3. This pull−up resistor is not required if the controller drives a valid logic level on SCLK at all times.
4. The GPI pins can be statically pulled HIGH or LOW to be used as module IDs, or they can be programmed to perform special functions (TRIGGER, OE_N, SADDR, STANDBY) to be dynamically controlled.
5. VPP, which can be used during the module manufacturing process, is not shown in Figure 4. This pad is left unconnected during normal operation.
6. The serial interface output pads can be left unconnected if the parallel output interface is used.
7. **onsemi** recommends that 0.1 μF and 10 μF 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. Check the MT9J003 demo head-board schematics for circuit recommendations.
8. **onsemi** recommends that analog power planes are placed in a manner such that coupling with the digital power planes is minimized.
9. **onsemi** recommends that VDD_TX0 is tied to VDD when the sensor is using the parallel interface.

**Figure 4. Typical Configuration: Parallel Pixel Data Interface**

## SIGNAL DESCRIPTIONS

Table 3 provides signal descriptions for MT9J003 die. For pad location and aperture information, refer to the MT9J003 die data sheet.

**Table 3. SIGNAL DESCRIPTIONS**

| Pad Name | Pad Type | Description |
|---|---|---|
| EXTCLK | Input | Master Clock Input, 6–48 MHz |
| RESET_BAR (XSHUTDOWN) | Input | Asynchronous active LOW reset. When asserted, data output stops and all internal registers are restored to their factory default settings |
| SCLK | Input | Serial clock for access to control and status registers |
| GPI[3:0] | Input | General purpose inputs. After reset, these pads are powered−down by default; this means that it is not necessary to bond to these pads. Any of these pads can be configured to provide hardware control of the standby, output enable, SADDR select, and shutter trigger functions.<br>Can be left floating if not used |
| TEST | Input | Enable manufacturing test modes. It should not be left floating. It can be tied to ground or VDD_IO when used in parallel or HiSPi. It should be connected to DGND for normal operation of the CCP2 configured sensor, or connected to VDD_IO power for the MIPI−configured sensor |
| SDATA | I/O | Serial data from READs and WRITEs to control and status registers |
| LINE_VALID | Output | LINE_VALID (LV) output. Qualified by PIXCLK |
| FRAME_VALID | Output | FRAME_VALID (FV) output. Qualified by PIXCLK |
| DOUT[11:0] | Output | Parallel pixel data output. Qualified by PIXCLK |
| PIXCLK | Output | Pixel clock. Used to qualify the LV, FV, and DOUT[11:0] outputs |
| FLASH | Output | Flash output. Synchronization pulse for external light source. Can be left floating if not used |
| SHUTTER | Output | Control for external mechanical shutter. Can be left floating if not used |
| VPP | Supply | Power supply used to program one−time programmable (OTP) memory. Disconnect pad when not programming or when feature is not used |
| VDD_TX0 | Supply | PHY power supply. Digital power supply for the MIPI or CCP2 serial data interface. **onsemi** recommends that VDD_TX0 is always tied to VDD when using an unpackaged sensor |
| VDD_SLVS | Supply | HiSPi power supply for data and clock output. This should be tied to VDD |
| VDD_SLVS_TX | Supply | Digital Power Supply for the HiSPi I/O |
| VAA | Supply | Analog Power Supply |
| VAA_PIX | Supply | Analog Power Supply for the Pixel Array |
| AGND | Supply | Analog Ground |
| VDD | Supply | Digital Power Supply |
| VDD_IO | Supply | I/O Power Supply |
| DGND | Supply | Common Ground for Digital and I/O |
| VDD_PLL | Supply | PLL Power Supply |
| GND_PLL | Supply | PLL Ground |
| PIXGND | Supply | Pixel Ground |
| SLVS_0P | Output | Lane 1 Differential HiSPi (LVDS) Serial Data (positive). Qualified by the SLVS Serial Clock |
| SLVS_0N | Output | Lane 1 Differential HiSPi (LVDS) Serial Data (negative). Qualified by the SLVS Serial Clock |
| SLVS_1P | Output | Lane 2 Differential HiSPi (LVDS) Serial Data (positive). Qualified by the SLVS Serial Clock |
| SLVS_1N | Output | Lane 2 Differential HiSPi (LVDS) Serial Data (negative). Qualified by the SLVS Serial Clock |
| SLVS_2P | Output | Lane 3 Differential HiSPi (LVDS) Serial Data (positive). Qualified by the SLVS Serial Clock |

**Table 3. SIGNAL DESCRIPTIONS** (continued)

| Pad Name | Pad Type | Description |
|---|---|---|
| SLVS_2N | Output | Lane 3 Differential HiSPi (LVDS) Serial Data (negative). Qualified by the SLVS Serial Clock |
| SLVS_3P | Output | Lane 4 Differential HiSPi (LVDS) Serial Data (positive). Qualified by the SLVS Serial Clock |
| SLVS_3N | Output | Lane 4 Differential HiSPi (LVDS) Serial Data (negative). Qualified by the SLVS Serial Clock |
| SLVS_CP | Output | Differential HiSPi (LVDS) Serial Clock (positive). Qualified by the SLVS Serial Clock |
| SLVS_CN | Output | Differential HiSPi (LVDS) Serial Clock (positive). Qualified by the SLVS Serial Clock |

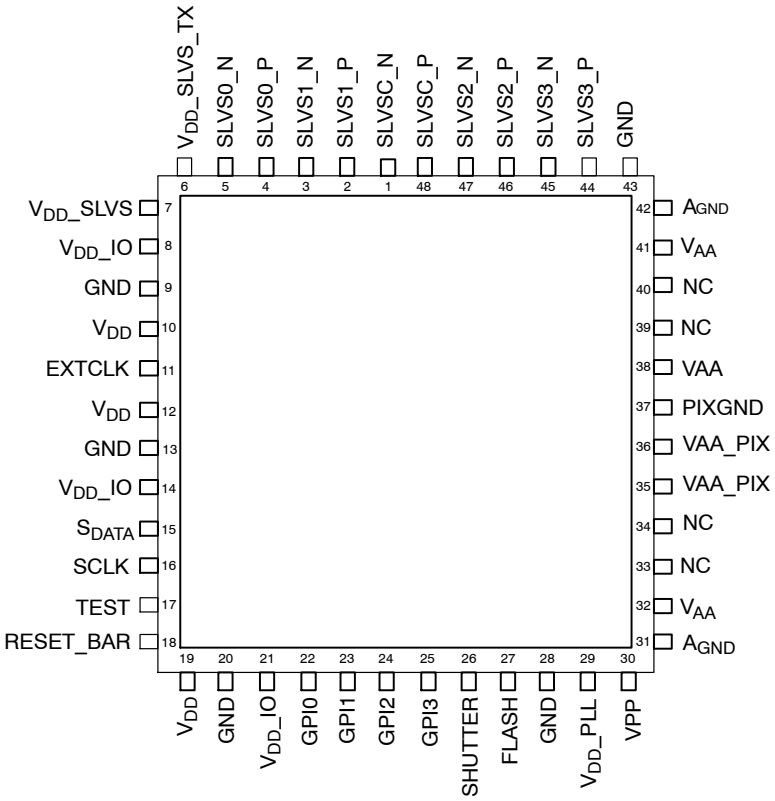



**Figure 5. HiSPi Package Pinout Diagram**

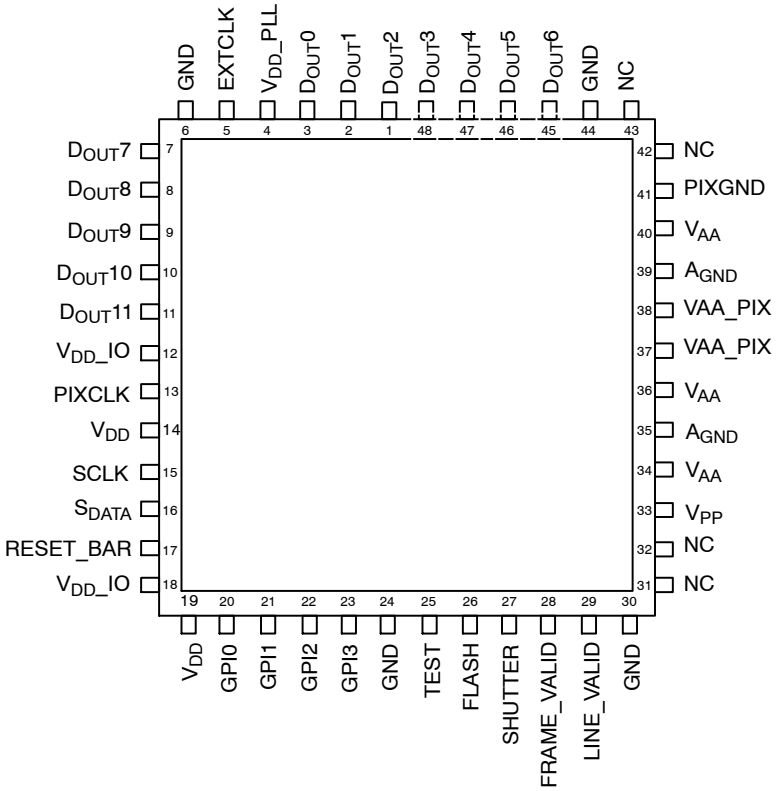



**Figure 6. 48−Pin iLCC Parallel Package Pinout Diagram**

## OUTPUT DATA FORMAT

### Serial Pixel Data Interface

The MT9J003 supports RAW8, RAW10, and RAW12 image data formats over a serial interface. The sensor supports a 1 and 2−lane MIPI as well as the HiSPi interface. These interfaces are not described in the data sheet.

### High Speed Serial Pixel Interface

The High Speed Serial Pixel (HiSPi) interface uses four data and one clock low voltage differential signaling (LVDS) outputs.

- SLVS_CP, SLVS_CN
- SLVS_[0:3]P, SLVS_[0:3]N

The HiSPi interface supports two protocols, streaming and packetized. The streaming protocol conforms to a standard video application where each line of active or intra−frame blanking provided by the sensor is transmitted at the same length. The packetized protocol will transmit only the active data ignoring line−to−line and frame−to−frame blanking data.

The HiSPi interface building block is a unidirectional differential serial interface with four data and one double data rate (DDR) clock lanes. One clock for every four serial data lanes is provided for phase alignment across multiple lanes. Figure 7 shows the configuration between the HiSPi transmitter and the receiver.

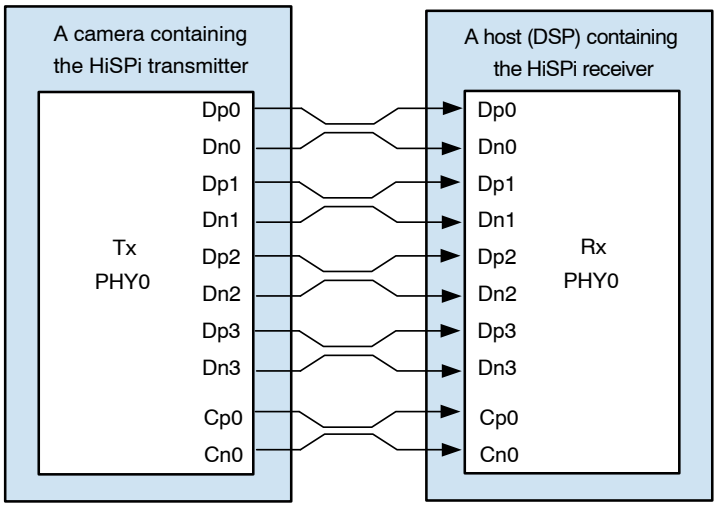



**Figure 7. HiSPi Transmitter and Receiver Interface Block Diagram**

*HiSPi Physical Layer*

The HiSPi physical layer is partitioned into blocks of four data lanes and an associated clock lane. Any reference to the PHY in the remainder of this document is referring to this minimum building block.

The PHY will serialize a 10−, 12−, 14− or 16−bit data word and transmit each bit of data centered on a rising edge of the clock, the second on the following edge of clock. Figure 8 shows bit transmission. In this example, the word is transmitted in order of MSB to LSB. The receiver latches data at the rising and falling edge of the clock.

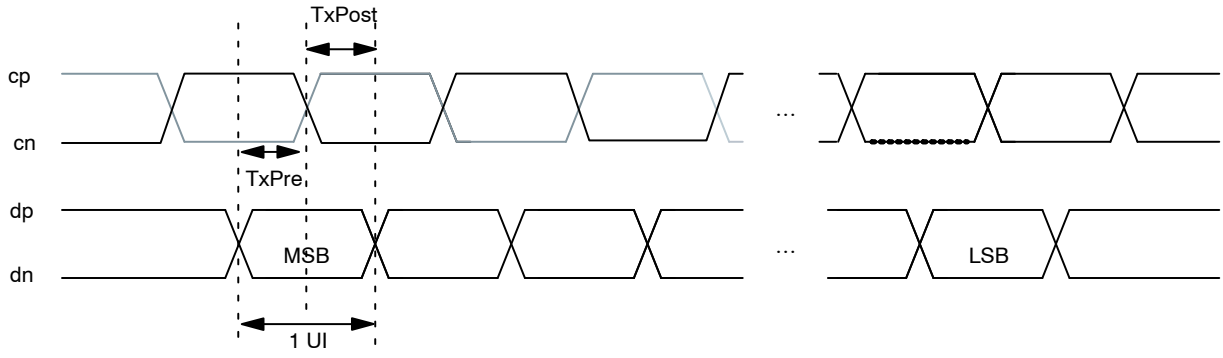



**Figure 8. Timing Diagram**

### DLL Timing Adjustment

The specification includes a DLL to compensate for differences in group delay for each data lane. The DLL is connected to the clock lane and each data lane, which acts as a control master for the output delay buffers. Once the DLL has gained phase lock, each lane can be delayed in 1/8 unit interval (UI) steps. This additional delay allows the user to increase the setup or hold time at the receiver circuits and can be used to compensate for skew introduced in PCB design.

If the DLL timing adjustment is not required, the data and clock lane delay settings should be set to a default code of 0x000 to reduce jitter, skew, and power dissipation.

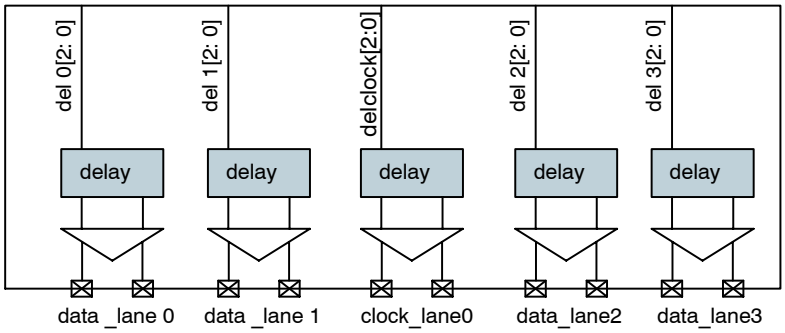



**Figure 9. Block Diagram of DLL Timing Adjustment**

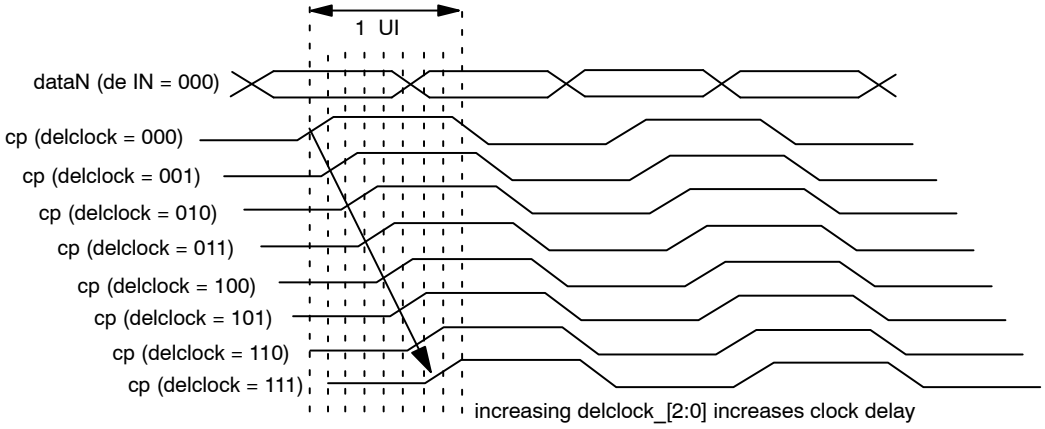



**Figure 10. Delaying the clock_lane with Respect to data_lane**

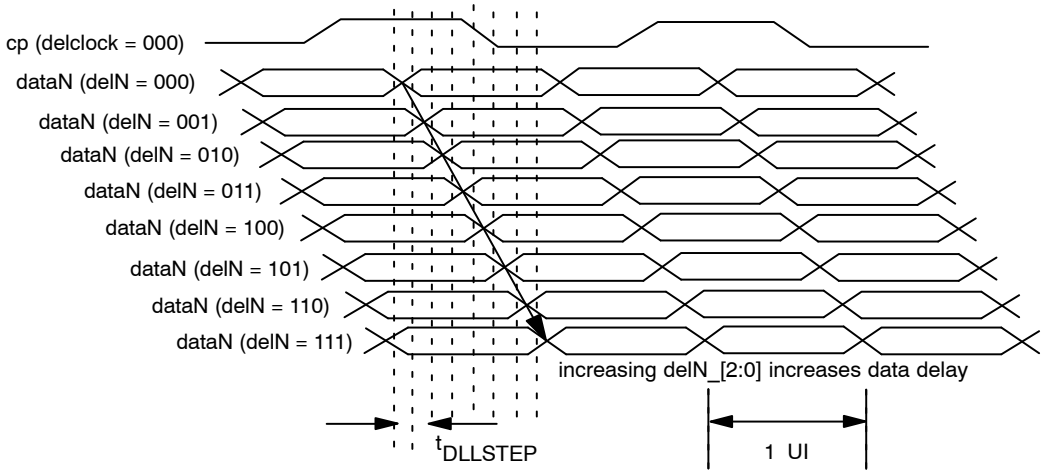



**Figure 11. Delaying data_lane with Respect to the clock_lane**

*HiSPi Streaming Mode Protocol Layer*

The protocol layer is positioned between the output data path of the sensor and the physical layer. The main functions of the protocol layer are generating sync codes, formatting pixel data, inserting horizontal/vertical blanking codes, and distributing pixel data over defined data lanes.

The HiSPi interface can only be configured when the sensor is in standby. This includes configuring the interface to transmit across 1, 2, or all 4 data lanes.

### Protocol Fundamentals

Referring to Figure 12, it can be seen that a SYNC code is inserted in the serial data stream prior to each line of image data. The streaming protocol will insert a SYNC code to transmit each active data line and vertical blanking lines.

The packetized protocol will transmit a SYNC code to note the start and end of each row. The packetized protocol uses sync a “Start of Frame” (SOF) sync code at the start of a frame and a “Start of Line” (SOL) sync code at the start of a line within the frame. The protocol will also transmit an “End of Frame” (EOF) at the end of a frame and an “End of Line” (EOL) sync code at the end of a row within the frame

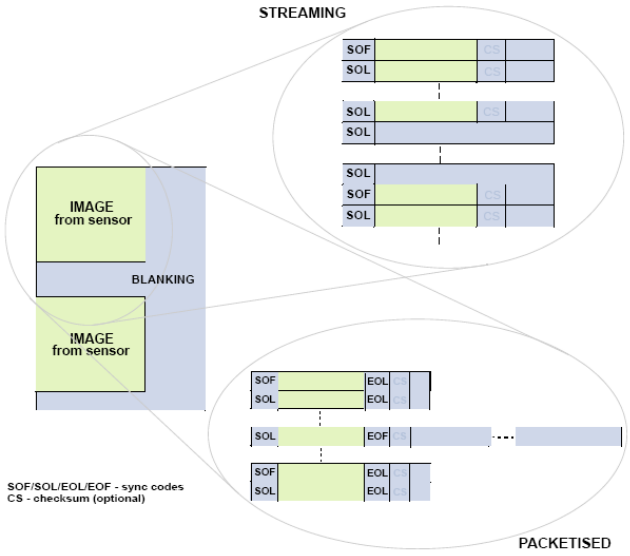



**Figure 12. Steaming vs. Packetized Transmission**

### Parallel Pixel Data Interface

MT9J003 image data is read out in a progressive scan. Valid image data is surrounded by horizontal blanking and vertical blanking, as shown in Figure 13. The amount of horizontal blanking and vertical blanking is programmable; LV is HIGH during the shaded region of the figure. FV timing is described in the "Output Data Timing (Parallel Pixel Data Interface)".

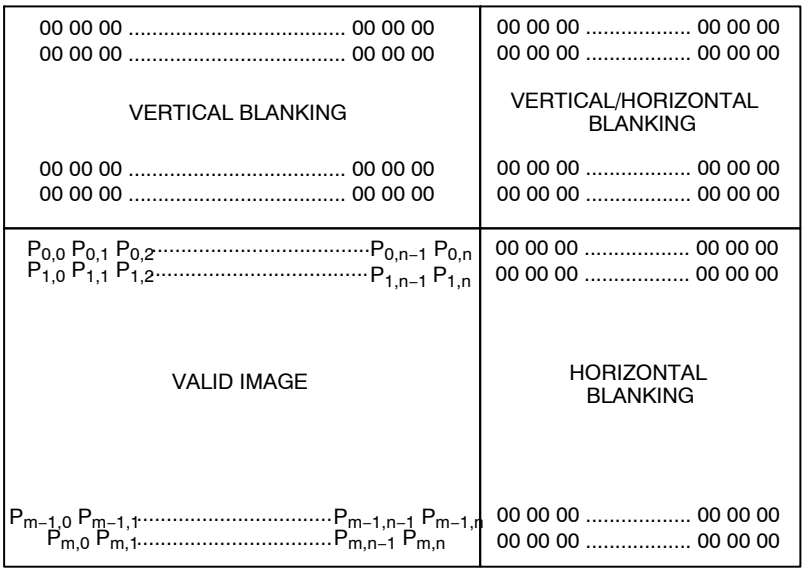



**Figure 13. Spatial Illustration of Image Readout**

### Output Data Timing (Parallel Pixel Data Interface)

MT9J003 output data is synchronized with the PIXCLK output. When LV is HIGH, one pixel value is output on the 12−bit DOUT output every PIXCLK period. The pixel clock frequency can be determined based on the sensor's master input clock and internal PLL configuration. The rising edges on the PIXCLK signal occurs one−half of a pixel clock period after transitions on LV, FV, and DOUT (see Figure 14). This allows PIXCLK to be used as a clock to sample the data. PIXCLK is continuously enabled, even during the blanking period. The MT9J003 can be programmed to delay the PIXCLK edge relative to the DOUT transitions. This can be achieved by programming the corresponding bits in the row_speed register. The parameters P, A, and Q in Figure 15 are defined in Table 4.

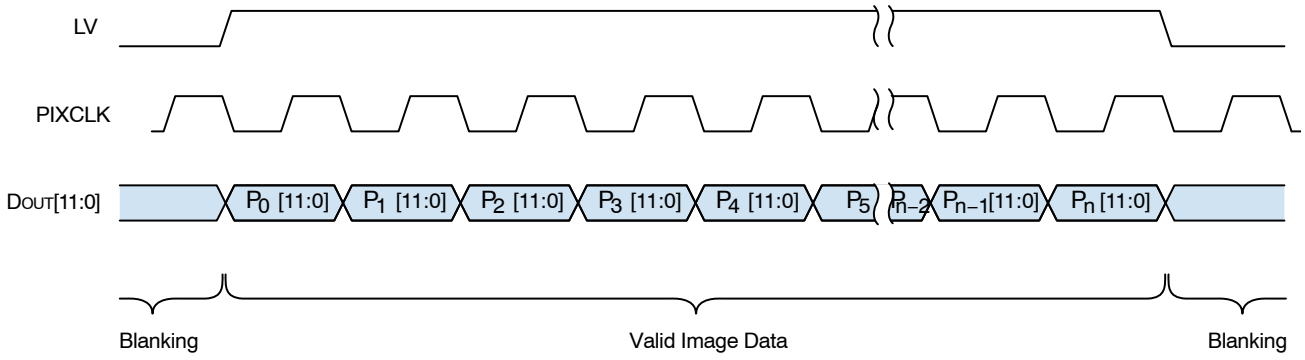



**Figure 14. Pixel Data Timing Example**

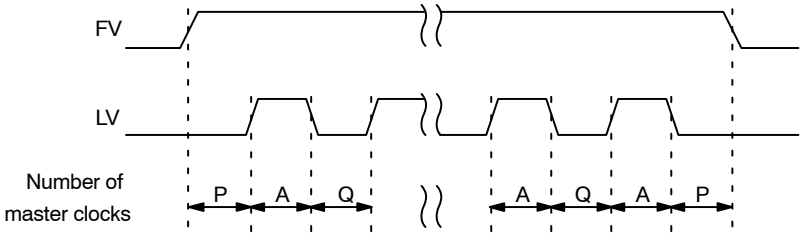



**Figure 15. Row Timing and FV/LV Signals**

The sensor timing (shown in Table 4) is shown in terms of pixel clock and master clock cycles (see Figure 14). The default settings for the on−chip PLL generate a pixel array clock (vt_pix_clk) of 160 MHz and an output clock (op_pix_clk) of 40 MHz given a 20 MHz input clock to the MT9J003. Equations for calculating the frame rate are given in “Frame Rate Control”.

**Table 4. ROW TIMING WITH HiSPi INTERFACE**

| Parameter | Name | Equation | Default Timing |
|---|---|---|---|
| PIXCLK_ PERIOD | Pixel Clock Period | 1/vt_pix_clk_freq_mhz | 1 Pixel Clock<br>= 6.25 ns |
| S | Skip (Subsampling) Factor | For x_odd_inc = y_odd_inc = 3, S = 2.<br>For x_odd_inc = y_odd_inc = 7, S = 4.<br>otherwise, S = 1<br>For y_odd_inc = 3, S = 2<br>For y_odd_inc = 7, S = 4<br>For y_odd_inc = 15, S = 8<br>For y_odd_inc = 31, S = 16<br>For y_odd_inc = 63, S = 32 | 1 |
| A | Active Data Time | (x_addr_end − x_addr_start + x_odd_inc) * 0.5 * PIXCLK_PERIOD/S<br>= 3775 − 112 + 12 | 1832 Pixel Clock<br>= 11.45 μs |
| P | Frame Start/end Blanking | 6 * PIXCLK_PERIOD | 6 Pixel Clock<br>= 37.5 ns |
| Q | Horizontal Blanking | (line_length_pck − A) * PIXCLK_PERIOD<br>= 3694 − 1832 | 1862 Pixel Clock<br>= 11.63 μs |
| A + Q | Row Time | line_length_pck * PIXCLK_PERIOD | 3694 Pixel Clock<br>= 23.09 μs |
| N | Number of Rows | (y_addr_end − y_addr_start + y_odd_inc) / S = (2755 − 8 + 1)/1 | 2748 Rows |
| V | Vertical Blanking | ((frame_length_lines − N) * (A + Q)) + Q − (2 * P)<br>= (2891 − 2748) * 3694 + 1862 − 12 | 530092 Pixel Clock<br>= 3.31 ms |
| T | Frame Valid Time | (N * (A + Q)) − Q + (2 * P)<br>= 2748*3694 − 1862 + 12 | 10149262 Pixel Clock<br>= 63.42 ms |
| F | Total Frame Time | line_length_pck * frame_length_lines * PIXCLK_PERIOD<br>= 2891 * 3694 | 10679354 Pixel Clock<br>= 66.75 ms |

**Table 5. ROW TIMING WITH PARALLEL INTERFACE**

| Parameter | Name | Equation | Default Timing |
|---|---|---|---|
| PIXCLK_ PERIOD | Pixel Clock Period | 1/vt_pix_clk_freq_mhz | 1 Pixel Clock = 6.25 ns |
| S | Skip (Subsampling) Factor | For x_odd_inc = y_odd_inc = 3, S = 2.<br>For x_odd_inc = y_odd_inc = 7, S = 4.<br>otherwise, S = 1<br>For y_odd_inc = 3, S = 2<br>For y_odd_inc = 7, S = 4<br>For y_odd_inc = 15, S = 8<br>For y_odd_inc = 31, S = 16<br>For y_odd_inc = 63, S = 32 | 1 |
| A | Active Data Time | (x_addr_end – x_addr_start + x_odd_inc) * 0.5 * PIXCLK_PERIOD/S<br>= (3775 – 112+1)/2 | 1832 Pixel Clocks<br>= 11.45 µs |
| P | Frame Start/end Blanking | 6 * PIXCLK_PERIOD | 6 Pixel Clocks<br>= 75 ns |
| Q | Array Horizontal Blanking | (line_length_pck – A) * PIXCLK_PERIOD = 7358 – 1832 | 5526 Pixel Clocks<br>= 34.5 µs<br>External horizontal blanking is 30 pixel clocks or 187 ns. |
| A + Q | Row Time Limited by Output Interface Speed | x_output_size * clk_pixel/clk_op + 30 = 3664 * 160 MHz/80 MHz + 30 | 7358 Pixel Clocks<br>= 46.1 µs |
| N | Number of Rows | (y_addr_end – y_addr_start + y_odd_inc) / S = (2755 – 8 + 1)/1 | 2748 rows |
| V | Vertical Blanking | ((frame_length_lines – N) * (A + Q)) + Q – (2 * P)<br>= (2891 – 2748)*7358 + 1862 – 12 | 1054044 Pixel Clocks<br>= 6.59 ms |
| T | Frame Valid Time | (N * (A + Q)) – Q + (2 * P) = 2748 * 7358 – 1862 + 12 | 20217934 Pixel Clocks<br>= 126.36 ms |
| F | Total Frame Time | line_length_pck * frame_length_lines * PIXCLK_PERIOD<br>= 2891 * 37358 | 21271978 Pixel Clocks<br>= 132.95 ms |

**Table 6. ROW TIMING WITH PARALLEL INTERFACE USING LOW POWER MODE**

| Parameter | Name | Equation | Default Timing |
|---|---|---|---|
| PIXCLK_ PERIOD | Pixel Clock Period | 1/vt_pix_clk_freq_mhz | 1 Pixel Clock<br>= 12.5 ns |
| S | Skip (Subsampling) Factor | For x_odd_inc = y_odd_inc = 3, S = 2.<br>For x_odd_inc = y_odd_inc = 7, S = 4.<br>otherwise, S = 1<br>For y_odd_inc = 3, S = 2<br>For y_odd_inc = 7, S = 4<br>For y_odd_inc = 15, S = 8<br>For y_odd_inc = 31, S = 16<br>For y_odd_inc = 63, S = 32 | 1 |
| A | Active Data Time | (x_addr_end – x_addr_start + x_odd_inc) * 0.5 * PIXCLK_PERIOD/S<br>= (3775–112+1)/2 | 1832 Pixel Clocks<br>= 22.9 µs |
| P | Frame Start/end Blanking | 6 * PIXCLK_PERIOD | 6 Pixel Clocks<br>= 75 ns |
| Q | Array Horizontal Blanking | (line_length_pck – A) * PIXCLK_PERIOD<br>= 3694 – 1832 | 1862 Pixel Clocks<br>= 23.2 µs<br>External horizontal blanking is 30 pixel clocks or 375 ns. |

**Table 6. ROW TIMING WITH PARALLEL INTERFACE USING LOW POWER MODE** (continued)

| Parameter | Name | Equation | Default Timing |
|---|---|---|---|
| A + Q | Row Time Limited by Output Interface Speed | x_output_size * clk_pixel/clk_op + 30<br>= 3664 * 80 MHz/80 MHz + 30 | 3694 Pixel Clocks<br>= 46.1 μs |
| N | Number of Rows | (y_addr_end – y_addr_start + y_odd_inc) / S<br>= (2755 – 8 + 1)/1 | 2748 Rows |
| V | Vertical Blanking | ((frame_length_lines – N) * (A + Q)) + Q – (2 * P)<br>= (2891 – 2748) * 7358 + 1862 – 12 | 530092 Pixel Clocks<br>= 6.63 ms |
| T | Frame Valid Time | (N * (A + Q)) – Q + (2 * P)<br>= 2748 * 3694 – 1862 + 12 | 10149262 Pixel Clocks<br>= 126.86 ms |
| F | Total Frame Time | line_length_pck * frame_length_lines * PIXCLK_PERIOD<br>= 2891 * 3694 | 10679354 Pixel Clocks<br>= 133.5 ms |

*Frame Rates at Common Resolutions*

Table 7 shows examples of register settings to achieve common resolutions and their frame rates.

**Table 7. REGISTER SETTINGS FOR COMMON RESOLUTIONS**

| Resolution | Interface | Frame Rate | Subsampling Mode | x_addr_start | x_addr_end | y_addr_start | y_addr_end |
|---|---|---|---|---|---|---|---|
| 3664x2748 (Full Resolution) | HiSPi | 14.7 fps | N/A | 112 | 3775 | 8 | 2755 |
| | Parallel | 7.5 fps | | | | | |
| 1920x1080 (1080p HDTV) | HiSPi | 59.94 fps | 2 x 2 Summing | 32 | 3873 | 296 | 2453 |
| | Parallel | 29.97 fps | | | | | |
| 1280x720 (720p HDTV) | HiSPi and Parallel | 59.94 fps | 2 x 2 Summing | 32 | 3873 | 296 | 2453 |
| 1408x792 + 10% EIS (720p HDTV + 10% EIS) | HiSPi and Parallel | 59.94 fps | 2 x 2 Summing | 624 | 3437 | 304 | 1885 |
| 640x480 (Low Power Monitor) | HiSPi and Parallel | 29.97 fps | Sum2Skip2 | 112 | 3769 | 8 | 2753 |

## TWO−WIRE SERIAL REGISTER INTERFACE

The two−wire serial interface bus enables read/write access to control and status registers within the MT9J003.

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

The protocols described in the two−wire serial interface specification allow the slave device to drive SCLKLOW; the MT9J003 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:

10. a (repeated) start condition
11. a slave address/data direction byte
12. an (a no−) acknowledge bit
13. a message byte
14. a stop condition

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

*Start Condition*

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

*Stop Condition*

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

*Data Transfer*

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

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

*Slave Address*

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 MT9J003 for the MIPI configured sensor are 0x6C (write address) and 0x6D (read address) in accordance with the MIPI specification. Alternate slave addresses of 0x6E (write address) and 0x6F (read address) can be selected by enabling and asserting the SADDR signal through the GPI pad. But for the CCP2 configured sensor, the default slave addresses used are 0x20 (write address) and 0x21 (read address) in accordance with the SMIA specification. Also, alternate slave addresses of 0x30 (write address) and 0x31 (read address) can be selected by enabling and asserting the SADDR signal through the GPI pad.

An alternate slave address can also be programmed through R0x31FC.

*Message Byte*

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

*Acknowledge Bit*

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

*No−Acknowledge Bit*

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

### Typical Sequence

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

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

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

### Single READ From Random Location

This sequence (Figure 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 16 shows how the internal register address maintained by the MT9J003 is loaded and incremented as the sequence proceeds.

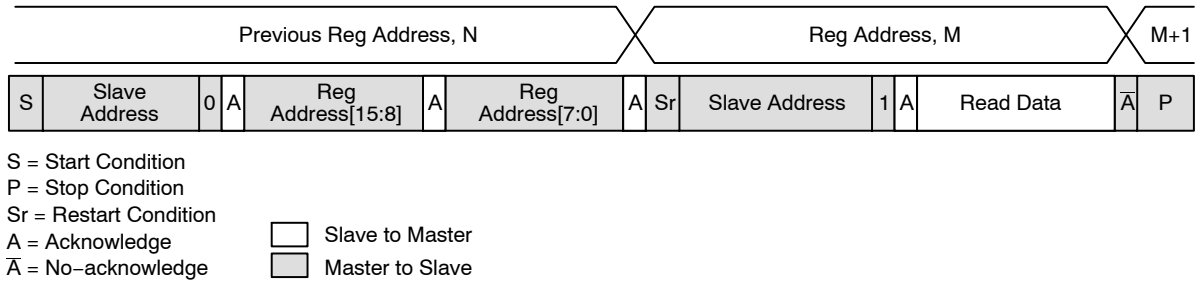



**Figure 16. Single READ from Random Location**

### Single READ From Current Location

This sequence (Figure 17) performs a read using the current value of the MT9J003 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.

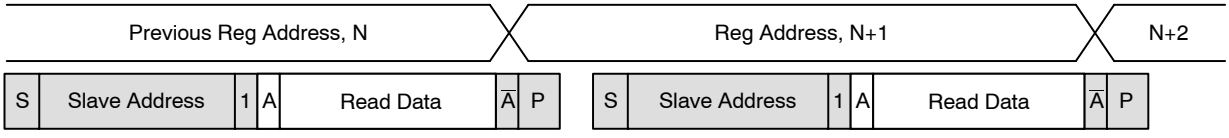



**Figure 17. Single READ from Current Location**

### Sequential READ, Start From Random Location

This sequence (Figure 18) starts in the same way as the single READ from random location (Figure 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.

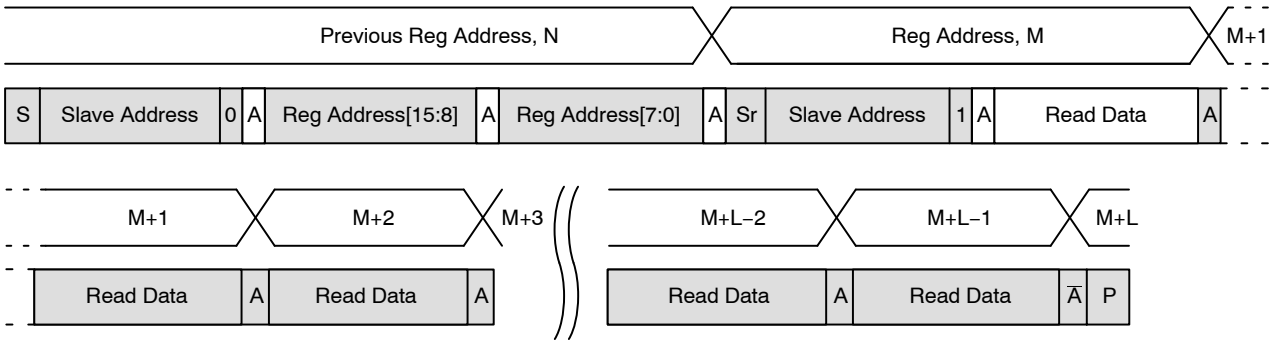



**Figure 18. Sequential READ, Start from Random Location**

### Sequential READ, Start From Current Location

This sequence (Figure 19) starts in the same way as the single READ from current location (Figure 17). Instead of generating a no−acknowledge bit after the first byte of data has been transferred, the master generates an acknowledge bit and continues to perform byte READs until “L” bytes have been read.

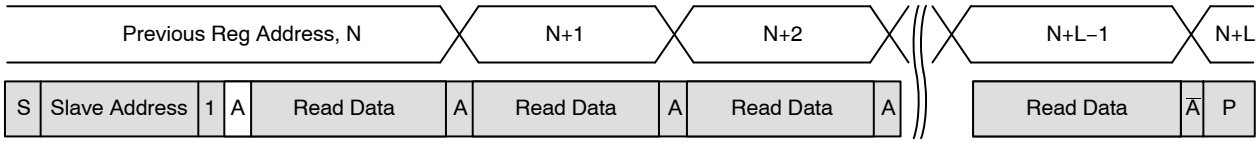



**Figure 19. Sequential READ, Start from Current Location**

### Single WRITE to Random Location

This sequence (Figure 20) 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.

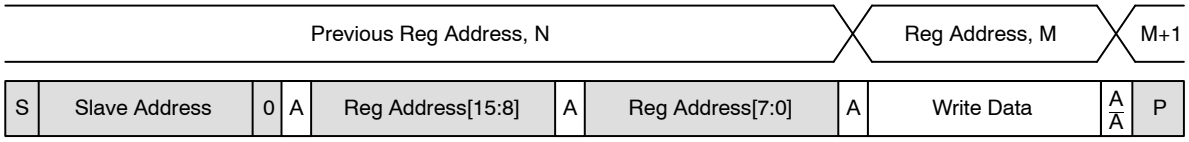



**Figure 20. Single WRITE to Random Location**

### Sequential WRITE, Start at Random Location

This sequence (Figure 21) starts in the same way as the single WRITE to random location (Figure 20). 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.

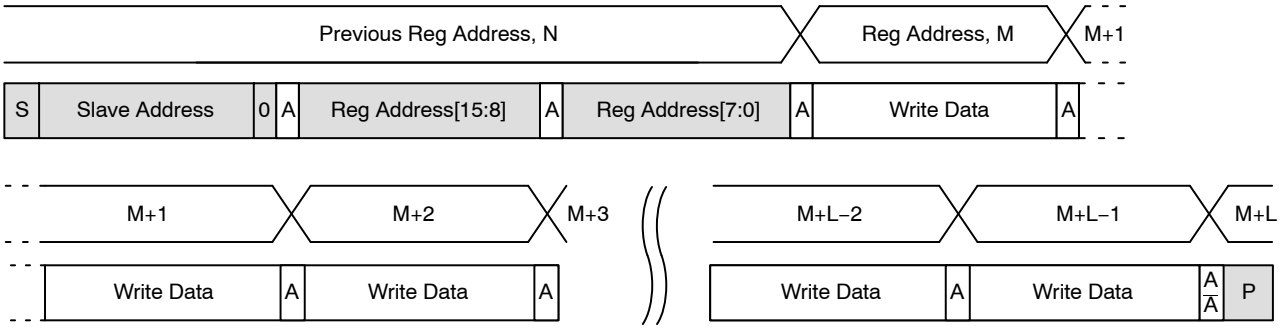



**Figure 21. Sequential WRITE, Start at Random Location**

## PROGRAMMING RESTRICTIONS

The following sections list programming rules that must be adhered to for correct operation of the MT9J003.

**Table 8. DEFINITIONS FOR PROGRAMMING RULES**

| Name | Definition |
|---|---|
| xskip | xskip = 1 if x_odd_inc = 1; xskip = 2 if x_odd_inc = 3; xskip = 4 if x_odd_inc = 7 |
| yskip | yskip = 1 if y_odd_inc = 1; yskip = 2 if y_odd_inc = 3; yskip = 4 if y_odd_inc = 7; yskip = 8 if y_odd_inc = 15; yskip = 16 if y_odd_inc = 31; yskip = 32 if y_odd_inc = 63 |

*X Address Restrictions*

The minimum column address available for the sensor is 24. The maximum value is 3879.

*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 MT9J003 will operate incorrectly if the x_output_size and y_output_size are significantly larger than the output image. To understand the reason for this, consider the situation where the sensor is operating at full resolution and the scaler is enabled with a scaling factor of 32 (half the number of pixels in each direction). This situation is shown in Figure 22.

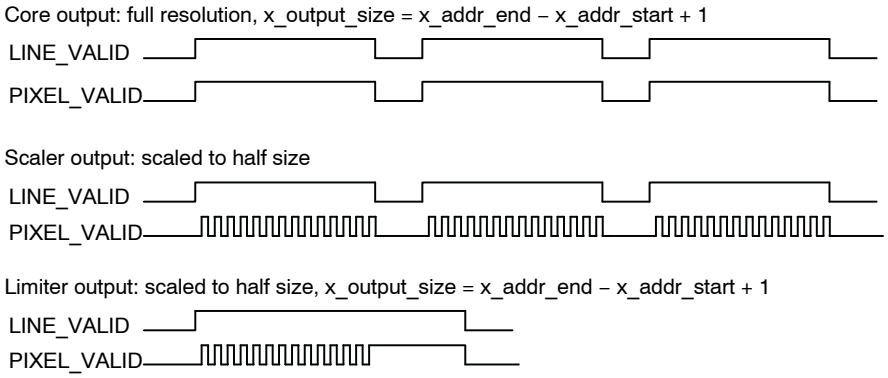



**Figure 22. Effect of Limiter on the Data Path**

In Figure 22, three different stages in the data path (see "Timing Specifications") 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 signaled 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 MT9J003 will cease to generate output frames.

A correct configuration is shown in Figure 23, in addition to showing the x_output_size reduced to match the output size of the scaler. In this configuration, the output of the limiter does not extend LV.

Figure 23 also shows the effect of the output FIFO, which forms the final stage in the data path. The output FIFO merges the intermittent pixel data back into a contiguous stream. Although not shown in this example, the output FIFO is also capable of operating with an output clock that is at a different frequency from its input clock.

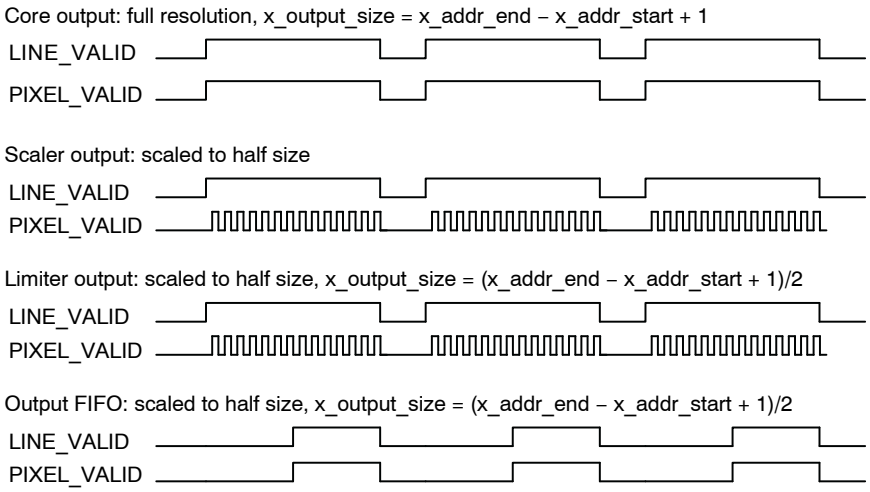



**Figure 23. Timing of Data Path**

*Output Data Timing*

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

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

A key constraint when configuring the clock for the output FIFO is that the frame rate out of the FIFO must exactly match the frame rate into the FIFO. When the scaler is disabled, this constraint can be met by imposing the rule that the row time on the serial data stream must be greater than or equal to the row time at the pixel array. The row time on the serial data stream is calculated from the x_output_size and the data_format (8, 10, or 12 bits per pixel), and must include the time taken in the serial data stream for start of frame/row, end of row/frame and checksum symbols.

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

*Changing Registers While Streaming*

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

- vt_pix_clk_div
- vt_sys_clk_div
- pre_pll_clk_div
- pll_multiplier
- op_pix_clk_div
- op_sys_clk_div

*Programming Restrictions When Using Global Reset*

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

## CONTROL OF THE SIGNAL INTERFACE

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

### Serial Register Interface

The serial register interface uses these signals:

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

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

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

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

This interface is described in detail in ”Two−Wire Serial Register Interface”.

### Parallel Pixel Data Interface

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

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

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

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

*Output Enable Control*

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

**Table 9. 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 10. CONFIGURATION OF THE PIXEL DATA INTERFACE**

| Serializer Disable R0x301 A−B[12] | Parallel Enable R0x301A−B[7] | Standby End−of−Frame R0x301A−B[4] | Description |
|---|---|---|---|
| 0 | 0 | 1 | Power up default.<br>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 MT9J003 are represented as a state diagram in Figure 24 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.

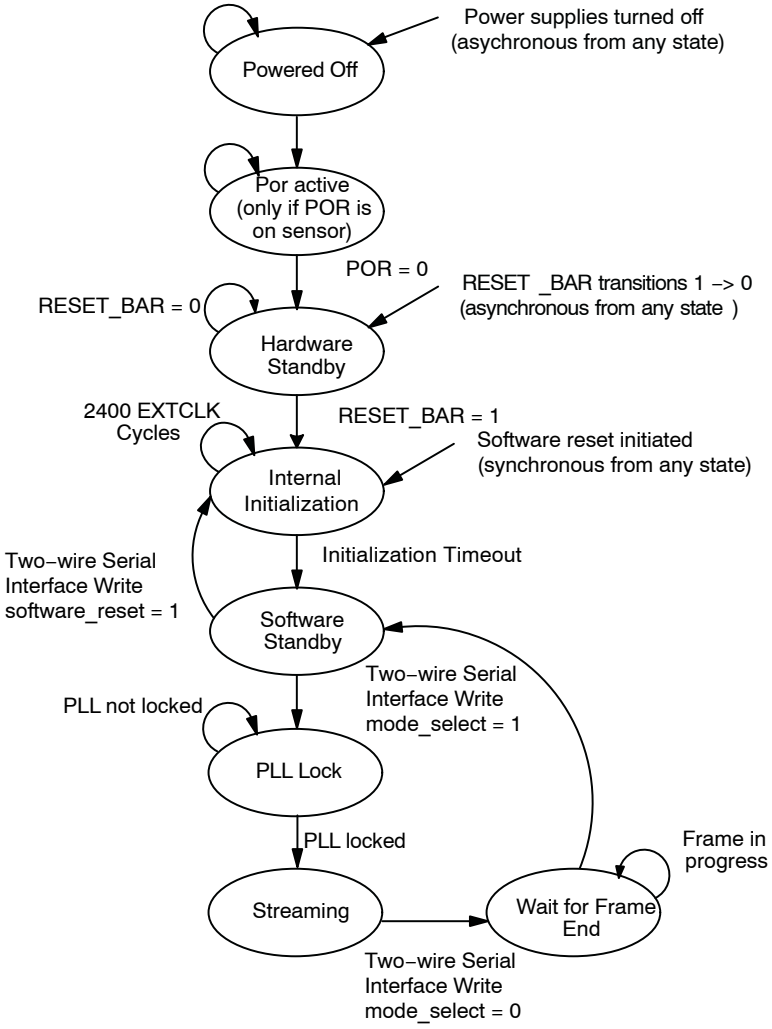



**Figure 24. MT9J003 System States**

**Table 11. RESET_BAR AND PLL IN SYSTEM STATES**

| State | EXTCLKs | PLL |
|---|---|---|
| Powered Off | – | VCO Powered Down (Note 1) |
| POR Active | – | |
| Hardware Standby | 0 | |
| Internal Initialization | 1 | |
| Software Standby | | |
| PLL Lock | | VCO Powering Up and Locking, PLL Output Bypassed |
| Streaming | | VCO Running, PLL Output Active |
| Wait for Frame End | | |

1. VCO = voltage−controlled oscillator.

### Power−On Reset Sequence

When power is applied to the MT9J003, it enters a low−power hardware standby state. Exit from this state is controlled by the later of two events:

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

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

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

When the sensor leaves the hardware standby state it performs an internal initialization sequence that takes 2400 EXTCLK cycles. After this, it enters a low−power software standby state. While the initialization sequence is in progress, the MT9J003 will not respond to READ transactions on its two−wire serial interface. Therefore, a method to determine when the initialization sequence has completed is to poll a sensor register; for example, R0x0000. While the initialization sequence is in progress, the sensor will not respond to its device address and READs from the sensor will result in a NACK on the two−wire serial interface bus. When the sequence has completed, READs will return the operational value for the register (0x2800 if R0x0000 is read).

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

### Soft Reset Sequence

The MT9J003 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 12. 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 | |
| GPI[3:0] | | Powered down. Can be left disconnected/floating | |
| TEST | | Enabled. Must be driven to a logic 0 for a serial CCP2−configured sensor, or 1 for a serial MIPI−configured sensor | |
| SCLK | | Enabled. Must be pulled up or driven to a valid logic level | |
| SDATA | I/O | Enabled as an input. Must be pulled up or driven to a valid logic level | |
| LINE_VALID | Output | High−Z. Can be left disconnected or floating | |
| FRAME_VALID | | | |
| DOUT[11:0] | | | |
| PIXCLK | | | |
| SLVS0_P | | | |
| SLVS0_N | | | |
| SLVS1_P | | | |
| SLVS1_N | | | |
| SLVS2_P | | | |
| SLVS2_N | | | |
| SLVS3_P | | | |
| SLVS3_N | | | |
| CLK_P | | | |
| CLK_N | | | |
| FLASH | | High−Z. | Logic 0. |
| SHUTTER | | | |

### General Purpose Inputs

The MT9J003 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”)
- Trigger (see the sections below)
- Standby functions
- SADDR selection (see “Serial Register Interface”)

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

### Streaming/Standby Control

The MT9J003 can be switched between its soft standby and streaming states under pin or register control, as shown in Table 13. Selection of a pin to use for the STANDBY function is described in “General Purpose Inputs”. The state diagram for transitions between soft standby and streaming states is shown in Figure 24.

**Table 13. STREAMING/STANDBY**

| Standby | Streaming R0x301A–B[2] | Description |
|---|---|---|
| Disabled | 0 | Soft Standby |
| Disabled | 1 | Streaming |
| – | 0 | Soft Standby |
| 0 | 1 | Streaming |
| 1 | – | Soft Standby |

### Trigger Control

When the global reset feature is in use, the trigger for the sequence can be initiated either under pin or register control, as shown in Table 14. Selection of a pin to use for the TRIGGER function is described in “General Purpose Inputs”.

**Table 14. TRIGGER CONTROL**

| Trigger | Global Trigger R0x3160–1[0] | Description |
|---|---|---|
| Disabled | 0 | Idle |
| Disabled | 1 | Trigger |
| 0 | 0 | Idle |
| – | 1 | Trigger |
| 1 | – | Trigger |

## PLL

The sensor contains a PLL for timing generation and control. The PLL contains a prescaler to divide the input clock applied on EXTCLK, a VCO to multiply the prescaler output, and a set of dividers to generate the output clocks. The clocking structure is shown in Figure 25.

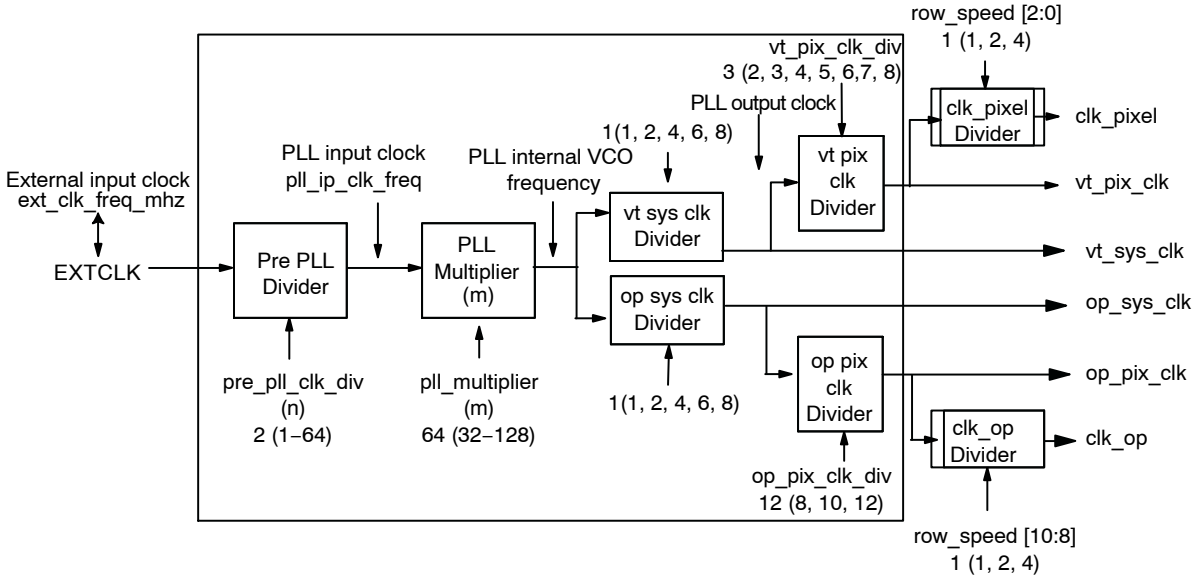



**Figure 25. Clocking Structure**

Figure 25 shows the different clocks and the register names. It also shows the default setting for each divider/multiplier control register, and the range of legal values for each divider/multiplier control register. The vt and op sys clk Divider is hardwired in the design. The PLL default settings support the HiSPi interface.

From the diagram, the clock frequencies can be calculated for the HiSPi interface using a 15 MHz input clock as follows:

Internal pixel clock used to readout the pixel array:

$$clk_pixel_freq_mhz = \frac{ext_clk_freq_mhz \times pll_multiplier}{pre_pll_clk_div \times vt_pix_clk_div \times row_speed\,[2:0]} = \frac{15\ MHz \times 64}{2 \times 3 \times 1} = 160\ MHz \quad \text{(eq. 1)}$$

The external pixel clock used to output the data:

$$clk_op_freq_mhz = \frac{ext_clk_freq_mhz \times pll_multiplier}{pre_pll_clk_div \times op_pix_clk_div \times row_speed\,[10:8]} = \frac{15\ MHz \times 64}{2 \times 12 \times 1} = 40\ MHz \quad \text{(eq. 2)}$$

Internal master clock:

$$\frac{vt_pix_clk_freq_mhz}{2} \quad \text{(eq. 3)}$$

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

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

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

- The minimum/maximum frequency limits for the associated clock must be met.
  - pll_ip_clk_freq must be in the range 6–48 MHz. Higher frequencies are preferred.
- PLL internal VCO frequency must be in the range 384–768 MHz.

  The minimum/maximum value for the divider/multiplier must be met.
  - Range for m: 32–128.
  - Range for (n): 1–64.
- The op_pix_clk must never run faster than the vt_pix_clk to ensure that the output data stream is contiguous.

  When using the HiSPi serial interface, the op_pix_clk must be 1/4 of the vt_pix_clk.
- The op_pix_clk_div divider must match the bit−depth of the image when using HiSPi. For example, op_pix_clk_div must be set to 12 for a 12−bit HiSPi output. The is not required when using the parallel interface.
- When using the parallel interface, the op_pix_clk must be half of the vt_pix_clk.
- The output line time (including the necessary blanking) must be output in a time equal to or less than the time defined by line_length_pck.

Although the PLL VCO input frequency range is advertised as 6–48 MHz, superior performance is obtained by keeping the VCO input frequency as high as possible.

The usage of the output clocks is shown below:

- clk_pixel is used by the sensor core to control the timing of the pixel array. The sensor core produces one 12−bit pixel each vt_pix_clk period. The line length (line_length_pck) and fine integration time (fine_integration_time) are controlled in increments of the clk_pixel period.
- clk_op is used to load parallel pixel data from the output FIFO. The output FIFO generates one pixel each op_pix_clk period.

Internal pixel clock used to readout the pixel array:

$$\text{clk_pixel_freq_mhz} = \frac{\text{ext_clk_freq_mhz} \times \text{pll_multiplier}}{\text{pre_pll_clk_div} \times \text{vt_pix_clk_div} \times \text{row_speed [2 : 0]}} = \frac{10\ \text{MHz} \times 64}{1 \times 2 \times 2} = 160\ \text{MHz} \quad \text{(eq. 4)}$$

The external pixel clock used to output the data:

$$\text{clk_op_freq_mhz} = \frac{\text{ext_clk_freq_mhz} \times \text{pll_multiplier}}{\text{pre_pll_clk_div} \times \text{op_pix_clk_div} \times \text{row_speed [10 : 8]}} = \frac{10\ \text{MHz} \times 64}{1 \times 1 \times 8} = 80\ \text{MHz} \quad \text{(eq. 5)}$$

### Programming the PLL Divisors

The PLL divisors must be programmed while the MT9J003 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 MT9J003 is in the streaming state is undefined.

An example of the parallel configuration for the PLL will uses an input clock of 10 MHz, an internal pixel clock of 160 MHz, and an output clock of 80 MHz. In this configuration:

- n = 1
- m = 64
- vt_sys_clk_div = 2
- op_sys_clk_div = 1
- vt_pix_clk_div = 2
- op_pix_clk_div = 8

### Clock Control

The MT9J003 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 MT9J003 enters a low−power state, almost all of the internal clocks are stopped. The only exception is that a small amount of logic is clocked so that the two−wire serial interface continues to respond to READ and WRITE requests.

## FEATURES

### Scaler

The MT9J003 sensor includes scaling capabilities. This allows the user to generate full field−of−view, low resolution images. Scaling is advantageous because it uses all pixel values to calculate the output image which helps to avoid aliasing. It is also more convenient than binning because the scale factor varies smoothly and the user is not limited to certain ratios of size resolution.

The scaling factor is programmable in 1/16 steps.

$$\text{ScaleFactor} = \frac{\text{scale_n}}{\text{scale_m}} = \frac{16}{\text{scale_m}} \qquad \text{(eq. 6)}$$

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

### Shading Correction

Lenses tend to produce images whose brightness is significantly attenuated near the edges. There are also other factors causing color plane nonuniformity in images captured by image sensors. The cumulative result of all these factors is known as image shading. The MT9J003 has an embedded shading correction module that can be programmed to counter the shading effects on each individual Red, GreenB, GreenR, and Blue color signal.

### The Correction Function

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

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

$$P_{corrected}(row, col) = P_{sensor}(row, col) \times f(row, col) \qquad \text{(eq. 7)}$$

- 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 Memory

The MT9J003 has a two−byte OTP memory that can be utilized during module manufacturing to store specific information about the module. This feature provides system integrators and module manufacturers the ability to label and distinguish various module types based on lens, IR−cut filter, or other properties.

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

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

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

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

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

1. Apply power to all the power rails of the sensor ($V_{DD}$, $V_{DD}$_IO, $V_{AA}$, $V_{AA}$_PIX, $V_{DD}$_PLL, and $V_{DD}$_TX0).
   - Set $V_{AA}$ to 3.1 V during OTP memory programming phase.
   - $V_{PP}$ needs to be floated during this phase.
   - Other supplies at nominal.
2. Provide 24 MHz EXTCLK clock input. The PLL settings are discussed at the end of the document.
3. Perform the proper reset sequence to the sensor.
4. Place the sensor in soft standby (sensor default state upon power−up) or ensure the streaming is turned OFF when the part is in active mode.

5. VPP ramps to 8.5 V in preparation to program. Power supply (VPP) slew rate should be slower than 1 V/µs.
6. Program R0x3052 to the value 0x045C.
7. Program R0x3054 to the value 0XEA99.
8. Write the 16 bit word data by programming R0x304C.
9. Initiate the OTP memory programming process by programming R0x304A[0] to the value 0x0001.
10. Check R0x304A [2] = 1, until bit is set to “1” to check for program completion.
11. Repeat steps 9 and 10 two more times.
12. Remove high voltage and float VPP pin.
13. Power down the sensor.
14. Apply nominal power to all the power rails of the sensor VDD, VDD_IO, VAA, $V_{AA}$_PIX and VDD_PLL). VPP must be floated.
15. Set EXTCLK to normal or customer defined operating frequency.
16. Perform the proper reset sequence to the sensor.
17. Initiate the OTP memory reading process by setting R0x304A[4] to the value 0x0010.
18. Poll the register bit R0x304A[6] until bit set to “1” to check for read completion.
19. Read the 16 bit word data from the R0x304E.

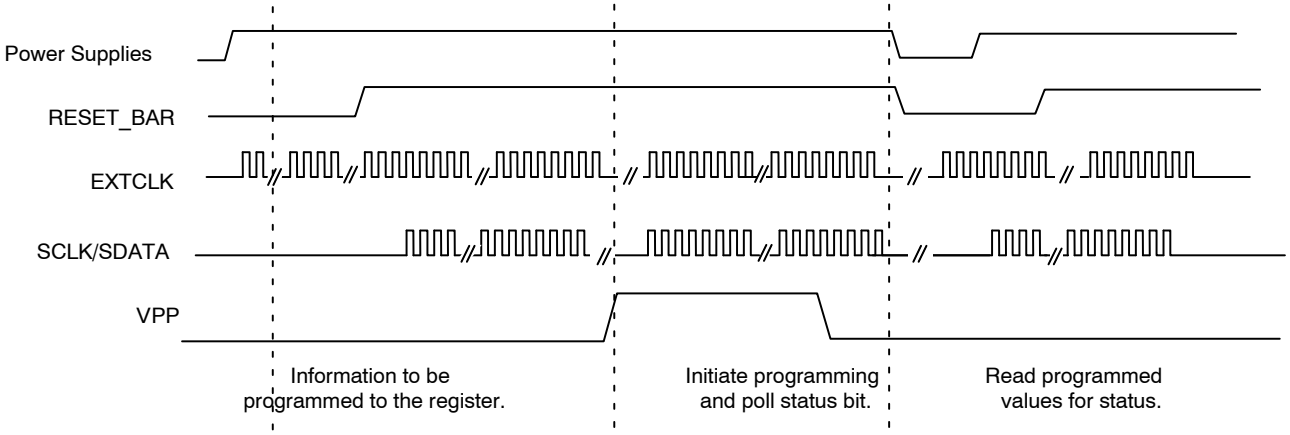



**Figure 26. Sequence for Programming the MT9J003**

## SENSOR READOUT CONFIGURATION

### Image Acquisition Modes

The MT9J003 supports two image acquisition modes:

1. Electronic rolling shutter (ERS) mode:
   This is the normal mode of operation. When the MT9J003 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 MT9J003 switches cleanly from the old integration time to the new while only generating frames with uniform integration. See “Changes to Integration Time” in the MT9J003 Register Reference.
2. Global reset mode:
   This mode can be used to acquire a single image at the current resolution. In this mode, the end point of the pixel integration time is controlled by an external electromechanical shutter, and the MT9J003 provides control signals to interface to that shutter. The operation of this mode is described in detail in ”Global Reset”.

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

### Window Control

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

### Pixel Border

The default settings of the sensor provide a 3840 (H) x 2748 (V) image. A border of up to 8 pixels (4 in binning) on each edge can be enabled by reprogramming the x_addr_start, y_addr_start, x_addr_end, y_addr_end, x_output_size, and y_output_size registers accordingly. This provides a total active pixel array of 3856 (H) x 2764 (V) including border pixels.

### 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 27 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.

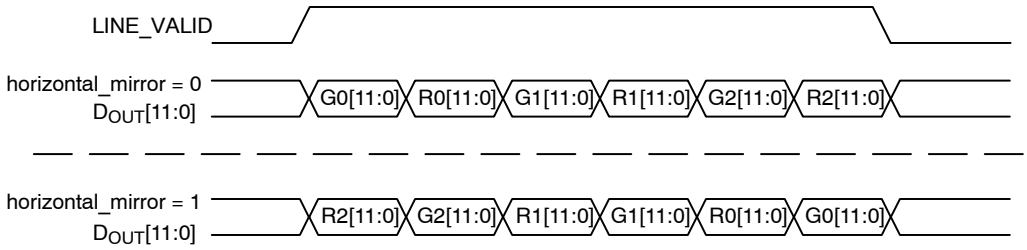



**Figure 27. Effect of Horizontal Mirror on Readout Order**

*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 28 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.

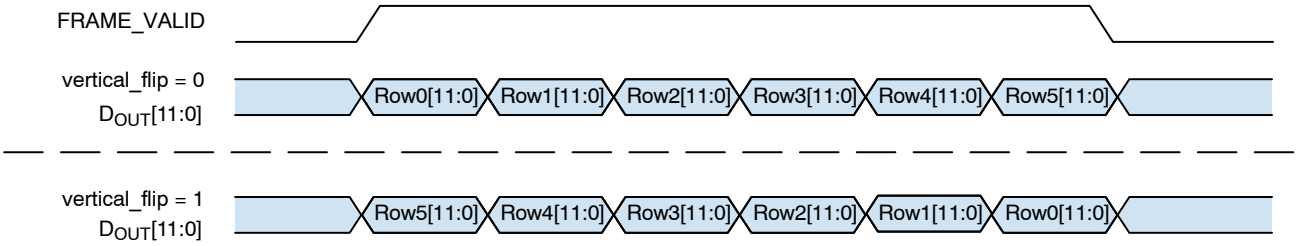



**Figure 28. Effect of Horizontal Mirror on Readout Order**

*Subsampling*

The MT9J003 supports subsampling. This feature allows the sensor to read out a sample of pixels available on the array. The most common subsampling used is either a 2x2 or 4x4 where every 2nd or 4th pixel is read in the x and y direction.

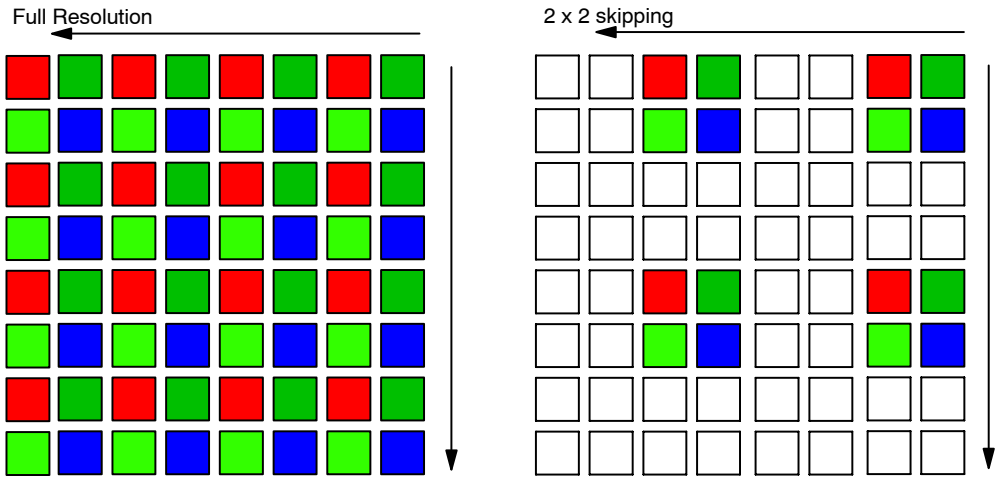



**Figure 29. Pixel Array Readout Without Subsampling and With 2x2 Skipping**

Pixel skipping can be configured up to 4x in the x−direction and 32x in the y−direction. Skipping pixels in the x−direction will reduce the row−time while skipping in the y−direction will reduce the number of rows readout from the sensor. Skipping in both directions will reduce the frame−time and is a common method used to increase the sensor frame−rate. Skipping will introduce image artifacts from aliasing.

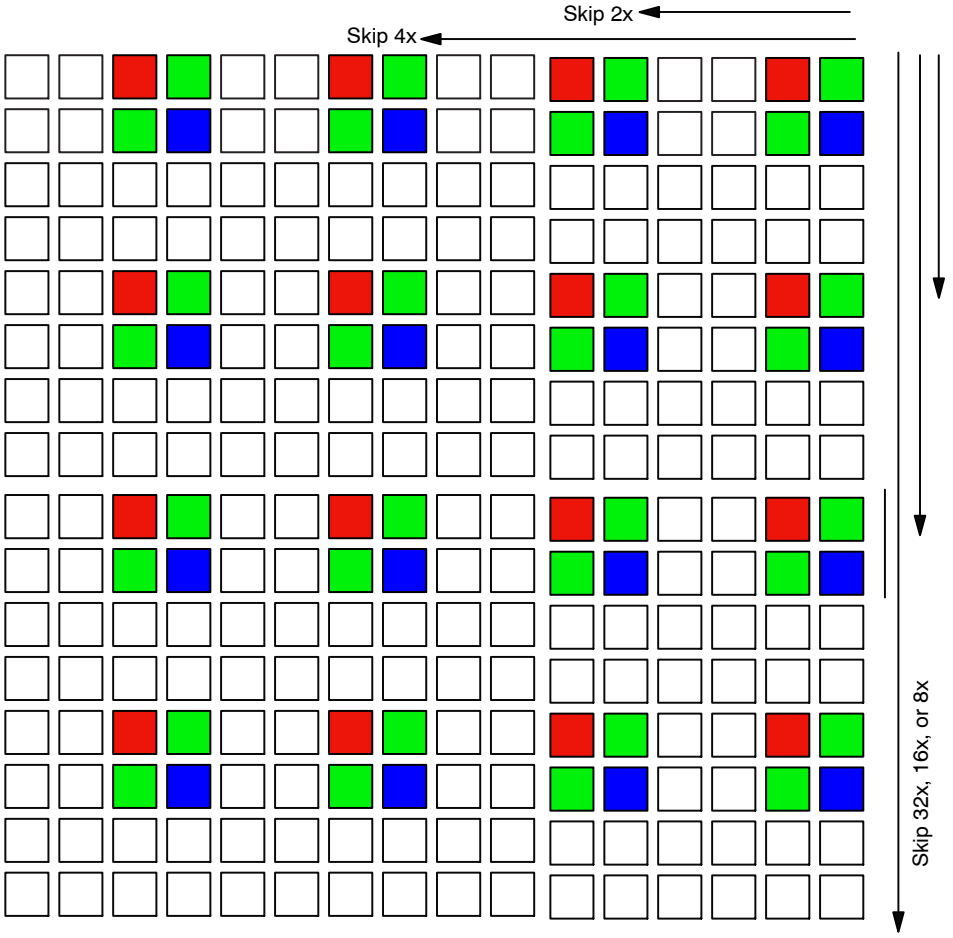



**Figure 30. Combinations of Pixel Skipping in the MT9J003 Sensor**

The subsampling feature can also bin or sum the skipped pixels. Pixel binning will sample pixels and average the value together in the analog domain. Summing will add the charge or voltage values of the neighboring pixels together.

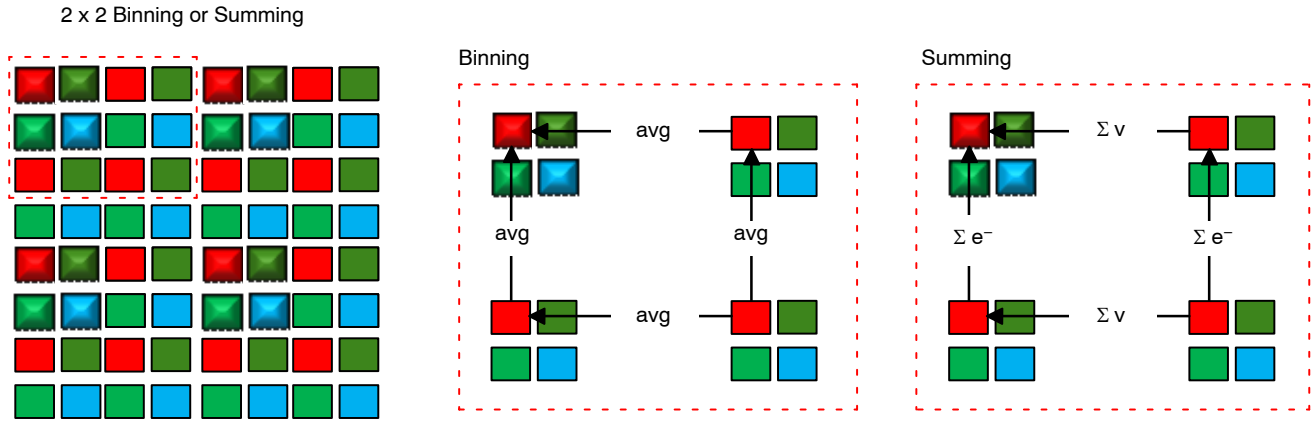



**Figure 31. Pixel Binning and Summing**

The pixel summing must be done with adjacent pixels within the same color plane. The pixel binning can be configured to combine adjacent pixels or to combine every other pixel.

The pixel subsampling can be configured as a combination of skipping and binning or summing. This type of subsampling is typically used to achieve the best combination of pixel responsivity and frame rate. The summing and skipping implementation will sum neighboring pixels on the same color plane and skip over the adjacent group of pixels.

Figure 32 shows that neighboring pixels are summed together. In the case that a subsampling factor of 4x or greater is used with summing, the neighboring pixels will also be summed together.

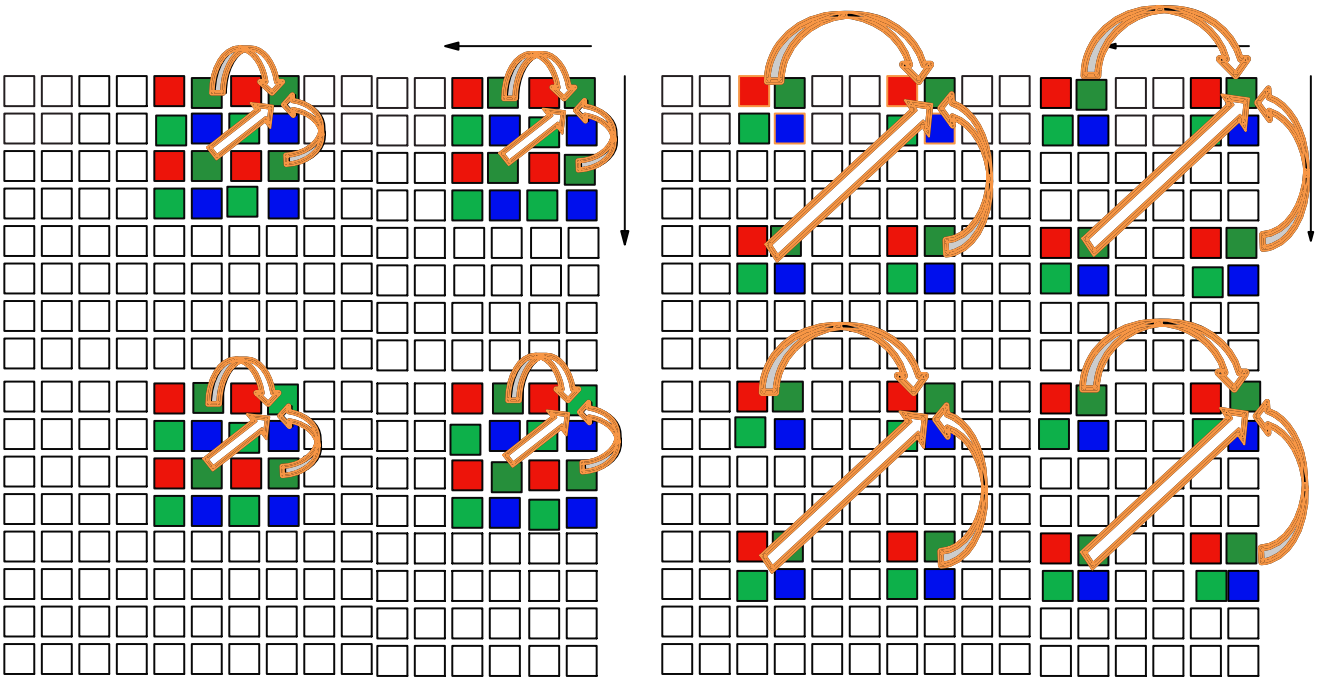


**Figure 32. Pixel Skipping Combined with Summing or Binning**

Table 15 shows the different combinations of subsampling available with the MT9J003 sensor. The sensor cannot combine pixels using two different methods in the same direction. This means that bin−xy and sum−y are not valid combinations with the sensor. As well, the bin−xy is limited to a skip of 4x in the vertical direction.

**Table 15. SUBSAMPLING COMBINATIONS**

| Skip Y | Skip X | Bin X | Bin XY | Sum X | Sum XY |
|---|---|---|---|---|---|
| 1 | 1 | – | – | – | – |
| | 2 | Y | – | Y | – |
| | 4 | Y | – | Y | – |
| 2 | 1 | – | – | – | Y |
| | 2 | Y | Y | Y | Y |
| | 4 | Y | Y | Y | Y |
| 4 | 1 | – | – | – | Y |
| | 2 | Y | Y | Y | Y |
| | 4 | Y | Y | Y | Y |
| 8 | 1 | – | – | – | Y |
| | 2 | Y | – | Y | Y |
| | 4 | Y | – | Y | Y |
| 16 | 1 | – | – | – | Y |
| | 2 | Y | – | Y | Y |
| | 4 | Y | – | Y | Y |
| 32 | 1 | – | – | – | Y |
| | 2 | Y | – | Y | Y |
| | 4 | Y | – | Y | Y |

### Frame Rate Control

The frame−time is calculated as the row−time multiplied by the number of rows (frame_length_lines). The row−time is referred to in these calculations as the number of pixel clocks read per row (line_length_pck) multiplied by the vt_pix_clk frequency.

The formulas for calculating the frame rate of the MT9J003 are shown below.

The line length is programmed in pixel clock periods through the register line_length_pck. The minimum value can be determined as the largest value found in Equation 8. These are the required values for either the array readout or the bandwidth available to the parallel or serial interface.

*Absolute Minimum Array Line Length Pck*

minimum line_length_pck = min_line_length_pck (see Table 16, “Minimum Row Time and Blanking Numbers”)

*Array Readout Line Length Pck*

$$\left[\frac{x_addr_end - x_addr_start + x_odd_inc}{2x\left(\frac{x_odd_inc + 1}{2}\right)} + min_line_blanking_pck\right] \quad \text{(eq. 8)}$$

*Interface Line Length Pck*

$$x_output_size\left(\frac{op_pix_clk}{vt_pix_clk}\right) + 30 \text{ (For Parallel)} \quad \text{(eq. 9)}$$

$$\left(\frac{x_output_size}{4}\right) \times \frac{op_pix_clk}{vt_pix_clk} + 30 \text{ (ForHiSPi)} \quad \text{(eq. 10)}$$

Note that line_length_pck will be the maximum of the three equations. The second equation describes the limitations from the readout of the pixel array while the third determines the frame−rate of the output interface. The frame−rate using HiSPi will always be higher than using the parallel interface. Values for min_line_blanking_pck are provided in “Minimum Row Time”.

The frame length is programmed directly in number of lines in the register frame_line_length. For a specific window size, the minimum frame length is shown in Equation 11:

$$minimumframe_length_lines = \left(\frac{y_addr_end - y_addr_start + 1}{subsampling\ factor} + min_frame_blanking_lines\right) \quad \text{(eq. 11)}$$

The frame rate can be calculated from these variables and the pixel clock speed as shown in Equation 12:

$$\text{frame rate} = \frac{\text{vt_pixel_clock_mhz} \times 1 \times 10^6}{\text{line_length_pck_xframe_length_lines}} \quad \text{(eq. 12)}$$

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

*Minimum Row Time*

The minimum row time and blanking values with default register settings are shown in Table 16.

**Table 16. MINIMUM ROW TIME AND BLANKING NUMBERS**

| Register | No Row Binning | | | Row Binning | | |
|---|---|---|---|---|---|---|
| row_speed[2:0] | 1 | 2 | 4 | 1 | 2 | 4 |
| min_line_blanking_pck | 0x046E | 0x029A | 0x01B0 | 0x0822 | 0x046C | 0x0292 |
| min_line_length_pck | 0x0670 | 0x03E0 | 0x02F0 | 0x0CC0 | 0x0660 | 0x03D8 |

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

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

1. line_length_pck> min_line_length_pck in Table 16.
2. line_length_pck > 0.5 * (x_addr_end – x_addr_start + x_odd_inc)/((1+x_odd_inc)/2) + min_line_blanking_pck in Table 16.
3. The row time must allow the FIFO to output all data during each row.

Parallel – line_length_pck > (x_output_size) * “vt_pix_clk period” / “op_pix_clk period” + 0x005E.

HiSPi (4−lane) – line_length_pck > (1/4)*(x_output_size) * “vt_pix_clk period” / “op_pix_clk period” + 0x005E.

*Minimum Frame Time*

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

**Table 17. MINIMUM FRAME TIME AND BLANKING NUMBERS**

| Register | |
|---|---|
| min_frame_blanking_lines | 0x008F |
| min_frame_length_lines | 0x0091 |

*Fine Integration Time Limits*

The limits for the fine_integration_time can be found from fine_integration_time_min and fine_integration_time_max_margin. Values for different mode combinations are shown in Table 18.

**Table 18. FINE_INTEGRATION_TIME LIMITS**

| Register | No Row Binning | | | Row Binning | | |
|---|---|---|---|---|---|---|
| row_speed[2:0] | 1 | 2 | 4 | 1 | 2 | 4 |
| fine_integration_time_min | 0x03F2 | 0x020A | 0x094 | 0x07B2 | 0x03AE | 0x010C |
| fine_integration_time_max_margin | 0x027E | 0x012E | 0x0108 | 0x050E | 0x0276 | 0x0224 |

*Fine Correction*

For the fine_integration_time limits, the fine_correction constant will change with the pixel clock speed and binning mode. These values are shown in Table 19.

**Table 19. FINE_CORECTION VALUES**

| Register | No Row Binning | | | Row Binning | | |
|---|---|---|---|---|---|---|
| row_speed[2:0] | 1 | 2 | 4 | 1 | 2 | 4 |
| fine_correction | 0x09C | 0x048 | 0x01E | 0x0134 | 0x094 | 0x044 |

### Low Power Mode

The MT9J003 sensor supports a low power mode, which can be entered by programming register bit read_mode[9]. Setting this bit will do the following:

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

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

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

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

### Integration Time

The integration (exposure) time of the MT9J003 is controlled by the fine_integration_time and coarse_integration_time registers.

The limits for the fine integration time are defined by:

$$\text{fine_integration_time_min} \le \text{fine_integration_time} \le (\text{line_length_pck} - \text{fine_integration_time_max_margin}) \quad \text{(eq. 13)}$$

The limits for the coarse integration time are defined by:

$$\text{coarse_integration_time_min} < \text{coarse_integration_time} \quad \text{(eq. 14)}$$

The actual integration time is given by:

$$\text{integration_time} = \frac{((\text{coarse_integration_time} \times \text{line_length_pck}) + \text{fine_integration_time})}{(\text{vt_pix_clk_freq_mhz} \times 10^6)} \quad \text{(eq. 15)}$$

It is required that:

$$\text{coarse_integration_time} <= (\text{frame_length_lines} - \text{coarse_integration_time_max_margin}) \quad \text{(eq. 16)}$$

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

### onsemi Gain Model

The **onsemi** gain model uses color−specific registers to control both analog and digital gain to the sensor. These registers are:

- global_gain
- greenR_gain
- red_gain
- blue_gain
- greenB_gain

The registers provide three 2x and one 4x analog gain stages. The first analog gain stage has a granularity of 64 steps over 2x gain. A digital gain from 1−7x can also be applied.

$$\text{analog gain} = (8/(8 - g(\text{colamp}) < 11:9 >)x(1 + \text{color_gain}[8])(1 + \text{color_gain}[7])(\text{color_gain}[6:0]/64) \quad \text{(eq. 17)}$$

Bits 11 to 9 are also restricted to 0, 4, and 6. This limits the particular gain stage to 4x.

As a result of the different gain stages, analog gain levels can be achieved in different ways. The recommended gain sequence is shown below in Table 20.

**Table 20. RECOMMENDED GAIN STAGES**

| Desired Gain | Recommended Gain Register Setting |
|---|---|
| 1–1.98 | 0x1040–0x107F |
| 2–3.97 | 0x1840–0x187F |
| 4–7.94 | 0x1C40–0x1C7F |
| 8–15.875 | 0x1CC0–0x1CFF |
| 16–31.75 | 0x1DC0–0x1DFF |

### Flash Control

The MT9J003 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 33, and in Figures 34 and 35. The flash and flash_count registers allow the timing of the flash to be changed. The flash can be programmed to fire only once, delayed by a few frames when asserted, and (for xenon flash) the flash duration can be programmed.

Enabling the LED flash will cause one bad frame, where several of the rows only have the flash on for part of their integration time. This can be avoided either by first enabling mask bad frames (write reset_register[9] = 1) before the enabling the flash or by forcing a restart (write reset_register[1] = 1) immediately after enabling the flash; the first bad frame will then be masked out, as shown in Figure 35. Read−only bit flash[14] is set during frames that are correctly integrated; the state of this bit is shown in Figures 33, 34, and 35.

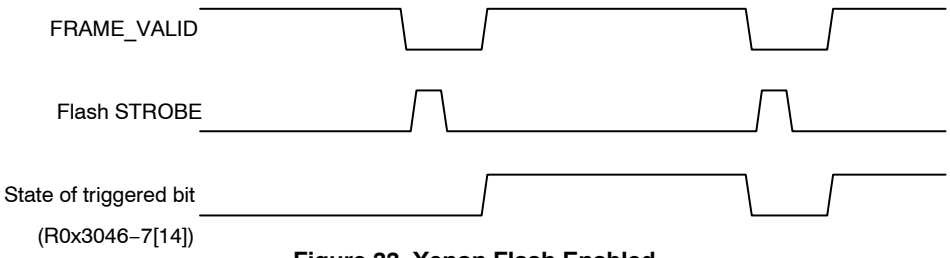
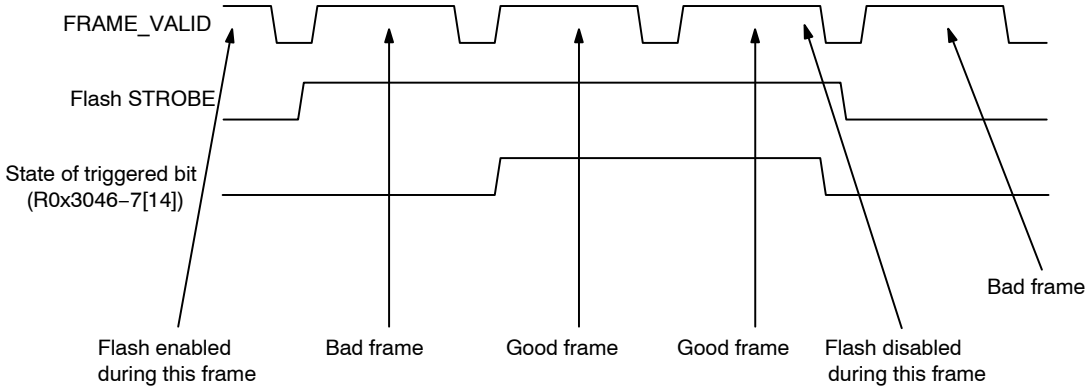



**Figure 33. Xenon Flash Enabled**




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

**Figure 34. LED Flash Enabled**

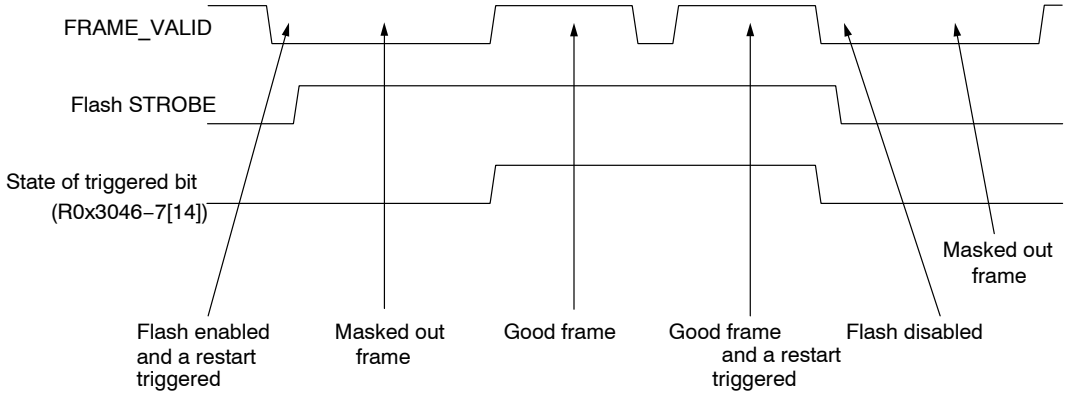



**Figure 35. LED Flash Enabled Following Forced Restart**

### Global Reset

Global reset mode allows the integration time of the MT9J003 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.

*Overview of Global Reset Sequence*

The basic elements of the global reset sequence are:

1. By default, the sensor operates in ERS mode and the SHUTTER output signal is LOW. The electromechanical shutter must be open to allow light to fall on the pixel array. Integration time is controlled by the coarse_integration_time and fine_integration_time registers.
2. A global reset sequence is triggered.
3. All of the rows of the pixel array are placed in reset.
4. All of the rows of the pixel array are taken out of reset simultaneously. All rows start to integrate incident light. The electromechanical shutter may be open or closed at this time.
5. If the electromechanical shutter has been closed, it is opened.
6. After the desired integration time (controlled internally or externally to the MT9J003), 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 36. The following sections expand to show how the timing of this sequence is controlled.

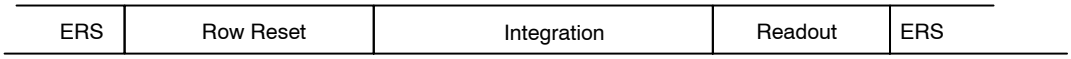



**Figure 36. Overview of Global Reset Sequence**

*Entering and Leaving the Global Reset Sequence*

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

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

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

((13 + coarse_integration_time) * line_length_pck)

This sequence is shown in Figure 37.

While operating in ERS mode, double−buffered registers are updated at the start of each frame in the usual way. During the global reset sequence, double−buffered registers are updated just before the start of the readout phase.

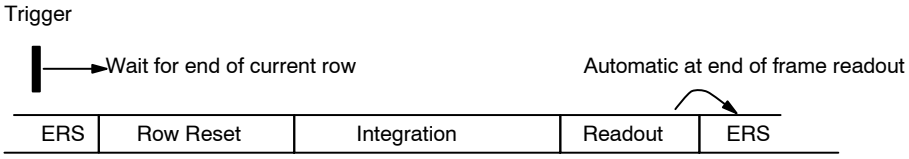



**Figure 37. Entering and Leaving a Global Reset Sequence**

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

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

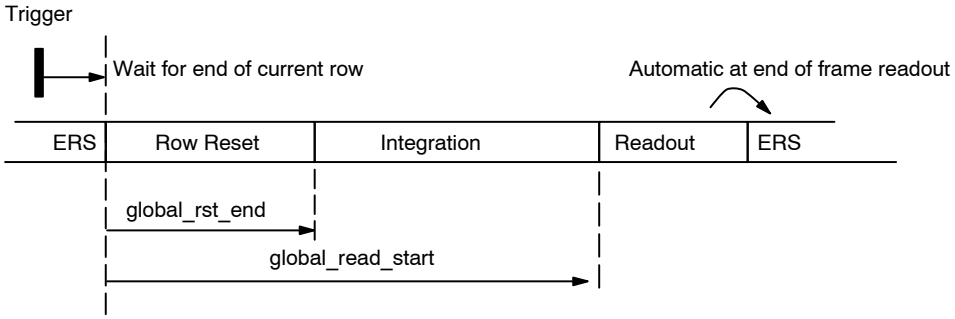



**Figure 38. Controlling the Reset and Integration Phases of the Global Reset Sequence**

*Control of the Electromechanical Shutter*

Figure 39 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.

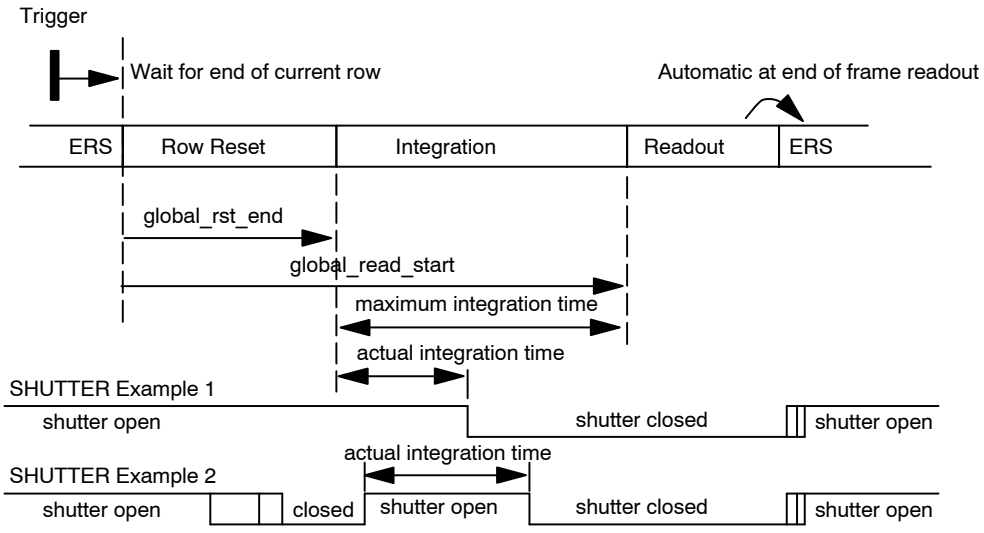



**Figure 39. Control of the Electromechanical Shutter**

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 MT9J003 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 40. SHUTTER is de−asserted by default. The point at which it asserts is controlled by the programming of global_shutter_start. At the end of the global reset readout phase, SHUTTER de−asserts approximately 2 * line_length_pck after the de−assertion of FV.

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

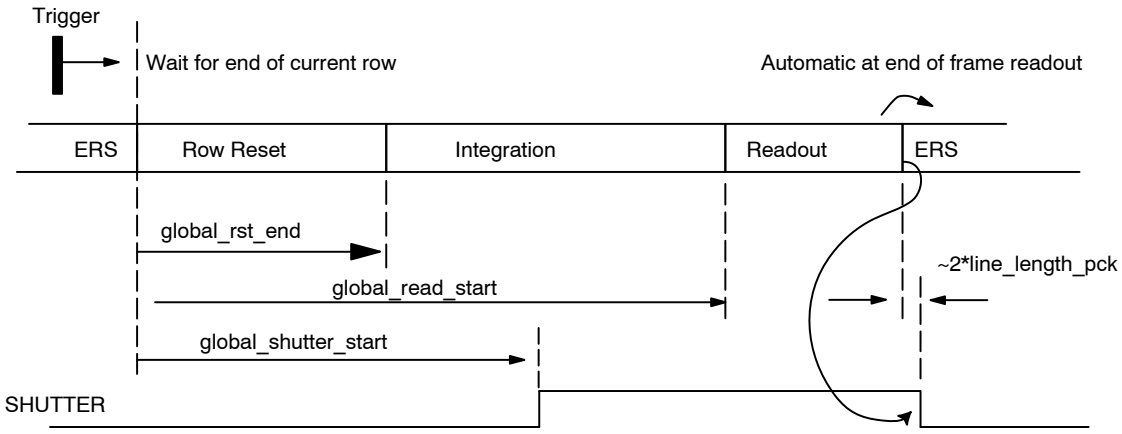



**Figure 40. Controlling the SHUTTER Output**

*Using FLASH with Global Reset*

If global_seq_trigger[2] = 1 (global flash enabled) when a global reset sequence is triggered, the FLASH output signal will be pulsed during the integration phase of the global reset sequence. The FLASH output will assert a fixed number of cycles after the start of the integration phase and will remain asserted for a time that is controlled by the value of the flash_count register, as shown in Figure 41.

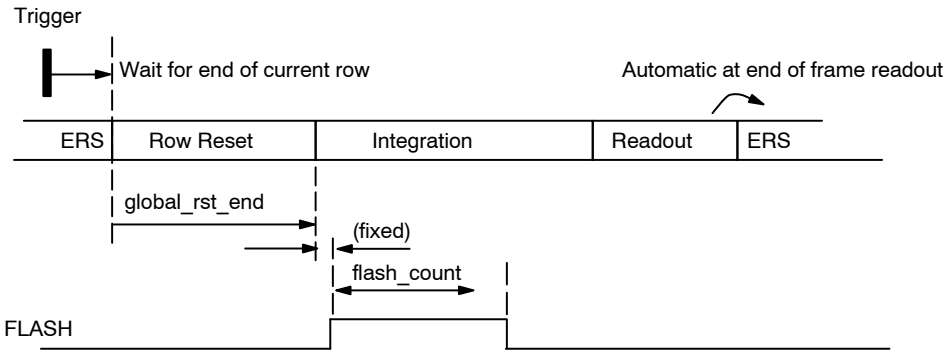



**Figure 41. Using FLASH With Global Reset**

*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.

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 42. The figure shows the global reset sequence being triggered by the GPI2 input, but it could be triggered by any of the GPI inputs or by the setting and subsequence clearing of the global_seq_trigger[0] under software control.

The integration time of the GRR sequence is defined as:

$$\text{IntegrationTime} = \frac{\text{global_scale} \times [\text{global_read_start} - \text{global_shutter_start} - \text{global_rst_end}]}{\text{vt_pix_clk_freq_mhz}} \quad \text{(eq. 18)}$$

Where:

$$\text{global_read_start} = (2^{16} \times \text{global_read_start2}[7:0] + \text{global_read_start1}[15:0] \quad \text{(eq. 19)}$$

$$\text{global_shutter_start} = (2^{16} \times \text{global_shutter_start2}[7:0] + \text{global_shutter_start1}[15:0] \quad \text{(eq. 20)}$$

The integration equation allows for 24−bit precision when calculating both the shutter and readout of the image. The global_rst_end has only 16−bit as the array reset function and requires a short amount of time.

The integration time can also be scaled using global_scale. The variable can be set to 0–512, 1–2048, 2–128, and 3–32.

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

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

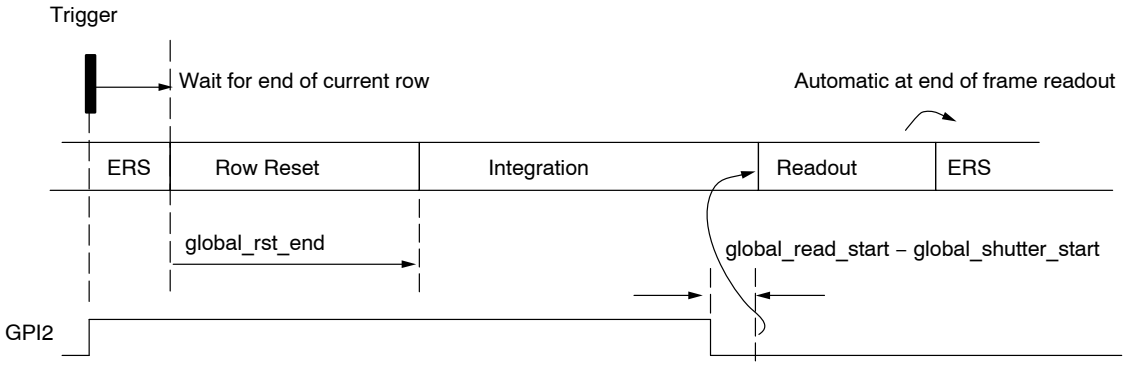



**Figure 42. Global Reset Bulb**

*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.

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 MT9J003 will remain in streaming state until the global reset sequence (including frame readout) has completed, as shown in Figure 43.

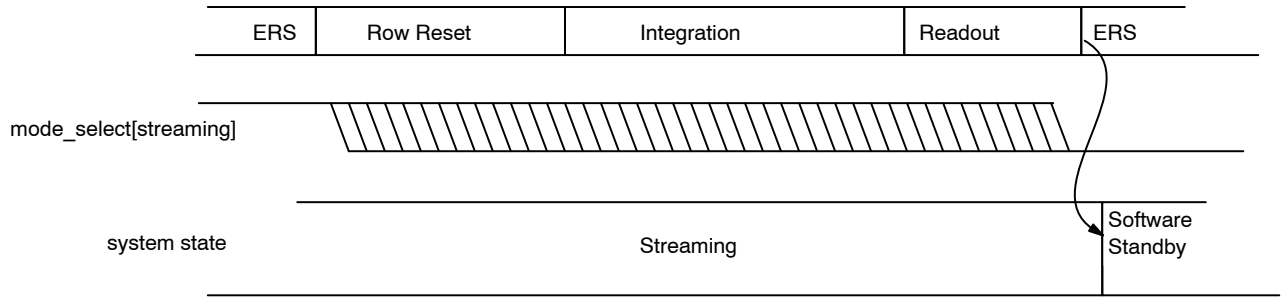



**Figure 43. Entering Soft Standby During a Global Reset Sequence**

## SENSOR CORE DIGITAL DATA PATH

### Test Patterns

The MT9J003 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 21.

**Table 21. TEST PATTERNS**

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

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

*HiSPi Test Patterns*

Test patterns specific to the HiSPi are also generated. The test patterns are enabled by using test_enable (R0x31C6 – 7) and controlled by test_mode (R0x31C6[6:4]).

**Table 22. HiSPi TEST PATTERNS**

| test_mode | Description |
|---|---|
| 0 | Transmit a constant 0 on all enabled data lanes |
| 1 | Transmit a constant 1 on all enabled data lanes |
| 2 | Transmit a square wave at half the serial data rate on all enabled data lanes |
| 3 | Transmit a square wave at the pixel rate on all enabled data lanes |
| 4 | Transmit a continuous sequence of pseudo random data, with no SAV code, copied on all enabled data lanes |
| 5 | Replace data from the sensor with a known sequence copied on all enabled data lanes |

For all of the test patterns, the MT9J003 registers must be set appropriately to control the frame rate and output timing. This includes:

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

*Test Cursors*

The MT9J003 supports one horizontal and one vertical cursor, allowing a crosshair to be superimposed on the image or on test patterns 1–3. The position and width of each cursor are programmable in R0x31E8–R0x31EE. Both even and odd cursor positions and widths are supported.

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

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

The effect of enabling the test cursors when the image_orientation register is non−zero is not defined by the design specification. The behavior of the MT9J003 is shown in Figure 44 and the test cursors are shown as translucent, for clarity. In practice, they are opaque (they overlay the imaged scene). The manner in which the test cursors are affected by the value of image_orientation can be understood from these implementation details:

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

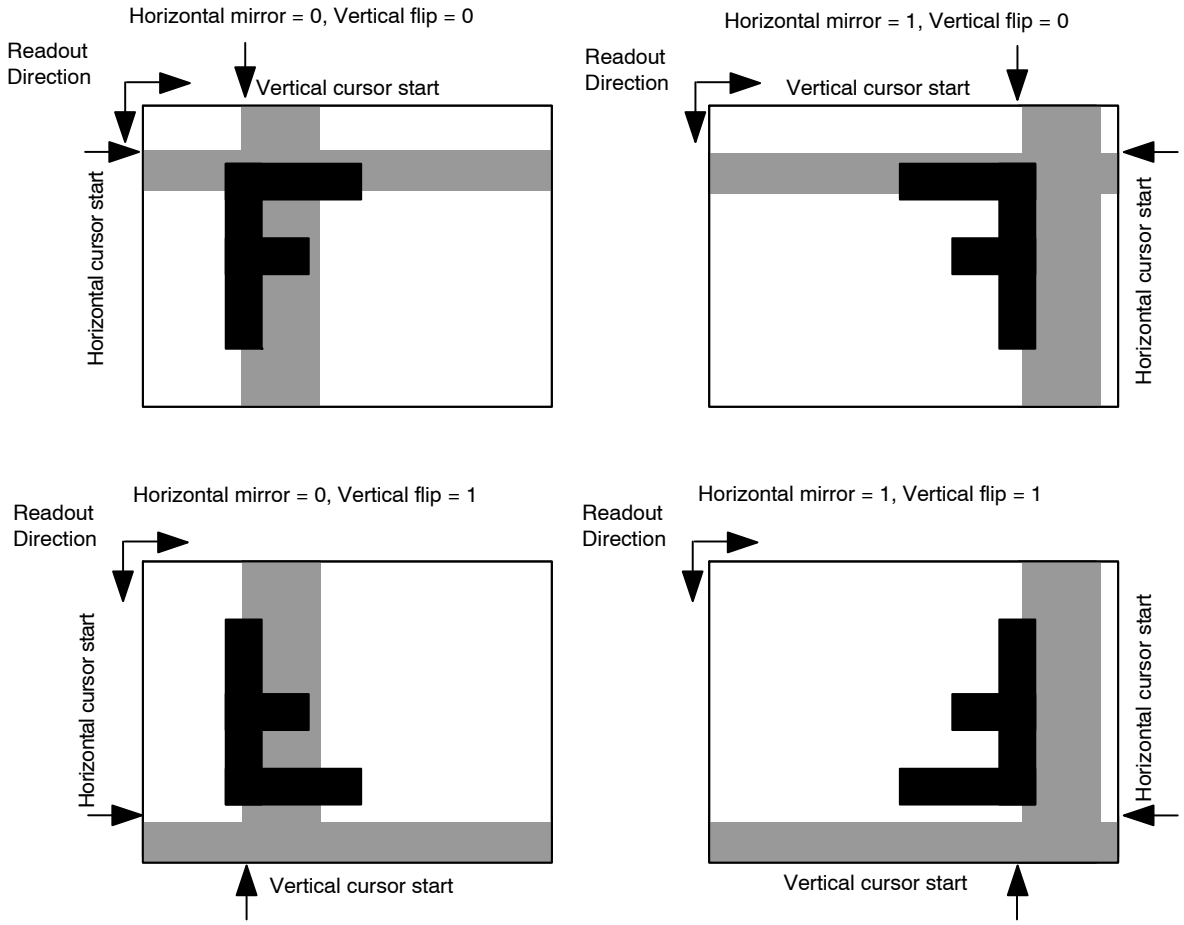



**Figure 44. Test Cursor Behavior With Image Orientation**

## TIMING SPECIFICATIONS

### Power−Up Sequence

The recommended power−up sequence for the MT9J003 is shown in Figure 45. The available power supplies− VDD_IO, VDD, VDD_TX, VDD_PLL, VAA, VAA_PIX, VDD_SLVS, VDD_SLVS_TX− can be turned on at the same time or have the separation specified below.

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

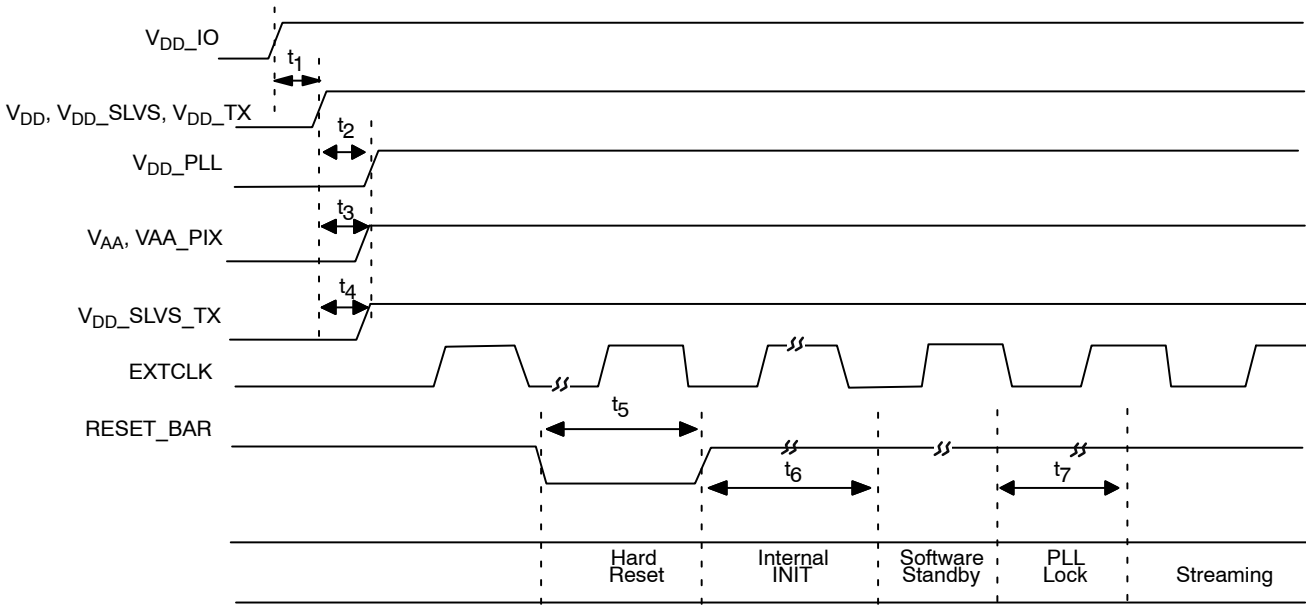



**Figure 45. Power−Up Sequence**

**Table 23. POWER−UP SEQUENCE**

| Definition | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| VDD_IO to VDD, VDD_TX Time | t1 | 0 | – | 500 | ms |
| VDD, VDD_TX to VDD_PLL Time | t2 | 0 | – | 500 | ms |
| VDD, VDD_TX to VAA/VAA_PIX Time | t3 | 0 | – | 500 | ms |
| VAA, VAA_PIX to VDD_SLVS_TX | t4 | – | – | 500 | ms |
| Active Hard Reset | t5 | 1 | – | – | ms |
| Internal Initialization | t6 | 2400 | – | – | EXTCLKs |
| PLL Lock Time | t7 | 1 | – | – | ms |

2. Digital supplies must be turned on before analog supplies.

### Power−Down Sequence

The recommended power−down sequence for the MT9J003 is shown in Figure 46. The available power supplies− VDD_IO, VDD, VDD_TX0, VDD_PLL, VAA, VAA_PIX, VDD_SLVS, VDD_SLVS_TX− 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 $V_{AA}$/$V_{AA}$_PIX and $V_{DD}$_PLL power supplies.
5. After 1–500 ms, turn off $V_{DD}$ and $V_{DD}$_TX0 power supply.
6. After 1–500 ms, turn off $V_{DD}$_IO power supply.

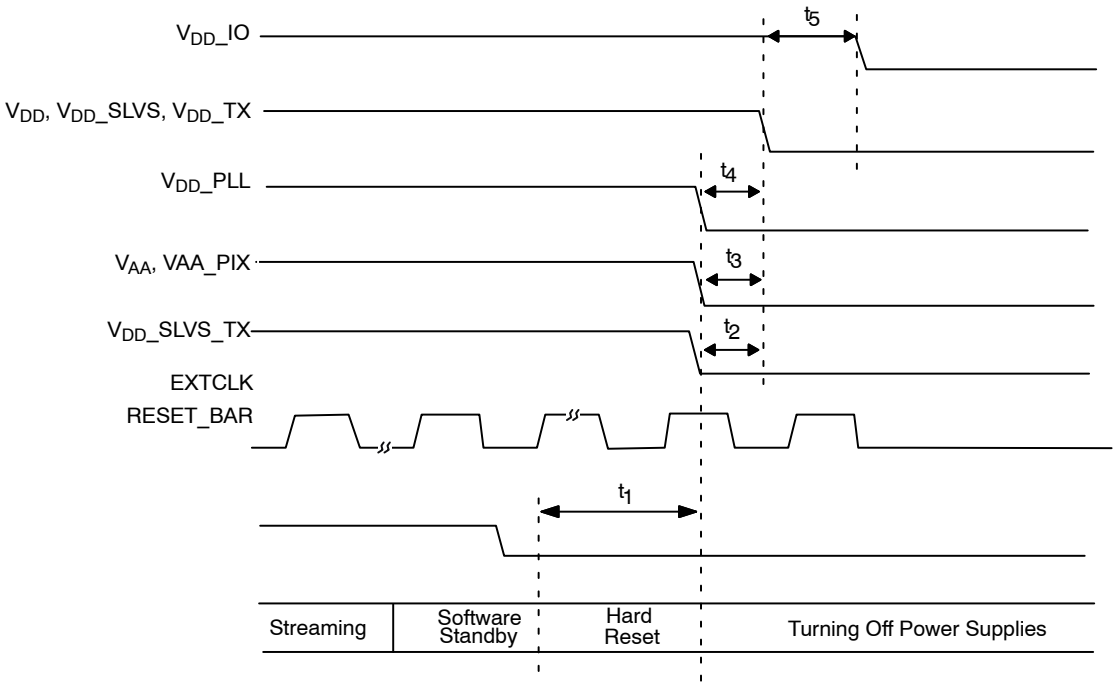



**Figure 46. Test Cursor Behavior With Image Orientation**

**Table 24. POWER−DOWN SEQUENCE**

| Definition | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| Hard reset | $t_1$ | 1 | – | – | ms |
| $V_{DD}$_SLVS_TX to $V_{DD}$ time | $t_2$ | 0 | – | 500 | ms |
| $V_{DD}$/$V_{AA}$/$V_{AA}$_PIX to $V_{DD}$ time | $t_3$ | 0 | – | 500 | ms |
| $V_{DD}$_PLL to $V_{DD}$ time | $t_4$ | 0 | – | 500 | ms |
| $V_{DD}$ to $V_{DD}$_IO time | $t_5$ | 0 | – | 500 | ms |

**Hard Standby and Hard Reset**

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

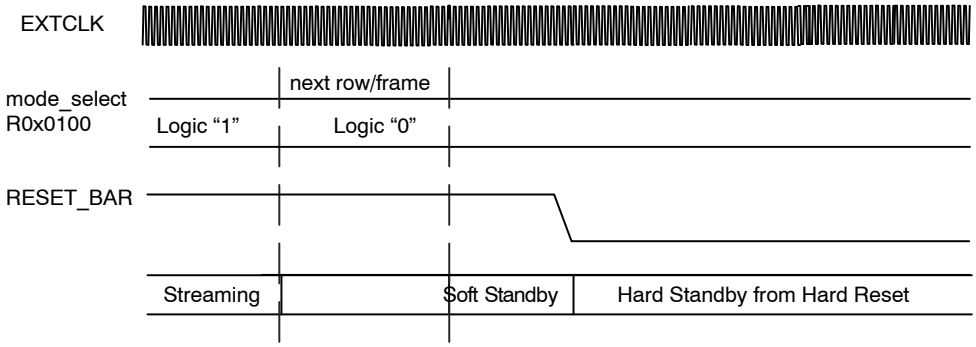



**Figure 47. Hard Standby and Hard Reset**

### Soft Standby and Soft Reset

The MT9J003 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*

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

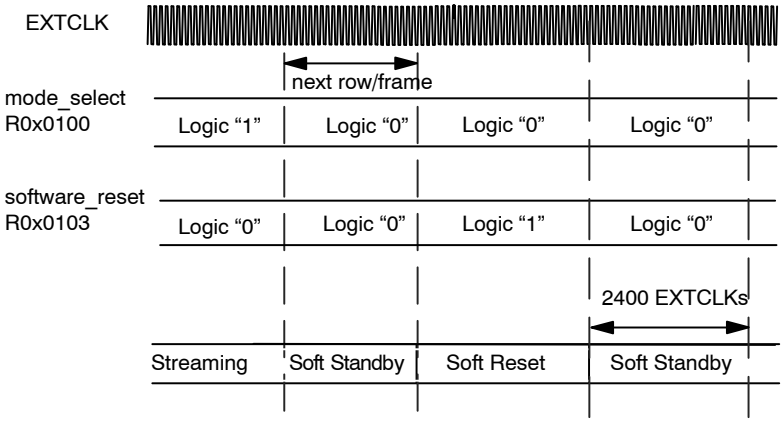



**Figure 48. Soft Standby and Soft Reset**

## SPECTRAL CHARACTERISTICS

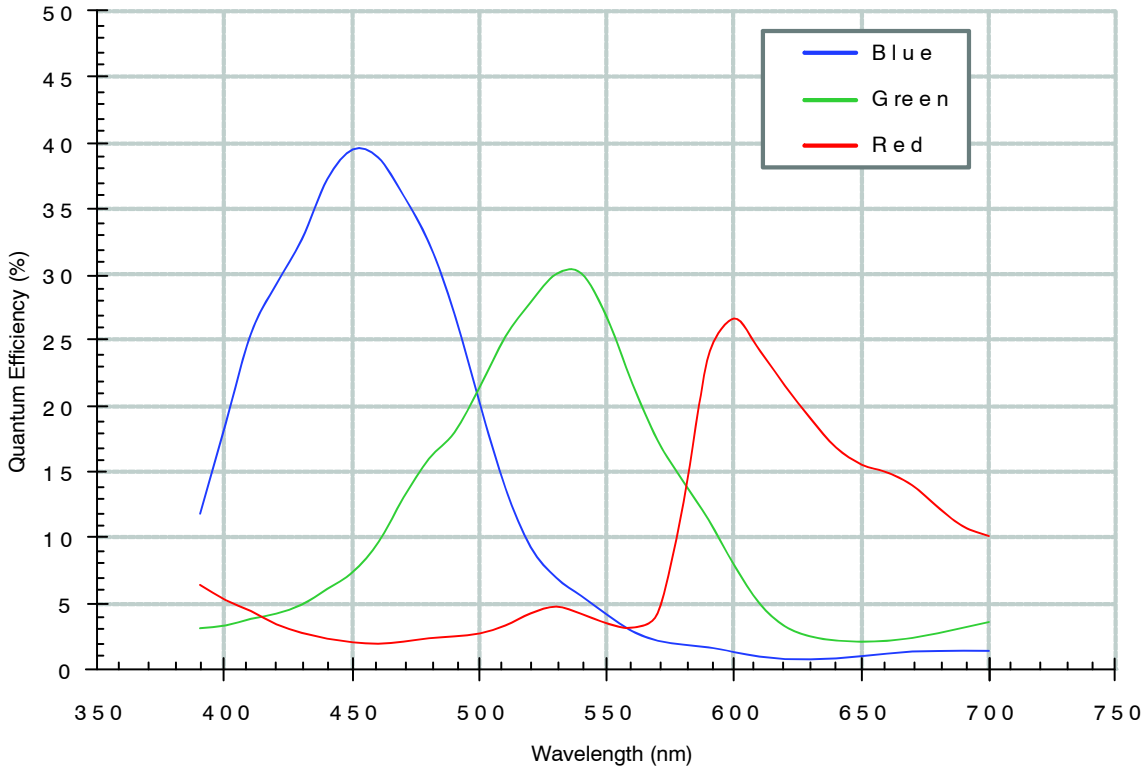



**Figure 49. Quantum Efficiency**

**CRA vs. Image Height Plot**

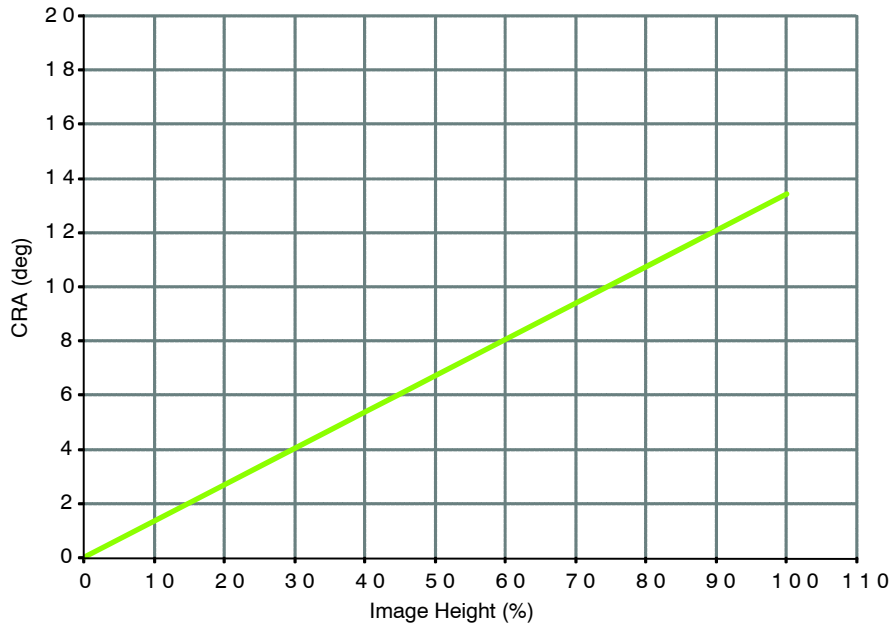



**Figure 50. CRA (13.4°)**

| Image Height | | CRA (°) |
|---|---|---|
| 0 | 0 | 0 |
| 5 | 0.191 | 0.67 |
| 10 | 0.382 | 1.34 |
| 15 | 0.574 | 2.01 |
| 20 | 0.765 | 2.68 |
| 25 | 0.956 | 3.35 |
| 30 | 1.147 | 4.03 |
| 35 | 1.339 | 4.70 |
| 40 | 1.530 | 5.37 |
| 45 | 1.721 | 6.04 |
| 50 | 1.912 | 6.71 |
| 55 | 2.103 | 7.38 |
| 60 | 2.295 | 8.05 |
| 65 | 2.486 | 8.72 |
| 70 | 2.677 | 9.39 |
| 75 | 2.868 | 10.06 |
| 80 | 3.059 | 10.73 |
| 85 | 3.251 | 11.41 |
| 90 | 3.442 | 12.08 |
| 95 | 3.633 | 12.75 |
| 100 | 3.824 | 13.42 |

## ELECTRICAL CHARACTERISTICS

### Table 25. DC ELECTRICAL DEFINITIONS AND CHARACTERISTICS

($^{t}$EXTCLK = 15 MHz; $V_{DD}$ = 1.8 V; $V_{DD}$_IO = 1.8 V; $V_{AA}$ = 2.8 V; VAA_PIX = 2.8 V; $V_{DD}$_PLL = 2.8 V; $V_{DD}$_SLVS = 1.8 V, $V_{DD}$_SLVS_TX = 0.8 V; Output load = 68.5 pF; TJ = 60°C; Data Rate = 480 MHz,; DLL set to 0, 10 Mp frame−rate at 14.7 fps)

| Definition | Condition | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| Core Digital Voltage | | VDD | 1.7 | 1.8 | 1.9 | V |
| Core Digital Voltage | | VDD | 1.7 | 1.8 | 1.9 | V |
| I/O Digital Voltage | Parallel Pixel Data Interface | VDD_IO | 1.7 | 1.8 | 1.9 | V |
| Analog Voltage | | VAA | 2.4 | 2.8 | 3.1 | V |
| Pixel Supply Voltage | | VAA_PIX | 2.4 | 2.8 | 3.1 | V |
| PLL Supply Voltage | | VDD_PLL | 2.4 | 2.8 | 3.1 | V |
| HiSPi Digital Voltage | | VDD_SLVS | 1.7 | 1.8 | 1.9 | V |
| HiSPi I/O Digital Voltage | | VDD_SLVS_TX | 0.3 | 0.4 | 0.9 | V |
| Digital Operating Current | Streaming, Full Resolution | | 35 | 41 | 45 | mA |
| I/O Digital Operating Current | Streaming, Full Resolution | | 0 | 0 | 0 | mA |
| Analog Operating Current | Streaming, Full Resolution | | 132 | 169 | 190 | mA |
| Pixel Supply Current | Streaming, Full Resolution | | 2.7 | 7.6 | 13.3 | mA |
| PLL Supply Current | Streaming, Full Resolution | | 6.5 | 7 | 7.5 | mA |
| HiSPi Digital Operating Current | Streaming, Full Resolution | | n/a | 20 | n/a | mA |
| HiSPi I/O Digital Operating Current | Streaming, Full Resolution | | 13 | 13.5 | 14 | mA |
| Soft Standby (Clock On) | | | 1.3 | 1.5 | 1.9 | mW |

**CAUTION:** Stresses greater than those listed in Table 26 may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

### Table 26. ABSOLUTE MAXIMUM RATINGS

| Symbol | Definition | Condition | Min | Max | Unit |
|---|---|---|---|---|---|
| VDD_MAX | Core Digital Voltage | | –0.3 | 1.9 | V |
| VDD_IO_MAX | I/O Digital Voltage | | –0.3 | 3.1 | V |
| VAA_MAX | Analog Voltage | | –0.3 | 3.5 | V |
| VAA_PIX | Pixel Supply Voltage | | –0.3 | 3.5 | V |
| VDD_PLL | PLL Supply Voltage | | –0.3 | 3.5 | V |
| VDD_SLVS_MAX | HiSPi Digital Voltage | | –0.3 | 1.9 | V |
| VDD_SLVS_TX_MAX | HiSPi I/O Digital Voltage | | –0.3 | 1.2 | V |
| IDD | Digital Operating Current | | – | 90 | mA |
| IDD_IO | I/O Digital Operating Current | | – | 100 | mA |
| IAA_MAX | Analog Operating Current | | – | 225 | mA |
| IAA_PIX | Pixel Supply Current | | – 6 | 25 | mA |
| IDD_PLL | PLL Supply Current | | – | 25 | mA |
| $^{t}$OP | Operating Temperature | Measure at Junction | –30 | 70 | °C |
| $^{t}$ST | Storage Temperature | | –40 | 85 | °C |

Stresses exceeding those listed in the Maximum Ratings table may damage the device. If any of these limits are exceeded, device functionality should not be assumed, damage may occur and reliability may be affected.

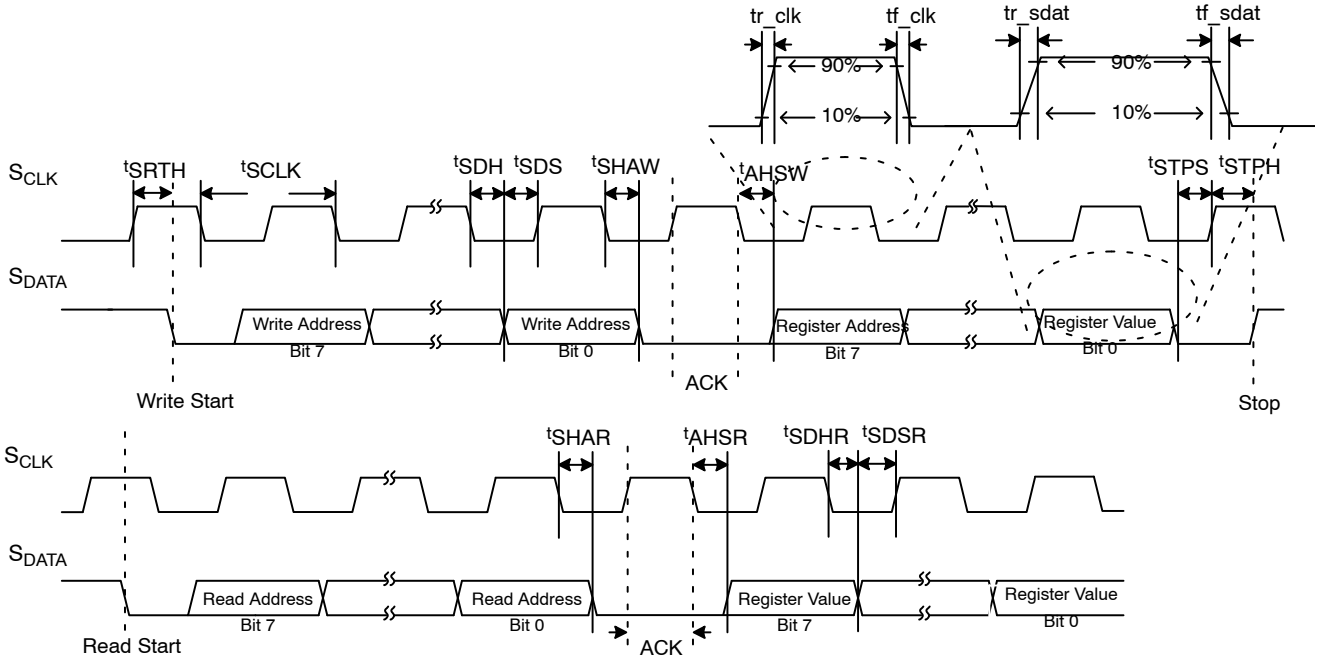
3. Exposure to absolute maximum rating conditions for extended periods may affect reliability.
4. To keep dark current and shot noise artifacts from impacting image quality, care should be taken to keep $^{t}$OP at a minimum.

### Table 27. PARALLEL INTERFACE CONFIGURED TO USE LOW POWER MODE

($f_{EXTCLK}$ = 15 MHz; $V_{DD}$ = 1.8 V; $V_{DD}$_IO = 1.8 V; $V_{AA}$ = 2.8 V; VAA_PIX = 2.8 V; $V_{DD}$_PLL = 2.8 V; $T_J$ = 60°C; Parallel Data Rate = 80 Mp/s)

| | Frame Rate | $I_{AA}$ | $I_{DDPLL}$ | $I_{DD}$ | $I_{DDIO}$ | $IA_{APIX}$ | |
|---|---|---|---|---|---|---|---|
| 10 MP | 7.5 fps | 103.29 | 10.26 | 23.93 | 11.53 | 2.33 | 388 mW |
| 720p60 | 59.94 fps | 122.78 | 10.25 | 23.85 | 11.21 | 5.39 | 451 mW |
| 1080p30 | 29.97 fps | 114.67 | 10.26 | 22.89 | 4.49 | 4.14 | 411 mW |
| VGA60 | 59.94 fps | 82.66 | 10.27 | 18.5 | 4.51 | 5.25 | 316 mW |
| Monitor | 29.97 fps | 69.22 | 10.28 | 16.3 | 6.35 | 2.76 | 271 mW |

5. Monitor is a low power VGA preview mode. The power consumption values in this table represent a small sample of MT9J003 sensors. The $I_{DD}$IO current will double if the $V_{DD}$_IO voltage is raised to 2.8 V.




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

**Figure 51. Two−Wire Serial Bus Timing Parameters**

### Table 28. TWO−WIRE SERIAL REGISTER INTERFACE ELECTRICAL CHARACTERISTICS

($f_{EXTCLK}$ = 15 MHz; $V_{DD}$ = 1.8 V; $V_{DD}$_IO = 1.8 V; $V_{AA}$ = 2.8 V; VAA_PIX = 2.8 V; $V_{DD}$_PLL = 2.8 V; $V_{DD}$_SLVS = 1.8 V, VDD_SLVS_TX = 0.4 V; Output load = 68.5 pF; $T_J$ = 60°C; Data Rate = 480 MHz; DLL set to 0)

| Symbol | Parameter | Condition | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $V_{IL}$ | Input LOW voltage | | −0.5 | 0.73 | 0.3 x $V_{DD}$_IO | V |
| $I_{IN}$ | Input leakage current | No pull up resistor; $V_{IN}$ = $V_{DD}$_IO or $D_{GND}$ | −2 | | 2 | μA |
| $V_{OL}$ | Output LOW voltage | At specified 2 mA | 0.031 | 0.032 | 0.035 | V |
| $I_{OL}$ | Output LOW current | At specified $V_{OL}$ 0.1 V | | | 3 | mA |
| $C_{IN}$ | Input pad capacitance | | | | 6 | pF |
| $C_{LOAD}$ | Load capacitance | | | | | pF |

**Table 29. TWO−WIRE SERIAL REGISTER INTERFACE TIMING SPECIFICATION**

($^{f}$EXTCLK = 15 MHz; $V_{DD}$ = 1.8 V; $V_{DD}$_IO = 1.8 V; $V_{AA}$ = 2.8 V; VAA_PIX = 2.8 V; $V_{DD}$_PLL = 2.8 V; $V_{DD}$_SLVS = 1.8 V, VDD_SLVS_TX = 0.4 V; Output load = 68.5 pF; $T_J$ = 60°C; Data Rate = 480 MHz,; DLL set to 0)

| Symbol | Parameter | Condition | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| fSCLK | Serial Interface Input Clock | – | 0 | 100 | 400 | kHz |
| | SCLK Duty Cycle | VOD | 45 | 50 | 60 | % |
| tR | SCLK/SDATA Rise Time | | | | 300 | µs |
| tSRTS | Start Setup Time | Master WRITE to Slave | 0.6 | | | µs |
| tSRTH | Start Hold Time | Master WRITE to Slave | 0.4 | | | µs |
| tSDH | SDATA Hold | Master WRITE to Slave | 0.3 | | 0.65 | µs |
| tSDS | SDATA Setup | Master WRITE to Slave | 0.3 | | | µs |
| tSHAW | SDATA Hold to ACK | Master READ to Slave | 0.15 | | 0.65 | µs |
| tAHSW | ACK Hold to SDATA | Master WRITE to Slave | 0.15 | | 0.70 | µs |
| tSTPS | Stop Setup Time | Master WRITE to Slave | 0.3 | | | µs |
| tSTPH | Stop Hold Time | Master WRITE to Slave | 0.6 | | | µs |
| tSHAR | SDATA Hold to ACK | Master WRITE to Slave | 0.3 | | 1.65 | µs |
| tAHSR | ACK Hold to SDATA | Master WRITE to Slave | 0.3 | | 0.65 | µs |
| tSDHR | SDATA Hold | Master READ from Slave | 012 | | 0.70 | µs |
| tSDSR | SDATA Setup | Master READ from Slave | 0.3 | | | µs |

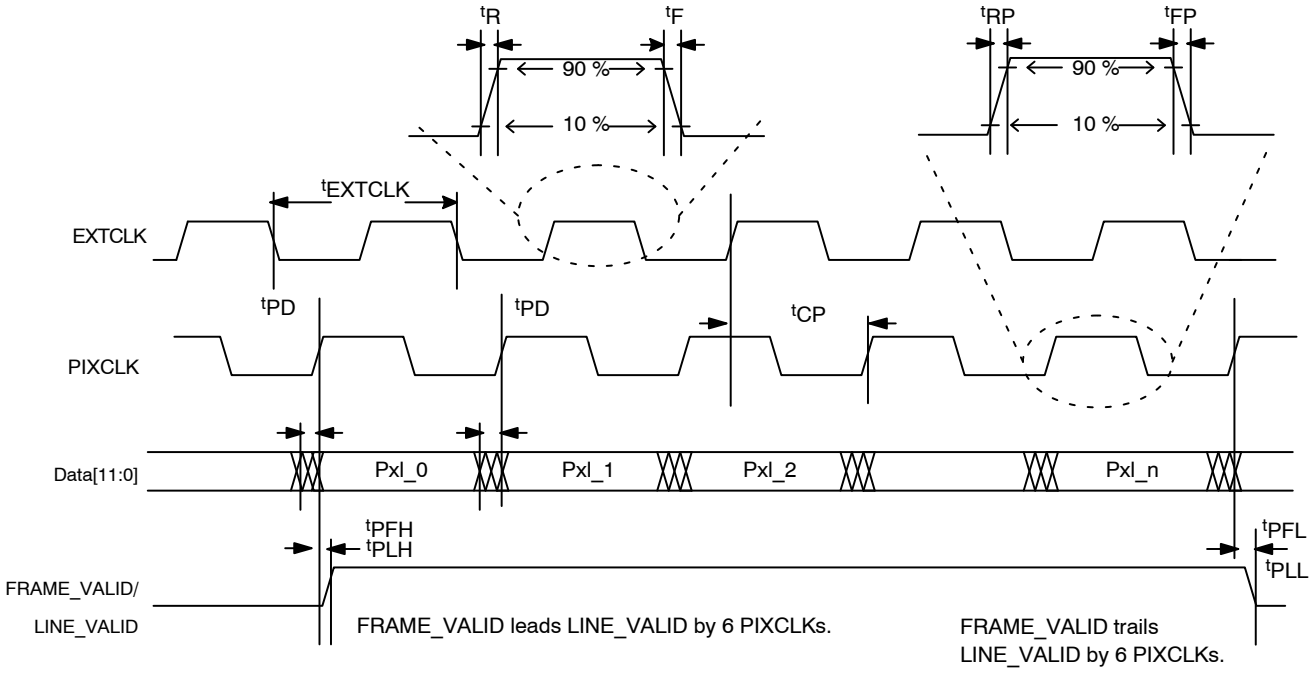



**Figure 52. I/O Timing Diagram**

**Table 30. I/O PARAMETERS**

($^{f}$EXTCLK = 15 MHz; $V_{DD}$ = 1.8 V; $V_{AA}$ = 2.8 V; VAA_PIX = 2.8 V; $V_{DD}$_PLL = 2.8 V; $V_{DD}$_SLVS = 1.8 V, VDD_SLVS_TX = 0.4 V; Output load = 68.5 pF; $T_J$ = 60°C; Data Rate = 480 MHz,; DLL set to 0)

| Symbol | Definition | Conditions | Min | Max | Unit |
|---|---|---|---|---|---|
| $V_{IH}$ | Input HIGH Voltage | $V_{DD}$_IO = 1.8V | 1.4 | $V_{DD}$_IO + 0.3 | V |
| | | $V_{DD}$_IO = 2.8V | 2.4 | | |
| $V_{IL}$ | Input LOW Voltage | $V_{DD}$_IO = 1.8V | GND – 0.3 | 0.4 | |
| | | $V_{DD}$_IO = 2.8V | GND – 0.3 | 0.8 | |
| $I_{IN}$ | Input Leakage Current | No Pull–up Resistor; $V_{IN}$ = $V_{DD}$ or $D_{GND}$ | – 20 | 20 | μA |
| $V_{OH}$ | Output HIGH Voltage | At Specified $I_{OH}$ | $V_{DD}$_IO – 0.4V | – | V |
| $V_{OL}$ | Output LOW Voltage | At Specified $I_{OL}$ | – | 0.4 | V |
| $I_{OH}$ | Output HIGH Current | At Specified $V_{OH}$ | – | –12 | mA |
| $I_{OL}$ | Output LOW Current | At Specified $V_{OL}$ | – | 9 | mA |
| $I_{OZ}$ | Tri–state Output Leakage Current | | – | 10 | μA |

**Table 31. I/O TIMING**

($^{f}$EXTCLK = 15 MHz; $V_{DD}$ = 1.8 V; $V_{DD}$_IO = 1.8 V; $V_{AA}$ = 2.8 V; VAA_PIX = 2.8 V; $V_{DD}$_PLL = 2.8 V; $V_{DD}$_SLVS = 1.8 V, VDD_SLVS_TX = 0.4 V; Output load = 68.5 pF; $T_J$ = 60°C; Data Rate = 480 MHz; DLL set to 0)

| Symbol | Definition | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $^{f}$EXTCLK | Input Clock Frequency | PLL Enabled | 6 | 24 | 48 | MHz |
| $^{t}$EXTCLK | Input Clock Period | PLL Enabled | 166 | 41 | 20 | ns |
| $^{t}$R | Input Clock Rise Time | | 0.1 | – | 1 | V/ns |
| $^{t}$F | Input Clock Fall Time | | 0.1 | – | 1 | V/ns |
| | Clock Duty Cycle | | 45 | 50 | 55 | % |
| $^{t}$JITTER | Input Clock Jitter | | – | – | 0.3 | ns |
| Output Pin Slew | Fastest | $C_{LOAD}$ = 15 pF | – | 0.7 | – | V/ns |
| $^{f}$PIXCLK | PIXCLK Frequency | Default | – | 80 | – | MHz |
| $^{t}$PD | PIXCLK to Data Valid | Default | – | – | 3 | ns |
| $^{t}$PFH | PIXCLK to FRAME_VALID HIGH | Default | – | – | 3 | ns |
| $^{t}$PLH | PIXCLK to LINE_VALID HIGH | Default | – | – | 3 | ns |
| $^{t}$PFL | PIXCLK to FRAME_VALID LOW | Default | – | – | 3 | ns |
| $^{t}$PLL | PIXCLK to LINE_VALID LOW | Default | – | – | 3 | ns |

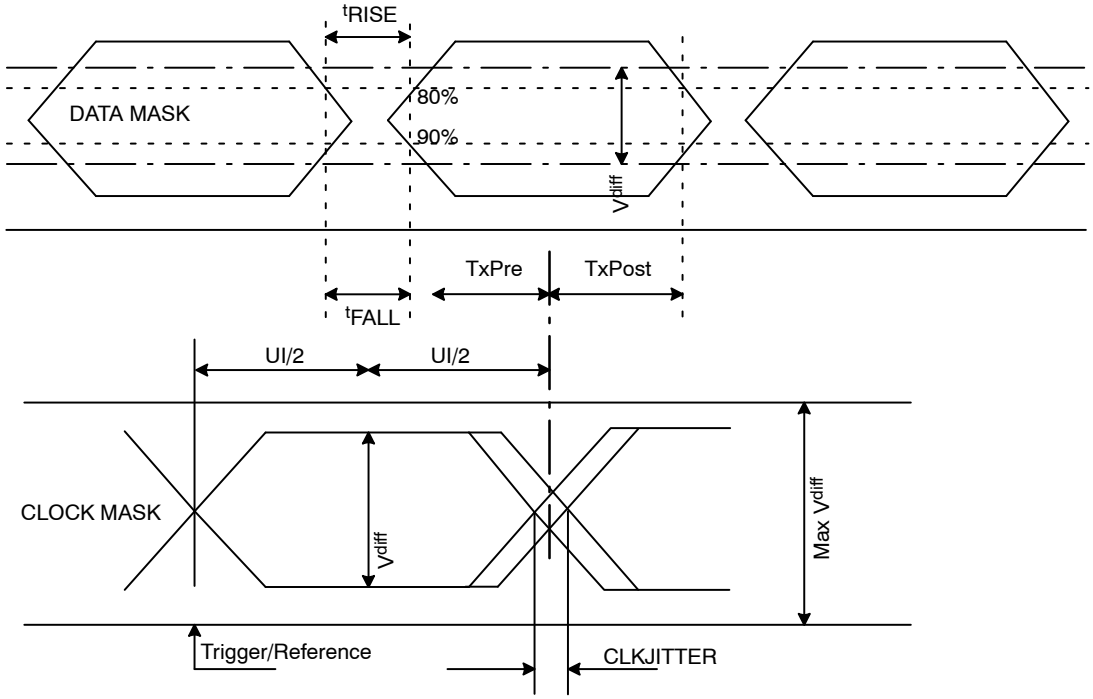



**Figure 53. HiSPi Eye Diagram for Both Clock and Data Signals**

**Table 32. HiSPi RISE AND FALL TIMES AT 480 MHz**

(Measurement Conditions: PHY Supply 1.8 V, HiSPi Power Supply 0.8 V, Data Rate 480 MHz, DLL set to 0)

| Parameter | Name | Value | Unit |
|---|---|---|---|
| Max Setup Time from Transmitter | TxPRE | 0.44 | UI |
| Max Hold Time from Transmitter | TxPost | 0.44 | UI |
| Rise Time t | $^{t}$RISE | 350 | ps |
| Fall Time t | $^{t}$FALL | 350 | ps |
| Output Impedance | | 66 | Ω |

**Table 33. HiSPi RISE AND FALL TIMES AT 360 MHz**

(Measurement Conditions: PHY Supply 1.8 V, HiSPi Power Supply 0.8 V, Data Rate 480 MHz, DLL set to 0)

| Parameter | Name | Value | Unit |
|---|---|---|---|
| Max Setup Time from Transmitter | TxPRE | 0.48 | UI |
| Max Hold Time from Transmitter | TxPost | 0.42 | UI |
| Rise Time t | $^{t}$RISE | 350 | ps |
| Fall Time t | $^{t}$FALL | 350 | ps |
| Output Impedance | | 66 | Ω |

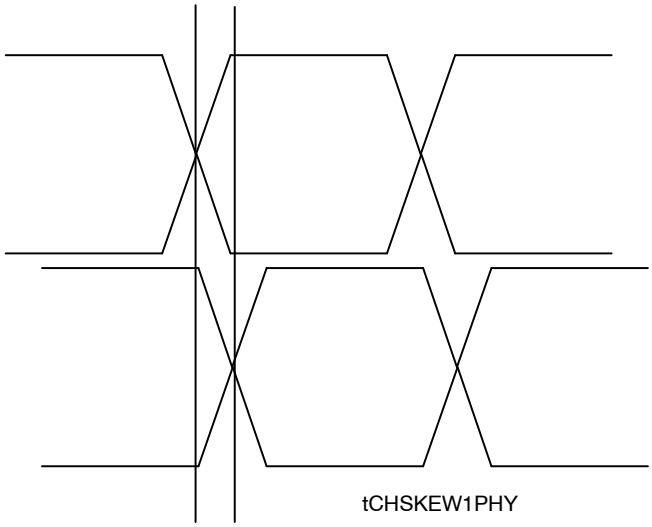



**Figure 54. HiSPi Skew Between Data Signals Within the PHY**

### Table 34. CHANNEL, PHY AND INTRA−PHY SKEW

(Measurement Conditions: PHY Supply 1.8 V, HiSPi Power Supply 0.8 V, Data Rate 480 MHz, DLL set to 0)

| | | | |
|---|---|---|---|
| Data Lane Skew in Reference to Clock | tCHSKEW1PHY | −150 | ps |

### Table 35. CLOCK DLL STEPS

(Measurement Conditions: PHY Supply 1.8 V, HiSPi Power Supply 0.8 V, Data DLL set to 0)

| Clock DLL Step | 1 | 2 | 3 | 4 | 5 | Step |
|---|---|---|---|---|---|---|
| Delay @ 480 MHz | 0.25 | 0.375 | 0.5 | 0.625 | 0.75 | UI |
| Eye_opening@ 480 MHz | 0.85 | 0.78 | 0.71 | 0.71 | 0.69 | UI |
| Eye_opening@ 360 MHz | 0.89 | 0.83 | 0.81 | 0.60 | 046 | UI |

6. The Clock DLL Steps 6 and 7 are not recommended by **onsemi** for the MT9J003 Rev. 2.

### Table 36. DATA DLL STEPS

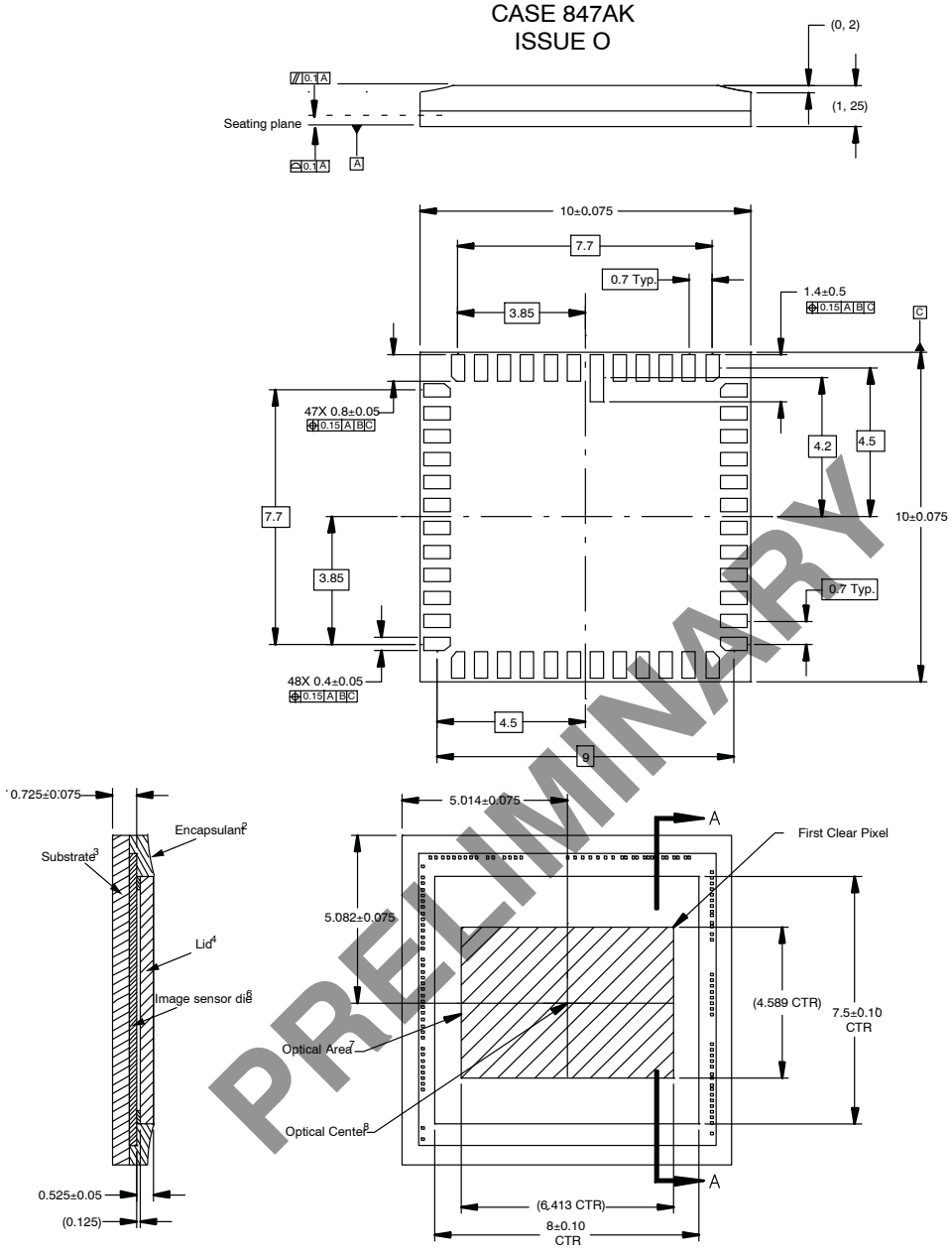
(Measurement Conditions: PHY Supply 1.8 V, HiSPi Power Supply 0.8 V, Clock DLL set to 0)

| Data DLL Step | 1 | 2 | 4 | 6 | Step |
|---|---|---|---|---|---|
| Delay @ 480MHz | 0.25 | 0.375 | 0.625 | 0.875 | UI |
| Eye_opening@ 480 MHz | 0.79 | 0.84 | 0.71 | 0.61 | UI |
| Eye_opening@ 360 MHz | 0.85 | 0.83 | 0.82 | 0.77 | UI |

7. The Data DLL Steps 3, 5, and 7 are not recommended by **onsemi** for the MT9J003 Rev. 2.

## PACKAGE DIMENSIONS

**ILCC48 10x10**
CASE 847AK
ISSUE O




PRELIMINARY

Notes:

1. Dimensions in mm. Dimensions in () are for reference only.
2. Encapsulant: Epoxy
3. Substrate material: Plastic laminate 0.5 thickness
4. List material: Borosilicate glass 0.4 thickness. Refractive index at 20°C = 1.5255 @ 546 nm and 1.5231 @ 588 nm.
5. Lead finish: Gold plating, 0.5 microns minimum thickness.
6. Image sensor die 0.2 thickness.
7. Maximum rotation of optical area relative to seating plane A: 25 microns.
   Maximum tilt of optical area relative to top of cover glass: 20 microns.
   Maximum tilt of optical area relative to top of cover glass: 50 microns.
8. Die center = package center; optical center offset from package center: X = 0.01356, Y = −0.081705

**onsemi**, , and other names, marks, and brands are registered and/or common law trademarks of Semiconductor Components Industries, LLC dba "**onsemi**" or its affiliates and/or subsidiaries in the United States and/or other countries. **onsemi** owns the rights to a number of patents, trademarks, copyrights, trade secrets, and other intellectual property. A listing of **onsemi**'s product/patent coverage may be accessed at www.onsemi.com/site/pdf/Patent−Marking.pdf. **onsemi** reserves the right to make changes at any time to any products or information herein, without notice. The information herein is provided "as−is" and **onsemi** makes no warranty, representation or guarantee regarding the accuracy of the information, product features, availability, functionality, or suitability of its products for any particular purpose, nor does **onsemi** assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation special, consequential or incidental damages. Buyer is responsible for its products and applications using **onsemi** products, including compliance with all laws, regulations and safety requirements or standards, regardless of any support or applications information provided by **onsemi**. "Typical" parameters which may be provided in **onsemi** data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. **onsemi** does not convey any license under any of its intellectual property rights nor the rights of others. **onsemi** products are not designed, intended, or authorized for use as a critical component in life support systems or any FDA Class 3 medical devices or medical devices with a same or similar classification in a foreign jurisdiction or any devices intended for implantation in the human body. Should Buyer purchase or use **onsemi** products for any such unintended or unauthorized application, Buyer shall indemnify and hold **onsemi** and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that **onsemi** was negligent regarding the design or manufacture of the part. **onsemi** is an Equal Opportunity/Affirmative Action Employer. This literature is subject to all applicable copyright laws and is not for resale in any manner.

## ADDITIONAL INFORMATION

**TECHNICAL PUBLICATIONS**:
**Technical Library:** www.onsemi.com/design/resources/technical−documentation
**onsemi Website:** www.onsemi.com

**ONLINE SUPPORT**: www.onsemi.com/support
**For additional information, please contact your local Sales Representative at** www.onsemi.com/support/sales