1/2.3-Inch 14 Mp CMOS Digital Image Sensor

MT9F002 Data Sheet, Rev. 9

For the latest data sheet, please visit: www.onsemi.com

Features

- 1.4 μm pixel with ON Semiconductor A-Pix™ technology
- Simple two-wire serial interface
- Auto black level calibration
- Full HD support at 60 fps for maximum video performance
- 20 percent extra image array area in full HD to enable electronic image stabilization (EIS).
- 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, 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 (SLVS)
- On-chip phase-locked loop (PLL) oscillator
- Bayer pattern downsize scaler

Applications

- Digital video cameras
- Digital still cameras

General Description

The ON Semiconductor MT9F002 is a 1/2.3-inch CMOS active-pixel digital imaging sensor with an active pixel array of 4608H x 3288V (4640H x 3320V including border pixels). It can support 14-megapixel (4384H x 3288V) digital still images and a 1080p plus additional 20 percent pixels for electronic image stabilization (4608H x 2592V) in digital video mode. The MT9F002 sensor is programmable through a simple two-wire serial interface, and has low power consumption.

Table 1: Key Performance Parameters

Parameter		Value
Optical format		1/2.3-inch (4:3)
Active pixels and imager size		• 4608H x 3288V: (entire array): 6.451mm (H) x 4.603mm (V), 7.925mm diagonal • 4384H x 3288V (4:3, still mode): 6.138mm (H) x 4.603mm (V), 7.672mm diagonal • 4608H x 2592V (16:9, video mode): 6.451mm (H) x 3.629mm (V), 7.402mm diagonal
Pixel size		1.4 μm x 1.4μm
Chief ray angle		0°, 11.4°, and 25°
Color filter array		RGB Bayer pattern
Shutter type		Electronic rolling shutter (ERS) with global reset release (GRR)
Input clock frequency		2–64 MHz
Maximum data rate	Parallel	96 Mp/s at 96 MHz PIXCLK
	HiSPi (4-lane)	700 Mbps/lane
Frame rate	14M resolution (4384H x 3288V)	Programmable up to 13.7 fps for HiSPi I/F, 6.3 fps for parallel I/F
	Preview VGA mode	• 30 fps with binning • 60 fps with skip2bin2
	1080p mode:	• 60 fps using HiSPi interface 2304H x 1296V (1080p +20%EIS) • 30 fps using parallel interface 2256H x 1268V (1080p +17%EIS)
ADC resolution		12-bit, on-chip
Responsivity		0.724 V/lux-sec (550nm)
Dynamic range		65.3 dB
SNR_{MAX}		35.5 dB
Supply voltage	I/O Digital	1.7–1.9 V (1.8 V nominal) or 2.4–3.1 V (2.8 V nominal)
	Digital	1.7–1.9 V (1.8 V nominal)
	Analog	2.7–3.1 V (2.8 V nominal)
	HiSPi PHY HiSPi I/O (SLVS) HiSPi I/O (HiVCM)	1.7–1.9 V (1.8 V nominal) 0.3 - 0.9 V (0.4 or 0.8 V nominal) 1.7–1.9 V (1.8 V nominal)
Power Consumption	Full resolution 13.65 fps (HiSPi serial I/F, 12-bit)	724 mW
	1080p60 (HiSPi serial I/F, 10-bit)	XYbin2: 596 mW
	1080p30 (HiSPi serial I/F, 10-bit)	XYbin2: 443 mW
Package		48-pin iLCC (10 mm x 10 mm) and bare die
Operating temperature		–30°C to +70°C (at junction)

Ordering Information

Table 2: Available Part Numbers

Part Number	Product Description	Orderable Product Attribute Description
MT9F002I12STCV-DP	RGB, 0deg CRA, HiSPi, iLCC Package	Drypack, Protective Film
MT9F002I12-N4000-DP1	RGB, 12deg CRA, HiSPi, iLCC Package	Drypack, Protective Film
MT9F002I12STCVD3-GEVK	0deg CRA, HiSPi, Demo Kit	
MT9F002I12STCVH-GEVB	0deg CRA, HiSPi, Head Board	
MT9F002I12-N4000D-GEVK	12deg CRA, HiSPi, Demo Kit	
MT9F002I12-N4000H-GEVB	12deg CRA, HiSPi, Head Board	

Table of Contents

General Description

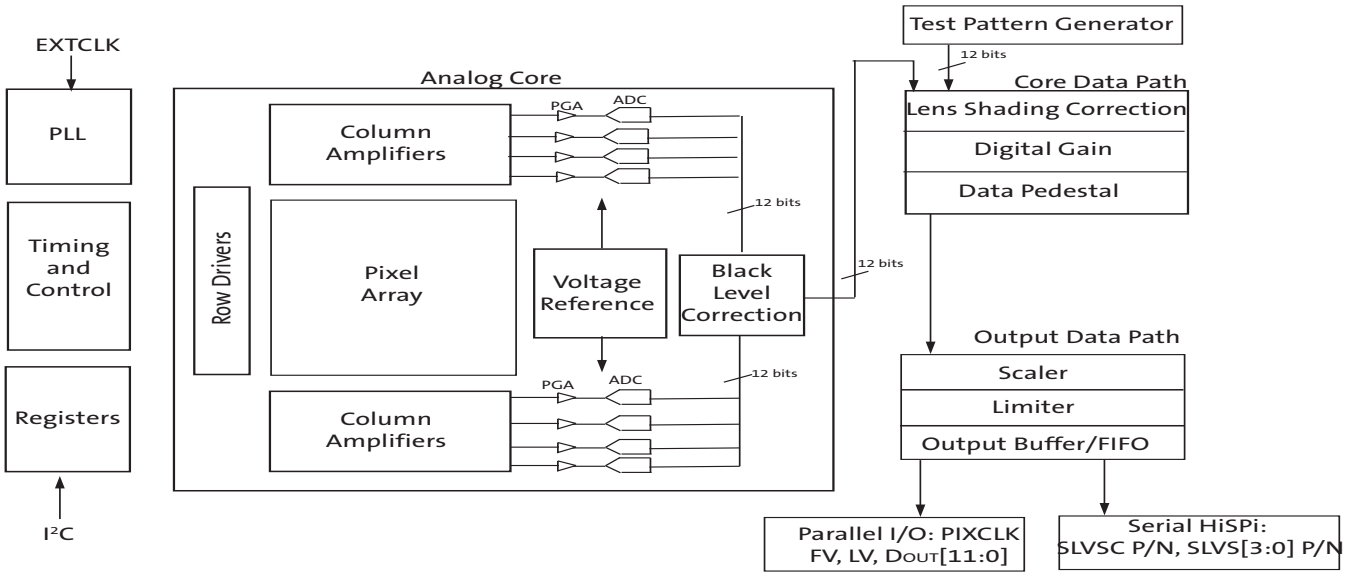
The MT9F002 digital image sensor features ON Semiconductor's breakthrough low-noise CMOS imaging technology that achieves near-CCD image quality (based on signal-to-noise ratio and low-light sensitivity) while maintaining the inherent size, cost, and integration advantages of CMOS.

When operated in its default 4:3 still-mode, the sensor generates a full resolution (4384x3288)image at 13 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.

Functional Overview

The MT9F002 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 2 and 64 MHz. The maximum output pixel rate is 220 Mp/s for serial HiSPi I/F and 96 Mp/s for parallel I/F, corresponding to a pixel clock rate of 220 MHz and 96 MHz, respectively. A block diagram of the sensor is shown in Figure 1.

Figure 1: Block Diagram

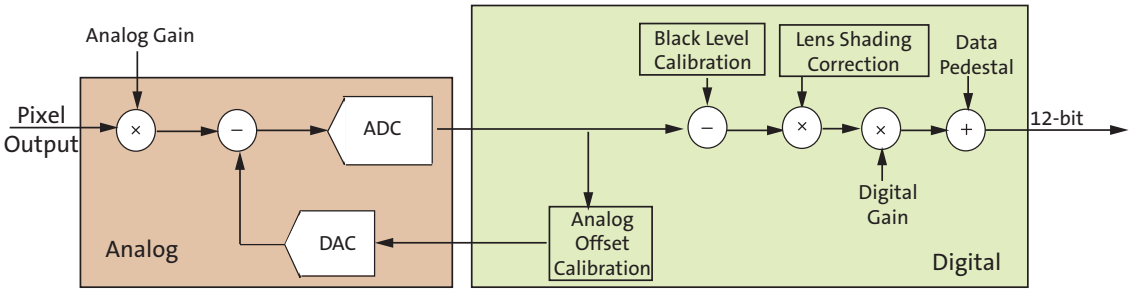


The core of the sensor is a 14Mp active-pixel array. The timing and control circuitry sequences through the rows of the array, resetting and then reading each row in turn. In the time interval between resetting a row and reading that row, the pixels in the row integrate incident light. The exposure is 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 image black level is calibrated to compensate for analog offset and ensure that the ADC range is utilized well. It also reduces row noise in the image. The black level in the output image involves Fine Digital Correction and addition of Data Pedestal (42 LSB for 10-bit ADC, 168 LSB for 12-bit ADC)

Figure 2: Data Flow Diagram



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

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

The control registers, timing and control, and digital processing functions shown in Figure 1 on page 4 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, 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 3. 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 3: Pixel Color Pattern Detail (Top Right Corner)

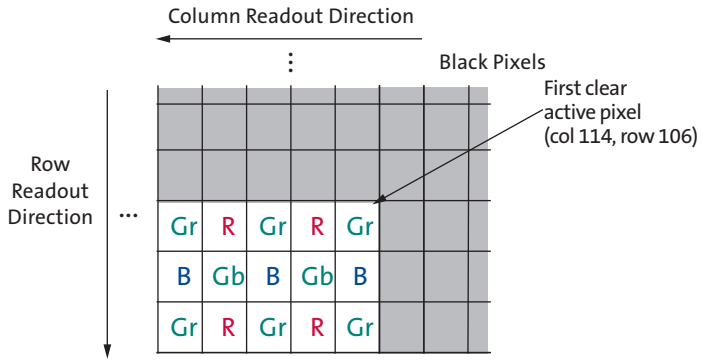
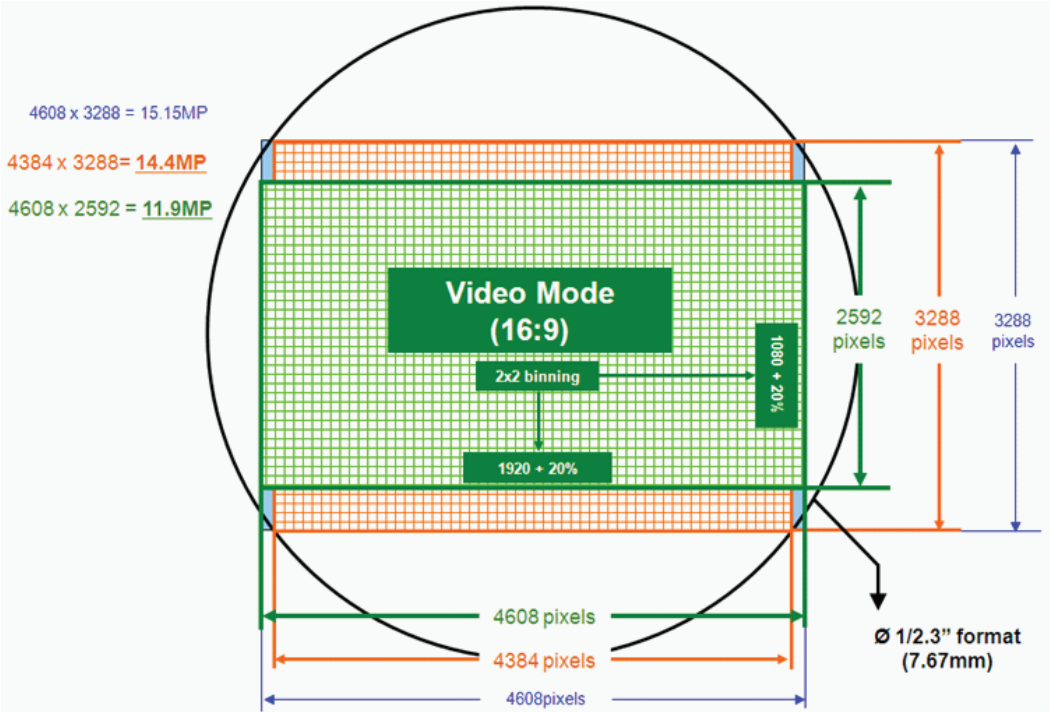


Figure 4: High-Resolution Still Image Capture + Full HD Video



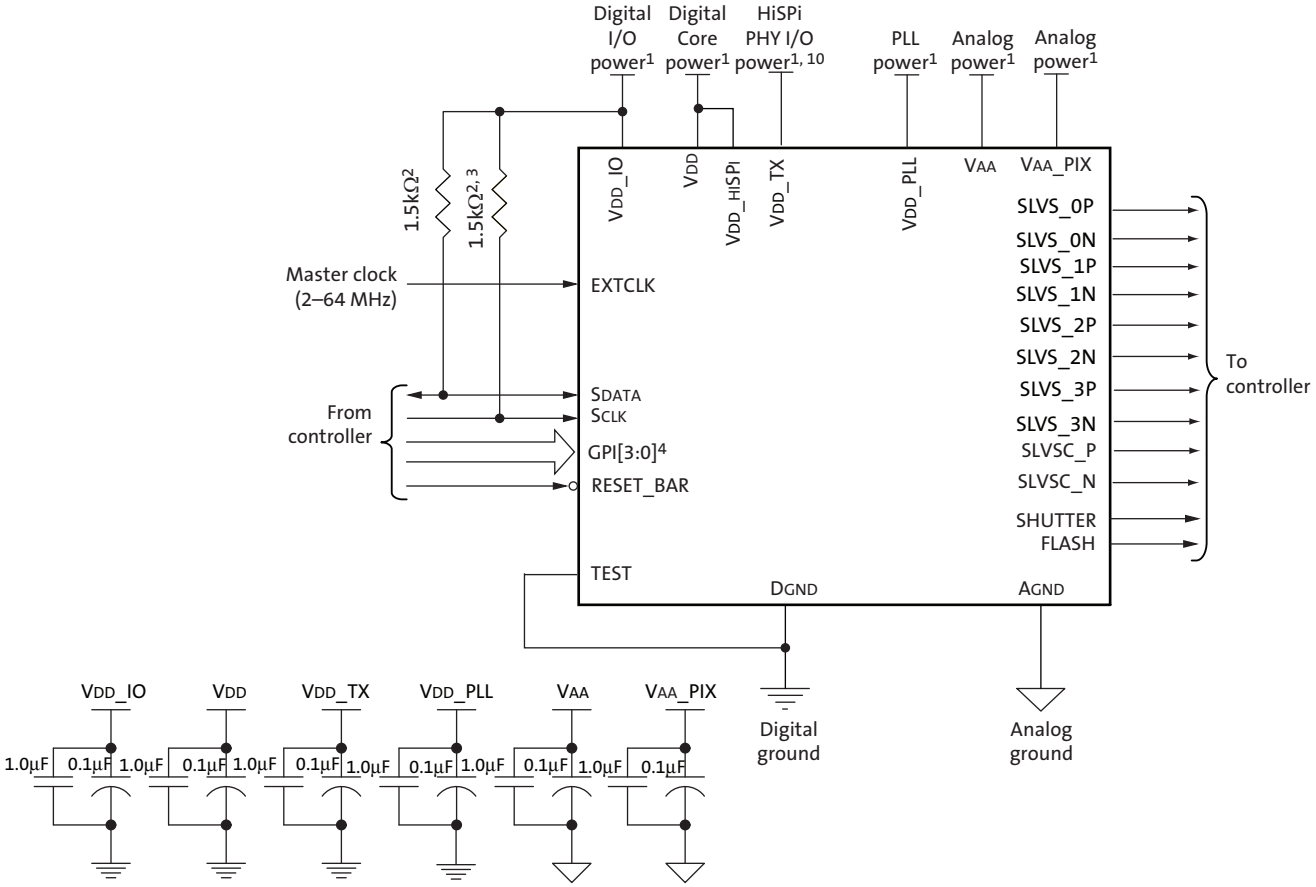
Operating Modes

By default, the MT9F002 powers up with the serial pixel data interface enabled. The sensor can operate in serial HiSPi or parallel mode.

For low-noise operation, the MT9F002 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 **ON Semiconductor does not recommend the use of inductance filters on the power supplies or output signals.**

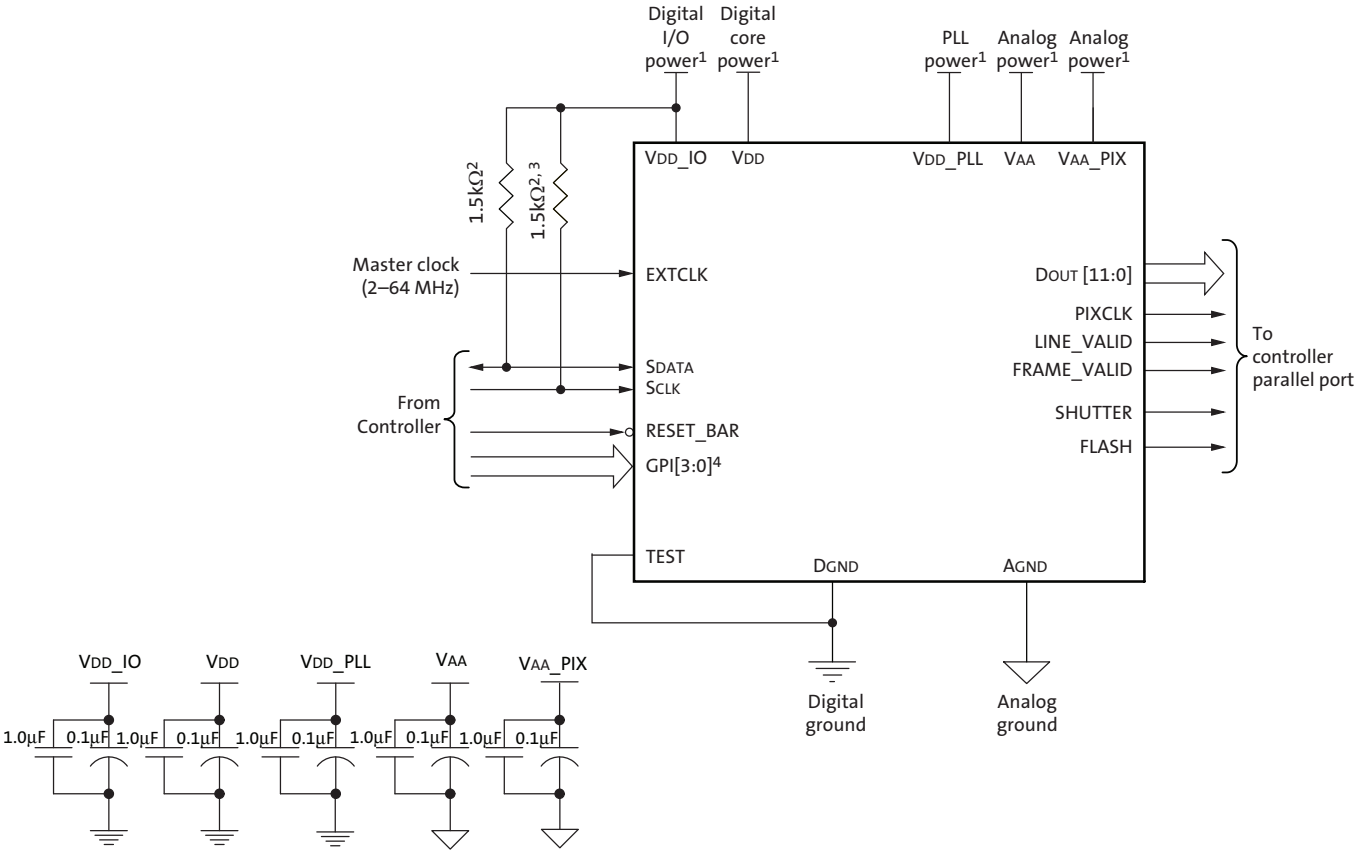
Figure 5: Typical Configuration: Serial Four-Lane HiSPi Interface



Notes:
1. All power supplies should be adequately decoupled. ON Semiconductor recommends having 1.0µF and 0.1µF decoupling capacitors for every power supply.
2. ON Semiconductor recommends a resistor value of 1.5kΩ, but a greater value may be used for slower two-wire speed.
3. This pull-up resistor is not required if the controller drives a valid logic level on SCLK at all times.
4. The GPI pins can be statically pulled HIGH or LOW and can be programmed to perform special functions (TRIGGER/VD, OE_BAR, SADDR, STANDBY) to be dynamically controlled. GPI pads can be left floating, when not used.
5. VPP, which is not shown in Figure 5, is left unconnected during normal operation.

6. The parallel interface output pads can be left unconnected when the serial output interface is used.
7. ON Semiconductor recommends that 0.1µF and 10µF 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 MT9F002 demo headboard schematics for circuit recommendations.
8. TEST signals must be tied to DGND for normal sensor operation.
9. ON Semiconductor recommends that analog power planes are placed in a manner such that coupling with the digital power planes is minimized.
10. For serial HiSPi HiVCM mode, set register bit R0x306E[9] = 1 and VDD_TX = VDD_IO = 1.8V.

Figure 6: Typical Configuration: Parallel Pixel Data Interface



Notes:
1. All power supplies should be adequately decoupled. ON Semiconductor recommends having 1.0µF and 0.1µF decoupling capacitors for every power supply.
2. ON Semiconductor recommends a resistor value of 1.5kΩ, but a greater value may be used for slower two-wire speed.
3. This pull-up resistor is not required if the controller drives a valid logic level on SCLK at all times.
4. The GPI pins can be statically pulled HIGH or LOW and can be programmed to perform special functions (TRIGGER/VD, OE_BAR, SADDR, STANDBY) to be dynamically controlled. GPI pads can be left floating, when not used.
5. VPP, which is not shown in Figure 6, is left unconnected during normal operation.
6. The serial interface output pads can be left unconnected when the parallel output interface is used.
7. ON Semiconductor recommends that 0.1µF and 10µF 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 MT9F002 demo headboard schematics for circuit recommendations.
8. TEST signals must be tied to DGND for normal sensor operation.

9. ON Semiconductor recommends that analog power planes are placed in a manner such that coupling with the digital power planes is minimized.

Signal Descriptions

Table 3 provides signal descriptions for MT9F002 die. For pad location and aperture information, refer to the MT9F002 die data sheet.

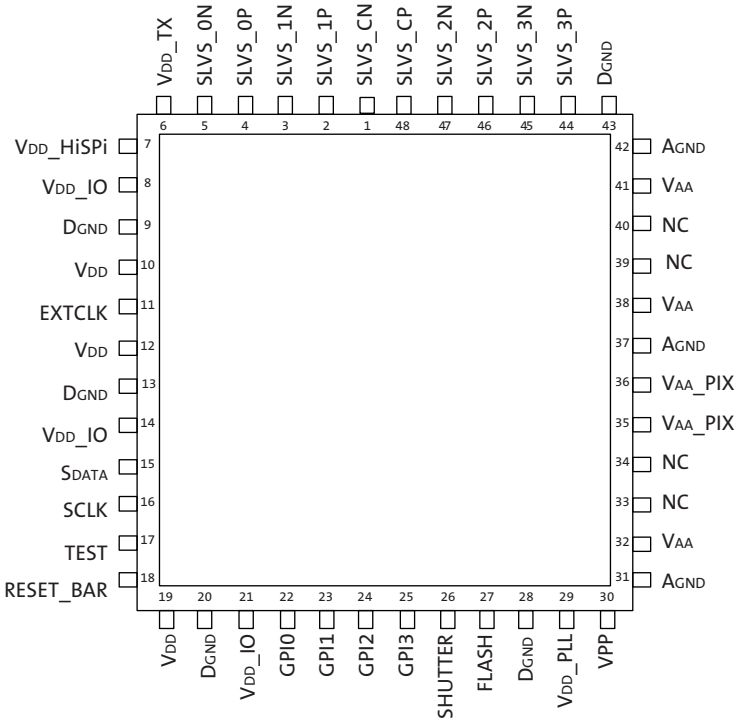
Table 3: Signal Descriptions

Signal	Type	Description
EXTCLK	Input	Master clock input, 2-64 MHz.
RESET_BAR	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 programmed (through register R0x3026) to provide hardware control of the standby, output enable, SADDR select, shutter trigger or slave mode trigger (VD) function. Can be left floating if not used.
TEST	Input	Enable manufacturing test modes. Tie to DGND for normal sensor operation.
SDATA	I/O	Serial data from READs and WRITEs to control and status registers.
VPP	Supply	Disconnect pad for normal operation. Power supply used to program one-time programmable (OTP) memory. Manufacturing use only.
VDD_HiSPi	Supply	HiSPi PHY power supply. Digital power supply for the HiSPi serial data interface. This should be tied to VDD.
VDD_TX	Supply	Digital power supply for the HiSPi I/O. For HiSPi SLVS mode, set register bit R0x306E[9] = 0 (default), and VDD_TX to 0.4V. For HiSPi HiVCM mode, set register bit R0x306E[9] = 1, and VDD_TX = VDD_IO.
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.
SLVS_0P	Output	Lane 1 differential HiSPi (SLVS) serial data (positive). Qualified by the SLVS serial clock.
SLVS_0N	Output	Lane 1 differential HiSPi (SLVS) serial data (negative). Qualified by the SLVS serial clock.
SLVS_1P	Output	Lane 2 differential HiSPi (SLVS) serial data (positive). Qualified by the SLVS serial clock.
SLVS_1N	Output	Lane 2 differential HiSPi (SLVS) serial data (negative). Qualified by the SLVS serial clock.
SLVS_2P	Output	Lane 3 differential HiSPi (SLVS) serial data (positive). Qualified by the SLVS serial clock.
SLVS_2N	Output	Lane 3 differential HiSPi (SLVS) serial data (negative). Qualified by the SLVS serial clock.
SLVS_3P	Output	Lane 4 differential HiSPi (SLVS) serial data (positive). Qualified by the SLVS serial clock.

Table 3: Signal Descriptions (continued)

Signal	Type	Description
SLVS_3N	Output	Lane 4 differential HiSPi (SLVS) serial data (negative). Qualified by the SLVS serial clock.
SLVS_CP	Output	Differential HiSPi (SLVS) serial clock (positive). Qualified by the SLVS serial clock.
SLVS_CN	Output	Differential HiSPi (SLVS) serial clock (negative). Qualified by the SLVS serial clock.
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.

Figure 7: 48-Pin iLCC HiSPi Package Pinout Diagram

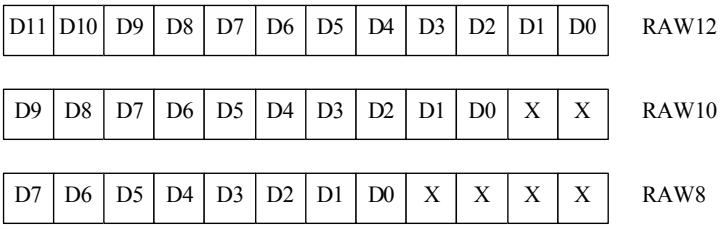


Output Data Format

Pixel Data Interface

The MT9F002 reads data out of the pixel array in a progressive scan over a High Speed serial data interface, or parallel data interface. RAW8, RAW10, and RAW12 image data formats are supported.

Figure 8: Data Formats



High Speed Serial Pixel Data Interface

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

- SLVS_CP
- SLVS_CN
- SLVS_0P
- SLVS_0N
- SLVS_1P
- SLVS_1N
- SLVS_2P
- SLVS_2N
- SLVS_3P
- SLVS_3N

The HiSPi interface supports the following protocols: Streaming-S and Packetized-SP. 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.

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.

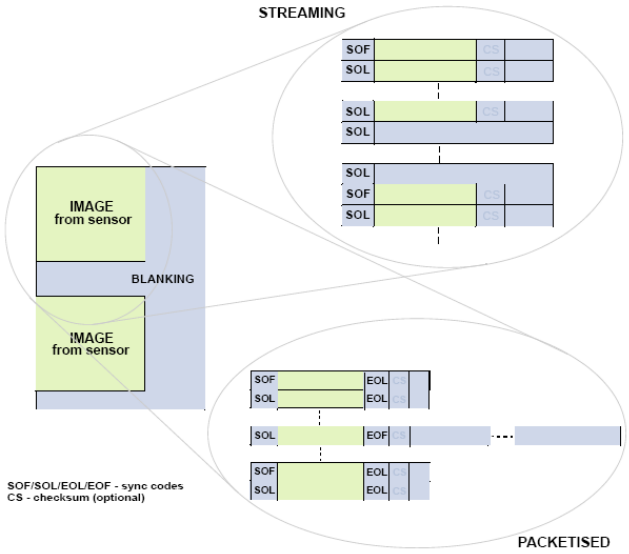
The HiSPi interface can only be configured when the sensor is in standby. This includes configuring the interface to transmit across 1, 2, or all 4 data lanes.

Protocol Fundamentals

Referring to Figure 9, 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 9: Steaming vs. Packetized Transmission



Note: See the High-Speed Serial Pixel (HiSPi)™ Protocol Specification V1.00.00 for HiSPi details.

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 HiSPi PHY uses a low voltage serial differential output. The HiSPi PHY drivers use a simple current steering driver scheme with two outputs that are complementary to each other (V_{OA} and V_{OB}). It is intended that these drivers be attached to short-length 100Ω differential interconnect to a receiver with a 100Ω termination. CL represents the total parasitic excess capacitance loading of the receiver and the interconnect.

There are two standards:

• Scalable Low Voltage Serial (SLVS) which has low amplitude and common-mode voltage (VCM) but scalable using an external supply.

• High VCM scalable serial interface (HiVCM), which has larger scalable amplitude and a high common-mode voltage.

Comparison of SLVS and HiVCM

Here is a comparison of the differences between SLVS and HiVCM.

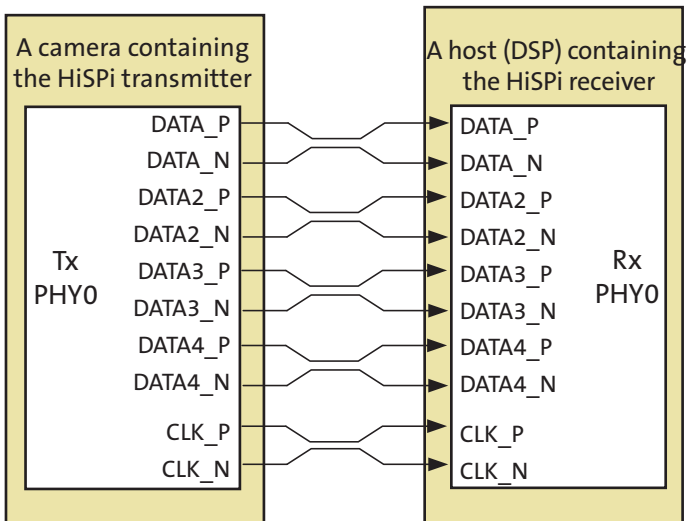
Table 4: SLVS and HiVCM Comparison

Parameter	HiVCM	SLVS
Typical Differential Amplitude[1]	280mV	200mV
Typical Common Mode[1]	0.9V	200mV
Typical Power Consumption[2]	45mW	4mW
Transmission Distance	Longer distance	Short distance
LVDS FPGA Receiver Compatible	Yes	No

Notes:
1. These are nominal values
2. Power from load driving stage, digital/serializer logic (VDD_HiSPi) not included.

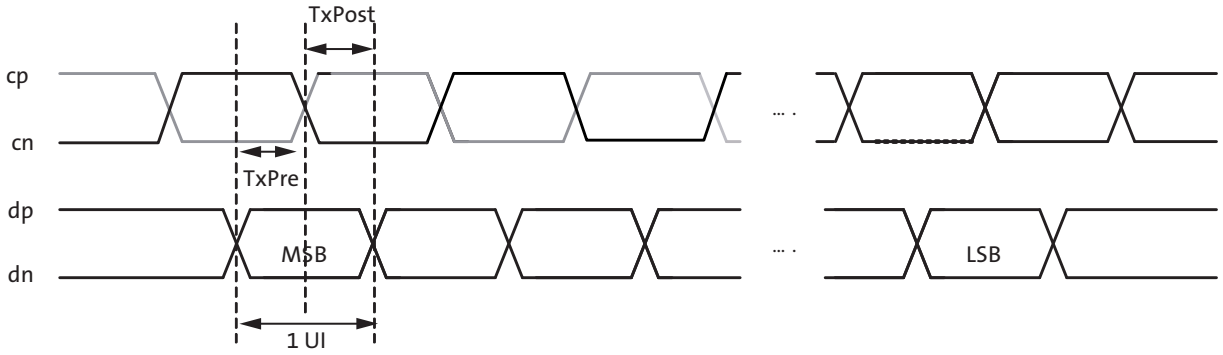
The HiSPi interface building block is a unidirectional differential serial interface with four data and one double data rate (DDR) clock lanes. The four Data lanes are 90 degrees out of phase with the Clock lanes. One clock for every four serial data lanes is provided for phase alignment across multiple lanes. Figure 10 shows the configuration between the HiSPi transmitter and the receiver.

Figure 10: HiSPi Transmitter and Receiver Interface Block Diagram



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 falling edge of clock. Figure 11 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 11: 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 12: Block Diagram of DLL Timing Adjustment

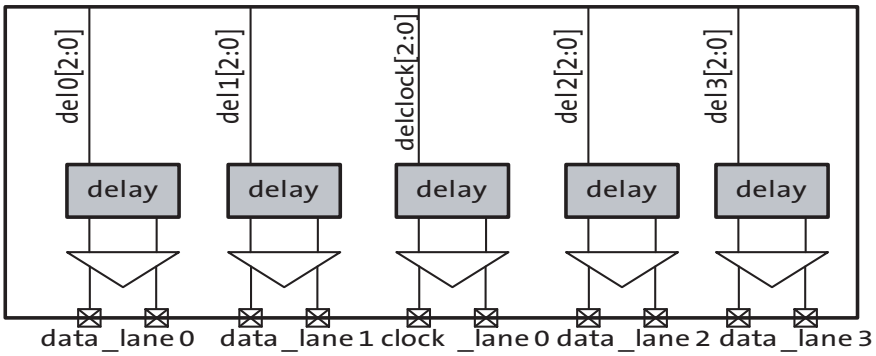


Figure 13: Delaying the clock_lane with Respect to data_lane

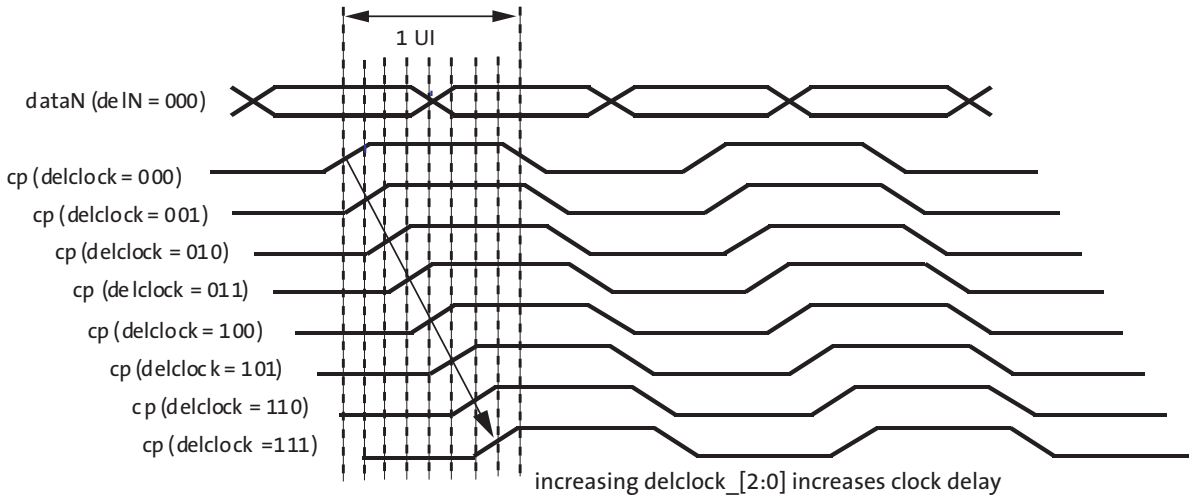
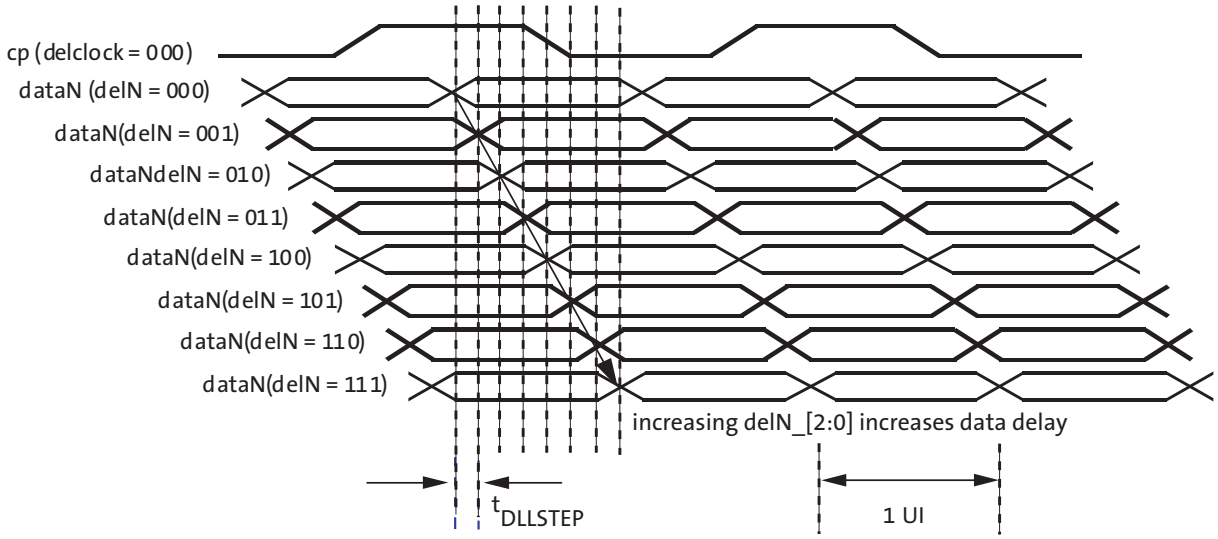


Figure 14: Delaying data_lane with Respect to the clock_lane

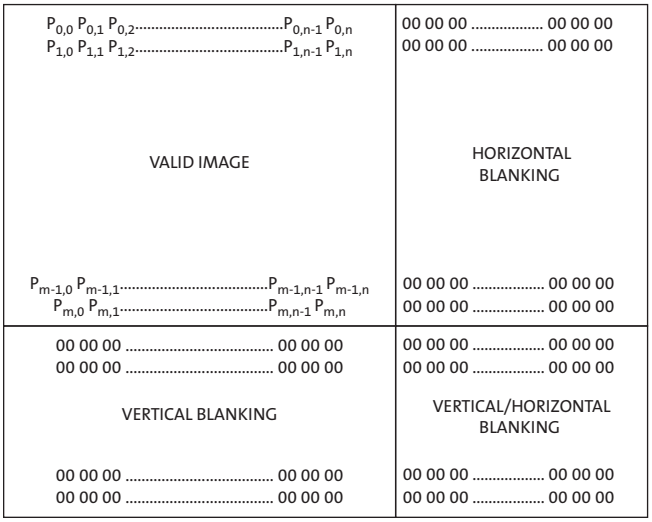


Note: See the High-Speed Serial Pixel (HiSPi)™ Physical Layer Specification V2.00.00 for details.

Parallel Pixel Data Interface

MT9F002 image data is read out in a progressive scan. Valid image data is surrounded by horizontal blanking and vertical blanking, as shown in Figure 15. 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 15: Spatial Illustration of Image Readout



Output Data Timing (Parallel Pixel Data Interface)

MT9F002 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 16). This allows PIXCLK to be used as a clock to sample the data. PIXCLK is continuously enabled, even during the blanking period. The MT9F002 can be programmed to delay the PIXCLK edge relative to the DOUT transitions. This can be achieved by programming the corresponding bits in the row_speed register.

Figure 16: Pixel Data Timing Example

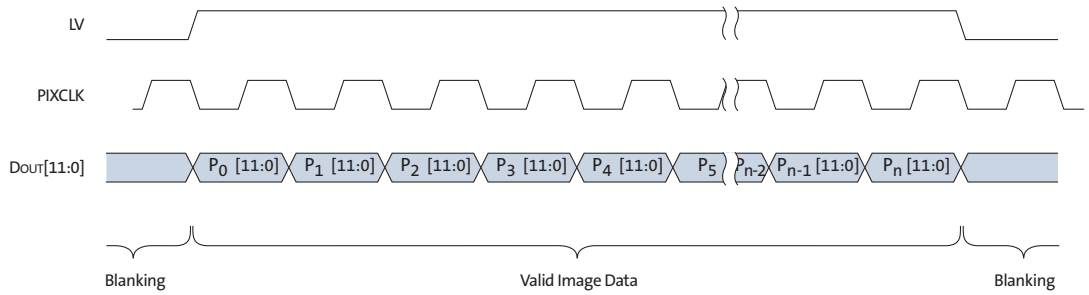
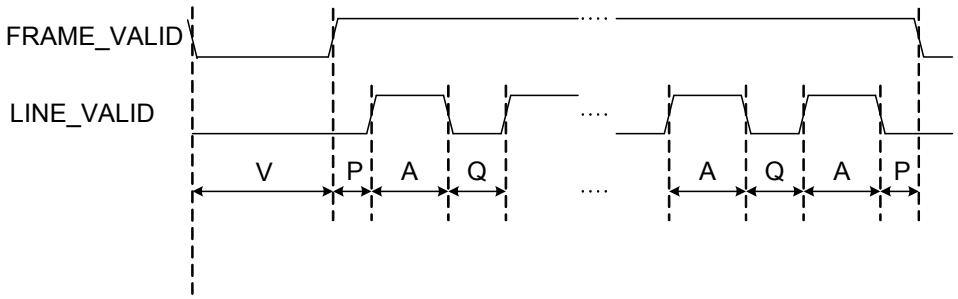


Figure 17: Frame Timing and FV/LV Signals



The sensor timing is shown in terms of pixel clock cycles (see Figure 16 on page 18). The default settings for the on-chip PLL generate a pixel array clock (vt_pix_clk) of 110 MHz and an output clock (op_pix_clk) of 55 MHz given a 24 MHz input clock to the MT9F002. Equations for calculating the frame rate are given in “Frame Rate Control” on page 48.

Table 5: Common Sensor Readout Modes

Key Readout Modes	Output Resolution	Aspect Ratio	DFOV: 7.67 mm (%)	Subsampling Mode	Frame Rate	ADC Effective Bit-Depth	Data Rate (Mbps/Lane)
14M Capture	4384H x 3288V	(4:3)	100	n/a	13.7	12	660
1080p +20% EIS (3Mp) Video	2304H x 1296V	(16:9)	96	x: Bin2 y: Bin2	60	10	550
	2304H x 1296V	(16:9)	96	x: Bin2 y: Bin2	30	10	275
720p +20%EIS (1.3Mp) Video	1536H x 864V	(16:9)	64	x: Bin2 y: Bin2	60	10	550
	1536H x 864V	(16:9)	64	x: Bin2 y: Bin2	30	10	275
VGA Video (High Quality)	1096H x 822V	(4:3)	100	x: Skip2Bin2 y: Bin4	60	10	550
EVF1 - Preview (Low Power)	1096H x 822V	(4:3)	100	x: Skip2Bin2 y: Bin4	30	10	275
EVF2 - Preview (Low Power)	1152H x 648V	(16:9)	96	x: Skip2Bin2 y: Bin4	30	10	275

Two-Wire Serial Register Interface

The two-wire serial interface bus enables read/write access to control and status registers within the MT9F002. The interface protocol uses a master/slave model in which a master controls one or more slave devices. The sensor acts as a slave device. The master generates a clock (SCLK) that is an input to the sensor and is used to synchronize transfers. Data is transferred between the master and the slave on a bidirectional signal (SDATA). SDATA is pulled up to VDD_IO off-chip by a 1.5kΩ resistor. Either the slave or master device can drive SDATA LOW—the interface protocol determines which device is allowed to drive SDATA at any given time.

The protocols described in the two-wire serial interface specification allow the slave device to drive SCLK LOW; the MT9F002 uses SCLK as an input only and therefore never drives it LOW.

Protocol

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

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

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

Start Condition

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

Stop Condition

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

Data Transfer

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

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

Slave Address/Data Direction Byte

Bits [7:1] of this byte represent the device slave address and bit [0] indicates the data transfer direction. A “0” in bit [0] indicates a WRITE, and a “1” indicates a READ. The default slave addresses used by the MT9F002 sensor are 0x20 (write address) and 0x21 (read address). Alternative slave addresses of 0x30 (write address) and 0x31 (read address) can be selected by enabling and asserting the SADDR signal through the GPI pin.

Alternate slave addresses can also be programmed through the i2c_ids register (R0x31FC-31FD). Note that this register needs to be unlocked through reset_register_lock_reg (R0x301A[3]) before is can be written to..

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.

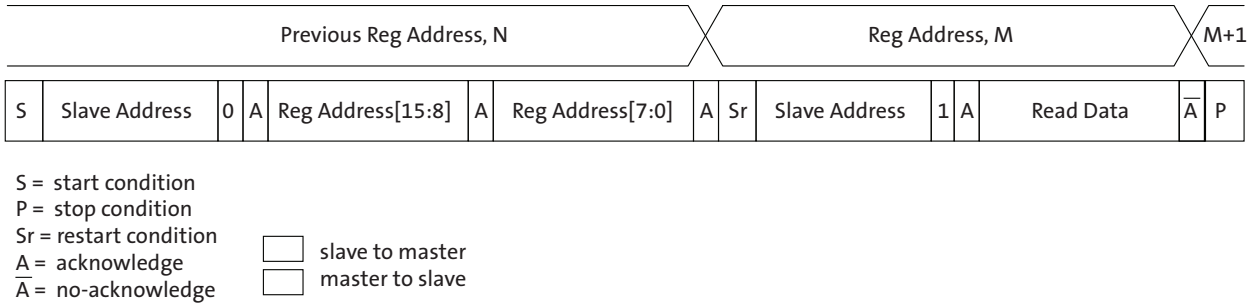
If the request was a WRITE, the master then transfers the 16-bit register address to which the WRITE should take place. This transfer takes place as two 8-bit sequences and the slave sends an acknowledge bit after each sequence to indicate that the byte has been received. The master then transfers the 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 18) 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 18 shows how the internal register address maintained by the MT9F002 is loaded and incremented as the sequence proceeds.

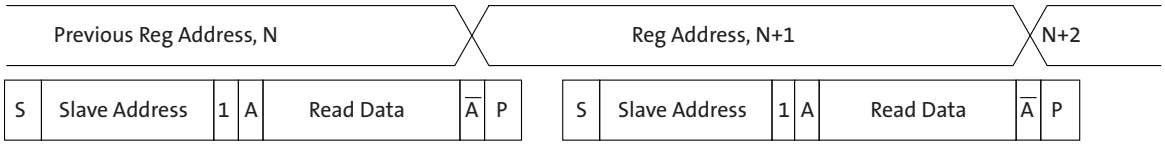
Figure 18: Single READ From Random Location



Single READ From Current Location

This sequence (Figure 19) performs a read using the current value of the MT9F002 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.

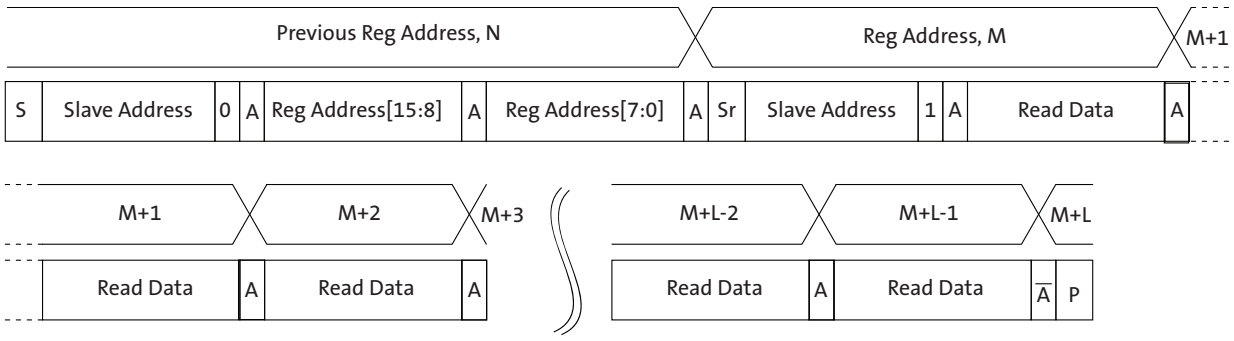
Figure 19: Single READ From Current Location



Sequential READ, Start From Random Location

This sequence (Figure 20) starts in the same way as the single READ from random location (Figure 18). 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.

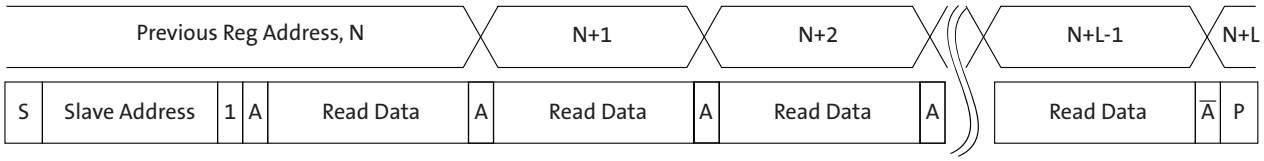
Figure 20: Sequential READ, Start From Random Location



Sequential READ, Start From Current Location

This sequence (Figure 21) starts in the same way as the single READ from current location (Figure 19 on page 22). 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.

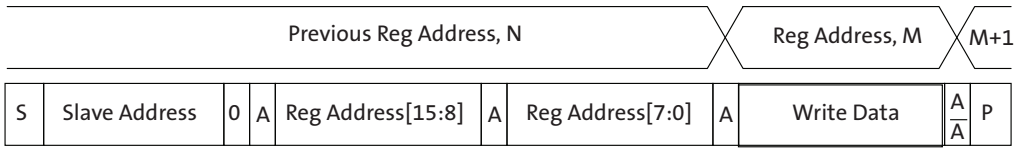
Figure 21: Sequential READ, Start From Current Location



Single WRITE to Random Location

This sequence (Figure 22) 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.

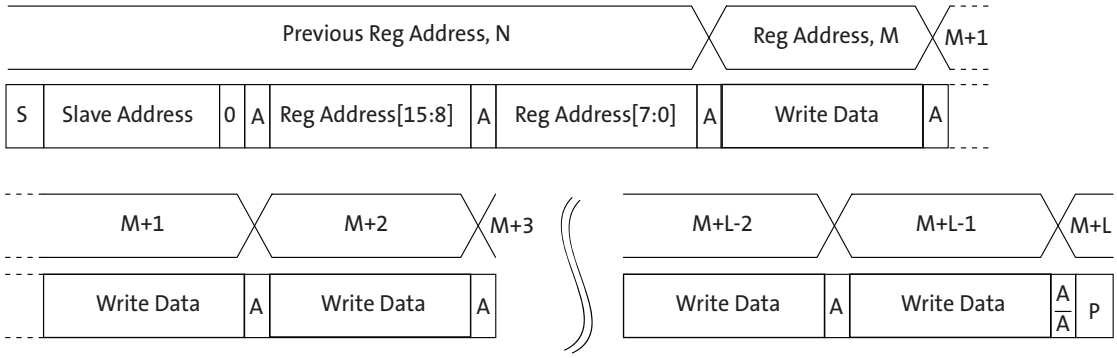
Figure 22: Single WRITE to Random Location



Sequential WRITE, Start at Random Location

This sequence (Figure 23) starts in the same way as the single WRITE to random location (Figure 22 on page 23). 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 23: Sequential WRITE, Start at Random Location



Programming Restrictions

The following sections list programming rules that must be adhered to for correct operation of the MT9F002. Refer to the MT9F002 Register Reference document for register programming details.

Table 6: 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

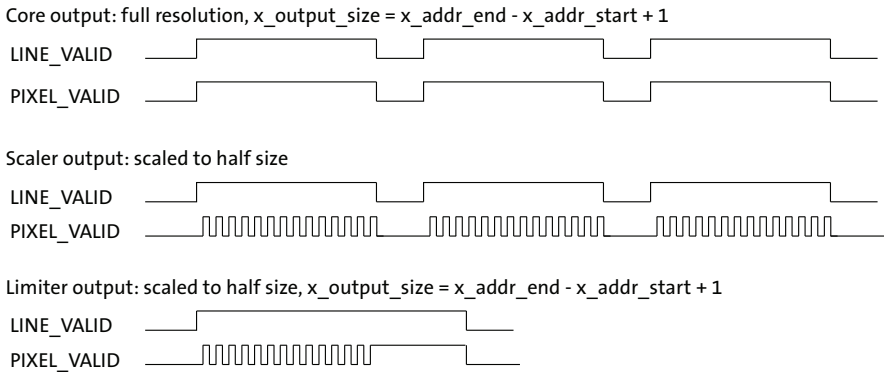
X Address Restrictions

The minimum column address available for the sensor is 24. The maximum value is 4647.

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 MT9F002 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 24.

Figure 24: Effect of Limiter on the Data Path



In Figure 24, three different stages in the data path (see “Timing Specifications” on page 68) 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.

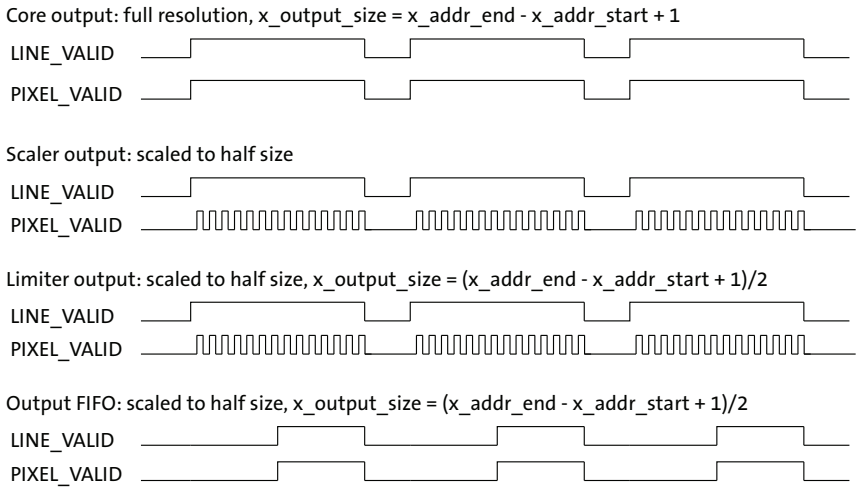
The second stage is the output of the scaler, when the scaler is set to reduce the image size by one-half in each dimension. The effect of the scaler is to combine groups of pixels. Therefore, the row time remains the same, but only half the pixels out of the scaler are valid. This is signaled by transitions 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 MT9F002 will cease to generate output frames.

A correct configuration is shown in Figure 25, 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 25 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 25: 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 signaled 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" on page 55.

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 pin)

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 that can be optionally enabled and controlled by a GPI pin to select an alternate slave address. These slave addresses can also be programmed through R0x31FC.

This interface is described in detail in “Two-Wire Serial Register Interface” on page 20.

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 8 on page 29 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 7. Selection of a pin to use for the OE_N function is described in "General Purpose Inputs" on page 32.

Table 7: 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 8.

Table 8: 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. 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 MT9F002 are represented as a state diagram in Figure 26 and described in subsequent sections. The effect of RESET_BAR on the system state and the configuration of the PLL in the different states are shown in Table 9 on page 31.

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

Figure 26: MT9F002 System States

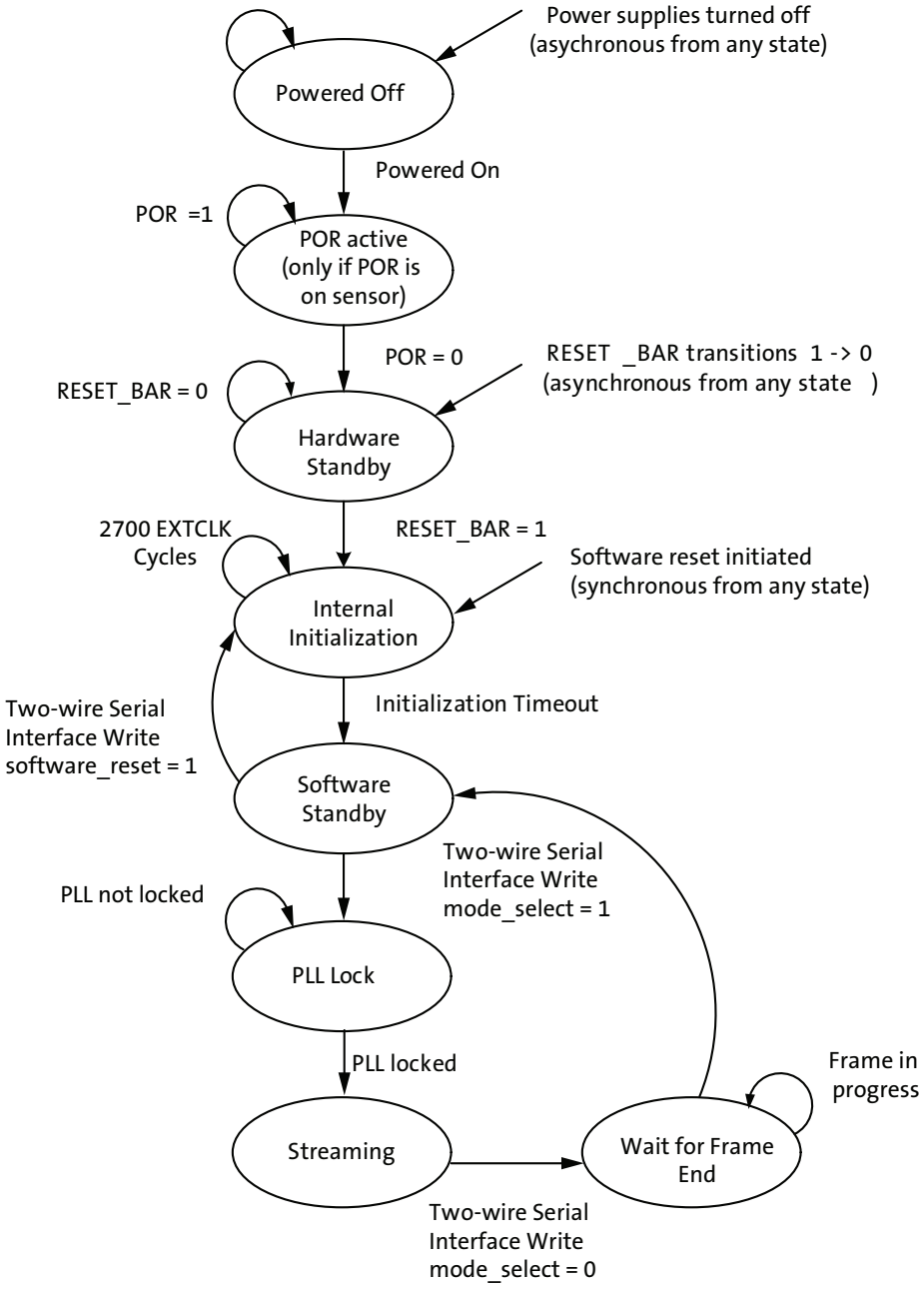


Table 9: RESET_BAR and PLL in System States

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

Note: VCO = voltage-controlled oscillator.

Power-On Reset Sequence

When power is applied to the MT9F002, 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 2700 EXTCLK cycles. After this, it enters a low-power software standby state. While the initialization sequence is in progress, the MT9F002 will not respond to READ transactions on its two-wire serial interface. Therefore, a method to determine when the initialization sequence has completed is to poll a sensor register; for example, R0x0000. While the initialization sequence is in progress, the sensor will not respond to its device address and READs from the sensor will result in a NACK on the two-wire serial interface bus. When the sequence has completed, READs will return the operational value for the register (0x2800 if R0x0000 is read).

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

Soft Reset Sequence

The MT9F002 can be reset under software control by writing “1” to software_reset (R0x0103). A software reset asynchronously resets the sensor, truncating any frame that is in progress. The sensor starts the internal initialization sequence, while the PLL and analog blocks are turned off. At this point, the behavior is exactly the same as for the power-on reset sequence.

Signal State During Reset

Table 10 on page 32 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 10: Signal State During Reset

Pad Name	Pad Type	Hardware Standby	Software Standby
EXTCLK	Input	Enabled. Must be driven to a valid logic level.	
RESET_BAR (XSHUTDOWN)			
GPI[3:0]		Powered down. Can be left disconnected/floating.	
TEST		Enabled. Must be driven to a logic 0.	
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			
SLVS_0P			
SLVS_0N			
SLVS_1P			
SLVS_1N			
SLVS_2P			
SLVS_2N			
SLVS_3P			
SLVS_3N			
SLVS_CP			
SLVS_CN			
FLASH		High-Z.	Logic 0.
SHUTTER			

General Purpose Inputs

The MT9F002 provides four general purpose inputs. After reset, the input pads associated with these signals are powered down by default, allowing the pads to be left disconnected/floating.

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

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

- Output enable (see “Output Enable Control” on page 28)
- Trigger/VD (slave mode) - see the sections below
- Standby functions
- SADDR selection (see “Serial Register Interface” on page 28)

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

Streaming/Standby Control

The MT9F002 can be switched between its soft standby and streaming states under pin or register control, as shown in Table 11. Selection of a pin to use for the STANDBY function is described in "General Purpose Inputs" on page 32. The state diagram for transitions between soft standby and streaming states is shown in Figure 26 on page 30.

Table 11: Streaming/STANDBY

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

Trigger Control

When the global reset feature is in use, the trigger for the sequence can be initiated either under pin or register control, as shown in Table 12. Selection of a pin to use for the TRIGGER function is described in "General Purpose Inputs" on page 32. In slave mode, the GPI pin also serves as VD signal input.

Table 12: Trigger Control

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

Clocking

The sensor contains a phase-locked loop (PLL) for timing generation and control. The PLL contains a prescaler to divide the input clock applied on EXTCLK, a VCO to multiply the prescaler output, and a set of dividers to generate the output clocks. The PLL structure is shown in Figure 27

Figure 27: Clocking Configuration

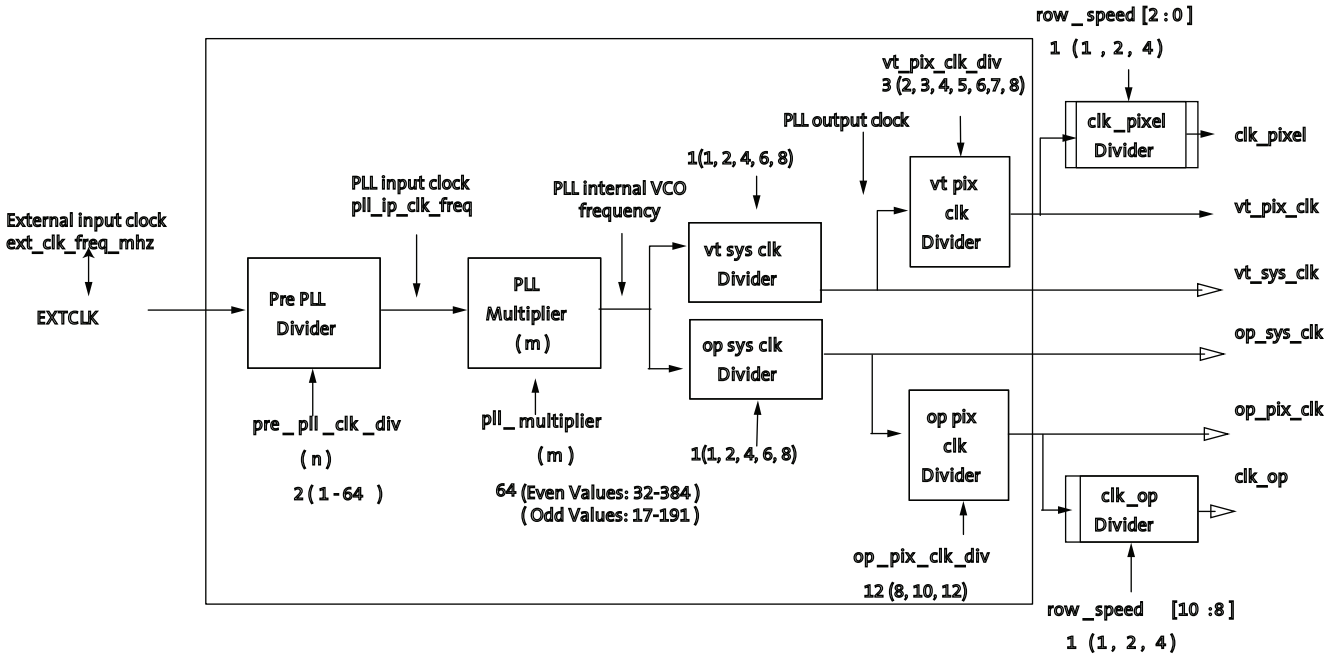


Table 13: PLL Parameter Range

Parameter	Symbol	Min	Max	Unit
External Input Frequency	fin	2	64	MHz
PLL Input (PFD) Frequency		2	24	MHz
VCO Clock Frequency	fvco	384	768	MHz

$$f_{PFD} = f_{in} / (n + 1), \quad 2\ MHz \le f_{PFD} \le 24\ MHz \qquad \text{(EQ 1)}$$

$$f_{VCO} = f_{in} * m / (n + 1), \quad 384\ MHz \le f_{VCO} \le 768\ MHz \qquad \text{(EQ 2)}$$

Figure 27 shows the different clocks and (in courier font) the names of the registers that contain or are used to control their values. Figure 27 also shows the default setting for each divider/multiplier control register and the range of legal values for each divider/multiplier control register. Default setup gives a physical 110 MHz internal clock for an input clock of 24 MHz. The maximum is 120 MHz.

From the diagram, the clock frequencies can be calculated as follows:

Note: Virtual pixel clock is used as the basis for frame timing equations.

$$vt\_pix\_clk = \frac{ext\_clk\_freq\_mhz \times pll\_multiplier \times (1 + shift\_vt\_pix\_clk\_div)}{pre\_pll\_clk\_div \times vt\_sys\_clk\_div \times vt\_pix\_clk\_div} = \frac{24\ MHz \times 165 \times 2}{6 \times 1 \times 6} = 220\ MHz \quad \text{(EQ 3)}$$

Internal pixel clock used to readout the pixel array:

$$clk\_pixel = \frac{ext\_clk\_freq\_mhz \times pll\_multiplier \times (1 + shift\_vt\_pix\_clk\_div)}{pre\_pll\_clk\_div \times vt\_sys\_clk\_div \times vt\_pix\_clk\_div \times 2 \times row\_speed[2:0]} = \frac{24\ MHz \times 165 \times 2}{6 \times 1 \times 6 \times 2 \times 1} = 110\ MHz \quad \text{(EQ 4)}$$

External pixel clock used to output the data:

$$clk\_op = \frac{ext\_clk\_freq\_mhz \times pll\_multiplier}{pre\_pll\_clk\_div \times op\_sys\_clk\_div \times op\_pix\_clk\_div \times row\_speed[10:8]} = \frac{24\ MHz \times 165}{6 \times 1 \times 12 \times 1} = 55\ MHz \quad \text{(EQ 5)}$$

Serial output clock:

$$op\_sys\_clk\_freq\_mhz = \frac{ext\_clk\_freq\_mhz \times pll\_multiplier}{pre\_pll\_clk\_div \times op\_sys\_clk\_div} = \frac{24\ MHx \times 165}{6 \times 1} = 660\ MHz \quad \text{(EQ 6)}$$

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 2-24 MHz. Lower 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 pre_pll_clk_div: 1-64.
- clk_op must never run faster than clk_pixel to ensure that the output data stream is contiguous.
- When the serial interface is used the clk_op divider cannot be used; row_speed[10:8] must equal 1.
- The value of op_sys_clk_div must match the bit-depth of the image when using serial interface. R0x0112-3 controls whether the pixel data interface will generate 12, 10, or 8 bits per pixel. When the pixel data interface is generating 8 bits per-pixel, op_pix_-clk_div must be programmed with the value 8. When the pixel data interface is generating 10 bits per pixel, op_pix_clk_div must be programmed with the value 10. And when the pixel data interface is generating 12 bits per pixel, op_pix_clk_div must be programmed with the value 12. This is not required when using the parallel interface.
- Although the PLL VCO input frequency range is advertised as 2-24 MHz, superior performance (better PLL stability) 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 two 10-bit pixels each clk_pixel period. The line length (line_length_pck) and fine integration time (fine_integration_time) are controlled in increments of half 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 clk_op period. This clock also equals the output PIXCLK.
- Master clock frequency corresponds to vt_pix_clk/2.
- Serial clock (op_sys_clk) used for the serial output interface.

Programming the PLL Divisors

The PLL divisors must be programmed while the MT9F002 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 MT9F002 is in the streaming state is undefined.

Clock Control

The MT9F002 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 MT9F002 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 MT9F002 supports scaling capability. Scaling is a “zoom out” operation to reduce the size of the output image while covering the same extent as the original image. That is, low resolution images can be generated with full field-of-view. Each scaled output pixel is calculated by taking a weighted average of a group input pixels which is composed of neighboring pixels. The input and output of the scaler is in Bayer format.

When compared to skipping, scaling is advantageous because it uses all pixel values to calculate the output image which helps avoid aliasing. Also, it is also more convenient than binning because the scale factor varies smoothly and the user is not limited to certain ratios of size reduction.

The MT9F002 sensor is capable of horizontal scaling and full (horizontal and vertical) scaling.

The scaling factor is programmable in 1/16 steps and is determined by.

$$ScaleFactor = \frac{scale\_n}{scale\_m} = \frac{16}{scale\_m} \quad \text{(EQ 7)}$$

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).

Scaler Example

When horizontal and vertical scaling is enabled for a 1:2 scale factor, an image is reduced by half in both the horizontal and vertical directions. This results in an output image that is one-fourth of the original image size. This can be achieved with the following register settings:
R0x0400 = 0x0002 // horizontal and vertical scaling mode
R0x0402 = 0x0020 // scale factor m = 32

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 MT9F002 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:

$$Pcorrected(row, col) = Psensor(row,col) * f(row,col) \quad \text{(EQ 8)}$$

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.

Sensor Readout Configuration

Image Acquisition Modes

The MT9F002 supports two image acquisition modes:

1. Electronic rolling shutter (ERS) mode
   This is the normal mode of operation. When the MT9F002 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 MT9F002 switches cleanly from the old integration time to the new while only generating frames with uniform integration. See “Changes to Integration Time” in the MT9F002 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 MT9F002 provides control signals to interface to that shutter. The operation of this mode is described in detail in "Global Reset" on page 55.

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 HiSPi interfaces, the output image size is controlled by the x_output_size and y_output_size registers.

Pixel Border

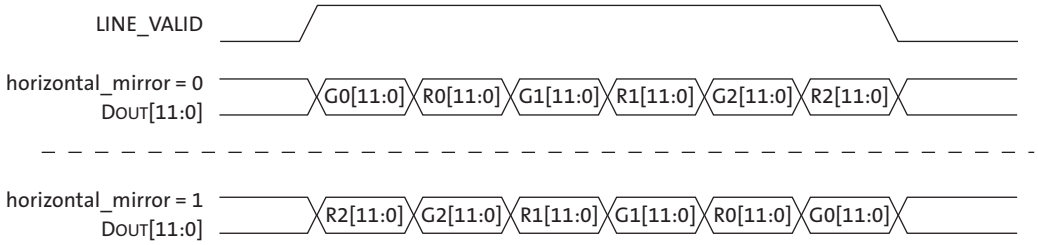
The default settings of the sensor provide a 4608H x3288V image. A border of up to 8 pixels (4 in binning) on each edge can be enabled by reprogramming the x_addr_start, y_addr_start, x_addr_end, y_addr_end, x_output_size, and y_output_size registers accordingly. This provides a total active pixel array of 4640H x 3320V 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 28 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 28: Effect of Horizontal Mirror on Readout Order



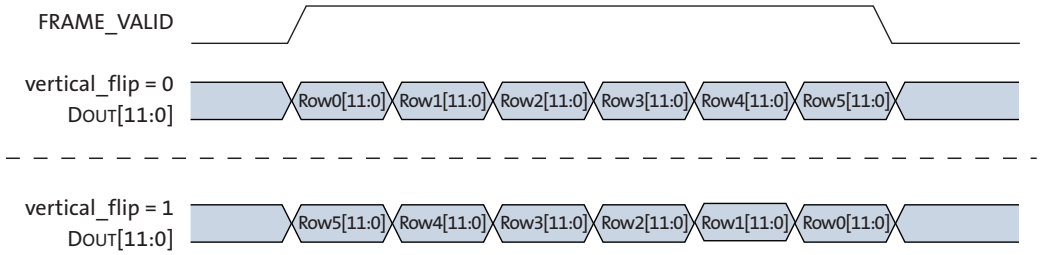
To enable image horizontal mirror mode, set register bit R0x3040[14]=1.

- 0 = Normal readout
- 1 = Readout is mirrored horizontally so that the column specified by x_addr_end_ is read out of the sensor first.

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 29 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 29: Effect of Vertical Flip on Readout Order



To enable image vertical flip mode, set register bit R0x3040[15]=1.

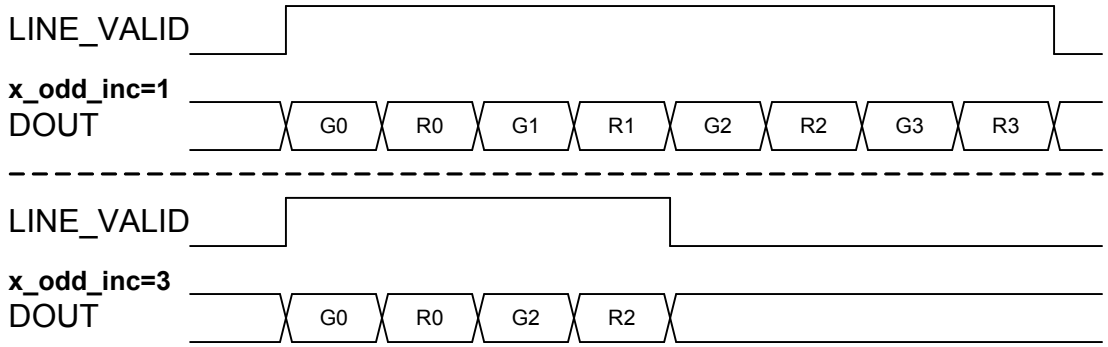
- 0 = Normal readout
- 1 = Readout is flipped vertically so that the row specified by y_addr_end_ is read out of the sensor first.

Subsampling

The MT9F002 supports subsampling. subsampling reduces the amount of data processed by the analogue signal chain in the sensor and thereby allows the frame rate to be increased. subsampling is enabled by changing x_odd_inc and/or y_odd_inc. Values of 1, 3 and 7 can be supported for x_odd_inc, while values 1, 3, 7, 15 and 31 can be supported for y_odd_inc.

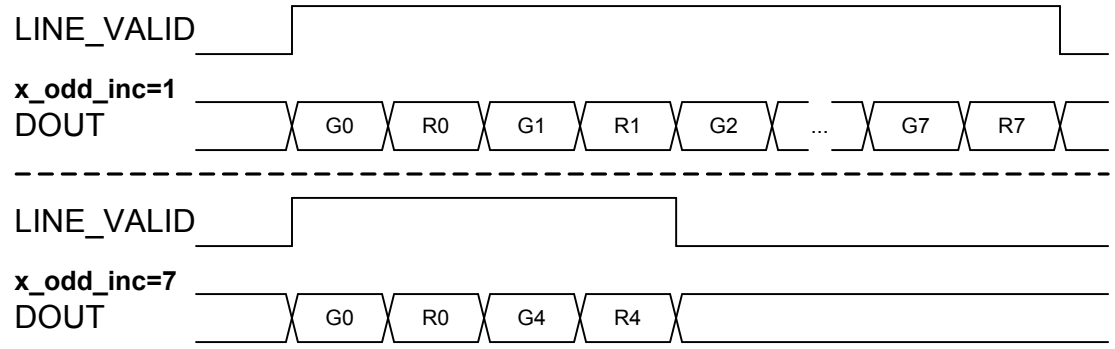
Setting both of these variables to 3 reduces the amount of row and column data processed and is equivalent to the skip2 readout mode provided by earlier Micron Imaging sensors. Figure 3 shows a sequence of 8 columns being read out with x_odd_inc=3 and y_odd_inc=1.

Figure 30: Effect of x_odd_inc=3 on readout sequence



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

Figure 31: Effect of x_odd_inc=7 on readout sequence



The effect of the different subsampling settings on the pixel array readout is shown in Figure 32through Figure 38.

Figure 32: Pixel Readout (no subsampling)

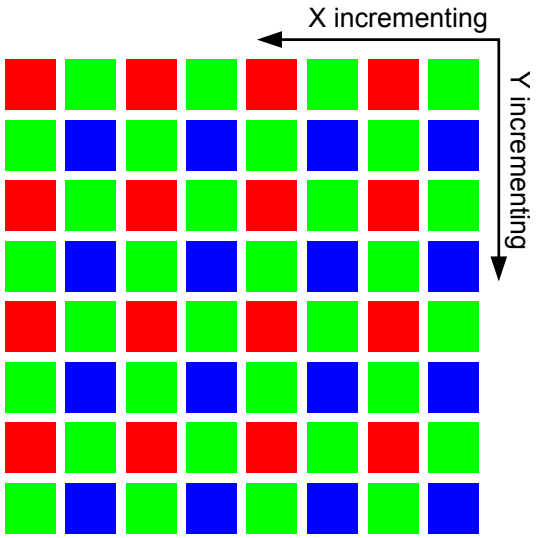


Figure 33: Pixel Readout (x_odd_inc=3, y_odd_inc=1)

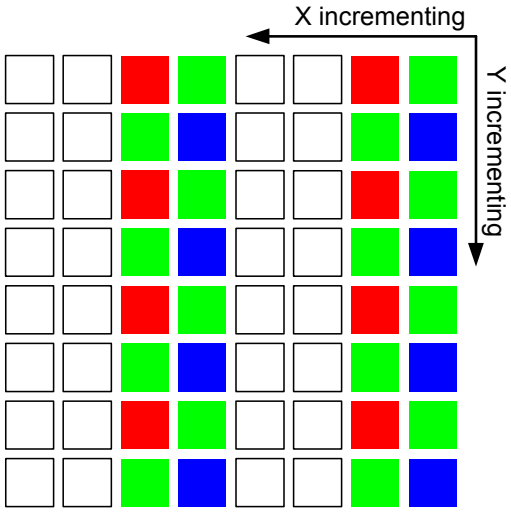


Figure 34: Pixel Readout (x_odd_inc=1, y_odd_inc=3)

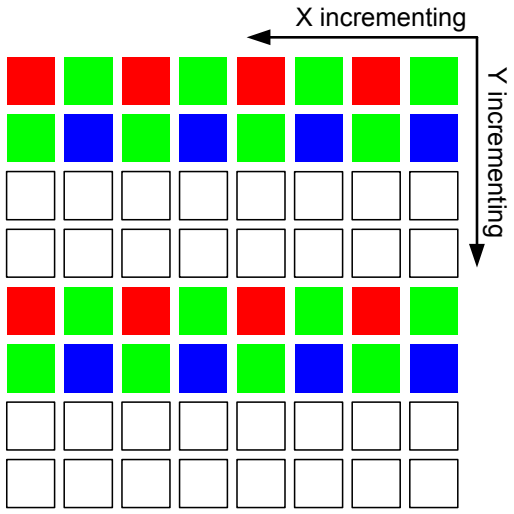


Figure 35: Pixel Readout (x_odd_inc=3, y_odd_inc=3)

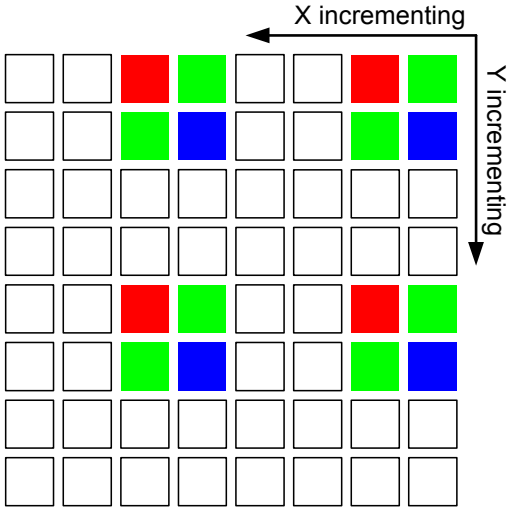


Figure 36: Pixel Readout (x_odd_inc=7, y_odd_inc=7)

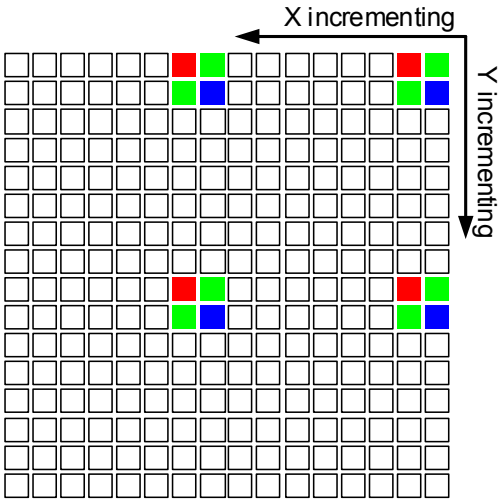


Figure 37: Pixel Readout (x_odd_inc=7, y_odd_inc=15)

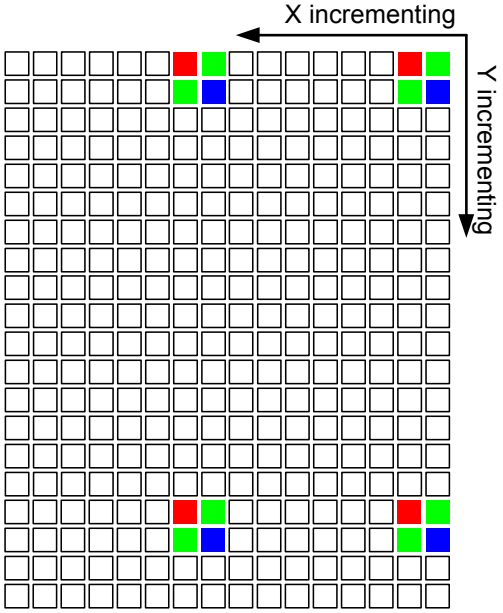
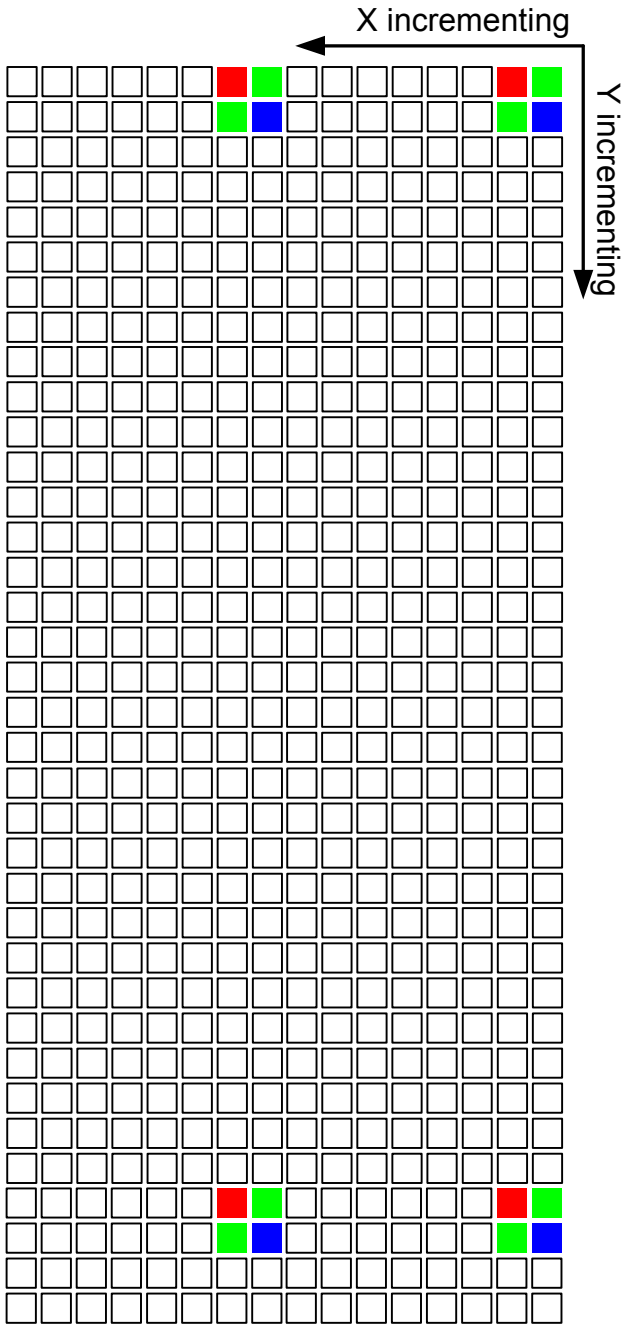


Figure 38: Pixel Readout (x_odd_inc=7, y_odd_inc=31)



Programming Restrictions When Subsampling

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

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

x_skip_factor = (x_odd_inc + 1) / 2

y_skip_factor = (y_odd_inc + 1) / 2

- x_addr_start should be a multiple of x_skip_factor*8
- (x_addr_end - x_addr_start + x_odd_inc) should be a multiple of x_skip_factor*8

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

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

Summing Mode

Summing can be enabled with binning. Unlike binning mode where the values of adjacent same color pixels are averaged together, summing adds the pixel values together, resulting in better sensor sensitivity. Summing normally provides two times the sensitivity compared to the binning only mode.

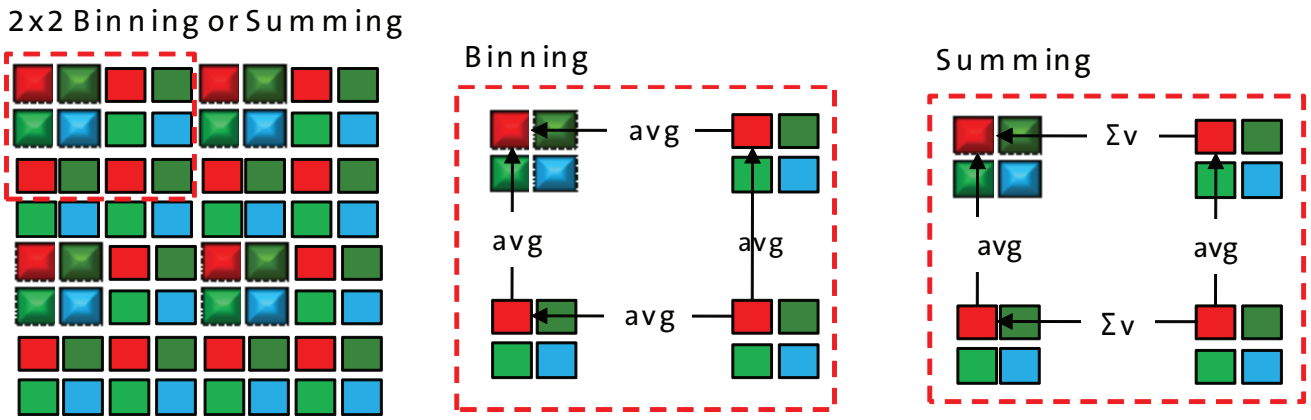
The 2x2 summing mode can be enabled by programming the following register bit fields:

R0x3178[5:4] = 3

R0x3178[7:6] = 1

To disable summing, program register bit fields above to 0.

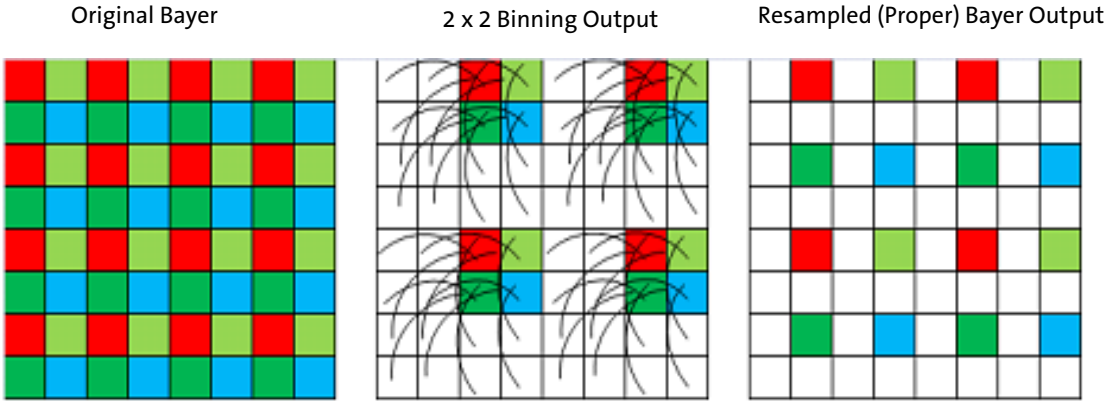
Figure 39: Pixel Binning and Summing



Bayer Resampler

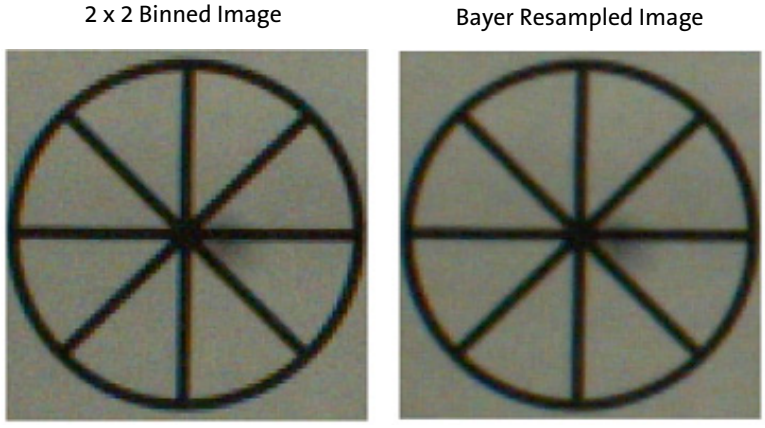
The imaging artifacts found from a 2 x 2 binning will show image artifacts from aliasing. These can be corrected by resampling the sampled pixels in order to filter these artifacts. Figure 40 shows the pixel location resulting from 2 x 2 binning located in the middle diagram, and the resulting pixel locations after the Bayer resampling function has been applied.

Figure 40: Bayer Resampling



The improvements from using the Bayer resampling feature can be seen in Figure 41. In this example, image edges seen on a diagonal have smoother edges when the Bayer resampling feature is applied. This feature is designed to be used only with modes configured with 2 x 2 binning. The feature will not remove aliasing artifacts that are caused skipping pixels.

Figure 41: Results of Resampling



To enable the Bayer resampling feature:

1. Set 0x0400 to 0x02 // Enable the on-chip scalar.
2. Set 0x306E to 0x90B0 // Configure the on-chip scalar to resample Bayer data.

To disable the Bayer resampling feature:

1. Set 0x0400 to 0x00 // Disable the on-chip scalar.
2. Set 0x306E to 0x9080 // Configure the on-chip scalar to resample Bayer data.

Frame Rate Control

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

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

$$minimum\_line\_length = \frac{x\_addr\_end - x\_addr\_start + 1}{subsampling\ factor} + min\_line\_blanking\_pck \quad \text{(EQ 9)}$$

Note that line_length_pck also needs to meet the minimum line length requirement set in register min_line_length_pck. The row time can either be limited by the time it takes to sample and reset the pixel array for each row, or by the time it takes to sample and read out a row. Values for min_line_blanking_pck are provided in Table 14 on page 49.

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 10:

$$minimum\ frame\_length\_lines = \left(\frac{y\_addr\_end - y\_addr\_start + 1}{subsampling\ factor} + min\_frame\_blanking\_lines\right) \quad \text{(EQ 10)}$$

The frame rate can be calculated from these variables and the pixel clock speed as shown in Equation 11:

$$frame\ rate = \frac{vt\ pixel\ clock\ mhz \times 1 \times 10^6}{line\_length\_pck \times frame\_length\_lines} \quad \text{(EQ 11)}$$

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 14.

Table 14: 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	0x0138	0x0138	0x0138	0x00E8	0x00E8	0x00E8
min_line_length_pck	0x04C8	0x0278	0x0278	0x0968	0x04B8	0x0260

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
2. line_length_pck ≥ 0.5*(x_addr_end - x_addr_start + x_odd_inc)/((1+x_odd_inc)/2) + min_line_blanking_pck
3. The row time must allow the FIFO to output all data during each row. That is,

- For parallel interface:
  line_length_pck > (x_output_size) * “vt_pix_clk period” / “op_pix_clk period” + 0x005E
- For 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 15: Minimum Frame Time and Blanking Numbers

Register	
min_frame_blanking_lines	0x0092
min_frame_length_lines	0x0094

Integration Time

The integration (exposure) time of the MT9F002 is controlled by the fine_integration_-time and coarse_integration_time registers.

The limits for the fine integration time are defined by:

$$fine\_integration\_time\_min \leq fine\_integration\_time \leq (line\_length\_pck - fine\_integration\_time\_max\_margin \quad \text{(EQ 12)}$$

The limits for the coarse integration time are defined by:

$$coarse\_integration\_time\_min \leq coarse\_integration\_time \quad \text{(EQ 13)}$$

The actual integration time is given by:

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

It is required that:

$$coarse\_integration\_time <= (frame\_length\_lines - coarse\_integration\_time\_max\_margin) \quad \text{(EQ 15)}$$

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.

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. It is necessary to change fine_correction (R0x3010) when binning is enabled or the pixel clock divider (row_speed[2:0]) is used. The corresponding fine_correction values are shown in Table 16.

Table 16: Fine_Integration_Time Limits

Register	No Row Binning			Row Binning		
row_speed[2:0]	1	2	4	1	2	4
fine_integration_time_min	0x02B0	0x0158	0x0AC	0x05F2	0x02FA	0x017E
fine_integration_time_max_margin	0x0212	0x0109	0x0086	0x0376	0x01BA	0x00DC

Fine Correction

For the fine_integration_time limits, the fine_correction constant will change with the pixel clock speed and binning mode.

Table 17: Fine_Correction Values

Register	No Row Binning			Row Binning		
row_speed[2:0]	1	2	4	1	2	4
fine_correction	0x094	0x044	0x01C	0x0183	0x0BB	0x057

Power Mode Contexts

The MT9F002 sensor supports power consumption optimization through the power mode contexts. Depending on the sensor operating mode, the appropriate power context can be programmed through register R0x30E8 as shown in Table 18 below.

Programming register R0x30E8 will internally set the analog bias current reserved registers to predetermined values which result in optimized bias currents in the analog domain. Register R0x30E8 is not “Frame Sync'd,” and should be programmed when FRAME_VALID is not active, in order to avoid a “Bad Frame.”

Table 18: Power Mode Contexts

Power Mode Context	Register Address	Recommended Value	Description
1	R0x30E8	0x8001	Reserved
2	R0x30E8	0x8002	Reserved
3	R0x30E8	0x8003	Reserved
4	Rr0x30E8	0x8004	Reserved
5	R0x30E8	0x8005	Reserved
6	R0x30E8	0x8006	Reserved
7	R0x30E8	0x8007	Reserved

ON Semiconductor Gain Model

The ON Semiconductor 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 analog gain stages. The analog_gain_2 analog gain stage has a granularity of 64 steps over 2x gain. A digital gain (GAIN<15:12>) from 1-15x can also be applied.

$$analog\ gain = 2\text{^}GAIN\langle 11:10\rangle \times 2\text{^}GAIN\langle 9:7\rangle \times GAIN\langle 6:0\rangle/64 \quad (EQ\ 16)$$

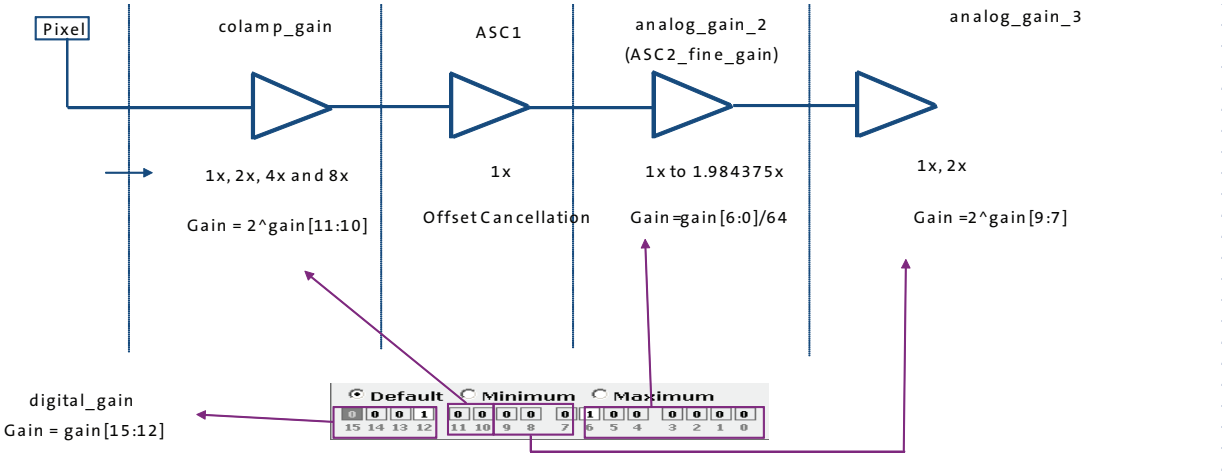
$$digital\_gain = GAIN\langle 15:12\rangle \quad (EQ\ 17)$$

$$Total\ gain = digital\_gain \times analog\_gain \quad (EQ\ 18)$$

Analog Gain Stages

The analog gain stages of the MT9F002 sensor are shown in Figure 1. The recommended gain settings enable gain increases very early in the signal chain (such as in the colamp), so the signal can be effectively boosted while amplifying as few noise sources as possible.

Figure 42: Analog Gain Stages



As a result of the different gain stages, analog gain levels can be achieved in different ways. The recommended gain settings are shown in Table 19 on page 53.

Table 19: Recommended Register Settings

Gain Range	Register Setting	Colamp_gain	Analog_gain 3	Analog_gain_2	Digital Gain
1.50 - 2.969	0x1430 - 0x145F	2x	1x	0.75 - 1.484	1x
3.00 - 5.938	0x1830 - 0x185F	4x	1x	0.75 - 1.484	1x
6.00 - 15.875	0x1C30 - 0x1C7F	8x	1x	0.75 - 1.984	1x
16.00 - 31.75	0x2C40 - 0x2C7F	8x	1x	1.00 - 1.984	2x
32.00 - 63.50	0x4C40 - 0x4C7F	8x	1x	1.00 - 1.984	4x

Note: These gain settings reflects maximizing the front-end Colamp_gain, while meeting the minimum requirement of 0.75 for the Analog_gain_2 stage.

In order to ensure ADC saturation, the recommended minimum gain (minimum ISO speed equivalent gain) setting for the MT9F002 sensor (Rev3) is 1.50.

Also, the recommended maximum analog gain is 15.875. For total gain values greater than 15.875, use or increase digital gain.

Flash Control

The MT9F002 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 43, and in Figure 44 and Figure 45 on page 54. 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 45 on page 54. Read-only bit flash[14] is set during frames that are correctly integrated; the state of this bit is shown in Figures 43, 44, and 45.

Figure 43: Xenon Flash Enabled

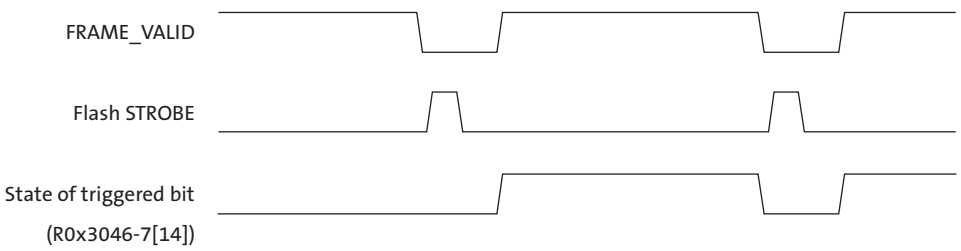
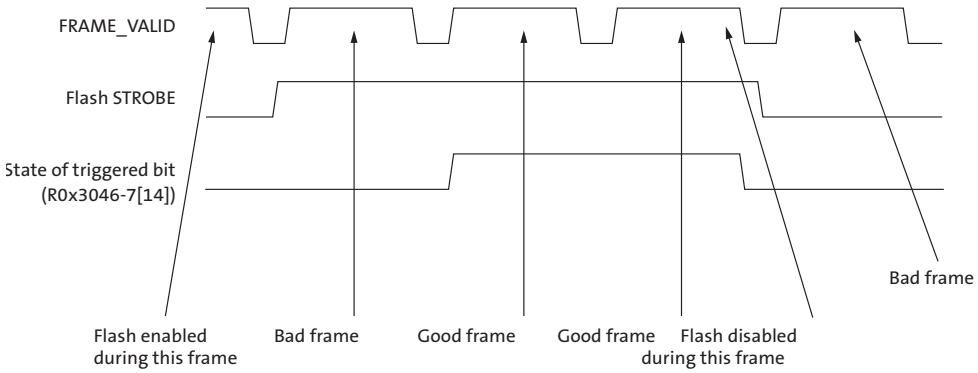
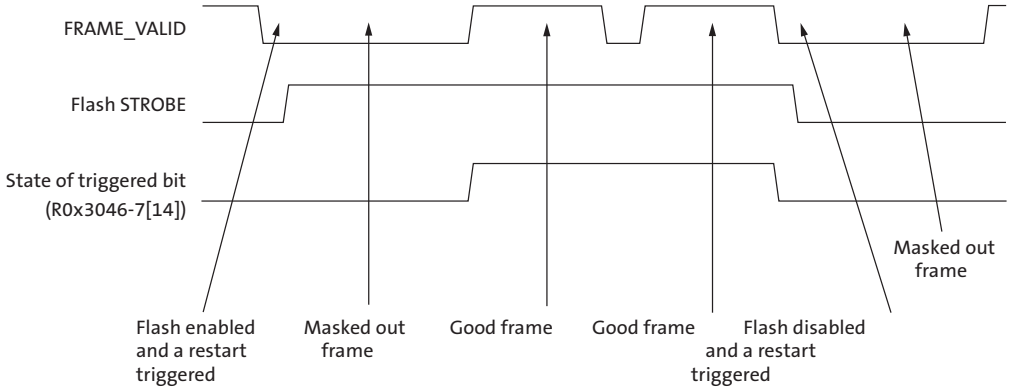


Figure 44: LED Flash Enabled



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

Figure 45: LED Flash Enabled Following Forced Restart



Global Reset

Global reset mode allows the integration time of the MT9F002 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.

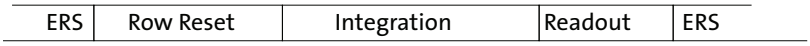
Overview of Global Reset Sequence

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 MT9F002), 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 46. The following sections expand to show how the timing of this sequence is controlled.

Figure 46: Overview of Global Reset Sequence



Entering and Leaving the Global Reset Sequence

A global reset sequence can be triggered by a register write to global_seq_trigger[0] (global trigger, to transition this bit from a 0 to a 1) or by a rising edge on a suitably-configured GPI input (see “Trigger Control” on page 33).

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.

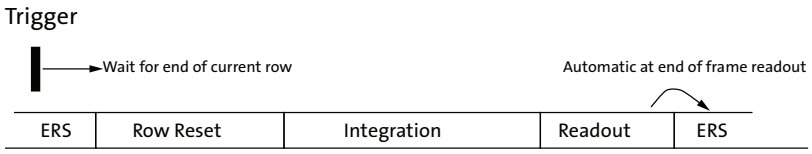
The global reset sequence completes with a frame readout. At the end of this readout phase, the sensor automatically resumes operation in ERS mode. The first frame integrated with ERS will be generated after a delay of approximately:

*((13 + coarse_integration_time) * line_length_pck).*

This sequence is shown in Figure 47.

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 47: Entering and Leaving a Global Reset Sequence

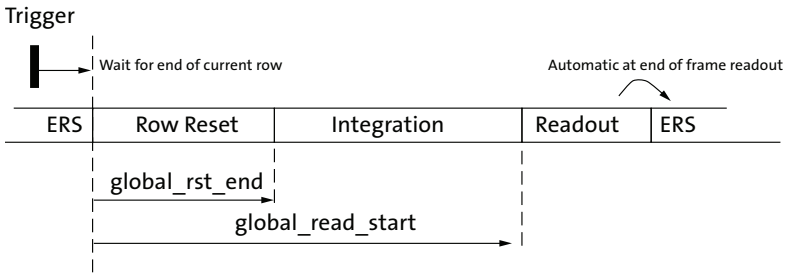


Programmable Settings

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 48. 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.

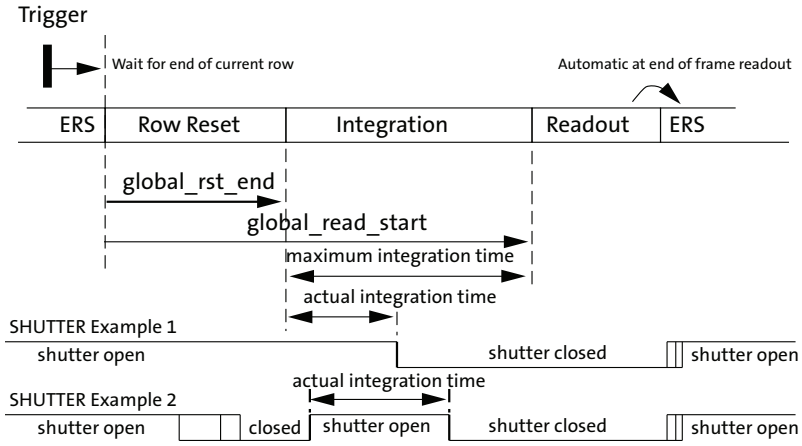
Figure 48: Controlling the Reset and Integration Phases of the Global Reset Sequence



Control of the Electromechanical Shutter

Figure 49 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 49: 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).

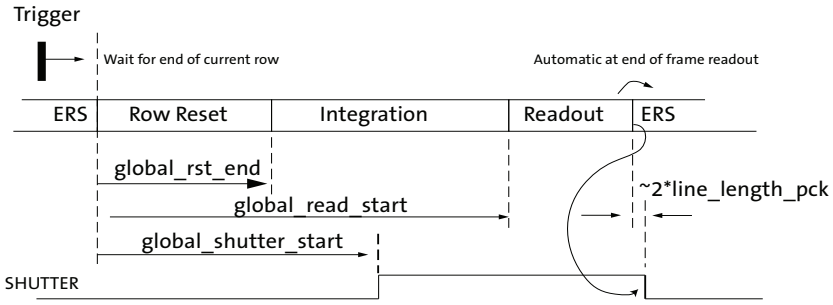
It is essential that the shutter closes before the end of the integration phase. If the row readout phase is allowed to start before the shutter closes, each row in turn will be integrated for one row-time longer than the previous row.

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 MT9F002 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 50 on page 58. 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

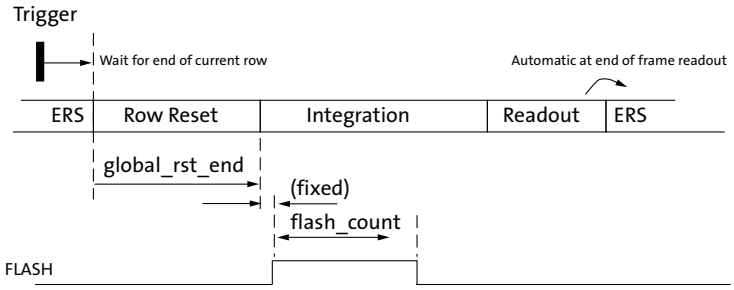
Figure 50: Controlling the SHUTTER Output



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 51.

Figure 51: 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 52 on page 59. 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:

$$Integration\ Time = \frac{global\_scale \times [global\_read\_start - global\_shutter\_start - global\_rst\_end]}{vt\_pix\_clk\_freq\_mhz} \quad \text{(EQ 19)}$$

Where:

$$global\_read\_start = (2^{16} \times global\_read\_start2[7:0] + global\_read\_start1[15:0]) \quad \text{(EQ 20)}$$

$$global\_shutter\_start = (2^{16} \times global\_shutter\_start2[7:0] + global\_shutter\_start1[15:0]) \quad \text{(EQ 21)}$$

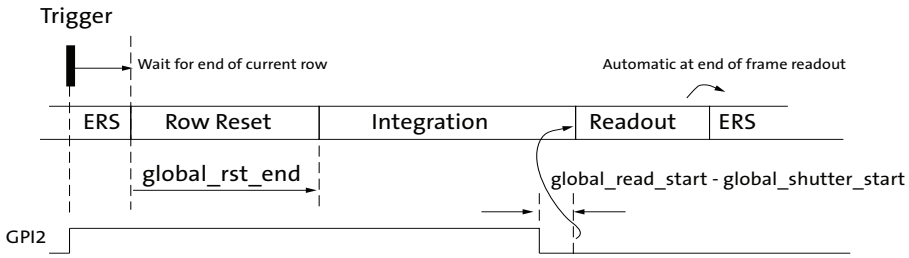
The integration equation allows for 24-bit precision when calculating both the shutter and readout of the image. The global_rst_end has only 16-bit as the array reset function and requires a short amount of time.

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 52: 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.

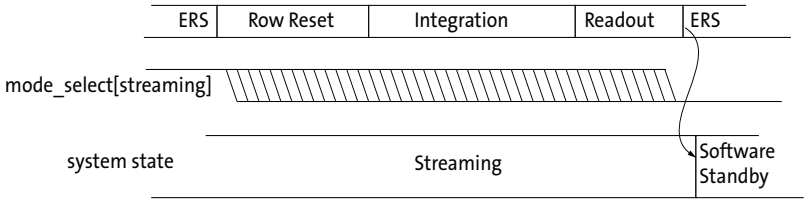
The earliest time that the global reset sequence can be retriggered is the point at which the SHUTTER output de-asserts; this occurs approximately *2 * line_length_pck* after the negation of FV for the global reset readout phase.

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 MT9F002 will remain in streaming state until the global reset sequence (including frame readout) has completed, as shown in Figure 53.

Figure 53: Entering Soft Standby During a Global Reset Sequence



Slave Mode

The MT9F002 sensor supports Slave mode to sync the frame rate more precisely, and simply by the VD signal from external ASIC. The VD signal also allows for precise control of frame rate and register change updates.

The VD signal for slave GRR mode is synchronized to ERS frame time, so that sensor can complete the current frame readout in ERS mode before moving to GRR mode, and avoid ERS broken frame before moving into GRR mode. Control bit vd_trigger_new_-frame bit allows VD triggering every new frame.

A GPI pin on the sensor can be programmed to act as VD input pin signal whose rising edge can be used to start every new frame (see Figure 55 for details).

An optional functionality to limit the duration counters are halted is given by setting vd_timer bit to 1. When this bit is set the counters will not wait indefinitely for VD rising edge & resume normal counting after halting for a limited time. Otherwise when vd_timer is set to 0, internal row and column counters are halted until the arrival of VD's positive edge.

Slave Mode GRR

Global reset sequence is triggered by programming the global_seq_trigger bit. After this register bit is written the sensor will wait for rising edge of VD signal at the end of the current frame to go into GRR mode. The control bit needed to be set to enable this functionality is vd_trigger_grst. Once in the GRR integration phase, the sensor will wait for the next VD rising edge to begin the readout.

At the end of the readout phase, the sensor automatically resumes operation in ERS mode with readout of successive frames starting with rising edge of VD. Figure 54: “Slave Mode GRR Timing,” on page 61 and Figure 55: “Slave Mode HiSPi Output (ERS to GRR Transition),” on page 61 are related timing diagrams:

Slave Mode GRR Timing

For example, to switch between ERS and GRR (and back to ERS), see Figure 54:

Figure 54: Slave Mode GRR Timing

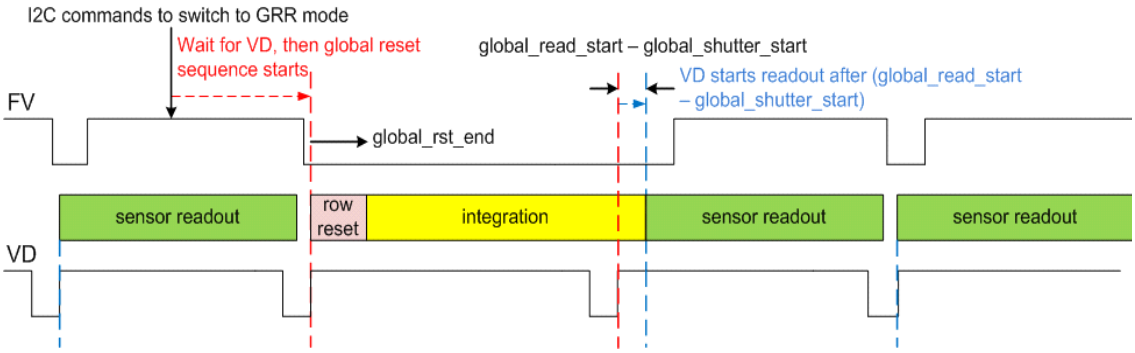
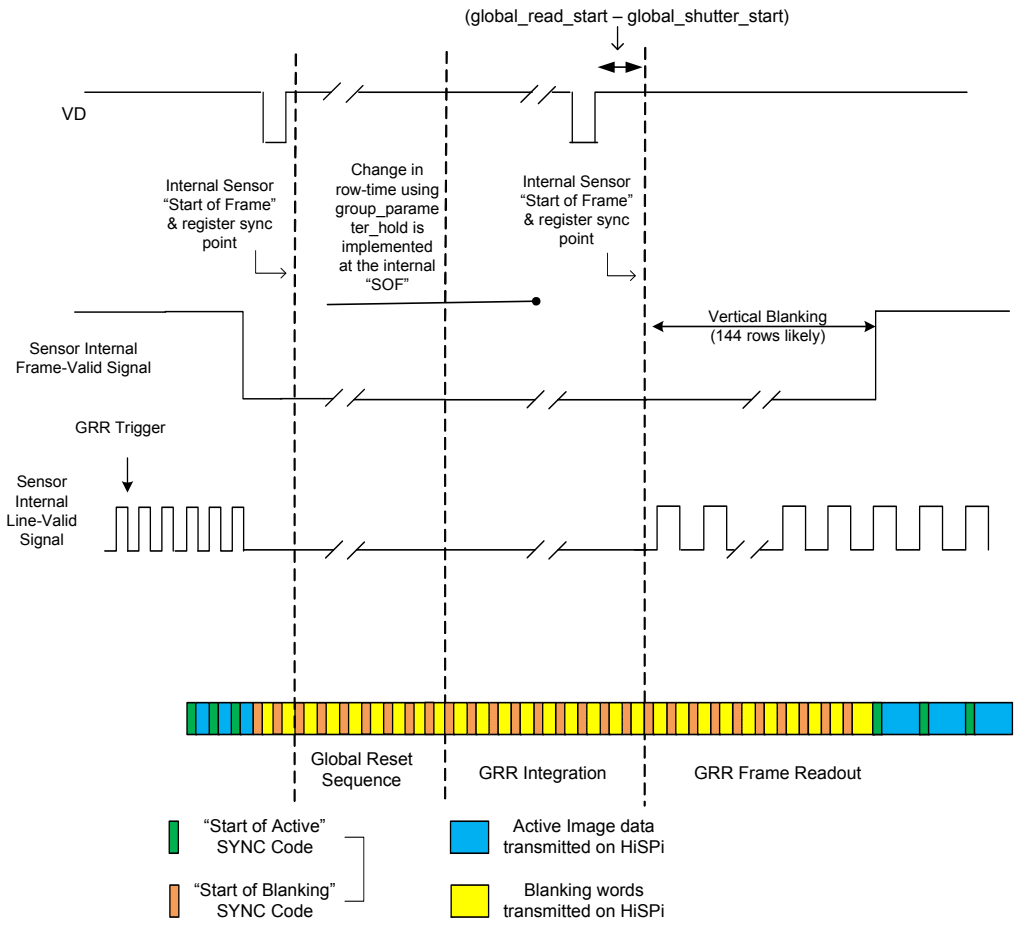


Figure 55: Slave Mode HiSPi Output (ERS to GRR Transition)



When GRR is triggered (by the rising edge of VD signal), the MT9F002 sensor starts GRR sequence and also send a start-of-blanking (SOB) SYNC code at the end of current ERS frame. It continues to send SOB sync codes during the entire GRR sequence.

Sensor Core Digital Data Path

Test Patterns

The MT9F002 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 20.

Table 20: 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 21: 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 MT9F002 registers must be set appropriately to control the frame rate and output timing. This includes:

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

Effect of Data Path Processing on Test Patterns

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

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

These effects can be eliminated by the following register settings:

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

Solid Color Test Pattern

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

100% Color Bars Test Pattern

In this test pattern, shown in Figure 41 on page 127, all pixel data is replaced by a Bayer version of an 8-color, color-bar chart (white, yellow, cyan, green, magenta, red, blue, black). Each bar is 1/8 of the width of the pixel array. The pattern repeats after eight bars.

Each color component of each bar is set to either 0 (fully off) or 0x3FF (fully on for 10-bit data). The pattern occupies the full height of the output image.

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

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

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

Figure 56: 100% Color Bars Test Pattern

Fade-to-gray Color Bars Test Pattern

In this test pattern, shown in Figure 42 on page 128, all pixel data is replaced by a Bayer version of an 8-color, color-bar chart (white, yellow, cyan, green, magenta, red, blue, black). Each bar is 1/8 of the width of the pixel array (2592/8 = 324 pixels). The test pattern repeats after 2592 pixels. Each color bar fades vertically from zero or full intensity at the top of the image to 50 percent intensity (mid-gray) on the last (968th) row of the pattern.

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

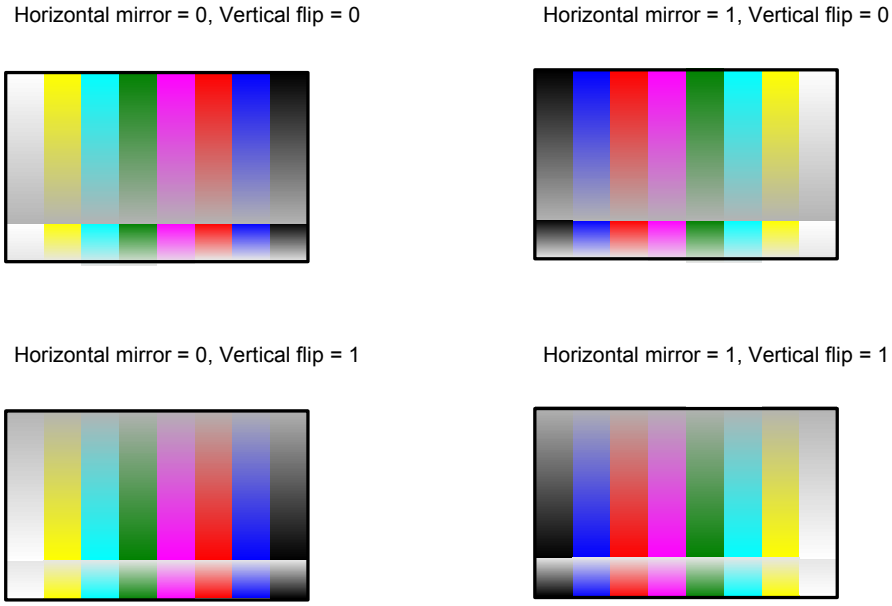
The rate-of-fade of the Bayer pattern is set so that there is at least one full pattern within a full-sized image for the sensor. Factors that affect this are the resolution of the ADC (10-bit or 12-bit) and the image height. For example, the MT9P013 fades the pixels by 2 LSB for each two rows. With 12-bit data, the pattern is 2048 pixels high and repeats after that, if the window is higher.

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

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

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

Figure 57: Fade-to-Gray Color Bar Test Pattern



PN9 Link Integrity Pattern

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

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

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

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

This polynomial generates this sequence of 10-bit values: 0x1FF, 0x378, 0x1A1, 0x336, 0x385... On the parallel pixel data output, these values are presented 10-bits per PIXCLK.

On the serial pixel data output, these values are streamed out sequentially without performing the RAW10 packing to bytes that normally occurs on this interface.

Walking 1s

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

Figure 58: Walking 1s 12-Bit Pattern

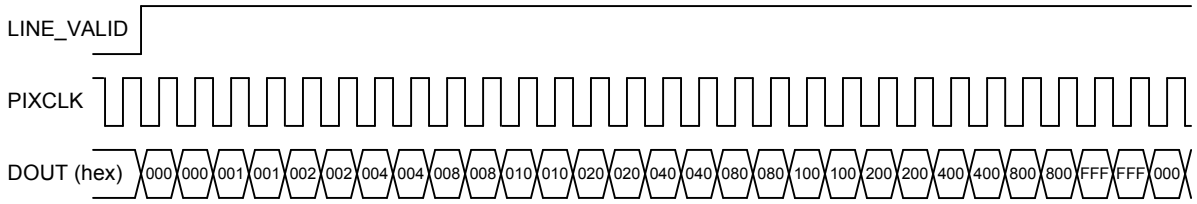


Figure 59: Walking 1s 10-Bit Pattern

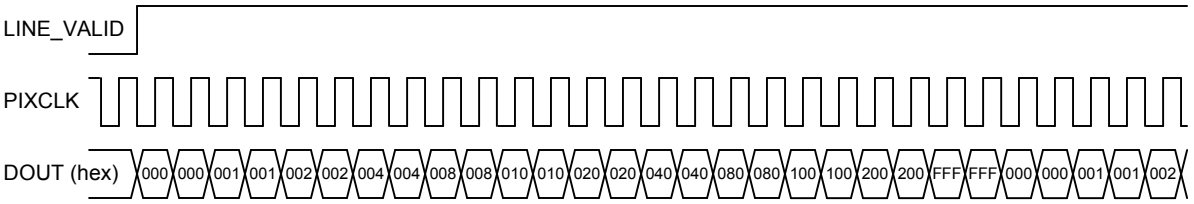
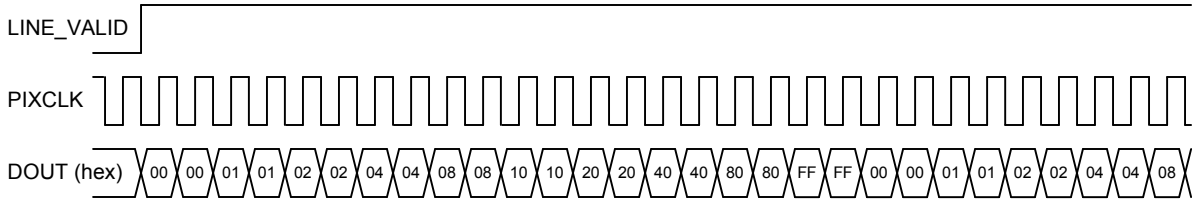


Figure 60: Walking 1s 8-Bit Pattern



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

Test Cursors

The MT9F002 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

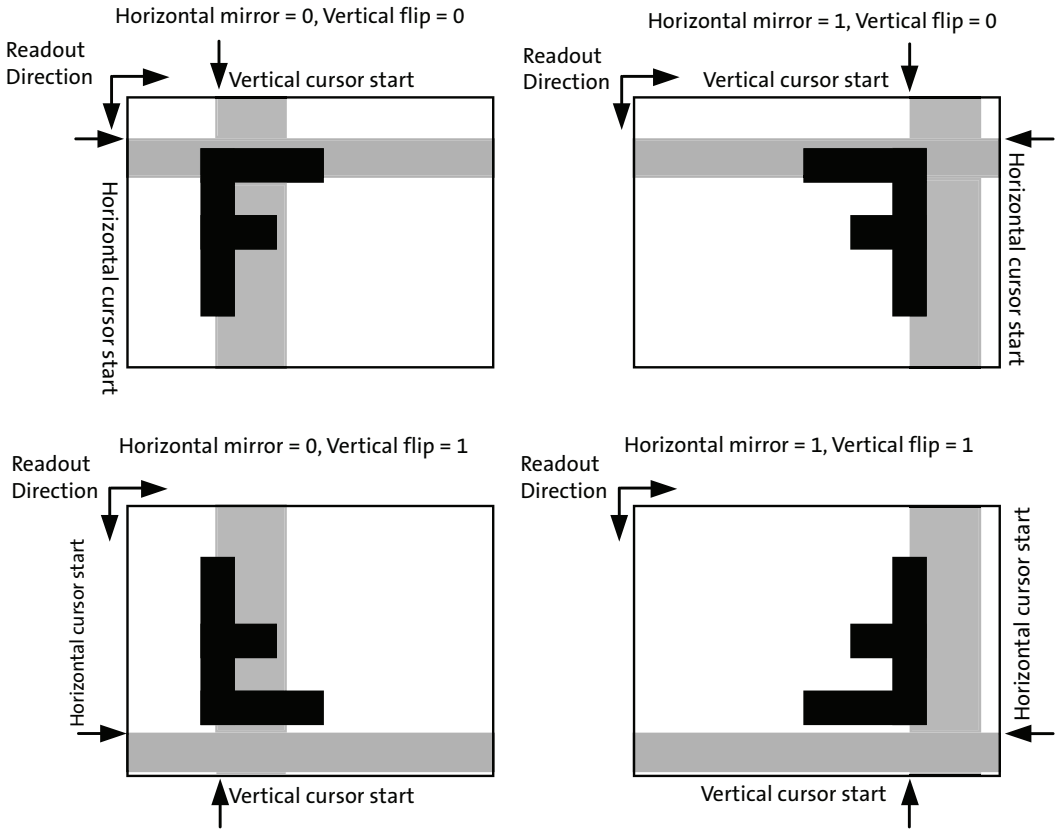
being drawn starting on the first row of the image. The cursors are opaque (they replace data from the imaged scene or test pattern). The color of each cursor is set by the values of the Bayer components in the test_data_red, test_data_greenR, test_data_blue and test_data_greenB registers. As a consequence, 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 MT9F002 is shown in Figure 61 on page 67 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 61: Test Cursor Behavior With Image Orientation



Timing Specifications

Power-Up Sequence

The recommended power-up sequence for the MT9F002 is shown in Figure 62. The available power supplies—VDD_IO, VDD, VDD_PLL, VAA, VAA_PIX, VDD_HISPI, VDD_TX can be turned on at the same time or have the separation specified below.

1. Turn on VDD_IO power supply.
2. After 1–500ms, turn on VDD and VDD_ HiSPi power supplies.
3. After 1–500ms, turn on VDD_PLL and VAA/VAA_PIX power supplies.
4. After 1–500ms, turn on VDD_TX power supply
5. After the last power supply is stable, enable EXTCLK.
6. Assert RESET_BAR for at least 1ms.
7. Wait 2700 EXTCLKs for internal initialization into software standby.
8. Configure PLL, output, and image settings to desired values
9. Set mode_select = 1 (R0x0100).
10. Wait 1ms for the PLL to lock before streaming state is reached.

Figure 62: Power-Up Sequence

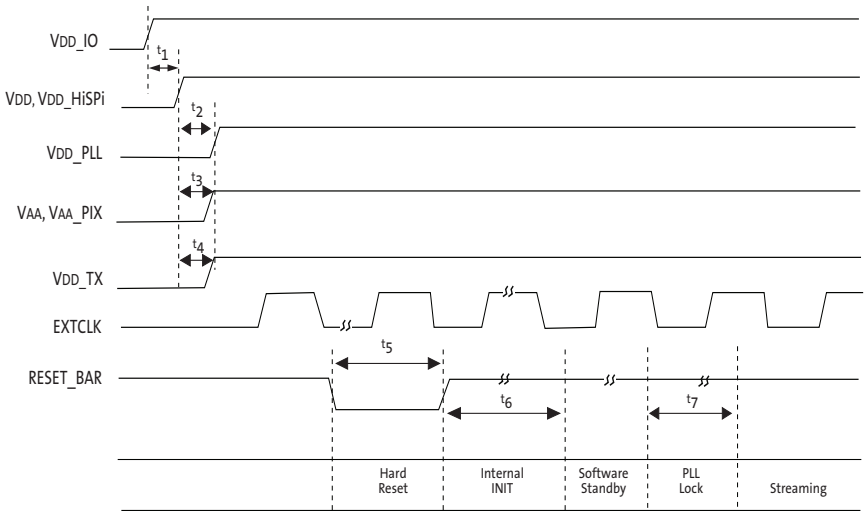


Table 22: Power-Up Sequence

Definition	Symbol	Min	Typ	Max	Unit
VDD_IO to VDD, VDD_HiSPi time	t_1	0	–	500	ms
VDD, VDD_HiSPi to VDD_PLL time	t_2	0	–	500	ms
VDD_PLL to VAA/VAA_PIX time	t_3	0	–	500	ms
VAA, VAA_PIX to VDD_TX	t_4	–	–	500	ms
Active hard reset	t_5	1	–	–	ms
Internal initialization	t_6	2700	–	–	EXTCLKs
PLL lock time	t_7	1	–	–	ms

Note: Digital supplies must be turned on before analog supplies.

Power-Down Sequence

The recommended power-down sequence for the MT9F002 is shown in Figure 63. The available power supplies—VDD_IO, VDD, VDD_PLL, VAA, VAA_PIX, VDD_HiSPi, and VDD_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 VDD_TX, VAA/VAA_PIX, and VDD_PLL power supplies.
5. After 1–500ms, turn off VDD and VDD_HiSPi power supply.
6. After 1–500ms, turn off VDD_IO power supply.

Figure 63: Power-Down Sequence

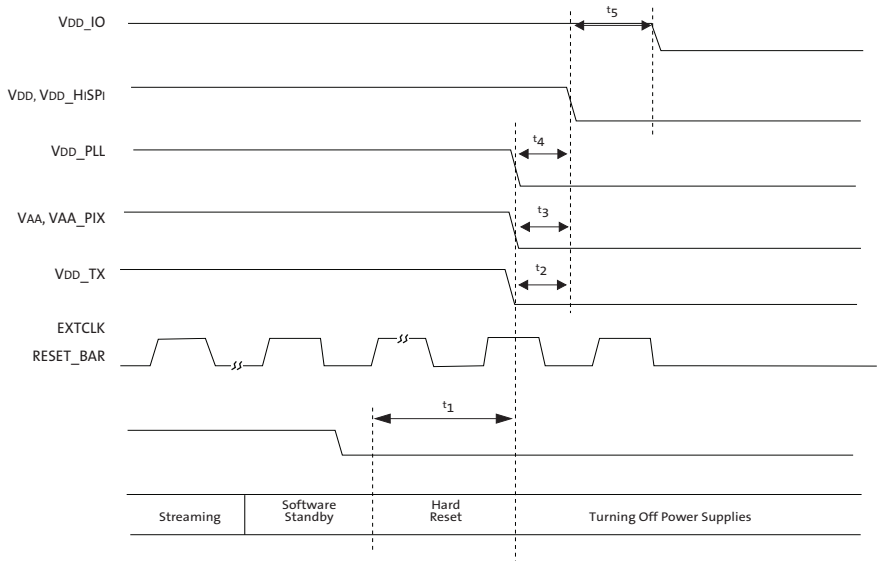


Table 23: Power-Down Sequence

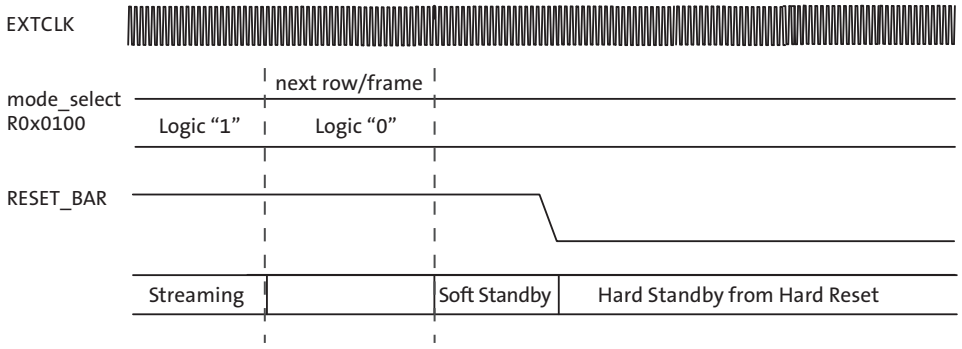
Definition	Symbol	Min	Typ	Max	Unit
Hard reset	t_1	1	–	–	ms
VDD_TX to VDD time	t_2	0	–	500	ms
VDD/VAA/VAA_PIX to VDD time	t_3	0	–	500	ms
VDD_PLL to VDD time	t_4	0	–	500	ms
VDD to VDD_IO time	t_5	0	–	500	ms

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 64 on page 70.

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 64: Hard Standby and Hard Reset



Soft Standby and Soft Reset

The MT9F002 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 65.

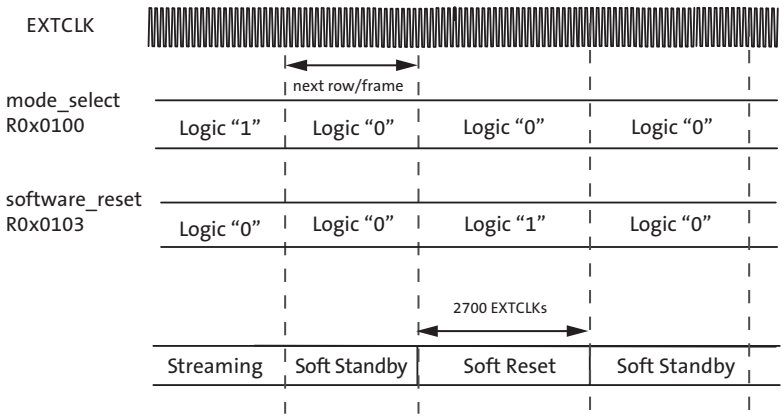
Soft Standby

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 2700 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 65: Soft Standby and Soft Reset



Spectral Characteristics

Figure 66: Quantum Efficiency

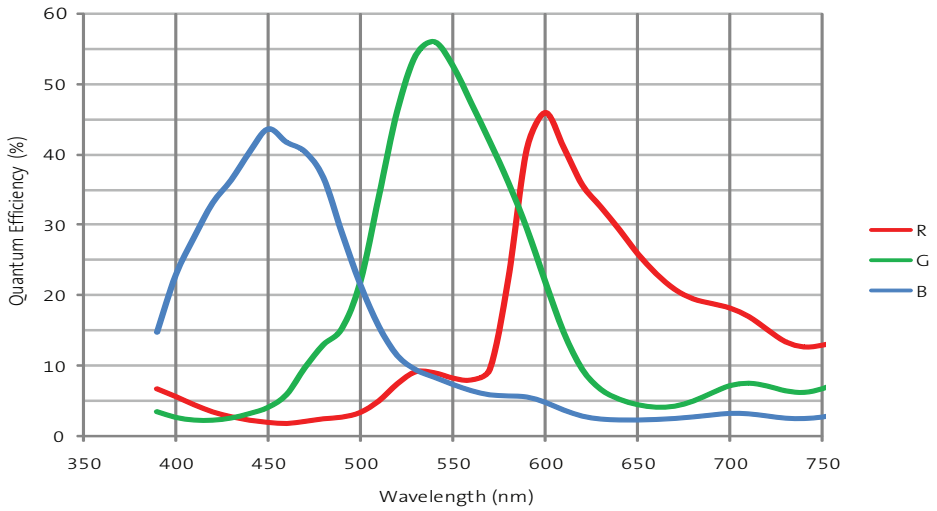


Table 24: 11.4° Chief Ray Angle

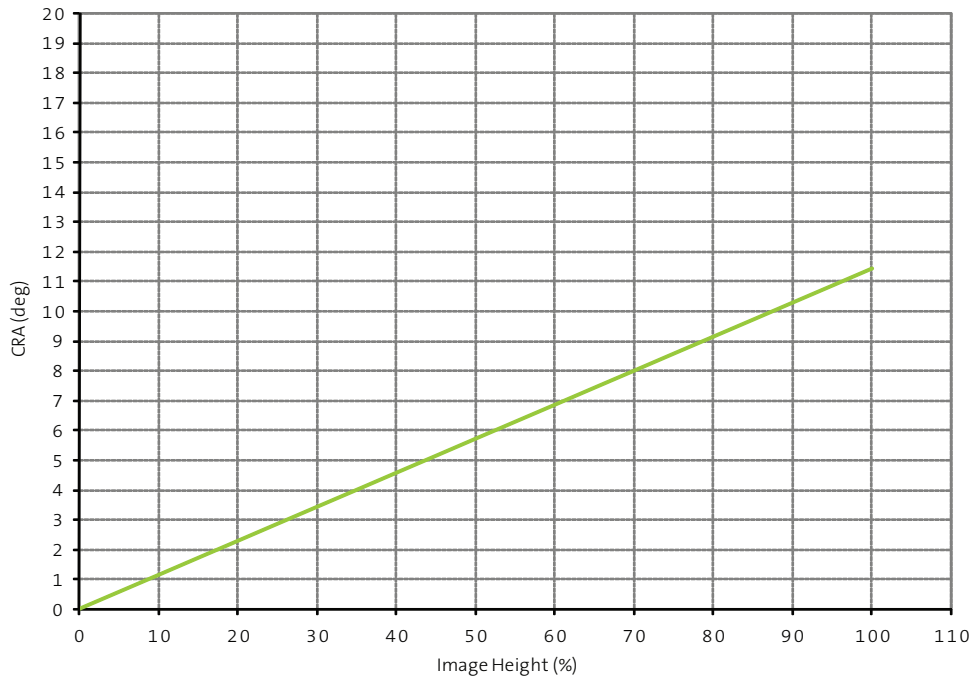


Image Height		CRA
(%)	(mm)	(deg)
0	0	0
5	0.192	0.57
10	0.384	1.14
15	0.575	1.71
20	0.767	2.28
25	0.959	2.85
30	1.151	3.42
35	1.343	3.99
40	1.534	4.56
45	1.726	5.13
50	1.918	5.70
55	2.110	6.27
60	2.302	6.84
65	2.493	7.41
70	2.685	7.98
75	2.877	8.55
80	3.069	9.14
85	3.261	9.69
90	3.452	10.26
95	3.644	10.83
100	3.836	11.40

Table 25: 25° Chief Ray Angle

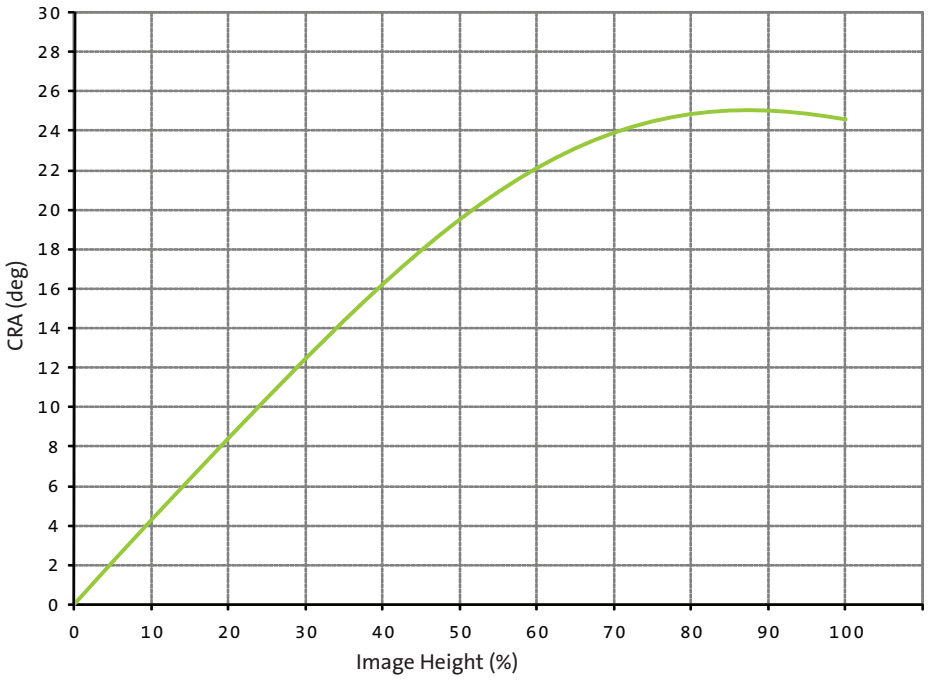


Image Height		CRA
(%)	(mm)	(deg)
0	0	0
5	0.192	2.16
10	0.384	4.27
15	0.575	6.35
20	0.767	8.41
25	0.959	10.45
30	1.151	12.44
35	1.343	14.37
40	1.534	16.21
45	1.726	17.93
50	1.918	19.49
55	2.110	20.89
60	2.302	22.10
65	2.493	23.10
70	2.685	23.88
75	2.877	24.46
80	3.069	24.83
85	3.261	25.00
90	3.452	25.00
95	3.644	24.84
100	3.836	24.56

Reading the Sensor CRA

Follow the steps below to obtain the CRA value of the image sensor:

1. Set the register bit field R0x301A[5] = 1.
2. Read the register bit fields R0x31FA[11:9].
3. Determine the CRA value according to Table 26.

Table 26: CRA Value

Binary Value of R0x31FA[11:9]	CRA Value
000	0
001	25
010	11.4

Electrical Characteristics

Table 27: DC Electrical Definitions and Characteristics

[f]EXTCLK = 24 MHz; V_{DD} = 1.8V; V_{DD_IO} = 1.8V; V_{AA} = 2.8V; V_{AA_PIX} = 2.8V; V_{DD_PLL} = 2.8V; V_{DD_HiSPI} = 1.8V, V_{DD_TX} = 0.4V; Output load = 68.5pF; T_J = 60°C; Data Rate = 660 Mbps; DLL set to 0, 14Mp frame-rate at 13.65 fps

Definition	Condition	Symbol	Min	Typ	Max	Unit
Core digital voltage		V_{DD}	1.7	1.8	1.9	V
I/O digital voltage		V_{DD_IO}	1.7	1.8	1.9	V
Analog voltage		V_{AA}	2.7	2.8	3.1	V
Pixel supply voltage		V_{AA_PIX}	2.7	2.8	3.1	V
PLL supply voltage		V_{DD_PLL}	2.4	2.8	3.1	V
HiSPi digital voltage		V_{DD_HiSPi}	1.7	1.8	1.9	V
HiSPi I/O digital voltage SLVS HiVCM		V_{DD_TX}	 0.3 1.7	 0.4 1.8	 0.9 1.9	 V V
Digital operating current	Serial HiSPi SLVS @ 13.65fps			75.0		mA
I/O digital operating current	Serial HiSPi SLVS @ 13.65fps			1.2		mA
Analog operating current	Serial HiSPi SLVS @ 13.65fps			172		mA
Pixel supply current	Serial HiSPi SLVS @ 13.65fps			5.6		mA
PLL supply current	Serial HiSPi SLVS @ 13.65fps			12.3		mA
HiSPi digital operating current	Serial HiSPi SLVS @ 13.65fps			28.6		mA
HiSPi I/O digital operating current	Serial HiSPi SLVS @ 13.65fps			10.5		mA
Digital operating current	Parallel interface @ 6.3fps			65.0		mA
I/O digital operating current	Parallel interface @ 6.3fps			41.5		mA
Analog operating current	Parallel interface @ 6.3fps			101.0		mA
Pixel supply current	Parallel interface @ 6.3fps			2.5		mA
PLL supply current	Parallel interface @ 6.3fps			13.7		mA
Soft standby (clock on)						mW

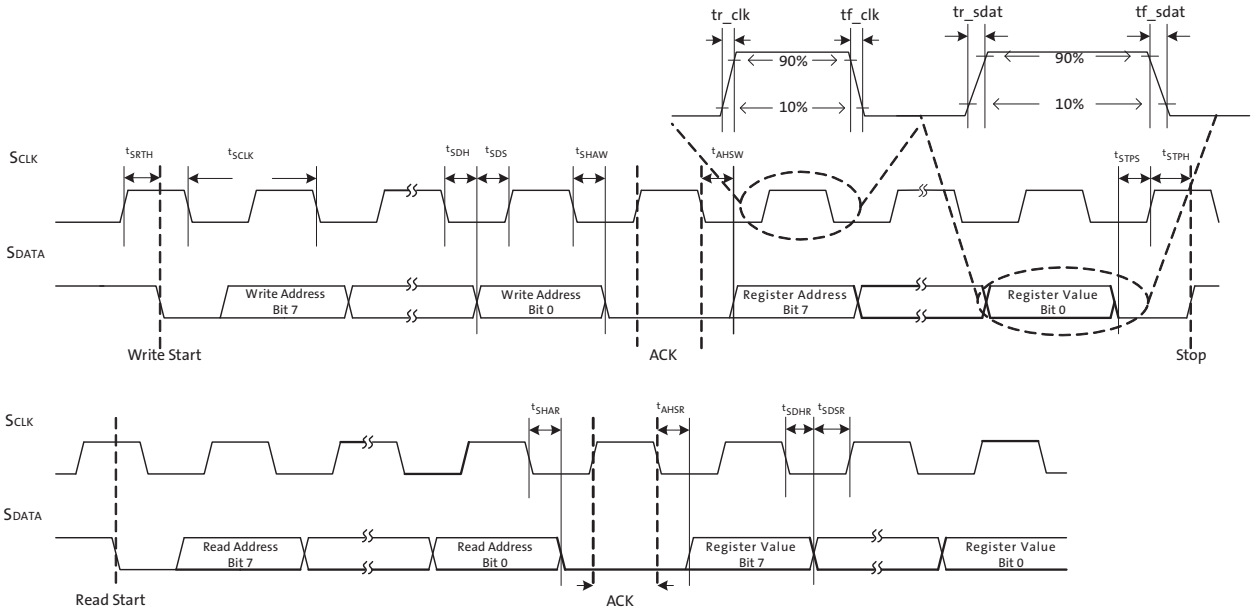
Caution **Stresses greater than those listed in Table 28 may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.**

Table 28: Absolute Maximum Ratings

Symbol	Definition	Condition	Min	Max	Unit
V_{DD_MAX}	Core digital voltage		–0.3	1.9	V
$V_{DD\_IO\_MAX}$	I/O digital voltage		–0.3	3.1	V
V_{AA_MAX}	Analog voltage		–0.3	3.5	V
V_{AA_PIX}	Pixel supply voltage		–0.3	3.5	V
V_{DD_PLL}	PLL supply voltage		–0.3	3.5	V
$V_{DD\_HiSPi\_MAX}$	HiSPi digital voltage		–0.3	1.9	V
$V_{DD\_TX\_MAX}$	HiSPi I/O digital voltage		–0.3	1.9	V
[t]ST	Storage temperature		–40	125	°C

Notes: 1. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

Figure 67: Two-Wire Serial Bus Timing Parameters



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

Table 29: Two-Wire Serial Register Interface Electrical Characteristics

^{f}EXTCLK = 24 MHz; V_{DD} = 1.8V; V_{DD_IO} = 1.8V; V_{AA} = 2.8V; V_{AA_PIX} = 2.8V; V_{DD_PLL} = 2.8V; V_{DD_HiSPi} = 1.8V, V_{DD_TX} = 0.4V; Output load = 68.5pF; T_J = 60°C; Data Rate =660 Mbps; 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 2mA	0.031	0.032	0.035	V
I_{OL}	Output LOW current	At specified V_{OL} 0.1V			3	mA
C_{IN}	Input pad capacitance				6	pF
C_{LOAD}	Load capacitance					pF

Table 30: Two-Wire Serial Register Interface Timing Specification

^{f}EXTCLK = 24 MHz; VDD = 1.8V; VDD_IO = 1.8V; VAA = 2.8V; VAA_PIX = 2.8V; VDD_PLL = 2.8V; VDD_HiSPi = 1.8V, VDD_TX = 0.4V; Output load = 68.5pF; T_J = 60°C; Data Rate = 660 Mbps,; DLL set to 0

Symbol	Parameter	Condition	Min	Typ	Max	Unit
^{f}SCLK	Serial interface input clock	–	0	100	400	kHz
	SCLK duty cycle	VOD	45	50	60	%
^{t}R	SCLK/SDATA rise time				300	µs
^{t}SRTS	Start setup time	Master WRITE to slave	0.6			µs
^{t}SRTH	Start hold time	Master WRITE to slave	0.4			µs
^{t}SDH	SDATA hold	Master WRITE to slave	0.3		0.65	µs
^{t}SDS	SDATA setup	Master WRITE to slave	0.3			µs
^{t}SHAW	SDATA hold to ACK	Master READ to slave	0.15		0.65	µs
^{t}AHSW	ACK hold to SDATA	Master WRITE to slave	0.15		0.70	µs
^{t}STPS	Stop setup time	Master WRITE to slave	0.3			µs
^{t}STPH	Stop hold time	Master WRITE to slave	0.6			µs
^{t}SHAR	SDATA hold to ACK	Master WRITE to slave	0.3		1.65	µs
^{t}AHSR	ACK hold to SDATA	Master WRITE to slave	0.3		0.65	µs
^{t}SDHR	SDATA hold	Master READ from slave	.012		0.70	µs
^{t}SDSR	SDATA setup	Master READ from slave	0.3			µs

Figure 68: I/O Timing Diagram

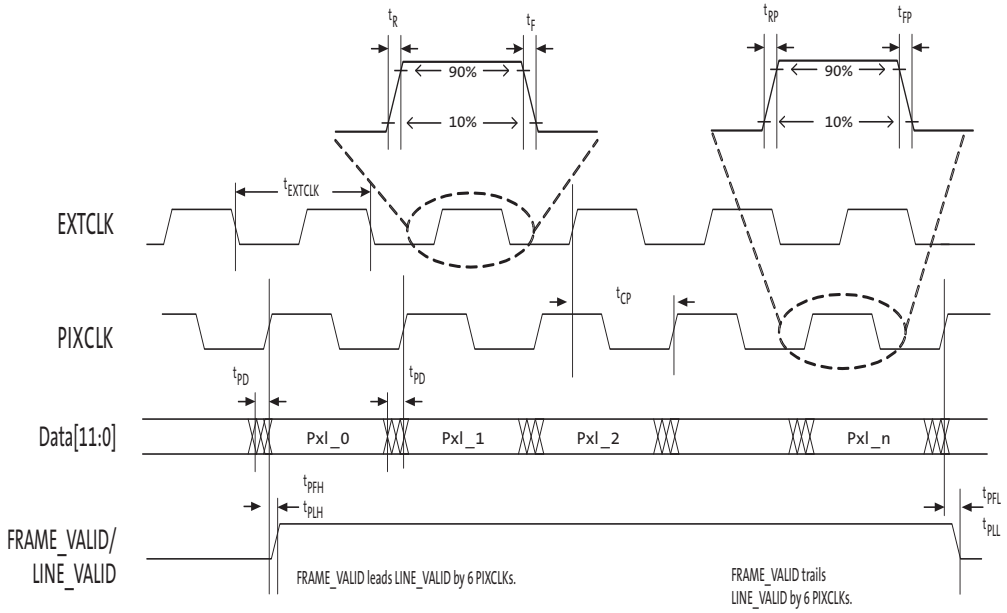


Table 31: I/O Parameters

^{f}EXTCLK = 24 MHz; V_{DD} = 1.8V; V_{AA} = 2.8V; V_{AA_PIX} = 2.8V; V_{DD_PLL} = 2.8V; V_{DD_HiSPi} = 1.8V, V_{DD_TX} = 0.4V; Output load = 68.5pF; T_J = 60°C; Data Rate = 660 Mbps,; DLL set to 0

Symbol	Definition	Conditions	Min	Max	Units
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 32: I/O Timing

^{f}EXTCLK = 24 MHz; V_{DD} = 1.8V; V_{DD_IO} = 1.8V; V_{AA} = 2.8V; V_{AA_PIX} = 2.8V; V_{DD_PLL} = 2.8V; V_{DD_HiSPi} = 1.8V, V_{DD_TX} = 0.4V; Output load = 68.5pF; T_J = 60°C; Data Rate = 660 Mbps,; DLL set to 0

Symbol	Definition	Conditions	Min	Typ	Max	Units
^{f}EXTCLK	Input clock frequency	PLL enabled	2	24	64	MHz
^{t}EXTCLK	Input clock period	PLL enabled	200	41.7	15.6	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} = 15pF	–	0.7	–	V/ns
^{f}PIXCLK	PIXCLK frequency	Default	–	–	96	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

SLVS Electrical Specifications

Table 33: Power Supply and Operating Temperature

Parameter	Symbol	Min	Typ	Max	Unit	Notes
SLVS Current Consumption	I_{DD_TX}			n*18	mA	1, 2
HiSPi PHY Current Consumption	I_{DD_HiSPi}			n*45	mA	1, 2, 3
Operating temperature	T_J	-30		70	°C	4

Notes:
1. Where 'n' is the number of PHYs
2. Temperature of 25°C

3. Up to 700 Mbps
4. Specification values may be exceeded when outside this temperature range.

Table 34: SLVS Electrical DC Specification
Tj = 25°C

Parameter	Symbol	Min	Typ	Max	Unit
SLVS DC mean common mode voltage	V_{CM}	0.45*VDD_TX	0.5*VDD_TX	0.55*VDD_TX	V
SLVS DC mean differential output voltage	$\|V_{OD}\|$	0.36*VDD_TX	0.5*VDD_TX	0.64*VDD_TX	V
Change in V_{CM} between logic 1 and 0	ΔV_{CM}			25	mV
Change in $\|V_{OD}\|$ between logic 1 and 0	$\|V_{OD}\|$			25	mV
V_{OD} noise margin	NM			±30	%
Difference in V_{CM} between any two channels	$\|\Delta V_{CM}\|$			50	mV
Difference in V_{OD} between any two channels	$\|\Delta V_{OD}\|$			100	mV
Common-mode AC Voltage (pk) without VCM cap termination	V_{CM_AC}			50	mV
Common-mode AC Voltage (pk) with VCM cap termination	V_{CM_AC}			30	mV
Maximum overshoot peak $\|V_{OD}\|$	V_{OD_AC}			$1.3*\|V_{OD}\|$	V
Maximum overshoot $V_{diff\ pk\text{-}pk}$	V_{diff_pkpk}			$2.6*V_{OD}$	V
Single-ended Output impedance	R_O	35	50	70	Ω
Output Impedance Mismatch	ΔR_O			20	%

Table 35: SLVS Electrical Timing Specification

Parameter	Symbol	Min	Max	Unit	Notes
Data Rate	1/UI	280	700	Mbps	1
Bitrate Period	t_{PW}	1.43	3.57	ns	1
Max setup time from transmitter	t_{PRE}	0.3		UI	1, 2
Max hold time from transmitter	t_{POST}	0.3		UI	1, 2
Eye Width	t_{EYE}		0.6	UI	1, 2
Data Total Jitter (pk-pk) @1e-9	$t_{TOTALJIT}$		0.2	UI	1, 2
Clock Period Jitter (RMS)	t_{CKJIT}		50	ps	2
Clock Cycle-to-Cycle Jitter (RMS)	t_{CYCJIT}		100	ps	2
Rise time (20% - 80%)	tR	150ps	0.25	UI	3
Fall time (20% - 80%)	tF	150ps	0.25	UI	3
Clock duty cycle	DCYC	45	55	%	2
Mean Clock to Data Skew	t_{CHSKEW}	-0.1	0.1	UI	1, 4
PHY-to-PHY Skew	$t_{PHYSKEW}$		2.1	UI	1, 5
Mean differential skew	$t_{DIFFSKEW}$	-100	100	ps	6

Notes:
1. One UI is defined as the normalized mean time between one edge and the following edge of the clock.
2. Taken from the 0V crossing point with the DLL off.
3. Also defined with a maximum loading capacitance of 10 pF on any pin. The loading capacitance may also need to be less for higher bitrates so the rise and fall times do not exceed the maximum 0.3 UI.
4. The absolute mean skew between the Clock lane and any Data Lane in the same PHY between any edges.
5. The absolute skew between any Clock in one PHY and any Data lane in any other PHY between any edges.

Differential skew is defined as the skew between complementary outputs. It is measured as the absolute time between the two complementary edges at mean VCM point. Note that differential skew also is related to the ΔVCM_AC spec which also must not be exceeded.

HiVCM Electrical Specifications

The HiSPi 2.0 specification also defines an alternative signaling level mode called HiVCM. Both V_{OD} and V_{CM} are still scalable with VDD_TX, but with VDD_TX nominal set to 1.8V the common-mode is elevated to around 0.9V.

Table 36: HiVCM Power Supply and Operating Temperatures

Parameter	Symbol	Min	Typ	Max	Unit	Notes
HiVCM Current Consumption	IDD_TX			n*34	mA	1, 2
HiSPi PHY Current Consumption	IDD_HiSPi			n*45	mA	1, 2, 3
Operating temperature	T_J	-30		70	°C	4

Notes:
1. Where 'n' is the number of PHYs
2. Temperature of 25°C
3. Up to 700 Mbps
4. Specification values may be exceeded when outside this temperature range.

Table 37: HiVCM Electrical Voltage and Impedance Specification
Tj = 25° C

Parameter	Symbol	Min	Typ	Max	Unit
HiVCM DC mean common mode voltage	V_{CM}	0.76	0.90	1.07	V
HiVCM DC mean differential output voltage	$\|V_{OD}\|$	200	280	350	mV
Change in V_{CM} between logic 1 and 0	ΔV_{CM}			25	mV
Change in $\|V_{OD}\|$ between logic 1 and 0	$\|\ V_{OD}\|$			25	mV
V_{OD} noise margin	NM			±30	%
Difference in V_{CM} between any two channels	$\|\Delta V_{CM}\|$			50	mV
Difference in V_{OD} between any two channels	$\|\Delta V_{OD}\|$			100	mV
Common-mode AC Voltage (pk) without V_{CM} cap termination	ΔV_{CM_AC}			50	mV
Common-mode AC Voltage (pk) with V_{CM} cap termination	ΔV_{CM_AC}			30	mV
Maximum overshoot peak $\|V_{OD}\|$	V_{OD_AC}			$1.3*\|V_{OD}\|$	V
Maximum overshoot Vdiff pk-pk	V_{diff_pkpk}			$2.6*V_{OD}$	V
Single-ended Output impedance	R_O	40	70	100	Ω
Output Impedance Mismatch	ΔR_O			20	%

Table 38: HiVCM Electrical AC Specification

Parameter	Symbol	Min	Max	Unit	Notes
Data Rate	1/UI	280	700	Mbps	1
Bitrate Period	t_{PW}	1.43	3.57	ns	1
Max setup time from transmitter	t_{PRE}	0.3		UI	1, 2
Max hold time from transmitter	t_{POST}	0.3		UI	1, 2
Eye Width	t_{EYE}		0.6	UI	1, 2
Data Total Jitter (pk-pk) @1e-9	$t_{TOTALJIT}$		0.2	UI	1, 2
Clock Period Jitter (RMS)	t_{CKJIT}		50	ps	2
Clock Cycle-to-Cycle Jitter (RMS)	t_{CYCJIT}		100	ps	2
Rise time (20% - 80%)	t_R	150ps	0.3	UI	3
Fall time (20% - 80%)	t_F	150ps	0.3	UI	3
Clock duty cycle	D_{CYC}	45	55	%	2
Clock to Data Skew	t_{CHSKEW}	-0.1	0.1	UI	1, 4
PHY-to-PHY Skew	tP_{HYSKEW}		2.1	UI	1, 5
Mean differential skew	$t_{DIFFSKEW}$	-100	100	ps	6

Notes:
1. One UI is defined as the normalized mean time between one edge and the following edge of the clock.
2. Taken from the 0 V crossing point with the DLL off.
3. Also defined with a maximum loading capacitance of 10pF on any pin. The loading capacitance may also need to be less for higher bitrates so the rise and fall times do not exceed the maximum 0.3 UI.
4. The absolute mean skew between the Clock lane and any Data Lane in the same PHY between any edges.
5. The absolute mean skew between any Clock in one PHY and any Data lane in any other PHY between any edges.
6. Differential skew is defined as the skew between complementary outputs. It is measured as the absolute time between the two complementary edges at mean V_{CM} point. Note that differential skew also is related to the ΔVCM_AC spec which also must not be exceeded.

Electrical Definitions

Figure 69 is the diagram defining differential amplitude V_{OD}, $V_{CM,}$ and rise and fall times. To measure V_{OD} and V_{CM} use the DC test circuit shown in Figure 70 on page 83 and set the HiSPi PHY to constant Logic 1 and Logic 0. Measure V_{oa}, V_{ob} and V_{CM} with voltmeters for both Logic 1 and Logic 0.

Figure 69: Single-Ended and Differential Signals

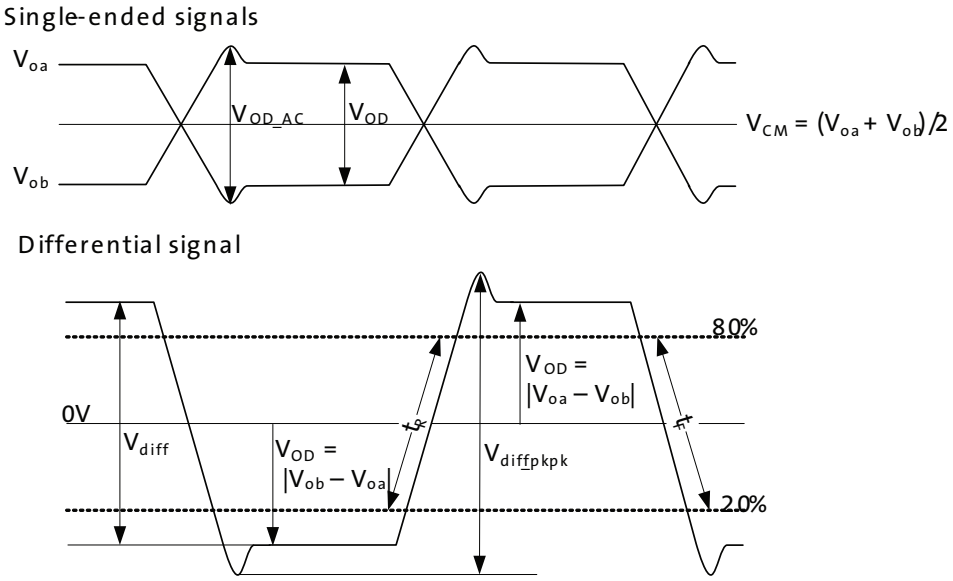
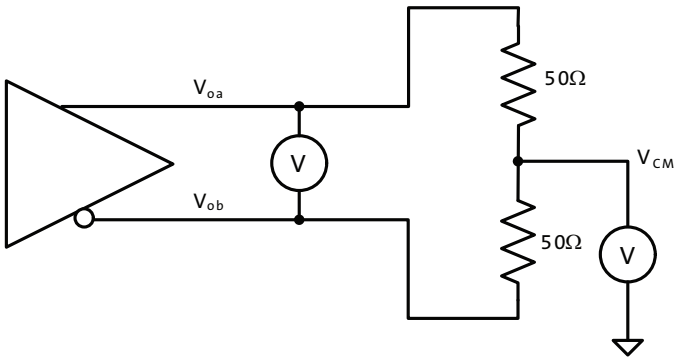


Figure 70: DC Test Circuit



$V_{OD}(m) = |V_{oa}(m) - V_{ob}(m)|$ where 'm' is either "1" for logic 1 or "0" for logic 0 (EQ 22)

$$V_{OD} = \frac{V_{OD}(1) + V_{OD}(0)}{2} \quad \textit{(EQ 23)}$$

$$V_{diff} = V_{OD}(1) + V_{OD}(0) \quad \text{(EQ 24)}$$

$$\Delta V_{OD} = |V_{OD}(1) - V_{OD}(0)| \quad \text{(EQ 25)}$$

$$V_{CM} = \frac{V_{CM}(1) + V_{CM}(0)}{2} \quad \textit{(EQ 26)}$$

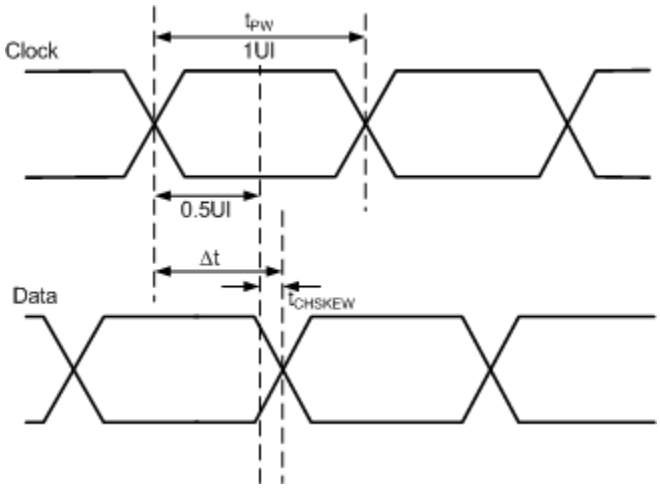
$$\Delta V_{CM} = |V_{CM}(1) - V_{CM}(0)| \quad \text{(EQ 27)}$$

Both V_{OD} and V_{CM} are measured for all output channels. The worst case ΔV_{OD} is defined as the largest difference in V_{OD} between all channels regardless of logic level. And the worst case ΔV_{CM} is similarly defined as the largest difference in V_{CM} between all channels regardless of logic level.

Timing Definitions

1. Timing measurements are to be taken using the Square Wave test mode.
2. Rise and fall times are measured between 20% to 80% positions on the differential waveform, as shown in Figure 69: “Single-Ended and Differential Signals,” on page 83.
3. Mean Clock-to-Data skew should be measured from the 0V crossing point on Clock to the 0V crossing point on any Data channel regardless of edge, as shown in Figure 71 on page 84. This time is compared with the ideal Data transition point of 0.5UI with the difference being the Clock-to-Data Skew (see Equation 28 on page 84).

Figure 71: Clock-to-Data Skew Timing Diagram



$$t_{CHSKEW}(ps) = \Delta t - \frac{t_{pw}}{2} \quad \text{(EQ 28)}$$

$$t_{CHSKEW}(UI) = \frac{\Delta t}{t_{pw}} - 0.5 \quad \text{(EQ 29)}$$

4. The differential skew is measured on the two single-ended signals for any channel. The time is taken from a transition on V_{oa} signal to corresponding transition on V_{ob} signal at V_{CM} crossing point.

Figure 72: Differential Skew

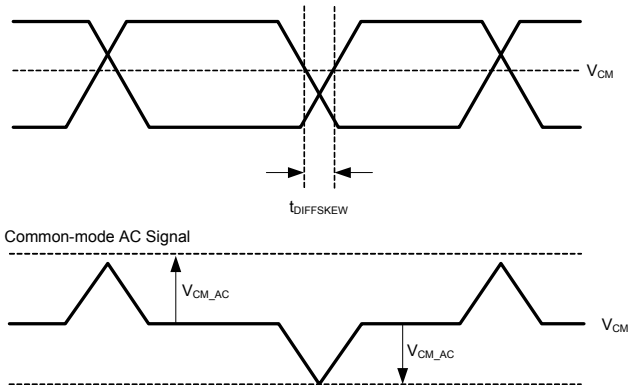


Figure 72 on page 85 also shows the corresponding AC V_{CM} common-mode signal. Differential skew between the V_{oa} and V_{ob} signals can cause spikes in the common-mode, which the receiver needs to be able to reject. V_{CM_AC} is measured as the absolute peak deviation from the mean DC V_{CM} common-mode.

Transmitter Eye Mask

Figure 73: Transmitter Eye Mask

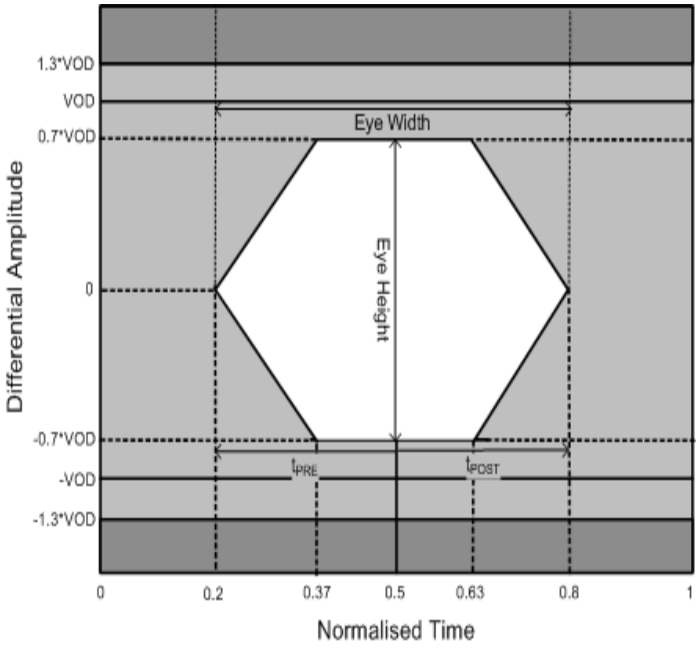
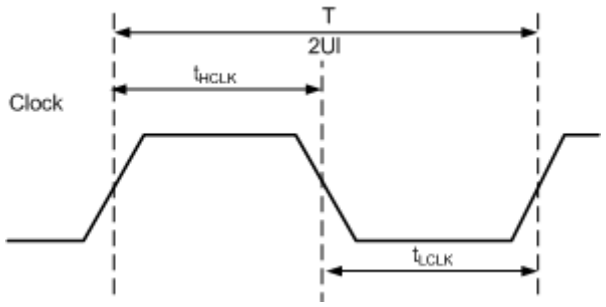


Figure 73 defines the **eye mask** for the transmitter. 0.5 UI point is the instantaneous crossing point of the Clock. The area in white shows the area Data is prohibited from crossing into. The **eye mask** also defines the minimum eye height, the data $\mathbf{t_{pre}}$ and $\mathbf{t_{post}}$ times, and the **total jitter pk-pk +mean skew** (t_{TJSKEW}) for Data.

Clock Signal

t_{HCLK} is defined as the high clock period, and t_{LCLK} is defined as the low clock period as shown in Figure 74. The clock duty cycle D_{CYC} is defined as the percentage time the clock is either high (t_{HCLK}) or low (t_{LCLK}) compared with the clock period T.

Figure 74: Clock Duty Cycle



$$D_{CYC}(1) = \frac{t_{HCLK}}{T} \quad \text{(EQ 30)}$$

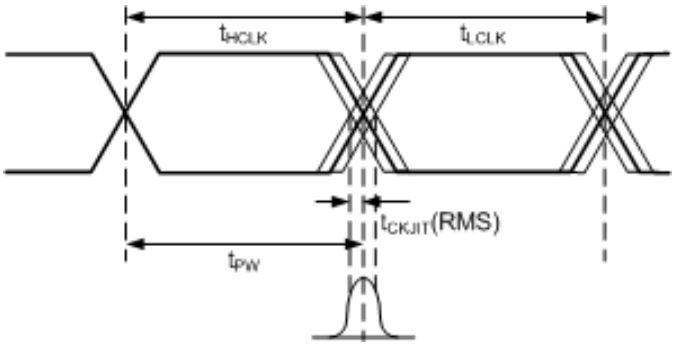
$$D_{CYC}(0) = \frac{t_{LCLK}}{T} \quad \text{(EQ 31)}$$

$$t_{pw} = \frac{T}{2} \quad \text{(i.e, 1 UI)} \quad \text{(EQ 32)}$$

$$Bitrate = \frac{1}{t_{pw}} \quad \text{(EQ 33)}$$

Figure 75 shows the definition of clock jitter for both the period and the cycle-to-cycle jitter.

Figure 75: Clock Jitter



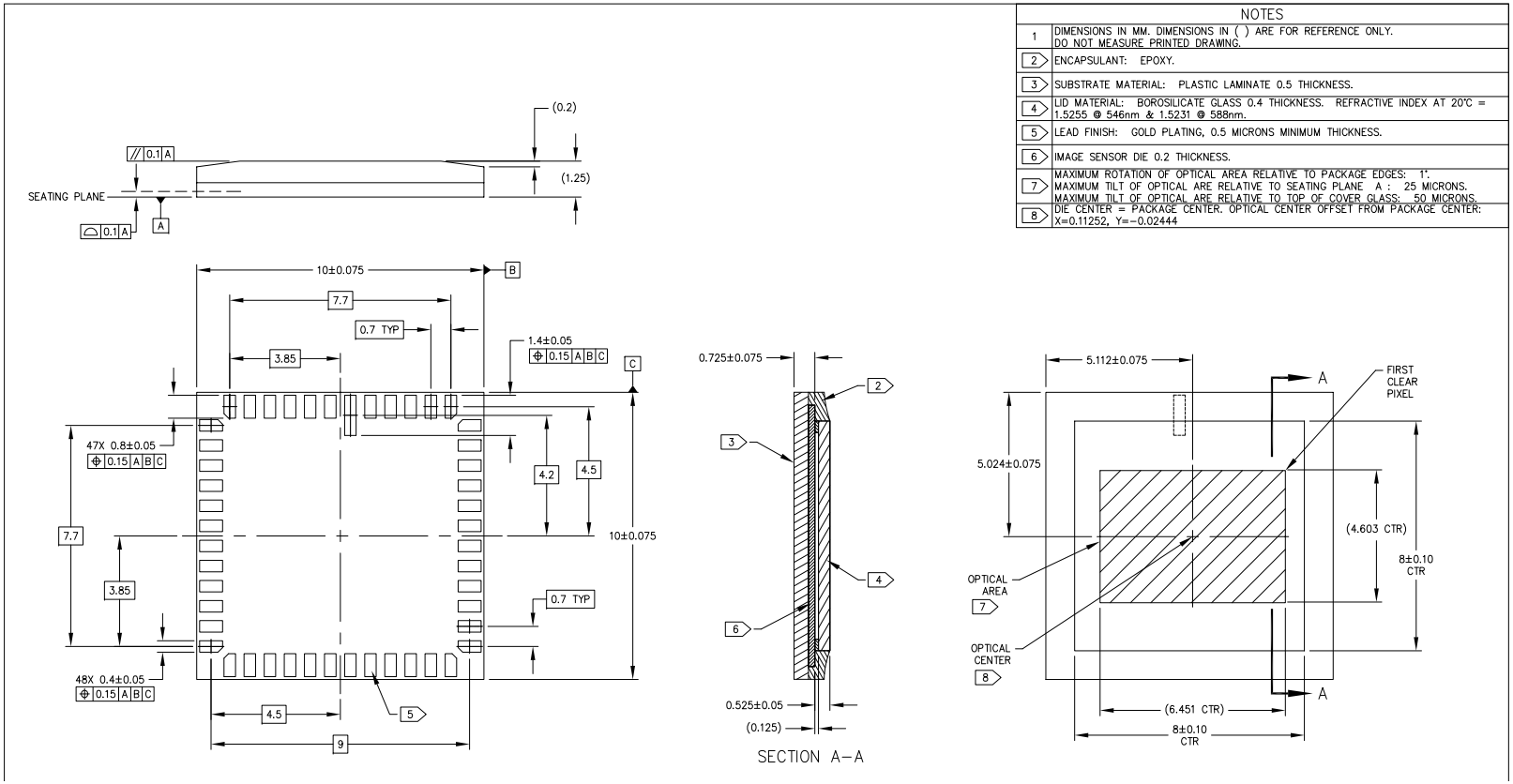
Period Jitter (t_{CKJIT}) is defined as the deviation of the instantaneous clock t_{PW} from an ideal 1UI. This should be measured for both the clock high period variation Δt_{HCLK}, and the clock low period variation Δt_{LCLK} taking the RMS or 1-sigma standard deviation and quoting the worse case jitter between Δt_{HCLK} and Δt_{LCLK}.

Cycle-to-cycle jitter (t_{CYCJIT}) is defined as the difference in time between consecutive clock high and clock low periods t_{HCLK} and t_{LCLK}, quoting the RMS value of the variation $\Delta(t_{HCLK} - t_{LCLK})$.

If pk-pk jitter is also measured, this should be limited to ±3-sigma.

Package Dimensions

Figure 76: 48-Pin iLCC Package Outline Drawing



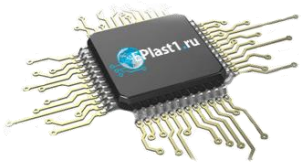
Компания «ЭлектроПласт» предлагает заключение долгосрочных отношений при поставках импортных электронных компонентов на взаимовыгодных условиях!

Наши преимущества:

- Оперативные поставки широкого спектра электронных компонентов отечественного и импортного производства напрямую от производителей и с крупнейших мировых складов;
- Поставка более 17-ти миллионов наименований электронных компонентов;
- Поставка сложных, дефицитных, либо снятых с производства позиций;
- Оперативные сроки поставки под заказ (от 5 рабочих дней);
- Экспресс доставка в любую точку России;
- Техническая поддержка проекта, помощь в подборе аналогов, поставка прототипов;
- Система менеджмента качества сертифицирована по Международному стандарту ISO 9001;
- Лицензия ФСБ на осуществление работ с использованием сведений, составляющих государственную тайну;
- Поставка специализированных компонентов (Xilinx, Altera, Analog Devices, Intersil, Interpoint, Microsemi, Aeroflex, Peregrine, Syfer, Eurofarad, Texas Instrument, Miteq, Cobham, E2V, MA-COM, Hittite, Mini-Circuits,General Dynamics и др.);

Помимо этого, одним из направлений компании «ЭлектроПласт» является направление «Источники питания». Мы предлагаем Вам помощь Конструкторского отдела:

- Подбор оптимального решения, техническое обоснование при выборе компонента;
- Подбор аналогов;
- Консультации по применению компонента;
- Поставка образцов и прототипов;
- Техническая поддержка проекта;
- Защита от снятия компонента с производства.



Как с нами связаться

Телефон: 8 (812) 309 58 32 (многоканальный)
Факс: 8 (812) 320-02-42
Электронная почта: org@eplast1.ru
Адрес: 198099, г. Санкт-Петербург, ул. Калинина, дом 2, корпус 4, литера А.