MT9D015

MT9D015 1/5-Inch 2 Mp CMOS Digital Image Sensor

Table 1. KEY PERFORMANCE PARAMETERS

Parameter		Value
Die Size		4356.15 µm (H) × 4354.85 µm (V)
Optical Format		1/5−inch UXGA (4:3)
Active Imager Size		2.828 mm (H) × 2.128 (V)
Active Pixels		1608 (H) × 1208 (V)
Pixel Size		1.75 × 1.75 µm
Color Filter Array		RGB Bayer Pattern
Shutter Type		Electronic Rolling Shutter (ERS)
Input Clock Frequency		6−27 MHz
Maximum Data Rate		640 Mb/s (CCP) and 768 Mb/s (MIPI)
CCP Frame Rate	UXGA (1600 x 1200)	Programmable Up to 21 fps in Profile 0 Mode (RAW10) Programmable Up to 30 fps in Profile 1/2 Mode (RAW10)
	XGA (1024 x 768)	Programmable Up to 42 fps in Profile 0 Mode (RAW10) Programmable Up to 61 fps in Profile 1/2 Mode (RAW10)
	HD (1280 x 720)	30 fps
MIPI Frame Rate	UXGA (1600 x 1200)	30 fps (RAW10)
	VGA (640 x 480)	60 fps (RAW10)
	QVGA (320X240)	120 fps (RAW10)
	HD (1280 x 720)	30 fps (RAW10)
ADC Resolution		10−bit
Responsivity		0.86 V/lux−sec
Dynamic Range		62 dB
SNR_{MAX}		38.7 dB
Supply Voltage	Analog	2.40−2.90 V (2.80 V Nominal)
	Digital	1.70−1.90 V (1.80 V Nominal)
Power Consumption		272 mW at 30 fps (TYP)
Operating Temperature		−30°C to +70°C
Packaging		Bare Die

ON Semiconductor®

www.onsemi.com

ORDERING INFORMATION

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

Features

- Superior Low Light Performance
- High Sensitivity
- Low Dark Current
- Simple Two−wire Serial Interface
- Auto Black Level Calibration
- Programmable Controls: Gain, Frame Size/Rate, Exposure, Left−right and Top−bottom Image Reversal, Window Size and Panning
- Data Interface: CCP2 Compliant Sub−low−voltage Differential Signaling (sub−LVDS) or Single Lane Serial Mobile Industry Processor Interface (MIPI)
- SMIA 1.0 Compatible; MIPI 1.0 Compliant
- On−chip Phase−locked Loop (PLL) Oscillator
- Bayer−pattern Down−size Scaler
- Integrated Lens Shading Correction
- Internal Power Switch for Ultra−low Standby Current Consumption
- 30 fps at Full Resolution
- 2D Defect Pixel Correction
- 2624−bit One−time Programmable Memory (OTPM) for Storing Module Information and Lens Shading Correction

Applications

- Cellular Phones
- Digital Still Cameras
- PC Cameras
- PDAs

TABLE OF CONTENTS

ORDERING INFORMATION

Table 2. AVAILABLE PART NUMBERS

Part Number	Product Description	Orderable Product Attribute Description
MT9D015D00STCMC25BC1−200	2 MP 1/4” CIS	Die Sales, 200 μm Thickness
MT9D015D00STCPC25BC1−400	2 MP 1/4” CIS	Die Sales, 400 μm Thickness

GENERAL DESCRIPTION

The ON Semiconductor MT9D015 is a 1/5−inch UXGA−format CMOS active−pixel digital image sensor with a pixel array of 1600 (H) × 1200 (V) (1608 (H) × 1208 (V) including border pixels). It incorporates sophisticated on−chip camera functions such as windowing, mirroring, column and subsampling modes. It is programmable through a simple two−wire serial interface and has very low power consumption.

The MT9D015 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 mode, the sensor generates a UXGA image at 21 frames per second (fps) when ext_clk_freq_mhz = 16 MHz. An on−chip analog−to−digital converter (ADC) generates a 10−bit value for each pixel.

FUNCTIONAL OVERVIEW

The MT9D015 is a progressive−scan sensor that generates a stream of pixel data at a constant frame rate. It uses an on−chip, phase−locked loop (PLL) to generate all internal clocks from a single master input clock running between 6 and 27 MHz. The maximum pixel rate is 64 Mp/s, corresponding to a video timing pixel clock rate of 91.4 MHz. A block diagram of the sensor is shown in Figure 1.

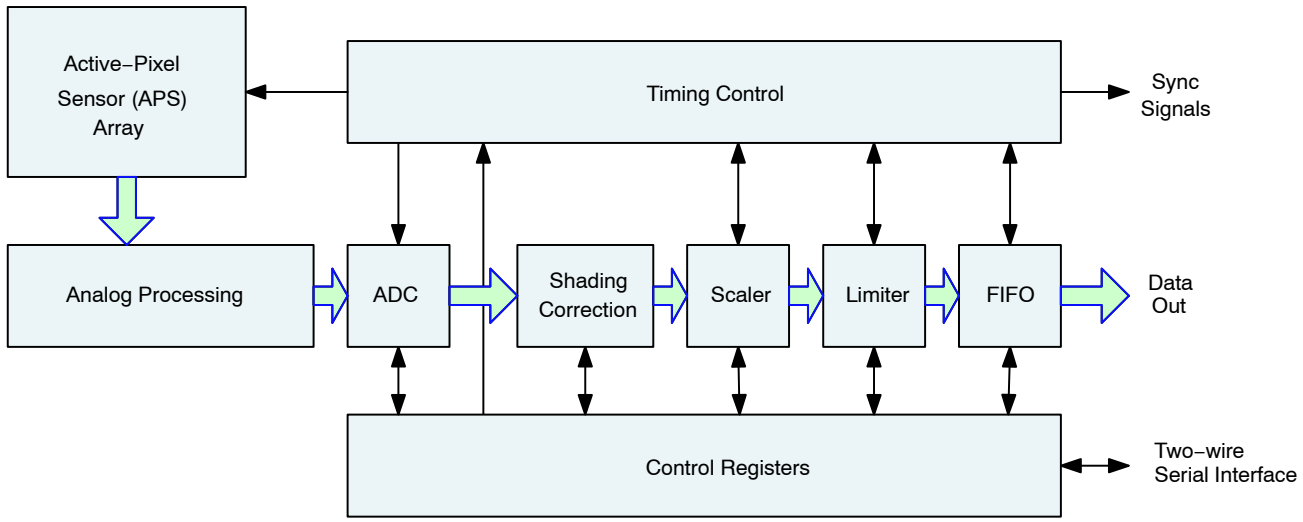


Figure 1. Block Diagram

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

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

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

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

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

- A sensor core that provides array control and data path corrections
- A digital shading correction block to compensate for color/brightness shading introduced by the lens or CRA curve mismatch
- Functionality to support the SMIA standard. This includes a horizontal and vertical image scaler, a limiter, a data compressor, an output FIFO, and a serializer.

The output FIFO prevents data bursts by keeping the data rate continuous.

Pixel Array

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

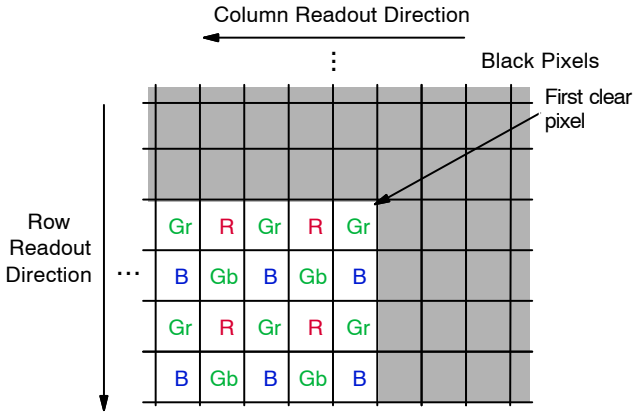


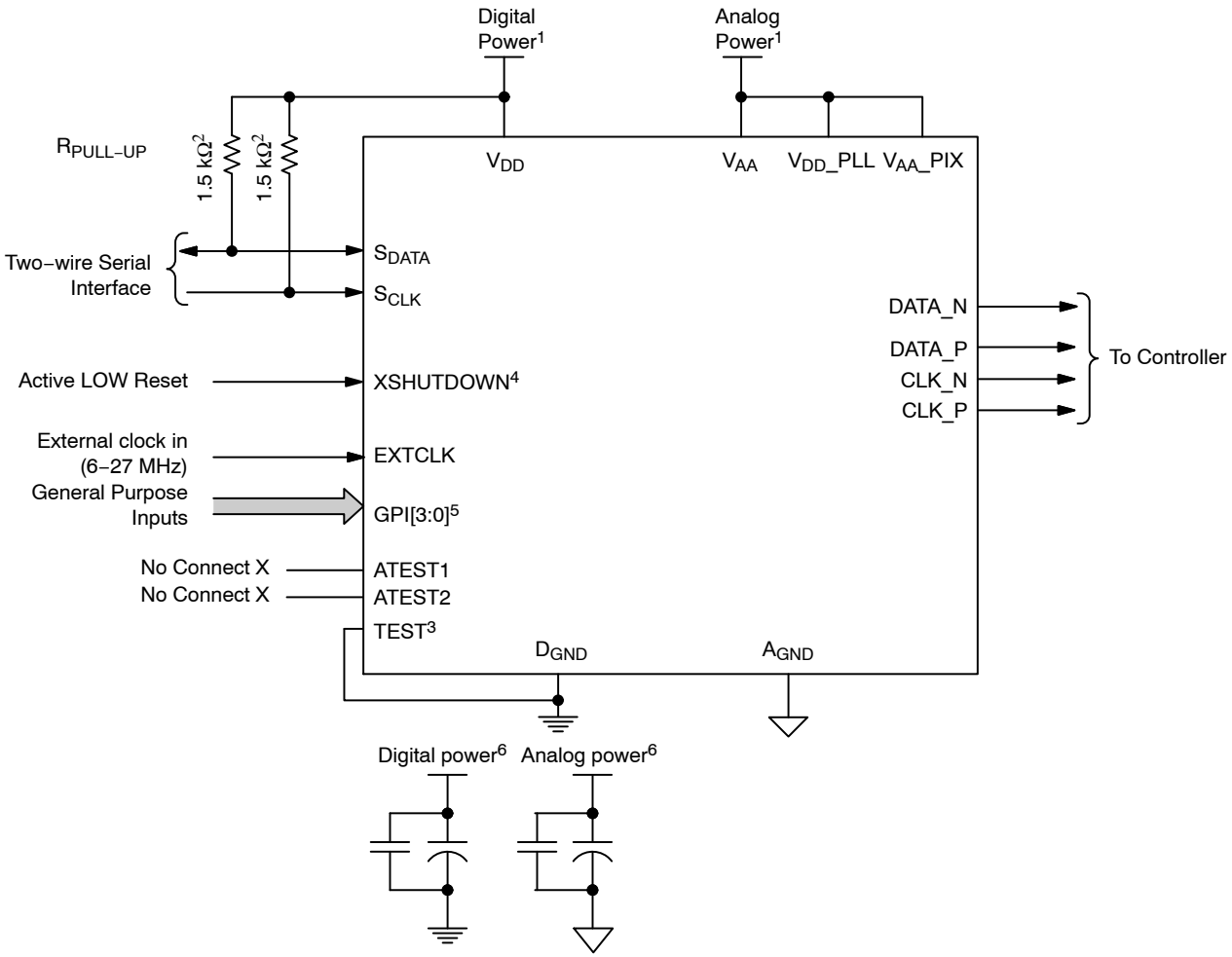
Figure 2. Pixel Color Pattern Detail (Top Right Corner)

OPERATING MODES

The MT9D015 can operate in either serial CCP2 or serial MIPI mode (preconfigured at the factory). In both cases, the sensor has a SMIA−compatible register interface while the I^2C device address is compliant with SMIA or MIPI requirements as appropriate. The reset level on the TEST pin must be tied in a way that is compatible with the configured serial interface of the sensor, for instance TEST = 0 for CCP2 and TEST = 1 for MIPI.

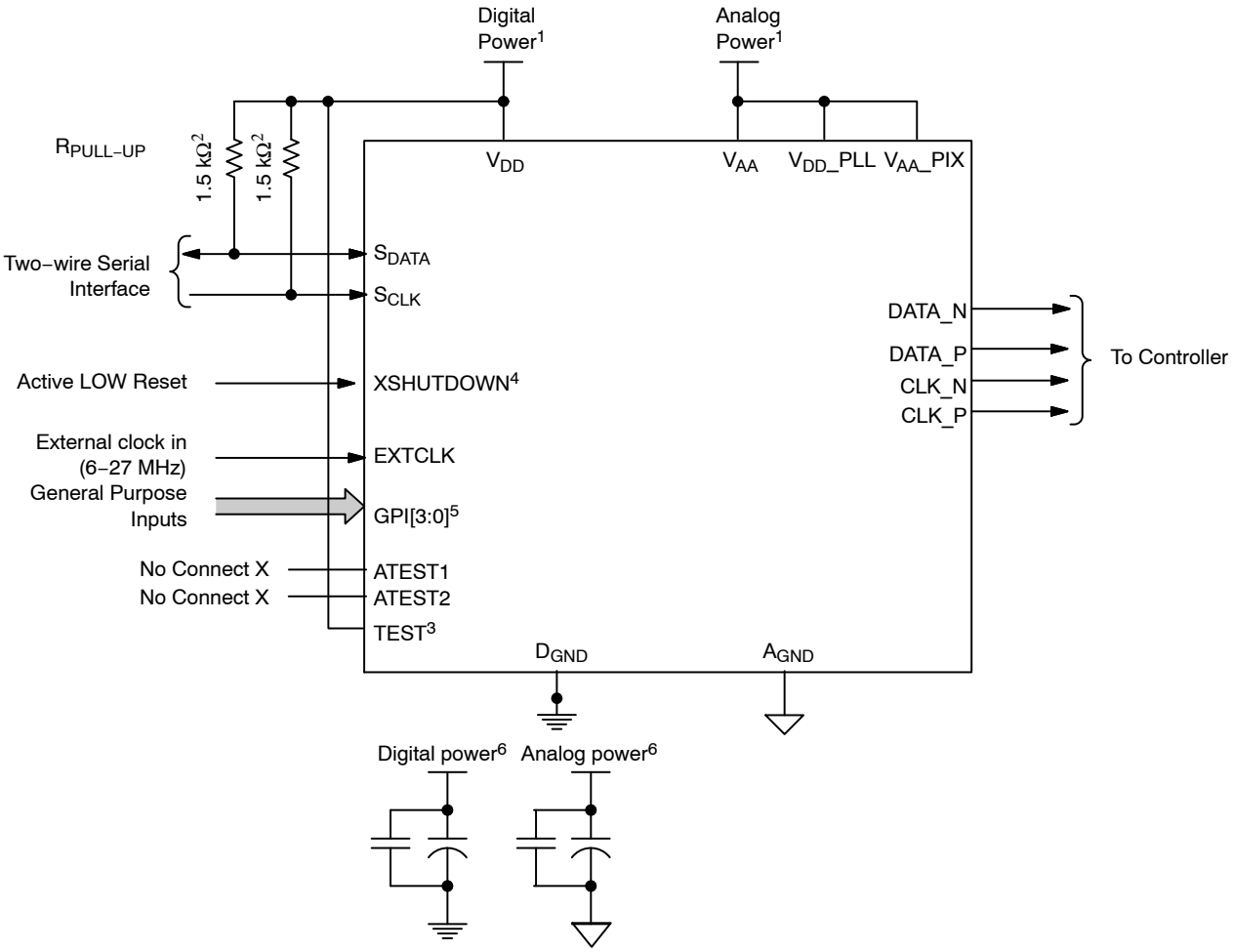
Typical configurations are shown in Figure 3 and Figure 4. These operating modes are described in "Control of the Signal Interface".

For low−noise operation, the MT9D015 requires separate power supplies for analog and digital. Incoming digital and analog ground conductors can be tied together next to the die. Both power supply rails should be decoupled from ground using capacitors as close as possible to the die. The use of inductance filters is not recommended on the power supplies or output signals.



Notes:
1. All power supplies must be adequately decoupled.
2. A resistor value of 1.5 kΩ is recommended, but it may be greater for slower two−wire speed.
3. TEST must be tied to GND for SMIA configuration.
4. Also referred to as RESET_BAR.
5. The GPI pins can be statically pulled HIGH or LOW and used as module IDs. All GPI pins must be driven to avoid leakage current.
6. ON Semiconductor recommends that 0.1 μF and 1 μF decoupling capacitors for each power supply are mounted as close as possible to the pad. Actual values and results may vary depending on layout and design considerations.
7. V_{PP}, which can be used during the module manufacturing process, is not shown in Figure 3. This pad is left unconnected during normal operation.
8. ATEST1 and ATEST2 must be floating.

Figure 3. Typical Configuration (Connection) −Serial Output Mode



Notes:
1. All power supplies must be adequately decoupled.
2. A resistor value of 1.5 kΩ is recommended, but it may be greater for slower two−wire speed.
3. TEST must be tied to V_{DD} for MIPI configuration.
4. Also referred to as RESET_BAR.
5. The GPI pins can be statically pulled HIGH or LOW and used as module IDs. All GPI pins must be driven to avoid leakage current.
6. ON Semiconductor recommends that 0.1 μF and 1 μF decoupling capacitors for each power supply are mounted as close as possible to the pad. Actual values and results may vary depending on layout and design considerations.
7. V_{PP}, which can be used during the module manufacturing process, is not shown in Figure 3. This pad is left unconnected during normal operation.
8. ATEST1 and ATEST2 must be floating.

Figure 4. Typical Configuration (Connection) – MIPI Mode

SIGNAL DESCRIPTIONS

Table 3 provides signal descriptions for MT9D015 die. For pad location and aperture information, refer to the MT9D015 die data sheet.

Table 3. SIGNAL DESCRIPTION

Pad Name	Pad Type	Description
EXTCLK	Input	Master clock input. PLL input clock. 6–27 MHz
RESET_BAR (XSHUTDOWN)	Input	Asynchronous active LOW reset. When asserted, data output stops and all internal registers are restored to their factory default settings
SCLK	Input	Serial clock for access to control and status registers
GPI[3:0]	Input	General purpose inputs After reset, these pads are powered up (enabled−see R0x301A) by default; these pads must be bonded to a HIGH or LOW state Failure to bond as required will result in excessive power consumption
TEST	Input	Enable manufacturing test modes. Connect to DGND for normal operation of the CCP2−configured sensor, or connect to VDD power for the MIPI configured sensor.
SDATA	I/O	Serial data for reads from and writes to control and status registers
DATA_P	Output	Differential CCP2/MIPI (sub−LVDS) serial data (positive)
DATA_N	Output	Differential CCP2/MIPI (sub−LVDS) serial data (negative)
CLK_P	Output	Differential CCP2/MIPI (sub−LVDS) serial clock/strobe (positive)
CLK_N	Output	Differential CCP2/MIPI (sub−LVDS) serial clock/strobe (negative)
VAA	Supply	Analog power supply
VDD_PLL	Supply	PLL power supply
VAA_PIX	Supply	Analog power supply
AGND	Supply	Analog ground
VDD	Supply	Digital power supply
DGND	Supply	Digital ground
VPP	Supply	OTPM programming power supply

TWO−WIRE SERIAL REGISTER INTERFACE

The two−wire serial interface bus enables read/write access to control and status registers within the sensor. This interface is designed to be compatible with the SMIA 1.0 Part 2: CCP2 Specification camera control interface (CCI), which uses the electrical characteristics and transfer protocols of the I^2C specification.

The protocols described in the I^2C specification allow the slave device to drive SCLK LOW; the sensor uses SCLK as an input only and therefore never drives it LOW.

Protocol

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

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

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

Start Condition

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

Slave Address/Data Direction Byte

Bits [7:1] of this byte represent the device slave address and bit [0] indicates the data transfer direction. A "0" in bit [0] indicates a write, and a "1" indicates a read. The default slave addresses used by the MT9D015 for the MIPI configured sensor are 0x6C (write address) and 0x6D (read address) in accordance with the MIPI specification. But for the CCP2 configured sensor, the default slave addresses used are 0x20 (write address) and 0x21 (read address) in accordance with the SMIA specification.

Acknowledge Bit

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

No−Acknowledge Bit

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

Message Byte

Message bytes are used for sending register addresses and register write data to the slave device and for retrieving register read data. The protocol used is outside the scope of the SMIA CCI.

Stop Condition

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

Data Transfer

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

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

Typical Sequence

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

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

Single READ from Random Location

This sequence (Figure 5) 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 1 byte of register data. The master terminates the READ by generating a no−acknowledge bit followed by a stop condition. Figure 5 shows how the internal register address maintained by the MT9D015 is loaded and incremented as the sequence proceeds.

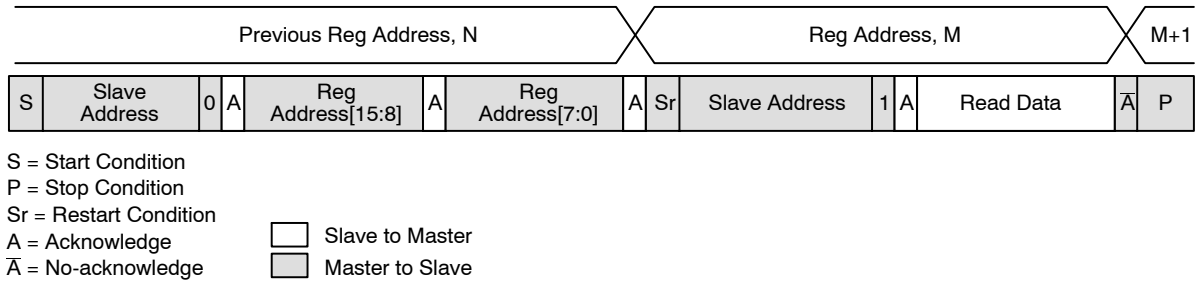


Figure 5. Single READ from Random Location

Single READ From Current Location

This sequence (Figure 6) performs a read using the current value of the MT9D015 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.

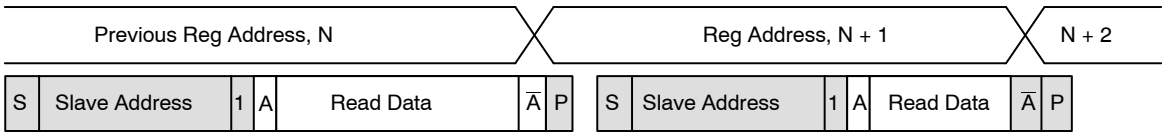


Figure 6. Single READ from Current Location

Sequential READ, Start From Random Location

This sequence (Figure 7) starts in the same way as the single READ from random location (Figure 5). 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.

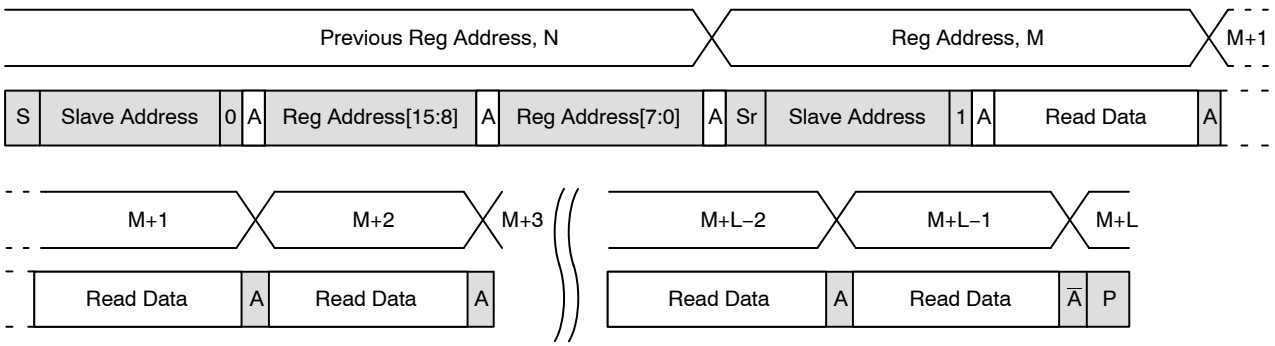


Figure 7. Sequential READ, Start from Random Location

Sequential READ, Start From Current Location

This sequence (Figure 8) starts in the same way as the single READ from current location (Figure 6). Instead of generating a no−acknowledge bit after the first byte of data has been transferred, the master generates an acknowledge bit and continues to perform byte reads until L bytes have been read.

Figure 8. Sequential READ, Start from Current Location

Single WRITE to Random Location

This sequence (Figure 9) 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.

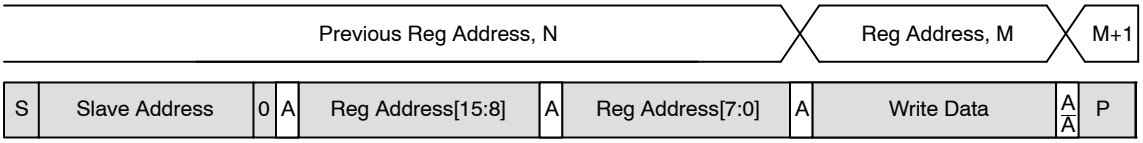


Figure 9. Single WRITE to Random Location

Sequential WRITE, Start at Random Location

This sequence (Figure 10) starts in the same way as the single WRITE to random location (Figure 9). Instead of generating a stop condition after the first byte of data has been transferred, the master continues to perform byte writes until *L* bytes have been written. The WRITE is terminated by the master generating a stop condition.

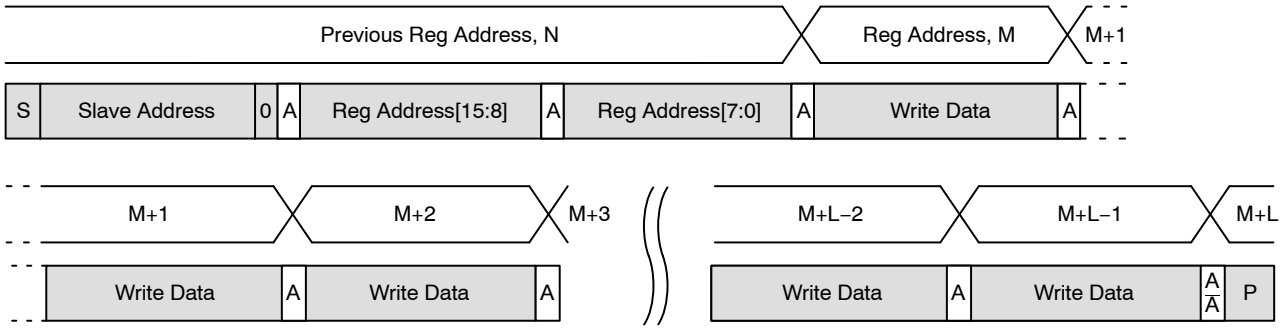


Figure 10. Sequential WRITE, Start at Random Location

REGISTERS

NOTE: The detailed register lists and descriptions are in a separate document, the MT9D015 Register Reference.

The MT9D015 provides a 32−bit register address space accessed through a serial interface (“Single READ from Random Location”). Each register location is 8 or 16 bits in size.

The address space is divided into the five major regions shown in Table 4.

Table 4. ADDRESS SPACE REGIONS

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

Register Notation

The underlying mechanism for reading and writing registers provides byte write capability. However, it is convenient to consider some registers as multiple adjacent bytes. The MT9D015 uses 8−bit, 16−bit, and 32−bit registers, all implemented as 1 or more bytes at naturally aligned, contiguous locations in the address space.

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

Register Aliases

A consequence of the internal architecture of the MT9D015 is that some registers are decoded at multiple addresses. Some registers in “configuration space” are also decoded in “manufacturer−specific space.” To provide unique names for all registers, the name of the register within manufacturer−specific register space has a trailing underscore. For example, R0x0000–1 is model_id, and R0x3000–1 is model_id_ (see the register table for more examples). The effect of reading or writing a register through any of its aliases is identical.

Bit Fields

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

Bit Field Aliases

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

Byte Ordering

Registers that occupy more than 1 byte of address space are shown with the lowest address in the highest−order byte lane to match the byte−ordering on the SMIA bus. For example, the model_id register is R0x0000–1. In the register table the default value is shown as 0x1501. This means that a READ from address 0x0000 would return 0x15, and a READ from address 0x0001 would return 0x01. When reading this register as two 8−bit transfers on the serial interface, the 0x15 will appear on the serial interface first, followed by the 0x01.

Address Alignment

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

Bit Representation

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

Data Format

Most registers represent an unsigned binary value or set of bit fields. For all other register formats, the format is stated explicitly at the start of the register description. The notation for these formats is shown in Table 5.

Table 5. DATA FORMATS

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

Register Behavior

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

Double−Buffered Registers

Some sensor settings cannot be changed during frame readout. For example, changing R0x0344–5 (x_addr_start) partway through frame readout would result in inconsistent row lengths within a frame. To avoid this, the MT9D015 double−buffers many registers by implementing a "pending" and a "live" version. READs and WRITEs access the pending register. The live register controls the sensor operation.

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

Using grouped_parameter_hold

Register grouped_parameter_hold (R0x0104) can be used to inhibit transfers from the pending to the live registers. When the MT9D015 is in streaming mode, write "1" to this register before making changes to any group of registers where a set of changes is required to take effect simultaneously. When this register is set to "0," all transfers from pending to live registers take place on the next frame start.

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

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

Bad Frames

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

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

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

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

Changes to Integration Time

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

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

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

Changes to Gain Settings

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

Embedded Data

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

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

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

EMBEDDED DATA FORMAT AND CONTROL

When the serial pixel data path is selected, the first two rows of the output image contain register values that are appropriate for the image. In this mode, the first two lines and the last line of data are not equally spaced. The format of this data is shown in Table 6. In the table, 8−bit (RAW8) and 10−bit (RAW10) versions of the data are shown. The 10−bit format places the data byte in bits [9:2] and sets bits [1:0] to a constant value of 01. Register values that are shown as "??" are dynamic and may change from frame to frame.

When the parallel pixel data path is selected and R0x306E − F[2:0] = 2 (parallel pixel data output MUX selects FIFO data). The output image contains two rows of embedded data.

Table 6. EMBEDDED DATA

Row 0					Row 1			
Offset	10−bit	8−bit	Two−wire Serial Interface Address	Comment	10−Bit	8−Bit	Two−wire Serial Interface Address	Comment
0	0X029	0X0A		2−byte tagged data format (embedded data)	0X029	0X0A		2−byte tagged data format (embedded data)
1	0X2A9	0XAA		cci register index msb	0X2A9	0XAA		CCI register index MSB
2	0X001	0X00		address 00xx	0X009	0X02		Address 02xx
3	0X295	0XA5		cci register index lsb	0X295	0XA5		CCI register index LSB
4	0X001	0X00		address xx00	0X001	0X00		Address xx00
5	0X169	0X5A		auto increment	0X169	0X5A		auto increment
6	0X055	0X15	0	model_id hi	??	??	200	fine_integration_time hi
7	0X169	0X5A			0X169	0X5A		
8	0X005	0X01	1	model_id lo	??	??	201	fine_integration_time lo
9	0X169	0X5A			0X169	0X5A		
10	0X080	0X20	2	revision_number	??	??	202	coarse_integration_time hi
11	0X169	0X5A			0X169	0X5A		
12	0X019	0X06	3	manufacturer_id	??	??	203	coarse_integration_time lo
13	0X169	0X5A			0X169	0X5A		
14	0X029	0X0A	4	smia_version	??	??	204	analogue_gain_code_global hi
15	0X169	0X5A			0X169	0X5A		
16	??	??	5	frame_count	??	??	205	analogue_gain_code_global lo
17	0X169	0X5A			0X169	0X5A		
18	??	??	6	pixel_order	??	??	206	analogue_gain_code_greenR hi
19	0X169	0X5A			0X169	0X5A		
20	??	??	7	reserved	??	??	207	analogue_gain_code_greenR lo
21	0X169	0X5A			0X169	0X5A		
22	0X001	0X00	8	data_pedestal_hi	??	??	208	analogue_gain_code_red hi
23	0X169	0X5A			0X169	0X5A		
24	0X0A9	0X2A	9	data_pedestal lo	??	??	209	analogue_gain_code_red lo
25	0X2A9	0XAA		cci register index msb	0X169	0X5A		
26	0X001	0X00		address 00xx	??	??	020a	analogue_gain_code_blue hi
27	0X295	0XA5		cci register index lsb	0X169	0X5A		
28	0X041	0X10		address xx10	??	??	020b	analogue_gain_code_blue lo
29	0X169	0X5A		auto increment	0X169	0X5A		
30	0X005	0X01	10	revision_number_minor	??	??	020c	analogue_gain_code_greenB hi

Table 6. EMBEDDED DATA (continued)

Row 0					Row 1			
Offset	10−bit	8−bit	Two−wire Serial Interface Address	Comment	10−Bit	8−Bit	Two−wire Serial Interface Address	Comment
31	0X169	0X5A			0X169	0X5A		
32	0X001	0X00	11	smia_pp_version	??	??	020d	analogue_gain_codegreenB lo
33	0X169	0X5A			0X169	0X5A		
34	0X001	0X00	12	module_date_year	??	??	020e	digital_gain_greenR hi
35	0X169	0X5A			0X169	0X5A		
36	0X001	0X00	13	module_date_month	??	??	020f	digital_gain_greenR lo
37	0X169	0X5A			0X169	0X5A		
38	0X001	0X00	14	module_date_day	??	??	210	digital_gain_red hi
39	0X169	0X5A			0X169	0X5A		
40	0X001	0X00	15	module_date_phase	??	??	211	digital_gain_red lo
41	0X169	0X5A			0X169	0X5A		
42	0X001	0X00	16	sensor_model_id hi	??	??	212	digital_gain_blue hi
43	0X169	0X5A			0X169	0X5A		
44	0X001	0X00	17	sensor_model_id lo	??	??	213	digital_gain_blue lo
45	0X169	0X5A			0X169	0X5A		
46	0X005	0X01	18	sensor_revision_number	??	??	214	digital_gain_greenB hi
47	0X169	0X5A			0X169	0X5A		
48	0X001	0X00	19	sensor_manufacturer_id	??	??	215	digital_gain_greenB lo
49	0X169	0X5A			0X2A9	0XAA		CCI register index MSB
50	0X001	0X00	1A	sensor_firmwave_version	0X00D	0X03		Address 03xx
51	0X169	0X5A			0X295	0XA5		CCI register index LSB
52	??	??	1B	reserved	0X001	0X00		Address xx00
53	0X169	0X5A			0X169	0X5A		auto increment
54	0X001	0X00	1C	serial_number_0 hi	??	??	300	vt_pix_clk_div hi
55	0X169	0X5A			0X169	0X5A		
56	0X001	0X00	1D	serial_number_0 lo	??	??	301	vt_pix_clk_div lo
57	0X169	0X5A			0X169	0X5A		
58	0X001	0X00	1E	serial_number_1 hi	??	??	302	vt_sys_clk_div hi
59	0X169	0X5A			0X169	0X5A		
60	0X001	0X00	1F	serial_number_1 lo	??	??	303	vt_sys_clk_div lo
61	0X2A9	0XAA		cci register index msb	0X169	0X5A		
62	0X001	0X00		address 00xx	??	??	304	pre_pll_clk_div hi
63	0X295	0XA5		cci register index lsb	0X169	0X5A		
64	0X101	0X40		address xx40	??	??	305	pre_pll_clk_div lo
65	0X169	0X5A		auto increment	0X169	0X5A		
66	0X005	0X01	40	frame_format_ model_type	??	??	306	pll_multiplier_hi
67	0X169	0X5A			0X169	0X5A		

Table 6. EMBEDDED DATA (continued)

Row 0					Row 1			
Offset	**10−bit**	**8−bit**	**Two−wire Serial Interface Address**	**Comment**	**10−Bit**	**8−Bit**	**Two−wire Serial Interface Address**	**Comment**
68	0X049	0X12	41	frame_format_ model_subtype	??	??	307	pll_multiplier_lo
69	0X169	0X5A			0X169	0X5A		
70	??	??	42	frame_format_ descriptor_0 hi	??	??	308	op_pix_clk_div hi
71	0X169	0X5A			0X169	0X5A		
72	??	??	43	frame_format_ descriptor_0 lo	??	??	309	op_pix_clk_div lo
73	0X169	0X5A			0X169	0X5A		
74	??	??	44	frame_format_ descriptor_1 hi	??	??	030a	op_sys_clk_div hi
75	0X169	0X5A			0X169	0X5A		
76	??	??	45	frame_format_ descriptor_1 lo	??	??	030b	op_sys_clk_div lo
77	0X169	0X5A			0X2A9	0XAA		CCI register index MSB
78	??	??	46	frame_format_ descriptor_2 hi	0X00D	0X03		Address 03xx
79	0X169	0X5A			0X295	0XA5		CCI register index LSB
80	??	??	47	frame_format_ descriptor_2 lo	0X101	0X40		Address xx40
81	0X169	0X5A			0X169	0X5A		auto increment
82	0X001	0X00	48	frame_format_ descriptor_3 hi	??	??	340	frame_length_lines hi
83	0X169	0X5A			0X169	0X5A		
84	0X001	0X00	49	frame_format_ descriptor_3 lo	??	??	341	frame_length_lines lo
85	0X169	0X5A			0X169	0X5A		
86	0X001	0X00	004a	frame_format_ descriptor_4 hi	??	??	342	line_length_pck hi
87	0X169	0X5A			0X169	0X5A		
88	0X001	0X00	004b	frame_format_ descriptor_4 lo	??	??	343	line_length_pck lo
89	0X169	0X5A			0X169	0X5A		
90	0X001	0X00	004c	frame_format_ descriptor_5 hi	??	??	344	x_addr_start hi
91	0X169	0X5A			0X169	0X5A		
92	0X001	0X00	004d	frame_format_ descriptor_5 lo	??	??	345	x_addr_start lo
93	0X169	0X5A			0X169	0X5A		
94	0X001	0X00	004e	frame_format_ descriptor_6 hi	??	??	346	y_addr_start hi
95	0X169	0X5A			0X169	0X5A		
96	0X001	0X00	004f	frame_format_ descriptor_6 lo	??	??	347	y_addr_start lo

Table 6. EMBEDDED DATA (continued)

Row 0					Row 1			
Offset	10−bit	8−bit	Two−wire Serial Interface Address	Comment	10−Bit	8−Bit	Two−wire Serial Interface Address	Comment
97	0X169	0X5A			0X169	0X5A		
98	0X001	0X00	50	frame_format_ descriptor_7 hi	??	??	348	x_addr_end hi
99	0X169	0X5A			0X169	0X5A		
100	0X001	0X00	51	frame_format_ descriptor_7 lo	??	??	349	x_addr_end lo
101	0X169	0X5A			0X169	0X5A		
102	0X001	0X00	52	frame_format_ descriptor_8 hi	??	??	034a	y_addr_end hi
103	0X169	0X5A			0X169	0X5A		
104	0X001	0X00	53	frame_format_ descriptor_8 lo	??	??	034b	y_addr_end lo
105	0X169	0X5A			0X169	0X5A		
106	0X001	0X00	54	frame_format_ descriptor_9 hi	??	??	034c	x_output_size hi
107	0X169	0X5A			0X169	0X5A		
108	0X001	0X00	55	frame_format_ descriptor_9 lo	??	??	034d	x_output_size lo
109	0X169	0X5A			0X169	0X5A		
110	0X001	0X00	56	frame_format_ descriptor_10 hi	??	??	034e	y_output_size hi
111	0X169	0X5A			0X169	0X5A		
112	0X001	0X00	57	frame_format_ descriptor_10 lo	??	??	034f	y_output_size lo
113	0X169	0X5A			0X2A9	0XAA		CCI register index MSB
114	0X001	0X00	58	frame_format_ descriptor_11 hi	0X00D	0X03		Address 02xx
115	0X169	0X5A			0X295	0XA5		CCI register index LSB
116	0X001	0X00	59	frame_format_ descriptor_11 lo	0X201	0X80		Address xx80
117	0X169	0X5A			0X169	0X5A		auto increment
118	0X001	0X00	005a	frame_format_ descriptor_12 hi	??	??	380	x_even_inc hi
119	0X169	0X5A			0X169	0X5A		
120	0X001	0X00	005b	frame_format_ descriptor_12 lo	??	??	381	x_even_inc lo
121	0X169	0X5A			0X169	0X5A		
122	0X001	0X00	005c	frame_format_ descriptor_13 hi	??	??	382	y_odd_inc hi
123	0X169	0X5A			0X169	0X5A		
124	0X001	0X00	005d	frame_format_ descriptor_13 lo	??	??	383	y_odd_inc lo
125	0X169	0X5A			0X169	0X5A		

Table 6. EMBEDDED DATA (continued)

Row 0					Row 1			
Offset	**10−bit**	**8−bit**	**Two−wire Serial Interface Address**	**Comment**	**10−Bit**	**8−Bit**	**Two−wire Serial Interface Address**	**Comment**
126	0X001	0X00	005e	frame_format_ descriptor_14 hi	??	??	384	y_even_inc hi
127	0X169	0X5A			0X169	0X5A		
128	0X001	0X00	005f	frame_format_ descriptor_14 lo	??	??	385	y_even_inc lo
129	0X2A9	0XAA		cci register index msb	0X169	0X5A		
130	0X001	0X00		address 00xx	??	??	386	x_odd_inc hi
131	0X295	0XA5		cci register index lsb	0X169	0X5A		
132	0X201	0X80		address xx80	??	??	387	x_odd_inc lo
133	0X169	0X5A		auto increment	0X2A9	0XAA		CCI register index MSB
134	0X001	0X00	80	analogue_gain_capability hi	0X011	0X04		Address 04xx
135	0X169	0X5A			0X295	0XA5		CCI register index LSB
136	0X005	0X01	81	analogue_gain_capability lo	0X001	0X00		Address xx00
137	0X2A9	0XAA		cci register index msb	0X169	0X5A		auto increment
138	0X001	0X00		address 00xx	??	??	400	scaling_mode hi
139	0X295	0XA5		cci register index lsb	0X169	0X5A		
140	0X211	0X84		address xx84	??	??	401	scaling_mode lo
141	0X169	0X5A		auto increment	0X169	0X5A		
142	0X001	0X00	84	analogue_gain_code_min hi	??	??	402	spatial_sampling hi
143	0X169	0X5A			0X169	0X5A		
144	0X021	0X08	85	analogue_gain_code_ min lo	??	??	403	spatial_sampling lo
145	0X169	0X5A			0X169	0X5A		
146	0X001	0X00	86	analogue_gain_code_ max hi	??	??	404	scale_m hi
147	0X169	0X5A			0X169	0X5A		
148	0X1FD	0X7F	87	analogue_gain_code_ max lo	??	??	405	scale_m lo
149	0X169	0X5A			0X169	0X5A		
150	0X001	0X00	88	ana- logue_gain_code_step hi	0X001	0X00	406	scale_n hi
151	0X169	0X5A			0X169	0X5A		
152	0X005	0X01	89	analogue_gain_code_ step lo	0X041	0X10	407	scale_n lo
153	0X169	0X5A			0X2A9	0XAA		CCI register index MSB
154	0X001	0X00	008a	analogue_gain_type hi	0X015	0X05		Address 05xx
155	0X169	0X5A			0X295	0XA5		CCI register index LSB
156	0X001	0X00	008b	analogue_gain_type lo	0X001	0X00		Address xx00
157	0X169	0X5A			0X169	0X5A		auto increment

Table 6. EMBEDDED DATA (continued)

Row 0					Row 1			
Offset	10−bit	8−bit	Two−wire Serial Interface Address	Comment	10−Bit	8−Bit	Two−wire Serial Interface Address	Comment
158	0X001	0X00	008c	analogue_gain_m0 lo	0X001	0X00	500	compression_mode hi
159	0X169	0X5A			0X169	0X5A		
160	0X005	0X01	008d	analogue_gain_m0 lo	0X005	0X01	501	compression_mode lo
161	0X169	0X5A			0X2A9	0XAA		CCI register index MSB
162	0X001	0X00	008e	analogue_gain_c0 lo	0X019	0X06		Address 06xx
163	0X169	0X5A			0X295	0XA5		CCI register index LSB
164	0X001	0X00	008f	analogue_gain_c0 lo	0X001	0X00		Address xx00
165	0X169	0X5A			0X169	0X5A		auto increment
166	0X001	0X00	90	analogue_gain_m1 lo	??	??	600	test_pattern_mode hi
167	0X169	0X5A			0X169	0X5A		
168	0X001	0X00	91	analogue_gain_m1 lo	??	??	601	test_pattern_mode lo
169	0X169	0X5A			0X169	0X5A		
170	0X001	0X00	92	analogue_gain_c1 lo	??	??	602	test_data_red hi
171	0X169	0X5A			0X169	0X5A		
172	0X021	0X08	93	analogue_gain_c1 lo	??	??	603	test_data_red lo
173	0X2A9	0XAA		cci register index msb	0X169	0X5A		
174	0X001	0X00		address 00xx	??	??	604	test_data_greenR hi
175	0X295	0XA5		cci register index lsb	0X169	0X5A		
176	0X301	0XC0		address xxc0	??	??	605	test_data_greenR lo
177	0X169	0X5A		auto increment	0X169	0X5A		
178	0X005	0X01	00C0	data_format_model_type	??	??	606	test_data_blue hi
179	0X169	0X5A			0X169	0X5A		
180	0X00D	0X03	00c1	data_format_model_ subtype	??	??	607	test_data_blue lo
181	0X169	0X5A			0X169	0X5A		
182	0X029	0X0A	00c2	data_format_descriptor_ 0 hi	??	??	608	test_data_greenB hi
183	0X169	0X5A			0X169	0X5A		
184	0X029	0X0A	00c3	data_format_descriptor_ 0 lo	??	??	609	test_data_greenB lo
185	0X169	0X5A			0X169	0X5A		
186	0X021	0X08	00c4	data_format_descriptor_ 1 hi	??	??	060a	horizontal_cursor_width hi
187	0X169	0X5A			0X169	0X5A		
188	0X021	0X08	00c5	data_format_descriptor_ 1 lo	??	??	060b	horizontal_cursor_width lo
189	0X169	0X5A			0X169	0X5A		
190	0X029	0X0A	00c6	data_format_descriptor_ 2 hi	??	??	060c	horizontal_cursor_position hi
191	0X169	0X5A			0X169	0X5A		

Table 6. EMBEDDED DATA (continued)

Row 0					Row 1			
Offset	**10−bit**	**8−bit**	**Two−wire Serial Interface Address**	**Comment**	**10−Bit**	**8−Bit**	**Two−wire Serial Interface Address**	**Comment**
192	0X021	0X08	00c7	data_format_descriptor_ 2 lo	??	??	060d	horizontal_cursor_position lo
193	0X169	0X5A			0X169	0X5A		
194	0X001	0X00	00c8	data_format_descriptor_ 3 hi	??	??	060e	vertical_cursor_width hi
195	0X169	0X5A			0X169	0X5A		
196	0X001	0X00	00c9	data_format_descriptor_ 3 lo	??	??	060f	vertical_cursor_width lo
197	0X169	0X5A			0X169	0X5A		
198	0X001	0X00	00ca	data_format_descriptor_ 4 hi	??	??	610	vertical_cursor_position hi
199	0X169	0X5A			0X169	0X5A		
200	0X001	0X00	00cb	data_format_descriptor_ 4 lo	??	??	611	vertical_cursor_position lo
201	0X169	0X5A			0X01D	0X07		Null Data
202	0X001	0X00	00cc	data_format_descriptor_ 5 hi	0X01D	0X07		Null Data − up to end−of−line
203	0X169	0X5A						
204	0X001	0X00	00cd	data_format_descriptor_ 5 lo				
205	0X169	0X5A						
206	0X001	0X00	00ce	data_format_descriptor_ 6 hi				
207	0X169	0X5A						
208	0X001	0X00	00cf	data_format_descriptor_ 6 lo				
209	0X2A9	0XAA		cci register index msb				
210	0X005	0X01		address 01xx				
211	0X295	0XA5		cci register index lsb				
212	0X001	0X00		address xx00				
213	0X169	0X5A		auto increment				
214	??	??	100	mode_select				
215	0X169	0X5A						
216	??	??	101	image_orientation				
217	0X169	0X5A						
218	??	??	102	reserved				
219	0X169	0X5A						
220	0X001	0X00	103	software_reset				
221	0X169	0X5A						
222	??	??	104	grouped_parameter_hold				
223	0X169	0X5A						
224	??	??	105	mask_corrupted_frames				

Table 6. EMBEDDED DATA (continued)

Row 0					Row 1			
Offset	**10−bit**	**8−bit**	**Two−wire Serial Interface Address**	**Comment**	**10−Bit**	**8−Bit**	**Two−wire Serial Interface Address**	**Comment**
225	0X2A9	0XAA		cci register index msb				
226	0X005	0X01		address 01xx				
227	0X295	0XA5		cci register index lsb				
228	0X041	0X10		address xx10				
229	0X169	0X5A		auto increment				
230	??	??	110	ccp2_channel_identifier				
231	0X169	0X5A						
232	??	??	111	ccp2_signalling_mode				
233	0X169	0X5A						
234	??	??	112	ccp_data_format_hi				
235	0X169	0X5A						
236	??	??	113	ccp_data_format_lo				
237	0X2A9	0XAA		cci register index msb				
238	0X005	0X01		address 01xx				
239	0X295	0XA5		cci register index lsb				
240	0X081	0X20		address xx20				
241	0X169	0X5A		auto increment				
242	0X001	0X00	120	gain_mode				
243	0X169	0X5A						
244	??	??	121	reserved				
245	0X01D	0X07		null data				
246	0X01D	0X07		null data − up to end−of−line				

PROGRAMMING RESTRICTIONS

The SMIA specification imposes a number of programming restrictions. An implementation naturally imposes additional restrictions. Table 7 shows a list of programming rules that must be adhered to for correct operation of the MT9D015. ON Semiconductor recommends that these rules are encoded into the device driver stack−either implicitly or explicitly.

Table 7. DEFINITIONS FOR PROGRAMMING RULES

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

Table 8. PROGRAMMING RULES

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

Table 8. PROGRAMMING RULES (continued)

Parameter	Minimum Value	Maximum Value	Origin
scale_m	scaler_m_min	scaler_m_max	SMIA
scale_n	scaler_n_min	scaler_n_max	SMIA
x_output_size	256	1608	Note 2
x_output_size[0]	0 (this is enforced in hardware: bit[0] is read−only)	0	Note 4
y_output_size	2	frame_length_lines	Note 3
y_output_size[0]	0 (this is enforced in hardware: bit[0] is read−only)	0	Note 4

1. With subsampling, start and end pixels must be addressed (impact on x/y start/end addresses, function of image orientation bits). SMIA FS Errata see “Subsampling”.
2. Minimum from SMIA FS Section 5.2.2.5. Maximum is a consequence of the output FIFO size on this implementation.
3. Minimum ensures 1 Bayer row−pair. Maximum avoids output frame being longer than pixel array frame.
4. SMIA FS Section 5.2.2.2.

Output Size Restrictions

The SMIA CCP2 specification imposes the restriction that an output line shall be a multiple of 32 bits in length. This imposes an additional restriction on the legal values of x_output_size:

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

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

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

Effect of Scaler on Legal Range of Output Sizes

When the scaler is enabled, it is necessary to adjust the values of x_output_size and y_output_size to match the image size generated by the scaler. The MT9D015 will not operate properly if the x_output_size and y_output_size are significantly larger than the output image. To understand the reason for this, consider the situation where the sensor is operating at full resolution and the scaler is enabled with a scaling factor of 32 (half the number of pixels in each direction). This situation is shown in Figure 11.

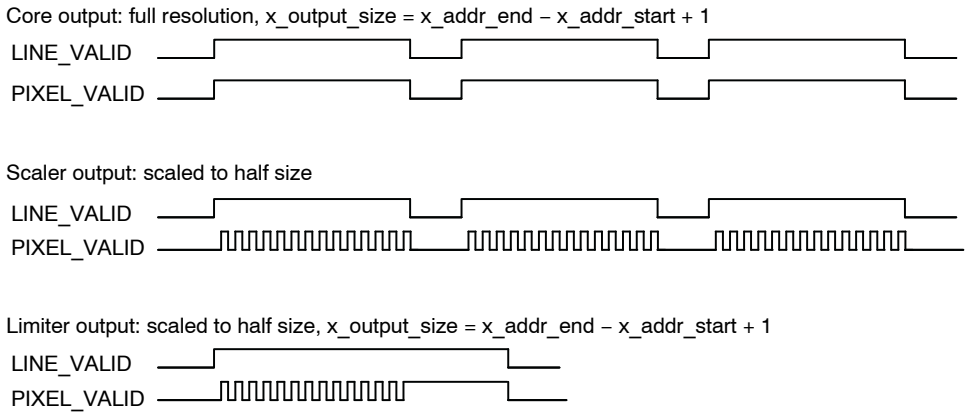


Figure 11. Effect of Limiter on the SMIA Data Path

In Figure 11, three different stages in the SMIA data path (see “Digital Data Path”) 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 LINE_VALID (LV) signal is asserted once per row and remains asserted for *N* pixel times. The PIXEL_VALID signal toggles with the same timing as LV, indicating that all pixels in the row are valid.

The second stage is the output of the scaler, when the scaler is set to reduce the image size by one−half in each dimension. The effect of the scaler is to combine groups of pixels. Therefore, the row time remains the same, but only

half the pixels out of the scaler are valid. This is signalled by transitions in PIXEL_VALID. Overall, PIXEL_VALID is asserted for (*N*/2) pixel times per row.

The third stage is the output of the limiter when the x_output_size is still set to match the active array size. Because the scaler has reduced the amount of valid pixel data without reducing the row time, the limiter attempts to pad the row with (*N*/2) additional pixels. If this has the effect of extending LV across the whole of the horizontal blanking time, the MT9D015 will cease to generate output frames.

A correct configuration is shown in Figure 12, 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 12 also shows the effect of the output FIFO, which forms the final stage in the SMIA data path. The output FIFO merges the intermittent pixel data back into a contiguous stream. Although not shown in this example, the output FIFO is also capable of operating with an output clock that is at a different frequency from its input clock.

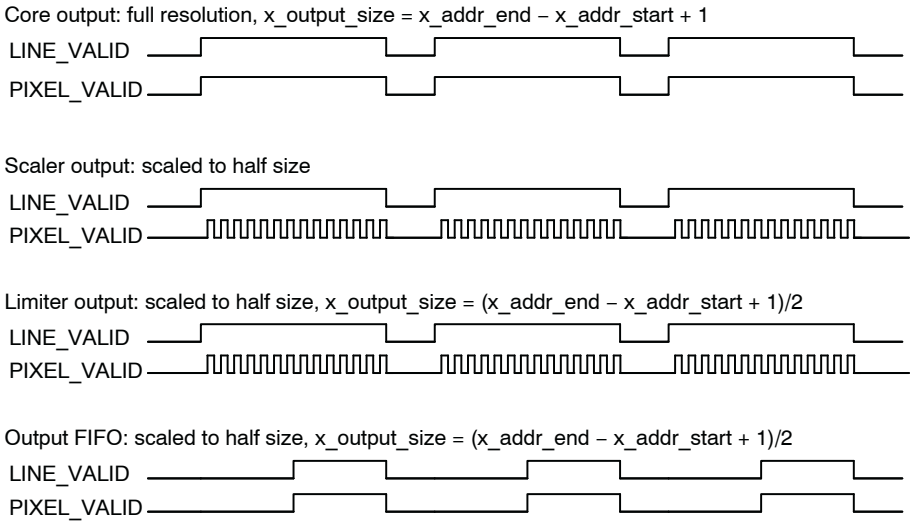


Figure 12. Timing of SMIA Data Path

Effect of CCP2 Class on Legal Range of Output Sizes/Frame Rate

The pixel array readout rate is set by *line_length_pck x frame_length_lines*. With the default register values, one frame time takes *2360 x 1283 = 3027880* pixel periods. This value includes vertical and horizontal blanking times so that the full−size image *1600 x 1202* (1200 lines of pixel data, 2 lines of embedded information) forms a subset of these pixels.

When the internal clock is running at 64 MHz, this frame time corresponds to *3027880/64e6 = 47.31* ms, giving rise to a frame rate of 21.14 fps.

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

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

The only way to get a full image out is to reduce the pixel clock rate until it is appropriate for the maximum CCP2 class 0 data rate. This requires the pixel rate to be reduced to 20.8 MHz. This has the side effect of reducing the frame rate. Repeating the calculation above, at 20.8 MHz internal clock, this corresponds to *3027880/20.8e6 = 145* ms, giving rise to a frame rate of 6.87 fps.

To use CCP2 class 0 with an internal clock of 64 MHz, it is necessary to reduce the amount of output data. This can be achieved by changing x_output_size, y_output_size so that less data comes out per frame. A change to the output size can be done in conjunction with windowing the image from the sensor (by adjusting x_addr_start, x_addr_end, y_addr_start, y_addr_end) or by enabling the scaler.

Output Data Timing

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

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

Maximum frame rate is achieved by setting the video timing clock (vt_clk_freq_mhz) to 91 MHz and using the FIFO to reduce horizontal blanking data rate to 640 Mb/s. At this setting, a maximum frame rate of 30 fps can be achieved.

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

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

Changing Registers while Streaming

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

- ccp2_channel_identifier
- ccp2_signalling_mode
- ccp_data_format
- scale_m
- vt_pix_clk_div
- vt_sys_clk_div
- pre_pll_clk_div
- pll_multiplier
- op_pix_clk_div
- op_sys_clk_div

CONTROL OF THE SIGNAL INTERFACE

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

Serial Register Interface

The serial register interface uses the following signals:

- SCLK
- SDATA

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 state, 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.

This interface is described in detail in “EXTCLK”.

Default Power−Up State

The MT9D015 provides interfaces for pixel data through the CCP2 high−speed serial interface described by the SMIA specification or the MIPI serial interface.

At power up and after a hard or soft reset, the reset state of the MT9D015 is to enable the SMIA CCP2 high speed serial interface for a CCP2−configured sensor, and CSI−2 high speed serial interface for a MIPI−configured sensor.

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

Serial Pixel Data Interface

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

- DATA_P
- DATA_N
- CLK_P
- CLK_N

The signal pairs are driven differentially using sub−LVDS switching levels. This interface conforms to the MIPI 1.0 CSI−2 and SMIA CCP2 requirements and supports both data/ clock signalling and data/strobe signalling.

The serial pixel data interface is enabled by default at power up and after reset. DATA_P and DATA_N are the data pair for the CCP2 or MIPI serial interface.

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

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

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

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

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

- 0x0101 – sensor supports single−lane CCP2 operation
- 0x0201 – sensor supports single−lane MIPI operation

System States

The system states of the MT9D015 are represented as a state diagram in Figure 13 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.

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

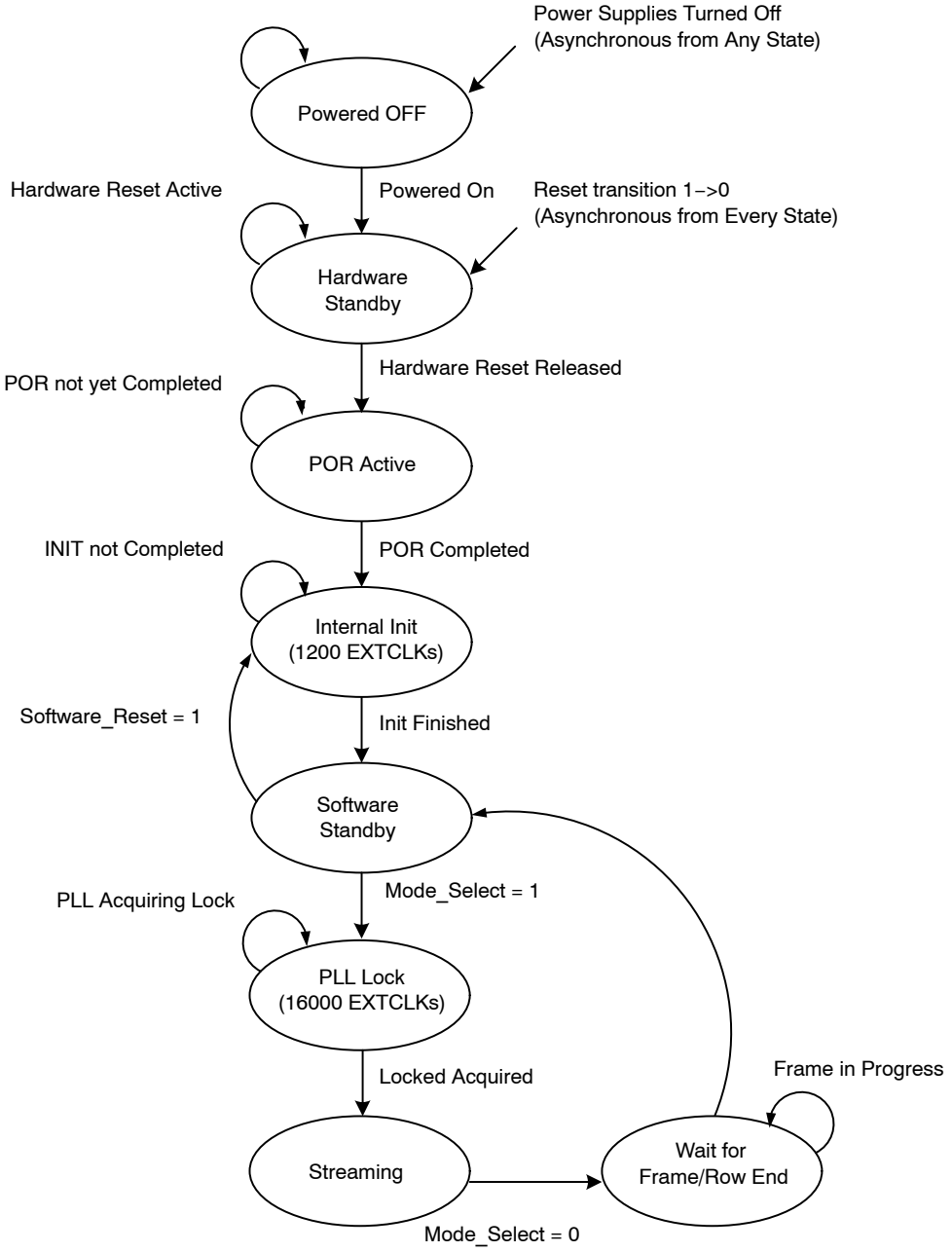


Figure 13. MT9D015 System States

Table 9. PLL IN SYSTEM STATES

State	EXTCLKs	PLL
Powered Off		
Hardware Standby		
POR Active		
Internal Initialization	1200	VCO powered down
Software Standby		
PLL Lock	16000	VCO powering up and locking, PLL output bypassed
Streaming		VCO running, PLL output active
Wait for Frame End		

1. VCO = voltage−controlled oscillator.

Power−On Reset Sequence

When power is applied to the MT9D015, it enters a low−power hardware standby state. Exit from this state is controlled by the later of two events:

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

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

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 waits for power−on reset and performs an internal initialization sequence that takes 1200 EXTCLK cycles. After this time, it enters a low−power soft standby state. While the initialization sequence is in progress, the MT9D015 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 (0x1501 if R0x0000 is read).

When the sensor leaves soft standby mode and enables the VCO, an internal delay will keep the PLL disconnected for up to 16000 EXTCLKs so that the PLL can lock.

Soft Reset Sequence

The MT9D015 can be reset under software control by writing “1” to software_reset (R0x0103). A software reset asynchronously resets the sensor, truncating any frame that is in progress. The sensor starts the internal initialization sequence, while the PLL and analog blocks are turned off. At this point, the behavior is exactly the same as for the power−on reset sequence.

Signal State During Reset

Table 10 shows the state of the signal interface during hardware standby (RESET_BAR asserted) and the default state during soft 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	Self−biased. Can be left disconnected/floating	
RESET_BAR (XSHUTDOWN)	Input	Enabled. Must be driven to a valid logic level	
SCLK	Input	Enabled. Must be pulled up or driven to a valid logic level	
SDATA	I/O	Enabled as an input. Must be pulled up or driven to a valid logic level	
DATA_P	Output	CCP2: High−Z MIPI: Ultra Low−Power State (ULPS), represented as an LP−00 state on the output (both wires at 0V)	
DATA_N	Output		
CLK_P	Output		
CLK_N	Output		

Table 10. SIGNAL STATE DURING RESET (continued)

Pad Name	Pad Type	Hardware Standby	Software Standby
GPI[3:0]	Input	Powered up. Must be connected to V_{DD} or D_{GND}	
TEST	Input	Enabled. Must be driven to a logic 0 for a serial CCP2−configured sensor, or 1 for a serial MIPI−configured sensor	

General Purpose Inputs

The MT9D015 provides four general purpose inputs. After reset, the input pads associated with these signals are powered on by default, requiring the pads to be tied to a defined logic level.

The general purpose inputs are disabled by setting reset_register[8] (R0x301A–B). Once disabled, the inputs can be left floating. The state of the general purpose inputs can be read through gpi_status[3:0] (R0x3026–7).

Streaming/Standby Control

The MT9D015 can be switched between its soft standby and streaming states under register control, as shown in Table 11. The state diagram for transitions between soft standby and streaming states is shown in Figure 13.

Table 11. STREAMING/STANDBY

Streaming R0x301A–B[2] or R0x0100[0]	Description
0	Soft standby
1	Streaming

CLOCKING

The MT9D015 contains a PLL for timing generation and control. The PLL contains a prescaler to divide the input clock applied on EXTCLK, a VCO to multiply the prescaler output, and a set of dividers to generate the output clocks.

Profile 0 Behavior

ON Semiconductor SMIA sensors are profile 2 sensors and have separate video timing and output clock domains.

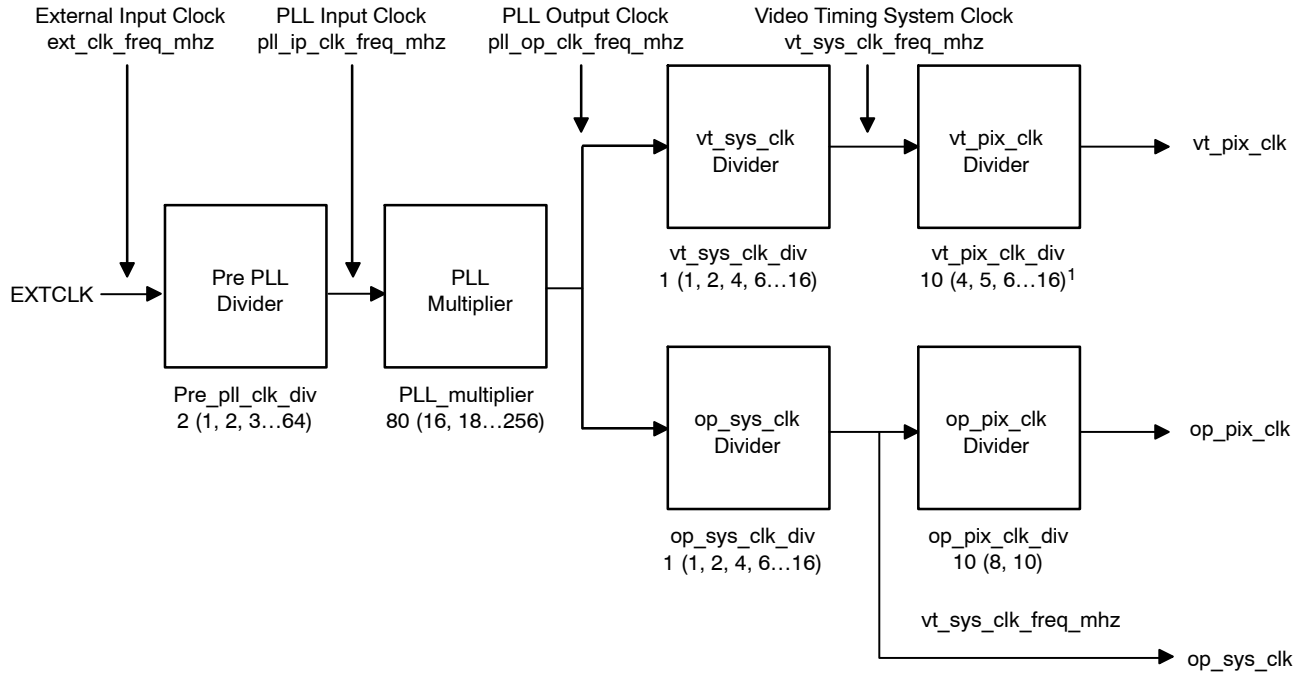
If the video timing and output clock domains are programmed with the same dividers, the part will operate in profile 0 mode as indicated by R0x306E–F[7]. For example, if Equation 1 is true, then the PLL will have profile 0 behavior:

$$\text{Profile0\_behavior} = (\text{vt\_sys\_clk\_div} \equiv \text{op\_sys\_clk\_div}) \ \& \ (\text{vt\_pix\_clk\_div} \equiv \text{op\_pix\_clk\_div}) \quad \text{(eq. 1)}$$

When the PLL is programmed to be in profile 0 behavior then the output clock domain is connected internally to the video timing domain thus ensuring that the sensor behave as an profile 0 sensor with respect to the PLL.

In profile 0 mode the number of bits between one sync code and the subsequent one are guaranteed to be equal.

Note that legacy sensors used the profile bit in the datapath_select register R0x306E[7] to set this behavior. The new behavior of profile 0 mode is equivalent with the old one once it is set by the host system.



NOTES:
1. The combinations vt_sys_clk_div = 1 and vt_pix_clk_div = (4,5, 6,... 16) are also supported even though the capability register does not advertise this.
2. The pll_multiplier only accepts even values when ccp2_class is set to data/clock signalling. Odd values will be rounded down to the first even number by setting LSB to "0."
3. The default value for vt_sys_clk_div is outside the range of legal values defined by the capability registers. This results in correct behavior for the cases listed in Note 1. The default setting is selected to ensure profile 0 behavior as default with the highest possible frame rate.

Figure 14. MT9D015 SMIA Profile 1/2 Clocking Structure

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

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

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

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

PLL input clock frequency range, after the pre−PLL divider stage, is 2.0–24 MHz.

The usage of the output clocks is:

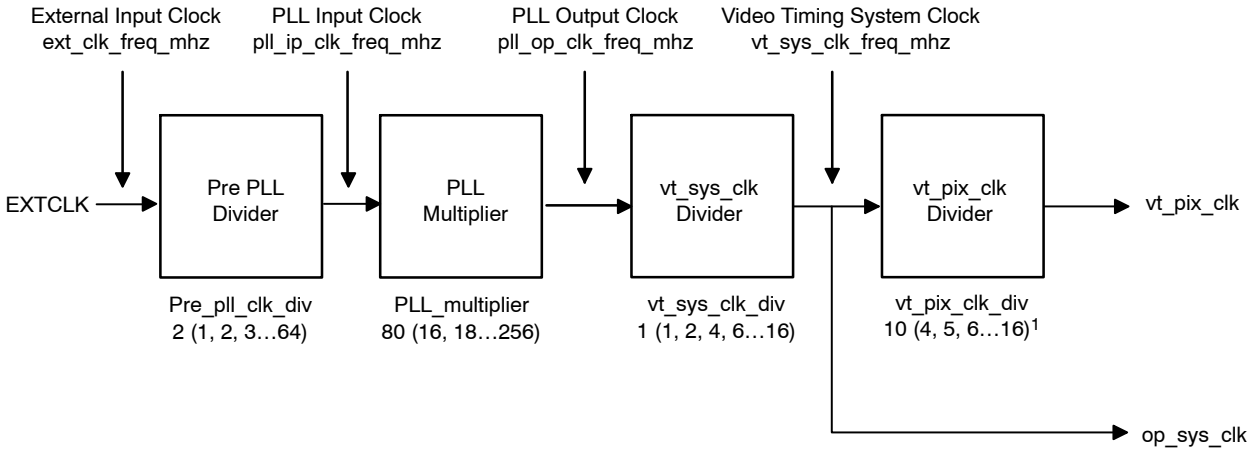
- vt_pix_clk is used by the sensor core to control the timing of the pixel array. The sensor core produces one 10−bit pixel each vt_pix_clk period. The line length (line_length_pck) and fine integration time (fine_integration_time) are controlled in increments of the vt_pix_clk period
- op_pix_clk is used to load parallel pixel data from the output FIFO (see Figure 26) to the CCP2 serializer. The output FIFO generates one pixel each op_pix_clk period. The pixel is either 8−bit or 10−bit depending upon the output data format, controlled by R0x0112–3 (ccp_data_format)
- op_sys_clk is used to generate the serial data stream on the CCP2 output. The relationship between this clock frequency and the op_pix_clk frequency is dependent upon the output data format

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

$$vt\_pix\_clk\_freq\_mhz = \frac{ext\_clk\_freq\_mhz \times pll\_multiplier}{pre\_pll\_clk\_div \times vt\_sys\_clk\_div \times vt\_pix\_clk\_div} \quad \text{(eq. 2)}$$

$$op\_pix\_clk\_freq\_mhz = \frac{ext\_clk\_freq\_mhz \times pll\_multiplier}{pre\_pll\_clk\_div \times op\_sys\_clk\_div \times op\_pix\_clk\_div} \quad \text{(eq. 3)}$$

$$op\_sys\_clk\_freq\_mhz = \frac{ext\_clk\_freq\_mhz \times pll\_multiplier}{pre\_pll\_clk\_div \times op\_sys\_clk\_div} \quad \text{(eq. 4)}$$



NOTES: 1. The legal range yielding profile 0 behavior is limited to the PLL values where the “vt domain” equals the “op domain”. The vt_sys_clk_div values in the parentheses are therefore the legal values for both vt_sys_clk_div and op_sys_clk_div, and the vt_pix_clk_div values in the parentheses are legal values for both vt_pix_clk_div and op_pix_clk_div.

2. The default value for vt_sys_clk_div is outside the range of legal values defined by the capability registers. This will result in correct behavior for the cases listed in Note 1. The default setting is selected to ensure profile 0 behavior as default with the highest possible frame rate.

Figure 15. MT9D015 SMIA Profile 0 Clocking Structure

When the video timing domain and the output timing domain have the same divider values, the PLL is equivalent to the SMIA profile 0 clocking structure. This is achieved by driving the op_sys_clk domain from the vt_sys_clk output and by driving the op_pix_clk domain from the vt_pix_clk output.

Programming the PLL Divisors

The PLL divisors should be programmed while the MT9D015 is in the soft standby state. After programming the divisors, it is necessary to wait for the VCO lock time before enabling the PLL. The PLL is enabled by entering the streaming state.

An external timer needs to delay the entering of streaming mode by 1ms so that the PLL can lock.

The effect of programming the PLL divisors while the MT9D015 is in the streaming state is undefined.

Influence of ccp_data_format

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

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

FEATURES

Lens Shading Correction (LC)

Lenses tend to produce images whose brightness is significantly attenuated near the edges. There are also other factors causing fixed pattern signal gradients in images captured by image sensors. The cumulative result of all these factors is known as image shading. The MT9D015 has an embedded shading correction module that can be programmed to counter the shading effects on each individual Red, GreenB, GreenR, and Blue color signal.

The Correction Function

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

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

$$Pcorrected(row, col) = Psensor(row, col) \times f(row, col) \quad \text{(eq. 5)}$$

where *P* are the pixel values and *f* is the color−dependent correction function 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 correct sequence to write to the LC registers is as follows:

1. Set R0x3780–1 = 0x0000
2. Load LC coefficients
3. Set R0x3780–1 = 0x8000

To read the LC coefficients, disable LC (set R0x3780 – 1 = 0x0000) before reading the register values.

One−Time Programmable Memory (OTPM)

The MT9D015 has 2624 bits of OTP memory that can be used during module manufacturing to store specific module information. This feature enables system integrators and module manufacturers to label and distinguish various module types based on lenses, IR−cut filters, or other properties. MT9D015 can support one set of LSC to save in OTPM. For OTPM programming details, please refer TN−09−248.

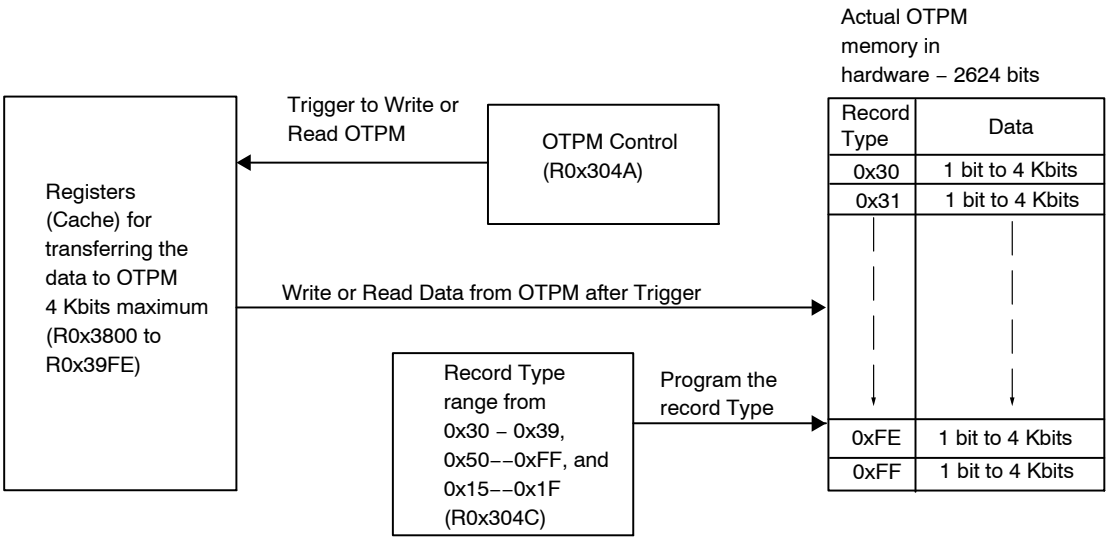


Figure 16. OTPM Block Diagram

Image Acquisition Modes

The MT9D015 supports ERS mode. When the MT9D015 is streaming, it generates frames at a fixed rate, and each frame is integrated (exposed) using the ERS. 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 MT9D015 switches cleanly from the old integration time to the new while only generating frames with uniform integration.

Window Control

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

Pixel Border

The default settings of the sensor provide a 1600 (H) × 1200 (V) image. A border of up to 4 pixels on each edge can be enabled by reprogramming the x_addr_start, y_addr_start, x_addr_end, y_addr_end, and x_output_size and y_output_size registers accordingly.

Full Resolution Frame Structure With Embedded Data

Figure 17 shows a full resolution frame structure example. The embedded data enable or disable is controlled by R0x3064[8], when set the bit to “1”, two lines of embedded data will output on start of image frame.

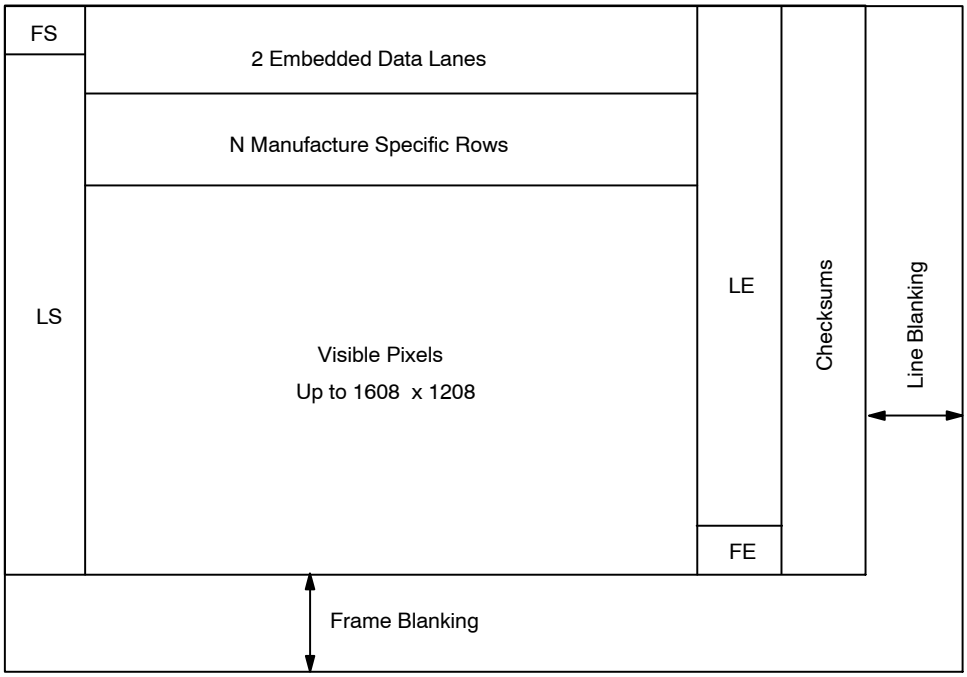


Figure 17. Full Resolution Frame Structure Example

Readout Modes

Horizontal Mirror

When the horizontal_mirror bit is set in the image_orientation register, the order of pixel readout within a row is reversed, so that readout starts from x_addr_end and ends at x_addr_start. Figure 18 shows a sequence of 6 pixels being read out with horizontal_mirror = 0 and horizontal_mirror = 1. Changing horizontal_mirror causes the Bayer order of the output image to change; the new Bayer order is reflected in the value of the pixel_order register.

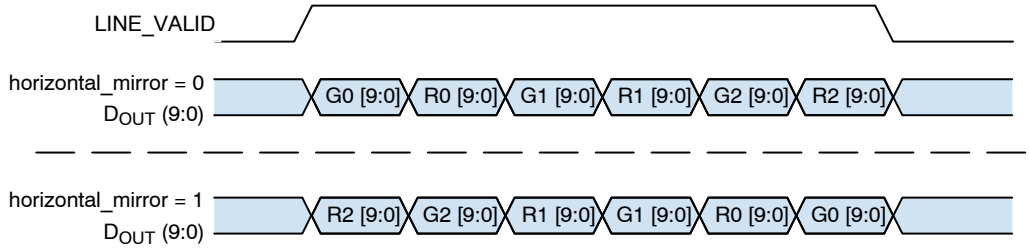


Figure 18. Effect of horizontal_mirror on Readout Order

Vertical Flip

When the vertical_flip bit is set in the image_orientation register, the order 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 19 shows a sequence of 6 rows being read out with vertical_flip = 0 and vertical_flip = 1. Changing vertical_flip causes the Bayer order of the output image to change; the new Bayer order is reflected in the value of the pixel_order register.

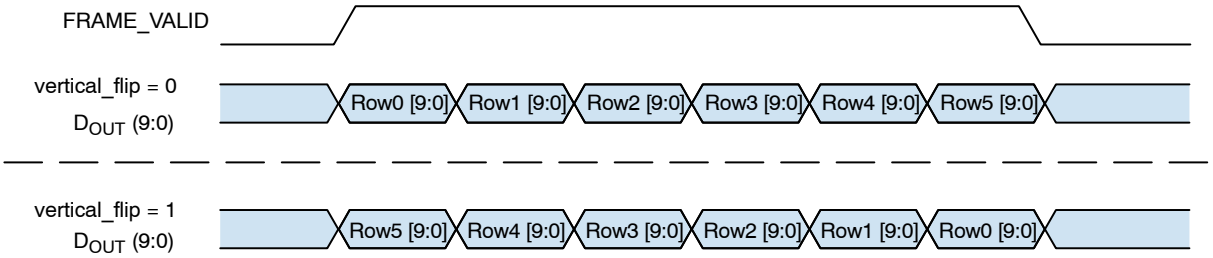


Figure 19. Effect of vertical_flip on Readout Order

Subsampling

The MT9D015 supports subsampling. Subsampling reduces the amount of data processed by the analog signal chain in the MT9D015 thereby allowing the frame rate to be increased. Subsampling is enabled by setting x_odd_inc and/or y_odd_inc. Values of 1 and 3 can be supported. Setting both of these variables to 3 reduces the amount of row and column data processed. Figure 20 shows a sequence of 8 columns being read out with x_odd_inc = 3 and y_odd_inc = 1.

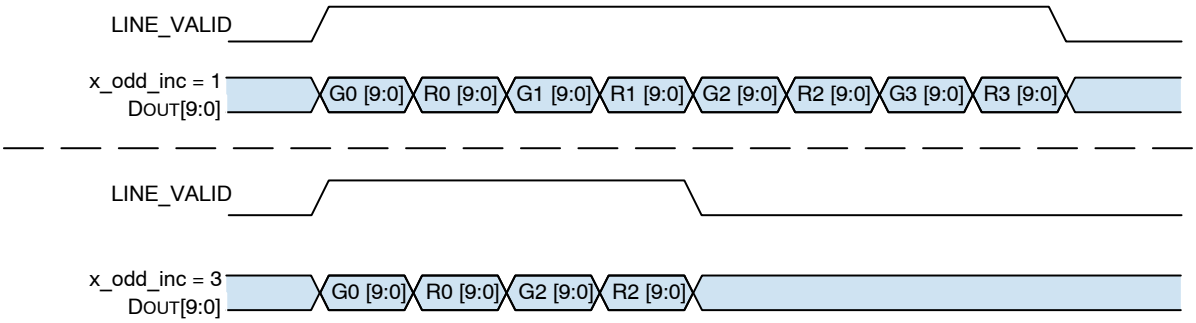


Figure 20. Effect of x_odd_inc = 3 on Readout Sequence

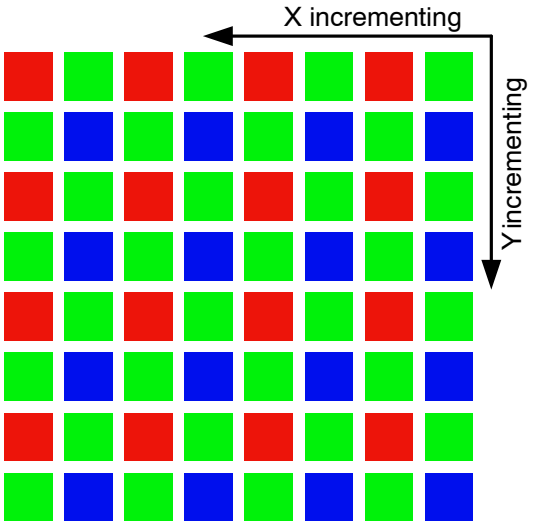


Figure 21. Pixel Readout (No Subsampling)

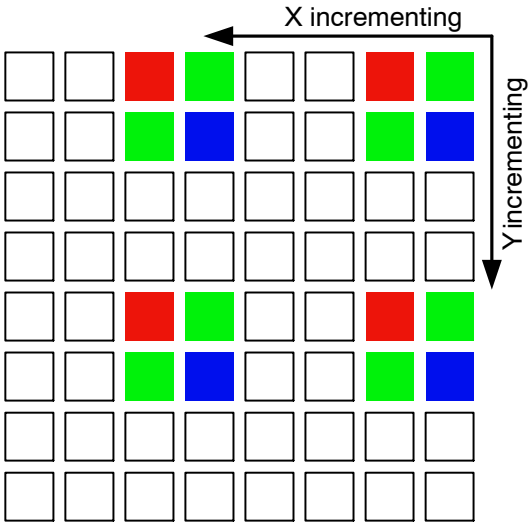


Figure 22. Pixel Readout (x_odd_inc = 3, y_odd_inc = 3)

Programming Restrictions when Subsampling

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

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

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

Example:

To achieve 1600×1200 full resolution without subsampling, the recommended register settings are:

```
[full resolution starting address with (4, 4)]
REG=0x0104, 1                   //GROUPED_PARAMETER_HOLD
REG=0x0382, 1                   //X_ODD_INC
REG=0x0386, 1                   //Y_ODD_INC
REG=0x0344, 4                   //X_ADDR_START
REG=0x0346, 4                   //Y_ADDR_START
REG=0x0348, 1603                //X_ADDR_END
REG=0x034A, 1203                //Y_ADDR_END
REG=0x034C, 1600                //X_OUTPUT_SIZE
REG=0x034E, 1200                //Y_OUTPUT_SIZE
REG=0x0104, 0                   //GROUPED_PARAMETER_HOLD
```

To achieve a 800 × 600 resolution with 1/2 subsampling, the recommended register settings are:

```
[1/2 subsampling starting address with (8, 8)]
REG=0x0104, 1                   //GROUPED_PARAMETER_HOLD
REG=0x0382, 3                   //X_ODD_INC
REG=0x0386, 3                   //Y_ODD_INC
REG=0x0344, 8                   //X_ADDR_START
REG=0x0346, 8                   //Y_ADDR_START
REG=0x0348, 1605                //X_ADDR_END
REG=0x034A, 1205                //Y_ADDR_END
REG=0x034C, 800                 // X_OUTPUT_SIZE
REG=0x034E, 600                 //Y_OUTPUT_SIZE
REG=0x0104, 0                   //GROUPED_PARAMETER_HOLD
```

Table 12 shows the row address sequencing for normal and subsampled readout. The same sequencing applies to column addresses for subsampled readout. There are two possible subsampling sequences for the rows (because the subsampling sequence only read half of the rows) depending upon the alignment of the start address.

Table 12. ROW ADDRESS SEQUENCING

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

Frame Rate Control

The formula for calculating the frame rate of the MT9D015 are shown below:

$$\text{line\_length\_pck} = \left(\frac{\text{x\_addr\_end} - \text{x\_addr\_start} + \text{x\_odd\_inc}}{\text{subsampling factor}} + \text{min\_line\_blanking\_pck}\right) \quad \text{(eq. 6)}$$

$$\text{frame\_length\_lines} = \left(\frac{\text{y\_addr\_end} - \text{y\_addr\_start} + \text{y\_odd\_inc}}{\text{subsampling factor}} + \text{min\_frame\_blanking\_lines}\right) \quad \text{(eq. 7)}$$

$$\text{frame rate[FPS]} = \frac{(\text{vt\_pixel\_clock\_mhz} \times 1 \times 10^6)}{(\text{line\_length\_pck} \times \text{frame\_length\_lines})} \quad \text{(eq. 8)}$$

NOTE: Subsampling factor = xskip or yskip
xskip = 1 if x_odd_inc = 1;
xskip = 2 if x_odd_inc = 3
yskip = 1 if y_odd_inc = 1;
yskip = 2 if y_odd_inc = 3

Minimum Row Time

The minimum row time and blanking values with default register settings are shown in Table 13.

Table 13. MINIMUM ROW TIME AND BLANKING NUMBERS

row_speed[2:0]	1	2	4
min_line_blanking_pck	0x02E1	0x01C9	0x013D
min_line_length_pck	0x03E1	0x0370	0x02E0

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:

- line_length_pck > min_line_length_pck in Table 13
- line_length_pck > (x_addr_end − x_addr_start + x_odd_inc)/((1 + x_odd_inc)/2) + min_line_blanking_pck in Table 13

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

- line_length_pck > min_line_length_pck in Table 13
- line_length_pck > (x_addr_end − x_addr_start + x_odd_inc)/((1 + x_odd_inc)/2) + min_line_blanking_pck in Table 13
- The row time must allow the FIFO to output all data during each row. That is, line_length_pck > (x_output_size / 2) × (vt_pix_clk_freq)/(op_pix_clk_freq)

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 14. MINIMUM FRAME TIME AND BLANKING NUMBERS

min_frame_blanking_lines	0x0093
min_frame_length_lines	0x054B

Integration Time

The integration (exposure) time of the MT9D015 is controlled by the fine_integration_time and coarse_integration_time registers.

The limits for the fine integration time are defined by:

$$\text{fine\_integration\_time\_min} <= \text{fine\_integration\_time} <= (\text{line\_length\_pck} - \text{fine\_integration\_time\_max\_margin}) \quad \text{(eq. 9)}$$

The limits for the coarse integration time are defined by:

$$\text{coarse\_integration\_time\_min} <= \text{coarse\_integration\_t} \quad \text{(eq. 10)}$$

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

The actual integration time is given by:

$$\text{integration\_time[sec]} = \frac{((\text{coarse\_integration\_time} \times \text{line\_length\_pck}) + \text{fine\_integration\_time})}{(\text{vt\_pix\_clk\_freq\_mhz} \times 1 \times 10^6)} \quad \text{(eq. 11)}$$

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

$$\text{Maximum integration time [sec]} = \frac{(((\text{frame\_length\_lines} - 1) \times \text{line\_length\_pck}) + (\text{line\_length\_pck} - \text{fine\_integration\_time\_max\_margin}}{(\text{vt\_pix\_clk\_freq\_mhz} \times 1 \times 10^6)}$$

$$= \frac{(((0x0503) \times 0x0938) + 0x7A3)}{(64\text{MHz} \times 1 \times 10^6)} = 47.34 \text{ ms} \quad \text{(eq. 12)}$$

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

Fine Integration Time Limits

The limits for the fine_integration_time can be found from fine_integration_time_min and fine_integration_time_max_margin. Values for different mode combinations are shown in Table 15.

Table 15. FINE_INTEGRATION_TIME LIMITS

row_speed[2:0]	1	2	4
fine_integration_time_min	0x01E5	0x0104	0x0052
fine_integration_time_max_margin	0x0191	0x00B7	0x008B

Analog Gain

The MT9D015 provides two mechanisms for setting the analog gain. The first uses the SMIA gain model; the second uses the traditional ON Semiconductor gain model. The following sections describe both models, the mapping between the models, and the operation of the per−color and global gain control.

Using Per−color or Global Gain Control

The read−only analogue_gain_capability register returns a value of “1,” indicating that the MT9D015 provides per−color gain control. However, the MT9D015 also provides the option of global gain control. Per−color and global gain control can be used interchangeably. A write to a global gain register is aliased as a write of the same data to the four associated color−dependent gain registers. A read from a global gain register is aliased to a read of the associated greenB/greenR gain register.

The read/write gain mode register required by SMIA has no defined function in the SMIA specification. In the MT9D015 this register has no side effects on the operation of the gain; per−color and global gain control can be used interchangeably regardless of the state of the gain_mode register.

SMIA Gain Model

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

- analogue_gain_code_global
- analogue_gain_code_greenR
- analogue_gain_code_red
- analogue_gain_code_blue
- analogue_gain_code_greenB

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

$$\text{gain} = \frac{\text{analogue\_gain\_m0} \times \text{analogue\_gain\_code}}{\text{analogue\_gain\_c1}} = \frac{\text{analog\_gain\_code\_} < \text{color} >}{8} \quad \text{(eq. 13)}$$

ON Semiconductor Gain Model

The ON Semiconductor gain model uses the following registers to set the analog gain:

- global_gain
- greenr_gain
- red_gain
- blue_gain
- greenb_gain

This gain model maps directly to the control settings applied to the gain stages of the analog signal chain. This provides a 7−bit gain stage and a number of 2X gain stages. As a result, the step size varies depending upon whether the 2X gain stages are enabled. The analog gain is given by:

$$\text{gain} = (<\text{color}>\_\text{gain}[8] + 1) \times (<\text{color}>\_\text{gain}[7] + 1) \times \frac{<\text{color}>\_\text{gain}[6:0]}{32} \quad \text{(eq. 14)}$$

As a result of the 2X gain stage, many of the possible gain settings can be achieved in two different ways. In all cases, the preferred setting is the setting that uses <*color*>*_gain[7]* first, <*color*>*_gain[6:0]*, and then <*color*>*_gain[8]* to apply the desired gain range,, because this will result in lower noise.

Gain Code Mapping

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

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

When the ON Semiconductor gain model is in use and values have been written to the gain_<color> registers, data read from the associated analogue_gain_code_<color> register is undefined. The reason for this is that many of the gain codes available in the ON Semiconductor gain model have no corresponding value in the SMIA gain model.

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

Minimum Gain and Gain Table

To make the sensor reach saturation status in high light condition, the gain value applied to this part must larger than 1.375X. In other words, the 1.375X is the minimum gain.

Table 16. GAIN TABLE

SMIA Gain (R0x0204)	Global Gain (R0x305E)	Gain
0x0008	0x1020	1
0x0009	0x1024	1.125
0x000A	0x1028	1.25
0x000B	0x102C	1.375
0x000C	0x1030	1.5
0x000D	0x1034	1.625
0x000E	0x1038	1.75
0x000F	0x103C	1.875
0x0010	0x10A0	2
0x0011	0x10A2	2.125
0x0012	0x10A4	2.25
0x0013	0x10A6	2.375
0x0014	0x10A8	2.5
0x0015	0x10AA	2.625
0x0016	0x10AC	2.75
0x0017	0x10AE	2.875
0x0018	0x10B0	3
0x0019	0x10B2	3.125
0x001A	0x10B4	3.25

Table 16. GAIN TABLE (continued)

SMIA Gain (R0x0204)	Global Gain (R0x305E)	Gain
0x001B	0x10B6	3.375
0x001C	0x10B8	3.5
0x001D	0x10BA	3.625
0x001E	0x10BC	3.75
0x001F	0x10BE	3.875
0x0020	0x10C0	4
0x0021	0x10C2	4.125
0x0022	0x10C4	4.25
0x0023	0x10C6	4.375
0x0024	0x10C8	4.5
0x0025	0x10CA	4.625
0x0026	0x10CC	4.75
0x0027	0x10CE	4.875
0x0028	0x10D0	5
0x0029	0x10D2	5.125
0x002A	0x10D4	5.25
0x002B	0x10D6	5.375
0x002C	0x10D8	5.5
0x002D	0x10DA	5.625
0x002E	0x10DC	5.75
0x002F	0x10DE	5.875
0x0030	0x10E0	6
0x0031	0x10E2	6.125
0x0032	0x10E4	6.25
0x0033	0x10E6	6.375
0x0034	0x10E8	6.5
0x0035	0x10EA	6.625
0x0036	0x10EC	6.75
0x0037	0x10EE	6.875
0x0038	0x10F0	7
0x0039	0x10F2	7.125
0x003A	0x10F4	7.25
0x003B	0x10F6	7.375
0x003C	0x10F8	7.5
0x003D	0x10FA	7.625
0x003E	0x10FC	7.75
0x003F	0x10FE	7.875
0x0040	0x11C0	8
0x0041	0x11C1	8.125
0x0042	0x11C2	8.25
0x0043	0x11C3	8.375
0x0044	0x11C4	8.5

Table 16. GAIN TABLE (continued)

SMIA Gain (R0x0204)	Global Gain (R0x305E)	Gain
0x0045	0x11C5	8.625
0x0046	0x11C6	8.75
0x0047	0x11C7	8.875
0x0048	0x11C8	9
0x0049	0x11C9	9.125
0x004A	0x11CA	9.25
0x004B	0x11CB	9.375
0x004C	0x11CC	9.5
0x004D	0x11CD	9.625
0x004E	0x11CE	9.75
0x004F	0x11CF	9.875
0x0050	0x11D0	10
0x0051	0x11D1	10.125
0x0052	0x11D2	10.25
0x0053	0x11D3	10.375
0x0054	0x11D4	10.5
0x0055	0x11D5	10.625
0x0056	0x11D6	10.75
0x0057	0x11D7	10.875
0x0058	0x11D8	11
0x0059	0x11D9	11.125
0x005A	0x11DA	11.25
0x005B	0x11DB	11.375
0x005C	0x11DC	11.5
0x005D	0x11DD	11.625
0x005E	0x11DE	11.75
0x005F	0x11DF	11.875
0x0060	0x11E0	12
0x0061	0x11E1	12.125
0x0062	0x11E2	12.25
0x0063	0x11E3	12.375
0x0064	0x11E4	12.5
0x0065	0x11E5	12.625
0x0066	0x11E6	12.75
0x0067	0x11E7	12.875
0x0068	0x11E8	13
0x0069	0x11E9	13.125
0x006A	0x11EA	13.25
0x006B	0x11EB	13.375
0x006C	0x11EC	13.5
0x006D	0x11ED	13.625
0x006E	0x11EE	13.75

Table 16. GAIN TABLE (continued)

SMIA Gain (R0x0204)	Global Gain (R0x305E)	Gain
0x006F	0x11EF	13.875
0x0070	0x11F0	14
0x0071	0x11F1	14.125
0x0072	0x11F2	14.25
0x0073	0x11F3	14.375
0x0074	0x11F4	14.5
0x0075	0x11F5	14.625
0x0076	0x11F6	14.75
0x0077	0x11F7	14.875
0x0078	0x11F8	15
0x0079	0x11F9	15.125
0x007A	0x11FA	15.25
0x007B	0x11FB	15.375
0x007C	0x11FC	15.5
0x007D	0x11FD	15.625
0x007E	0x11FE	15.75
0x007F	0x11FF	15.875

1. 1−1.25 gains have been grayed out. Customers should not use less than x1.375 gain to avoid un−saturation issue.
2. The gain range 1−7.875 used analog gain only. The range 8 − 15.875 used 2X digital gain. When R0x0204[6]= R0x305E[8]=1, 2X digital gain is enabled.

SENSOR CORE DIGITAL DATA PATH

Test Patterns

The MT9D015 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 17.

Table 17. TEST PATTERNS

test_pattern_mode	Description
0	Normal operation: no test pattern
1	Solid color
2	100% color bars
3	Fade−to−gray color bars
4	PN9 link integrity pattern

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

For all of the test patterns, the MT9D015 registers must be set appropriately to control the frame rate and output timing. These include:

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

The MT9D015 will disable digital corrections automatically when test patterns are activated. The test cursor is now added to the end of the data path.

Solid Color Test Pattern

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

100 Percent Color Bars Test Pattern

In this test pattern, shown in Figure 23, all pixel data is replaced by a Bayer version of an 8−color, color−bar chart (white, yellow, cyan, green, magenta, red, blue, and black). Each bar is 200 pixels wide and occupies the full height of the output image. Each color component of each bar is set to either “0” (fully off) or 0x3FF (fully on for 10−bit data). The pattern repeats after $8 \times 200 = 1600$ pixels. The image size is set by x_addr_start, x_addr_end, y_addr_start, y_addr_end and may be affected by the setting of x_output_size, y_output_size. The color−bar pattern starts at the column identified by x_addr_start. The number of colors that are visible in the output is dependent upon x_addr_end – x_addr_start and the setting of x_output_size. The width of each color−bar is fixed at 200 pixels.

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

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

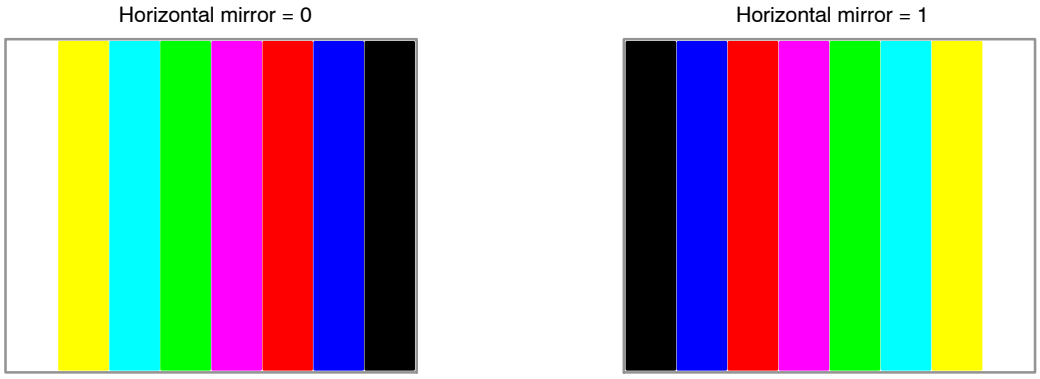


Figure 23. 100 Percent Color Bars Test Pattern

Fade−to−Gray Color Bars Test Pattern

In this test pattern, shown in Figure 24, all pixel data is replaced by a Bayer version of an 8−color, color−bar chart (white, yellow, cyan, green, magenta, red, blue, and black). Each bar is 200 pixels wide and occupies 1024 rows of the output image. Each color bar fades vertically from full intensity at the top of the image to 50 percent intensity (mid−gray) on the 1024th row. Each color bar is divided into a left and a right half, in which the left half fades smoothly and the right half fades in quantized steps every 8 pixels for a given color. Due to the Bayer pattern of the colors this means that the level changes every 16 rows. The pattern repeats horizontally after 8 × 200 = 1600 pixels and vertically after 1024 rows (using 10−bit data, the fade−to−gray pattern goes from 100 to 50 percent or from 0 to 50 percent for each color component, so only half of the 2^{10} states of the 10−bit data are used. However, to get all of the gray levels, each state must be held for two rows, hence the vertical size of $2^{10} / 2 \times 2 = 1024$). The image size is set by x_addr_start, x_addr_end, y_addr_start, and y_addr_end and may be affected by the setting of x_output_size and y_output_size. The color−bar pattern starts at the column identified by x_addr_start. The number of colors that are visible in the output is dependent upon x_addr_end − x_addr_start and the setting of x_output_size. The width of each color−bar is fixed at 200 pixels.

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

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

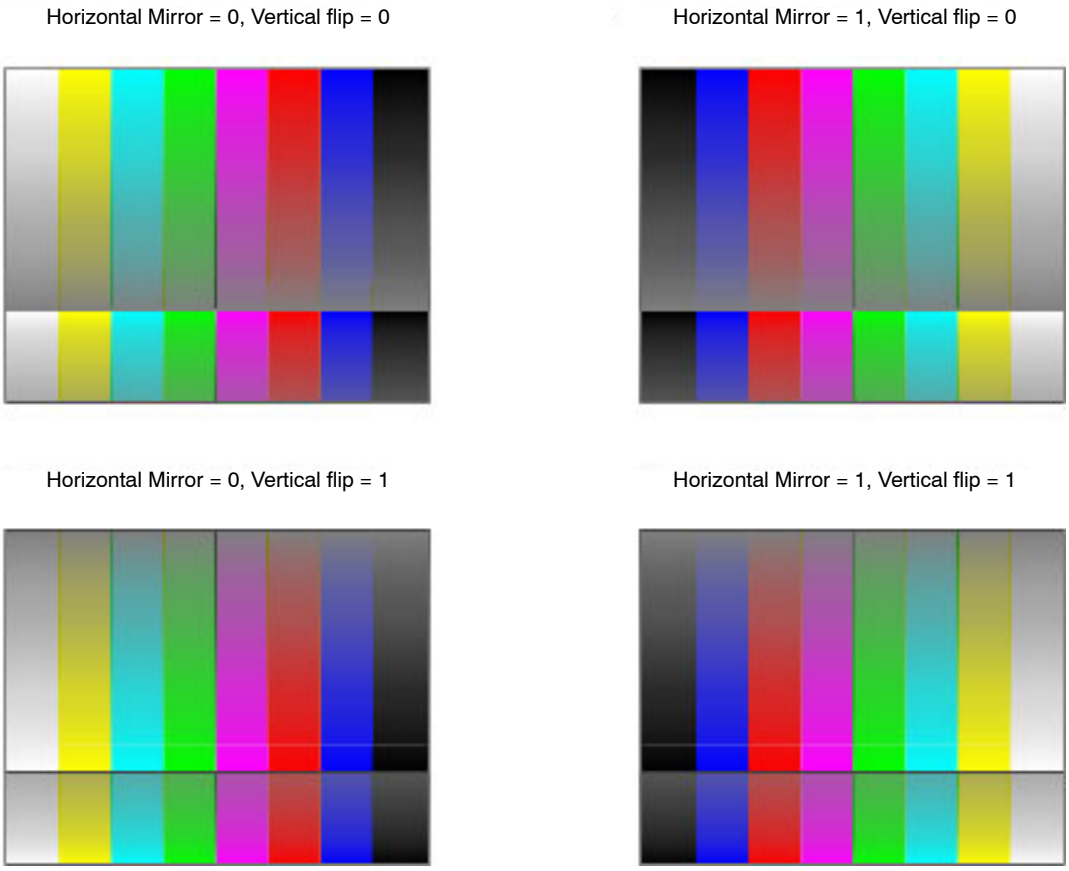


Figure 24. Fade−to−Gray Color Bars Test Pattern

PN9 Link Integrity Pattern

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

When this test pattern is enabled:

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

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

Test Cursors

The MT9D015 supports one horizontal and one vertical cursor, allowing a “cross hair” to be superimposed on the image or on test patterns 1–3.

The position and width of each cursor are programmable in R0x31E8–0x31EE. Each cursor can be inhibited by setting its width to “0.”

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

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

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

The effect of enabling the test cursors when the image_orientation register is non−zero is not defined by the SMIA specification. The behavior of the MT9D015 is shown in Figure 25. In this figure the test cursors are shown as translucent for clarity. In practice, they are opaque (they overlay the imaged scene). The manner in which the test cursors are affected by the value of image_orientation can be understood from the following implementation details:

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

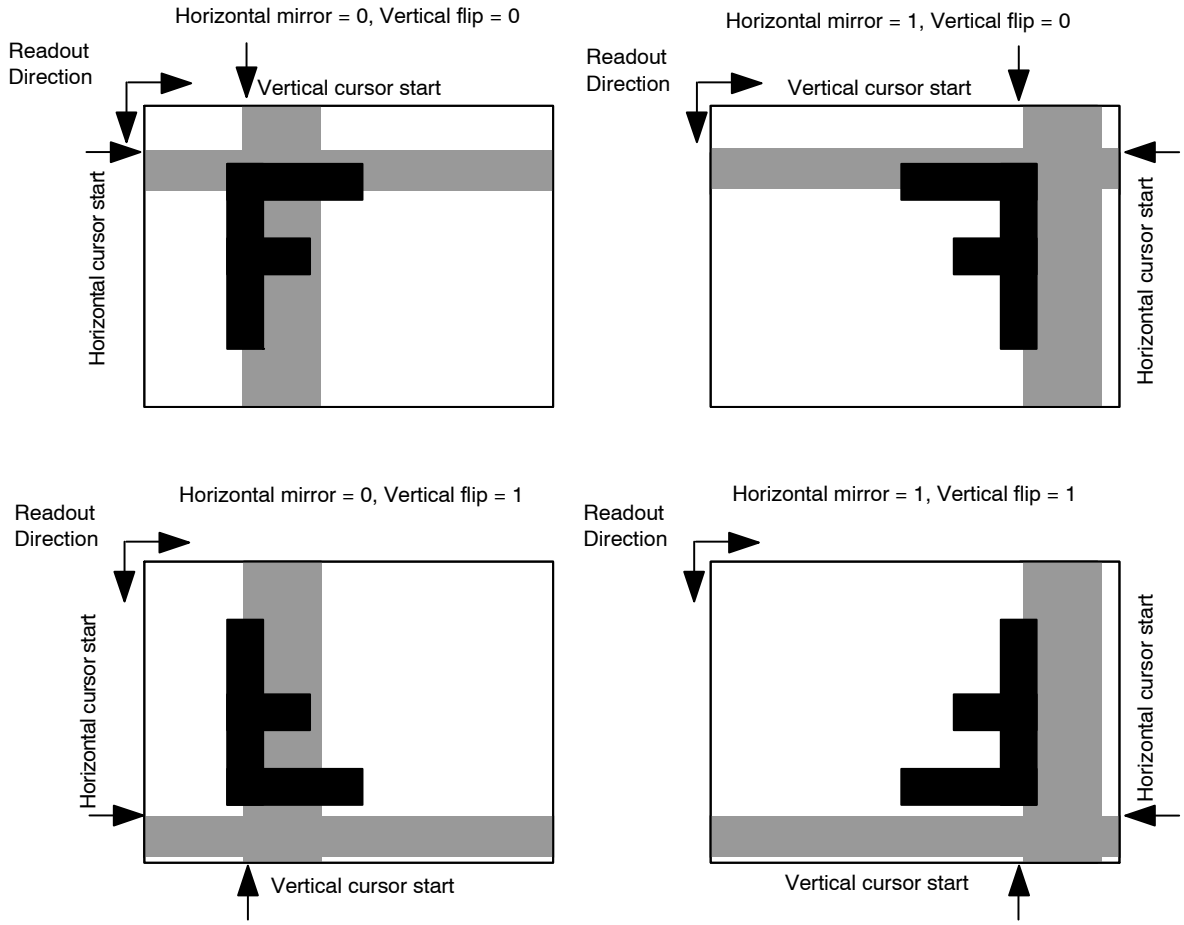


Figure 25. Test Cursor Behavior when image_orientation

Digital Gain

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

Pedestal

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

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

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

DIGITAL DATA PATH

The digital data path after the sensor core is shown in Figure 26.

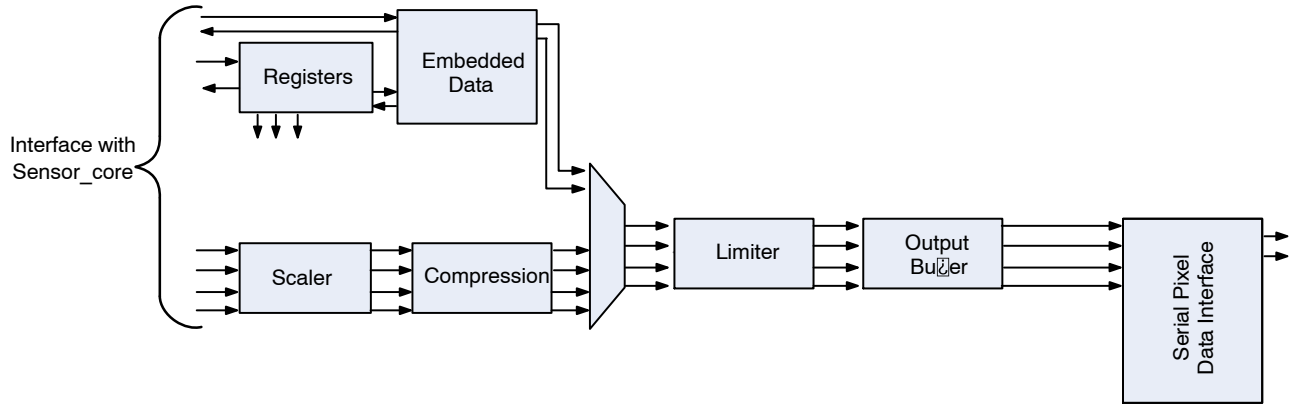


Figure 26. Data Path

TIMING SPECIFICATIONS

Power−Up Specifications

The digital and analog supply voltages can be powered up in any order. However, ON Semiconductor recommends the following power−up sequence to minimize current consumption. The power−up sequence corresponds to the requirements described in section 3.2 of the SMIA functional specification.

Power−Up Sequence

The recommended power−up sequence for the MT9D015 is shown in Figure 27 and Table 18. The available power supplies−VDD, VDD_PLL, VAA, VAA_PIX−can be turned on at any point or have the separation specified below for reducing current consumption during power−up sequence.

1. Turn on the VDD power supply
2. After 0–500 ms, turn on VDD_PLL and VAA/VAA_PIX power supplies
3. After the last power supply is stable, enable EXTCLK
4. After EXTCLK is stable, assert RESET_BAR for at least 1 ms
5. Wait 1200 EXTCLKs for internal initialization into soft standby
6. Configure PLL, output, and image settings to desired values
7. Wait 16000 EXTCLKs for the PLL to lock before streaming state is reached (enforced in hardware)
8. Set mode_select = 1 (R0x0100) to start streaming

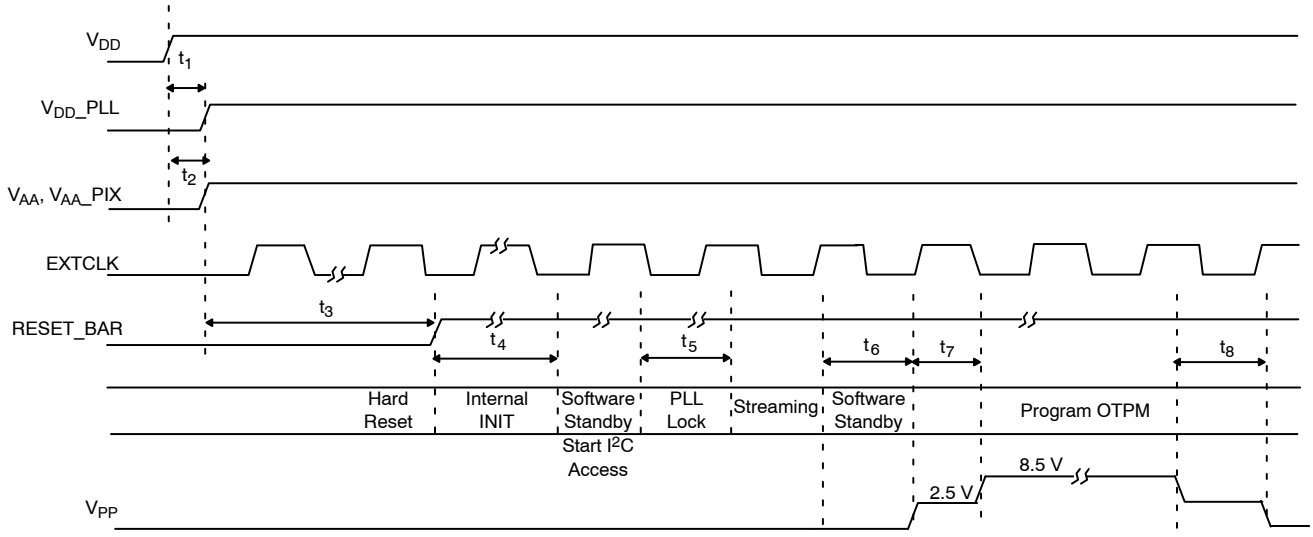


Figure 27. Power−up Sequence

Table 18. POWER−UP SEQUENCE

Definition	Symbol	Min	Typ	Max	Unit
VDD to VDD_PLL Time	t1	0	–	500	ms
VDD to VAA/VAA_PIX Time	t2	0	–	500	ms
Active Hard Reset	t3	1	–	–	ms
Internal Initialization	t4	1200	–	–	EXTCLKs
PLL Lock Time	t5	16000	–	–	EXTCLKs
Soft Standby	t6	500	–		ms
Delay 1	t7	600	–		ms
Delay 2	t8	600	–		ms

Power−Down Specification

The digital and analog supply voltages can be powered down in any order. However, ON Semiconductor recommends the following power−down sequence to minimize current consumption. The power−down sequence corresponds to the requirements described in section 3.2 of the SMIA functional specification.

Power−Down Sequence

The recommended power−down sequence for the MT9D015 is shown in Figure 28 and in Table 19. The available power supplies−VDD, VDD_PLL, VAA, VAA_PIX−can be turned off at any point or have the separation specified below for reducing current consumption during power−down sequence.

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” at least 1 ms
4. Stop EXTCLK; drive this pin to logic “0”
5. Turn off the VAA/VAA_PIX and VDD_PLL power supplies
6. After 0–500 ms, turn off VDD and power supply

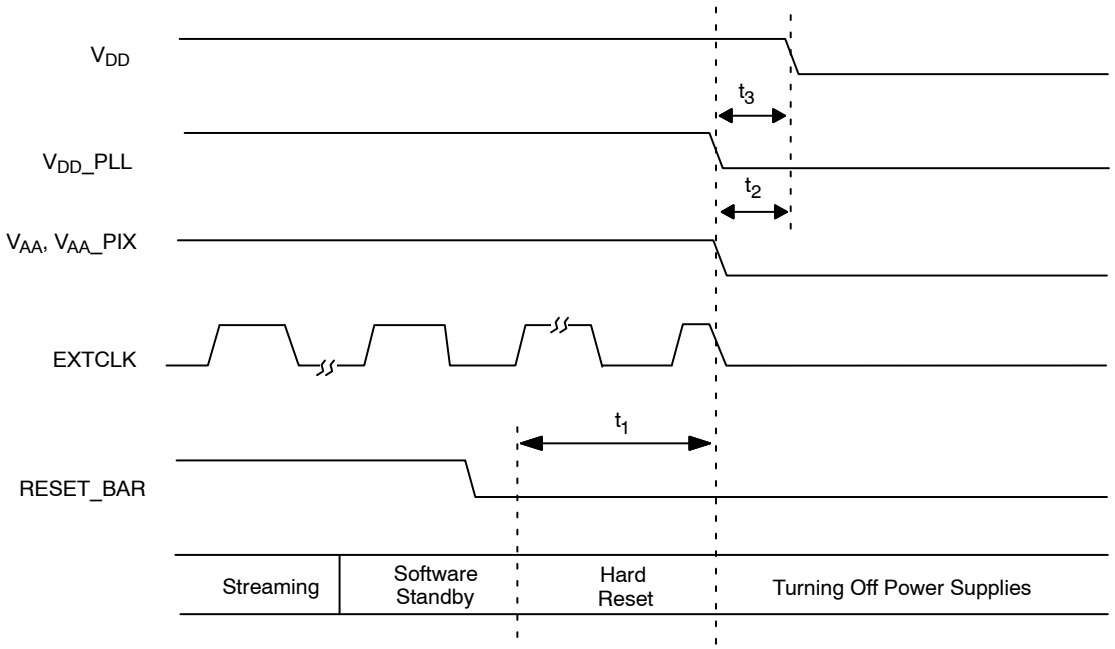


Figure 28. Power−Down Sequence

Table 19. POWER−DOWN SEQUENCE

Definition	Symbol	Min	Typ	Max	Unit
Hard reset	t1	1	–	–	ms
V_{AA}/V_{AA}_PIX to V_{DD} time	t2	0	–	500	ms
V_{DD}_PLL to V_{DD} time	t3	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. This operating mode complies with section 3.1 of the SMIA Functional Specification.

Soft Standby and Soft Reset

The MT9D015 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 optionally enabled to return all register values back to the default. The details of the sequence are shown in Figure 29.

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 list
2. Delay 1 frame time; set software_reset = 1 (R0x0103) to start the internal initialization sequence
3. After 700 EXTCLKs, the internal initialization sequence is completed and the current state returns to soft standby automatically. All registers, including software_reset, returns to their default values

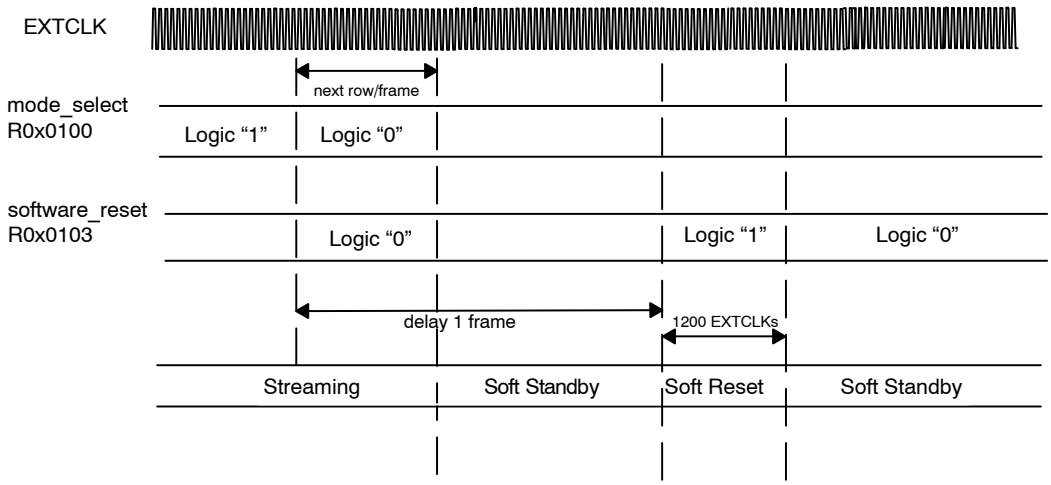


Figure 29. Soft Standby and Soft Reset

ELECTRICAL SPECIFICATIONS

EXTCLK

The electrical characteristics of the EXTCLK input are shown in Table 20. The EXTCLK input supports an AC−coupled sine−wave input clock or a DC−coupled square−wave input clock.

Table 20. ELECTRICAL CHARACTERISTICS (EXTCLK)

Definition	Condition	Symbol	Min	Typ	Max	Unit
Input Clock Frequency		f_{EXTCLK}	6	16	27	MHz
Input Clock Period		t_{EXTCLK}	166.7	62.5	37	ns
Input Clock Minimum Voltage Swing (AC Coupled Sine Wave)		V_{IN_AC}	0.5	1	1.2	V (pk−pk)
Input Clock Duty Cycle			45	50	55	%
Input Clock Jitter	cycle−to−cycle	t_{JITTER}		545	600	ps
PLL VCO Lock Time (at 16 MHz EXTCLK)		t_{LOCK}		3200		ext clk cycles
Input Pad Capacitance		C_{IN}		3.75		pF
Input HIGH Leakage Current	V_{IN} = VVD	I_{IH}	−10	−	10	mA
Input LOW Leakage Current	V_{IN} = D_{GND}	I_{IL}	−10	−	10	mA
Input HIGH Voltage(DC Coupled)		V_{IH}	0.7 y VDIG	−	2.9	V
Input LOW Voltage (DC Coupled)		V_{IL}	−0.3	−	0.3 × VDD	V

Two−Wire Serial Register Interface

Table 21 describes the I/O levels, I/O current and pin capacitance for Two−Wire Serial Interface. Table 22 describes timing specification for Two−Wire Serial Interface. Figure 30: “Definition of Timing for Two−Wire Serial Interface,” shows timing parameters definition.

Table 21. TWO−WIRE SERIAL INTERFACE ELECTRICAL CHARACTERISTICS

(V_{DD} = 1.7−1.9 V; V_{AA} = 2.4 −3.1 V; Environment Temperature = −30°C to 50°C)

Symbol	Definition	Condition	Min	Max	Unit
V_{IH}	Input High Voltage		0.7 V_{DD}	2.9	V
VIL	Input Low Voltage		−0.3	0.3 V_{DD}	V
V_{OL}	Output Low Voltage at 3 mA Sink Current		0	0.4	V
I_{OL}	Output Low Current	V_{OL} = 0.4 V	3	−	mA
Ii	Input Current Each I/O Pin	No pull−up resistor; V_{IN} = V_{DD} or D_{GND}	−10	+10	μA
Ci	Capacitance for Each I/O Pin		−	6	pF
CLOAD	Load Capacitance		N/A	N/A	pF

Table 22. TWO−WIRE SERIAL INTERFACE TIMING SPECIFICATION

(V_{DD} = 1.7−1.9 V; V_{AA} = 2.4 −3.1 V; Environment Temperature = −30°C to 50°C)

Symbol	Definition	Min	Max	Unit
f_{SCLK}	SCLK Frequency	0	400	KHz
t_{HIGH}	SCLK High Period	0.6	−	μs
t_{LOW}	SCLK Low Period	1.3	−	μs
t_{SRTS}	START Setup Time	0.6	−	μs
t_{SRTH}	START Hold Time	0.6	−	μs

Table 22. TWO−WIRE SERIAL INTERFACE TIMING SPECIFICATION (continued)
(V_{DD} = 1.7−1.9 V; V_{AA} = 2.4 −3.1 V; Environment Temperature = −30°C to 50°C)

Symbol	Definition	Min	Max	Unit
t_{SDS}	Data Setup Time	100	−	ns
t_{SDH}	Data Hold Time	0	Note 1	μs
t_{SDV}	Data Valid Time	−	0.9	μs
t_{ACV}	Data Valid Acknowledge Time	−	0.9	μs
t_{STPS}	STOP Setup Time	0.6	−	μs
t_{BUF}	Bus Free Time between STOP and START	1.3	−	μs
tr	SCLK and S_{DATA} Rise Time	−	300	ns
tf	SCLK and SDATA Fall Time	−	300	ns

1. Max t_{SDH} could be 0.9 μs but must be less than max of t_{SDV} and t_{ACV} by a transition time.

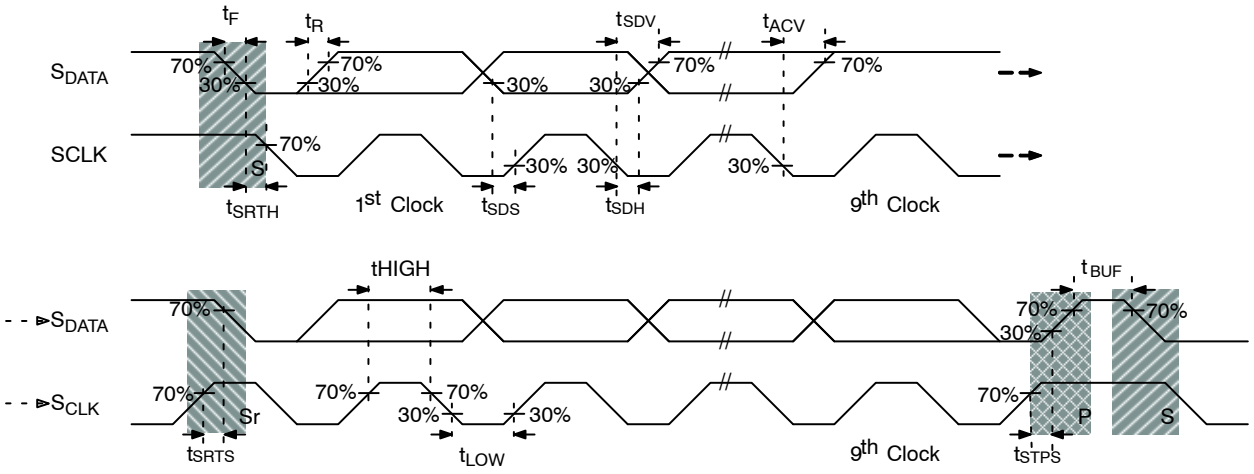


Figure 30. Definition of Timing for Two-Wire Serial Interface

Serial Pixel Data Interface

The electrical characteristics of the serial pixel data interface (CLK_P, CLK_N, DATA_P, DATA_N are shown in Table 23).

Table 23. ELECTRICAL CHARACTERISTICS (SERIAL CCP2 PIXEL DATA INTERFACE)
(V_{DD} = 1.8 V; V_{AA} = 2.8 V; V_{AA}_PIX = 2.8 V; V_{DD}_PLL = 2.8 V; Ambient Temperature)

Definition	Symbol	Min	Typ	Max	Unit
Operating Frequency		180		360	MHz
Fixed Common Mode Voltage	VCMF	0.8	0.9	1	V
Differential Voltage Swing	VOD	100	150	200	mV
Drive Current Range		0.83	1.5	2	mA
Drive Current Variation				15	%
Output Impedance		40		140	?
Output Impedance Mismatch				10	%
Clock Duty Cycle at 416 MHz		45	50	55	%
Rise Time (20–80%)		300		400	ps
Fall Time (20–80%)		300		400	ps

Table 23. ELECTRICAL CHARACTERISTICS (SERIAL CCP2 PIXEL DATA INTERFACE) (continued)
(V_{DD} = 1.8 V; V_{AA} = 2.8 V; V_{AA}_PIX = 2.8 V; V_{DD}_PLL = 2.8 V; Ambient Temperature)

Definition	Symbol	Min	Typ	Max	Unit
Differential Skew				500	ps
Channel−to−channel Slew				200	ps
Maximum Data Rate Data/Strobe Mode Data/Clock Mode				 640 208	Mb/s
Power Supply Rejection Ratio (PSRR) 0−100 MHz		30			dB
Power Supply Rejection Ratio (PSRR) 100−1000 MHz		10			dB

Table 24. ELECTRICAL CHARACTERISTICS (SERIAL MIPI PIXEL DATA INTERFACE)
(V_{DD} = 1.8 V; V_{AA} = 2.8 V; V_{AA}_PIX = 2.8 V; V_{DD}_PLL = 2.8 V; Ambient Temperature)

Definition	Symbol	Min	Typ	Max	Unit
High Speed Transmit Differential Voltage	VOD	140		270	mV
High Speed Transmit Static Common−mode Voltage	VCMTX	150		250	mV
VOD Mismatch when Output is Differential−1 or Differential−0	dVOD			<14	mV
High Speed Output High Voltage Mismatch	dVCMTX			16	mV
Single Ended Output Impedance	ZOS	40		64	Ω
Single Ended Output Impedance Mismatch	dZOS			10	%
20−80% Rise Time	tR			250	ps
20−80% Fall Time	tF			250	ps
Output LOW Level	VOL			50	mV
Output HIGH Level	VOH			1.3	V
Output Impedance of Low Power Parameter	ZOLP			110	Ω
15−85% Rise Time	TRLP			25	ns
15−85% Fall Time	TFLP			25	ns
Slew Rate (CLOAD = 5−20pF)	dv/dtsr			235	mV/ns
Slew Rate (CLOAD = 20−70pF)	dv/dstr			200	mV/ns
Power Supply Rejection Ratio (PSRR) 0−100 MHz		30			dB
Power Supply Rejection Ratio (PSRR) 100−1000 MHz		10			dB

Control Interface

The electrical characteristics of the control interface (RESET_BAR, TEST, GPI0, GPI1, GPI2, and GPI3) are shown in Table 25.

Table 25. ELECTRICAL CHARACTERISTICS

Definition	Condition	Symbol	Min	Typ	Max	Unit
Input HIGH Voltage		V_{IH}	0.7 x V_{DD}		2.9	V
Input LOW Voltage		V_{IL}	−0.3		0.3 x V_{DD}	V
Input Leakage Current	No pull−up resistor; V_{IN} = V_{DD} or D_{GND}	I_{IN}			10	μA
Input Pad Capacitance		C_{IN}		6.5		pF

Power−On Reset

Table 26. POWER−ON RESET CHARACTERISTICS

Definition	Condition	Symbol	Min	Typ	Max	Unit
V_{DD} rising, crossing V_{TRIG_RISING}; Internal reset being released		t_1	7	10	15	μs
V_{DD} falling, crossing $V_{TRIG\_FALLING}$; Internal reset active		t_2		0.5	1	μs
Minimum V_{DD} spike width below $V_{TRIG\_FALLING}$; considered to be a reset when POR cell output is HIGH		t_3		0.5		μs
Minimum V_{DD} spike width below $V_{TRIG\_FALLING}$; considered to be a reset when POR cell output is LOW		t_4		1		μs
Minimum V_{DD} spike width above V_{TRIG_RISING}; considered to be a stable supply when POR cell output is LOW	While the POR cell output is LOW, all V_{DD} spikes above V_{TRIG_RISING} less than t_5 must be ignored	t_5		50		ns
V_{DD} rising trigger voltage		V_{TRIG_RISING}	1.15		1.55	V
V_{DD} falling trigger voltage		$V_{TRIG\_FALLING}$	1		1.45	V

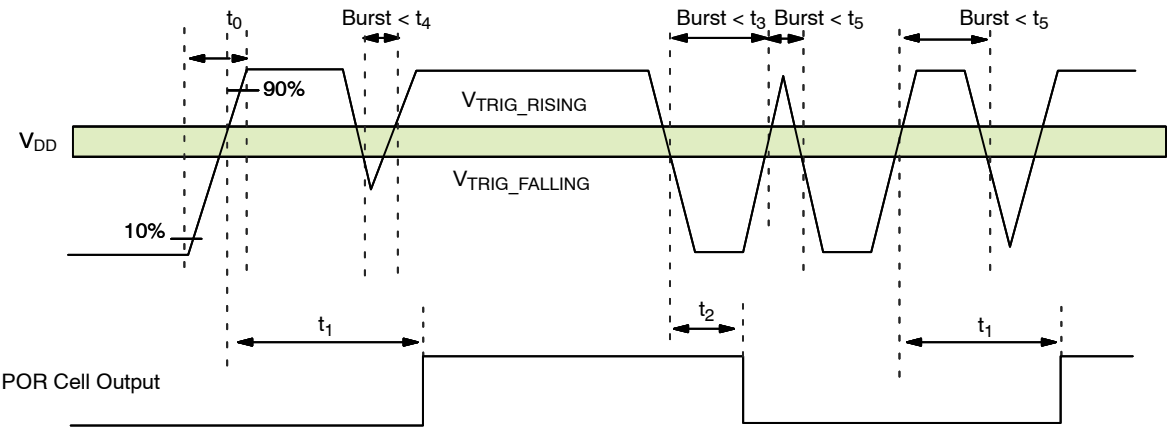


Figure 31. Internal Power−On Reset

Operating Voltages

V_{AA} and V_{AA}_PIX must be at the same potential for correct operation of the MT9D015.

Table 27. DC Electrical Definitions and Characteristics

(Setup Conditions: T_J = 25°C; Dark Lighting Conditions)

Definition	Condition	Symbol	Min	Typ	Max	Unit
Core Digital Voltage		V_{DD}	1.7	1.8	1.9	V
Analog Voltage		V_{AA}	2.4	2.8	2.9	V
Pixel Supply Voltage		V_{AA_PIX}	2.4	2.8	2.9	V
PLL Supply Voltage		V_{DD_PLL}	2.4	2.8	2.9	V
Digital Operating Current	1600x1200 at 30 fps	IDIG	10	50	71	mA
Analog Operating Current		IANA	25	65	80	mA
Digital Operating Current	1600x1200 at 15 fps	IDIG		25	30	mA
Analog Operating Current		IANA		52	58	mA
Digital Operating Current	1280x720 at 30 fps	IDIG		48	55	mA
Analog Operating Current	Full field of view	IANA		65	73	mA
Digital Operating Current	640x480 at 30 fps	IDIG		24	30	mA
Analog Operating Current		IANA		40	45	mA
Digital Operating Current	320x240 at 120 fps	IDIG		30	35	mA
Analog Operating Current		IANA		49	55	mA
Hard Standby	Analog				10	uA
	Digital				15	uA
Soft Standby (Clock Off)	Analog			20	275	uA
	Digital			30	300	uA
Soft Standby (Clock on 6 MHz)	Analog			20	275	uA
	Digital			50	500	uA
Soft Standby (Clock on 27 MHz)	Analog			20	275	uA
	Digital			1	3	mA

Absolute Maximum Ratings

Table 28. ABSOLUTE MAXIMUM VALUES

Definition	Condition	Symbol	Min	Max	Unit
Core Digital Voltage		V_{DD_MAX}	–0.3	2.2	V
Analog Voltage		V_{AA_MAX}	–0.3	3.2	V
Pixel Supply Voltage		$V_{AA\_PIX\_MAX}$	–0.3	3.2	V
PLL Supply Voltage		$V_{DD\_PLL\_MAX}$	–0.3	3.2	V
Input HIGH Voltage		V_{IH_MAX}	0.7 x V_{DD}	V_{AA} + 0.3	V
Input LOW Voltage		V_{IH_MAX}	–0.3	0.3 x V_{DD}	V
Operating Temperature	Measure at Junction	T_{OP}	–30	70	°C
Storage Temperature		T_{STG}	–40	125	°C

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

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

CHIEF RAY ANGLE

CRA vs. Image Height Plot

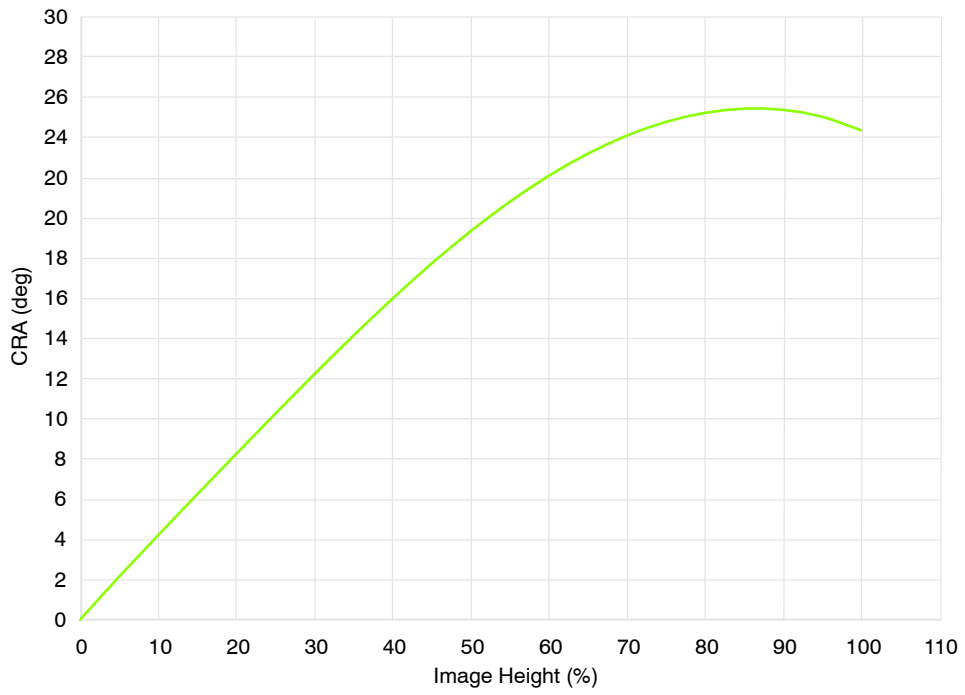


Figure 32. Chief Ray Angle

Image Height (%)	Image Height (mm)	CRA (deg)
0	0	0
5	0.088	2.14
10	0.175	4.21
15	0.263	6.25
20	0.350	8.27
25	0.438	10.27
30	0.525	12.24
35	0.613	14.16
40	0.700	16.00
45	0.788	17.74
50	0.875	19.36
55	0.963	20.83
60	1.050	22.12
65	1.138	23.23
70	1.225	24.13
75	1.313	24.81
80	1.400	25.26
85	1.488	25.47
90	1.575	25.42
95	1.663	25.07
100	1.750	24.40

SMIA AND MIPI SPECIFICATION REFERENCE

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

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

PUBLICATION ORDERING INFORMATION

LITERATURE FULFILLMENT:
Literature Distribution Center for ON Semiconductor
19521 E. 32nd Pkwy, Aurora, Colorado 80011 USA
Phone: 303−675−2175 or 800−344−3860 Toll Free USA/Canada
Fax: 303−675−2176 or 800−344−3867 Toll Free USA/Canada
Email: orderlit@onsemi.com

N. American Technical Support: 800−282−9855 Toll Free USA/Canada
Europe, Middle East and Africa Technical Support:
Phone: 421 33 790 2910
Japan Customer Focus Center
Phone: 81−3−5817−1050

ON Semiconductor Website: **www.onsemi.com**

Order Literature: http://www.onsemi.com/orderlit

For additional information, please contact your local Sales Representative