Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

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.

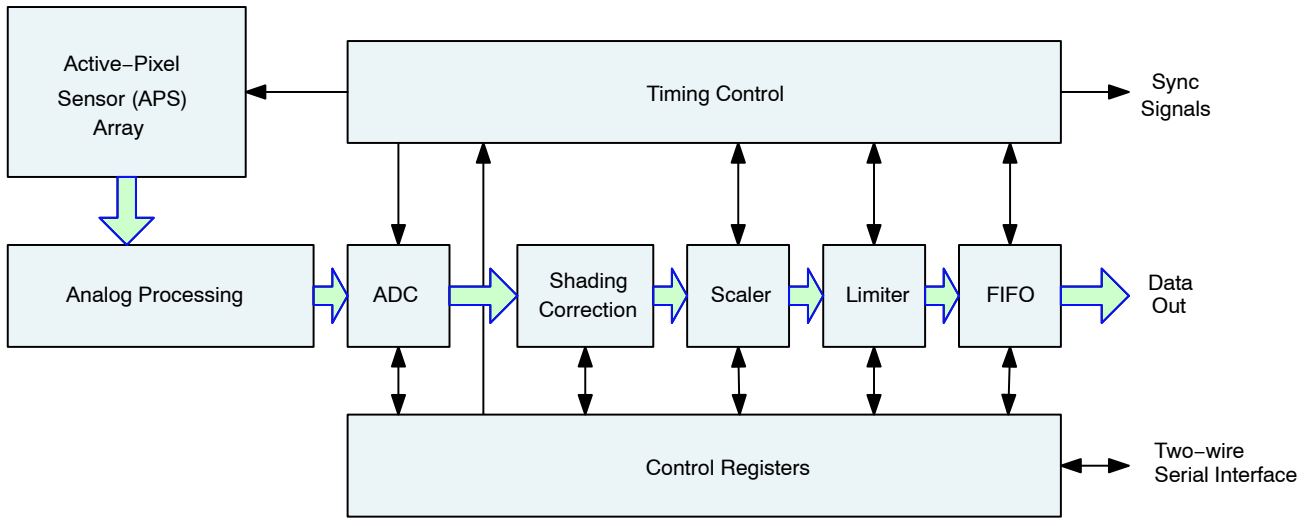


Figure 1. Block Diagram

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.

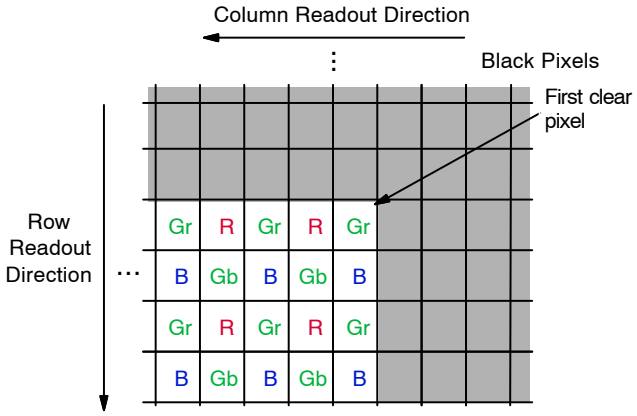


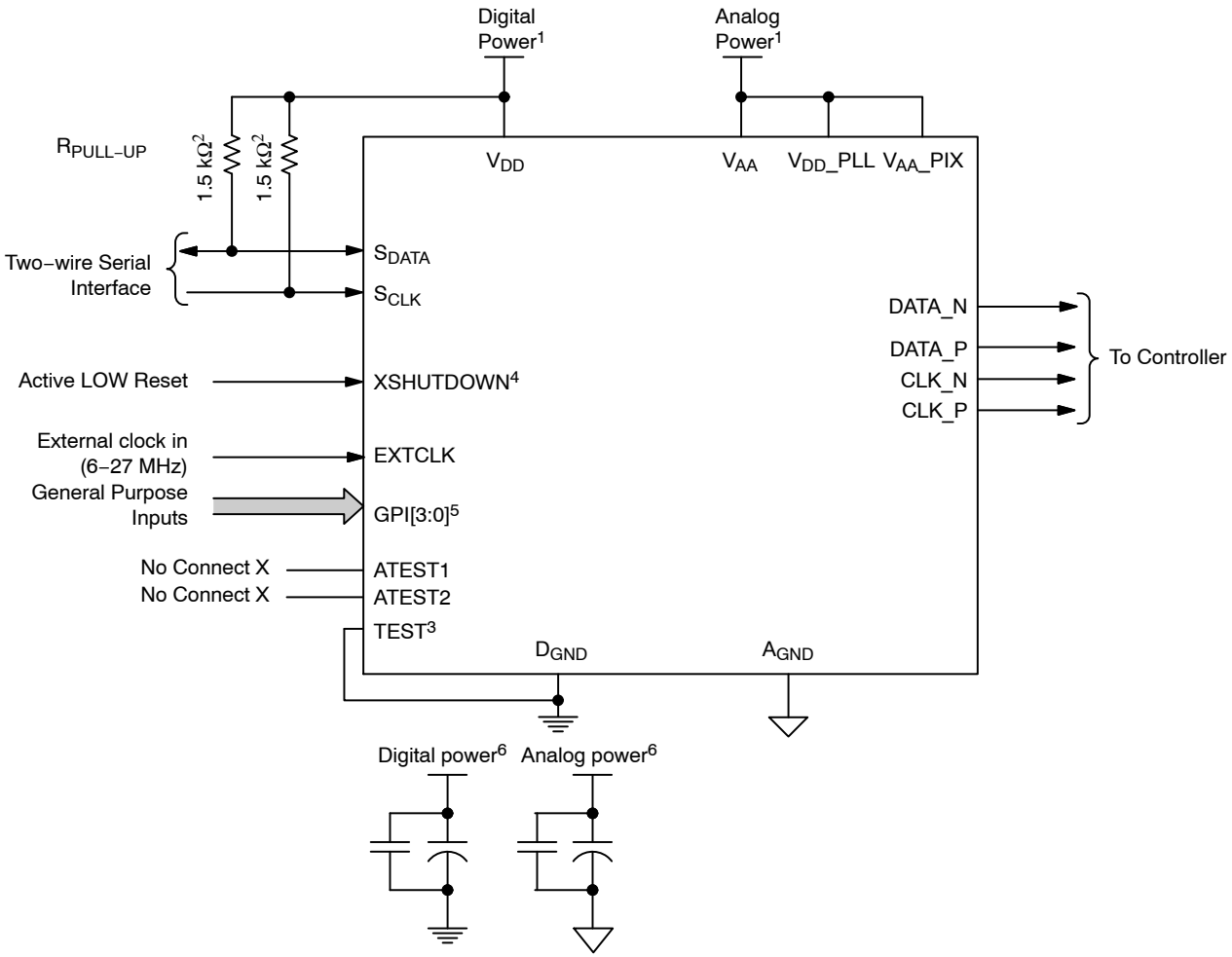
Figure 2. Pixel Color Pattern Detail (Top Right Corner)

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.

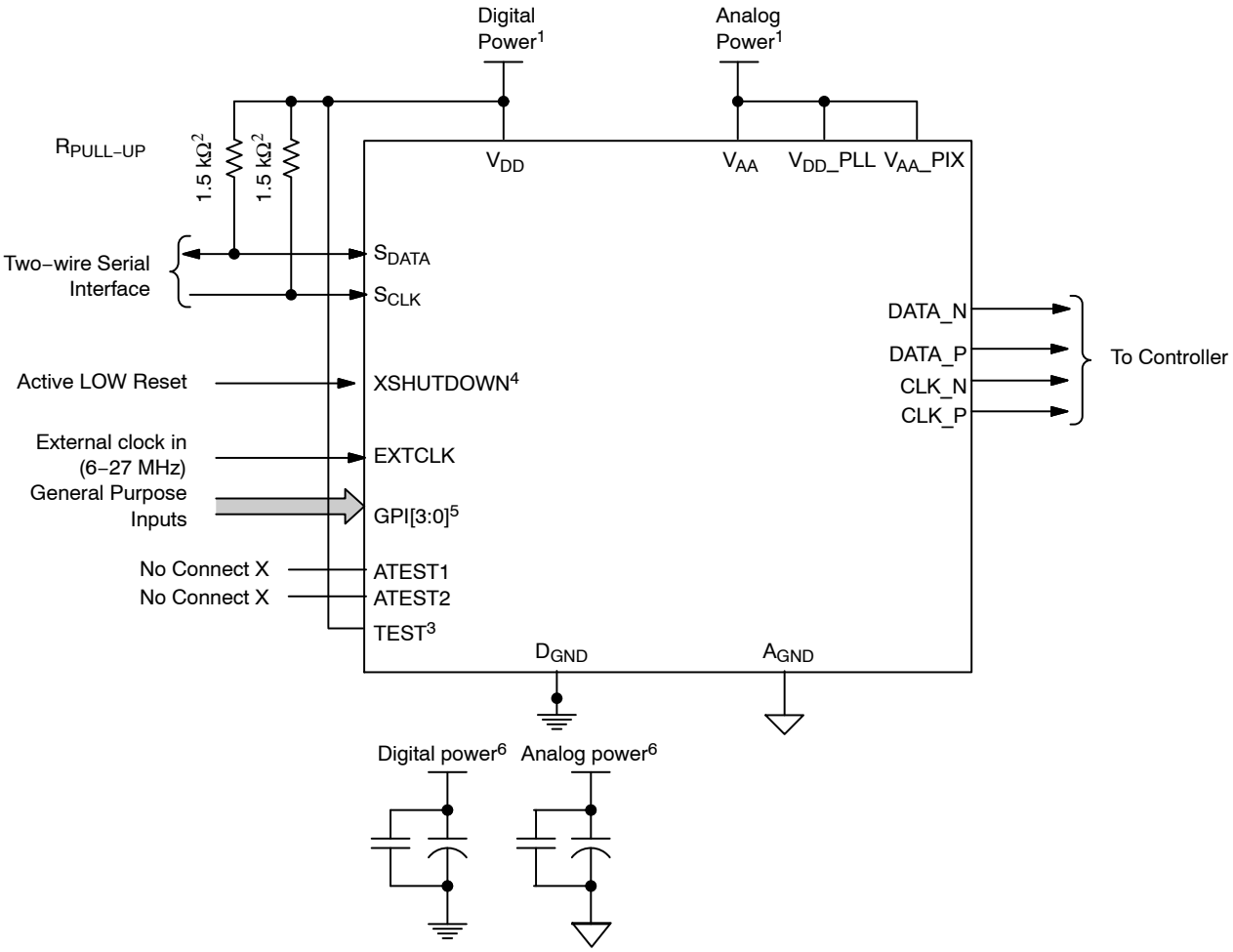
Typical configurations are shown in Figure 3 and Figure 4. These operating modes are described in “Control of the Signal Interface”.

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.

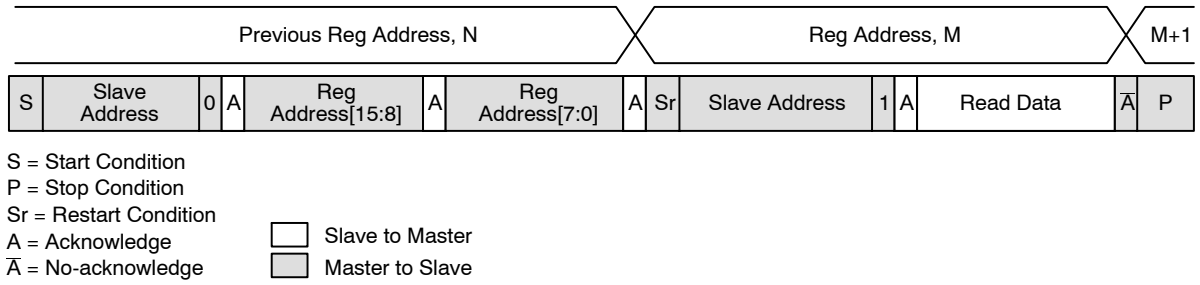


Figure 5. Single READ from Random Location

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.

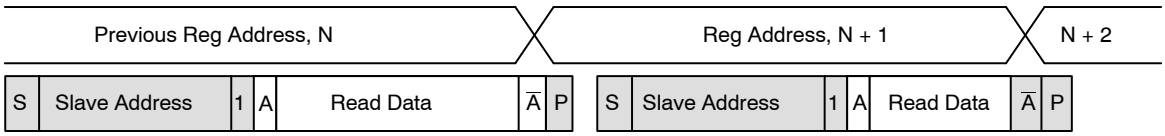


Figure 6. Single READ from Current Location

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.

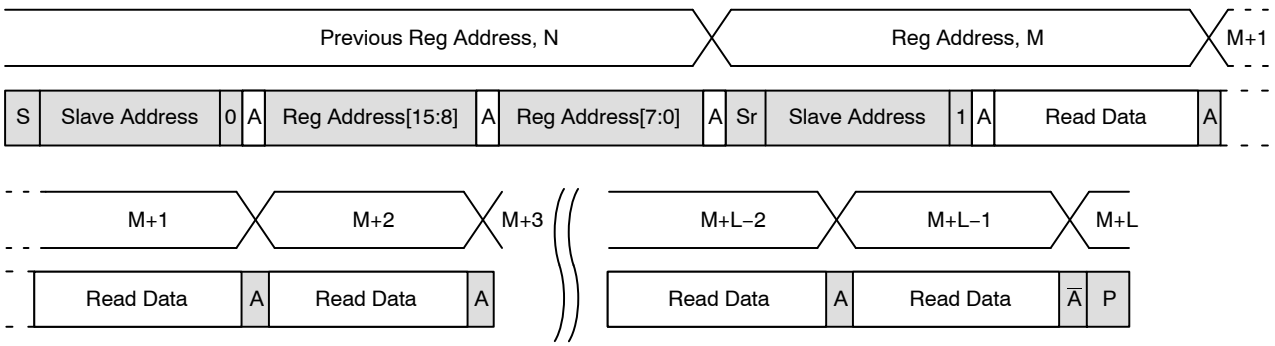


Figure 7. Sequential READ, Start from Random Location

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.

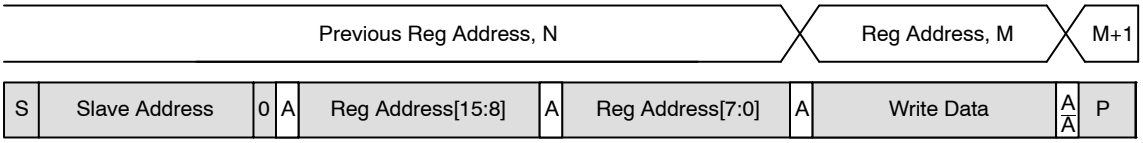


Figure 9. Single WRITE to Random Location

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.

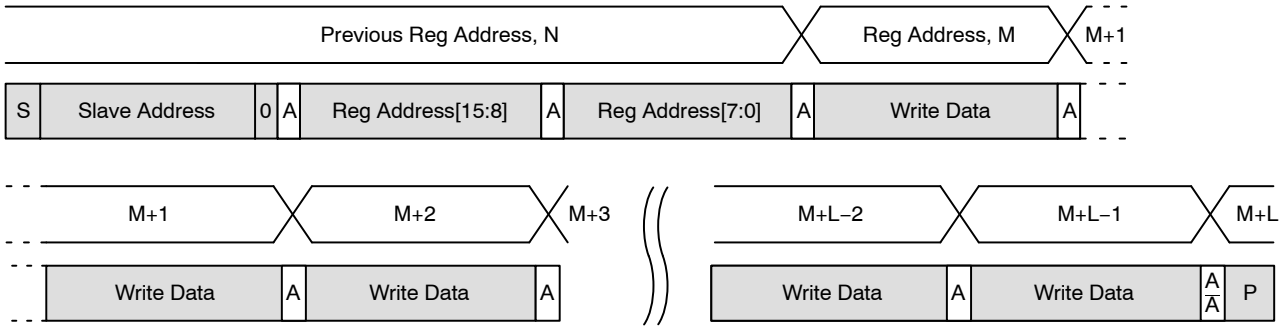


Figure 10. Sequential WRITE, Start at Random Location

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.

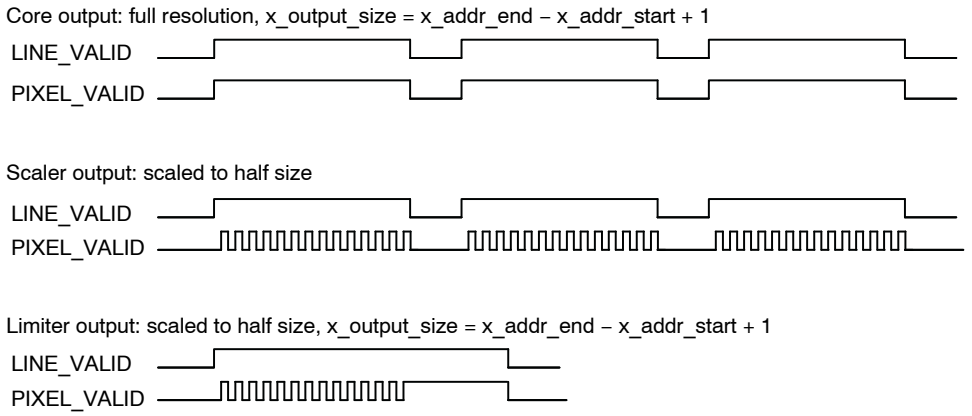


Figure 11. Effect of Limiter on the SMIA Data Path

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.

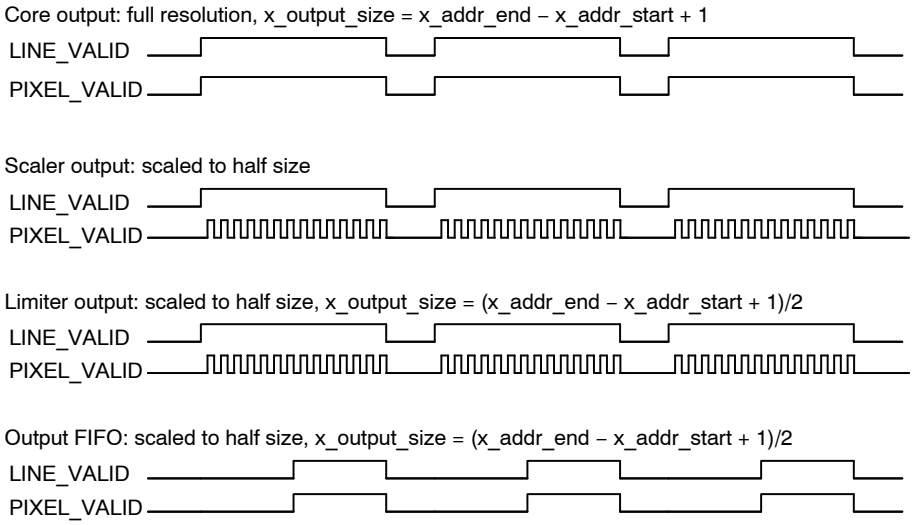


Figure 12. Timing of SMIA Data Path

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