1/4-Inch 2Mp System-On-A-Chip (SOC) CMOS Digital Image Sensor

MT9D112

For the latest data sheet, refer to Aptina's Web site: www.aptina.com

Features

- DigitalClarity® CMOS imaging technology
- Superior low-light performance
- Ultra-low-power, low-cost
- Internal master clock generated by on-chip phase-locked loop (PLL) oscillator
- Electronic rolling shutter (ERS), progressive scan
- Integrated image flow processor (IFP) for single-die camera module
- Automatic image correction and enhancement, including 4-channel lens shading correction with independent corner correction
- Arbitrary image scaling with anti-aliasing
- Integrated microcontroller for flexibility
- Two-wire serial interface providing access to registers and microcontroller memory
- Selectable output data format: YCbCr, 565RGB, 555RGB, 444RGB, processed Bayer, RAW8, and RAW10 bit
- Output FIFO for data rate equalization
- Programmable I/O slew rate
- Parallel and serial MIPI data output
- Xenon and LED flash support with fast exposure adaptation
- Flexible support for external auto focus, optical zoom, and mechanical shutter
- Independently configurable gamma correction

Applications

- Cellular phones
- PC cameras
- PDAs

Table 1: Key Performance Parameters

Parameter		Value
Optical format		1/4-inch (4:3)
Full resolution		1600 x 1200 pixels (UXGA)
Pixel size		2.2 x 2.2µm
Chief ray angle		22.1 deg maximum at 75% image height 27.0 deg maximum at 80% image height
Color filter array		RGB Bayer pattern
Active pixel array area		3.56mm x 2.68mm
Shutter type		Electronic rolling shutter (ERS) with global reset
Input clock frequency		6–54 MHz
Maximum frame rate		15 fps at full resolution, 24 fps in preview mode, 30 fps in video mode
Maximum data rate/ master clock		80 Mb/s 6 MHz to 80 MHz
Supply voltage	Analog	2.5–3.1V
	Digital	1.7–1.95V
	I/O	1.7–1.95V or 2.5–3.1V
	PLL	2.5–3.1V
	AF	1.7–3.1V
ADC resolution		10-bit, on-die
Responsivity		0.53 V/lux-sec (preliminary)
Dynamic range		59.5dB (preliminary)
SNR_{MAX}		37.7dB (preliminary)
Power consumption		245mW at 15 fps, full resolution
		230mW at 30 fps, video mode 168mW at 24 fps, preview mode
		10µW, standby/shutdown
Operating temperature		–30°C to +70°C (at junction)
Packaging		Bare die

Ordering Information

Table 2: Available Part Numbers

Part Number	Description
MT9D112D00STCK15AC1	Bare die (22.1 deg CRA)
MT9D112D00STCZK15AC1	Bare die (27.0 deg CRA)

Table of Contents

List of Figures

List of Tables

General Description

The Aptina® MT9D112 is a 1/4-inch 2Mp CMOS image sensor with an integrated advanced camera system. This camera system features a microcontroller (MCU), a sophisticated image flow processor (IFP), and both parallel and serial mobile industry processor interface (MIPI) interfaces. It also includes a programmable general purpose I/O module (GPIO), which can be used to control external auto focus (AF), optical zoom, or mechanical shutter. The microcontroller manages all components of the camera system and sets key operation parameters for the sensor core to optimize the quality of raw image data entering the IFP. The sensor core consists of an active pixel array of 1616 x 1216 pixels, programmable timing and control circuitry including a PLL and external flash support, analog signal chain with automatic offset correction and programmable gain, and two 10-bit analog-to-digital converters (ADC). The entire system-on-a-chip (SOC) has ultra-low power requirements and superior low light performance that is particularly suitable for mobile applications. The MT9D112 is based on DigitalClarity® Technology—Aptina's breakthrough low-noise CMOS imaging technology that achieves near-CCD image quality (based on signal-to-noise ratio and low-light sensitivity) while maintaining the inherent size, cost, power consumption, and integration advantages of CMOS.

MT9D112 Overview

The MT9D112 has a color image sensor with a Bayer color filter arrangement and a 2Mp active-pixel array with electronic rolling shutter and global reset. The sensor core readout is 10-bit and supports skipping, binning and can be flipped and/or mirrored. The sensor core also supports separate analog and digital gain for all four color channels (R, Gr, Gb, B).

The MT9D112 also has an embedded phase-locked loop oscillator (PLL) that can generate the internal sensor clock from the common wireless system clock. When in use, the PLL adjusts the incoming clock frequency up, allowing the MT9D112 to run at almost any desired resolution and frame rate within the sensor's capabilities. The PLL can be bypassed and powered down to reduce power consumption.

Low power consumption is a very important requirement for all components of wireless devices. The MT9D112 has numerous power-conserving features, including soft and hard standby modes, as well as an external SHUTDOWN pin that allows the internal power bus to be disabled.

Another important consideration for wireless devices is their electromagnetic interface (EMI). The MT9D112 can be used with either a serial MIPI interface or the parallel data output interface which has a programmable I/O slew rate to minimize EMI and an output FIFO to eliminate output data bursts.

The advanced image flow processor and flexible programmability of the MT9D112 provide a variety of ways to enhance and optimize the image sensor performance. Built-in optimization algorithms enable the MT9D112 to operate at factory settings as a fully automatic, highly adaptable camera; however, most of its settings are user-programmable.

These algorithms include black level conditioning, lens shading correction, defect correction, noise reduction, color interpolation, edge detection, color correction, aperture correction, and image formatting such as cropping and scaling.

The MT9D112 also includes a sequencer that coordinates all events triggered by the user. The sequencer manages auto focus, auto white balance, flicker detection, and auto exposure for the different operating modes which include preview, still capture, video, and snapshot with flash.

A two-wire serial register interface bus enables read/write access to control registers, variables, and special function registers within the MT9D112. The hardware registers are grouped internally by pages and include sensor core controls, color pipeline controls, and output controls. Variables are located in the microcontroller's RAM memory and are used for drivers such as the auto exposure (AE), auto white balance (AWB), and auto focus (AF). Special function registers are registers connected to the local bus of the microcontroller and include GPIO and the waveform generator.

The general purpose I/O can be configured to allow the user to output a flash or shutter pulse or to achieve 10-bit parallel output, or they can be configured as inputs to enable the user to use features such as an external trigger.

Signal Description

Table 3 provides the signal descriptions for the MT9D112.

Table 3: Signal Description

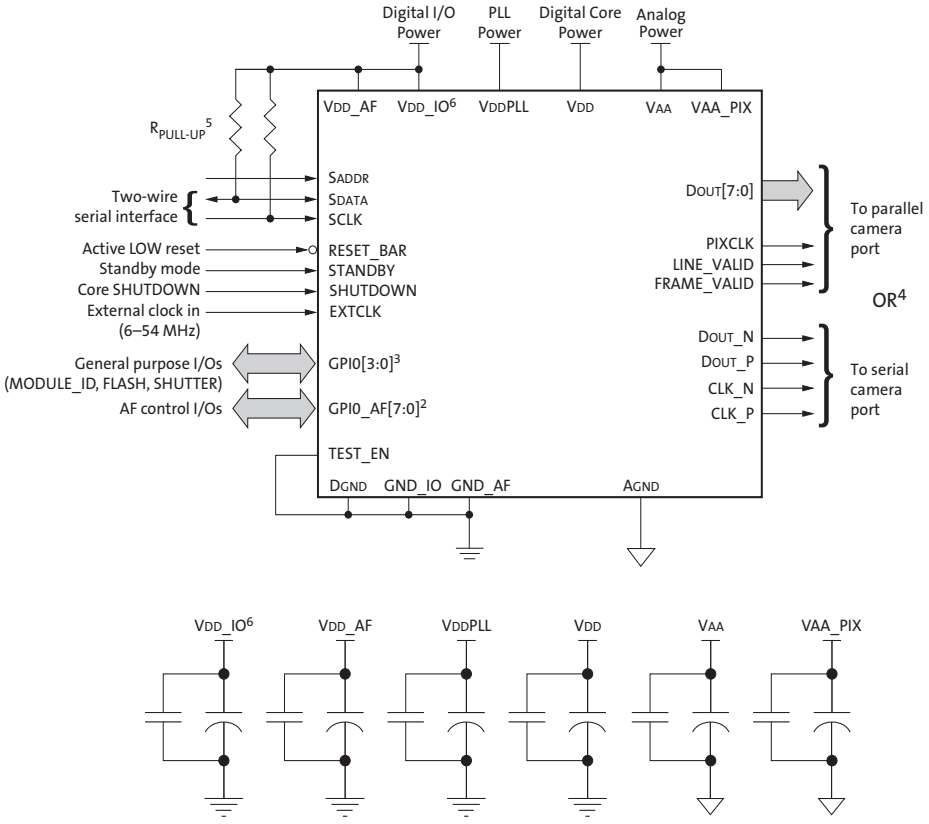
Name	Type	Description
SHUTDOWN	Input	Power down VDD, active HIGH.
TEST_EN	Input	Reserved for factory test. Tie to digital ground during normal operation (can leave floating if not used).
STANDBY	Input	Controls sensor standby mode, active HIGH.
SCLK	Input	Two-wire serial interface clock.
SADDR	Input	Selects device address for the two-wire serial interface. The address is 0x78 when SADDR is tied LOW, 0x7A if tied HIGH.
RESET_BAR	Input	Master reset signal, active LOW.
EXTCLK	Input	Input clock signal 6–54 MHz.
GPIO[3:0]	I/O	General purpose digital I/O, could be configured for FLASH/SHUTTER/DOUT_LSB0/DOUT_LSB1/ MODULE_ID/OE_BAR/TRIGGER.
SDATA	I/O	Two-wire serial interface data.
GPIO_AF[7:0]	I/O	General purpose digital I/O. Used for auto focus function (can leave floating if not used).
DOUT[7:0]	Output	Eight-bit image data output or most significant bits (MSB) of 10-bit sensor bypass mode.
DATA_OUT_N	Output	Differential MIPI data (sub-LVDS, negative) (must leave floating if not used).
DATA_OUT_P	Output	Differential MIPI data (sub-LVDS, positive) (must leave floating if not used).
CLK_OUT_N	Output	Differential MIPI clock (sub-LVDS, negative) (must leave floating if not used).
CLK_OUT_P	Output	Differential MIPI clock (sub-LVDS, positive) (must leave floating if not used).
PIXCLK	Output	Pixel clock. Used for sampling DOUT, FRAME_VALID, and LINE_VALID.
LINE_VALID	Output	Identifies lines in the active image.
FRAME_VALID	Output	Identifies rows in the active image.
VDD	Supply	Digital power (1.8V).
VAA_PIX	Supply	Pixel array power (2.8V).
VAA	Supply	Analog power (2.8V).
VDD_PLL	Supply	PLL power (2.8V).
VDD_IO	Supply	I/O power supply (1.7–1.95V or 2.5–3.1V).
GND_IO	Supply	I/O ground.
DGND	Supply	Digital, I/O, and PLL ground.
AGND	Supply	Analog ground.
VDD_AF	Supply	I/O power supply for GPIO_AF[7:0] pads (can leave floating if not used).
GND_AF	Supply	IO ground for GPIO_AF[7:0].

Typical Connections

Figure 1 shows typical MT9D112 device connections. For low-noise operation, the MT9D112 requires separate power supplies for analog and digital. Incoming digital and analog ground conductors can be tied together next to the die. Both power supply rails should be decoupled to ground using capacitors as close as possible to the die. The use of inductance filters is not recommended on the power supplies or output signals.

The MT9D112 also supports different digital core (VDD/DGND) and I/O power (VDD_IO/DGND) power domains that can be at different voltages. The PLL requires a clean power source (VDDPLL).

Figure 1: Typical Configuration (connections)



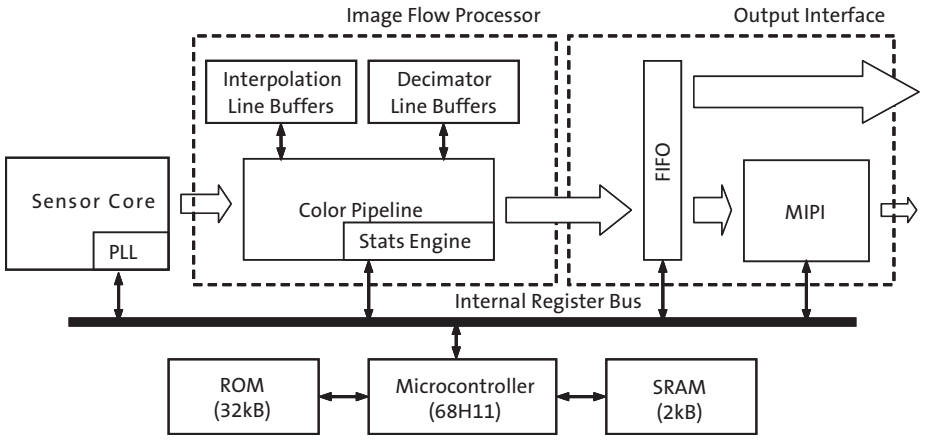
It is recommended that 0.1µF and 1µF decoupling capacitors for each power supply are mounted as close as possible to the pad. Actual values and results may vary depending on layout and design considerations.

Notes:
1. This typical configuration shows only one scenario out of multiple possible variations for this sensor.
2. If auto focus is not required, the following pads can be left floating: VDD_AF, GND_AF, and GPIO_AF.
3. The GPIO pads can serve multiple functions and can be reconfigured. The function and direction will vary by application.
4. Only one of the output modes (serial or parallel) can be used at any time.
5. A resister value of 1.5KΩ is recommended for the two-wire serial interface RPULL-UP, however, a greater value may be used for slower transmission speed.
6. All inputs must be configured with VDD_IO.
7. VAA and VAA_PIX must be tied together.

Architecture Overview

The MT9D112 combines a 2Mp sensor core together with an image flow processor (IFP) to form a stand-alone solution that includes both image acquisition and processing. The processed image data is transmitted to the host-system either via a parallel bus or a serial data interface through the output interface. In normal operation, an integrated microcontroller (MCU) controls most aspects of operation autonomously. This includes the control over key internal registers for both sensor core and the IFP settings. However, registers can be controlled by the user as well. This can be achieved by either controlling registers through the MCU, or by disabling the MCU for direct control.

Figure 2: SOC Block Diagram



Sensor Core Description

The sensor core of the MT9D112 is a progressive-scan sensor that generates a stream of pixel data at a constant frame rate, qualified by LINE_VALID and FRAME_VALID. The maximum pixel rate is 40 megapixels/second, corresponding to a pixel clock rate of 80 MHz. Figure 3 on page 12 shows a block diagram of the sensor core. It includes a 2-megapixel 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.

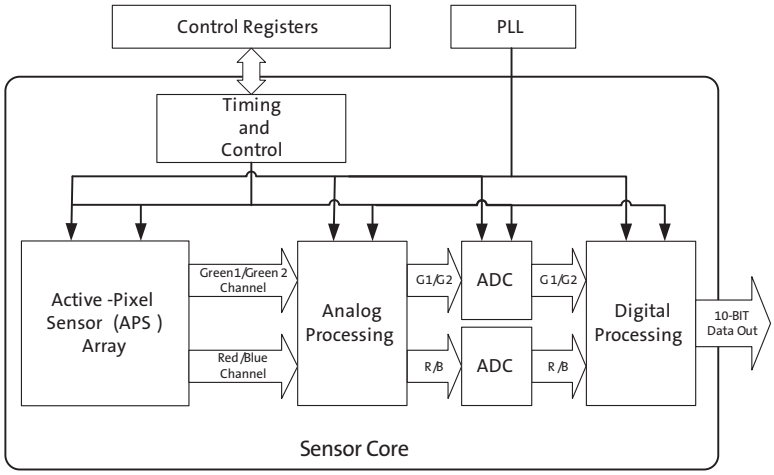
The pixel array contains optically active and light-shielded (dark) pixels. The dark pixels are used to provide data for the offset-correction algorithms (black level control).

The sensor core 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 are controlled by the SOC firmware and can be accessed through the two-wire serial interface. Note that register values written to the sensor core may be overwritten by firmware.

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

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

Figure 3: Sensor Core Block Diagram



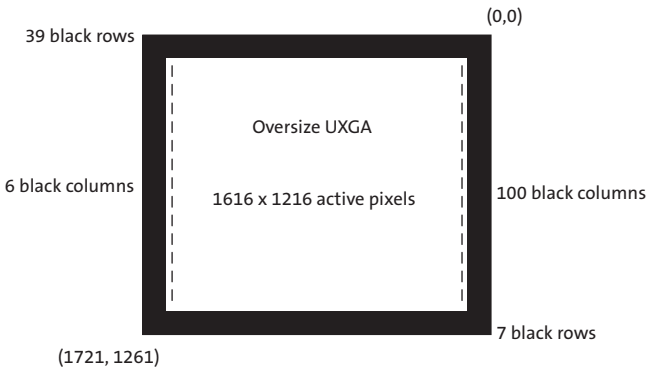
Pixel Array

Pixel Array Structure

The sensor core pixel array is configured as 1,722 columns by 1,262 rows (shown in Figure 4**)**. The first 100 columns and the first 39 rows of pixels are optically black and are used for the automatic black level adjustment; the last 6 columns are also optically black.

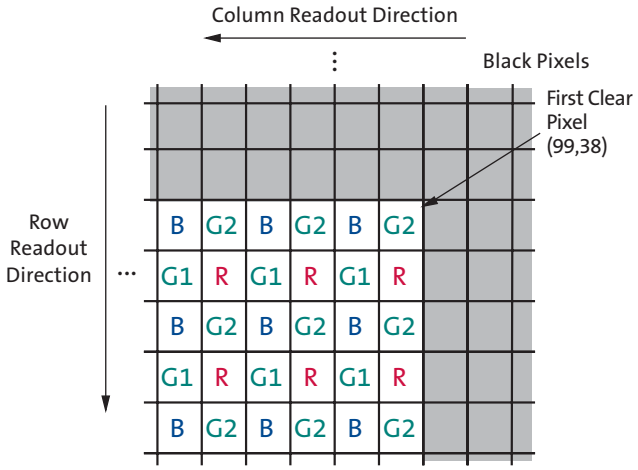
The optically active pixels are used as follows: In default mode a UXGA image (1,616 columns by 1,216 rows) is generated, starting at row 40, column 101. An 8-pixel boundary of active pixels is enabled around the image to avoid boundary effects during color interpolation and correction.

Figure 4: Pixel Array



The sensor core uses a Bayer color pattern, as shown in Figure 5. The even-numbered rows contain green and red color pixels; odd-numbered rows contain blue and green color pixels. Even-numbered columns contain green and blue color pixels; odd-numbered columns contain red and green color pixels.

Figure 5: Pixel Color Pattern Detail (Top Right Corner)

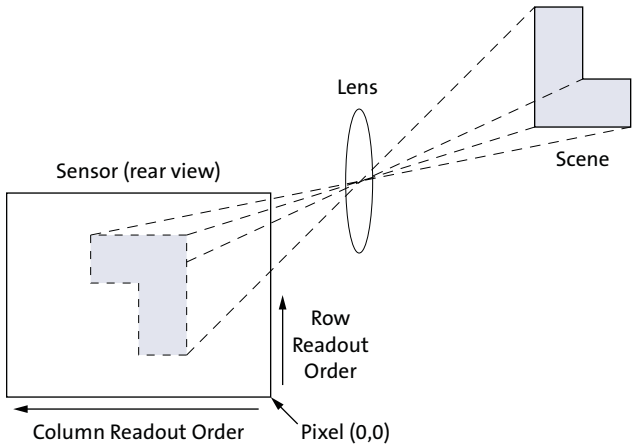


Default Readout Order

By convention, the sensor core pixel array is shown with pixel (0,0) in the top right-hand corner (see Figure 5). This reflects the actual layout of the array on the die. When the sensor is imaging in a system, the active surface of the sensor faces the scene as shown in Figure 6.

When the image is read out of the sensor, it is read one row at a time, with the rows and columns sequenced. By convention, data from the sensor is shown with the first pixel read out—pixel (99,38) in the case of the sensor core—in the top lefthand corner.

Figure 6: Imaging a Scene



Analog Processing

Analog Readout Channel

The sensor core features two identical analog readout channels as shown in Figure 3 on page 12. The readout channel consists of two gain stages, a sample-and-hold stage with black level calibration capability, and a 10-bit ADC.

Timing and Control

Gain Options

The MT9D112 provides per-color gain control as well as 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.

Integer digital gains in the range 0–7 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). Gain settings are updated every frame by the MCU auto feature; to make manual adjustments to gain settings, the MCU auto features must be disabled.

Integration Time

The integration (exposure) time of the MT9D112 is controlled by the fine_integration_time and coarse_integration_time registers. While coarse integration time controls the integration duration steps of lines, the fine time allows for sub-line accuracy. Integration time is updated every frame by the MCU auto feature; to make manual adjustments to integration time, the MCU auto features must be disabled.

The limits for the fine integration time are defined by:

sensor_fine_IT_min_A/B≤fine_integration_time≤(sensor_line_length_pck_A/B–sensor_fine_IT_max_margin_A/B)

Course integration time is < frame_length_lines_margin.

The actual integration time is given by:

$$integration\_time = \frac{((coarse\_integration\_time \quad x \quad line\_length\_pck) + fine\_integration\_time)}{vt\_pix\_clk\_freq\_mhz/1x10^{6}} \qquad \text{(EQ 1)}$$

line_length_pck is the value of R0x300C

The minimum allowable value for line_length_pck is given by the line_length_pck> = ((x_addr_end - x_addr_start + 1)/xskip) + min_line_blanking_pck.

min_line_blanking_pck is 476 in full resolution mode, 743 in high-power preview mode, and 442 in low-power preview mode.

coarse_integration_time is value of R0x3012

fine_integration_time is value of R0x3014

x_addr_end is a value of R0x3008

x_addr_start is a value of R0x3004

xskip is a value of R0x3016[7:5]

vt_pix_clk_freq_mhz is MCLK/2

It is fundamental to the operation of an electronic rolling shutter (ERS) that it is not possible to set an integration time that is greater than the frame time. Unlike earlier Aptina Imaging parts, setting an integration time that is greater than the frame time does not affect the frame time; the behavior is undefined. On the MT9D112, it is necessary to reprogram the frame time (frame_length_lines) in order to make longer exposure times available. Long integration times increase the likelihood of image degradation due to increased accumulation of dark current.

If the integration time is changed while FRAME_VALID 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 is 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.

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.

PLL

PLL-Generated Master Clock

The PLL can generate a master clock signal whose frequency is up to 80 MHz (input clock from 6 MHz through 54 MHz). R0x341C controls the frequency of the PLL-generated clock. It is possible to bypass the PLL and use CLKIN as master clock. In order to do so, one must set R0x341E[0] to “1.” If power consumption is a concern, R0x341E[1] should be also set to “1” a short time later, to put the bypassed PLL in power-down mode. To enable the PLL again, the two bits must be set to “0” in the reverse order. By default, the PLL is bypassed and powered down.

PLL Setup

Because the input clock frequency is unknown, the part starts with the PLL disabled. The PLL takes time to power up. During this time, the behavior of its output clock signal is not guaranteed. The PLL output frequency is determined by two constants, M and N, and the input clock frequency.

PLL programming and power-up sequence is as follows:

1. Program PLL frequency settings, R0x341C (pll_m, pll_n) (master clock frequency is equal to fVCO_pll/8). With default settings master clock frequency of 80 MHz is obtained with ^{f}CLKIN=16 MHz.
2. Power up PLL, R0x341E[1] = 0.
3. Wait for PLL settling time > 1ms.
4. Turn off PLL bypass, R0x341E[0] = 0.

Allow one complete frame to effect the correct integration time after enabling PLL.

Note: Before enabling PLL, ensure the slew rate is optimized for pads.

The default M and N values are for 16 MHz.

Note: Any changes to PLL settings must be done with PLL bypassed (R0x341E[0]=1). Also, EXTCLK must remain at a stable frequency with no missing pulses to ensure correct PLL operation. If the clock frequency is changed (or if pulses are missed) while the PLL is enabled, sensor operation is UNDEFINED. Depending on the duration and magnitude of the disturbance, frame rate will drop, frames will be corrupted, and the sensor may become non-functional. The proper sequence for changing PLL settings is: bypass the PLL, change the PLL settings, wait at least 1ms, disable PLL bypass.

Table 4: Frequency Parameters

Frequency	Equation	Min (MHz)	Max (MHz)
f_{IN}	–	6	54
f_{PFD}	f_{clkin} /(pll_n+1)	2	20
f_{VCO}	f_{clkin} * pll_m/(pll_n+1)	320	640

Readout Options

The sensor core supports different readout options to modify the image before it is sent to the IFP. The readout can be limited to a specific window of the original pixel array.

For preview modes, the sensor core supports both skipping and pixel averaging in x and y directions.

By changing the readout direction the image can be flipped in the vertical and/or mirrored in the horizontal.

Window Size

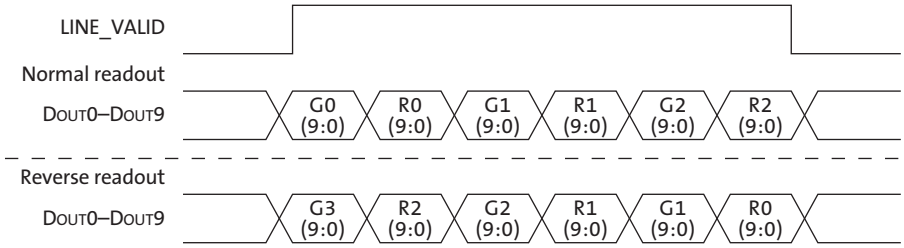
The image output size is set with registers x_addr_start, x_addr_end, y_addr_start, and y_addr_end. The edge pixels in the 1,616 x 1,216 array are present to avoid edge defects and should not be included in the visible window. Binning will change the image output size.

Readout Modes

Horizontal Mirror

When the horizontal_mirror bit (R0x3040[0]) is set in the read mode 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 7 on page 17 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. This change in sensor core output is corrected for by the SOC.

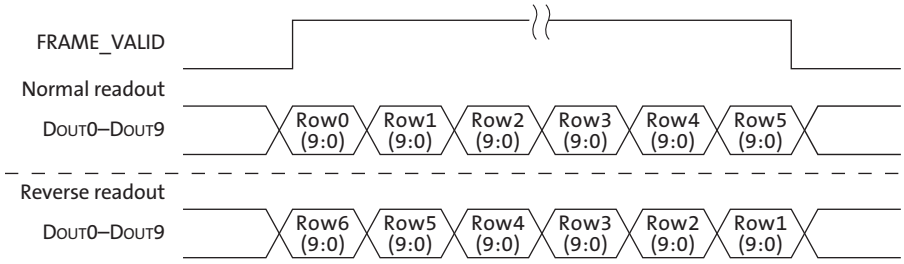
Figure 7: 6 Pixels in Normal and Column Mirror Readout Modes



Vertical Flip

When the vertical flip bit (R0x3040[1]) is set in the read mode 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 8 shows a sequence of 6 rows being read out with row mirror=0 and row mirror=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. This change in sensor core output is corrected for by the SOC.

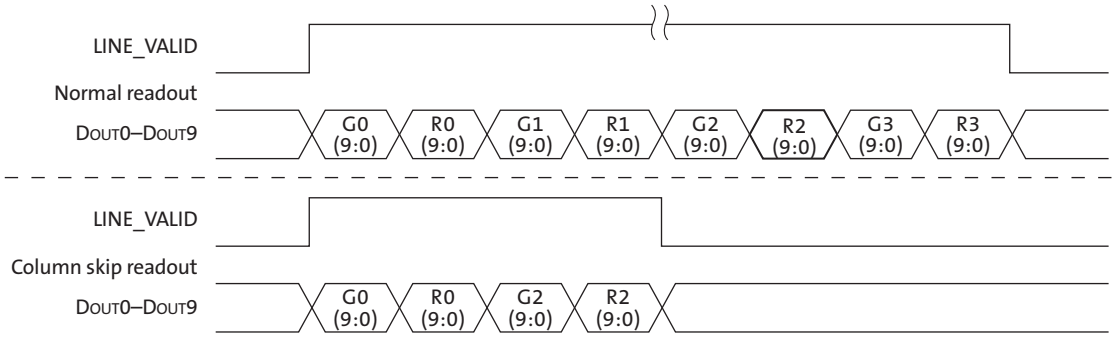
Figure 8: 6 Rows in Normal and Row Mirror Readout Modes



Column and Row Skip

The sensor core supports subsampling. Subsampling reduces the amount of data processed by the analog signal chain in the sensor and thereby allows the frame rate to be increased. Subsampling is enabled by setting x_odd_inc=3 and/or y_odd_inc=3. This reduces the amount of row and column data processed and is equivalent to the skip2 readout mode provided by earlier Aptina Imaging sensors. When enabling skipping, the proper image output and crop sizes must be updated beforehand.

Figure 9: 8 Pixels in Normal and Column Skip 2X Readout Modes



The following waveform shows a sequence of data being read out with x_odd_inc=3 and y_odd_inc=1. The effect of the different subsampling settings on the pixel array readout is shown in Figures 10 through 13.

Figure 10: Pixel Readout (no skipping)

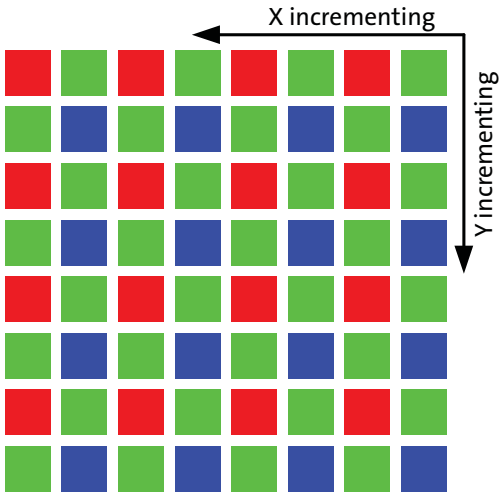


Figure 11: Pixel Readout (x_odd_inc=3, y_odd_inc=1)

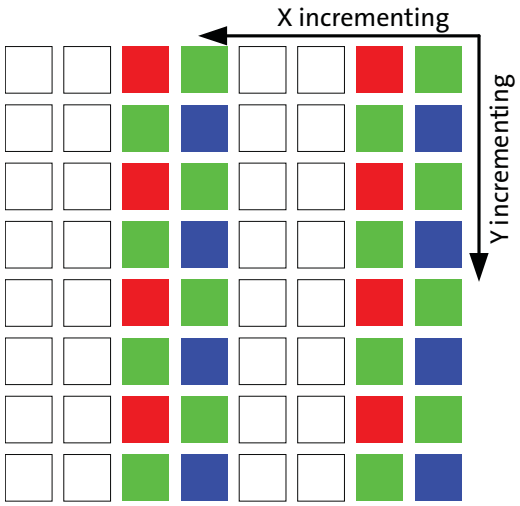


Figure 12: Pixel Readout (x_odd_inc=1, y_odd_inc=3)

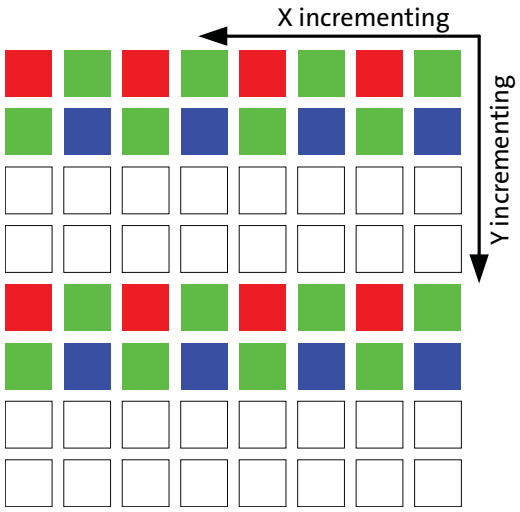
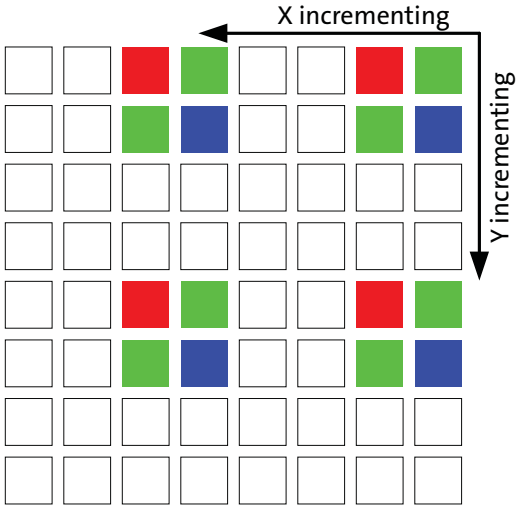


Figure 13: Pixel Readout (x_odd_inc=3, y_odd_inc=3)



Programming Restrictions when Skipping

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

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

remainder = (addr_end - addr_start + 1) AND 4;

if (remainder == 0) addr_end = addr_end - 2;

Table 5 shows the row address sequencing for normal and subsampled (with y_odd_inc=3) readout. The same sequencing applies to column addresses for subsampled readout. There are two possible subsampling sequences (because the subsampling sequence only reads half of the rows and columns) depending upon the alignment of the start address.

Table 5: Row Address Sequencing

Normal	Skipping (start divides by 4)	Skipping (start does not divide by 4)	Binned (start divides by 4)	Binned (start does not divide by 4)
0	0		0, 2	
1	1		1, 3	
2		2		2, 4
3		3		3, 5
4	4		4, 6	
5	5		5, 7	
6		6		6, 8
7		7		7, 9

Binning

The MT9D112 sensor core supports 2x1 and 2x2 analog binning (column binning, also called x-binning and row/column binning, also called xy-binning). Binning has many of the same characteristics as subsampling but because it gathers image data from all pixels in the active window (rather than a subset of them), it achieves superior image quality and avoids the aliasing artifacts that can be a characteristic side effect of subsampling.

Binning is enabled by selecting the appropriate subsampling settings (x_odd_inc=3 and y_odd_inc=1 for x-binning, x_odd_inc=3 and y_odd_inc=3 for xy-binning) and setting the appropriate binning bit in read_mode (R0x3040-1). As for subsampling, x_addr_end and y_addr_end may require adjustment when binning is enabled.

The effect of the different subsampling settings is shown in Figure 14 and Figure 15.

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

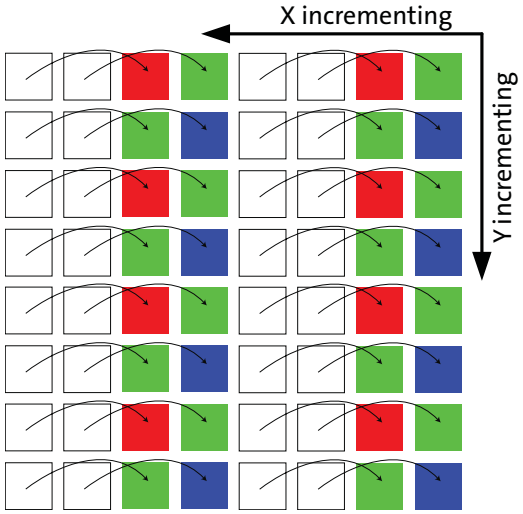
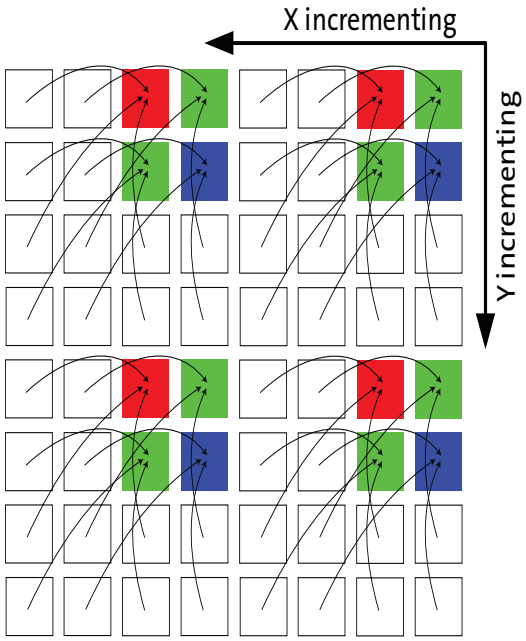


Figure 15: Pixel Readout (x_odd_inc=3, y_odd_inc=3, x_ybin=1)



Binning Limitations

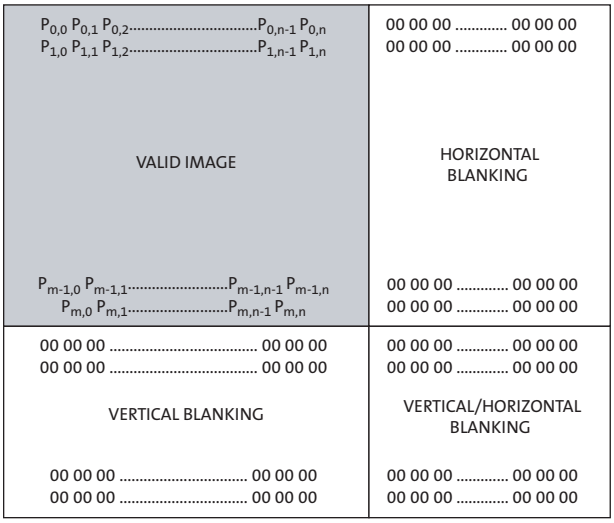
Binning requires different sequencing of the pixel array and imposes different timing limits on the operation of the sensor. In particular, xy-binning requires two read operations from the pixel array for each line of output data, which has the effect of increasing the minimum line blanking time.

As a result, when xy-binning is enabled, some of the programming limits declared in the parameter limit registers are no longer valid. In addition, the default values for some of the manufacturer-specific registers need to be reprogrammed. None of these adjustments are required for x-binning. The sensor must be taken out of streaming mode before switching between binned and non-binned operation.

Raw Data Format

The sensor core image data is read out in a progressive scan. Valid image data is surrounded by horizontal blanking and vertical blanking as shown in Figure 16. The amount of horizontal blanking and vertical blanking is programmable. LINE_VALID is HIGH during the shaded region of the figure. FRAME_VALID timing is described in the next section.

Figure 16: Pixel Data Timing Example



Raw Data Timing

The sensor core output data is synchronized with the PIXCLK output. When LINE_VALID is HIGH, one pixel data is output on the 10-bit DOUT output every PIXCLK period. By default, the PIXCLK signal runs at half the frequency as the master clock, and its falling edges occur one master clock period (half PIXCLK period) after transitions on LINE_VALID, FRAME_VALID, and DOUT (Figure 17). This allows PIXCLK to be used as a clock to sample the data. PIXCLK is continuously enabled, even during the blanking period.

Figure 17: Pixel Data Timing Example

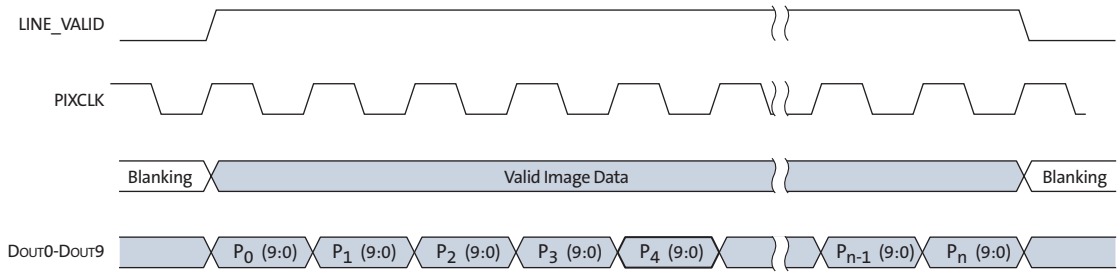
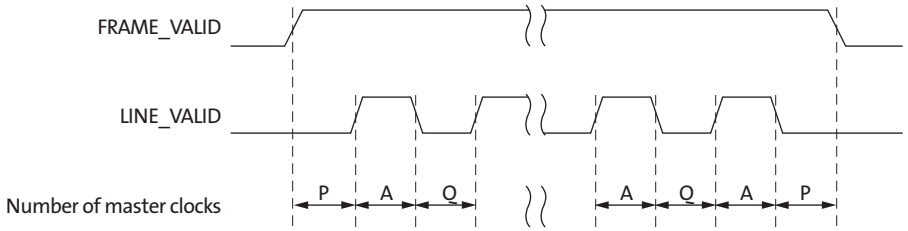


Figure 18: Row Timing and FRAME_VALID/LINE_VALID Signals



The sensor timing is shown in terms of pixel clock and master clock cycles (Figure 17 and Figure 18). Increasing the integration time to more than one frame causes the frame time to be extended.

Table 6: Row Timing Parameters

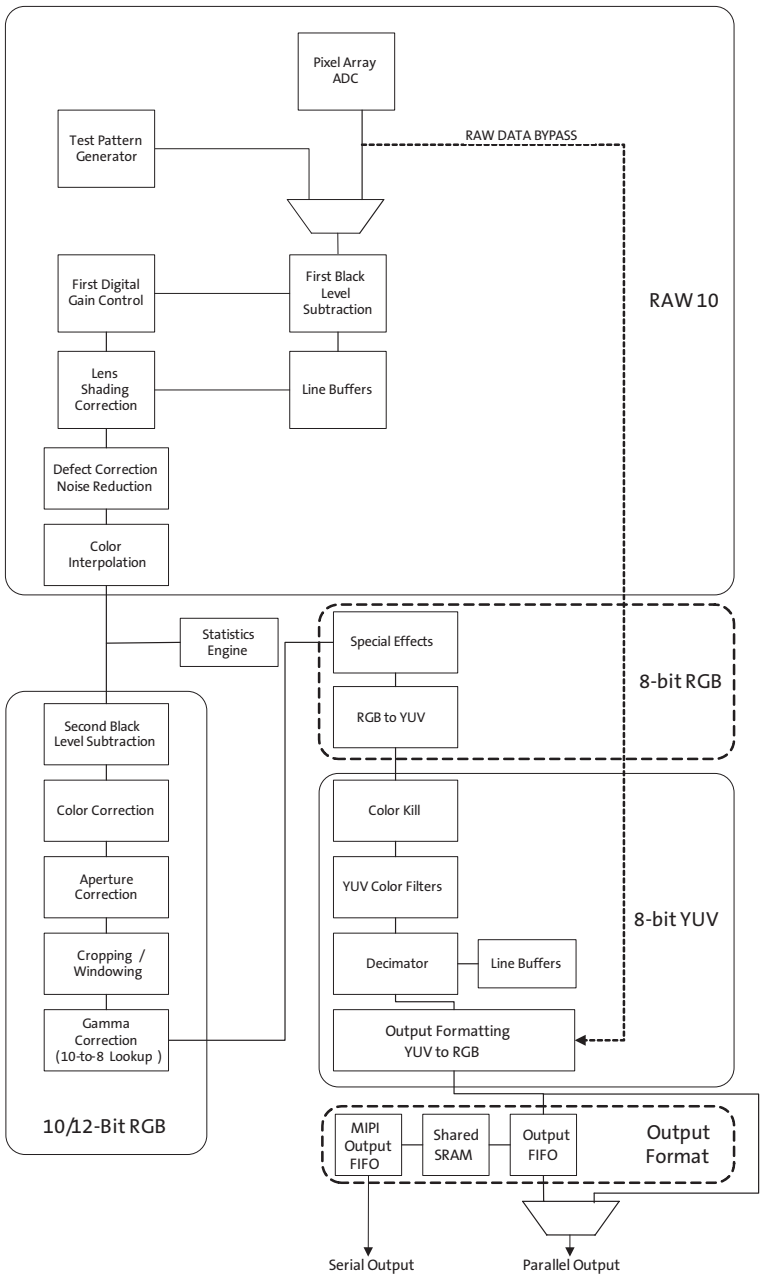
Parameter	Name		Equation	Default at CLKIN = 54 MHz (PLL Off)
PIXCLK_PERIOD	Pixel clock period	PLL Off	2*R0x3016[2:0]/^{f}CLKIN	1 pixel clock = 37.04ns
		PLL On	(16*R0x3016[2:0]*(pll_n+1))/(^{f}CLKIN*pll_m)	
S	Skip (subsampling) factor		For x_odd_inc=y_odd_inc=3, S=2 Otherwise, S=1	1
A	Active data time		(x_addr_end-x_addr_start+1)*PIXCLK_PERIOD/S	1,608 pixel clocks = 59.56?s
P	Frame start/end blanking		6*PIXCLK_PERIOD	6 pixel clocks = 222.22ns
Q	Horizontal blanking		(line_length_pck*PIXCLK_PERIOD)-A	476 pixel clocks = 17.63µs
A + Q	Row time		line_length_pck*PIXCLK_PERIOD	2,084 pixel clocks = 77.19µs
N	Number of rows		(y_addr_end-y_addr_start +1)/S	1,208 rows
V	Vertical blanking		([frame_length_lines-N]*[A+Q])+Q-(2*P)	110,916 pixel clocks = 4.108ms
Tfv	Frame valid time		(N*(A+Q))-Q+(2*P)	2,517,008 pixel clocks = 93.22ms
F	Total frame time		line_length_pck*frame_length_lines*PIXCLK_PERIOD	2,627,924 pixel clocks = 97.33ms

SOC Description

Image Flow Processor

Image and color processing in the MT9D112 is implemented as an image flow processor coded in hardware logic. The IFP can be controlled by registers from the outside but during normal operation, the embedded microcontroller will automatically adjust the operation parameters. The IFP is broken down into different sections outlined in Figure 19.

Figure 19: Color Pipeline

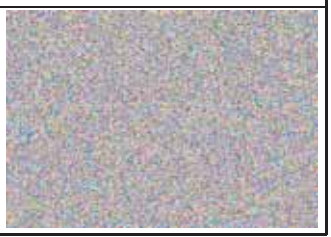


Test Patterns

During normal operation of the MT9D112, a stream of raw image data from the sensor core is continuously fed into the color pipeline. For test purposes, this stream can be replaced with a fixed image generated by a special test module in the pipeline. The module provides a selection of test patterns sufficient for basic testing of the pipeline.

Test patterns are accessible using R0x3290 and are shown in Figure 20. Disabling MCU is recommended before enabling test patterns.

Figure 20: Test Patterns

Test Pattern	Register Value	Example
Flat Field	R0x3290 = 1	
Vertical Ramp	R0x3290 = 2	
Color Bar	R0x3290 = 3	
Vertical Stripes	R0x3290 = 4	
Pseudo-Random	R0x3290 = 5	

First Black Level Subtraction and Digital Gain

Image stream processing starts with black level subtraction (R0x3278) and multiplication of all pixel values by a programmable digital gain (R0x32DC). Both operations can be independently set to separate values for each color channel (R, Gr, Gb, B). Independent color channel digital gain is adjusted with registers R0x32D4 through R0x32DA. Independent color channel black level adjustments can be made with R0x327A through R0x3280. If the black level subtraction produces a negative result for a particular pixel, the value of this pixel is set to "1."

Lens Shading Correction

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 lens shading. The MT9D112 has an embedded lens shading correction (LC) module that can be programmed to counter the shading effect of a lens on each individual R, Gb, Gr, and B color signal. The LC module multiplies R, Gb, Gr, and B signals by a 2-dimensional correction function F (x,y), whose profile in both x and y direction is a piecewise quadratic polynomial with coefficients independently programmable for each direction and color. Lens shading correction can be enabled and disabled with R0x3210[2].

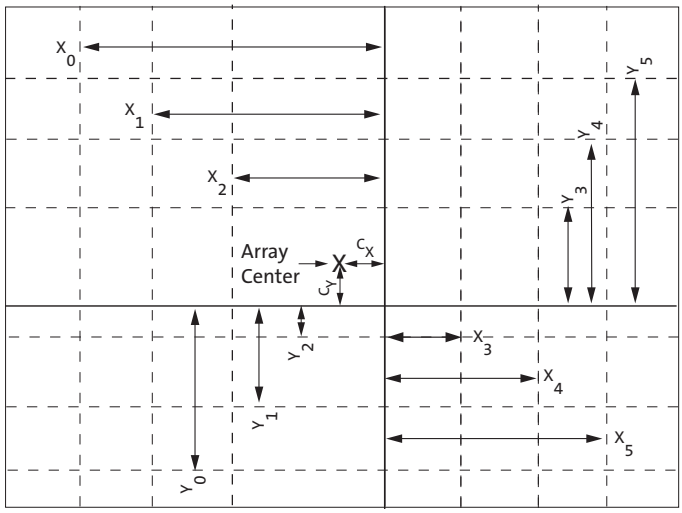
The MT9D112 also includes 16 independent corner parameters, K, for each of the color channels. The K factors are independently adjustable at each corner and are not dependent on the vertical and horizontal spatial dimension. They can be adjusted with R0x3544 through R0x3562.

Lens shading correction can be enabled and disabled with R0x3210[2].

Lens Correction Zones

In order to increase the precision of the correction function, the image plane is divided into 8 zones in each dimension. The coordinates of zone boundaries are referenced with respect to the lens center, C. Each boundary as well as C (Cx, Cy) coordinate is stored as a byte, which represents the coordinate value divided by 4. There always three boundaries to the left (top) of the center and three to the right (bottom) of the center. These boundaries apply uniformly for all color channels. However, the correction functions are programmable independently for each color component. Boundary and lens center positions are also valid for the preview mode. Figure 21 on page 27 illustrates the lens correction zones, which are accessible through registers R0x34CE through R0x34DC.

Figure 21: Lens Correction Zones



Defect Correction and Noise Reduction

The IFP performs on-the-fly defect correction that can mask pixel array defects such as high dark current (hot) pixels and pixels that are darker or brighter than their neighbors due to photoresponse nonuniformity. Each pixel is compared with its 8 nearest neighbors having the same filter color. Suppose the value of the compared pixel is P and the values of the 8 neighbors range from PMIN to PMAX. If P > PMAX, P is replaced with PMAX. Likewise, if P < PMIN, P is replaced with PMIN. Otherwise, the pixel is not considered defective, and its value P is not changed. Defect correction can be enabled and disabled with R0x3210[3].

The image data for each color channel can be passed through an adaptive noise suppression module which averages over flat field areas while preserving edge information. The module is edge aware with exposure that is based on configurable thresholds. The thresholds are changed on the fly based on the brightness of the current scene. Noise reduction can be enabled and disabled with R0x33F4[3] and thresholds are set by R0x33F6 through R0x33FC.

Color Interpolation and Edge Detection

In the raw data stream fed by the sensor core to the IFP, each pixel is represented by a 10-bit integer number, which can be considered proportional to the pixel's response to a one-color light stimulus, red, green or blue, depending on the pixel's position under the color filter array. Initial data processing steps, up to and including the defect correction, preserve the 1-color-per-pixel nature of the data stream, but after the defect correction it must be converted to a 3-colors-per-pixel stream appropriate for standard color processing. The conversion is done by an edge-sensitive color interpolation module. The module pads the incomplete color information available for each pixel with information extracted from an appropriate set of neighboring pixels. The algorithm used to select this set and extract the information seeks the best compromise between preserving edges and filtering out high frequency noise in flat field areas. The edge threshold can be set with R0x328E.

Second Black Level Correction

After interpolation it might be necessary to recalibrate the black level. A second global black level substraction is possible at this stage in the pipeline, which is controlled via R0x3276. If the subtraction produces a negative result for a particular pixel, the value of this pixel is set to "0."

Color Correction and Aperture Correction

To achieve good color fidelity of IFP output, interpolated RGB values of all pixels are subjected to color correction. The IFP multiplies each vector of three pixel colors by a 3 x 3 color correction matrix. The three components of the resulting color vector are all sums of three 10-bit numbers. Since such sums can have up to 12 significant bits, the bit width of the image data stream is widened to 12 bits per color (36 bits per pixel). The color correction matrix can be either programmed by the user or automatically selected by the auto white balance (AWB) algorithm implemented in the IFP. Color correction should ideally produce output colors that are independent of the spectral sensitivity and color crosstalk characteristics of the image sensor. The optimal values of color correction matrix elements depend on those sensor characteristics and on the spectrum of light incident on the sensor. The color correction variables can be adjusted with R0x006 through R0x042.

To increase image sharpness, a programmable 1D or 2D aperture correction (sharpening filter) is applied to color corrected image data. Aperture correction is enabled using R0x3210. The gain and threshold for 1D and 2D correction can be defined via R0x326A and R0x326C.

Image Cropping

Image cropping takes place when the sensor core is programmed to output pixel values from a rectangular portion of its pixel array-a window-smaller than the default 1,600 x 1,200 window. Pixels outside the selected cropping window are not read out, resulting in a narrower field of view than at the default sensor settings. Irrespective of the size and position of the cropping window, the MT9D112 sensor core can also decimate outgoing images by skipping columns and/or rows of the pixel array, and/or by binning 2 x 2 groups of pixels of the same color. Since scaling by skipping (deletion) can cause aliasing (even if pixel binning is simultaneously enabled), it is generally better to change image size only by cropping and pixel binning.

In context A, the cropped window is defined by variables 0x051 through 0x057. In context B, it is defined by 0x05F and 0x065. In context A and B, the height and width definitions for the output window must be equal to or smaller than the cropped image.

Output width will always be as with no scaling; crop will still be in effect.

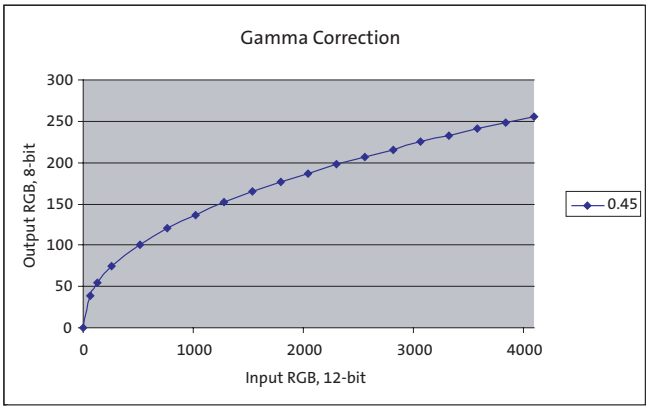
Contrast and Gamma Correction

Gamma correction is independently applied and independently configurable for each of the 12-bit R, G, and B color channels. The gamma correction curve is implemented as a piecewise linear function with 19 knee points, taking 12-bit arguments and mapping them to 8-bit output. The abscissas of the knee points are fixed at 0, 64, 128, 256, 512, 768, 1024, 1280, 1536, 1792, 2048, 2304, 2560, 2816, 3072, 3328, 3584, 3840, and 4096. The 8-bit ordinates are programmable through the IFP registers.

The MT9D112 IFP includes a block for gamma and contrast correction. A custom gamma/contrast correction table may be uploaded, or preset gamma and contrast settings may be selected.

The gamma and contrast correction block uses the following 12-bit input data points to form a piecewise linear transformation curve: 0, 64, 128, 256, 512, 768, 1024, 1280, 1536, 1792, 2048, 2304, 2560, 2816, 3072, 3328, 3584, 3840, and 4096. These input points have been selected to provide more detail to the low end of the curve where gamma correction changes are typically the greatest. These points correspond to 8-bit output values that can be uploaded to the appropriate registers.

Figure 22: Gamma Correction Curve



For simplicity, predefined gamma and contrast tables may be selected, and the MT9D112 automatically combines these tables and upload them to the appropriate gamma correction registers.

The gamma and contrast tables may be selected at mode driver (ID = 7) offsets 109 and 110 (decimal) for context A and context B, respectively. The gamma settings are established at bits 0–2, and the contrast settings are established at bits 4–6.

The gamma setting values are shown in Table 7.

Table 7: Gamma Settings

Gamma Setting	Definition
0	Gamma = 1.0 (no gamma correction)
1	Gamma = 0.56
2	Gamma = 0.45
3	Use user-defined gamma table

S-Curve

The predefined contrast table values have been established by creating an “S” curve with highlight and shadow regions that blend smoothly with a linear midtone region. The slope and value of the highlight and shadow regions match the linear region at these transitions. In addition, the slope of the “S” curve is zero at the top (white) and bottom (black) points. The slope of the linear region determines how much contrast is applied; more contrast corresponds to a higher, midtone linear slope.

Figure 23: Contrast "S" Curve

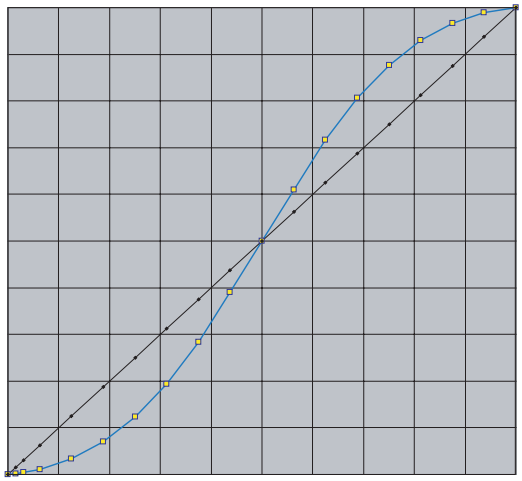


Table 8: Contrast Values

Contrast Setting	Definition
0	No contrast correction
1	Contrast slope = 1.25
2	Contrast slope = 1.50
3	Contrast slope = 1.75
4	Noise reduction contrast

The contrast curve function is applied to the gamma curve points used (whether the gamma curve points are predefined or user-uploaded).

S-curve is a function to correct image pixel values. When applied to pixel values, it typically compresses dark and bright tones, while stretching the midtones.

Special effects like negative image, sepia, or B/W can be applied to the data stream at this point. Special effects are enabled with R0x3348.

RGB to YUV Conversion

For further processing the data is converted from RGB color space to YUV color space.

Color Kill

To remove high light or low light color artifacts, a color kill circuit is included. It affects only pixels whose luminance exceeds a certain preprogrammed threshold. The U and V values of those pixels are attenuated proportionally to the difference between their luminance and the threshold.

YUV Color Filter

As an optional processing step noise suppression by 1-dimensional low-pass filtering of Y and/or UV signals is possible. A 3- or 5-tap filter can be selected for each signal.

Image Scaling

To ensure that the size of images output by the MT9D112 can be tailored to the needs of all users, the IFP includes a scaler module. When enabled, this module performs rescaling of incoming images—shrinks them to arbitrarily selected width and height without reducing the field of view and without discarding any pixel values.

The scaler performs pixel binning—divides each input image into rectangular bins corresponding to individual pixels of the desired output image, averages pixel values in these bins, and assembles the output image from the bin averages. Pixels lying on bin boundaries contribute to more than one bin average: their values are added to bin-wide sums of pixel values with fractional weights. The entire procedure preserves all image information that can be included in the downsized output image and filters out high frequency features that could cause aliasing.

The image cropping and scaler module can be used together to implement a digital zoom and pan. If the scaler is programmed to output images smaller than images coming from the sensor core, zoom effect can be produced by cropping the latter from their maximum size down to the size of the output images. The ratio of these two sizes determines the maximum attainable zoom factor. For example, a 1,600 x 1,200 image rendered on a 160 x 120 display can be zoomed up to 10 times, since 1,600/160 = 1,200/120 = 10. Panning effect can be achieved by fixing the size of the cropping window and moving it around the pixel array.

Due to the loss of sharpness due to pixel binning during image scaling, 1D aperture correction may be applied to increase image sharpness lost due to pixel binning during image scaling.

YUV-to-RGB/YUV Conversion and Output Formatting

The YUV data stream emerging from the scaler module can either exit the color pipeline as-is or be converted before exit to an alternative YUV or RGB data format as selected by R0x332E.

Color Conversion Formulas

Y'U'V'

This conversion is BT 601 scaled to make YUV range from 0 through 255. This setting is recommended for JPEG encoding and is the most popular, although it is not well defined and is often misused in Windows.

$$Y' = 0.299\ R' + 0.587\ G' + 0.114\ B'$$

$$U' = 0.564\ (B' - Y') + 128$$

$$V' = 0.713\ (R' - Y') + 128$$

There is an option where 128 is not added to U'V'.

Y'Cb'Cr' Using sRGB Formulas

The MT9D112 implements the sRGB standard. This option provides YCbCr coefficients for a correct 4:2:2 transmission. Note that 16 < Y60 1< 235, 16 < Cb, Cr < 240 and 0 < = RGB < = 255.

Y' = (0.2126*R' + 0.7152*G' + 0.0722*B')*219/256 + 16

Cb' = 0.5389*(B '- Y')*224/256 + 128

Cr' = 0.635*(R' - Y')*224/256 + 128

Y'U'V' Using sRGB Formulas

Similar to the previous set of formulas, but has YUV spanning a range of 0 through 255.

Y' = 0.2126*R' + 0.7152*G' + 0.0722*B'

U' = 0.5389*(B' - Y') + 128 = -0.1146*R - 0.3854*G + 0.5*B

V' = 0.635*(R' - Y') + 128 = 0.5*R - 0.4542*G - 0.0458*B

There is an option to disable adding 128 to U'V'. The reverse transform is as follows:

R' = Y + 1.5748*V

G' = Y - 0.1873*(U - 128) - 0.4681*(V - 128)

B' = Y + 1.8556*(U - 128)

Output Interface

Parallel and MIPI Output

The user can select to either use the serial MIPI output or the 8-bit parallel output to transmit the data. Only one of the output modes can be used at any time.

The parallel output, enabled by R0x301A[7], can be used with an output FIFO whose memory is shared with the MIPI output FIFO to retain a constant pixel output clock independent from the scaling factor.

When scaling the image or skipping lines, the data would be generated in bursts and the pixel clock would turn on and off in intervals, which might lead to EMI problems. The output FIFO will group all active pixel data together so the pixel clock can be run at a constant speed. The output FIFO is enabled by setting R0x3212[1:0] to 10. There are two registers to configure the output FIFO, watermark and line length.

The MIPI output transmitter implements a fully configurable serial differential subLVDS transmitter capable of up to 640 Mb/s. It supports multiple formats, error checking, and custom short packets.

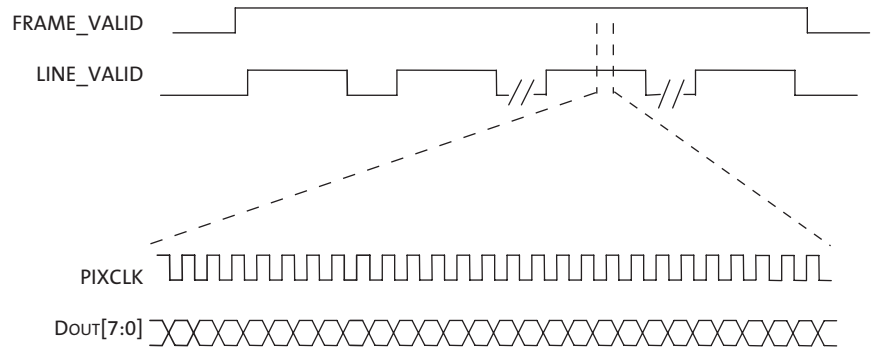
Output Format and Timing

YUV/RGB Uncompressed Output

Uncompressed YUV or RGB data can be output either directly from the output formatting block or through a FIFO buffer with a capacity of 800 bytes, enough to hold one-fourth of an uncompressed line at full resolution. Buffering of data is a way to equalize the data output rate when image scaling is used. Scaling produces an intermittent data stream consisting of short high-rate bursts separated by idle periods. High pixel clock frequency during bursts may be undesirable due to EMI concerns.

Figure 24 on page 33 depicts the output timing of uncompressed YUV/RGB when a decimated data stream is equalized by buffering or when no scaling takes place. The pixel clock frequency remains constant during each LINE_VALID HIGH period. Decimated data are output at a lower frequency than full size frames, which helps to reduce EMI.

Figure 24: Timing of Uncompressed Full Frame Output or Decimated Output Passing Through the FIFO



Uncompressed YUV/RGB Data Ordering

The MT9D112 supports swapping YCrCb mode, as illustrated in Table 9.

Table 9: YCrCb Output Data Ordering

Mode				
Default (no swap)	Cb_i	Y_i	Cr_i	Y_{i+1}
Swapped CrCb	Cr_i	Y_i	Cb_i	Y_{i+1}
Swapped YC	Y_i	Cb_i	Y_{i+1}	Cr_i
Swapped CrCb, YC	Y_i	Cr_i	Y_{i+1}	Cb_i

The RGB output data ordering in default mode is shown in Table 10. The odd and even bytes are swapped when luma/chroma swap is enabled. R and B channels are bit-wise swapped when chroma swap is enabled.

Table 10: RGB Ordering in Default Mode

Mode (Swap Disabled)	Byte	$D_7D_6D_5D_4D_3D_2D_1D_0$
RGB 565	Odd	$R_7R_6R_5R_4R_3G_7G_6G_5$
	Even	$G_4G_3G_2B_7B_6B_5B_4B_3$
RGB 555	Odd	$0\ R_7R_6R_5R_4R_3G_7G_6$
	Even	$G_4G_3G_2B_7B_6B_5B_4B_3$
RGB 444x	Odd	$R_7R_6R_5R_4G_7G_6G_5G_4$
	Even	$B_7B_6B_5B_4\ 0\ 0\ 0\ 0$
RGB x444	Odd	$0\ 0\ 0\ 0\ R_7R6_6R_5R_4$
	Even	$G_7G_6G_5G_4B_7B_6B_5B_4$

Uncompressed 10-Bit Bypass Output

Raw 10-bit Bayer data from the sensor core can be output in bypass mode in two ways:

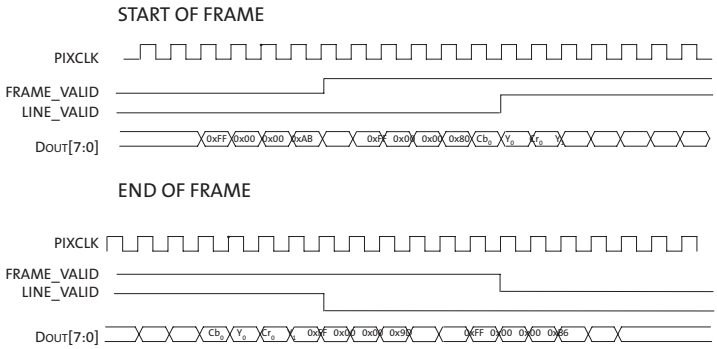
1. Using 8 data output pads (DOUT0-DOUT7), and GPIO[0:1].
2. Using only 8 pads (DOUT0-DOUT7) and a special 8 + 2 data format, shown in Table 11.

The timing of 10-bit or 8-bit data stream output in the bypass mode is qualitatively the same as that depicted in Figure 25 on page 34.

Table 11: 2-Byte RGB Format

Odd bytes	8 data bits	$D_9D_8D_7D_6D_5D_4D_3D_2$
Even bytes	2 data bits + 6 unused bits	0 0 0 0 0 0 D_1D_0

Figure 25: Example of Timing for Non-Decimated Uncompressed Output Bypassing Output FIFO



FIFO

During normal pipeline operation the output data rate is determined by a number of factors, for instance, input image size, degree of scaling, and sensor operation mode. As these parameters change during normal sensor operation, output frequency changes. This output frequency may generate RF noise, interfering with the mobile device. By using an output FIFO to maintain a constant output clock frequency, noise is easily filtered out.

The FIFO accumulates data and after a certain number of bytes are stored, it will yield them in a single burst making sure that data rate within the burst remains constant. This approach may utilize a free running clock thus minimizes possible RF interference.

In the default operation the mode user must provide two values for proper FIFO operation:

1. the expected line length (R0x3220)
2. watermark (R0x321E)

The value for both has to be specified in bytes. Calculation of line length is straightforward; multiply the number of pixels by the number of bytes per pixel. The number of bytes per pixel is 2 for all output modes except processed Bayer in which it equals one byte per pixel.

Watermark

Calculation of the watermark should be done based on the maximum number of bytes that the pipeline could output in the given mode (with activated clip and no scaling) and the number of bytes it will yield, taking into account scaling. Also, accounting for the horizontal sensor binning mode if it is activated. In this mode, the number of pixels that the pipeline can output should be doubled.

Camera Control

General Purpose I/Os

The four general purpose I/Os of the MT9D112 can be configured in multiple ways. Each of the I/Os can be used as a simple input/output that can be programmed from the host. The status of the GPIO is read at power up and can be used as a module ID to separate different module suppliers.

If 10-bit RAW output is required, GPIO[1:0] can be configured as bit 0 and bit 1 of a 10-bit data bus.

GPIO[3:2] can be configured to output a flash pulse to trigger an external Xenon or LED flash or a shutter pulse to control an external shutter.

The general purpose inputs are enabled by setting R0x301A[8]. Once enabled, all four inputs must be driven to valid logic levels by external signals. The state of the general purpose inputs can be read through R0x3026[3:0].
In addition, each of the following functions can be associated with none, one, or more of the general purpose inputs so that the function can be directly controlled by a hardware input:

- Output enable
- Trigger
- Standby functions

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

Output Enable Control

When the parallel pixel data interface is enabled, its signals can be switched asynchronously between the driven and High-Z under pin or register control, as shown in Table 12.

Table 12: Output Enable Control

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

Trigger Control

When the global reset feature is in use, the trigger for the sequence can be initiated either under pin or register control, as shown in Table 13.

Table 13: Trigger Control

TRIGGER	Global Trigger R0x3060-1[0]	Description
Disabled	0	Idle
Disabled	1	Trigger
0	0	Idle
X	1	Trigger
1	X	Trigger

Streaming/Standby Control

The MT9D112 can be switched between its soft standby and streaming states under pin or register control, as shown in Table 14.

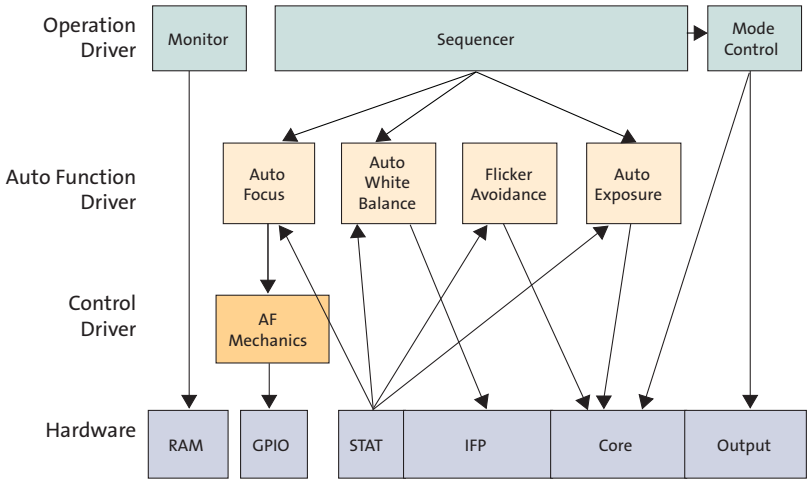
Table 14: Streaming/STANDBY

STANDBY	Streaming R0x301A[2]	Description
Disabled	0	Soft Standby
Disabled	1	Streaming
X	0	Soft Standby
0	1	Streaming
1	X	Soft Standby

Firmware Architecture

The firmware for the MT9D112 is implemented in multiple drivers that are responsible for different parts of operation. All drivers are executed on the 68HC11 microcontroller from the 32kB ROM and use the 2kB RAM to store variables and firmware updates. The firmware can access registers in the sensor core and the IFP independently from the user through an internal bus. The GPIO interface is accessible through special function registers (SFRs) in memory space.

Figure 26: Software Architecture



Sequencer

The sequencer is responsible for coordinating all events triggered by the user. It is implemented as a state machine. For example, sending a capture command to the sequencer will change the resolution from preview to full resolution, settle the AWB/AE, turn on or off an external LED and switch back to preview after capturing the frame. The setup of the sensor can be defined by the user for the following states:

- on entering preview
- in preview
- when leaving preview
- when entering capture

Context and Operational Modes

The MT9D112 can operate in several modes including preview, still capture (snapshot), and video. All modes of operation are individually configurable and are organized as two contexts-context A and context B. A context is defined by sensor image size, frame rate, resolution and other associated parameters. The user can switch between the two contexts by sending a command via the two-wire serial interface.

Preview Mode

Context A is primarily intended for use in the preview mode. During preview, the sensor usually outputs low resolution images at a relatively high frame rate, and its power consumption is kept to a minimum.

Still Capture and Video Modes

Context B can be configured for the still capture or video mode, as required by the user. For still capture configuration, the user typically specifies the desired output image size, if flash should be enabled, how many frames to capture. For video, the user might select a different image size and a fixed frame rate.

Snapshot and Flash

To take a snapshot, the user must send a command that changes the context from A to B. Typical sequence of events after this command are:

- First, the camera may turn on its LED flash, if it has one and is required to use it. With the flash on, the camera exposure and white balance are automatically adjusted to the changed illumination of the scene.
- Next, the camera performs auto focusing. Once in focus, it captures one or more frames of desired size. A camera equipped with a xenon flash strobes it during the capture. Completing the sequence, the camera automatically returns to context A and resumes running preview.

Note: This sequence of events can take up to 10 frames or more depending on AF hardware capability.

Video

To start video capture, the user must change relevant context B settings, such as capture mode, image size and frame rate, and again send a context change command. Upon receiving it, the MT9D112 switches to the modified context B settings, while continuing to output YUV-encoded image data. Auto exposure and auto focus automatically continues smooth operation. To exit the video capture mode, the user has to send another context change command causing the sensor to be switched back to context A.

Auto Exposure

The auto exposure algorithm performs automatic adjustments of the image brightness by controlling exposure time and analog gains of the sensor core as well as digital gains applied to the image.

Auto exposure is implemented by a firmware driver that analyzes image statistics collected by the exposure measurement engine, makes a decision and programs the sensor core and color pipeline to achieve the desired exposure. The measurement engine subdivides the image into 16 windows organized as a 4 x 4 grid.

Two auto exposure algorithm modes are available:

1. preview (luminance-based)
2. evaluative (histogram-based), known as MDR (Maximum Dynamic Range)

Preview Mode

This exposure mode is activated during preview or video capture. It relies on the statistics engine that tracks speed and amplitude of the change of the overall luminance in the selected windows of the image.

Backlight compensation is achieved by weighting the luminance in the center of the image higher than the luminance on the periphery. Other algorithm features include the rejection of fast fluctuations in illumination (time averaging), control of speed of response, and control of the sensitivity to small changes. While the default settings are adequate in most situations, the user can program target brightness, measurement window, and other parameters described above.

In preview AE, the driver calculates image brightness based on average luma values received from 16 programmable equal-size rectangular windows forming a 4 x 4 grid. In preview mode, 16 windows are combined in 2 segments: central and peripheral. The central segment includes four central windows. All remaining windows belong to the peripheral segment. Scene brightness is calculated as average luma in each segment taken with certain weights. Variable ae.weights[3:0] specifies the central zone weight, ae.weights[7:4] specifies the peripheral zone weight.

The driver changes AE parameters (IT, Gains, and so on) to drive brightness (ae.CurrentY) to a programmable target (ae.Target). The magnitude of each single step used to approach the target is defined by the ae.JumpDivisor variable.

Expected brightness is:

Ynew = ae.CurrentY+(ae.Target-ae.CurrentY)/ ae.JumpDivisor.

To avoid unwanted reactions of AE to small fluctuations of scene brightness or momentary scene changes, the AE driver uses a temporal filter for luma and gates around the AE luma target. The driver changes AE parameters only if the buffered luma outsteps AE target gates.

Variable *ae.lumaBufferSpeed* defines the buffering level:

*32*Ybuf1 = Ybuf0*(32-ae.lumaBufferSpeed)+Ycurr* ae.lumaBufferSpeed*

Values *ae.lumaBufferSpeed = 32* and *ae.JumpDivisor = 1* specify maximal AE speed.

Evaluative Algorithm (MDR)

A scene-evaluative AE algorithm is available for use in snapshot mode. The algorithm performs scene analysis and classification with respect to its brightness, contrast, and composure and then decides to increase, decrease, or keep the original exposure target. It makes most difference for backlight and bright outdoor conditions.

Accelerated Settling During Overexposure

The AE speed is direction-dependent. Transitioning from over saturation to the target can take more time than transitioning from under-saturation. The AE driver has a mode which speeds up AE for over exposed scenes (ae.status[7]).

The AE driver counts the number of AE windows (driver variable ae.numOE) which have average brightness equal to or greater than some value, 250 by default. For a scene having saturated regions, the average luma is underestimated due to signal clipping. The driver compensates underestimation by a factor defined by the ae.numOE variable.

currentY = ae.CurrentY * coeffOE[ae.numOE] / 16;

where:

const BYTE coeffOE[17] = {16,17, 18, 19, 20, 21, 22, 24, 26, 28, 30, 32, 36, 40, 48, 54, 64};

Exposure Control

To achieve the required amount of exposure, the AE driver adjusts the sensor integration time R0x3012, R0x3018, gains, ADC reference, and IFP digital gains. To reject flicker, integration time is typically adjusted in increments of ae.R9_step. ae.R9_step specifies duration in row times equal to one flicker period. Thus, flicker is rejected if integration time is kept a natural factor of the flicker period.

Exposure is adjusted differently depending on illumination situation.

- In extremely bright conditions, the exposure is set using R0x3018, R0x3012, and analog gains.
- R0x3018 is used to achieve very short integration times.

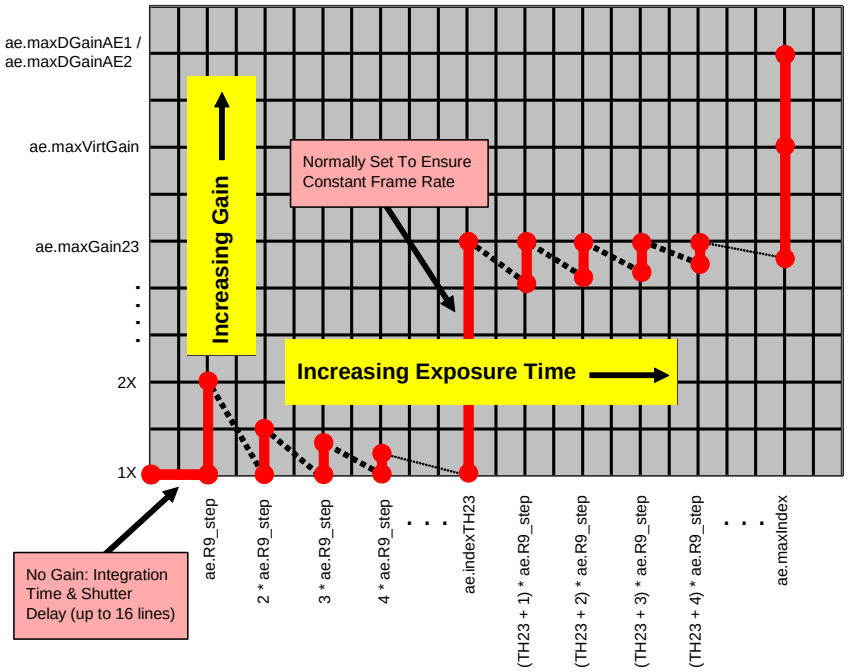
In this situation,

- R0x3012<ae.R9_step and flicker is not rejected.
- In bright conditions where R0x3012>=ae.R9_step R0x3012 is set as a natural factor of ae.R9_step. Analog gains are also used, but the green gain, also called virtual gain, does not exceed 2x. ae.minVirtGain limits minimal integration time and is expressed in flicker periods. ae.Index indicates the current integration time expressed in the same form.
- Under medium-intensity illumination, the integration time can increase further. For any given exposure, the best signal-to-noise ratio can be typically obtained by using the longest exposure and the smallest gain setting. However, a long exposure time can slow down the output frame rate if the former exceeds the default frame rate, R0x3012 >R0x3006 + R0x300C + 1. Integration ae.IndexTH23 specifies the breakpoint where the AE scheme, giving preference to increasing the shutter width, is replaced with another scheme giving preference to an increase in gain. ae.maxGain23 specifies maximum allowed gain in this situation. ae.VirtGain indicates current green channel gain.
- In darker situations, the gain reaches ae.maxGain23 and the integration time is allowed to increase again up to ae.maxIndex.

- In yet darker situations, once the integration time reaches ae.maxIndex, the analog gain is allowed to increase up to *ae.maxVirtGain*.
- In very dark conditions, the digital IFP gains are allowed to increase up to *ae.maxDGainAE1* and *ae.maxDGainAE2*.

ADC is used as an additional gain stage by adjusting reference levels. See Table 33 on page 50 for the ae.ADC* variables.

Figure 27: Gain vs. Exposure



AE MDR Mode

The MT9D112 also has an AE MDR mode to extend the dynamic range of a scene after AE has settled brightness. This mode can only be selected in PreviewEnter, PreviewLeave, and CaptureEnter states.

To set this mode, the following registers must be set:

- seq.previewParEnter.ae = 4
- seq.previewParLeave.ae = 4
- seq.capParEnter.ae = 4

Auto White Balance

The MT9D112 has a built-in auto white balance algorithm designed to compensate for the effects of changing spectra of the scene illumination on the quality of the color rendition. The algorithm consists of two major parts: a measurement engine performing statistical analysis of the image and a driver performing the selection of the optimal color correction matrix, digital, and sensor core analog gains. While default settings of these algorithms are adequate in most situations, the user can reprogram base color correction matrices, place limits on color channel gains, and control the speed of both matrix and gain adjustments.

The MT9D112 AWB does not require the presence of gray or white elements in the image for good color rendition. The AWB does not attempt to locate the brightest or grayest element of the image but instead performs statistical image analysis to differentiate between changes in predominant spectra of illumination and changes in predominant colors of the scene.

The MT9D112 also includes a module to detect edges based on hue variation which allows high frequency patterns of the same color to be excluded from the AWB calculation.

Flicker Detection

Flicker occurs when the integration time is not an integer multiple of the period of the light intensity. The automatic flicker detection block does not compensate for the flicker, but rather avoids it by detecting the flicker frequency and adjusting the integration time. For integration times below the light intensity period (10ms for 50Hz environments), flicker cannot be avoided.

Flicker shows as horizontal bars rolling up or down. The MCU looks for these rolling bars using a thin horizontal window, which outputs luma average and is applied to 48 points in the upper half of the image. The MCU repeats the same sampling on the next frame and subtracts 48 samples from the previous frame from corresponding samples from the current frame. fd.skipFrame allows skipping more frames between subtraction. The MCU then smooths the 48 sampling points, see fd.smooth_cnt, and applies an amplitude threshold fl.minAmplitude to avoid false detection then looks at the resulting waveform. If flicker is present, the waveform should have a frequency within the search range, see fd.search_f1_50, fd.search_f2_50, fd.search_f1_60 and fd.search_f2_60. Assuming the flicker power is a sine wave, subtracting two frames results in

$$sin(wt)-sin(wt+a)=2sin(a/2)cos(wt+a/2)$$

which is a cosine wave of the original frequency.

Auto Focus

Algorithm

The auto focus algorithm implemented in the MT9D112 firmware seeks to maximize sharpness of vertical lines in images output by the sensor, by guiding an external lens actuator to the position of best lens focus. The algorithm is actuator-independent, providing guidance by means of an abstract 1-dimensional position variable, leaving the translation of its changes into physical lens movements to a separate AF mechanics (AFM) driver.

For measuring line sharpness, the AF algorithm relies on the focus measurement engine in the color pipeline, which is a programmable vertical-edge-filtering module. The module convolves two preprogrammed 1-dimensional digital filters with luminance (Y) data it receives row by row from the color interpolation module. In every interpolated image, the pixels whose Y values are used in the convolution form a rectangular block that can be arbitrarily positioned and sized, and in addition divided into up to 16 equal-size sub-blocks, referred to as AF windows or zones. The absolute values of convolution results are summed separately for each filter over each of the AF windows, yielding up to 32 sums per frame.

There are several motion sequences through which the MT9D112 AF algorithm can bring a lens to best focus position. All these sequences begin with a jump to a preselected start position, for example, the infinity focus position. This jump is referred to as the first flyback. It is followed by a unidirectional series of steps that puts the lens at up to 19 preselected positions different from the start position. This series of steps is called the first scan.

During the scan, the AF algorithm stops the lens at each preselected position long enough to obtain valid sharpness scores. The normalized score for each AF window is stored along with information on how many zones had a high sharpness score and the position with the maximum sharpness score is determined taking into account the zone information. This way the algorithm can handle scenes with objects at different distances.

After the first scan the AF algorithm provides a number of ways to proceed with final lens positioning. The user should select a way that best fits the magnitude of lens actuator hysteresis and desired lens proximity to the truly optimal position. Actuators with large, unknown or variable hysteresis should do a second flyback and either jump or retrace the steps of the first scan to the best scanned position. Actuators with constant hysteresis (like stepper motors) can be moved to that position directly from the end position of the scan-the AF algorithm offers an option to automatically increase the length of this move by a preprogrammed backlash-compensating step. Finally, if the first scan is coarse relative to the positioning precision of the lens actuator and depth of field of the lens, an optional second fine scan can be performed around the lens position voted best after the first scan. This second scan is done in the same way as the first, except that the positions it covers are not preselected. Instead, the AF algorithm user must set step size and number of steps for the second scan. The second scan must be followed by the same hysteresis-matching motion sequence as the first scan, for example, a third flyback and jump to the best position.

The AF driver is disabled on power-up. If AF is desired, it must be explicitly enabled after every power up or reset before it can be used. This can be accomplished by establishing AF motor type:

To establish AF motor type:

- Set afm.type (0x002) = 129 (for helimorph AF)
  afm.type (0x002) =130 (for stepper motor AF)
  afm.type (0x002) =131 (for AD5398)
- Turn on AF in preview mode seq.previewPar.af = 1
- Call “refresh” sequencer command seq.cmd = 5

Modes

There are four AF camera modes that the MT9D112 can fully support if it controls the position of the camera lens.

1. Snapshot mode
   In this mode, a camera performs auto focusing upon a user command to do so. When the auto focusing is finished, a snapshot is normally taken and there is no further AF activity until the next appropriate user command. The MT9D112 can do the auto focusing using its own AF algorithm described above or a substitute algorithm loaded into RAM. It can then wait or automatically proceed with other operations required to take a snapshot.
2. Locked mode
   The MT9D112 can be commanded to lock the lens in its current position. Between the command to lock the lens and another to release it, the lens does not respond to other commands or scene changes.
3. Focus-free mode
   In many situations, for example, under low light or during video recording, it may be impossible or undesirable to focus the lens prior to every image capture. Instead, the lens can be locked in a position most likely to produce satisfactory images, for example, the hyperfocal position. This position can be programmed into the MT9D112, and it can move and hold the lens there on command.
4. Manual mode
   In this mode there is no AF activity-focusing the camera is left to the user. The user typically can move the camera lens in steps, by manually issuing commands to the lens actuator, and observe the effect of his actions on a preview display. The MT9D112 can provide 30 fps image input for the display and simultaneously translate user commands received via two-wire serial interface into digital waveforms driving the lens actuator.

Lens Actuator Interface

Actuators used to move lenses in AF cameras can be classified into several broad categories that differ significantly in their requirements for driving signals. These requirements also vary from one device to another within each category. To ensure its compatibility with many different actuators, the MT9D112 includes a general purpose input/output module (GPIO_AF).

The GPIO is a programmable rectangular waveform generator, with 8 individually controllable output pads (GPIO0 through GPIO7), a separate power supply pad (VDD_AF), and a separate clock domain that can be disconnected from the master clock to save power when the GPIO is not in use. The GPIO can toggle its output pads as fast as half the master clock frequency.

An external host processor or the embedded microcontroller of the MT9D112 have two ways to control the voltages on the GPIO output pads:

1. Setting or clearing bits in a control register
   The state of the GPIO pads is updated immediately after writing to the register. Since writing via the two-wire serial interface takes some time, this way does not give the host processor a very precise control over GPIO output timing.
2. Waveform programming
   The second way to obtain a desired output from the GPIO is to program a set of periodic waveforms to the control registers and initialize their generation. The GPIO then generates the programmed waveforms on its own, without waiting for any further input, and therefore with the best attainable timing precision. If necessary, the GPIO can notify the MCU and the host processor about reaching certain points in the waveforms generation, for example, the end of a particular waveform. Every GPIO notification has two components: the GPIO sends a wakeup signal to the MCU and sets a bit in its status register that can be polled by the MCU and/or the host processor. The wakeup signals have an effect only when the MCU is in sleep mode.

The MT9D112 can be set up not only to output digital signals to a lens actuator and/or other similar devices, but also to receive their digital feedback. All GPIO output pads are reconfigurable as high-impedance digital inputs. The logical state of each GPIO pad is mirrored by the state of a bit in a dedicated register, which allows the MCU and host processor to sample digital input signals at intervals equal to their respective register read times.

Two-Wire Serial Interface

The two-wire serial interface bus enables read/write access to control and status registers within the MT9D112.The interface protocol uses a master/slave model in which a master controls one or more slave devices. The sensor acts as a slave device. The master generates a clock (SCLK) that is an input to the MT9D112 and used to synchronize transfers.

Data is transferred between the master and the slave on a bidirectional signal (SDATA). SDATA is pulled up to VDD_IO off-chip by a 1.5KΩ resistor. Either the slave or master device can drive SDATA LOW—the interface protocol determines which device is allowed to drive SDATA at any given time.

Protocol

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

- a (repeated) start condition
- a slave address/data direction byte
- a 16-bit register address
- an (a no) acknowledge bit
- two data bytes
- 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, and this is known as a “repeated start” or “restart” condition.

Stop Condition

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

Data Transfer

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

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

Slave Address/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 MT9D112 are 0x78 (write address) and 0x79 (read address). Alternate slave addresses of 0x7A (write address) and 0x7B (read address) can be selected by asserting the SADDR input signal.

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 two-wire serial interface 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.

Typical Serial Transfer

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

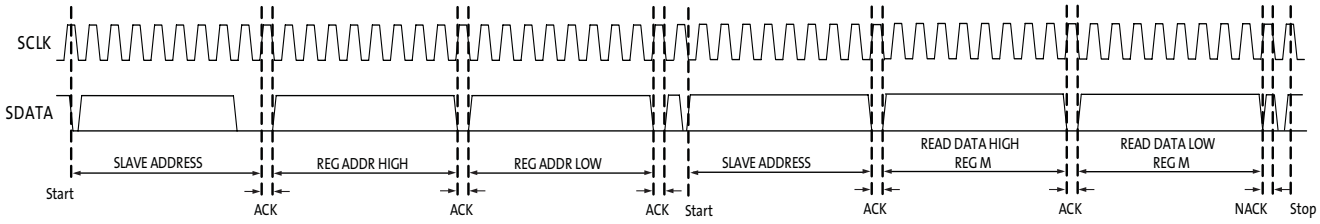
If the request was a write, the master then transfers the 16-bit register address to which a write should take place. This transfer takes place as two 8-bit sequences and the slave sends an acknowledge bit after each sequence to indicate that the byte has been received. The master then transfers the data as an 8-bit sequence; the slave sends acknowledge bit at the end of the sequence. After 8 bits have been transferred, the slave’s internal register address is incremented automatically, so that the next 8 bits are written to the next register address. The master stops writing by generating a (re)start or stop condition.

If the request was a read, the master sends the 8-bit write slave address/data direction byte and 16-bit register address, 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 auto-incremented after every 8 bits are transferred. The data transfer is stopped when the master sends a no-acknowledge bit.

Single Read from Random Location

This sequence (Figure 28) 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 two bytes of register data. The master terminates the read by generating a no-acknowledge bit followed by a stop condition.

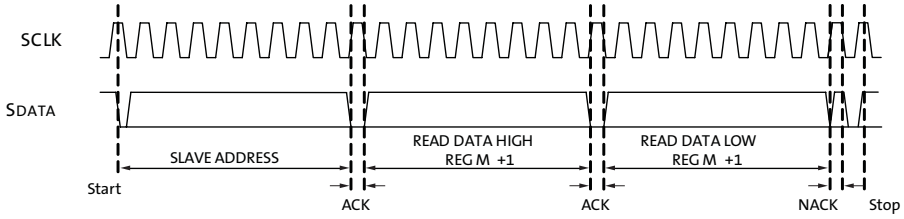
Figure 28: Single Read from Random Location



Single Read from Current Location

This sequence (Figure 29) performs a read using the current value of the MT9D112 internal register address. The master terminates the read by generating a no-acknowledge bit followed by a stop condition.

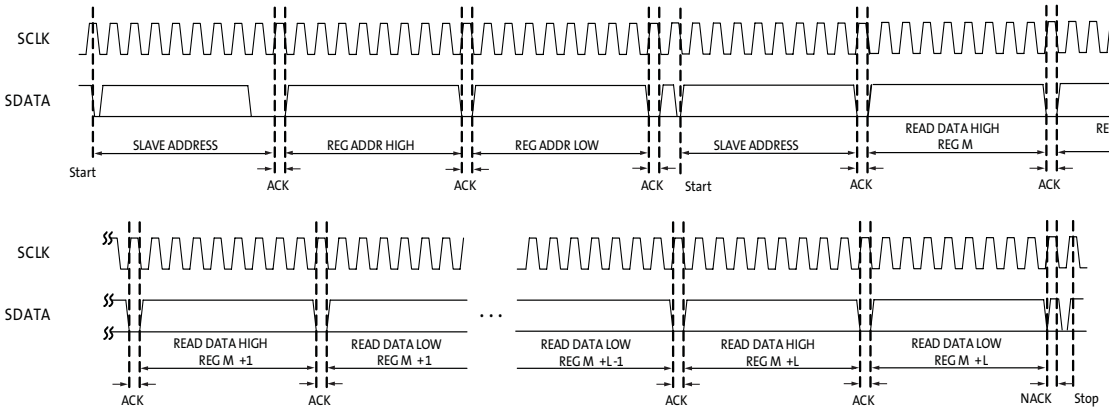
Figure 29: Single Read from Current Location



Sequential Read, Start from Random Location

This sequence (Figure 30) starts in the same way as the single read from random location (Figure 28 on page 48). Instead of generating a no-acknowledge bit after the first two bytes of data have been transferred, the master generates an acknowledge bit, and continues to perform byte reads until "L" registers have been read.

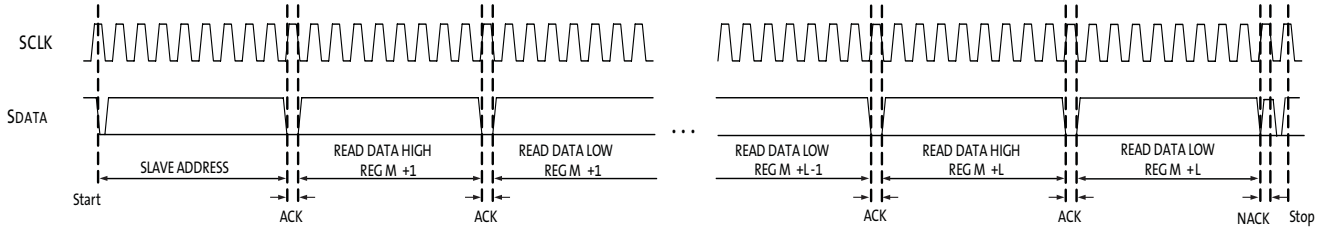
Figure 30: Sequential Read, Start from Random Location



Sequential Read, Start from Current Location

This sequence (Figure 31 on page 49) starts in the same way as the single read from current location (Figure 29). Instead of generating a no-acknowledge bit after the first two bytes of data have been transferred, the master generates an acknowledge bit, and continues to perform byte reads until "L" registers have been read.

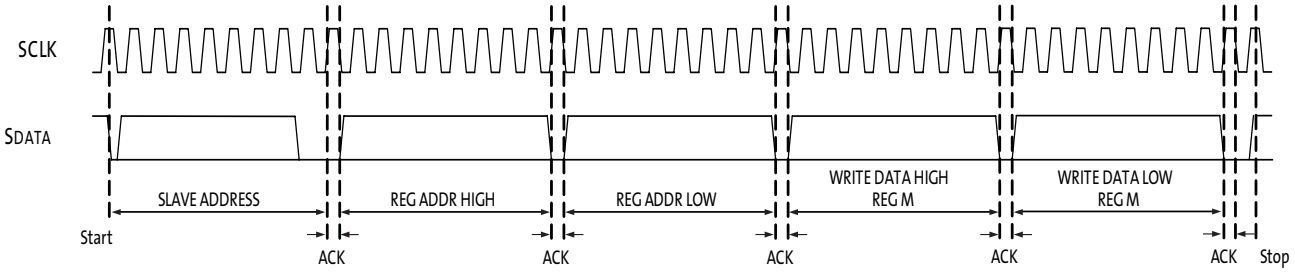
Figure 31: Sequential Read, Start from Current Location



Single Write to Random Location

This sequence (Figure 32) 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 two bytes of write data. The write is terminated by the master generating a stop condition.

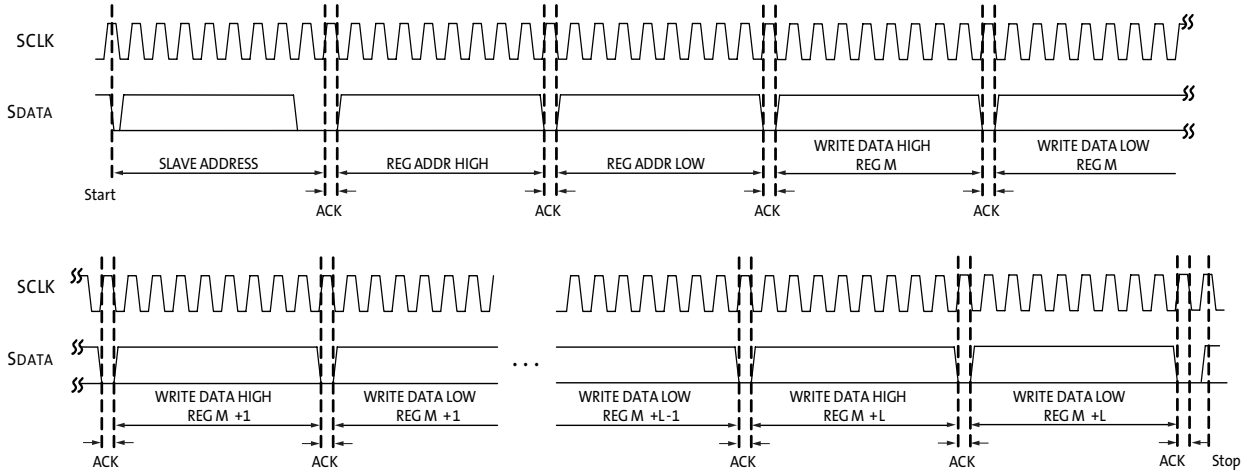
Figure 32: Single Write to Random Location



Sequential Write, Start at Random Location

This sequence (Figure 33) starts in the same way as the single write to random location (Figure 32 on page 50). Instead of generating a no-acknowledge bit after two bytes of data have been transferred, the master generates an acknowledge bit, and continues to perform byte writes until "L" registers have been written. The write is terminated by the master generating a stop condition.

Figure 33: Sequential Write, Start at Random Location



Registers and Variables

Four types of configuration controls are available:

1. Hardware registers
2. Driver variables
3. Special function registers
4. MCU SRAM

The following convention is used in the text below to designate registers and variables:

R0x3012, R0x3012[3:0] or R12306, R12306[3:0]

These refer to two-wire accessible register number 12306. [3:0] indicate bits. Registers numbers range 0..65535 and bits range 15..0. Not all register numbers are used.

- ae.Target
  This refers to variable “Target” in the AE driver.
- SFR 0x1080 or SRAM 0x0400
  This refers to special function register or SRAM located at address 0x1080 in MCU memory space.

How to Access Registers and Variables

Registers, variables, and SFRs are accessed in different ways.

Registers

Hardware registers are grouped internally by pages.

R0x3000 – R0x31FE = Sensor Core Page 0 (sensor controls)

R0x3200 – R0x33FE = SoC Page 1 (color pipeline controls)

R0X3400 – R0x35BE = SoC Page 2 (color pipeline, MIPI, output FIFO)

Not all the registers in each page are used, not all the bits in each register might be used. Registers are accessed by serial interface reads and writes consisting of a 16-bit address and 16-bit data.

Variables

Variables are located in the microcontroller RAM memory. Each driver, such as auto exposure, white balance, and auto focus, has a unique driver ID (0..31) and a set of public variables organized as a structure. Each variable in this structure is uniquely identified by its offset from the top of the structure and its size. The size can be 1 or 2 bytes, while the offset is 1 byte.

All driver variables (public and private) can be accessed via R0x338C and R0x3390. While two access modes are available-access by physical address and by logical address, the public variables are typically accessed by the logical method. The logical address, which is set in R0x338C, consists of a 5-bit driver ID number and a variable offset. Examples are provided below.

To set the variable ae.Target=50:

- The variable has a driver ID of 2. Therefore, set R0x338C[12:8]=2.
- The variable has an offset of 6. Therefore, set R0x338C[7:0]=6.
- This is a logical access. Therefore, set R0x338C[14:13]=01.
- The size of the variable is 8 bits. Therefore, set R0x338C[15]=1.

- By combining these bits, R0x338C=0xA206.
- Set R0x3390=50 for the value of the variable.

To read the variable ae.Target:
- Since this is the same variable as the above example, R0x338C=0xA206.
- Read R0x3390 for the current variable value.

To set the variable mon.arg1=0x1234:
- The variable has a driver ID of 0. Therefore, set R338C[12:8]=0.
- The variable has an offset of 3. Therefore, set R0x3390[7:0]=3.
- This is a logical access. Therefore, set R0x338C[14:13]=01.
- The size of the variable is 16 bits. Therefore, set R0x338C[15]=0.
- By combining these bits, R0x338C=0x2003.
- Set R0x3390=0x1234 for the value of the variable.

To read the variable mon.arg1:
- Since this is the same variable as the above example, R0x338C=0x2003.
- Read R0x3390 for the current variable value.

Special Function Registers and MCU SRAM

Special function registers are registers connected to the local bus of the MCU. These registers include GPIO, waveform generator, and those important for firmware operation. Special function registers are accessed by physical address.

MCU SRAM consists of 1K system memory and 1K user memory. Examples of access:
- Write into user SRAM. Use to upload code
  a. R0x338C = 0x8400 // address
  b. R0x3390 = 0x1234 // write 16-bit value
- Read from user SRAM
  c. R0x338C = 0x8400 // address
  d. Read R0x3390 // read 16-bit value

GPIO access example for writing 0x00FE to GPIO R0x1078:
- Write to 8-bit GPIO register
  a. R0x338C = 0x9078 // GPIO direction at 0x1078
  b. R0x3390 = 0x00FE // Configure GPIO[0] as output
- Read from 8-bit GPIO register
  c. R0x338C = 0x9078 // GPIO direction at 0x1078
  d. Read R0x3390 // Check GPIO[7:0] pad state

Note: 8-bit access: R0x338C[15] = 1, R0x338C[14:0] = GPIO register
16-bit access: R0x338C[15] = 0, R0x338C[14:0] = GPIO register

Core Registers

Double-buffered Registers

Some sensor settings cannot be changed during frame readout. For example, changing x_addr_end R0x3008 part way through frame readout results in inconsistent LINE_VALID behavior. To avoid this, the sensor core 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. By default, this occurs 10 row times before FRAME_VALID goes HIGH.

R0x301A[15] can be used to inhibit transfers from the pending to the live registers. This control should be used when making many register changes that must take effect simultaneously.

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 changed during the frame.

Many changes to the sensor register settings can cause a bad frame. For example, when x_addr_end R0x3008 is changed, the new register value does not affect sensor behavior until the next frame starts. However, the frame that would be read out at that frame-start has been integrated using the old row width. Consequently, reading it out using the new row width results in a frame with an incorrect integration time.

By default, most bad frames are masked: LINE_VALID and FRAME_VALID are inhibited for these frames so that the vertical blanking time between frames is extended by the frame time. This feature can be disabled with R0x301A[10].

Changes to Integration Time

If the integration time is changed while FRAME_VALID 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. 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 on the new value of the integration time.
3. When frame (*n* + 2) is read out, it is integrated using the new integration time. If the integration time is changed on successive frames, each value written is applied to a single frame; the latency between writing a value and it affecting the frame readout remains at two frames.

Changes to Gain Settings

When the gain settings are changed, the gain is usually updated on the next frame start. When the integration time and the gain are changed simultaneously, 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.

If the gain and integration time are both changed on successive frames, some gain value is overwritten without being applied, while each integration time is used for one single frame.

Timing Specifications

Power-up Sequence

Aptina recommends starting the power-up sequence with VDD. Powering down the sensor requires power supplies to be removed in the reverse order. For applications that require complete power down of the MT9D112 when not in use, refer to TN-09-120 for more details on how to perform power removal and application.

The sensor includes a power-on reset feature that initiates a reset upon power up. Unless the initial VDD ramp occurs within 5µs, it is advised that the user still manually assert a hard reset upon power up.

Figure 34: Internal Power-On Reset

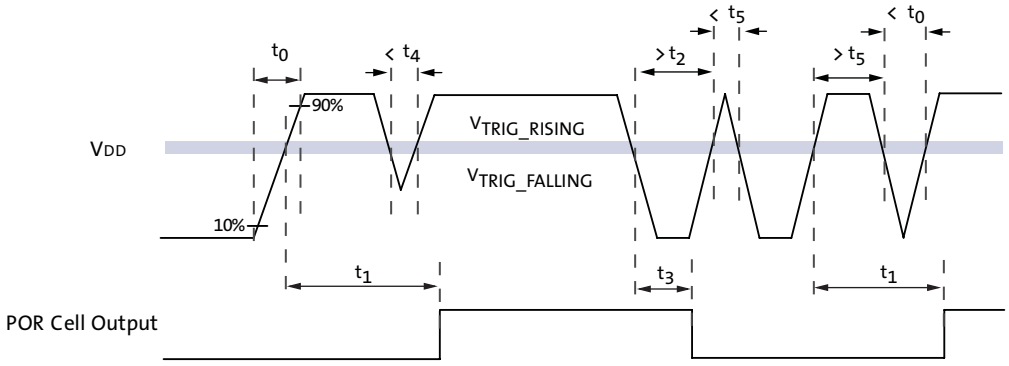


Table 15: POR Parameters

Symbol	Parameter	Min	Typ	Max	Unit
t_0	VDD ramp time	–	–	10	µs
t_1	VDD rising, crossing VTRIG_RISING internal reset being released	7	10	15	µs
t_2	VDD falling, crossing VTRIG_FALLING; internal reset active	–	0.5	1	µs
t_3	Minimum VDD spike width below VTRIG_FALLING; considered to be a reset when POR cell output is HIGH	–	0.5	–	µs
t_4	Minimum VDD spike width below VTRIG_FALLING considered to be a reset when POR cell output is LOW	–	1	–	µs
t_5	Minimum VDD spike width above VTRIG_RISING considered to be a stable supply when POR cell output is LOW	–	50	–	ns
VTRIG_RISING	VDD rising trigger voltage	1.12	–	1.55	V
VTRIG_FALLING	VDD falling trigger voltage	1.0	–	1.45	V

Notes: 1. The hard reset and POR signals are gated, therefore, MCU ROM read will begin after both have been de-asserted.

The VDD ramp time required are max = 10µs. If external VDD ramp is more than 10µs, an external reset is required.

Hard Reset

Figure 35: Hard Reset

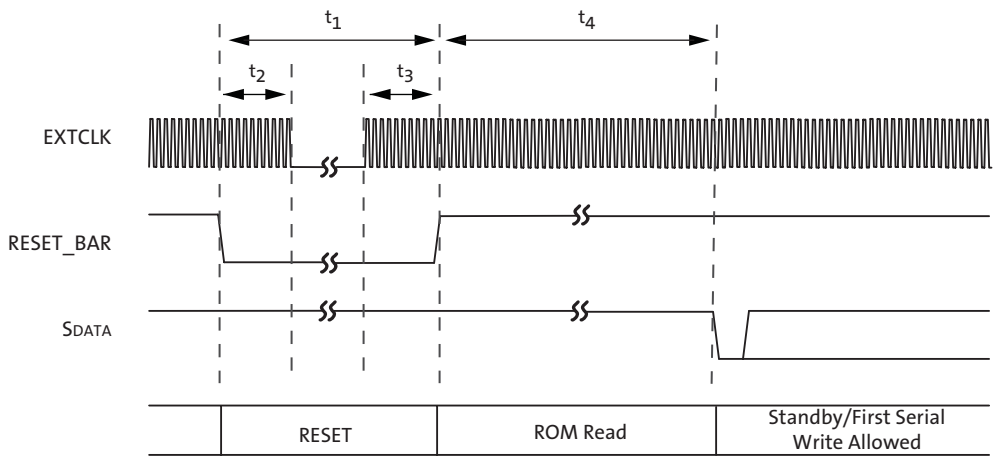


Table 16: Hard Reset

Symbol	Parameter	Min	Typ	Max	Unit
t_1	RESET_BAR pulse width	30	–	–	EXTCLK Cycles
t_2	Active EXTCLK after RESET_BAR asserted	10	–	–	
t_3	Active EXTCLK before RESET_BAR de-asserted	10	–	–	
t_4	Max ROM read time	–	–	6000	

A hard reset sequence to the camera can be activated by the following steps:

1. Wait for all supplies to be stable.
2. Assert RESET_BAR for at least 30 EXTCLK cycles.
3. De-assert RESET_BAR (input clock must be running for at least 10 EXTCLK cycles).
4. Wait 6000 clock cycles before using the two-wire serial interface.

After hard reset, the output FIFO is configured for operation but disabled and all outputs are tri-stated.

Soft Reset

A soft reset sequence to the camera can be activated by the following steps:

1. Enable soft reset by setting R0x301A[0].
2. Wait 6000 clock cycles before using the two-wire serial interface.

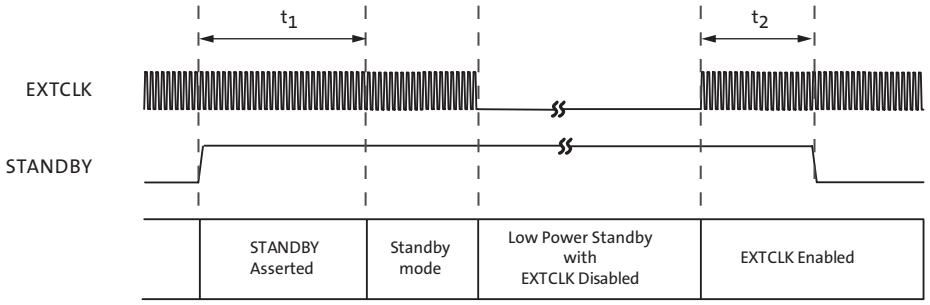
Standby Sequences

Standby mode can be activated by two methods:

1. Assert STANDBY (hard standby.)
2. Issue standby through the two-wire serial interface (soft standby).

Hard Standby

Figure 36: Hard Standby



Notes: 1. Standby mode with EXTCLK enabled is allowed, but the lowest power consumption is achieved in standby mode with the EXTCLK disabled.

Table 17: Hard Standby

Symbol	Parameter	Min	Typ	Max	Unit
t_1	Active EXTCLK after STANDBY asserted	–	20	60	ms
t_2	Active EXTCLK before STANDBY de-asserted	10	–	–	EXTCLK cycles

Hard standby can be performed in any sequencer state after all AE, AWB, AF, HG, and flicker calculations are finished and the chip is in sleep mode. During hard standby, the sequencer does the following:

- Configures MIPI, PLL, sensor core, and AFM driver
- Saves clock control registers
- Sets standby done bit
- Disables internal clocks
- MCU goes into sleep mode

After leaving hard standby, the MCU wakes up and performs the following sequence:

1. Restores internal clocks
2. Clears standby done bit
3. Leaves standby for MIPI, PLL, sensor core, and AFM driver

To enter hard standby:

a. Set STANDBY = 1
   The MCU will run standby routine and will set bit R0x3204[0 = 1 when standby entry is completed. By default, the clock is gated off when STANDBY is asserted, therefore the status of R0x3204[0] cannot be polled. During this time, no registers or variables are accessible.
b. To ensure standby mode has been entered, wait until the two-wire serial interface becomes non-responsive.
c. Optionally, stop the EXTCLK to minimize the standby current.

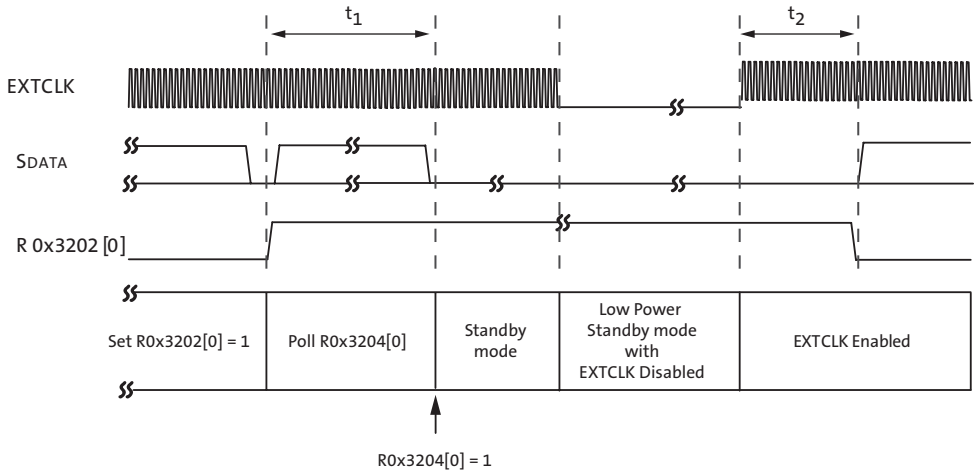
To leave hard standby:

a. Provide EXTCLK 10 clock cycles before de-asserting STANDBY.
b. De-assert STANDBY.
c. Set R0x301A[7] = 1 in order to enable parallel output from the sensor.

No frames are dropped coming out of standby.

Soft Standby (Two-wire Serial Interface)

Figure 37: Soft Standby



Notes: 1. Standby mode with EXTCLK enabled is allowed, but the lowest power consumption is achieved in standby mode with the EXTCLK disabled.

Table 18: Soft Standby

Symbol	Parameter	Min	Typ	Max	Unit
t_1	Active EXTCLK after R0x3202[0] =1 is set	–	20	60	ms
t_2	Active EXTCLK before soft standby de-activates	10	–	–	EXTCLK cycles

Soft standby can be performed in any sequencer state after all AE, AWB, AF, HG, and flicker calculations are finished and the chip is in sleep mode. During soft standby, the sequencer performs the following functions:

1. Configures MIPI, PLL, sensor core, and AFM driver into perspective standby states
2. Saves clock control registers
3. Sets standby done bit
4. Disables internal clocks
5. MCU goes into sleep mode

After leaving soft standby, the MCU wakes up and performs the following functions:

1. Restores internal clocks
2. Clears standby done bit
3. Leaves standby for MIPI, PLL, sensor core, and AFM driver

To enter soft standby:

a. Set R0x3202[3] = 1
b. Set R0x3202[0] = 1

c. To ensure Standby mode is entered, poll R0x3204[0] until it equals 1, which indicates that the standby entry is completed.
d. Optionally, stop the EXTCLK to minimize the standby current.

By default, internal clocks are running.

To leave soft standby:

a. Provide EXTCLK for 10 clock cycles .
b. Set R0x3202[0] = 0.
c. Set R0x301A[7] = 1. // This will enable parallel output from the sensor.

No frames are dropped coming out of standby.

Low-Power Standby (VDD_Disable)

Low power standby mode uses the SHUTDOWN pin to shut down digital power (VDD_Disable) and ensure the lowest power consumption. All the two-wire serial interface settings and firmware variables including patches will be lost in this mode. Waking up from this mode is equivalent to power up.

To enter standby:

a. Follow hard standby or soft standby procedure to enter into standby and check for standby entry.
b. Pull up the SHUTDOWN pin.

To leave standby:

a. Pull down SHUTDOWN pin .
b. Follow power-up procedure.

How to Inhibit Standby Entry

For both hard and soft standby, standby entry can be inhibited by setting R0x3202[3] = 0.

How to Synchronize Standby with End of Frame

If standby command is issued within a frame, the execution of standby will take place after the completion of the current line by default. In order to synchronize the execution of standby with the end of frame, set R0x3202[1] = 1.

How to Reproduce the Sequence of Soft Standby in DevWare

To reproduce entering soft standby mode in the demo system with DevWare, proceed with the following sequence:

1. Monitor the standby done status bit (R0x3204[0]), when the bit = 1 the MT9D112 has entered the standby state.
2. Stop the streaming by checking the stop button in DevWare.
3. Set R0x3202[3] = 1 // Enables the use of standby function.
4. Set R0x3202[0] = 1 // Enables standby through two-wire serial interface.
5. Monitor R0x3204[0] // This bit will be set to “1” when the enter standby sequence has been completed.

To reproduce leaving soft standby in the demo system with DevWare, proceed with the following sequence:

1. Monitor the standby done status bit (R0x3204[0]), when the bit = 0 the MT9D112 has left the standby state.
2. Set R0x3202[0] = 0 // Disables soft standby.
3. Monitor R0x3204[0] // This bit will be set to “0” when the exit standby sequence has been completed.
4. Set R0x301A[7] = 1 // This will enable parallel output from the sensor.
5. Enable streaming by checking the play button in DevWare.

Table 19: Summary of Standby States

	Two-wire Serial Interface Standby	Pin Activated Standby	Shutdown
Trigger to enter standby	Set two-wire serial interface register standby two-wire serial interface bit	Assert "standby" pin	Assert "Shutdown" pin. HOST should only assert "Shutdown" pin after host poll standby_done bit asserted or wait certain period of time.
Trigger for leaving standby	Clear two-wire serial interface register standby two-wire serial interface bit	De-assert "standby" pin	De-assert "Shutdown" pin. Internal reset will occur once pin de-asserted.
Final state of standby	MCU will control the reset of the system to go into standby gracefully (sensor, MIPI, PLL) and update the standby_done bit	MCU will control the reset of the system to go into standby gracefully (sensor, MIPI, PLL), and update standby_done bit. By default the two-wire serial interface will not be responsive in this mode and this fact could be used as a criteria for standby entry completion.	Digital circuit power down.
Time to complete standby	20ms (TYP), 60ms (MAX). Depends on whether sync with end of frame is enabled. If it is, time varies with frame rate.	20ms (typical), 60ms (max). Depends on whether sync with end of frame is enabled. If it is, time varies with frame rate.	Time to enter standby plus Internal V_{DD} discharge time. (in order of ns).
Clocks gated off	After reaching the end-point of standby, all internal clocks gate off, except two-wire serial interface decoder. Clocks to gate can be specified in standby_mode variable	After reaching the end-point of standby, all internal clocks gate off by default, including two-wire serial interface decoder. Clocks to gate can be specified in standby_mode variable.	All clocks gated off in digital circuit. Input clock pad is powered down.

Table 19: Summary of Standby States (continued)

	Two-wire Serial Interface Standby	Pin Activated Standby	Shutdown
Register/SRAM contents Variable contents	Keep	Keep	Lost due to power down. Everything will be reset upon exiting this mode.
Pre-state before standby trigger	Any mode	Any mode	Soft or Hard Standby complete
Power consumption	If external CLKIN is stopped after reaching the end-point, the power consumption is the same as Hard STANDBY. Otherwise, if external CLKIN is left running, the two-wire serial interface decoder is active and will consume power.	Stdcell leakage power is the main power consumption.	Digital power down. Analog core in low power mode; MIPI in ultra low power mode; PLL power down lowest power consumption.

Pin States

Table 20: Pin States During Conditions

Pin	Reset	Post-reset	Standby	Standby w/ SHUTDOWN	Power Down
DOUT[7:0]	Z	Z	Z by default (configurable via OE_BAR or two-wire serial interface reg)	Z by default	X
PIXCLK	Z	Z	Z by default (configurable)	Z by default	X
LV	Z	Z	Z by default (configurable)	Z by default	X
FV	Z	Z	Z by default (configurable)	Z by default	X
DATA_OUT_N	0	0	0	0	X
DATA_OUT_P	0	0	0	0	X
CLK_OUT_N	0	0	0	0	X
CLK_OUT_P	0	0	0	0	X
GPIO[3:0]	Z	Z	Depending on how system use them as DOUT_LSB1/DOUT_LSB2/SHUTTER/FLASH/ MODULE_ID/TRIGGER/OE_BAR	Depending on how system use them as DOUTPSB/SHUTTER/ FLASH/MODULE_ ID/ TRIGGER/OE_BAR	X
AF_GPIO[7:0]	Z	Z	Will maintain state that was set before Standby		X
SADDR	Input	Input	Input	Input	
SDATA	Input	I/O	Input	Input	
SCLK	Input	Input	Input	Input	
ATEST	I/O (Analog pad)	I/O (Analog pad)	I/O (Analog pad)	I/O (Analog pad)	X

Spectral Characteristics

Figure 38: Chief Ray Angle (CRA 22.1 Deg) vs. Image Height

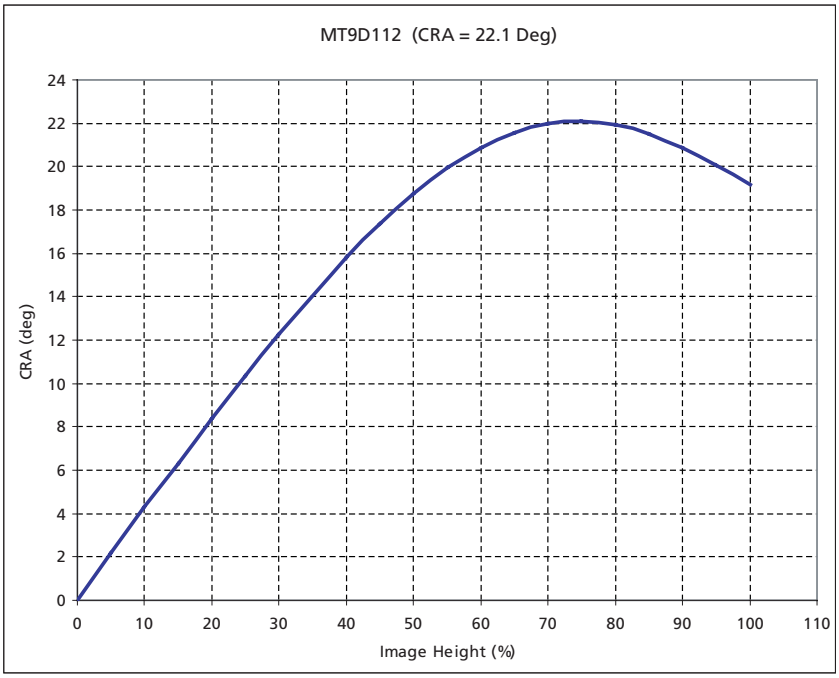
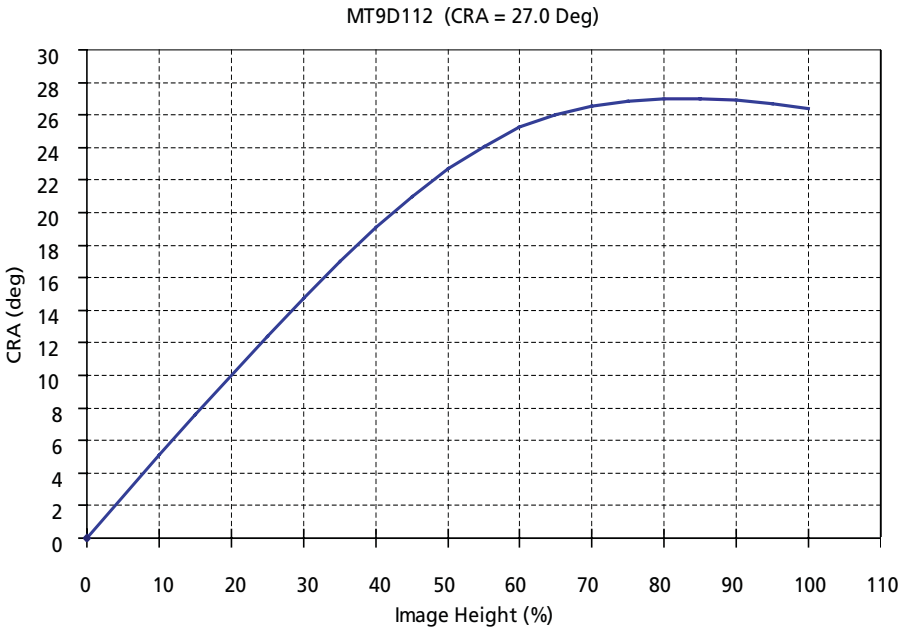
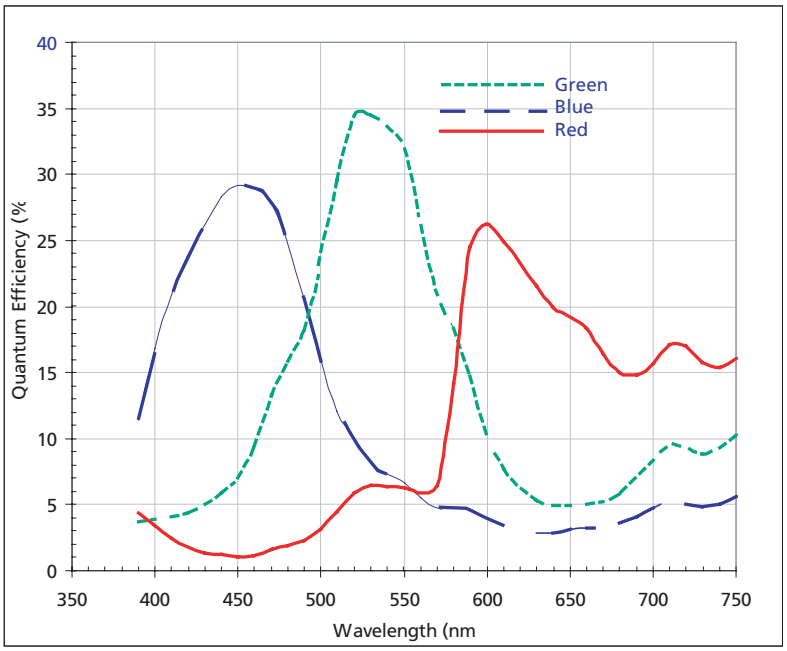
CRA vs. Image Height Plot	Image Height		CRA (Deg)
	(%)	(mm)	
	0	0	0
	5	0.110	2.17
	10	0.220	4.28
	15	0.330	6.34
	20	0.440	8.37
	25	0.550	10.34
	30	0.660	12.26
	35	0.770	14.09
	40	0.880	15.80
	45	0.990	17.37
	50	1.100	18.76
	55	1.210	19.95
	60	1.320	20.89
	65	1.430	21.57
	70	1.540	21.98
	75	1.650	22.10
	80	1.760	21.94
	85	1.870	21.53
	90	1.980	20.88
	95	2.090	20.06
	100	2.200	19.14

Figure 39: Chief Ray Angle (27.0 Deg CRA) vs. Image Height



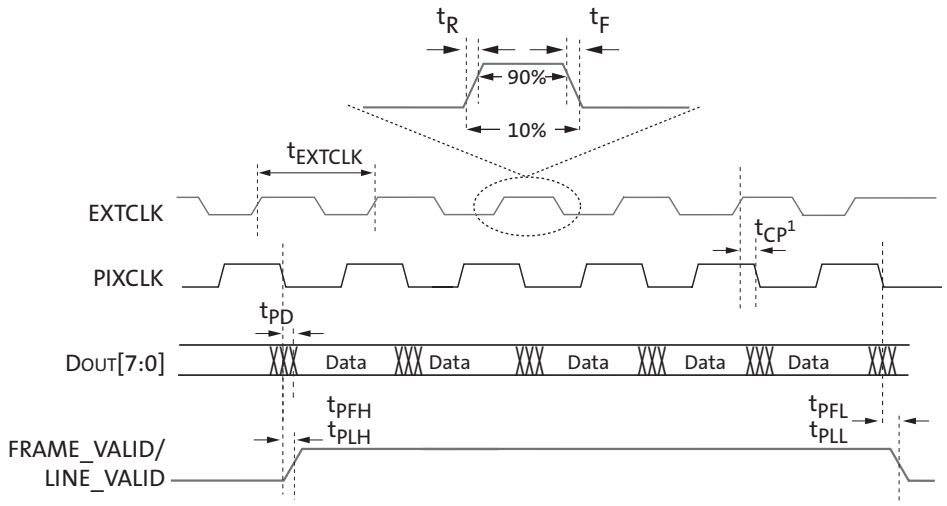
CRA vs. Image Height Plot	Image Height		CRA (Deg)
	(%)	(mm)	
	0	0	0
	5	0.110	2.55
	10	0.220	5.09
	15	0.330	7.56
	20	0.440	10.02
	25	0.550	12.42
	30	0.660	14.77
	35	0.770	16.96
	40	0.880	19.10
	45	0.990	20.99
	50	1.100	22.73
	55	1.210	24.07
	60	1.320	25.26
	65	1.430	25.98
	70	1.540	26.57
	75	1.650	26.85
	80	1.760	27.02
	85	1.870	26.99
	90	1.980	26.90
	95	2.090	26.71
	100	2.200	26.42

Figure 40: Quantum Efficiency



Electrical Specifications

Figure 41: I/O Timing Diagram



Note: PLL disabled for ^{t}CP.

Table 21: AC Electrical Characteristics

^{f}EXTCLK = 6–54 MHz; V_{DD} = 1.8V; V_{DD}_IO = 1.8–2.8V; V_{AA} = 2.8V; V_{AA}_PIX = 2.8V; V_{DD}PLL = 2.8V; T_Junction = 25°C; C_{LOAD} = 15–20p

Symbol	Parameter	Conditions		Min	Typ	Max	Unit	Note
^{f}EXTCLK	External clock frequency	PLL enabled and disabled		6	–	54	MHz	
^{f}PIXCLK	PIXCLK/internal master clock frequency	PLL disabled		6	–	54	MHz	
		PLL enabled		40	–	80	MHz	
^{t}R	Input clock rise time	From 10 to 90% Vp-p		–	2	5	ns	3
				–	–	–	ns	4
^{t}F	Input clock fall time	From 90 to 10% Vp-p		–	2	5	ns	3
				–	–	–	ns	4
	Clock duty cycle			45	50	55	%	
^{t}JITTER	Input clock jitter (peak-peak cycle jitter)			–	0.5	–	ns	1
^{t}CP	EXTCLK rising to PIXCLK falling propagation delay	PLL disabled		–	–	15	ns	
Output pin slew	Programmable slew = 7	V_{DD}_IO = 2.8V	C_LOAD = 30 ±3pF	0.68	–	1.90	V/ns	
			C_LOAD = 15 ±3pF	1.60	–	3.70	V/ns	
		V_{DD}_IO = 1.8V	C_LOAD = 30 ±3pF	0.26	–	0.50	V/ns	
			C_LOAD = 15 ±3pF	0.45	–	1.40	V/ns	
Output pin slew	Programmable slew = 4	V_{DD}_IO = 2.8V	C_{LOAD} = 30 ±3pF	0.40	–	0.68	V/ns	
			C_{LOAD} = 15 ±3pF	0.59	–	0.77	V/ns	
		V_{DD}_IO = 1.8V	C_{LOAD} = 30 ±3pF	0.17	–	0.26	V/ns	
			C_{LOAD} = 15 ±3pF	0.21	–	0.37	V/ns	
	Programmable slew = 0	V_{DD}_IO = 2.8V	C_{LOAD} = 30 ±3pF	0.26	–	0.34	V/ns	
			C_{LOAD} = 15 ±3pF	0.31	–	0.37	V/ns	
		V_{DD}_IO = 1.8V	C_{LOAD} = 30 ±3pF	0.12	–	0.16	V/ns	
			C_{LOAD} = 15 ±3pF	0.14	–	0.21	V/ns	
^{t}PD	PIXCLK to data valid	Default		–	–	5	ns	2
^{t}PFH	PIXCLK to FV HIGH	Default		–	–	5	ns	2
^{t}PLH	PIXCLK to LV HIGH	Default		–	–	5	ns	2
^{t}PFL	PIXCLK to FV LOW	Default		–	–	5	ns	2
^{t}PLL	PIXCLK to LV LOW	Default		–	–	5	ns	2
C_{IN}	Input pin capacitance			–	3.5	–	pF	

Notes:
1. Value equal to jitter on tester.
2. Valid for C_LOAD <20pF on PIXCLK, DOUT, LINE_VALID, and FRAME_VALID pads. Loads must be matched as closely as possible.
3. PLL is off.
4. When PLL is on, the input clock rise/fall time specificatons are not applicable; the input HIGH/LOW voltage should still meet the specifications included in Table 22 on page 67.

Table 22: I/O Parameters

Symbol	Parameter	Condition	Min	Typ	Max	Unit
V_{IH}	Input HIGH voltage	V_{DD}_IO = 2.8V at specified I_{IN} V_{DD}_IO = 1.8V at specified I_{IN}	2.4 1.4		V_{DD}_IO + 0.3 V_{DD}_IO + 0.3	V
V_{IL}	Input LOW voltage	V_{DD}_IO = 2.8V at specified I_{IN} V_{DD}_IO = 1.8V at specified I_{IN}	GND –0.3 GND –0.3		0.8 0.4	V
I_{IN}	Input leakage current	No pull-up resistor; VIN = V_{DD} or D_{GND}	–50	–	50	?A
V_{OH}	Output HIGH voltage	At specified I_{OH}	V_{DD}_IO –0.4	–		V
V_{OL}	Output LOW voltage	At specified I_{OL}	–	–	0.4	V
I_{OH}	Output HIGH current	At specified V_{OH}	–	–	–7	mA
I_{OL}	Output HIGH current	At specified V_{OL}	–	–	7	mA
I_{OZ}	Tri-state output leakage current	VIN = V_{DD}_IO or GND	–10	–	10	µA

Table 23: DC Electrical Definitions and Characteristics

[f]EXTCLK = 6–54 MHz; V_{DD} = 1.8V; V_{DD}_IO = 2.8V; V_{AA} = 2.8V; VAA_PIX = 2.8V; VDDPLL = 2.8V; T_Junction = 25°C; Dark lighting conditions

Symbol	Parameter	Conditions		Min	Typ	Max	Unit	Note
V_{DD}	Core digital voltage			1.7	1.8	1.95	V	
V_{DD}_IO	I/O digital voltage	Serial data interface		1.7	1.8	1.95	V	
		Parallel data interface		2.5	2.8	3.1	V	
				1.7	1.8	1.95	V	
V_{AA}	Analog voltage			2.5	2.8	3.1	V	
VAA_PIX	Pixel supply voltage			2.5	2.8	3.1	V	
VDDPLL	PLL supply voltage			2.5	2.8	3.1	V	
VDD_AF	AF supply voltage			1.7	1.8	1.95	V	
				2.5	2.8	3.1	V	
IDD_1	Digital operating current	Low power preview	PLL off	–	16	23	mA	
			PLL on	–	25	28	mA	
IDD_IO_1	I/O digital supply current	See note 2		–	25	–	mA	2
IAA_1	Analog operating current	Low power preview		–	–	24	mA	
IAA_PIX_1	Pixel supply current	Low power preview		–	–	1.25	mA	
IDD_PLL_1	PLL supply current	Low power preview with PLL enabled		–	–	10	mA	
	Total power consumption	Low power preview	PLL off	–	–	140	mW	1
			PLL on	–	–	150	mW	1
IDD_2	Digital operating current	Full resolution	PLL off	–	22	33	mA	
			PLL on	–	38	42	mA	
IDD_IO_2	I/O digital supply current	See note 3		–	37	–	mA	3
IAA_2	Analog operating current	Full resolution		–	–	40	mA	
IAA_PIX_2	Pixel supply current	Full resolution		–	–	0.75	mA	
IDD_PLL_2	PLL supply current	Full resolution with PLL enabled		–	–	10	mA	
	Total power consumption	Full resolution	PLL off	–	–	202	mW	1
			PLL on	–	–	218	mW	1
STDBY_1	Standby current 1	Hard standby with SHUTDOWN enabled		–	1	5	µA	4
STDBY_2	Standby current 2	Soft standby with SHUTDOWN enabled		–	1	5	µA	4
STDBY_3	Standby current 3	Soft standby with EXTCLK disabled		–	35	100	µA	
STDBY_4	Standby current 4	Hard standby with EXTCLK disabled		–	35	100	µA	

Notes:
1. Total power consumption does not include current from VDD_IO or VDD_AF.
2. High power preview mode with 29 fps at illuminated flat field target.
3. Full resolution mode with 15 fps at illuminated flat field target.
4. EXTCLK does not effect current consumption when SHUTDOWN is enabled.

Caution

Stresses above those listed in Table 24 may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Table 24: Absolute Maximum Ratings

Symbol	Parameter	Rating		Unit
		Min	Max	
VDD_MAX	Core digital voltage	–0.3	2.4	V
VDD_IO_MAX	I/O digital voltage	–0.3	4.0	V
VAA_MAX	Analog voltage	–0.3	4.0	V
VAA_PIX_MAX	Pixel supply voltage	–0.3	4.0	V
VDD_PLL MAX	PLL supply voltage	–0.3	4.0	V
VIN	DC input voltage	–0.3	VDD_IO + 0.3	V
IIN	Transient input current (0.5 sec duration)	–	150	mA
T_OP	Operating temperature (measure at junction)	–30	75	°C
T_STG[1]	Storage temperature	–40	85	°C

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

Figure 42: Two-Wire Serial Interface Timing

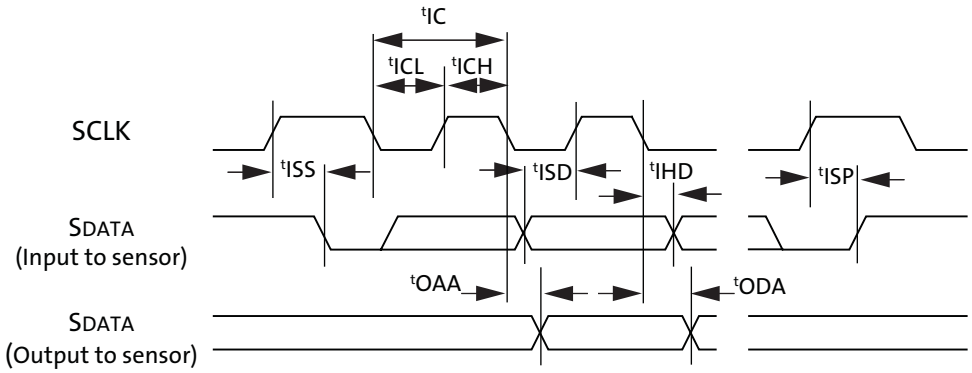


Figure 43: Two-Wire Serial Interface Start and Stop Timing

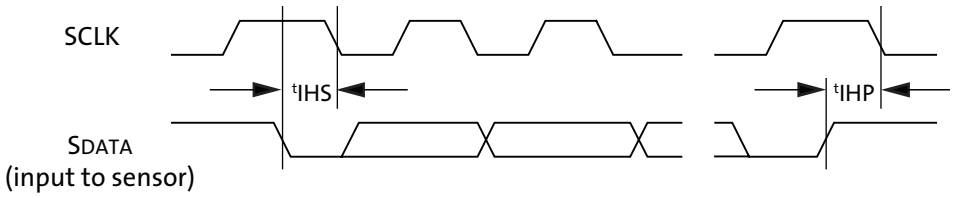


Table 25: Two-Wire Serial Interface Timing Specifications

V_{DD} = 1.8V; V_{AA} = 2.8V; V_{AA_PIX} = 2.8V; V_{DD_IO} = 1.8–2.8V; T_J = 30°C + 70°C, unless otherwise specified

Symbol	Parameter	Min	Typ	Max	Unit	Note
f_{SCLK}	Serial interface input clock frequency	0	100	400	kHz	
	SCLK duty cycle	35		65	%	
t_{ISS}	Setup time for start condition	820	–	–	ns	
t_{IHS}	Hold time for start condition	660	–	–	ns	
t_{IHD}	Hold time for input data	10	–	–	ns	
t_{ISD}	Setup time for input data	300	–	–	ns	
t_{ISP}	Setup time for stop condition	300	–	–	ns	
t_{IHP}	Hold time for stop condition	300	–	–	ns	
t_{OAA}	Output data acknowledge time	–	–	350	ns	1
				1200		2
t_{ODA}	Output data delay time	550	–	–	ns	1
		1200				2
C_{IN_SI}	Serial interface input pin capacitance	–	–	6	pF	
C_{LOAD_SD}	S_{DATA} max load capacitance	–	–	15	pF	
R_SD	S_{DATA} pull-up resistor	–	1.5	–	KΩ	

Notes:
1. Master clock frequency of 50 MHz.
2. Master clock frequency of 6 MHz.

Revision History

Rev. J 5/25/10

- Updated to non-confidential

Rev. H 3/10/10

- Updated to Aptina template
- Register tables moved to new document, the MT9D112 Register Reference

Rev. G, Production 1/14/08

- Added new part number to Table 2 on page 1
- Added more detail to “Special Function Registers and MCU SRAM” on page 52
- Added Figure 39: “Chief Ray Angle (27.0 Deg CRA) vs. Image Height,” on page 64
- Updated Figure 40: “Quantum Efficiency,” on page 64 with thicker lines

Rev. F, Production 5/24/07

- Added reference to TN-09-120 in "Power-up Sequence" on page 55
- Updated Table 41, “I/O Timing Diagram,” on page 65
- Updated Figure 35: “Hard Reset,” on page 56
- Updated Figure 36: “Hard Standby,” on page 57
- Updated Figure 37: “Soft Standby,” on page 58
- Reworded “Standby Sequences” on page 57 through page 60
- Updated Table 21, “AC Electrical Characteristics,” on page 66
- Updated Figure 41: “I/O Timing Diagram,” on page 65
- Updated Figure 42: “Two-Wire Serial Interface Timing,” on page 68
- Updated Figure 43: “Two-Wire Serial Interface Start and Stop Timing,” on page 68
- Updated Table 25, “Two-Wire Serial Interface Timing Specifications,” on page 69
- Fixed minor typos throughout the document

Rev. E, Production 11/06

- Updated Table 1, “Key Performance Parameters,” on page 1
- Updated "MT9D112 Overview" on page 7
- Updated Table 3, “Signal Description,” on page 9
- Updated "Architecture Overview" on page 11
- Updated "Sensor Core Description" on page 11
- Updated "Integration Time" on page 14
- Updated "Raw Data Timing" on page 22
- Updated Figure 25: “Example of Timing for Non-Decimated Uncompressed Output Bypassing Output FIFO,” on page 34
- Updated "Single Read from Random Location" on page 48
- Updated Figure 28: “Single Read from Random Location,” on page 48
- Updated "Single Read from Current Location" on page 48
- Updated "Sequential Read, Start from Random Location" on page 49
- Updated "Sequential Read, Start from Current Location" on page 49
- Updated Figure 29: “Single Read from Current Location,” on page 48
- Updated Figure 30: “Sequential Read, Start from Random Location,” on page 49
- Updated Figure 31: “Sequential Read, Start from Current Location,” on page 49
- Updated Figure 32: “Single Write to Random Location,” on page 50
- Updated "Single Write to Random Location" on page 50

- Updated "Sequential Write, Start at Random Location" on page 50
- Updated Figure 33: “Sequential Write, Start at Random Location,” on page 50
- Updated Table 15, “0: Core Registers,” on page 1
- Updated Table 16, “1: SOC1 Registers,” on page 5
- Updated Table 17, “2: SOC2 Registers,” on page 8
- Updated Table 18, “0: Monitor Variables,” on page 12
- Updated Table 19, “1: Sequencer Variables,” on page 12
- Updated Table 20, “2: Auto Exposure Variables,” on page 14
- Updated Table 21, “3: Auto White Balance Variables,” on page 16
- Updated Table 22, “4: Anti-Flicker Variables,” on page 18
- Updated Table 23, “5: Auto Focus Variables,” on page 20
- Updated Table 24, “6: Auto Focus Mechanism Variables,” on page 21
- Updated Table 25, “7: Mode Variables,” on page 22
- Updated Table 26, “11: Histogram Variables,” on page 25
- Updated Table 27, “0: GPIO Registers,” on page 25
- Updated Table 28, “0: Core Registers,” on page 1
- Updated Table 29, “1: SOC1 Registers,” on page 11
- Updated Table 30, “2: SOC Registers,” on page 25
- Updated Table 31, “0: Monitor Variables,” on page 41
- Updated Table 32, “1: Sequencer Variables,” on page 42
- Updated Table 33, “2: Auto Exposure Variables,” on page 50
- Updated Table 34, “3: Auto White Balance Variables,” on page 54
- Updated Table 35, “4: Anti-Flicker Variables,” on page 59
- Updated Table 36, “5: Auto Focus Variables,” on page 62
- Updated Table 37, “6: Auto Focus Mechanism Variables,” on page 66
- Updated Table 38, “7: Mode Variables,” on page 71
- Updated Table 39, “11: Histogram Variables,” on page 82
- Updated Table 40, “0: GPIO Registers,” on page 83
- Updated "Power-up Sequence" on page 55
- Updated Table 16, “Hard Reset,” on page 56
- Updated Figure 35: “Hard Reset,” on page 56
- Updated Table 17, “Hard Standby,” on page 57
- Updated "Hard Standby" on page 57
- Updated "Soft Standby (Two-wire Serial Interface)" on page 58
- Updated Table 18, “Soft Standby,” on page 58
- Updated "How to Synchronize Standby with End of Frame" on page 59
- Updated Table 19, “Summary of Standby States,” on page 60
- Updated Figure 41: “I/O Timing Diagram,” on page 65
- Updated Table 23, “DC Electrical Definitions and Characteristics,” on page 67
- Updated Table 24, “Absolute Maximum Ratings,” on page 68
- Updated Table 25, “Two-Wire Serial Interface Timing Specifications,” on page 69

Rev. D, Production 8/06

- Updated Table 1, “Key Performance Parameters,” on page 1
- Updated Table 6, “Row Timing Parameters,” on page 23
- Updated Figure 28: “Single Read from Random Location,” on page 48
- Updated Figure 30: “Sequential Read, Start from Random Location,” on page 49
- Updated"Power-up Sequence" on page 55
- Updated Figure 41: “I/O Timing Diagram,” on page 65

- Updated Table 21, “AC Electrical Characteristics,” on page 66
- Updated Table 22, “I/O Parameters,” on page 67
- Updated Table 23, “DC Electrical Definitions and Characteristics,” on page 67
- Updated Figure 42: “Two-Wire Serial Interface Timing,” on page 68
- Updated Table 25, “Two-Wire Serial Interface Timing Specifications,” on page 69

Rev. C, Preliminary 5/06

- Updated “Features” on page 1
- Updated Table 3, “Signal Description,” on page 9
- Updated Figure 1: “Typical Configuration (connections),” on page 10
- Updated “Gain Options” on page 14
- Updated “Integration Time” on page 14
- Updated “Pixel Array” on page 12
- Updated Figure 7: “6 Pixels in Normal and Column Mirror Readout Modes,” on page 17
- Updated Figure 8: “6 Rows in Normal and Row Mirror Readout Modes,” on page 17
- Updated “Lens Shading Correction” on page 26
- Updated “Contrast and Gamma Correction” on page 28
- Updated Figure 25: “Example of Timing for Non-Decimated Uncompressed Output Bypassing Output FIFO,” on page 34
- Updated “General Purpose I/Os” on page 36
- Updated “Snapshot and Flash” on page 38
- Updated “Exposure Control” on page 40
- Updated “Slave Address/Data Direction Byte” on page 46
- Updated Figure 30: “Sequential Read, Start from Random Location,” on page 49
- Updated Figure 33: “Sequential Write, Start at Random Location,” on page 50
- Updated Table 15, “0: Core Registers,” on page 1
- Updated Table 16, “1: SOC1 Registers,” on page 5
- Updated Table 17, “2: SOC2 Registers,” on page 8
- Updated Table 18, “0: Monitor Variables,” on page 12
- Updated Table 19, “1: Sequencer Variables,” on page 12
- Updated Table 20, “2: Auto Exposure Variables,” on page 14
- Updated Table 21, “3: Auto White Balance Variables,” on page 16
- Updated Table 22, “4: Anti-Flicker Variables,” on page 18
- Updated Table 23, “5: Auto Focus Variables,” on page 20
- Updated Table 24, “6: Auto Focus Mechanism Variables,” on page 21
- Updated Table 25, “7: Mode Variables,” on page 22
- Updated Table 26, “11: Histogram Variables,” on page 25
- Updated Table 27, “0: GPIO Registers,” on page 25
- Updated Table 28, “0: Core Registers,” on page 1
- Updated Table 29, “1: SOC1 Registers,” on page 11
- Updated Table 30, “2: SOC Registers,” on page 25
- Updated Table 31, “0: Monitor Variables,” on page 41
- Updated Table 32, “1: Sequencer Variables,” on page 42
- Updated Table 33, “2: Auto Exposure Variables,” on page 50
- Updated Table 34, “3: Auto White Balance Variables,” on page 54
- Updated Table 35, “4: Anti-Flicker Variables,” on page 59
- Updated Table 36, “5: Auto Focus Variables,” on page 62
- Updated Table 37, “6: Auto Focus Mechanism Variables,” on page 66

- Updated Table 38, “7: Mode Variables,” on page 71
- Updated Table 39, “11: Histogram Variables,” on page 82
- Updated Table 40, “0: GPIO Registers,” on page 83
- Updated “Power-up Sequence” on page 55
- Added “Pin States” on page 62
- Updated Table 20, “Pin States During Conditions,” on page 62
- Added Figure 40: “Quantum Efficiency,” on page 64

Rev. B, Preliminary 2/06

- Updated “Features” on page 1
- Updated Table 1, “Key Performance Parameters,” on page 1
- Updated “General Description” on page 7
- Updated “MT9D112 Overview” on page 7
- Updated Table 3, “Signal Description,” on page 9
- Updated Figure 1: “Typical Configuration (connections),” on page 10
- Updated “Sensor Core Description” on page 11
- Updated “Pixel Array” on page 12
- Updated Figure 4: “Pixel Array,” on page 12
- Updated Figure 5: “Pixel Color Pattern Detail (Top Right Corner),” on page 13
- Updated “Default Readout Order” on page 13
- Updated “Integration Time” on page 14
- Updated “PLL Setup” on page 15
- Updated “Horizontal Mirror” on page 16
- Updated Table 4, “Frequency Parameters,” on page 16
- Updated “Window Size” on page 16
- Updated “Raw Data Timing” on page 22
- Updated Table 6, “Row Timing Parameters,” on page 23
- Updated Figure 20: “Test Patterns,” on page 25
- Updated “Test Patterns” on page 25
- Updated “First Black Level Subtraction and Digital Gain” on page 26
- Updated “Contrast and Gamma Correction” on page 28
- Updated “Color Kill” on page 30
- Updated “Parallel and MIPI Output” on page 32
- Updated “YUV/RGB Uncompressed Output” on page 32
- Updated “Uncompressed 10-Bit Bypass Output” on page 34
- Updated “Firmware Architecture” on page 37
- Updated “Slave Address/Data Direction Byte” on page 46
- Updated “Message Byte” on page 47
- Updated “Protocol” on page 46
- Updated “Data Transfer” on page 46
- Updated “Typical Serial Transfer” on page 47
- Updated Figure 28: “Single Read from Random Location,” on page 48
- Updated Figure 29: “Single Read from Current Location,” on page 48
- Updated Figure 30: “Sequential Read, Start from Random Location,” on page 49
- Updated Figure 31: “Sequential Read, Start from Current Location,” on page 49
- Updated “Registers” on page 51
- Updated “Variables” on page 51
- Updated “Double-buffered Registers” on page 53
- Updated “Bad Frames” on page 53

- Updated “Changes to Integration Time” on page 53
- Updated “Changes to Gain Settings” on page 54
- Updated Table 15, “0: Core Registers,” on page 1
- Updated Table 16, “1: SOC1 Registers,” on page 5
- Updated Table 17, “2: SOC2 Registers,” on page 8
- Updated Table 18, “0: Monitor Variables,” on page 12
- Updated Table 19, “1: Sequencer Variables,” on page 12
- Updated Table 20, “2: Auto Exposure Variables,” on page 14
- Updated Table 21, “3: Auto White Balance Variables,” on page 16
- Updated Table 22, “4: Anti-Flicker Variables,” on page 18
- Updated Table 23, “5: Auto Focus Variables,” on page 20
- Updated Table 24, “6: Auto Focus Mechanism Variables,” on page 21
- Updated Table 25, “7: Mode Variables,” on page 22
- Updated Table 26, “11: Histogram Variables,” on page 25
- Updated Table 27, “0: GPIO Registers,” on page 25
- Updated Table 28, “0: Core Registers,” on page 1
- Updated Table 29, “1: SOC1 Registers,” on page 11
- Updated Table 30, “2: SOC Registers,” on page 25
- Updated Table 31, “0: Monitor Variables,” on page 41
- Updated Table 32, “1: Sequencer Variables,” on page 42
- Updated Table 33, “2: Auto Exposure Variables,” on page 50
- Updated Table 34, “3: Auto White Balance Variables,” on page 54
- Updated Table 35, “4: Anti-Flicker Variables,” on page 59
- Updated Table 36, “5: Auto Focus Variables,” on page 62
- Updated Table 37, “6: Auto Focus Mechanism Variables,” on page 66
- Updated Table 38, “7: Mode Variables,” on page 71
- Updated Table 39, “11: Histogram Variables,” on page 82
- Updated Table 40, “0: GPIO Registers,” on page 83
- Updated “Power-up Sequence” on page 55
- Updated “Hard Reset” on page 56
- Updated “Standby Sequences” on page 57
- Updated “Soft Reset” on page 56
- Updated “Hard Standby” on page 57
- Updated “Soft Standby (Two-wire Serial Interface)” on page 58
- Updated Figure 36: “Hard Standby,” on page 57
- Updated Table 17: “Hard Standby,” on page 57
- Updated Figure 37: “Soft Standby,” on page 58
- Updated “Low-Power Standby (Vdd_Disable)” on page 59
- Updated Table 20, “Pin States During Conditions,” on page 62
- Updated Figure 38: “Chief Ray Angle (CRA 22.1 Deg) vs. Image Height,” on page 63
- Updated Figure 41: “I/O Timing Diagram,” on page 65
- Updated Table 21, “AC Electrical Characteristics,” on page 66
- Updated Table 22, “I/O Parameters,” on page 67
- Updated Table 23, “DC Electrical Definitions and Characteristics,” on page 67
- Updated Table 24, “Absolute Maximum Ratings,” on page 68
- Updated Table 15, “POR Parameters,” on page 55
- Updated Table 25, “Two-Wire Serial Interface Timing Specifications,” on page 69

Rev, A, Advance 10/05

- Initial release

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