# 1/6-Inch 1.3Mp System-On-A-Chip (SOC) CMOS Digital Image Sensor

## MT9M113 Data Sheet

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

## Features

- 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 four-channel lens shading correction
- Arbitrary image scaling with anti-aliasing
- 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 mobile industry processor interface (MIPI) data output
- Xenon and LED flash support with fast exposure adaptation
- Independently configurable gamma correction

## Applications

- Cellular phones
- PC cameras
- PDAs

**Table 1: Key Performance Parameters**

| Parameter | | Value |
|---|---|---|
| Optical format | | 1/6-inch (5:4) |
| Full resolution | | 1280 x 1024 pixels (SXGA) |
| Pixel size | | 1.75μm x 1.75μm |
| Dynamic range | | 66dB |
| SNR MAX | | 38.5dB |
| Responsivity | | 0.54 V/lux-sec |
| Chief ray angle (CRA) | | 24.63° MAX at 90% image height |
| Color filter array | | RGB Bayer pattern |
| Active pixel array area | | 2.28mm x 1.83mm |
| Shutter type | | Electronic rolling shutter (ERS) |
| Input clock frequency | | 8−54 MHz |
| Maximum frame rate | | 15 fps at full resolution, 30 fps in preview mode, 30 fps in video mode |
| Maximum pixel data output | | 30 Mp/s |
| Maximum pixel clock frequency | | 60 MHz |
| Supply voltage | Analog | 2.5−3.1V |
| | Digital | 1.7−1.95V |
| | I/O | 1.7−3.1V |
| | PLL | 2.5−3.1V |
| | MIPI | 1.7−1.95V |
| ADC resolution | | 10-bit, on-die |
| Power consumption per mode | Full resolution | 222mW at 15 fps |
| | Video | 156mW at 30 fps |
| | Preview | 112mW at 20fps |
| | Standby | 50μW |
| Operating temperature | | −30°C to +70°C (at junction) |

## Ordering Information

**Table 2: Available Part Numbers**

| Part Number | Description |
|---|---|
| MT9M113D00STCK24AC1 | Bare die |

## Table of Contents

## List of Figures

## List of Tables

## General Description

Aptina's MT9M113 is a 1/6-inch 1.3Mp CMOS digital image sensor with an integrated advanced camera system. This camera system features a microcontroller (MCU), a sophisticated IFP, and both parallel and serial MIPI interfaces. It also includes a one-time programmable memory for storing module-specific information for identification purposes. 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 1300 x 1044 pixels, programmable timing and control circuitry including a PLL and external flash support, analog signal chain with automatic offset correction and programmable gain, and a 10-bit analog-to-digital converter (ADC). The entire SOC has ultralow power requirements and superior low light performance that is particularly suitable for mobile applications. The MT9M113 is based on Aptina's breakthrough low-noise CMOS imaging technology that achieves CCD-like 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.

## MT9M113 Overview

The MT9M113 has a color image sensor with a Bayer color filter arrangement and a 1.3Mp active-pixel array with electronic rolling shutter (ERS). The sensor core readout is 10-bit, supports skipping and 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 MT9M113 also has an embedded 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 MT9M113 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 MT9M113 has numerous power-conserving features, including an internal soft standby mode and a hard standby mode, which both allow the internal power bus to be disabled.

Another important consideration for wireless devices is their electromagnetic interference (EMI). The MT9M113 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 IFP and flexible programmability of the MT9M113 provide a variety of ways to enhance and optimize the image sensor's performance. Built-in optimization algorithms enable the MT9M113 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, shading correction, defect correction, noise reduction, color interpolation, edge detection, color correction, aperture correction, and image formatting such as cropping and scaling.

The MT9M113 also includes a sequencer that coordinates all events triggered by the user. The sequencer manages auto white balance (AWB), flicker detection, and auto exposure (AE) for the different operating modes, which include preview, still capture, video, and snapshot with flash.

The MT9M113 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 through the two-wire serial interface.

A two-wire serial register interface bus enables read/write access to control registers, variables, and special function registers within the MT9M113. The hardware registers 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 those for AWB and AE. Special function registers are registers connected to the local bus of the microcontroller and include general purpose I/O (GPIO).

The GPIOs can be configured to allow the user to output a flash or achieve a 10-bit parallel output.

## Signal Description

Table 3 provides the signal types and descriptions for the MT9M113. Power supply domains are also identified.

**Table 3:       Signal Descriptions**

| Name | Type | Description | Notes |
|---|---|---|---|
| STANDBY | Input | Controls sensor's 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 (can leave floating if not used). | |
| EXTCLK | Input | Input clock signal 8–54 MHz. | |
| GPIO[4:0] | I/O | General purpose digital I/O, can be configured for FLASH/SHUTTER/DOUT_LSB0/DOUT_LSB1/MODULE_ID/OE_BAR. | |
| SDATA | I/O | Two-wire serial interface data. | |
| VPP | Input | High voltage programming pin for anti-fuse of module ID (must leave floating for normal operation). | |
| DOUT[7:0] | Output | Eight-bit image data output or most significant bits (MSB) of 10-bit sensor bypass mode. | |
| DOUT_N | Output | Differential MIPI data (sub-LVDS, negative) (must leave floating if not used). | |
| DOUT_P | Output | Differential MIPI data (sub-LVDS, positive) (must leave floating if not used). | |
| CLK_N | Output | Differential MIPI clock (sub-LVDS, negative) (must leave floating if not used). | |
| CLK_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 pixels in the active line. | |
| FRAME_VALID | Output | Identifies rows in the active image. | |
| VDD | Supply | Digital power (1.8V typical). | |
| VAA_PIX | Supply | Pixel array power (2.8V typical). | |
| VAA | Supply | Analog power (2.8V typical). | |
| VDD_PLL | Supply | PLL power (2.8V typical). | |
| VDD_IO | Supply | I/O power supply (1.8V or 2.8V typical). | |
| GND_IO | Supply | I/O ground. | |
| DGND | Supply | Digital ground. | 1 |
| GND_PLL | Supply | PLL ground. | |
| AGND | Supply | Analog ground. | 1 |
| VDDIO_TX | Supply | I/O power supply for the MIPI interface 1.8V typical. | |

Notes:     1.  AGND and DGND are not connected internally (inside the chip).

## Typical Connections

Figure 1 shows typical MT9M113 device connections. For low-noise operation, the MT9M113 requires separate power supplies for analog and digital circuitry. Incoming digital and analog ground conductors can be tied together next to the die. Both power supply rails should be decoupled from ground using capacitors as close as possible to the die. The use of inductance filters is not recommended on the power supplies or output signals. The MT9M113 also supports different digital core ($V_{DD}$/$D_{GND}$), MIPI ($V_{DD}IO\_TX$), and I/O ($V_{DD}\_IO$/$GND\_IO$) power domains that can be at different voltages. The PLL requires a clean power source ($V_{DD}\_PLL$/$GND\_PLL$).

**Figure 1:**  **Typical Configuration[1] (connection)**
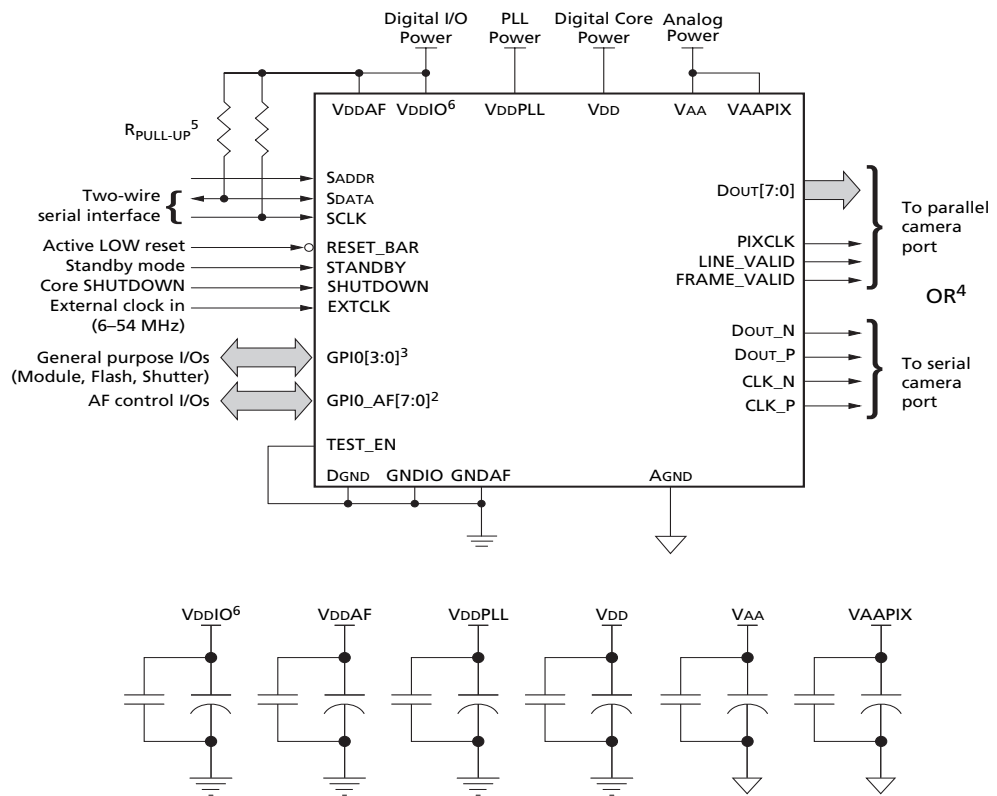


It is recommended that 0.1µF and 1µF decoupling capacitors for each power supply are mounted as close as possible to the pad. Actual values and results may vary depending on layout and design considerations.

Notes:
1. This typical configuration shows only one scenario out of multiple possible variations for this sensor.
2. If a MIPI device is not required, the following pads must be left floating: DOUT_P, DOUT_N, CLK_P, CLK_N, and $V_{DD}IO\_TX$.
3. The GPIO pads can serve multiple features that can be reconfigured. The function and direction will vary by applications.
4. Only one of the output modes (serial or parallel) can be used at any time.
5. Aptina recommends a resistor value of 1.5KΩ for the two-wire serial interface $R_{PULL-UP}$; however, greater values may be used for slower transmission speed.
6. All inputs must be configured with $V_{DD}\_IO$.
7. $V_{AA}$ and VAA_PIX must be tied together.
8. $V_{PP}$ is the one-time programmable (OTP) memory signal and should be left floating during normal operation.

## Architecture Overview

The MT9M113 combines a 1.3Mp sensor core together with an IFP to form a stand-alone solution that includes both image acquisition and processing. Both the sensor core and the IFP have internal registers that can be controlled by the user. In normal operation though, an integrated microcontroller controls most aspects of operation autonomously. The processed image data is transmitted to the host system either through a parallel bus or a serial data interface through the output interface.

**Figure 2:**     **SOC Block Diagram**
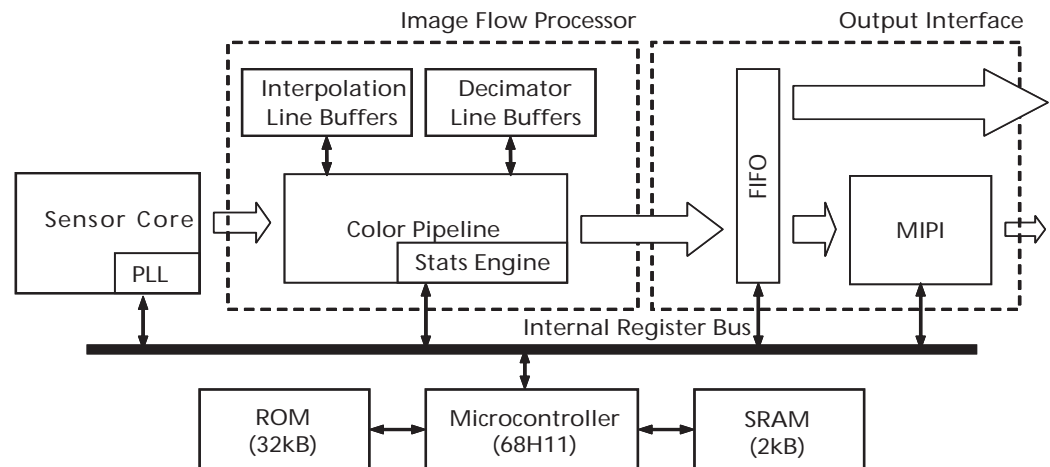


## Sensor Core Description

The sensor core of the MT9M113 is a progressive-scan sensor that generates a stream of pixel data at a constant frame rate, qualified by LINE_VALID (LV) and FRAME_VALID (FV). The maximum pixel rate is 30 Mp/s, corresponding to a pixel clock rate of 60 MHz. Figure 3 on page 12 shows a block diagram of the sensor core. It includes a 1.3-Mp active-pixel array. The timing and control circuitry sequences through the rows of the array, resetting and then reading each row in turn. In the time interval between resetting a row and reading that row, the pixels in the row integrate incident light. The exposure is controlled by varying the time interval between reset and readout. Once a row has been read, the data from the columns is sequenced through an analog signal chain (providing offset correction and gain), and then through an ADC. The output from the ADC is a 10-bit value for each pixel in the array.

The 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 settings. These registers are controlled by the SOC firmware and can be accessed through a two-wire serial interface. 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 and red pixels or blue and green pixels. The offset and gain stages of the analog signal chain provide per-color control of the pixel data.

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**
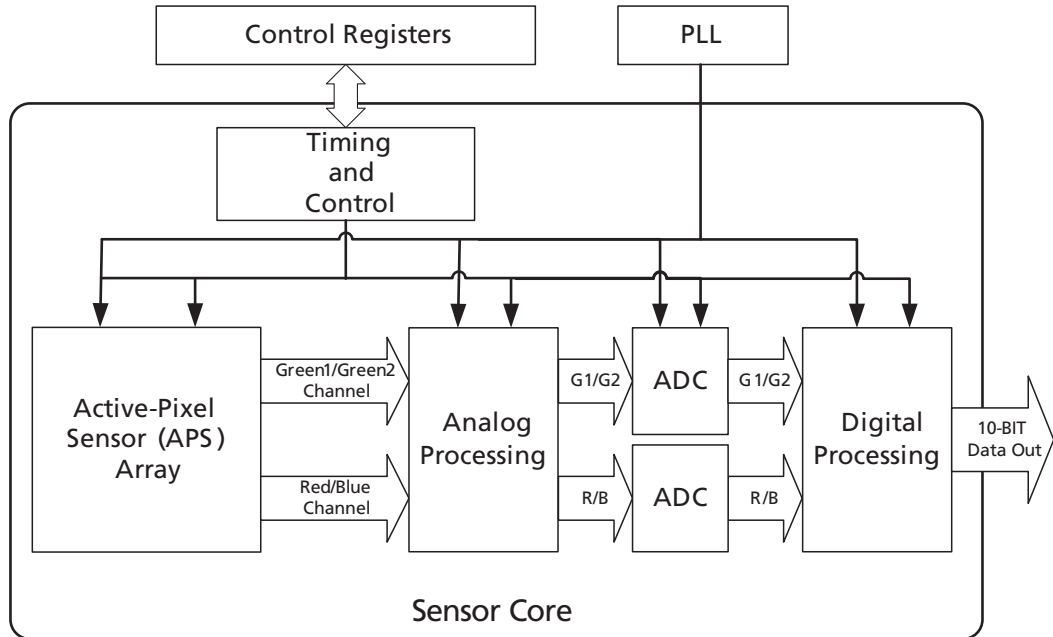
## Pixel Array

The sensor core uses a Bayer color pattern, as shown in Figure 4 on page 13. The even-numbered rows contain green and red pixels; odd-numbered rows contain blue and green pixels. Even-numbered columns contain green and blue pixels; odd-numbered columns contain red and green pixels.

**Figure 4: Pixel Color Pattern Detail (Top Right Corner)**
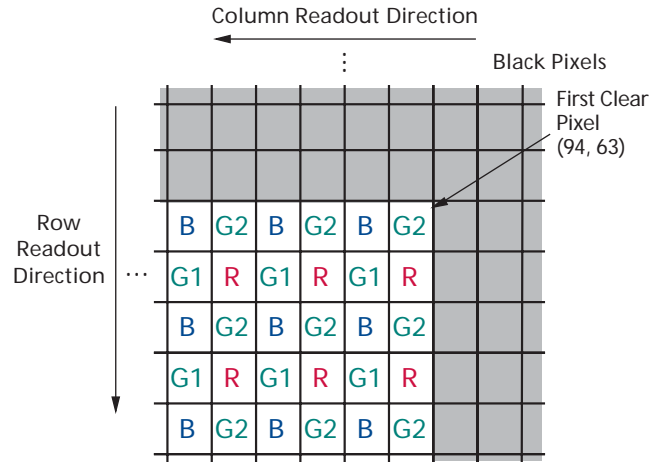


**Note:** Gr (Green R) corresponds to Green1; Gr (Green B) corresponds to Green2.

## Default Readout Order

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

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 in the case of the sensor core in the top left corner.

**Figure 5:       Imaging a Scene**
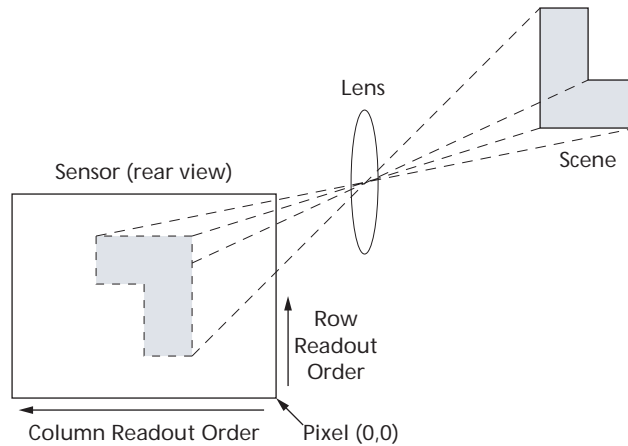


## Analog Processing

### Analog Readout Channel

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

### Gain Options

The MT9M113 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 in 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 time (exposure) of the MT9M113 is controlled by variables. While coarse integration time controls the integration duration steps of lines, fine integration time allows for sub-line accuracy. Integration time is updated in every frame by the MCU auto feature; to make manual adjustments to integration time, the MCU auto features must be disabled.

Unlike earlier Aptinaparts, setting an integration time that is greater than the frame time does not affect the frame time; for the MT9M113 the behavior is undefined. On the MT9M113, it is not necessary to reprogram the frame time in order to make longer integration times available, as the frame time adjusts automatically. Long integration times increase the likelihood of image degradation due to increased accumulation of dark current.

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

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

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

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 60 MHz (input clock from 8 through 54 MHz).

### PLL Setup

Because the input clock frequency is unknown, the sensor starts up 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.

## Digital Processing

### 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 direction and/or mirrored in the horizontal direction.

### 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 1300 x 1044 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 sensor is configured to mirror the image horizontally, 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 6 shows a sequence of 6 pixels being read out with normal readout and reverse readout. The SOC corrects for this change in sensor core output.

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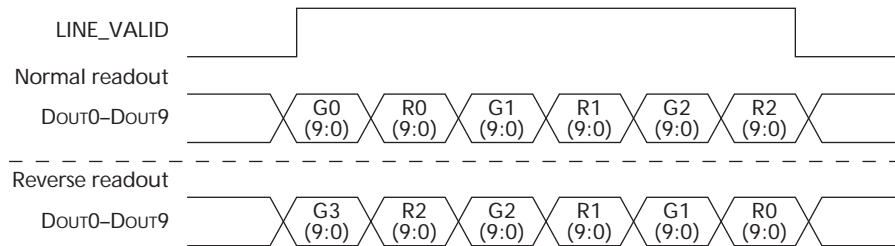


### Vertical Flip

When the sensor is configured to flip the image vertically, 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 7 shows a sequence of 6 rows being read out with normal readout and reverse readout. The SOC corrects for this change in sensor core output.

**Figure 7:    6 Rows in Normal and Row Mirror Readout Modes**
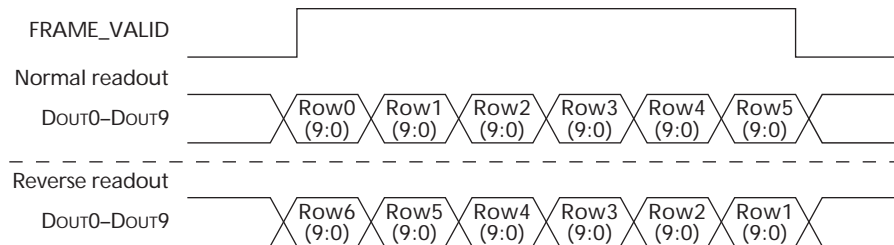


### Column and Row Skip

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. This reduces the amount of row and column data processed and is equivalent to the skip2 readout mode provided by earlier Aptina image sensors. Set the proper image output and crop sizes before enabling subsampling.

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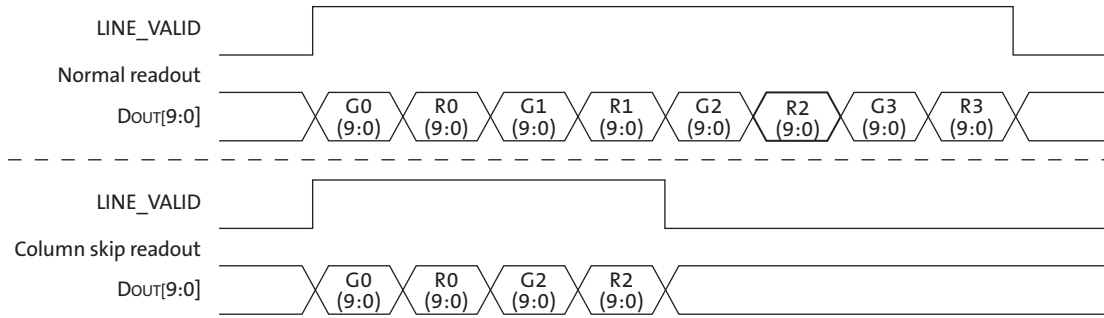


## Pixel Readouts

Figure 10 through Figure 12 on page 19 show a sequence of data being read out with no skipping, with *x_odd_inc = 3* and *y_odd_inc = 1*, with *x_odd_inc = 1* and *y_odd_inc = 3*, and with *x_odd_inc = 3* and *y_odd_inc = 3*.

**Figure 9:** **Pixel Readout (no skipping)**
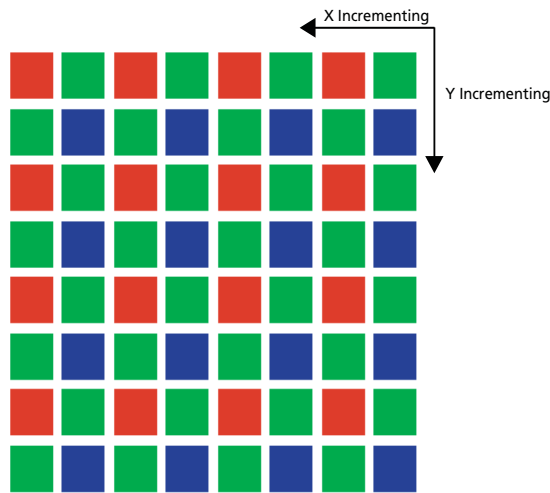


**Figure 10:** **Pixel Readout (x_odd_inc = 3, y_odd_inc = 1)**
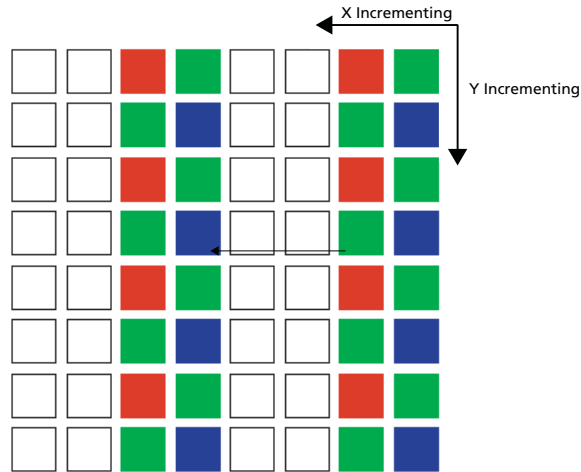
**Figure 11:**     **Pixel Readout (x_odd_inc = 1, y_odd_inc = 3)**
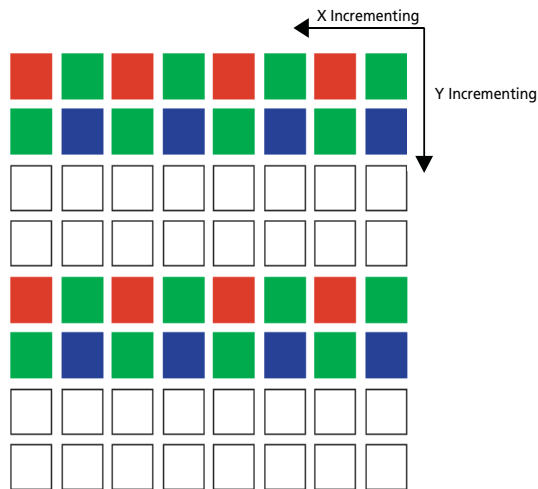
**Figure 12:  Pixel Readout (x_odd_inc = 3, y_odd_inc = 3)**
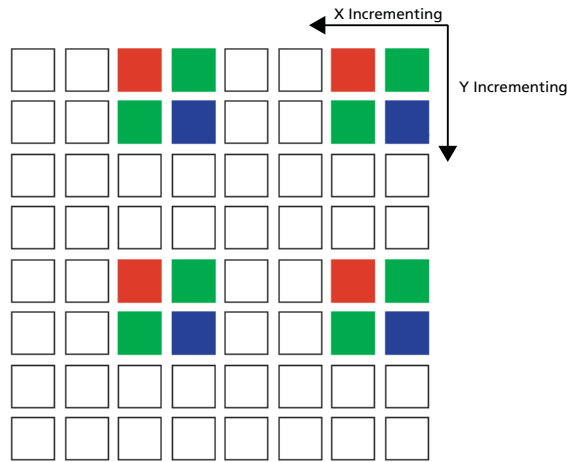


**Programming Restrictions When Skipping**

When skipping is enabled as a viewfinder mode, and the sensor is switched back and forth between full resolution and skipping, keep *line_length_pck* constant.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 must correspond with rows/ columns that form part of the subsampling sequence. To make the adjustment, use the following rules:

$$remainder = (addr\_end - addr\_start + 1)\ AND\ 4 \qquad \text{(EQ 1)}$$

$$if\ (remainder == 0)\ addr\_end = addr\_end - 2 \qquad \text{(EQ 2)}$$

Table 4 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 4:  Row Address Sequencing (Sampling)**

| Normal | Subsampled Sequence 1 | Subsampled Sequence 2 |
|--------|-----------------------|-----------------------|
| 0 | 0 | No data |
| 1 | 1 | No data |
| 2 | No data | 2 |
| 3 | No data | 3 |
| 4 | 4 | No data |
| 5 | 5 | No data |
| 6 | No data | 6 |
| 7 | No data | 7 |

**Binning**

The MT9M113 sensor core supports 2 x 1 and 2 x 2 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 register. 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 13 and in Figure 14 on page 21.

**Figure 13:      Pixel Readout (x_odd_inc = 3, y_odd_inc = 1, x_bin = 1)**
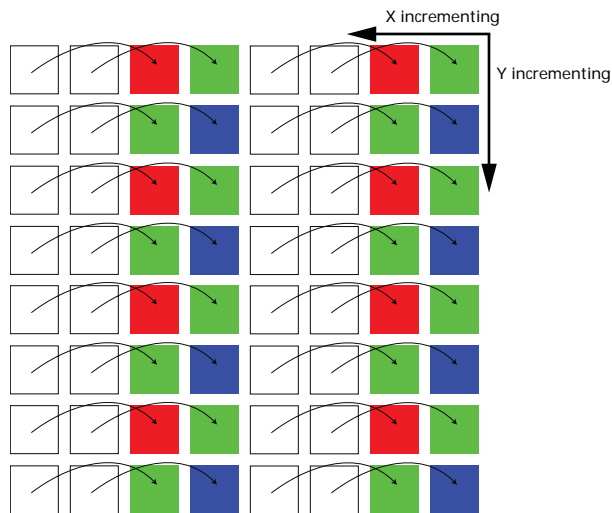
**Figure 14:** Pixel Readout (x_odd_inc = 3, y_odd_inc = 3, x_ybin = 1)
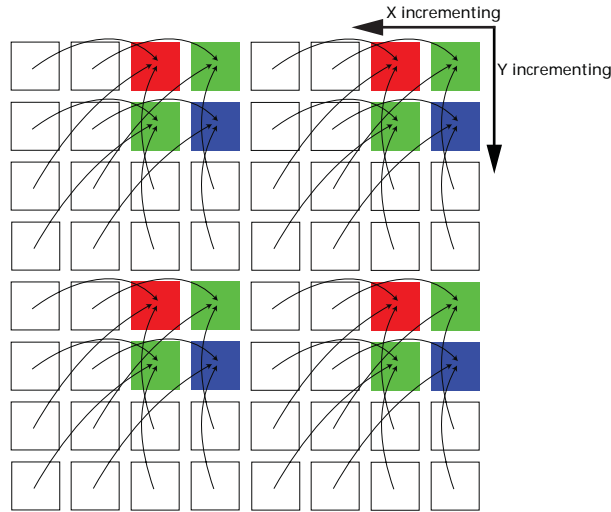


### Binning Limitations

Binning requires a 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. The row addresses for various binning modes are shown in Table 5 on page 21.

**Table 5:** Row Address Sequencing (Binning)

| Normal | Binning Sequence 1 | Binning Sequence 2 |
|--------|--------------------|--------------------|
| 0 | 0,2 | No data |
| 1 | 1,3 | No data |
| 2 | No data | 2,4 |
| 3 | No data | 3,5 |
| 4 | 4,6 | No data |
| 5 | 5,7 | No data |
| 6 | No data | 6,8 |
| 7 | No data | 7,9 |

### 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 15 on page 22. The amount of horizontal blanking and vertical blanking is programmable. LV is HIGH during the shaded region of the figure.

**Figure 15:** **Valid Image Data**

```
P₀,₀ P₀,₁ P₀,₂.........................P₀,ₙ₋₁ P₀,ₙ    00 00 00 ............. 00 00 00
P₁,₀ P₁,₁ P₁,₂.........................P₁,ₙ₋₁ P₁,ₙ    00 00 00 ............. 00 00 00


              VALID IMAGE                          HORIZONTAL
                                                    BLANKING


Pₘ₋₁,₀ Pₘ₋₁,₁...................Pₘ₋₁,ₙ₋₁ Pₘ₋₁,ₙ    00 00 00 ............. 00 00 00
Pₘ,₀ Pₘ,₁.........................Pₘ,ₙ₋₁ Pₘ,ₙ       00 00 00 ............. 00 00 00

00 00 00 ..................................... 00 00 00    00 00 00 ............. 00 00 00
00 00 00 ..................................... 00 00 00    00 00 00 ............. 00 00 00

         VERTICAL BLANKING                      VERTICAL/HORIZONTAL
                                                     BLANKING

00 00 00 ................................. 00 00 00    00 00 00 ............. 00 00 00
00 00 00 ................................. 00 00 00    00 00 00 ............. 00 00 00
```

**Raw Data Timing**

The sensor core output data is synchronized with the PIXCLK output. When LV is HIGH, one pixel's data is output on the 10-bit DOUT output bus every PIXCLK period. By default, the PIXCLK signal runs at the same frequency as the master clock, and its falling edges occur one-half of a master clock period after transitions on LV, FV, and DOUT[9:0] (see Figure 16 on page 22). This allows PIXCLK to be used as a clock to sample the data. PIXCLK is continuously enabled, even during the blanking period.

**Figure 16:** **Pixel Data Timing Example**
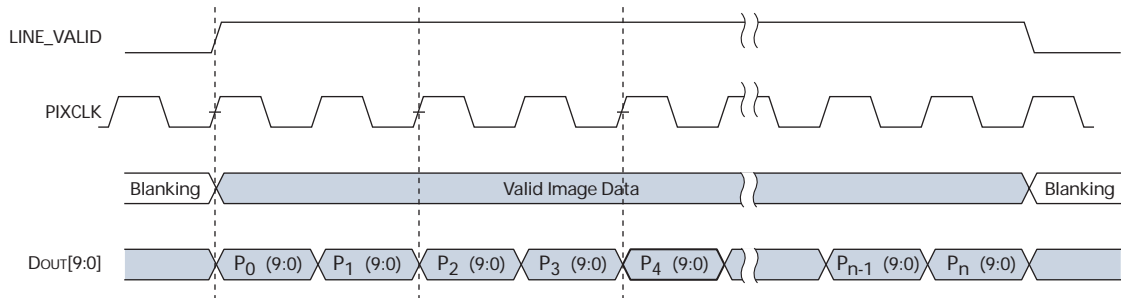
# SOC Description

## Image Flow Processor

Image and color processing in the MT9M113 are implemented as an IFP 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, as outlined in Figure 17 on page 24.

**Figure 17:    Image Flow Processor**
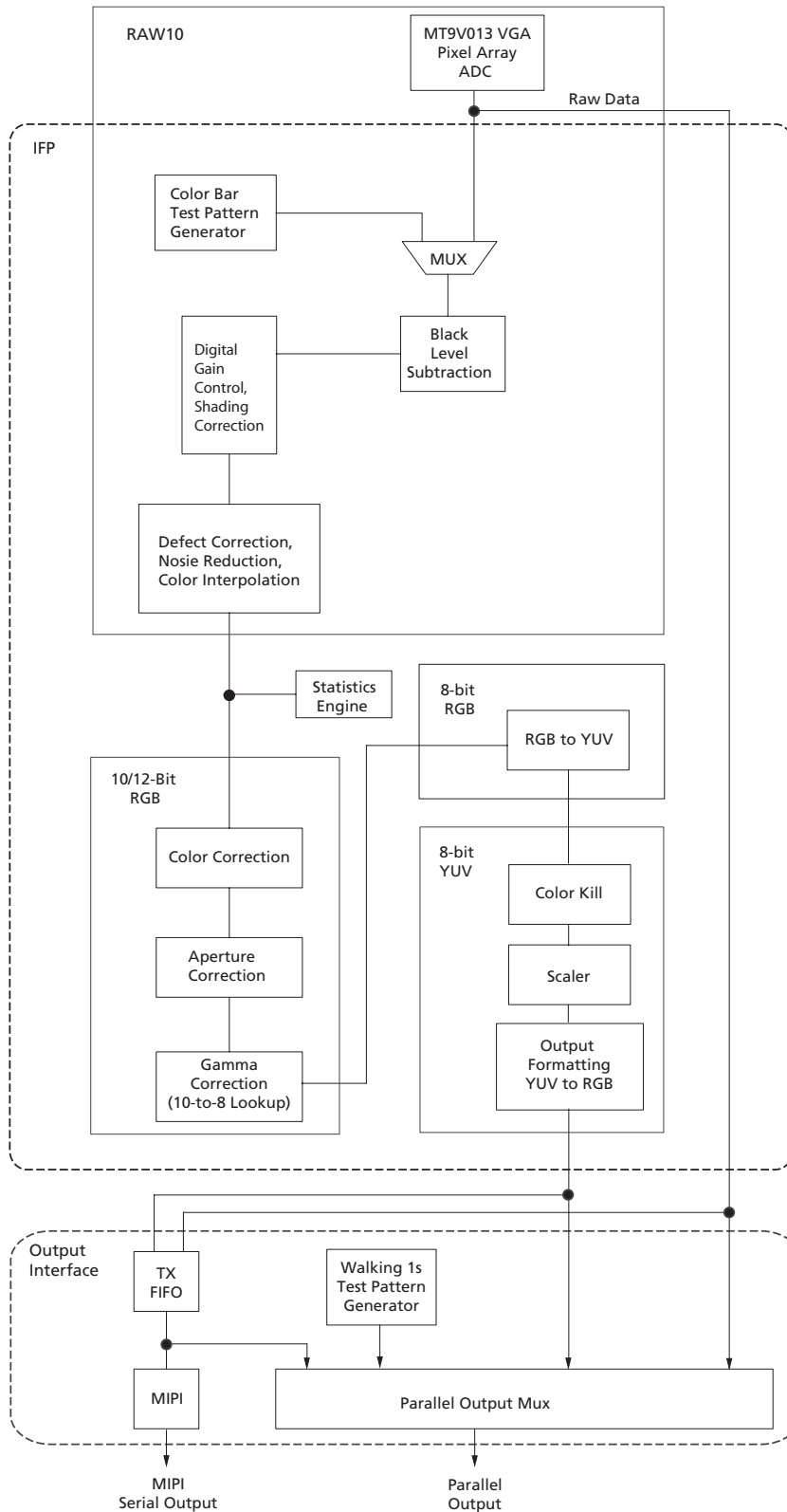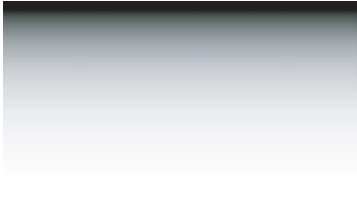
## Test Patterns

During normal operation of the MT9M113, 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 by programming a register and are shown in Figure 18. Disabling the MCU is recommended before enabling test patterns.

**Figure 18:    Color Bar Test Pattern**

| Test Pattern | Example |
|---|---|
| Flat Field | |
| Vertical Ramp | |
| Color Bar | |
| Vertical Stripes | |
| Pseudo-Random | |

## First Black Level Subtraction and Digital Gain

Image stream processing starts with black level subtraction and multiplication of all pixel values by a programmable digital gain. Both operations can be independently set to separate values for each color channel (R, Gr, Gb, B). Independent color channel digital gain can be adjusted with registers. Independent color channel black level adjustments can also be made. If the black level subtraction produces a negative result for a particular pixel, the value of this pixel is set to "0."

## Shading Correction (SC)

Lenses tend to produce images whose brightness is significantly attenuated near the edges. There are also other factors causing fixed pattern signal gradients in images captured by image sensors. The cumulative result of all these factors is known as image shading. The MT9M113 has an embedded shading correction module that can be programmed to counter the shading effects on each individual R, Gb, Gr, and B color signal.

### The Correction Function

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

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

$$P_{corrected}(row,col)=P_{sensor}(row,col)*f(row,col) \tag{EQ 3}$$

where $P$ are the pixel values and $f$ is the color dependent correction functions for each color channel.

## Defect Correction and Noise Reduction

The IFP performs continuous 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. The module is edge-aware with exposure that is based on configurable thresholds. The thresholds are changed continuously based on the brightness of the current scene. Noise reduction can be enabled and disabled and thresholds can be set through register settings.

### 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 one-color-per-pixel nature of the data stream, but after the defect correction it must be converted to a three-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 through register settings.

## Color Correction and Aperture Correction

To achieve good color fidelity of the 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 the 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 through register settings.

To increase image sharpness, a programmable 1D or 2D aperture correction (sharpening filter) is applied to color-corrected image data. The gain and threshold for 1D and 2D correction can be defined through register settings.

## Image Cropping

By configuring the cropped and output windows to various sizes, different zooming levels–for example 4X, 2X, and 1X–can be achieved. The location of the cropped window is also configurable so that panning is also supported. A separate cropped window is defined for context A and context B. In both contexts, the height and width definitions for the output window must be equal to or smaller than the cropped image.

## Gamma Correction

The gamma correction curve (as shown in Figure 19 on page 28) 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 IFP registers.

The MT9M113 IFP includes a block for gamma correction that can adjust its shape based on brightness to enhance the performance under certain lighting conditions. Two custom gamma correction tables may be uploaded, one corresponding to a brighter lighting condition, the other corresponding to a darker lighting condition. At power-up, the IFP loads the two tables with default values. The final gamma correction table used depends on the brightness of the scene and can take the form of either uploaded tables or an interpolated version of the two tables. A single (non-adjusting) table for all conditions can also be used.

**Figure 19:**     **Gamma Correction Curve**
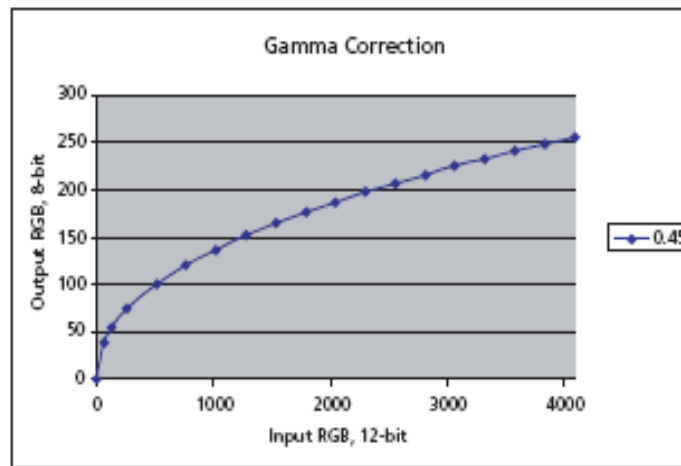


**Special Effects**

Special effects like negative image, sepia, or black and white can be applied to the data stream at this point. These effects can be enabled and selected by registers.

**RGB to YUV Conversion**

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

**Color Kill**

To remove high 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 one-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 MT9M113 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 1280 x 1024 image rendered on a 160 x 128 display can be zoomed up to eight times, since 1280/160 = 1024/128 = 8. 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.

## YUV-to-RGB/YUV Conversion and Output Formatting

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

## 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 often misused in various operating systems.

$$Y' = 0.299 \times R' + 0.587 \times G' + 0.114 \times B' \tag{EQ 4}$$

$$U' = 0.564 \times (B' - Y') + 128 \tag{EQ 5}$$

$$V' = 0.713 \times (R' - Y') + 128 \tag{EQ 6}$$

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

### Y'Cb'Cr' Using sRGB Formulas

The MT9M113 implements the sRGB standard. This option provides YCbCr coefficients for a correct 4:2:2 transmission.

**Note:** *$16 < Y601 < 235$; $16 < Cb < 240$; $16 < Cr < 240$; and $0 <= RGB <= 255$*

$$Y' = (0.2126 \times R' + 0.7152 \times G' + 0.0722 \times B') \times (219/256) + 16 \tag{EQ 7}$$

$$Cb' = 0.5389 \times (B' - Y') \times (224/256) + 128 \tag{EQ 8}$$

$$Cr' = 0.635 \times (R' - Y') \times (224/256) + 128 \tag{EQ 9}$$

**Y'U'V' Using sRGB Formulas**

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

$$Y' = 0.2126 \times R' + 0.7152 \times G' + 0.0722 \times B' \tag{EQ 10}$$

$$U' = (0.5389 \times (B' - Y') + 128 = -0.1146 \times R' - 0.3854 \times G' + 0.5 \times B' + 128) \tag{EQ 11}$$

$$V' = 0.635 \times (R' - Y') + 128 = (0.5 \times R' - 0.4542 \times G' - 0.0458 \times B' + 128) \tag{EQ 12}$$

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

$$R' = Y + 1.5748 \times V \tag{EQ 13}$$

$$G' = Y - 0.1873 \times (U - 128) - 0.4681 \times (V - 128) \tag{EQ 14}$$

$$B' = Y + 1.8556 \times (U - 128) \tag{EQ 15}$$

## Output Interface

**Parallel and MIPI Output**

The user can select to use either 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 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 MIPI output transmitter implements a serial differential sub-LVDS transmitter capable of up to 480 Mb/s. It supports multiple formats, error checking, and custom short packets.

**Table 6: Data Formats Supported by MIPI Interface**

| Data Format | Data Type |
|---|---|
| YUV 422 8-bit | 0x1E |
| RGB565 | 0x22 |
| RGB555 | 0x21 |
| RGB444 | 0x20 |
| RAW8 | 0x2A |
| RAW10 | 0x2B |

Notes: 1. Data will be packed as RAW8 if the data type specified does not match any of the above data types.

## 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. This size is large 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 20 depicts the output timing of uncompressed YUV/RGB when a scaled data stream is equalized by buffering or when no scaling takes place. The pixel clock frequency remains constant during each LV high period. Scaled data is output at a lower frequency than full size frames, which helps to reduce EMI.

**Figure 20: Timing of Uncompressed Full Frame Data or Scaled Data Passing Through the FIFO**
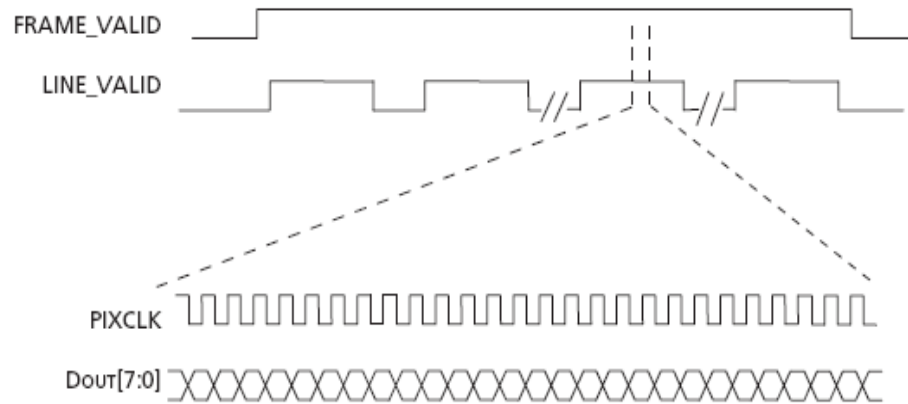


### Uncompressed YUV/RGB Data Ordering

The MT9M113 supports swapping YCrCb mode, as illustrated in Table 7.

**Table 7: YCrCb Output Data Ordering**

| Mode | Data Sequence | | | |
|---|---|---|---|---|
| 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 8. The odd and even bytes are swapped when luma/chroma swap is enabled. R and B channels are bitwise swapped when chroma swap is enabled.

**Table 8: 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 | $G_4G_3G_2B_7B_6B_5B_4B_3$ |

**Table 8:** **RGB Ordering in Default Mode (continued)**

| Mode (Swap Disabled) | Byte | $D_7D_6D_5D_4D_3D_2D_1D_0$ |
|---|---|---|
| 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 (DOUT[7:0]) and GPIO[1:0]. The GPIO pads are the lowest two bits of data.
2. Using only 8 pads (DOUT[7:0]) and a special 8 + 2 data format, shown in Table 9.

**Table 9:** **2-Byte RGB Format**

| Byte | Bits Used | Bit Sequence |
|---|---|---|
| Odd bytes | 8 data bits | $D_9D_8D_7D_6D_5D_4D_3D_2$ |
| Even bytes | 2 data bits + 6 unused bits | $0\ 0\ 0\ 0\ 0\ 0\ D_1D_0$ |

## FIFO

During normal pipeline operation, the output data rate is determined by a number of factors: 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 output them in a single burst, making sure that the data rate within the burst remains constant. This approach utilizes a free-running clock, thus making possible minimal RF interference.

## Camera Control

### General Purpose I/Os

The five GPIOs of the MT9M113 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 identify different module suppliers. In addition, module ID can be stored in the one-time programmable memory of the sensor. Information on the OTP memory can be found in "One-Time Programmable (OTP) Memory Operation."

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

GPIO[4] 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.

GPIO[2] can also be configured as inputs to be used as an OE_BAR signal for the data bus.

The general purpose inputs are enabled or disabled through register settings. Once enabled, all five inputs must be driven to valid logic levels by external signals. The state of the general purpose inputs can be read from a register.

### Output Enable Control

When the parallel pixel data interface is enabled, its signals can be switched asynchronously between being driven and High-Z under signal or register control.

## One-Time Programmable (OTP) Memory Operation

The MT9M113 has a 16-bit word OTP memory that can be utilized during module manufacturing to store specific information about the module. This feature provides system integrators and module manufacturers the ability to label and distinguish various module types based on lens, IR-cut filter, or other properties.

For more information on OTP memory, see the MT9M113 Developer Guide.

## Firmware Architecture

The firmware for the MT9M113 is implemented in multiple drivers that are responsible for different parts of operation.

**Figure 21:    Software Architecture**
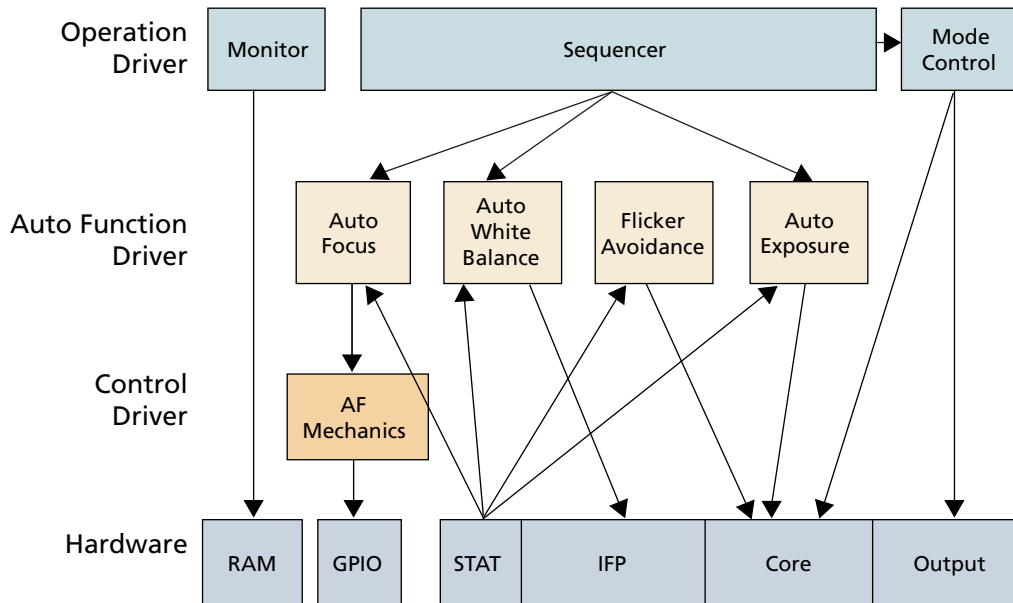


## Sequencer

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, turn on or off an external LED, and switch back to preview after capturing the frame. The user can define the sensor setup for preview and capture.

## Context and Operational Modes

The MT9M113 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. Context switching can be accomplished by sending a command through 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, and so forth. 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. The typical sequence of events after this command is:

1. 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.
2. The camera captures one or more frames of desired size. A camera equipped with a xenon flash strobes while capturing images. When capturing images is completed, the camera automatically returns to context A and resumes running in preview mode.

Note:     This sequence of events can take up to 10 frames.

**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 MT9M113 switches to the modified context B settings, while continuing to output YUV-encoded image data. AE adjusts automatically and provides a smooth continuous operation. To exit the video capture mode, the user must send another context change command causing the sensor to switch 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. Average brightness tracking (ABT)
2. Dynamic range tracking (DRT)

The average brightness tracking AE uses a constant average tracking algorithm where a target brightness value is compared to a current brightness value, and the gain and integration time are adjusted accordingly to meet the target requirement.

The dynamic range tracking AE examines the data from the statistics engine and produces a corrected brightness target so that overexposed or underexposed scene can be avoided. This also allows a manual control of the image brightness. Both algorithms are available in preview and capture modes.

**AE Driver**

This exposure mode is activated during preview. This mode can also be enabled during video capture mode. 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 the small changes. While the default settings are adequate in most situations, the user can program target brightness and other parameters described above.

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 two segments: central and peripheral. The central segment includes 4 central windows. All remaining windows belong to peripheral segment. Scene brightness is calculated as average luma in each segment taken with certain weights.

The driver changes AE parameters (integration time, gains, and so on) to drive brightness to the programmable target. The value of the single step approach to the target value can be controlled.

To avoid unwanted reaction of AE on small fluctuations of scene brightness or momentary scene changes, the AE driver uses a temporal filter for luma and a threshold around the AE luma target. The driver changes AE parameters only if the buffered luma is larger than the AE target step and pushes the luma beyond the threshold.

**Evaluative Algorithm**

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 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 oversaturation to target can take more time than transitioning from undersaturation. The AE driver has a mode that speeds up AE for overexposed scenes.

The AE driver counts the number of AE windows whose average brightness is 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 that can be defined.

**Exposure Control**

To achieve the required amount of exposure, the AE driver adjusts the sensor integration time, gains, ADC reference, and IFP digital gains. In addition, a variable is available for the user to adjust the overall brightness of the scene. To reject flicker, integration time is typically adjusted in increments of steps. The incremental step specifies the 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.

## Auto White Balance

The MT9M113 has a built-in AWB 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.

## 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 environment), 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 previous frame from corresponding samples from the current frame. Skipping more frames between subtraction can be set by a variable. The MCU then smooths the 48 sampling points, applies an amplitude threshold to avoid false detection, and looks at the resulting waveform. If flicker is present, the waveform should have a frequency within the search range. Assuming the flicker power is a sine wave, subtracting two frames results in:

$$sin(wt) - sin(wt + a) = 2 \times sin(a/2) \times cos(wt + (a/2)) \qquad \text{(EQ 16)}$$

which is a cosine wave of the original frequency.

# Two-Wire Serial Interface

The two-wire serial interface bus enables READ and WRITE access to control and status registers within the MT9M113. This interface is designed to be compatible with the MIPI Alliance Standard for Camera Serial Interface 2 (CSI-2) 1.0, which uses the electrical characteristics and transfer protocols of the two-wire serial interface specification.

The interface protocol uses a master/slave model in which a master controls one or more slave devices. The sensor acts as a slave device. The master generates a clock (SCLK) that is an input to the sensor and 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:

1. a (repeated) start condition
2. a slave address/data direction byte
3. a 16-bit register address
4. an (a no) acknowledge bit
5. two data bytes
6. a stop condition

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

### Start Condition

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

### Stop Condition

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

### Data Transfer

Data is transferred serially, 8 bits at a time, with the MSB transmitted first. Each byte of data is followed by an acknowledge bit or a no-acknowledge bit. This data transfer mechanism is used for the slave address and 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 MT9M113 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 transfers 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, eight bits at a time. The master generates an acknowledge bit after each 8-bit transfer. The slave's internal register address is 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 (see Figure 22 on page 40) starts with a dummy WRITE to the 16-bit address that is to be used for the READ. The master terminates the WRITE by generating a restart condition. The master then sends the 8-bit READ slave address/data direction byte and clocks out one byte of register data. The master terminates the READ by generating a no-acknowledge bit followed by a stop condition. Figure 22 shows how the internal register address maintained by the MT9M113 is loaded and incremented as the sequence proceeds.

**Figure 22:    Single READ from Random Location**

| Previous Reg Address, N | | | | | | | Reg Address, M | | | | M+1 |

| S | Slave Address | 0 | A | Reg Address[15:8] | A | Reg Address[7:0] | A | Sr | Slave Address | 1 | A | Read Data | $\overline{A}$ | P |
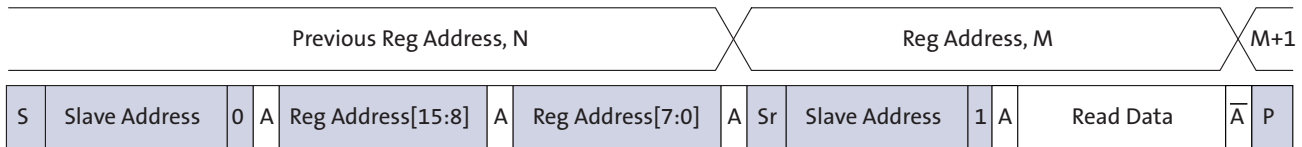
S = start condition
P = stop condition
Sr = restart condition
A = acknowledge
$\overline{A}$ = no-acknowledge

☐ slave to master
▨ master to slave

## Single READ from Current Location

This sequence (Figure 23) performs a READ using the current value of the MT9M113 internal register address. The master terminates the READ by generating a no-acknowledge bit followed by a stop condition. The figure shows two independent READ sequences.

**Figure 23:    Single Read from Current Location**

| Previous Reg Address, N | | | | | | Reg Address, N+1 | | | | | | | N+2 |

| S | Slave Address | 1 | A | Read Data | $\overline{A}$ | P | S | Slave Address | 1 | A | Read Data | $\overline{A}$ | P |
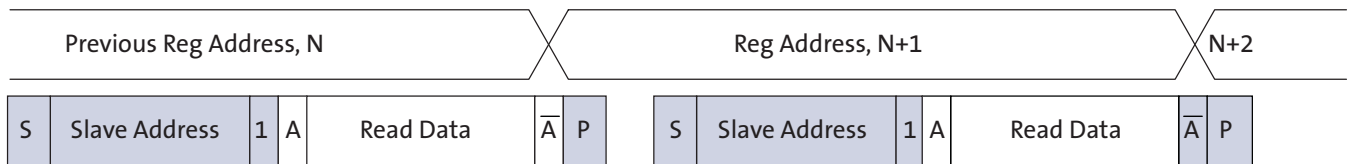
## Sequential READ, Start from Random Location

This sequence (Figure 24) starts in the same way as the single READ from random location (Figure 22 on page 40). Instead of generating a no-acknowledge bit after the first byte of data has been transferred, the master generates an acknowledge bit, and continues to perform byte reads until "L" bytes have been read.

**Figure 24:**     **Sequential READ, Start from Random Location**
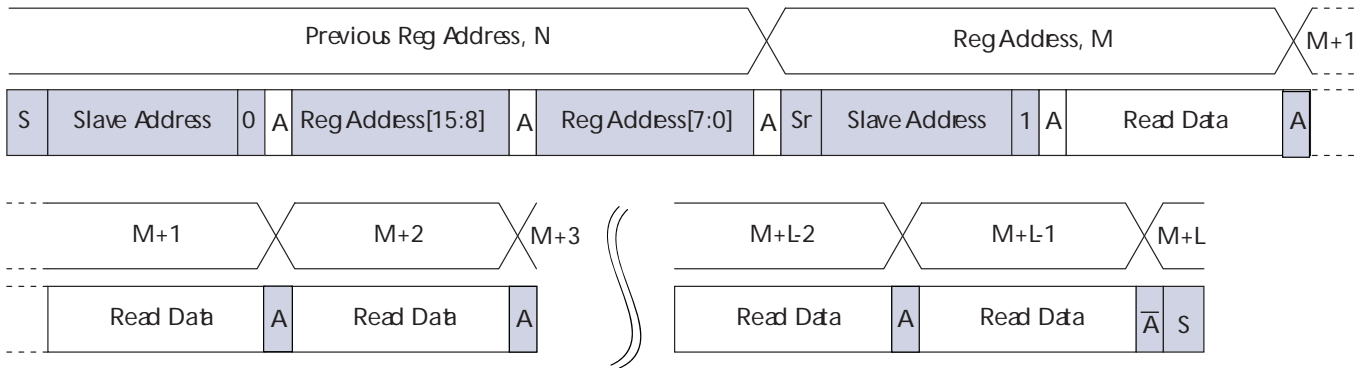
## Sequential READ, Start from Current Location

This sequence (Figure 25) starts in the same way as the single READ from current location (Figure 23 on page 40). Instead of generating a no-acknowledge bit after the first byte of data has been transferred, the master generates an acknowledge bit, and continues to perform byte reads until "L" bytes have been read.

**Figure 25:**     **Sequential READ, Start from Current Location**
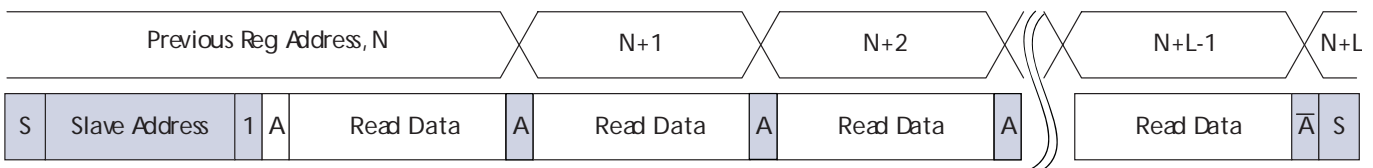
## Single WRITE to Random Location

This sequence (Figure 26) begins with the master generating a start condition. The slave address/data direction byte signals a WRITE and is followed by the high then low bytes of the register address that is to be written. The master follows this with the byte of WRITE data. The WRITE is terminated by the master generating a stop condition.

**Figure 26:**     **Single WRITE to Random Location**
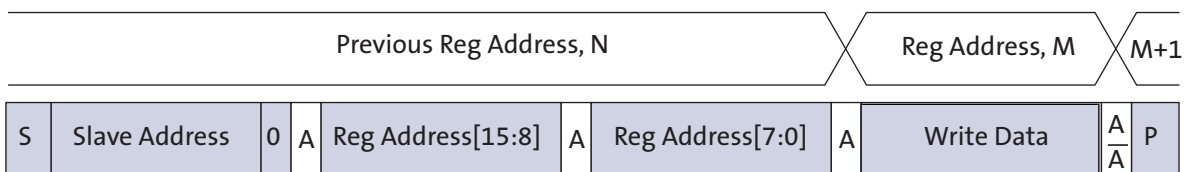
## Sequential WRITE, Start at Random Location

This sequence (Figure 27) starts in the same way as the single WRITE to random location (Figure 26 on page 41). Instead of generating a no-acknowledge bit after the first byte of data has been transferred, the master generates an acknowledge bit, and continues to perform byte writes until "L" bytes have been written. The WRITE is terminated by the master generating a stop condition.

**Figure 27:** **Sequential WRITE, Start at Random Location**
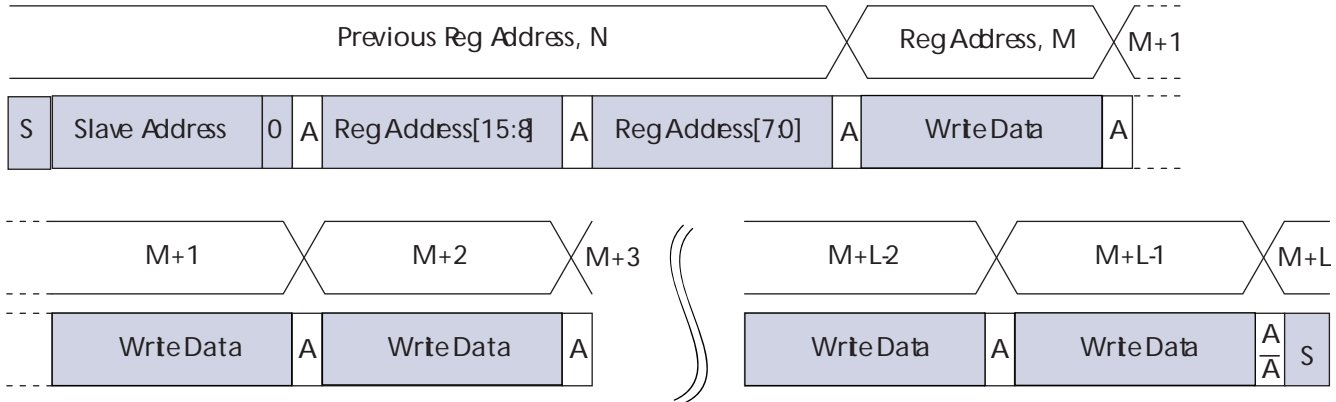
# Registers and Variables

Four types of configuration controls are available:

- Hardware registers
- Driver variables
- Special function registers (SFRs)
- MCU SRAM

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

R0x3012 or R0x3012[3:0]

These refer to two-wire accessible register number 3012. [3:0] indicate bits. Registers numbers range from 0x0000 through 0xFFFF and bits range from 15 through 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

R0x3200 – R0x33FE = SOC Page 1

R0X3400 – R0x35BE = SOC Page 2

- Page 0 contains sensor controls.
- Page 1 contains color pipeline controls.
- Page 2 contains MIPI, output FIFO and more color pipeline controls.

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 READ and WRITE 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 (from 0 through 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 through R0x098C and R0x0990. 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 R0x098C, consists of a 5-bit driver ID number and a variable offset. Examples are provided below.

To set the variable ae.Target = 0x0050 first set up the logical address, then write out the variable value:
- The variable has a driver ID of 2. Therefore, R0x098C[12:8] = 0x0002.
- The variable has an offset of 6. Therefore, R0x098C[7:0] = 0x0006.
- This is a logical access. Therefore, R0x098C[14:13] = 01.
- The size of the variable is 8 bits. Therefore, R0x098C[15] = 0x0001.
1. By combining these bits [15:0], set the logical address (R0x098C) = 0xA206.
2. Setting R0x0990 = 0x0050 writes out the value of the variable.

To read the variable ae.Target, first set up the logical address, then write out the variable value:
1. Since this is the same variable as the above example, set R0x098C = 0xA206.
2. Read R0x0990 for the current variable value.

To set the variable mon.arg1 = 0x1234, first set up the logical address, then read the variable value:
- The variable has a driver ID of 0. Therefore, R098C[12:8] = 0.
- The variable has an offset of 3. Therefore, R0x0990[7:0] = 0x0003.
- This is a logical access. Therefore, R0x098C[14:13] = 01.

- The size of the variable is 16 bits. Therefore, R0x098C[15] = 0.
1. By combining these bits [15:0], set the logical address (R0x098C) = 0x2003.
2. Set R0x0990 = 0x1234 for the value of the variable.

To read the variable mon.arg1, first set up the logical address, then read the variable value:
1. Since this is the same variable as the above example, set R0x098C = 0x2003.
2. Read R0x0990 for the current variable value.

## Special Function Registers and MCU SRAM

Special function registers are registers connected to the local bus of the microcontroller. These registers include GPIO, waveform generator, and those important for firmware operation. SFRs 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
1. Set R0x098C = 0x0400 // address
2. Set R0x0990 = 0x1234 // write 16-bit value
- Read from user SRAM
1. Set R0x098C = 0x0400 // address
2. Read R0x0990 // read 16-bit value
- Write to 8-bit GPIO register
1. Set R0x098C = 0x9079 // GPIO_DIR_L at 0x1079
2. Set R0x0990 = 0x00FE // Configure GPIO[0] as output
- Read from 8-bit GPIO register
1. Set R0x338C = 0x9079 // GPIO_DIR_L at 0x1079
2. Read R0x0990 //Check GPIO[7:0] pad state

## 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.

R0x0248[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 start. 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 R0x0248[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

Powering up the sensor requires voltages to be applied in a particular order as seen in Figure 28. The timing requirements are shown in Table 10. The sensor includes a power-on reset feature that initiates a reset upon power up, as shown in Figure 29 on page 49. Even though this feature is included on the device, it is advised that the user still manally assert a hard reset upon power up.

**Figure 28:    Power-Up Sequence**
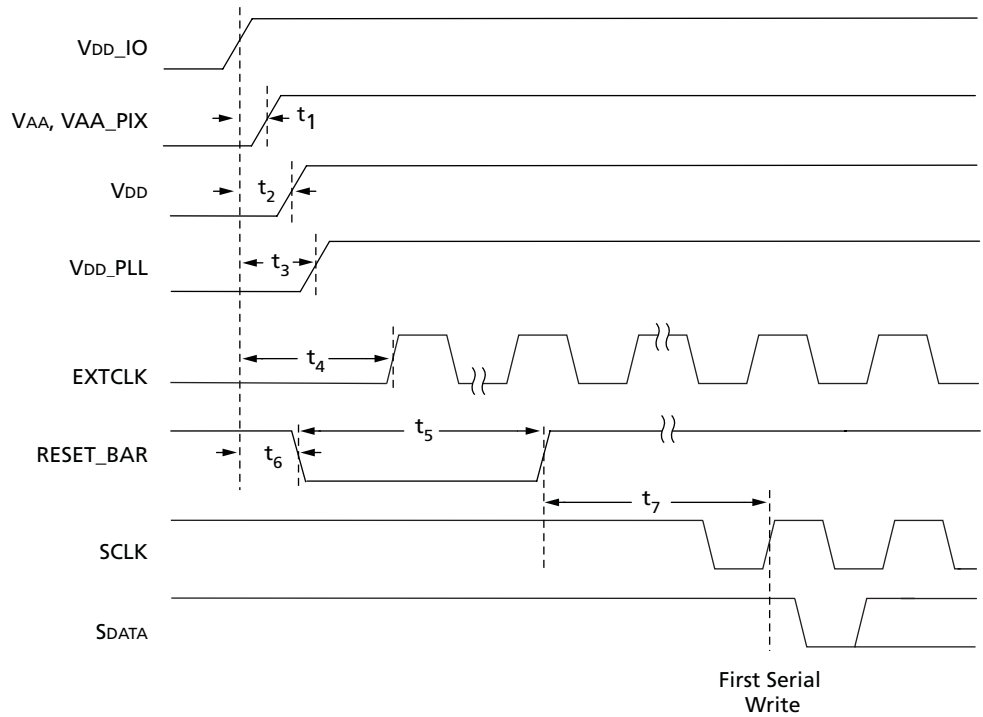


**Table 10:    Power-Up Signal Timing**

| Parameter | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| Delay from $V_{DD}$_IO to $V_{AA}$ and VAA_PIX | $t_1$ | 0 | – | 500 | ms |
| Delay from $V_{DD}$_IO to $V_{DD}$ | $t_2$ | 0 | – | 500 | ms |
| Delay from $V_{DD}$_IO to $V_{DD}$_PLL | $t_3$ | 0 | – | 500 | ms |
| EXTCLK Activation | $t_4$ | – | 500 | – | ms |
| RESET_BAR initialization time | $t_6$ | – | 2 | – | ms |
| RESET_BAR activation time | $t_5$ | – | 70 | – | EXTCLKs |
| First serial write | $t_7$ | – | 100 | – | EXTCLKs |

Notes:    1.   $V_{DD}$ and $V_{DD}$IO_TX should be tied together.

## Power On Reset

**Figure 29:    Power On Reset Sequence**
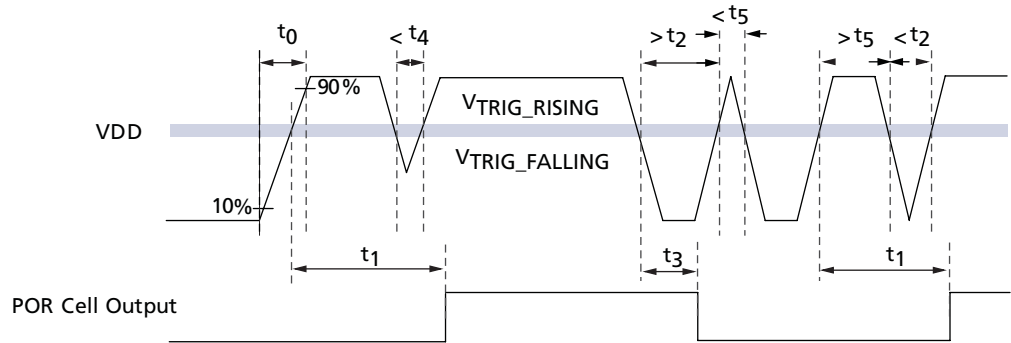


**Table 11:    POR Parameters**

| Parameter | Symbol | Condition | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $V_{DD}$ ramp time | $t_0$ | | – | – | – | ?s |
| $V_{DD}$ rising, crossing $V_{TRIG\_RISING}$ internal reset being released | $t_1$ | | – | 10 | 15 | ?s |
| $V_{DD}$ falling, crossing $V_{TRIG\_FALLING}$; internal reset active | $t_2$ | | – | 0.5 | 1 | ?s |
| Minimum $V_{DD}$ spike width below $V_{TRIG\_FALLING}$; considered to be a reset when POR cell output is HIGH | $t_3$ | | – | 0.5 | – | ?s |
| Minimum $V_{DD}$ spike width below $V_{TRIG\_FALLING}$ considered to be a reset when POR cell output is LOW | $t_4$ | | – | 1 | – | ?s |
| Minimum $V_{DD}$ spike width above $V_{TRIG\_RISING}$ considered to be a stable supply when POR cell output is LOW | $t_5$ | | – | 50 | – | ns |
| $V_{DD}$ rising trigger voltage | $V_{TRIG\_RISING}$ | | 1.15 | 1.4 | 1.55 | V |
| $V_{DD}$ falling trigger voltage | $V_{TRIG\_FALLING}$ | | 1 | 1.25 | 1.45 | V |

Note:    The hard reset and POR signals are gated, therefore, M3 ROM READ will begin after both have been de-asserted.

The $V_{DD}$ ramp time required are MAX = 10µs. If external $V_{DD}$ ramp is more than 10µs, an external reset is required.

## Reset

Two types of reset are available:
- A hard reset is issued by toggling RESET_BAR.
- A soft reset is issues by writing commands through the two-wire serial interface.

### Hard Reset

After hard reset, the output FIFO is configured for operation but disabled and all outputs are tri-stated. These outputs can be enabled through the two-wire serial interface. The hard reset signal sequence is shown in Figure 30. Hard reset timing is shown in Table 12.

**Figure 30:    Hard Reset Signal Sequence**
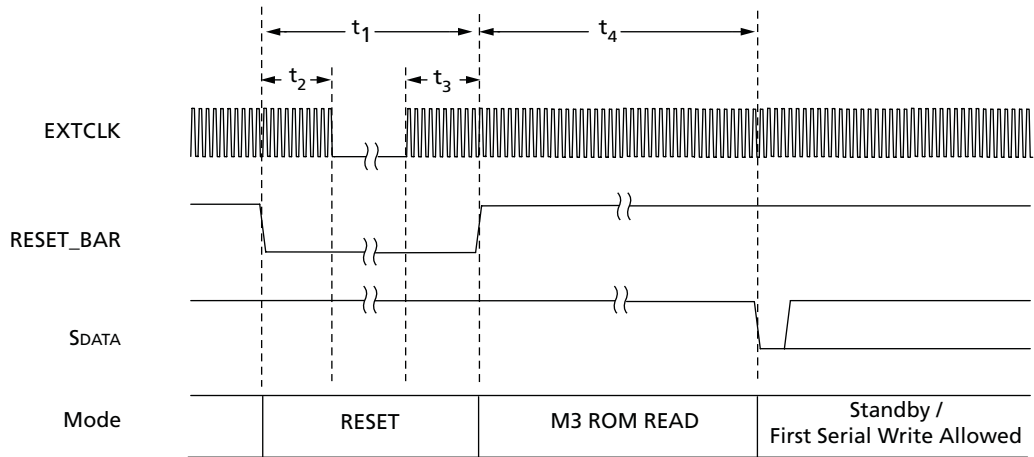


**Table 12:    Hard Reset Signal Timing**

| Parameter | Symbol | Condition | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| RESET_BAR pulse width | $t_1$ | | 70 | – | – | EXTCLKs |
| Active ECXTCLK after RESET_BAR asserted | $t_2$ | | 10 | – | – | EXTCLKs |
| Active EXTCLK before RESET_BAR de-asserted | $t_3$ | | 10 | – | – | EXTCLKs |
| Max ROM read time | $t_4$ | | – | – | 6000 | EXTCLKs |

**Soft Reset**

A soft reset sequence to the sensor has the same affect as the hard reset and can be activated writing to a register through the two-wire serial interface. The soft reset signal sequence is shown in Figure 31. Soft reset timing is shown in Table 13.

**Figure 31:     Soft Reset Signal Sequence**
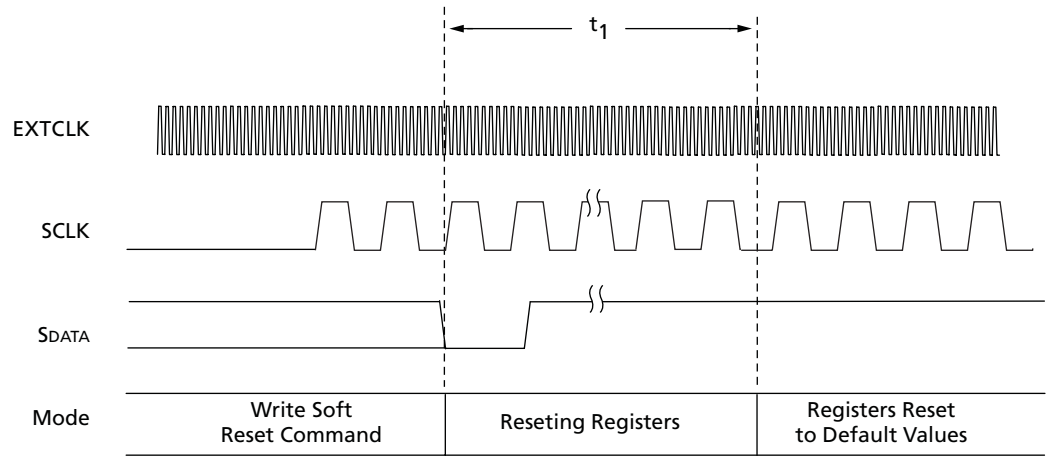


**Table 13:     Soft Reset Signal Timing**

| Parameter | Symbol | Condition | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| Active EXTCLK after soft reset command asserted | $t_1$ | | – | 6000 | – | EXTCLKs |

## Standby Modes

The MT9M113 supports two different standby modes:
- Hard standby mode
- Soft standby mode

For each mode, entry can be inhibited by programming the standby_control register.

### Hard Standby

Hard standby mode uses STANDBY to shut down digital power (VDD) and ensure the lowest power consumption. All the two-wire serial interface settings and firmware variables including patches will be lost in this mode. Starting up from this mode is equivalent to power up. The two-wire serial interface will be inactive and the sensor must be started up by de-asserting STANDBY. The hard standby signal sequence is shown in Figure 32. Hard standby timing is shown in Table 14.

**Figure 32:    Hard Standby Signal Sequence**
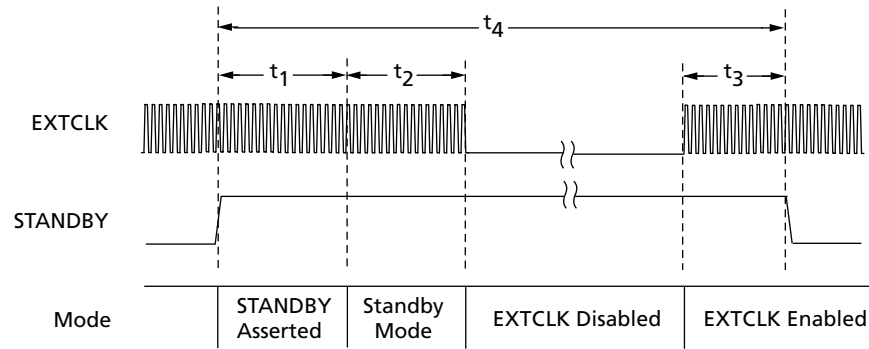


**Table 14:    Hard Standby Signal Timing**

| Symbol | Parameter | Min | Typ | Max | Unit |
|--------|-----------|-----|-----|-----|------|
| $t_1$ | Standby entry complete (0x301A[4] = 1) | 1 Frame + 20 | – | 1 Frame + 40 | EXTCLKs |
| $t_2$ | Active EXTCLK required after STANDBY asserted | 10 | – | – | EXTCLKs |
| $t_3$ | Active EXTCLK required before STANDBY de-asserted | 10 | – | – | EXTCLKs |
| $t_4$ | STANDBY pulse width | 1 Frame + 120 | – | 1 Frame + 120 | EXTCLKs |

**Soft Standby**

Soft standby can be enabled by register access and disables the sensor core and most of the digital logic. The two-wire serial interface is still active and the sensor can be programmed through register commands. All register settings and RAM content will be preserved. Soft standby can be performed in any sequencer state after all AE, AWB, histogram, and flicker calculations are finished and the sensor is in low-power mode.

The execution of standby will take place after the completion of the current line by default. It is possible to synchronize the execution of standby with the end of frame through the standby_control register. The soft standby signal sequence is shown in Figure 33. Soft standby timing is shown in Table 15.

**Figure 33:     Soft Standby Signal Sequence**
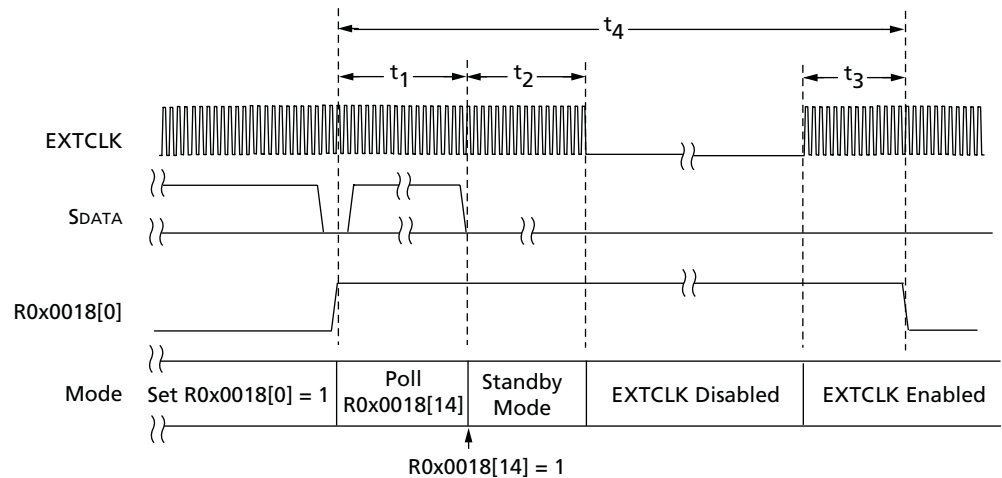


**Table 15:     Soft Standby Signal Timing**

| Symbol | Parameter | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| $t_1$ | Standby entry complete (0x301A[4] = 1) | 1 Frame + 20 | – | 1 Frame + 40 | EXTCLKs |
| $t_2$ | Active EXTCLK required after soft standby activates | 10 | – | – | EXTCLKs |
| $t_3$ | Active EXTCLK required before soft standby de-activates | 10 | – | – | EXTCLKs |
| $t_4$ | Minimum standby time | 1 Frame + 100 | – | 1 Frame + 120 | EXTCLKs |

## Signal States

**Table 16:     Status of Signals During Different States**

| Signal | Reset | Post-Reset | Standby | Standby with SHUTDOWN | Power Down |
|--------|-------|-----------|---------|----------------------|-----------|
| Dout[7:0] | High-Z | High-Z | High-Z by default (configurable through OE_BAR or two-wire serial interface register) | High-Z by default | X |
| PIXCLK | High-Z | High-Z | High-Z by default (configurable) | High-Z by default | X |
| LV | High-Z | High-Z | High-Z by default (configurable) | High-Z by default | X |
| FV | High-Z | High-Z | High-Z by default (configurable) | High-Z by default | X |
| DOUT_N | 0 | 0 | 0 | 0 | X |
| DOUT_P | 0 | 0 | 0 | 0 | X |
| CLK_N | 0 | 0 | 0 | 0 | X |
| CLK_P | 0 | 0 | 0 | 0 | X |
| GPIO[4:0] | High-Z | High-Z | Depending on how the system uses them as Dout_LSB1/Dout_LSB2/SHUTTER/FLASH/ MODULE_ID/TRIGGER/OE_BAR | Depending on how the system uses them as DoutLSB/SHUTTER/ FLASH/MODULE_ID/ TRIGGER/OE_BAR | X |
| Saddr | Input | Input | Input | Input | |
| Sdata | Input | I/O | Input | Input | |
| SCLK | Input | Input | Input | Input | |

Note:      X = "Don't Care."

## Spectral Characteristics

**Figure 34:** **Chief Ray Angle (CRA) vs. Image Height**

| CRA vs. Image Height Plot | | Image Height | | CRA |
|---|---|---|---|---|
| | | (%) | (mm) | (°) |
| | | 0 | 0 | 0 |
| | | 5 | 0.072 | 1.89 |
| | | 10 | 0.143 | 3.77 |
| | | 15 | 0.215 | 5.63 |
| | | 20 | 0.287 | 7.46 |
| | | 25 | 0.359 | 9.26 |
| | | 30 | 0.430 | 11.01 |
| | | 35 | 0.502 | 12.70 |
| | | 40 | 0.574 | 14.34 |
| | | 45 | 0.645 | 15.90 |
| | | 50 | 0.717 | 17.38 |
| | | 55 | 0.789 | 18.77 |
| | | 60 | 0.861 | 20.06 |
| | | 65 | 0.932 | 21.23 |
| | | 70 | 1.004 | 22.27 |
| | | 75 | 1.076 | 23.15 |
| | | 80 | 1.147 | 23.86 |
| | | 85 | 1.219 | 24.37 |
| | | 90 | 1.291 | 24.63 |
| | | 95 | 1.363 | 24.63 |
| | | 100 | 1.434 | 24.30 |
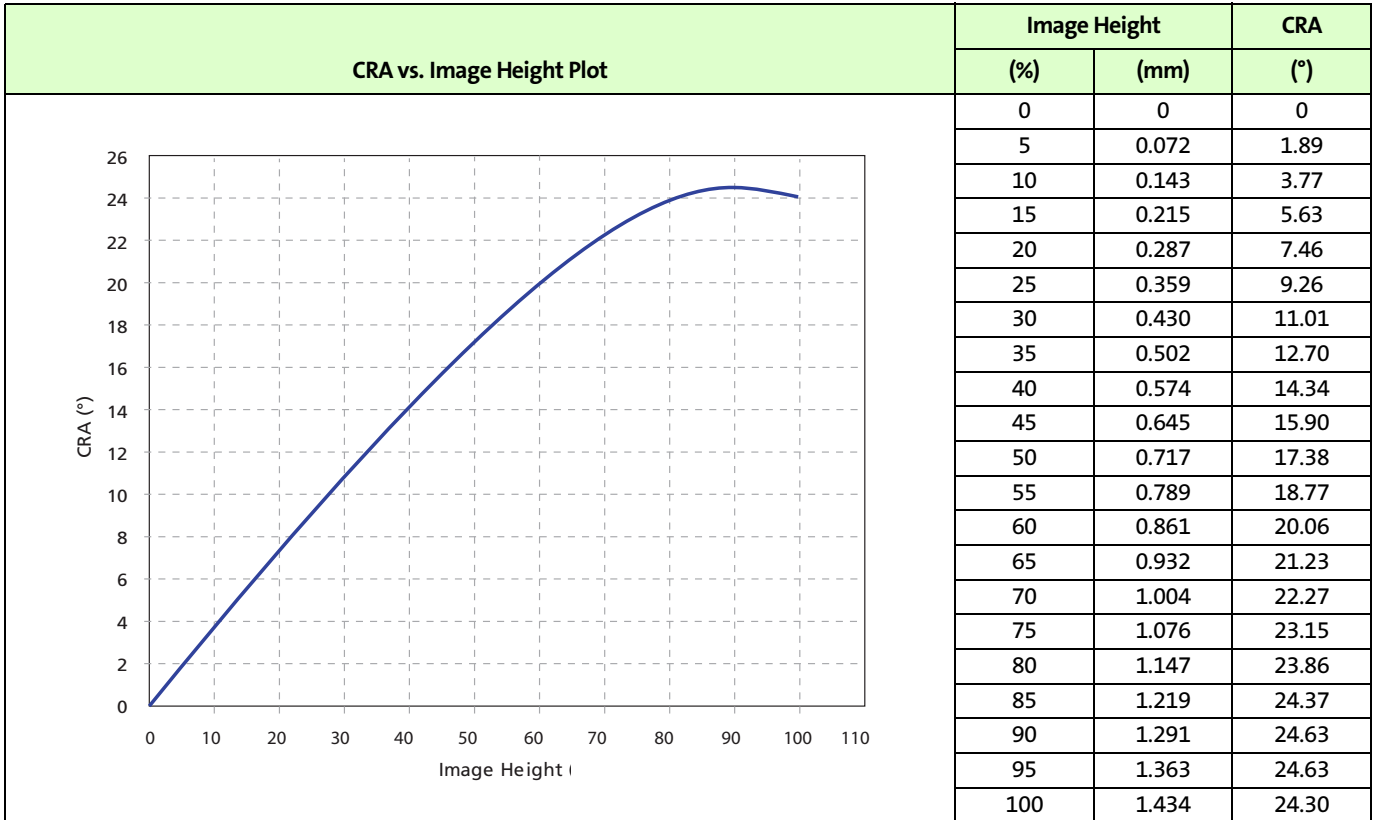
## Electrical Specifications

**Figure 35:** I/O Timing Diagram
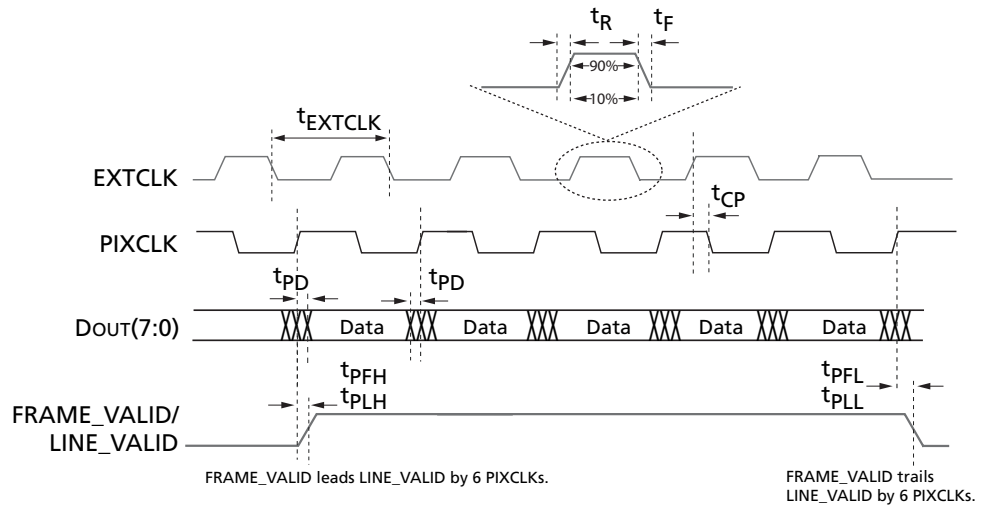


Notes:  1. PLL disabled for $^t$CP.
2. FV leads LV by 6 PIXCLKs.
3. FV trails LV by 6 PIXCLKs.

**Table 17:** I/O Timing Specifications

$^f$EXTCLK = 24 MHz; V$_{DD}$ = 1.8V; V$_{DD}$_IO = 1.8V; V$_{AA}$ = 2.8V; VAA_PIX = 2.8V;
V$_{DD}$_PLL = 2.8V; V$_{DD}$_PHY = NA; T$_J$ = 70°C; C$_{LOAD}$ = 68.5pF

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit | Note |
|---|---|---|---|---|---|---|---|
| $^f$EXTCLK | External clock frequency | PLL enabled | 8 | – | 54 | MHz | |
| $^t$EXTCLK | External clock period | PLL enabled | 18.5 | – | 125 | ns | |
| $^t$R | Input clock rise time | | – | 2 | 5 | V/ns | |
| $^t$F | Input clock fall time | | – | 2 | 5 | V/ns | |
| | Clock duty cycle | | 40 | 50 | 60 | % | |
| $^t$JITTER | Input clock jitter (peak-to-peak cycle jitter) | | – | 0.5 | 1 | ns | 1 |
| $^t$CP | EXTCLK rising to PIXCLK propagation delay | | 5 | 0.25*$^t$EXTCLK | 45 | ns | 2 |
| Output pin slew | Default slew rate | TYP V$_{DD}$_IO C$_{LOAD}$ = 30pf | – | 0.7 | – | V/ns | |
| $^f$PIXCLK | PIXCLK frequency | Default | 30 | – | 60 | MHz | 3 |
| $^t$PD | PIXCLK to data valid | Default | 0.4 * $^t$PIXCLK | 0.5 * $^t$PIXCLK | 0.6 * $^t$PIXCLK | ns | 4 |
| $^t$PFH | PIXCLK to FV HIGH | Default | 0.4 * $^t$PIXCLK | 0.5 * $^t$PIXCLK | 0.6 * $^t$PIXCLK | ns | 4 |
| $^t$PLH | PIXCLK to LV HIGH | Default | 0.4 * $^t$PIXCLK | 0.5 * $^t$PIXCLK | 0.6 * $^t$PIXCLK | ns | 4 |
| $^t$PFL | PIXCLK to FV LOW | Default | 0.4 * $^t$PIXCLK | 0.5 * $^t$PIXCLK | 0.6 * $^t$PIXCLK | ns | 4 |
| $^t$PLL | PIXCLK to LV LOW | Default | 0.4 * $^t$PIXCLK | 0.5 * $^t$PIXCLK | 0.6 * $^t$PIXCLK | ns | 4 |
| C$_{IN}$ | Input pin capacitance | | – | – | 6 | pF | |

Notes:  1. Jitter to be specified.
2. PLL disabled.

3. Should include information on adaptive pixel clock scheme if included.
4. From falling edge of PIXCLK.

**Table 18:**     **EXTCLK Electrical Specifications**

$^f$EXTCLK = 24 MHz; $V_{DD}$ = 1.8V; $V_{DD\_IO}$ = 1.8V; $V_{AA}$ = 2.8V; $V_{AA\_PIX}$ = 2.8V; $V_{DD\_PLL}$ = 2.8V;
$C_{LOAD}$ = 68.5pF; $T_J$ = 70°C

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|--------|-----------|-----------|-----|-----|-----|------|
| $^f$EXTCLK1 | Input clock frequency | | 8 | | 54 | MHz |
| $^t$EXTCLK1 | Input clock period | | 18.5 | | 125 | ns |
| $^t$R | Input clock rise time | | 0.04 | – | 1.4 | ns |
| $^t$F | Input clock fall time | | 0.04 | – | 1.4 | ns |
| VIN_AC | Input clock minimum voltage swing (AC coupled) | | 0.5 | – | – | V (p-p) |
| VIN_DC | Input clock maximum voltage (DC coupled) | | – | | $V_{DD\_IO}$ + 0.5 | V |
| $^f$CLKMAX(AC) | Input clock signalling frequency (low amplitude) | VIN = VIN_AC (MIN) | – | – | 27 | MHz |
| $^f$CLKMAX(DC) | Input clock signalling frequency (full amplitude) | VIN = $V_{DD\_IO}$ | – | – | 48 | MHz |
| | Clock duty cycle | | 45 | 50 | 55 | % |
| $^t$JITTER | Input clock jitter | Cycle to cycle | – | – | 500 | ps |
| $^t$LOCK | PLL VCO lock time | | – | 0.2 | 1 | ms |
| CIN | Input pad capacitance | | – | 3.5 | | pF |
| IIH | Input HIGH leakage current | | – | – | 10 | μA |
| IIL | Input LOW leakage current | | – | – | -10 | μA |
| VIH | Input HIGH voltage | | 0.7 * $V_{DD\_IO}$ | – | $V_{DD\_IO}$ + 0.5 | V |
| VIL | Input LOW voltage | | -0.5 | – | 0.3 * $V_{DD\_IO}$ | V |

**Table 19:** **DC Electrical Definitions and Characteristics—Parallel Mode**

$^f$EXTCLK = 24 MHz; V$_{DD}$ = 1.8V; V$_{DD}$_IO = 1.8V; V$_{AA}$ = 2.8V; VAA_PIX = 2.8V;
V$_{DD}$_PLL = 2.8V; V$_{DD}$_PHY = NA; T$_J$ = 70°C; C$_{LOAD}$ = 68.5pF

| Symbol | Parameter | Condition | Min | Typ | Max | Unit | Note |
|---|---|---|---|---|---|---|---|
| V$_{DD}$ | Core digital voltage | | 1.7 | 1.8 | 1.95 | V | |
| V$_{DD}$_IO1 | I/O digital voltage | | 1.7 | 1.8 | 1.95 | V | |
| V$_{DD}$_IO2 | I/O digital voltage | | 2.5 | 2.8 | 3.1 | V | |
| V$_{AA}$ | Analog voltage | | 2.5 | 2.8 | 3.1 | V | |
| VAA_PIX | Pixel supply voltage | | 2.5 | 2.8 | 3.1 | V | |
| V$_{DD}$_PLL | PLL supply voltage | | 2.5 | 2.8 | 3.1 | V | |
| I$_{DD}$_1 | Digital operating current | Context A/EXTCLK | – | 14 | – | mA | |
| I$_{DD}$_IO1 | I/O digital operating current (1.8V) | Context A/EXTCLK | – | 14 | – | mA | |
| I$_{DD}$_IO2 | I/O digital operating current (2.8V) | Context A/EXTCLK | – | 25 | – | mA | |
| I$_{AA}$_1 | Analog operating current | Context A/EXTCLK | – | 22 | – | mA | |
| IAA_PIX1 | Pixel supply current | Context A/EXTCLK | – | 1 | – | mA | |
| I$_{DD}$_PLL1 | PLL supply current | Context A/EXTCLK | – | NA | – | mA | |
| I$_{DD}$IO_TX | MIPI supply current | Context A/EXTCLK | – | NA | – | mA | |
| | Total supply current | Context A/EXTCLK | – | 37 | – | mA | 1 |
| | Total power consumption | Context A/EXTCLK | – | 90 | – | mW | 1 |
| I$_{DD}$_2 | Digital operating current | Context B/EXTCLK | – | 25 | – | mA | |
| I$_{DD}$_IO1 | I/O digital operating current (1.8V) | Context B/EXTCLK | – | 9 | – | mA | |
| I$_{DD}$_IO2 | I/O digital operating current (2.8V) | Context B/EXTCLK | – | 18 | – | mA | |
| I$_{AA}$_2 | Analog operating current | Context B/EXTCLK | – | 35 | – | mA | |
| IAA_PIX2 | Pixel supply current | Context B/EXTCLK | – | 1 | – | mA | |
| I$_{DD}$_PLL2 | PLL supply current | Context B/EXTCLK | – | NA | – | mA | |
| I$_{DD}$IO_TX | MIPI supply current | Context B/EXTCLK | – | NA | – | mA | |
| | Total supply current | Context B/EXTCLK | – | 61 | – | mW | 1 |
| | Total power consumption | Context B/EXTCLK | – | 146 | – | mW | 1 |
| I$_{DD}$_1 | Digital operating current | Context A/PLL | – | 18 | – | mA | |
| I$_{DD}$_IO1 | I/O digital operating current (1.8V) | Context A/PLL | – | 14 | – | mA | |
| I$_{DD}$_IO2 | I/O digital operating current (2.8V) | Context A/PLL | – | 25 | – | mA | |
| I$_{AA}$_1 | Analog operating current | Context A/PLL | – | 22 | – | mA | |
| IAA_PIX1 | Pixel supply current | Context A/PLL | – | 1 | – | mA | |
| I$_{DD}$_PLL1 | PLL supply current | Context A/PLL | – | 21 | – | mA | |
| I$_{DD}$IO_TX | MIPI supply current | Context A/PLL | – | NA | – | mA | |
| | Total supply current | Context A/PLL | – | 62 | – | mA | 1 |
| | Total power consumption | Context A/PLL | – | 156 | – | mW | 1 |
| I$_{DD}$_2 | Digital operating current | Context B/PLL | – | 28 | – | mA | |
| I$_{DD}$_IO1 | I/O digital operating current (1.8V) | Context B/PLL | – | 9 | – | mA | |
| I$_{DD}$_IO2 | I/O digital operating current (2.8V) | Context B/PLL | – | 18 | – | mA | |
| I$_{AA}$_2 | Analog operating current | Context B/PLL | – | 36 | – | mA | |
| IAA_PIX2 | Pixel supply current | Context B/PLL | – | 1 | – | mA | |
| I$_{DD}$_PLL2 | PLL supply current | Context B/PLL | – | 24 | – | mA | |
| I$_{DD}$IO_TX | MIPI supply current | Context B/PLL | – | NA | – | mA | |
| | Total supply current | Context B/PLL | – | 89 | – | mw | 1 |
| | Total power consumption | Context B/PLL | – | 222 | – | mw | 1 |
| Hard standby (clock on) | Total Standby Current when asserting the STANDBY Pin | V$_{DD}$ Disable ON | | | 20 | μA | |

**Table 19:** **DC Electrical Definitions and Characteristics—Parallel Mode (continued)**

$^f$EXTCLK = 24 MHz; VDD = 1.8V; VDD_IO = 1.8V; VAA = 2.8V; VAA_PIX = 2.8V;
VDD_PLL = 2.8V; VDD_PHY = NA; $T_J$ = 70°C; CLOAD = 68.5pF

| Symbol | Parameter | Condition | Min | Typ | Max | Unit | Note |
|---|---|---|---|---|---|---|---|
| Hard standby (clock on) | Total Standby Current when asserting the STANDBY Pin | | | | 400 | μA | |
| Hard standby (clock off) | Total Standby Current when asserting the STANDBY Pin | VDD Disable ON | | | 20 | μA | |
| Hard standby (clock off) | Total Standby Current when asserting the STANDBY Pin | | | | 400 | μA | |
| Soft standby (clock on | Total Standby Current when asserting R0x0018[0] = 1 | VDD Disable ON | | | 2500 | μA | |
| Soft standby (clock on) | Total Standby Current when asserting R0x0018[0] = 1 | VDD Disable OFF R0x0028[0] = 0 | | | 2500 | μA | |
| Soft standby (clock off) | Total Standby Current when asserting R0x0018[0] = 1 | VDD Disable ON | | | 20 | μA | |
| Soft standby (clock off) | Total Standby Current when asserting R0x0018[0] = 1 | | | | 400 | μA | |

Notes: 1. Total power consumption does not include current from VDD_IO.
2. Context A: 30 fps preview mode
Context B: 15 fps XSGA

**Table 20:** **DC Electrical Definitions and Characteristics—Serial Mode**

$^f$EXTCLK = 27 MHz; VDD = 1.8V; VDD_IO = 1.8V; VAA = 2.8V; VAA_PIX = 2.8V;
VDD_PLL = 2.8V; VDD_PHY = NA; $T_J$ = 70°C; CLOAD = 68.5pF

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit | Note |
|---|---|---|---|---|---|---|---|
| VDD | Core digital voltage | | 1.7 | 1.95 | – | V | |
| VDD_IO1 | I/O digital voltage | | 1.7 | 1.95 | – | V | |
| VDD_IO2 | I/O digital voltage | | 2.5 | 3.1 | – | V | |
| VAA | Analog voltage | | 2.5 | 3.1 | – | V | |
| VAA_PIX | Pixel supply voltage | | 2.5 | 3.1 | – | V | |
| VDD_PLL | PLL supply voltage | | 2.5 | 3.1 | – | V | |
| IDD_1 | Digital operating current | Context A/PLL | – | 22 | – | mA | |
| IDD_IO1 | I/O digital operating current (1.8V) | Context A/PLL | – | 6 | – | mA | |
| IDD_IO2 | I/O digital operating current (2.8V) | Context A/PLL | – | 12 | – | mA | |
| IAA_1 | Analog operating current | Context A/PLL | – | 26 | – | mA | |
| IAA_PIX1 | Pixel supply current | Context A/PLL | – | 1 | – | mA | |
| IDD_PLL1 | PLL supply current | Context A/PLL | – | 13 | – | mA | |
| IDDIO_TX | MIPI supply current | Context A/PLL | – | 2 | – | mA | |
| | Total supply current | Context A/PLL | – | 64 | – | mA | 1 |
| | Total power consumption | Context A/PLL | – | 156 | – | mW | 1 |
| IDD_2 | Digital operating current | Context B/PLL | – | – | – | mA | |
| IDD_IO1 | I/O digital operating current (1.8V) | Context B/PLL | – | 9 | – | mA | |
| IDD_IO2 | I/O digital operating current (2.8V) | Context B/PLL | – | 17 | – | mA | |
| IAA_2 | Analog operating current | Context B/PLL | – | 43 | – | mA | |
| IAA_PIX2 | Pixel supply current | Context B/PLL | – | 1 | – | mA | |
| IDD_PLL2 | PLL supply current | Context B/PLL | – | 13 | – | mA | |
| IDDIO_TX | MIPI supply current | Context B/PLL | – | 5 | – | mA | |

**Table 20:** **DC Electrical Definitions and Characteristics—Serial Mode (continued)**

$^f$EXTCLK = 27 MHz; $V_{DD}$ = 1.8V; $V_{DD}$_IO = 1.8V; $V_{AA}$ = 2.8V; $V_{AA}$_PIX = 2.8V;
$V_{DD}$_PLL = 2.8V; $V_{DD}$_PHY = NA; $T_J$ = 70°C; $C_{LOAD}$ = 68.5pF

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit | Note |
|---|---|---|---|---|---|---|---|
| | Total supply current | Context B/PLL | – | 94 | – | mw | 1 |
| | Total power consumption | Context B/PLL | – | 226 | – | mw | 1 |
| Hard standby (clock on) | Total Standby Current when asserting the STANDBY Pin | R0x0028[0] = 1 | – | 20 | – | μA | |
| Hard standby (clock on) | Total Standby Current when asserting the STANDBY Pin | | – | 400 | – | μA | |
| Hard standby (clock off) | Total Standby Current when asserting the STANDBY Pin | R0x0028[0] = 1 | – | 20 | – | μA | |
| Hard standby (clock off) | Total Standby Current when asserting the STANDBY Pin | | – | 400 | – | μA | |
| Soft standby (clock on) | Total Standby Current when asserting R0x0018[0] = 1 | $V_{DD}$ Disable ON | – | 2500 | – | μA | |
| Soft standby (clock on) | Total Standby Current when asserting R0x0018[0] = 1 | $V_{DD}$ Disable OFF R0x0028[0] = 0 | – | 2500 | – | μA | |
| Soft standby (clock off) | Total Standby Current when asserting R0x0018[0] = 1 | $V_{DD}$ Disable ON | – | 20 | – | μA | |
| Soft standby (clock off) | Total Standby Current when asserting R0x0018[0] = 1 | | – | 400 | – | μA | |

Notes: 1. Total power consumption does not include current from $V_{DD}$_IO.

**Table 21:** **I/O Parameters**

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $V_{IH}$ | Input HIGH voltage | At specified $I_{IN}$ | 0.7 * $V_{DD}$_IO | | $V_{DD}$_IO + 0.5 | V |
| $V_{IL}$ | Input LOW voltage | $V_{DD}$_IO = 2.8V at specified $I_{IN}$ | −0.5 | | 0.6 | V |
| | | $V_{DD}$_IO = 1.8V at specified $I_{IN}$ | −0.5 | | 0.4 | |
| $I_{IN}$ | Input leakage current | No pull-up resistor; $V_{IN}$ = $V_{DD}$ or $D_{GND}$ | −10 | | 10 | μA |
| $V_{OH}$ | 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 $V_{OH}$, $V_{DD}$_IO = 1.8V | – | | 13 | mA |
| | | At specified $V_{OH}$, $V_{DD}$_IO = 2.8V | – | | 20 | mA |
| $I_{OL}$ | Output LOW current | At specified $V_{OL}$, $V_{DD}$_IO = 0.1V | −15 | | – | mA |
| | | At specified $V_{OL}$, $V_{DD}$_IO = 0.4V | −25 | | 15 | mA |
| $I_{OZ}$ | Tri-state output leakage current | | | | 10 | μA |

PDF: 5223098994/Source: 9024576921
MT9M113_DS - Rev. J Pub. 6/12 EN

60

**Caution**    Table 22 shows stress ratings only, and functional operation of the device at these or any other conditions above those indicated in the product specification is not implied. Stresses above those listed may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

**Table 22:     Absolute Maximum Ratings**

| Symbol | Parameter | Rating | | Unit |
|---|---|---|---|---|
| | | Min | Max | |
| $V_{DD\_MAX}$ | Core digital voltage | −0.3 | 2.4 | V |
| $V_{DD\_IO\_MAX}$ | I/O digital voltage | −0.3 | 4.0 | V |
| $V_{AA\_MAX}$ | Analog voltage | −0.3 | 4.0 | V |
| $V_{AA\_PIX\_MAX}$ | Pixel supply voltage | −0.3 | 4.0 | V |
| $V_{DD\_PLL\_MAX}$ | PLL supply voltage | −0.3 | 4.0 | V |
| $V_{IH\_MAX}$ | Input HIGH voltage | −0.3 | $V_{DD}\_IO + 0.3$ | V |
| $V_{IL\_MAX}$ | Input LOW voltage | −0.3 | − | V |
| T_OP | Operating temperature (measure at junction) | −30 | 75 | °C |
| T_STG | Storage temperature | −40 | 85 | °C |

## Two-Wire Serial Bus Timing

Figure 36 and Table 23 on page 63 illustrate the timing for the two-wire serial interface.

**Figure 36:     Two-Wire Serial Bus Timing Parameters**
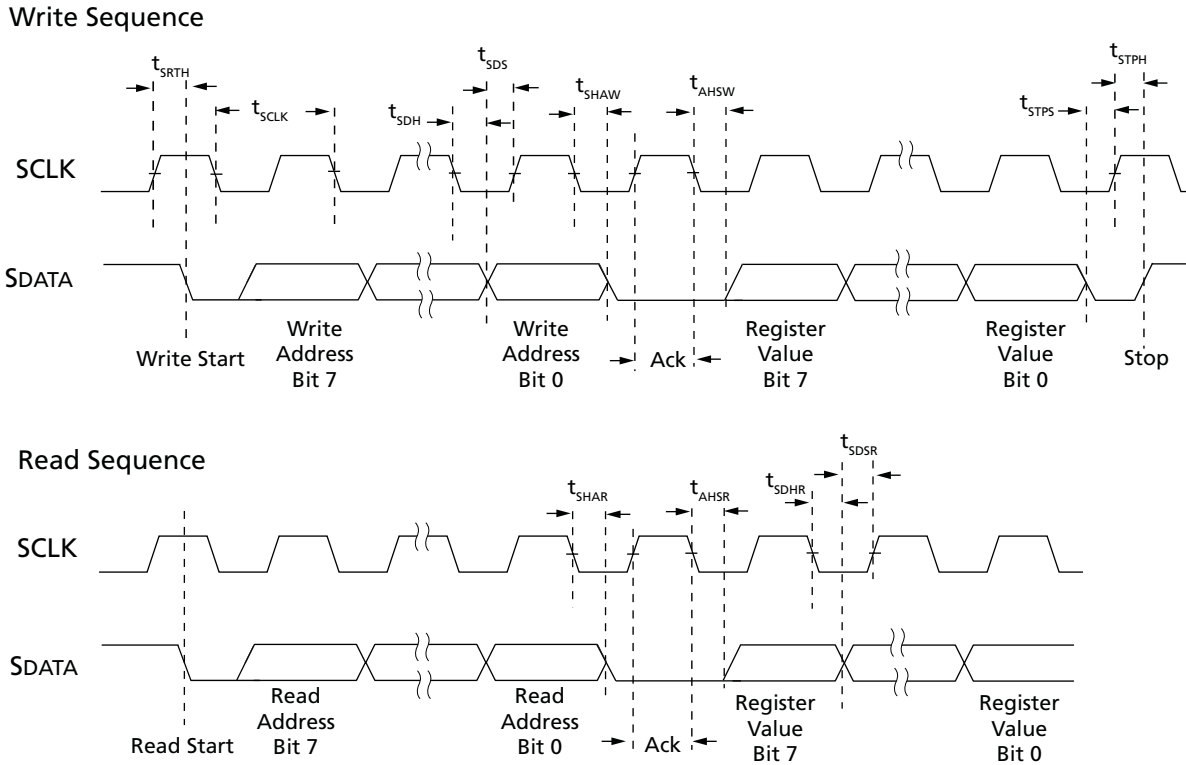
**Table 23: Two-Wire Serial Bus Characteristics**

$^f$EXTCLK = 24 MHz; V$_{DD}$ = 1.8V; V$_{DD\_IO}$ = 1.8V; V$_{AA}$ = 2.8V; VAA_PIX = 2.8V;
V$_{DD\_PLL}$ = 2.8V; V$_{DD\_PHY}$ = NA; T$_J$ = 70°C; C$_{LOAD}$ = 68.5pF

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit | Note |
|---|---|---|---|---|---|---|---|
| $^f$SCLK | Serial interface input clock frequency | | 100 | – | 400 | kHz | |
| $^t$SCLK | Serial interface input clock period | | 2.5 | – | 10 | MS | Master clock cycle units or PLL cycles if enabled |
| | SCLK duty cycle | | 40 | 50 | 60 | % | |
| $^t$SRTH | Start hold time | Write/Read | 0.6 | – | – | μs | Master clock cycle units or PLL cycles if enabled |
| $^t$SDH | SDATA hold | Write | 0.6 | – | – | μs | Master clock cycle units or PLL cycles if enabled |
| $^t$SDS | SDATA setup | Write | 0.6 | – | – | μs | Master clock cycle units or PLL cycles if enabled |
| $^t$SHAW | SDATA hold to ack | Write | 0.6 | – | – | μs | Master clock cycle units or PLL cycles if enabled |
| $^t$AHSW | Ack hold to SDATA | Write | 0.6 | – | – | μs | Master clock cycle units or PLL cycles if enabled |
| $^t$STPS | Stop setup time | Write/Read | 0.6 | – | – | μs | Master clock cycle units or PLL cycles if enabled |
| $^t$STPH | Stop hold time | Write/Read | 0.6 | – | – | μs | Master clock cycle units or PLL cycles if enabled |
| $^t$SHAR | SDATA hold to ack | Read | 0.3 | – | 1 | μs | Master clock cycle units or PLL cycles if enabled |
| $^t$AHSR | Ack hold to SDATA | Read | 0.3 | – | 1 | μs | Master clock cycle units or PLL cycles if enabled |
| $^t$SDHR | SDATA hold | Read | 0.3 | – | 1 | μs | Master clock cycle units or PLL cycles if enabled |
| $^t$SDSR | SDATA setup | Read | 0.1 | – | 1 | μs | Master clock cycle units or PLL cycles if enabled |
| SDATA Slew Rate | No programmable skew | TYP V$_{DD\_IO}$ C$_{LOAD}$= 35pF | 0.3 | 0.7 | – | V/ns | |
| C$_{IN\_SI}$ | Serial interface input pin capacitance | | – | 6 | – | pF | |
| C$_{LOAD\_SD}$ | SDATA max load capacitance | | – | – | 30 | pF | |
| R$_{SD}$ | SDATA pull-up resistor | | – | 1.5 | – | KΩ | |

## Revision History

**Rev. J** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **6/12/12**
- Applied updated Aptina template
- Restored Figures 23, 24, 25, 26, and 27 whose links got broken in previous release
- Deleted extra blank page

**Rev. H** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **4/19/12**
- Updated trademarks
- Applied updated Aptina template
- Updated active pixel array size in "General Description" on page 7
- Restored missing Figure 8: "8 Pixels in Normal and Column Skip 2X Readout Modes," on page 17
- Updated "Y'U'V' Using sRGB Formulas" on page 30
- Updated Figure 28: "Power-Up Sequence," on page 48
- • Updated Table 11, "POR Parameters," on page 49

**Rev. G** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **10/22/10**
- Updated to latest Aptina template (fixed hyperlinks, updated trademarks)
- Fixed micron symbols that got corrupted in last update

**Rev. F** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **2/4/09**
- Updated to Aptina template; transferred register tables to separate document

**Rev. E** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **3/18/08**
- Updated Table 14, "Hard Standby Signal Timing," on page 52
- Updated Table 15, "Soft Standby Signal Timing," on page 53
- Updated document to Production

**Rev. D** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **12/13/07**
- Revised "One-Time Programmable (OTP) Memory Operation" on page 33.
- Deleted measurement window from last sentence of 2nd paragraph of "AE Driver" on page 36.
- Removed ae_min_index from Table 16 and Table 28.
- Added R0x0020 and R0x0024 to Table 20 on page 57.
- Added note to R0x3402 bit 0.
- Removed "reserved" status of R0x0055 and added information for bits 4:0.
- Changed Max value to 400 for the following in Table 19, "DC Electrical Definitions and Characteristics—Parallel Mode," on page 58 and Table 20, "DC Electrical Definitions and Characteristics—Serial Mode," on page 59:
  - Hard standby (clock on), Total Standby Current when asserting the STANDBY Pin
  - Hard standby (clock off), Total Standby Current when asserting the STANDBY Pin
  - Soft standby (clock off), Total Standby Current when asserting R0x0018[0] = 1

**Rev. C** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **5/10/07**
- Updated document from Advance to Preliminary.
- Added power consumption at preview mode and standby current in Table 1.
- Modified procedure in "One-Time Programmable (OTP) Memory Operation" on page 33.
- Modified Figure 21 on page 34.
- Deleted note for Table 10 on page 48.

- Replaced Figure 31 on page 51.
- Modified assumptions for Table 17 on page 56 through Table 23 on page 63.
- Modified Table 17 on page 56.
- Added Table 18, EXTCLK Electrical Specifications, on page 57.
- Modified Table 19, DC Electrical Definitions and Characteristics—Parallel Mode, on page 58.
- Deleted Table 45, Power Consumptions at Different Resolutions.

**Rev. B** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **3/13/07**

- Updated Table 10 on page 48 through Table 33 on page 101 with new data from RegDB. Edited RegDB format.
- EXTCLK minimum changed from 6 MHz to 8 MHz in various places.
- Updated electrical timing from Figure 28 on page 48 through Figure 36 on page 62.
- Updated electrical data from Table 10 on page 48 through Table 23 on page 63.
- Added DigitalClarity to trademarks on last page.

**Rev. A** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **10/06**

- Initial release