1/3.2-Inch System-On-A-Chip (SOC) CMOS Digital Image Sensor

MT9D131 Datasheet, Rev. J

For the latest MT9D131 data sheet, please visit www.onsemi.com

Features

- Superior low-light performance
- Ultra-low-power, cost effective
- Internal master clock generated by on-chip phase-locked loop oscillator (PLL)
- Electronic rolling shutter (ERS), progressive scan
- Integrated image flow processor (IFP) for single-die camera module
- Automatic image correction and enhancement, including lens shading correction
- Arbitrary image decimation with anti-aliasing
- Integrated real-time JPEG encoder
- Integrated microcontroller for flexibility
- Two-wire serial interface providing access to registers and microcontroller memory
- Selectable output data format: ITU-R BT.601 (YCbCr), 565RGB, 555RGB, 444RGB, JPEG 4:2:2, JPEG 4:2:0, and raw 10-bit
- Output FIFO for data rate equalization
- Programmable I/O slew rate

Applications

- Network Security Cameras
- ePTZ cameras
- Wireless cameras
- Consumer video products
- High resolution security camera

Table 1: Key Performance Parameters

Parameter		Value
Optical format		1/3.2-inch (4:3)
Full resolution		1600 x 1200 pixels (UXGA)
Pixel size		2.8 µm x 2.8 µm
Active pixel array area		4.73 mm x 3.52 mm
Shutter type		Electronic rolling shutter (ERS)
Maximum frame rate		15 fps at full resolution, 30 fps in preview mode, (800 x 600)
Maximum data rate/ master clock		80 Mp/s 6–80 MHz
Supply voltage	Analog	2.5–3.1 V
	Digital	1.7–1.95 V
	I/O	1.7–3.1 V
	PLL	2.5–3.1 V
ADC resolution		10-bit, on-die
Responsivity		1.0 V/lux-sec (550 nm)
Dynamic range		71 dB
SNR_{MAX}		42.3 dB
Power consumption		348 mW at 15 fps, full resolution
		223 mW at 30 fps, preview mode
Operating temperature		–30°C to +70°C
Package		48-pin CLCC

Ordering Information

Table 2: Available Part Numbers

Part Number	Product Description	Orderable Product Attribute Description
MT9D131C12STC-DP	2.0 MP 1/3" SOC	Dry Pack with Protective Film
MT9D131C12STC-DR	2.0 MP 1/3" SOC	Dry Pack without Protective Film
MT9D131C12STC-TP	2.0 MP 1/3" SOC	Tape & Reel with Protective Film
MT9D131D00STCK15LC1-305	2.0 MP 1/3" SOC	Die Sales, 305 μm Thickness

Table of Contents

List of Figures

List of Tables

General Description

The ON Semiconductor MT9D131 is a 1/3.2 inch, 2 Mp CMOS image sensor with an integrated advanced camera system. The camera system features a microcontroller (MCU) and a sophisticated image flow processor (IFP) with a real-time JPEG encoder.

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 1668 x 1248 pixels, programmable timing and control circuitry including a PLL, analog signal chain with automatic offset correction and programmable gain, and two 10-bit A/D converters (ADC). The entire system-on-a-chip (SOC) has ultra-low power requirements and superior low-light performance that is particularly suitable for surveillance applications.

The excellent low-light performance of MT9D131 is one of the hallmarks of ON Semiconductor's breakthrough low-noise CMOS imaging technology that achieves CCD image quality (based on signal-to-noise ratio and low-light sensitivity) while maintaining the inherent size, cost, power consumption, and integration advantages of CMOS.

Feature Overview

The MT9D131 is a color image sensor with a Bayer color filter arrangement. Its basic characteristics are described in Table 1 on page 1.

The MT9D131 has an embedded phase-locked loop oscillator (PLL) that can be used with the common wireless system clock. When in use, the PLL adjusts the incoming clock frequency, allowing the MT9D131 to run at almost any desired resolution and frame rate. To reduce power consumption, the PLL can be bypassed and powered down.

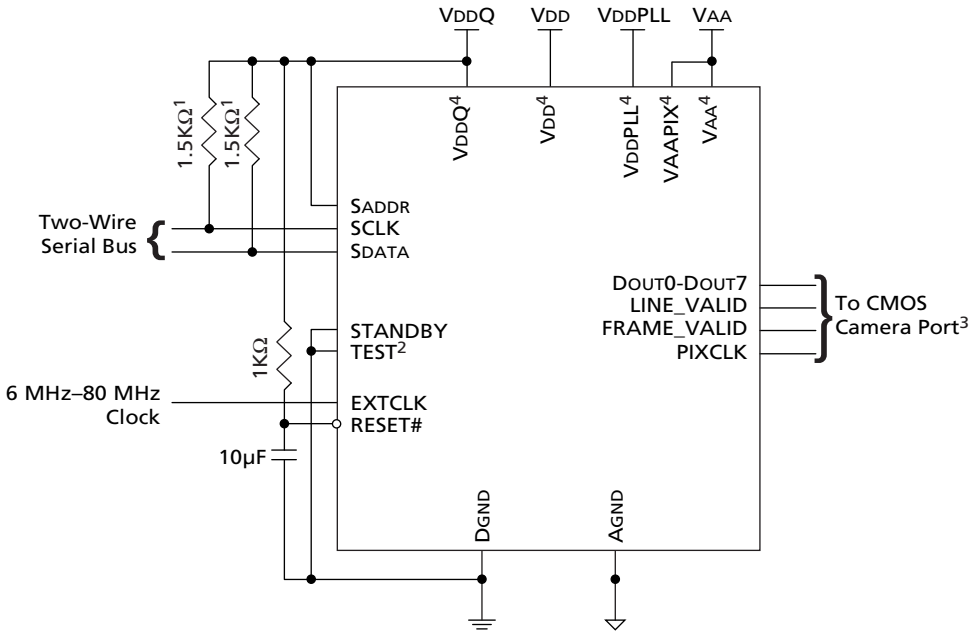
Low power consumption is a very important requirement for all components of wireless devices. The MT9D131 has numerous power conserving features, including an ultra low-power standby mode and the ability to individually shut down unused digital blocks.

Another important consideration for wireless devices is their electromagnetic emission or interference (EMI). The MT9D131 has a programmable I/O slew rate to minimize its EMI and an output FIFO to eliminate output data bursts.

The advanced IFP and flexible programmability of the MT9D131 provide a variety of ways to enhance and optimize the image sensor performance. Built-in optimization algorithms enable the MT9D131 to operate at factory settings as a fully automatic, highly adaptable camera. However, most of its settings are user-programmable.

Typical Connection

Figure 1: Typical Configuration (Connection)



Note:
1. Resistor value 1.5KΩ is recommended, but may be greater for slower two-wire speed.
2. TEST must be connected to digital ground for normal device operation.
3. See “Standby Hardware Configuration” on page 109.
4. All power supply pads must be used.

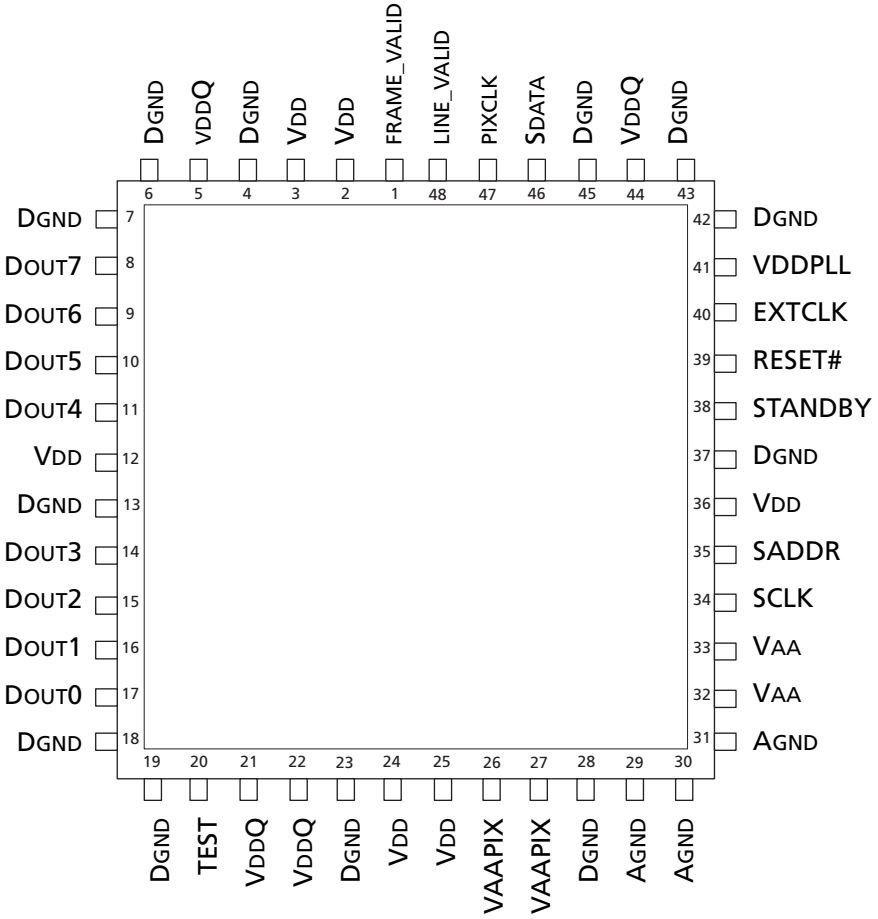
Signal Description

Table 3: Signal Description

Name	Type	Description	Note
EXTCLK	Input	Master clock signal (can either drive the on-chip PLL or bypass it).	
RESET_BAR	Input	Master reset signal, active LOW.	
STANDBY	Input	Controls sensor's standby mode.	
TEST	Input	Reserved for factory test. Tie to digital ground during normal operation.	
SCLK	Input	Two-wire serial interface clock.	
SADDR	Input	Selects device address for the two-wire serial interface. The address is 0x90 when SADDR is tied LOW, 0xBA if tied HIGH. See also R0x0D:0[10].	
DOUT0-DOUT7	Output	Eight-bit image data output or most significant bits (MSB) of 10-bit sensor bypass mode.	1
FRAME_VALID	Output	Identifies rows in the active image.	1
LINE_VALID	Output	Identifies lines in the active image.	1
PIXCLK	Output	Pixel clock. To be used for sampling DOUT, FRAME_VALID, and LINE_VALID.	1
SDATA	I/O	Two-wire serial interface data.	
VDD	Supply	Digital power (1.8V).	
VDDPLL	Supply	PLL power (2.8V).	
VAA	Supply	Analog power (2.8V).	
VAAPIX	Supply	Pixel array power (2.8V).	
VDDQ	Supply	I/O power (nominal 1.8V or 2.8V).	
AGND	Supply	Analog ground.	
DGND	Supply	Digital, I/O, and PLL ground.	

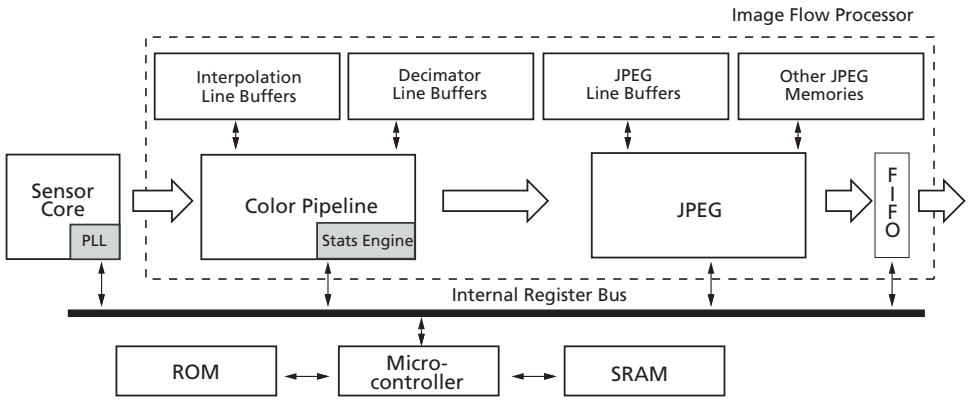
Note: 1. See "Standby Hardware Configuration" on page 109.

Figure 2: 48-Pin CLCC Pinout Diagram



Architecture Overview

Figure 3: Block Diagram

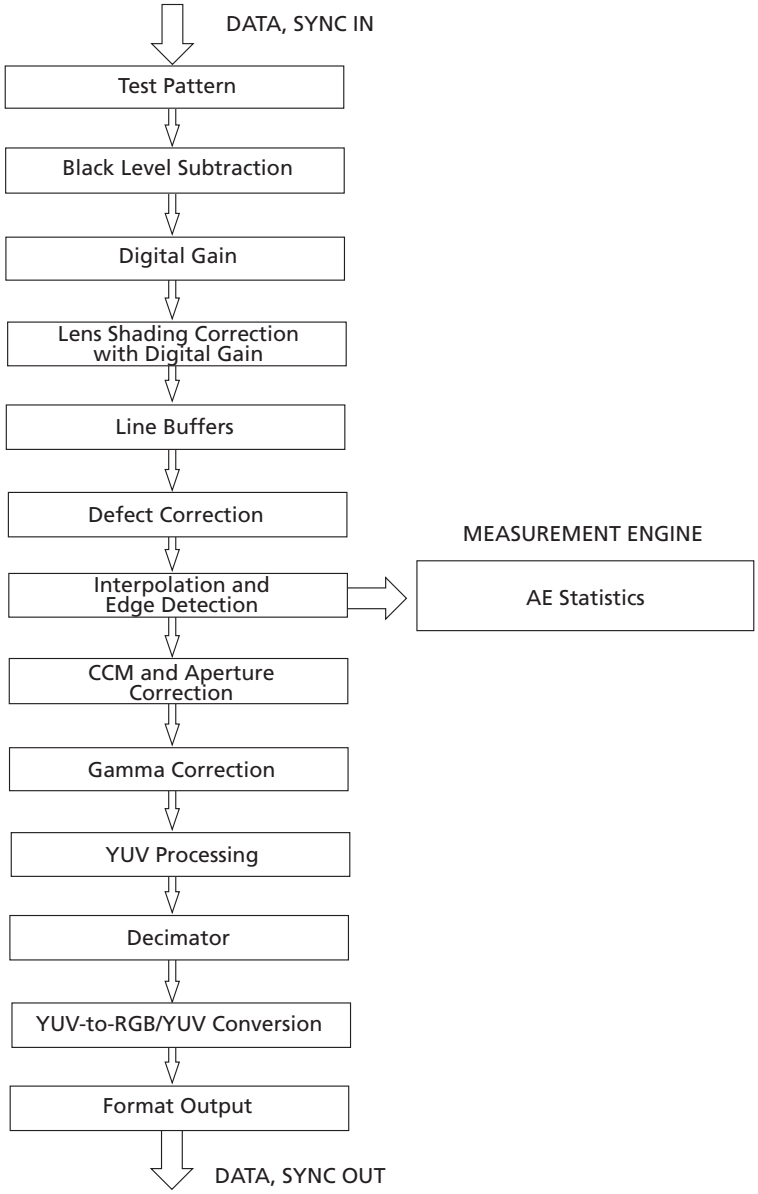


Sensor Core

The MT9D131 sensor core is based on ON Semiconductor's MT9D011, a stand-alone, 2-megapixel CMOS image sensor with a 2.8µm pixel size. Both image sensors have the same optical size (1/3.2 inches) and maximum resolution (UXGA). Like the MT9D011, the MT9D131 sensor core includes a phase-locked loop oscillator (PLL), to facilitate camera integration and minimize the system cost for surveillance applications. When in use, the PLL generates internal master clock signal whose frequency can be set higher than the frequency of external clock signal EXTCLK. This allows the MT9D131 to run at any desired resolution and frame rate up to the specified maximum values, irrespective of the EXTCLK frequency.

Color Pipeline

Figure 4: Color Pipeline



Test Pattern

During normal operation of MT9D131, 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.

Black Level Conditioning and Digital Gain

Image stream processing starts with black level conditioning and multiplication of all pixel values by a programmable digital gain.

Lens Shading Correction

Inexpensive lenses tend to produce images whose brightness is significantly attenuated near the edges. Chromatic aberration in such lenses can cause color variation across the field of view. There are also other factors causing fixed-pattern signal gradients in images captured by image sensors. The cumulative result of all these factors is known as lens shading. The MT9D131 has an embedded lens shading correction (LC) module that can be programmed to precisely counter the shading effect of a lens on each RGB color signal. The LC module multiplies RGB signals by a two-dimensional correction function F(x,y), whose profile in both x and y direction is a piecewise quadratic polynomial with coefficients independently programmable for each direction and color.

Line Buffers

Several data processing steps following the lens shading correction require access to pixel values from up to 8 consecutive image lines. For these lines to be simultaneously available for processing, they must be buffered. The IFP includes a number of SRAM line buffers that are used to perform defect correction, color interpolation, image decimation, and JPEG encoding.

Defect Correction

The IFP performs on-the-fly defect correction that can mask pixel array defects such as high-dark-current (“hot”) pixels and pixels that are darker or brighter than their neighbors due to photoresponse nonuniformity. The defect correction algorithm uses several pixel features to distinguish between normal and defective pixels. After identifying the latter, it replaces their actual values with values inferred from the values of nearest same-color neighbors.

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, to make things simple, 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 maintaining the sharpness of the image and filtering out high-frequency noise. The simplest interpolation algorithm is to sort the nearest eight neighbors of every pixel into three sets—red, green, and blue: discard the set of pixels of the same color as the center pixel (if there are any): calculate average pixel values for the remaining two sets, and use the averages instead of the missing color data for the center pixel. Such averaging reduces high-frequency noise, but it also blurs and distorts sharp transitions (edges) in the image. To avoid this problem, the interpolation module performs edge detection in the neighborhood of every processed pixel and, depending

on its results, extracts color information from neighboring pixels in a number of different ways. In effect, it does low-pass filtering in flat-field image areas and avoids doing it near edges.

Color Correction and Aperture Correction

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

To increase image sharpness, a programmable aperture correction is applied to color corrected image data, equally to each of the 12-bit R, G, and B color channels.

Gamma Correction

Like the aperture correction, gamma correction is applied equally to each of the 12-bit R, G, and B color channels. Gamma correction curve is implemented as a piecewise linear function with 19 knee points, taking 12-bit arguments and mapping them to 8-bit output. The abscissas of the knee points are fixed at 0, 64, 128, 256, 512, 768, 1024, 1280, 1536, 1792, 2048, 2304, 2560, 2816, 3072, 3328, 3584, 3840, and 4095. The 8-bit ordinates are programmable through IFP registers or public variables of mode driver (ID = 7). The driver variables include two arrays of knee point ordinates defining two separate gamma curves for sensor operation contexts A and B.

YUV Processing

After the gamma correction, the image data stream undergoes RGB to YUV conversion and, optionally, further corrective processing. The first step in this processing is removal of highlight coloration, also referred to as “color kill.” It affects only pixels whose brightness exceeds a certain preprogrammed threshold. The U and V values of those pixels are attenuated proportionally to the difference between their brightness and the threshold. The second optional processing step is noise suppression by one-dimensional low-pass filtering of Y and/or UV signals. A 3- or 5-tap filter can be selected for each signal.

Image Cropping and Decimation

To ensure that the size of images output by MT9D131 can be tailored to the needs of all users, the IFP includes a decimator module. When enabled, this module performs “decimation” of incoming images (shrinks them to arbitrarily selected width and height without reducing the field of view and without discarding any pixel values). The latter point merits underscoring, because the terms “decimator” and “image decimation” suggest image size reduction by deleting columns and/or rows at regular intervals. Despite the terminology, no such deletions take place in the decimator module. Instead, it 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 decimation in the IFP can be preceded by image cropping and/or image decimation in the sensor core. Image cropping takes place when the sensor core is programmed to output pixel values from a rectangular portion of its pixel array - a window - smaller than the default 1600 x 1200 window. Pixels outside the selected cropping window are not read out, which results in narrower field of view than at the default sensor settings. Irrespective of the size and position of the cropping window, the MT9D131 sensor core can also decimate outgoing images by skipping columns and/or rows of the pixel array, and/or by binning 2 x 2 groups of pixels of the same color. Because decimation by skipping (that is, deletion) can cause aliasing (even if pixel binning is simultaneously enabled), it is generally better to change image size only by cropping and pixel binning.

The image cropping and decimator module can be used to do digital zoom and pan. If the decimator 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 1600 x 1200 image rendered on a 160 x 120 display can be zoomed up to 10 times, since 1600/160 = 1200/120 = 10. Panning effect can be achieved by fixing the size of the cropping window and moving it around the pixel array.

YUV-to-RGB/YUV Conversion and Output Formatting

The YUV data stream emerging from the decimator module can either exit the color pipeline as-is or be converted before exit to an alternative YUV or RGB data format. See “Color Conversion Formulas” on page 78 and the description of register R0x97:1 for more details.

JPEG Encoder and FIFO

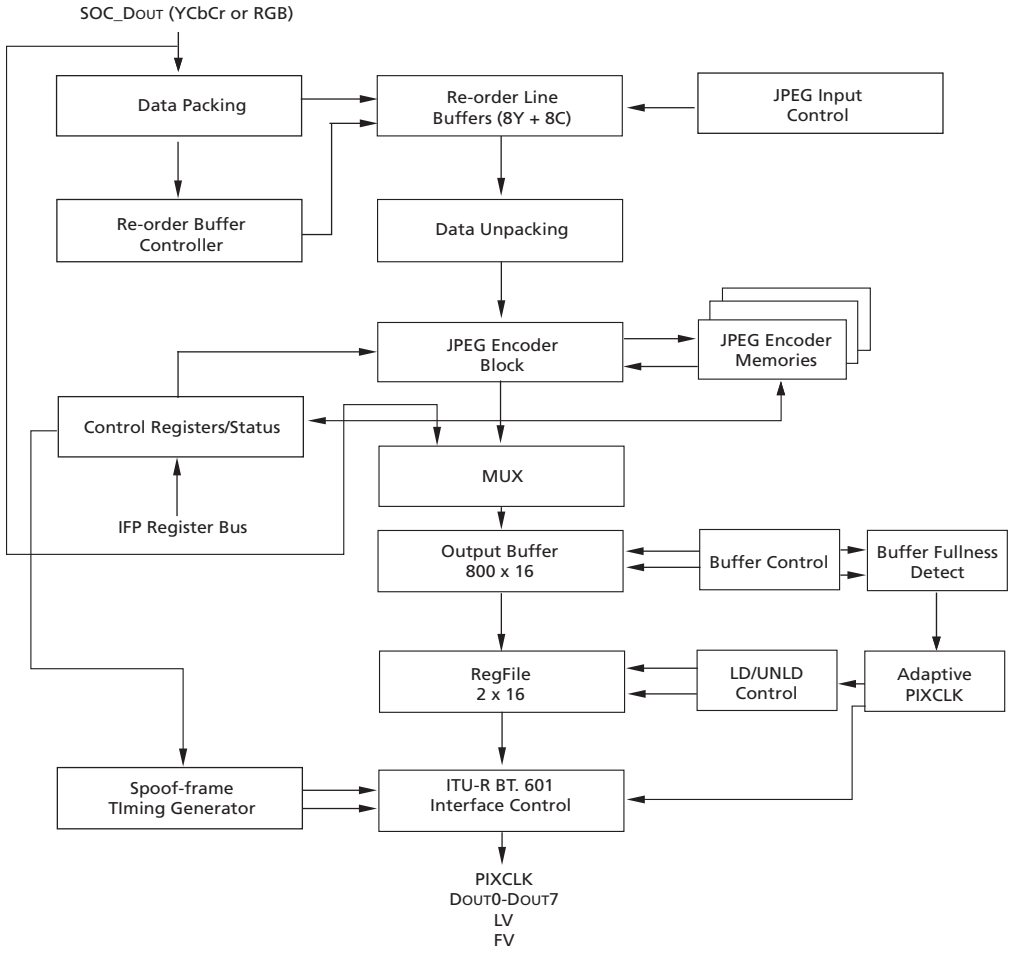
The JPEG compression engine in the MT9D131 is a highly integrated, high-performance solution that can provide sustained data rates of almost 80 MB/s for image sizes up to 1600 x 1200. Additionally, the solution provides for low power consumption and full programmability of JPEG compression parameters for image quality control.

The JPEG encoding block is designed for continuous image flow and is ideal for low-power applications. After initial configuration for a target application, it can be controlled easily for instantaneous stop/restart. A flexible configuration and control interface allows for full programmability of various JPEG-specific parameters and tables.

JPEG Encoding Highlights

1. Sequential DCT (baseline) ISO/IEC 10918-1 JPEG-compliant
2. YCbCr 4:2:2 format compression
3. Programmable quantization tables
   - One each for luminance and chrominance (active)
   - Support for three pairs of quantization tables—two pairs serve as a backup for buffer overflow
4. Programmable Huffman Tables
   - 2 AC, 2 DC tables—separate for luminance and chrominance
5. Quality/compression ratio control capability
6. 15 fps MJPEG capability (header processing in external host processor)

Figure 5: JPEG Encoder Block Diagram



Output Buffer Overflow Prevention

The MT9D131 integrates SRAM for the storage of JPEG data. To prevent output buffer overflow, the MT9D131 implements an adaptive pixel clock (PIXCLK) rate scheme. When the adaptive pixel clock rate scheme is enabled, PIXCLK can run at clock frequencies of *(EXTCLK freq/N1)*, *(EXTCLK freq/N2)*, *(EXTCLK freq/N3)*, where *N1*, *N2*, *N3* are register values programmed by the host through the two-wire serial interface. A clock divider block from the master clock EXTCLK generates the three clocks, PCLK1, PCLK2, and PCLK3.

At the start of the frame encode, PIXCLK is sourced by PCLK1. The buffer fullness detection block of the SOC switches PIXCLK to PCLK2 and then to PCLK3, if necessary, based on the watermark at the output buffer (that is, percentage filled up). When the output buffer watermark reaches 50 percent, PIXCLK switches to PCLK2. This increase in PIXCLK rate unloads the output buffer at a higher rate. However, depending on the image complexity and quantization table setting, the compressed image data may still be generated by the JPEG encoder faster than PIXCLK can unload it. Should the output buffer watermark equal 75 percent or higher, PIXCLK is switched to PCLK3. When the output buffer watermark drops back to 50 percent, PIXCLK is switched back to PCLK2. When the output buffer watermark drops to 25 percent, PIXCLK is switched to PCLK1.

When a decision to adapt PIXCLK frequency is made, LINE_VALID, which qualifies the 8-bit data output (DOUT), is de-asserted until PIXCLK is safely switched to the new clock. LINE_VALID is independent of the horizontal timing of the uncompressed imaged. Its assertion is strictly based on compressed image data availability.

Should an output buffer overflow still occur with PIXCLK at the maximum frequency, the output buffer and the small asynchronous FIFO are flushed immediately. This causes LINE_VALID to be de-asserted. FRAME_VALID is also de-asserted.

In addition to the adaptive PIXCLK rate scheme, the MT9D131 also has storage for three sets of quantization tables (six tables). In the event of output buffer overflow during the compression of the current frame, another set of the preloaded quantization tables can be used for the encoding of the immediate next frame. Then, the MT9D131 starts compressing the next frame starting with the nominal PIXCLK frequency.

Output Interface

Control (Two-Wire Serial Interface)

Camera control and JPEG configuration/control are accomplished through a two-wire serial interface. The interface supports individual access to all camera function registers and JPEG control registers. In particular, all tables located in the JPEG quantization and Huffman memories are accessible through the two-wire interface. To write to a particular register, the external host processor must send the MT9D131 device address (selected by SADDR or R0x0D:0[10]), the address of the register, and data to be written to it. See "Appendix A: Two-Wire Serial Register Interface" on page 125 for a description of read sequence and for details of the two-wire serial interface protocol.

Data

JPEG data is output in a BT656-like 8-bit parallel bus DOUT0–DOUT7, with FRAME_VALID, LINE_VALID, and PIXCLK. JPEG output data is valid when both FRAME_VALID and LINE_VALID are asserted. When the JPEG data output for the frame completes, or buffer overflow occurs, LINE_VALID and FRAME_VALID are de-asserted. The output clock runs at frequencies selected by frequency divisors N1, N2, and N3 (registers R0x0E:2 and R0x0F:2), depending on output buffer fullness.

Context and Operational Modes

The MT9D131 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.

Preview

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. 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; for JPEG compression, how many frames to capture, and so on. For video, the user might select a different image size and a fixed frame rate.

Snapshot

To take a snapshot, the user must send a command that changes the context from A to context B. Typical sequence of events after this command is as follows. First, the camera exposure and white balance adjusts automatically to the changed illumination of the scene. Next, the camera enables JPEG compression and captures one or more frames of desired size. Once the sequence is complete, the camera automatically returns to context A and resumes running preview.

Video

To start video capture, the user has to change relevant context B settings, such as capture mode, image size, and frame rate, and again send a context change command. Upon receiving the command, the MT9D131 switches to the modified context B settings, while continuing to output YUV-encoded image data. Auto exposure automatically switches to smooth continuous operation. To exit the video capture mode, the user has to send another context change command causing the sensor to switch back to context A.

Auto Exposure

The auto exposure (AE) 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.

Two auto exposure algorithm modes are available:

1. preview
2. scene-evaluative

Auto exposure is implemented by means of a firmware driver that analyzes image statistics collected by 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.

Preview Mode

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

The 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, measurement window, and other parameters described above.

Scene-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 the most difference for backlight and bright outdoor conditions.

Auto White Balance

The MT9D131 has a built-in auto white balance (AWB) algorithm designed to compensate for the effects of changing spectra of the scene illumination on the quality of the color rendition. This sophisticated 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. Unlike simple white balancing algorithms found in many PC cameras, the MT9D131 AWB does not require the presence of gray or white elements in the image for good color rendition. The AWB does not attempt to locate “brightest” or “grayest” element of the image but instead performs sophisticated image analysis to differentiate between changes in predominant spectra of illumination and changes in predominant colors of the scene. While defaults are suitable for most applications, a wide range of algorithm parameters can be overwritten by the user through the serial interface.

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.

Registers and Variables

Three types of configuration controls are available:

- Hardware registers
- Driver variables
- MCU SRAM

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

R0x12:1, R0x12:1[3:0] or R18:1, R18:1[3:0]

These refer to two-wire accessible register number 18, or 0x12 hexadecimal, located on page 1. [3:0] indicate bits. Registers numbers range from 0 through FF and bits range from 15 through 0.

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

How to Access

Registers, variables are accessed in different ways.

Registers

Hardware registers are organized into several pages. Page 0 contains sensor controls. Page 1 contains color pipeline controls. Page 2 contains JPEG, output FIFO, and more color pipeline controls. The desired page is selected by writing the desired value to R0xF0. After that, all READs and WRITEs to registers from 0 through FF except R0xF0 and R0xF1, are directed to the selected page. R0xF0 and R0xF1 are special registers and are present on all pages. See "Appendix A: Two-Wire Serial Register Interface" on page 125 for a description of two-wire register access.

Variables

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

All driver variables (public and private) can be accessed through R0xC6:1 and R0xC8:1. While two access modes are available (accessed by physical address and by logical address) the public variables are typically accessed by the logical method. The logical address, which is set in R0C6:1, consists of a 5-bit driver ID number and a variable offset. Examples are provided below.

To set the variable ae.Target = 50:

- The variable has a driver ID of 2. Therefore, set R0xC6:1[12:8] = 2
- The variable has an offset of 6. Therefore, set R0xC6:1[7:0] = 6
- This is a logical access. Therefore, set R0xC6:1[14:13] = 01
- The size of the variable is 8 bits. Therefore, set R0xC6:1[15] = 1
- By combining these bits, R0xC6:1 = 0xA206.
- Set R0xC8:1 = 50 for the value of the variable

To read the variable ae.Target:

- Since this is the same variable as the above example, R0xC6:1 = 0xA206
- Read R0xC8:1 for the current variable value

MCU SRAM consists of 1K system memory and 1K user memory. Examples of access:

- Write into user SRAM. Use to upload code
  a. R0xC6:1 = 0x400 // address
  b. R0xC8:1 = 0x1234 // write 16-bit value
- Read from user SRAM
  a. R0xC6:1 = 0x400 // address
  b. Read R0xC8:1 // read 16-bit value

See R0xC6:1 and R0xC8:1 description in Table 7, “IFP Registers, Page 2,” on page 43 for more detail.

Registers

Sensor Core Registers

Table 4: Sensor Core Register Defaults

Register Number (HEX)	Register Description	Default PRE MCU BOOT	Default AFTER MCU BOOT
0x00	Reserved	0x1519	0x1519
0x01	Row Start	0x001C	0x001C
0x02	Column Start	0x003C	0x003C
0x03	Row Width	0x04B0	0x04B0
0x04	Col Width	0x0640	0x0640
0x05	Horizontal Blanking B	0x015C	0x0204
0x06	Vertical Blanking B	0x0020	0x002F
0x07	Horizontal Blanking A	0x00AE	0x00FE
0x08	Vertical Blanking A	0x0010	0x000C
0x09	Shutter Width	0x04D0	N/A
0x0A	Row Speed	0x00011	0x0001
0x0B	Extra Delay	0x0000	0x0000
0x0C	Shutter Delay	0x0000	0x0000
0x0D	Reset	0x0000	0x0000
0x1F	Frame Valid Control	0x0000	0x0000
0x20	Read Mode B	0x0000	0x0300
0x21	Read Mode A	0x0490	0x8400
0x22	Dark Col/Rows	0x010F	0x010F
0x23	Reserved	0x0608	0x0608
0x24	Extra Reset	0x8000	0x8000
0x25	Line Valid Control	0x0000	0x0000
0x26	Bottom Dark Rows	0x0007	0x0007
0x2B	Green Gain	0x0020	N/A
0x2C	Blue Gain	0x0020	N/A
0x2D	Red Gain	0x0020	N/A
0x2E	Green2 Gain	0x0020	N/A
0x2F	Global Gain	0x0020	N/A
0x30	Row Noise	0x042A	0x042A
0x59	Black Rows	0x00FF	0x00FF
0x5B	Dark G1 average	N/A	N/A
0x5C	Dark B average	N/A	N/A
0x5D	Dark R average	N/A	N/A
0x5E	Dark G2 average	N/A	N/A
0x5F	Calib Threshold	0x231D	0x231D
0x60	Calib Control	0x0080	0x0080
0x61	Calib Green1	0x0000	0x0000
0x62	Calib Blue	0x0000	0x0000
0x63	Calib Red	0x0000	0x0000

Table 4: Sensor Core Register Defaults (continued)

Register Number (HEX)	Register Description	Default PRE MCU BOOT	Default AFTER MCU BOOT
0x64	Calib Green2	0x0000	0x0000
0x65	Clock Control	0xE000	0xE000
0x66	PLL Control 1	0x2809	0x1000
0x67	PLL Control 2	0x0501	0x0500
0xC0	Reserved	0x1519	0x1519
0xC1	Reserved	0x1519	0x1519
0xC2	Reserved	0x1519	0x1519
0xC3	Reserved	0x1519	0x1519
0xC4	Reserved	0x1519	0x1519
0xC5	Reserved	0x1519	0x1519
0xC6	Reserved	0x1519	0x1519
0xE0	Reserved	0x1519	0x1519
0xE1	Reserved	0x1519	0x1519
0xE2	Reserved	0x1519	0x1519
0xE3	Reserved	0x1519	0x1519
0xF0	Page Register	0x0000	0x0000
0xF1	Bytewise Address	0x0000	0x0000
0xF2	Context Control	0x000B	0x0000
0xFF	Reserved	0x1519	0x1519

Registers

Notation used in the sensor register description table:
Sync'd to frame start
N = No. The register value is updated and used immediately.
Y = Yes. The register value is updated at next frame start as long as the synchronize changes bit is 0. Frame start is defined as when the first dark row is read out. By default, this is 8 rows before FRAME_VALID goes HIGH.
Bad frame
A bad frame is a frame where all rows do not have the same integration time, or offsets to the pixel values changed during the frame.
N = No. Changing the register value does not produce a bad frame.
Y = Yes. Changing the register value might produce a bad frame.
YM = Yes, but the bad frame is masked out unless the "show bad frames" feature is (R0x0D:0[8]) is enabled.

Table 5: Sensor Core Register Description

Bit Field	Description		Default (Hex)	Sync'd to Frame Start	Bad Frame
R0—0x00 - Reserved (R/O)					
Bits 15:0	Reserved	Reserved.	1519		
R1—0x01 - Row Start (R/W)					
Bits 10:0	Row Start	The first row to be read out, excluding any dark rows that may be read. To window the image down, set this register to the starting Y value. Setting a value less than 20 is not recommended because the dark rows should be read using R0x22:0.	1C	Y	YM
R2—0x02 - Column Start (R/W)					
Bits 10:0	Column Start	The first column to be read out, excluding dark columns that may be read. To window the image down, set this register to the starting X value. Setting a value below 52 is not recommended because readout of dark columns should be controlled by R0x22:0.	3C	Y	YM
R3—0x03 - Row Width (R/W)					
Bits 10:0	Row Width	Number of rows in the image to be read out, excluding any dark rows or border rows that may be read. The minimum supported value is 2.	4B0	Y	YM
R4—0x04 - Column Width (R/W)					
Bits 10:0	Column Width	Number of columns in image to be read out, excluding any dark columns or border columns that may be read. The minimum supported value is 9 in 1 ADC mode and 17 in 2 ADC mode.	640	Y	YM
R5—0x05 - Horizontal Blanking—Context B (R/W)					
Bits 13:0	Horizontal Blanking—Context B	Number of blank columns in a row when context B is selected (R0xF2:0[0] = 1). The extra columns are added at the beginning of a row. See "Frame Rate Control" on page 93 for more information on supported register values.	15C	Y	YM

Table 5: Sensor Core Register Description (continued)

Bit Field		Description	Default (Hex)	Sync'd to Frame Start	Bad Frame
R6—0x06 - Vertical Blanking—Context B (R/W)					
Bits 14:0	Vertical Blanking—Context B	Number of blank rows in a frame when context B is selected (R0xF2:0[1] = 1). The minimum supported value is (4 + R0x22:0[2:0]). The actual vertical blanking time may be controlled by the shutter width (R0x09:0). See "Raw Data Timing" on page 83.	20	Y	N
R7—0x07 - Horizontal Blanking—Context A (R/W)					
Bits 13:0	Horizontal Blanking—Context A	Number of blank columns in a row when context A is selected (R0xF2:0[0] = 0). The extra columns are added at the beginning of a row. See "Frame Rate Control" on page 93 for more information on supported register values.	AE	Y	YM
R8—0x08 - Vertical Blanking—Context A (R/W)					
Bits 14:0	Vertical Blanking—Context A	Number of blank rows in a frame when context A is chosen (R0xF2:0[1] = 1). The minimum supported value is (4 + R0x22:0[2:0]). The actual vertical blanking time may be controlled by the shutter width (R0x9:0). See "Raw Data Timing" on page 83.	10	Y	N
R9—0x09 - Shutter Width (R/W)					
Bits 15:0	Shutter Width	Integration time in number of rows. The integration time is also influenced by the shutter delay (R0x0C:0) and the overhead time.	4D0	Y	N
R10—0x0A - Row Speed (R/W)					
Bits 15:14	Reserved	Do not change from default value.			
Bit 13	Reserved	Do not change from default value.			
Bit 8	Invert Pixel Clock	Invert PIXCLK. When clear, FRAME_VALID, LINE_VALID, and DOUT are set up relative to the delayed rising edge of PIXCLK. When set, FRAME_VALID, LINE_VALID, and DOUT are set up relative to the delayed falling edge of PIXCLK.	0	N	N
Bits 7:4	Delay Pixel Clock	Number of half master clock cycle increments to delay the rising edge of PIXCLK relative to transitions on FRAME_VALID, LINE_VALID, and DOUT.	1	N	N
Bit 3	Reserved	Do not change from default value.			
Bits 2:0	Pixel Clock Speed	A programmed value of N gives a pixel clock period of N master clocks in 2 ADC mode and 2*N master clocks in 1 ADC mode. A value of "0" is treated like (and reads back as) a value of "1."	1	Y	YM
R11—0x0B - Extra Delay (R/W)					
Bits 13:0	Extra Delay	Extra blanking inserted between frames. A programmed value of N increases the vertical blanking time by N pixel clock periods. Can be used to get a more exact frame rate. It may affect the integration times of parts of the image when the integration time is less than one frame. Bad frame does not occur unless integration time is less than one frame.	0	Y	N[1]

Table 5: Sensor Core Register Description (continued)

Bit Field	Description		Default (Hex)	Sync'd to Frame Start	Bad Frame
R12—0x0C - Shutter Delay (R/W)					
Bits 13:0	Shutter Delay	The amount of time from the end of the sampling sequence to the beginning of the pixel reset sequence. If the value in this register exceeds the row time, the reset of the row does not complete before the associated row is sampled, and the sensor does not generate an image. A programmed value of N reduces the integration time by N/2 pixel clock periods in 1 ADC mode and by N pixel clock periods in 2 ADC mode.	0	Y	N
R13—0x0D - Reset (R/W)					
Bit 15	Synchronize Changes	By default, update of many registers are synchronized to frame start. Setting this bit inhibits this update; register changes remain pending until this bit is returned to "0." When this bit is returned to "0," all pending register updates are made on the next frame start.	0	N	N
Bit 10	Toggle SADDR	By default, the sensor serial bus responds to addresses 0xBA and 0xBB. When this bit is set, the sensor serial bus responds to addresses 0x90 and 0x91. WRITEs to this bit are ignored when STANDBY is asserted. See "Slave Address" on page 126.	0	N	N
Bit 9	Restart Bad Frames	When set, a restart is forced to take place whenever a bad frame is detected. This can shorten the delay when waiting for a good frame because the delay, when masking out a bad frame, is the integration time rather than the full frame time.	0	N	N
Bit 8	Show Bad Frames	0: Outputs only good frames (default). 1: Output all frames (including bad frames). A bad frame is defined as the first frame following a change to: window size or position, horizontal blanking, pixel clock speed, zoom, row or column skip, binning, mirroring, or use of border.	0	N	N
Bit 7:6	Inhibit Standby / Drive Pins	00 or 01: setting STANDBY HIGH puts sensor into standby state with high-impedance outputs 10: Setting STANDBY HIGH only puts the outputs in High-Z 11: Causes STANDBY to be ignored	0	N	N
Bit 5	Reset SOC	When this bit is set to "1", SOC is put in reset state. It exits this state when the bit is set back to "0. " Any attempt to access SOC registers (IFP pages 1 and 2) in the reset state results in a sensor hang-up. The sensor cannot recover from it without a hard reset or power cycle.	0	N	
Bit 4	Output Disable	Setting this bit to "1" puts the pin interface in a High-Z. See "Output Enable Control" on page 109. If the DOUT*, PIXCLK, Frame_Valid, or Line_Valid are floating during STANDBY, this bit should be set to "0" to turn off the input buffer, reducing standby current (see technical note TN0934 "Standby Sequence"). This bit must work together with bit 6 to take effect.	0		
Bit 3	Reserved	Keep at default value.	0		
Bit 2	Standby	Setting this bit to "1" places the sensor in a low-power state. Any attempt to access registers R[0xF7:0xFD]:0 in this state results in a sensor hang-up. The sensor cannot recover from it without a hard reset or power cycle.	0	N	YM

Table 5: Sensor Core Register Description (continued)

Bit Field	Description		Default (Hex)	Sync'd to Frame Start	Bad Frame
Bit 1	Restart	Setting this bit to "1" causes the sensor to truncate the current frame and start resetting the first row. The delay before the first valid frame is read out is equal to the integration time. This bit is write - "1" but always reads back as "0".	0	N	YM
Bit 0	Reset	Setting this bit puts the sensor in reset; the frame being generated is truncated and the pin interface goes to an idle state. All internal registers (except for this bit) go to the default power-up state. Clearing this bit resumes normal operation.	0	N	YM
R31—0x1F - FRAME_VALID Control (R/W)					
Bit 15	Enable Early FRAME_VALID Fall	0: Default. FRAME_VALID goes low 6 pixel clocks after last LINE_VALID. 1: Enables the early disabling of FRAME_VALID as set in bits 14:8. LINE_VALID is still generated for all active rows.	0	N	N
Bits 14:8	Early FRAME_VALID Fall	When enabled, the FRAME_VALID falling edge occurs within the programmed number of rows before the end of the last LINE_VALID. (1 + bits 14:8)*row time + constant (constant = 3 in default mode) The value of this field must not be larger than row width R0x03:0.	0	N	N
Bit 7	Enable Early FRAME_VALID Rise	0: Default. FRAME_VALID goes HIGH 6 pixel clocks before first LINE_VALID. 1: Enables the early rise of FRAME_VALID as set in bits 6:0.	0	N	N
Bits 6:0	Early FRAME_VALID Rise	When enabled, the FRAME_VALID rising edge is set HIGH the programmed number of rows before the first LINE_VALID: (1 + bits 6:0)*row time + horizontal blank + constant (constant = 3 in default mode).	0	N	N
R32—0x20 - Read Mode—Context B (R/W)					
Bit 15	Binning—Context B	When read mode context B is selected (R0xF2:0[3] = 1): 0: Normal operation. 1: Binning enabled. See "Binning" on page 92 and See "Frame Rate Control" on page 93 for a full description.	0	Y	YM
Bit 13	Zoom Enable	0: Normal operation. 1: Zoom is enabled, with zoom factor [zoom] defined in bits 12:11. In zoom mode, the pixel data rate is slowed by a factor of [zoom]. This is achieved by outputting [zoom - 1] blank rows between each output row. Setting this mode allows the user to fill a window that is [zoom] times larger with interpolated data. The pixel clock speed is not affected by this operation, and the output data for each pixel is valid for [zoom] pixel clocks. Every row is followed by [zoom - 1] blank rows (with their own LINE_VALID, but all data bits = 0) of equal time. The combination of this register and an appropriate change to the window sizing registers allows the user to zoom to a region of interest without affecting the frame rate.	0	Y	YM

Table 5: Sensor Core Register Description (continued)

Bit Field	Description		Default (Hex)	Sync'd to Frame Start	Bad Frame
Bits 12:11	Zoom	When zoom is enabled by bit 13, this field determines the zoom amount: 00: Zoom 2x 01: Zoom 4x 10: Zoom 8x 11: Zoom 16x	0	Y	YM
Bit 10	Use 1 ADC—Context B	When read mode context B is selected (bit 3, R0xF2:0 = 1): 0: Use both ADCs to achieve maximum speed. 1: Use 1 ADC to reduce power. Maximum readout frequency is now half the master clock frequency, and the pixel clock is automatically adjusted as described for the pixel clock speed register.	0	Y	YM
Bit 9	Show Border	This bit indicates whether to show the border enabled by bit 8. 0: Border is enabled but not shown; vertical blanking is increased by 8 rows and horizontal blanking is increased by 8 pixels. 1: Border is enabled and shown; FRAME_VALID time is extended by 8 rows and LINE_VALID is extended by 8 pixels. See "Pixel Border" on page 89.	0	N	N
Bit 8	Over Sized	0: Normal UXGA size. 1: Adds a 4-pixel border around the active image array independent of readout mode (skip, zoom, mirror, and so on). Setting this bit adds 8 to the number of rows and columns in the frame.	0	Y	YM
Bit 7	Column Skip Enable—Context B	When read mode context B is selected (R0xF2:0[3] = 1): 0: Normal readout. 1: Enable column skip.	0	Y	YM
Bits 6:5	Column Skip—Context B	When read mode context B is selected (R0xF2:0[3] = 1) and column skip is enabled (bit 7 = 1): 00: Column Skip 2x 01: Column Skip 4x 10: Column Skip 8x 11: Column Skip 16x See "Column and Row Skip" on page 90 for more information.	0	Y	YM
Bit 4	Row Skip Enable—Context B	When read mode context B is selected (R0xF2:0[3] = 1): 0: Normal readout. 1: Enable row skip.	0	Y	YM
Bits 3:2	Row Skip—Context B	When read mode context B is selected (R0xF2:0[3] = 1) and Row skip is enabled (bit 4 = 1): 00: Row Skip 2x 01: Row Skip 4x 10: Row Skip 8x 11: Row Skip 16x See "Column and Row Skip" on page 90 for more information.	0	Y	YM
Bit 1	Mirror Columns	Read out columns from right to left (mirrored). When set to "1", column readout starts from column [column start + column size] and continues down to [column start + 1]. When set to "0", readout starts at column start and continues to [column start + column size – 1]. This ensures that the starting color is maintained.	0	Y	YM

Table 5: Sensor Core Register Description (continued)

Bit Field	Description		Default (Hex)	Sync'd to Frame Start	Bad Frame
Bit 0	Mirror Rows	Read out rows from bottom to top (upside down). When set, row readout starts from row [row start + row size] and continues down to [row start + 1]. When clear, readout starts at row start and continues to [row start + row size –1]. This ensures that the starting color is maintained.	0	Y	YM
R33—0x21 - Read Mode—Context A (R/W)					
Bit 15	Binning—Context A	When read mode context A is selected (R0xF2:0[3] = 0): 0: Normal operation. 1: Binning enabled. See "Binning" on page 92.	1	Y	YM
Bit 10	Use 1 ADC—Context A	When read mode context A is selected (R0xF2:0[3] = 0): 0: Use both ADCs to achieve maximum speed. 1: Use one ADC to reduce power. Maximum readout frequency is now half of the master clock, and the pixel clock is automatically adjusted as described for the pixel clock speed register.	1	Y	YM
Bit 7	Column Skip Enable—Context A	When read mode context A is selected (R0xF2:0[3] = 0): 0: Normal readout. 1: Enable column skip.	1	Y	YM
Bits 6:5	Column Skip—Context A	When read mode context A is selected (R0xF2:0[3] = 0) and column skip is enabled (bit 7 = 1): 00: Column Skip 2x 01: Column Skip 4x 10: Column Skip 8x 11: Column Skip 16x See "Column and Row Skip" on page 90 for more information.	0	Y	YM
Bit 4	Row Skip Enable—Context A	When read mode context A is selected (R0xF2:0[3] = 0): 0: Normal readout. 1: Enable row skip.	1	Y	YM
Bits 3:2	Row Skip—Context A	When read mode context A is selected (R0xF2:0[3] = 0) and Row skip is enabled (bit 4 = 1): 00: Row Skip 2x 01: Row Skip 4x 10: Row Skip 8x 11: Row Skip 16x See "Column and Row Skip" on page 90 for more information.	0	Y	YM
R34—0x22 - Show Control (R/W)					
Bit 10	Number of Dark Columns	The MT9D131 has 40 dark columns. 0: Read out 20 dark columns (4–23). 1: Read out 36 dark columns (4–39). Ignored during binning, where all 40 dark columns are used.	0	N	N
Bit 9	Show Dark Columns	When set to "1", the 20 or 36 (dependent on bit 10) dark columns are output before the active pixels in a line. There is an idle period of 2 pixels between readout of the dark columns and readout of the active image. Therefore, when set to "1", LINE_VALID is asserted 22 pixel times earlier than normal, and the horizontal blanking time is decreased by the same amount.	0	N	N

Table 5: Sensor Core Register Description (continued)

Bit Field	Description		Default (Hex)	Sync'd to Frame Start	Bad Frame
Bit 8	Read Dark Columns	0: When disabled, an arbitrary number of dark columns can be read out by including them in the active image. Enabling the dark columns increases the minimum value for horizontal blanking but does not affect the row time. 1: Enables the readout of dark columns for use in the row-wise noise correction algorithm. The number of columns used are 40 in binning mode, and otherwise determined by bit 10.	1	N	Y
Bit 7	Show Dark Rows	When set to "1", the programmed dark rows is output before the active window. FRAME_VALID is thus asserted earlier than normal. This has no effect on integration time or frame rate.	0	N	N
Bits 6:4	Dark Start Address	The start address for the dark rows within the 8 available rows (an offset of 4 is added to compensate for the guard pixels). Must be set so all dark rows read out falls in the address space 0:7.	0	N	N
Bit 3	Reserved	Do not change from default value.			
Bits 2:0	Num Dark Rows	A value of N causes n + 1 dark rows to be read out at the start of each frame when dark row readout is enabled (bit 3).	7	N	Y
R36—0x24 - Extra Reset (R/W)					
Bit 15	Extra Reset Enable	0: Only programmed window (set by R0x01:0 through R0x04:0) and black pixels are read. 1: Two additional rows are read and reset above and below programmed window to prevent blooming to active area.	1	N	N
Bit 14	Next Row Reset	When set, and the integration time is less than one frame time, row n + 1 is reset immediately prior to resetting row n. This is intended to prevent blooming across rows under conditions of very high illumination.	0	N	N
Bits 13:0	Reserved	Do not change from default value.			
R37—0x25 - LINE_VALID Control (R/W)					
Bit 15	Xor LINE_VALID	0: Normal LINE_VALID (default, no XORing of LINE_VALID). Ineffective if continuous LINE_VALID is set. 1: LINE_VALID = "continuous" LINE_VALID XOR FRAME_VALID.	0	N	N
Bit 14	Continuous LINE_VALID	0: Normal LINE_VALID (default, no LINE_VALID during vertical blanking). Bad frame 1: "Continuous" LINE_VALID (continue producing LINE_VALID during vertical blanking).	0	N	N^2
R38—0x26 - Bottom Dark Rows (R/W)					
Bit 7	Show	The bottom dark rows are visible in the image if the bit is set.	0	N	N
Bits 6:4	Start Address	Defines the start address within the 8 bottom dark rows.	0	N	N
Bit 3	Enable Readout	Enables readout of the bottom dark rows.	0	N	Y
Bits 2:0	Number of Dark Rows	Defines the number of bottom dark rows to be used. (The number of rows used is the specified value + 1.)	7	N	Y
R43—0x2B - Green1 Gain (R/W)					
Bits 11:9	Digital Gain	Total gain = (bit 9 + 1)*(bit 10 + 1)*(bit 11 + 1)*analog gain (each bit gives 2x gain).	0	Y	N
Bits 8:7	Analog Gain	Analog gain = (bit 8 + 1)*(bit 7 + 1)*initial gain (each bit gives 2x gain).	0	Y	N

Table 5: Sensor Core Register Description (continued)

Bit Field	Description		Default (Hex)	Sync'd to Frame Start	Bad Frame
Bits 6:0	Initial Gain	Initial gain = bits 6:0*0.03125.	20	Y	N
R44—0x2C - Blue Gain (R/W)					
Bits 11:9	Digital Gain	Total gain = (bit 9 + 1)*(bit 10 + 1)*(bit 11 + 1)*analog gain (each bit gives 2x gain).	0	Y	N
Bits 6:0	Initial Gain	Initial gain = bits [6:0]*0.03125.	20	Y	N
Bits 8:7	Analog Gain	Analog gain = (bit 8 + 1)*(bit 7 + 1)*initial gain (each bit gives 2x gain).	0	Y	N
R45—0x2D - Red Gain (R/W)					
Bits 11:9	Digital Gain	Total gain = (bit 9 + 1)*(bit 10] + 1)*(bit 11 + 1)*analog gain (each bit gives 2x gain).	0	Y	N
Bits 8:7	Analog Gain	Analog gain = (bit 8 + 1)*(bit 7 + 1)*initial gain (each bit gives 2x gain).	0	Y	N
Bits 6:0	Initial Gain	Initial gain = bits 6:0*0.03125.	20	Y	N
R46—0x2E - Green2 Gain (R/W)					
Bits 11:9	Digital Gain	Total gain = (bit 9 + 1)*(bit 10 + 1)*(bit 11 + 1)*analog gain (each bit gives 2x gain).	0	Y	N
Bits 8:7	Analog Gain	Analog gain = (bit 8 + 1)*(bit 7 + 1)*initial gain (each bit gives 2x gain).	0	Y	N
Bits 6:0	Initial Gain	Initial gain = bits 6:0*0.03125.	20	Y	N
R47—0x2F - Global Gain (R/W)					
Bits 11:0	Global Gain	This register can be used to simultaneously set all 4 gains. When read, it returns the value stored in R0x2B:0.	20	Y	N
R48—0x30 - Row Noise (R/W)					
Bit 15	Frame-wise Digital Correction	By default, the row noise is calculated and compensated for individually for each color of each row. When this bit is set to "1", the row noise is calculated and applied for each color of each of the first two 2 pairs of values and the same values are applied to each subsequent row, so that new values are calculated and applied once per frame.	0	N	N
Bits 14:12	Gain Threshold	When the upper analog gain bits are equal to or larger than this threshold, the dark column average is used in the row noise correction algorithm. Otherwise, the subtracted value is determined by bit 11. This check is independently performed for each color, and is a means to turn off the black level algorithm for lower gains.	0	N	N
Bit 11	Use Black Level Average	0: Use mean of black level programmed threshold in the row noise correction algorithm for low gains. 1: Use black level frame average from the dark rows in the row noise correction algorithm for low gains. This frame average was taken before the last adjustment of the offset DAC for that frame, so it might be slightly off.	0	N	Y
Bit 10	Enable Correction	0: Normal operation. 1: Enable row noise cancellation algorithm. When this bit is set, the average value of the dark columns read out is used as a correction for the whole row. The dark average is subtracted from each pixel on the row, and then a constant is added (bits 9:0).	1	N	Y

Table 5: Sensor Core Register Description (continued)

Bit Field	Description		Default (Hex)	Sync'd to Frame Start	Bad Frame
Bits 9:0	Row Noise Constant	Constant used in the row noise cancellation algorithm. It should be set to the dark level targeted by the black level algorithm plus the noise expected between the averaged values of the dark columns. The default constant is set to 42 LSB.	2A	N	Y
R89—0x59 - Black Rows (R/W)					
Bits 7:0	Black Rows	For each bit set, the corresponding dark row (rows 0–7) are used in the black level algorithm. For this to occur, the reading of those rows must be enabled by the settings in R0x22:0.	FF	N	N
R91—0x5B - Green1 Frame Average (R/O)					
Bits 6:0	Green1 Frame Average	The frame-averaged green1 black level that is used in the black level calibration algorithm.			
R92—0x5C - Blue Frame Average (R/O)					
Bits 6:0	Blue Frame Average	The frame-averaged blue black level that is used in the black level calibration algorithm.			
R93—0x5D - Red Frame Average (R/O)					
Bits 6:0	Red Frame Average	The frame-averaged red black level that is used in the black level calibration algorithm.			
R94—0x5E - Green2 Frame Average (R/O)					
Bits 6:0	Green2 Frame Average	The frame-averaged green2 black level that is used in the black level calibration algorithm.			
R95—0x5F - Threshold (R/W)					
Bits 14:8	Upper Threshold	Upper threshold for targeted black level in ADC LSBs.	23	N	N
Bits 6:0	Lower Threshold	Lower threshold for targeted black level in ADC LSBs.	1D	N	N
R96—0x60 - Calibration Control (R/W)					
Bit 15	Disable Rapid Sweep Mode	Disables the rapid sweep mode in the black level algorithm. The averaging mode remains enabled.	0	Y	N
Bit 12	Recalculate	When set to "1", the rapid sweep mode is triggered if enabled, and the running frame average is reset to the current frame average. This bit is write – 1, but always reads back as "0."	0	Y	N
Bit 10	Limit Rapid Sweep	0: All dark rows can be used for the black level algorithm. This means that the internal average might not correspond to the calibration value used for the frame, so the dark row average should in this case not be used as the starting point for the digital frame-wise black level algorithm. 1: Dark rows 8–11 are not used for the black level algorithm controlling the calibration value. Instead, these rows are used to calculate dark averages that can be a starting point for the digital frame-wise black level algorithm.	0	N	N
Bit 9	Freeze Calibration	When set to "1", does not let the averaging mode of the black level algorithm change the calibration value. Use this with the feature in the frame-wise black level algorithm that allows you to trigger the rapid sweep mode when the dark column average gets away from the black level target.	0	N	N

Table 5: Sensor Core Register Description (continued)

Bit Field	Description		Default (Hex)	Sync'd to Frame Start	Bad Frame
Bit 8	Sweep Mode	When set to "1", the calibration value is increased by one every frame, and all channels are the same. This can be used to get a ramp input to the ADC from the calibration DACs.	0	N	N
Bits 7:5	Frames To Average Over	Two to the power of this value determines how many frames to average when the black level algorithm is in the averaging mode. In this mode, the running frame average is calculated from the following formula: Running frame ave = old running frame ave – (old running frame ave)/2^n + (new frame ave)/ 2^n. n = frames to average over.	4	N	N
Bit 4	Step Size Forced To 1	When set to "1", the step size is forced to 1 for the rapid sweep algorithm. Default operation (0) is to start at a higher step size when in rapid sweep mode, to converge faster to the correct value.	0	N	N
Bit 3	Switch Calibration Values	Reserved.	0		
Bit 2	Same Red/Blue	When set to "1", the same calibration value is used for red and blue pixels: Calib blue = calib red.	0	N	Y
Bit 1	Same Green	When set to "1", the same calibration value is used for all green pixels: Calib green2 = calib green1.	0	N	Y
Bit 0	Manual Override	Manual override of black level correction. 0: Normal operation (default). 1: Override automatic black level correction with programmed values. (R0x61:0–R0x64:0).	0	N	Y
R97—0x61 - Green1 Calibration Value (R/W)					
Bits 8:0	Green1 Calibration Value	Analog calibration offset for green1 pixels, represented as a two's complement signed 8-bit value (if bit 8 is clear, the offset is positive and the magnitude is given by bits 7:0. If bit 8 is set, the offset is negative and the magnitude is given by not ([7:0]) + 1). If R0x60:0[0] = 0, this register is R/O and returns the current value computed by the black level calibration algorithm. If R0x60:0[0] = 1, this register is R/W and can be used to set the calibration offset manually. Green1 pixels share rows with red pixels.	0	N	Y
R98—0x62 - Blue Calibration Value (R/W)					
Bits 8:0	Blue Calibration Value	Analog calibration offset for blue pixels, represented as a two's complement signed 8-bit value (if bit 8 is clear, the offset is positive and the magnitude is given by bits 7:0. If bit 8 is set, the offset is negative and the magnitude is given by Not ([7:0]) + 1). If R0x60:0[0] = 0, this register is R/O and returns the current value computed by the black level calibration algorithm. If R0x60:0[0] = 1, this register is R/W and can be used to set the calibration offset manually.	0	N	Y

Table 5: Sensor Core Register Description (continued)

Bit Field		Description	Default (Hex)	Sync'd to Frame Start	Bad Frame
R99—0x63 - Red Calibration Value (R/W)					
Bits 8:0	Red Calibration Value	Analog calibration offset for red pixels, represented as a two's complement signed 8-bit value (if bit 8 is clear, the offset is positive and the magnitude is given by bits 7:0. If bit 8 is set, the offset is negative and the magnitude is given by Not ([7:0]) + 1). If R0x60:0[0] = 0, this register is R/O and returns the current value computed by the black level calibration algorithm. If R0x60:0[0] = 1, this register is R/W and can be used to manually set the calibration offset.	0	N	Y
R100—0x64 - Green2 Calibration Value (R/W)					
Bits 8:0	Green2 Calibration Value	Analog calibration offset for green2 pixels, represented as a two's complement signed 8-bit value (if bit 8 is clear, the offset is positive and the magnitude is given by bits 7:0. If bit 8 is set, the offset is negative and the magnitude is given by Not ([7:0]) + 1.) If R0x60:0[0] = 0, this register is R/O and returns the current value computed by the black level calibration algorithm. If R0x60:0[0] = 1, this register is R/W and can be used to manually set the calibration offset. Green2 pixels share rows with blue pixels.	0	N	Y
R101—0x65 - Clock (R/W)					
Bit 15	PLL Bypass	0: Use clock produced by PLL as master clock. 1: Bypass the PLL. Use EXTCLK input signal as master clock.	1	N	N
Bit 14	PLL Power-down	0: PLL powered-up. 1: Keep PLL in power-down to save power (default).	1	N	N
Bit 13	Power-down PLL During Standby	This register only has an effect when bit 14 = 0. 0: PLL powered-up during standby. 1: Turn off PLL (power-down) during standby to save power (default).	1	N	N
Bit 2	clk_newrow	Force clk_newrow to be on continuously.	0	N	N
Bit 1	clk_newframe	Force clk_newframe to be on continuously.	0	N	N
Bit 0	clk_ship	Force clk_ship to be on continuously.	0	N	N
R102—0x66 - PLL Control 1 (R/W)					
Bits 15:8	M	M value for PLL must be 16 or higher.	10	N	N
Bits 5:0	N	N value for PLL.	00	N	N
R103—0x67 - PLL Control 2 (R/W)					
Bits 11:8	Reserved	Do not change from default value.			
Bits 6:0	P	P value for PLL.	00	N	N
R240—0xF0 - Page Register (R/W)					
Bits 2:0	Page Register	Must be kept at "0" to be able to write/read from sensor. Used in SOC to access other pages with registers.	0	N	N
R241—0xF1 - ByteWise Address (R/W)					
Bits 15:0	Bytewise Address	Special address to perform 16-bit reads and writes to the sensor in 8-bit chunks. See "8-Bit Write Sequence" on page 128.	0	N	N

Table 5: Sensor Core Register Description (continued)

Bit Field	Description		Default (Hex)	Sync'd to Frame Start	Bad Frame
R242—0xF2 - Context Control (R/W)					
Bit 15	Restart	Setting this bit causes the sensor to abandon the current frame and start resetting the first row. Same physical register as R0x0D:0[1].	0	N	YM
Bit 7	Reserved	Reserved.	0	Y	N
Bit 3	Read Mode Select	0: Use read mode context A, R0x21:0. 1: Use read mode context B, R0x20:0. Bits only found in read mode context B register are always taken from that register.	1	Y	YM
Bit 2	Reserved	Reserved.	0	Y	Y
Bit 1	Vertical Blank Select	0: Use vertical blanking context A, R0x08:0. 1: Use vertical blanking context B, R0x06:0.	1	Y	YM
Bit 0	Horizontal Blank Select	0: Use horizontal blanking context A, R0x07:0. 1: Use horizontal blanking context B, R0x05:0.	1	Y	YM
R255—0xFF - Reserved (R/O)					
Bits 15:0	Reserved	Reserved.	1519		

Notes: 1. Unless integration time is less than one frame (see R0x0B[13;0]).
2. f enabled by bit 3 (see R0x25[14]).

IFP Registers, Page 1

Table 6: IFP Registers, Page 1

Reg #	Bits	Default	Name
8 0x08	10:0	0x01F8	Color Pipeline Control
	0	0	Toggles the assumption about Bayer CFA (vertical shift). 0: Row containing Blue comes first. 1: Row with Red comes first.
	1	0	Toggles the assumption about Bayer CFA (horizontal shift). 0: Green comes first. 1: Red or Blue comes first.
	2	0	1: Enables lens shading correction.
	3	1	1: Enables on-the-fly defect correction.
	4	1	1: Enables 2D aperture correction.
	5	1	0: Bypasses color correction (unity color matrix). 1: Enables color correction.
	6	1	1: Inverts output pixel clock (in all modes - JPEG, SOC, sensor).
	7	1	1: Enables gamma correction.
	8	1	1: Enables decimator.
	9	0	1: Enables minblank. Allows len generation 2 lines earlier. Also adds 4 pixels to the line size.
	10	0	1: Enables 1D aperture correction.
9 0x09	4:0	0x000A	Factory Bypass
	1:0	1	Data output bypass. Selects data going to DOUT pads. 00: 10-bit sensor. 01: SOC. 10: JPEG and output FIFO (no bypass). Reg16 and Reg17 (Page 2) have to be set to the correct output frame size. 11: Reserved.
	2	0	Reserved.
	4:3	1	Reserved.
10 0x0A	10:0	0x0488	Pad Slew
	2:0	0	In bypass mode (R9[1:0] = 2): slew rate for DOUT[7:0], PIXCLK, FRAME_VALID, and LINE_VALID. During normal operation, the slew of listed pads is set by JPEG configuration registers. Actual slew depends on load, temperature, and I/O voltage. Set this register based on customers' characterization results.
	3	1	Unused.
	6:4	0	Reserved.
	7	0	0: Disable I/O pad input clamp. Set this bit to "1" before going to soft/hard standby to reduce the leakage current. When coming out of standby, set this bit back to "0." 1: Enables I/O pad input clamp during standby.That prevents elevated standby current if pad input floating. The pads are DOUT[7:0], FRAME_VALID, LINE_VALID, PIXCLK.
	10:8	4	Slew rate for SDATA. 7 = fastest slew; 0 = slowest. Actual slew depends on load, temperature, and I/O voltage. Actual slew depends on load, temperature, and I/O voltage. Set this register based on customers' characterization results.

Table 6: IFP Registers, Page 1 (continued)

Reg #	Bits	Default	Name
11 0x0B	8:0	0x00DF	Internal Clock Control
	0	1	Reserved.
	1	1	Reserved.
	2	1	Reserved.
	3	1	1: Enables output FIFO clock.
	4	1	1: Enables JPEG clock.
	5	0	Reserved.
	6	1	Reserved.
	7	1	Reserved.
	8	0	Reserved.
17 0x11	10:0	0x0000	X0 Coordinate for Crop Window
	Use the crop window for pan and zoom. Crop coordinates are updated synchronously with frame enable, unless freeze is enabled. In preview mode crop coordinates are automatically divided by X skip factor. Coordinates are specified as (X0, Y0) and (X1, Y1), where X0 < X1 and Y0 < Y1. Value is overwritten by mode driver (ID = 7).		
18 0x12	10:0	0x0640	X1 Coordinate for Crop Window +1
	See R17:1. Value is overwritten by mode driver (ID = 7).		
19 0x13	10:0	0x0000	Y0 Coordinate for Crop Window
	See R17:1. Value is overwritten by mode driver (ID = 7).		
20 0x14	10:0	0x04B0	Y1 Coordinate for Crop Window +1
	See R17:1. Value is overwritten by mode driver (ID = 7).		
21 0x15	13:0	0x0000	Decimator Control
	0	0	Reserved.
	1	0	Reserved.
	2	0	High precision mode. Additional bits for result are stored. Can only be used for decimation > 2.
	3	0	Reserved.
	4	0	Enables 4:2:0 mode.
	5:6	0	Reserved.
	This register controls operation of the decimator. Value is overwritten by mode driver (ID = 7).		
22 0x16	12:0	0x0800	Weight for Horizontal Decimation X output = int (X input/2048*reg. value). Value is calculated and overwritten by mode driver (ID = 7).
23 0x17	12:0	0x0800	Weight for Vertical Decimation Y output = int (Y input/2048*reg. value). Value is calculated and overwritten by mode driver (ID = 7).
	InputSize is defined by Registers 17–20. Minimal output size supported by the decimator is 3 x 1 pixel.		
32 0x20	15:0	0xC814	Luminance Range of Pixels Considered in WB Statistics
	7:0	20	Lower limit of luminance for WB statistics.
	15:8	200	Upper limit of luminance for WB statistics.
	To avoid skewing WB statistics by very dark or very bright values, this register allows programming the luminance range of pixels to be used for WB computation.		

Table 6: IFP Registers, Page 1 (continued)

Reg #	Bits	Default	Name
45 0x2D	15:0	0x5000	Right/Left Coordinates of AWB Measurement Window
	7:0	0	Left window boundary.
	15:8	80	Right window boundary.
	This register specifies the right/left coordinates of the window used by AWB measurement engine. The values programmed in the registers are desired boundaries divided by 8.		
46 0x2E	15:0	0x3C00	Bottom/Top Coordinates of AWB Measurement Window
	7:0	0	Top window boundary.
	15:8	60	Bottom window boundary.
	This register specifies the bottom/top coordinates of the window used by AWB measurement engine. The values programmed in the registers are desired boundaries divided by 8.		
48 0x30	7:0	R/O	Red Chrominance Measure Calculated by AWB
	This register contains a measure of red chrominance obtained using AWB measurement algorithm. The measure is normalized to an arbitrary maximum value; the same for R48:1, R49:1, and R50:1. Because of this normalization, only the ratios of values of registers R48:1 R49:1, and R50:1 should be used.		
49 0x31	7:0	R/O	Luminance Measure Calculated by AWB
	This register contains a measure of image luminance obtained using AWB measurement algorithm. The measure is normalized to an arbitrary maximum value; the same for R48:1, R49:1, and R50:1. Because of this normalization, only the ratios of values of registers R48:1, R49:1, and R50:1 should be used.		
50 0x32	7:0	R/O	Blue Chrominance Measure Calculated by AWB
	This register contains a measure of blue chrominance obtained using AWB measurement algorithm. The measure is normalized to an arbitrary maximum value; the same for R48:1, R49:1, and R50:1. Because of this normalization, only the ratios of values of registers R48:1, R49:1, and R50:1 should be used.		
53 0x35	15:0	0x1208	1D Aperture Correction Parameters
	7:0	8	Ap_knee; threshold for aperture signal.
	10:8	2	Ap_gain; gain for aperture signal.
	13:11	2	Ap_exp; exponent for gain for aperture signal.
54 0x36	15:0	0x1208	2D Aperture Correction Parameters
	7:0	8	Ap_knee; threshold for aperture signal.
	10:8	2	Ap_gain; gain for aperture signal.
	13:11	2	Ap_exp; exponent for gain for aperture signal.
	14	0	Reserved.
	Defines 2D aperture gain and threshold.		

Table 6: IFP Registers, Page 1 (continued)

Reg #	Bits	Default	Name
55 0x37	7:0	0x080	Filters
	2:0	0	UV filter: 000: No filter 001: 11110 averaging filter 010: 01100 averaging filter 011: 01210 averaging filter 100: 12221 averaging filter 101: Median 3 110: Median 5 111: Reserved
	4:3	0	Y filter mode: 00: No filter 01: Median 3 10: Median 5 11: Reserved
	5	0	Permanently enables Y filter.
59 0x3B	9:0	0	Second Black Level
	This register contains the value subtracted by IFP from pixel values prior to CCM. If the subtraction produces a negative result for a particular pixel, the value of this pixel is set to "0."		
60 0x3C	9:0	42	First Black Level
	This register contains the value subtracted by IFP from raw pixel values before applying lens shading correction and digital gains. If the subtraction produces a negative result for a particular pixel, the value of this pixel is set to "0." Typically, the subtracted value should be equal to the black level targeted by the sensor. This value is subtracted from all test patterns as well.		
67 0x43	0	1	Enable Support for Preview Modes
	1: Enables automatic recalculation operation of coefficient lens correction dependent on sensor output resolution.		
68 0x44	15:0	N/A	Mirrors Sensor Register 0x20
	Read only		
69 0x45	15:0	N/A	Mirrors Sensor Register 0xF2
	Read only		
70 0x46	15:0	N/A	Mirrors Sensor Register 0x21
	Read only		
71 0x47	7:0	16	Edge Threshold for Interpolation
	Threshold for identifying pixel neighborhood as having an edge.		
72 0x48	2:0	0	Test Pattern
	2:0	0	Test mode. 001: Flat field; RGB values are specified in R73-75:1 010: Vertical monochrome ramp 011: Vertical color bars 100: Vertical monochrome bars; set bar intensity in R73:1 and R74:1 101: Pseudo-random test pattern 110: Horizontal monochrome bars; set bar intensity in R73:1 and R74:1
	Injects test pattern into beginning of color pipeline.		
73 0x49	9:0	256	Test Pattern R/Monochrome Value
74 0x4A	9:0	256	Test Pattern G/Monochrome Value

Table 6: IFP Registers, Page 1 (continued)

Reg #	Bits	Default	Name
75 0x4B	9:0	256	Test Pattern B Value
78 0x4E	7:0	32	Digital Gain 2
	Default setting 32 corresponds to gain value of 1. Gain scales linearly with value.		
96 0x60	14:0	0x2923	Color Correction Matrix Exponents for C11...C22 2:0: Matrix element 1 (C11) exponent 5:3: Matrix element 2 (C12) exponent 8:6: Matrix element 3 (C13) exponent 11:9: Matrix element 4 (C21) exponent 14:12: Matrix element 5 (C22) exponent
97 0x61	11:0	0x0524	Color Correction Matrix Exponents for C22...C33 2:0: Matrix element 6 (C23) exponent 5:3: Matrix element 7(C31) exponent 8:6: matrix element 8(C32) exponent 11:9: Matrix element 9(C33) exponent The value of a matrix coefficient is calculated Cij = (1 - 2*Sij)*Mij*2^-(Eij + 4), $0 <= Eij < = 4$ Where Sij is coefficient's sign, Mij is the mantissa, and Eij is the exponent.
98 0x62	15:0	0xBBC8	Color Correction Matrix Elements 1 and 2 Mantissas
	7:0	200	Color correction matrix element 1 (C11).
	15:8	187	Color correction matrix element 2 (C12).
99 0x63	15:0	0xCA3A	Color Correction Matrix Elements 3 and 4 Mantissas
	7:0	58	Color correction matrix element 3 (C13).
	15:8	202	Color correction matrix element 4 (C21).
100 0x64	15:0	0x3B85	Color Correction Matrix Elements 5 and 6 Mantissas
	7:0	133	Color correction matrix element 5 (C22).
	15:8	59	Color correction matrix element 6 (C23).
101 0x65	15:0	0xF26F	Color Correction Matrix Elements 7 and 8 Mantissas
	7:0	111	Color correction matrix element 7 (C31).
	15:8	242	Color correction matrix element 8 (C32).
102 0x66	13:0	0x3D9C	Color Correction Matrix Element 9 Mantissa and Signs
	7:0	156	Color correction matrix element 9 (C33).
	13:8	61	Signs for off-diagonal CCM elements. Bit 8: sign for C12 Bit 9: sign for C13 Bit 10: sign for C21 Bit 11: sign for C23 Bit 12: sign for C31 Bit 13: sign for C32 1: indicates negative 0: indicates positive Signs for C11, C22, and C33 are assumed always positive.
106 0x6A	9:0	128	Digital Gain 1 for Red Pixels
	Default setting 128 corresponds to gain value of 1. Gain scales linearly with value.		
107 0x6B	9:0	128	Digital Gain 1 for Green1 Pixels
	Default setting 128 corresponds to gain value of 1. Gain scales linearly with value.		

Table 6: IFP Registers, Page 1 (continued)

Reg #	Bits	Default	Name
108 0x6C	9:0	128	Digital Gain 1 for Green2 Pixels
	Default setting 128 corresponds to gain value of 1. Gain scales linearly with value.		
109 0x6D	9:0	128	Digital Gain 1 for Blue Pixels
	Default setting 128 corresponds to gain value of 1. Gain scales linearly with value.		
110 0x6E	9:0	128	Digital Gain 1 for All Colors
	Write 128 to set all gains (R106-R109) to 1. When read, this register returns the value of R107.		
122 0x7A	15:0	80	Boundaries of Flicker Measurement Window (Left/Width)
	7:0	80	Window width.
	15:8	0	Left window boundary.
	This register specifies the boundaries of the window used by the flicker measurement engine. The values programmed in the registers are the desired boundaries divided by 8.		
123 0x7B	15:0	0x0088	Boundaries of Flicker Measurement Window (Top/Height)
	5:0	8	Window height.
	15:6	2	Top window boundary.
	This register specifies the boundaries of the window used by the flicker measurement engine.		
124 0x7C	15:0	5120	Flicker Measurement Window Size
	This register specifies the number of pixels in the window used by the flicker measurement engine.		
125 0x7D	15:0	R/O	Measure of Average Luminance in Flicker Measurement Window
150 0x96	0	0	Blank Frames
	0	0	0: When bit is unset, output resumes with the next frame. The bit is synchronized with frame enable. See freeze bit (R152[7])1: Blank outgoing frames. 1: When bit is set, current frame completes and following frames are not output, FEN = LEN = 0.
151 0x97	7:0	0	Output Format Configuration
	0	0	In YUV output mode, swaps Cb and Cr channels. In RGB mode, swaps R and B. This bit is subject to synchronous update.
	1	0	Swaps chrominance byte with luminance byte in YUV output. In RGB mode, swaps odd and even bytes. This bit is subject to synchronous update.
	2	0	Reserved.
	3	0	Monochrome output.
	4	0	1: Uses ITU-R BT.601 codes when bypassing FIFO. (0xAB = frame start; 0x80 = line start; 0x9D = line end; 0xB6 = frame end.)
	5	0	0: Output YUV 1: Output RGB (see R151[7:6])
	7:6	0	RGB output format: 00: 16-bit RGB565 01: 15-bit RGB555 10: 12-bit RGB444x 11: 12-bit RGBx444

Table 6: IFP Registers, Page 1 (continued)

Reg #	Bits	Default	Name
152 0x98	2:0	0	Output Format Test
	0	0	1: Disables Cr channel (R in RGB mode).
	1	0	1: Disables Y channel (G in RGB mode).
	2	0	1: Disables Cb channel (B in RGB mode).
	5:3	0	Test ramp output: 00: Off 01: By column 10: By row 11: By frame
	6	0	1: Enables 8+2 bypass.
	7	0	1: Freezes update of SOC registers affecting output size and format. These include R151:1, R17-23:1.
153 0x99	11:0	R/O	Line Count
154 0x9A	15:0	R/O	Frame Count
164 0xA4	15:0	0x6440	Special Effects
	2:0	0	Special effect selection bits: 0: Disabled 1: Monochrome 2: Sepia 3: Negative 4: Solarization with unmodified UV 5: Solarization with -UV
	5:3	0	Dither enable and amplitude. Valid values are 1...4, others disable.
	6	1	1: Dithers only in luma channel, 0: dither in all color channels.
	15:8	0	Solarization threshold for luma, divided by 2; 64...127.
165 0xA5	15:0	0xB023	Sepia Constants
	7:0	35	Sepia constant for Cr.
	15:8	176	Sepia constant for Cb.
178 0xB2	15:0	0x2700	Gamma Curve Knees 0 and 1
	7:0		Ordinate of gamma curve knee point 0 (its abscissa is 0).
	15:8		Gamma curve knee point 1.
	Gamma correction curve is implemented in the MT9D131 as a piecewise linear function mapping 12-bit arguments to 8-bit output. Seventeen line fragments forming the curve connect 19 knee points, whose abscissas are fixed and ordinates are programmable through registers R[178:187]:1. The abscissas of the knee points are 0, 64, 128, 256, 512, 768, 1024, 1280, 1536, 1792, 2048, 2304, 2560, 2816, 3072, 3328, 3584, 3840, and 4095. Ordinates of knee points written directly to the registers R[178:187]:1 may be overwritten by mode driver (ID = 7). The recommended way to program a gamma curve into the MT9D131 is to write desired knee ordinates to one of the two mode.gamma_table[A/B][] arrays that hold gamma curves used in contexts A and B. Once the desired ordinates are in one of those arrays, all that is needed to put them into effect (that is, into the registers) is a short command telling the sequencer to go to proper context or through REFRESH loop.		
179 0xB3	15:0	0x4936	Gamma Curve Knees 2 and 3
	7:0		Gamma curve knee point 2.
	15:8		Gamma curve knee point 3.

Table 6: IFP Registers, Page 1 (continued)

Reg #	Bits	Default	Name
180 0xB4	15:0	0x7864	Gamma Curve Knees 4 and 5
	7:0		Gamma curve knee point 4.
	15:8		Gamma curve knee point 5.
181 0xB5	15:0	0x9789	Gamma Curve Knees 6 and 7
	7:0		Gamma curve knee point 6.
	15:8		Gamma curve knee point 7.
182 0xB6	15:0	0xB0A4	Gamma Curve Knees 8 and 9
	7:0		Gamma curve knee point 8.
	15:8		Gamma curve knee point 9.
183 0xB7	15:0	0xC5BB	Gamma Curve Knees 10 and 11
	7:0		Gamma curve knee point 10.
	15:8		Gamma curve knee point 11.
184 0xB8	15:0	0xD7CE	Gamma Curve Knees 12 and 13
	7:0		Gamma curve knee point 12.
	15:8		Gamma curve knee point 13.
185 0xB9	15:0	0xE8E0	Gamma Curve Knees 14 and 15
	7:0		Gamma curve knee point 14.
	15:8		Gamma curve knee point 15.
186 0xBA	15:0	0xF8F0	Gamma Curve Knees 16 and 17
	7:0		Gamma curve knee point 16.
	15:8		Gamma curve knee point 17.
187 0xBB	7:0	0x00FF	Gamma Curve Knee 18
190 0xBE	3:0	6	YUV/YCbCr control
	3	0	Clips Y values to 16–235; clips UV values to 16–240.
	2	1	Adds 128 to U and V values.
	1	1	0: Uses sRGB coefficients. 1: ITU-R BT.601 coefficients
	0	0	0: No scaling 1: Scales Y data by 219/256 and UV data by 224/256.
191 0xBF	15:0	0	Y/RGB Offset
	15:8	0	Y offset.
	7:0	0	RGB offset.
195 0xC3	15:0	0	Microcontroller Boot Mode
	0	0	1: Reset microcontroller.
	1,2,3,6:4, 6:5,7:5	0	Reserved.
	7	0	Microcontroller debug indicator.
	11:8,12,13,1 4,15	R/O	Reserved.

Table 6: IFP Registers, Page 1 (continued)

Reg #	Bits	Default	Name
198 0xC6	15:0	0	Microcontroller Variable Address
	7:0	0	Bits 7:0 of address for physical access; driver variable offset for logical access.
	12:8	0	Bits 12:8 of address for physical access; driver ID for logical access.
	14:13	0	Bits 14:13 of address for physical access; R0xC6:1[14:13] = 01 select logical access.
	15	0	0: 16-bit 1: 8-bit access
	Microcontroller variables are similar to two-wire serial interface registers, except that they are located in the microcontroller's memory and have different bit widths. Registers 198:1 and R200:1 provide easy access to variables that can be represented as 8- or 16-bit unsigned integers (bytes or words). To access such a variable, one must write its address to R0xC6:1 and then read its value from R200:1 or write a new value to the same register. Variables having more than 16 bits (for example 32-bit unsigned long integers) must be accessed as arrays of bytes or words - there is no way to read or write their values without parsing. Variable address written to R0xC6:1 can be physical or logical. Physical address is the actual address of a byte or word in the microcontroller's address space. The logical addressing option is provided only to facilitate access to public variables of various firmware drivers. Logical address of a public variable consists of a 5-bit driver ID 1 = sequencer, and so on) and 8-bit offset of the variable in the driver's data structure (which cannot be larger than 256 bytes).		
200 0xC8	15:0	0	Microcontroller Variable Data
	To read current value of an 8- or 16-bit variable from microcontroller memory, write its address to R0xC6:1 and read R200:1. To change value of a variable, write its address to R0xC6:1 and its new value to R200:1. When bit width of a variable is specified as 8 bits (R0xC6:1[15] = 1), the upper byte of R200:1 is irrelevant. It is set to "0" when the register is read and ignored when it is written to. See R0xC6:1 above for explanation how to read and set variables having more than 16 bits.		
201-209 0xC9-D1	15:0	0	Microcontroller Variable Data using Burst Two-Wire Serial Interface Access
	Use these registers to read or write up to 16 bytes of variable data using the burst two-wire serial interface access mode. The variables must have consecutive addresses.		
240 0xF0	2:0	1	Page Register 0: Sensor core 1: IFP page 1 2: IFP page 2
241 0xF1	15:0	0	Bytewise Address Special address to perform 16-bit READs and WRITEs to the sensor in 8-bit chunks. See "8-Bit Write Sequence" on page 128.

IFP Registers, Page 2

Table 7: IFP Registers, Page 2

Reg #	Bits	Default	Name
0 0x00	15:0	0	JPEG Control Register
	0	0	Start/Enable Encoder: Enable JPEG encoding at the start of next frame.
	1	0	Test SRAM: When set, allows host or microcontroller to take control of the output FIFO buffer and the sixteen 800 x 16 RAMs in the re-order buffer for testing. Used in conjunction with JPEG RAM test controls register to simultaneously write all 17 RAMs and individually read each RAM.
	[14:2]		Return zero when read.
	15	0	Soft Reset.

Table 7: IFP Registers, Page 2 (continued)

Reg #	Bits	Default	Name
2 0x02	15:0	0	JPEG Status Register 0
	0	0	Transfer done status flag. When bit is set to "1", it indicates that the completion of transfer of the JPEG-compressed image. This status flag remains set until cleared by the host or microcontroller by writing "1" to bit [0]. Subsequently, the output FIFO overflow, the spoof oversize error and the re-order buffer error status bits are reset. The output buffer clock must be present to clear this bit.
	1	0	Output FIFO overflow status flag. When bit is set to "1", it indicates that an overflow condition was detected in the output FIFO during the frame transfer and that transfer was terminated prematurely. This status flag remains set until cleared by the host or microcontroller as it clears transfer done flag. Valid for JPEG compressed images only.
	2	0	Spoof oversize error status flag. When bit is set to "1", it indicates that the spoof frame size is too small for JPEG data stream. This status flag remains set until cleared by the host or microcontroller as it clears transfer done flag. Valid for JPEG compressed images only.
	3	0	Re-order buffer error status flag: When the re-order buffer detects an overflow or underflow condition, this bit is set to "1." This bit is cleared by writing a "1" to bit[0] of this register.
	5:4	0	Watermark of the output FIFO. 00: Less than 25% full 01: 25% to less than 50% full 10: 50% to less than 75% full 11: 75% full or more Watermark is cleared when the host or microcontroller writes "1" to bit 4 of this register (R258).
	7:6	0	QTable_ID. 00: Quantization table set 0 01: Quantization table set 1 10: Quantization table set 2 11: Reserved
	15:8	0	JPEG data length bits 23:16. Highest byte of 24-bit JPEG data length.
3 0x03	15:0	0	JPEG Status Register 1 - JPEG Data Length Bits 15:0
	This register combined with R2:2[15:8] gives the total number of data bytes successfully encoded—a 24-bit JPEG data length. If an output FIFO overflow occurs, this register holds the total number of data bytes already sent out by the JPEG encoder (up to the point where the overflow occurs).		
4 0x04	2:0	0	JPEG Status Register 2 - Output FIFO Fullness Status
	Instantaneous FIFO fullness status code: 000: FIFO is empty 001: 0% < fullness < 25% 011: 25% < = fullness < 50% 010: 50% < = fullness < 75% 110: 75% < = fullness < = 100%		
5 0x05	5:0	0	JPEG Front End Configuration Register
	1	0	JPEG monochrome mode. When this bit is set, the re-order buffer control sends only luma data to the JPEG encoder.
	0	0	Color component composition: 0: 4:2:2 format 1: 4:2:0 format
6 0x06	0:0	0	JPEG Core Configuration Register - Extend JPEG Quantization Matrix
	Presently, the JPEG encoder core supports two sets of quantization tables. (Each set contains a pair of quantization tables - one for luma, one for chroma.) The quantization memory available allows an additional set of quantization tables to be stored. By setting this bit to "1", the encoder uses the third table pair. If the bit is set to "0", then whichever set of tables 1 and 2 was programmed will be used.		

Table 7: IFP Registers, Page 2 (continued)

Reg #	Bits	Default	Name
10 0x0A	0:0	1	JPEG Encoder Bypass
	Set bit 0 to "1" of this register to have uncompressed frames from the SOC captured in the output FIFO and transferred out as spoof frames (JPEG encoder is bypassed). Register 13, bit 0, should be set to "1" for spoof-mode output. When the bit is cleared (set to "0"), JPEG-encoded frames are transferred out through the FIFO.		
13 0x0D	10:0	0x0007	Output Configuration Register
	0	1	Enable spoof frame: When this bit is set to "1", the data captured in the output FIFO is sent out as a spoofed frame, formatted according to information stored in the various spoof registers. This output mode can be used to output both JPEG-compressed and uncompressed image data. JPEG data may be padded if dummy data is needed. During LINE_VALID assertion period, the output clock is only clocked when there is valid JPEG data or padded dummy data to be transmitted. This may result in a non-uniform clock period. When LINE_VALID is de-asserted, the output clock is enabled according to SPOOF_LV_LEAD and SPOOF_LV_TRAIL setting. JPEG SOI/EOI markers cannot be inserted into spoof frames.
	1	1	Enable output pixel clock between frames: When this bit is set to "1", this bit enables the pixel clock to run whether FRAME_VALID is LOW or HIGH. Clearing the bit disables the clock during the periods when FRAME_VALID is de-asserted except for SOI/EOI markers transmission if they are outside the FV assertion period. The gating off of this output pixel clock saves power.
	2	1	Enable pixel clock during invalid data: When this bit is set to "1", the pixel clock runs continuously while FRAME_VALID is asserted but LINE_VALID is de-asserted. When cleared, it causes the pixel clock to be active only when valid JPEG data are output. Disabling pixel clock during LINE_VALID de-assertion is not support in spoof frame.
	3	0	Enable SOI/EOI insertion: When this bit is set to "1", this bit causes SOI and EOI markers to be output at the beginning and end of every JPEG-encoded frame. When the bit is cleared, only JPEG data bytes are output.
	4	0	Insert SOI and EOI when FRAME_VALID is HIGH: When this bit is set to "1", SOI and EOI are inside FV assertion period. When it is"0," SOI, and EOI are outside FRAME_VALID assertion period. This bit is relevant only when SOI/EOI insertion is enabled.
	5	0	Ignore spoof frame height: This bit is used in conjunction with bit 0 that enables spoof framing. When this bit is set to "1", the JPEG unit output section ignores the spoof frame height register. The spoof frame ends when either JPEG bytes or uncompressed image data are exhausted. Both kinds of data are always padded with dummy data to the programmed spoof frame width.
	6	0	Enable variable pixel clock rate: When this bit is set to "1", it enables the adaptive pixel clock frequency feature. The pixel clock is switched from PCLK1 to PCLK2 to PCLK3, depending on the output FIFO fullness. When fullness reaches 50%, the pixel clock is switched from PCLK1 to PCLK2. When the fullness reaches 75%, the clock is switched to PCLK3. As the FIFO fullness drops below 50%, PIXCLK switches from PCLK3 to PCLK2; as it drops below 25%, it switches back to PCLK1. At the start of a frame, it always starts with PCLK1.
	7	0	Enable byte swap: Toggling this bit swaps the even and odd bytes in JPEG data stream. Byte swapping supported only when enable spoof frame is set.

Table 7: IFP Registers, Page 2 (continued)

Reg #	Bits	Default	Name
	8	0	Duplicate FRAME_VALID on LINE_VALID: When this bit is set to "1", the FRAME_VALID waveform is output on LINE_VALID output also; therefore, the two are identical.
	9	0	Enable status insertion: When this bit is set to "1", the JPEG module appends the status byte to the end of the JPEG byte stream. This register should only be set when transferring in continuous mode. The status byte inserted at the end of the JPEG byte stream is Reg2 bit [7:0].
	10	0	Enable spoof ITU-R BT.601 codes: This bit is relevant matters when uncompressed frames are output in the spoof mode. When this bit is set to "1", the bit causes the ITU-R BT.601 markers SOF, EOF, SOL, EOL to be inserted into every frame. Codes are: Start of Frame: FF0000AB End of Frame: FF0000B6 Start of Line: FF000080 End of Line: FF00009D
	11	0	Freeze_update; Default = 0. When this bit is set to "1", it disables the transfer of values from registers 0x0A, 0x0D (except freeze_update), 0x0E, 0x0F, 0x10, 0x11, 0x12 into their corresponding shadow registers. When cleared, the shadow registers are updated with values from their corresponding configuration registers at the end of vertical blanking of the input image. The shadow registers allow the microcontroller to change configuration registers values during the active frame time without corrupting the current frame transfer.
14 0x0E	15:0	0x0501	Output PCLK1 & PCLK2 Configuration Register
	3:0	0x1	Output clock frequency divisor N1: This 4-bit register contains an integer divisor used to divide master clock frequency to obtain the frequency of output clock source PCLK1. A value of "0" in this register has the same effect as "1."
	7:5	0	PCLK1 slew rate: The value contained in this 3-bit register determines the slew rate of the output clock when PCLK1 is selected as its source.
	11:8	0x5	Output Clock Frequency Divisor N2: This 4-bit register contains an integer divisor used to divide master clock frequency to obtain the frequency of output clock source PCLK2. The output clock switches from PCLK1 to PCLK2 when the output buffer fullness reaches 50%. A value of "0" in this register has the same effect as "1."
	12	0	Not used.
	15:13	0	PCLK2 slew rate: The value contained in this 3-bit register determines the slew rate of the output clock when PCLK2 is selected as its source.
15 0x0F	7:0	0x0003	Output PCLK3 Configuration Register
	3:0	0x03	Output clock frequency divisor N3: This 4-bit register contains an integer divisor used to divide master clock frequency to obtain the frequency of output clock source PCLK3. The output clock switches from PCLK2 to PCLK3 when the output buffer fullness reaches 75 %. A value of "0" in this register has the same effect as 1.
	4	0	Not used.
	7:5	0	PCLK3 slew rate: The value contained in this 3-bit register determines the slew rate of the output clock when PCLK3 is selected as its source.
16 0x10	11:0	0x0640	Spoof Frame Width
	This register defines the width of the spoof frame used to output data captured in the output FIFO. It corresponds to the real time assertion of LINE_VALID in terms of number of PIXCLKs. It must be an even number.		
17 0x11	11:0	0x0258	Spoof Frame Height
	This register defines the height of the spoof frame used to output data captured in the output FIFO. The height is equal to the number of assertions of LINE_VALID within one assertion of FRAME_VALID. The value stored in this register is ignored if bit R13:2[5] is set.		

Table 7: IFP Registers, Page 2 (continued)

Reg #	Bits	Default	Name
18 0x12	15:0	0x0606	Spoof Frame Line Timing
	7:0	0x6	Spoof LINE_VALID lead. The number of clocks before LINE_VALID is asserted in a spoof frame. This must be a minimum value of "5."
	15:8	0x6	Spoof LINE_VALID trail. The number of clocks after LINE_VALID is de-asserted in a spoof frame. This must be a minimum value of "5."
29 0x1D	15:0	0	JPEG RAM Test Control Register
	9:0	0	Tested SRAM address: This register defines the location in the seventeen 800 x 16 RAMs that is being accessed by the host or microcontroller while testing the group of SRAMs. A specified 16-bit location can be selected from 0 through 799. The address is automatically incremented when R31:2 is written during SRAM test data WRITE or READ during SRAM test data READ.
	12:10	0	Test Y/C SRAM Register: Address the same set of 8 SRAMs (either for Y or for C), this register setting identifies which of the 8 SRAMs is selected. 000 for SRAM1, 001 for SRAM2, ..., 111 for SRAM8.
	13	0	Test Y/C SRAM Select Register: Select a bank of 8 SRAMs from luminance or from chrominance. Y = 1, C = 0.
	14	0	Test output buffer SRAM: Select to read output buffer SRAM and supersedes Y/C SRAMs. If this bit is set to "1", data from the output buffer RAM is selected to be read. During data WRITE, this has no effect.
	15	0	SRAM write enable. This bit is used in conjunction with the RAM selection register. When this bit is set to "1", the RAM specified in the selection register undergoes a test write cycle. Data residing in the indirect data register R31:2 is loaded into all seventeen 800 x 16 SRAMS simultaneously. Resetting the bit to "0", thereafter causes a test READ cycle to be performed and the data is read from the SRAMs and loaded back into R31:2. This bit is write_enable when 1; read_enable when 0. The READ and WRITE cycles occur when R31:2 is accessed.
30 0x1E	15:0	0	JPEG Indirect Access Control Register
	10:0	0	Indirect access address register: This 11-bit register contains the address of the register or memory to be accessed indirectly.
	12:11	0	Unused.
	13	0	Enable two-wire serial interface burst: When this bit is set to "1", the two-wire serial interface decoder operates in burst mode for the indirect data register (READ burst and WRITE burst). The longest burst supported is 16 (128 READ or WRITE cycles).
	14	0	Enable indirect writing: When this bit is set to "1", data from the indirect data register is written to the Indirect address location specified by [10:0] of this register except when auto-increment is set. Reading the same address location when this bit is reset to "0."
	15	0	Address auto-increment: When this bit is set to "1", the value in the indirect access address register is automatically incremented after every READ or WRITE, to the JPEG indirect access data register. This feature is used to emulate a burst access to memory or registers being accessed indirectly.
31 0x1F	15:0	0	JPEG Indirect Access Data Register
	Writing to, and reading from this register, is equivalent to performing these operations on registers or memory being indirectly accessed. When address auto-increment bit is set in JPEG indirect access control register, multiple writes or reads from this register affect a burst data transfer. Data is written to or read from indirect registers (when TestSRAM REG 0x0[1] is set to "0"), or the 800 x 16 SRAMs (when TestSRAM is set to "1").		
32-46 0x20- 0x2E	15:0	0	JPEG Indirect Access Data Register Same as R31:2 when doing two-wire serial interface burst.

Table 7: IFP Registers, Page 2 (continued)

Reg #	Bits	Default	Name
128 0x80	10:0	0x0160	Lens Correction Control
	0	0	Sign for the K*F(x)*F(y). If 0, then K is positive; if 1, then K is negative.
	1	0	When this bit is set to "1", all the second X derivatives are doubled.
	2	0	When this bit is set to "1", all the second Y derivatives are doubled.
	3:5	100	Divisor for the first derivative, X direction. 000-/1, 001-/2, 010-/4, ... 111-/128. Also applies to second derivative.
	6:8	101	Divisor for the first derivative, Y direction. 000-/1, 001-/2, 010-/4, ... 111-/128. Also applies to second derivative.
	9	0	Shift column, LC specific.
	10	0	Shift row, LC specific.
	This register controls behavior of lens correction.		
129 0x81	15:0	0x6432	Zone Boundaries X1 and X2
	7:0		X2 boundary (/4) position.
	15:8		X1 boundary (/4) position.
	Definition of X1 and X2 boundaries		
130 0x82	15:0	0x3296	Zone Boundaries X0 and X3
	7:0		X0 boundary (/4) position.
	15:8		X3 boundary (/4) position.
	Definition of X0 and X3 boundaries.		
131 0x83	15:0	0x9664	Zone Boundaries X4 and X5
	7:0		X4 boundary (/4) position.
	15:8		X5 boundary (/4) position.
	Definition of X4 and X5 boundaries.		
132 0x84	15:0	0x5028	Zone Boundaries Y1 and Y2
	7:0		Y2 boundary (/4) position.
	15:8		Y1 boundary (/4) position.
	Definition of Y1 and Y2 boundaries.		
133 0x85	15:0	0x2878	Zone Boundaries Y0 and Y3
	7:0		Y0 boundary (/4) position.
	15:8		Y3 boundary (/4) position.
	Definition of Y0 and Y3 boundaries.		
134 0x86	15:0	0x7850	Zone Boundaries Y4 and Y5
	7:0		Y4 boundary (/4) position.
	15:8		Y5 boundary (/4) position.
	Definition of Y4 and Y5 boundaries.		
135 0x87	15:0	0x0000	Center Offset
	7:0		Offset of LC center in respect to geometrical center. X coordinate(/4).
	15:8		Offset of LC center in respect to geometrical center. Ycoordinate(/4).
	Offset of the central point for the lens correction (relative to the center of imaging array).		
136 0x88	11:0	0x015E	F(x) for Red Color at the First Pixel of the Array
137 0x89	11:0	0x0143	F(x) for Green Color at the First Pixel of the Array

Table 7: IFP Registers, Page 2 (continued)

Reg #	Bits	Default	Name
138 0x8A	11:0	0x0127	F(x) for Blue Color at the First Pixel of the Array
139 0x8B	11:0	0x012C	F(y) for Red Color at the First Pixel of the Array
140 0x8C	11:0	0x0103	F(y) for Green Color at the First Pixel of the Array
141 0x8D	11:0	0x00FD	F(y) for Blue Color at the First Pixel of the Array
142 0x8E	11:0	0x0CEF	dF/dx for Red Color at the First Pixel of the Array
143 0x8F	11:0	0x0D38	dF/dx for Green Color at the First Pixel of the Array
144 0x90	11:0	0x0D92	dF/dx for Blue Color at the First Pixel of the Array
145 0x91	11:0	0x0C18	dF/dy for Red Color at the First Pixel of the Array
146 0x92	11:0	0x0CF1	dF/dy for Green Color at the First Pixel of the Array
147 0x93	11:0	0x0D05	dF/dy for Blue Color at the First Pixel of the Array
148 0x94	15:0	0x0B03	Second Derivative for Zone 0 Red Color
	7:0		d2F/dx2 for zone 0 red color.
	15:8		d2F/dy2 for zone 0 red color.
	Second derivative for red color in zone 0.		
149 0x95	15:0	0x0000	Second Derivative for Zone 0 Green Color
	7:0		d2F/dx2 for zone 0 green color.
	15:8		d2F/dy2 for zone 0 green color.
	Second derivative for green color in zone 0.		
150 0x96	15:0	0x0100	Second Derivative for Zone 0 Blue Color
	7:0		d2F/dx2 for zone 0 blue color.
	15:8		d2F/dy2 for zone 0 blue color.
	Second derivative for green color in zone 0.		
151 0x97	15:0	0x2534	Second Derivative for Zone 1 Red Color
	7:0		d2F/dx2 for zone 1 red color.
	15:8		d2F/dy2 for zone 1 red color.
	Second derivative for red color in zone 1.		
152 0x98	15:0	0x1C33	Second Derivative for Zone 1 Green Color
	7:0		d2F/dx2 for zone 1 green color.
	15:8		d2F/dy2 for zone 1 green color.
	Second derivative for green color in zone 1.		
153 0x99	15:0	0x1A2E	Second Derivative for Zone 1 Blue Color
	7:0		d2F/dx2 for zone 1 blue color.
	15:8		d2F/dy2 for zone 1 blue color.
	Second derivative for green color in zone 1.		

Table 7: IFP Registers, Page 2 (continued)

Reg #	Bits	Default	Name
154 0x9A	15:0	0x2A3A	Second Derivative for Zone 2 Red color
	7:0		d2F/dx2 for zone 2 red color.
	15:8		d2F/dy2 for zone 2 red color.
	Second derivative for red color in zone 2.		
155 0x9B	15:0	0x252D	Second Derivative for Zone 2 Green Color
	7:0		d2F/dx2 for zone 2 green color.
	15:8		d2F/dy2 for zone 2 green color.
	Second derivative for green color in zone 2.		
156 0x9C	15:0	0x2823	Second Derivative for Zone 2 Blue Color
	7:0		d2F/dx2 for zone 2 blue color.
	15:8		d2F/dy2 for zone 2 blue color.
	Second derivative for green color in zone 2.		
157 0x9D	15:0	0x0F14	Second Derivative for Zone 3 Red Color
	7:0		d2F/dx2 for zone 3 red color.
	15:8		d2F/dy2 for zone 3 red color.
	Second derivative for red color in zone 3.		
158 0x9E	15:0	0x0D20	Second Derivative for Zone 3 Green Color
	7:0		d2F/dx2 for zone 3 green color.
	15:8		d2F/dy2 for zone 3 green color.
	Second derivative for green color in zone 3.		
159 0x9F	15:0	0x0421	Second Derivative for Zone 3 Blue Color
	7:0		d2F/dx2 for zone 3 blue color.
	15:8		d2F/dy2 for zone 3 blue color.
	Second derivative for green color in zone 3.		
160 0xA0	15:0	0x0D2A	Second Derivative for Zone 4 Red Color
	7:0		d2F/dx2 for zone 4 red color.
	15:8		d2F/dy2 for zone 4 red color.
	Second derivative for red color in zone 4.		
161 0xA1	15:0	0x1017	Second Derivative for Zone 4 Green Color
	7:0		d2F/dx2 for zone 4 green color.
	15:8		d2F/dy2 for zone 4 green color.
	Second derivative for green color in zone 4.		
162 0xA2	15:0	0x1617	Second Derivative for Zone 4 Blue Color
	7:0		d2F/dx2 for zone 4 blue color.
	15:8		d2F/dy2 for zone 4 blue color.
	Second derivative for green color in zone 4.		
163 0xA3	15:0	0x1642	Second Derivative for Zone 5 Red Color
	7:0		d2F/dx2 for zone 5 red color.
	15:8		d2F/dy2 for zone 5 red color.
	Second derivative for red color in zone 5.		

Table 7: IFP Registers, Page 2 (continued)

Reg #	Bits	Default	Name
164 0xA4	15:0	0x1448	Second Derivative for Zone 5 Green Color
	7:0		d2F/dx2 for zone 5 green color.
	15:8		d2F/dy2 for zone 5 green color.
	Second derivative for green color in zone 5.		
165 0xA5	15:0	0x0F4E	Second Derivative for Zone 5 Blue Color
	7:0		d2F/dx2 for zone 5 blue color.
	15:8		d2F/dy2 for zone 5 blue color.
	Second derivative for green color in zone 5.		
166 0xA6	15:0	0x1F27	Second Derivative for Zone 6 Red Color
	7:0		d2F/dx2 for zone 6 red color.
	15:8		d2F/dy2 for zone 6 red color.
	Second derivative for green color in zone 6.		
167 0xA7	15:0	0x1A12	Second Derivative for Zone 6 Green Color
	7:0		d2F/dx2 for zone 6 green color.
	15:8		d2F/dy2 for zone 6 green color.
	Second derivative for green color in zone 6.		
168 0xA8	15:0	0x1E16	Second Derivative for Zone 6 Blue Color
	7:0		d2F/dx2 for zone 6 blue color.
	15:8		d2F/dy2 for zone 6 blue color.
	Second derivative for blue color in zone 6.		
169 0xA9	15:0	0x16C7	Second Derivative for Zone 7 Red Color
	7:0		d2F/dx2 for zone 7 red color.
	15:8		d2F/dy2 for zone 7 red color.
	Second derivative for red color in zone 7.		
170 0xAA	15:0	0x31C5	Second Derivative for Zone 7 Green Color
	7:0		d2F/dx2 for zone 7 green color.
	15:8		d2F/dy2 for zone 7 green color.
	Second derivative for green color in zone 7.		
171 0xAB	15:0	0x2AB6	Second Derivative for Zone 7 Blue Color
	7:0		d2F/dx2 for zone 7 blue color.
	15:8		d2F/dy2 for zone 7 blue color.
	Second derivative for blue color in zone 7.		
172 0xAC	15:0	0x0000	X2 Factors
	7:0		X2 factors for X zones. If 1 value of the value of d2F/dx2 is multiplied by 2.
	15:8		X2 factors for Y zones. If 1 value of the value of d2F/dy2 is multiplied by 2.
173 0xAD	7:0	0x0002	Global Offset of F(x,y) Function Signed 8 bit number. Value of 32 corresponds to gain of 1. Works as digital gain
174 0xAE	15:0	0x018E	K Factor in K F(x) F(y) This factor can increase lens compensation in the corners to up to 2 times.

Table 7: IFP Registers, Page 2 (continued)

Reg #	Bits	Default	Name
192 0xC0	15:0	0	Boundaries of First AE Measurement Window (Top/Left)
	7:0		Left window boundary.
	15:8		Top window boundary.
	This register specifies top and left boundaries of the first from 16 (left-top) window used by AE measurement engine. The values programmed in the registers are desired boundaries divided by 8.		
193 0xC1	15:0	0x3C50	Boundaries of First AE Measurement Window (Height/Width)
	7:0		Window width.
	15:8		Window height.
	This register specifies height and width of the first from 16 window used by AE measurement engine. The values programmed in the registers are desired boundaries divided by 2.		
194 0xC2	15:0	0x4B00	AE Measurement Window Size (LSW)
	This register number of pixels in the window used by AE measurement engine.		
195 0xC3	2:0	6	AE Measurement Enable
	0	0	MS bit of the AE measurement window size.
	1	1	AE statistic enable.
	2	1	Reserved.
	This register specifies number of pixels in the window used by AE measurement engine.		
196 0xC4	15:0	R/O	Average Luminance in AE Windows W12 and W11
	7:0		Average Y in W11 (top left).
	15:8		Average Y in W12.
197 0xC5	15:0	R/O	Average Luminance in AE Windows W14 and W13
	7:0		Average Y in W13.
	15:8		Average Y in W14 (top right).
198 0xC6	15:0	R/O	Average Luminance in AE Windows W22 and W21
	7:0		Average Y in W21.
	15:8		Average Y in W21.
199 0xC7	15:0	R/O	Average Luminance in AE Windows W24 and W23
	7:0		Average Y in W23.
	15:8		Average Y in W24.
200 0xC8	15:0	R/O	Average Luminance in AE Windows W32 and W31
	7:0		Average Y in W31.
	15:8		Average Y in W32.
201 0xC9	15:0	R/O	Average Luminance in AE Windows W34 and W33
	7:0		Average Y in W33.
	15:8		Average Y in W34.
202 0xCA	15:0	R/O	Average Luminance in AE Windows W42 and W41
	7:0		Average Y in W41 (bottom left).
	15:8		Average Y in W43.
203 0xCB	15:0	R/O	Average Luminance in AE Windows W44 and W43
	7:0		Average Y in W43.
	15:8		Average Y in W44 (bottom right).

Table 7: IFP Registers, Page 2 (continued)

Reg #	Bits	Default	Name
210 0xD2	9:0	0x0194	Saturation and Color Kill
	2:0	4	Saturation; 100 = 100%. Scales linearly with value.
	5:3	2	Color kill gain. Determines rate of gain decrease above threshold. If pixel value is above threshold the difference is multiplied by 2^ value-1 (for 0 factor is “0”) and subtracted from the base gain.
	8:6	6	Color kill threshold. Color kill affects only pixels with value larger then 128*value.
	9	0	Reserved.
211 0xD3	15:0	0	Histogram Window Lower Boundaries
	7:0		X0/8.
	15:8		Y0/8.
212 0xD4	15:0	0x95C7	Histogram Window Upper Boundaries
	7:0		X1/8.
	15:8		Y1/8.
213 0xD5	10:0	0x030A	First Set of Bin Definitions
	7:0		Offset for bin 0, divided by 4 on 10-bit scale.
	10:8		Bin width, 0-4LSB,1-8LSB,2-16LSB,...7-512LSB on a 10-bit scale.
214 0xD6	10:0	0x030A	Second Set of Bin Definitions
	7:0		Offset for bin 0, divided by 4 on 10-bit scale.
	10:8		Bin width, 0-4LSB,1-8LSB,2-16LSB,...7-512LSB on a 10-bit scale.
215 0xD7	13:0	0x000A	Histogram Window Size
	12:0	1	Number of pixels in the window used by the histogram, set to width*height.
	13	0	Select a bin set to read. 0: bin set 1 1: bin set 2
216 0xD8	15:0	0	Pixel Counts for Bin 0 and Bin 1
	7:0		Pixel count for bin 0 (lowest values) divided window size, R215:2.
	15:8		Pixel count for bin 1 divided by window size, R215:2.
	Histogram module uses luma after interpolation. No color correction or gamma is applied. Digital gains and first black level are applied. This register may not be read out when image portion containing histogram window is being output. There are two bin sets which could be read through registers 216 and 217, based on value of bit 13 in reg 215[2].		
217 0xD9	15:0	0x0002	Pixel Counts for Bin 2 and Bin 3
	7:0		Pixel count for bin 2 divided by G factor.
	15:8		Pixel count for bin 3 (highest values) divided by G factor.
240 0xF0	2:0	2	Page Register 0: sensor core 1: IFP page 1 2: IFP page 2
241 0xF1	15:0	0	Bytewise Address Special address to perform 16-bit reads and writes to the sensor in 8-bit chunks. See “8-Bit Write Sequence” on page 128.

JPEG Indirect Registers

Table 8: JPEG Indirect Registers (See Registers 30 and 31, Page 2)

Reg #	Bits	Default	Name
1 0x01	2:0	0	JPEG Core Register 1
	1:0	0	NCol: Number of color components–1.
	2	0	Re: When set, enables restart marker insertion into JPEG byte stream.
3 0x03	15:0	0	JPEG Core Register 3 – NRST
	Re-start interval - 1. Valid only when Re in Core Register 1 is set. This register defines the number of MCUs between Restart Markers.		
4 0x04	7:0	0	JPEG Core Register 4
	0	0	HD0: DC Huffman table pointer for color component 0.
	1	0	HA0: AC Huffman table pointer for color component 0.
	3:2	0	QT0: Quantization table pointer for color component 0.
	7:4	0	NBlock0: Number of 8 x 8 blocks—1 for color component 0 in MCU.
5 0x05	7:0	0	JPEG Core Register 5
	0	0	HD1: DC Huffman table pointer for color component 1.
	1	0	HA1: AC Huffman table pointer for color component 1.
	3:2	0	QT1: Quantization table pointer for color component 1.
	7:4	0	NBlock1: Number of 8 x 8 blocks—1 for color component 1 in MCU.
6 0x06	7:0	0	JPEG Core Register 6
	0	0	HD2: DC Huffman table pointer for color component 2.
	1	0	HA2: AC Huffman table pointer for color component 2.
	3:2	0	QT2: Quantization table pointer for color component 2.
	7:4	0	NBlock2: Number of 8 x 8 blocks—1 for color component 2 in MCU.
7 0x07	7:0	0	JPEG Core Register 7
	0	0	HD3: DC Huffman table pointer for color component 3.
	1	0	HA3: AC Huffman table pointer for color component 3.
	3:2	0	QT3: Quantization table pointer for color component 3.
	7:4	0	NBlock3: Number of 8 x 8 blocks—1 for color component 3 in MCU.
8 0x08	15:0	0	JPEG Core Register8 – NMCU LSB
	Low-order word of NMCU (NMCU = number of MCUs contained in the image to be encoded minus 1).		
9 0x09	9:0	0	JPEG Core Register9 – NMCU_MSB
	High-order word of NMCU (NMCU = number of MCUs contained in the image to be encoded minus 1).		
128-511 0x080 - 0x1FF	13:0	0	JPEG Quantization Memory
	0 - 63: Quantization table 0 64 - 127: Quantization table 1 128 - 191: Quantization table 2 192 - 255: Quantization table 3 256 - 319: Quantization table 4 320 - 383: Quantization table 5		
512 - 895 0x200 - 0x37F	11:0	0	JPEG Huffman Memory
	0 - 175: AC Huffman table 0 176 - 351: AC Huffman table 1 352 - 367: DC Huffman table 0 368 - 383: DC Huffman table 1		

Table 8: JPEG Indirect Registers (See Registers 30 and 31, Page 2) (continued)

Reg #	Bits	Default	Name
1024 - 1407 0x400 - 0x43F	13:0	0	JPEG DCT Memory
	64 x 14 dual-port RAM is accessible to host and microcontroller indirectly for testing purposes.		
1408 - 1471 0x440 - 0x47F	13:0	0	JPEG Zigzag Memory 0
	64 x 14 dual-port RAM is accessible to host and microcontroller indirectly for testing purposes.		
1472 - 1535 0x480 - 0x4BF	13:0	0	JPEG Zigzag Memory 1
	64 x 14 dual-port RAM is accessible to host and microcontroller indirectly for testing purposes.		

Firmware Driver Variables

Table 9: Drivers IDs

ID	VARNAME	Description
0		Reserved
1	seq	Sequencer
2	ae	Auto exposure
3	awb	Auto white balance
4	fd	Flicker detection
5		Reserved
6		Reserved
7	mode	Mode
8		Reserved
9	jpeg	JPEG
11		Reserved
12–15		Reserved
Driver Extensions		
16		Reserved
17		Sequencer extension
18		Auto exposure extension
19		Auto white balance extension
20		Flicker detection extension
21		Reserved
22		Reserved
23		Mode extension
24		Reserved
25		JPEG extension
26		Reserved
27–31		Reserved

Sequencer

Table 10: Driver Variables–Sequencer Driver (ID = 1)

Offs	Name	Type	Default	RW	Description
0	vmt	void*	61547	RW	Reserved
2	mode	uchar	0x0F	RW	Set of 1-bit switches enabling various drivers and outdoor white balance option. Writing 1 to a particular switch enables the corresponding driver or option. Bit 0: Auto exposure driver (ID = 2) Bit 1: Reserved Bit 2: Reserved Bit 3: Reserved Bit 4: Reserved Bit 5: Option to force outdoor white balance settings if exposure value exceeds sharedParams.outdoorTH threshold. Bits 6:7: Reserved Bit 7: TBD
3	cmd	uchar	0	RW	Command or program to execute 0: Run 1: Do Preview 2: Do Capture 3: Do Standby 4: Do lock 5: Refresh 6: Refresh mode Writing a positive value to this variable commands the sequencer to execute the corresponding program. The sequencer resets cmd to 0, executes the program, and then resumes running in its current state.
4	state	uchar	3	R	Current state of sequencer 0: Initialize 1: Mode Change to Preview 2: Enter Preview 3: Preview 4: Leave Preview 5: Mode Change to Capture 6: Enter Capture 7: Capture 8: Leave Capture 9: Standby
5	stepMode	uchar	0	RW	Step-by-step mode for sequencer Bit 0: Step mode On/Off (1 = On) Bit 1: 1 forces the sequencer to do next step
7	sharedParams.aeContBuff	uchar	8	RW	Weighting factor determining to what extent AE driver averages luma over time in continuous AE mode. Setting of 32 disables luma averaging—only current luma values are used. Lower settings enable luma averaging.
8	sharedParams.aeContStep	uchar	2	RW	Number of steps in which AE driver is expected to reach target brightness in continuous AE mode.
9	sharedParams.aeFastBuff	uchar	32	RW	Weighting factor determining to what extent AE driver averages luma over time in fast mode. Setting of 32 disables luma averaging—only current luma values are used. Lower settings enable luma averaging.

Table 10: Driver Variables–Sequencer Driver (ID = 1) (continued)

Offs	Name	Type	Default	RW	Description
10	sharedParams.aeFastStep	uchar	1	RW	Number of steps in which AE driver is expected to reach target brightness in fast AE mode.
11	sharedParams.awbContBuff	uchar	8	RW	Weighting factor determining to what extent AWB driver averages luma over time in continuous AWB mode. Setting of 32 disables luma averaging—only current luma values are used. Lower settings enable luma averaging.
12	sharedParams.awbContStep	uchar	2	RW	Number of steps in which AWB driver is expected to reach target color balance in continuous AWB mode.
13	sharedParams.awbFastBuff	uchar	32	RW	Weighting factor determining to what extent AWB driver averages luma over time in fast mode. Setting of 32 disables luma averaging—only current luma values are used. Lower settings enable luma averaging.
14	sharedParams.awbFastStep	uchar	1	RW	Number of steps in which AWB driver is expected to reach target color balance in fast AWB mode.
18	sharedParams.totalMaxFrames	uchar	30	RW	Number of frames after which every driver must time out.
20	sharedParams.outdoorTH	uchar	10	RW	Exposure value (EV) threshold. If current EV is above this threshold and bit 5 of seq.mode equals 1, AWB driver is forced to choose color correction settings suitable for daylight color temperature of 6500 K.
21	sharedParams.LLmode	uchar	0x60	RW	Bit 0: Change interpolation threshold Bit 1: Reduce color saturation Bit 2: Reduce aperture correction Bit 3: Increase aperture correction threshold Bit 4: Enable Y filter
22	sharedParams.LLvirtGain1	uchar	0x51	RW	Minimum LL virtual gain. Defined as (ae.VirtGain/2 + ae.DGainAE1/16 + ae.DGainAE2/4).
23	sharedParams.LLvirtGain2	uchar	0x5A	RW	Maximum LL virtual gain. Min. and max. LL virtual gains define when low-light parameters start and stop changing.
24	sharedParams.LLSat1	uchar	128	RW	Maximum color saturation for color correction matrix (128 means 100%).
25	sharedParams.LLSat2	uchar	0	RW	Minimum color saturation
26	sharedParams.LLInterpThresh1	uchar	16	RW	Minimum threshold for interpolation
27	sharedParams.LLInterpThresh2	uchar	64	RW	Maximum threshold for interpolation
28	sharedParams.LLApCorr1	uchar	2	RW	Maximum aperture correction
29	sharedParams.LLApCorr2	uchar	0	RW	Minimum aperture correction
30	sharedParams.LLApThresh1	uchar	8	RW	Minimum aperture correction threshold
31	sharedParams.LLApThresh2	uchar	64	RW	Maximum aperture correction threshold
32	captureParams.mode	uchar	0x00	RW	Capture mode Bit 0: Reserved Bit 1: capture video Bit 2: reserved Bit 3: Reserved Bit 4: AE On (video only) Bit 5: AWB On (video only) Bit 6: HG On (video only)
33	captureParams.numFrames	uchar	3	RW	Number of frames captured in still capture mode

Table 10: Driver Variables–Sequencer Driver (ID = 1) (continued)

Offs	Name	Type	Default	RW	Description
34	previewParams[0].ae	uchar	1	RW	Auto exposure configuration in PreviewEnter state 0: Off 1: Fast settling 2: Manual 3: Continuous 4: Fast settling + metering
35	previewParams[0].fd	uchar	0	RW	Flicker detection configuration in PreviewEnter state 0: Off 1: Manual 2: Continuous
36	previewParams[0].awb	uchar	1	RW	Auto white balance configuration in PreviewEnter state 0: Off 1: Fast settling 2: Manual 3: Continuous
38	previewParams[0].hg	uchar	1	RW	Histogram configuration in PreviewEnter state 0: Off 1: Fast 2: Manual 3: Continuous
40	previewParams[0].skipframe	uchar	0x00	RW	Bit 7: 1 means turn off frame enable Bit 6: 1 means skip state Bit 5: 1 means skip first frame when LED On
41	previewParams[1].ae	uchar	3	RW	Auto exposure configuration in Preview state 0: Off 1: Fast settling 2: Manual 3: Continuous 4: Fast settling + metering
42	previewParams[1].fd	uchar	2	RW	Flicker detection configuration in Preview state 0: Off 1: Manual 2: Continuous
43	previewParams[1].awb	uchar	3	RW	Auto white balance configuration in Preview state 0: Off 1: Fast settling 2: Manual 3: Continuous
45	previewParams[1].hg	uchar	3	RW	Histogram configuration in Preview state 0: Off 1: Fast 2: Manual 3: Continuous
47	previewParams[1].skipframe	uchar	0x00	RW	Bit 7: 1 means turn off frame enable Bit 6: 1 means skip state Bit 5: 1 means skip first frame when LED On

Table 10: Driver Variables–Sequencer Driver (ID = 1) (continued)

Offs	Name	Type	Default	RW	Description
48	previewParams[2].ae	uchar	4	RW	Auto exposure configuration in PreviewLeave state 0: Off 1: Fast settling 2: Manual 3: Continuous 4: Fast settling + metering
49	previewParams[2].fd	uchar	0	RW	Flicker detection configuration in PreviewLeave state 0: Off 1: Manual 2: Continuous
50	previewParams[2].awb	uchar	1	RW	Auto white balance configuration in PreviewLeave state 0: Off 1: Fast settling 2: Manual 3: Continuous
52	previewParams[2].hg	uchar	1	RW	Histogram configuration in PreviewLeave state 0: Off 1: Fast 2: Manual 3: Continuous
54	previewParams[2].skipframe	uchar	0x00	RW	Bit 7: 1 means turn off frame enable Bit 6: 1 means skip state Bit 5: 1 means skip first frame when LED On
55	previewParams[3].ae	uchar	0	RW	Auto exposure configuration in CaptureEnter state 0: Off 1: Fast settling 2: Manual 3: Continuous 4: Fast settling + metering
56	previewParams[3].fd	uchar	0	RW	Flicker detection configuration in CaptureEnter state 0: Off 1: Manual 2: Continuous
57	previewParams[3].awb	uchar	0	RW	Auto white balance configuration in CaptureEnter state 0: Off 1: Fast settling 2: Manual 3: Continuous
59	previewParams[3].hg	uchar	0	RW	Histogram configuration in CaptureEnter state 0: Off 1: Fast 2: Manual 3: Continuous
61	previewParams[3].skipframe	uchar	0x00	RW	Bit 7: 1 means turn off frame enable Bit 6: 1 means skip state Bit 5: 1 means skip first frame when LED On

Auto Exposure

Table 11: Driver Variables–Auto Exposure Driver (ID = 2)

Offs	Name	Type	Default	RW	Description
0	vmt	void*	59518	RW	Reserved
2	windowPos	uchar	0	RW	Position of upper left corner of first AE window Bits [3:0]: X0 in units of 1/16 of frame width Bits [7:4]: Y0 in units 1/16 of frame height New position written to this variable takes effect only after REFRESH_MODE command is given to the sequencer (seq.cmd is set to 6).
3	windowSize	uchar	0xEF	RW	Dimensions of 4 x 4 array of AE windows Bits [3:0]: width (in units of 1/16 of frame width) decremented by 1 Bits [7:4]: height (in units of 1/16 of frame height) decremented by 1 New dimensions written to this variable take effect only after REFRESH_MODE command is given to the sequencer (seq.cmd is set to 6).
4	wakeUpLine	uint		R	Number of image row at which MCU wakes up to do AE calculations. This number is automatically adjusted after each change of the position and dimensions of the AE windows.
6	Target	uchar	60	RW	Target brightness
7	Gate	uchar	10	RW	AE sensitivity
8	SkipFrames	uchar	0	RW	Frequency of AE updates
9	JumpDivisor	uchar	2	RW	Factor reducing exposure jumps (1 = fastest jumps)
10	lumaBufferSpeed	uchar	8	RW	Speed of buffering (32 = fastest, 1 = slowest)
11	maxR12	uint	1200	RW	Maximum value of R12:0 (shutter delay)
13	minIndex	uchar	0	RW	Minimum allowed zone number (that is minimum integration time)
14	maxIndex	uchar	24	RW	Maximum allowed zone number (that is maximum integration time)
15	minVirtGain	uchar	16	RW	Minimum allowed virtual gain
16	maxVirtGain	uchar	128	RW	Maximum allowed virtual gain
17	maxADChi	uchar	13	RW	Maximum ADC Vref_hi setting
18	minADChi	uchar	11	RW	Minimum ADC Vref_hi setting
19	minADClo	uchar	7	RW	Minimum ADC Vref_lo setting
20	maxDGainAE1	uint	128	RW	Maximum digital gain pre-LC
22	maxDGainAE2	uchar	32	RW	Maximum digital gain post-LC
23	IndexTH23	uchar	8	RW	Zone number to start gain increase in low-light. Sets frame rate at normal illumination.
24	maxGain23	uchar	120	RW	Maximum gain to increase in low-light before dropping frame rate. Sets frame rate at low-light.
25	weights	uchar	0xFF	RW	Bits [7:4]: background weight Bits [3:0]: center zone weight

Table 11: Driver Variables–Auto Exposure Driver (ID = 2) (continued)

Offs	Name	Type	Default	RW	Description
26	status	uchar	192	RW	AE status Bit 0: 1 if AE reached the limit Bit 1: 1 if R9 is changed (need to skip frame) Bit 2: ready Bit 3: do metering mode Bit 4: metering mode is done Bit 6: On/Off overexpose fit Bit 7: On/Off overexpose compensation
27	CurrentY	uchar		R	Last measured luminance
28	R12	uint		RW	Current shutter delay value
30	Index	uchar		R	Current zone (integration time)
31	VirtGain	uchar		RW	Current virtual gain
32	ADC_hi	uchar		R	Current ADC VREF_HI value
33	ADC_lo	uchar		R	Current ADC VREF_LO value
34	DGainAE1	uint		RW	Current digital gain pre-LC
36	DGainAE2	uchar		RW	Current digital gain post-LC
37	R9	uint		RW	Current R9:0 value (integration time)
39	R65	uchar		RW	Current ADC VREF values
40	rowTime	uint	531	RW	Row time divided by 4 (in master clock periods)
42	gainR12	uchar	128	R	Gain compensating too low maxR12 (128 = unit gain)
43	SkipFrames_cnt	uchar		R	Counter of skipped frames
44	BufferedLuma	uint		R	Current time-buffered measured luma
46	dirAE_prev	uchar		R	Previous state of AE
47	R9_step	uint	157	RW	Integration time of one zone
50	maxADClo	uchar		R	Maximum ADC Vref_lo setting
51	physGainR	uint		R	Physical (analog) R gain
53	physGainG	uint		R	Physical (analog) G gain
55	physGainB	uint		R	Physical (analog) B gain
82	mmEVZone1	uchar	16	RW	EV threshold for scenes containing the sun
83	mmEVZone2	uchar	12	RW	EV threshold for bright outdoor scenes
84	mmEVZone3	uchar	8	RW	EV threshold for indoor or outdoor twilight scenes
85	mmEVZone4	uchar	2	RW	EV threshold for night scenes
86	mmShiftEV	uchar	13	RW	Exposure Value shift. Adjust this value for given lens.
87	numOE	uchar		R	Number of overexposed zones

Auto White Balance

Table 12: Driver Variables–Auto White Balance (ID = 3)

Offs	Name	Type	Default	RW	Description
0	vmt	void*	59731	RW	Reserved
2	windowPos	uchar	0	RW	Position of upper left corner of AWB window Bits [3:0]: X0 in units of 1/16 of frame width Bits [7:4]: Y0 in units 1/16 of frame height New position written to this variable takes effect only after REFRESH_MODE command is given to the sequencer (seq.cmd is set to 6).
3	windowSize	uchar	0xEF	RW	Dimensions of AWB window Bits [3:0]: width (in units of 1/16 of frame width) decremented by 1. Bits [7:4]: height (in units of 1/16 of frame height) decremented by 1. New dimensions written to this variable take effect only after REFRESH_MODE command is given to the sequencer (seq.cmd is set to 6).
4	wakeUpLine	uint		R	Number of image row at which MCU wakes up to perform AWB calculations. This number is automatically adjusted after each change of the position and dimensions of the AWB window.
6	ccmL[0]	int	0x0219	RW	Left color correction matrix element K11. Value is encoded in signed two's complement form; 256 means 1.0. New values written to awb.ccmL array take effect only after REFRESH command is given to the sequencer (seq.cmd is set to 5).
8	ccmL[1]	int	0xFEEC	RW	Left color correction matrix element K12
10	ccmL[2]	int	0xFFFF	RW	Left color correction matrix element K13
12	ccmL[3]	int	0xFF6A	RW	Left color correction matrix element K21
14	ccmL[4]	int	0x01D4	RW	Left color correction matrix element K22
16	ccmL[5]	int	0xFFC9	RW	Left color correction matrix element K23
18	ccmL[6]	int	0xFF71	RW	Left color correction matrix element K31
20	ccmL[7]	int	0xFDD3	RW	Left color correction matrix element K32
22	ccmL[8]	int	0x03ED	RW	Left color correction matrix element K33
24	ccmL[9]	int	0x0020	RW	Left color correction matrix red/green gain ratio. Value is interpreted as unsigned; 32 means 1.0.
26	ccmL[10]	int	0x0035	RW	Left color correction matrix blue/green gain ratio. Value is interpreted as unsigned; 32 means 1.0.
28	ccmRL[0]	int	0xFF92	RW	Delta color correction matrix element D11. Value is encoded in signed two's complement form; 256 means 1.0. New values written to awb.ccmRL array take effect only after REFRESH command is given to the sequencer (seq.cmd is set to 5).
30	ccmRL[1]	int	0x0098	RW	Delta color correction matrix element D12
32	ccmRL[2]	int	0xFFE1	RW	Delta color correction matrix element D13
34	ccmRL[3]	int	0x0014	RW	Delta color correction matrix element D21
36	ccmRL[4]	int	0xFFFC	RW	Delta color correction matrix element D22
38	ccmRL[5]	int	0xFFF2	RW	Delta color correction matrix element D23
40	ccmRL[6]	int	0x004E	RW	Delta color correction matrix element D31
42	ccmRL[7]	int	0x0148	RW	Delta color correction matrix element D32
44	ccmRL[8]	int	0xFE3F	RW	Delta color correction matrix element D33

Table 12: Driver Variables–Auto White Balance (ID = 3) (continued)

Offs	Name	Type	Default	RW	Description
46	ccmRL[9]	int	0x000A	RW	Delta color correction matrix red/green gain ratio. Value is interpreted as unsigned; 32 means 1.0.
48	ccmRL[10]	int	0xFFF1	RW	Delta color correction matrix blue/green gain ratio. Value is interpreted as unsigned; 32 means 1.0.
50	ccm[0]	int		RW	Current color correction matrix element C11. Value is stored in signed two's complement form; 256 means 1.0.
52	ccm[1]	int		RW	Current color correction matrix element C12
54	ccm[2]	int		RW	Current color correction matrix element C13
56	ccm[3]	int		RW	Current color correction matrix element C21
58	ccm[4]	int		RW	Current color correction matrix element C22
60	ccm[5]	int		RW	Current color correction matrix element C23
62	ccm[6]	int		RW	Current color correction matrix element C31
64	ccm[7]	int		RW	Current color correction matrix element C32
66	ccm[8]	int		RW	Current color correction matrix element C33
68	ccm[9]	int		RW	Current color correction matrix red/green gain ratio. Value is interpreted as unsigned; 32 means 1.0.
70	ccm[10]	int		RW	Current color correction matrix blue/green gain ratio. Value is interpreted as unsigned; 32 means 1.0.
72	GainBufferSpeed	uchar	8	RW	Speed of time-buffering (32 = fastest, 1 = slowest)
73	JumpDivisor	uchar	2	RW	Derating factor for white balance changes (1 = fastest changes)
74	GainMin	uchar	83	RW	Minimum allowed AWB digital gain (128 means 1.0)
75	GainMax	uchar	192	RW	Maximum allowed AWB digital gain (128 means 1.0)
76	GainR	uchar		R	Current R digital gain (128 means1.0)
77	GainG	uchar		R	Current G digital gain (128 means1.0)
78	GainB	uchar		R	Current B digital gain (128 means 1.0)
79	CCMpositionMin	uchar	0	RW	Leftmost position of color correction matrix (corresponding to incandescent light)
80	CCMpositionMax	uchar	127	RW	Rightmost position of color correction matrix (corresponding to daylight)
81	CCMposition	uchar	64	RW	Current position of color correction matrix (0 corresponds to incandescent light, 127 to daylight)
82	saturation	uchar	128	RW	Saturation of color correction matrix (128 means 100%)
83	mode	uchar	0	RW	Bit 0: 1 = AWB is in steady state Bit 1: 1 = limits are reached Bit 2: 1 = reserved Bit 3: 1 = debug Bit 4: 1 = no awb statistic Bit 5: 1 = force AWB DGains to unit; program value into awb.CCMPosition to manually set matrix position Bit 6: 1 = disable CCM normalization Bit 7: 1 = force outdoor settings, set by the sequencer
84	GainR_buf	uint	0x1000	RW	Time-buffered R gain (0x1000 means 1.0)
86	GainB_buf	uint	0x1000	RW	Time-buffered B gain (0x1000 means 1.0)
88	sumR	uchar		R	AWB statistics (normalized to an arbitrary number)
89	sumY	uchar		R	AWB statistics (normalized to an arbitrary number)
90	sumB	uchar		R	AWB statistics (normalized to an arbitrary number)

Table 12: Driver Variables–Auto White Balance (ID = 3) (continued)

Offs	Name	Type	Default	RW	Description
91	steadyBGainOutMin	uchar	115	RW	Blue gain min. threshold to start search. When the blue gain falls below this threshold, the AWB driver starts searching for new color correction matrix position. Threshold setting of 128 corresponds to gain of 1.0; higher setting means higher gain.
92	steadyBGainOutMax	uchar	140	RW	Blue gain max. threshold to start search. When the blue gain goes above this threshold, the AWB driver starts searching for new color correction matrix position. Threshold setting of 128 corresponds to gain of 1.0; higher setting means higher gain.
93	steadyBGainInMin	uchar	124	RW	Blue gain min. threshold to end search. When the blue gain is between awb.steadyBGainInMin and awb.steadyBGainInMax, the AWB driver maintains current color correction matrix position. Threshold setting of 128 corresponds to gain of 1.0; higher setting means higher gain.
94	steadyBGainInMax	uchar	131	RW	Blue gain max. threshold to end search. See awb.steadyBGainInMin above.
95	cntPxlTH	uint	100	RW	Reserved
97	TG_min0	uchar	226	RW	Reserved
98	TG_max0	uchar	246	RW	Reserved
99	XO		16	RW	CCM position threshold between Day and Incandescent normalization coefficients.
100	kR_L	uchar	170	RW	CCM red row normalization coefficient for left matrix position (128 - minimal number)
101	kG_L	uchar	150	RW	CCM green row normalization coefficient for left matrix position (128 - minimal number)
102	kB_L	uchar	128	RW	CCM blue row normalization coefficient for left matrix position (128 - minimal number)
103	kR_R	uchar	128	RW	CCM red row normalization coefficient for right matrix position (128 - minimal number)
104	kG_R	uchar	128	RW	CCM green row normalization coefficient for right matrix position (128 - minimal number)
105	kB_R	uchar	128	RW	CCM blue row normalization coefficient for right matrix position (128 - minimal number)

Flicker Detection

Table 13: Driver Variables–Flicker Detection Driver (ID = 4)

Offs	Name	Type	Default	RW	Description
0	vmt	void*	59884	RW	Reserved
2	windowPosH	uchar	29	RW	Width of flicker measurement window and position of its left boundary X0 Bits [3:0]: width (in units of 1/16 of frame width) decremented by 1 Bits [7:4]: X0 (in the same units as the width) At the beginning of each flicker measurement, the top boundary (Y0) of the flicker measurement window is set at row 64. Average luminance in the window is measured by the IFP measurement engine and stored in fd.Buffer[0]. Then Y0 is incremented by fd.windowHeight and the buffer index by 1. By repeating these steps 47 times, the entire fd.Buffer is filled with average luminance values from a vertical array of 48 equal-size windows.
3	windowHeight	uchar	4	RW	Bits [5:0]: flicker measurement window height in rows
4	mode	uchar	0	RW	Mode switches and indicators Bit 7: 1 enables manual mode, 0 disables it Bit 6: 0 selects 60Hz settings for manual mode, 1 selects 50Hz settings Bit 5: read-only, indicates current settings (0–60Hz, 1–50Hz) Bit 4: 1 enables debug mode, 0 disables it Bits [3:0]: read-only, reserved for auto FD
5	wakeUpLine	uint	64	R	Number of image row at which MCU wakes up to perform flicker detection. Changing the value of fd.wakeUpLine has no effect on the functioning of the FD driver, because it does not use this variable. It simply sets it to 64 at initialization, to indicate the top of the flicker measurement area. See fd.windowPosH above.
7	smooth_cnt	uchar	5	RW	Reserved
8	search_f1_50	uchar	30	RW	Lower limit of period range of interest in 50Hz flicker detection. If the FD driver is searching for 50Hz flicker and detects in a frame a flicker-like signal whose period in rows is between fl.search_f1_50 and fl.search_f2_50, it counts the frame as one containing flicker.
9	search_f2_50	uchar	32	RW	Upper limit of period range of interest in 50Hz flicker detection. See fd.search_f1_50 above.
10	search_f1_60	uchar	37	RW	Lower limit of period range of interest in 60Hz flicker detection. If the FD driver is searching for 60Hz flicker and detects in a frame a flicker-like signal whose period in rows is between fl.search_f1_60 and fl.search_f2_60, it counts the frame as one containing flicker.
11	search_f2_60	uchar	39	RW	Upper limit of period range of interest in 60Hz flicker detection. See fd.search_f1_60 above.
12	skipFrame	uchar	0	RW	Skip frame before subtracting two frames.
13	stat_min	uchar	3	RW	Flicker is considered detected if fd.stat_min out of fd.stat_max consecutive frames contain flicker (periodic signal of sought frequency).

Table 13: Driver Variables–Flicker Detection Driver (ID = 4) (continued)

Offs	Name	Type	Default	RW	Description
14	stat_max	uchar	5	RW	See fl.stat_min.
15	stat	uchar		R	Reserved
16	minAmplitude	uchar	5	RW	Reserved
17	R9_step60	uint	157	RW	Minimal shutter width step for 60Hz AC
19	R9_step50	uint	188	RW	Minimal shutter width step for 50Hz AC
21	Buffer[48]	uchar[48]		R	Reserved

Context

Table 14: Driver Variables–Mode/Context Driver (ID = 7)

Offs	Name	Type	ASSOC.H/W Reg	RW	Default	Description
0	VMT Pointer	uint	local	RW	59871	Pointer to virtual method table.
2	context	uchar	local	RW	0	Current context (0 = A, 1 = B).
3	output width_A	uint	R0x16:1	RW	800	Output size of final image for context A. Must be equal to or smaller than crop dimension.
5	output height_A	uint	R0x17:1	RW	600	Output size of final image for context A. Must be equal to or smaller than crop dimension.
7	output width_B	uint	R0x16:1	RW	1600	Output size of final image for context B. Must be equal to or smaller than crop dimension.
9	output height_B	uint	R0x17:1	RW	1200	Output size of final image for context B. Must be equal to or smaller than crop dimension.
11	mode_config	uint	R0xA:2	RW	0x10	Bit 4: JPG bypass: context A. Bit 5: JPG bypass: context B. Set to “0” to enable JPEG.
13	PLL_lock_delay	uint	local	RW	200	Delay between PLL enable and start of microcontroller operation (no. of clock cycles x 100).
15	s_row_start_A	uint	R0x01:0	RW	28	First sensor-readout row (context A shadow register).
17	s_col_start_A	uint	R0x02:0	RW	60	First sensor-readout column (context A shadow register).
19	s_row_height_A	uint	R0x03:0	RW	1200	Number of sensor-readout rows (context A shadow register).
21	s_col_width_A	uint	R0x04:0	RW	1600	Number of sensor-readout columns (context A shadow register).
23	s_ext_delay_A	uint	R0x0B:0	RW	0	Extra sensor delay per frame (context A shadow register).
25	s_row_speed_A	uint	R0x0A:0	RW	1	Row speed (context A shadow register).
27	s_row_start_B	uint	R0x01:0	RW	28	First sensor-readout row (context B shadow register).
29	s_col_start_B	uint	R0x02:0	RW	60	First sensor-readout column (context B shadow register).
31	s_row_height_B	uint	R0x03:0	RW	1200	Number of sensor-readout rows (context B shadow register).
33	s_col_width_B	uint	R0x04:0	RW	1600	Number of sensor-readout columns (context B shadow register).
35	s_ext_delay_B	uint	R0x0B:0	RW	1050	Extra sensor delay per frame (context B shadow register).
37	s_row_speed_B	uint	R0x0A:0	RW	1	Row speed (context B shadow register).
39	crop_X0_A	uint	R0x11:1	RW	0	Lower-x decimator zoom window (context A shadow register).
41	crop_X1_A	uint	R0x12:1	RW	800	Upper-x decimator zoom window (context A shadow register).
43	crop_Y0_A	uint	R0x13:1	RW	0	Lower-y decimator zoom window (context A shadow register).

Table 14: Driver Variables–Mode/Context Driver (ID = 7) (continued)

Offs	Name	Type	ASSOC.H/W Reg	RW	Default	Description
45	crop_Y1_A	uint	IR0x14:1	/W	600	Upper-y decimator zoom window (context A shadow register).
47	dec_ctrl_A	uint	R0x15:1	R/W	0	Decimator control register (context A shadow register).
53	crop_X0_B	uint	R0x11:1	RW	0	Lower-x decimator zoom window (context B shadow register).
55	crop_X1_B	uint	R0x12:1	RW	1600	Upper-x decimator zoom window (context B shadow register)
57	crop_Y0_B	uint	R0x13:1	RW	0	Lower-y decimator zoom window (context B shadow register).
59	crop_Y1_B	uint	R0x14:1	RW	1200	Upper-y decimator zoom window (context B shadow register).
61	dec_ctrl_B	uint	R0x15:1	RW	0	Decimator control register (context B shadow register).
67	gam_cont_A	uchar	local	RW	0x42	Gamma and contrast settings (context A shadow register) Gamma Setting: Bits 0:2: 0: gamma = 1.0 1: gamma = 0.56 2: gamma = 0.45 3: use user-defined gamma table Contrast Setting: Bits 4:6: 0: no contrast increase (slope = 100%) 1: some contrast increase (slope = 125%) 2: more contrast increase (slope = 150%) 3: most contrast increase (slope = 175%) 4: noise-reduction contrast
68	gam_cont_B	uchar	local	RW	0x42	Gamma and contrast settings (context B shadow register) Gamma Setting: Bits 0:2: 0: gamma = 1.0 1: gamma = 0.56 2: gamma = 0.45 3: use user-defined gamma table Contrast Setting: Bits 4:6: 0: no contrast increase (slope = 100%) 1: some contrast increase (slope = 125%) 2: more contrast increase (slope = 150%) 3: most contrast increase (slope = 175%) 4: noise-reduction contrast
69	gamma_table_A_0	uchar	R0xB2:1[7:0]	RW	0	User-defined gamma table values (context A shadow register).
70	gamma_table_A_1	uchar	R0xB2:1[15:8]	RW	39	User-defined gamma table values (context A shadow register).
71	gamma_table_A_2	uchar	R0xB3:1[7:0]	RW	53	User-defined gamma table values (context A shadow register).
72	gamma_table_A_3	uchar	R0xB3:1[15:8]	RW	72	User-defined gamma table values (context A shadow register).
73	gamma_table_A_4	uchar	R0xB4:1[7:0]	RW	99	User-defined gamma table values (context A shadow register).

Table 14: Driver Variables–Mode/Context Driver (ID = 7) (continued)

Offs	Name	Type	ASSOC.H/W Reg	RW	Default	Description
74	gamma_table_A_5	uchar	R0xB4:1[15:8]	RW	119	User-defined gamma table values (context A shadow register).
75	gamma_table_A_6	uchar	R0xB5:1[7:0]	RW	136	User-defined gamma table values (context A shadow register).
76	gamma_table_A_7	uchar	R0xB5:1[15:8]	RW	150	User-defined gamma table values (context A shadow register).
77	gamma_table_A_8	uchar	R0xB6:1[7:0]	RW	163	User-defined gamma table values (context A shadow register).
78	gamma_table_A_9	uchar	R0xB6:1[15:8]	RW	175	User-defined gamma table values (context A shadow register).
79	gamma_table_A_10	uchar	R0xB7:1[7:0]	RW	186	User-defined gamma table values (context A shadow register).
80	gamma_table_A_11	uchar	R0xB7:1[15:8]	RW	196	User-defined gamma table values (context A shadow register).
81	gamma_table_A_12	uchar	R0xB8:1[7:0]	RW	206	User-defined gamma table values (context A shadow register).
82	gamma_table_A_13	uchar	R0xB8:1[15:8]	RW	215	User-defined gamma table values (context A shadow register).
83	gamma_table_A_14	uchar	R0xB9:1[7:0]	RW	224	User-defined gamma table values (context A shadow register).
84	gamma_table_A_15	uchar	R0xB9:1[15:8]	RW	232	User-defined gamma table values (context A shadow register).
85	gamma_table_A_16	uchar	R0xBA:1[7:0]	RW	240	User-defined gamma table values (context A shadow register).
86	gamma_table_A_17	uchar	R0xBA:1[15:8]	RW	248	User-defined gamma table values (context A shadow register).
87	gamma_table_A_18	uchar	R0xBB:1[7:0]	RW	255	User-defined gamma table values (context A shadow register).
88	gamma_table_B_0	uchar	R0xB2:1[7:0]	RW	0	User-defined gamma table values (context B shadow register).
89	gamma_table_B_1	uchar	R0xB2:1[15:8]	RW	39	User-defined gamma table values (context B shadow register).
90	gamma_table_B_2	uchar	R0xB3:1[7:0]	RW	53	User-defined gamma table values (context B shadow register).
91	gamma_table_B_3	uchar	R0xB3:1[15:8]	RW	72	User-defined gamma table values (context B shadow register).
92	gamma_table_B_4	uchar	R0xB4:1[7:0]	RW	99	User-defined gamma table values (context B shadow register).
93	gamma_table_B_5	uchar	R0xB4:1[15:8]	RW	119	User-defined gamma table values (context B shadow register).
94	gamma_table_B_6	uchar	R0xB5:1[7:0]	RW	136	User-defined gamma table values (context B shadow register).
95	gamma_table_B_7	uchar	R0xB5:1[15:8]	RW	150	User-defined gamma table values (context B shadow register).
96	gamma_table_B_8	uchar	R0xB6:1[7:0]	RW	163	User-defined gamma table values (context B shadow register).
97	gamma_table_B_9	uchar	R0xB6:1[15:8]	RW	175	User-defined gamma table values (context B shadow register).

Table 14: Driver Variables–Mode/Context Driver (ID = 7) (continued)

Offs	Name	Type	ASSOC.H/W Reg	RW	Default	Description
98	gamma_table_B_10	uchar	R0xB7:1[7:0]	RW	186	User-defined gamma table values (context B shadow register).
99	gamma_table_B_11	uchar	R0xB7:1[15:8]	RW	196	User-defined gamma table values (context B shadow register).
100	gamma_table_B_12	uchar	R0xB8:1[7:0]	RW	206	User-defined gamma table values (context B shadow register).
101	gamma_table_B_13	uchar	R0xB8:1[15:8]	RW	215	User-defined gamma table values (context B shadow register).
102	gamma_table_B_14	uchar	R0xB9:1[7:0]	RW	224	User-defined gamma table values (context B shadow register).
103	gamma_table_B_15	uchar	R0xB9:1[15:8]	RW	232	User-defined gamma table values (context B shadow register).
104	gamma_table_B_16	uchar	R0xBA:1[7:0]	RW	240	User-defined gamma table values (context B shadow register).
105	gamma_table_B_17	uchar	R0xBA:1[15:8]	RW	248	User-defined gamma table values (context B shadow register).
106	gamma_table_B_18	uchar	R0xBB:1[7:0]	RW	255	User-defined gamma table values (context B shadow register).
107	FIFO_config0_A	uint	R0x0D:2	RW	0x0027	FIFO Buffer configuration 0 (context A shadow register).
109	FIFO_config1_A	uint	R0x0E:2	RW	0x0002	FIFO Buffer configuration 1 (context A shadow register).
111	FIFO_config2_A	uchar	R0x0F:2	RW	0x21	FIFO Buffer configuration 2 (context A shadow register).
112	FIFO_len_timing_A	uint	R0x12:2	RW	0x0606	FIFO spoof frame line timing (context B shadow register).
114	FIFO_config0_B	uint	R0x0D:2	RW	0x0067	FIFO Buffer configuration 0 (context B shadow register).
116	FIFO_config1_B	uint	R0x0E:2	RW	0x050A	FIFO Buffer configuration 1 (context B shadow register).
118	FIFO_config2_B	uchar	R0x0F:2	RW	0x0003	FIFO Buffer configuration 2 (context B shadow register).
119	FIFO_len_timing_B	uint	R0x12:2	RW	0x0606	FIFO spoof frame line timing (context B shadow register).
121	spoof_width_B	uint	R0x10:2*	RW	1600	FIFO spoof frame line timing (context B shadow register). *If in uncompressed mode, 2x this value is uploaded to R0x10.
123	spoof_height_B	uint	R0x11:2	RW	600	FIFO spoof frame line timing (context B shadow register).
125	out_format_A	uchar	R0x97:1	RW	0	Output Format Config. (context A shadow register).
126	out_format_B	uchar	R0x97:1	RW	0	Output Format Config. (context B shadow register).
127	spec_effects_A	uint	R0xA4:1	RW	0x6440	Special effects selection (context A shadow register).
129	spec_effects_B	uint	R0xA4:1	RW	0x6440	Special effects selection (context B shadow register).

Table 14: Driver Variables–Mode/Context Driver (ID = 7) (continued)

Offs	Name	Type	ASSOC.H/W Reg	RW	Default	Description
131	y_rgb_offset_A	uchar	R0xBF:1	RW	0	Y/RGB Offset setting (context A shadow register).
132	y_rgb_offset_B	uchar	R0xBF:1	RW	0	Y/RGB Offset setting (context B shadow register).

JPEG

Table 15: Driver Variables–JPEG Driver (ID = 9)

Offs	Name	Type	Def	RW	Description
0	vmt	void*	58451	RW	Reserved.
2	width	uint	1600	R	Image width (see mode.output_width).
4	height	uint	1200	R	Image height (see mode.output_height).
6	format	uchar	0	RW	Image format: 0: YCbCr 4:2:2 1: YCbCr 4:2:0 2: Monochrome
7	config	uchar	52	RW	Configuration and handshaking: Bit 0: if 1, video; if 0, still snapshot Bit 1: enable handshaking with host at every error frame Bit 2: enable retry after an unsuccessful encode or transfer Bit 3: host indicates it is ready for next frame Bit 4: enable scaled quantization table generation Bit 5: enable auto-select quantization table Bit 7:6: quantization table ID
8	restartInt	uint	0	RW	Restart marker interval: 0 = restart marker is disabled
10	qscale1	uchar	6	RW	Bit 6:0: scaling factor for first set of quantization tables Bit 7: 1, new scaling factor value
11	qscale2	uchar	9	RW	Bit 6:0: scaling factor for second set of quantization tables Bit 7: if 1, new scaling factor value
12	qscale3	uchar	12	RW	Bit 6:0: scaling factor for third set of quantization tables Bit 7: if 1, new scaling factor value
13	timeoutFrames	uchar	10	RW	Number of frames to time out when host is not responding (setting bit 3 of config) to an unsuccessful JPEG frame while bit 1 of config is set.
14	state	uchar	0	R	JPEG driver state.
15	dataLengthMSB	uchar		R	Bit [23:16] of previous frame JPEG data length.
16	dataLengthLSBs	uint		R	Bit [15:0] of previous frame JPEG data length.

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 1,600 bytes, enough to hold one half uncompressed line at full resolution. Buffering of data is a way to equalize the data output rate when image decimation is used. Decimation produces an intermittent data stream consisting of short high-rate bursts separated by idle periods. Figure 6 depicts such a stream. High pixel clock frequency during bursts may be undesirable due to EMI concerns.

Figure 6: Timing of Decimated Uncompressed Output Bypassing the FIFO

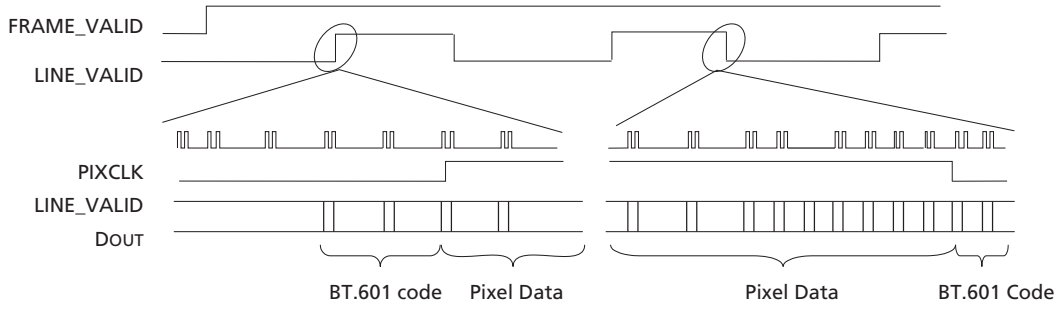
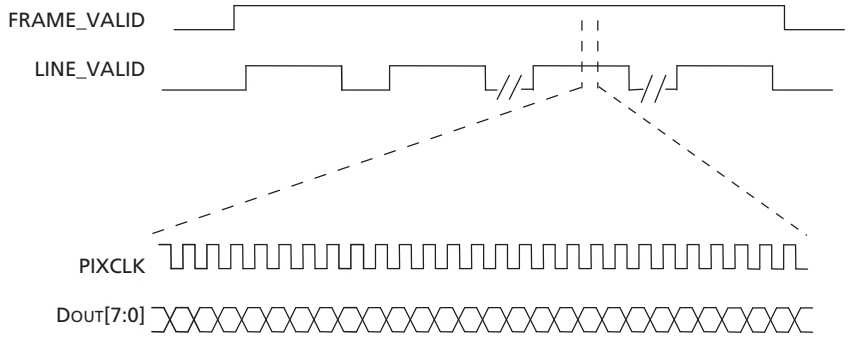


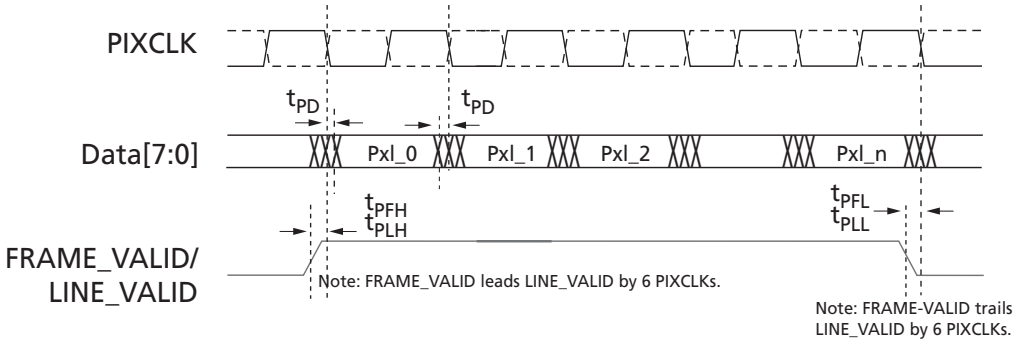
Figure 7 depicts the output timing of uncompressed YUV/RGB when a decimated data stream is equalized by buffering or when no decimation takes place. The pixel clock frequency remains constant during each LINE_VALID high period. Decimated data are output at a lower frequency than full size frames, which helps to reduce EMI.

Figure 7: Timing of Uncompressed Full Frame Output or Decimated Output Passing Through the FIFO



Timing details of uncompressed YUV/RGB output are shown in Figure 8 on page 74.

Figure 8: Details of Uncompressed YUV/RGB Output Timing



Symbol	Definition	Conditions	Min	Max	Units
^{f}PIXCLK	PIXCLK frequency	Default		80	MHz
^{t}PD	PIXCLK to data valid	Default	-3	3	ns
^{t}PFH	PIXCLK to FV high	Default	-3	3	ns
^{t}PLH	PIXCLK to LV high	Default	-3	3	ns
^{t}PFL	PIXCLK to FV low	Default	-3	3	ns
^{t}PLL	PIXCLK to LV low	Default	-3	3	ns

Uncompressed YUV/RGB Data Ordering

The MT9D131 supports swapping YCrCb mode, as illustrated in Table 16.

Table 16: YCrCb Output Data Ordering

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

The RGB output data ordering in default mode is shown in Table 17. The odd and even bytes are swapped when luma/chroma swap is enabled. R and B channels are bit-wise swapped when chroma swap is enabled.

Table 17: RGB Ordering in Default Mode

Mode (Swap Disabled)	Byte	$D_7D_6D_5D_4D_3D_2D_1D_0$
565RGB	Odd	$R_7R_6R_5R_4R_3G_7G_6G_5$
	Even	$G_4G_3G_2B_7B_6B_5B_4B_3$
555RGB	Odd	$0\ R_7R_6R_5R_4R_3G_7G_6$
	Even	$G_4G_3G_2B_7B_6B_5B_4B_3$
444xRGB	Odd	$R_7R_6R_5R_4G_7G_6G_5G_4$
	Even	$B_7B_6B_5B_4\ 0\ 0\ 0\ 0$
x444RGB	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

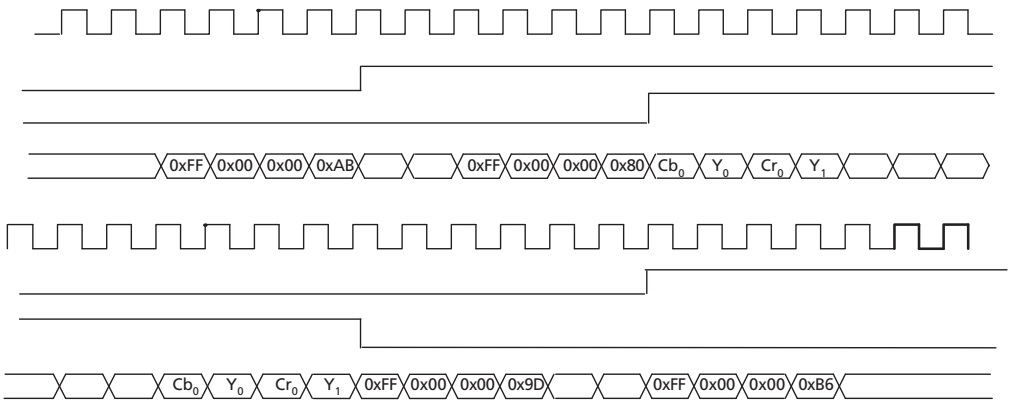
Raw 10-bit Bayer data from the sensor core can be output in bypass mode by using the eight output signals (DOUT[7:0]) in a special 8 + 2 data format, as shown in Table 18.

The timing of 10-bit or 8-bit data stream output in the bypass mode is qualitatively the same as that depicted in Figure 7.

Table 18: 2-Byte RGB Format

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

Figure 9: Example of Timing for Non-Decimated Uncompressed Output Bypassing Output FIFO



JPEG Compressed Output

JPEG compression of IFP output produces a data stream whose structure differs from that of an uncompressed YUV/RGB stream. Frames are no longer sequences of lines of constant length. This difference is reflected in the timing of the LINE_VALID signal. When JPEG compression is enabled, logical HIGHs on LINE_VALID do not correspond to image lines, but to variably sized packets of valid data. In other words, the LINE_VALID signal is in fact a DATA_VALID signal. It is a good analogy to compare the JPEG output of the MT9D131 to an 8-bit parallel data port wherein the LINE_VALID signal indicates valid data and the FRAME_VALID signal indicates frame timing.

The JPEG compressed data can be output either continuously or in blocks simulating image lines. The latter output scheme is intended to spoof a standard video pixel port connected to the MT9D131 and for that purpose treats JPEG entropy-coded segments as if they were standard video pixels. In the continuous output mode, JPEG output clock can be free-running or gated. In all, three timing modes are available and are depicted in Figure 10 on page 77, Figure 11 on page 77, and Figure 12 on page 77. These timing diagrams are merely three typical examples of many variations of JPEG output. Description of output configuration register R0xD:2 in Table 7, “IFP Registers, Page 2,” on page 43 provides more information on different output interface configuration options.

The “continuous” and spoof JPEG output modes differ primarily in how the LINE_VALID output is asserted. In the continuous mode, LINE_VALID is asserted only during output clock cycles containing valid JPEG data. The resulting LINE_VALID signal pattern is non-

uniform and highly image-dependent, reflecting the inherent nature of JPEG data stream. In the spoof mode, LINE_VALID is asserted and de-asserted in a more uniform pattern emulating uncompressed video output with horizontal blanking intervals. When LINE_VALID is de-asserted, available JPEG data are not output, but instead remain in the FIFO until LINE_VALID is asserted again. During the time when LINE_VALID is asserted, the output clock is gated off whenever there is no valid JPEG data in the FIFO.

Note: As a result, spoof "lines" containing the same number of valid data bytes may be output within different time intervals depending on constantly varying JPEG data rate.

The host processor configures the spoof pattern by programming the total number of LINE_VALID assertion intervals, as well as the number of output clock periods during and between LINE_VALID assertions. In other words, the host processor can define a temporal "frame" for JPEG output, preferably with "size" tailored to the expected JPEG file size. If this frame is too large for the total number of JPEG bytes actually produced, the MT9D131 either de-asserts FRAME_VALID or continues to pad unused "lines" with zeros until the end of the frame. If the frame is too small, the MT9D131 either continues to output the excess JPEG bytes until the entire JPEG compressed image is output or discards the excess JPEG bytes and sets an error flag in a status register accessible to the host processor.

In the continuous output mode, the JPEG output clock can be configured to be either gated off or running while LINE_VALID is de-asserted. To save extra power, the JPEG output clock can also be gated off between frames (when FRAME_VALID is de-asserted) in both continuous and spoof output mode. In the continuous output mode, there is an option to insert JPEG SOI (0xFFD8) and EOI (0xFFD9) markers respectively before and after valid JPEG data. SOI and EOI can be inserted either inside or outside the FRAME_VALID assertion period, but always outside LINE_VALID assertions.

The output order of even and odd bytes of JPEG data can be swapped in the spoof output mode. This option is not supported in the continuous mode.

Output clock speed can optionally be made to vary according to the fullness of the FIFO, to reduce the likelihood of FIFO overflow. When this option is enabled, the output clock switches at three fixed levels of FIFO fullness (25 percent, 50 percent, and 75 percent) to a higher or lower frequency, depending on the direction of fullness change. The set of possible output clock frequencies is restricted by the fact that its period must be an integer multiple of the master clock period. The frequencies to be used are chosen by programming three output clock frequency divisors in registers R0xE:2 and R0xF:2. Divisor N1 is used if the FIFO is less than 50 percent full and last fullness threshold crossed has been 25 percent. When the FIFO reaches 50 percent and 75 percent fullness, the output clock switches to divisor N2 and N3, respectively. When the FIFO fullness level drops to 50 percent and 25 percent, the output clock is switched back to divisor N2 and N1, respectively.

The host processor can read registers containing JPEG status flags and JPEG data length (total byte count of valid JPEG data) through a two-wire serial interface. In addition, the JPEG data length and JPEG status byte are always appended at the end of JPEG spoof frame. JPEG status byte can be optionally appended at the end of JPEG continuous frame. JPEG data stream sent to the host does not have a header.

Figure 10: Timing of JPEG Compressed Output in Free-Running Clock Mode

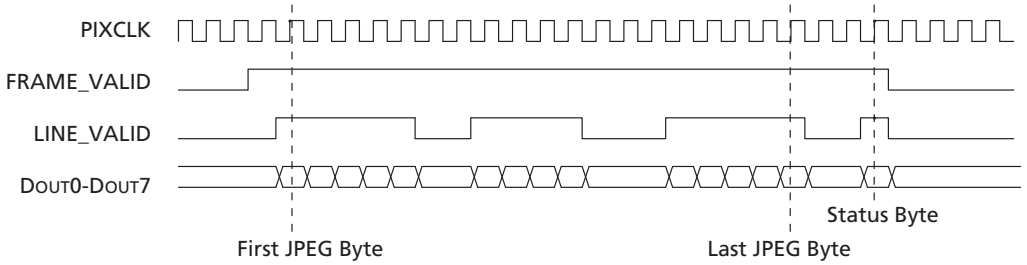


Figure 11: Timing of JPEG Compressed Output in Gated Clock Mode

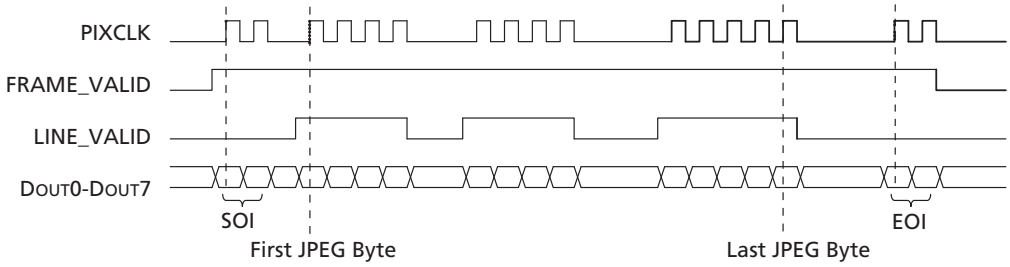
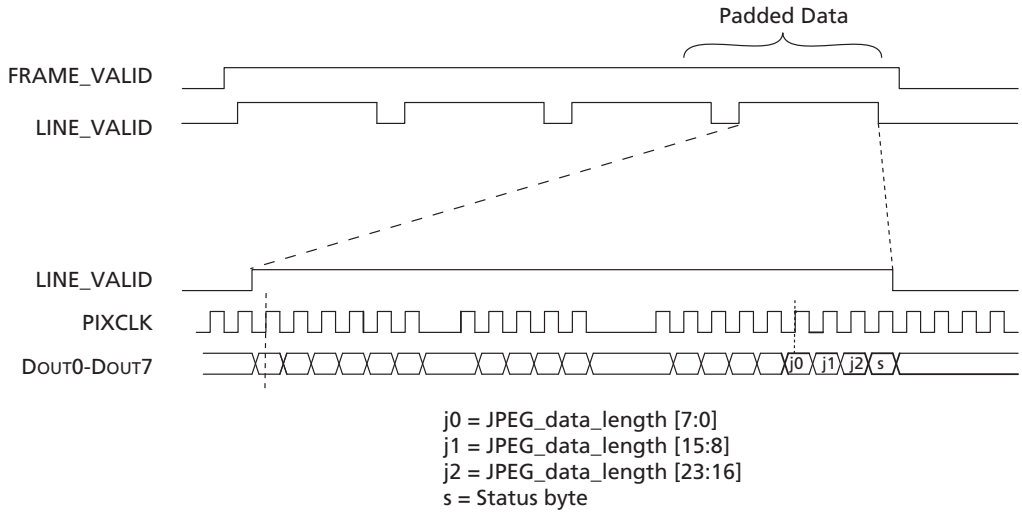


Figure 12: Timing of JPEG Compressed Output in Spoof Mode



In the spoof mode, the timing of FRAME_VALID and LINE_VALID mimics an uncompressed output. The timing of PIXCLK and DOUT within each LINE_VALID assertion period is variable and therefore unlike that of uncompressed data. Valid JPEG data are padded with dummy data to the size of the original uncompressed frame.

Color Conversion Formulas

Y'Cb'Cr' ITU-R BT.601

A widely known color conversion standard. Note that $16 < Y_{601} < 235$ and $16 < Cb, Cr < 240$. $0 <= RGB <= 255$.

$Y_{601}' = Y'*219/256 + 16$

$Cr' = 0.713 (R' - Y')*224/256 + 128$

$Cb' = 0.564 (B' - Y')*224/256 + 128$

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

The reverse formulas to convert YCbCr into RGB, $0 <= RGB <= 255$:

$R' = 1.164(Y_{601}' - 16) + 1.596(Cr' - 128)$

$G' = 1.164(Y_{601}' - 16) - 0.813(Cr' - 128) - 0.391(Cb' - 128)$

$B' = 1.164(Y_{601}' - 16) + 2.018(Cb' - 128)$

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 Windows.

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

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

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

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

Y'Cb'Cr Using sRGB Formulas

The MT9D131 implements the sRGB standard. This option provides YCbCr coefficients for a correct 4:2:2 transmission. Note that $16 < Y60\ 1 < 235$, $16 < Cb, Cr < 240$ and $0 <= RGB <= 255$.

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

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

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

Y'U'V' Using sRGB Formulas

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

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

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

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

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

$$R' = Y + 1.5748*V$$

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

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

Sensor Core

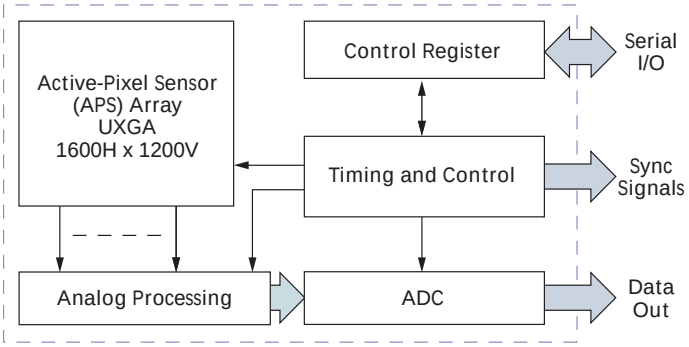
This section describes the sensor core. The core is based entirely on Micron's MT9D011 sensor.

The SOC firmware controls a key sensor core registers, such as exposure, window size, gains, and contexts. When firmware or MCU are disabled, the sensor core can be programmed directly.

Introduction

The sensor core is a progressive-scan sensor that generates a stream of pixel data qualified by LINE_VALID and FRAME_VALID signals. An on-chip PLL generates the master clock from an input clock of 6 MHz to 40 MHz. In default mode, the data rate (pixel clock) is the same as the master clock frequency, which means that one pixel is generated every master clock cycle. The sensor block diagram is shown in Figure 13.

Figure 13: Sensor Core Block Diagram



The core of the sensor is an active-pixel array. The timing and control circuitry sequences through the rows of the array, resetting and then reading each row. In the time interval between resetting a row and reading that row, the pixels in that row integrate incident light. The exposure is controlled by varying the time interval between reset and readout. After a row is 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 "black" pixels. The black pixels are used to provide data for on-chip offset correction algorithms (black level control).

The sensor contains a set of 16-bit control and status registers that can be used to control many aspects of the sensor operations. These registers can be accessed through a two-wire serial interface. In this document, registers are specified either by name (Column Start) or by register address (R0x04:0). Fields within a register are specified by bit or by bit range (R0x20:0[0] or R0x0B:0[13:0]). The control and status registers are described in Table 5, "Sensor Core Register Description," on page 23.

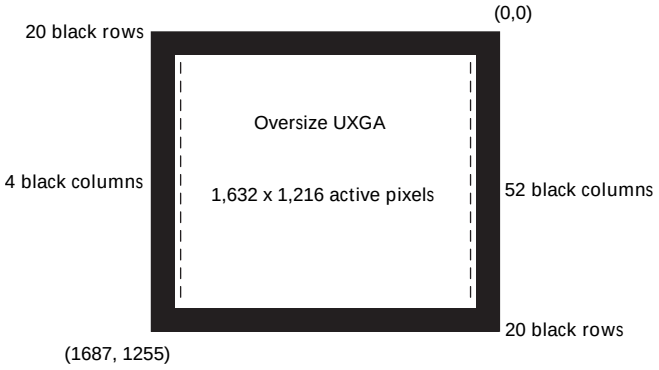
The output from the sensor is a Bayer pattern: alternate rows are a sequence of either green/red pixels or blue/green pixels. The offset and gain stages of the analog signal chain provide per-color control of the pixel data.

Pixel Array Structure

The sensor core pixel array is configured as 1,688 columns by 1,256 rows (shown in Figure 14). The first 52 columns and the first and the last 20 rows of pixels are optically black and are used for the automatic black level adjustment. The last 4 columns are also optically black.

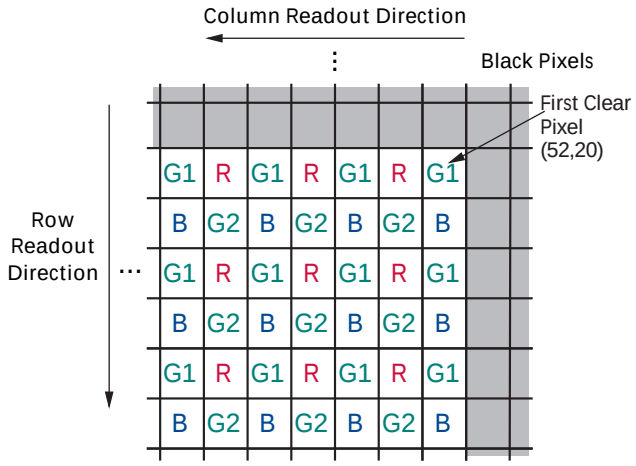
The optically active pixels are used as follows: In default mode a UXGA image (1,600 columns by 1,200 rows) is generated, starting at row 28, column 60. A 4-pixel boundary of active pixels can be enabled around the image to avoid boundary effects during color interpolation and correction. During mirrored readout, the region of active pixels used to generate the image is offset by 1 pixel in each mirrored direction so that the readout always starts on the same color pixel.

Figure 14: Pixel Array



The sensor core uses a Bayer color pattern, as shown in Figure 15. The even-numbered rows contain green and red color pixels; odd-numbered rows contain blue and green color pixels. Even-numbered columns contain green and blue color pixels; odd-numbered columns contain red and green color pixels. The color order is preserved during mirrored readout.

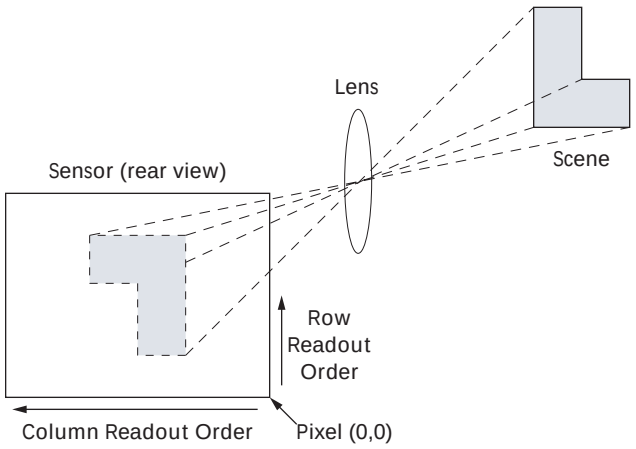
Figure 15: Pixel Color Pattern Detail (Top Right Corner)



Default Readout Order

By convention, the sensor core pixel array is shown with pixel (0,0) in the top right corner (see Figure 15). This reflects the actual layout of the array on the die. When the sensor is imaging, the active surface of the sensor faces the scene as shown in Figure 16. When the image is read out of the sensor, it is read one row at a time, with the rows and columns sequenced as shown in Figure 14. By convention, data from the sensor is shown with the first pixel read out—pixel (52,20) in the case of the sensor core—in the top left corner.

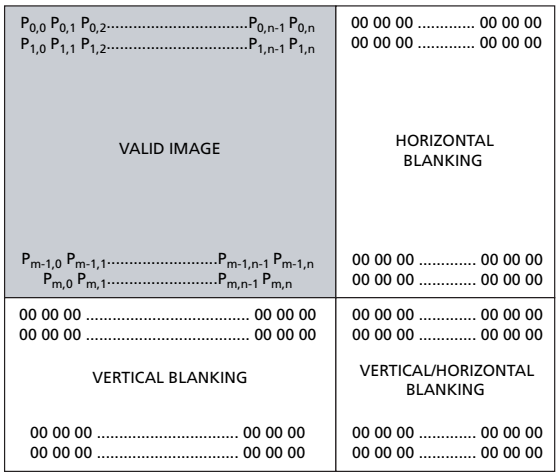
Figure 16: Imaging a Scene



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 17. The amount of horizontal blanking and vertical blanking is programmable. LINE_VALID is HIGH during the shaded region of the figure. FRAME_VALID timing is described in the next section.

Figure 17: Spatial Illustration of Image Readout



Raw Data Timing

The sensor core output data is synchronized with the PIXCLK output. When LINE_VALID is HIGH, one pixel datum is output on the 10-bit DOUT output every PIXCLK period. By default, the PIXCLK signal runs at the same frequency as the master clock, and its rising edges occur one-half of a master clock period after transitions on LINE_VALID, FRAME_VALID, and DOUT (see Figure 18). This allows PIXCLK to be used as a clock to sample the data. PIXCLK is continuously enabled, even during the blanking period. The sensor core can be programmed to delay the PIXCLK edge relative to the DOUT transitions from 0 to 3.5 master clocks, in steps of one-half of a master clock. This can be achieved by programming the corresponding bits in R0x0A:0. The parameters P, A, and Q in Figure 19 are defined in Table 19 on page 85.

Figure 18: Pixel Data Timing Example

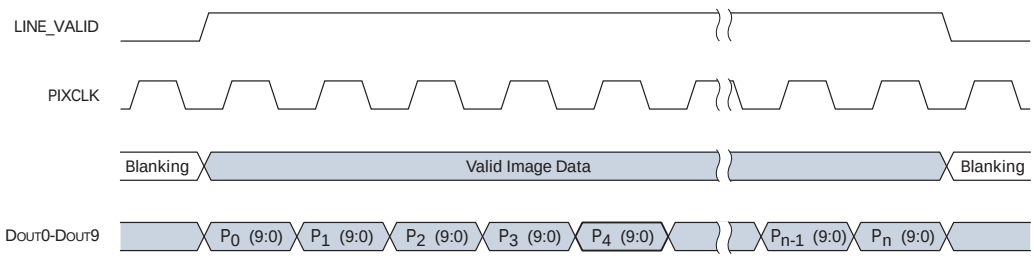
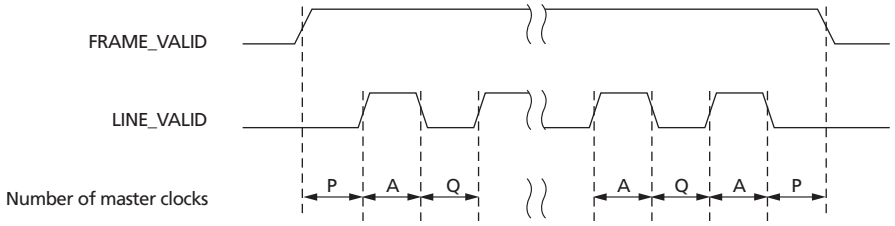


Figure 19: Row Timing and FRAME_VALID/LINE_VALID Signals



The sensor timing is shown in terms of pixel clock and master clock cycles (see Figure 18 on page 83). The recommended master clock frequency is 36 MHz. Increasing the integration time to more than one frame causes the frame time to be extended. The equations in Table 19 on page 85 assume integration time is less than the number of rows in a frame (R0x09:0 < R0x03:0/S + BORDER + VBLANK_REG). If this is not the case, the number of integration rows must be used instead to determine the frame time, as shown in Table 20 on page 85.

Table 19: Frame Time

Parameter	Name	Equation	Default Timing at 36 MHz Dual ADC Mode
HBLANK_REG	Horizontal blanking register	R0x07:0 if R0xF2:0[0] = 0 R0x05:0 if R0xF2:0[0] = 1	0x15C = 348 pixels
VBLANK_REG	Vertical blanking register	R0x8:0 if R0xF2:0[1] = 0 R0x6:0 if R0xF2:0[1] = 1	0x20 = 32 rows
ADC_MODE	ADC mode	R0xF2:0[3] = 0: R0x20:0[10] R0xF2:0[3] = 1: R0x21:0[10]	
PIXCLK_PERIOD	Pixel clock period	ADC_MODE = 0: R0x0A:0[2:0] ADC_MODE = 1: R0x0A:0[2:0]*2	1 ADC_MODE: 55.556ns 2 ADC_MODE: 27.778ns
S	Skip factor	For skip 2x mode: S = 2 For skip 4x mode: S = 4 For skip 8x mode: S = 8 For skip 16x mode: S = 16 otherwise, S = 1	1
A	Active data time	(R0x04:0/S) * PIXCLK_PERIOD	1,600 pixel clocks = 1,600 master = 44.44µs
P	Frame start/end blanking	6 * PIXCLK_PERIOD (can be controlled by R0x1F:0)	6 pixel clocks = 12 master = 0.166µs
Q	Horizontal blanking	HBLANK_REG * PIXCLK_PERIOD	348 pixel clocks = 348 master = 9.667µs
A + Q	Row time	((R0x04:0/S) + HBLANK_REG) * PIXCLK_PERIOD	1,948 pixel clocks = 1,948 master = 54.112µs
V	Vertical blanking	VBLANK_REG * (A + Q) + (Q - 2*P)	62,672 pixel clocks = 62,672 master = 1.741ms
Nrows * (A + Q)	Frame valid time	(R0x03:0/S) * (A + Q) - (Q - 2*P)	2,337,264 pixel clocks = 2,337,264 master = 64.925ms
F	Total frame time	((R0x03:0/S) + VBLANK_REG) * (A + Q)	2,399,936 pixel clocks = 2,399,936 master = 66.665ms

Note: Skip factor should be multiplied by 2 if binning is enabled.

Table 20: Frame—Long Integration Time

Parameter	Name	Equation (Master Clock)
V'	Vertical blanking (long integration time)	(R0x09:0 – (R0x03:0)/S) * (A + Q) + (Q - 2*P)
F'	Total frame time (long integration time)	(R0x09:0) * (A + Q)

Registers

Table 5, “Sensor Core Register Description,” on page 23 provides a detailed description of the registers. Bit fields that are not identified in the table are read-only.

Double-Buffered Registers

Some sensor settings cannot be changed during frame readout. For example, changing row width R0x03:0 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. R0x22:0 enables the dark rows to be shown in the image, but this has no effect on the position of frame start.

To determine which registers or register fields are double-buffered in this way, see Table 5, “Sensor Core Register Description,” on page 23, the “sync’d-to-frame-start” column.

R0x0D:0[15] can be used to inhibit transfers from the pending to the live registers. This control bit 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 row width R0x03:0 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.

To determine which register or register field changes can produce a bad frame, see Table 5, “Sensor Core Register Description,” on page 23, the “bad frame” column, and these notations:

- N—No. Changing the register value does not produce a bad frame
- Y—Yes. Changing the register value might produce a bad frame
- YM—Yes; but the bad frame is masked out unless the “show bad frames” feature (R0x0D:0[8]) is enabled

Changes to Integration Time

If the integration time (R0x09:0) 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 R0x09:0 pending register.
2. At the start of frame (*n +1*), the new integration time is transferred to the R0x09:0 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.

If the integration time is changed (R0x09:0 written) 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 (R0x2B:0, R0x2C:0, R0x2D:0, R0x2E:0, and R0x2F:0) 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. All the firmware drivers must be off (that is, seq.cmd = 0) before manual control of gains or integration time (R0x09:0) is possible.

Feature Description

PLL Generated Master Clock

The PLL embedded in the sensor core can generate a master clock signal whose frequency is up to 80 MHz (input clock from 6 MHz through 64 MHz). Registers R0x66:0 and R0x67:0 control the frequency of the PLL-generated clock. It is possible to bypass the PLL and use EXTCLK as master clock. To do so, one must set R0x65:0[15] to 1. If power consumption is a concern, R0x65:0[14] should be also set to 1 a short time later, to put the bypassed PLL in power down mode. To enable the PLL again, the two bits must be set to 0 in the reverse order. By default, the PLL is bypassed and powered down.

PLL Setup

The PLL output frequency is determined by three constants (M, N, and P) and the input clock frequency. These three values are set in:

R0x66:0// [15:8] for M; [5:0] for N

R0x67:0// [6:0] for P

Their relations can be shown by the following equation:

$$^{f}PLL, ^{f}OUT = {}^{f}PLL, {}^{f}IN, \times M / [2 \times (N+1) \times (P+1)]$$

However, since the following requirements must be satisfied, then not all combinations of M/N/P are valid:

M must be 16 or higher

$^{f}PFD, ^{f}VCO, ^{f}OUT$ ranges are satisfied

Table 21: Frequency Parameters

Frequency	Equation	Min (MHz)	Max (MHz)
^{f}PFD	$^{f}IN / (N+1)$	2	13
^{f}VCO	$^{f}PFD \times M$	110	240
^{f}OUT	$^{f}VCO / [2 \times (P+1)]$	6	80
^{f}IN	–	6	64

After determining the proper M, N, and P values and setting them in R102:0/R103:0, the PLL can be enabled by the following sequence:

R0x65:0[14] = 0 // powers on PLL

R0x65:0[15] = 0 // disable PLL bypass (enabling PLL)

Note: If PLL is used, bypass the PLL (R0x65:0[15] = 1) before going into hard standby. It can be enabled again (R0x65:0[15] = 0) once the sensor is out of standby.

PLL Power-up

The PLL takes time to power up. During this time, the behavior of its output clock signal is not guaranteed. The PLL is in the power-down mode by default and must be turned on manually. When using the PLL, the correct power-up sequence after chip reset is as follows:

1. Program PLL frequency settings (R0x66:0 and R0x67:0)
2. Power up the PLL (R0x65:0[14] = 0)
3. Wait for a time longer than PLL locking time (> 1ms)
4. Turn off the PLL bypass (R0x65:0[15] = 0)

Window Control

Window Start

The row and column start address of the displayed image can be set by R0x01:0 (row start) and R0x02:0 (column start).

Window Size

The size of the sensor core image is controlled by R0x03:1 (Row Width) and R0x04:0 (column width). The default image size is 1,600 columns and 1,200 rows (UXGA).

The window start and size registers can be used to change the number of columns in the image from 17 through 1,632 and the number of rows from 2 through 1,216. The displayed image size is controlled by the output and crop variables in the mode (ID = 7) driver.

Pixel Border

When bits R0x20:0[9:8] are both set, a 4-pixel border is added around the specified image. This border can be used as extra pixels for image processing algorithms. The border is independent of the readout mode, which means that even in skip, zoom, and binning modes, a 4-pixel border is output in the image. When enabled, the row and column widths are 8 pixels larger than the values programmed in R0x03:0 and R0x04:0. If the border is enabled but not shown in the image (R0x20[9:8] = 01), the horizontal blanking and vertical blanking values are 8 pixels larger than the values programmed in the blanking registers.

Readout Modes

Readout Speeds and Power Savings

The sensor core has two ADCs to convert the pixel values to digital data. Because the ADCs run at half the master clock frequency, it is possible to achieve a data rate equal to the master clock frequency. By turning off one of the ADCs, the power consumption of the sensor is reduced. The pixel clock is then reduced by a factor of two.

In R0x20:0 or R0x21:0, bit 10 chooses between the two modes:

- 0: Use both ADCs and read out at the set pixel clock frequency (R0x0A:0[3:0])
- 1: Use 1 ADC and read out at half the set pixel clock frequency (R0x0A:0[3:0])

This can be used, for instance, when the camera is in preview mode. To make the transitions between two sensor settings easier, some simple context switching is described in “Context Switching” on page 94.

Column Mirror Image

By setting R0x20:0[1] = 1 (R0x21:0 in context A), the readout order of the columns are reversed as shown in Figure 20. The starting color is preserved when mirroring the columns.

Row Mirror Image

By setting R0x20:0[0] = 1 (R0x21:0 in context A), the readout order of the rows are reversed as shown in Figure 21. The starting color is preserved when mirroring the rows.

Figure 20: 6 Pixels in Normal and Column Mirror Readout Modes

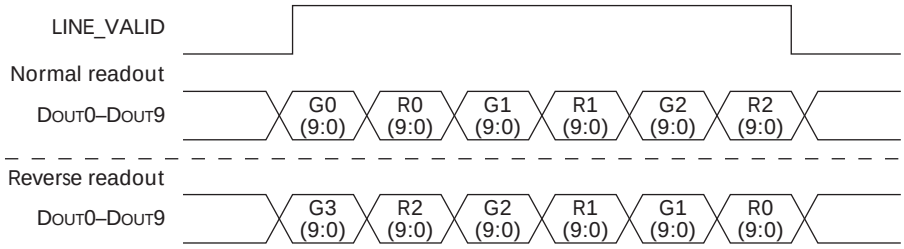
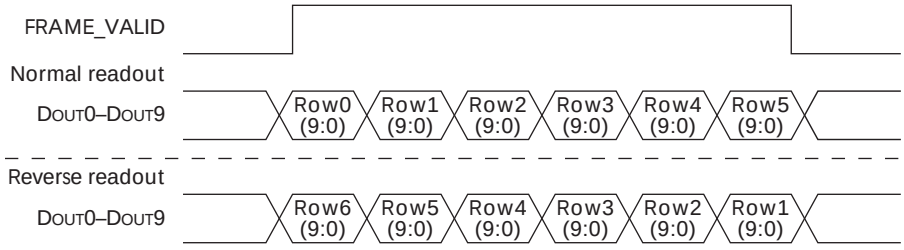


Figure 21: 6 Rows in Normal and Row Mirror Readout Modes



Column and Row Skip

This section assumes context B. If context A is used, replace all references to R0x20:0 with R0x21:0.

By setting R0x20:0[4] = 1 or R0x20:0[7] = 1, skip is enabled for rows or columns, respectively. When skip is enabled, the image is subsampled. The amount of skipping is set by R0x20:0[3:2] (rows) and R0x20:0[6:5] (columns) according to Table 22. Depending on the skip value, the output size variable in the mode (ID = 7) driver might need adjustments.

Table 22: Skip Values

Bit Values	Skip Value
00	2
01	4
10	8
11	16

The number of rows or columns read out is what is set in R0x03:0 or R0x04:0, respectively, divided by the skip value in this table.

In all cases, the row and column sequencing ensures that the Bayer pattern is preserved. Column skip examples are shown in Figures 22 through 25.

Figure 22: 8 Pixels in Normal and Column Skip 2x Readout Modes

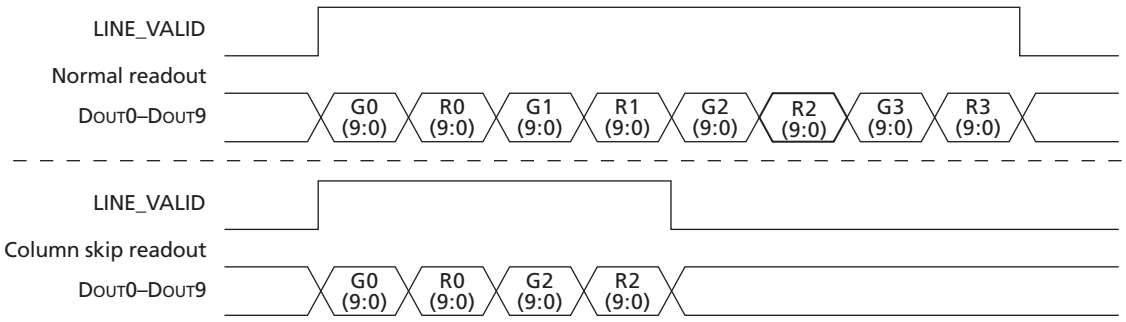


Figure 23: 16 Pixels in Normal and Column Skip 4x Readout Modes

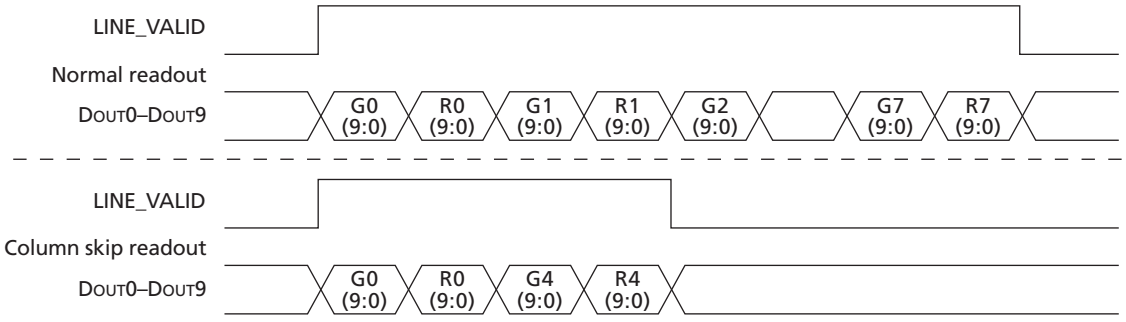


Figure 24: 32 Pixels in Normal and Column Skip 8x Readout Modes

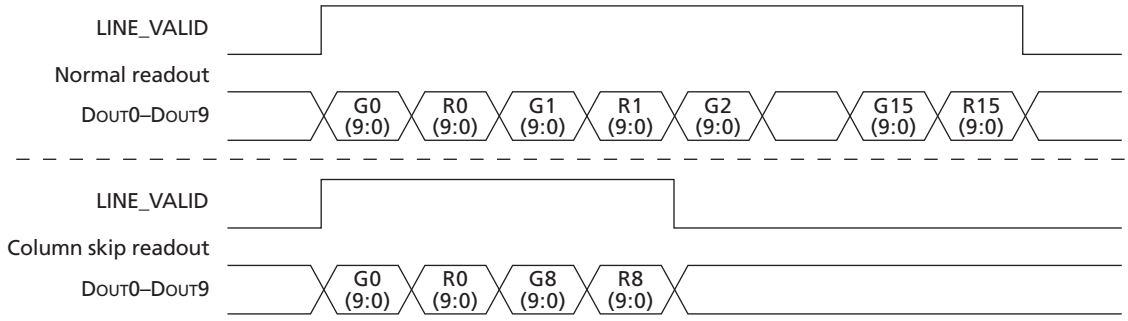
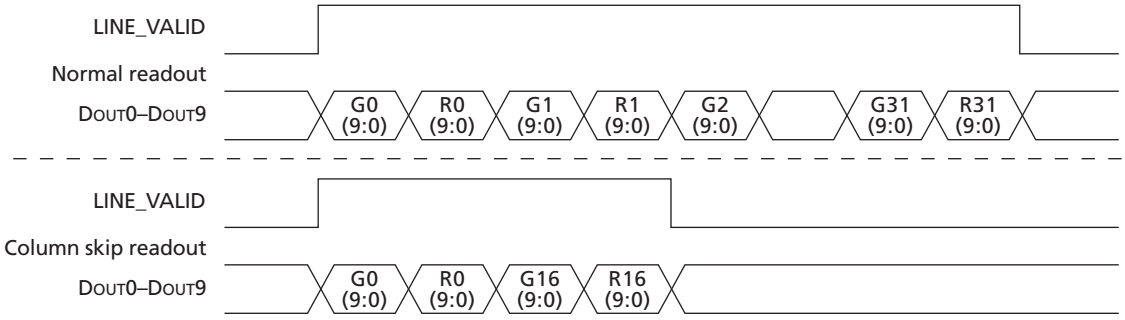


Figure 25: 64 Pixels in Normal and Column Skip 16x Readout Modes



Digital Zoom

Digital zoom is not used in SOC.

Binning

The sensor core supports 2 x 2 binning of 4 pixels of the same color. This mode can be activated by asserting R0x20:0[15] (R0x21:0 if context A is used).

Binning is primarily used instead of 2x skip as a way of decimating the picture without losing information. The effect of aliasing in preview mode is eliminated when binning is used instead of just skipping rows and columns.

Activating binning has a few implications:

- It adds a level of skip, so the picture that comes out has the same dimensions as a picture read out with the next higher skip setting.
- It increases the minimum hblank and minimum row time requirements (see Table 25 and Table 26).

Table 23: Row Addressing

Number of ADCs	Binning	Row Addressing (not considering start address)
1	No	0, 1, 2, 3, 4, 5, 6, 7,...
2	No	0, 1, 2, 3, 4, 5, 6, 7,...
1	Yes	0/2, 1/3, 4/6, 5/7,...
2	Yes	0/2, 1/3, 4/6, 5/7,...

Table 24: Column Addressing

Number of ADCs	Binning	Column Addressing (not considering start address)
1	No	0, 1, 2, 3, 4, 5, 6, 7,...
2	No	0/1, 2/3, 4/5, 6/7,...
1	Yes	0/2, 1/3, 4/6, 5/7,...
2	Yes	0/1/2/3, 4/5/6/7,...

Binning Limitations

To achieve correct operation, the following conditions must be met:

- Start address must be divisible by four (row and column).
- Window size must be divisible by four in both directions, after dividing by zoom factor and skip factor (because they both reduce the effective window size from the sensor's point of view).

Example: Default row size = 1,200. 8x zoom means the actual window on the sensor is divided by 8, so 8x zoom and binning is not allowed with default window size, because 1,200 / 8 = 150, which is not divisible by 4.

- Binning can be seen as an extra level of skip. The combination binning/16x skip is therefore not legal.

Frame Rate Control

For a given window size, the blanking registers (R0x05:0 - R0x08:0) along with the row speed register (R0x0A:0) can be used to set a particular frame rate.

The frame timing equations (Table 19 and Table 20 on page 85) can be rearranged to express the horizontal blanking or vertical blanking values as a function of the frame rate:

HBLANK_REG = master clock freq / (frame rate*
((R0x03:0/S + BORDER) + VBLANK_REG)*PIXCLK_PERIOD) - (R0x04:0/S + BORDER)

VBLANK_REG = master clock freq / (frame rate*
((R0x04:0/S + BORDER) + HBLANK_REG)*PIXCLK_PERIOD) - (R0x03:0/S + BORDER)

The HBLANK_REG value allows the frame rate to be adjusted with a minimum resolution of one PIXCLK_PERIOD multiplied by the total number of rows (displayed plus blanking). When finer resolution is required, R0x0B:0 (extra delay) can be used. R0x0B:0 allows the frame time to be changed in increments of pixel clocks.

Minimum Horizontal Blanking

The minimum horizontal blanking value is constrained by the time used for sampling a row of pixels and the overhead in the row readout. This is expressed in Table 25.

Table 25: Minimum Horizontal Blanking Parameters

Parameter	Default / 2 ADC Mode, No Binning	1 ADC Mode, No Binning	2 ADC Mode, Binning	1 ADC Mode, Binning
HBLANK(MIN)	286 mclks	324 mclks = 162 pixclks	470 mclks	508 mclks = 254 pixclks

Minimum Row Time Requirement

The total row time must be sufficient to allow all row operations (readout and shutter operations). The row time is the sum of column width (halved during binning divided by column skip factor) and horizontal blanking, and can therefore be adjusted by programming these.

Table 26 shows minimum row time as a function of mode of operation.

This is a particularly strict requirement during binning because twice as many row operations are required per row and the column width is halved.

Table 26: Minimum Row Time Parameters

Parameter	Default / 2 ADC Mode, No Binning	1 ADC Mode, No Binning	2 ADC Mode, Binning	1 ADC Mode, Binning
ROW_TIME(MIN)	473 mclks	488 mclks = 244 pixclks	931 mclks	946 mclks = 473 pixclks
pointer_operations	461 mclks	464 mclks	919 mclks	922 mclks

Context Switching

When the sensor is in the SOC bypass mode, R0xF2:0 is designed to enable easy switching between sensor modes. Some key registers and bits in the sensor have two physical register locations, called contexts. Bits 0, 1, and 3 of R0xF2:0 control which context register context is currently in use. A "1" in a bit selects context B, while a "0" selects context A for this parameter. The select bits can be used in any combination, but by default are set up to allow easy switching between preview mode and full resolution mode:

Context B (Default context)

R0xF2:0	= 0x000B	(Context B)
R0x05:0	= 0x015C	(Horizontal blanking, context B)
R0x06:0	= 0x0020	(Vertical Blanking, context B)
R0x20:0	= 0x0000	(2 ADCs, no column or row skip)

Description: Full resolution UXGA (1,600 x 1,200) image at 15 fps

Context A (Alternate context, preview mode)

R0xF2:0	= 0x0000	(Context A)
R0x07:0	= 0x00AE	(Horizontal blanking, context A)
R0x08:0	= 0x0010	(Vertical blanking, context A)
R0x21:0	= 0x0490	(1 ADC, 2x column and row skip)

Description: Half-resolution SVGA (800 x 600) image at 30 fps

The horizontal blanking and vertical blanking values for the two contexts are chosen so that row time is preserved between contexts. This ensures that changing contexts does not affect integration time. See Table 5, "Sensor Core Register Description," on page 23 for more information.

Settings for skip, 1 ADC mode, and binning can be set separately for context B and context A using R0x20:0 and R0x21:0, respectively. When these settings are referred to in this document, the register is dependent on what context is set in R0xF2:0. For the default (no bypass) mode, context switching is controlled by the sequencer (ID = 1) driver.

Integration Time

Integration time is controlled by R0x09:0 (shutter width in multiples of the row time) and R0x0C:0 (shutter delay, in PIXCLK_PERIOD/2). R0x0C:0 is used to control sub-row integration times and only has a visible effect for small values of R0x09:0. The total integration time, ^{t}INT, is shown in the equation below:

^{t}INT = R0x09:0 * Row Time - Integration Overhead - Shutter Delay

where:

Row time = (R0x04:0/S + BORDER + HBLANK_REG)*PIXCLK_PERIOD master clock periods (from Table on page 85)

S = Skip Factor, multiplied by 2 if binning is enabled

Overhead time = 260 master clock periods (262 in 1 ADC mode)

Shutter delay = R0x0C:0 * PIXCLK_PERIOD master clock periods (/2 in 1 ADC mode)

with default settings:

^{t}INT = (1,232 * (1,600 + 348)) - 260 - 0

= 2,399,676 master clock periods = 66.66ms at 36 MHz

In the equation, the integration overhead corresponds to the delay between the row reset sequence and the row sample (read) sequence.

Typically, the value of R0x09:0 is limited to the number of rows per frame (which includes vertical blanking rows), so that the frame rate is not affected by the integration time. If R0x09:0 is increased beyond the total number of rows per frame, the sensor adds blanking rows as needed. Additionally, ^{t}INT must be adjusted to avoid banding in the image caused by light flicker. Therefore, ^{t}INT must be a multiple of 1/120 of a second under 60Hz flicker, and a multiple of 1/100 of a second under 50Hz flicker.

Maximum Shutter Delay

The shutter delay can be used to reduce the integration time. A programmed value of *N* reduces the integration time by *N* master clock periods. The maximum shutter delay is set by the row time and the sample time, as shown in the equation below:

Maximum shutter delay = (Row Time - pointer_operations)

where:

Row time = (R0x04:0/S + BORDER + HBLANK_REG)*PIXCLK_PERIOD master clock periods (from Table on page 85)

S = Skip Factor, multiplied by 2 if binning is enabled

pointer_operations = see Table 26 on page 94.

with default settings:

Maximum shutter delay = (1,600 + 348) - 461

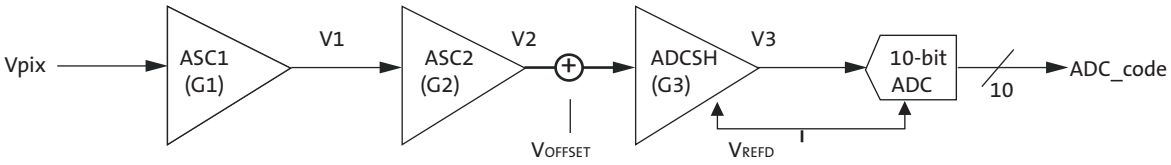
= 1,487 (master clock periods)

If the value in this register exceeds the maximum value given by this equation, the sensor may not generate an image.

Analog Signal Path

The sensor core features two identical analog readout channels. A block diagram for one channel is shown in Figure 26. The readout channel consists of two gain stages (ASC1 and ASC2), a sample-and-hold (ADCSH) stage with black level calibration capability (V_{OFFSET}), and a 10-bit ADC.

Figure 26: Analog Readout Channel



Stage-by-Stage Transfer Functions

Transfer functions proceed stage-by-stage, as follows:

Let V_{PIX} be the input of the signal path:	V_{PIX} = pixel output voltage = signal path input voltage,	
The output voltage of ASC 1st stage is:	$V1 = -1*G1*V_{PIX}$	(1)
The output voltage of ASC 2nd stage is:	$V2 = -1*G2*V1$	(2)
The output voltage of ADC Sample-and-Hold stage is:	$V3 = 2*G3*V2 - V_{REFD} + V_{OFFSET}$	(3)
and the ADC output code is:	ADC output code = $511*(1 + (V3 / V_{REFD}))$	(4)
From (1) to (4), the ADC output code can also be written as:	ADC code = $(1022/V_{REFD})*[G1*G2*G3*V_{PIX} + (Voffset/(2*G3))]$	(5)

Where G1, G2, and G3 are the gain settings, V_{OFFSET} is the offset (calibration) voltage, and V_{REFD} is the reference voltage of the ADC. The gain setting G3 is applied to the signal but is not applied to V_{OFFSET}. The parameters V_{REFD}, G1, G2, G3, and V_{OFFSET} are described next.

VREFD

The V_{REFD} parameters are as follows:

The ADC reference voltage V_{REFD} is:	$V_{REFD} = V_{REF\_HI} - V_{REF\_LO}$	(6)
where	$V_{REF\_HI} = 55.5mV*(R0x41:0[7:4] + 23)$	(7)
using default register values:	$V_{REF\_HI} = 55.5mV*(13 + 23) = 1.998V$	
and	$V_{REF\_LO} = 55.5mV*(R0x41:0[3:0] +11)$	(8)
using default register values:	$V_{REF\_LO} = 55.5mV*(7 +11) = 0.999V$	
so	$V_{REFD} = 55.5mV*(R0x41:0[7:4] - R0x41:0[3:0] + 12)$	(9)
using default register values	$V_{REFD} = 1.998 - 0.999 = 0.999V$	

Analog Gain Settings: G1, G2, G3

The analog gains for green1, blue, red, and green2 pixels (as shown in Figure 26 on page 96) are set by registers R0x2B:0, R0x2C:0, R0x2D:0, and R0x2E:0, respectively. Gain can also be globally set by R0x2F:0. The analog gain is set by bits 8:0 of the corresponding register as follows:

G1 = bit 7 + 1	(10)
G2 = bit 6:0 / 32	(11)
G3 = bit 8 + 1	(12)

Digital gain is set by bits 11:9 of the same registers.

Offset Voltage: VOFFSET

The offset voltage provides a constant offset to the ADC to fully utilize the ADC input dynamic range. The offset voltages for green1, blue, red, and green2 pixels are manually set by registers R0x61:0, R0x62:0, R0x63:0, and R0x64:0, respectively. The offset voltages also can be automatically set by the black level calibration loop.

For a given color, the offset voltage, VOFFSET, is determined by:

$$V_{OFFSET} = 0.50V * offset\_gain * offset\_sign * offset\_code[7:0]/255 \quad (13)$$

where: "offset_sign" is determined by bit 8 as:

if bit 8 = 0, offset_sign = +1 (14)

if bit 8 = 1, offset_sign = -1 (15)

"offset_code" is the decimal value of bit<7:0>

OFFSET_GAIN is determined by the 2-bit code from R0x5A:0[1:0], as shown in Table 27. These step sizes are not exact; increasing the stage0 ADC gain from 2 to 4 decreases the step size significance; decreasing the ADC VREFD increases the step size significance.

Table 27: Offset Gain

R0x5A:0[1:0]	OFFSET_GAIN
00	OFFSET_GAIN = 0 (no calibration voltage is applied)
01	OFFSET_GAIN = 0.25 (1 calibration LSB is equal to 0.5 ADC LSB when VREFD = 1V)
10	OFFSET_GAIN = 0.50 (1 calibration LSB is equal to 1 ADC LSB when VREFD = 1V)
11	OFFSET_GAIN = 1 (1 calibration LSB is equal to 2 ADC LSB when VREFD = 1V)

Recommended Gain Settings

The analog gain circuitry in the sensor core provides signal gains from 1 through 15.875.

Table 28: Recommended Gain Settings

Desired Gain	Recommended Gain Register Setting
1–1.969	0x020–0x03F
2–7.938	0x0A0–0x0FF
8–15.875	0x1C0–0x1FF

Firmware

Firmware implements all automatic functionality of the camera, including auto exposure, white balance, flicker detection and so on. The firmware consists of drivers, generally one driver for each major control function. The firmware runs on a 6811-compatible microcontroller with a mathematical coprocessor.

Overview of Architecture

Bootstrap Routine

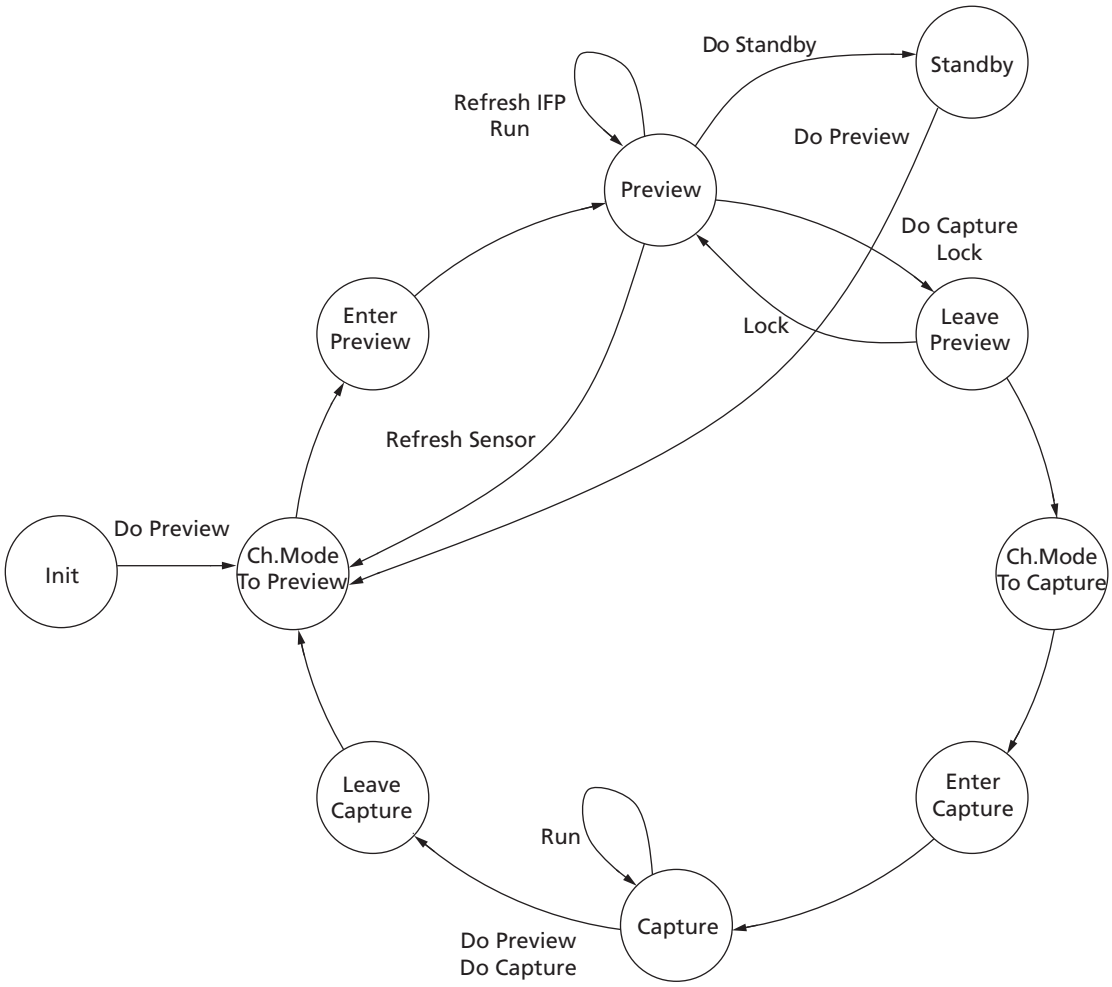
The firmware consists of a bootstrap code, drivers, and a test module. The bootstrap code initializes internal low-level MCU registers, performs a mini-self-test, and sets up a 128-bytes deep stack. The main routine is then invoked.

Drivers

Firmware applications are organized into drivers. Each driver performs a specific function. Each driver has its own data structure with various variables. The user can access the data structure through R0xC6:1 and R0xC8:1 to read or write variables. The access is implemented using hardware direct memory access (DMA). Drivers are static objects with virtual methods. Each driver has an associated data structure and a driver ID. To access driver variable, the user must know driver ID and variable offset in the data structure.

Drivers are static objects with virtual methods. Each driver has an associated data structure and a driver ID. To access a driver variable, the user must know driver ID and variable offset in the data structure.

Figure 27: Sequencer Driver



Sequencer Driver

The sequencer is a finite state machine that controls the operation of main functions and the switching between modes. Camera operation is organized as states, such as preview or capture. The sequencer carries out a number of programs, such as previewing, preview lock, capture, and so on.

The state of the sequencer is indicated in variable state. To have the sequencer execute a certain program, set the program number in cmd. The host then should monitor state to know when to change resolution and/or capture frames.

Each state has its configuration (see "Firmware Driver Variables" on page 56). Before executing programs, set up state configurations to customize the program. For example, to capture compressed frames, enable compression in capture state configuration.

A typical scenario includes the following:

1. Configure mode variables after hardware reset.
    a. Set up sensor image size for preview.
    b. Set up displayed image size.

c. Set up FIFO to smooth data rate.

2. Configure preview mode.
   a. Set auto exposure, white balance speed.
3. Execute sensor REFRESH command.
4. Run in preview until shutter button is depressed.
   a. If the shutter button is depressed halfway, execute the lock program.
   i. Configure preview state to disable necessary functions to be locked, for example, auto exposure, and white balance.
   ii. Configure PreviewLeave state for fast settling of those functions.
   iii. Execute the lock program.
   iv. The lock program transits PreviewLeave state, perform fast settling, and return to Preview state, which was configured to freeze (lock).
   b. If the shutter button is depressed all the way, execute the capture program.
   i. If already in lock mode, disable PreviewLeave state and execute capture program.
   ii. If not in lock state and some settling is required, configure PreviewLeave state for fast settling.
   iii. Execute capture program. The program transitions to PreviewLeave, perform required settling and proceed through mode change to capture.
5. Capture a frame.
   a. Preconfigure capture mode variables for correct image size, compression, and select video/still.
   b. Monitor the "state" variable. When the state is capture, grab the frame(s).

Optionally, configure Capture Enter or Preview Leave to have output blanked. This helps grabbing correct frame for capture.

Low-Light Operation

A number features are available to improve image quality in low-light conditions:

- Increased noise suppression in interpolation
- Reduced color saturation
- Reduced aperture correction
- Increased aperture correction threshold
- Y-filter

The seq.sharedParams.LLmode variable enables individual low-light features. seq.sharedParams.LLvirtGain1 and 2 specify gain thresholds to enable some of these features. Values are given by seq.sharedParams.LLInterpThresh1/2, LLSat1/2, LLApCorr1/2, and LLApThresh1/22.

Auto Exposure Driver

The AE driver works to achieve desirable exposure by adjusting sensor core's integration time, analog, and digital gains. The driver can be configured with respect to desired AE speed, maximum and minimum frame rate, the range of gains, brightness, backlight compensation, and so on.

AE driver typically runs in one of these modes. The modes are set in the sequencer driver for each sequencer state.

- Fast settling preview—reach target exposure as fast as possible
- Continuous preview—a slow-changing mode good for video capture
- Evaluative—evaluate current scene and adjust exposure for still capture

Some key variables affecting all modes:

- ae.Target— controls target exposure for all modes. Increasing or decreasing this variable makes the image brighter or darker
- ae.Gate—how accurate AE driver tracks the target exposure
- ae.weights—specify weights for central and peripheral backlight compensation in preview modes

AE driver adjusts exposure by programming sensor gains, R0x2B–R0x2E:0 and R0x41:0, sensor integration time R0x9:0 and R0xC:0 and IFP digital gains R0x4E:2 and R0x6E:1.

Evaluative Auto Exposure

Evaluative AE (EAE) selects optimum exposure for scenes where conventional AE does not give good results:

- Scenes involving sun
- Back light scenes
- Strong contrast scenes and so on

EAE breaks down input image into a 4 x 4 grid of subwindows and analyzes their exposure value (EV) readings. It evaluates brightness and contrast in the image, the scene and adjusts exposure appropriately. Variables ae.mmEVZone1/2/3/4 keep programmable thresholds defining brightness classes. Variable ae.mmShiftEV is used to calibrate EV readings for particular module type.

Mode Driver

The mode driver reduces integration efforts by managing most aspects of switching between preview (mode A) and capture (mode B) modes. It remembers vital register values for each image acquisition mode and uploads these values to the appropriate registers upon each mode switch. In addition to remembering the register data, it also creates preloaded and custom gamma and contrast tables for each mode.

To change the mode driver parameters, upload the new values to the mode driver table (ID = 7). Upon the next mode change or sequencer REFRESH (CM_REFRESH) command, these mode driver values are uploaded to the appropriate sensor and system registers, and the image processing then reflects the new values (at the beginning of the next frame acquired).

To control the output image size, upload the dimensions to the mode driver variables: output width_A, output height_A, output width_B and output height_B. The mode driver automatically applies any appropriate downsizing filter to achieve this output image size as well as updating the FIFO output size when in JPG bypass (RAW) mode. It is important so set up the imager to output an image equal to or larger than the target output image. It is also important for the crop window (crop_left_A/B and crop_right_A/B) to be equal to or larger than the target output image.

To upload a custom gamma table, upload the values to the appropriate mode driver locations (see Table 15, “Driver Variables-JPEG Driver (ID = 9),” on page 72), and select “3” (user-defined) for the gamma table type for the particular mode. If a contrast level is selected, it is applied to the user-uploaded gamma table.

The driver contains settings for raw and output image size and pan for each mode. See variables such as mode.sensor_col_width_A/B, mode.sensor_row_height_A/B and mode.fifo_width_A/B, mode.fifo_height_A/B. Blanking and readout mode— the use of skip, binning, and 1ADC modes— is configured using sensor core registers R0x5:1–R0x8:1 and R0x20:1, R0x21:1.

To change overall image brightness by changing adding luma offset at output, set

- mode.y_rgb_offset_A for preview
- mode.y_rgb_offset_B for capture

To change output format, set

- mode.output_format_A for preview
- mode.output_format_B for capture

Similarly, the user can set special effects for each mode using mode.spec_effects_A/B.

JPEG Driver

The JPEG driver programs Huffman table and quantization table memories, sets up the MT9D131 to output JPEG compressed data, and handles error checking and handshaking with the host processor.

Usage

JPEG is enabled using mode.mode_config. For example, to enable compression for capture set mode.mode_congif = 16. jpeg.qscale1/2/3, and specify the quantization factor to control compression ratio. Bigger number indicates more compression. mode.fifo* configure spoof and non-spoof output and specify FIFO output clock divider. If necessary, use jpeg.config for error handshaking with the host.

Table Programming

At power-up initialization, the JPEG driver loads standard Huffman tables into Huffman memory. Scaled versions of standard luma and chroma quantization tables and are loaded into quantization memory. The calculation of the scaled quantization table is as follows:

Scaled_Q = (scaling_factor * standard_Q + 16) >> 5

Scaling factor is bit 6:0 of JPEG driver variables qscale1, qscale2, and qscale3. The standard quantization and Huffman tables are Tables K.1–K.4 of the ISO/IEC 10918-1 Specification. Host processor can override Huffman and quantization memory with any arbitrary Huffman and quantization tables through indirect register access.

If the host chooses to use the scaled standard quantization tables, bit 4 of JPEG driver variable config must be set to “1.” Scaling factor can be changed at any time. Whenever it is changed, bit 7 of the corresponding JPEG driver variable qscale must be set to “1,” and the new value takes effect in the next frame JPEG encoding.

Since the quantization memory stores three sets (with luma and chroma information) of quantization tables, the one that is used for the current frame JPEG encoding is indicated in bit 7:6 of jpeg.config (quantization table ID). Bit 5 of jpeg.config determines who is responsible for setting the quantization table ID. If it is “0,” the host processor must program quantization table ID for every JPEG compressed frame (no need to

reprogram it if subsequent JPEG frames use the same quantization tables). If it is “1,” the JEPG driver sets quantization table ID to “0” at the start of JPEG capture (be it still or video) and automatically select next set of quantization tables for the subsequent frames when the current JPEG frame is unsuccessful.

Bit 7:6 of jpeg.config = 0, 1, and 2 indicates first, second, and third set of quantization tables, respectively.

Error Handling and Handshaking

When the capturing of a JPEG snapshot is unsuccessful, the JPEG driver can be configured to enable encoding subsequent frames until it is successful by setting bit 2 of jpeg.config to “1.” For capturing JPEG video, MT9D131 always encodes subsequent frames no matter what value bit 2 of jpeg.config is set to.

If bit 1 of jpeg.config is “1,” handshaking with the host processor at every erroneous JPEG frame is required. When JPEG status register indicates that there is an error in the current JPEG frame, JPEG encoding is stopped until the host processor sets bit 3 of jpeg.config to “1” to indicate it is ready to receive next JPEG frame. If the host processor does not respond to an erroneous JPEG frame within jpeg.timeoutFrames frames, JPEG capture is terminated. If bit 1 of jpeg.config is “0” or if there is no error in JPEG status register, no handshaking is required.

The handshaking mechanism is provided so that the host processor has sufficient time to handle the erroneous JPEG frame and react upon the error condition. For example, if the JPEG status register indicates FIFO overflow, the host processor should increase quantization value by changing quantization table scaling factor or selecting another set of the preloaded quantization tables. If spoof oversize error occurs, the host processor should increase the spoof frame size by programming registers R0x10 and R0x11 on IFP Register Page 2 and/or increase quantization value.

Start-Up and Usage

The start-up sequence consists of the following:

1. Power-up
2. Hardware reset
3. Configure and enable PLL
4. Configure pad slew rate
5. Configure preview mode
6. Configure capture mode
7. Perform lock or capture

To start the part, power up power supplies, provide an input clock, and perform a hardware reset.

Power-Up

There are no specific requirements to the order in which different supplies are turned on. Once the last supply is stable within the valid ranges specified below, follow the hard reset sequence to complete the power-up sequence. To minimize leakage current, all the power supplies should be turned on at the same time

.

Analog Voltage:	2.8V for best image performance
Digital Voltage:	1.8V ±0.1V (1.7V–1.9V)
I/O Voltage:	1.7V–3.1V

Power-Down

Before powering down the sensor, it is recommended to bypass the PLL. Once this is completed, the input clock (EXTCLK) can be turned off. After EXTCLK is off, turn off all the power supplies as shown in Figure 28 on page 106. RESET_BAR should also be LOW at this time.

Note: Turning the power supplies off cannot be used as a method of achieving low power consumption. Power to the sensor needs to be provided as long as the system is active. For the lowest power consumption, refer to the standby procedure.

Hard Reset Sequence

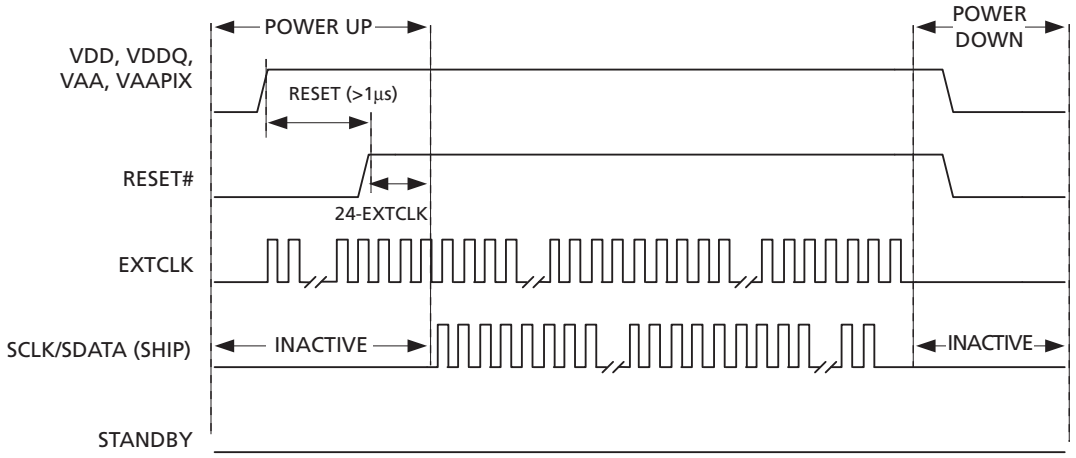
After power-up, a hard reset is required. Assuming all supplies are stable, the assertion of RESET_BAR (active LOW) sets the device in reset mode. The clock is required to be active when RESET_BAR is released. Hence, leaving the input clock running during the reset duration is recommended. After 24 clock cycles (EXTCLK), the two-wire serial interface is ready to accept commands. Reset should not be activated while STANDBY is asserted.

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

1. Wait for all power supplies to be stable and within specification.
2. Supply the sensor with an input clock.
3. Assert RESET_BAR (active LOW) for at least 1µs.
4. De-assert RESET_BAR (input clock must be running).
5. Wait 24 clock cycles before using the two-wire serial interface.

Refer to Figure 28 for the timing diagram.

Figure 28: Power On/Off Sequence



Notes:
1. For a safe RESET to occur, EXTCLK should be running during RESET with STANDBY LOW, as shown in the sequence above.
2. After RESET_BAR is HIGH, wait 24 EXTCLK rising edges before the two-wire serial interface communication is initiated.
3. After the power-up sequence, the preview state is reached when the firmware variable seq.state (ID = 1, Offset = 4) is equal to 3. This transition time varies depending on the input clock frequency and scene conditions.
4. To go into the firmware standby state, go to capture mode (also known as context B), or execute the firmware REFRESH/REFRESH_MODE commands after the power-up sequence (the preview state [seq.state = 3] must be reached first).

Soft Reset Sequence

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

1. Bypass the PLL, R0x65:0 = 0xA000, if it is currently used
2. Perform MCU reset by setting R0xC3:1 = 0x0501
3. Enable soft reset by setting R0x0D:0 = 0x0021. Bit 0 is used for the sensor core reset while bit 5 refers to SOC reset.
4. Disable soft reset by setting R0x0D:0 = 0x0000
5. Wait 24 clock cycles before using the two-wire serial interface

Note: No access to MT9D131 registers—both page 1 and page 2—is possible during soft reset.

Enable PLL

Since the input clock frequency is unknown, the part starts with PLL disabled. The default MNP values are for 10 MHz, with 80 MHz as target. For other frequencies, calculate and program appropriate M, N, and P values. PLL programming and power-up sequence is as follows:

1. Program PLL frequency settings, R0x66–R0x67:0
2. Power up PLL, R0x65:0[14] = 0
3. Wait for PLL settling time >150µs
4. Turn off PLL bypass, R0x65:2[15] = 0

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

Note: Until PLL is enabled the two-wire serial interface may be limited in speed. After PLL is enabled, the two-wire serial interface master can increase its communication speed.

Configure Pad Slew

Program the desired slew rate for DOUT, PIXCLK, FRAME_VALID, and LINE_VALID at the variables, mode.fifo_conf1/2_A/B. Program R0xA:1 with desired slew rate for two-wire serial interface SDATA and SCLK.

Configure Preview Mode

The default preview image size is 800 x 600, running at up to 30 fps at 80 MHz internal clock. To change the default size, program mode driver variables mode.output_width_A and mode.output_height_A and issue a REFRESH command, seq.cmd = 5.

For example, to configure 160x120 LCD RGB preview, program

- mode.output_width_A = 160
- mode.output_width_B = 120
- mode.out_format_A = 0x20
- seq.cmd = 5

Preview contrast, brightness, gamma, frame rate, and many other parameters can be loaded here as well. If known at this time, the user can also program capture parameters. If necessary, set up the AE/WB lock command here.

Configure Capture Mode

Program the mode and other drivers, when desired capture parameters (video, still, compression, and resolution) are known.

Perform Lock or Capture

See "Sequencer Driver" on page 100 for details.

Standby Sequence

Standby mode can be activated by two methods. The first method is to assert STANDBY, which places the chip into hard standby. Turning off the input clock (EXTCLK) reduces the standby power consumption to the maximum specification of 100µA at 55°C. There is no serial interface access for hard standby.

The second method is activated through the serial interface by setting R0xD:0[2] = 1 to the register, known as the soft standby. As long as the input clock remains on, the chip allows access through the serial interface in soft standby.

Standby should only be activated from the preview mode (context A), and not the capture mode (context B). In addition, the PLL state (off/bypassed/activated) is recorded at the time of firmware standby (seq.cmd = 3) and restored once the camera is out of firmware standby. In both hard and soft standby scenarios, internal clocks are turned off and the analog circuitry is put into a low power state. Exit from standby must go through the same interface as entry to standby. If the input clock is turned off, the clock must be restarted before leaving standby. If the PLL is used to generate the master clock, ensure that the PLL is powered down during standby because it uses a relatively high amount of power. By default, R0x65:0[13] powers down the PLL when the chip enters standby mode. Turn on the PLL bypass (R0x65:0[15] = 1) to prepare the PLL for standby.

To Enter Standby

1. Preparing for standby
   a. Issue the STANDBY command to the firmware by setting seq.cmd = 3
   b. Poll seq.state until the current state is in standby (seq.state = 9)
   c. Bypass the PLL if used by setting R0x65:0[15] = 1
2. Preventing additional leakage current during standby
   a. Set R0xA:1[7] = 1 to prevent elevated standby current. It controls the bidirectional pads DOUT, LINE_VALID, FRAME_VALID, and PIXCLK.
   b. If the outputs are allowed to be left in an unknown state while in standby, the current can increase. Therefore, either have the receiver hold the camera outputs HIGH or LOW, or allow the camera to drives its outputs to a known state by setting R0xD:0[6] = 1, R0xD:0[4] needs to remain at the default value of “0.” In this case, some pads are HIGH while some are LOW. For dual camera systems, at least one camera has to be driving the bus at any time so that the outputs are not left floating.
3. Putting the camera in standby
   a. Assert STANDBY = 1. Optionally, stop the EXTCLK clock to minimize the standby current specified in the MT9D131 data sheet. For soft standby, program standby R0xD:0[2] = 1 instead.

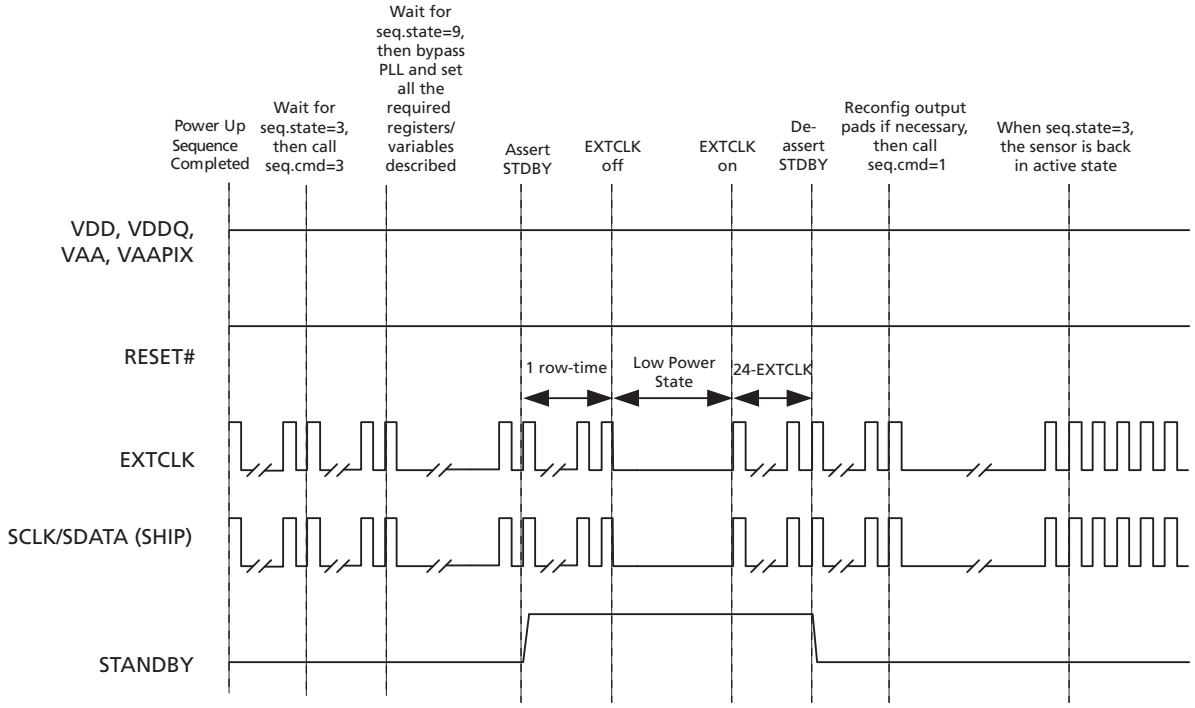
To Exit Standby

1. De-assert standby
   a. Provide EXTCLK clock, if it was disabled when using STANDBY
   b. De-assert STANDBY = 0 if hard standby was used. Or program R0xD:0[2] = 0 if soft standby was used
2. Reconfiguring output pads
3. Issue a GO_PREVIEW command to the firmware by setting seq.cmd = 1
4. Poll seq.state until the current state is preview (seq.state = 3)

The following timing requirements should be met to turn off EXTCLK during hard standby:

1. After asserting standby, wait 1 row time before stopping the clock
2. Restart the clock 24 clock cycles before de-asserting standby

Figure 29: Hard Standby Sequence



Standby Hardware Configuration

While in standby, floating IO signals may cause the standby current to rise significantly. Therefore, it is recommended that the following signals be maintained HIGH or LOW during standby and not floating: DOUT[7:0], PIXCLK, FRAME_VALID, and LINE_VALID.

Output Enable Control

When the sensor is configured to operate in default mode, the DOUT, FRAME_VALID, LINE_VALID, and PIXCLK outputs can be placed in High-Z under hardware or software control, as shown in Table 29 on page 110.

Table 29: Output Enable Control

Standby	R0x0D:0[4] (output_dis)	R0x0D:0[6] (drive_pins)	Output State
0	0 (default)	0 (default)	Driven
1	0 (default)	0 (default)	High-Z
"Don't Care"	0 (default)	1	Driven
"Don't Care"	1	"Don't Care"	High-Z

The pin transition between driven and High-Z always occurs asynchronously. Output-enable control is provided as a mechanism to allow multiple sensors to share a single set of interface pins with a host controller.

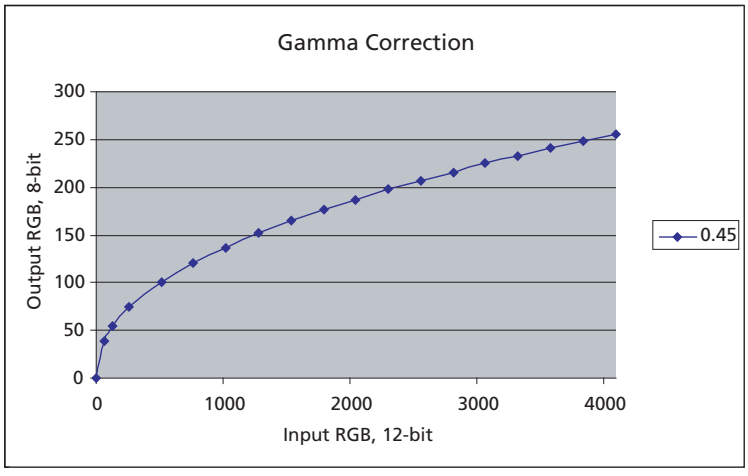
There is no benefit in placing the pins in a High-Z while the part is in its low-power standby state. Therefore, in single-sensor applications that use STANDBY to enter and leave the standby state, programming R0x0D:0[6] = 1 is recommended.

Contrast and Gamma Settings

The MT9D131 IFP includes a block for gamma and contrast correction. A custom gamma/contrast correction table may be uploaded, or pre-set gamma and contrast settings may be selected.

The gamma and contrast correction block uses the following 12-bit input data points to form a piecewise linear transformation curve: 0, 64, 128, 256, 512, 768, 1024, 1280, 1536, 1792, 2048, 2304, 2560, 2816, 3072, 3328, 3584, 3840, and 4095. These input points have been selected to provide more detail to the low end of the curve where gamma correction changes are typically the greatest. These points correspond to 8-bit output values that can be uploaded to the appropriate registers.

Figure 30: Gamma Correction Curve



For simplicity, predefined gamma and contrast tables may be selected, and the MT9D131 automatically combines these tables and upload them to the appropriate gamma correction registers.

The gamma and contrast tables may be selected at mode driver (ID = 7) offsets 67 and 68 (decimal) for mode A and mode B, respectively. The gamma settings are established at bits 0–2, and the contrast settings are established at bits 4–6.

The gamma setting values are shown in Table 30.

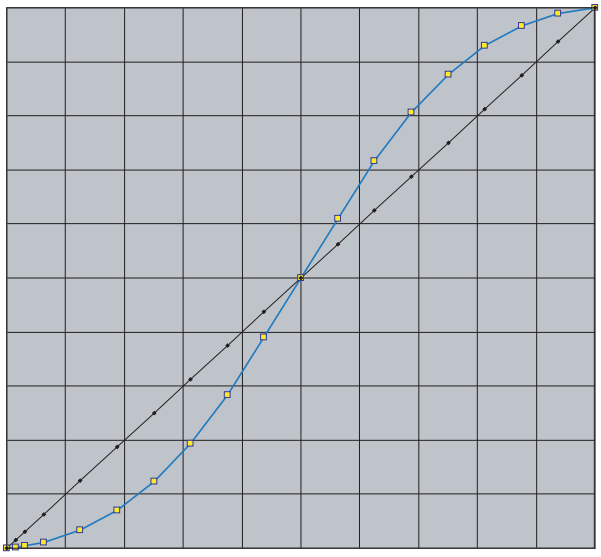
Table 30: Gamma Settings

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

S-Curve

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

Figure 31: Contrast “S” Curve



The contrast values are shown in Table 31 on page 112.

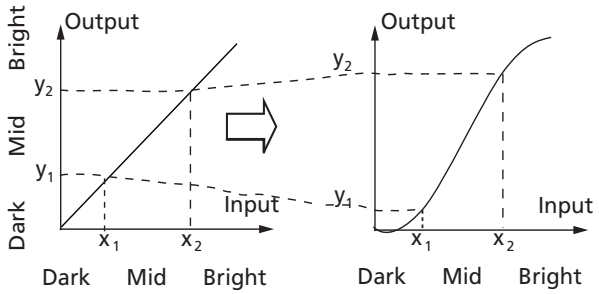
Table 31: Contrast Values

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

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

S-curve is a function to correct image pixel values. When applied to pixel values, it typically compresses dark and bright tones, while stretching the midtones. Figure 32 shows how input tone range is remapped to output tone range. Categorize input pixel values as dark, midlevel, and bright. When no S-curve is applied, pixel values are unchanged. That is, dark tones 0...x_1 map to same dark tones 0...$x_1 = y_1$, and midlevel maps to identical midlevel $x_1 = x_2$, and bright maps to identical bright. When an S-curve is applied, the mapping is changed. In particular, midtones are stretched $(y_1 - y_2) > (x_1 - x_2)$, causing increase of contrast. Here $(y_1 - y_2) / (x_1 - x_2)$ is a measure of contrast. Value of 1 corresponds to no change; >1 and <1 indicate contrast increase and decrease, respectively. Dark tones are compressed, $y_1 < x_1$, causing suppression of noise.

Figure 32: Tonal Mapping



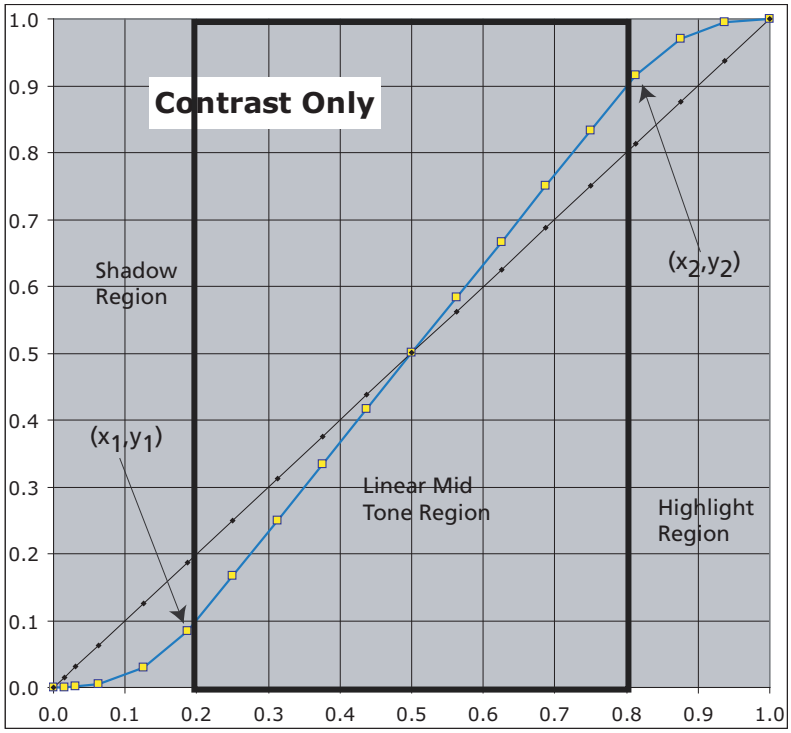
The contrast settings on the MT9D131 are implemented by applying an S-curve function on to the data points of the gamma curve. These resulting points then replace the existing gamma curve points, and the image is processed using this new contrast-enhanced gamma curve. Identical S-curve is applied to all three RGB components.

The S-curve is created by joining a linear midsection (often with a slope greater than one) to curved sections for highlights and shadows. The following constraints apply to the overall curve to ensure a smooth curve appropriate for increased contrast:

1. The first/lowest point must be (0,0) and the last/highest point must be (1,1) (normalized).
2. The midsection and each of the end-curves must intersect on the same points.
3. The slope of the midsection and each of the end-curves must be equal at the intersection points.

The midsections a simple linear function of the form: y = mx + b. Each of the end-curves (shadow and highlights) must be a trinomial to satisfy all boundary conditions.

Figure 33: Contrast Diagram



Auto Exposure

Two types of auto exposure are available—preview and evaluative.

Preview

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

The driver changes AE parameters (IT, Gains, and so on) to drive brightness (ae.CurrentY) to programmable target (ae.Target). Value of one step approach to target is defined by ae.JumpDivisor variable. Expected brightness is

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

To avoid unwanted reaction of AE on small fluctuations of scene brightness or momentary scene changes, the AE driver uses temporal filter for luma and gate around AE luma target. The driver changes AE parameters only if buffered luma outsteps AE target gates. Variable ae.lumaBufferSpeed defines buffering level.

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

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

Evaluative Algorithm

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.

Exposure Control

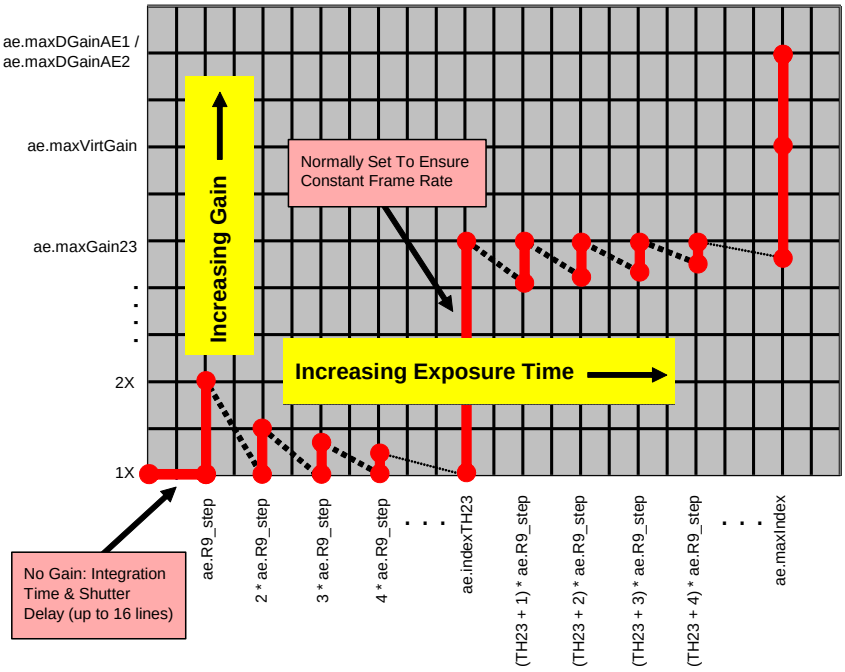
To achieve the required amount of exposure, the AE driver adjusts the sensor integration time R0x9:0, R0xC:0, gains, ADC reference, and IFP digital gains. To reject flicker, integration time is typically adjusted in increments of ae.R9_step. ae.R9_step specifies duration in row times equal to one flicker period. Thus, flicker is rejected if integration time is kept a natural factor of the flicker period.

Exposure is adjusted differently depending on illumination situation.

- In extremely bright conditions, the exposure is set using R0xC:0, R0x9:0, and analog gains. R0xC:0 is used to achieve very short integration times. In this situation, R0x9:0<ae.R9_step and flicker are not rejected.
- In bright conditions where R0x9:0> = ae., R9_step R0x9:0 is set as a natural factor of ae.R9_step. Analog gains are also used, but the green gain, also called virtual gain, does not exceed 2x. ae.minVirtGain limits minimal integration time and is expressed in flicker periods. ae.Index indicates the current integration time expressed in the same form.
- Under medium-intensity illumination, the integration time can increase further. For any given exposure, the best signal-to-noise ratio can be typically obtained by using the longest exposure and the smallest gain setting. However, a long exposure time can slow down the output frame rate if the former exceeds the default frame rate, R0x9:0 > R0x3:0 + R0x6:0 + 1. Integration ae.IndexTH23 specifies the breakpoint where AE scheme, giving preference to increasing the shutter width, is replaced with another scheme giving preference to increase in gain. ae.maxGain23 specifies maximum allowed gain in this situation. ae.VirtGain indicates current green channel gain.
- In darker situations, the gain achieves ae.maxGain23 and the integration time is allowed to increase again up to ae.maxIndex.
- In yet darker situations, once the integration time achieves ae.maxIndex, the analog gain is allowed to increase up to ae.maxVirtGain.
- In very dark conditions, the digital IFP gains are allowed to increase up to ae.maxDGainAE1 and ae.maxDGainAE2.

ADC is used as an additional gain stage by adjusting reference levels. See ae.ADC* variables.

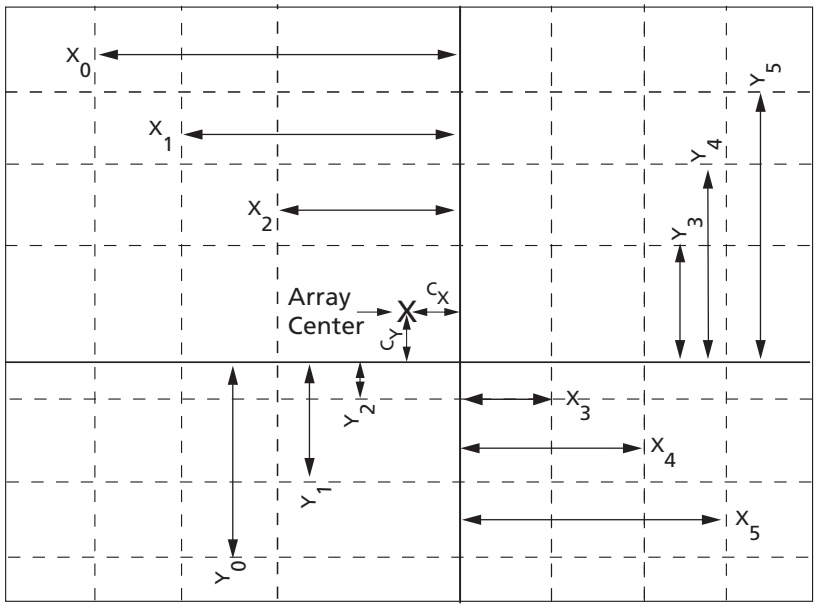
Figure 34: Gain vs. Exposure



Lens Correction Zones

To increase the precision of the correction function, the image plane is divided into 8 zones in each dimension. The coordinates of zone boundaries are referenced with respect to the lens center, C. Each boundary as well as C (Cx, Cy) coordinate is stored as a byte, which represents the coordinate value divided by 4. There are always three boundaries to the left (top) of the center and three to the right (bottom) of the center. These boundaries apply uniformly for each color channel. However, the correction functions are programmable independently for each color component. Boundary and lens center positions are also valid for the preview mode. Figure 35 on page 116 illustrates the lens correction zones.

Figure 35: Lens Correction Zones



Lens Correction Procedure

The goal of the lens shading correction is to achieve a constant sensitivity across the entire image area after the correction is applied. To accomplish this, each incoming pixel value is multiplied with a correction function F (x, y), which is dependent on the location of the pixel. The corrected pixel data, P_{OUT}, can be expressed in the following term:

$$P_{OUT}(x,y) = P_{IN}(x,y) * F(x,y)$$

(EQ 1)

Within each zone described above, the correction function can be expressed with a following equation, as follows:

$$F(x,y) = \phi(x,x^2) + \varphi(y,y^2) + k*\phi(x,x^2)*\varphi(x,x^2)*\varphi(y,y^2) - G$$

(EQ 2)

where

$\phi(x,x^2)$ and $\varphi(y,y^2)$ are piecewise quadratic polynomial functions that are independent in each x and y dimension, and k and G are constants that can be used to increase lens correction at the image corners (k) and to offset the LC magnitude for all zones and colors (G).

The expressions
$\phi(x, x^2)$ *and* $\varphi(y, y^2)$ are defined independently for each dimension. These can be expressed further as:

$$\phi(x, x^2) = a_i x^2 + b_i x + c_i \quad \text{(EQ 3)}$$

and

$$\varphi(y, y^2) = d_i y^2 + e_i y + f_i \quad \text{(EQ 4)}$$

To implement the function F (x, y) for each zone, the MT9D131 provides a set of registers that allow flexible definition of the function F (x, y). These registers contain the following parameters:

- Operation mode R0x80:2
- Zone boundaries and center offset R0x81:2–R0x87:2
- Initial conditions of:

  $\phi(x, x^2)$ *and* $\varphi(y, y^2)$

  for each color (12-bit wide) R0x88:2–R0x8D:2
- Initial conditions of the first derivative of:

  $\phi(x, x^2)$ *and* $\varphi(y, y^2)$

  for each color (12-bit wide) R0x8E:2–R0x93:2
- The second derivatives of:

  $\phi(x, x^2)$ *and* $\varphi(y, y^2)$

- The second derivatives of

for each color and each zone (8-bit wide) R0x94:2–R0xAB:2

- Values for k(10-bit wide) and G(8-bit wide) in (2) for all colors and zones are specified in R0xAD:2 and R0xAE:2. Sign for k is specified in R0x80:2.

There are x2 factor that can be applied to the second derivatives inside each zone (R0xAC:2) if correction curvature in the particular zone is to be increased. There are also global x2 factors for all zones in X and Y direction located in R0x80:2. The first derivative (for both X and Y directions) can be divided by a number (devisor) specified in R0x80:2 before it is applied to the F function. Higher numbers result in more precise curve (full-scale expanse).

Color Correction

Color correction in the color pipeline is achieved by:

1. Apply digital gain to raw RGB, R0x6A:1–R0x6E:1
2. Subtract second black level D from raw RGB, see R0x3B:1
3. Multiply result by CCM, R0x60:1–R0x66:1
4. Clip the result

$$\begin{vmatrix} R \\ G \\ B \end{vmatrix} = \text{CLIP}\left(\begin{vmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{vmatrix} \bullet \begin{vmatrix} R_{raw} * G_R - D \\ G_{raw} * G_G - D \\ B_{raw} * G_B - D \end{vmatrix} \right)$$

MCU controls both CCM and D.

Decimator

To fit image size to customer needs, the image size of the SOC can be scaled down. The decimator can reduce image to arbitrary size using filtering. The scale-down procedure is performed by transferring an incoming pixel from the image space into a decimated (scaled down) space. The procedure can be performed in both X and Y dimensions. All the standard formats with resolution lower than 2 megapixels as well as customer-specified resolutions are supported.

Transfer of pixel from the image into the decimated space is done in the following way, which is the same in X and Y directions:

Each incoming pixel is split in two parts. The first part (P1) goes into the currently formed output-space pixel while part two (P2) goes to the next pixel. P1 could be equal or less than value of the incoming pixel while P2 is always less.

These two parts are obtained by multiplying the value of incoming pixel by scaling factors f1 and f2, which sum is always constant for the given decimation degree and proportional to X1/X0 where X1 is size of the output image and X0 is the size of the input image. It is denoted as “decimation weight.”

Coefficients f1 and f2 are calculated in the microcontroller transparently for user based on the specified output image size and mode of SOC operation.

At large decimation degrees, several incoming pixels may be averaged into one decimated pixel. Averaging of the pixels during decimation provides a low pass filter, which removes high-frequency components from the incoming image, and thus avoids aliasing in the decimated space.

The decimator has two operational modes—normal and high-precision. Since the intermediate result for Y decimated pixels has to be stored in a memory buffer with certain word width, there is a need for additional precision at larger decimation degrees when scaling factors are small. This is done by increasing the number of digits for each stored value when decimation is greater than 2.

Spectral Characteristics

Figure 36: Typical Spectral Characteristics

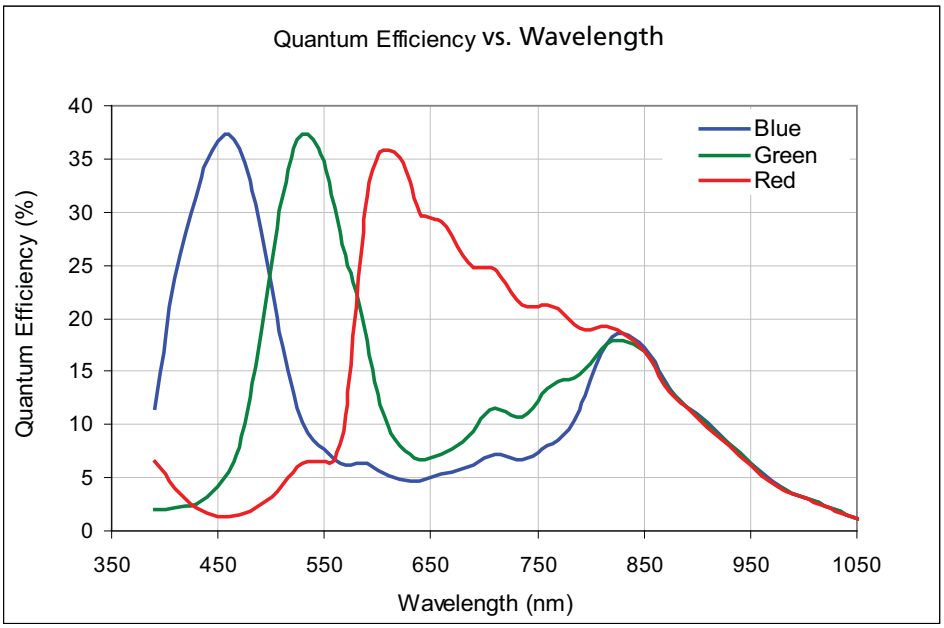
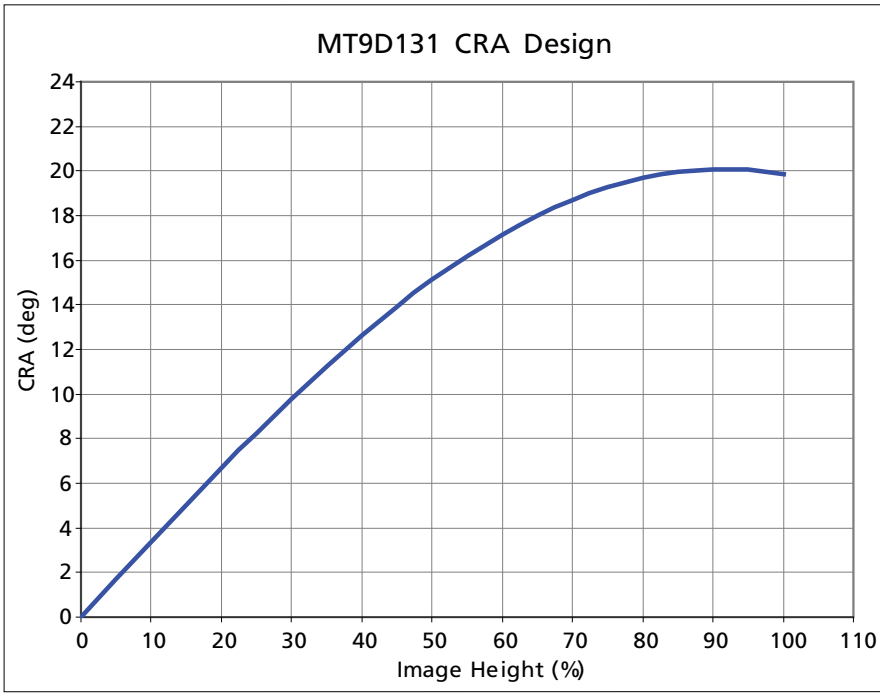


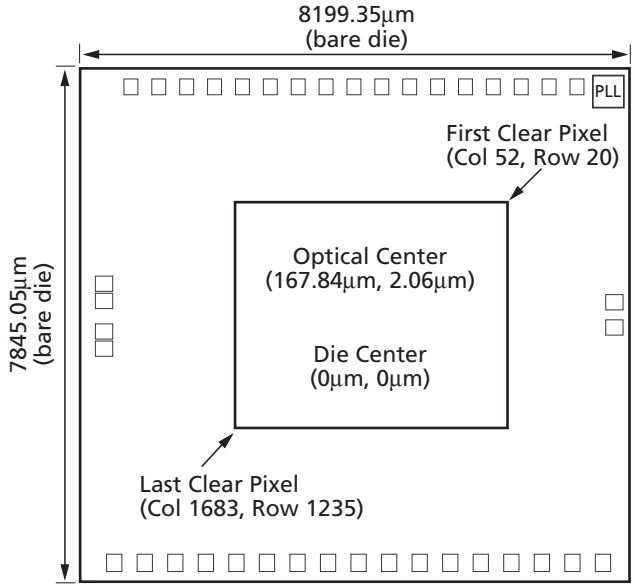
Figure 37: Chief Ray Angle (CRA) vs. Image Height



CRA vs. Image Height Plot	Image Height		CRA
	(%)	(mm)	(deg)
	0	0	0
	5	0.140	1.68
	10	0.280	3.37
	15	0.420	5.03
	20	0.560	6.67
	25	0.700	8.25
	30	0.840	9.79
	35	0.980	11.25
	40	1.120	12.63
	45	1.260	13.92
	50	1.400	15.11
	55	1.540	16.19
	60	1.680	17.15
	65	1.820	17.98
	70	1.960	18.69
	75	2.100	19.26
	80	2.240	19.68
	85	2.380	19.96
	90	2.520	20.08
	95	2.660	20.05
	100	2.800	19.87

Die Outline

Figure 38: Optical Center Offset



Note: Figure not to scale.

Electrical Specifications

Recommended die operating temperature range is from –20° to +55°C. The sensor image quality may degrade above +55°C.

Table 32: AC Electrical Characteristics

Parameter	Symbol	Conditions	Min	Type	Max	Units
Input clock frequency	$f_{EXTCLK1}$	PLL enabled (MCLK max = 80 MHz)	6	10	64	MHz
Input clock period	$t_{EXTCLK1}$	PLL enabled (MCLK max = 80 MHz)	15.625	100	166.7	ns
Input clock frequency	$f_{EXTCLK2}$	PLL disabled	6		80	MHz
Input clock period	$t_{EXTCLK2}$	PLL disabled	12.5		166.7	ns
Input clock rise time	t_R		0.5		1	V/ns
Input clock fall time	t_F		0.5		1	V/ns
Clock duty cycle			40	50	60	%
PIXCLK frequency	f_{PIXCLK}	Default	6		80	MHz
PIXCLK to data valid	t_{PD}	Default	-3		3	ns
PIXCLK to FV HIGH	t_{PFH}	Default	-3		3	ns
PIXCLK to LV HIGH	t_{PLH}	Default	-3		3	ns
PIXCLK to FV LOW	t_{PFL}	Default	-3		3	ns
PIXCLK to LV LOW	t_{PLL}	Default	-3		3	ns
Input pin capacitance	C_{IN}			3.5		pF
Load capacitance	C_{LOAD}			15	20	pF
AC Setup Conditions						
$f_{EXTCLK1}$			6		64	MHz
V_{DD}			1.7	1.8	1.95	V
V_{DDQ}			1.7	2.8	3.1	V
V_{AA}			2.5	2.8	3.1	V
VAAPIX			2.5	2.8	3.1	V
V_{DD}PLL			2.5	2.8	3.1	V
Output load				15		pF

Table 33: DC Electrical Definitions and Characteristics

Parameter	Symbol	Conditions	Min	Typ	Max	Units	Note
Core digital voltage	V_{DD}		1.7	1.8	1.95	V	
I/O digital voltage	V_{DDQ}		1.7	2.8	3.1	V	
Analog voltage	V_{AA}		2.5	2.8	3.1	V	
Pixel supply voltage	VAAPIX		2.5	2.8	3.1	V	
PLL supply voltage	V_{DDPLL}		2.5	2.8	3.1	V	
Input high voltage	V_{IH}	V_{DDQ} = 2.8V	2.4		V_{DDQ}+0.3	V	
		V_{DDQ} = 1.8V	1.4		V_{DDQ}+0.3		
Input low voltage	V_{IL}	V_{DDQ} = 2.8V	GND-0.3		0.8	V	
		V_{DDQ} = 1.8V	GND-0.3		0.5		
Input leakage current	I_{IN}	No pull-up resistor; V_{IN} = V_{DDQ} or D_{GND}	–10	+/-0.5	10	µA	
Output high voltage	V_{OH}	At specified I_{OH}	V_{DDQ}–0.4			V	
Output low voltage	V_{OL}	At specified I_{OL}			0.4	V	
Output high current	I_{OH}	At specified V_{OH} = V_{DDQ}~400mV at 1.7V V_{DDQ}			–7	mA	
Output low current	I_{OL}	At specified V_{OL}~400MV at 1.7V V_{DDQ}			7	mA	
Tri-state output leakage current	I_{OZ}	V_{IN} = V_{DDQ} or GND	–10	+/-0.5	10	µA	
Digital operating current	I_{DD}1	Context B, 1600 x 1200, JPEG on, MCLK = MAX, PIXCLK = MAX		92	110	mA	
I/O digital operating current	I_{DDQ}1	Context B, 1600 x 1200, JPEG on, MCLK = MAX, PIXCLK = MAX		15		mA	1
Analog operating current	I_{AA}1	Context B, 1600 x 1200, JPEG on, MCLK = MAX, PIXCLK = MAX		43	55	mA	
Pixel supply current	IAAPIX1	Context B, 1600 x 1200, JPEG on, MCLK = MAX, PIXCLK = MAX		2.1	3.5	mA	
PLL supply current	IDDPLL1	Context B, 1600 x 1200, JPEG on, MCLK = MAX, PIXCLK = MAX		2.3	3	mA	
Digital operating current	I_{DD}2	Context A, 800 x 600, No JPEG, MCLK = MAX, PIXCLK = MAX		48	60	mA	
I/O digital operating current	I_{DDQ}2	Context A, 800 x 600, No JPEG, MCLK = MAX, PIXCLK = MAX		15		mA	1
Analog operating current	IAA2	Context A, 800 x 600, No JPEG, MCLK = MAX, PIXCLK = MAX		27	35	mA	
Pixel supply current	IAAPIX2	Context A, 800 x 600, No JPEG, MCLK = MAX, PIXCLK = MAX		4	5.5	mA	
PLL supply current	IDDPLL2	Context A, 800 x 600, No JPEG, MCLK = MAX, PIXCLK = MAX		2.3	3	mA	
Standby current PLL enabled	ISTDBY1	PLL enabled (MCLK = 0Hz, held at either V_{IL} or V_{IH})		35	100	µA	
Standby current PLL disabled	ISTDBY2	PLL disabled (MCLK = 0Hz, held at V_{IL} or V_{IH})		35	100	µA	

Notes: 1. Due to the influence of several variables (scene illumination, output load) maximum values are not available.

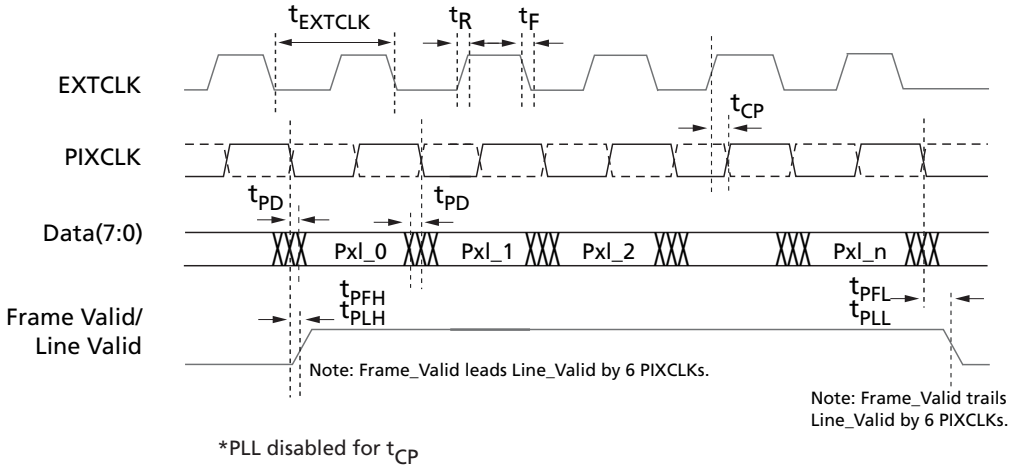
Table 34: Absolute Maximum Ratings

Symbol	Parameter	Min	Max	Unit
V_{DD}	Digital power	–0.3	2.4	V
V_{DDQ}	I/O power	–0.3	4.0	V
V_{DDPLL}	PLL power	–0.3	4.0	V
V_{AA}	Analog power (2.8V)	–0.3	4.0	V
VAAPIX	Pixel array power	–0.3	4.0	V
V_{IN}	DC input voltage	–0.3	V_{DDQ} + 0.3	V
V_{OUT}	DC output voltage	–0.3	V_{DDQ} + 0.3	V
T_{OP}	Operation temperature	–30	70	°C
T_{STG}[1]	Storage temperature	–40	85	°C

Note: [1]Stresses above those listed may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other conditions above those indicated in the product specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

I/O Timing

Figure 39: I/O Timing Diagram



Appendix A: Two-Wire Serial Register Interface

This section describes the two-wire serial interface bus that can be used in any functional sensor mode.

The two-wire serial interface bus enables R/W access to control and status registers within the sensor core.

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. The master is responsible for driving a valid logic level on SCLK at all times. Data is transferred between the master and the slave on a bidirectional signal (SDATA). Both the SDATA AND SCLK signal are pulled up to VDD off-chip by a 1.5KΩ resistor. Either the slave or master device can drive the SDATA line low—the interface protocol determines which device is allowed to drive the SDATA line at any given time.

Protocol

The two-wire serial interface bus defines the transmission codes as follows:

- a start bit
- the slave device 8-bit address
- a(an) (no) acknowledge bit
- an 8-bit message
- a stop bit

Sequence

A typical read or write sequence is executed as follows:

1. The master sends a start bit.
2. The master sends the 8-bit slave device address. The last bit of the address determines if the request is a read or a write, where a “0” indicates a write and a “1” indicates a read.
3. The slave device acknowledges receipt of the address by sending an acknowledge bit to the master.
4. If the request is a write, the master then transfers the 8-bit register address, indicating where the write takes place.
5. The slave sends an acknowledge bit, indicating that the register address has been received.
6. The master then transfers the data, 8 bits at a time, with the slave sending an acknowledge bit after each 8 bits.

The sensor core uses 16-bit data for its internal registers, thus requiring two 8-bit transfers to write to one register. After 16 bits are transferred, the register address is automatically incremented so that the next 16 bits are written to the next register address. The master stops writing by sending a start or stop bit.

A typical read sequence is executed as follows.

1. The master sends the write-mode slave address and 8-bit register address, just as in the write request.
2. The master then sends a start bit and the read-mode slave address, and clocks out the register data, 8 bits at a time.
3. The master sends an acknowledge bit after each 8-bit transfer. The register address is auto-incremented after every 16 bits is transferred.
4. The data transfer is stopped when the master sends a no-acknowledge bit.

Bus Idle State

The bus is idle when both the data and clock lines are high. Control of the bus is initiated with a start bit, and the bus is released with a stop bit. Only the master can generate start and stop bits.

Start Bit

The start bit is defined as a HIGH-to-LOW data line transition while the clock line is HIGH.

Stop Bit

The stop bit is defined as a LOW-to-HIGH data line transition while the clock line is HIGH.

Slave Address

The 8-bit address of a two-wire serial interface device consists of 7 bits of address and 1 bit of direction. A "0" in the LSB (least significant bit) of the address indicates write mode, and a "1" indicates read mode. The default slave addresses used by the sensor core are 0xBA (write address) and 0xBB (read address). R0x0D:0[10] or the SADDR pin can be used to select the alternate slave addresses 0x90 (write address) and 0x91 (read address).

Writes to R0x0D:0[10] are inhibited when the STANDBY signal is asserted (all other writes proceed normally). This allows two sensors to co-exist as slaves on this interface, but they must be addressed independently. Enable this capability as follows:

After RESET, both sensors use the default slave address. READS or WRITES on the serial register interface to the default slave address are decoded by both sensors simultaneously.

1. After RESET, assert the STANDBY signal to one sensor and negate the STANDBY signal to the other sensor.
2. Perform a write to R0x0D:0 with bit 10 set. The sensor with STANDBY asserted ignores the write to bit 10 and continues to decode at the default slave address.

The sensor with STANDBY negated has its R0x0D:0[10] set and responds to the alternate slave address for all subsequent READ and WRITE operations, as shown in Table 35.

Table 35: Slave Address Options

SADDR	R0xD:0[10]	Slave Address	
		WRITE	READ
0	0	0x090	0x091
0	1	0x0BA	0x0BB
1	0	0x0BA	0x0BB
1	1	0x090	0x091

Data Bit Transfer

One data bit is transferred during each clock pulse. The serial interface clock pulse is provided by the master. The data must be stable during the high period of the two-wire serial interface clock—it can only change when the serial clock is LOW. Data is transferred 8 bits at a time, followed by an acknowledge bit.

Acknowledge Bit

The master generates the acknowledge clock pulse. The transmitter (which is the master when writing, or the slave when reading) releases the data line, and the receiver indicates an acknowledge bit by pulling the data line LOW during the acknowledge clock pulse.

No-Acknowledge Bit

The no-acknowledge bit is generated when the data line is not pulled down by the receiver during the acknowledge clock pulse. A no-acknowledge bit is used to terminate a read sequence.

Page Register

The MT9D131 two-wire serial interface and its associated protocols support an address space of 256 16-bit locations. This address space is extended by a 3-bit page prefix, and controlled through accesses to R0xF0:0.

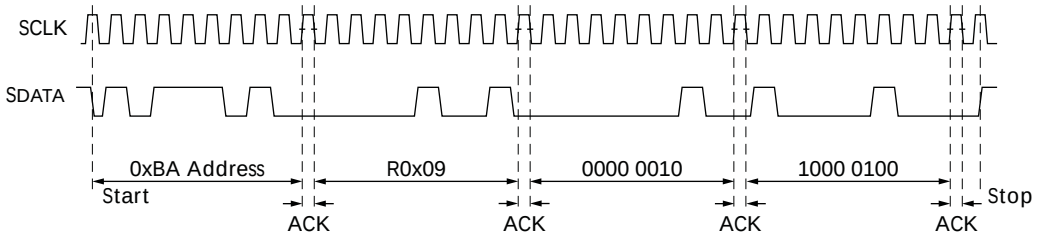
The paging mechanism is intended to allow access to other sets of registers when the sensor is embedded as part of a more complex integrated subsystem, for example, in an SOC. All registers within the sensor core are accessible on page 0 (the default page).

Sample Write and Read Sequences

16-Bit Write Sequence

A typical write sequence for writing 16 bits to a register is shown in Figure 40. A start bit given by the master starts the sequence, followed by the write address. The image sensor then sends an acknowledge bit and expects the register address to come first, followed by the 16-bit data. After each 8-bit transfer, the image sensor sends an acknowledge bit. All 16 bits must be written before the register is updated. After 16 bits are transferred, the register address is automatically incremented so that the next 16 bits are written to the next register. The master stops writing by sending a start or stop bit.

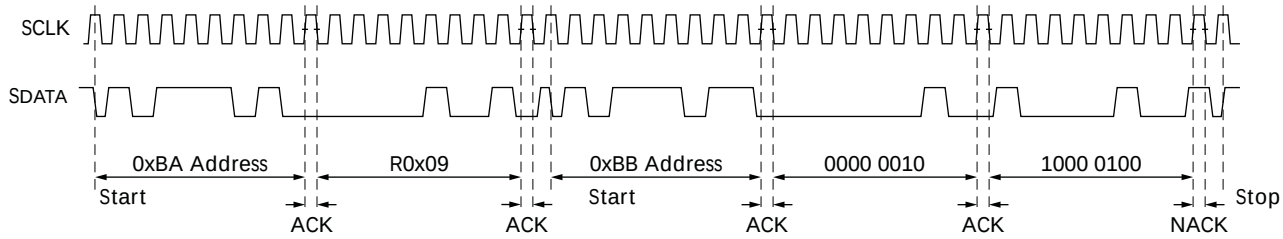
Figure 40: WRITE Timing to R0x09:0—Value 0x0284



16-Bit Read Sequence

A typical read sequence is shown in Figure 41 on page 128. First the master writes the register address, as in a write sequence. Then a start bit and the read address specify that a read is about to happen from the register. The master clocks out the register data, eight bits at a time. The master sends an acknowledge bit after each 8-bit transfer. The register address should be incremented after every 16 bits is transferred. The data transfer is stopped when the master sends a no-acknowledge bit.

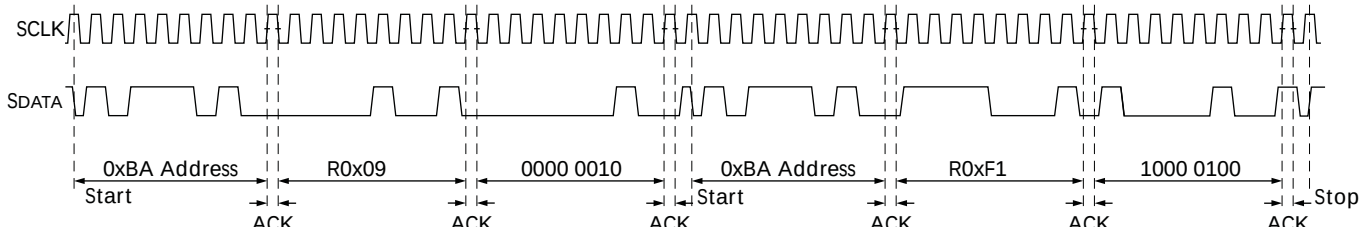
Figure 41: READ Timing from R0x09:0; Returned Value 0x0284



8-Bit Write Sequence

To be able to write 1 byte at a time to the register, a special register address is added. The 8-bit write is done by writing the upper 8 bits to the desired register, then writing the lower 8 bits to the special register address (R0xF1:0). The register is not updated until all 16 bits have been written. It is not possible to update just half of a register. In Figure 42, a typical sequence for 8-bit writes is shown. The second byte is written to the special register (R0xF1:0).

Figure 42: WRITE Timing to R0x09:0—Value 0x0284



8-Bit Read Sequence

To read one byte at a time, the same special register address is used for the lower byte. The upper 8 bits are read from the desired register. By following this with a read from the special register (R0xF1:0), the lower 8 bits are accessed (Figure 43 on page 129). The master sets the no-acknowledge bits.

Figure 43: READ Timing from R0x09:0; Returned Value 0x0284

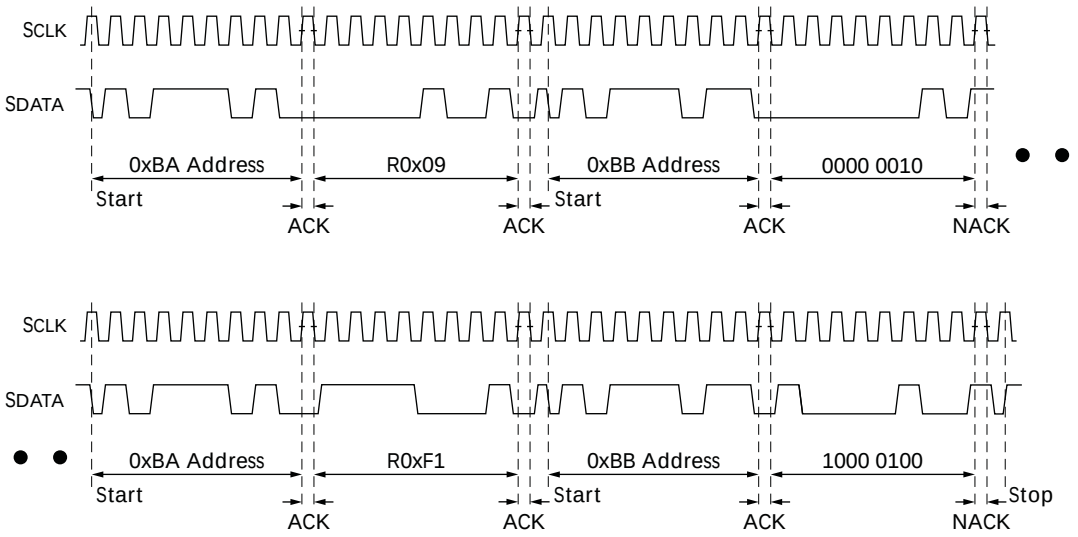
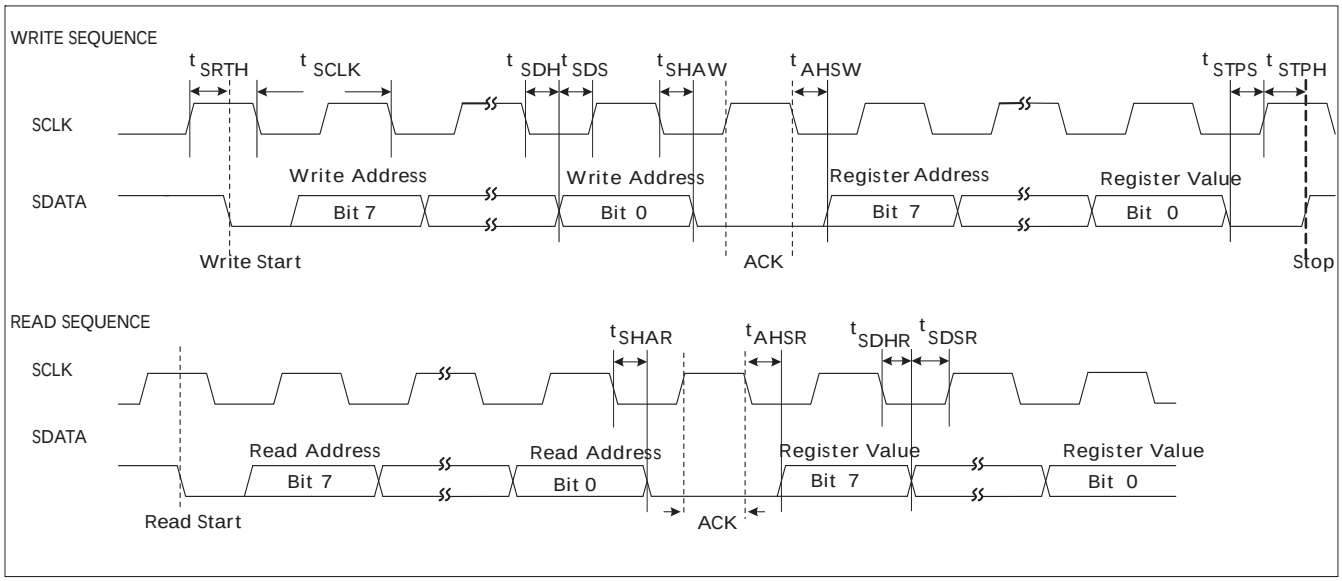


Figure 44: Two-Wire Serial Bus Timing Parameters



Notes: 1. Read waveforms start after a write command and register address are issued. They are for an 8-bit read.

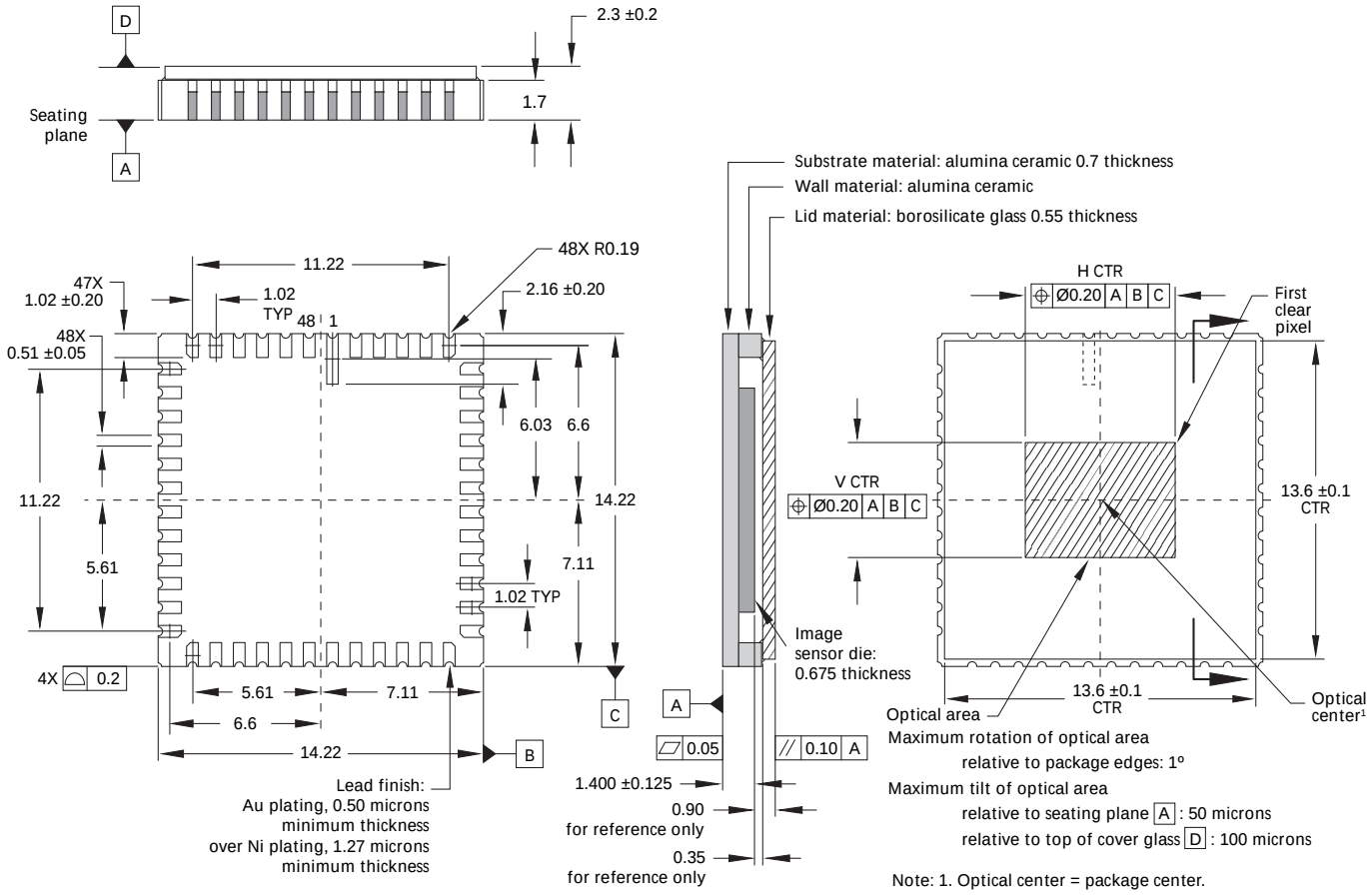
Table 36: Two-Wire Serial Bus Characteristics

Parameter	Symbol	Conditions	Min	Typ	Max	Units
Serial interface input clock frequency	^{f}SCLK				^{f}EXTCLK/16	kHz
Serial interface input clock period	^{t}SCLK		1/^{f}SCLK			ns
SCLK duty cycle			40	50	60	%
Start hold time	^{t}SRTH	WRITE/READ	4*^{t}EXTCLK			ns
SDATA hold	^{t}SDH	WRITE	4*^{t}EXTCLK			ns
SDATA setup	^{t}SDS	WRITE	4*^{t}EXTCLK			ns
SDATA hold to ACK	^{t}SHAW	WRITE	4*^{t}EXTCLK			ns
ACK hold to SDATA	^{t}AHSW	WRITE	4*^{t}EXTCLK			ns
Stop setup time	^{t}STPS	WRITE/READ	4*^{t}EXTCLK			ns
Stop hold time	^{t}STPH	WRITE/READ	4*^{t}EXTCLK			ns
SDATA hold to ACK	^{t}SHAR	READ	4*^{t}EXTCLK			ns
ACK hold to SDATA	^{t}AHSR	READ	4*^{t}EXTCLK			ns
SDATA hold	^{t}SDHR	READ	4*^{t}EXTCLK			ns
SDATA setup	^{t}SDSR	READ	4*^{t}EXTCLK			ns
Serial interface input pin capacitance	CIN_SI			3.5		pF
SDATA max load capacitance	CLOAD_SD			15		pF
SDATA pull-up resistor	RSD			1.5		KΩ

Note: Either the slave or master device can drive the SCLK line LOW—the interface protocol determines which device is allowed to drive the SCLK line at any given time.

Package Dimensions

Figure 45: 48-Pin CLCC Package Outline Drawing



Revision History

Rev. J 5/4/15

- Updated “Ordering Information” on page 2

Rev. H 3/26/15

- Converted to ON Semiconductor template

Rev. G 8/13/12

- Removed iCSP package information because it is no longer supported.

Rev. F 5/11

- Updated trademarks
- Updated to new Aptina template

Rev. E 8/10

- Updated to non-confidential

Rev. D 4/10

- Updated to Aptina template

Rev. C, ... 09/07

- Updated Table 9, “Drivers IDs,” on page 56
- Updated Table 11, “Driver Variables-Auto Exposure Driver (ID = 2),” on page 61
- Added Table 12, “Driver Variables-Auto White Balance (ID = 3),” on page 63
- Added Table 13, “Driver Variables-Flicker Detection Driver (ID = 4),” on page 66
- Updated Table 14, “Driver Variables-Mode/Context Driver (ID = 7),” on page 68
- Updated Table 32, “AC Electrical Characteristics,” on page 122
- Updated Table 2, “Available Part Numbers,” on page 2
- Added “Architecture Overview,” on page 10
- Added Figure 37: “Chief Ray Angle (CRA) vs. Image Height,” on page 120
- Added Figure 47: “64-Ball iCSP Package Outline Drawing,” on page 130

Rev. B 2/07

- Figure 36 on page 119, Spectral Characteristics updated.
- Minor changes to Table 5 on page 23, Table 32 on page 122, and Table 33 on page 123.
- Updated register references to R0x hexadecimal format.
- Updated package from LLCC to CLCC (see Figure 45 on page 131).

Rev. A 7/06

- Initial release

Общество с ограниченной ответственностью «МосЧип» ИНН 7719860671 / КПП 771901001
Адрес: 105318, г.Москва, ул.Щербаковская д.3, офис 1107

Данный компонент на территории Российской Федерации

Вы можете приобрести в компании MosChip.

Для оперативного оформления запроса Вам необходимо перейти по данной ссылке:

http://moschip.ru/get-element

Вы можете разместить у нас заказ для любого Вашего проекта, будь то серийное производство или разработка единичного прибора.

В нашем ассортименте представлены ведущие мировые производители активных и пассивных электронных компонентов.

Нашей специализацией является поставка электронной компонентной базы двойного назначения, продукции таких производителей как XILINX, Intel (ex.ALTERA), Vicor, Microchip, Texas Instruments, Analog Devices, Mini-Circuits, Amphenol, Glenair.

Сотрудничество с глобальными дистрибьюторами электронных компонентов, предоставляет возможность заказывать и получать с международных складов практически любой перечень компонентов в оптимальные для Вас сроки.

На всех этапах разработки и производства наши партнеры могут получить квалифицированную поддержку опытных инженеров.

Система менеджмента качества компании отвечает требованиям в соответствии с ГОСТ Р ИСО 9001, ГОСТ РВ 0015-002 и ЭС РД 009

Офис по работе с юридическими лицами:

105318, г.Москва, ул.Щербаковская д.3, офис 1107, 1118, ДЦ «Щербаковский»

Телефон: +7 495 668-12-70 (многоканальный)

Факс: +7 495 668-12-70 (доб.304)

E-mail: info@moschip.ru

Skype отдела продаж:
moschip.ru
moschip.ru_4
moschip.ru_6
moschip.ru_9