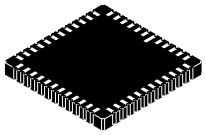
1/3.2-inch 8 Mp CMOS Digital Image Sensor

AR0835HS



CLCC48 10 × 10
CASE 848AJ

Table 1. KEY PERFORMANCE PARAMETERS

Parameter	Value
Array Format	8 Mp: 3264 × 2448 6 Mp: 3264 × 1836
Primary Modes	4:3 – 8 Mp 46 fps Max (HiSPi) and 42 fps Max (MIPI)
	16:9 – 6 Mp at 60 fps Max
	1080p 60 fps / 720p 120 fps Max
Pixel Size	1.4 μm Back Side Illuminated (BSI)
Optical Format	1/3.2″
Die Size	6.86 mm × 6.44 mm (Area: 44.17 mm^2)
Input Clock Frequency	6–27 MHz
Interface	HiSPi Mode: 4 lanes at 1 Gbps Max. MIPI Mode: CSI–2 (2, 3, 4 lanes) at 896 Mbps max.
Subsampling Modes	X – Bin2, Sum2 Skip: 2×, 4×
	Y – Sum2, Skip: 2×, 4×, 8×
Output Data Depth	10–bit Raw, 10–to–8 bit A–Law, 8/6–bit DPCM
Analog Gain	1×, 2×, 3×, 4×, 6×, 8×
High Quality Bayer Scalar	Adjustable Scaling Up to 1/6× Scaling
Temperature Sensor	10–bit, Single Instance on Chip, Controlled by Two–wire Serial I/F
VCM AF Driver	8–bit Resolution with Slew Rate Control
3–D Support	Frame Rate and Exposure Synchronization
Supply Voltage Analog Digital Pixel I/O HiSPi/MIPI	 2.5–3.1 V (2.8 V Nominal) 1.14–1.3 V (1.2 V Nominal) 2.5–3.1 V (2.8 V Nominal) 1.7–1.9 V (1.8 V Nominal) or 2.5–3.1 V (2.8 V Nominal) 1.14–1.3 V (1.2 V Nominal)
OTPM Program Voltage	6.5 V
Power Consumption	Typical 420 mW at 25°C for 8 Mp/46 fps and 6 Mp/60 fps
Responsivity	0.6 V/lux–sec
SNR_{MAX}	36 dB
Dynamic Range	64 dB
Operating Temperature Range (at Junction) $–T_J$	–30°C to +70°C

Features

- High Speed Sensor Supporting 8 Mp (4:3) and 6 Mp (16:9) Up to 60 fps
- 1.4μ Pixel with **onsemi** A–PixHS™ Technology Providing Best–in–class Low–light Performance
- Optional On–chip High–quality Bayer Scaler to Resize Image to Desired Size

ORDERING INFORMATION

See detailed ordering and shipping information on page 2 of this data sheet.

Features (Continued)

- Data Output Serial Interface: Four–lane High–speed Serial Pixel Interface (HiSPi) or Mobile Industry Processor Interface (MIPI)
- Bit–depth Compression Available for Serial Interface: 10–to–8 and 10–6 Bit Compression to Enable Lower Bandwidth Receivers for Full Frame Rate Applications
- On–chip Temperature Sensor
- On–die Phase–locked Loop (PLL) Oscillator
- 5.6 kbits One–time Programmable Memory (OTPM) for Storing Module Information and Calibration Data
- On–chip 8–bit VCM Driver
- 3D Synchronization Controls to Enable Stereo Video Capture
- Interlaced Multi–exposure Readout Enabling High Dynamic Range (HDR) Still and Video Applications
- Programmable Controls: Gain, Horizontal and Vertical Blanking, Auto Black Level Offset Correction, Frame Size/Rate, Exposure, Left–right and Top–bottom Image Reversal, Window Size, and Panning
- Support for External Mechanical Shutter
- Support for External LED or Xenon Flash

Applications

- Sports Cameras
- Digital Still Cameras
- Digital Video Cameras

Table 2. MODE OF OPERATION AND POWER CONSUMPTION

Mode	Active Readout Window (Col × Row)	Sensor Output Resolution (Col × Row)	Mode	FPS	Typical Power Consumption (Note 2)
FULL RESOLUTION 4:3					
8 Mp	3264 × 2448	3264 × 2448	Full Mode	46/42	420 mW
8 Mp	3264 × 2448	3264 × 2448	Full Mode	30	370 mW
FULL RESOLUTION 16:9					
6 Mp	3264 × 1836	3264 × 1836	Full Mode	60	420 mW
6 Mp	3264 × 1836	3264 × 1836	Full Mode	30	370 mW
4:3 VIDEO MODE					
VGA	3264 × 2448	640 × 480	Skip4	180	370 mW
QVGA	3264 × 2448	320 × 240	Skip4	240	370 mW
16:9 VIDEO MODE					
1080p	3264 × 1836	1920 × 1080	Scaling	30	360 mW
1080p	3264 × 1836	1920 × 1080	Scaling	60	420 mW
720p	3264 × 1836	1280 × 720	Bin2−Sum2	120	390 mW
720p	3264 × 1836	1280 × 720	Scaling	60	420 mW
720p	3264 × 1836	1280 × 720	Bin2 + Scaling	60	250 mW

1. Gbps/Lane HiSPi and 896 Mbps/Lane MIPI data transfer rate.
2. Values measured at T = 25°C and nominal voltages.

ORDERING INFORMATION

Table 3. AVAILABLE PART NUMBERS

Part Number	Product Description	Orderable Product Attribute Description
AR0835HS3C12SUAA0−DP	8 MP 1/3″ CIS	Dry Pack with Protective Film
AR0835HS3C12SUAA0−DR	8 MP 1/3″ CIS	Dry Pack without Protective Film

See the **onsemi** Device Nomenclature document (TND310/D) for a full description of the naming convention used for image sensors. For reference documentation, including information on evaluation kits, please visit our web site at www.onsemi.com.

GENERAL DESCRIPTION

The AR0835HS from **onsemi** is a 1/3.2−inch BSI (back side illuminated) CMOS active−pixel digital image sensor with a pixel array of 3264 (H) × 2448 (V) (3280 (H) × 2464 (V) including border pixels). It incorporates sophisticated on−chip camera functions such as mirroring, column and row skip modes, and context switching for zero shutter lag snapshot mode. It is programmable through a simple two−wire serial interface and has very low power consumption.

The AR0835HS digital image sensor features **onsemi**'s breakthrough low−noise 1.4 μm pixel CMOS imaging technology that achieves near−CCD image quality (based on signal−to−noise ratio and low−light sensitivity) while maintaining the inherent size, cost, and integration advantages of CMOS.

FUNCTIONAL OVERVIEW

In order to meet higher frame rates in AR0835HS sensor, the architecture has been re−designed. The analog core has a column parallel architecture with 4 data paths. Digital block has been re−architected to have 4 data paths.

Figure 1 shows the block diagram of the AR0835HS.

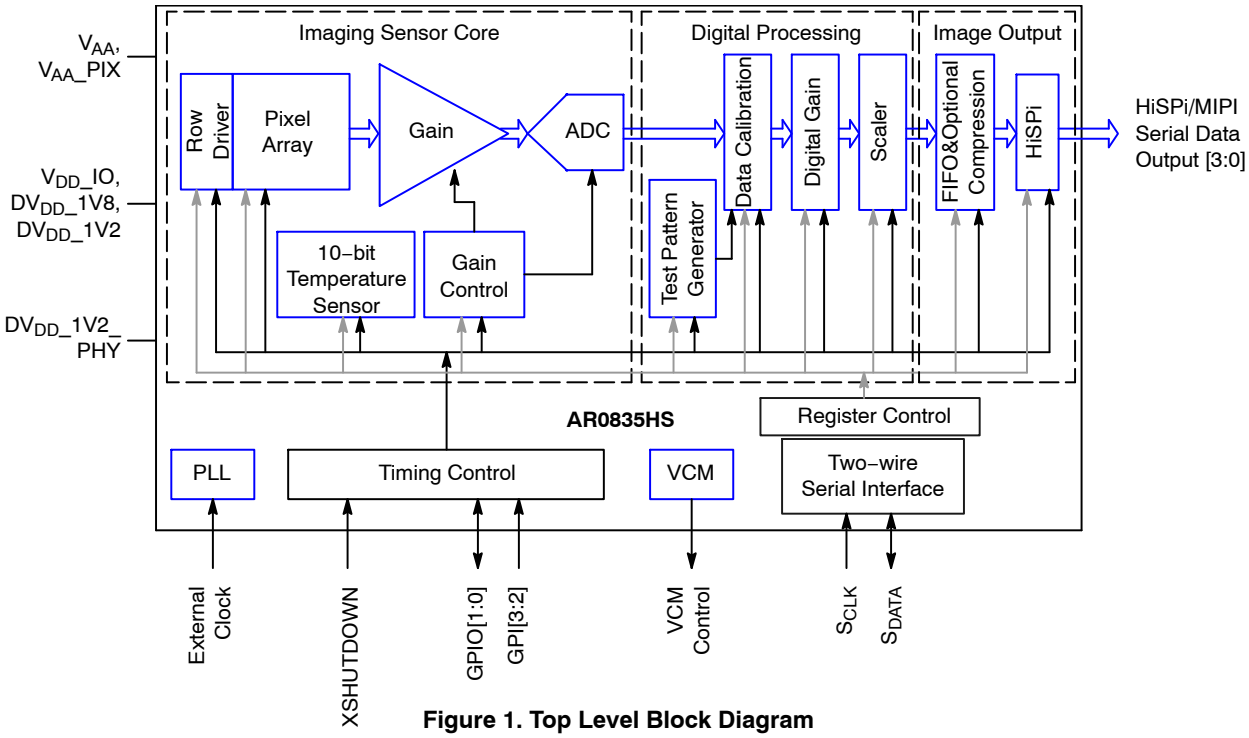


Figure 1. Top Level Block Diagram

The core of the sensor is an 8 Mp active−pixel array. The timing and control circuitry sequences through the rows of the array, resetting and then reading each row in turn. In the time interval between resetting a row and reading that row, the pixels in the row integrate incident light. The exposure is controlled by varying the time interval between reset and readout. Once a row has been read, the data from the columns is sequenced through an analog signal chain (providing gain), and then through an ADC. The output from the ADC is a 10−bit value for each pixel in the array. The ADC output passes through a digital processing signal chain (which provides further data path corrections and applies digital gain).

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

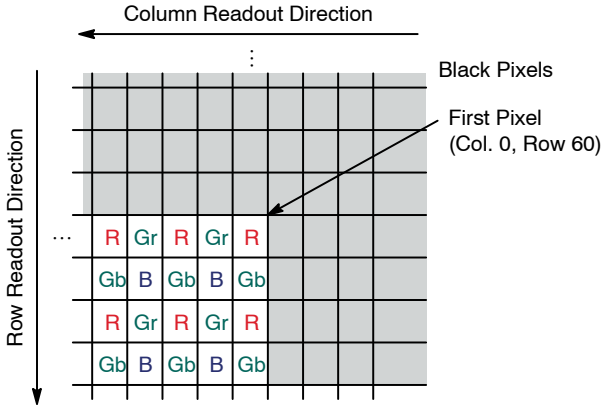
The sensor contains a set of control and status registers that can be used to control many aspects of the sensor behavior including the frame size, exposure, and gain setting. These registers can be accessed through a two−wire serial interface.

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

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

Pixel Array

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

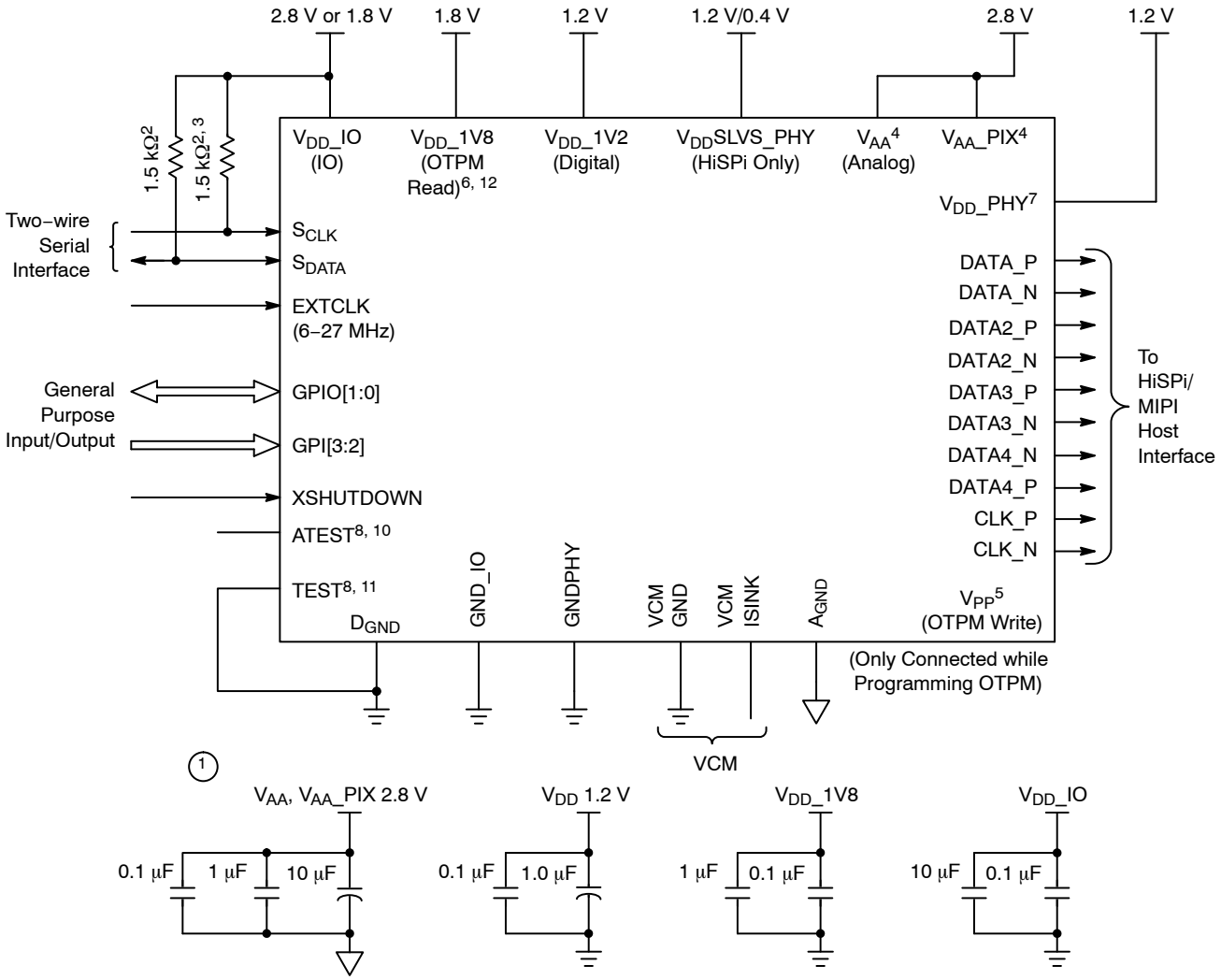


NOTE: By default the mirror bit is set, so the read−out direction is from right to left.

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

TYPICAL CONNECTIONS

The chip supports HiSPi/MIPI output protocol. HiSPi and MIPI are configured to work in 4−lane mode. There are no parallel data output ports.



1. All power supplies should be adequately decoupled; recommended cap values are:
   − 2.8 V: 1.0 μF, 0.1 μF, and then 0.01 μF
   − 1.2 V: 10 μF, 1 μF, and then 0.1 μF
   − 1.8 V: 1 μF and 0.1 μF
2. Resistor value 1.5 kΩ is recommended, but may be greater for slower two−wire speed.
3. This pull−up resistor is not required if the controller drives a valid logic level on S_{CLK} at all times.
4. V_{AA} and V_{AA}_PIX can be tied together. However, for noise immunity it is recommended to have them separate (i.e. two sets of 2.8 V decoupling caps).
5. V_{PP}, 6.5 V, is used for programming OTPM. This pad is left unconnected if OTPM is not being programmed.
6. V_{DD}_1V8 can be combined with V_{DD}_IO, if V_{DD}_IO = 1.8 V.
7. V_{DD}_1V2 and V_{DD}_PHY can be tied together.
8. HiSPi mode only: V_{DD}SLVS_PHY is set to 0.4 V externally. Alternatively, V_{DD}SLVS_PHY may be tied to 1.2 V if the user chooses to have the HiSPi SLVS PHY TX voltage supplied using the AR0835HS's internal 1.2 V−to−0.4 V regulator.
9. Register 31BE[2:3] can be used to program the option of internal of external regulator, **onsemi** recommends using external regulator.
10. ATEST can be left floating.
11. TEST pin must be tied to D_{GND}.
12. V_{DD}_1V8 is the OTPM read voltage.

Figure 3. Typical Application Circuit − HiSPi Connection

SIGNAL DESCRIPTIONS

AR0835HS has 66 pads placed in a two sided pad frame. It has only serial outputs. The part may be configured as HiSPi with different bit depths. The pad description is tabulated in Table 4.

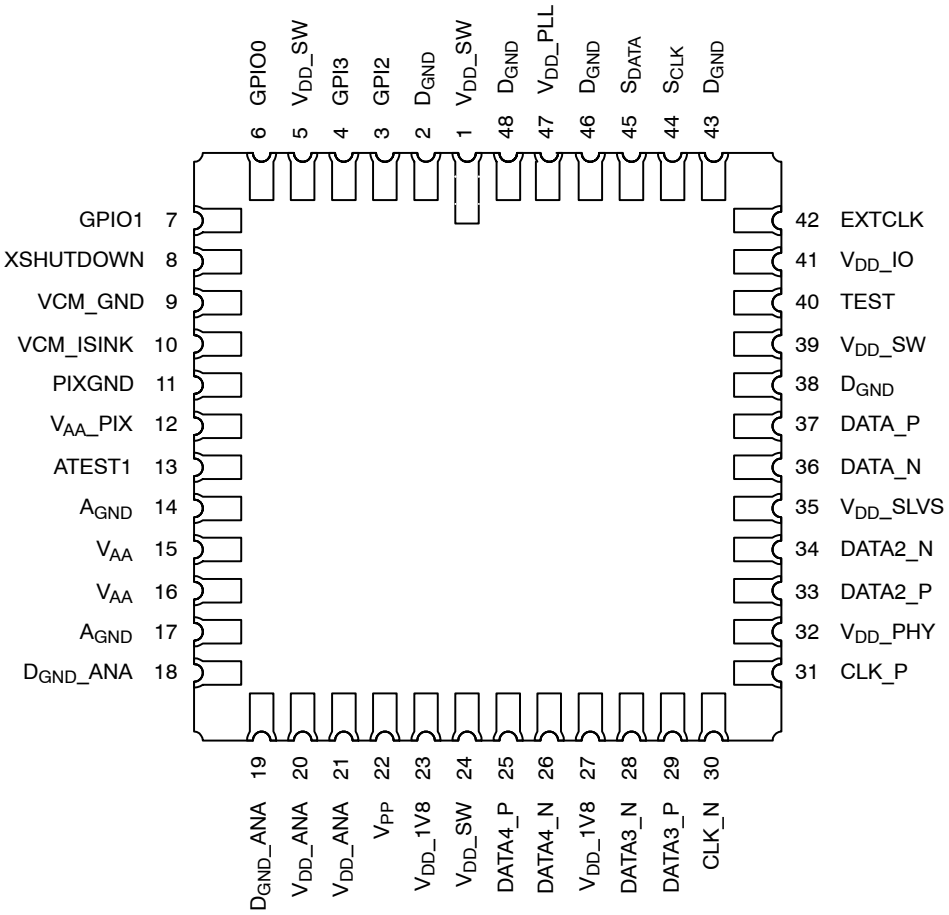


Figure 4. CLCC Package Pinout Diagram (Top Side View)

Table 4. PAD DESCRIPTIONS

Pad Name	Pad Type	Description
SENSOR CONTROL		
EXTCLK	Input	Master clock input; PLL input clock. 6−27 MHz. This is a SMIA−compliant pad.
GPIO0	Input/Output	General Input and one Output function include: a. (Default Output) Flash b. (Input) all options in GPI2 High−Z before XSHUTDOWN going high; default value is ‘0’ after all three voltages in place and XSHUTDOWN being high. After reset, this pad is not powered down since its default use is as Flash pin. If not used, can be left floating.
GPIO0	Input/Output	General Input and one Output function include: a. (Default Output) Flash b. (Input) all options in GPI2 High−Z before XSHUTDOWN going high; default value is ‘0’ after all three voltages in place and XSHUTDOWN being high. After reset, this pad is not powered down since its default use is as Flash pin. If not used, can be left floating.

Table 4. PAD DESCRIPTIONS (continued)

Pad Name	Pad Type	Description
SENSOR CONTROL		
GPIO1	Input/Output	General Input and 2 Output functions include: a. (Default Output) Shutter b. (Output) 3−D daisy chain communication output c. (Input) all options in GPI2 High−Z before XSHUTDOWN going high; default value is '0' after all three voltages in place and XSHUTDOWN being high. After reset, this pad is not powered down since its default use is as Shutter pin. If not used, can be left floating.
GPI2	Input	General Input; After reset, these pads are powered down by default; this means that it is not necessary to bond to these pads. Functions include: a. S_{ADDR}, switch to the second two−wire serial interface device address (see "Slave Address/Data Direction Byte") b. Trigger signal for Slave Mode c. Standby If not used, can be left floating.
GPI3	Input	General Input; After reset, these pads are powered−down by default; this means that it is not necessary to bond to these pads. Functions include: a. 3−D daisy chain communication input b. All options in GPI2 If not used, can be left floating.
TWO−WIRE SERIAL INTERFACE		
S_{CLK}	Input	Serial clock for access to control and status registers
S_{DATA}	I/O	Serial data for reads from and writes to control and status registers
SERIAL OUTPUT		
DATA[4:1]P	Output	Differential serial data (positive)
DATA[4:1]N	Output	Differential serial data (negative)
CLK_P	Output	Differential serial clock/strobe (positive)
CLK_N	Output	Differential serial clock/strobe (negative)
XSHUTDOWN	Input	Asynchronous active LOW reset. When asserted, data output stops and all internal registers are restored to their factory default settings. This pin will turn off the digital power domain and is the lowest power state of the sensor.
VCM DRIVER		
VCM_ISINK	Input/Output	VCM Driver current sink output. If not used, it could be left floating.
VCM_GND	Input/Output	Ground connection to VCM Driver. If not used, needs to be connected to ground (D_{GND}). This ground must be separate from the other grounds.
POWER		
V_{PP}	Supply	High−voltage pin for programming OTPM, present on sensors with that capability. This pin can be left floating during normal operation.
V_{AA}, V_{AA}_PIX, V_{DD}_1V2_[V_{DD}SW, V_{DD}_ANA, V_{DD}_PLL], V_{DD}_1V8, V_{DD}_IO, V_{DD}_PHY, V_{DD}SLVS_PHY, A_{GND}, PIXGND, D_{GND}	Supply	Power supply. The domains are specified in the next table. The brackets indicate the number of individual pins. V_{DD}SLVS_PHY is for HiSPi mode only.

There are standard GPI and GPIO pads, 2 each. Chip can also be communicated to through the two−wire serial interface.

The chip has four unique power supply requirements: 1.2 V (digital), 1.8 V, 2.8 V, and an analog 1.2 V or 0.4 V. These are further divided and in all there are seven power domains and five independent ground domains from the ESD perspective.

Table 5. INDEPENDENT POWER AND GROUND DOMAINS

Pad Name	Power Supply	Description
GROUNDS		
D_{GND}	0 V	Digital
VCM_GND	0 V	
A_{GND}, PIXGND	0 V	Analog
POWER		
V_{AA}	2.8 V	Analog
V_{AA}_PIX	2.8 V	Pixel
V_{DD}SLVS_PHY	0.4 V or 1.2 V	HiSPi PHY. (HiSPi Mode Only)
V_{DD}SW, V_{DD}_ANA, V_{DD}_PLL	1.2 V	Digital
V_{DD}_IO	1.8 V/2.8 V	IO
V_{DD}_PHY	1.2 V	HiSPi/MIPI
V_{DD}_1V8	1.8 V	OTPM

SYSTEM STATES

The system states of the AR0835HS are represented as a state diagram in Figure 5 and described in subsequent sections.

The sensor's operation is broken down into three separate states: hardware standby, software standby, and streaming. The transition between these states might take a certain amount of clock cycles as outlined in Figure 5 and Figure 6.

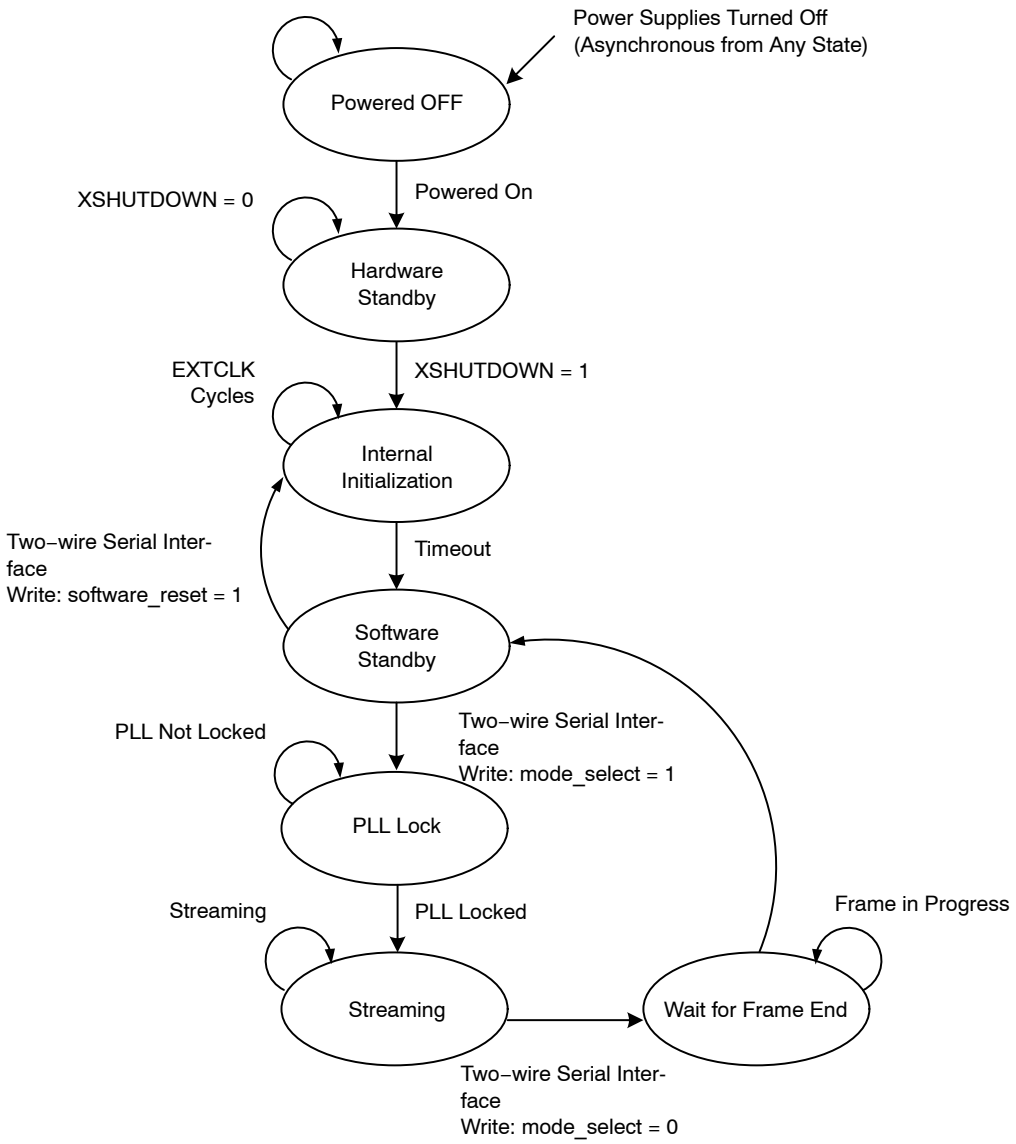


Figure 5. System States

SENSOR INITIALIZATION

Power−Up Sequence

AR0835HS has four voltage supplies divided into several domains. The four voltages are 1.2 V (digital), 1.8 V, 2.8 V, and analog 1.2 V or 0.4 V. For proper operation of the chip, a power−up sequence is recommended as shown in Figure 6.

The power sequence is governed by controlled vs controlling behavior of a power supply and the inrush current (ie current that exists when not all power supplies are present).

Table 6. INRUSH CONSIDERATION

XSHUTDOWN	1.2 V	1.8 V (V_{DD}_IO)	2.8 V	Comment
x	Present	Absent	Absent	Not Supported
x	Absent	Present	Absent	Supported
x	Absent	Absent	Present	Supported
x	Present	Present	Absent	Supported
x	Present	Absent	Present	Not Supported
x	Absent	Present	Present	Supported
0	Present	Present	Present	Powered Down State
1	Present	Present	Present	Powered Up State

Since V_{DD}_IO supply controls the XSHUTDOWN, it should be turned on first. The sequence of powering up the other two domains is not too critical. While turning on 2.8 V supply before 1.2 V supply shouldn't be an issue as shown in Table 1, it is still not recommended since the 2.8 V domain is controlled by 1.2 V signals. The dedicated 1.8 V domain is used only for OTPM read function, so can turn on along with 1.8 V supply.

Due to the above considerations, the suggested power−on sequence is as shown in Figure 6:

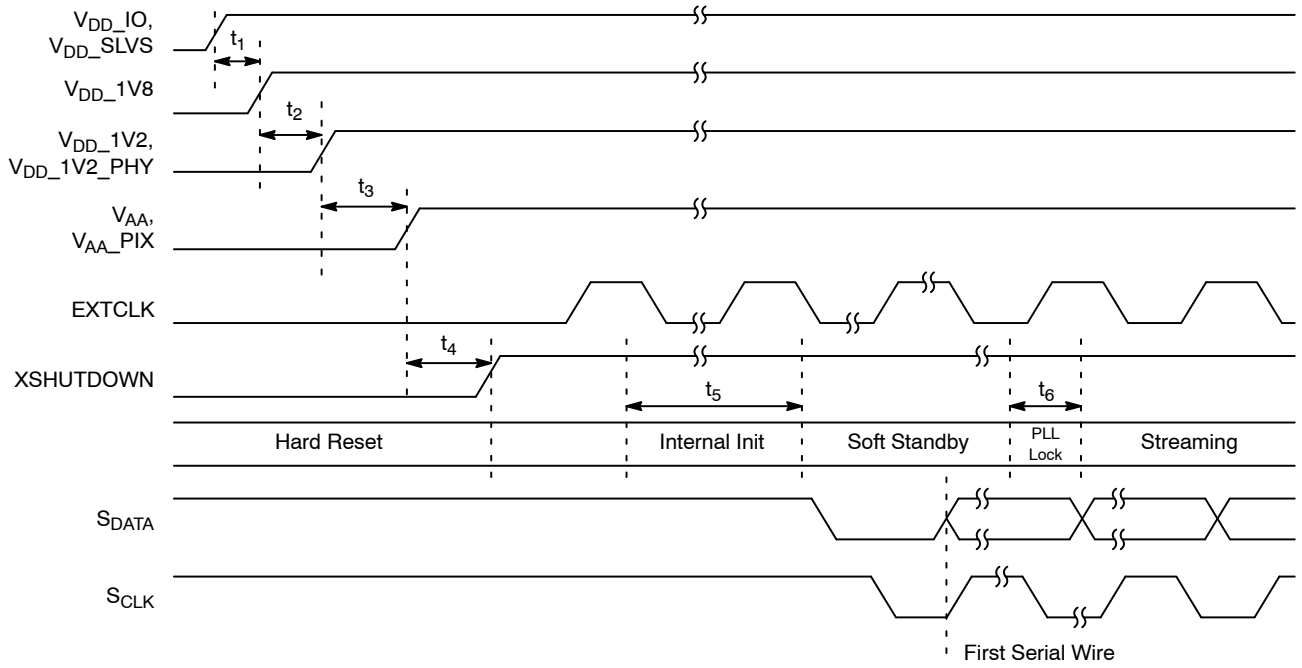


Figure 6. Recommended Power−Up Sequence

Table 7. POWER−UP SEQUENCE

Symbol	Definition	Minimum	Typical	Maximum	Unit
t_1	V_{DD}_IO to V_{DD}_1V8	–	–	500	ms
t_2	V_{DD}_1V8 to V_{DD}_1V2	0.2	–	500	ms
t_3	V_{DD}_1V2 to V_{AA}	0.2	–	500	ms
t_4	Active Hard Reset	1	–	500	ms
t_5	Internal Initialization	2400	–	–	EXTCLKs
t_6	PLL Lock Time	1	–	5	ms

Power−Down Sequence

The recommended power−down sequence for the AR0835HS is shown in Figure 7. The three power supply domains (1.2 V, 1.8 V, and 2.8 V) must have the separation specified below.

1. Disable streaming if output is active by setting standby R0x301a[2] = 0.
2. After disabling the internal clock EXTCLK, disable XSHUTDOWN.
3. After XSHUTDOWN is LOW disable the 2.8 V/1.8 V supply.
4. After the 2.8 V/1.8 V supply is LOW disable the 1.2 V supply.
5. After the 1.2 V supply is LOW disable the V_{DD}_IO supply.

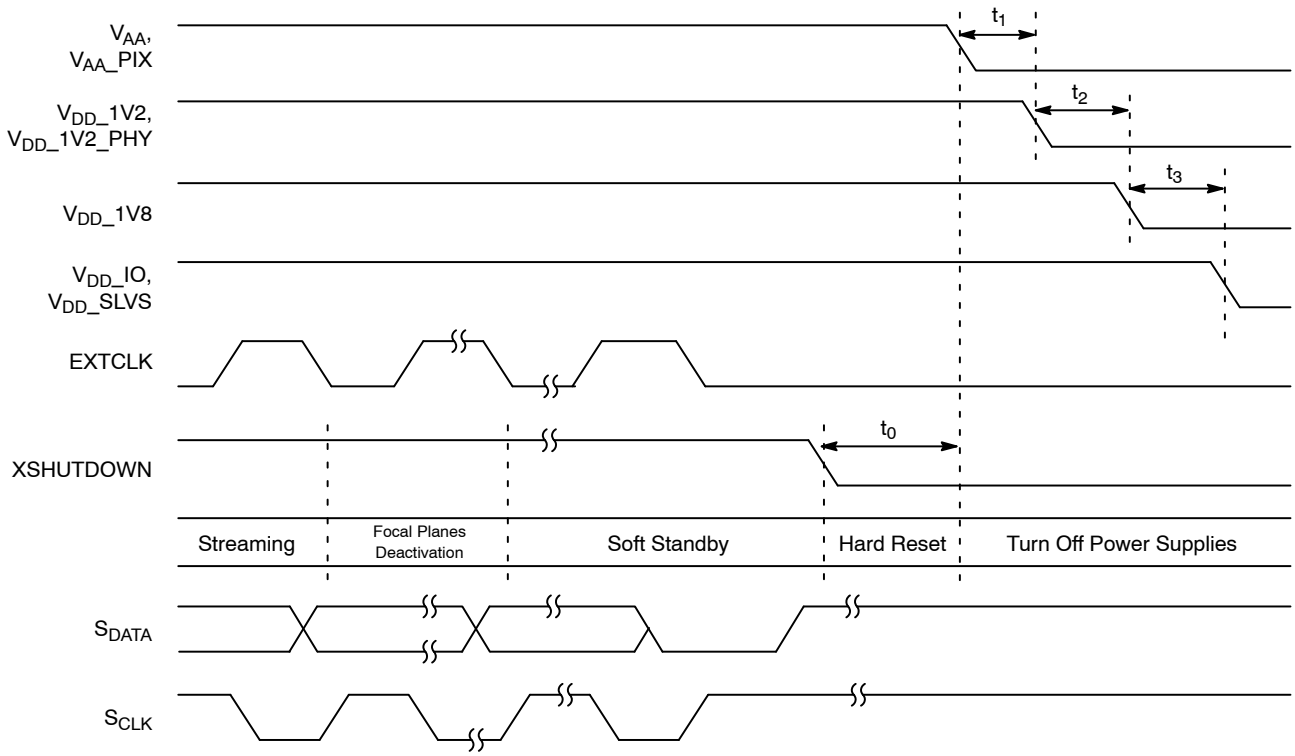


Figure 7. Recommended Power−Down Sequence

Table 8. POWER−DOWN SEQUENCE

Symbol	Definition	Minimum	Typical	Maximum	Unit
	EXTCLK Inactive to XSHUTDOWN Active	100	–	–	μs
t_0	XSHUTDOWN to V_{AA}	200	–	–	μs
t_1	V_{AA} to $V_{DD_}$1V2	0	–	–	μs
t_2	$V_{DD_}$1V2 to $V_{DD_}$1V8	0	–	–	μs
t_3	$V_{DD_}$1V8 to $V_{DD_}$IO	0	–	–	μs

Hard Standby and Hard Reset

The hard standby state is reached by the assertion of the XSHUTDOWN pad (hard reset). Register values are not retained by this action, and will be returned to their default values once hard reset is completed. The minimum power consumption is achieved by the hard standby state. The details of the sequence are described below and shown in Figure 8.

1. Disable streaming if output is active by setting mode_select 0x301A[2] = 0.
2. The soft standby state is reached after the current row or frame, depending on configuration, has ended.
3. Assert XSHUTDOWN (active LOW) to reset the sensor.
4. The sensor remains in hard standby state if XSHUTDOWN remains in the logic “0” state.

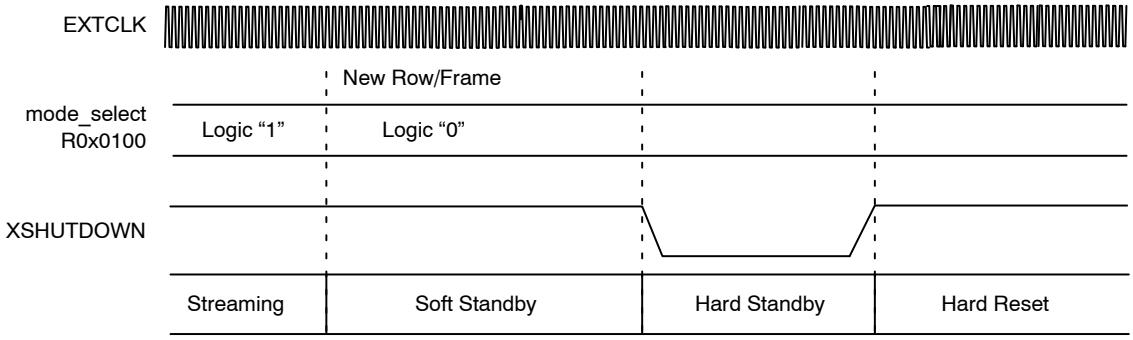


Figure 8. Hard Standby and Hard Reset

Soft Standby and Soft Reset

The AR0835HS can reduce power consumption by switching to the soft standby state when the output is not needed. Register values are retained in the soft standby state. The details of the sequence are described below and shown in Figure 9.

Soft Standby

1. Disable streaming if output is active by setting mode_select 0x301A[2] = 0.
2. The soft standby state is reached after the current row or frame, depending on configuration, has ended.

Soft Reset

1. Follow the soft standby sequence list above.
2. Set software_reset = 1 (R0x3021) to start the internal initialization sequence.
3. After 2400 EXTCLKs , the internal initialization sequence is completed and the current state returns to soft standby automatically.

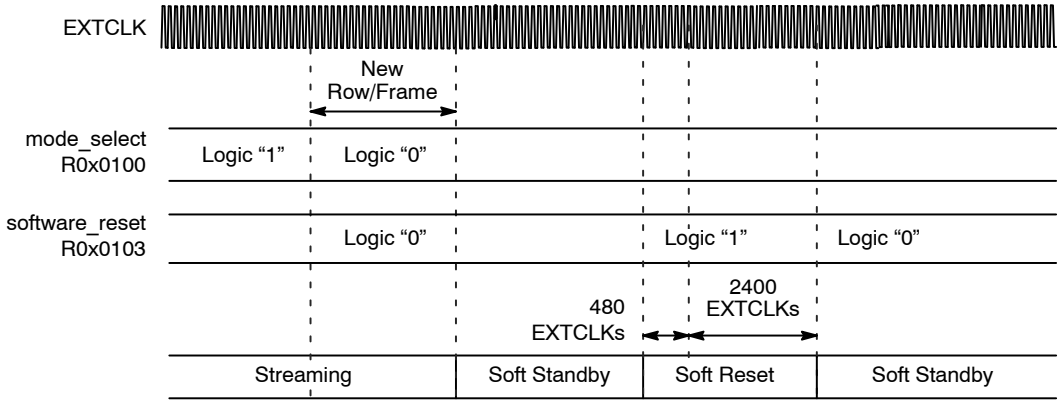


Figure 9. Soft Standby and Soft Reset

TWO−WIRE SERIAL REGISTER INTERFACE

A two−wire serial interface bus enables read/write access to control and status registers within the AR0835HS. The two−wire serial interface is fully compatible with the I^2C standard.

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 (S_{CLK}) that is an input to the sensor and is used to synchronize transfers. Data is transferred between the master and the slave on a bidirectional signal (S_{DATA}). S_{DATA} is pulled up to V_{DD} off−chip by a 1.5 kΩ resistor. Either the slave or master device can drive S_{DATA} LOW − the interface protocol determines which device is allowed to drive S_{DATA} at any given time.

The protocols described in the two−wire serial interface specification allow the slave device to drive S_{CLK} LOW; the AR0835HS uses S_{CLK} as an input only and therefore never drives it LOW. The electrical and timing specifications are further detailed on “Two−Wire Serial Register Interface”.

Protocol

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

1. a (repeated) start condition
2. a slave address/data direction byte

3. an (a no) acknowledge bit
4. a message byte
5. a stop condition

The bus is idle when both S_{CLK} and S_{DATA} are HIGH. Control of the bus is initiated with a start condition, and the bus is released with a stop condition. Only the master can generate the start and stop conditions.

Start Condition

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

Stop Condition

A stop condition is defined as a LOW−to−HIGH transition on S_{DATA} while S_{CLK} is HIGH.

Data Transfer

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

One data bit is transferred during each S_{CLK} clock period. S_{DATA} can change when S_{CLK} is LOW and must be stable while S_{CLK} is HIGH.

Slave Address/Data Direction Byte

Bits [7:1] of this byte represent the device slave address and bit [0] indicates the data transfer direction. A "0" in bit [0] indicates a WRITE, and a "1" indicates a READ. Alternate slave addresses of 0x6E(write address) and 0x6F(read address) can be selected by enabling and asserting the S_{ADDR} signal through the GPI pad.

The alternate slave addresses can also be programmed through R0x31FC.

Message Byte

Message bytes are used for sending register addresses and register write data to the slave device and for retrieving register read data.

Acknowledge Bit

Each 8−bit data transfer is followed by an acknowledge bit or a no−acknowledge bit in the S_{CLK} clock period following the data transfer. The transmitter (which is the master when writing, or the slave when reading) releases S_{DATA}. The receiver indicates an acknowledge bit by driving S_{DATA} LOW.

No−Acknowledge Bit

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

Typical Sequence

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

If the request was a WRITE, the master then transfers the 16−bit register address to which the WRITE should take place. This transfer takes place as two 8−bit sequences and the slave sends an acknowledge bit after each sequence to indicate that the byte has been received. The master then transfers the data as an 8−bit sequence; the slave sends an acknowledge bit at the end of the sequence. The master stops writing by generating a (re)start or stop condition.

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

Single READ from Random Location

This sequence (Figure 10) starts with a dummy WRITE to the 16−bit address that is to be used for the READ. The master terminates the WRITE by generating a restart condition. The master then sends the 8−bit read slave address/data direction byte and clocks out one byte of register data. The master terminates the READ by generating a no−acknowledge bit followed by a stop condition. Figure 10 shows how the internal register address maintained by the AR0835HS is loaded and incremented as the sequence proceeds.

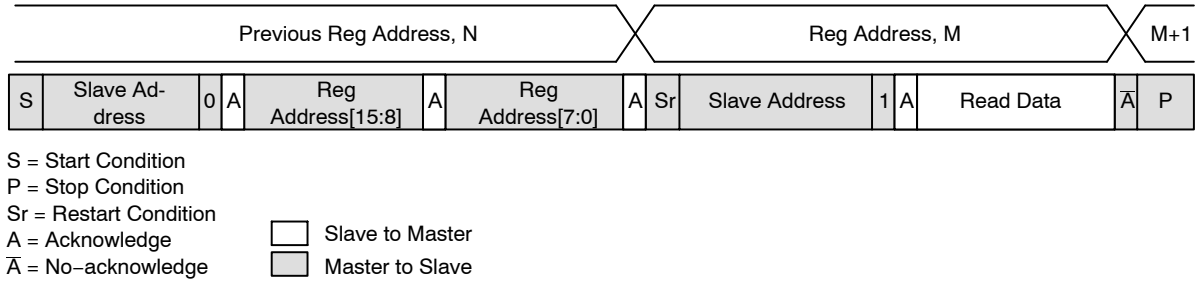


Figure 10. Single READ from Random Location

Single READ from Current Location

This sequence (Figure 11) performs a read using the current value of the AR0835HS internal register address. The master terminates the READ by generating a no−acknowledge bit followed by a stop condition. The figure shows two independent READ sequences.

Figure 11. Single READ from Current Location

Sequential READ, Start from Random Location

This sequence (Figure 12) starts in the same way as the single READ from random location (Figure 10). Instead of generating a no−acknowledge bit after the first byte of data has been transferred, the master generates an acknowledge bit and continues to perform byte READs until “L” bytes have been read.

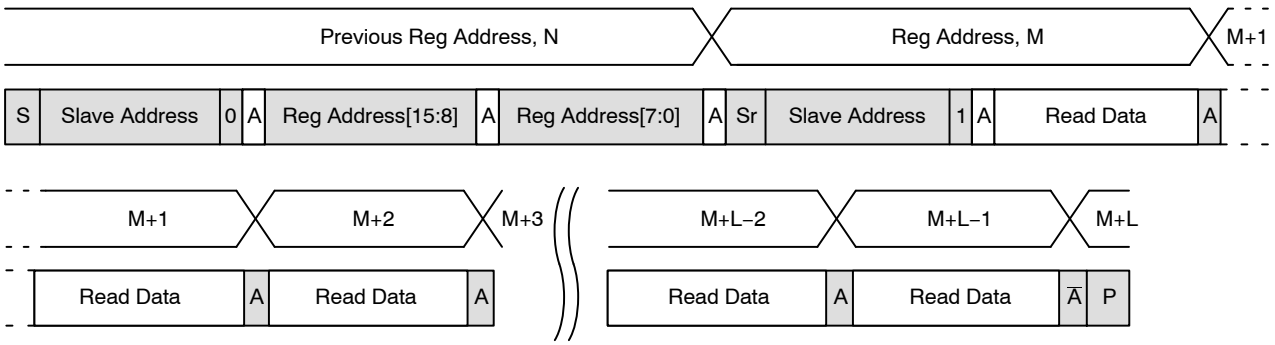


Figure 12. Sequential READ, Start from Random Location

Sequential READ, Start from Current Location

This sequence (Figure 13) starts in the same way as the single READ from current location (Figure 11). Instead of generating a no−acknowledge bit after the first byte of data has been transferred, the master generates an acknowledge bit and continues to perform byte READs until “L” bytes have been read.

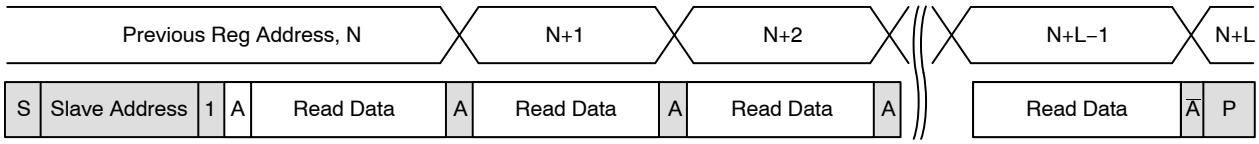


Figure 13. Sequential READ, Start from Current Location

Single WRITE to Random Location

This sequence (Figure 14) begins with the master generating a start condition. The slave address/data direction byte signals a WRITE and is followed by the HIGH then LOW bytes of the register address that is to be written. The master follows this with the byte of write data. The WRITE is terminated by the master generating a stop condition.

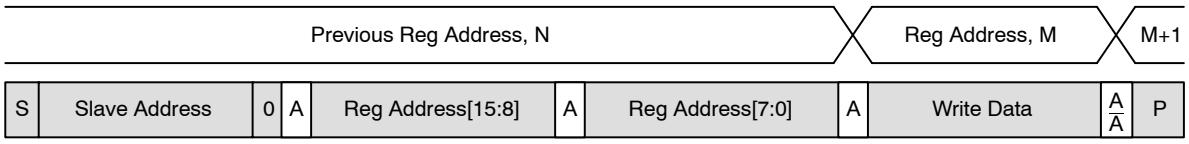


Figure 14. Single WRITE to Random Location

Sequential WRITE, Start at Random Location

This sequence (Figure 15) starts in the same way as the single WRITE to random location (Figure 14). Instead of generating a no−acknowledge bit after the first byte of data has been transferred, the master generates an acknowledge bit and continues to perform byte WRITEs until “L” bytes have been written. The WRITE is terminated by the master generating a stop condition.

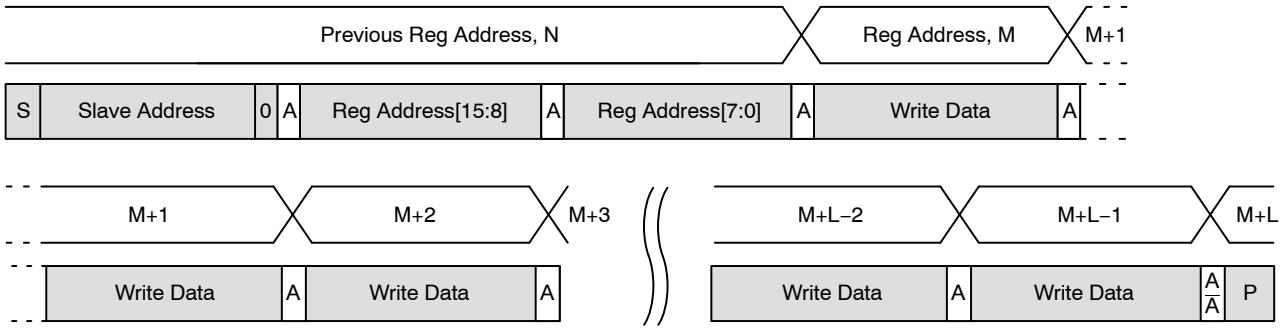


Figure 15. Sequential WRITE, Start at Random Location

REGISTERS

The AR0835HS provides a 16−bit register address space accessed through a serial interface (“Two−Wire Serial Register Interface”). Each register location is 8 or 16 bits in size.

The address space is divided into the five major regions shown in Table 9. The remainder of this section describes these registers in detail.

Table 9. ADDRESS SPACE REGIONS

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

Register Notation

The underlying mechanism for reading and writing registers provides byte write capability. However, it is convenient to consider some registers as multiple adjacent bytes. The AR0835HS uses 8−bit, 16−bit, and 32−bit registers, all implemented as 1 or more bytes at naturally aligned, contiguous locations in the address space.

In this document, registers are described either by address or by name. When registers are described by address, the size of the registers is explicit. For example, R0x3024 is a 2−bit register at address 0x3024, and R0x3000–1 is a 16−bit register at address 0x3000–0x3001. When registers are described by name, the size of the register is implicit. It is necessary to refer to the register table to determine that model_id is a 16−bit register.

Register Aliases

A consequence of the internal architecture of the AR0835HS is that some registers are decoded at multiple addresses. Some registers in “configuration space” are also decoded in “manufacturer−specific space”. To provide unique names for all registers, the name of the register within manufacturer−specific register space has a trailing underscore. For example, R0x0202 is coarse_integration_time and R0x3012 is coarse_integration_time_. The effect of reading or writing a register through any of its aliases is identical.

Bit Fields

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

Bit Field Aliases

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

Byte Ordering

Registers that occupy more than one byte of address space are shown with the lowest address in the highest−order byte lane to match the byte−ordering on the data bus. For example, the chip_version_reg register is R0x0000–1. In the register table the default value is shown as 0x4B00. This means that a read from address 0x0000 would return 0x4B, and a read from address 0x0001 would return 0x00. When reading this register as two 8−bit transfers on the serial interface, the 0x4B will appear on the serial interface first, followed by the 0x00.

Address Alignment

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

Bit Representation

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

Data Format

Most registers represent an unsigned binary value or set of bit fields. For all other register formats, the format is stated explicitly at the start of the register description. The notation for these formats is shown in Table 10.

Table 10. DATA FORMATS

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

Register Behavior

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

Double−Buffered Registers

Some sensor settings cannot be changed during frame readout. For example, changing R0x3004–5 (x_addr_start) partway through frame readout would result in inconsistent row lengths within a frame. To avoid this, the AR0835HS double−buffers many registers by implementing a “pending” and a “live” version. Reads and writes access the pending register. The live register controls the sensor operation.

The value in the pending register is transferred to a live register at a fixed point in the frame timing, called frame

start. Frame start is defined as the point at which the first dark row is read out internally to the sensor. In the register tables the “Frame Sync’d” column shows which registers or register fields are double−buffered in this way.

Using grouped_parameter_hold

Register grouped_parameter_hold (R0x301A[15]) can be used to inhibit transfers from the pending to the live registers. When the AR0835HS is in streaming mode, this register should be written to “1” before making changes to any group of registers where a set of changes is required to take effect simultaneously. When this register is written to “0”, all transfers from pending to live registers take place on the next frame start.

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

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

Bad Frames

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

Many changes to the sensor register settings can cause a bad frame. For example, when line_length_pck (R0x300C) is changed, the new register value does not affect sensor behavior until the next frame start. However, the frame that would be read out at that frame start will have been integrated using the old row width, so reading it out using the new row width would result in a frame with an incorrect integration time.

By default, bad frames are masked. If the masked bad frame option is enabled, both LV and FV are inhibited for these frames so that the vertical blanking time between frames is extended by the frame time.

In the register tables, the “Bad Frame” column shows where changing a register or register field will cause a bad frame. This notation is used:

- *N* – No. Changing the register value will not produce a bad frame.
- *Y* – Yes. Changing the register value might produce a bad frame.
- *YM* – Yes; but the bad frame will be masked out when mask_corrupted_frames (R0x301A[9]) is set to “1”.

Changes to Integration Time

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

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

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

Changes to Gain Settings

Usually, when the gain settings are changed, the gain is updated on the next frame start. When the integration time and the gain are changed at the same time, the gain update is held off by one frame so that the first frame output with the new integration time also has the new gain applied. In this case, a new gain should not be set during the extra frame delay. There is an option to turn off the extra frame delay by setting extra_delay (R0x3018).

CLOCKING

Default setup gives a physical 73.2 MHz internal clock for an external input clock of 24 MHz.

The sensor contains a phase−locked loop (PLL) for timing generation and control. The PLL contains a prescaler to divide the input clock applied on EXTCLK, a VCO to multiply the prescaler output, and a set of dividers to generate the output clocks. The PLL structure is shown in Figure 16.

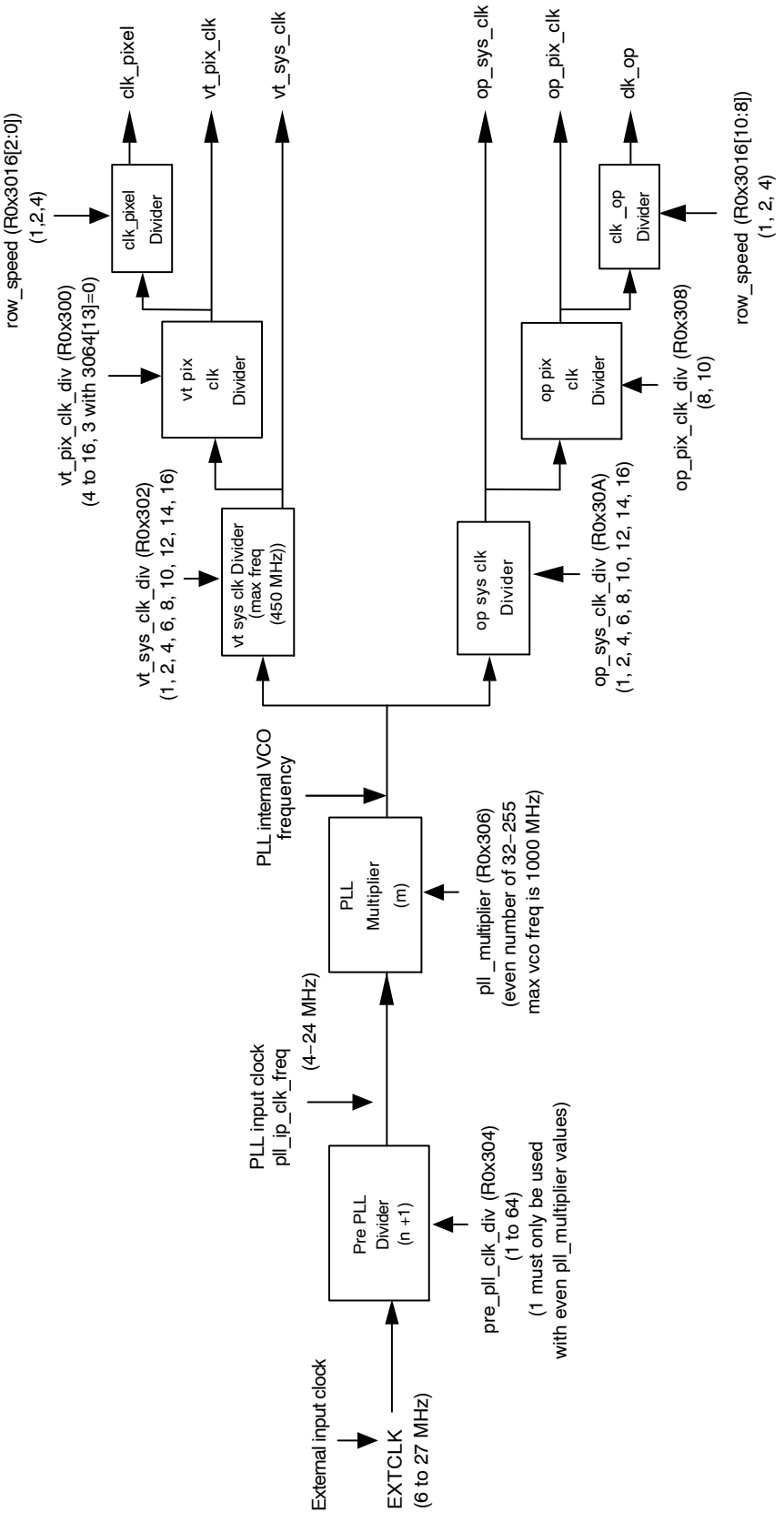


Figure 16. Clocking Configuration (PLL)

Figure 16 shows the different clocks and the names of the registers that contain or are used to control their values. The vt_pix_clk is divided by two to compensate for the fact that the design has 2 digital data paths. This divider should always remain turned on.

AR0835HS has 10−to−8 compression.

The usage of the output clocks is shown below:

- clk_pixel (vt_pix_clk/row_speed[2:0]) is used by the sensor core to readout and control the timing of the pixel array. The sensor core produces one 10−bit pixel each vt_pix_clk period. The line length (line_length_pck) is controlled in increments of the clk_pixel period
- clk_op (op_pix_clk/row_speed[10:8]) is used to load parallel pixel data from the output FIFO (see Figure 41) to the serializer. The output FIFO generates one pixel each op_pix_clk period
- op_sys_clk is used to generate the serial data stream on the output. The relationship between this clock frequency and the op_pix_clk frequency is dependent upon the output data format

The pixel frequency can be calculated in general as:

$$pixel\_clock\_mhz = \frac{ext\_clk\_freq\_mhz \times pll\_multiplier}{pre\_pll\_clk\_div \times vt\_sys\_clk\_div \times 2 \times vt\_pix\_clk\_div \times row\_speed[2:0]} \quad \text{(eq. 1)}$$

The output clock frequency can be calculated as:

$$clk\_op\_freq\_mhz = \frac{ext\_clk\_freq\_mhz \times pll\_multiplier}{pre\_pll\_clk\_div \times op\_sys\_clk\_div \times op\_pix\_clk\_div \times row\_speed[10:8]} \quad \text{(eq. 2)}$$

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

PLL Clocking

The PLL divisors should be programmed while the AR0835HS is in the software standby state. After programming the divisors, it is necessary to wait for the VCO lock time before enabling the PLL. The PLL is enabled by entering the streaming state.

An external timer will need to delay the entrance of the streaming mode by 1 millisecond so that the PLL can lock.

The effect of programming the PLL divisors while the AR0835HS is in the streaming state is undefined.

Clock Control

The AR0835HS uses an aggressive clock−gating methodology to reduce power consumption. The clocked logic is divided into a number of separate domains, each of which is only clocked when required.

When the AR0835HS enters a soft standby state, almost all of the internal clocks are stopped. The only exception is that a small amount of logic is clocked so that the two−wire serial interface continues to respond to read and write requests.

FEATURES

Interlaced HDR Readout

The sensor enables HDR by outputting frames where even and odd row pairs within a single frame are captured at different integration times. This output is then matched with an algorithm designed to reconstruct this output into an HDR still image or video.

The sensor HDR is controlled by two shutter pointers (Shutter pointer1, Shutter pointer2) that control the integration of the odd (Shutter pointer1) and even (Shutter pointer2) row pairs.

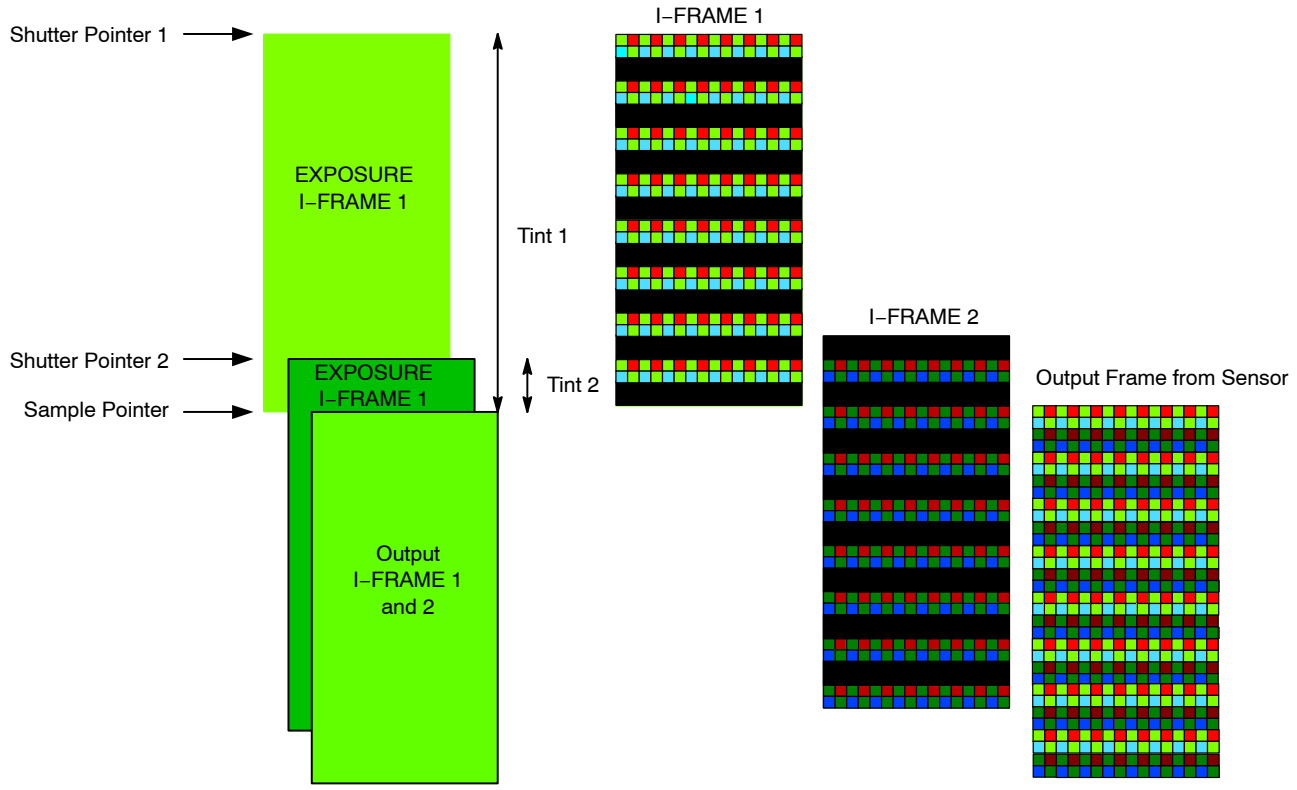


Figure 17. HDR Integration Time

INTEGRATION TIME FOR INTERLACED HDR READOUT

Tint1 (Integration Time 1) and Tint2 (Integration Time 2)

The limits for the coarse integration time are defined by:

$$\text{coarse\_integration\_time\_min} \le \text{coarse\_integration\_time} \le (\text{frame\_length\_lines} - \text{coarse\_integration\_time\_max\_margin}) \quad \text{(eq. 4)}$$

$$\text{coarse\_integration\_time2\_min} \le \text{coarse\_integration\_time2} \le (\text{frame\_length\_lines} - \text{coarse\_integration\_time2\_max\_margin}) \quad \text{(eq. 5)}$$

The actual integration time is given by:

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

$$\text{integration\_time2} = \frac{\text{coarse\_integration\_time2} \times \text{line\_length\_pck}}{\text{vt\_pix\_clk\_freq\_mhz} \times 10^6} \quad \text{(eq. 7)}$$

If this limit is broken, the frame time will automatically be extended to *(coarse_integration_time + coarse_integration_time_max_margin)* to accommodate the larger integration time.

The ratio between even and odd rows is typically adjusted to 1×, 2×, 4×, and 8×.

Bayer Resampler

The imaging artifacts found from a 2×2 binning or summing will show image artifacts from aliasing. These can be corrected by resampling the sampled pixels in order to filter these artifacts. Figure 18 shows the pixel location resulting from 2×2 summing or binning located in the middle and the resulting pixel locations after the Bayer re−sampling function has been applied.

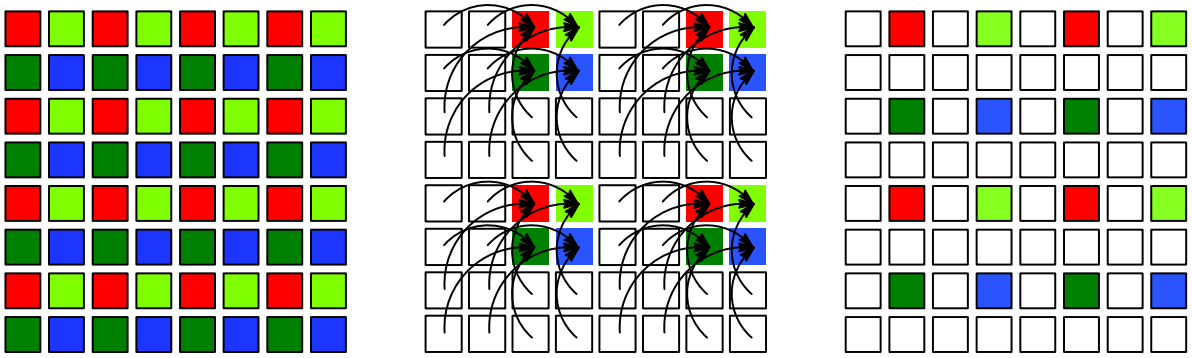


Figure 18. Bayer Resampling

The improvements from using the Bayer resampling feature can be seen in Figure 19. In this example, image edges seen on a diagonal have smoother edges when the Bayer re−sampling feature is applied. This feature is only designed to be used with modes configured with 2 × 2 binning or summing. The feature will not remove aliasing artifacts that are caused skipping pixels.

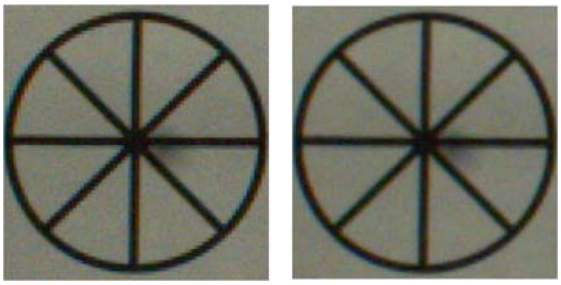


Figure 19. Results of Resampling

To enable the Bayer resampling feature:

1. Set R0x400 = 2 // Enable the on−chip scalar.
2. Set R0x306E to 0x90B0 // Configure the on−chip scalar to resample Bayer data.

To disable the Bayer resampling feature:

1. Set R0x400 = 0 // Disable the on−chip scalar.
2. Set R0x306E to 0x9080 // Configure the on−chip scalar to resample Bayer data.

NOTE: The image readout (rows and columns) has to have two extra rows and two extra columns when using the resample feature.

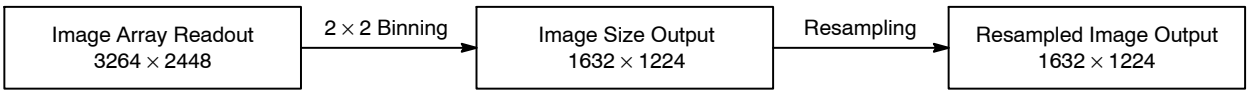


Figure 20. Illustration of Resampling Operation

One−Time Programmable Memory (OTPM)

The AR0835HS features 5.6 kbits of one−time programmable memory (OTPM) for storing shading correction coefficients, individual module, and customer−specific information. The user may program the data before shipping. OTPM can be accessed through two−wire serial interface. The AR0835HS uses the auto mode for fast OTPM programming and read operations.

To read out the OTPM, 1.8 V supply is required. As a result, a dedicated DV_{DD}_1V8 pad has been implemented. During the programming process, a dedicated pin for high voltage needs to be provided to perform the anti−fusing operation. This voltage (V_{PP}) would need to be 6.5 V. The completion of the programming process will be communicated by a register through the two−wire serial interface.

If the V_{PP} pin does not need to be bonded out as a pin on the module, it should be left floating inside the module.

The programming of the OTPM requires the sensor to be fully powered and remain in software standby with its clock input applied. The information will be programmed through the use of the two−wire serial interface, and once the data is written to an internal register, the programming host machine will apply a high voltage to the programming pin, and send a program command to initiate the anti−fusing process. After the sensor has finished programming the OTPM, a status bit will be set to indicate the end of the programming cycle, and the host machine can poll the setting of the status bit through the two−wire serial interface. Only one programming cycle for the 16−bit word can be performed.

Reading the OTPM data requires the sensor to be fully powered and operational with its clock input applied. The data can be read through a register from the two−wire serial interface.

Programming and Verifying the OTPM

The procedure for programming and verifying the AR0835HS OTPM follows:

1. Apply power to all the power rails of the sensor.
2. Provide a 12−MHz EXTCLK clock input.
3. Set R0x301A = 0x18, to put sensor in the soft standby mode.
4. Set R0x3130 = 0xFF01 (Timing configuration).
5. Set R0x304C[15:8] = Record type (e.g. 0x30).
6. Set R0x304C[7:0] = Length of the record which is the number of OTPM data registers that are filled in.
7. Set R0x3054[9] = 0 to ensure that the error checking and correction is enabled.
8. Write data into all the OTPM data registers: R0x3800−R0x39FE.
9. Ramp up V_{PP} to 6.5 V.
10. Set the otpm_control_auto_wr_start bit in the otpm_control register R0x304A[0] = 1, to initiate the auto program sequence. The sensor will now program the data into the OTPM.
11. Poll otpm_control_auto_wr_end (R0x304A [1]) to determine when the sensor is finished programming the word.
12. Verify that the otpm_control_auto_wr_success (0x304A[2]) bit is set.
13. If the above bits are not set to 1, then examine otpm_status register R0x304E[9] to verify if the OTPM memory is full and 0x304E[10] to verify if OTPM memory is insufficient.
14. Remove the high voltage (V_{PP}) and float V_{PP} pin.

Reading the OTPM

1. Apply power to all the power rails of the sensor (V_{DD}_IO, V_{AA}, V_{AA}_PIX, DV_{DD}_1V2, DV_{DD}_1V2_PHY, and DV_{DD}_1V8) at their nominal voltage.
2. Set EXTCLK to normal operating frequency.
3. Perform proper reset sequence to the sensor.
4. Set R0x3134 = 0xCD95 (Timing Configuration)
5. Set R0x304C[15:8] = Record Type (for example, 0x30)
6. Set R0x304C[7:0] = Length of the record which is the number of data registers to be read back. This could be set to 0 during OTPM auto read if length is unknown.

7. Set R0x3054 = 0x0400.
8. Initiate the auto read sequence by setting the otpm_control_auto_read_start bit (R0x304A[4]) = 1.
9. Poll the otpm_control_auto_rd_end bit (R0x304A[5]) to determine when the sensor is finished reading the word(s). When this bit becomes 1, the otpm_control_auto_rd_success bit (R0x304A[6]) will indicate whether the memory was read successfully or not.
10. Data can now be read back from the otpm_data registers (R0x3800−R0x39FE).

Image Acquisition Modes

The AR0835HS supports two image acquisition modes:

1. Electronic Rolling Shutter (ERS) Mode:
   This is the normal mode of operation. When the AR0835HS is streaming, it generates frames at a fixed rate, and each frame is integrated (exposed) using the ERS. When the ERS is in use, timing and control logic within the sensor sequences through the rows of the array, resetting and then reading each row in turn. In the time interval between resetting a row and subsequently reading that row, the pixels in the row integrate incident light. The integration (exposure) time is controlled by varying the time between row reset and row readout. For each row in a frame, the time between row reset and row readout is fixed, leading to a uniform integration time across the frame. When the integration time is changed (by using the two−wire serial interface to change register settings), the timing and control logic controls the transition from old to new integration time in such a way that the stream of output frames from the AR0835HS switches cleanly from the old integration time to the new while only generating frames with uniform integration. See “Changes to Integration Time”.
2. Global Reset Release (GRR) Mode:
   This mode can be used to acquire a single image at the current resolution. In this mode, the end point of the pixel integration time is controlled by an external electromechanical shutter, and the AR0835HS provides control signals to interface to that shutter. The operation of this mode is described in detail in “Global Reset Release (GRR)”.

The benefit for the use of an external electromechanical shutter is that it eliminates the visual artifacts associated with ERS operation. Visual artifacts arise in ERS operation, particularly at low frame rates, because an ERS image effectively integrates each row of the pixel array at a different point in time.

Window Control

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

Pixel Border

The default settings of the sensor provide a 3264 (H) × 2448 (V) image. A border of up to 8 pixels (4 in binning) on each edge can be enabled by reprogramming the x_addr_start, y_addr_start, x_addr_end, y_addr_end, x_output_size, and y_output_size registers accordingly. These border pixels can be used but are disabled by default.

Readout Modes

Horizontal Mirror

The horizontal_mirror bit in the image_orientation register is set by default. The result of this is that the order of pixel readout within a row is reversed, so that readout starts from x_addr_end and ends at x_addr_start. Figure 21 shows a sequence of 6 pixels being read out with horizontal_mirror = 0 and horizontal_mirror = 1. Changing horizontal_mirror causes the Bayer order of the output image to change; the new Bayer order is reflected in the value of the pixel_order register.

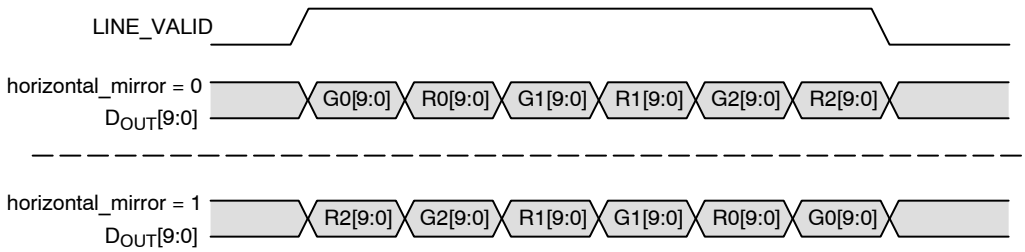


Figure 21. Effect of horizontal_mirror on Readout Order

Vertical Flip

When the vertical_flip bit is set in the image_orientation register, the order in which pixel rows are read out is reversed, so that row readout starts from y_addr_end and ends at y_addr_start. Figure 22 shows a sequence of 6 rows being read out with vertical_flip = 0 and vertical_flip = 1. Changing vertical_flip causes the Bayer order of the output image to change; the new Bayer order is reflected in the value of the pixel_order register.

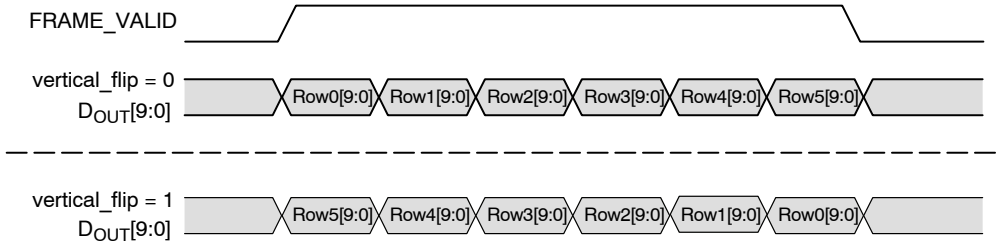


Figure 22. Effect of vertical_flip on Readout Order

Subsampling

The AR0835HS supports subsampling to reduce the amount of data processed by the signal chains in the AR0835HS, thereby allowing the frame rate to be increased and power consumption reduced. Subsampling is enabled by setting x_odd_inc and/or y_odd_inc. Values of 1, 3, and 7 can be supported. Setting both of these variables to 3 reduces the amount of row and column data processed and is equivalent to the 2 × 2 skipping readout mode provided by the AR0835HS. Setting x_odd_inc = 3 and y_odd_inc = 3 results in a quarter reduction in output image size. Figure23 shows a sequence of 8 columns being read out with x_odd_inc = 3 and y_odd_inc = 1.

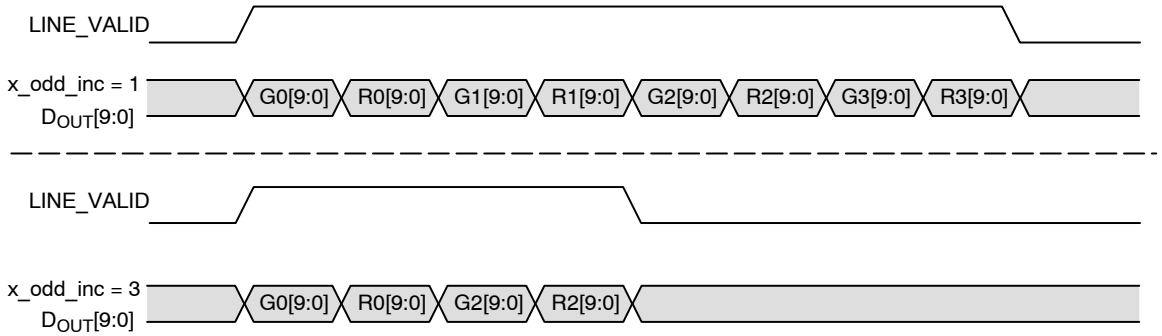


Figure 23. Effect of x_odd_inc = 3 on Readout Sequence

A 1/16 reduction in resolution is achieved by setting both x_odd_inc and y_odd_inc to 7. This is equivalent to 4 × 4 skipping readout mode provided by the AR0835HS. Figure 24 shows a sequence of 16 columns being read out with x_odd_inc = 7 and y_odd_inc = 1.

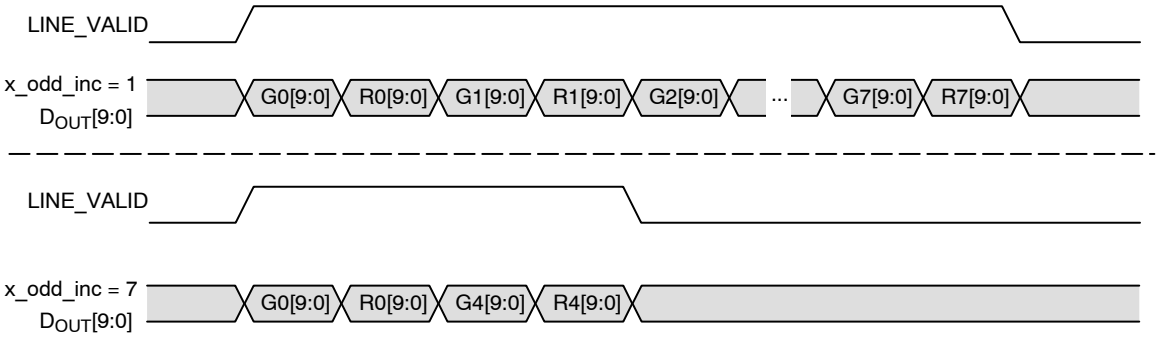


Figure 24. Effect of x_odd_inc = 7 on Readout Sequence

The effect of the different subsampling settings on the pixel array readout is shown in Figure 25 through Figure 27.

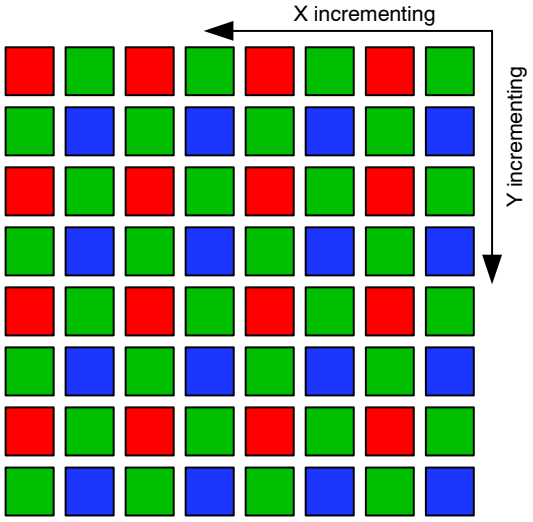


Figure 25. Pixel Readout (No Subsampling)

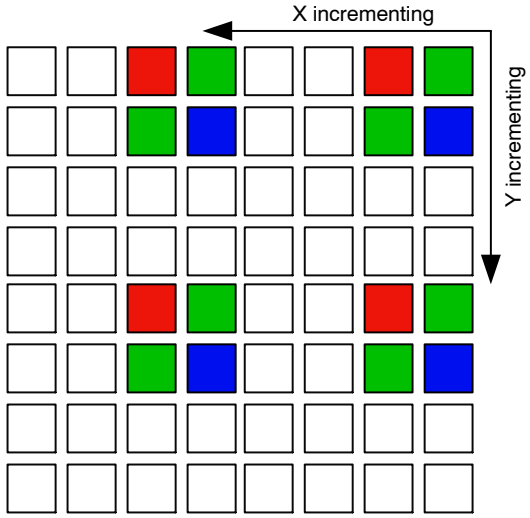


Figure 26. Skip2 Pixel Readout (x_odd_inc = 3, y_odd_inc = 3)

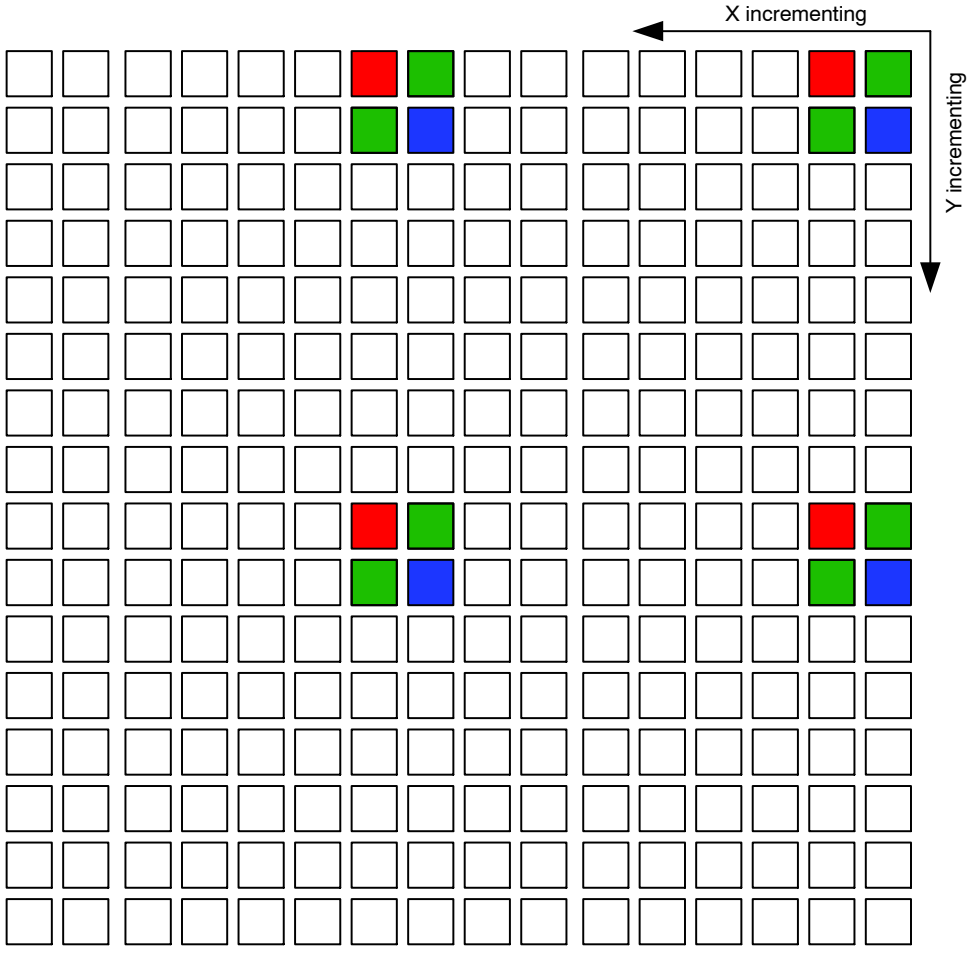


Figure 27. Skip4 Pixel Readout (x_odd_inc = 7, y_odd_inc = 7)

Programming Restrictions when Subsampling

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

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

x_skip_factor = (x_odd_inc + 1) / 2
y_skip_factor = (y_odd_inc + 1) / 2

- x_addr_start should be a multiple of x_skip_factor × 4
- (x_addr_end − x_addr_start + x_odd_inc) should be a multiple of x_skip_factor × 4
- (y_addr_end − y_addr_start + y_odd_inc) should be a multiple of y_skip_factor × 4

The number of columns/rows read out with subsampling can be found from the equation below:

- columns/rows = (addr_end − addr_start + odd_inc) / skip_factor

Table 11 shows the row or column address sequencing for normal and subsampled readout. In the 2× skip case, there are two possible subsampling sequences (because the subsampling sequence only reads half of the pixels) depending upon the alignment of the start address. Similarly, there will be four possible subsampling sequences in the 4× skip case (though only the first two are shown in Table 11).

Table 11. ROW ADDRESS SEQUENCING DURING SUBSAMPLING

odd_inc = 1 (Normal)	odd_inc = 3 (2× Skip)	odd_inc = 7 (4× Skip)
Start = 0	**Start = 0**	**Start = 0**
0	0	0
1	1	1
2		
3		
4	4	
5	5	
6		
7		
8	8	8
9	9	9
10		
11		
12	12	
13	13	
14		
15		

Binning

The AR0835HS supports 2 × 1 (column binning, also called x−binning). Binning has many of the same characteristics as skipping, but because it gathers image data from all pixels in the active window (rather than a subset of them), it achieves superior image quality and avoids the aliasing artifacts that can be a characteristic side effect of skipping.

Binning is enabled by selecting the appropriate subsampling settings (in read_mode, the sub−register x_odd_inc = 3 and y_odd_inc = 1 for x−binning and setting the appropriate binning bit in read_mode R0x3040[11] = 1 for x_bin_enable). As with skipping, x_addr_end and y_addr_end may require adjustment when binning is enabled. It is the first of the two columns/rows binned together that should be the end column/row in binning, so the requirements to the end address are exactly the same as in skipping mode. The effect of the different binning is shown in Figure 28 below and Figure 29.

Binning can also be enabled when the 4× subsampling mode is enabled (x_odd_inc = 7 and y_odd_inc = 1 for x−binning, x_odd_inc = 7 and y_odd_inc = 7 for 4× xy−binning). In this mode, however, not all pixels will be used so this is not a 4× binning implementation. An implementation providing a combination of skip2 and bin2 is used to achieve 4× subsampling with better image quality. The effect of this subsampling mode is shown in Figure 29.

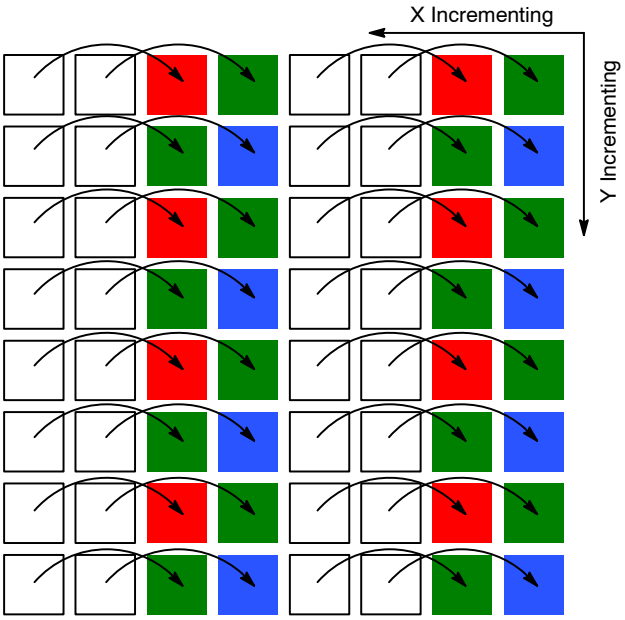


Figure 28. Bin2 Pixel Readout (x_odd_inc = 3, y_odd_inc = 1, x_bin = 1)

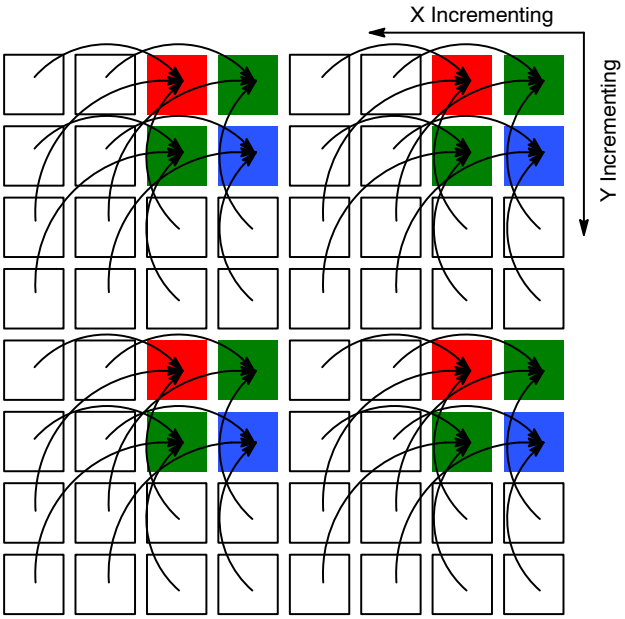


Figure 29. Bin2 Pixel Readout (x_odd_inc = 3, y_odd_inc = 3, x_bin = 1)

Binning address sequencing is a bit more complicated than during subsampling only, because of the implementation of the binning itself.

For a given column *n*, there is only one other column, n_bin, that can be binned with, because of physical limitations in the column readout circuitry. The possible address sequences are shown in Table 12.

Table 12. COLUMN ADDRESS SEQUENCING DURING BINNING

odd_inc = 1 (Normal)	odd_inc = 3 (2× Bin)	odd_inc = 7 (2× Skip + 2× Bin)
x_addr_start = 0	x_addr_start = 0	x_addr_start = 0
0	0/2	0/4
1	1/3	1/5
2		
3		
4	4/6	
5	5/7	
6		
7		
8	8/10	8/12
9	9/11	9/13
10		
11		
12	12/14	
13	13/15	
14		
15		

There are no physical limitations on what can be binned together in the row direction. A given row *n* will always be binned with row n + 2 in 2× subsampling mode and with row n + 4 in 4× subsampling mode. Therefore, which rows get binned together depends upon the alignment of y_addr_start. The possible sequences are shown in Table 13.

Table 13. ROW ADDRESS SEQUENCING DURING BINNING

odd_inc = 1 (Normal)	odd_inc = 3 (2× Bin)	odd_inc = 7 (2× Skip + 2× Bin)
y_addr_start = 0	y_addr_start = 0	y_addr_start = 0
0	0/2	0/4
1	1/3	1/5
2		
3		
4	4/6	
5	5/7	
6		
7		
8	8/10	8/12
9	9/11	9/13
10		
11		
12	12/14	
13	13/15	
14		
15		

Programming Restrictions When Binning and Summing

Binning and summing require different sequencing of the pixel array and impose different timing limits on the operation of the sensor.

As a result, when xy−subsampling is enabled, some of the programming limits declared in the parameter limit registers are no longer valid. In addition, the default values for some of the manufacturer−specific registers need to be reprogrammed. See “Minimum Frame Time” and “Minimum Row Time”.

Subsampling/Binning Options:

1. XskipYskip
   R0x3040[11], x_bin_en: 0
   R0x3040[13], row_sum: 0
   R0x0382: x_odd_inc = 3 (xskip2) or 7 (xskip4)
   R0x0386: y_odd_inc = 3 (yskip2), 7 (yskip4) or 15 (yskip8)
2. XbinYskip
   R0x3040[11], x_bin_en: 1
   R0x3040[13], row_sum: 0
   R0x0382: x_odd_inc = 3 (xbin2)
   R0x0386: y_odd_inc = 3 (yskip2), 7 (yskip4) or 15 (yskip8)
3. XskipYsum
   R0x3040[11], x_bin_en: 0
   R0x3040[13], row_sum: 1
   R0x0382: x_odd_inc = 3 (xskip2) or 7 (xskip4)
   R0x0386: y_odd_inc = 3 (ysum2)
4. XbinYsum
   R0x3040[11], x_bin_en: 1
   R0x3040[13], row_sum: 1
   R0x0382: x_odd_inc = 3 (xbin2)
   R0x0386: y_odd_inc = 3 (ysum2)
5. XsumYsum
   R0x3040[11], x_bin_en: 1
   R0x3040[13], row_sum: 1
   R0x3EE4[0], sreg_colamp_sum2: 1
   (cannot write to this bit when streaming – have to write to entire register)
   R0x0382: x_odd_inc = 3 (xsum2)
   R0x0386: y_odd_inc = 3 (ysum2)

Binning, Skipping, and Summing Mode

Summing, skipping, and binning can be combined in the modes listed in Table 14. Unlike binning mode where the values of adjacent same color pixels are averaged together, summing adds the pixel values together resulting in better sensor sensitivity. Summing is supposed to provide two times the sensitivity compared to the binning only mode.

Table 14. AVAILABLE SKIP, BIN, AND SUM MODE IN THE AR0835HS SENSOR

Subsampling Method	Horizontal	Vertical
Skipping	2×, 4×	2×, 4×, 8×
Binning	2×	
Summing	2×	2×

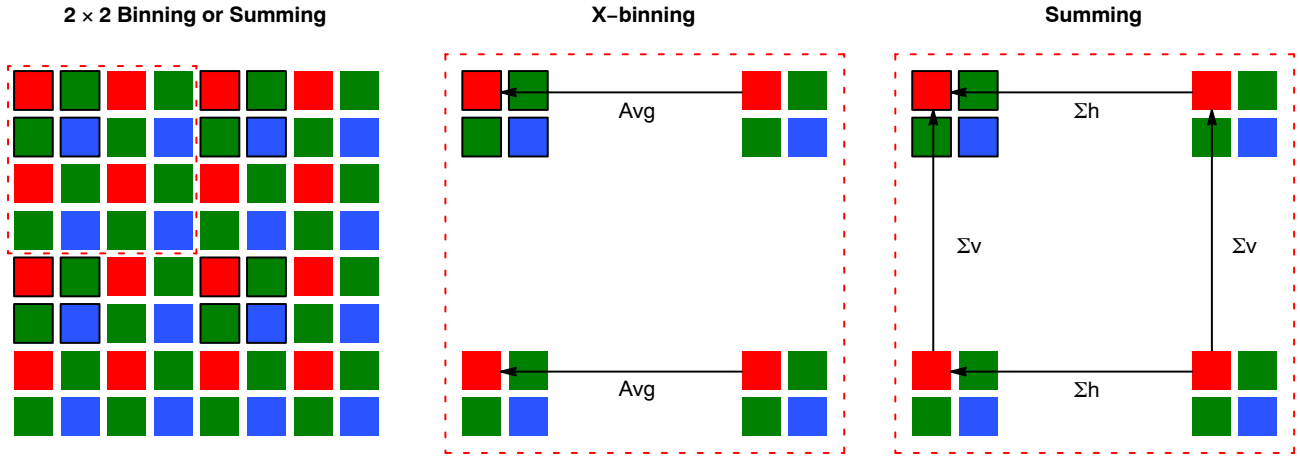


Figure 30. Pixel Binning and Summing

Scaler

Scaling reduces the size of the output image while maintaining the same field−of−view. The input and output of the scaler is in Bayer format.

When compared to skipping, scaling is advantageous as it avoids aliasing. The scaling factor, programmable in 1/16 steps, is used for horizontal and vertical scalers.

The AR0835HS sensor is capable of horizontal scaling and full (horizontal and vertical) scaling.

The scale factor is determined by:

- *n*, which is fixed at 16
- *m*, which is adjustable with register R0x0404
- Legal values for m are 16 through 96, giving the user the ability to scale from 1:1 (m=16) to 1:6 (m=96)

Frame Rate Control

The formulas for calculating the frame rate of the AR0835HS are shown below.

The line length is programmed directly in pixel clock periods through register line_length_pck. For a specific window size, the minimum line length can be found from Equation 8:

$$\text{minimum line\_length\_pck} = \left(\frac{\text{x\_addr\_end} - \text{x\_addr\_start} + 1}{\text{subsampling factor}} + \text{min\_line\_blanking\_pck}\right) \quad \text{(eq. 8)}$$

Note that line_length_pck also needs to meet the minimum line length requirement set in register min_line_length_pck. The row time can either be limited by the time it takes to sample and reset the pixel array for each row, or by the time it takes to sample and read out a row. Values for min_line_blanking_pck are provided in “Minimum Row Time”.

The frame length is programmed directly in number of lines in the register frame_line_length. For a specific window size, the minimum frame length can be found in Equation 9:

$$\text{minimum frame\_length\_lines} = \left(\frac{\text{y\_addr\_end} - \text{y\_addr\_start} + 1}{\text{subsampling factor}} + \text{min\_frame\_blanking\_lines}\right) \quad \text{(eq. 9)}$$

The frame rate can be calculated from these variables and the pixel clock speed as shown in Equation 10:

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

If coarse_integration_time is set larger than frame_length_lines the frame size will be expanded to coarse_integration_time + 1.

Minimum Row Time

Enough time must be given to the output FIFO so it can output all data at the set frequency within one row time.

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

- line_length_pck ≥ min_line_length_pck in Table 15
- The row time must allow the FIFO to output all data during each row. That is, line_length_pck ≥ (x_output_size × 2 + 0x005E) × “vt_pix_clk period” / “op_pix_clk period”

Minimum Frame Time

The minimum number of rows in the image is 1, so min_frame_length_lines will always equal (min_frame_blanking_lines + 1).

Table 15. MINIMUM FRAME TIME AND BLANKING NUMBERS

min_frame_blanking_lines	0x008F
min_frame_length_lines	0x0A1F

Integration Time

The integration (exposure) time of the AR0835HS is controlled by the coarse_integration_time register.

The limits for the coarse integration time are defined by:

$$\text{coarse\_integration\_time\_min} \le \text{coarse\_integration\_time} \quad \text{(eq. 11)}$$

The actual integration time is given by:

$$\text{integration\_time} = \frac{\text{coarse\_integration\_time} \times \text{line\_length\_pck}}{\text{vt\_pix\_clk\_freq\_mhz} \times 10^6} \quad \text{(eq. 12)}$$

It is required that:

$$\text{coarse\_integration\_time} \le (\text{frame\_length\_lines} - \text{coarse\_integration\_time\_max\_margin}) \quad \text{(eq. 13)}$$

If this limit is broken, the frame time will automatically be extended to (coarse_integration_time + coarse_integartion_time_max_margin) to accommodate the larger integration time.

In binning mode, frame_length_lines should be set larger than coarse_integration_time by at least 3 to avoid column imbalance artifact.

Flash Timing Control

The AR0835HS supports both xenon and LED flash timing through the FLASH output signal. The timing of the FLASH signal with the default settings is shown in Figure 31 (Xenon) and Figure 32 (LED). The flash and flash_count registers allow the timing of the flash to be changed. The flash can be programmed to fire only once,

delayed by a few frames when asserted, and (for xenon flash) the flash duration can be programmed.

Enabling the LED flash will cause one bad frame, where several of the rows only have the flash on for part of their integration time. This can be avoided either by first enabling mask bad frames (R0x301A[9] = 1) before the enabling the flash or by forcing a restart (R0x301A[1] = 1) immediately after enabling the flash; the first bad frame will then be masked out, as shown in Figure 32. Read−only bit flash[14] is set during frames that are correctly integrated; the state of this bit is shown in Figures 31 and 32.

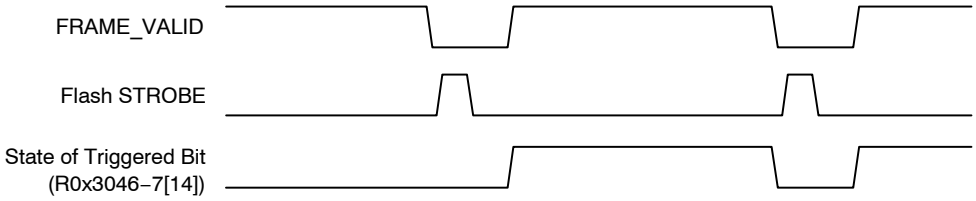
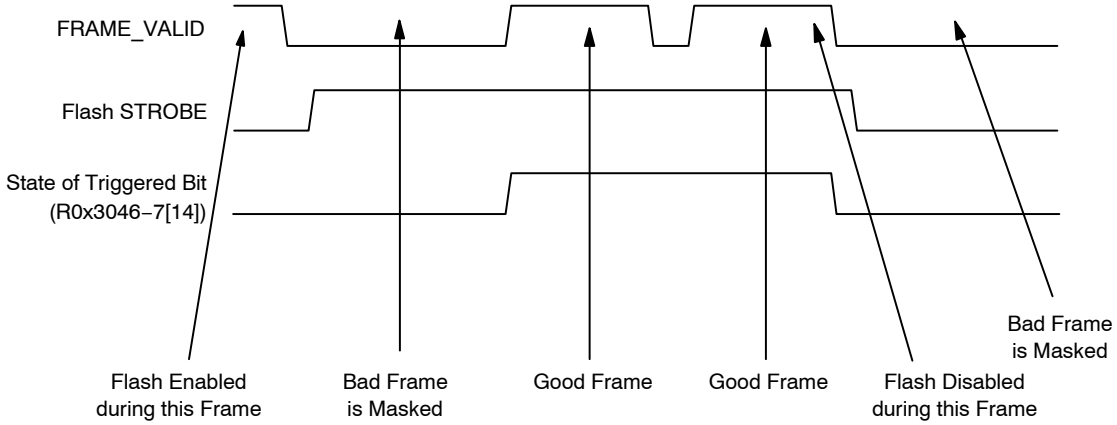


Figure 31. Xenon Flash Enabled



NOTE: An option to invert the flash output signal through R0x3046[7] is also available.

Figure 32. LED Flash Enabled

Global Reset Release (GRR)

Global reset release mode allows the integration time of the AR0835HS to be controlled by an external electromechanical shutter. GRR mode is generally used in conjunction with ERS mode. The ERS mode is used to provide viewfinder information, the sensor is switched into GRR mode to capture a single frame, and the sensor is then returned to ERS mode to restore viewfinder operation.

Overview of Global Reset Release Sequence

The basic elements of the GRR sequence are:

1. By default, the sensor operates in ERS mode and the SHUTTER output signal is LOW. The electromechanical shutter must be open to allow light to fall on the pixel array. Integration time is controlled by the coarse_integration_time register.
2. A global reset sequence is triggered.
3. All of the rows of the pixel array are placed in reset.
4. All of the rows of the pixel array are taken out of reset simultaneously. All rows start to integrate incident light. The electromechanical shutter may be open or closed at this time.
5. If the electromechanical shutter has been closed, it is opened.
6. After the desired integration time (controlled internally or externally to the AR0835HS), the electromechanical shutter is closed.
7. A single output frame is generated by the sensor with the usual LV, FV, PIXCLK, and D_{OUT} timing. As soon as the output frame has completed (FV negates), the electromechanical shutter may be opened again.
8. The sensor automatically resumes operation in ERS mode.

This sequence is shown in Figure 33. The following sections expand to show how the timing of this sequence is controlled.

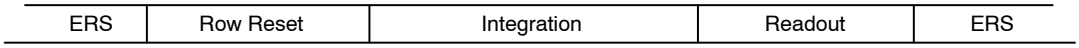


Figure 33. Overview of Global Reset Sequence

Entering and Leaving the Global Reset Sequence

A global reset sequence can be triggered by a register write to R0x315E global_seq_trigger[0] (global trigger, to transition this bit from a 0 to a 1) or by a rising edge on a suitably−configured GPI input).

When a global reset sequence is triggered, the sensor waits for the end of the current row. When LV negates for that row, FV is negated 6 PIXCLK periods later, potentially truncating the frame that was in progress.

The global reset sequence completes with a frame readout. At the end of this readout phase, the sensor automatically resumes operation in ERS mode. The first frame integrated with ERS will be generated after a delay of approximately ((13 + coarse_integration_time) × line_length_pck). This sequence is shown in Figure 34.

While operating in ERS mode, double−buffered registers (“Double−Buffered Registers”) are updated at the start of each frame in the usual way. During the global reset sequence, double−buffered registers are updated just before the start of the readout phase.

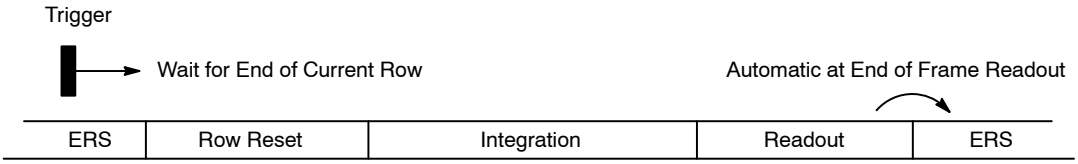


Figure 34. Entering and Leaving a Global Reset Sequence

Programmable Settings

The registers global_rst_end and global_read_start allow the duration of the row reset phase and the integration phase to be controlled, as shown in Figure 35. The duration of the readout phase is determined by the active image size.

The recommended setting for global_rst_end is 0x3160 (for example, 512 µs total reset time) with default vt_pix_clk. This allows sufficient time for all rows of the pixel array to be set to the correct reset voltage level. The row reset phase takes a finite amount of time due to the capacitance of the pixel array and the capability of the internal voltage booster circuit that is used to generate the reset voltage level.

As soon as the global_rst_end count has expired, all rows in the pixel array are taken out of reset simultaneously and the pixel array begins to integrate incident light.

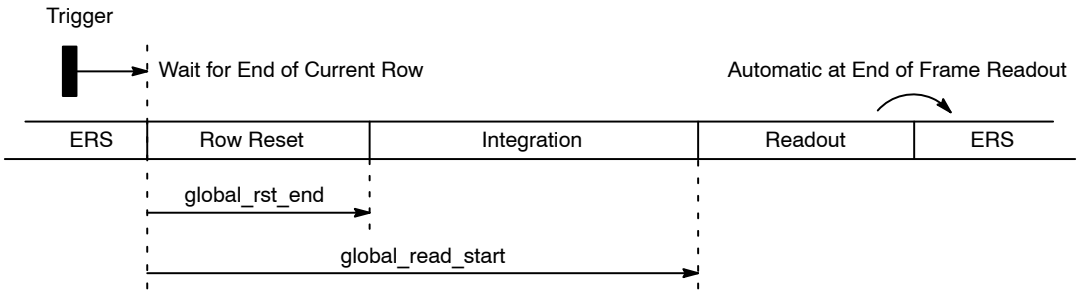


Figure 35. Controlling the Reset and Integration Phases of the Global Reset Sequence

Control of the Electromechanical Shutter

Figure 36 shows two different ways in which a shutter can be controlled during the global reset sequence. In both cases, the maximum integration time is set by the difference between global_read_start and global_rst_end. In shutter example 1, the shutter is open during the initial ERS sequence and during the row reset phase. The shutter closes during the integration phase. The pixel array is integrating incident light from the start of the integration phase to the point at which the shutter closes. Finally, the shutter opens again after the end of the readout phase. In shutter example 2, the shutter is open during the initial ERS sequence and closes sometime during the row reset phase. The shutter both opens and closes during the integration phase. The pixel array is integrating incident light for the part of the integration phase during which the shutter is open. As for the previous example, the shutter opens again after the end of the readout phase.

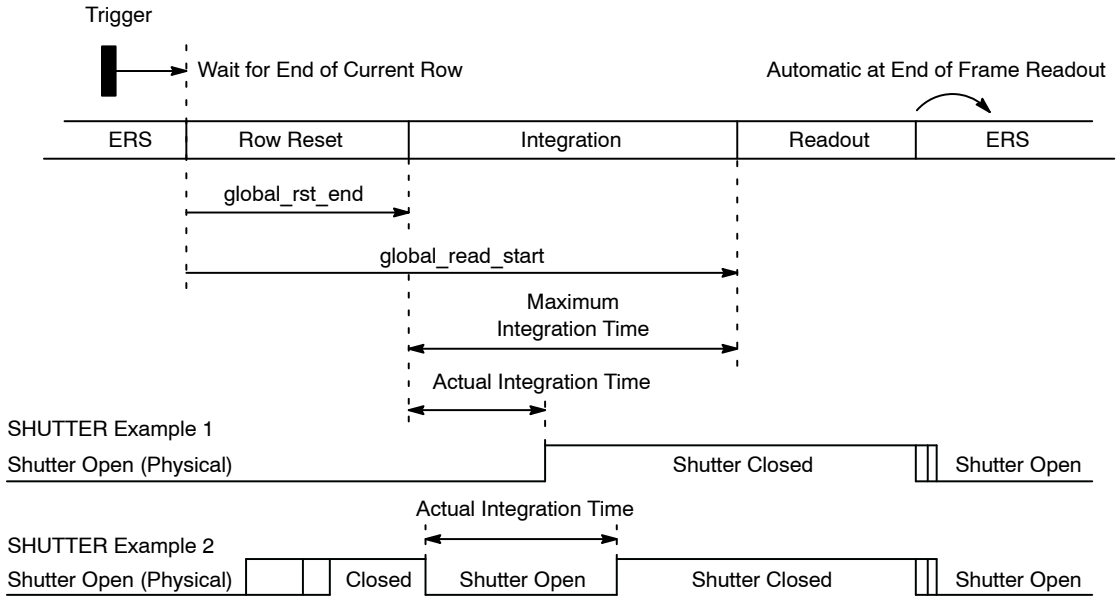


Figure 36. Control of the Electromechanical Shutter

It is essential that the shutter remains closed during the entire row readout phase (that is, until FV has negated for the frame readout); otherwise, some rows of data will be corrupted (over−integrated).

It is essential that the shutter closes before the end of the integration phase. If the row readout phase is allowed to start before the shutter closes, each row in turn will be integrated for one row−time longer than the previous row.

After FV negates to signal the completion of the readout phase, there is a time delay of approximately (10 x line_length_pck) before the sensor starts to integrate light−sensitive rows for the next ERS frame. It is essential that the shutter be opened at some point in this time window; otherwise, the first ERS frame will not be uniformly integrated.

The AR0835HS provides a SHUTTER output signal to control (or help the host system control) the electromechanical shutter. The timing of the SHUTTER output is shown in Figure 37. SHUTTER is negated by default. The point at which it asserts is controlled by the programming of global_shutter_start. At the end of the global reset readout phase, SHUTTER negates approximately (2 x line_length_pck) after the negation of FV.

This programming restriction must be met for correct operation:

- global_read_start > global_shutter_start

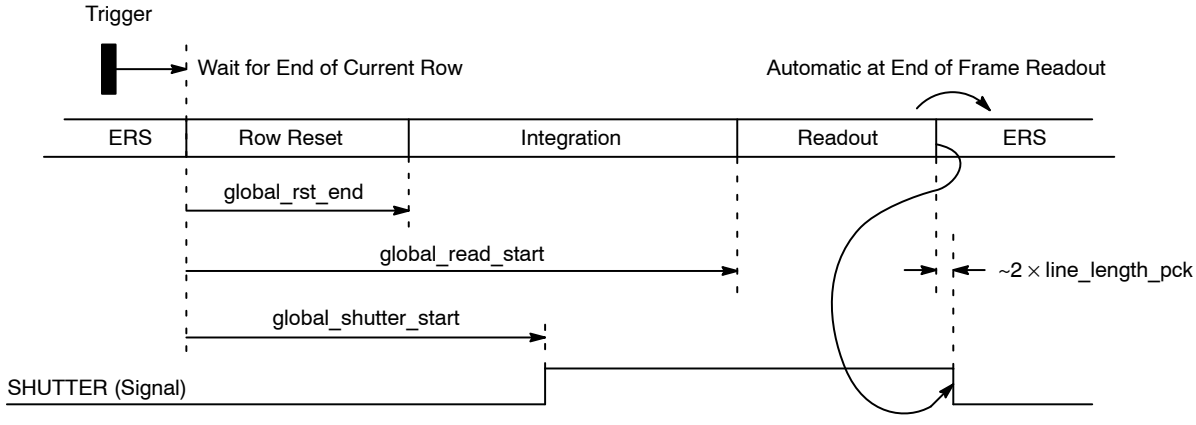


Figure 37. Controlling the SHUTTER Output

Using FLASH with Global Reset

If R0x315E global_seq_trigger[2] = 1 (global flash enabled) when a global reset sequence is triggered, the FLASH output signal will be pulsed during the integration phase of the global reset sequence. The FLASH output will assert a fixed number of cycles after the start of the integration phase and will remain asserted for a time that is controlled by the value of the flash_count register. When flash_count is programmed for value N, (where N is 0–0x3FE) the resulting flash duration is given by N × 512 x (1/vt_pix_clk_freq_mhz), as shown in Figure 38.

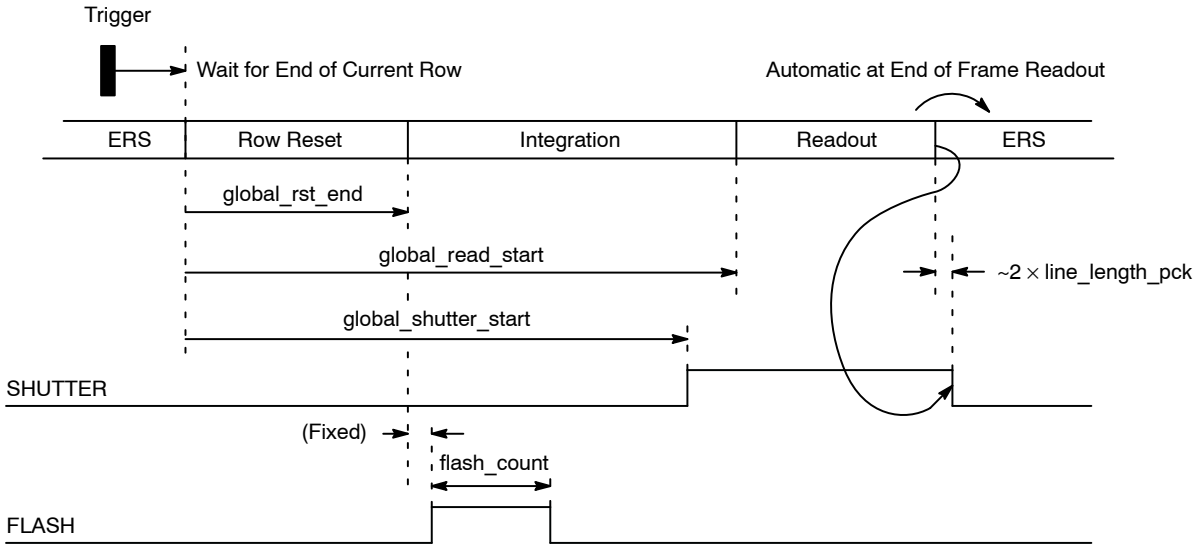


Figure 38. Using FLASH with Global Reset

When the flash_count = 0x3FF, the flash signal will be maximized and goes LOW when readout starts, as shown in Figure 39. This would be preferred if the latency in closing the shutter is longer than the latency for turning off the flash. This guarantees that the flash stays on while the shutter is open.

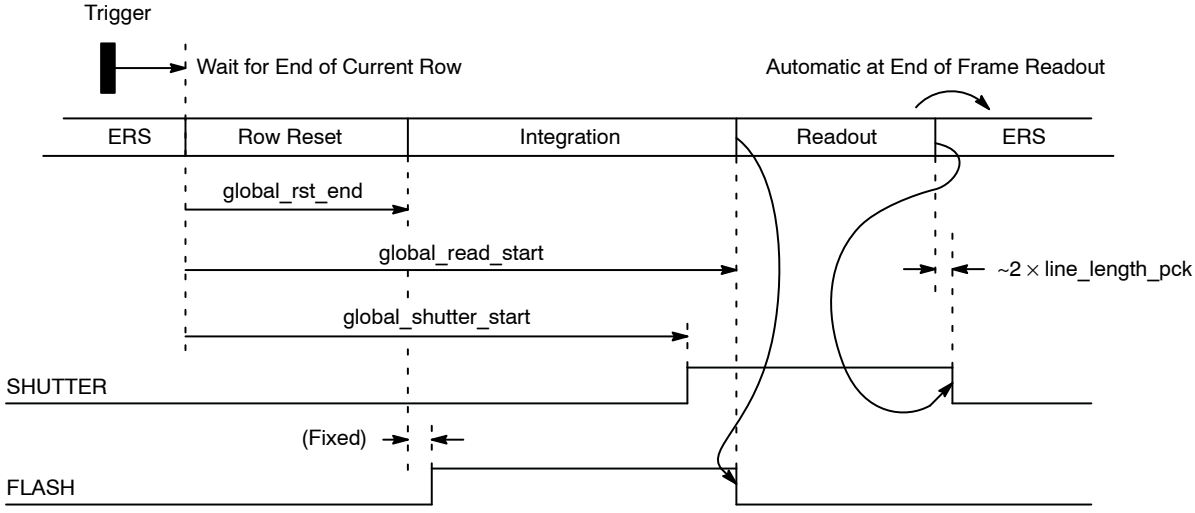


Figure 39. Extending FLASH Duration in Global Reset (Reference Readout Start)

External Control of Integration Time

If global_seq_trigger[1] = 1 (global bulb enabled) when a global reset sequence is triggered, the end of the integration phase is controlled by the level of trigger (global_seq_trigger[0] or the associated GPI input). This allows the integration time to be controlled directly by an input to the sensor.

This operation corresponds to the shutter "B" setting on a traditional camera, where "B" originally stood for "Bulb" (the shutter setting used for synchronization with a magnesium foil flash bulb) and was later considered to stand for "Brief" (an exposure that was longer than the shutter could automatically accommodate).

When the trigger is de−asserted to end integration, the integration phase is extended by a further time given by *global_read_start − global_shutter_start*. Usually this means that global_read_start should be set to *global_shutter_start + 1*.

The operation of this mode is shown in Figure 40. The figure shows the global reset sequence being triggered by the GPI2 input, but it could be triggered by any of the GPI inputs or by the setting and subsequence clearing of the global_seq_trigger[0] under software control.

The integration time of the GRR sequence is defined as:

$$\text{Integration Time} = \frac{\text{global\_scale} \times [\text{global\_read\_start} - \text{global\_shutter\_start} - \text{global\_rst\_end}]}{\text{vt\_pix\_clk\_freq\_mhz}} \quad \text{(eq. 14)}$$

where:

$$\text{global\_read\_start} = \left(2^{16} \times \text{global\_read\_start2[7:0]} + \text{global\_read\_start1[15:0]}\right) \quad \text{(eq. 15)}$$

$$\text{global\_shutter\_start} = \left(2^{16} \times \text{global\_shutter\_start2[7:0]} + \text{global\_shutter\_start1[15:0]}\right) \quad \text{(eq. 16)}$$

The integration equation allows for 24−bit precision when calculating both the shutter and readout of the image. The global_rst_end has only 16−bit as the array reset function and requires a short amount of time.

The integration time can also be scaled using global_scale. The variable can be set to 0–512, 1–2048, 2–128, and 3–32.

These programming restrictions must be met for correct operation of bulb exposures:

- global_read_start > global_shutter_start
- global_shutter_start > global_rst_end
- global_shutter_start must be smaller than the exposure time (that is, this counter must expire before the trigger is de−asserted)

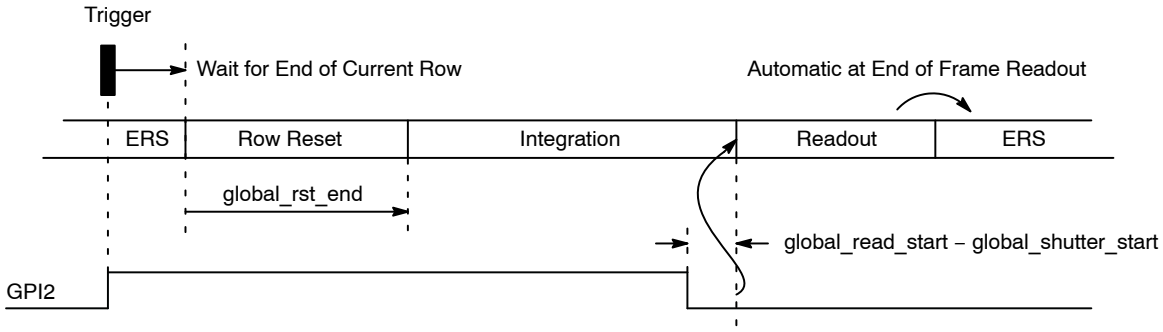


Figure 40. Global Reset Bulb

Retriggering the Global Reset Sequence

The trigger for the global reset sequence is edge−sensitive; the global reset sequence cannot be retriggered until the global trigger bit (in the R0x315E global_seq_trigger register) has been returned to "0", and the GPI (if any) associated with the trigger function has been negated.

The earliest time that the global reset sequence can be retriggered is the point at which the SHUTTER output negates; this occurs approximately (2 x line_length_pck) after the negation of FV for the global reset readout phase.

Using Global Reset with SMIA Data Path

When a global reset sequence is triggered, it usually results in the frame in progress being truncated (at the end of the current output line). The SMIA data path limiter function (see Figure 41) attempts to extend (pad) all frames to the programmed value of y_output_size. If this padding is still in progress when the global reset readout phase starts, the SMIA data path will not detect the start of the frame correctly. Therefore, to use global reset with the serial data path, this timing scenario must be avoided. One possible way of doing this would be to synchronize (under software control) the assertion of trigger to an end−of−frame marker on the serial data stream.

At the end of the readout phase of the global reset sequence, the sensor automatically resumes operation in ERS mode.

The frame that is read out of the sensor during the global reset readout phase has exactly the same format as any other frame out of the serial pixel data interface, including the addition of two lines of embedded data. The value of the coarse_integration_time register within the embedded data matches the programmed values of those registers and does not reflect the integration time used during the global reset sequence.

Global Reset and Soft Standby

If the R0x301A[2] mode_select[stream] bit is cleared while a global reset sequence is in progress, the AR0835HS will remain in streaming state until the global reset sequence (including frame readout) has completed, as shown in Figure 41.

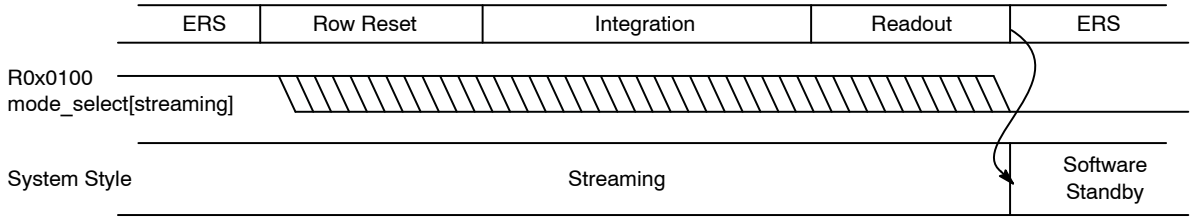


Figure 41. Entering Soft Standby During a Global Reset Sequence

Slave Mode

Slave mode is to ensure having an ERS−GRR−ERS transition without a broken ERS frame before GRR. It requests to trigger/end the GRR sequence through the pin which is similar to the GRR Bulb mode. The major difference to our existing sensor is to start the GRR sequence after the end of the current frame instead of to start immediately in the next following row.

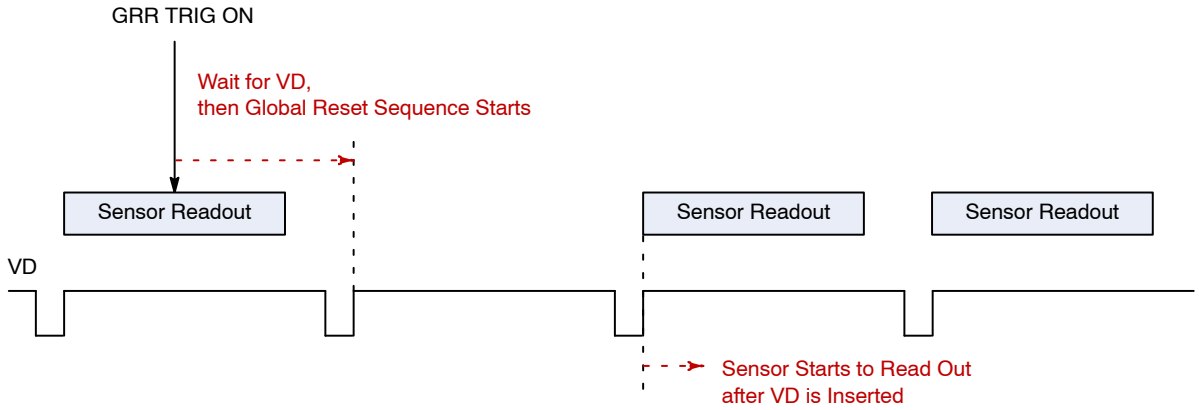


Figure 42. Slave Mode Transition

GAIN

AR0835HS supports both analog and digital gain.

Analog Gain

Analog gain is provided by colamp and ADC reference scaling (there is no ASC gain due to column parallel nature of architecture). Only global (not per−color) coarse gain can be set by analog gain. Global gain register (R0x305E) sets the analog gain. Bits [1:0] set the colamp gain while bits [4:2] are reserved for ADC gain. While the 2−bit colamp gain provides up to 4x analog gain, only LSB (bit [2]) of ADC gain bits is utilized to support 2x ADC gain. Table 16 is the recommended gain setting:

Table 16. RECOMMENDED ANALOG GAIN SETTING

Colamp Gain Codes (R0x305E[1:0])		ADC Gain Codes (R0x305E[4:2])			Colamp Gain	ADC Gain	Total Gain
0	0	0	0	0	1	1	1
0	1	0	0	0	2	1	2
1	0	0	0	0	3	1	3
1	1	0	0	0	4	1	4

Table 16. RECOMMENDED ANALOG GAIN SETTING (continued)

Colamp Gain Codes (R0x305E[1:0])		ADC Gain Codes (R0x305E[4:2])			Colamp Gain	ADC Gain	Total Gain
1	0	0	0	1	3	2	6
1	1	0	0	1	4	2	8

Digital Gain

Digital gain provides both per−color and fine (sub 1×) gain. The analog and digital gains are multiplicative to give the total gain. Digital gain is set by setting bits R0x305E[15:5] to set global gain or by individually setting digital color gain R0x3056−C[15:5] where these 11 bits are designed in 4p7 format i.e. 4 MSB provide gain up to 15× in step of 1× while 7 LSB provide sub−1× gain with a step size of 1/128. This sub−1× gain provides the fine gain control for the sensor.

Total Gain

Max. total gain required by design spec is 8× (analog) and 16× (digital) with min. step size of 1/8. The total gain equation can be formulated as:

$$\text{Total Gain} = (1 + \text{dec(R0x305D[1:0])}) \times (1 + \text{R0x305E[2]}) \times \frac{\text{dec(R0x305X[15:5])}}{128} \quad \text{(eq. 17)}$$

where X is 6, 8, A, C, for Gr, B, R and Gb, respectively.

NOTE: **onsemi** recommends using the registers mentioned above for gain settings. Avoid R0x3028 to R0x3038 unless their mapping to above registers is well understood and taken into account.

TEMPERATURE SENSOR

A standalone PTAT based temperature sensor has been implemented. The block is controlled independent of sensor timing and all communication happens through the two−wire serial interface.

INTERNAL VCM DRIVER

The AR0835HS utilizes an internal Voice Coil Motor (VCM) driver. The VCM functions are register−controlled through the serial interface.

There are two output ports, VCM_ISINK and VCM_GND, which would connect directly to the AF actuator.

Take precautions in the design of the power supply routing to provide a low impedance path for the ground connection. Appropriate filtering would also be required on the actuator supply. Typical values would be a 0.1 μF and 10 μF in parallel.

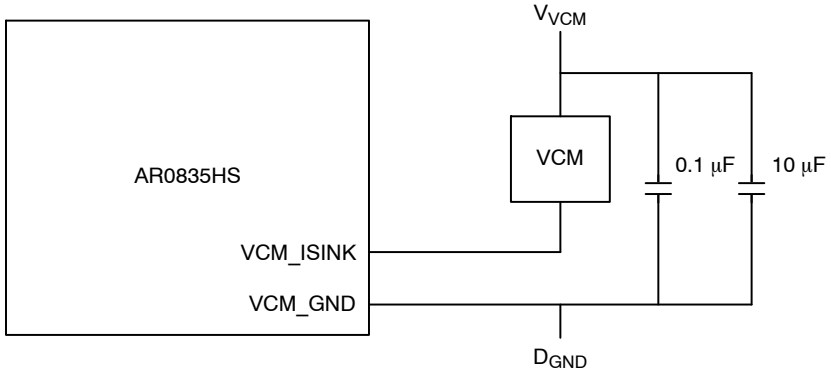


Figure 43. VCM Driver Typical Diagram

Table 17. VCM DRIVER TYPICAL

Characteristic	Parameter	Minimum	Typical	Maximum	Unit
VCM_OUT	Voltage at VCM Current Sink	2.5	2.8	3.3	V
WVCM	Voltage at VCM Actuator	2.5	2.8	3.3	V
INL	Relative Accuracy	–	±1.5	±4	LSB
RES	Resolution	–	8	–	bits
DNL	Differential Non−linearity	−1	–	+1	LSB
IVCM	Output Current	90	100	110	mA
	Slew Rate (User Programmable)	–	–	13	mA/ms

SPECTRAL CHARACTERISTICS

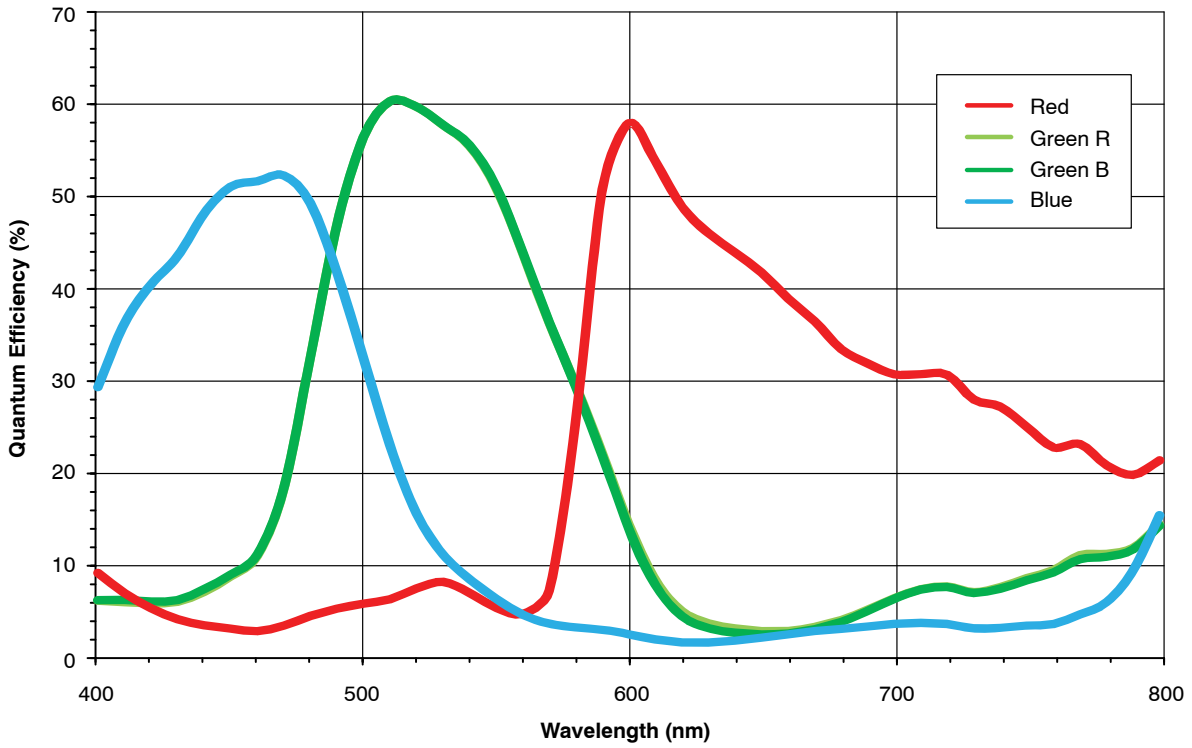


Figure 44. Quantum Efficiency

CRA vs. Image Height Plot

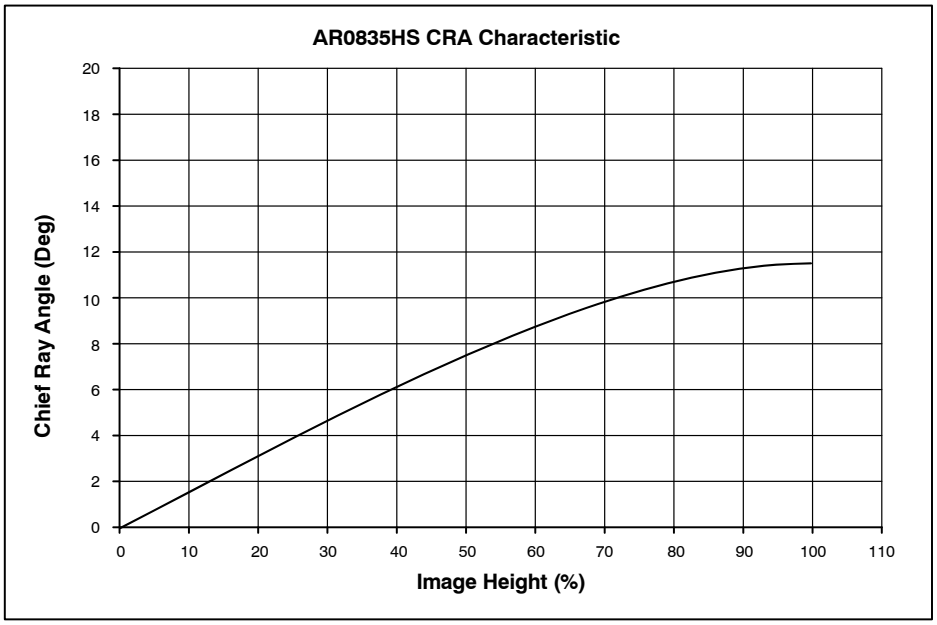


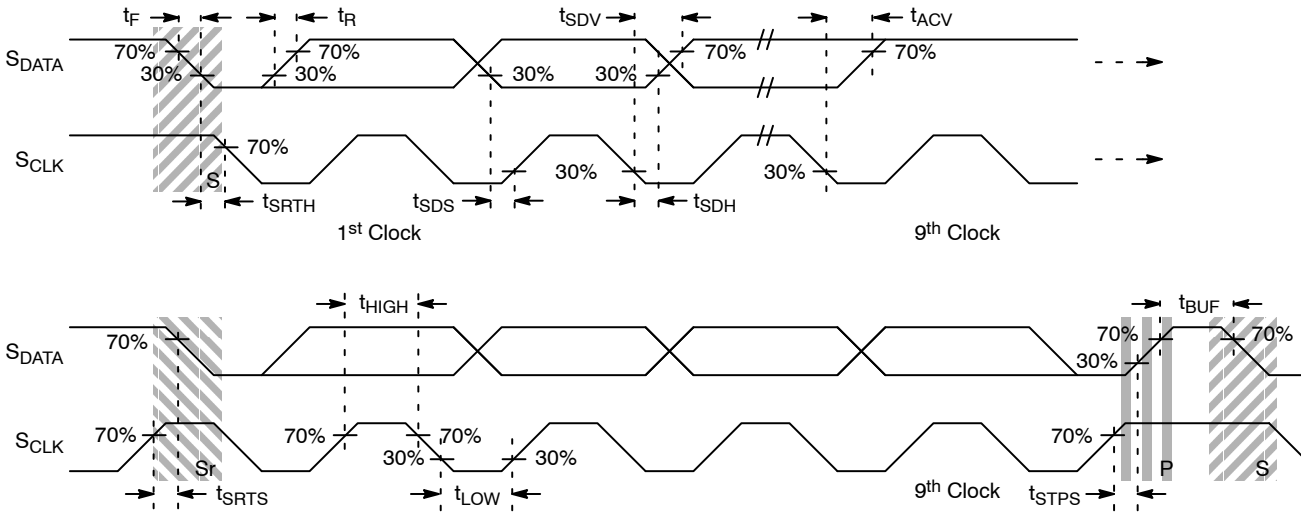
Figure 45. Chief Ray Angle vs. Image Height

Image Height		CRA
(%)	(mm)	(deg)
0	0	0
5	0.143	0.77
10	0.286	1.55
15	0.428	2.33
20	0.571	3.11
25	0.714	3.87
30	0.857	4.62
35	1.000	5.36
40	1.142	6.07
45	1.285	6.77
50	1.428	7.43
55	1.571	8.06
60	1.714	8.66
65	1.856	9.22
70	1.999	9.73
75	2.142	10.19
80	2.285	10.59
85	2.428	10.93
90	2.570	11.18
95	2.713	11.34
100	2.856	11.40

ELECTRICAL CHARACTERISTICS

Two−Wire Serial Register Interface

The electrical characteristics of the two−wire serial register interface (S_{CLK}, S_{DATA}) are shown in Figure 46 and Table 18. Table 19 shows the timing specification for the two−wire serial interface.



NOTE: Read sequence: For an 8−bit READ, read waveforms start after WRITE command and register address are issued.

Figure 46. Two−Wire Serial Bus Timing Parameters

Table 18. TWO−WIRE SERIAL REGISTER INTERFACE ELECTRICAL CHARACTERISTICS

(f_{EXTCLK} = 25 MHz; V_{AA} = 2.8 V; V_{AA_PIX} = 2.8 V; V_{DD_IO} = 1.8 V; V_{DD_1V2} = 1.2 V; V_{DD_PLL} = 1.2 V; V_{DD_1V8} = 1.8 V; Output load = 68.5 pF; T_J = 55°C)

Symbol	Parameter	Condition	Min	Typ	Max	Unit
V_{IL}	Input LOW Voltage		−0.5	–	0.3 × V_{DD_IO}	V
V_{IH}	Input HIGH Voltage		0.7 × V_{DD_IO}	–	V_{DD_IO} + 0.5	V
I_{IN}	Input Leakage Current	No Pull Up Resistor; V_{IN} = V_{DD_IO} or D_{GND}	10	–	14	μA
V_{OL}	Output LOW Voltage	At Specified 2 mA	0.11	–	0.3	V
I_{OL}	Output LOW Current	At Specified V_{OL} 0.1 V	–	–	6	mA
C_{IN}	Input Pad Capacitance		–	–	6	pF
C_{LOAD}	Load Capacitance		–	–	N/A	pF

Table 19. TWO−WIRE SERIAL INTERFACE TIMING SPECIFICATIONS

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

Symbol	Definition	Min	Max	Unit
f_{SCLK}	S_{CLK} Frequency	100	400	kHz
t_{HIGH}	S_{CLK} High Period	0.6	–	μs
t_{LOW}	S_{CLK} Low Period	1.3	–	μs
t_{SRTS}	START Setup Time	0.6	–	μs
t_{SRTH}	START Hold Time	0.6	–	μs
t_{SDS}	Data Setup Time	100	–	ns
t_{SDH}	Data Hold Time	0	–	μs
t_{SDV}	Data Valid Time	–	0.9	μs
t_{ACV}	Data Valid Acknowledge Time	–	0.9	μs
t_{STPS}	STOP Setup Time	0.6	–	μs

Table 19. TWO−WIRE SERIAL INTERFACE TIMING SPECIFICATIONS (continued)
(V_{DD}_IO = 1.7−1.9 V; V_{AA} = 2.4 −3.1 V; Environment temperature = −30°C to 50°C)

Symbol	Definition	Min	Max	Unit
t_{BUF}	Bus Free Time between STOP and START	1.3	–	μs
t_r	S_{CLK} and S_{DATA} Rise Time	–	300	ns
t_f	S_{CLK} and S_{DATA} Fall Time	–	300	ns

EXTCLK

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

If EXTCLK is AC−coupled to the AR0835HS and the clock is stopped, the EXTCLK input to the AR0835HS must be driven to ground or to V_{DD}_IO. Failure to do this will result in excessive current consumption within the EXTCLK input receiver.

Table 20. ELECTRICAL CHARACTERISTICS (EXTCLK)
(f_{EXTCLK} = 24 MHz; V_{AA} = 2.8 V; V_{AA}_PIX = 2.8 V; V_{DD}_IO = 1.8 V; V_{DD}_1V2 = 1.2 V, Output load = 68.5 pF; T_J = 55°C)

Symbol	Parameter	Condition	Min	Typ	Max	Unit
$f_{EXTCLK1}$	Input Clock Frequency	PLL Enabled	6	24	27	MHz
t_R	Input Clock Rise Slew Rate	C_{LOAD} < 20 pF	–	2.883	–	ns
t_F	Input Clock Fall Slew Rate	C_{LOAD} < 20 pF	–	2.687	–	ns
V_{IN_AC}	Input Clock Minimum Voltage Swing (AC Coupled)	–	0.5	–	–	V (p−p)
V_{IN_DC}	Input Clock Maximum Voltage Swing (DC Coupled)	–	–	–	V_{DD}_IO + 0.5	V
$f_{CLKMAX(AC)}$	Input Clock Signaling Frequency (Low Amplitude)	V_{IN} = V_{IN_AC} (MIN)	–	–	25	MHz
$f_{CLKMAX(DC)}$	Input Clock Signaling Frequency (Full Amplitude)	V_{IN} = V_{DD}_IO	–	–	48	MHz
	Clock Duty Cycle	–	45	50	55	%
t_{JITTER}	Input Clock Jitter	Cycle−to−Cycle	–	545	600	ps
t_{LOCK}	PLL VCO Lock Time	–	–	0.2	2	ms
C_{IN}	Input Pad Capacitance	–	–	6	–	pF
I_{IH}	Input HIGH Leakage Current	–	0	–	10	μA
V_{IH}	Input HIGH Voltage		0.7 × V_{DD}_IO	–	V_{DD}_IO + 0.5	V
V_{IL}	Input LOW Voltage		−0.5	–	0.3 × V_{DD}_IO	V

Product parametric performance is indicated in the Electrical Characteristics for the listed test conditions, unless otherwise noted. Product performance may not be indicated by the Electrical Characteristics if operated under different conditions.

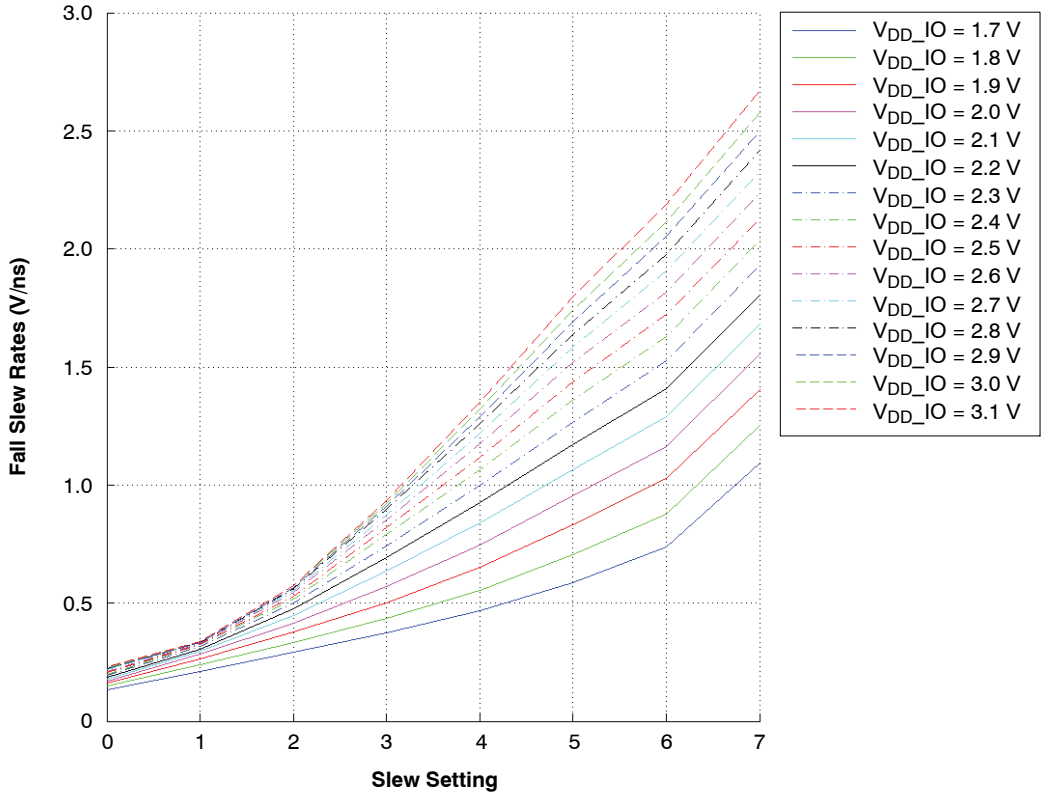


Figure 47. Fall Slew Rates (Cap Load = 25 pF)

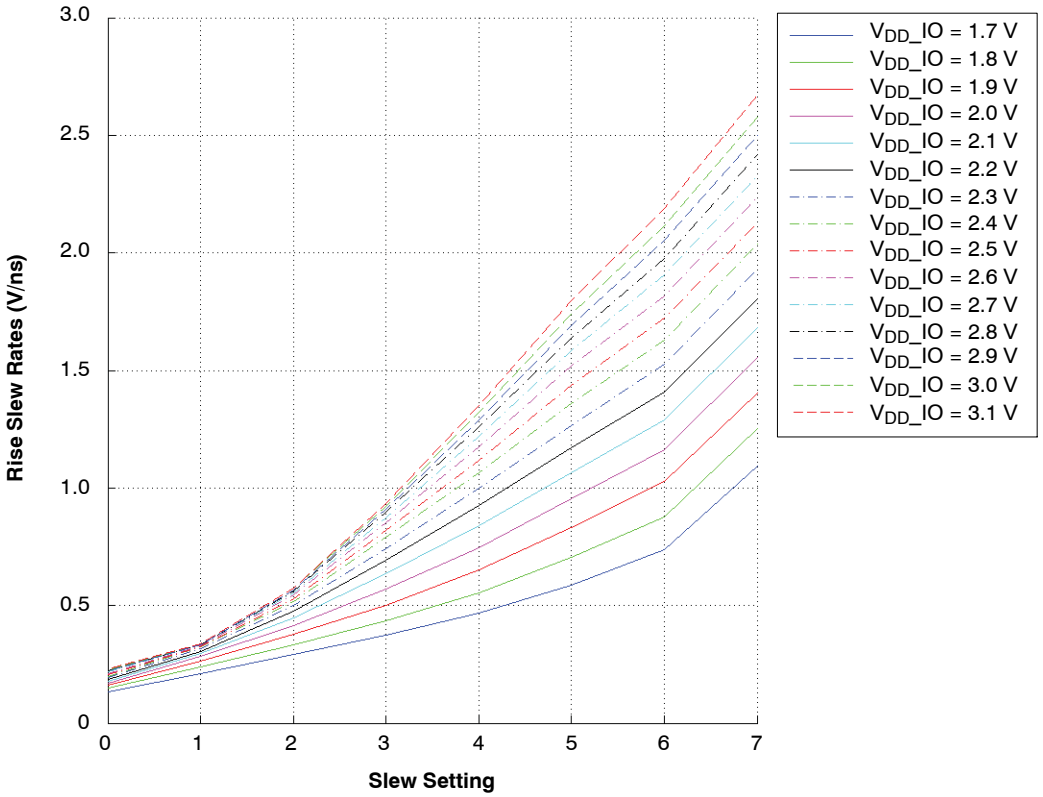


Figure 48. Rise Slew Rates (Cap Load = 25 pF)

High Speed Serial Pixel Data Interface

The High Speed Serial Pixel (HiSPi) interface uses four data and one clock low voltage differential signaling (LVDS) outputs.

- SLVSC_P
- SLVSC_N
- SLVS0_P
- SLVS0_N
- SLVS1_P
- SLVS1_N
- SLVS2_P
- SLVS2_N
- SLVS3_P
- SLVS3_N

The HiSPi interface supports three protocols, Streaming S, Streaming SP, and Packetized SP. The streaming protocols conform to a standard video application where each line of active or intra−frame blanking provided by the sensor is transmitted at the same length. The Packetized SP protocol will transmit only the active data ignoring line−to−line and frame−to−frame blanking data.

These protocols are further described in the High−Speed Serial Pixel (HiSPi) Interface Protocol Specification V1.50.00.

The HiSPi interface building block is a unidirectional differential serial interface with four data and one double data rate (DDR) clock lanes. One clock for every four serial data lanes is provided for phase alignment across multiple lanes. A collection of one clock lane plus four data lanes is called a PHY .Figure 49 shows the configuration between the HiSPi transmitter and the receiver.

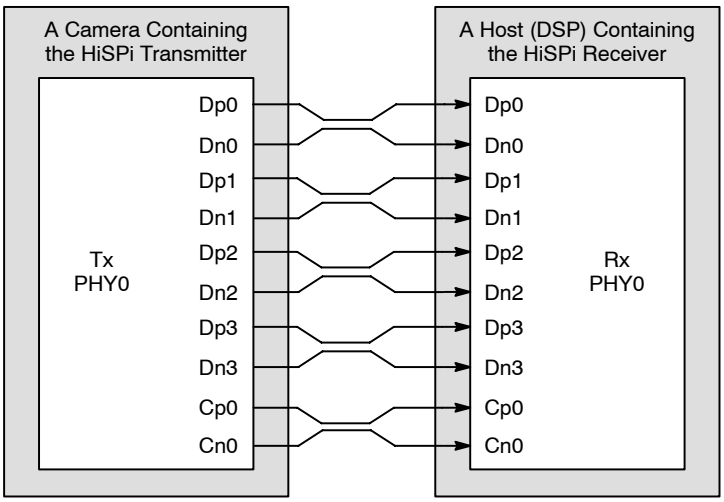


Figure 49. HiSPi Transmitter and Receiver Interface Block Diagram (HiSPi Mode Only)

HiSPi Physical Layer

The AR0835HS PHY will serialize an 8− or 10−bit data word and transmit each bit of data centered on a rising edge of the clock, the second on the falling edge of clock. Figure 50 shows bit transmission. In this example, the word is transmitted in order of MSB to LSB. The receiver latches data at the rising and falling edge of the clock.

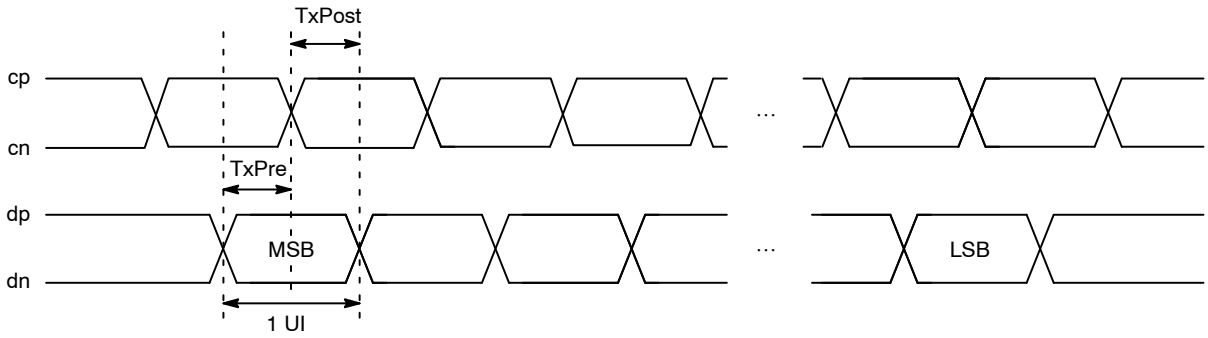


Figure 50. Timing Diagram

DLL Timing Adjustment

The specification includes a DLL to compensate for differences in group delay for each data lane. The DLL is connected to the clock lane and each data lane, which acts as a control master for the output delay buffers. Once the DLL has gained phase lock, each lane can be delayed in 1/8 unit interval (UI) steps. This additional delay allows the user to increase the setup or hold time at the receiver circuits and can be used to compensate for skew introduced in PCB design.

If the DLL timing adjustment is not required, the data and clock lane delay settings should be set to a default code of 0x000 to reduce jitter, skew, and power dissipation.

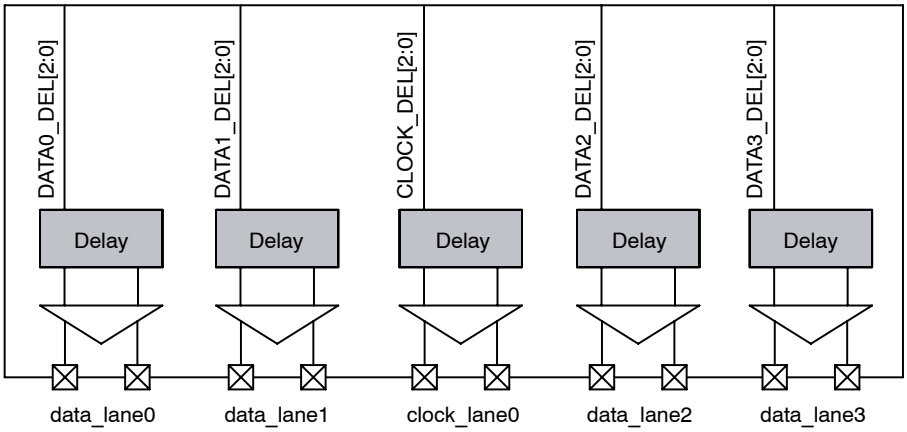


Figure 51. Block Diagram of DLL Timing Adjustment

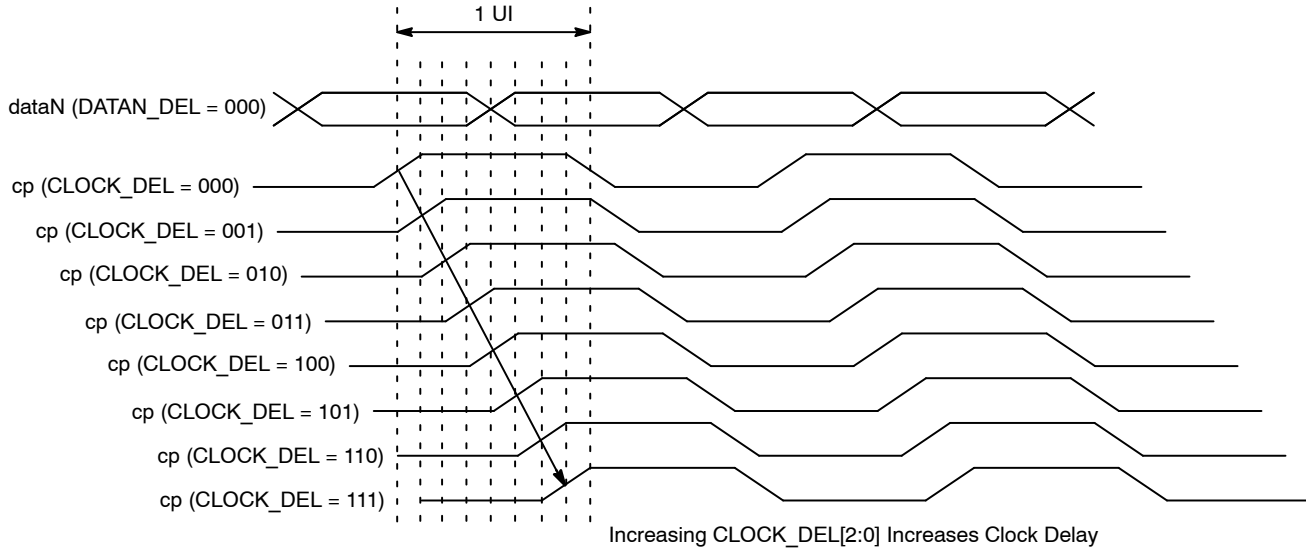


Figure 52. Delaying the clock_lane with Respect to data_lane

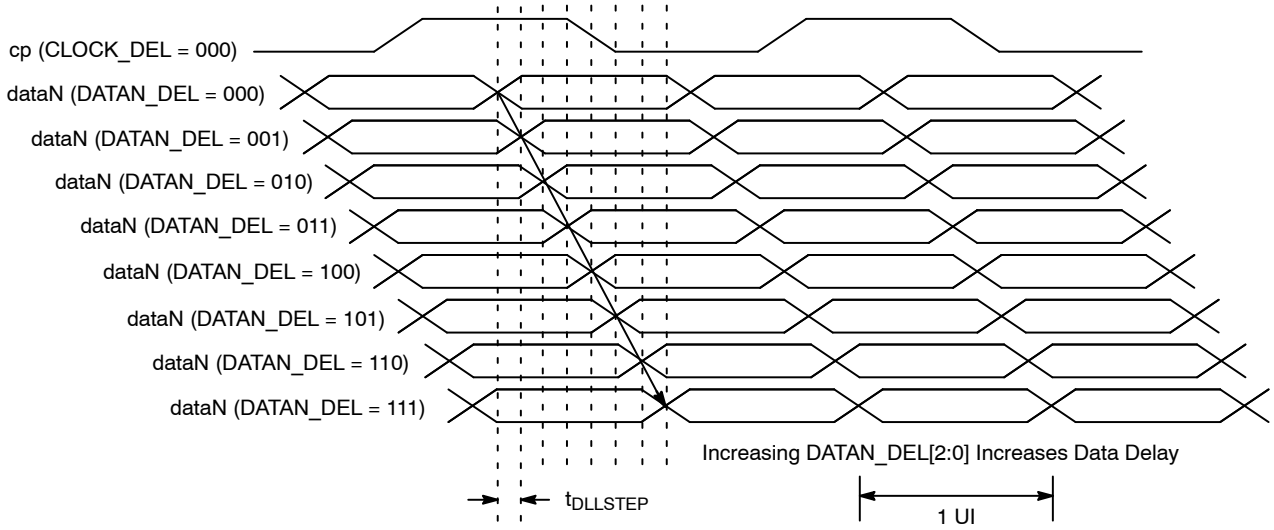


Figure 53. Delaying data_lane with Respect to the clock_lane

HiSPi Streaming Mode Protocol Layer

The HiSPi protocol is described HiSPi Protocol V1.50.00.

Serial Pixel Data Interface (HiSPi Mode)

The electrical characteristics of the serial pixel data interface (CLK_P, CLK_N, DATA[4:1]_P, and DATA[4:1]_N) are shown in Table 21 and Table 22.

Table 21. SLVS ELECTRICAL TIMING SPECIFICATION

Symbol	Parameter	Min	Max	Unit
1/UI	Data Rate (Note 3)	280	1000	Mbps
t_{PW}	Bitrate Period (Note 3)	1.00	3.57	ns
t_{PRE}	Max Setup Time from Transmitter (Notes 3, 4)		0.3	UI
t_{POST}	Max Hold Time from Transmitter (Notes 3, 4)		0.3	UI
t_{EYE}	Eye Width (Notes 3, 4)	0.6		UI
$t_{TOTALJIT}$	Data Total Jitter (pk−pk) @1e−9 (Notes 3, 4)		0.2	UI
t_{CKJIT}	Clock Period Jitter (RMS) (Note 4)		50	ps
t_{CYCJIT}	Clock Cycle−to−Cycle Jitter (RMS) (Note 4)		100	ps
t_R	Rise Time (20−80%) (Note 5)	150 ps	0.25	UI
t_F	Fall Time (20−80%) (Note 5)	150 ps	0.25	UI
D_{CYC}	Clock Duty Cycle (Note 4)	45	55	%
t_{CHSKEW}	Total Clock to Data Skew(Notes 3, 6)	−0.2	0.2	UI
$t_{DIFFSKEW}$	Mean Differential Skew (Note 7)	−100	100	ps

3. One UI is defined as the normalized mean time between one edge and the following edge of the clock.
4. Taken from the 0 V crossing point with the DLL off.
5. Also defined with a maximum loading capacitance of 10 pF on any pin. The loading capacitance may also need to be less for higher bitrates so the rise and fall times do not exceed the maximum 0.3 UI.
6. The total skew between the Clock lane and any Data Lane in the same PHY between any edges; it includes clock duty cycle, mean skew and total peak jitter at BER of 1E−9.
7. Differential skew is defined as the skew between complementary outputs. It is measured as the absolute time between the two complementary edges at mean V_{CM} point. Note that differential skew also is related to the ΔV_{CM_AC} spec which also must not be exceeded.

Table 22. SLVS ELECTRICAL DC SPECIFICATION (T_J = 25°C)

Symbol	Parameter	Min	Typ	Max	Unit
V_{CM}	SLVS DC Mean Common Mode Voltage	0.45 × V_{DD}_TX	0.5 × V_{DD}_TX	0.55 × V_{DD}_TX	V
$\|V_{OD}\|$	SLVS DC Mean Differential Output Voltage	0.36 × V_{DD}_T	0.5 × V_{DD}_TX	0.64 × V_{DD}_TX	V
ΔV_{CM}	Change in V_{CM} between Logic 1 and 0			25	mV
$\Delta\|V_{OD}\|$	Change in $\|V_{OD}\|$ between Logic 1 and 0			25	mV
NM	V_{OD} Noise Margin			±30	%
$\|\Delta V_{CM}\|$	Difference in V_{CM} between any Two Channels			30	mV
$\|\Delta V_{OD}\|$	Difference in V_{OD} between any Two Channels			50	mV
V_{CM_AC}	Common−mode AC Voltage (pk) without V_{CM} Cap Termination			50	mV
V_{CM_AC}	Common−mode AC Voltage (pk) with V_{CM} Cap Termination			30	mV
V_{OD_AC}	Maximum Overshoot Peak $\|V_{OD}\|$			1.3 × $\|V_{OD}\|$	V
V_{diff_pkpk}	Maximum Overshoot $V_{diff\ pk-pk}$			2.6 × V_{OD}	V
R_O	Single−ended Output Impedance	35	50	70	Ù
ΔR_O	Output Impedance Mismatch			20	%

Control Interface

The electrical characteristics of the control interface (RESET_BAR, TEST, GPIO0, GPIO1, GPI2, and GPI3) are shown in Table 23.

Table 23. DC ELECTRICAL CHARACTERISTICS (CONTROL INTERFACE)

(f_{EXTCLK} = 24 MHz; V_{AA} = 2.8 V; V_{AA}_PIX = 2.8 V; V_{DD}_IO = 1.8 V; DV_{DD}_1V2 = 1.2 V; Output load = 68.5 pF; T_J = 55°C)

Symbol	Parameter	Condition	Min	Typ	Max	Unit
V_{IH}	Input HIGH Voltage		0.7 × V_{DD}_IO	–	V_{DD}_IO + 0.5	V
V_{IL}	Input LOW Voltage		–0.5	–	V_{DD}_IO × 0.3	V
I_{IN}	Input Leakage Current	No pull−up resistor; V_{IN} = V_{DD}_IO or D_{GND}	–	–	10	μA
C_{IN}	Input Pad Capacitance		–	6	–	pF

Operating Voltages

V_{AA} and V_{AA}_PIX must be at the same potential for correct operation of the AR0835HS.

Table 24. DC ELECTRICAL DEFINITIONS AND CHARACTERISTICS

(f_{EXTCLK} = 24 MHz; V_{AA} = 2.8 V; V_{AA}_PIX = 2.8 V; V_{DD}_IO = 1.8 V; DV_{DD}_1V2 = 1.2 V; Output load = 68.5 pF; T_J = 70°C; Mode = Full Resolution (3264 × 2488); Frame rate = 30 fps)

Symbol	Parameter	Condition	Min	Typ	Max	Unit
V_{AA}	Analog Voltage		2.5	2.8	3.1	V
V_{AA}_PIX	Pixel Supply Voltage		2.5	2.8	3.1	V
V_{DD}_1V2	Digital Voltage		1.14	1.2	1.3	V
V_{DD}_1V8	PHY Digital Voltage		1.7	1.8	1.9	V
V_{DD}_IO	I/O Digital Voltage		1.7	1.8	1.9	V
			2.5	2.8	3.1	V
V_{DD}SLVS_PHY	HiSPi Analog Supply	Internal Regulator Disabled	0.35	0.4	0.45	V
		Internal Regulator Enabled	1.14	1.2	1.3	
V_{DD}_PHY	HiSPi Digital Supply		1.14	1.2	1.3	V
H/W Standby Current Consumption			–	–	30	μA

Table 24. DC ELECTRICAL DEFINITIONS AND CHARACTERISTICS (continued)
(f_{EXTCLK} = 24 MHz; V_{AA} = 2.8 V; V_{AA}_PIX = 2.8 V; V_{DD}_IO = 1.8 V; DV_{DD}_1V2 = 1.2 V; Output load = 68.5 pF; T_J = 70°C; Mode = Full Resolution (3264 × 2488); Frame rate = 30 fps)

Symbol	Parameter	Condition	Min	Typ	Max	Unit
Output Driving Strength			10	–	–	mA
Slew Rate			–	0.7	–	μV/sec
Programming Voltage for OTPM (V_{PP})			6	6.5	7	V

Table 25. TYPICAL OPERATING CURRENT CONSUMPTION
(Nominal Voltages: f_{EXTCLK} = 24 MHz; V_{AA} = 2.8 V; V_{AA}_PIX = 2.8 V; V_{DD}_1V8 = 1.8 V; V_{DD}_IO = 1.8 V; V_{DD}_1V2 = 1.2 V T_J = 25°C)

Symbol	Parameter		Min	Typ	Max	Unit
I(V_{AA})	Analog Supply Current	8 Mp, 46 fps	–	85	110	mA
		6 Mp, 60 fps	–	85	110	mA
		1080p, 60 fps	–	85	110	mA
		720p, 120 fps	–	85	110	mA
I(V_{AA}_PIX)	Pixel Supply Current	8 Mp, 46 fps	–	16	20	mA
		6 Mp, 60 fps	–	16	20	mA
		1080p, 60 fps	–	16	20	mA
		720p, 120 fps	–	16	20	mA
I(V_{DD}_1V8)	OTPM Read Supply Current	8 Mp, 46 fps	–	0	1	mA
		6 Mp, 60 fps	–	0	1	mA
		1080p, 60 fps	–	0	1	mA
		720p, 120 fps	–	0	1	mA
I(V_{DD}_IO)	I/O Supply Current	8 Mp, 46 fps	–	1	2	mA
		6 Mp, 60 fps	–	1	2	mA
		1080p, 60 fps	–	1	2	mA
		720p, 120 fps	–	1	2	mA
I(V_{DD}_1V2: V_{DD}SW, V_{DD}_ANA, V_{DD}_PLL)	Core Supply Current	8 Mp, 46 fps	–	150	175	mA
		6 Mp, 60 fps	–	150	175	mA
		1080p, 60 fps	–	130	165	mA
		720p, 120 fps	–	130	140	mA
I(V_{DD}_PHY, V_{DD}SLVS_PHY)	PHY Supply Current	8 Mp, 46 fps	–	12	14	mA
		6 Mp, 60 fps	–	12	14	mA
		1080p, 60 fps	–	10	12	mA
		720p, 120 fps	–	8	10	mA

Absolute Minimum and Maximum Ratings

CAUTION: Stresses greater than those listed in Table 26 may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect reliability. This is a stress rating only, and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

Table 26. ABSOLUTE MAXIMUM VOLTAGES

Symbol	Parameter	Min	Max	Unit
1.2V	All1.2 V Supply	−0.3	1.5	V
1.8V	All1.8 V Supply	−0.3	2.1	V
2.8V	All 2.8 V Supply	−0.3	3.5	V

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

HiSPi SPECIFICATION REFERENCE

The sensor design and this documentation is based on the following HiSPi Specifications:

- HiSPi Protocol Specification V1.50.00
- HiSPi Physical Layer Specification V3.0

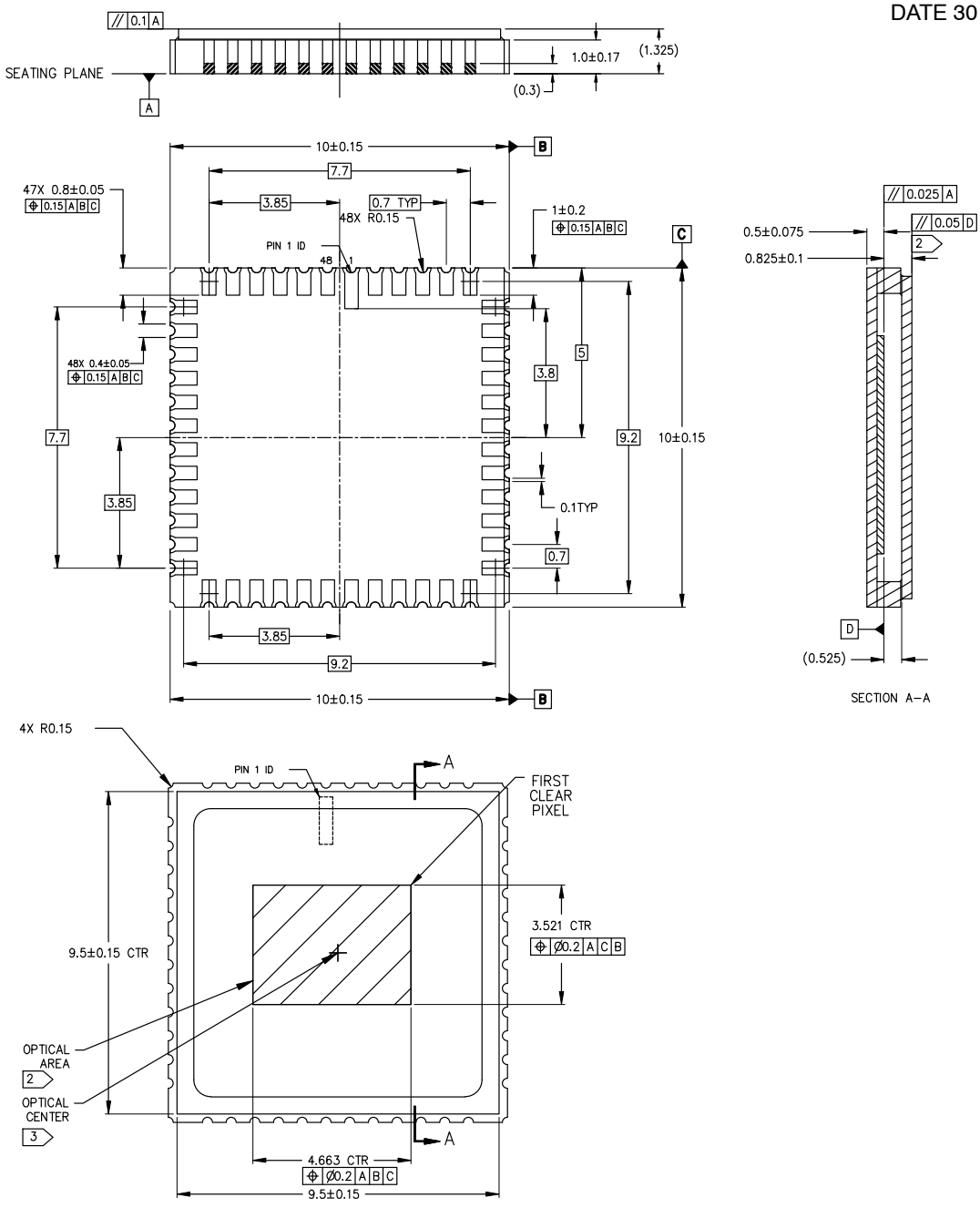
MIPI SPECIFICATION REFERENCE

The sensor design and this documentation is based on the following MIPI Specifications:

- MIPI Alliance Standard for CSI−2 version 1.0
- MIPI Alliance Standard for D−PHY version 1.0

CLCC48 10x10
CASE 848AJ
ISSUE O

DATE 30 DEC 2014



NOTES

1 DIMENSIONS IN MM. DIMENSIONS IN ( ) ARE FOR REFERENCE ONLY.

2 MAXIMUM ROTATION OF OPTICAL AREA RELATIVE TO PACKAGE EDGES: 1°.
MAXIMUM TILT OF OPTICAL AREA RELATIVE TO SUBSTRATE PLANE A: 25 MICRONS.
MAXIMUM TILT OF COVER GLASS RELATIVE TO OPTICAL AREA PLANE D: 50 MICRONS.

3 OPTICAL CENTER = PACKAGE CENTER.

DOCUMENT NUMBER:	**98AON93121F**	Electronic versions are uncontrolled except when accessed directly from the Document Repository. Printed versions are uncontrolled except when stamped "CONTROLLED COPY" in red.
DESCRIPTION:	**CLCC48 10X10**	**PAGE 1 OF 1**