S5K33DX

1/3.2" VGA 7um Pixel Indirect CW ToF sensor

Revision 0.40
August 2019

Application Note

SAMSUNG ELECTRONICS RESERVES THE RIGHT TO CHANGE PRODUCTS, INFORMATION AND SPECIFICATIONS WITHOUT NOTICE.

Products and specifications discussed herein are for reference purposes only. All information discussed herein is provided on an "AS IS" basis, without warranties of any kind.

This document and all information discussed herein remain the sole and exclusive property of Samsung Electronics. No license of any patent, copyright, mask work, trademark or any other intellectual property right is granted by one party to the other party under this document, by implication, estoppel or otherwise.

Samsung products are not intended for use in life support, medical, or safety equipment or any military or defense application, or any governmental procurement to which special terms or provisions may apply. Samsung products intended for automotive application are not fail-safe or error-free and that a driving assistance system or other similar system incorporation Samsung products may be prone to failures of safety-critical functions if used without proper safety technology measures in the form of hardware, software, information, and time redundancy. Accordingly, Samsung disclaims all liability arising from any and all failures of a safety-critical function caused by any error or failure of Samsung products.

For updates or additional information about Samsung products, contact your nearest Samsung office.
All brand names, trademarks and registered trademarks belong to their respective owners

Important Notice

Samsung Electronics Co. Ltd. (“Samsung”) reserves the right to make changes to the information in this publication at any time without prior notice. All information provided is for reference purpose only. Samsung assumes no responsibility for possible errors or omissions, or for any consequences resulting from the use of the information contained herein.

This publication on its own does not convey any license, either express or implied, relating to any Samsung and/or third-party products, under the intellectual property rights of Samsung and/or any third parties.

Samsung makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Samsung assume any liability arising out of the application or use of any product or circuit and specifically disclaims any and all liability, including without limitation any consequential or incidental damages.

Customers are responsible for their own products and applications. "Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by the customer's technical experts.

Samsung products are not designed, intended, or authorized for use in applications intended to support or sustain life, or for any other application in which the failure of the Samsung product could reasonably be expected to create a situation where personal injury or death may occur. Customers acknowledge and agree that they are solely responsible to meet all other legal and regulatory requirements regarding their applications using Samsung products notwithstanding any information provided in this publication. Customer shall indemnify and hold Samsung and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, expenses, and reasonable attorney fees arising out of, either directly or indirectly, any claim (including but not limited to personal injury or death) that may be associated with such unintended, unauthorized and/or illegal use.

Samsung Electronics Co., Ltd.
San #24 Nongseo-Dong, Giheung-Gu
Yongin-City, Gyeonggi-Do, Korea 446-711

Contact Us: dong.shin@samsung.com

Home Page: *http://www.samsungsemi.com*

Trademarks

All brand names, trademarks and registered trademarks belong to their respective owners.

- Exynos, FlexOneNAND, and OneNAND are trademarks of Samsung Electronics.
- ARM, Jazelle, TrustZone, and Thumb are registered trademarks of ARM Limited.
- Cortex, ETM, ETB, Coresight, ISA, and Neon are trademarks of ARM Limited.
- Java is a trademark of Sun Microsystems, Inc.
- SD is a registered trademark of Toshiba Corporation.
- MMC and eMMC are trademarks of MultiMediaCard Association.
- JTAG is a registered trademark of JTAG Technologies, Inc.
- Synopsys is a registered trademark of Synopsys, Inc.
- I2S is a trademark of Phillips Electronics.
- I^2C is a trademark of Phillips Semiconductor Corp.
- MIPI and Slimbus are registered trademarks of the Mobile Industry Processor Interface (MIPI) Alliance.

All other trademarks used in this publication are the property of their respective owners.

Chip Handling Guide

Precaution against Electrostatic Discharge

When using semiconductor devices, ensure that the environment is protected against static electricity:

1. Wear antistatic clothes and use earth band.
2. All objects that are in direct contact with devices must be made up of materials that do not produce static electricity.
3. Ensure that the equipment and work table are earthed.
4. Use ionizer to remove electron charge.

Contamination

Do not use semiconductor products in an environment exposed to dust or dirt adhesion.

Temperature/Humidity

Semiconductor devices are sensitive to:

- Environment
- Temperature
- Humidity

High temperature or humidity deteriorates the characteristics of semiconductor devices. Therefore, do not store or use semiconductor devices in such conditions.

Mechanical Shock

Do not to apply excessive mechanical shock or force on semiconductor devices.

Chemical

Do not expose semiconductor devices to chemicals because exposure to chemicals leads to reactions that deteriorate the characteristics of the devices.

Light Protection

In non- Epoxy Molding Compound (EMC) package, do not expose semiconductor IC to bright light. Exposure to bright light causes malfunctioning of the devices. However, a few special products that utilize light or with security functions are exempted from this guide.

Radioactive, Cosmic and X-ray

Radioactive substances, cosmic ray, or X-ray may influence semiconductor devices. These substances or rays may cause a soft error during a device operation. Therefore, ensure to shield the semiconductor devices under environment that may be exposed to radioactive substances, cosmic ray, or X-ray.

EMS (Electromagnetic Susceptibility)

Strong electromagnetic wave or magnetic field may affect the characteristic of semiconductor devices during the operation under insufficient PCB circuit design for Electromagnetic Susceptibility (EMS).

Revision History

Revision No.	Date	Description	Author (s)
0.0	Mar. 13, 2019	• Initial Version Draft	Y.D. Chang
0.1	Jul. 22, 2019	• Add “5.2 Modulation frequency control” and modify "Table 61 CXA4016 Write Control Registers"	Y.D. Chang
0.2	Jul.26, 2019	• Add examples for integration time, modulation frequency, frame rate and pixel rate	Y.D. Chang
0.3	August 9, 2019	• Add "Table 62 CXA4016 Write Control Registers"	Y.D. Chang
0.4	August 13, 2019	• Correct "Table 61 CXA4016 Write Control Registers"	Y.D. Chang

Table of Contents

List of Figures

List of Tables

List of Conventions

Register RW Access Type Conventions

Type	Definition	Description
R	Read Only	The application has permission to read the Register field. Writes to read-only fields have no effect.
W	Write Only	The application has permission to write in the Register field.
RW	Read & Write	The application has permission to read and writes in the Register field. The application sets this field by writing 1'b1 and clears it by writing 1'b0.

Register Value Conventions

Expression	Description
x	Undefined bit
X	Undefined multiple bits
?	Undefined, but depends on the device or pin status
Device dependent	The value depends on the device
Pin value	The value depends on the pin status

Reset Value Conventions

Expression	Description
0	Clears the register field
1	Sets the register field
x	Don't care condition

Warning: Some bits of control registers are driven by hardware or write operation only. As a result the indicated reset value and the read value after reset might be different.

1 Product Overview

1.1 Overview

S5K33DX is a highly integrated indirect Time-of-Flight (ToF) sensor aimed for 3D depth sensing. S5K33DX has VGA resolution and 7 µm pixel. The camera is integrated in a ToF system that also includes a modulated light source in Infrared (IR) band which is controlled by S5K33DX. S5K33DX chip includes demodulation ToF pixel array, analog/mixed-signal processing circuit including modulation signal generation and transmission, image correction functionality, low-power depth calculation chain, and serial transmission using up to 2-lane MIPI transmitter.

It is designed for fast but low-power operation, delivering full resolution capture at up to 60 frames per second (fps) for raw image data. It is fabricated by the Samsung CMOS image sensor process to realize a high-efficiency and low-power photo sensor. The sensor consists of 640×480 effective pixels (644×484 active pixels) that meet with the 1/3.2-inch optical format.

The indirect ToF depth sensing concept requires four different samples of the pixel data in four different phases to calculate its depth value.

This sensor is designed as a 2-tap or 4-tap sensor. In case of a 2-tap sensor, each pixel can sample two phases of data simultaneously. Each phase is considered internally as an independent pixel so conceptually, a 2-tap pixel array is expanded to 1280x480 output raw image data. In case of a 4-tap sensor, each pixel can sample four phases of data simultaneously. Each phase is considered internally as an independent pixel so conceptually a 4-tap pixel array is expanded to 1280x960 output raw image data.

The main application of the sensor is to output RAW phase data that are translated to depth image by post processing (either in S/W or in H/W) at the back-end chip which is an Application Processor (AP).

This sensor also includes simple H/W depth engine that provides low-accuracy depth image for simple depth applications such as proximity sensing. The depth engine can be enabled only in certain sensor output modes.

This sensor has on-chip 12-bit ADC arrays to digitize the pixel output and on-chip Correlated Double Sampling (CDS) to drastically reduce pixel reset noise and Fixed-Pattern Noise (FPN). It incorporates on-chip camera functions such as defect correction, exposure setting, image scaling, depth and proximity sensing, and auto exposure statistics.

S5K33DX CIS is programmable through a CCI or SPI serial interface and includes an on-chip One-Time Programmable (OTP) Non-Volatile Memory (NVM).

S5K33DX is suitable for a low-power camera module with a 2.8 V/1.05 V power supply.

1.2 Features

S5K33DX supports the following features:

- VGA 4:3 Indirect Time-Of-Flight sensor with 2-tap or 4-tap outputs per pixel and 1/3.2" optics
- Unit pixel size: 7.0 μm
- Effective resolution: 640 (H) × 480 (V) for pixel array, 1280 (H) × 480 (V) for 2-tap output raw image, 1280 (H) × 960 (V) for 4-tap output raw image
- Active resolution: 644 (H) × 484 (V) for pixel array, 1288 (H) × 484 (V) for 2-tap output raw image, 1288 (H) × 968 (V) for 4-tap output raw image
- Shutter type: Electronic global shutter
- Modes:

Resolution	Mode	Array Mode	Out Mode	H	V	Frame Rate
VGA	Full	2-tap	RAW phase	1288	484	120fps
VGA	Full	4-tap	RAW phase	1288	968	60fps
QVGA	2x2 binning	2-tap	RAW phase	640	240	180fps
QVGA	2x2 binning	4-tap	RAW phase	640	480	90fps
16x16	16x16 crop	4-tap	Proximity (no output)	32	32	10fps
VGA	Full	4-tap	Depth	644	484	60fps
VGA	Full	4-tap	Depth+Confidence	1932(**)	484	60fps

(**) - Depth + Confidence image contains 3 different attributes per pixel: Depth, Intensity and Amplitude so image width is 644x3=1932

- Interfaces:
  - Fine interface frequency control using additional dedicated PLL for EMI avoidance and integration flexibility.
  - MIPI CSI2: four lanes (1.6Gbps per lane)
  - Output formats: RAW12
- Control interface:
  - SPI interface: Four-wire serial communication circuit up to 20 MHz
  - Camera Control Interface (CCI) high-speed I2C-compatible - Two-wire serial communication circuit up to 1 MHz
- 32 Kb on-chip OTP memory mapped BPC and chip ID
- Vertical and horizontal flip mode
- Continuous frame capture mode
- Vertical average readout: x2, x4
- Horizontal digital binning: x2, x4

- On-sensor Depth Chain
- Low Power Proximity sensing mode
- Pixel skip readout function
- Mapped bad pixel correction
- Built-in test pattern generation
- Supply voltage: 2.8 V for analog, 1.8 V or 2.8 V for I/O, and 1.05 V for digital core supply
- Operating temperature: 0°C to + 60°C

1.3 Functional Block Diagram

1.3.1 Block Diagram

S5K33DX is a progressive-scan sensor that generates a stream of pixel data at a constant frame rate. It uses an on-chip Phase-Locked Loop (PLL) to generate all internal clocks from a single master input clock running between 12 MHz and 60 MHz. Dedicated PLL generates the output interface clocks for maximum flexibility in interface frequency and for avoiding EMI.

The block diagram of the sensor is illustrated in *Figure 1 Functional Block Diagram*.

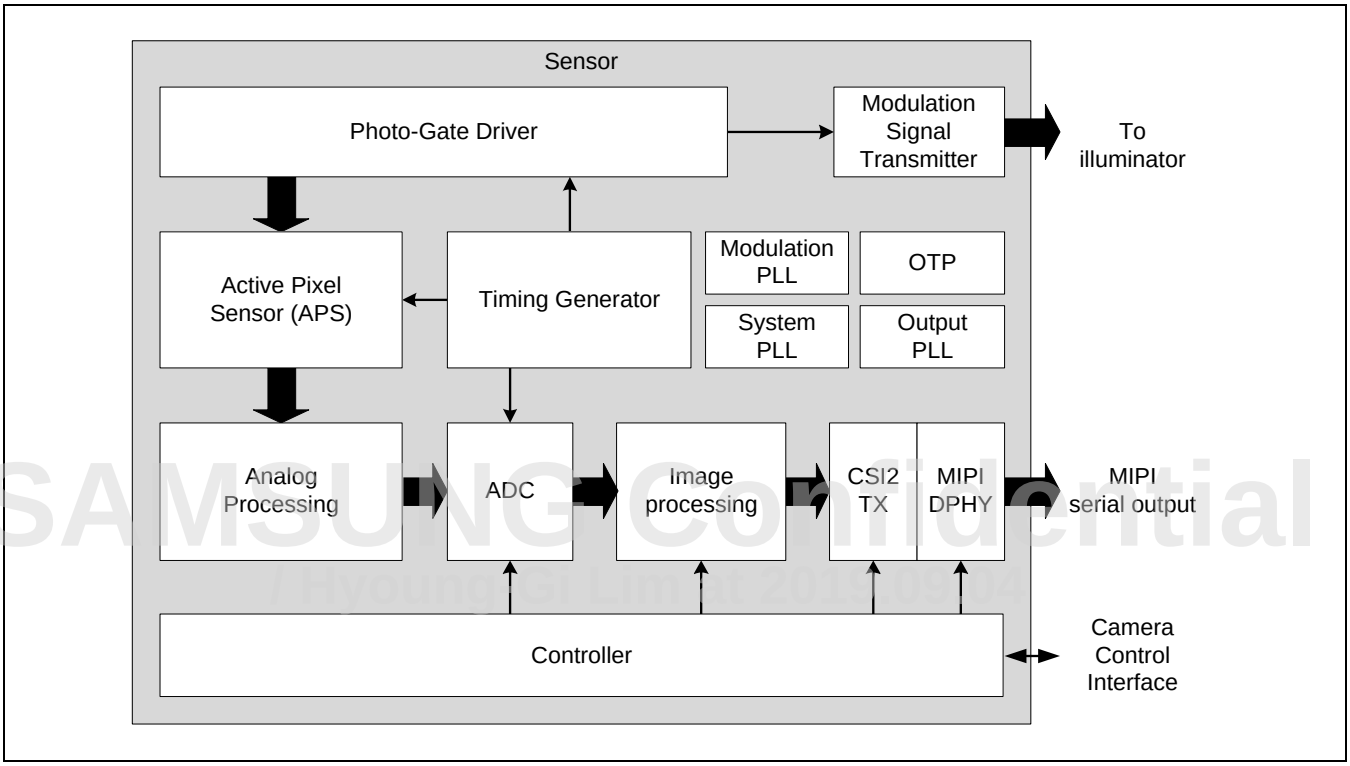


Figure 1 Functional Block Diagram

As an indirect ToF sensor, S5K33DX has a dedicated modulation PLL and a modulation signal transmitter to control the IR illuminator. The modulation clock is also used in the photo-gate driver to perform demodulation at the pixel array.

The image sensor has an on-chip ADC. A column parallel ADC scheme is used for low-power analog processing.

The analog output signal of each pixel includes some temporal random noise caused by the pixel reset action and some fixed pattern noise caused by the in-pixel amplifier offset deviation. To eliminate these noise components, a Correlated Double Sampling (CDS) circuit is used before converting to digital.

The output from the ADC is a 12-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 sensor is interfaced using a set of control and status registers that can be used to control many aspects of the sensor behavior, including frame size and exposure setting. These registers can be accessed through a CCI or SPI interface.

2 Pixel Array Information

The figure below shows top view of the sensor.

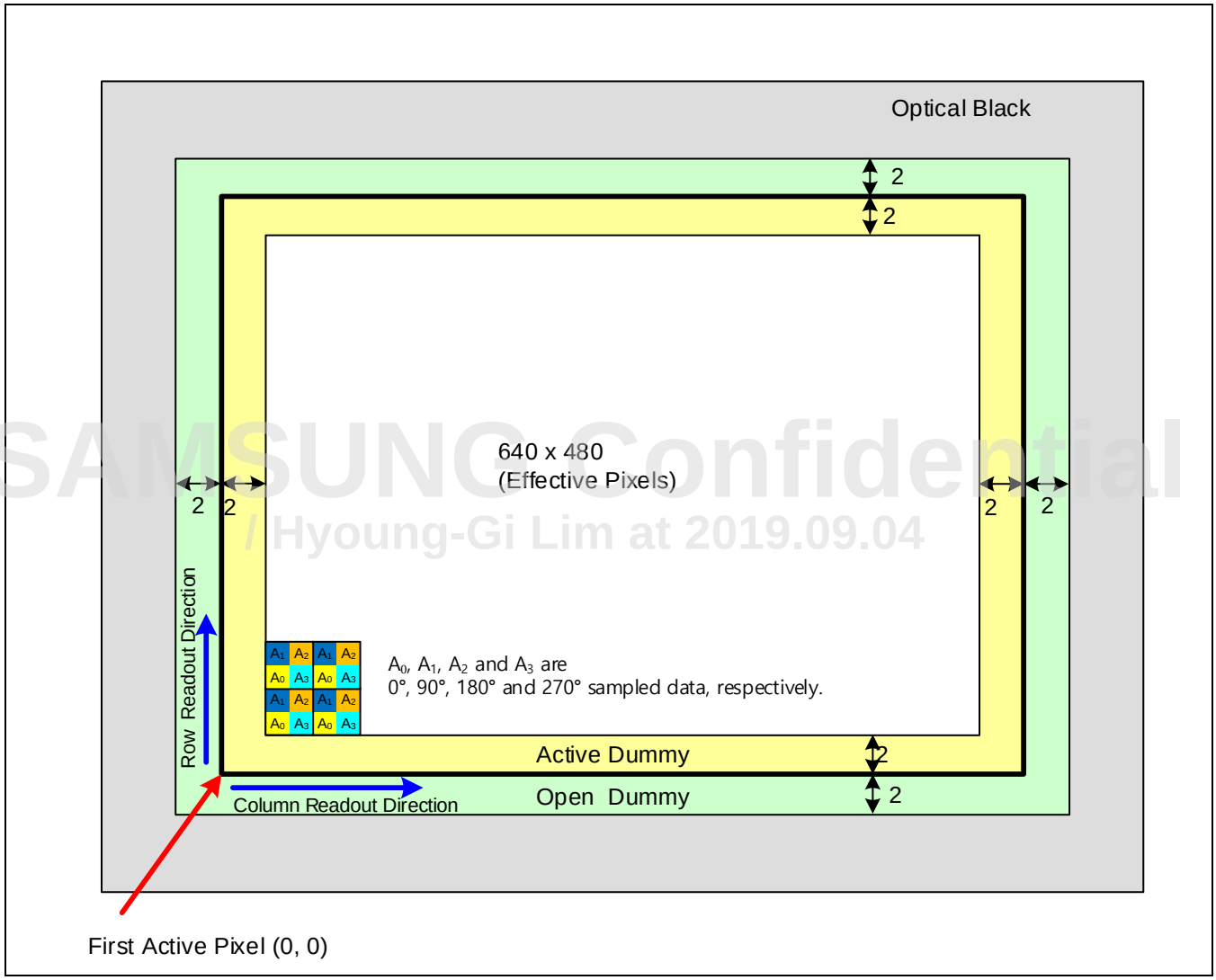


Figure 2 Pixel Array Diagram

NOTE: The displayed image will be flipped.

3 Embedded Controller

3.1 Firmware Initialization

The sensor has an embedded controller; the host must execute the following sequence for proper initialization of the embedded controller's firmware:

- **Issue Reset**. After reset (either through RSTN pin or through IIC/SPI commands), the embedded controller starts to execute its startup code. Once completed the embedded controller puts the sensor in SW Standby.

- **Load Initialization Settings**. Once the sensor enters SW Standby the host can load initialization settings. These include TnP, analog settings, front-end settings, ISP (if used), clock settings and initial mode. The first time the sensor enters SW Standby after reset (designated as Init Settings Wait state in below figure), differs from standard SW Standby mode because there are some settings that can be loaded only during this time (e.g. TnP settings).

- **Issue Stream On Command**. Once initialization settings are loaded host can issue the Stream On command. The embedded controller processes the initialization settings and configures the system accordingly. If there are any settings errors (e.g. illegal resolution, frame rate, etc.) the embedded control remains in the System Init state indefinitely. If there are no errors the embedded controller enters the Streaming state.

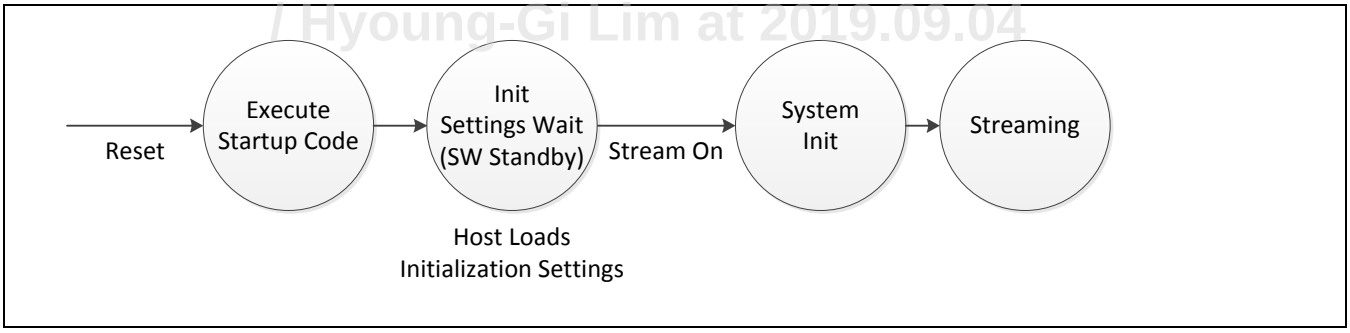


Figure 3 Embedded Controller Initialization States and Sequence

3.2 Initialization Settings File

Initialization settings, when operating S5K33DX either by Simmian or by a host (i.e. Application processor), are usually maintained as a file containing a list of commands. In order to ensure proper initialization of the sensor and easy maintenance and support the file should have the following structure:

```
// Reset
// Clocks opening
// TnP [optional…]
// Analog Settings [optional…]
// Operation Settings:
//    Clock Settings [optional…]
//    Initial Mode
// FE Settings [optional…]
// ISP Settings [optional…]
// Stream On
```

Initialization Settings File Flow-Chart:

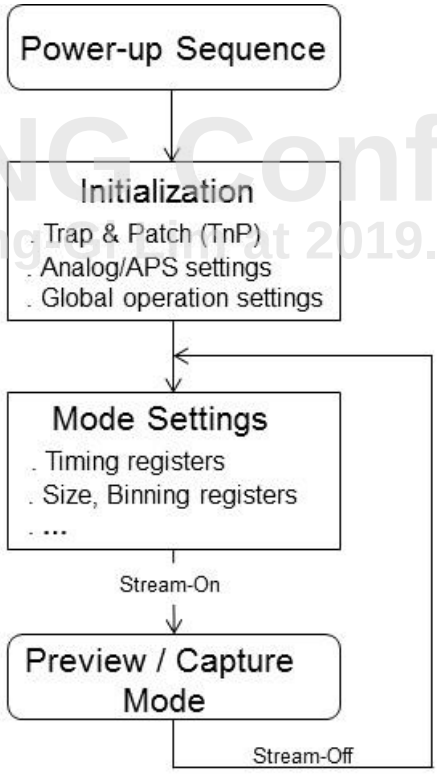


Figure 4 Initialization Settings File Flow-Chart

Caution: Take care not to place settings belonging to one section in another section; it makes settings file maintenance difficult and error prone.

3.2.1 Reset

Reset sequence:

```
WRITE 40006010 0001 // Reset
P3                          // Wait 3msec to allow startup code to run.
                            // 3msec consider the worst case of
                            // external clock set to 6 MHz.
```

This command sequence resets the sensor including the embedded controller. Once reset is complete the embedded controller executes its startup code and enters Init Settings Wait state where it waits for initialization settings to be loaded by the host.

Guide: The Reset sequence above is optional if initialization settings are loaded after power up sequence because the power up sequence itself resets the sensor (there is no need to reset the sensor twice).

3.2.2 Clocks Opening

Certain HW registers require clocks to be opened in order to be updated.

The clocks are opened at the beginning of set file, enabling these HW registers update at any further location in the set file.

```
WRITE 40006214 7970 // open all clocks
WRITE 40006218 7150 // open all clocks
```

3.2.3 TnP

In S5K33DX , the sensor's firmware code can reside only in ROM, so the embedded controller supports limited changes to firmware code by loading firmware code changes to RAM (AKA trap and patch).

3.2.4 Analog settings

This part of the set file contains analog settings of the sensor. It's configured by Samsung and must not be changed.

3.2.5 Operation Settings

3.2.5.1 Clock Settings

This section contains settings for configuration the sensor's clocks (system clock and output clock). If the definition of clocks contain errors (e.g. clocks are not within PLL capabilities), initialization will fail. Settings are defined according to guidelines provided in this document.

3.2.5.2 Initial Mode

The section defines the initial settings for input sizes, output sizes, output format, frame rate etc. Settings are defined by the user according to customer modes table provided in datasheet.

3.2.6 FE Settings

This section contains settings for digital blocks that assist with analog processing (e.g., dark level correction and analog offset calibration). Settings are provided by Samsung and must not be changed.

3.2.7 ISP Settings

These sections contain ISP settings that are done either by using calibration tools or manual tuning.

3.2.8 Stream On

Once initial settings are loaded, the stream on command is given by the host to initiate streaming:

```
WRITE 0x40000100 01
```

Once the sensor receives this command it processes the initialization settings (System Init state) and if no errors are found it changes to the Streaming state.

3.3 HW & FW Registers Host Access

Under normal operating condition the host must not access any HW registers directly (e.g. the host doesn't directly access registers of ISP, GTG, output interface, etc.); instead, the host communicates with the embedded controller (ARM7) through RAM based (FW) registers. The firmware that runs on the embedded controller in turn configures and controls the HW blocks. HW/FW register can be differentiated based on their address (see ***Table 1 Sensor's Address Space***).

Table 1 Sensor's Address Space

Address Base	Description
0x00000000	Embedded Controller's ROM
0x20000000 0x40000000 – 0x40005FFF	Embedded Controller's RAM (FW Registers)
0x40006000 – 0x4000FFFF	HW Blocks (HW Registers)

Caution: Accessing a HW register from the host might cause conflict with firmware (e.g. if firmware needs to access the same HW register) and system malfunction. Unless specifically directed by customer support engineer or this guide, do not directly access HW registers.

3.4 FW Register Usage

FW registers value can be set at different times, some must be set at initialization, some during SW standby and some can be set during streaming. The "Attr." column in register tables specified when it is possible to change registers values. It can have the following values:

Table 2 FW Register Types

Attr. Value	Description
RO	Read only access; monitoring register.
RW	Read/Write. Written value effect on the next frame.
RW/R	Read/Write. Written value effect only on exit from stand-by
RW/C	Read/Write. Changing value (may) cause configuration change (abort timing or corrupted frame)
RW/SR	Read/Write. Changing value may cause entering stand-by/software reset.
	Special rules apply to changing register value; they are described in accompanying text.

4 Control Interface (IIC/SPI)

4.1 SPI Control Interface

S5K33DX can use the SPI interface for control registers communication. SPI interface is illustrated in *Figure 5*.

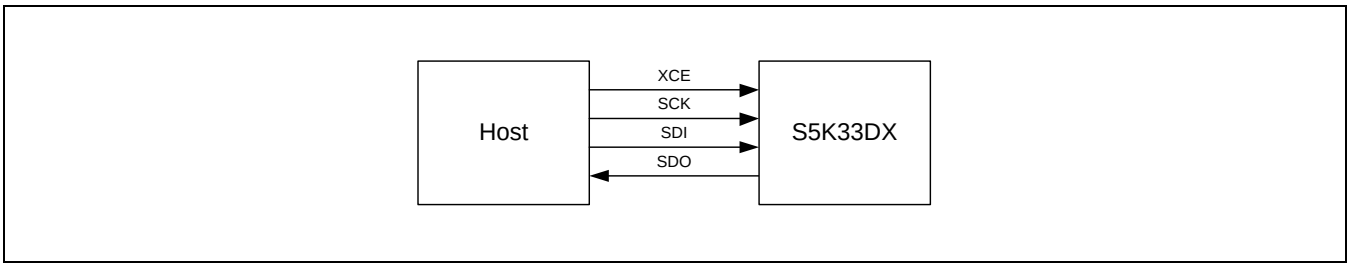


Figure 5 SPI Control Interface

The following legend table describes SPI signals in *Figure 5*.

Pin Name	Description
XCE	Serial communication enable
SCK	Serial communication clock input
SDI	Serial data input
SDO	Serial data output

If it is not necessary to read the control registers and chip ID, the SDO pad can be left unconnected.

4.1.1 SPI Timing Definitions

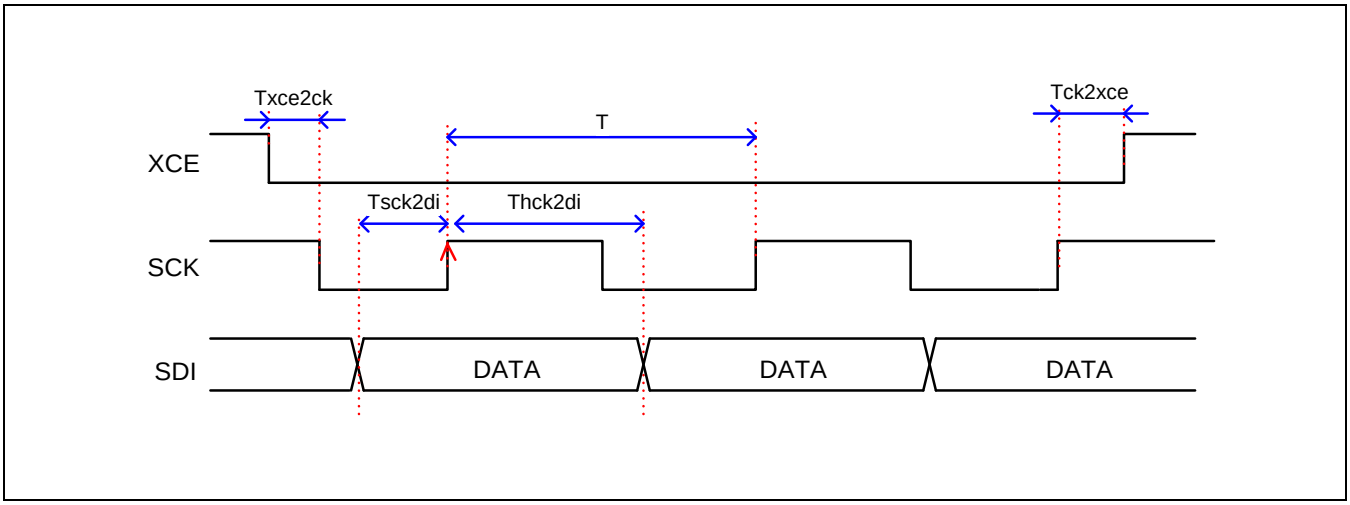


Figure 6 Timing Diagram of SPI Write Mode 11

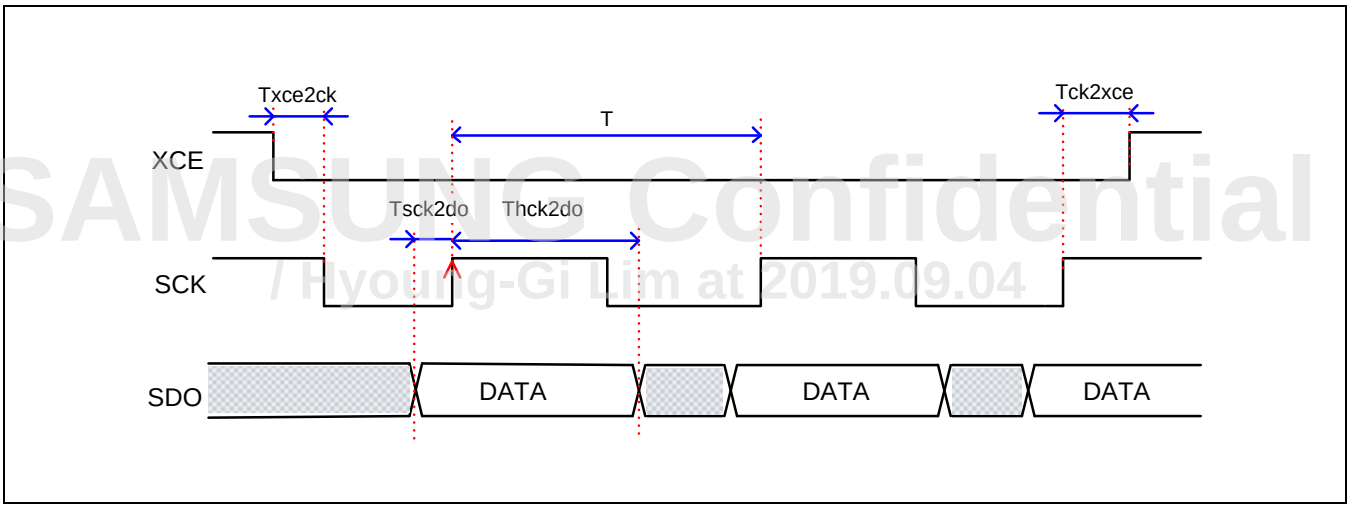


Figure 7 Timing Diagram of SPI Read Mode 11

Table 3 SPI Write/Read Timing Specification

Description	Symbol	Min.	Typ.	Max.	Unit
MCLK (external clock) frequency	Fext	6	–	64	MHz
SPI clock frequency	F (1)	–	–	Fext/3	
	F (2)	–	–	20	
SPI clock period	T	1/F	–	–	nSec
SPI clock duty cycle	Tdc	45	–	55	%
XCE negedge to SCK edge	Txce2ck	T	–	–	nSec
SCK posedge to XCE edge	Tck2xce	2T	–	–	
SCK to SDI setup time	Tsck2di	7	–	–	
SCK to SDI hold time	Thck2di	5	–	–	
SCK to SDO setup time	Tsck2do	T/2-20	–	–	
SCK to SDO hold time	Thck2do	T/2	–	–	

NOTE:

1. External clock is used.
2. PLL stable. Internal system clock can be set to maximum of 60 MHz.

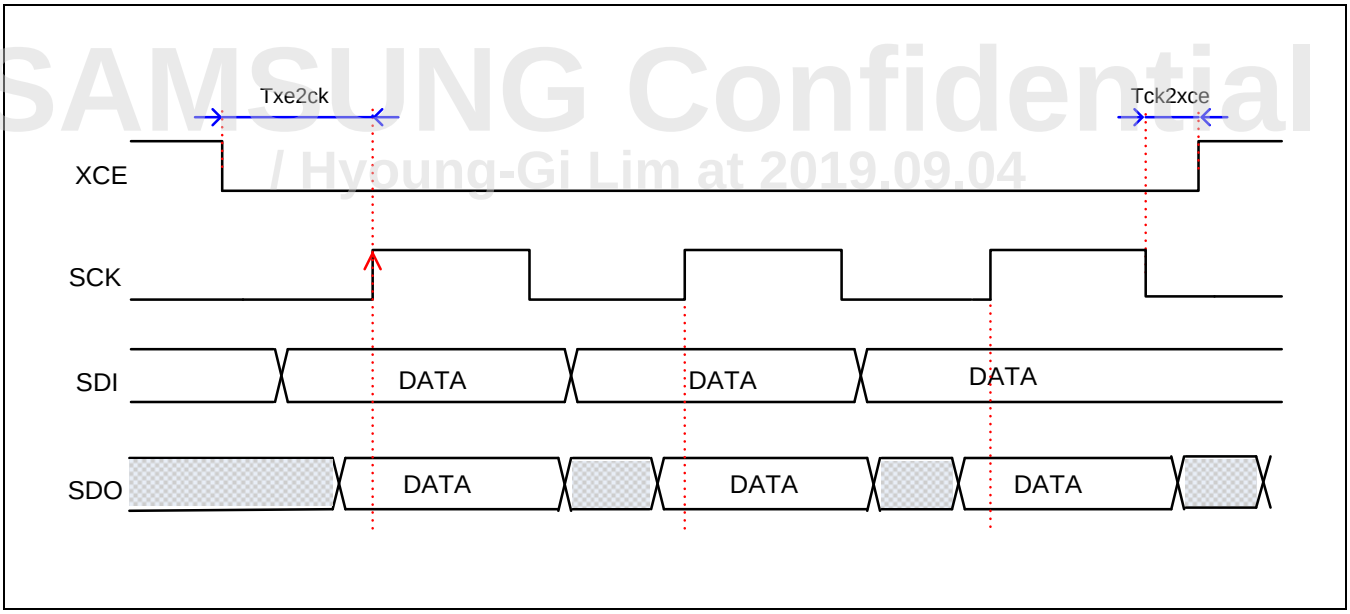


Figure 8 SPI Write/Read Mode 00

4.1.2 SPI Sequences

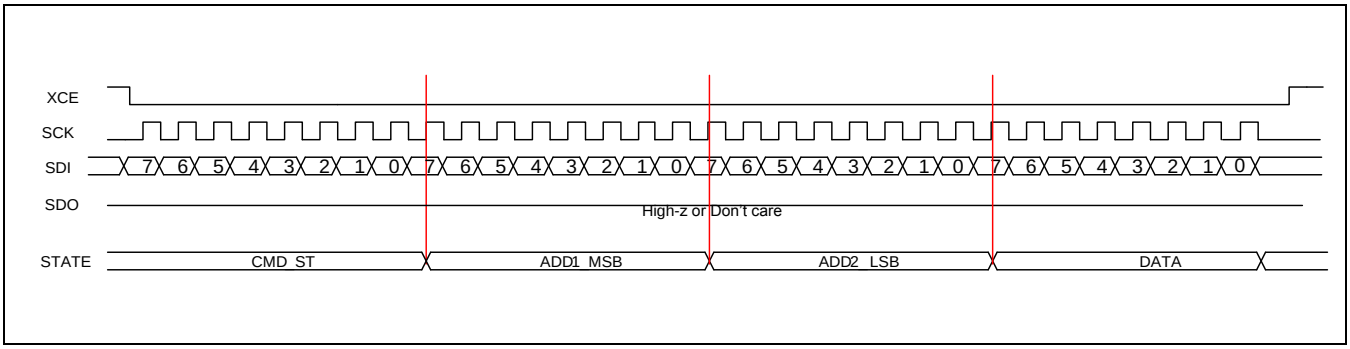


Figure 9 SPI Write Sequence (1 Cycle)

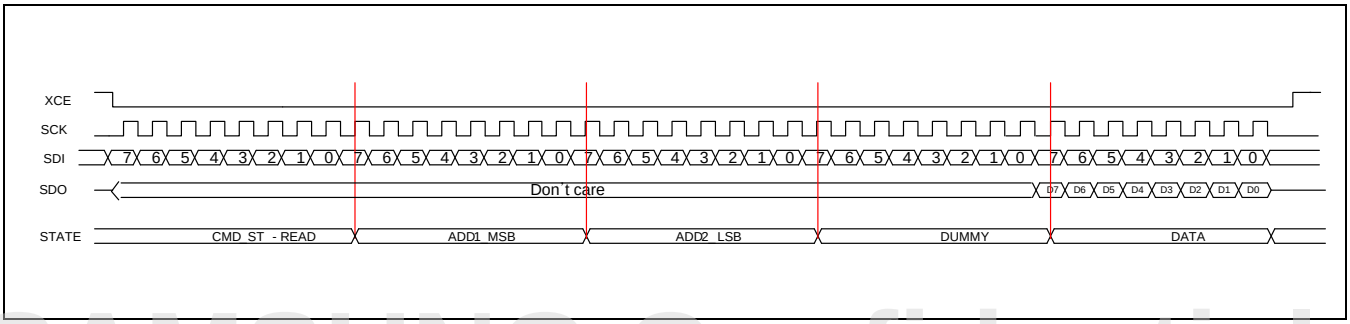


Figure 10 SPI Read

4.2 Camera Control Interface

S5K33DX supports the Camera Control Interface (CCI), which is an I2C fast-mode compatible interface for controlling the transmitter. S5K33DX always acts as a slave in the CCI bus. CCI is capable of handling several slaves in the bus, but multi-master mode is not supported. Typically, only the receiver and transmitter are connected to the CCI bus. This ensures a pure S/W implementation.

The CCI is different from the system I2C bus, but I2C-compatibility ensures that it is also possible to connect the transmitter to the system I2C bus. CCI is a subset of the I2C protocol, including the minimum combination of obligatory features for the I2C slave device specified in the I2C specification. Therefore, transmitters complying with the CCI specification can also be connected to the system I2C bus. However, it is important to ensure that the I2C masters do not try to utilize these I2C features, which are not supported in transmitters complying with the CCI specification. Each transmitter conformed to the CCI specification can have additional features implemented to support I2C.

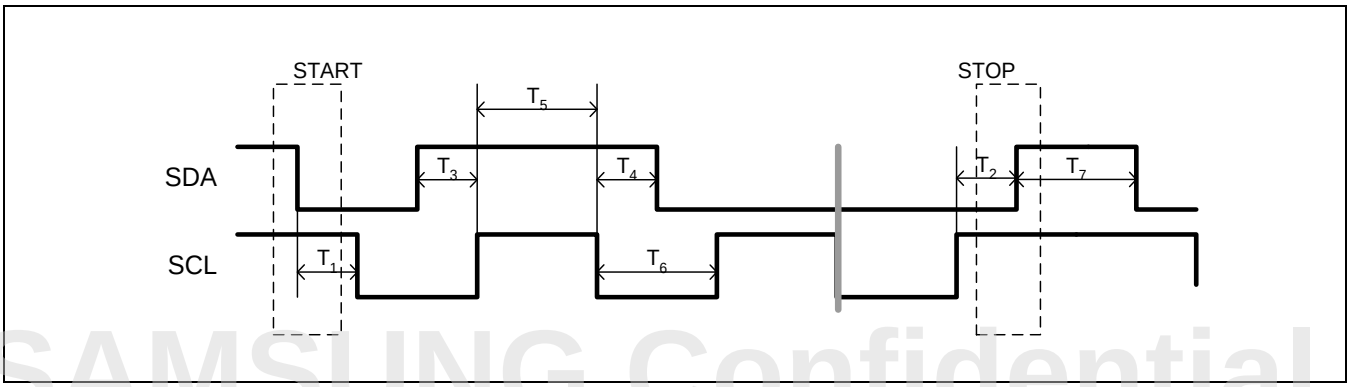


Figure 11 Timing Diagram of CCI

Table 4 I2C Standard Mode Timing Specifications

Parameter	Symbol	Min.	Max.	Unit
SCL clock frequency	–	0	100	kHz
Hold time for start condition	T_1	4.0	–	µs
Setup time for stop condition	T_2	4.0	–	
Data setup time	T_3	250	–	ns
Data hold time	T_4	0	3.45	µs
High period of the SCL clock	T_5	4.0	–	
Low period of the SCL clock	T_6	4.7	–	
Bus free time between stop and start conditions	T_7	4.7	–	
Rise time for both SDA and SCL signals	–	–	1000	ns
Fall time for both SDA and SCL signals	–	–	300	
Capacitive load for each bus line	CB	–	400	pF

Table 5 I2C Fast Mode Timing Specifications

Parameter	Symbol	Min.	Max.	Unit
SCL clock frequency	–	0	400	kHz
Hold time for start condition	T_1	0.6	–	µs
Setup time for stop condition	T_2	0.6	–	
Data setup time, external clock (MCLK) above 12.8 MHz	T_3	0.1	–	µs
Data setup time, external clock (MCLK) below 12.8 MHz		0.6	–	
Data hold time	T_4	0	0.9	µs
High period of the SCL clock	T_5	0.6	–	
Low period of the SCL clock	T_6	1.3	–	
Bus free time between stop and start conditions	T_7	1.3	–	
Rise time for both SDA and SCL signals	–	–	300	ns
Fall time for both SDA and SCL signals	–	–	300	
Capacitive load for each bus line	CB	–	400	pF

Table 6 I2C Fast Mode Plus (FM+) Timing Specifications

Parameter	Symbol	Min.	Max.	Unit
SCL clock frequency	–	0	1	MHz
Hold time for start condition	T_1	0.26	–	µs
Setup time for stop condition	T_2	0.26	–	
Data setup time, external clock (MCLK) above 24 MHz	T_3	0.05	–	µs
Data setup time, external clock (MCLK) between 12.8 and 24 MHz		0.1	–	
Data setup time, external clock (MCLK) below 12.8 MHz		0.6	–	
Data hold time	T_4	0	–	µs
High period of the SCL clock	T_5	0.26	–	
Low period of the SCL clock	T_6	0.5	–	
Bus free time between stop and start conditions	T_7	0.5	–	
Rise time for both SDA and SCL signals	–	–	120	ns
Fall time for both SDA and SCL signals	–	–	120	
Capacitive load for each bus line	CB	–	550	pF

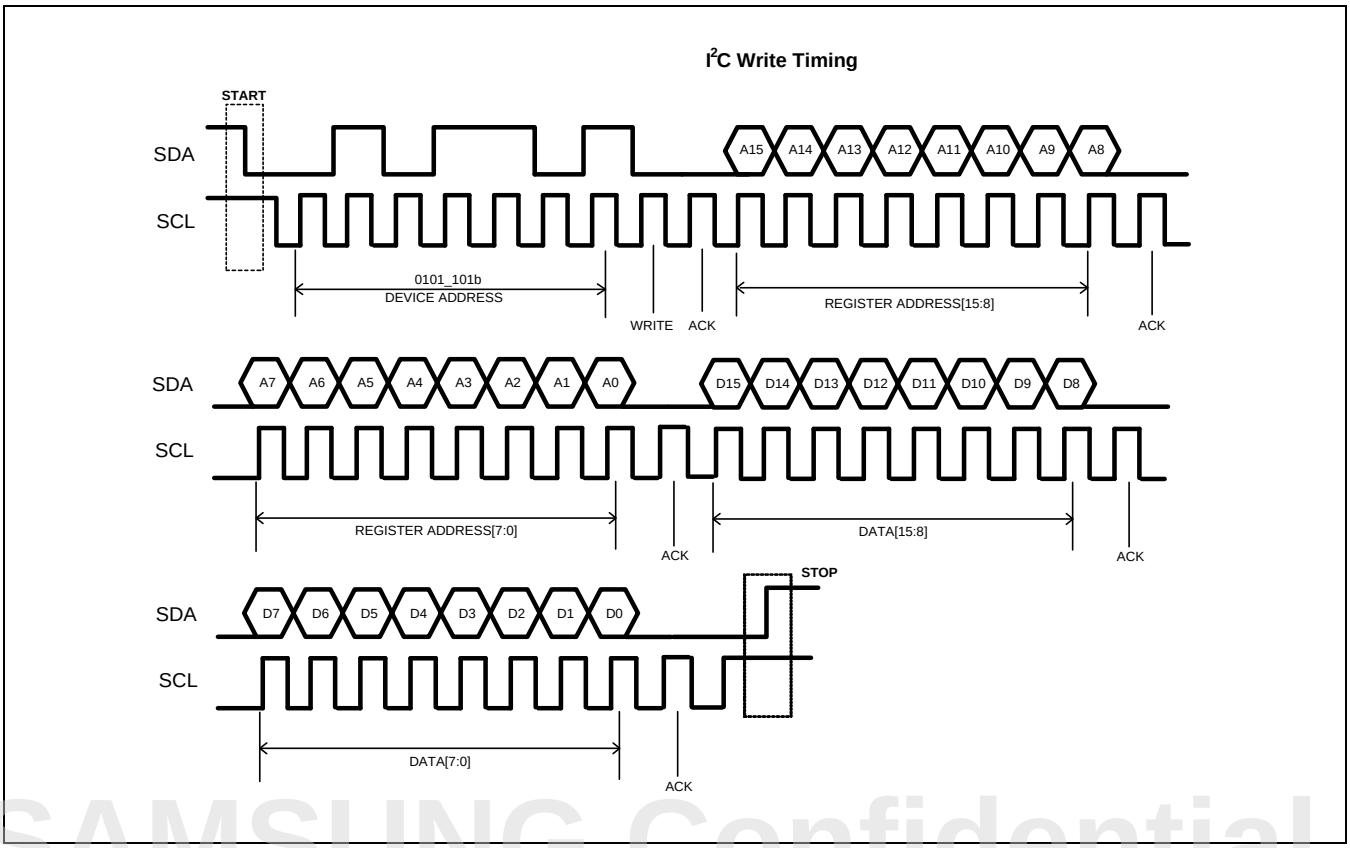


Figure 12 CCI Write Timing Example

NOTE: Pin configuration of I2C_ID changes the device address as described in the pad description.

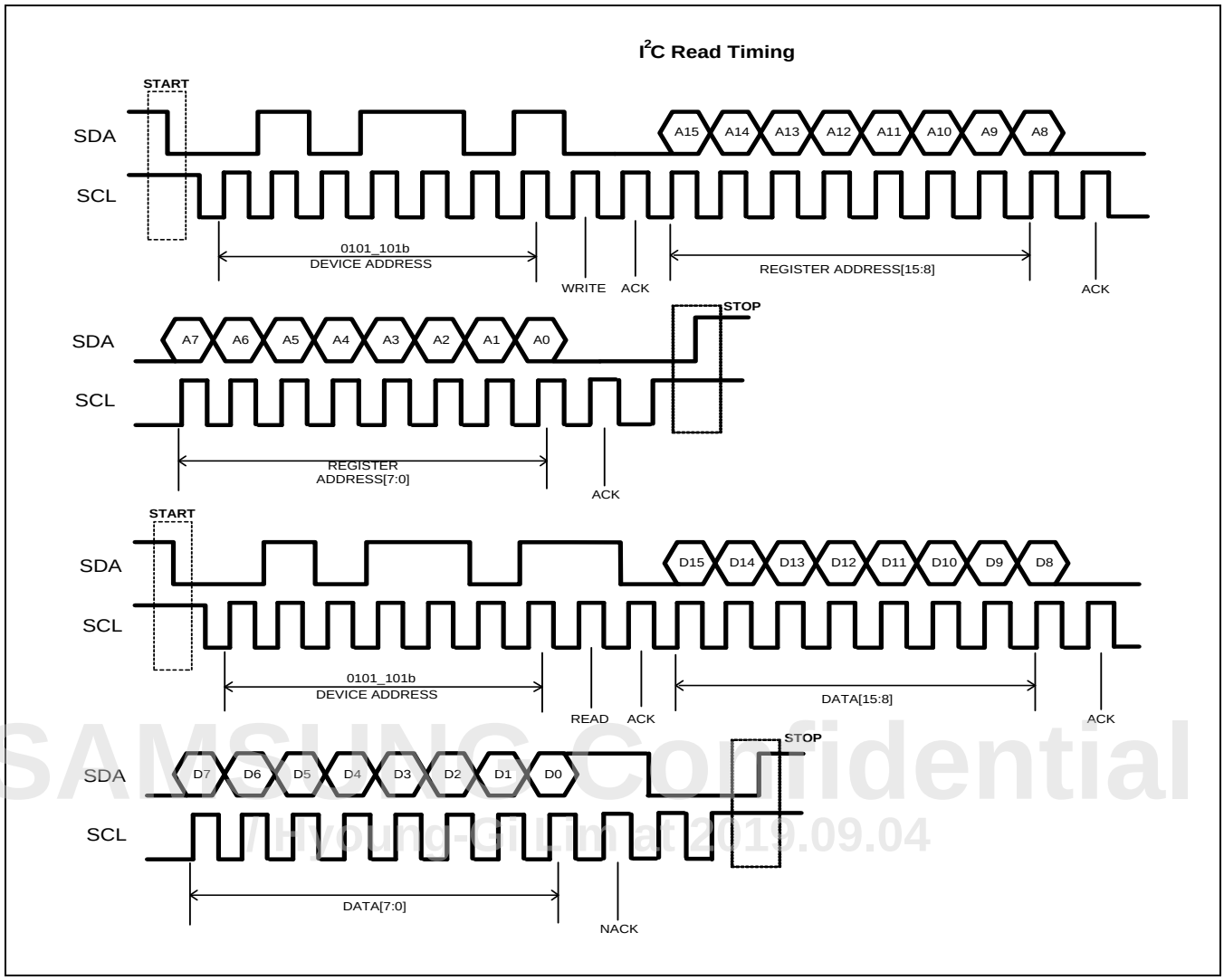


Figure 13 CCI Read Timing Example

NOTE: Pin configuration of I2C_ID changes the device address as described in the pad description.

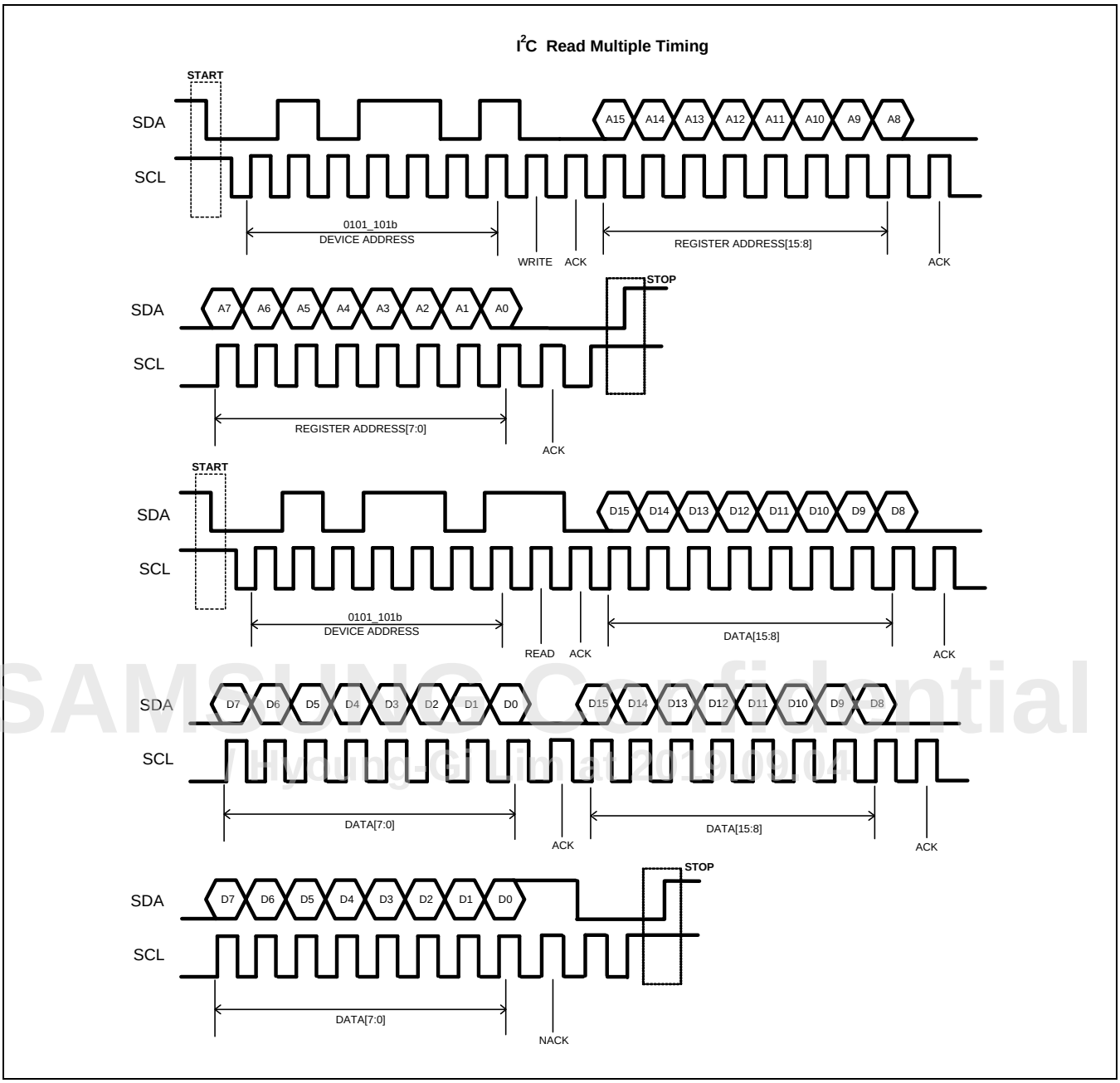


Figure 14 CCI Read Multiple Timing Example

NOTE: Pin configuration of I2C_ID changes the device address as described in the pad description.

You can configure up to two I2C slave addresses using I2C_ID pins.

Table 7 I2C ID Address (XCE Pad)

XCE	Slave Address (7-bit + Read Mode)	Slave Address (7-bit + Write Mode)	Comment
0	0010_0001b/21h	0010_0000b/20h	Address 1
1	0101_1011b/5Bh	0101_1010b/5Ah	Address 2

4.3 CIS Access Examples

For the examples below assume the following memory content:

Table 8 CIS Access Example

Address	Data	Address	Data
40000100	00	20000200	0A
40000101	01	20000201	0B
40000102	02	20000202	0C
40000103	03	20000203	0D
40000104	04	20000204	0E
40000105	05	20000205	0F
40000106	06	20000206	10
40000107	07	20000207	11
40000108	08	20000208	12
40000109	09	20000209	13

Each memory address can contain an 8-bit, 16-bit or 32-bit register. The internal bus is little-endian thus if address 0x40000100 contains an 8-bit register its value is 0x00; if it contains a 16-bit register its value 0x0100 and if it contains a 32-bit register its value is 0x03020100.

The SPI bus is bidirectional however it is used in half-duplex mode. In examples below a blue typeface indicates data read from sensor.
If the current indirect page is not mentioned it is assumed to be 0x4000.

For IIC examples:
- SR - IIC slave address for read (Slave Read)
- SW - IIC slave address for write (Slave Write)

For SPI transaction the first byte is a command byte:
- 02 – Read
- 03 – Write

4.3.1 8-bit

For this example assume 0x40000103 holds an 8-bit register.

Table 9 Example of 8-bit Direct Read/Write

Op	IIC Transaction	SPI Transaction	CIS Internal Data Bus
Direct read 8-bit	SW 01 03 SR 03	02 01 03 03	R 03 <- 40000103
Direct write 8-bit	SW 01 03 07	03 01 03 07	W 07 -> 40000103

Table 10 Example of 8-bit Indirect Read/Write

Op	IIC Transaction	SPI Transaction	CIS Internal Data Bus
Indirect read 8-bit	SW 60 2E 01 03 SW 6F 12 SR 03	02 60 2E 01 03 03 6F 12 03	R 03 <- 40000103
Indirect write 8-bit	SW 60 2A 01 03 SW 6F 12 07	02 60 2A 01 03 02 6F 12 07	W 07 -> 40000103

4.3.2 16-bit

For this example assume 0x40000104 holds a 16-bit register.

Table 11 Example of 16-bit Direct Read/Write

Op	IIC Transaction	SPI Transaction	CIS Internal Data Bus
Direct read 16-bit	SW 01 04 SR 04 05	02 01 04 04 05	R 04 <- 40000104 R 05 <- 40000105
Direct write 16-bit	SW 01 04 10 20	03 01 04 10 20	W 10 -> 40000104 W 20 -> 40000105

Table 12 Example of 16-bit Indirect Read/Write

Op	IIC Transaction	SPI Transaction	CIS Internal Data Bus
Indirect read 16-bit	SW 60 2E 01 04 SW 6F 12 SR 04 05	02 60 2E 01 04 03 6F 12 04 05	R 04 <- 40000104 R 05 <- 40000105
Indirect write 16-bit	SW 60 2A 01 04 SW 6F 12 10 20	02 60 2A 01 04 02 6F 12 10 20	W 10 -> 40000104 W 20 -> 40000105

4.3.3 32-bit

For correct 32-bit transactions the host needs to consider that the internal data bus is big-endian so values in the lower addresses are the most significant. For this example assume that address 0x40000106 holds a 32-bit register.

Table 13 Example of 32-bit Direct Read/Write

Op	IIC Transaction	SPI Transaction	CIS Internal Data Bus
Direct read 32-bit	SW 01 06 SR 06 07 SW 01 08 SR 08 09 Mapping read results to 32-bit value by host: 06070809	02 01 06 06 07 02 01 08 08 09	R 06 <- 40000106 R 07 <- 40000107 R 08 <- 40000108 R 09 <- 40000109
Direct write 32-bit	SW 01 06 12 34 SW 01 08 56 78 Host updates 32-bit value in register to 0x12345678.	03 01 06 12 34 03 01 08 56 78 Internal memory after write: 40000106 12 40000107 34 40000108 56 40000109 78	W 12 -> 40000106 W 34 -> 40000107 W 56 -> 40000108 W 78 -> 40000109

Table 14 Example of 32-bit Indirect Read/Write

Op	IIC Transaction	SPI Transaction	CIS Internal Data Bus
Indirect read 32-bit	SW 60 2E 01 06 SW 6F 12 SR 06 07 SW 60 2E 01 08 SW 6F 12 SR 08 09	02 60 2E 01 06 03 6F 12 06 07 02 60 2E 01 08 03 6F 12 08 09	R 06 <- 40000106 R 07 <- 40000107 R 08 <- 40000108 R 09 <- 40000109
Indirect write 32-bit	SW 60 2A 01 06 SW 6F 12 12 34 SW 60 2A 01 08 SW 6F 12 56 78	03 60 2A 01 06 03 6F 12 12 34 03 60 2A 01 08 03 6F 12 56 78	W 12 -> 40000106 W 34 -> 40000107 W 56 -> 40000108 W 78 -> 40000109

4.3.4 Page Change

There is no internal bus transaction for a page change operation. However all future transactions are done on the new page. For this example assume all addresses contain 16-bit registers.

Table 15 Example of Page Read/Write

Op	IIC Transaction	SPI Transaction	CIS Internal Data Bus
Indirect read page change	SW 60 2C 70 00	02 60 2C 20 00	
Indirect write page change	SW 60 28 70 00	02 60 28 20 00	
Indirect multi-page read	SW 60 2C 20 00 SW 60 2E 02 00 SW 6F 12 SR 0A 0B SW 60 2E 02 02 SW 6F 12 SR 0C 0D SW 60 2C 40 00 SW 60 2E 01 00 SW 6F 12 SR 00 01	02 60 2C 20 00 02 60 2E 02 00 03 6F 12 0A 0B 02 60 2E 02 02 03 6F 12 0C 0D 02 60 2C 40 00 02 60 2E 01 00 03 6F 12 00 01	 R 0A <- 20000200 R 0B <- 20000201 R 0C <- 20000202 R 0D <- 20000203 R 00 <- 40000100 R 01 <- 40000101
Indirect multi-page write	SW 60 28 70 00 SW 60 2A 02 00 SW 6F 12 12 34 SW 60 2A 02 02 SW 6F 12 56 78 SW 60 28 40 00 SW 60 2A 01 00 SW 6F 12 9A BC	03 60 28 70 00 03 60 2A 02 00 03 6F 12 12 34 03 60 2A 02 02 03 6F 12 56 78 03 60 28 40 00 03 60 2A 01 00 03 6F 12 9A BC	 W 12 -> 20000200 W 34 -> 20000201 W 56 -> 20000202 W 78 -> 20000203 W 9A -> 40000100 W BC -> 40000101

4.3.5 Burst

By default the IIC/SPI address is auto incremented after each internal bus transaction (auto increment is controlled by 0x40006004 register). This can be used to generate bursts of read and writes in direct mode. Indirect mode also supports bursts but this requires repeated writes to 0x6F12; thus, IIC/SPI address auto increment must be disabled. The base unit for the burst is 8-bit data as this is the internal data bus width.

The examples show a burst of 8 reads or writes. Actually they can be of any length but the page address is not incremented just the offset is incremented, so bursts cannot be done across page boundaries (see roll over example below).

When using burst the host must correctly map the burst data to register values. For these examples assume that addresses 0x40000100 and 0x40000101 contain 8-bit registers; 0x4000102 contains a 32-bit register and 0x40000106 contains a 16-bit register.

For example below in direct mode auto increment is on and in indirect mode auto increment is off.

Table 16 Example of 8-bit Direct Burst Mode

Op	IIC Transaction	SPI Transaction	CIS Internal Data Bus
Direct read burst	SW 01 00 SR 00 01 02 03 04 05 06 07 Mapping burst results to register values by host: 40000100 00 40000101 01 40000102 02030405 40000106 0607	03 01 00 00 01 02 03 04 05 06 07	R 00 <- 40000100 R 01 <- 40000101 R 02 <- 40000102 R 03 <- 40000103 R 04 <- 40000104 R 05 <- 40000105 R 06 <- 40000106 R 07 <- 40000107
Direct write burst	SW 01 00 12 34 56 78 9A BC DE F0 Host uses burst to update register as follows: 40000100 12 40000101 34	02 01 00 12 34 56 78 9A BC DE F0 Internal memory after burst: 4000 0100 12 4000 0101 34	W 12 -> 40000100 W 34 -> 40000101 W 56 -> 40000102 W 78 -> 40000103 W 9A -> 40000104 W BC -> 40000105 W DE -> 40000106 W F0 -> 40000107

Op	IIC Transaction	SPI Transaction	CIS Internal Data Bus
	40000102 56789ABC 40000106 DEF0	4000 0102 56 4000 0103 78 4000 0104 9A 4000 0105 BC 4000 0106 DE 4000 0107 F0	
Direct write burst roll-over	SW FF FE 12 34 56 78 9A BC	02 FF FE 12 34 56 78 9A BC	W 12 -> 4000FFFE W 34 -> 4000FFFF W 56 -> 40010000 W 78 -> 40010001 W 9A -> 40010002 W BC -> 40010003

Table 17 Example of Indirect Burst Mode

Op	IIC Transaction	SPI Transaction	CIS Internal Data Bus
Indirect read burst	SW 60 2E 01 00 SW 6F 12 SR 00 01 02 03 04 05 06 07 See notes for direct mode example.	02 60 2E 01 00 03 6F 12 00 01 02 03 04 05 06 07	R 00 <- 40000100 R 01 <- 40000101 R 02 <- 40000102 R 03 <- 40000103 R 04 <- 40000104 R 05 <- 40000105 R 06 <- 40000106 R 07 <- 40000107
Indirect write burst	SW 60 2A 01 00 SW 6F 12 12 34 56 78 9A BC DE F0 See notes for direct mode example.	02 60 2A 01 00 02 6F 12 12 34 56 78 9A BC DE F0	W 12 -> 40000100 W 34 -> 40000101 W 56 -> 40000102 W 78 -> 40000103 W 9A -> 40000104 W BC -> 40000105 W DE -> 40000106 W F0 -> 40000107
Indirect write burst roll-over	SW 60 2A FF FE SW 6F 12 12 34 56 78 9A BC	02 60 2A FF FE 02 6F 12 12 34 56 78 9A BC	W 12 -> 4000FFFE W 34 -> 4000FFFF W 56 -> 40000000 W 78 -> 40000001 W 9A -> 40000002 W BC -> 40000003

IIC/SPI address auto increment is controlled with 0x40006004. Value of 0 enables auto increment and value of 1 disables auto increment. When mixing direct and indirect bursts the value of auto increment must be changed.

Table 18 Example of Mixed Burst Mode

Op	IIC Transaction	SPI Transaction	CIS Internal Data Bus
Mixed read burst	SW 60 04 00 00 SW 01 00 SR 00 01 02 03 SW 60 04 00 01 SW 60 2E 01 06 SW 6F 12 SR 06 07 08 09	02 60 04 00 00 03 01 00 00 01 02 03 02 60 04 00 01 02 60 2E 01 06 03 6F 12 06 07 08 09	R 00 <- 40000100 R 01 <- 40000101 R 02 <- 40000102 R 03 <- 40000103 R 06 <- 40000106 R 07 <- 40000107 R 08 <- 40000108 R 09 <- 40000109
Mixed write burst	SW 60 04 00 00 SW 01 00 12 34 56 78 SW 60 04 00 01 SW 60 2A 01 06 SW 6F 12 9A BC DE F0	02 60 04 00 00 02 01 00 12 34 56 78 02 60 04 00 01 02 60 2A 01 06 02 6F 12 9A BC DE F0	W 12 -> 40000100 W 34 -> 40000101 W 56 -> 40000102 W 78 -> 40000103 W 9A -> 40000106 W BC -> 40000107 W DE -> 40000108 W F0 -> 40000109

4.3.5.1 32-bit Burst

This is a special case of using burst to read/write 32-bit registers. For this example assume that address 40000106h holds a 32-bit register.

Table 19 Example of 32-bit Burst Write

Op	IIC Transaction	SPI Transaction	CIS Internal Data Bus
Direct read 32-bit (burst)	SW 01 06 SR 06 07 08 09	03 01 06 06 07 08 09	R 06 <- 40000106 R 07 <- 40000107 R 08 <- 40000108 R 09 <- 40000109
Direct write 32-bit (burst)	SW 01 06 12 34 56 78	02 01 06 12 34 56 78	W 12 -> 40000106 W 34 -> 40000107 W 56 -> 40000108 W 78 -> 40000109

Table 20 Example of 32-bit Burst Read

Op	IIC Transaction	SPI Transaction	CIS Internal Data Bus
Indirect read 32-bit (burst)	SW 60 2E 01 06 SW 6F 12 SR 06 07 08 09	02 60 2E 01 06 03 6F 12 06 07 08 09	R 06 <- 40000106 R 07 <- 40000107 R 08 <- 40000108 R 09 <- 40000109
Indirect write 32-bit (burst)	SW 60 2A 01 06 SW 6F 12 12 34 56 78	02 60 2A 01 06 02 6F 12 12 34 56 78	W 12 -> 40000106 W 34 -> 40000107 W 56 -> 40000108 W 78 -> 40000109

5 Clock Settings

This chapter deals with the controlling over PLL and clock dividers in S5K33DX. In general, user should not apply clock control changes over the published modes of operation defined. In case the user requests to make adaptation in an operating mode for his application, it is strongly suggested that he contacts an FAE and describe its needs. It is not the intention of this document to give the user tools to create new modes of operation.

5.1 PLL and Clock GeneratorSetting

S5K33DX clock system uses system PLL, system clock dividers, output PLL, output clock dividers, and a modulation PLL to generate all internal clocks from a single master input clock running between 12 MHz and 60 MHz.

The maximum effective VCO frequency of output PLL for MIPI transmission is 1.6 GHz. The dedicated system PLL is used for maximal flexibility in interface frequency and for EMI avoidance. The maximum system PLL VCO frequency is 720 MHz. The modulation PLL allows flexible demodulation of the input modulation signal and for supplying sync signal for the IR illuminator. Modulation frequency can be changed for each frame to support dual-frequency mode. Clock dividers are used to generate all system clocks from a single PLL source.

The charge pump clock and the ADC clock are used for A/D conversion circuits, pixel clock is used for pixel processing and sensor control. Bit clock and output clock are set according to the required output rate.

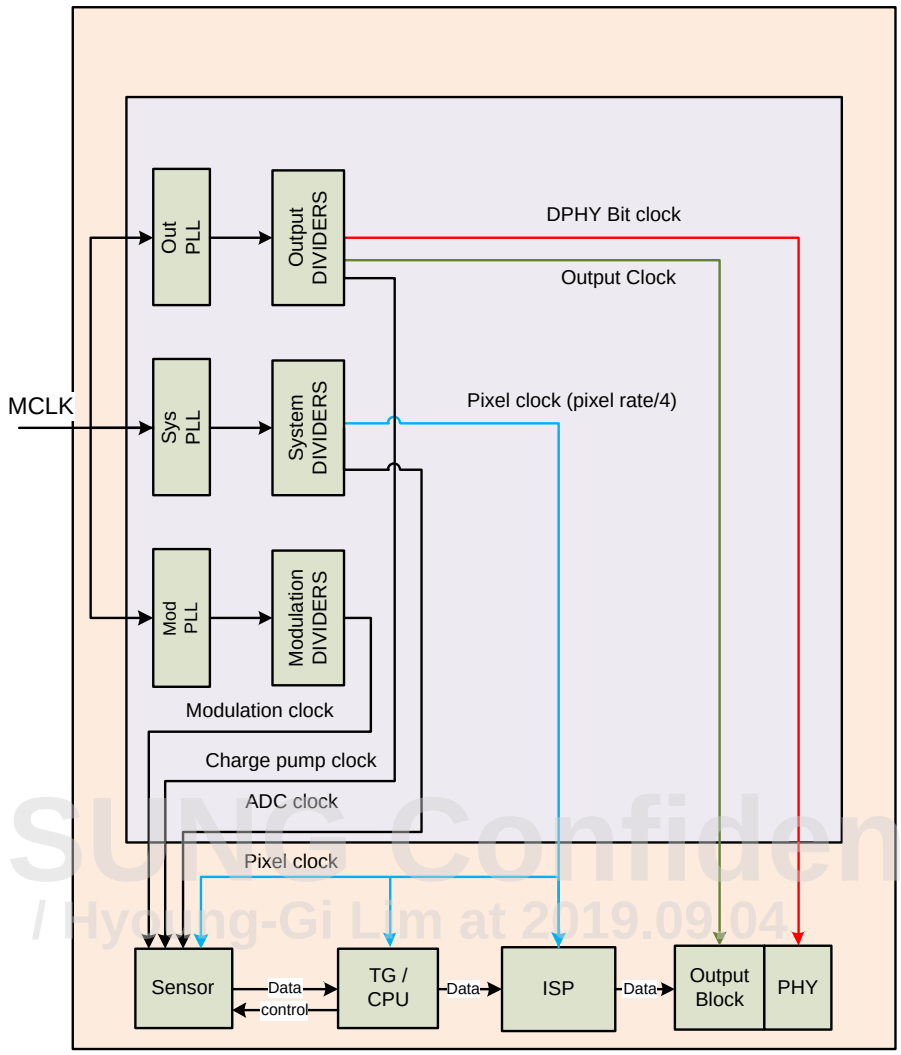


Figure 15 Clock System Block Diagram (Use of Output PLL in Addition to System PLL)

5.1.1 Clock Relationships

The host provides the external input clock (with values varying between 12 and 60 MHz) in addition to setting dividers and multipliers to get the required video timing and output pixel clocks.

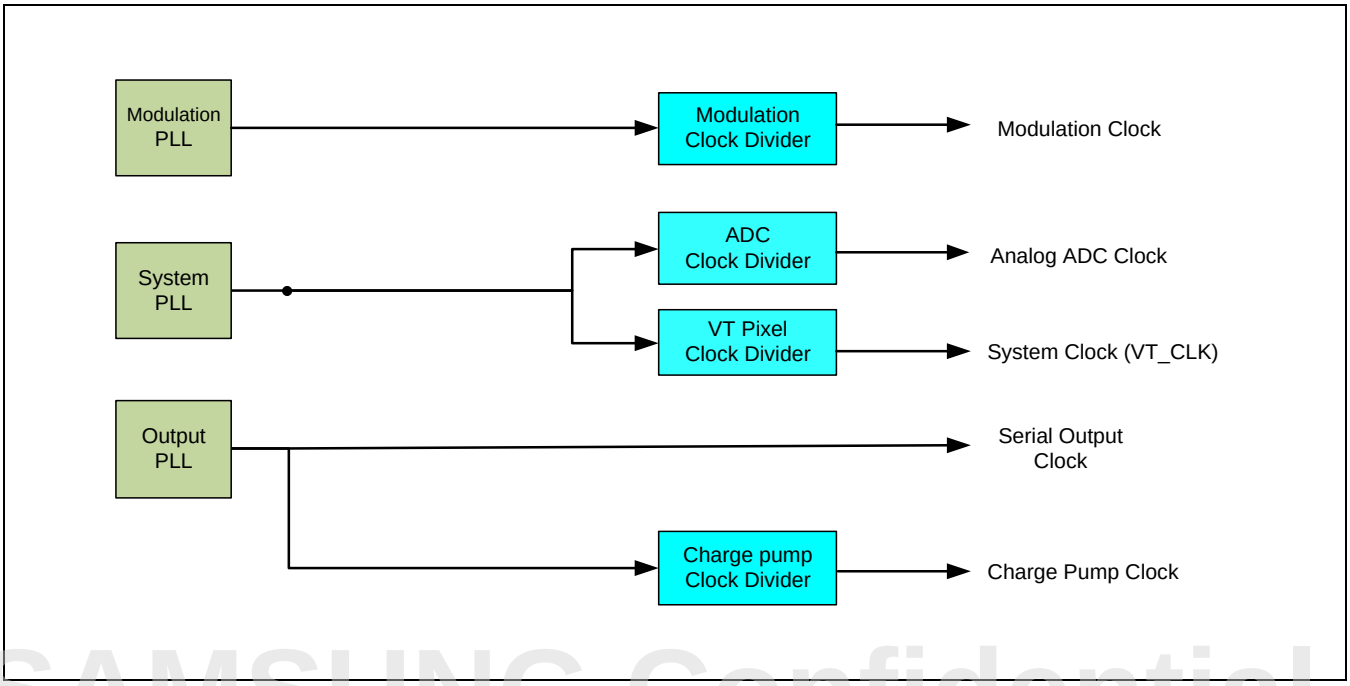


Figure 16 Clock Relationships

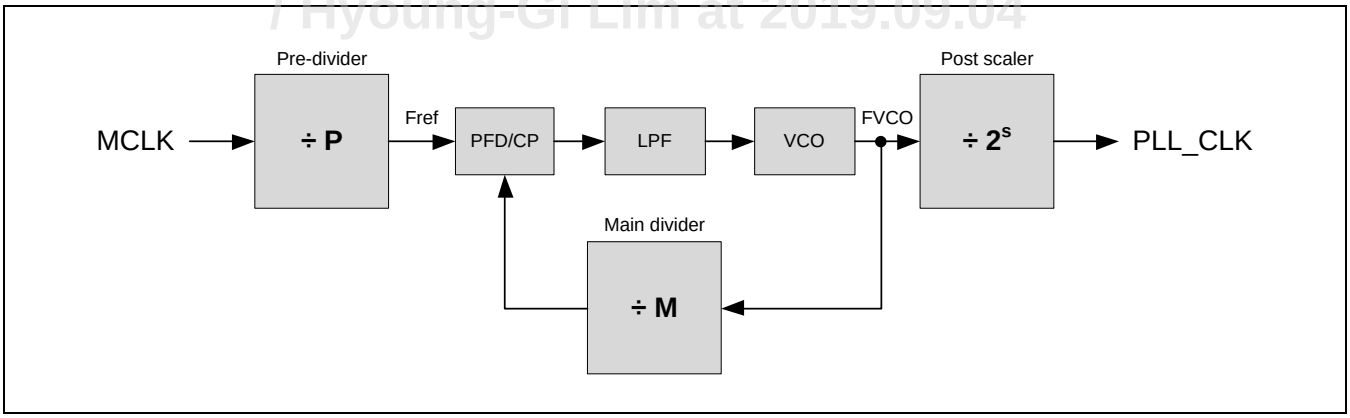


Figure 17 PLL Frequency Synthesis

Use the following formula to calculate PLL_CLK.

$$System\ PLL\_CLK = MCLK \times \frac{M}{P} \times \frac{1}{2^S}$$
$$Output\ PLL\_CLK = MCLK \times \frac{2M}{P} \times \frac{1}{2^S}$$

Post scaler (PLL_S) in S5K33DX is not controllable. Post dividers are used for lower frequency synthesis.

Table 21 PLL Component Output Frequency

Parameter	Min.	Typ.	Max.	Unit	Remarks
Input frequency range	6	–	64	MHz	EXTCLK frequency range
System PLL Reference frequency range	6	–	12	MHz	Output of system PLL pre-divider (Fref)
System PLL VCO frequency range	500	–	1000	MHz	Output of system PLL multiplier VCO oscillation range (Fvco)
System PLL output frequency range	32	–	1000	MHz	Output of system PLL post scaler. Minimum value is just for test. (6 > S ≥ 0)
Output PLL Reference frequency range	6	–	12	MHz	Output of output PLL pre-divider (Fref)
Output PLL VCO frequency range	1075	–	1600	MHz	Output of output PLL multiplier VCO oscillation range (Fvco).
Output PLL output frequency range	80	–	1600	MHz	Output of output PLL post scaler. Minimum value is just for test. (6 > S ≥ 0).
Modulation PLL IP (1st and 2nd)	6	–	12	MHz	Output of output PLL pre-divider (Fref) External Clock / PLL divider
Modulation PLL VCO (1st and 2nd)	500	–	1000	MHz	Output of output PLL multiplier VCO oscillation range (Fvco). (Fin/P)*M
Modulation PLL Fout (1st and 2nd)	500	–	1000	MHz	Output of system PLL post scaler. Minimum value is just for test. (6 > S ≥ 0)VCO/2^S

Example of main PLL calculation:

((24[external clock] / 4[pre pll divider]) × 120[pll multiplier]) / 1[pll post scalar] = 720[VCO output]

Example for calculation of pixel rate:

$$vt\_pix\_clk\_freq\_mhz = ext\_clk\_freq\_mhz * \frac{pll\_multiplier * 4}{2\text{^}pll\_s * pre\_pll\_clk\_div * vt\_sys\_clk\_div * vt\_pix\_clk\_div}$$

$$ext\_clk\_freq\_mhz = 24$$
$$pll\_multiplier = 120$$
$$pre\_pll\_clk\_div = 4$$
$$pll\_s = 0$$
$$vt\_sys\_clk\_div(dummy) = 1$$
$$vt\_pix\_clk\_div = 10$$

$$vt\_pix\_clk\_freq\_mhz = 24 * \frac{120 * 4}{2^0 * 4 * 1 * 10} = 288_{Mhz}$$

5.2 Modulation Frequency Control

S5K33DX modulation clock uses additional phase-locked loop (PLL). 33D can use 2 Modulation frequency, 100MHz and 80MHz.

$$Modualtion\ Frequency\ [Mhz] = \frac{ext\_clk\_freq\_mhz \times [\frac{M}{P} \times \frac{1}{2^S} \times \frac{1}{D}]}{4}$$

Where M is the modulation_pll_multiplier, P is the modulation_pre_pll_clk_div, S is the modulation_pll_post_scalar and D is the modualtion_pix_clk_div

Table 22 Modulation Frequency Registers

Register Name	Register Address	Default Value	Description
api_rw_clocks_modulation_pre_pll_clk_div	0x400002F0	0x3	1st MODULATION PLL frequency Pre PLL clock Divider Value (**P**)
api_rw_clocks_modulation_pll_multiplier	0x400002F2	0x64	1st MODULATION PLL frequency multiplier Value (**M**)
api_rw_clocks_modulation_pll_post_scalar	0x400002F4	0x0	1st MODULATION PLL frequency 2^S divider (**S**)
api_rw_clocks_modulation_pix_clk_div	0x400002F6	0x2	1st Modulation Pixel Clock divider (**D**)
api_rw_clocks_modulation_2nd_freq_pre_pll_clk_div	0x400002F8	0x3	2nd MODULATION PLL frequency Pre PLL clock Divider Value (**P**)
api_rw_clocks_modulation_2nd_freq_pll_multiplier	0x400002FA	0x50	2nd MODULATION PLL frequency multiplier Value (**M**)
api_rw_clocks_modulation_2nd_freq_pll_post_scalar	0x400002FC	0x0	2nd MODULATION PLL frequency 2^S divider (**S**)
api_rw_clocks_modulation_2nd_freq_pix_clk_div	0x400002FE	0x2	2nd Modulation Pixel Clock divider (**D**)

Example for control of Modulation Frequency :

MCLK= 24Mhz, P=3, M=0x64 (100 dec), S=0, D=2

$$Modualtion\ Frequency\ [Mhz] = \frac{24Mhz \times [\frac{100}{3} \times \frac{1}{2^0} \times \frac{1}{2}]}{4} = 100MHz$$

6 Frame Settings

This chapter deals with the controlling over frame timing and over the different methods of setting frame size and readout. In general, user should not apply frame control changes over the published modes of operation defined. In case the user requests to make adaptation in an operating mode for his application, it is strongly suggested that he contacts Samsung Field Application Engineer and describe its needs.

It is not the intention of this document to give the user tools to create new modes of operation.

6.1 Frame timing

The line rate and frame rate can be changed by varying the size of the timing frame. The timing frame's width and depth are controlled by the **line_length_pck** and **frame_length_lines** registers. The horizontal and vertical blanking times should meet system requirements.

Horizontal blanking time:

$$line\_length\_pck - x\_output\_size$$

Vertical blanking time:

$$(frame\_length\_lines - y\_output\_size) * line\_length\_pck$$

Frame rate calculation:

$$Frame\ Rate = \frac{vt\_pix\_clk\_freq\_mhz}{frame\_length\_lines * line\_length\_pck}$$

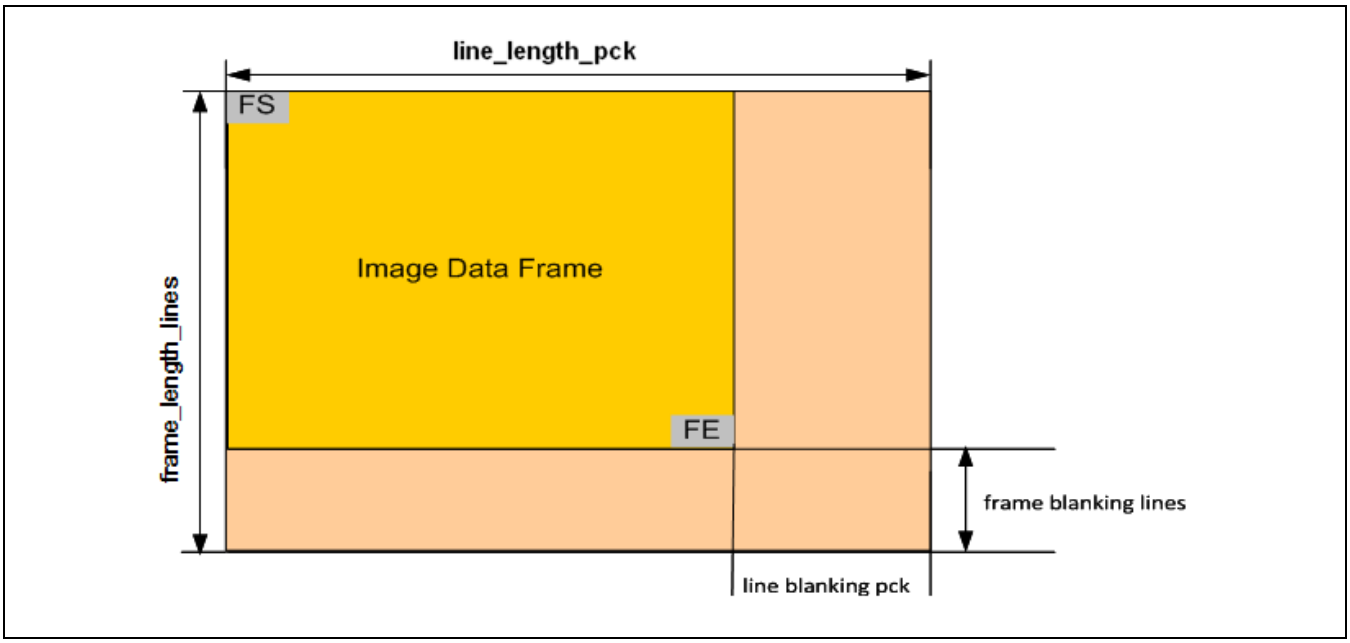


Figure 18 An Example of Timing Frame Format

Table 23 Frame Rate Control Registers

Register Name	Address	Default Value	Description
api_rw_frame_timing_frame_length_lines	0x40000340	0x0C44	1-Frame time control register. Format: 16-bit unsigned integer The minimum value of this register can be found in api_ro_frame_timing_min_frame_length_lines which considers minimum VBLANK lines required for FW processing time. This value may be lower in cases where api_rw_frame_timing_line_length_pck is large enough to compensate on shortage of frame_length_lines. In general, the FW protects the system from too low value by setting the actual value used to above the minimum
api_rw_frame_timing_line_length_pck	0x40000342	0x05F8	Recommend not to change it because all timing of control signals is changed.

Example for calculation of frame timing:

vt_pix_clk_freq_mhz=288MHz, frame_lenth_lines=3140(0x0C44), line_length_pck=1528(0x05F8)

$$Frame\ rate[fps] = \frac{288{,}000{,}000}{3140 \times 1528} = 60\ fps$$

6.1.1 Long Frame Timing

The v-blank can be changed by changing to slow capture of frames for 4tap. A set of frames required for Depth can be batched together with a long v-black between each set. By changing the numbers of frames with registers **long_frame_timing_n_frames_as_one_slow** and by changing the frame timing with register **long_frame_timing_total_slow_frame_timing_lines.**

Controllable timing parameters (T1+T2+T3 = 8.3 ms (min.) for 120 fps)

- T1 (V-blank time): minimum time exists for sensor internal operation
- T2 (integration time): depends on application and eye-safety limit
- T3 (readout time): depends on the number of vertical lines

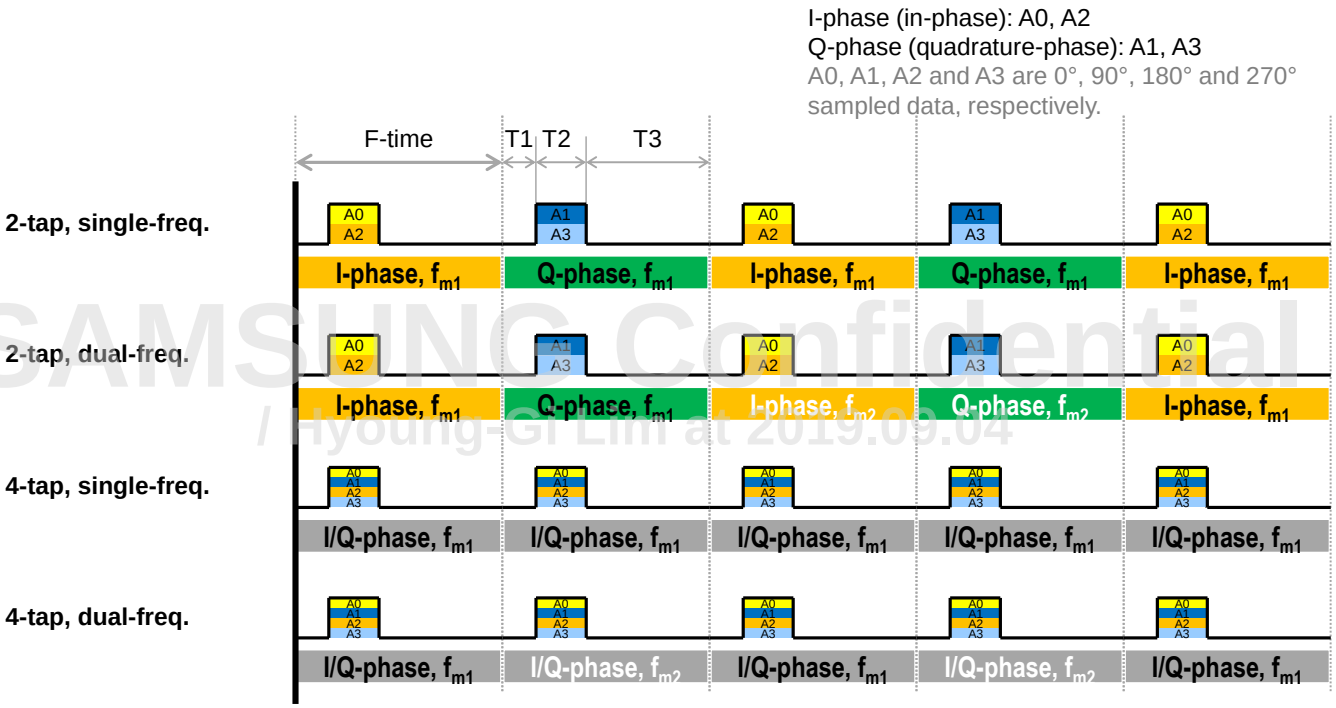


Figure 19 Frame Timing Control Diagram

In addition, S5K33DXX supports separating only the 'Slow' F/S and F/E by different VCs:

The F/S and F/E packets which arrive with a different VC will need to be filtered out

< MIPI output for a group of 4 frames >

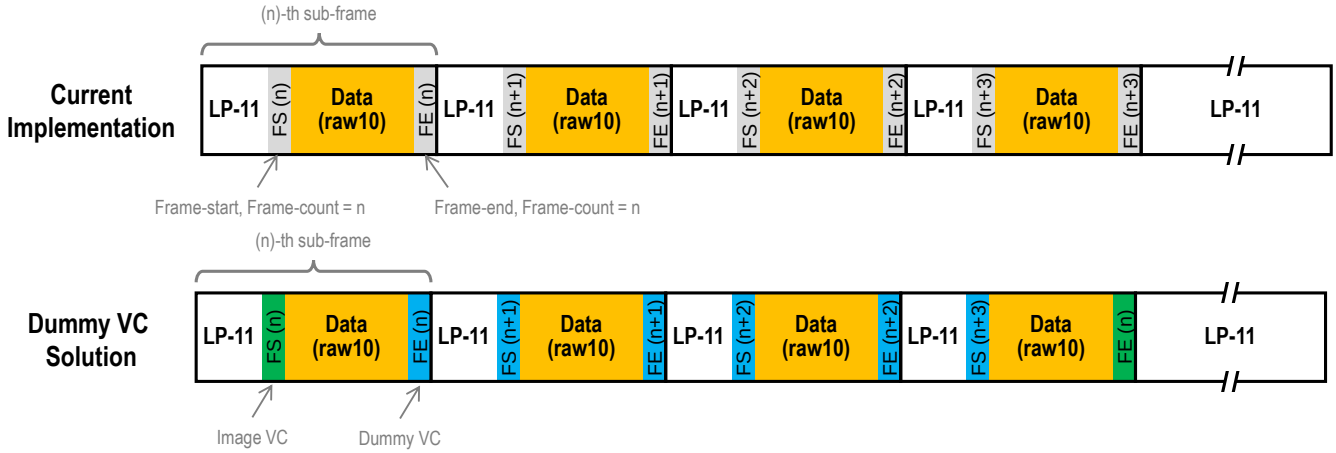


Figure 20 Slow Frame FS/FW Workaround

For very long frame timing and exposure it is possible to use a shifter in order to multiply the PCK and integration time by changing registers **long_frame_timing_line_length_pck_sh** and **long_frame_timing_coarse_integration_time_sh.**

Table 24 Long Frame Timing Registers

Register Name	Register Address	Default Value	Description
api_rw_long_frame_timing_line_length_pck_sh	0x40000703	0x0	shift-up value for line length pck.
api_rw_long_frame_timing_coarse_integration_time_sh	0x40000704	0x0	Left shifter for coarse integration time (max value is 11)
api_rw_long_frame_timing_n_frames_as_one_slow	0x4000070D	0x0	number of frames to be considered as 1 slow. 0 - disable mode, 1 - number of frames
api_rw_long_frame_timing_total_slow_frame_timing_lines	0x4000070E	0x0	number of total frame timing lines for the one
t_sys_oif_enable_frame_suppress_using_dummy_vc	0x200011A5	0x0	enable frame suppress using dummy virtual channels. will be considered only on n frame as one slow mode.
t_sys_oif_mipi_dummy_vc_on_frame_suppress	0x200011A6	0x3	mipi VC to use for FrameStart and FrameEnd workaround

6.2 Input Size

The addressed region of the horizontal pixel data output is controlled by the *x_addr_start, x_output_width* registers, and that of the vertical pixel data output is controlled by the *y_addr_start, y_addr_end* registers, respectively.

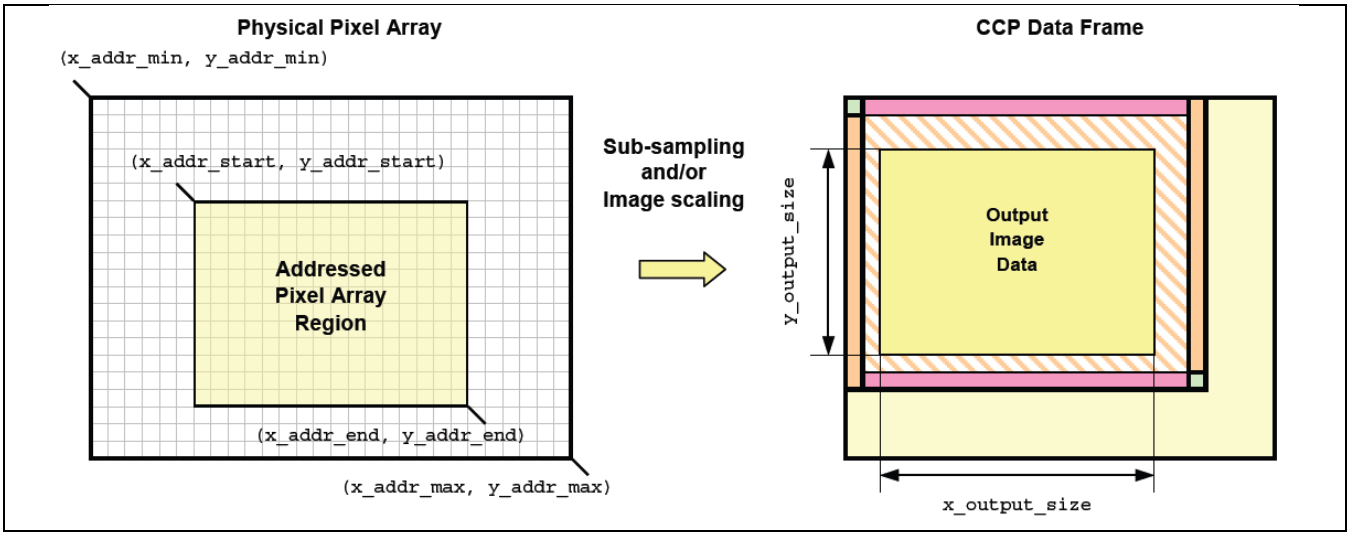


Figure 21 Controlling Input Size

6.3 Output cropping

One way of setting output size to be different than input size is cropping the input frame. This block handles cropping of the input frame according to the registers below.

In addition, it adds padding in order to complete the gap between the input frame and output frame.

Table 25 Setting Output Size Using Crop

Register Name	Address	Bit Width	Description
api_rw_image_size_x_addr_start	0x40000344	[12:0]	X-address of the top left corner of the visible pixel data (offset from full image size).
api_rw_image_size_y_addr_start	0x40000346	[12:0]	Y-address of the top left corner of the visible pixel data (offset from full image size).
api_rw_image_size_x_addr_end	0x40000348	[12:0]	X-address of the bottom right corner of the visible pixel data (offset from full image size).
api_rw_image_size_y_addr_end	0x4000034A	[12:0]	Y-address of the bottom right corner of the visible pixel data (offset from full image size).
api_rw_image_size_x_output_size	0x4000034C	[12:0]	Width of image data output from the sensor module.
api_rw_image_size_y_output_size	0x4000034E	[12:0]	Height of image data output from the sensor module.

Table 26 Setting Output Size Using Crop&Pad

Register Name	Address	Bit Width	Description
api_rw_image_size_digital_crop_x_offset	0x40000350	[15:0]	Allow to cut off left side of output image (Crop&Pad)
api_rw_image_size_digital_crop_y_offset	0x40000352	[15:0]	Allow to cut off top side of output image (Crop&Pad)

The values should correspond to the limits defined at:

Table 27 Control Registers for Frame Cropping

Register Name	Address	Bit Width	Default Value
api_ro_size_limits_min_x_output_size	0x400011B8	[15:0]	0x0100
api_ro_size_limits_min_y_output_size	0x400011BA	[15:0]	0x0090
api_ro_size_limits_max_x_output_size	0x400011BC	[15:0]	0x1230
api_ro_size_limits_max_y_output_size	0x400011BE	[15:0]	0x0DC0

The registers **api_rw_image_size_x_output_size** and **api_rw_image_size_y_output_size** are used to define

the ROI and the window of the output image.

In case digital crop is required the image width will be cropped according to **api_rw_image_size_digital_crop_x_offset** and **api_rw_image_size_digital_crop_y_offset**.

6.4 Image orientation

Mirror and flip are determined by SMIA registers.

The registers are as follows:

Table 28 Image Orientation Register

Register Name	Address	Bit Width	Default Value	Description
api_rw_general_setup_image_orientation	0x40000101	[1:0]	0x00	Image orientation i.e., horizontal mirror and vertical flip Bit 0: Hmirror – Horizontal Mirror Enable 0 = No mirror 1 = Horizontal Mirror Bit 1: Vflip – Vertical Flip Enable 0 = No flip 1 = Vertical Flip

6.5 Readout modes

S5K33DX have several different methods to read active pixels area:

1) **Normal** – The entire array is being readout. This is the normal method when number of reading pixels is equal to the size of the array defined by api_rw_frame_timing registers. It is not specified in this chapter.
2) **Analog Binning** – averaging neighbors before reading, output is smaller than the array size. Analog binning is done using a PLA circuit.
3) **Skipping** – one pixel value (each channel) is being readout from each data block.
4) **Sub sampling** – averaging without including all the pixels, some pixels are ignored.

S5K33DX also supports X downscaling and averaging in the digital domain. In the digital domain, this action is performed by the downscaler. Please refer to section 10.2 for more details.
The horizontal downscaler (HBin) is capable of producing an input: output ratio of 1:2; 1:3; 1:4.

In this section we will elaborate about the different modes.

Examples:

In the following examples we are presenting different modes to read the block presented in Figure 9.
The block size determined by ODD and EVEN increments and it will be explained later in this section.
In the examples below, the input block size is 8X8, containing 4 pixels from each channel in axis X and Y:

Input blocks of 4tap full readout:

	A	B	C	D	E	F	G	H
0	A0	A3	A0	A3	A0	A3	A0	A3
1	A1	A2	A1	A2	A1	A2	A1	A2
2	A0	A3	A0	A3	A0	A3	A0	A3
3	A1	A2	A1	A2	A1	A2	A1	A2
4	A0	A3	A0	A3	A0	A3	A0	A3
5	A1	A2	A1	A2	A1	A2	A1	A2
6	A0	A3	A0	A3	A0	A3	A0	A3
7	A1	A2	A1	A2	A1	A2	A1	A2

Figure 22 Readout Block Example

- Normal - Block size = 8X8 pixels , mode normal => output size 8X8

Normal mode, output equal to input.

- Analog Binning - Block size = 8X8 pixels , binning 4 (only Y) => output size 8X2:
  - Pixel A0 is equal to Avg(Phase-A0 [A0:A7])
  - Pixel B0 is equal to Avg(Phase-A2 [B0:B7])
  - Pixel C0 is equal to Avg(Phase-A0 [C0:C7])
  - Pixel D0 is equal to Avg(Phase-A2 [D0:D7])
  - Pixel E0 is equal to Avg(Phase-A0 [E0:E7])
  - Pixel F0 is equal to Avg(Phase-A2 [F0:F7])
  - Pixel G0 is equal to Avg(Phase-A0 [G0:G7])
  - Pixel H0 is equal to Avg(Phase-A2 [H0:H7])
  - Pixel A1 is equal to Avg(Phase-A1 [A0:A7])
  - Pixel B1 is equal to Avg(Phase-A3 [B0:B7])
  - Pixel C1 is equal to Avg(Phase-A1 [C0:C7])
  - Pixel D1 is equal to Avg(Phase-A3 [D0:D7])
  - Pixel E1 is equal to Avg(Phase-A1 [E0:E7])
  - Pixel F1 is equal to Avg(Phase-A3 [F0:F7])
  - Pixel G1 is equal to Avg(Phase-A1 [G0:G7])
  - Pixel H1 is equal to Avg(Phase-A3 [H0:H7])

	A	B	C	D	E	F	G	H
0								
1								

Notation:

Avg(Phase-A0[A0:A7]) – means average all Phase-A0 pixels in the square start at A0 (left up corner) to A7 (left bottom corner)

- Skipping - Block size = 8X8 pixels , skipping 4 (both X and Y) => output size 2X2:
  - Pixel A0 is equal to A0
  - Pixel B0 is equal to B0
  - Pixel A1 is equal to A1
  - Pixel B1 is equal to B1

	A	B
0		
1		

And continue to the next block:

  - Pixel A2 is equal to A8
  - Pixel B2 is equal to B8
  - Pixel C0 is equal to I0
  - Pixel C1 is equal to I1

Output block:

In the following matrix each cell contains the original pixel name, all the other are skipped

	A	B	C	D
0	A0	B0	I0	J0
1	A1	B1	I1	J1
2	A8	B8	I8	J8
3	A9	B9	I9	J9

- Subsampling – Block size = 8X8 pixels, binning 2 => some of the pixels are averaged and some are skipped:
  - Pixel A0 – Averaging A0,A2,C0,C2 Pixels in columns E,G and rows 4,6 are ignored
  - Pixel A1 – Averaging A1,A3,C1,C3 Pixels in columns E,G and rows 5,7 are ignored
  - Pixel B0 – Averaging B0,B2,D0,D2 Pixels in columns F,H and rows 4,6 are ignored
  - Pixel B1 – Averaging B1,B3,D1,D3 Pixels in columns F,H and rows 5,7 are ignored

  Output block:
  In the following matrix each cell contains average of 4 pixels, those not mentioned are skipped

	A	B	C	D
0	(A0+A2+C0+C2)/4	(B0+B2+D0+D2)/4	(I0+I2+K0+K2)/4	(J0+J2+L0+L2)/4
1	(A1+A3+C1+C3)/4	(B1+B3+D1+D3)/4	(I1+I3+K1+K3)/4	(J1+J3+L1+L3)/4
2	(A8+A10+C8+C10)/4	(B8+B10+D8+D10)/4	(I8+I10+K8+K10)/4	(J8+J10+L8+L10)/4
3	(A9+A11+C9+C11)/4	(B9+B11+D9+D11)/4	(I9+I11+K9+K11)/4	(J9+J11+L9+L11)/4

Note:
- Average of adjacent is done between pixels from the same channel / color only.
- In the example we assumed same binning type on axis x and y but if needed each axis configuration can be controlled separately

6.5.1 Binning and skipping modes

Analog binning is done using a PLA circuit.
Binning type values are 1, 2, 3, 4 and 6 ONLY for Y axis. For X axis only binning value 1 is supported.
By programming the binning enable register, the sensor can be configured to read out pixel data that has been averaged with adjacent pixels.

The following formula defines the sub_sampling_factor

$$sub\_sampling\_factor = \frac{even\_inc + odd\_inc}{2}$$

Examples:
- even inc = 1 , odd_inc = 5 => block size = 6, it consists of 3 pixels from each color
- even inc = 1 , odd_inc = 3 => block size = 4, it consists of 2 pixels from each color

If binning type is equal to sub_sampling_factor, full binning is taking place which means all sub_sampling_factor pixels are averaged.
If binning type is smaller than sub_sampling_factor, some pixels are skipped without reading.

If binning type value is invalid, no binning will be performed. Output will be sub-sampled with the defined even and odd increment parameters.
If the binning is performed in the analog domain, the timing of processing a single row will be decreased down to a minimum required by the analog for the conversion and readout processes.

Table 29 Binning Control Registers

Register Name	Address	Default Value	Description
api_rw_binning_mode	0x40000900	0x00	0 = Disabled analog binning 1 = Enabled analog binning 2 = Enabled analog summation, only in 2PD projects If the mode is set to none, then skipping or normal mode is applied.
api_rw_binning_type	0x40000901	0x11	[3:0] = Row (vertical) binning factor [7:4] = Column (horizontal) binning factor When any of the binning factors are set to 0 or 1, it means that no binning is applied in this axis.
api_rw_binning_weighting	0x40000902	0x00	0 = Averaged weighting 1 = Summed weighting 2 = Bayer weighting 3 = Module-specific weighting

The host input request is initially verified by the FW:
- The sum of the odd and even increment x, y is twice the x, y (respectively) binning type.
- Validity of binning type – ensure it is selected from the valid types only (1, 2, 3, 4 and 6).

If any of above is not valid, no binning will be performed. Output will be sub-sampled with the defined even and

odd increment parameters.

The binning type for the vertical axis defines with which row the current selected row being averaged, e.g., if the row factor is set to 2, then row index 0 will be averaged with row index 2 and row index 3 will be averaged with row index 5 for an even and odd y increment of 3, 3.

6.5.2 Sub-Sampled Mode

By programming the x and y odd and even increment registers (*x_even_inc, x_odd_inc, y_even_inc, y_odd_inc*), the sensor can be configured to read out sub-sampled pixel data.

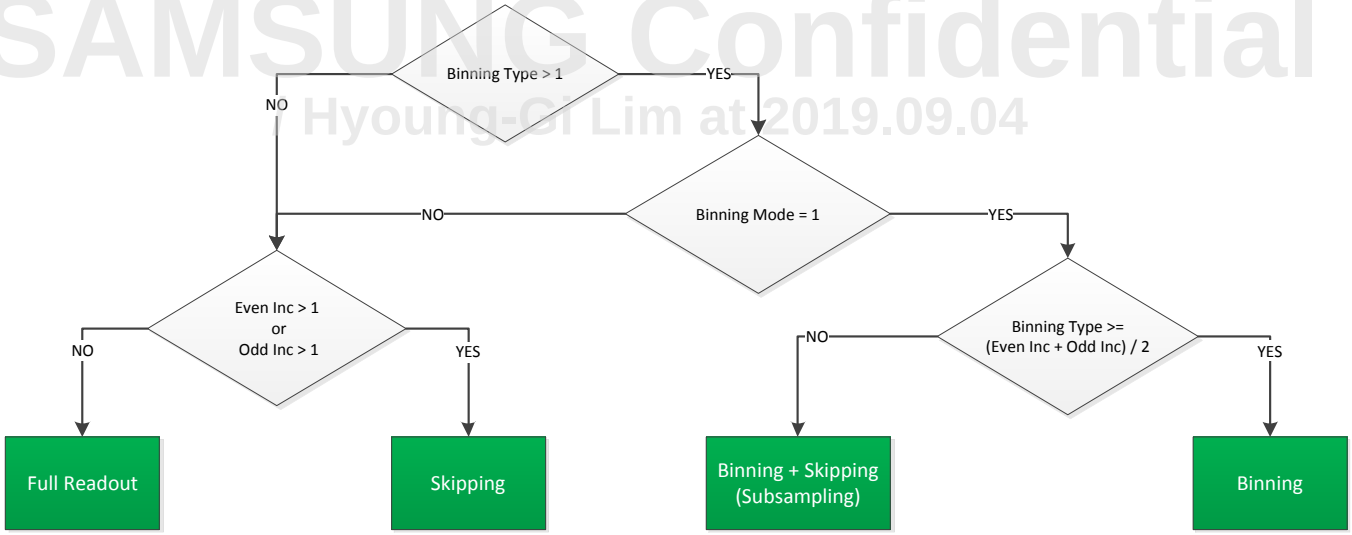
The system will follow the odd and even increments to progress.
If current pixel X/Y address is even, progress by x_even_inc/y_even_inc respectively.
If current pixel X/Y address is odd, progress by x_odd_inc/y_odd_inc respectively.

NOTE: The odd and even increment values must have odd values. If the host specifies non-odd values, then the firmware overrides these values accordingly.

6.5.3 Configuration flow

- **Binning type can be controlled separately for each axis (high nibble for x, low nibble for y).**
- **Binning mode is common for both axis X and Y**

Each one of the following charts can describe the possible configuration flow:



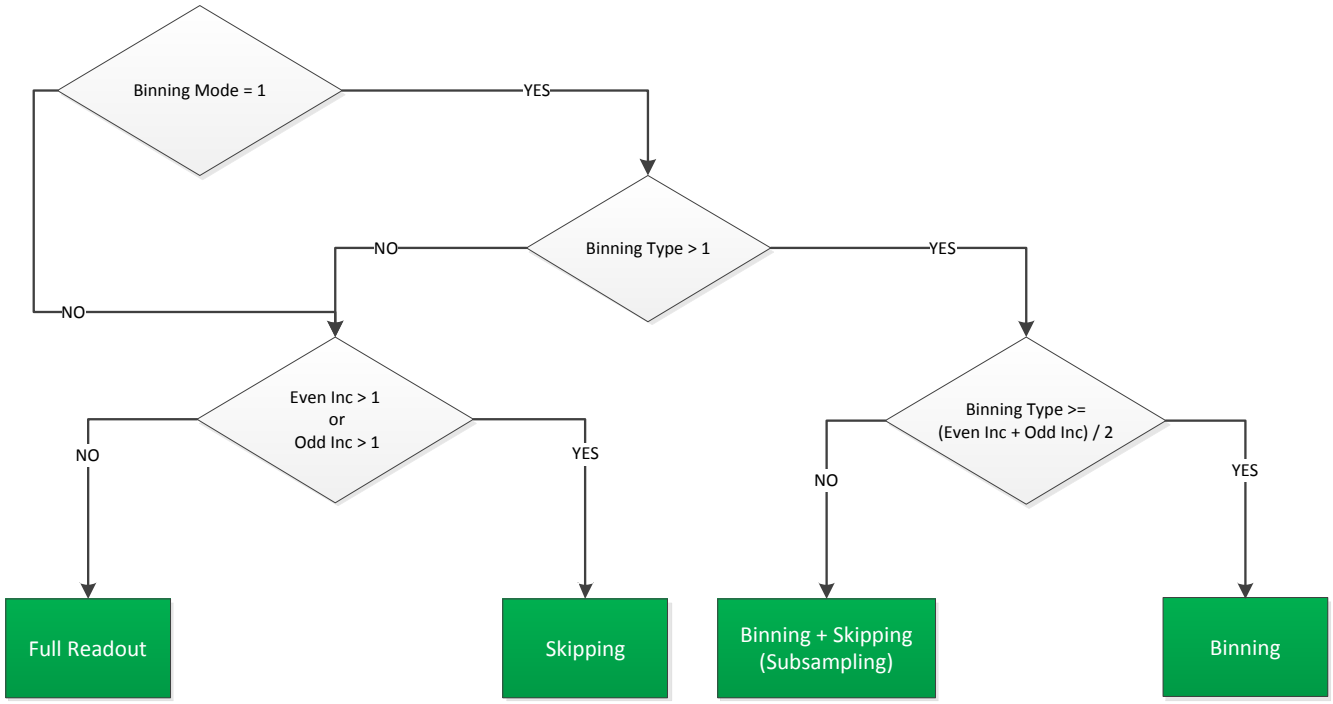


Figure 23 Binning Configuration Flows

6.5.4 Sub-Sampling Effect on Sensor's Timing

When defining a certain sub-sampling for the x axis, the timing of line readout will be shortened down to a limit derived from the sensor constraint of processing the line.

The sub sampling in the y axis may be of the pattern: even: 4n + 1, where n is an integer in the range 0-7(i.e. the even value may be up to 29); odd: any positive odd value.

Table 30 X- and Y-Address Increment Registers (Read/Write)

Register Name	Address	Bit Width	Description
api_rw_sub_sample_x_even_inc	0x40000380	[4:0]	Increment for even pixels – 0, 2, 4, etc. Format: 16-bit unsigned integer
api_rw_sub_sample_x_odd_inc	0x40000382	[4:0]	Increment for odd pixels – 1, 3, 5, etc. Format: 16-bit unsigned integer
api_rw_sub_sample_y_even_inc	0x40000384	[4:0]	Increment for even pixels – 0, 2, 4, etc. Format: 16-bit unsigned integer
api_rw_sub_sample_y_odd_inc	0x40000386	[4:0]	Increment for odd pixels – 1, 3, 5, etc. Format: 16-bit unsigned integer

The figure below illustrates how the above register values can be programmed to sub-sample in x and y by a factor of 2.

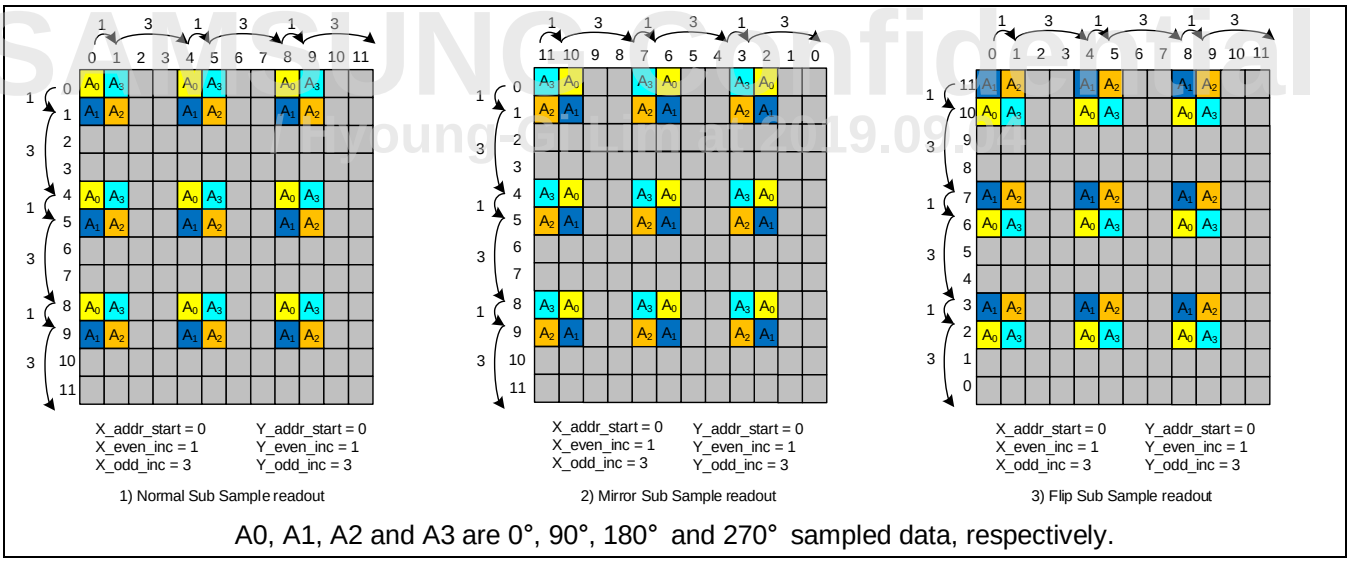


Figure 24 Example of Sub-Sampled Readout

The table below lists the registers that define the minimum and maximum limits for the odd and even increment values.

Table 31 X- and Y-Address Increment Parameter Limits (Read Only)

Register Name	Address	Default Value	Bit Width	Description
api_ro_sub_sample_cap_min_even_inc	0x400011C0	0x0001	[15:0]	Minimum Increment for even pixels
api_ro_sub_sample_cap_max_even_inc	0x400011C2	0x000F	[15:0]	Maximum increment for even pixels
api_ro_sub_sample_cap_min_odd_inc	0x400011C4	0x0001	[15:0]	Minimum Increment for odd pixels
api_ro_sub_sample_cap_max_odd_inc	0x400011C6	0x001F	[15:0]	Maximum Increment for odd pixels

The rules for sub-sampled readout are:

- The application of the sub-sample parameters must be re-timed internally to the start of frame boundary.
- Sub-sampled readout is disabled by setting both the odd and even increment values to 1.
- If the pixel being read out is even (0, 2, 4, etc.), then the address is incremented by the even increment value.
- If the pixel being read out is odd (1, 3, 5, etc.), then the address is incremented by the odd increment value.
- The equation for the sub-sampling factor expressed in terms of the even and odd increments is shown below:

$$sub\_sampling\_factor = \frac{even\_inc + odd\_inc}{2}$$

- The camera modules pixel readout order register value must track as the start address, the end address, the odd address increment and the even address increment values change.
- The host system when calculating the start address, the end address, the odd address increment and the even address increment values must ensure that the CSI data transmitted per line length (CCP2) or packet (CSI-2) is according to size rules.
- The host must ensure that x_addr_end and y_addr_end coincide with a selected pixel (rather than a skipped one), otherwise the output will be implementation-defined.
- It is the responsibility of the host to update the output image size registers according to the sub-sampling factors.

6.5.5 Limitations

- Visible column RO address: 4 * N. N=0, 1, 2, 3….
- Visible column RO size: 4 * Sub-sampling factor * N. N=0, 1, 2, 3…..
- Visible row RO address: 4 * M. M=0, 1, 2, 3…
- frame_timing_x_addr_end must be odd (frame_timing_x_addr_start + width – odd_increment)
- frame_timing_y_addr_end must be odd (frame_timing_y_addr_start + width – odd_increment)

6.6 Output Format

There are several output formats, which are pre-defined. These formats are selected the data format register:

Table 32 Output Format Register

Register Name	Address	Default Value	Description
api_rw_output_data_format	0x40000112	0x0C0C	Data format [15:8] = Max. sensors ADC resolution 0x08 = Top 8 bits of pixel data (RAW8) 0x0A = Top 10 bits of pixel data (RAW10) [7:0] = Output data format 0x08 = Top 8 bits of pixel data (RAW8) 0x0A = Top 10 bits of pixel data (RAW10)

7 Integration Time and Gain Control

7.1 Integration Time Control

Integration time is set in respect to system timing settings and timing frame size settings. System clock and line length (in pck units) are considered.

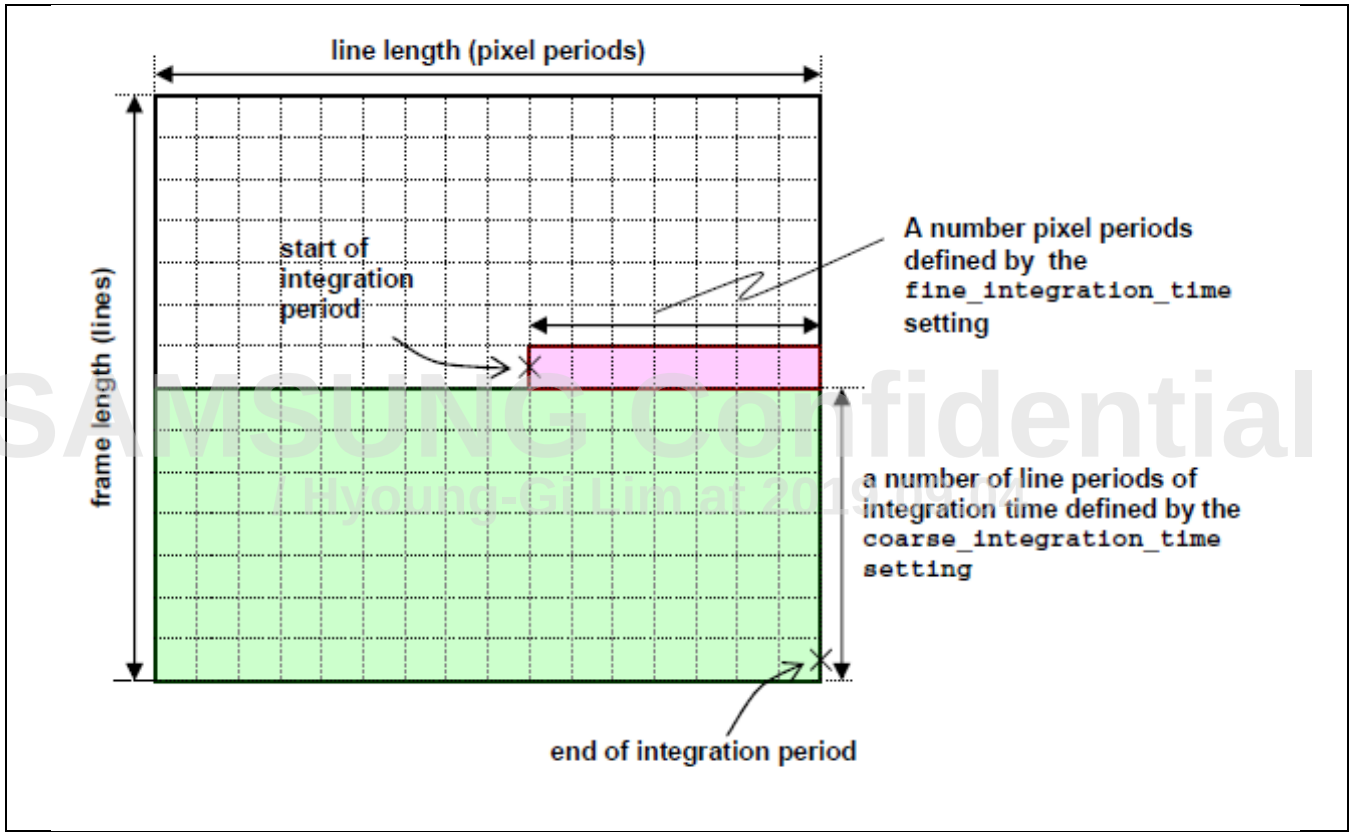


Figure 25 Integration Time

Values for coarse and fine parameters of integration time are limited to the values shown in the following figure.

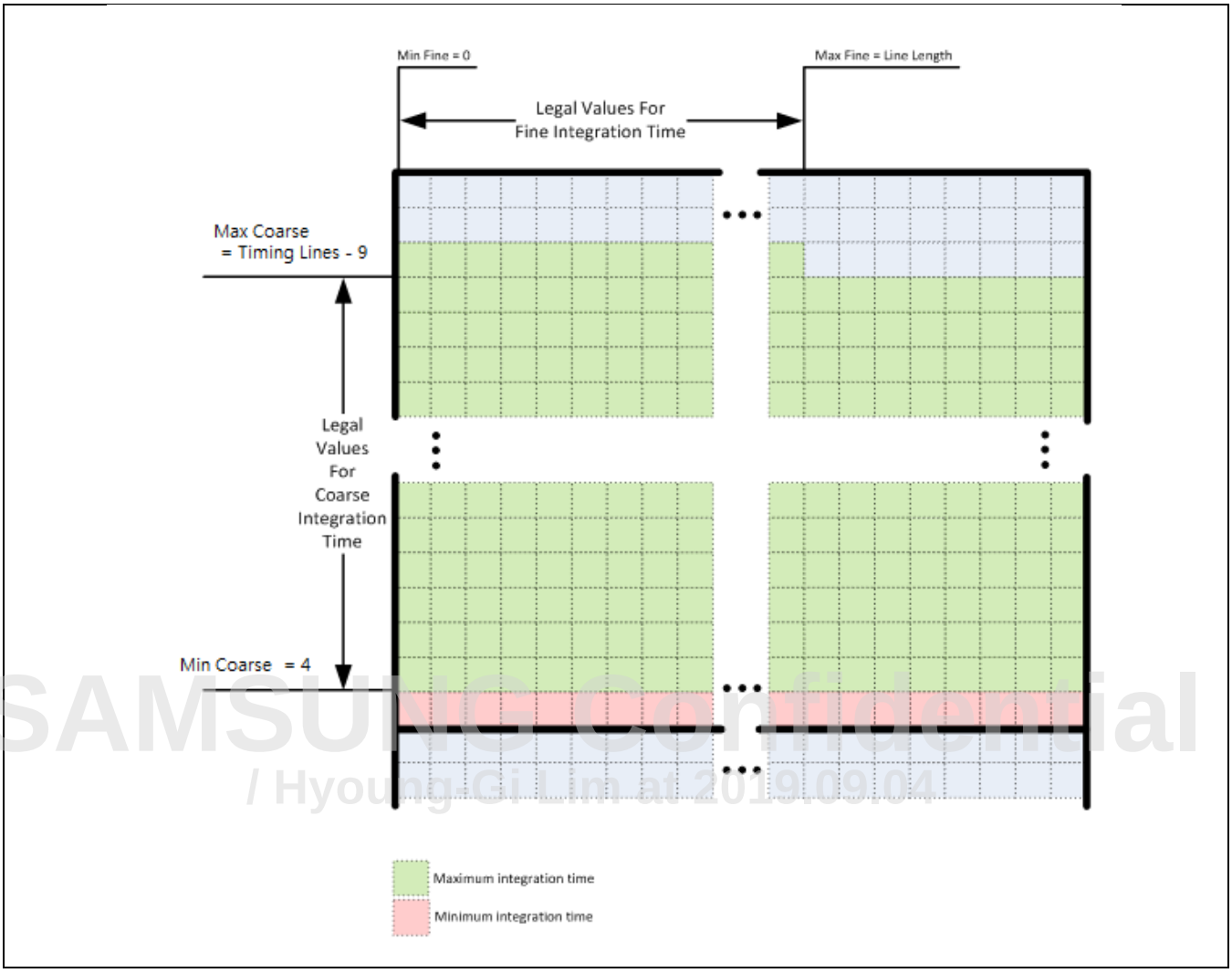


Figure 26 Coarse and Fine Integration Time Limitation

Note:

Since the exposure in 33D is global and must not occur during readout there is no actual min/max limit, therefore they are set to 0.

The pixel integration (exposure) time is controlled by the shutter operation. It is determined by the line Step Integration Time Control Register (coarse_integration_time). The total integration time of the sensor module can be calculated using the following formula:

$$total\_integration\_time = (coarse\_integration\_time * pixel\_periods\_per\_line) * pix\_clk\_period$$

Where pix_clk_period can be calculated as follows:

$$pix\_clk\_period = \frac{1}{(vt\_pix\_clk\_freq\_mhz * 10\text{^}6)}$$

For calculation of vt_pix_clk_freq_mhz, please refer to Ch. 5.1.

Table 33 Frame Rate Control Registers

Register Name	Address	Default Value	Description
api_rw_op_cond_extclk_frequency_mhz	0x40000136	0x0	Nominal external clock frequency (8.8 fixed point number)
api_rw_clocks_vt_pll_multiplier	0x40000306	0x0078	PLL multiplier value
api_rw_clocks_vt_pre_pll_clk_div	0x40000304	0x0004	Pre-PLL clock divider value
api_rw_clocks_vt_pix_clk_div	0x40000300	0x000A	Video timing pixel clock divider
api_rw_frame_timing_line_length_pck	0x40000342	0x0	Line length (H-time) in pixel clocks (VT clock domain) It can be increased by long_exposure registers. 0 = FW select shortest time possible

For example:

$$vt\_pix\_clk\_freq\_mhz = 1,280Mhz$$

$$pix\_clk\_period = 1/(vt\_pix\_clk\_freq\_mhz * 10^6) = 1/1,280,000,000\ sec$$

$$api\_rw\_frame\_timing\_line\_length\_pck = 5600$$

If the user is interested in a total integration time of 10 ms, coarse_integration_time can be calculated by:

$$coarse\_integration\_time = 10 * 10^{-3} * 1,280,000,000/5600 = 2285$$

Table 34 Integration Time Control Registers

Register Name	Address	Default Value	Description
api_rw_integration_time_coarse_integration_time	0x40000202	0x0095	Coarse integration time in h-times (of short exposure in WDR mode)Format: 16-bit unsigned integer (5.3056us/ code)

Example for calculation of integration time :

coarse_integration_time=149(0x0095)

Integration time = 5.3056us x 149 = 790.53 us

7.2 Digital GAIN control

This block handles digital gains and periodic gain mismatch.
It has two sets of register interfaces:

- SMIA gain registers interface, which is coarse and supports fractional gain of 1/256 scale.
- When digital gain is applied, the LSB(s) resulting data shall be padded with zeros.

Table 35 Digital Gain Control Registers

Register Name	Address	Default Value	Description
api_rw_digital_gain_gains_phase_gain	0x4000020E	0x0100	Digital gain control for all phases in case single digital gain is used, or Phase-A0 in case digital gain per phase channel is used
api_rw_digital_gain_gains_reserved1	0x40000210	0x0100	Phase-A2 in case digital gain per phase channel is used
api_rw_digital_gain_gains_reserved2	0x40000212	0x0100	Phase-A1 in case digital gain per phase channel is used
api_rw_digital_gain_gains_reserved3	0x40000214	0x0100	Phase-A3 in case digital gain per phase channel is used
api_rw_digital_gain_digital_gain_mode	0x4000020D	0x0000	0 = use a single digital gain for all phases, taken from api_rw_digital_gain_gains_phase_ gain. 1 = gain for each phase channel
api_ro_digital_gain_min	0x40001084	0x0100	The minimum valid limit of the digital gain control parameters
api_ro_digital_gain_max	0x40001086	0x8000	The maximum valid limit of the digital gain control parameters
api_ro_digital_gain_step_size	0x40001088	0x0100	Defines the resolution of the digital gain control parameters

8 Stream Preserving and Interrupting Register Updates

8.1 Stream Preserving Register Updates

8.1.1 Exposure/Gain Configuration Changes

Exposure and gain changes are very common in a standard sensor operation. Often, changes in exposure that are requested at frame "n", cannot be applied for frame "n + 1", and they will be applied only on frame "n + 2". The decision whether changes can be applied for next frame or not depends on current exposure time parameters, on new exposure time parameters, and on the timing of the change request within the frame.

Therefore, there are two different modes of apply method for changes in exposure and gain control registers changes:

- Normal mode- Changes will always be applied in frame "n + 2". This mode is intended for hosts who would like a stable system that updates constantly with a two frame delay.
- Immediate mode - When possible, changes will be applied in frame "n + 1". This mode is intended for hosts who would like to give their users a system that updates as fast as possible.

In both modes – exposure and gain both change together, at the same frame.

Table 36 Immediate Mode

Register Name	Address	Default Value	Description
api_rw_general_setup_immediate_exposure_enable	0x4000010F	0x00	Start exposure of next frame immediately (if exposure has not started yet) 0 = Disable immediate update for integration time 1 = Enable immediate update for integration time

The next flowchart describes the process described above:

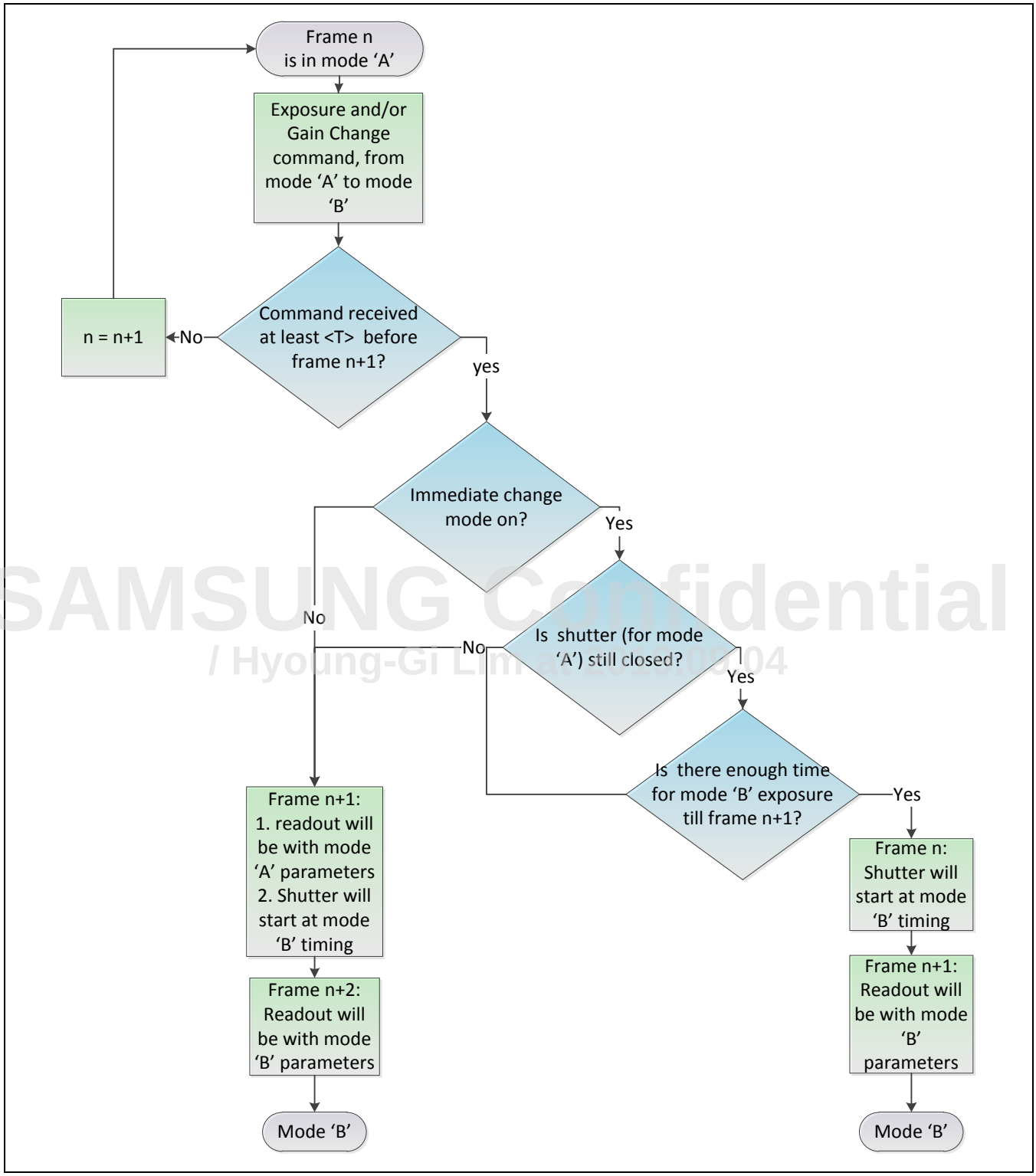


Figure 27 Exposure/Gain Configuration Changes

8.1.1.1 Exposure/Gain Normal Configuration Changes

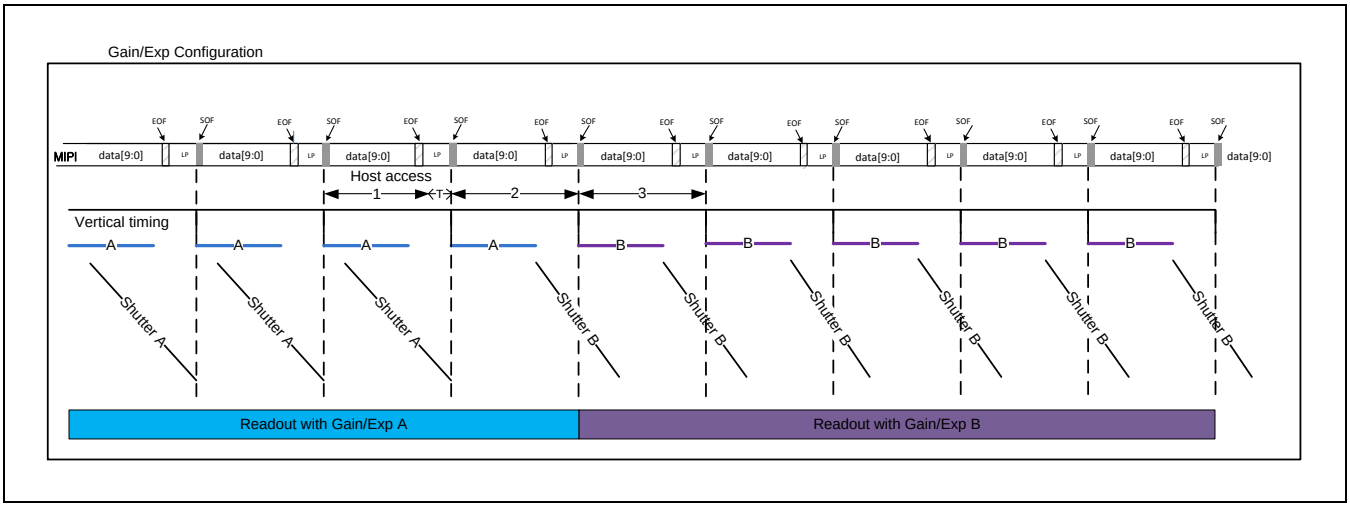


Figure 28 Exposure/Gain Normal Configuration Change

Table 37 Exposure/Gain Normal Configuration Change

Phase	Title	Description
1	Host Access	Host access in order to change Gain/Exp mode from mode A to mode B. Exposure & Gain configuration change is applicable within the timing frame boundary till <T> time before next frame readout.
2	Transition frame	At the following frame after host access, readout will still have mode A parameters, but shutter will start mode B operation.
3	New configuration frame	At the second following frame readout will have mode B parameters.

NOTE: <T> time is the time relatively to the Frame start (SOF)

8.1.1.2 Exposure/Gain Immediate Configuration Changes

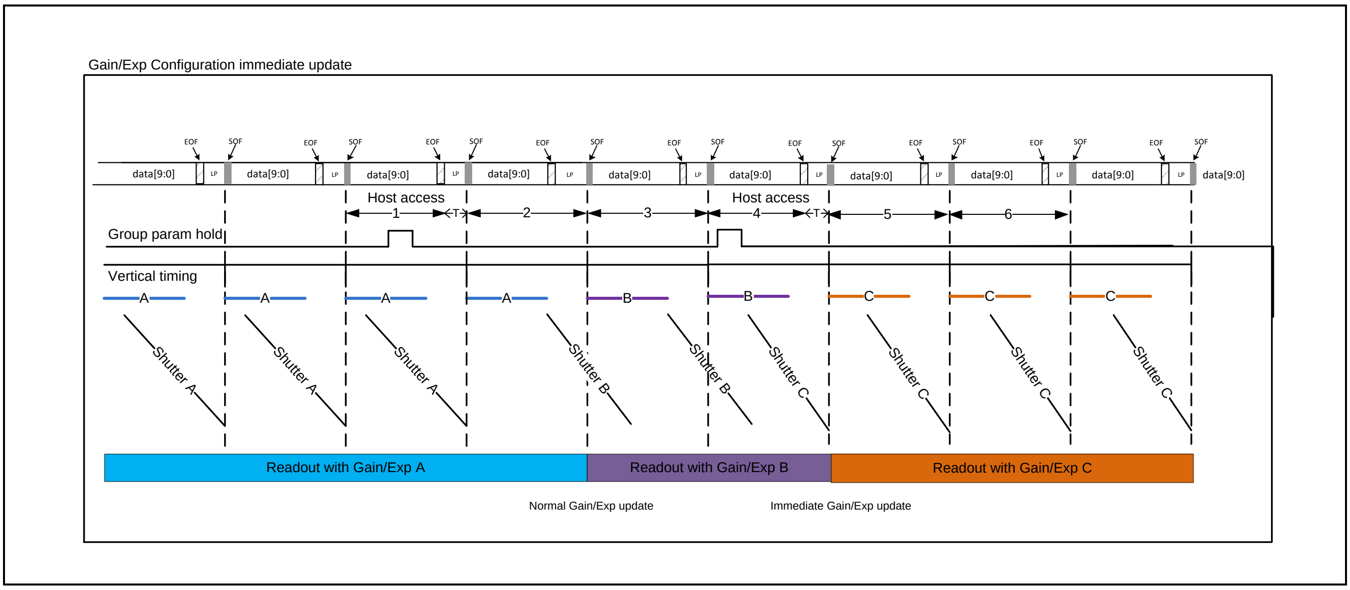


Figure 29 Exposure/Gain Immediate Configuration Change

Table 38 Exposure/Gain Immediate Configuration Change

Phase	Title	Description
1	Host access for immediate Gain/Exp	Host access for new Gain/Exp values. In case immediate update is required, S5K33DX will examine the option to apply new configuration for the following frame. Since Shutter already activated immediate update is denied.
2	Transition frame	At the following frame after host access, readout will still have mode A parameters, but shutter will start mode B operation.
3	New configuration frame	At the following frame readout will have mode B parameters.
4	Host access for immediate Gain/Exp	Host access for new Gain/Exp values (e.g. C). In case immediate update is required, S5K33DX will examine the option to apply new configuration for the following frame. Since Shutter did not start yet, new configuration is updated immediately. (The new shutter request is applicable only for the remaining time.) For immediate update, host must set the request <T> before the shutter begins.
5	New configuration frame	At the following frame readout will have mode C parameters.

8.2 Stream Interrupting Register Updates

When sensor frame timing configuration change is requested by host (Refer to section *8.2.1.2 Settings Affecting Frame Timing*) by register setting with "group parameter hold" request, firmware detects the change and forces frame timing change in one of the following options:

1. Preserve frame timing without outputting the corrupted frame to host (skip frame)

2. Preserve frame timing and output corrupted frame

3. Timing abort: After end frame (i.e. readout and VBLANK). In this mode, full frame timing will be completed before frame abort.

4. Timing abort: After frame readout ends but before VBLANK time.

5. Timing abort: Immediate abort (including during frame read out).

Frame timing examples may be found in the following figures;

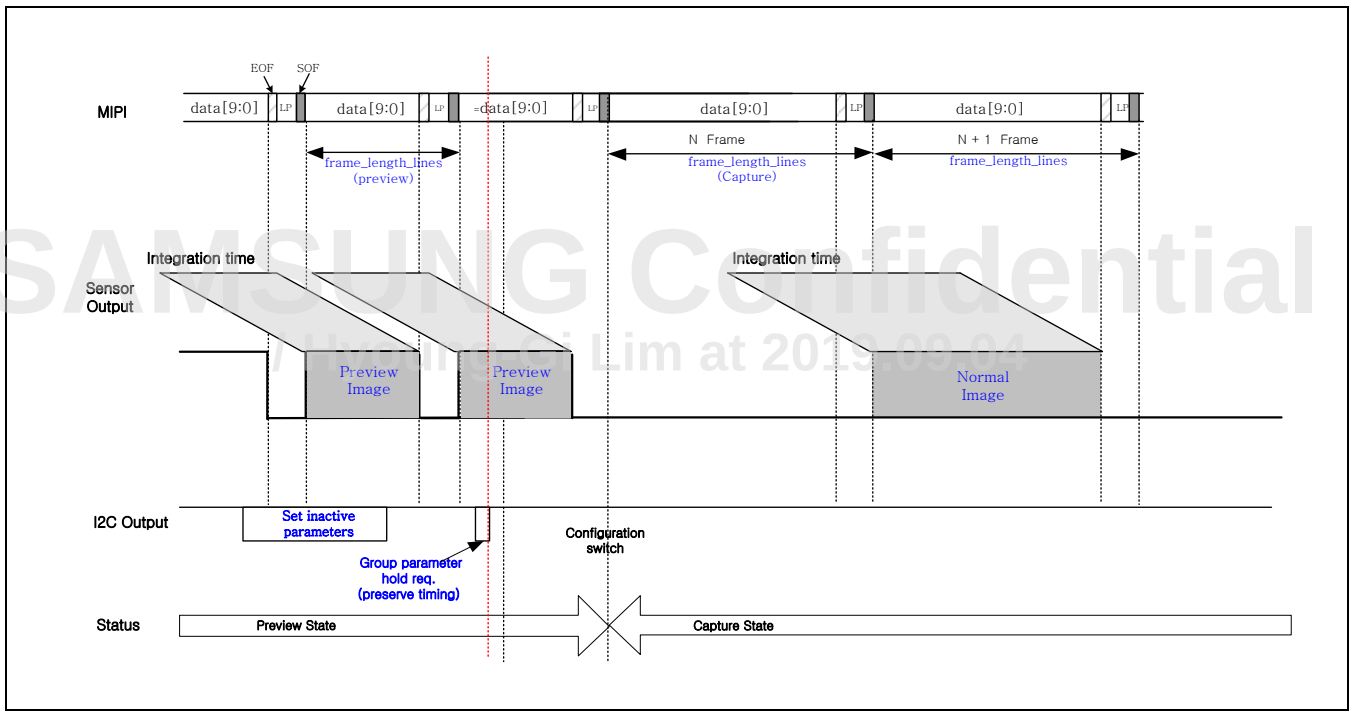


Figure 30 Example - Case (a); Preserve Frame Timing without Outputting the Corrupted Frame to Host

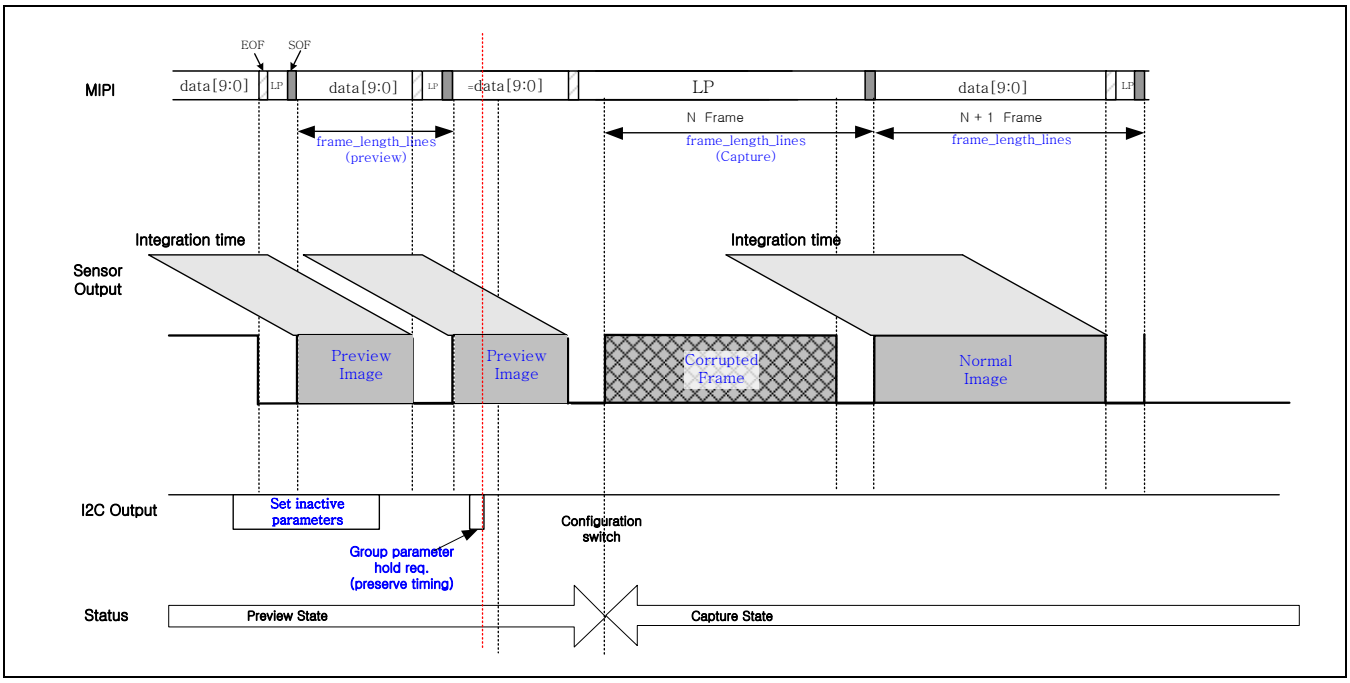


Figure 31 Example - Case (b); Preserve Frame Timing and Output Corrupted Frame

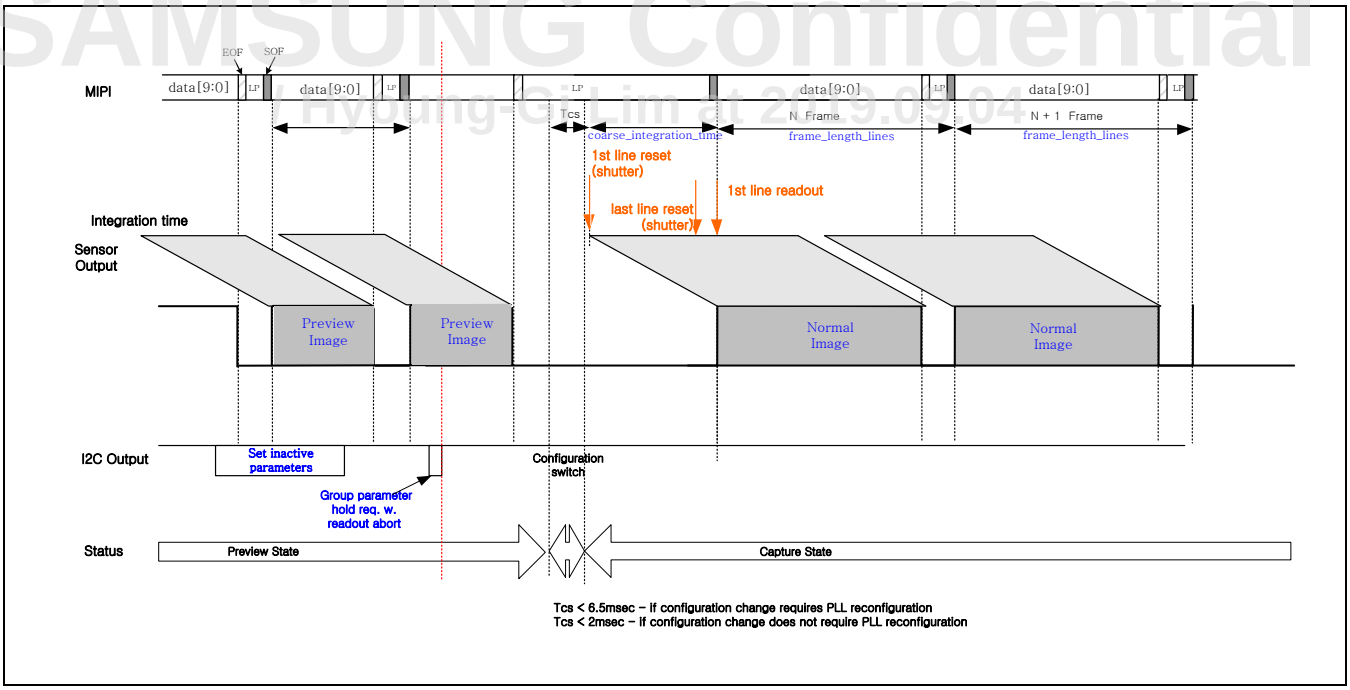


Figure 32 Example - Case (c); Timing abort: after End of Readout and VBLANK

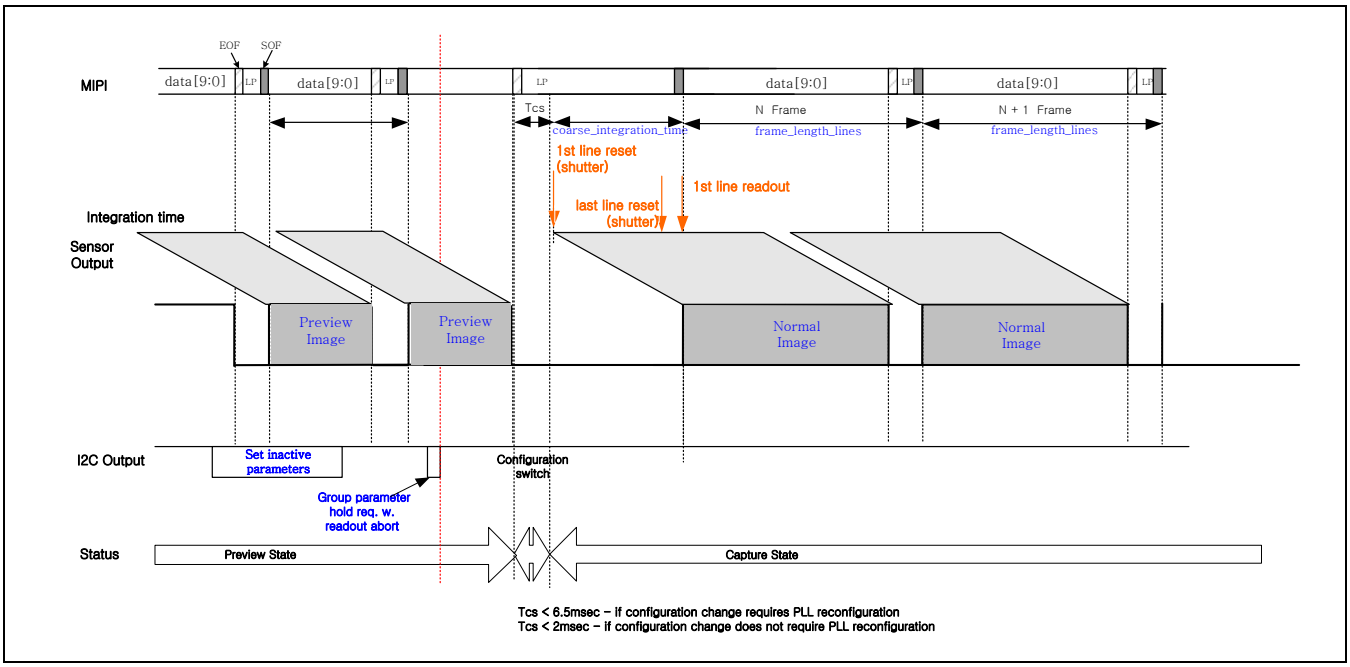


Figure 33 Example - Case (d); Timing abort: after Frame Readout Ends but before VBLANK Time

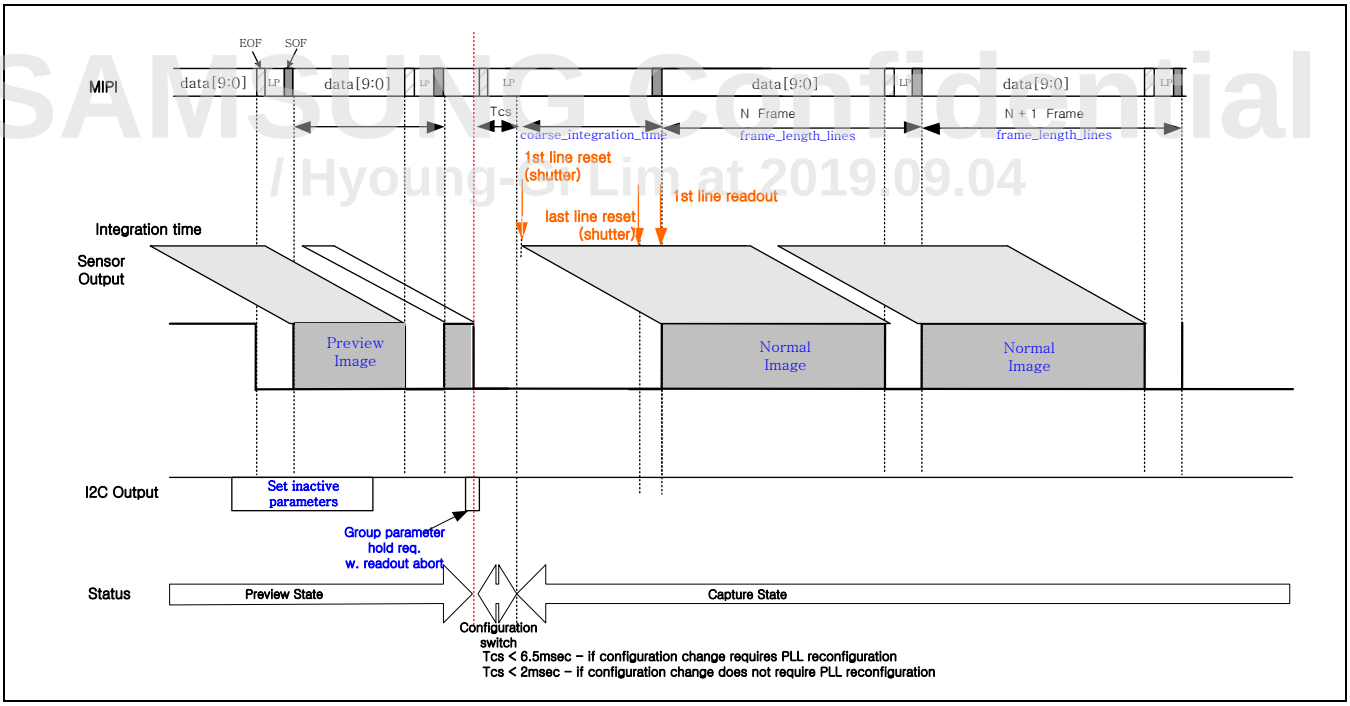


Figure 34 Example - Case (e); Timing abort: Immediate abort (Including During Frame Read Out)

8.2.1 Frame Timing Control Configuration

Table 39 Frame Timing Transition Options

	Timing Transition Type	Description
1	REQ_NO_ABORT_CORRUP_FRAME	Preserve timing and output the corrupted frame
2	REQ_NO_TIMING_ABORT	Preserve timing skip the corrupted frame
3	REQ_ABORT_ON_END_FRAME	Abort after end of timing frame
4	REQ_ABORT_ON_END_RO	Abort after frame read out ends
5	REQ_IMMEDIATE_ABORT	Abort immediately, even during frame read out (NOTE)

NOTE: The abort immediately has 3 exceptions where the command response will be delayed slightly:

1. Command detected during first 10 timing lines: In this case, firmware will delay the actual abort after the dark lines are finished, and OUTIF transmitted the MIPI short packet "frame-start".
2. Command detected during last 10 readout lines: In this case, firmware will delay the actual abort after readout ends (in order to prevent problematic timing of internal firmware interrupts).
3. Command detected during last 10 timing lines (just before frame ends anyway): In this case, firmware will delay the actual abort after normal API registers read that happens anyway around this time.

8.2.1.1 Rolling Shutter Frame Timing Configuration Change

Table 40 Rolling Shutter Frame Timing Transition Options

Timing Transition Type	Setting
REQ_NO_ABORT_CORRUP_FRAME	general_setup_mask_corrupted_frames = 0
REQ_NO_TIMING_ABORT	general_setup_mask_corrupted_frames = 1
REQ_ABORT_ON_END_FRAME	general_setup_mask_corrupted_frames = 2
REQ_ABORT_ON_END_RO	general_setup_mask_corrupted_frames = 3
REQ_IMMEDIATE_ABORT	general_setup_mask_corrupted_frames = 4

8.2.1.2 Settings Affecting Frame Timing

These are main parameters that if changed will result in frame timing configuration change:

Table 41 Parameters that Cause Frame Timing Change If Modified

Input Size, Mirror, Binning or Line Timing	PLL Modification
image_size_x_addr_start image_size_y_addr_start image_size_x_addr_end image_size_y_addr_end frame_timing_line_length_pck sub_sample_x_odd_inc sub_sample_x_even_inc sub_sample_y_odd_inc sub_sample_y_even_inc binning_mode binning_type general_setup_image_orientation api_rw_scaling_digital_scaling image_size_digital_crop_x_offset image_size_digital_crop_y_offset api_rw_scaling_hbin_digital_binning_factor	api_rw_clocks_vt_pix_clk_div api_rw_clocks_op_pre_pll_clk_div api_rw_clocks_op_pll_multiplier api_rw_clocks_vt_pre_pll_clk_div api_rw_clocks_vt_pll_multiplier api_rw_long_frame_timing_line_length_pck_shifter

8.3 Grouped Changes

8.3.1 Grouped Changes

The sensor must recognize a system by which a set of changes to these parameters can be grouped by the host system. These changes must be applied in such a way that they effect a change in the output pixel stream at a field boundary, as well as effecting each of the changes in the group that affect the output pixel stream at the same field boundary.

A group of parameter changes is marked by the host using a dedicated Boolean control parameter, grouped_parameter_hold. Any changes made to "retimed" parameters while the grouped_parameter_hold signal is in the "hold" state should be considered part of the same group.

The group of changes should be executed only when the grouped_parameter_hold control signal is moved back to the default "no-hold" state.

The usage of this mode is relevant only to SMIA registers.

Table 42 Grouped Change Control Parameter

Register Name	Address	Default Value	Description
api_rw_general_setup_grouped_parameter_hold	0x40000104	0x00	The grouped parameter hold register disables the consumption of integration, gain and video timing parameters. This register enables setting to an envelope a series of parameter changes as a group of changes that should be made so as to effect the output stream on the same frame boundary 0 = Consume as normal 1 = Hold

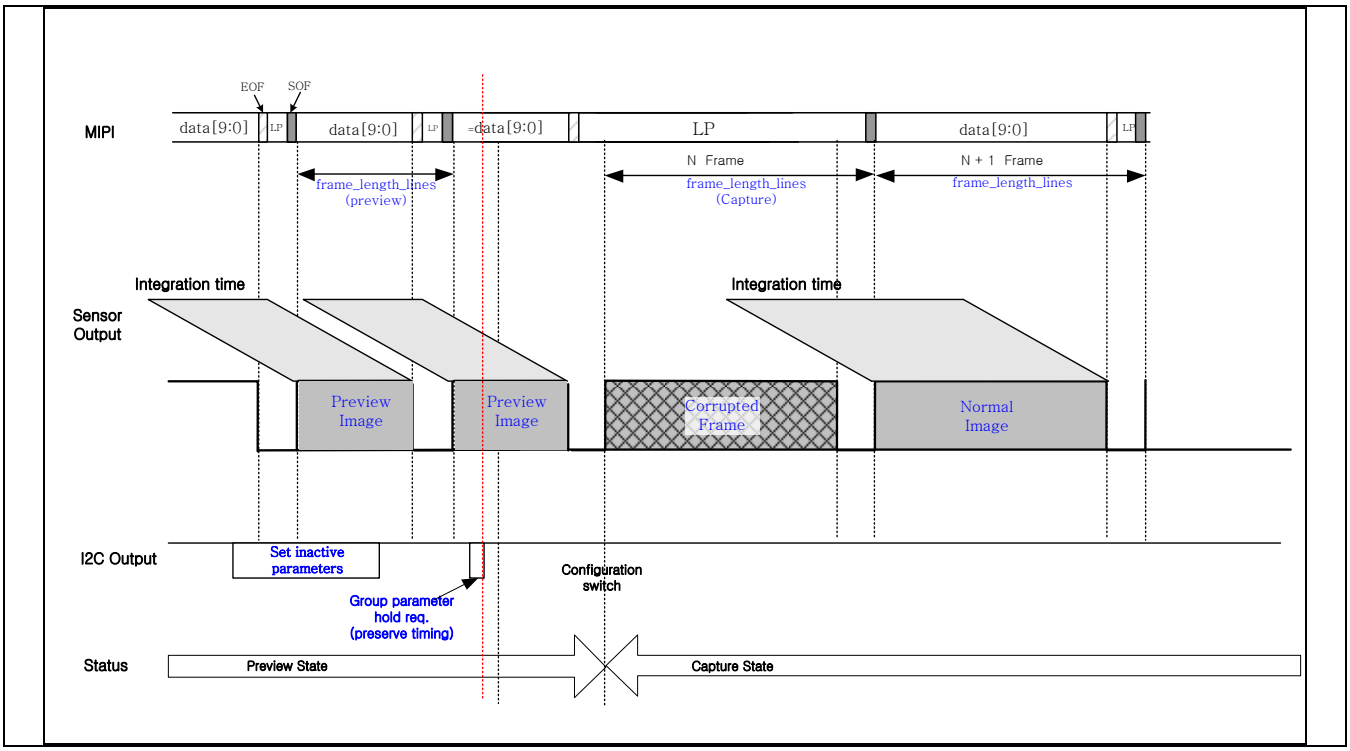


Figure 35 Group Parameter Hold

- Group parameter Hold is used to enable configuration to ignore synchronization issues
- Group parameter Hold enables cross-frame configuration

In the above figure, all new capture parameters are set when group parameters hold is applied. For example, when the group parameter hold is set to 0, the parameters are applied all at once, configuring the new frame after the current truncated frame.

8.4 Different PLL Settings

In case that the group of changes includes PLL settings modification, the sensor will automatically go through SW standby/wakeup sequence during the configuration change. This means that in MIPI mode, the sensor goes through the ULPM sequence (enter/exit) and re-calculates all timing generator functions (due to potentially different operating frequencies).

8.5 Frame Update Timing

See below timing diagram and calculations of timing periods required for frame update:

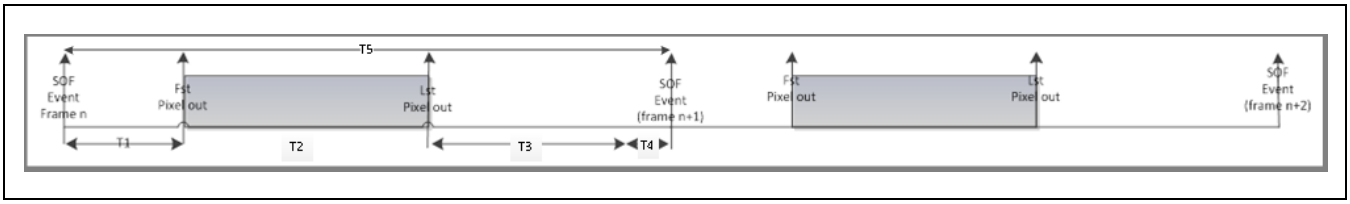


Figure 36 Frame Update Timing

Where:

<T1>	The period of time before output starts. Generally it is the time it takes to run FADLC lines and some extra FW calculation time (.)
<T2>	The period of output lines
<T3>	The period of time after Read Out, where user can give change commands that will be accepted in frame n timing. The time starts at last pixel out
<T4>	The period of time FW will not accept any change command given for frame n timing, but for frame n + 1 timing
<T5>	The total frame period timing

T1 (overhead lines):

Y Binning	Overhead Lines
1	82

T2 (output lines):

Case if api_rw_output_emb_use_header:

0 = api_rw_image_size_y_output_size + api_rw_output_emb_footer_lines
1 = api_rw_image_size_y_output_size + api_rw_output_emb_footer_lines+ api_rw_output_emb_header_lines

T4 (FW processing lines – host commands given during this period are delayed and will be processed by FW in next frame):

$$Ceil\left(\frac{30000 * 4}{api\_rw\_frame\_timing\_line\_length\_pck}\right)$$

Or by integer division:

$$\left(\frac{30000 * 4 + api\_rw\_frame\_timing\_line\_length\_pck - 1}{api\_rw\_frame\_timing\_line\_length\_pck}\right)$$

T5 (frame lines):

api_rw_frame_timing_frame_length_lines

T3 (host commands given during this period will be processed by FW in this frame):

$$T5 - (T1 + T2 + T4)$$

All above are in frame length lines units, In order to convert the values to time units, it is needed to multiply the values by line_lengh_pck value multiplied by time of 1 system clock tick. See calculation example below referring to the following sensor configuration:

$system\ clock\left(\frac{External\ clock}{pre\_div} * \frac{multiplyer*4}{vt\_sys}\right)$ **set to 1280 MHz**
api_rw_frame_timing_line_length_pck set to 5600
api_rw_frame_timing_frame_length_lines set to 3808

T1: 82 (359 uS)
T2: 3520 (15.04 ms)
T4: ceil(30000 ×4 / 5600) = 22 (~96 uS)
T5: 3808 (16.66 ms)
T3: 3808 – (82 + 3520 + 22) = 184 lines (~809 uS)

NOTE: The calculation of T2 period is under consideration that down scaling is not used.

9 DTP

This block handles the optional DTP injection which can be generated only at the beginning sensor data path, i.e. SMIA DTP is not supported.

9.1 Full Frame Deterministic Test Patterns

Two types of full frame deterministic test patterns are defined. Most are Bayer test patterns that more suitable for some tests than real image data and are injected early in the sensor data path. The only exception to this is a test pattern that is intended to test sensor-host link integrity. The data in this pattern is not Bayer data, and it is injected just prior to CSI framing. Use of these full frame test patterns is controlled by the test_pattern_mode parameter. The following table shows all the defined parameter settings.

Table 43 Main Full Frame Test Pattern Control Parameters

Register Name	Address	Default Value	Description
test_pattern_test_pattern_mode	0x40000600	0x0000	Controls the output of the test pattern module 0 = No pattern (default) 1 = Solid color 2 = 100% color bars 3 = Fade to grey color bars 4 = PN9

In both the default parameter state and in any undefined parameter states, normal array data should be output rather than a test pattern. The individual test patterns are described later in this chapter. Where the camera module supports non-baseline features (e.g., digital gain) or manufacturer-specific features, the operation of the test patterns may be undefined, outside the default (baseline) state.

9.2 Scaled & Cropped and Re-Orientated Output

If the sensor includes image-scaling options, the test patterns need only be generated according to the specification when the scaling hardware is disabled.

9.3 Solid Color Mode

In the solid color test pattern mode, all pixel data is replaced with fixed Bayer test data that is defined by the four 'test data color' parameters, these are described in the following table.

Table 44 Test Data Color Parameters

Register Name	Address	Bit Width	Default Value	Description
api_rw_test_data_red	0x40000602	[9:0]	0x0000	The test data used to replace Phase-A3 pixel data on even rows
api_rw test_data_green_red	0x40000604	[9:0]	0x0000	The test data used to replace Phase-A0 pixel data on even rows
api_rw test_data_blue	0x40000606	[9:0]	0x0000	The test data used to replace Phase-A1 pixel data on odd rows (not in use in 2tap readout modes)
api_rw test_data_geeen_blue	0x40000608	[9:0]	0x0000	The test data used to replace Phase-A2 pixel data on odd rows (not in use in 2tap readout modes)

9.4 Gradient Mode

Table 45 Test Data Color Parameters

Register Name	Address	Bit Width	Default Value	Description
api_rw_test_pattern_fedtp_gradient_enable	0x40000615	[7:0]	0x00	bit 1: gradient enable Y. bit 0: gradient enable X
api_rw_test_pattern_fedtp_gradient_invert	0x40000616	[7:0]	0x00	bit 1: Invert address Y. bit 0: Invert address X.
api_rw_test_pattern_fedtp_gradient_hor_shift	0x40000617	[7:0]	0x00	Scaling factor for X dependent gradient
api_rw_test_pattern_fedtp_gradient_ver_shift	0x40000618	[7:0]	0x00	Scaling factor for Y dependent gradient

10 Embedded Data Lines

Embedded data lines include extra information regarding sensor's status and operational mode, which is included within the output data stream. Examples for such data are frame timing parameters, integration time and gain parameters etc.

The embedded data is synchronized with the frame data, thus allowing the host to track mode changes.

10.1 Embedded Data Lines Usage in MIPI Mode

- The first byte of the 2-byte packet is the tag byte. The tag byte value defines the meaning of the second byte, the data byte.
- The embedded data comprises a sequence of the above tagged data packets and is terminated by a data packet with the special end of data tag, 07_H.
- The 2-byte tagged data format uses 5 different tag codes.

Table 46 MIPI Simplified 2-Byte Tagged Data Format Tag Codes

Tag	Data Byte Description
00_H	Illegal Tag. If found treat as end of data
07_H	End of data (data byte value = 07_H)
AA_H	CCI register index MSB [15:8]
$A5_H$	CCI register index LSB [7:0]
$5A_H$	Auto increment the CCI index after the data byte – valid data Data byte contains valid CCI register data
55_H	Auto increment the CCI index after the data byte – null data A CCI register does NOT exist for the current CCI index. The data byte value is the 07_H.
FF_H	Illegal Tag. If found treat as end of data

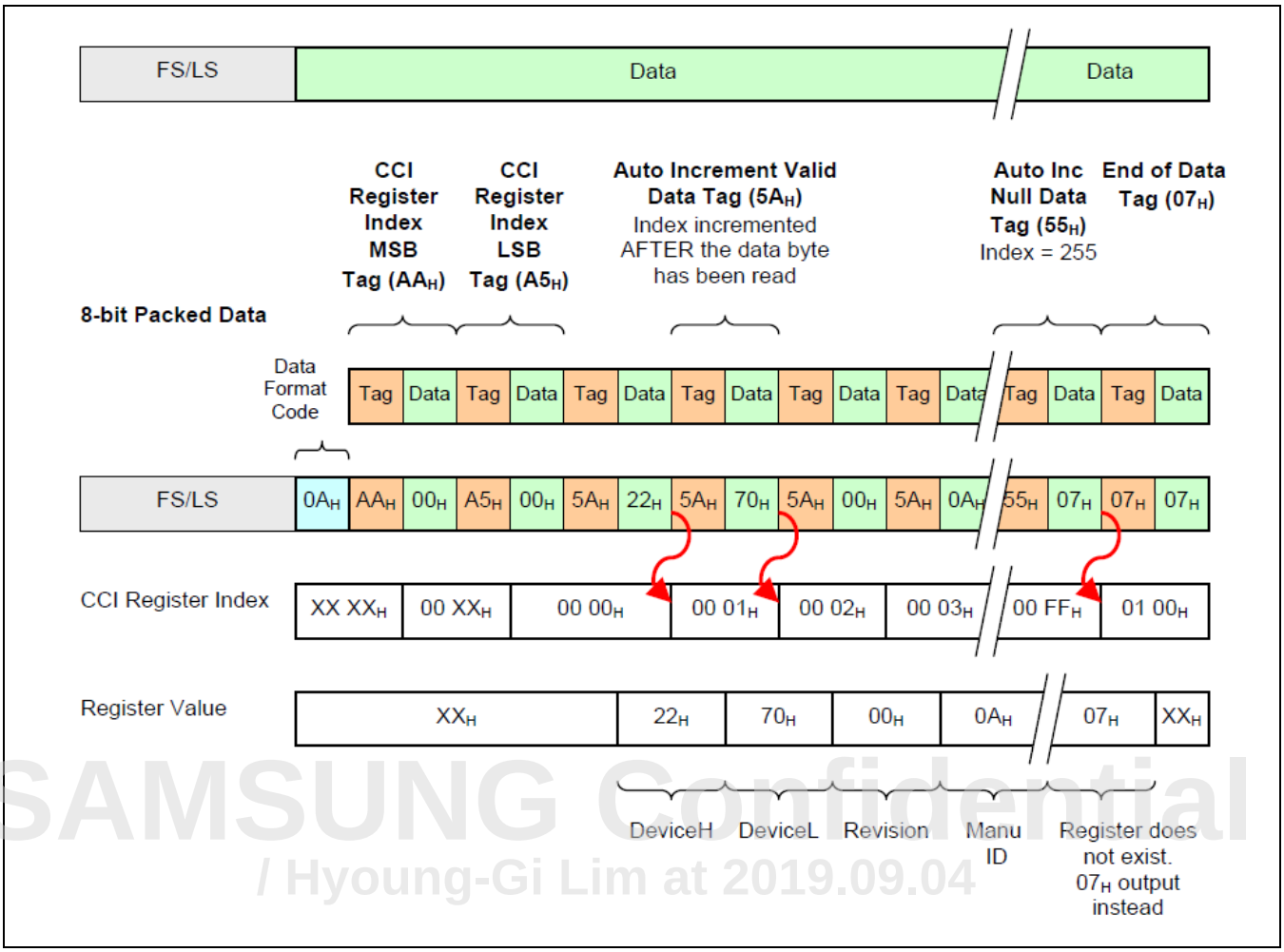


Figure 37 MIPI Simplified 2-Byte Tagged Data Format Tag Codes

S5K33DX has two modes for embedded data:

1. Embedded data line length is shorter then data line length.
2. Embedded data line length is same as visible line length.

Table 47 Mode for Embedded Data

Register Name	Address	Default Value	Description
api_rw_output_emb_use_header	0x40000118	0x00	Enable/Disable the embedded line header 0 = Disable 1 = Enable
api_rw_output_emb_header_lines	0x40000119	0x2	Defines the number of embedded lines (1-4)
api_rw_output_emb_footer_lines	0x4000011A	0x0	Enable/Disable the embedded line footer 0 = Disable 1 = Enable
api_rw_output_emb_equal_to_data	0x4000011B	0x1	Maintain embedded line same length as pixels line 0 = Embedded line length is shorter than data line length 1 = Embedded line length is equal to data line length Note: This is only the line length, not the actual byte that used for data (fixed to 256 pixels each line).

The simplified 2-byte tagged data format is used also when output interface is MIPI. See below diagrams for general line structure description.

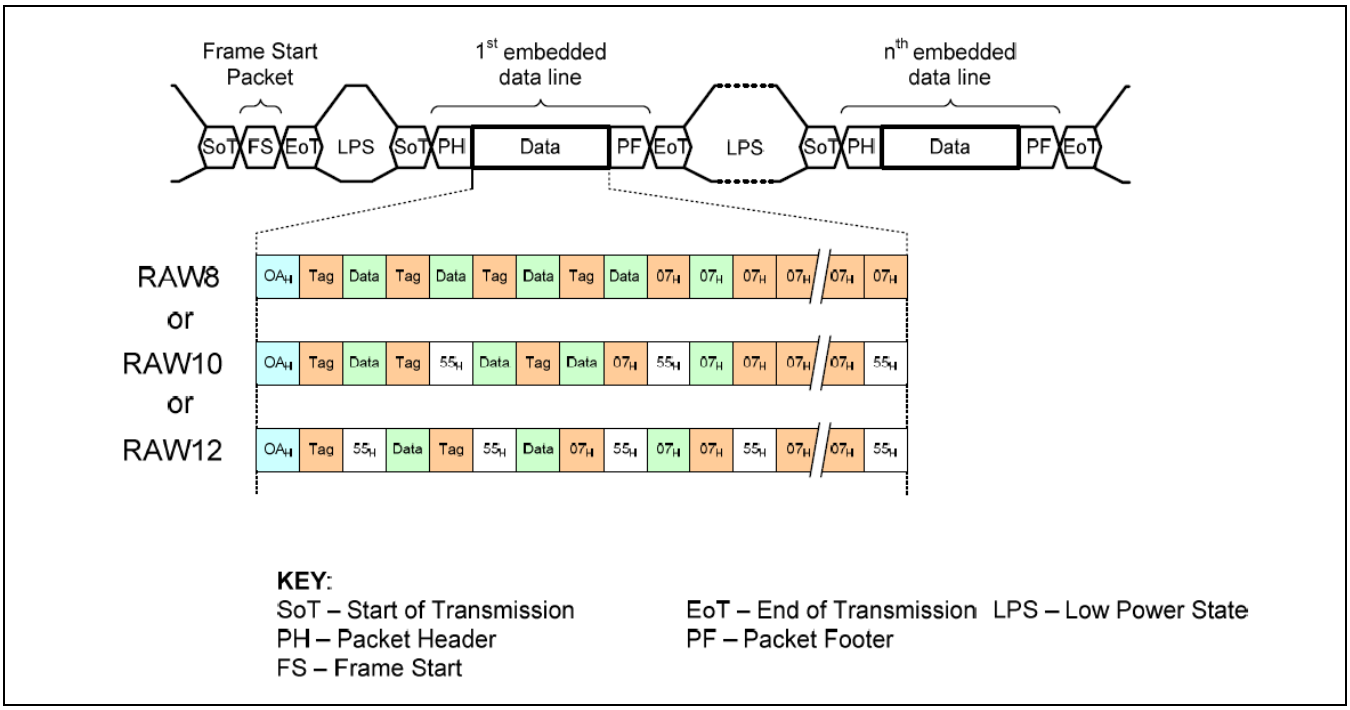


Figure 38 MIPI Simplified 2-Byte Tagged Data line Structure

10.1.1 Coding example

Set 02 to address MSB

Code	Data
AA	02

Set 02 to address MSB, Set 01 to address LSB => pointer is equal to 0201

Code	Data	Code	Data
AA	02	A5	01

Set the address to 0x0201, data in 0x201 is equal to 0x30,
Increment pointer by 1

Code	Data	Code	Data	Code	Data
AA	02	A5	01	5A	30

Set the address to 0x0201,
Data in 0x200 is equal to 0x30, increment pointer by 1
Data in 0x201 is equal to 0xC2, increment pointer by 1

Code	Data	Code	Data	Code	Data	Code	Data
AA	02	A5	01	5A	30	5A	C2

10.2 Embedded Data Lines Supported Fields List:

33D_EVT1_ELG_Example_V1.xlsx

10.3 Key Information in Embedded Data Lines

10.3.1 Frame Information

Table 48 Registers in Embedded Data Lines for Frame Information

Byte Index	Register Name	Size	Description
16	api_rd_general _frame_count	1Byte	Number of current frame
180	api_rd_current_state_time_of_flight _output_format	1Byte	0x01: 4-tap, single-freq., non-shuffle 0x0A: 4-tap, dual-freq., shuffle
182	api_rd_current_state_time_of_flight _phase_map	1Byte	0x10: non-shuffle 0x1B: shuffle
184	api_rd_current_state_time_of_flight _frequency_modulation_index	1Byte	0x00: f1 (100MHz) 0x01: f2 (80MHz)

Example:

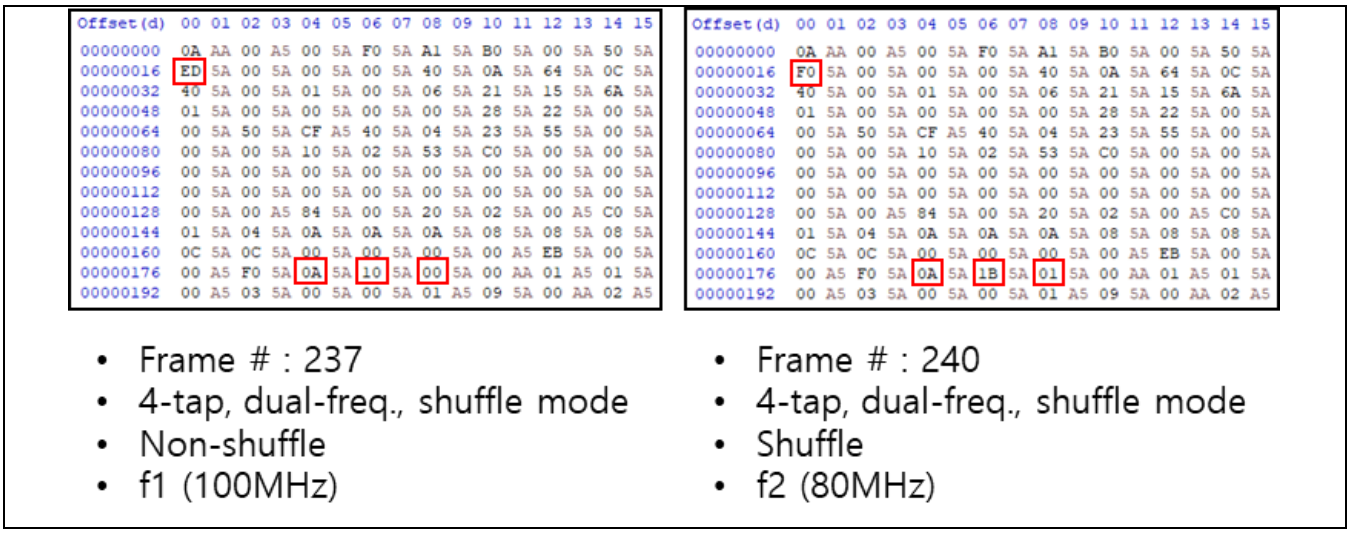


Figure 39 Example of Frame Information in Embedded Data Lines

10.3.2 Frame Rate

Table 49 Registers in Embedded Data Lines for Frame Rate

Byte Index	Register Name	Size	Description
320	api_rw_clocks_vt_pix_clk_div	2Byte	Refer to Table 33.
324	api_rw_clocks_dummy_reserved_vt_sys_clk_div	2Byte	
328	api_rw_clocks_vt_pre_pll_clk_div	2Byte	
332	api_rw_clocks_vt_pll_multiplier	2Byte	
362	api_rw_frame_timing_frame_length_lines	2Byte	Refer to Table 23.
366	api_rw_frame_timing_line_length_pck	2Byte	

Calculation:

$$vt\_pix\_clk\_freq = 24MHz \times \frac{\boldsymbol{vt\_pll\_multiplier} \times 4}{\boldsymbol{vt\_pre\_pll\_clk\_div} \times \boldsymbol{vt\_sys\_clk\_div} \times \boldsymbol{vt\_pix\_clk\_div}}$$

$$Frame\ Rate[frames/sec] = \frac{vt\_pix\_clk\_freq}{\boldsymbol{frame\_length\_lines} \times \boldsymbol{line\_length\_pck}}$$

Example:

Offset(d)	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
00000288	00	5A	00	5A	00	5A	00	A5	30	5A	01	5A	00	5A	01	5A
00000304	00	5A	01	5A	00	5A	01	5A	00	AA	03	A5	00	5A	00	5A
00000320	0A	5A	00	5A	01	5A	00	5A	04	5A	00	5A	78	5A	00	5A
00000336	08	5A	00	5A	01	5A	00	5A	00	5A	00	5A	04	5A	00	5A
00000352	85	5A	00	5A	01	A5	40	5A	06	5A	64	5A	16	5A	E8	5A
00000368	00	5A	04	5A	00	5A	04	5A	05	5A	03	5A	03	5A	C3	5A
00000384	05	5A	00	5A	03	5A	C0	A5	80	5A	00	5A	01	5A	00	5A
00000400	01	5A	00	5A	01	5A	00	5A	01	AA	04	A5	02	5A	10	5A
00000416	10	5A	10	5A	00	AA	09	A5	00	5A	00	5A	11	5A	00	AA
00000432	0B	A5	00	5A	00	5A	00	5A	01	A5	05	5A	01	5A	00	5A
00000448	01	5A	00	5A	00	5A	01	5A	01	5A	01	5A	00	AA	CC	A5
00000464	00	5A	01	5A	2A	5A	01	07	07	07	07	07	07	07	07	07
00000480	07	07	07	07	07	07	07	07	07	07	07	07	07	07	07	07
00000496	07	07	07	07	07	07	07	07	07	07	07	07	07	07	07	07

Figure 40 Example of Frame Rate in Embedded Data Lines

$$\boldsymbol{vt\_pix\_clk\_div} = 0\text{x}000\text{A} = 10$$

$$\boldsymbol{vt\_sys\_clk\_div} = 0\text{x}0001 = 1$$

$$\boldsymbol{vt\_pre\_pll\_clk\_div} = 0\text{x}0004 = 4$$

$$\boldsymbol{vt\_pll\_multiplier} = 0\text{x}0078 = 120$$

$$\boldsymbol{frame\_length\_lines} = 0\text{x}0664 = 1636$$
$$\boldsymbol{line\_length\_pck} = 0\text{x}16\text{E}8 = 5864$$

$$vt\_pix\_clk\_freq = 24MHz \times \frac{\mathbf{120} \times 4}{\mathbf{4 \times 1 \times 10}} = 288MHz$$
$$Frame\ Rate = \frac{288{,}000{,}000}{\mathbf{1636 \times 5864}} \cong 30.02\ frames/sec$$

10.3.3 Sensor Temperature

Table 50 Registers in Embedded Data Lines for Sensor Temperature

Byte Index	Register Name	Size	Description
28	api_rd_general_temperature	2Byte	Sensor temperature

Calculation:

$$Sensor\ Temp.\ [℃] = \boldsymbol{api\_rd\_general\_temperature}(Signed\ number\ with\ 8\ fraction\ bits)$$

Example:

Offset(d)	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
00000000	0A	AA	00	A5	00	5A	F0	5A	A1	5A	B0	5A	00	5A	50	5A
00000016	70	5A	00	5A	00	5A	01	5A	00	5A	20	5A	8C	5A	0C	5A
00000032	40	5A	00	5A	01	5A	00	5A	06	5A	21	5A	15	5A	6A	5A
00000048	01	5A	00	5A	00	5A	00	5A	00	5A	28	5A	1B	5A	00	5A
00000064	00	5A	50	5A	3C	A5	40	5A	04	5A	23	5A	55	5A	00	5A
00000080	00	5A	00	5A	10	5A	02	5A	53	5A	C0	5A	00	5A	00	5A
00000096	00	5A	00	5A	00	5A	00	5A	00	5A	00	5A	00	5A	00	5A
00000112	00	5A	00	5A	00	5A	00	5A	00	5A	00	5A	00	5A	00	5A
00000128	00	5A	00	A5	84	5A	00	5A	20	5A	02	5A	00	A5	C0	5A

Figure 41 Example of Sensor Temperature in Embedded Data Lines

$$\boldsymbol{api\_rd\_general\_temperature} = 0\text{x}208\text{C}$$

$$Sensor\ Temp. = (\mathbf{20.8C})_{hex}$$
$$= (0010\ 0000\ .\ 1000\ 1100)_2$$
$$= 32.546875\ ℃$$

10.3.4 Driver IC Temperature

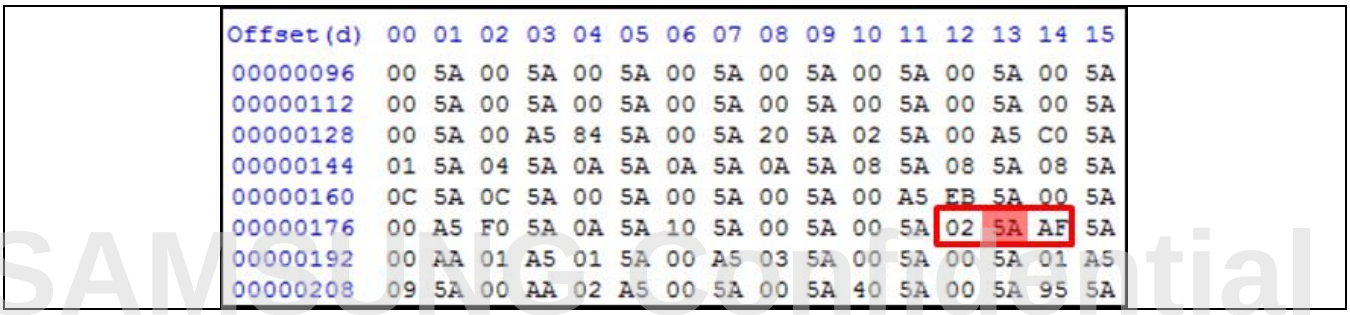
Table 51 Registers in Embedded Data Lines for Driver IC Temperature

Byte Index	Register Name	Size	Description
190	api_rd_current_state_led_driver_ic_temperature	2Byte	Driver IC temperature

Calculation:

$$Driver\ IC\ Temp.\ [℃] = 25 + \frac{\boldsymbol{tnp\_mon\_led\_driver\_ic\_temperature} - 296}{5.4}$$

Example:



Offset(d)	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
00000096	00	5A	00	5A	00	5A	00	5A	00	5A	00	5A	00	5A	00	5A
00000112	00	5A	00	5A	00	5A	00	5A	00	5A	00	5A	00	5A	00	5A
00000128	00	5A	00	A5	84	5A	00	5A	20	5A	02	5A	00	A5	C0	5A
00000144	01	5A	04	5A	0A	5A	0A	5A	0A	5A	08	5A	08	5A	08	5A
00000160	0C	5A	0C	5A	00	5A	00	5A	00	5A	00	A5	EB	5A	00	5A
00000176	00	A5	F0	5A	0A	5A	10	5A	00	5A	00	5A	02	5A	AF	5A
00000192	00	AA	01	A5	01	5A	00	A5	03	5A	00	5A	00	5A	01	A5
00000208	09	5A	00	AA	02	A5	00	5A	00	5A	40	5A	00	5A	95	5A

Figure 42 Example of Driver IC Temperature in Embedded Data Lines

$$\boldsymbol{tnp\_mon\_led\_driver\_ic\_temperature} = 0x02AF = 687$$

$$Driver\ IC\ Temp. = 25 + \frac{\mathbf{687} - 296}{5.4}$$

$$\cong 97.41\ ℃$$

11 Output Interface

The S5K33DX includes MIPI 4 lanes, for sensor pixel signal output. The MIPI protocol is output via a dedicated interface, and the selected interface can be configured using register settings. Unused pins can be left unconnected, as described in the assembly information document.

S5K33DX supports the following signaling options:

- 1-4 lane solution as specified in MIPI.

The **api_rw_output_signalling_mode** register controls the signaling method used. Mode switching is allowed in SW Standby mode.

Table 52 Signaling Mode Control Registers

Register Name	Address	Default Value	Description
api_rw_output_signalling_mode	0x40000111	0x02	0 = PVI 2 = CSI-2 with DPHY (MIPI)

Channel identifier – A virtual channel that is stored on MIPI packets in order to identify a specific stream for the host. Its value is 0 by default, and the MIPI can be attached to the channel in order to distinguish between stream types, e.g., between preview and capture. The receiver is expected to check this identifier and act accordingly. For example, a receiver can define that it would like to receive data on channel 4, and unless the data is packed with this identifier it will be rejected by the host.

The **api_rw_output_lane_mode** register determines with how many lanes the interface will work.

Table 53 Lane Mode Control Registers

Register Name	Address	Default value	Description
api_rw_output_lane_mode	0x40000114	0x00	Number of CSI-2 lanes to be used 0 = One lane 1 = Two lanes

11.1 Serial Output Interface (MIPI CSI-2)

S5K33DX MIPI CSI-2 interface is a four-lane high-speed serial interface that connects the camera sensor to a host processor. Maximum bitrate of MIPI of S5K33DX is 1.6Gbps per lane.

S5K33DX supports all mandatory requirements in MIPI CSI-2 version 1.00 and DPHY 1.2 specifications. Please see MIPI DPHY 1.2 specification for details.

- Main output frame rates:
    - VGA RAW Phase: 120fps
    - QVGA RAW Phase: 180fps
    - QQVGA RAW Phase: 240fps
    - VGA Depth: 60fps
    - VGA Depth + Confidence: 60fps
- MIPI CSI-2: Four lanes with a maximum of 1.6Gbps (bit clock rate)
-

11.1.1 ULPM Mode

The DPHY requires a significant setup time to wake up. Thus, when waking up, there is an option to move to Ultra Low Power Mode (ULPM),

Table 54 MIPI ULPM mode Control Register

Register Name	Address	Default Value	Description
api_rw_output_standby_power_mode	0x4000011F	0x00	MIPI DPHY standby power mode: 0 = ULPM. 1 = LP11 (lane lines stays high) 2 = Disabled (lane lines are disconnected)

11.1.2 Continuous and Non-Continuous Modes

MIPI interface clock can be active continuously during all frame length, or it can be shut down in different periods of times when there is no data running.

The purpose of the clock shut down is power saving.

The following register controls this function:

Table 55 MIPI Continuous and Non-Continuous modes Control Register

Register Name	Address	Default Value	Description
api_rw_output_qual_mode_continues_enable	0x4000011E	0x01	0 = Legacy mode of continuous clock (PVI clock is always on)

11.1.3 MIPI I/F General Control Registers

The following table describes the registers controlling the MIPI I/F general functionality:

Table 56 MIPI IF Control Registers

Register Name	Address	Default Value	Description
api_rw_output_emb_equal_to_data	0x4000011B	0x01	Maintain embedded line same length as pixels line 0 = Embedded line length is shorter than data line length 1 = Embedded line length is equal to data line length Note: This is only the line length, not the actual byte that used for data (fixed to 256 pixels each line).
api_rw_output_emb_in_bytes	0x4000011C	0x01	Maintain embedded line pixel bitage same as pixels data bitage 1 = Each embedded line pixel is byte 0 = Each embedded line pixel is equal to data pixel
api_rw_output_standby_power_mode	0x4000011F	0x00	MIPI DPHY stand-by power mode 0 = ULPM 1 = LP11 (lane lines stay high) 2 = Disabled (lane lines are disconnected)

12 Non-Volatile Memory (NVM)

The NVM is a type of flash with a size of 4 KByte (2x4KByte), used primarily in order to store tables, grids and parameters for MSM and static BPC. If additional space exists, it can be used to store host data.

The NVM memory module is a non-volatile OTP (one-timing programming) memory module. This module enables the saving of unique data to each chip at the production stage.

OTP memory is used to store the following unique information:
- Chip_id data – production history data to be stored during die sorting
- Parameter for Lens shading

There are three different ways to access each NVM memory:
- Externally, using IO pads by configuring the chip to a dedicated test mode and performing full OTP memory IP write/read protocol.
- Internally, through the APB bus using the controller.

The S5K33DX is capable of storing Firmware setfile in the OTP memory. This is useful for shortening its power-up sequence as loading setfile via the serial interface can be skipped in this case.

This memory is ordered in pages, 128 pages of 64 bytes each.
The data is written bit-by-bit, where the bit read time is 300 micro seconds, and write time is 40 micro seconds.
The mechanism of writing into this memory type is as follows:
The memory is initially all zeros. Upon writing, whenever a certain input bit has a value of 1, the bit at the related NVM is raised to 1.
NOTE: **The process of raising an NVM bit to 1 is irreversible (!).**

Page number 0 in the NVM is reserved for storing the chip number, EVT number, and any other model-specific data that cannot be put on the ROM.

There are two modes of operation:

- Direct – The input data is automatically written into the NVM. There is no need for polling, as the data writing is fast.
- Polling – Before each writing/reading operation, the host has to poll certain parameters. In addition, the input data has to be written to an intermediate buffer, from which the FW actually reads the input. Note that when the control interface is I2C, then there is no need to wait between signaling that a writing phase has begun and the actual placement of the data in the buffer. This is because I2C is relatively slow. On SPI, on the other hand, there is a need for timeout definition, which ensures that data written to the temp buffer remains valid.

12.1 Data Upload and Download Interface

12.1.1 NVM I/F Control Registers

Table 57 NVM Control Registers

Register Name	Address	Default Value	Description
api_ro_data_transfer_if_capability	0x40001800	0x01	Bit 0: 1 = interface 1 supported Bit 1: reserved, report value 0 Bit 2: 0 = polling not needed in reading 1 = polling needed in reading. Bit 3: 0 = polling not needed in writing 1 = polling needed in writing. Bit 4: 0 = supports single DTI functionality. 1 = supports multi DTI functionality Bits 0-3 are valid in single DTI functionality only
api_rw_data_transfer_ctrl	0x40000A00	0x00	To initiate read/write sequence: [0]: 1 = Enable, 0 = Disable [1]: 1 = Write enable, 0 = Read enable [2]: 1 = Pending errors, read errors register. (For more information, refer to Protocol Documentation.)
api_rw_data_transfer_errors	0x40000A01	0x00	Error flags: [0] = Data corrupted error (failed to read from OTP) [1] = Error bit is not cleared before operation [2] = Trying to write to bit that is already set in OTP [3] = Request set (ctrl bit 0 set) during processing request [4] = Wrong page number
api_rw_data_transfer_page_select	0x40000A02	0x00	Each page is 64 bytes. Do not exceed OTP size.
api_rw_data_transfer_data	0x40000A04 - 0x40000A43	0x00	64 x 8-bit register for read or write accesses

12.2 Functional Specification

12.2.1 Read Sequence

Data transfer interface can be used for reading. A typical read sequence would be as follows:

1. The host selects the appropriate page by writing the required value to register `api_rw_data_transfer_page_select`.
2. The host writes 0x01 to `api_rw_data_transfer_ctrl`. The camera module will start preparing data in read interface.
3. The host reads data from registers `api_rw_data_transfer_data`. The selected page is presented as 64 x 8-bit words (512 bits total).
4. If access to a further page is needed, the host can select a new page by register `api_rw_data_transfer_page_select` and after that follow step 3 again.
5. When host has read all the needed data, the host disables read interface by writing 0x00 to `api_rw_data_transfer_ctrl` register.

12.2.2 Example for Read Sequence

```
6010  0001
p10
0136  1800    // Input clock in MHz in higher byte (1800 = 24 MHz)
0304  0004    // vt_pre_pll_clk_div (default = 4 clock post div = 24/4 = 6 MHz)
0306  0078    // vt_pll_multiplier (6*120 = 720 MHz)
0300  000A    // vt_pix_clk_div
030E  0004    // op_pre_pll_clk_div
0312  0000    // op_pll_post_scalar
0310  005B    // op_pll_multiplier
0100  0100    // stream enable

p10
0A02  0500    // page 1-15 (default is page 0, but this page is saved) MSB is activate page selection (5 in this example)
0A00  0100    // NVM read enable
p5
// At the end of the sequence 1 page of NVM data (64 bytes) is available in RAM and may be read
// from addresses 0x0A04-0x0A43

In this example, Reading page 5
After this sequence, data is ready in bytes 0x0A04-0x0A43

0A00  0000    // NVM read disable
```

12.2.3 Write Sequence

Data transfer interface can also be used for writing. Write sequence would be as follows:

1. The host selects the appropriate page by writing the required value to register `api_rw_data _transfer_page_select`.
2. The host writes 0x03 to `api_rw_data _transfer_ctrl` register. This enables writing for interface 1. The camera module prepares write interface.
3. The host writes data to registers `api_rw_data_transfer_data`. This data represents the data in the selected page.
4. When writing has ended, a new page can be selected as in step 1 and the host can continue writing as defined in step 3.
5. Steps 1-4 are repeated until all needed data is written.
6. When writing has ended, the host writes 0x00 to `api_rw_data _transfer_ctrl` register. This disables the write interface.

Notes:

1. The host cannot read during write or write during read
2. Shadowing is not supported so there no way to be sure that the request was buffered before moving to new page.
3. Polling the write_if_ready bit is done for each page separately, so the host must wait for the interface to be ready before moving to next page.

12.2.4 Example for write Sequence


```
6010 0001
p10
0136  1800    // Input clock in MHz in higher byte (1800 = 24 MHz)
0304  0004    // vt_pre_pll_clk_div (default = 4 clock post div = 24/4 = 6 MHz)
0306  0078    // vt_pll_multiplier (6*120 = 720 MHz)
0300  000A    // vt_pix_clk_div
030E  0004    // op_pre_pll_clk_div
0312  0000    // op_pll_post_scalar
0310  005B    // op_pll_multiplier
0100  0100    // stream enable
p10
0A02  0500    // select page 0-15 (default is page 0 - page 5 in this example)
0A00  0300    // NVM write enable
0A04  0000    // NVM byte 0, 1   in selected page
0A06  0000    // NVM byte 2, 3   in selected page
0A08  0000    // NVM byte 4, 5   in selected page
0A0A  0000    // NVM byte 6, 7   in selected page
0A0C  0000    // NVM byte 8, 9   in selected page
0A0E  0000    // NVM byte 10,11 in selected page
0A10  ABCD    // NVM byte 12,13 in selected page
0A12  EFGH    // NVM byte 14,15 in selected page
0A14  1234    // NVM byte 16,17 in selected page
0A16  5678    // NVM byte 18,19 in selected page
0A18  0000    // NVM byte 20,21 in selected page
0A1A  0000    // NVM byte 22,23 in selected page
```

```
0A1C  0000    // NVM byte 24,25 in selected page
0A1E  0000    // NVM byte 26,27 in selected page
0A20  0000    // NVM byte 28,29 in selected page
0A22  0000    // NVM byte 30,31 in selected page
0A24  0000    // NVM byte 32,33 in selected page
0A26  0000    // NVM byte 34,35 in selected page
0A28  0000    // NVM byte 36,37 in selected page
0A2A  0000    // NVM byte 38,39 in selected page
0A2C  0000    // NVM byte 40,41 in selected page
0A2E  0000    // NVM byte 42,43 in selected page
0A30  0000    // NVM byte 44,45 in selected page
0A32  0000    // NVM byte 46,47 in selected page
0A34  0000    // NVM byte 48,49 in selected page
0A36  0000    // NVM byte 50,51 in selected page
0A38  0000    // NVM byte 52,53 in selected page
0A3A  0000    // NVM byte 54,55 in selected page
0A3C  0000    // NVM byte 56,57 in selected page
0A3E  0000    // NVM byte 58,59 in selected page
0A40  0000    // NVM byte 60,61 in selected page
0A42  0000    // NVM byte 62,63 in selected page
0A00  0000    // NVM write disable
p1
```

In this example, writing data to page 5

12.2.5 NVM map

In S5K33DX, NVM may be used to store, in addition to factory data, also module specific data such as:

- Chip ID data – production history data to be stored during die sorting
- Parameter for Lens shading

Except for NVM page #0, which is reserved for factory data, all other 127 pages may be used for storing the above optional data section in a user configured mapping.

For more details about the OTP structure, please refer to the official OTP map.

12.2.6 Write registers via OTP

FW configuration can be loaded online from host over I2C interface, or from a pre-burned data on NVM memory. When using the second option, data is written to NVM memory using data upload interface over I2C.

In some cases we will use NVM writing method in order to improve wake-up time (set file writing) by writing specific calibration to NVM.
After writing the data block, the start address (absolute) of the block needs to be written to *Set File & T&P table address* register in the NVM. The exact address can be found in the relevant *OTP map* document.

This NVM section supports two different extraction methods:

- Sequential block - hold a destination start address and a block of data to be copied into destination. Copy is done by bytes.
- None sequential - hold address and data to be copied into that address.

It is important to mention that the system distinguishes between 0x4000 to 0x2000 address space when reading the data and this should be considered when writing the data.
When reading word from 0x4000, big endian method is used and lower byte is placed at the significant byte.
When reading word from 0x2000, little endian method is used and higher byte is placed at the significant byte.

Example:

Relative address	Value
0x0100	03
0x0101	E8

Reading word from address 0x4000 0100, the value is 0x03E8
Reading word from address 0x2000 0100, the value is 0xE803

Guide:	None-sequential can start just after sequential. Sequential after none-sequential requires two zero bytes padding between the two blocks.
Guide:	Please make sure you are writing to the right location in the NVM. the location allocated to T&P / Set file can be found in the relevant OTP map document

In this section we can use several codes to improve the data usage and implement different kind of writing.

12.3 Sequential block

Byte index	Value	Description
0-1	0xFFFF	Code to indicate sequential block
2-3	Size	Size of the block
4-7	Address	First address
8	Data 0	First byte of data
9	Data 1	
.....		
8 + N - 1	Data N	Last byte of data

Sequential writing example – writing to 0x4000XXXX

OTP writing:

Code		Size		Address				Data							
FF	FF	00	08	40	00	03	44	00	04	00	04	10	73	0C	33

Memory output:

Address	Value before copy	Value after copy
0x40000343	00	00
0x40000344	00	00
0x40000345	00	04
0x40000346	00	00
0x40000347	00	04
0x40000348	00	10
0x40000349	00	73
0x4000034A	00	0C
0x4000034B	00	33
0x4000034C	00	00

```
WRITE #api_rw_data_transfer_page_select 0A //Page selecting (e.g. 0xA)
WRITE #api_rw_data_transfer_ctrl 03        //Write to NVM
p1000
WRITE #api_rw_data_transfer_data_0_ FF     //Sequential code (0xFF FF)
WRITE #api_rw_data_transfer_data_1_ FF
WRITE #api_rw_data_transfer_data_2_ 00     //Size of data (8 bytes)
WRITE #api_rw_data_transfer_data_3_ 08
WRITE #api_rw_data_transfer_data_4_ 40     //Address (0x40 00 03 44)
WRITE #api_rw_data_transfer_data_5_ 00
WRITE #api_rw_data_transfer_data_6_ 03
WRITE #api_rw_data_transfer_data_7_ 44
WRITE #api_rw_data_transfer_data_8_ 00     //Data
WRITE #api_rw_data_transfer_data_9_ 04
WRITE #api_rw_data_transfer_data_10_ 00
WRITE #api_rw_data_transfer_data_11_ 04
WRITE #api_rw_data_transfer_data_12_ 10
WRITE #api_rw_data_transfer_data_13_ 73
WRITE #api_rw_data_transfer_data_14_ 0C
WRITE #api_rw_data_transfer_data_15_ 33
p1000
WRITE #api_rw_data_transfer_ctrl 00        //Disable Write to NVM
```

Guide: You should clear (set to 0) the data registers which are not being used (16 to 63 in this example)

12.4 Non Sequential

None sequential block is composed of a sequence of Code and Data pairs ended by two zeroed bytes. The addresses are always considered as offset from the page select register.

Change mode codes

0x0000	End of sequence
0xFFFE	Set a global page select register with the value of the coming data. This register is used to generate a 32 bit address out of a 16 bit address. Will be followed by a 16 bit data. Page select register default value 0x4000.
0xFFFD	Change to 16 bit writing mode.
0xFFFC - Default	Change to 8 bit writing mode.
0xFFFB	Varying word size mode – support 8 and 16 bit writing.

The next two bytes after **0xFFFE** representing the page address.

12.4.1 8 bit writing mode – default configuration

Relative Address		Data
40	38	AA

Memory output:

Address	Value before copy	Value after copy
0x40004038	00	AA

12.4.2 16 bit writing mode

Change mode to 16 bit		Relative Address		Data	
FF	FD	40	38	AA	BB

Memory output:

Address	Value before copy	Value after copy
0x40004038	00	AA
0x40004039	00	BB

12.4.3 Varying word size mode

Relative address - one or two bytes writing selection

The most significant bit for 16 bit address will be used to indicate one or two bytes writing and therefore the address range will be consist of 15 bits hence from 1 to 0x7FFE (inclusive).

Address range	Description
[1-0x7FFA]	Bits 0-15 hold offset from the page select register. The generated address will be used to write **16** bit data and Will be followed by a 16 bit data.
[0x8001-0xFFFA]	Bits 0-14 hold offset from the page select register. The generated address will be used to write **8** bit data and

	will be followed by an 8 bit data.

End of sequence

End of none-sequential block will be ended by writing two zeroed bytes.

0x0000	**End of none-sequential**

Guide: In case it includes none sequential block before the ending bytes, the block four last bytes should be zeroed.

Guide: Sequential writing needs to be used in case the relative address is bigger than 0xFFFE

Guide: User can change mode by using one of the 'change mode' codes, default mode is one byte writing

None sequential writing example

OTP writing:

Change mode		Page Selection code		Page Value		Relative Address For Two bytes writing		Data		Relative Address For one byte writing (0x403B)		Data	End of sequence	
FF	FB	FF	FE	40	00	40	38	AA	BB	C0	3B	CC	00	00

Memory output:

Address	Value before copy	Value after copy
0x40004038	00	AA
0x40004039	00	BB
.....		
0x4000403B	00	CC
.....		

13 Appendix

13.1 Tx Laser Driver register setting example

This is a register setting example for Sony CXA4016 Tx Laser Driver.

Tx Laser wave control guide:

- Refer CXA4016GF Laser Driver Specification document

Laser Driver SPI register start address:

- Write to Laser Driver : 0x20001FC1 ~ 0x20001FD2

  The registers in CXA4016 are written just one time during loading a setfile. To change a CXA4016 register value, user should change the matched S5K33DXX register value in the setfile and then re-loading it.
- Read from Laser Driver : 0x20002881 ~ 0x200028CB

  The registers in CXA4016 are read just one time during loading a setfile and then the read values are written in matched S5K33DXX registers. These registers are for verifying that the initialization of Laser Driver is successful. If you need the temperature register value of Laser Driver which varies per frame, you can read it from embedded data.

Table 58 CXA4016 Write Control Registers

Register Address		Default Value
S5K33DXX (For Write Control)	CXA4016	
0x20001FC1	0x25	0x01
0x20001FC2	0x00	0x0C
0x20001FC3	0x01	0x00
0x20001FC4	0x02	0x00
0x20001FC5	0x03	0x00
0x20001FC6	0x04	0x00
0x20001FC7	0x05	0x00
0x20001FC8	0x06	0x00
0x20001FC9	0x07	0x00
0x20001FCA	0x08	0xFF

0x20001FCB	0x09	0x00
0x20001FCC	0x0A	0x04
0x20001FCD	0x0B	0x00
0x20001FCE	0x0C	0x28
0x20001FCF	0x0D	0x00
0x20001FD0	0x0E	0x77
0x20001FD1	0x0F	0xA0
0x20001FD2	0x10	0x00

Example for the location of CXA4061 write control values in a setfile:

```
s6F121325
s6F120001
s6F120203
s6F120405
s6F120607
s6F120809
s6F120A0B
s6F120C0D
s6F120E0F
s6F121013
s602A1FC0
s6F120001      // value for writing in CXA4016 register address 0x25
s6F120C00      // value for writing in CXA4016 register address 0x00 & 0x01
s6F120000      // value for writing in CXA4016 register address 0x02 & 0x03
s6F120000      // value for writing in CXA4016 register address 0x04 & 0x05
s6F120000      // value for writing in CXA4016 register address 0x06 & 0x07
s6F12FF00      // value for writing in CXA4016 register address 0x08 & 0x09
s6F120400      // value for writing in CXA4016 register address 0x0A & 0x0B
s6F122800      // value for writing in CXA4016 register address 0x0C & 0x0D
s6F1277A0      // value for writing in CXA4016 register address 0x0E & 0x0F
s6F120003      // value for writing in CXA4016 register address 0x10
```


Table 59 CXA4016 Read Control Registers

Register Address	
S5K33DXX (For Read Control)	**CXA4016**
0x20002881	0x00
0x20002883	0x01
0x20002885	0x02
0x20002887	0x03
0x20002889	0x04
0x2000288B	0x05
0x2000288D	0x06

0x2000288F	0x07
0x20002891	0x08
0x20002893	0x09
0x20002895	0x0A
0x20002897	0x0B
0x20002899	0x0C
0x2000289B	0x0D
0x2000289D	0x0E
0x2000289F	0x0F
0x200028A1	0x10
0x200028A3	0x11
0x200028A5	0x12
0x200028A7	0x13
0x200028A9	0x14
0x200028AB	0x15
0x200028AD	0x16
0x200028AF	0x17
0x200028B1	0x18
0x200028B3	0x19
0x200028B5	0x1A
0x200028B7	0x1B
0x200028B9	0x1C
0x200028BD	0x1E
0x200028BF	0x1F
0x200028C1	0x20
0x200028C3	0x21
0x200028C5	0x22
0x200028C7	0x23
0x200028C9	0x24
0x200028CB	0x25