ISP 8500 Firmware User Manual

Owner	
Name	Rohit Upadhyay
E-mail	rohit.upadhyay@st.com
Division	Imaging
Site	Greater Noida

Abstract

This document explains 8500 – Pictor firmware user manual

Revision History

Date	Version	Author	Reason/Changes
November 25, 2008	0.1	Rohit Upadhyay	First Release for firmware v 0.1.0
February 12, 2009	0.2	Atul Gupta	Updated for firmware release 0.3.0
February 26, 2009	0.3	Rohit Upadhyay	Updated for firmware release 0.4.0
March 17, 2009	0.4	Rohit Upadhyay	- Updated for firmware version 0.6.0 - Added details for ISP pipe opening and image data paths
April 27, 2009	0.5	Rohit Upadhyay	- Updated for firmware version 0.11.0 - Added details for Exposure and White Balance modules
June 26, 2009	0.6	Atul Gupta	- Update for firmware release 0.17.0 - Data path page elements - White balance constrainer - State machine
August 11, 2009	0.7	Rohit Upadhyay	- Update for firmware version 0.17.1 - Put in sections for Zoom
August 17, 2009	0.8	Rohit Upadhyay	- Update for firmware version 0.17.2 - Introduced new states in the firmware state machine
September 02, 2009	0.9	Rohit Upadhyay	- Updated for firmware version 0.20.0 - Introduced description for frame rate - Introduced significance of maximum frame rate
October 16, 2009	1.0	Manpreet Singh	- Updated for firmware version 0.22.0 - Added description of new page elements for the clock switch in Generic System Configuration.
October 16, 2009	1.1	Manpreet Singh	- Updated for firmware version 0.23.0 - Added an event which notifies host of any master i2c transaction failure. - Added a new state (RESET_ISP) in the host state machine. - Added a new event which notifies completion of RESET command to host.
October 28, 2009	1.2	Rohit Upadhyay	- Added the documentation about firmware support for dynamic change

			in pipe output image resolution.
October 29, 2009	1.3	Rohit Upadhyay	- Revamped the section on Device Streaming Operations. Introduced the concept of viewfinder, movie, direct and indirect stills.
November 5, 2009	1.4	Mamta Chalana	Updated for firmware release 0.24.0
November 10, 2009	1.5	Rohit Upadhyay	Added more details about the ISP_STOP_STREAMING, SENSOR_STOP_STREAMING, ISP_START_STREAMING and SENSOR_START_STREAMING notifications. Updated a few table formattings.
November 13, 2009	1.6	Rohit Upadhyay	Added details about the programming of `HostFrameConstraints.e_FrameDimensionProgMode` for VF, MOVIE, Single stage still grab and multi stage still grab
November 13, 2009	1.7	Mamta Chalana	Updated for firmware release 0.25.0
November 18, 2009	1.8	Rohit Upadhyay	Minor change in the document heading
January 07, 2010	1.9	Rohit Upadhyay	Added details about input interface selection and programming. Corrected bad table formattings.
January 11, 2010	2.0	Rohit Upadhyay	Pulled in documentation for Glace and Histogram IP.
January 20, 2010	2.1	Manpreet Singh	Added documentation of Events for Histogram, Glace and Exposure
February 8, 2010	2.2	Ashwani Chauhan	Added documentation for Auto Focus
February 10, 2010	2.3	Rohit Upadhyay	Added details about dynamic change in output image resolution Added the corresponding event notifications.
February 10, 2010	2.4	Ashwani Chauhan	Added Notification details for AUTOFOCUS_STATS_READY & FLADRIVER_LENS_STOP notifications.
February 23, 2010	2.5	Rohit Upadhyay	Included the Gamma module documentation
February 24, 2010	2.6	Rohit Upadhyay	Added event notification details for colour matrix for pipe0 and pipe1
February 25, 2010	2.7	Rohit Upadhyay	Updated for firmware version 0.32.0

March 4, 2010	2.8	Manpreet Singh	Added documentation to the SystemSetup page. Documentation of the metering of the system added.
March 16, 2010	2.9	Rohit Upadhyay	Added details about the host interface supported to perform single step exposure, white balance and frame rate.
March 17, 2010	3.0	Rohit Upadhyay	Refined details about single step programming of Exposure, White Balance and Frame Rate.
March 21, 2010	3.1	Rohit Upadhyay	Put in details about SDL_UPDATE_COMPLETE notification.
April 12, 2010	3.2	Manpreet Singh	Pulled in documentation of Adsoc, Babylon, Binning Repair, Duster, Gridiron, RSO & Scorpio modules.
May 21, 2010	3.3	Atul Gupta	Added documentation for Flash, brightness support, special effects, Aperture
June 22, 2010	3.4	Atul Gupta	Updated SDL documentation
July 2, 2010	3.5	Atul Gupta	Updated system config and system status page
July 5, 2010	3.6	Atul Gupta	Updated Flash Sensor Tuning and ND filter documentation.
July 7, 2010	3.7	Atul Gupta	Added Test pattern and memory mapping section
August 4, 2010	3.8	Partha Mitra	Updated flash and focus
August 13, 2010	3.9	Hem Dutt Dabral	Updated document for NVM, Frame rate, white balance, zoom, Grab Interface description, Adaptive over-scanning, and best sensor mode selection.
August 25, 2010	4.1	Hem Dutt Dabral	Correction in Sensor Tuning documentation regarding Sensor Tuning control page element
August 27, 2010	4.2	Hem Dutt Dabral	Corrections in the document
September 9, 2010	4.3	Hem Dutt Dabral	Updated documentation for FrameParamStatus, And added documentation for FrameParamStatus_Extn page
September 16, 2010	4.4	Atul Gupta	Documentation cleanup
October 1, 2010	4.5	Atul Gupta	Updated sensor tuning parameter

October 19, 2010	4.6	Atul Gupta	Updated PE for exposure compiler, exposure applied, frame param and frame param extension
November 1, 2010	4.7	Atul Gupta	export frame id along with AF stats and lens position/stability
November 30, 2010	4.8	Hem Dutt Dabral	Updated documentation corresponding to Flash in g_SystemSetup.e_Coin_Ctrl PE. Updated Gamma documentation.
December 24, 2010	4.9	Hem Dutt Dabral	Added docuementation for ISP FW versioning scheme and Boot Sequence. Updated Sensor Tuning documentation.
January 28, 2011	5.1	Hem Dutt Dabral	Updated Gridiron, Adsoc, Duster, Flash, Glace, Histogram, and Exposure, whitebalnce, framerate related documentation.
February 25, 2011	5.2	Hem Dutt Dabral	Updated Sensor output modes and Sensor mode selection related documentation. Valid for FW_4.0.0 series.
March 7, 2011	5.3	Atul Gupta	Updated documentation with reserved and debug PE's Added support for REC709 and Valid frame notification for OMX ISP FW manual should be used from ISP FW 5.0.0 series.
April 7, 2011	5.4	Atul Gupta	Documentation for dictionary trace and memory dump traces added ISP FW manual should be used from ISP FW 5.1.0 series onward.
May 02, 2011	5.5	Atul Gupta	Documentation update for Gridiron, Duster, FrameParamStatus_Extn, BML, BMS, CRM, Scalar, divider programming (u8_crm_clk_pip_in_div). ISP FW should be used from ISP FW 6.0.0 series onward
May 5, 2011	5.6	Atul Gupta	Updated FrameParamStatus_Af_ts in focus. When user request AF statistics, it will export focus parameter status in memory in same way as that of glace+histogram.
May 20, 2011	5.7	Hem Dutt Dabral	Updated documentation for Grab abort and Error handler. This ISP FW manual should be used with ISP FW 7.0.0 series.

June 1, 2011	5.8	Biswanath Goswami	Corrected BML parameter for Multi Stage Still Grab.
June 8, 2011	5.9	Hem Dutt Dabral	Added documentation for PE g_SystemSetup.e_Flag_ForceGrabOk
June 22, 2011	5.10	Atul Gupta	Added documentation for User restriction bit mask in run mode control and updated raw NVM data. This ISP FW manual should be used with ISP FW 7.2.0 series.
Aug 19, 2011	5.11	Hem Dutt Dabral Partha Mitra	Updated documentation for RGB2YUVCoder and Gridiron. Also added documentation for u8_NumOfFramesTobeSkipped and e_BMS_GrabMode_Ctrl. This ISP FW manual should be used with ISP FW 7.6.0 series.
Sep 7, 2011	5.12	Partha Mitra	Changed Name for Sensor Tuning
October 14, 2011	5.13	Hem Dutt Dabral	Added documentation for glace and histogram geometry mode selection. This ISP FW manual should be used with ISP FW 7.10.0 series.
Oct 27, 2011	5.14	Partha Mitra	Added documentation for Flash, SensorSettingsControl and PipeSettingsControl Interfaces. This version is corresponding to ISP FW 7.11.x and 7.12.x series.
November 11, 2011	5.15	Hem Dutt Dabral	Updated Gridiron documentation. Added documentation for Smia-Rx test mode. This ISP FW manual should be used with ISP FW 7.13.0 series.
November 17, 2011	5.16	Hem Dutt Dabral	Updated frame rate control documentation. This ISP FW manual should be used with ISP FW 7.14.0 series.
December 15,2011	5.17	Sudeep K S	Updated Generic System Configuration documentation This ISP FW manual should be used with ISP FW 7.20.0 series
December 23, 2011	5.18	Atul Gupta	Updated document for user restriction bitmask. The ISP FW manual should be used with FW 7.22.0 onwards.

TABLE OF CONTENTS

TABLE OF FIGURES

1. Overview

1.1 Objective

The purpose of this document is to specify the firmware interface and features exported by the ISP8500 firmware. It also explains firmware page element setup required for the features such as streaming, flash etc.

1.2 Referenced documents

Description	Version	Document Location
STXP70_Firmware_GuideLines_0_7.doc	0.7	https://codex.cro.st.com/plugins/docman/?group_id=1026&action=show&id=65523
Pictor Functional Specification	0.8	https://codex.cro.st.com/plugins/docman/?group_id=1041&action=show&id=30311

1.3 Naming convention used in document

For all the page elements, host should follow prefix naming convention to know the size of variable explained in table below.

Data Type	No of Bytes in firmware	Variable Prefix	Description
int8_t	1	s8_	signed short
uint8_t	1	u8_	unsigned short
int16_t	2	s16_	signed word
uint16_t	2	u16_	unsigned word
int32_t	4	s32_	unsigned DWord
uint32_t	4	u32_	signed DWord
int64_t	8	s64_	signed long
uint64_t	8	u64_	unsigned long
float_t	4	f_	IEEE float
bool_t	1	bo_	BOOL
enum	1	e_	Enum type. Can take only certain values

1.4 Glossary

Acronym	Description
AEC	Automatic Exposure Control

APE	Application Processor Engine
AWB	Automatic White Balance
CCI	Camera Control Interface
CCP2	Compact Camera Port 2
DMA	Direct Memory Access
DPCM	Differential Pulse Code Modulation
EMC	Electro Magnetic Compatibility
EMI	Electro Magnetic Interference
EOF	End of Frame
EXIF	Exchangeable Image File Format
FE	Frame End
Fps	Frames per second
FS	Frame Start
HWA	hardware Accelerator
I2C	Inter IC bus
IF	Interface
IO	Input/Output
ISP	Image Signal Processor
ITM	Interrupt Manager (hardware block)
LE	Line End
LS	Line Start
LSB	Least Significant Byte
LVDS	Low Voltage Differential Signaling
Mbps	Megabits per second
MIPI	Mobile Industry Processor Interface
MSB	Most Significant Byte
MSP	Manufacturer Specific Pixels
OC	Open Collector
OD	Open Drain

PCK	Pixel Clock
PCM	Pulse Code Modulation
PLL	Phase locked loop
QXGA	Quantum Extended Graphics Array (2048x1536)
RO	Read Only
RW	Read/Write
SCK	System Clock
SMIA	Standard Mobile Imaging Architecture
SOF	Start of Frame
SubLVDS	Sub-Low Voltage Differential Signaling
SVGA	Super Video Graphics Array (800x600)
SXGA	Super Extended Graphics Array (1280x1024)
SXVGA	Super Extended Video Graphics Array (1280x960)
UXGA	Ultra Extended Graphics Array (1600x1200)
VGA	Video Graphics Array (640x480)
VT	Video Timing
WOI	Window of Interest

2. ISP FW Versioning

2.1 FW Versioning

2.1.1 Overview

ISP firmware versioning format is FW_X.Y.Z_SENSOR_A_LLA_B.C_LLCD_D.E.

2.1.1.1 FW_X.Y.Z

FW_X.Y.Z signifies version of FW part used in release.

- X: Incremented when there is a compatibility break.
- Y: Incremented when feature addition has been done without any break in compatibility. It is reset to 0 if X is incremented.
- Z: Incremented when only bug fixes have been done in a release. It is reset to 0 if Y is incremented

2.1.1.2 SENSOR_A

'A' signifies sensor information. By this number, we can deduce which sensor is supported for this release. For different sensors, 'A' has different value. Value of 'A' for a given sensor will be communicated separately to ISP FW customer.

2.1.1.3 LLA_B.C

LLA_B.C signifies **Low Level API** version used in a given release.

- B: Incremented when there is a compatibility break in API and ISP FW update is necessary.
- C: Incremented when there is a change in API or feature addition that is backward compatible.

2.1.1.4 LLCD_D.E

LLCD_D.E signifies version of **Low Level Camera Driver** used in a given release. Significance of D and E are explained below:-

- D: Incremented with every release.
- E: Incremented each time when there is bug fix on old API.

2.1.2 Pages Exposed

DeviceParameters

DeviceParameters		
Page Elements	Description	Range
u32_DeviceId	Specifies the ID of the device [DEFAULT]: 0x8500	0 - 4294967295
u32_FirmwareVersionMajor	Firmware major version number [DEFAULT]: 0x2	0 - 4294967295
u32_FirmwareVersionMinor	Firmware minor version number [DEFAULT]: 0x0	0 - 4294967295
u32_FirmwareVersionMicro	Firmware micro version number [DEFAULT]: 0x0	0 - 4294967295
u32_LLA_Sensor	Sensor model to which low level api belong [DEFAULT]: 0x1	0 - 4294967295

u32_LLA_MajorVersion	Major release version of low level api integrated fw [DEFAULT]: 0x2	0 - 4294967295
u32_LLA_MinorVersion	Minor release version of low level api integrated fw [DEFAULT]: 0x0	0 - 4294967295
u32_LLCD_MajorVersion	Major release version of low level api integrated fw [DEFAULT]: 0x4	0 - 4294967295
u32_LLCD_MinorVersion	Minor release version of low level api integrated fw [DEFAULT]: 0x0	0 - 4294967295
e_SiliconVersion	Version of the Silicon. 1 for V1, 2 for V2. [DEFAULT]: SiliconVersion_e_8500v1	SiliconVersion_e_8500v1, SiliconVersion_e_8500v2

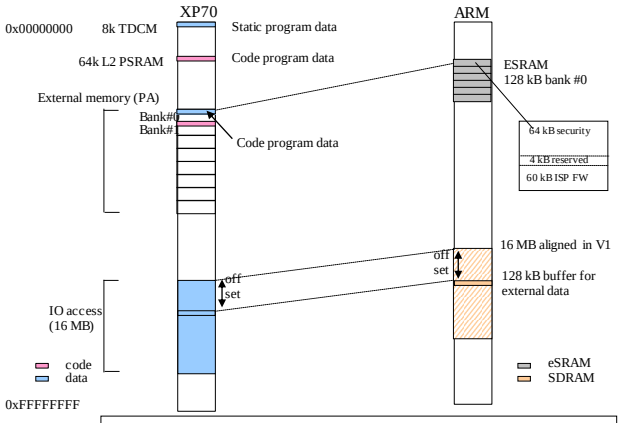
3. Memory mapping

Pictor ISP FW for 8500 runs on STxP70 in SIA. It use 4 different memory for program and data. The four different memory regions are:

1. IDM – 8 KB: This is referred as tightly coupled data memory. From xP70, the memory is mapped from 0x00000000 address. ISP FW uses this memory to store frequenty used data structures.
2. IPM – 64 KB: This is referred as level 2 program memory. ISP FW uses this memory for frequently accessed instructions.
3. eSRAM: The buffer is provided by host and is expected to be allocated in eSRAM. HOST allocate 64 KB buffer and map it to PA0 space. Please note that ISP FW generate only 60KB binary, so host should download this section leaving gap of 4KB. Please refer attached slides.
4. SDRAM: The buffer is provided by host and is expected to be allocated in SDRAM. Host allocate 64KB buffer and map it to PA1 space. ISP FW re-direct less frequenty excecuted program in this memory.

Name	Size	Address from xP70
IDM – Tightly coupled data memory	8 KB	0x00000000
IPM – L2 program memory	64KB	0x00400000
eSRAM	60 KB	0x10001000
SDRAM	64 KB	0x20000000

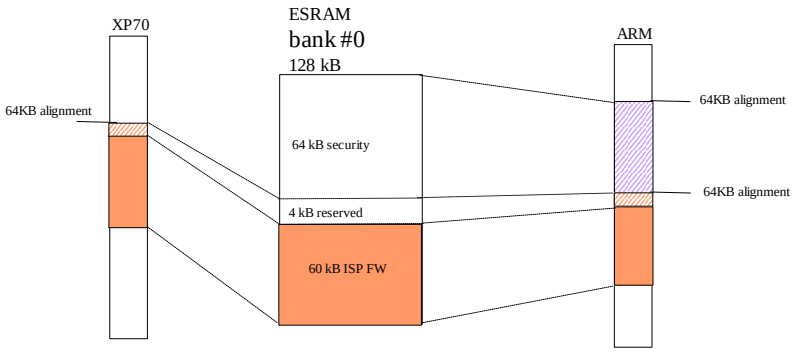
XP70 – ARM Memory Map



Code:
– 64 kB L2 PSRAM (internal program memory)
–eSRAM pipeline access memory (60 kB put in ESRAM bank #0 memory, external program memory)

Data:
– 8 kB TCDM (internal data memory) PA Bank0 (eSRAM)
– non pipelined external IO space (16MB) used for external memory. Can be remapped where ever in the 4GB system memory space (eSRAM or SDRAM). On V1 there are 2x16 MB IO spaces. On V2 there will be more.

ESRAM allocation



- XP70 can remap ARM memory
  - Minimum size is 64Kb
  - Minimum alignment is 64Kb
- ISP FW zone in ESRAM is not 64KB aligned !

ESRAM remapping

- ARM rounds address of ISP ESRAM zone to 64Ko
  - Programs ISP_MCU_SYS_ADDR0 accordingly
  - ISP_MCU_SYS_SIZE0 set to 64Kb
- XP70 remaps 64Kb
  - 4KB reserved
  - 60KB of ISP FW code
- XP70 must not access first 4KB remapped
  - This area contains code and XP70 can't modify it
  - **Must be taken into account by linker scripts (scatter loader ?)**

IO REGION

- The ISP FW will put shared data between the SIA and the ARM in IO area.
  - In V1 the buffer must be 16 MB aligned.
  - In V2 8 zones will be available.
- ARM allocates 128 kB SDRAM (Size to be refined)
  - The idea is to allocate without taking into account the alignment constraint.
- ARM retrieves the physical address of this buffer
  - Address is rounded to 16MB
    - program ISP_MCU_IO_ADDR0
  - Offset of this physical address to the 16MB aligned address
    - Sent to ISP FW through a PageElement
- XP70 must not modify SDRAM outside of these 128KB
    - Hardware does not prevent it.

4 CONFIDENTIAL ST ERICSSON

4. Host Interface

4.1 Host Comms

4.1.1 Overview

The host comms interface is used by the host to access the device firmware page elements. The current host comms implementation uses a mailbox mechanism to facilitate the read and write of page elements.

4.1.2 Concept of Page Elements

The external host communicates with the device through an interface called as page-elements. All firmware addresses are (internally within the firmware) clubbed into various groups based on the kind and usage of the address. Each of this group is called a “page” and each individual register within this page is called an “element” e.g. `ExposureControls` is called a page and the register `bMetering` an element of this page. A page may have certain properties (e.g. Read Only). In this case, all the elements within this page share the same property.

1. Read Only Page: Any attempt by the host to perform a write on any element within this page is not carried out.
2. Mode Static Page: Any change to the device functioning associated to a change in a mode static page is absorbed only when the device transitions from IDLE state to a streaming state.

4.1.3 Address Encoding

The device firmware page element has a 16 bit address range. This 16 bit address range is encoded as shown in Figure 1. The figure shows that the bit 15 is reserved, thereby restricting address for the Imaging Device to 15 bits. However, this bit can be used to extend the device address space for accessing non-Imaging Pages.

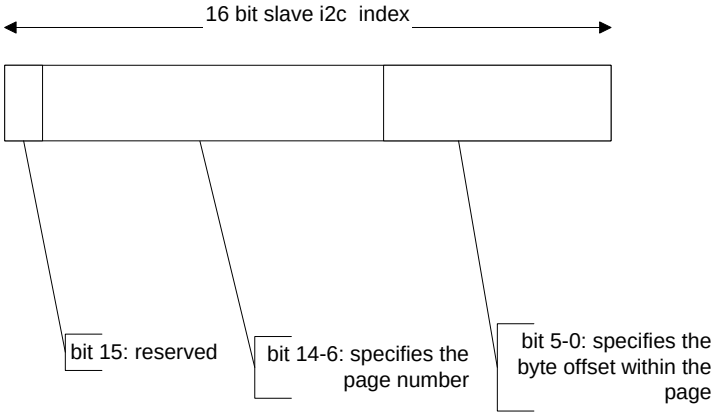


Figure 1 Address Encoding

4.1.4 Usage

4.1.4.1 Pre-Requisites

The host must know the following to perform a read/write operation on a device page element:

1. Address of the element to be read or written.
2. Number of bytes of to be read or written. This is normally the byte size of an element.
3. Address of USER_IF_PAGE_ELEMENT mailbox register. The host must specify the base address of the element to be read or written into this register.
4. Address of USER_IF_OPC.opcode mailbox register. The host must specify the opcode of the read or write operation into this register. A read opcode is denoted by 0 and a write opcode is denoted by 1.
5. Address of USER_IF_OPC.transaction_id and USER_IF_ACK. transaction_id_completed mailbox register. These two registers together are used to trigger the read or write operation. Once the host has programmed all the relevant registers for the operation, it must program transaction_id register to be different from transaction_id_completed. At the completion of the read or write operation by the device, the transaction_id_completed register will have a value equal to transaction_id.
6. Address of USER_IF_WR_DATA mailbox register. The host must specify the data to be written on to the page element in case of a write operation.
7. Address of USER_IF_RD_DATA mailbox register. This register contains the value of the element after the completion of a read operation.
8. The host must have a notion of Host Comms event notifications. These notifications are generally in the form of interrupts into the host processor. The host comms generates notifications (if enabled by the host) for the following two events:
    a. HOST_COMMS_READY: To signify that the device host comms is ready. Will be raised after device power on.
    b. HOST_COMMS_OPERATION_COMPLETE: To signify that the last host comms operation has completed.

4.1.4.2 Pre Boot Parameters

None

4.1.4.3 Pre Run Parameters

None

4.1.4.4 Live Parameters

None

4.1.4.5 Status Parameters

The following status parameters are exposed:

1. e_HostComms_Status: Specifies the status of the last read/write operation.
2. bo_ModeStaticSetupChanged: Set to TRUE is a mode static page is written onto. Is automatically cleared once the mode static changes have taken effect.

4.1.4.6 Write Operation

Figure 2 depicts the host end operations that are required to perform a write operation on a page element.

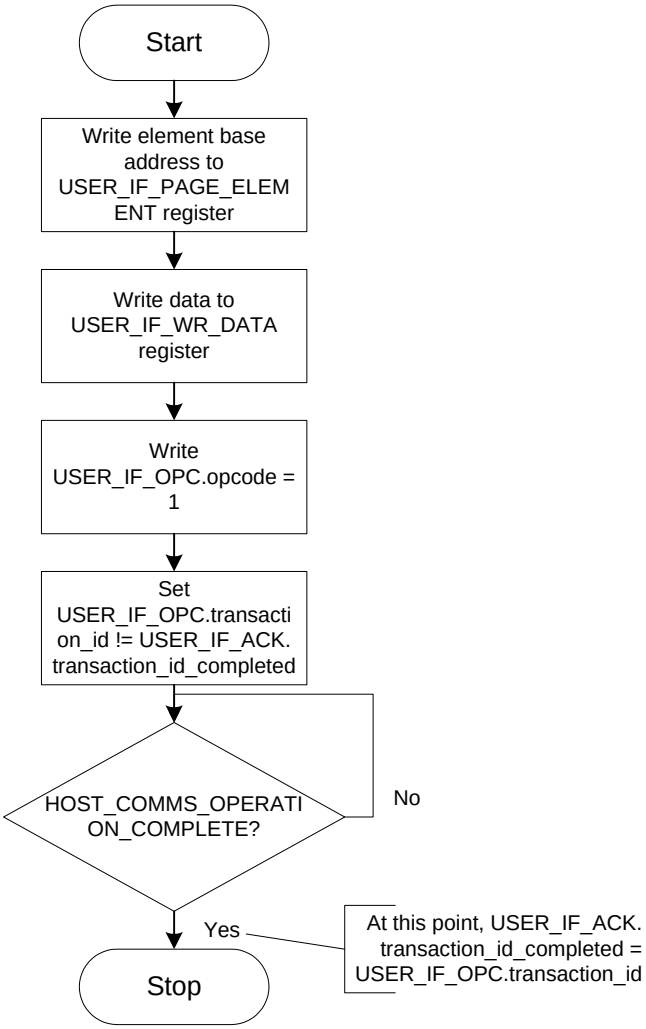


Figure 2 Host Comms Write Operation

4.1.4.7 Read Operation

Figure 3 depicts the host end operations that are required to perform a write operation on a page element.

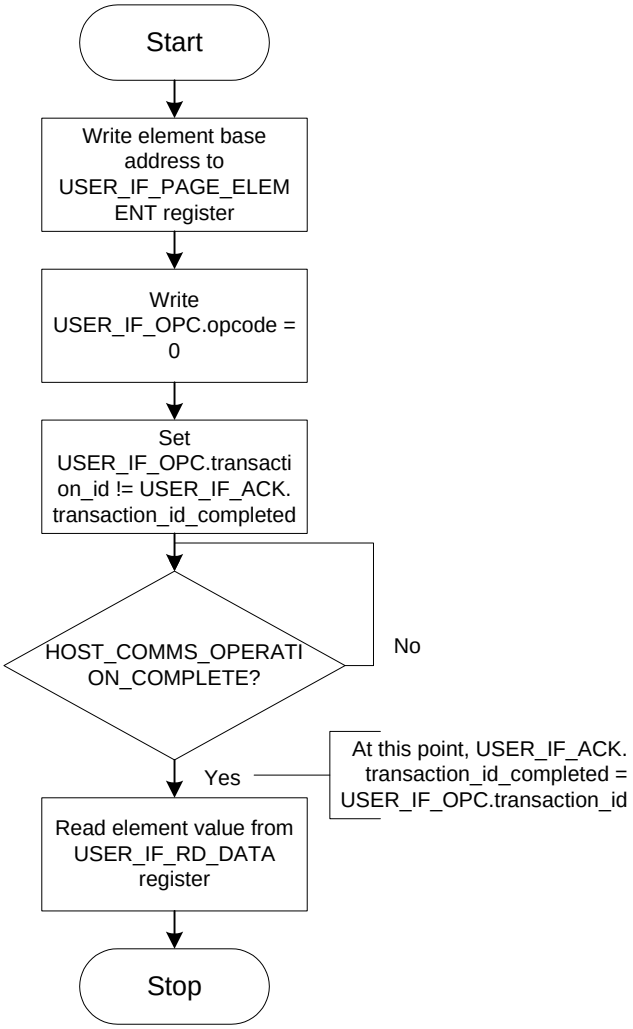


Figure 3 Host Comms Read Operation

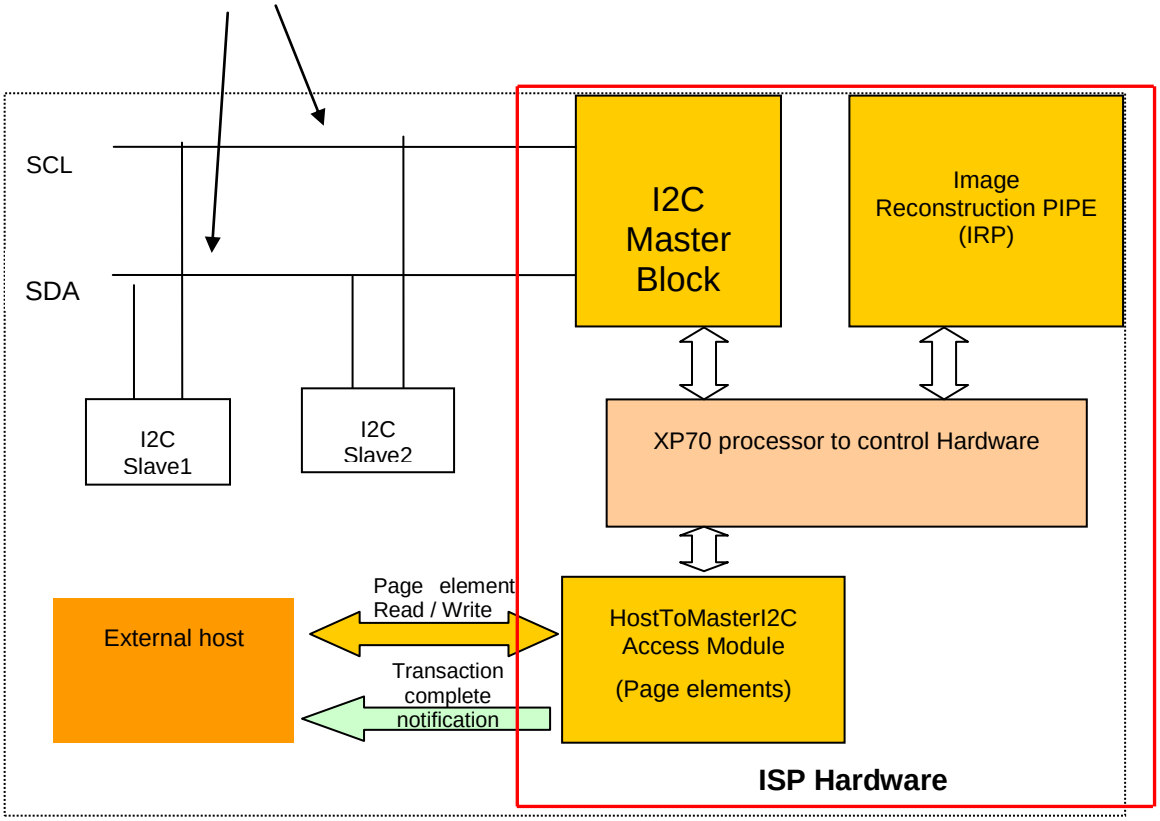
4.2 Host to Master I2C access

4.2.1 Host to Master I2C access overview

ISP communicates to external devices through I2C or CCI bus, e.g. Sensor, Flash etc. ISP acts as Master on I2C bus and the devices act as slave. I2C Master is hardware IP inside the ISP block. There is no way for Host to communicate with I2C slave, so ISP firmware provide interface to host to facilitate communication, this interface is called as HostToMasterI2C_Access.

- Description of ISP hardware, Host and connected I2C devices:
  1. IRP (Image reconstruction pipe)
  2. STxP70: Microcontroller or DSP to control IRP and other hardware blocks
  3. MasterI2C: Master I2C IP manages communication with I2C compliant salves on I2C bus.

2 Wire I2C bus



- Pre-requisites and recommendations for Using HostToMasterI2CAccess
  - It is suggested that the HostToMasterI2CAccess module should only be used when the device is in IDLE state. In the IDLE state, the HostToMasterI2CAccess module is the sole contender for the MasterI2c channel. It is hence guaranteed that the HostToMasterI2CAccess module would be granted the master i2c communication channel.
  - The HostToMasterI2CAccess module cannot be used in the SLEEP state of the device.
  - It is strongly suggested that the HostToMasterI2CAccess module should not be used in any active state of the device (i.e. while the device is streaming either finitely or infinitely). There are firmware modules that may be performing transactions with the MasterI2C. Due to the nature of i2c communication, only one transaction could be done at any given time. Hence there is a possibility that the HostToMasterI2CAccess module is not granted the i2c communication channel. In this case, the intended HostToMasterI2CAccess transaction will not be carried out.
  - If the host has to perform a read or write onto the MasterI2C, it must specify the MasterI2C (Sensor0 or Sensor1) onto which the operation has to be performed. However care has to be taken that if this operation has to be performed while the device is streaming, then no attempt must be made to perform a read/write operation from the MasterI2C which is currently not active.
  - Module does not support more than 16 bytes Read / Write

4.2.2 Host to Master I2C access usage

The module can be used to perform reads and writes to any registers of any device mounted on the master i2c bus.

4.2.2.1 Pre-requisites

Host should know following parameters to access the module:

1. Device address of I2C slave
2. Device endianess

4.2.2.2 Pre BOOT parameters

None

4.2.2.3 Pre RUN parameters

None

4.2.2.4 Live parameters

None

4.2.3 Features supported by Host to Master I2C access module

Read: Host can read from 1 byte to N bytes from I2C slave. Following types of read command are available in Host to master i2c access:

- READ N BYTES (8*N) read: Maximum 16 bytes read from sequentially addressed registers is possible

Write: Host can write from 1 byte to N bytes on to I2C slave. Following types of write command are available in Host to master i2c access.

- WRITE N BYTES (8*N) write. Maximum 16 byte write at registers are possible. The module will write these N bytes on sequentially addressed registers

4.2.4 Coin Mechanism to use Host to Master I2C access

The device uses a command-coin and status-coin mechanism to process commands from the host. Whenever ISP finds that command–coin e_Coin_Command element in the HostToMasterI2CAccessControl page is different from status-coin e_Coin_Status element of the HostToMasterI2CAccessStatus, it processes the command programmed in e_HostToMasterI2CRequest_Request element of the HostToMasterI2CAccessStatus page. After completion of the command, ISP sets:

HostToSensorAccessControl->e_Coin_Command= HostToSensorAccessStatus->e_Coin_Status

In case of error, it also updates the error status page elements.

Host can check transaction by two ways:

1. Polling till status coin becomes equal to command coin.
2. Wait for Master I2C host request complete event notification. Waiting on notification is better as it save polling time and unnecessary page element reads.

4.2.5 HostToMasterI2CAccess Read Operation

Read N Bytes

Step1: check if Control coin is same as that of status coin. Different coins means HostToMasterI2c Access is already busy with transaction. If they are equal, go to step1:

HostToMasterI2CAccessStatus.e_Coin_Status should be equal to HostToMasterI2CAccessControl.e_Coin_Command

Step2: Program Device ID (I2C Slave Address)

HostToMasterI2CAccessControl.u16_DeviceID = u16_DeviceID

Step3: Program index register(register address in the I2C slave)

HostToMasterI2CAccessControl.u16_Index = u16_Index

Step4: Program requested command: Read –NBytes

HostToMasterI2CAccessControl.e HostToMasterI2CRequest Request = e_HostToMasterI2C_ReadNBytes

Step5: Program Number of byte (1 - 16) to read from I2C Slave

HostToMasterI2CAccessControl.u8_NoBytesReadWrite = NoOfBytes

Step6: Toggle the coin, i.e. Change the value command coin, If it was head, and change it to tails or vice versa.

```
if (e_CommandCoin ==‘Coin_e_Heads’):
    e_CommandCoin = ‘Coin_e_Tails’
elif (e_CommandCoin == ‘Coin_e_Tails’):
    e_CommandCoin = ‘Coin_e_Heads’
self.device.setVariableValue('HostToMasterI2CAccessControl.e_Coin_Command',e_CommandCoin)
```

Step7: Wait for command coin and status coin to become equal or transaction Complete notification

```
while (e_CommandCoin != e_StatusCoin): WAIT
```

Step8: Read HostToMasterI2CAccessData.u8_arrData[0] … [15] based on no of bytes programmed

U8_Data[0] = HostToMasterI2CAccessData.u8_arrData[0]

U8_Data[1] = HostToMasterI2CAccessData.u8_arrData[1]

.
.

.

U8_Data[N] = HostToMasterI2CAccessData.u8_arrData[N]

Step9: Finish

4.2.6 HostToMasterI2CAccess Write Operation

Write N Bytes

Step1: check if Control coin is same as that of status coin. Different coins means HostToMasterI2c Access is already busy with transaction. If they are equal, go to step1:

HostToMasterI2CAccessStatus.e_Coin_Status should be equal to HostToMasterI2CAccessControl.e_Coin_Command

Step2: Program Device ID (I2C Slave Address)

HostToMasterI2CAccessControl.u16_DeviceID = u16_DeviceID

Step3: Program index register(register address in the I2C slave)

HostToMasterI2CAccessControl.u16_Index = u16_Index

Step4: Program requested command: Read –NBytes

HostToMasterI2CAccessControl.e HostToMasterI2CRequest Request = e_HostToMasterI2C_ReadNBytes

Step5: Program Number of byte (1 - 16) to read from I2C Slave

HostToMasterI2CAccessControl.u8_NoBytesReadWrite = NoOfBytes

Step6: Program HostToMasterI2CAccessData.u8_arrData[0] … [15] based on no of bytes to write on slave

HostToMasterI2CAccessData.u8_arrData[0] = U8_Data[0]

HostToMasterI2CAccessData.u8_arrData[1] = U8_Data[1]

.
.

.

HostToMasterI2CAccessData.u8_arrData[N] = U8_Data[N]

Step7: Toggle the coin, i.e. Change the value command coin, If it was head, and change it to tails or vice versa.

```
if (e_CommandCoin ==‘Coin_e_Heads’):
    e_CommandCoin = ‘Coin_e_Tails’
elif (e_CommandCoin == ‘Coin_e_Tails’):
    e_CommandCoin = ‘Coin_e_Heads’
```

self.device.setVariableValue('HostToMasterI2CAccessControl.e_Coin_Command',e_CommandCoin)

Step8: Wait for command coin and status coin to become equal or transaction Complete notification

```
while (e_CommandCoin != e_StatusCoin): WAIT
```

Step9: Finish

4.2.7 HostToMasterI2C_Access page elements

4.2.7.1 HostToMasterI2CAccessControl

HostToMasterI2CAccessControl	Control Structure for Host To MasterI2C Access	
Page Elements	**Description**	**Range**
u16_DeviceID	I2C Slave Device ID NOTE: Only 7 bit device is supported, 10 Bit slave address is not supported	
u16_Index	Index register in I2C slave to which host would like to read or write. NOTE: Currently 16 bit index supported, 8 bit index is not supported	
e_Coin_Command	Program CommandCoin != StatusCoin, After Read/Write access command complete, CommandCoin will become equal to StatusCoin	Coin_e_Heads(0) Coin_e_Tails(1)
e_HostToMasterI2CRequest_Request	The type of Read/Write request to connected I2C slave	HostToMasterI2CRequest_e_HostToMasterI2C_ReadNBytes(0) HostToMasterI2CRequest_e_HostToMasterI2C_WriteNBytes(1)
u8_NoBytesReadWrite	Number of bytes to be Read/Write on slave NOTE: Only valid if e_HostToMasterI2CRequest_Request set to e_HostToMasterI2C_ReadNBytes (Read) or e_HostToMasterI2C_WriteNBytes (Write)	N can take any value between 1 and 16
e_DeviceAddress_Type	10 bit i2c slave address 7 bit i2c slave address	DeviceAddress_e_10BitDeviceAddress(0), DeviceAddress_e_7BitDeviceAddress(1)
e_DeviceIndex_Type	16-bits data index 8-bits data index	DeviceIndex_e_16BitDataIndex(0) DeviceIndex_e_8BitDataIndex(1)

4.2.7.2 HostToMasterI2CAccessData

HostToMasterI2CAccessData	Data Read / Write Structure	
Page Elements	**Description**	**Range**
u8_arrData[16]	Read / Write data on slave when e_HostToMasterI2CRequest_Request set to e_HostToMasterI2C_ReadNBytes (Read) or e_HostToMasterI2C_WriteNBytes (Write)	

4.2.7.3 HostToMasterI2CAccessStatus

HostToMasterI2CAccessStatus	Status Structure	
Page Elements	**Description**	**Range**
e_Coin_Status	When e_Coin_Command != e_Coin_Status, an attempt is made to perform the required read/write operation When command is completed, e_Coin_Status is set to e_Coin_Command	Coin_e_Heads(0) Coin_e_Tails(1)
e_Result_result	I2C operation result	Result_e_failure(0) Result_e_success(1)
u8_HostToMasterI2CAccessErrorCount	Counter to keep track on no of errors occurred on MasterI2C	

4.2.8 Default settings recommendation

None

4.3 Host Interface Manager

4.3.1 Overview

This module provides an interface to the host through which the device is driven. All commands to the device to start streaming, stop streaming, boot up etc. will be through this module.

4.3.2 Device State Machine

The device state machine has been structured into few high level states. Figure 4 depicts the high level state machine of the device and various state transitions based on user commands.

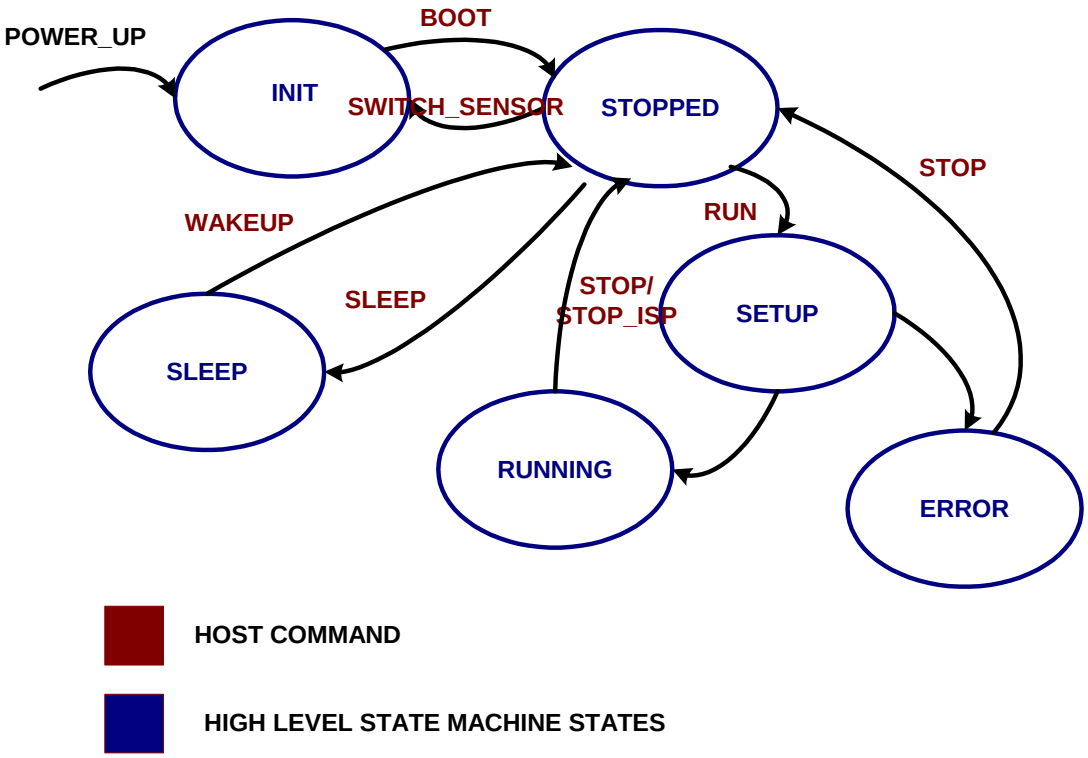


Figure 4 High Level State Machine

Each of the high level states is further composed of low level states. At certain low level state transitions, appropriate notifications are issued by the device.

Detailed description of the high level states with respect to their corresponding low level state transitions and their corresponding device notifications if any are shown in the following figures.

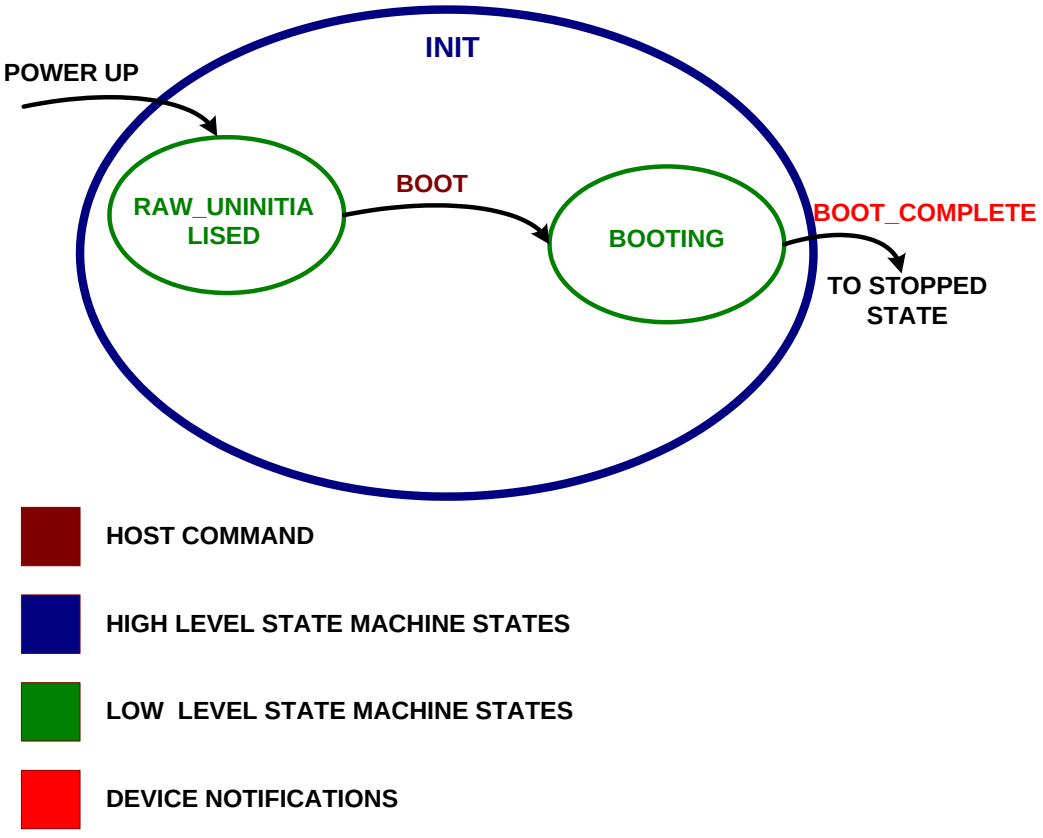


Figure 5 High Level State INIT

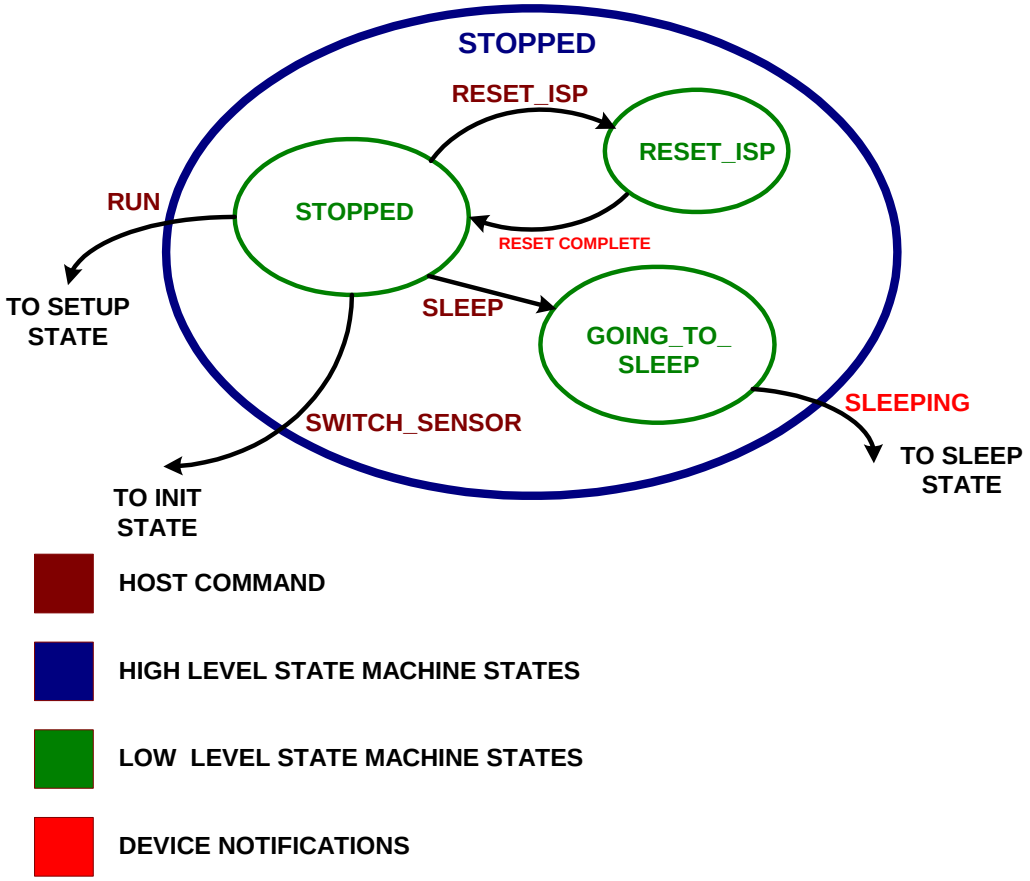


Figure 6 High Level State STOPPED

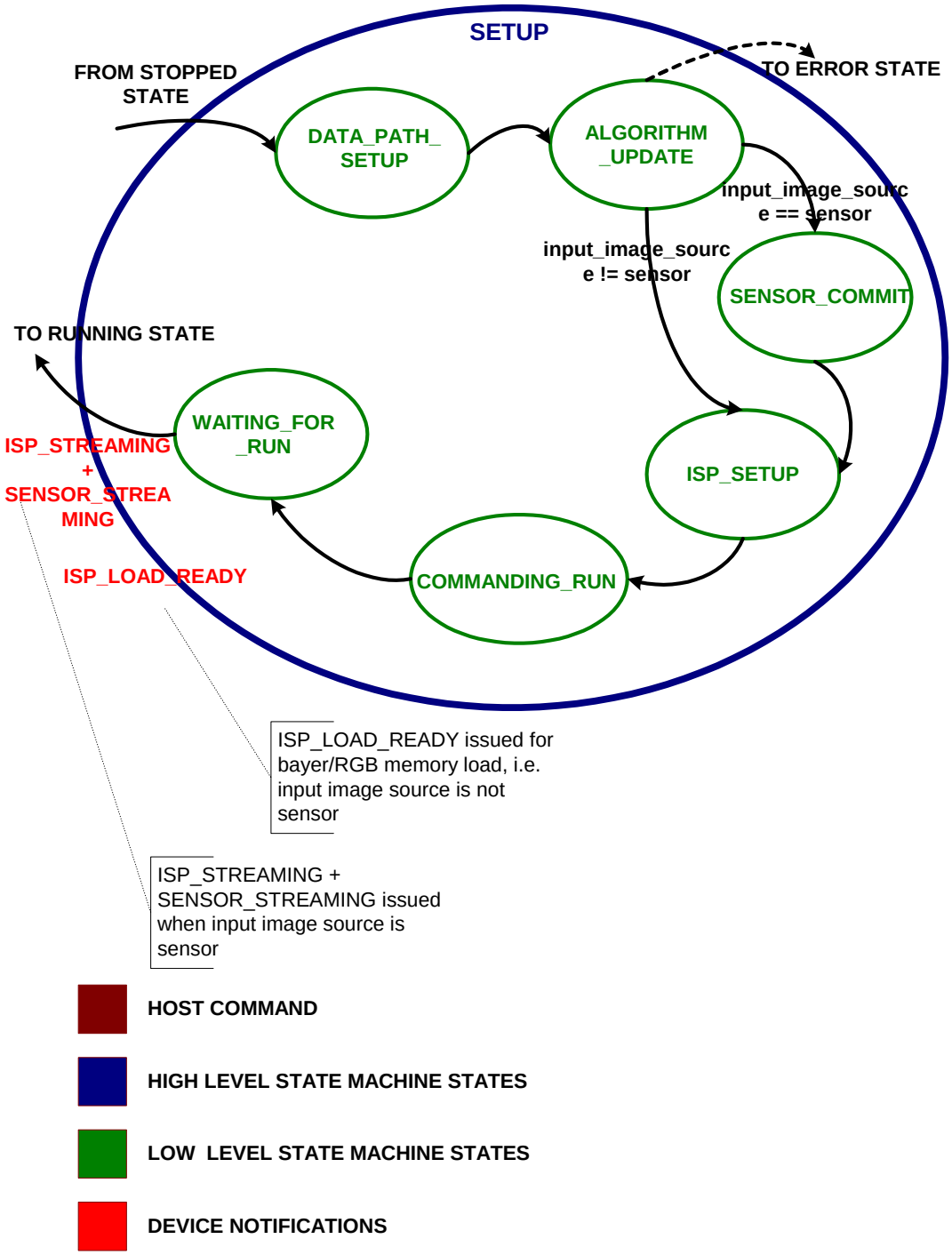


Figure 7 High Level State SETUP

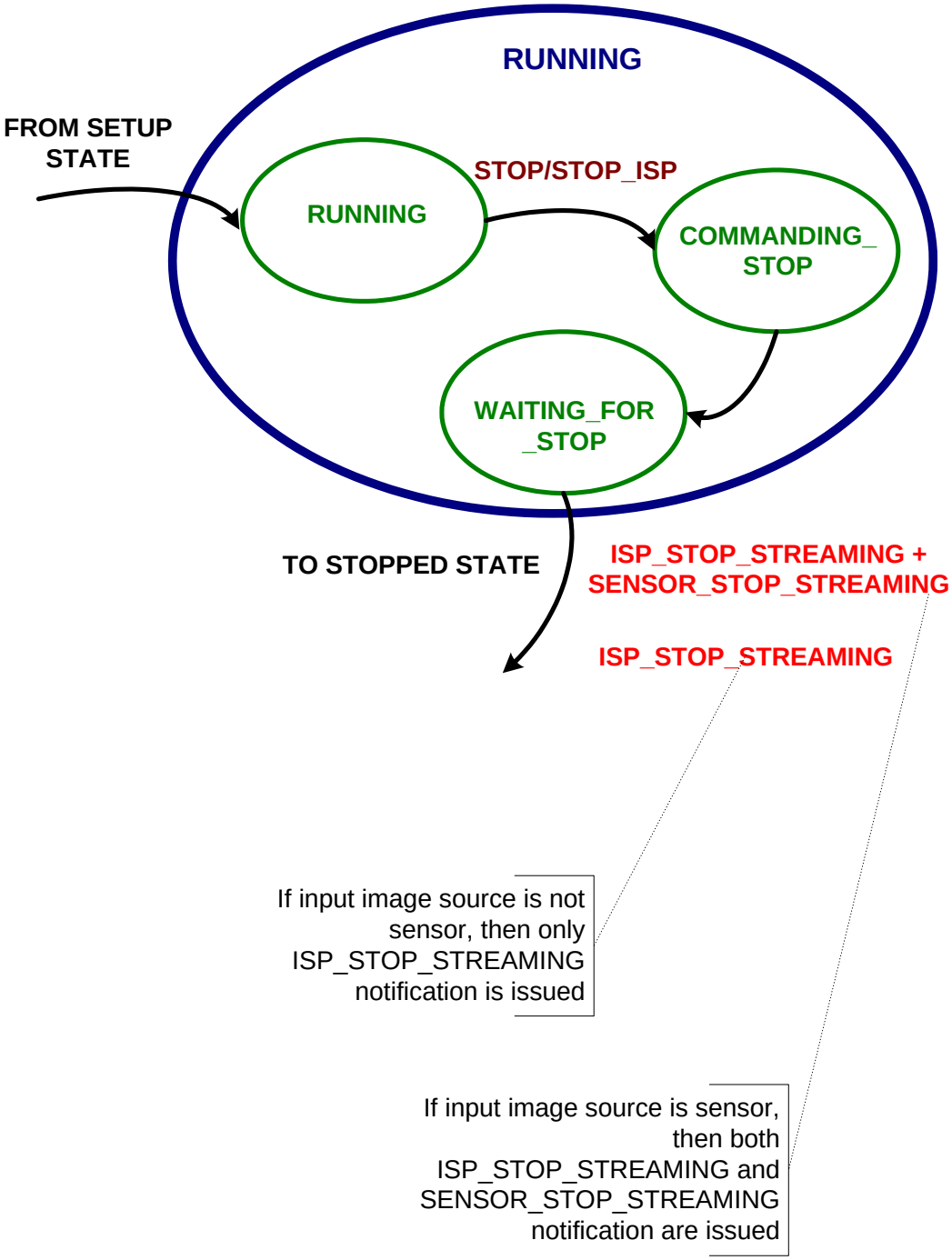


Figure 8 High Level State RUNNING

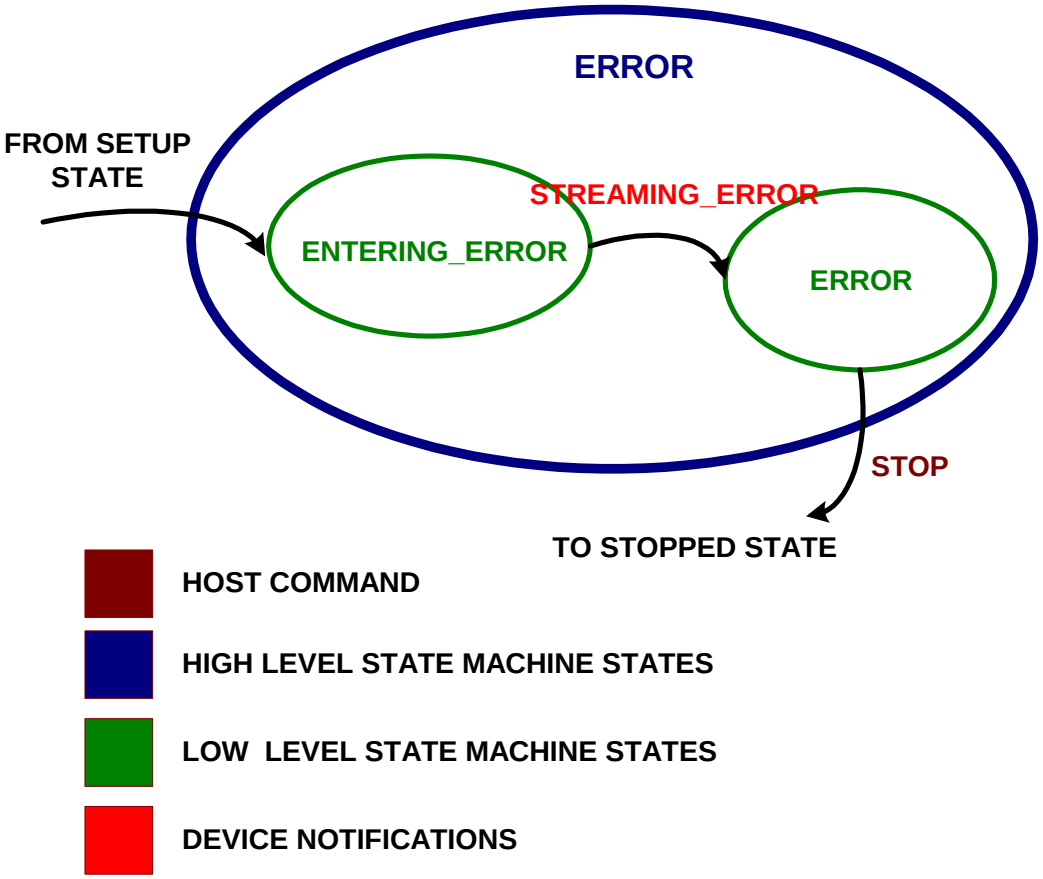


Figure 9 High Level State ERROR

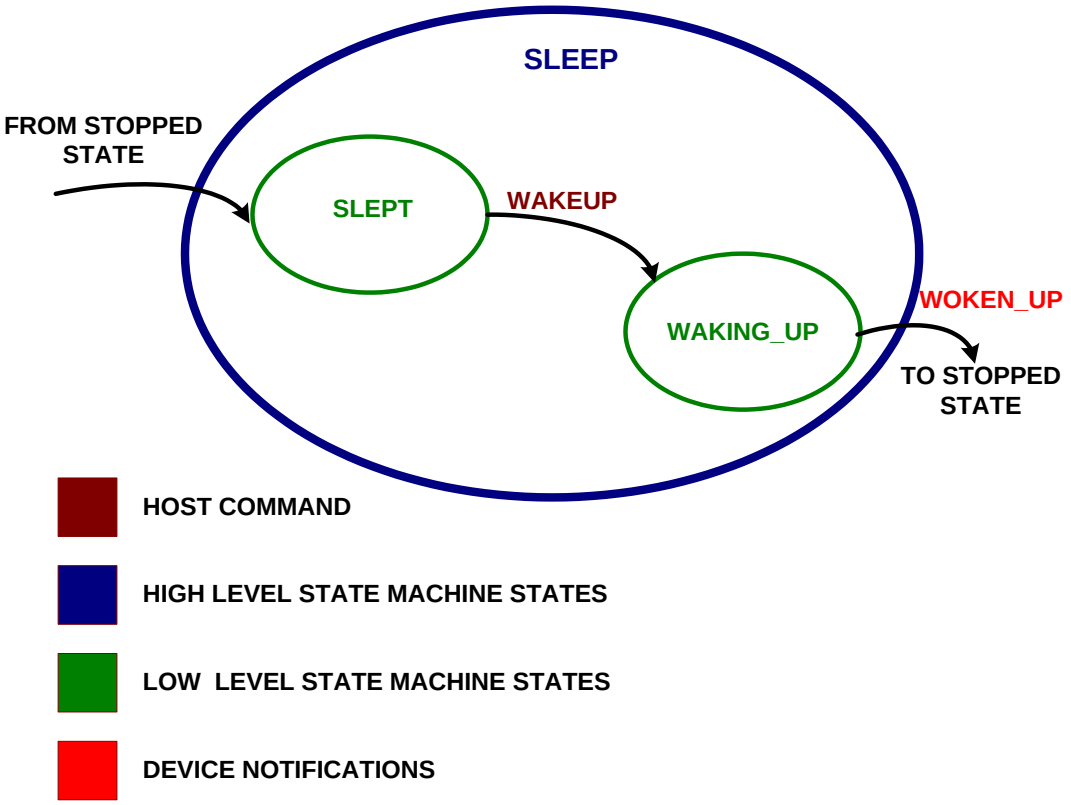


Figure 10 High Level State SLEEP

It is evident from the above figures, the device presents a set of user commands that are used to drive the device into its desired state. A description of the high level states is given in Table 1.

High Level State	Description
HostInterfaceHighLevelState_e_INIT	The default state of the device after power up. After issuing HOST_COMMS_READY notification, the device is in this state.
HostInterfaceHighLevelState_e_STOPPED	In this state, the device is not streaming.
HostInterfaceHighLevelState_e_SETUP	This is an intermediate state. The device performs all algorithm, sensor, data path and ISP setup in this state
HostInterfaceHighLevelState_e_RUNNING	In this state, the device is streaming data as per the setup done in SETUP state.
HostInterfaceHighLevelState_e_SLEPT	This is the low power state. All the clocks are powered down and the sensor XSHUTDOWN are pulled low.
HostInterfaceHighLevelState_e_ERROR	The state machine reaches this state when there has been an error in starting to stream.

Table 1 High Level State Description

A description of the user commands is given in Table 2.

User Commands	Description
HostInterfaceCommand_e_BOOT	Command to BOOT the device.
HostInterfaceCommand_e_RUN	Command to start streaming from the device.
HostInterfaceCommand_e_STOP	Command to stop both the sensor and the ISP.
HostInterfaceCommand_e_STOP_ISP	Command to stop only the ISP. The sensor continues to stream after the ISP stops.
HostInterfaceCommand_e_SLEEP	Command to put the device into low power SLEEP state.
HostInterfaceCommand_e_WAKEUP	Command to pull the device out of SLEEP.
HostInterfaceCommand_e_SWITCH_SENSOR	Command to select a sensor different from the currently selected sensor.
HostInterfaceCommand_e_RESET_ISP	Command to reset all the IPs in the Pipe.

Table 2 User Commands Description

Once the device has processed any of the user commands, it generates an appropriate event notification and transitions to an appropriate state. It must be noted that the device may generate different set of notifications for the same command depending on input image source for the operation. The event notifications generated in context of user commands are specified in Table 3.

Input Image Source	Command	Event Notification Generated
Sensor0/ Sensor1	HostInterfaceCommand_e_RUN	ISP_STREAMING, SENSOR_STREAMING
Sensor0/ Sensor1	HostInterfaceCommand_e_STOP	ISP_STOP_STREAMING, SENSOR_STOP_STREAMING
Sensor0/ Sensor1	HostInterfaceCommand_e_STOP_ISP	ISP_STOP_STREAMING
Bayer Load 1/ Bayer Load 2/ RGB Load	HostInterfaceCommand_e_RUN	ISP_LOAD_READY
Bayer Load 1/ Bayer Load 2/ RGB Load	HostInterfaceCommand_e_STOP/ HostInterfaceCommand_e_STOP_ISP	ISP_STOP_STREAMING
Any	HostInterfaceCommand_e_RESET_ISP	ISP_RESET_COMPLETE

Table 3 User Command Event Notifications

4.3.3 Usage

4.3.3.1 Pre-Requisites

The host must be capable of reading writing to the HostInterface_Control. e_HostInterfaceCommand_User.

It must be capable of enabling and handling device event notifications.

The host must be able to access the PERIPH_CTRL_CTRL hardware register.

4.3.3.2 Pre BOOT Parameters (Power up Process)

Figure 11 depicts the host end power up process that must be followed to correctly lead the device from power on to the POWER_UP state.

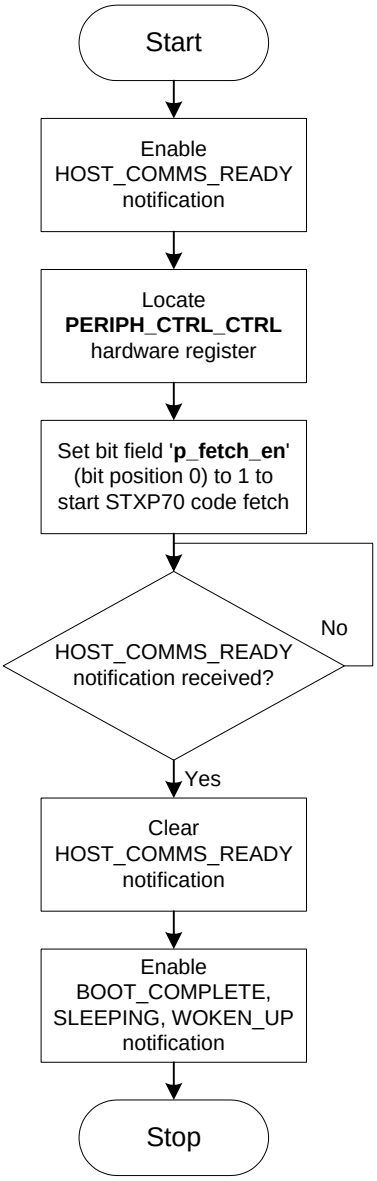
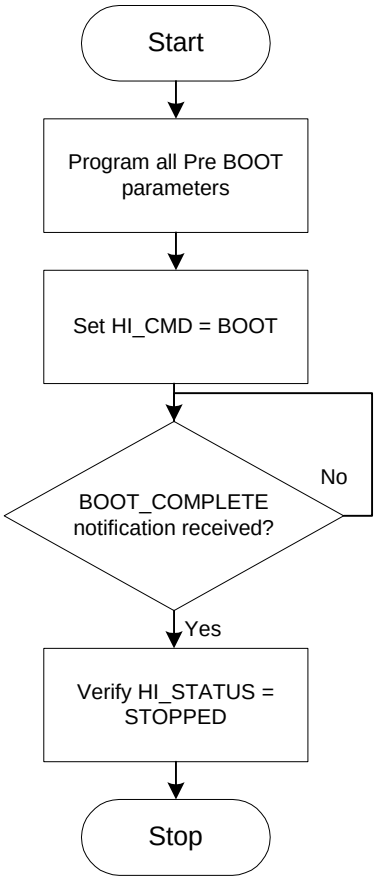


Figure 11 Power Up Process

4.3.3.3 BOOT Process

Once the HOST_COMMS_READY event notification has been received by the device, the host can perform page element reads and writes. Figure 12 depicts the BOOT process that must be followed by the host to correctly lead the device from the POWER_UP state to UNITIALISED state.



1.)HI_STATUS to be read as HostHostInterface_Status.e_HostInterfaceHighLevelState

2.)HI_CMD to be read as HostInterface_Control.e_HostInterfaceCommand_User

3.)Comamnd BOOT to be read as HostInterfaceCommand_e_BOOT

4.)Status STOPPED to be read as HostInterfaceHighLevelState_e_STOPPED

Figure 12 BOOT Process

4.3.3.4 SLEEP Process

In the UNINITIALISED state, the device can be commanded to enter a low power state through the SLEEP command. In this low power state, the device does the following[1]:

Pull XSHUTDOWN of all the sensors to low.

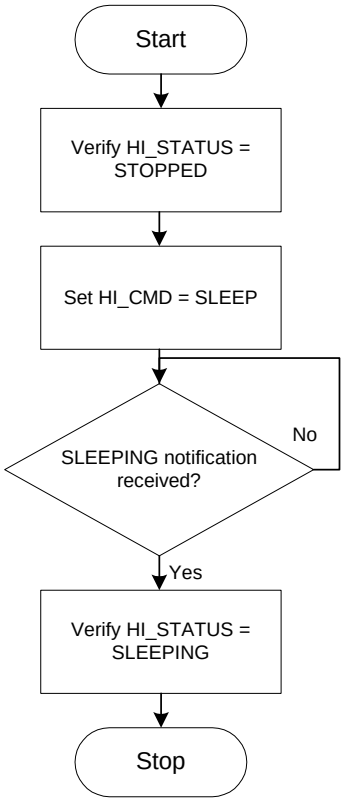
Power down all the clock inputs to the device.

Put the onboard STXP70 processor into idle state.

[1] At the moment, the device does not perform any action apart from generating a SLEEPING notification. These features will be implemented in future device firmware releases.

Generate SLEEPING event notification.

Figure 13 depicts the SLEEP process that must be followed by the host to lead the device into SLEEP state.



1.)HI_STATUS to be read as HostHostInterface_Status.e_HostInterfaceHighLevelState

2.)HI_CMD to be read as HostInterface_Control.e_HostInterfaceCommand_User

3.)Comamnd SLEEP to be read as HostInterfaceCommand_e_SLEEP

4.)Status SLEEPING to be read as HostInterfaceHighLevelState_e_SLEPT

5.) Status STOPPED to be read as HostInterfaceHighLevelState_e_STOPPED

Figure 13 Sleep Process

4.3.3.5 Wake Process

In the SLEEPING state, the device can be commanded to become ready for a new streaming command. In this low power state, the device does the following[2]:

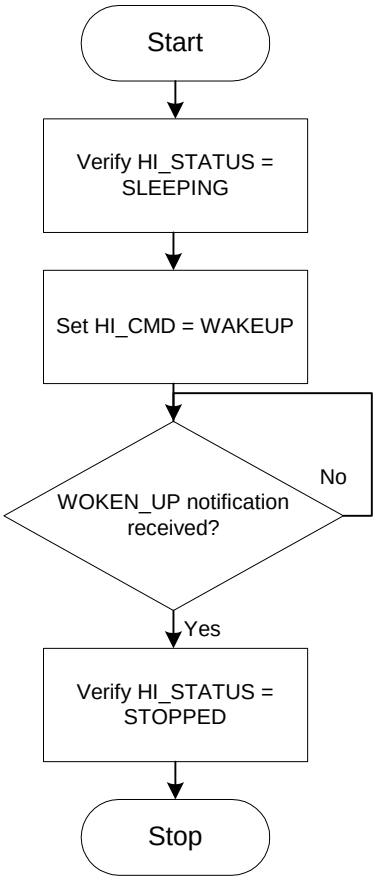
Pull the onboard STXP70 processor into active state.

Power up all the relevant clock inputs to the device.

Generate WOKEN_UP event notification.

Figure 14 depicts the WAKEUP process that must be followed by the host to lead the device into STOPPED state.

[2] At the moment, the device does not perform any action apart from generating a WOKEN_UP notification. These features will be implemented in future device firmware releases.



1.)HI_STATUS to be read as HostHostInterface_Status.e_HostInterfaceHighLevelState

2.)HI_CMD to be read as HostInterface_Control.e_HostInterfaceCommand_User

3.)Comamnd WAKEPUP to be read as HostInterfaceCommand_e_WAKEUP

4.)Status STOPPED to be read as HostInterfaceHighLevelState_e_STOPPED

5.) Status SLEEPING to be read as HostInterfaceHighLevelState_e_SLEEPING

Figure 14 Wake Process

4.3.3.6 RUN Process[3]

In the Stopped state, the device can accept e_RUN command. After Run, the device does the following:

Run pipe algorithm and setup pipe.

Update sensor.

Start Pipe receiver.

Start Sensor.

On successful streaming, notify Host.

4.3.3.7 Pre RUN Parameters

The following should be setup by the host before issuing a RUN command:

Override any sensor parameters if required. It is possible for the host to override certain sensor parameters like the default colour matrix, sensor frame constraints and sensor video timing constraints.

[3] A detailed description of streaming operation is given in section 13

This step may be required if the sensor does not report its video timing, frame dimension or other parameters correctly.

Setup video timing parameters. There are certain parameters that are needed by the device to setup the sensor clock chain correctly. Please refer to section **Error! Reference source not found.** for details.

Setup the output image resolution and output image format for the pipes. Refer to section 10.3.1 for details.

Setup zoom parameters.[4]

4.3.3.8 Live Parameters

Run Sub Commands TBD.

4.3.4 HostInterafceManager Page Elements

4.3.4.1 HostInterface_Control

HostInterface_Control	Control interface for the HostInterfaceManager	
Page Elements	Description	Range/ Possible Values
e_HostInterfaceCommand_User	The command element of the state machine. All commands to the state machine should be issued through this element. Description of the user commands have been provided in Table 2.	HostInterfaceCommand_e_BOOT HostInterfaceCommand_e_RUN HostInterfaceCommand_e_STOP HostInterfaceCommand_e_STOP_ISP HostInterfaceCommand_e_SLEEP HostInterfaceCommand_e_WAKEUP HostInterfaceCommand_e_SWITCH_SENSOR HostInterfaceCommand_e_RESET_ISP

Table 4 Host Interface Control Page

4.3.4.2 HostInterfaceStatus

HostInterface_Status	Status page of the HostInterfaceManager	
Page Elements	Description	Range/ Possible Values
e_HostInterfaceHighLevelState	Hi level status element of the state machine. Depicts the current status of the device. Detailed description along with state transitions have been provided in section 4.3	HostInterfaceHighLevelState_e_INIT HostInterfaceHighLevelState_e_STOPPED HostInterfaceHighLevelState_e_SETUP HostInterfaceHighLevelState_e_RUNNING HostInterfaceHighLevelState_e_SLEPT

[4] At the moment, the only user zoom parameter to be set is Zoom_Control.f_SetFOVX which must be set to the desired FOV. Eventually a complete zoom section will describe the zoom setup process.

Table 5 Host Interface Status Page

4.4 Firmware Events Signaling

4.4.1 Overview

The device signals the host about a few important events through the ITM. These event notifications normally trigger into the host as interrupts. It is the responsibility of the host to enable the events corresponding to which it wishes to receive notifications from the device.

The ITM is the Interrupt Manager module of the device. It routes device notifications to the host. It supports 4 event registers and each of these event registers can manage 32 device events. Hence in all the ITM can manage 128 events (4x32). For details about the ITM, refer to the device functional specifications.

4.4.2 Usage

4.4.2.1 Mapping of Device Events to ITM Events

As mentioned earlier, the ITM has 4 event registers and each of these event registers have 32 events each. Each of the device notification has been assigned an event register and an event within the corresponding register. This mapping is static and cannot be changed. Table 6 shows the mapping of the different device events to ITM events.

Event Name	ITM Mapping (X,Y) (Event register X, and source Y, within X)	Comments
HOST_COMMS_READY	0, 0	Raised when the device is ready to accept host comms requests
HOST_COMMS_OPERATION_COMPLETE	0, 1	Raised to signify that the last host comms operation has completed
BOOT_COMPLETE	0, 2	Raised in context of a BOOT command to the Host Interface. Signifies that the device has completed the BOOT process and is ready to receive a new command
SLEEPING	0, 3	Raised in context of a SLEEP command to the Host Interface. Signifies that the device has transitioned into the SLEEP state.
WOKEN_UP	0, 4	Raised in context of a WAKE_UP command to the Host Interface. Signifies that the device has transitioned out of the SLEEP state.
ISP_STREAMING	0, 5	Raised in context of a RUN command to the Host Interface. Signifies that the device has started the streaming operation. In context of a RUN command from the host, after issuing a SENSOR_START_STREAMING notification, the device programs the Rx to start streaming, and polls on the Rx hardware status to show RUNNING. Subsequently this notification is issued.
ISP_STOP_STREAMING	0, 6	Raised in context of a STOP or ISP_STOP command to the Host Interface. Signifies that

		the device has stopped steaming. No assumptions should be made about the state of the sensor. In context of a STOP command from the host, the device programs the Rx to stop streaming, and polls on the Rx hardware status to show IDLE. Subsequently it polls on the number of frames streamed out of the active pipes to be equal to the number of frames streamed into the ISP. This guarantees that the complete frame has been streamed out of the ISP. Subsequently this notification is issued.
SENSOR_STOP_STREAMING	0, 7	Raised in context of a STOP command to the Host Interface. Signifies that the sensor has stopped steaming. In context of a STOP command from the host, after issuing an ISP_STOP_STREAMING notification, the device initiates an i2c transaction to stop the sensor and waits for a time period equal to one frame time. This notification is issued after the expiry of this timeout. This guarantees that when this notification is issued, the sensor has stopped streaming.
SENSOR_START_STREAMING	0, 8	Raised in context of a RUN command to the Host Interface. Signifies that the sensor has started steaming. In context of a RUN command from the host, after initiating an i2c transaction to start the sensor, the device waits on a host programmed timeout (SystemSetup.f_SensorStartDelay_us). This notification is issued after the expiry of this timeout.
ISP_LOAD_READY	0, 9	Raised in context of a run command for a image load from memory operation. At this point, the host can ready off the line length requirements and program the DMA accordingly.
ZOOM_CONFIG_REQUEST_DENIED	0, 10	Raised when a zoom configuration request in context of a zoom command is denied by the frame dimension manager. It indicates that the requested zoom command violates a system level constraint like maximum frame rate requirement or maximum line length requirement etc.
ZOOM_CONFIG_REPROGRAM_REQUIRED	0, 11	Raised when a new sensor configuration is needed in context of a zoom command.
ZOOM_STEP_COMPLETE	0, 12	Raised when a zoom step has been absorbed by the device. After the host receives this notification, it is safe to issue another zoom command.
ZOOM_SET_OUT_OF_RANGE	0, 13	Raised in context of a Zoom_SetCenter command when the desired center cannot be achieved at the current field of view and

		current sensor constraints.
STREAMING_ERROR	0, 14	Raised in context of a streaming command when there has been an error in starting to stream (possibly due to wrong field of view or center requirements)
ISP_RESET_COMPLETE	0, 15	Raised in context of reset command from the host when the resetting of all the IPs in the ISP is complete.
MASTER_I2C_ACCESS_FAILURE	0, 16	Raised when master i2c access fails.
GLACE_STATS_READY	0, 17	Raised when Glace Statistics are Ready for HOST to be read from the memory.
HISTOGRAM_STATS_READY	0, 18	Raised when Histogram Statistics are Ready for HOST to be read from the memory.
EXPOSURE_AND_WB_PARAMETERS_UPDATED	0, 19	Raised to inform that the previous exposure, White Balance and frame rate parameters have been applied in the current frame.
AUTOFOCUS_STATS_READY	0, 20	Raised when AutoFocus accumulated Stats for all the zones are Ready for Current Frame , Now Host can read the Statistics from Memory .
FLADRIVER_LENS_STOP	0, 21	Raised to inform the Lens Movement has Stopped .Once the lens Start Moving it takes some time to Complete the Movement or Reach the Target Position , As soon as the Lens STOP its Movement or Target position is Achieved this Notification is raised .
ZOOM_OUTPUT_IMAGE_RESOLUTION_READY	0, 22	Raised to inform that the new output image resolution is ready to facilitate the host to adapt its DMA programming for dynamic change in output image resolution.
Reserved	0,23	Reserved
Reserved	0,24	Reserved
Reserved	0,25	Reserved
NVM_EXPORT_DONE	0,26	This event is used to notify HOST that NVM data export is complete. raised after exporting NVM data.
SENSORTUNING_AVAILABLE	0,27	This event is used to notify HOST that Sensor Tuning configuration is supported by sensor.
POWER_NOTIFICATION	0,28	This event is used to notify HOST a power operation is pending at his end. Once this notification is raised HOST can read power command and related PE values to take appropriate action.

SMS_NOTIFICATION	0,29	This event is used to notify the HOST that SMS request operation has completed, and the HOST may read the computed values from SMSStatus Page.
SENSOR_OUTPUT_MODE_EXPORT_NOTIFICATION	0, 30	This event is used to notify HOST that Sensor output mode data export has been completed.
Valid BMS frame notification	0,31	The event is raised when ISP FW detect valid BMS frame i.e. Nips target is absorbed with Flash (if active).
HOST_TO_SENSOR_ACCESS_COMPLETE	1, 0	Raised in context of a host to master i2c access operation initiated by the host. It indicates that the operation has completed. For a write operation, the notification indicates that the data has been written to the sensor while for the read operation it indicates that the data has been read from the sensor and is available in the data pages
SENSOR_COMMIT Notification	1, 1	Raised when the request for applying various sensor params through g_SensorPipeSettings_Control is complete.
ISP_COMMIT Notification	1, 2	Raised when the request for applying various Pipe params through g_SensorPipeSettings_Control is complete.

Table 6 Mapping of Device Events

4.4.2.2 Pre-Requisites

The following hardware registers must be known to the host to be able to setup the device event notifications:

Address of ITM_FW_EVENTn_EN_BCLR hardware register.

Address of ITM_FW_EVENTn_EN_BSET hardware register.

Address of ITM_FW_EVENTn_EN_STATUS hardware register.

Address of ITM_FW_EVENTn_STATUS hardware register.

4.4.2.3 Pre Boot Parameters

None

4.4.2.4 Pre Run Parameters

All the device events for which the host wants to be notified must be enabled before issuing a RUN command. Similarly, all the device events for which the host does not want to be notified must be disabled before issuing a RUN command.

4.4.2.5 Enabling Device Notification

The host must enable the device notifications that it wants to receive. Figure 15 depicts the procedure to enable a device notification mapped onto ITM event (X, Y). By default all device notifications are disabled.

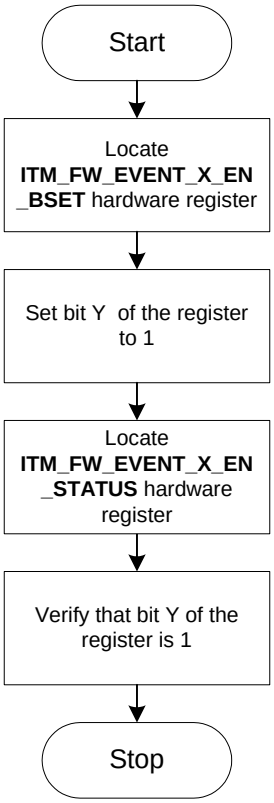


Figure 15 Enabling device event mapped to ITM (X, Y)

4.4.2.6 Disabling Device Notification

The host must disable the device notifications that it does not want to receive. Figure 16 depicts the procedure to disable a device notification mapped onto ITM event (X, Y).

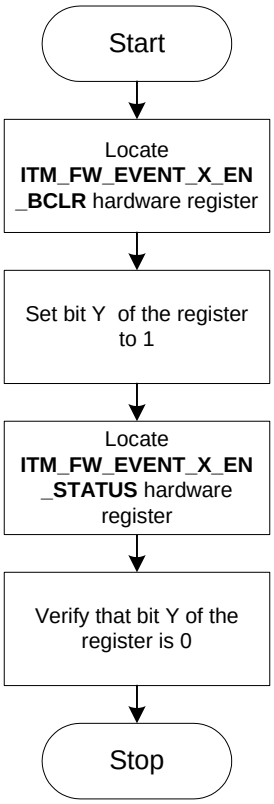


Figure 16 Disabling device event mapped to ITM (X, Y)

4.4.2.7 Handling Device Notification

All notifications raised by the device must be cleared by the host. Failure to do so will prevent any further notifications being raised into the host. Figure 17 depicts the procedure to clear a device notification mapped onto ITM event (X, Y).

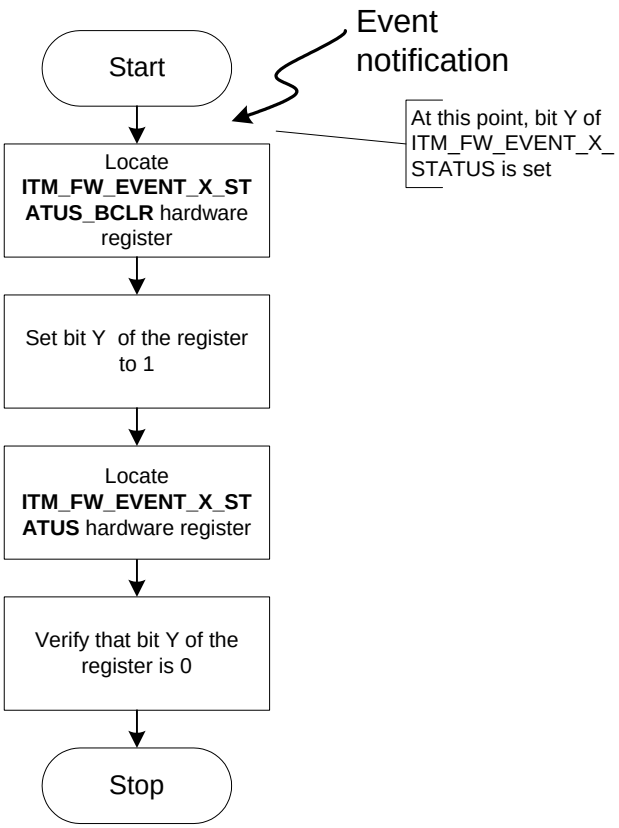


Figure 17 Clearing device event mapped to ITM (X, Y)

Event counts for each event	g_Event0Count	
Page Elements	Description	Range
u16_EVENT0_0_HOST_COMMS_READY	Event Count corresponding to Host Comms Ready Event	
u16_EVENT0_1_HOST_COMMS_OPERATION_COMPLETE	Event Count corresponding to Host Comms Operation Complete Event	
u16_EVENT0_2_BOOT_COMPLETE	Event Count corresponding to BOOT Complete Event	

u16_EVENT0_3_SLEEPING	Event Count corresponding to Sleeping Event	
u16_EVENT0_4_WOKEN_UP	Event Count corresponding to Woken Up Event	
u16_EVENT0_5_ISP_STREAMING	Event Count corresponding to ISP Streaming Event	
u16_EVENT0_6_ISP_STOP	Event Count corresponding to ISP Stop Event	
u16_EVENT0_7_SENSOR_STOP	Event Count corresponding to Sensor Stop Event	
u16_EVENT0_8_SENSOR_START	Event Count corresponding to Sensor Start Event	
u16_EVENT0_9_ISP_LOAD_READY	Event Count corresponding to ISP Load Ready Event	
u16_EVENT0_10_ZOOM_CONFIG_REQUEST_DENIED	Event Count corresponding to Zoom Config Request Denied Event	
u16_EVENT0_11_ZOOM_CONFIG_REQUEST_REPROGAM_REQUIRED	Event Count corresponding to Zoom Config Request Reprogram Required Event	
u16_EVENT0_12_ZOOM_STEP_COMPLETE	Event Count corresponding to Zoom Step Complete Event	
u16_EVENT0_13_ZOOM_SET_OUTOF_RANGE	Event Count corresponding to Zoom Set Out Of Range Event	
u16_EVENT0_14_STREAMING_ERROR	Event Count corresponding to Streaming Error Event	
u16_EVENT0_15_RESET_ISP_COMPLETE	Event Count corresponding to	

	Reset ISP Complete Event	
u16_EVENT0_16_MASTER_I2C_ACCESS_FAILED	Event Count corresponding to Master I2C Access Failed Event	
u16_EVENT0_17_GLACE_STATS_READY	Event Count corresponding to Glace Stats Ready Event	
u16_EVENT0_18_HISTOGRAM_STATS_READY	Event Count corresponding to Histogram Stats Ready Event	
u16_EVENT0_19_EXPOSURE_AND_WB_PARAMETERS_UDPATE	Event Count corresponding to Exposure And WB Parameters Update Event	
u16_EVENT0_20_AF_STATS_READY	Event Count corresponding to AF Stats Ready Event	
u16_EVENT0_21_FLA_DRIVER_LENS_STOP	Event Count corresponding to FLA Driver Lens Stop Event	
u16_EVENT0_22_ZOOM_OUTPUT_RESOLUTION_READY	Event Count corresponding to Zoom Output Resolution Ready Event	
u16_EVENT0_23_reserved01	reserved	
u16_EVENT0_24_ reserved02	Reserved	
u16_EVENT0_25_ reserved03	reserved	
u16_EVENT0_26_NVM_Export	Event count corresponding to NVM data export event.	
u16_EVENT0_27_ SensorTuning _Available	Event count corresponding to Sensor Tuning available event.	
u16_EVENT0_28_Power_Notification	Event count corresponding to Power notification event.	

u16_EVENT0_29_SMS_Notification	Event count corresponding to SMS notification event.	
u16_EVENT0_30_Sensor_Output_Mode_Export_Notification	Event count corresponding to Sensor output mode notifications sent to HOST.	
u16_EVENT0_31_Valid_Frame_Notification	Event count corresponding to BMS Valid frame notification	

Event counts for each event	g_Event1Count	
Page Elements	Description	Range
u16_EVENT1_0_HOST_TO_MASTER_I2C_ACCESS	Event Count corresponding to Host To Master I2C Access Event	
u16_EVENT1_1_SENSOR_COMMIT	Event Count corresponding to Sensor Commit Event	
u16_EVENT1_2_ISP_COMMIT	Event Count corresponding to ISP Commit Event	

Event counts for each event	g_Event2Count	
Page Elements	Description	Range
u16_Reserved	Reserved for future	

Event counts for each event	g_Event3Count	
Page Elements	Description	Range
u16_EVENT3_0_DMA_GRAB	Event Count corresponding to number of Grab-OK or not-OK frames ouput by the Pipe.	
u16_EVENT3_1_DMA_GRAB_VideoStab	Event count corresponding to GRAB about the output size change.	
u16_EVENT3_2_DMA_GRAB_Abort	Event count corresponding to GRAB abort.	

4.5 Grab Interface Overview

4.5.1 Grab Interface Overview

In case of BMS (BMS2 currently supported) being enabled, the provision of getting (Glace, Histo) statistics is also supported as described in Flash document. Another page element i.e., e_Flag_FlashMode is used in case of flash. If user sets Flash Mode to ON (true), and if flash lit frame arrives, the firmware updates a shared variable (GRAB OK/not OK) and the frame id (count) of the Flash-lit frame, both of which the Host can read. Three shared variables are being used for sharing information with grab. Base address of the structure containing these shared variables is 0x00001FE0. They are described below:-

g_GrabNotify.u32_DMA_GRAB_Indicator : The GRAB shared variable is a 32-bit variable and is located in the Fixed-Address segment in XP70 TCDM. The variable is set to 1 (Valid Frame/Grab OK) or reset to 0 (Invalid Frame / Grab Not OK) according to the algorithm.

g_GrabNotify.u32_DMA_GRAB_Frame_Id : The frame id is another 32-bit variable and is located at a Fixed-Address segment in XP70 TCDM. The Host may compare the frame Id obtained here with the Id exported in FrameParamStatus page, after stats for the flash lit frame is copied. This will facilitate matching frame with associated statistics.

Above two variables are also updated when Flash Mode is OFF, and stats are still requested by the user. In general, there are several ways of requesting Stats supported by the firmware. In the current context, shared variables will be updated and notified to Host when stats are requested by toggling system coin, or when a flash lit frame arrives.

g_GrabNotify.u32_DMA_GRAB_Indicator_For_VideoStab : ISP FW and host share a common look-up table to calculate actual pipe output for a given FOV. ISP FW dynamically change this output size on a new setFOV() request according to the lookup table and GRAB also can adapt to this new output size and program DMA accordingly on the fly. Synchronization between ISP FW and GRAB is achieved through a 32 bit share variable g_GrabNotify.g_DMA_GRAB_Indicator_For_VideoStab which has a fixed address in XP70 domain.

g_GrabNotify.u32_DMA_GRAB_Abort: The u32_DMA_GRAB_Abort is another 32-bit variable and is located at a Fixed-Address segment in XP70 TCDM. When this shared variable is set to 1, GRAB should proceed to abort frame being processed by it, And after aborting current frame DMA should reset this variable to 0.

4.5.2 Grab Interface from Flash and Exposure Perspective

In case of BMS (BMS2 currently supported) being enabled, the provision of getting (Glace, Histo) stats is also supported as per the algorithm below. Another page element i.e., e_Flag_FlashMode, comes into picture. It defines the Flash Mode. If user sets Flash Mode to ON (true), and if flash lit frame arrives, the firmware updates a shared variable (GRAB OK/not OK) and the frame id (count) of the Flash-lit frame, both of which the Host can read.

Flash Mode is set to ON or OFF through the page element e_Flag_FlashMode in the FlashControl Page by the host. This element is significant only in BMS use-case.

The GRAB shared variable is a 32-bit variable and is located in the Fixed-Address segment in XP70 TCDM. Its address is **0x00001FE0**. The variable is set to 1 (Valid Frame/Grab OK) or reset to 0 (Invalid Frame / Grab Not OK) according to the algorithm (Figure 1).

The frame id is another 32-bit variable and is located in the Fixed-Address segment in XP70 TCDM at **0x00001FE4**.

Internally both the above variables are members of a common structure in the firmware.

The Host may compare the frame Id obtained here with the Id exported in FrameParamStatus page, after stats for the flash lit frame is copied. This way he will be able to know the frame for which those stats have been exported.

The notification to MMDSP is sent by setting the zeroth bit of the ITM_FW_EVENT3_STATUS register. Also, the u16_EVENT3_0_DMA_GRAB element of the Event_Count Page is incremented on each notification.

The shared variables are also updated when Flash Mode is OFF, and stats are still requested by the user. In general, there are several ways of requesting Stats supported by the firmware. In the current context, shared variables will be updated and notified to Host when stats are requested by toggling system coin, or when a flash lit frame arrives.

When stats is required to be exported, SystemConfig Status Coin will be toggled (equaled to SystemSetup Control Coin) only after stats corresponding to the flash lit frame has been exported and glace notification sent to the Host.

Following is the algorithm for BMS use case, w.r.t. stats and flash lit frame:

```
If (BMS)
{
    If (((Grab Mode = NORMAL) AND (Flash and Exposure Converged)) OR (Grab Mode = FORCE_GRAB))
    {
        Update Grab Variable to “GRAB OK”; //valid frame
        Update ISL (Frame) ID;
        Send GRAB OK to Host;
        Send Valid Frame Notification to Host;
    }
    Else   // in all remaining use cases
    {
        Update Variable to “GRAB NOT OK”;//invalid frame
        Send Notification to MMDSP;
    }

} // Endif BMS
```

Figure: Algorithm for notification to MMDSP in BMS use case

PS: Only BMS2 is currently supported.

Also to be noted is that the update of the page elements u8_FlashFiredFrameCount and u32_flash_fired (as in the non-BMS use case) is also valid in this case, provided the host toggles flash (SystemSetup) coin.

The Host can further ensure that it **always** receives GRAB OK (valid) frames, by setting the value of e_BMS_GrabMode_Ctrl to BMS_GrabMode_e_FORCE_OK. This value could be used in the zero shutter lag use case.

On the contrary, the Host can also ensure that it **always** receives GRAB NOT OK (invalid) frames by setting the value of e_BMS_GrabMode_Ctrl to BMS_GrabMode_e_FORCE_NOK. This value could be used in the preflash use case.

In general, the (default) value of e_BMS_GrabMode_Ctrl will be BMS_GrabMode_e_NORMAL.

4.5.3 Modes of Operation

4.5.3.1 Pre-requisites

None

4.5.3.2 Pre BOOT Parameters

None

4.5.3.3 Pre RUN Parameters

4.5.3.4 Live Parameters

None

4.5.3.5 Event Description

Following events are involved:-

DMA_GRAB	3,0	The event is raised when ISP FW detect that the frame is illuminated i.e. sensor has triggered the flash strobe.
DMA_GRAB_VideoStab	3,1	This event is used to notify GRAB about the output size change.
DMA_GRAB_Abort	3,2	This event is used to notify GRAB that Smia-rx has been aborted in the middle of current frame. So GRAB should take appropriate action.

4.5.4 Grab Interface Page Elements

4.5.4.1 DMA_GRAB_Indicator_Params page element

Page Elements	Description	Range
u32_DMA_GRAB_Indicator	Only significant in BMS (BMS2) mode, this variable indicates to the Host whether the frame output from the Pipe is OK (to be Grabbed) or Not-OK. The use cases when this variable is set (1) or reset (0) is explained in the Flash section of User Manual.	0-1
u32_DMA_GRAB_Frame_Id	This variable is also significant in BMS (BMS2) mode, as well as when the first variable above (g_DMA_GRAB_Indicator) is set to true. This gives the frame count of the output frame (Grab OK), so that the Host may sync up when stats corresponding to the same frame are exported.	
u32_DMA_GRAB_Indicator_For_VideoStab	This shared variableis for synchronization between ISP FW and GRAB component which is responsible to setup DMA with correct OP size acording to the common lookup table. ISP FW calculates new o/p size on receiving of new FOVusing the look up table and updates scaler/cropper registers for the new pipe o/p. ISP FW sends a notification to GRAB notifying that new o/p size has been calculated and GRAB can reconfigure DMA according to the new o/p size and disable pipe output at vid complete. GRAB waits for rendezvous of EOT and ISP FW notification and reprograms the DMA with new o/p size (GRAB uses the same look up table as ISP FW to calculate DMA size) and sets the variable u32_DMA_GRAB_Indicator_For_VideoStab = 1. ISP FW poll on this variable and reenable pipe output when its value becomes 1.	
u32_DMA_GRAB_Abort	When this shared variable is set to 1, GRAB should proceed to abort frame being processed by it, And after aborting current frame DMA should reset this variable to 0.	

5. Generic System Configuration

5.1 Overview

Certain inputs (like host external frequency to the sensor, sensor device ID etc.) are needed by the device for proper functioning. These inputs are platform dependent and hence must be specified by the host.

5.2 Single Step Exposure, White Balance and Frame Rate

It is possible that the host may want to disable the auto mode of operation for Exposure, White Balance, and frame rate. It is envisaged that in such cases, the host will use Exposure in DRIVER_MANUAL_MODE, White Balance in MANUAL_RGB and Frame Rate in `FrameRateMode_e_SINGLE_STEP` mode (by programming `FrameRateControl.e_FrameRateMode`). The device allows the host to program the parameters to be applied for these modules in an atomic manner protected by a coin mechanism. To allow the host end algorithm to close the loop with respect to programming of parameters and the absorption of these parameters, the device also raises an event notification once the host specified parameters have been absorbed.

For this mechanism, the device must be put into single step metering mode by programming `SystemSetup.e_SystemMeteringMode_Control = SystemMeteringMode_e_SINGLE_STEP`. This must be done before starting to stream. Whenever, the host wishes to change the exposure, white balance or frame rate parameters, it can program these parameters and toggle `SystemSetup.e_Coin_Ctrl`. The device will consider these parameters and program them onto the sensor and ISP at appropriate point in the frame. Whenever, the exposure parameters sent to the sensor match with those reported by the sensor at the start of the frame, the corresponding white balance gain parameters are applied onto the ISP. At the same time, whenever the exposure and frame rate parameters sent to the sensor, match with those reported by the sensor at the start of the frame, the device raises an `EXPOSURE_AND_WB_PARAMETERS_UPDATED` notification. The host must wait on this notification after toggling `SystemSetup.e_Coin_Ctrl` before it can attempt to change the exposure, white balance or frame rate again.

There is a coin mechanism built in the frame rate module, which can be used by the host to run the frame rate in the Single step mode. However, this functionality should not be used by the host. The frame rate coin is used internally by the firmware.

It is important to note that the transition from `SystemMeteringMode_e_CONTINUOUS` to `SystemMeteringMode_e_SINGLE_STEP` and vice versa is not possible while streaming.

5.2.1 Pre-requisites

The host should be able to program `SystemSetup` page.

5.2.2 Pre BOOT parameters

All the parameters are pre boot and must be programmed prior to issuing a BOOT command.

5.2.3 Pre RUN parameters

None

5.2.4 Live parameters

None

5.3 Page Elements

System Configuration page elements

SystemSetup	System configuration page	
Page Elements	Description	Range/ Possible Values
f_SensorInputClockFreq_Mhz	Specifies the frequency of the external clock being input to the sensor	Sensor dependent. SMIA range is 6-27Mhz
f_MCUClockFreq_Mhz	Frequency of the input clock to the MCU	Platform Dependent
f_ClockHost_Mhz	Value of clk_host_ipp in Mhz. It is the clock from which the STXP70 is clocked and the emulated sensor clocks are generated	Platform Dependent
f_SensorStartDelay_us	Specifies the number of micro seconds after which the sensor will output good clocks after been given the start sensor command. The device will switch to sensor clocks after this time expires.	Sensor dependent.
u16_SensorDeviceID	Specifies the I2C device ID of the active sensor	Sensor dependent
u16_SensorXshutdownDelay_us	Specifies the number of micro seconds that the device will wait before initiating an i2c read/write transaction after having pulled the XSHUTDOWN of the sensor high	Sensor dependent
e_DeviceAddress_Type	Specifies whether the device address of the sensor is a 7 bit device address or 10 bit device address	DeviceAddress_e_10BitDeviceAddress (if the device address is a 10 bit device address), DeviceAddress_e_7BitDeviceAddress (if the device address is a 7 bit device address)
e_DeviceIndex_Type	Specifies whether the index of the sensor is a 16 bit index or an 8 bit index	DeviceIndex_e_16BitDataIndex (if the device address is a 16 bit device index), DeviceIndex_e_8BitDataIndex (if the device address is a 8 bit device index),
e_InputImageSource	Source of input image to the ISP	InputImageSource_e_Sensor0 (0), InputImageSource_e_Sensor1 (1), InputImageSource_e_Rx (2), InputImageSource_e_BayerLoad1(3), InputImageSource_e_BayerLoad2(4), InputImageSource_e_RGBLoad (5)
e_InputImageInterface	Specifies the input image interface (i.e. CSI0, CS1 or CCP) Only valid if e_InputImageSource is sensor0 or sensor1	
u8_NumberOfCSI2DataLines	Specifies the number of CSI2 data lines Valid only if input image source is sensor and input mage interface is CSI	

e_RxTestPattern	Specifies the Rx test pattern type. Valid only when e_InputImageSource == e_Rx	
e_Flag_PerformIPPSetup	Specifies if the IPP setup has to be done by the ISP firmware	
u8_FrameCount[5]	Specifies the number of frames to stream	Program this element to the number of frames to stream.
u8_LinesToWaitAtRxStop	Specifies the number of lines to wait after Rx STOP command to allow the ISP pipe to be flushed.	
e_SystemMeteringMode_Control	Specifies the ype of Metering the host want to use.CONTINUOUS or SINGLE_STEP Default value is CONTINUOUS	SystemMeteringMode_e_CONTINUOUS, SystemMeteringMode_e_SINGLE_STEP
e_Flag_InhibitExpMetering	Specifies whether Exposure Module is to be used in the metering process Default value is FALSE	Flag_e_TRUE Flag_e_FALSE
e_Flag_InhibitWbMetering	Specifies whether Exposure Module is to be used in the metering process. Default value is FALSE	Flag_e_TRUE Flag_e_FALSE
e_Coin_Ctrl	Control coin to available to HOST for controlling the Flash, Exposure, channel gains and image quality related IP. The coin is used to apply Flash, Exp, AWB and new IP values. Host should calculate all the values i.e. exposure, channel gains, frame rate, RSO etc. and program them in their page elements. Once all the parameters for the particular frame is programmed in the firmware, host should toggle this coin. FW will apply all these values as one atomic transaction. Default value = Tails	Each time if coin is toggled, FW will export glace and histogram statistics.
e_Flag_mirror	Image will be horizontally mirrored if set to Flag_e_True [DEFAULT]: Flag_e_False	
e_Flag_flip	Image will be vertically flipped if set to Flag_e_True [DEFAULT]: Flag_e_False	
e_Coin_Glace_Histogram_Ctrl_debug	Coin to get only Glace and histogram statistics. This will not apply new AEC values. **NOTE: To be used only for debug purpose**	
e_IdleMode_Ctrl	XP70 Processor Idle Mode Control. [DEFAULT]: IdleMode_e_IDLE_MODE_NONE	

[5] Not used for current firmware release.

u8_crm_clk_pip_in_div	u8_crm_clk_pip_in_div specify the value of divider to be used in BML phase. The value should be provided before BOOT command. Based on value explained below, ISP FW will generate BML clock from input clock. For example, if value = 0x8 is set and input clock is 200MHz, ISP FW will use 200 MHz as BML clock.	/// [4:0] CLK_PIPE_IN_DIV: /// 5'b01_00_0 : 1 : 0x8 /// 5'b01_00_1: 7/6 (1.16...): 0x9 /// 5'b01_01_0: 8/6 (1.33...): 0xA /// 5'b01_10_0: 6/4 (1.5) : 0xC /// 5'b01_10_1: 10/6 (1.66 : 0xD /// 5'b01_11_0: 7/4 (1.75) : 0xE /// 5'b10_00_0: 2 : 0x10 /// 5'b10_10_0: 10/4 (2.5) : 0x14 /// 5'b11_10_0: 14/4 (3.5) : 0x1C
e_Flag_abortRx_OnStop	SmiaRx will be aborted on STOP command Flag_e_TRUE : smia-rx should abort when HOST sends STOP command Flag_e_FALSE : smia-rx should do stop when HOST sends STOP command [DEFAULT]: Flag_e_FALSE	Flag_e_TRUE Flag_e_FALSE
e_Flag_AecStatsCancel	Significant when there is a pending stats request, and the Host issues a STOP command. If TRUE, pending stats request will be cancelled before FW moves to STOPPED state. If FALSE, pending stats request will be completed, but notification will send to the Host in STOPPED state. [DEFAULT]: Flag_e_TRUE	Flag_e_TRUE Flag_e_FALSE
u8_NumOfFramesTobeSkipped	start_grab_dly setting, i.e., num of frames to be skipped in still capture [Default value] 0	
e_BMS_GrabMode_Ctrl	Send DMA Grab OK or NOTOK forcefully based on the value of this flag, otherwise, it is based on the whether stats are valid for the frame. [Default] BMS_GrabMode_e_NORMAL	BMS_GrabMode_e_NORMAL=0 BMS_GrabMode_e_FORCE_OK=1 BMS_GrabMode_e_FORCE_NOK=2

SystemConfig_Status	System configuration status page	
Page Elements	Description	Range/ Possible Values
e_Flag_InputPipeUpdated	Shows the status of Input Pipe Updation e_FALSE: Not Updated. e_TRUE: Updated.	Flag_e_TRUE Flag_e_FALSE
e_Flag_ExposureOutOfSync	This page element specifies whether the exposure parameters applied and the sensor sending the frame have the same parameters	Flag_e_TRUE

	applied. When TRUE, it indicates that Exposure is in sync.	Flag_e_FALSE
e_Coin_Status	Status coin for fw to know, when to apply the parameters programmed by HOST	
e_Flag_ZoomUpdateDone	Flag to indicate that a zoom step was completed. This information will be used by the SOF interrupt to setup its stats accordingly	Flag_e_TRUE Flag_e_FALSE
e_Flag_mirror	Image is horizontally mirrored if set to Flag_e_True	Flag_e_TRUE Flag_e_FALSE
e_Flag_flip	Image is vertically flipped if set to Flag_e_True	Flag_e_TRUE Flag_e_FALSE
e_Flag_UpdateNotificationPending	Flag to indicate that the notification of the update is pending to the Host	Flag_e_TRUE Flag_e_FALSE
e_UpdateStatus_Exposure	Update status for the exposure cycle	
e_UpdateStatus_FrameRate	Update status of the Frame Rate cycle	
e_UpdateStatus_Flash	Update status of Flash	
e_Coin_Glace_Histogram_Status	status coin for glace-histogram control	
e_PixelOrder	Update status of pixel order from Sensor Bayer pattern	
e_Flag_RunBestSensormode	This flag is used to indicate to zoom manager that Zoom_Run() is to be called when any pipe is enabled on the fly	
e_Flag_FireDummyVidComplete0		
e_Flag_FireDummyVidComplete1		

Input Interface Selection and Programming

The device supports the following three serial data interfaces for accepting image from the sensor:

Three data lane CSI2 interface (referred to as CSI2_0 interface)

Single data lane CSI2 interface (referred to as CSI2_1 interface)

SMIA CCP2 interface (referred to as CCP interface)

Up to two sensors (referred to as sensor0 and sensor1 in the device context) can be mounted on the above three interfaces. The device makes no assumptions about sensor0 or sensor1 being mounted on any of the above three physical interfaces i.e. sensor0 could be mounted on any of the above three interfaces and sensor1 can be mounted on any of the above three interfaces. For this reason, the host must explicitly specify the active input interface in addition to the currently active sensor. This can be specified using `SystemSetup.e_InputImageInterface` element.

CSI2_0 interface supports up to three data lanes. If CSI2_0 interface has been selected, then the host must specify the number of data lanes being actively used for the current streaming operation in the `SystemSetup.u8_NumberOfCSI2DataLines` element. For CSI2_1 interface, the number of data lanes is fixed to one only and cannot be programmed. When CSI2_0 or CSI2_1 interface is selected, then there are other parameters which can be overridden by the host through the page element interface. This has been specified in Table 8.

SystemSetup		
Page Elements	Description	Range/ Possible Values
e_InputImageInterface	Specifies the currently active input image interface	InputImageInterface_CSI2_0: The active interface is CSI2_0 InputImageInterface_CSI2_1: The active interface is CSI2_1 InputImageInterface_CCP: The active interface is CCP
u8_NumberOfCSI2DataLines	Specifies the number of data lanes active for CSI2_0 interface. Valid only when InputImageInterface_CSI2_0 is selected for streaming.	

Table 7 Input Interface Selection Elements

CSIControl[6]	CSI2 interface programming page	
Page Elements	Description	Range/ Possible Values
u16_DataLanesMapCSI2_0	Specifies the data lanes mapping for CSI2_0 interface Needed only when the input image interface is InputImageInterface_CSI2_0	
u16_DataLanesMapCSI2_1	Specifies the data lanes mapping for CSI2_1 interface Needed only when the input image interface is InputImageInterface_CSI2_1	
u8_DPHY0Ctrl	Specifies the DPHY0 clk control. Needed only when input image interface is InputImageInterface_CSI2_0 Consists of bit fields to be interpreted as follows: u8_DPHY0Ctrl[0] : CSI0_SWAP_PIN_CL u8_DPHY0Ctrl[1] : CSI0_HS_INVERT_CL u8_DPHY0Ctrl[2] : CSI0_SWAP_PIN_DL1 u8_DPHY0Ctrl[3] : CSI0_HS_INVERT_DL1 u8_DPHY0Ctrl[4] : CSI0_SWAP_PIN_DL2 u8_DPHY0Ctrl[5] : CSI0_HS_INVERT_DL2 u8_DPHY0Ctrl[6] : CSI0_SWAP_PIN_DL3 u8_DPHY0Ctrl[7] : CSI0_HS_INVERT_DL2	
u8_DPHY1Ctrl	Specifies the DPHY1 clk control. Needed only when input image interface is InputImageInterface_CSI2_1 Consists of bit fields to be interpreted as follows: u8_DPHY1Ctrl[0] : CSI1_SWAP_PIN_CL u8_DPHY1Ctrl[1] : CSI1_HS_INVERT_CL u8_DPHY1Ctrl[2] : CSI1_SWAP_PIN_DL1 u8_DPHY1Ctrl[3] : CSI1_HS_INVERT_DL1 u8_DPHY1Ctrl[7:4] : reserved	
e_SensorCSI2Version_0	Specifies the sensor CSI2 version of sensor mounted on interface 0	
e_SensorCSI2Version_1	Specifies the sensor CSI2 version of sensor mounted on interface 1	
u8_DPHY0Ctrl_4th_lane	Specifies the DPHY0 clk control only Needed only when input image interface is	

[6] This page is not present in firmware v0.28.0, hence these parameters cannot be overridden in firmware v0.28.0.

	InputImageInterface_CSI2_0 with support for 4 Datalanes u8_DPHY0Ctrl_4th_lane[0] : CSI0_SWAP_PIN_DL4 u8_DPHY0Ctrl_4th_lane[1] : CSI0_HS_INVERT_DL4 u8_DPHY1Ctrl[7:2] : reserved	

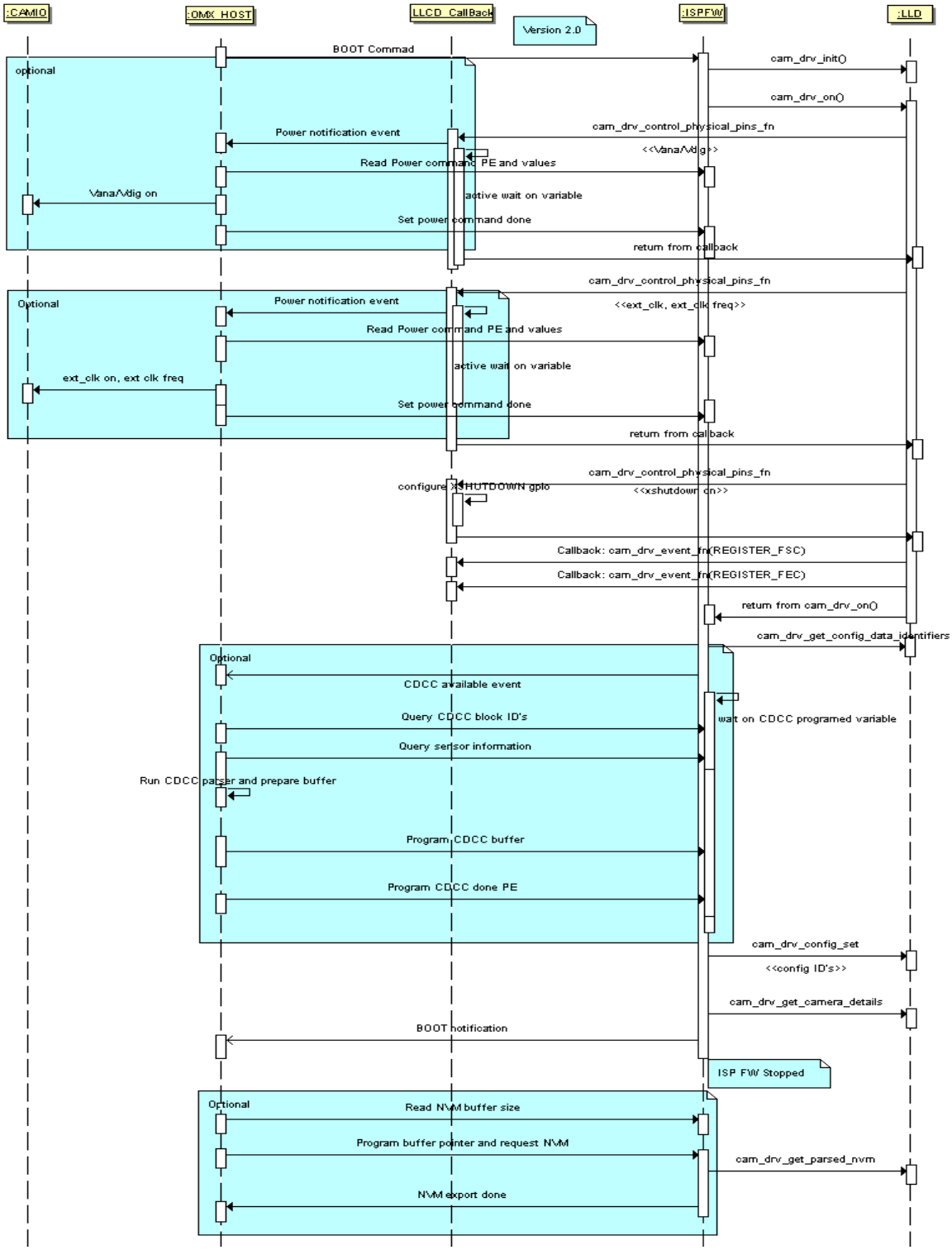
Table 8 CSI2 Interface Programming Page

6. Boot Sequence

6.1 Boot Sequence

6.1.1 Overview

Boot sequence can be best described by following sequence diagram :-



Initially ISP FW is in HostInterfaceLowLevelState_e_RAW_UNINITIALISED state. In order to boot ISP FW HOST must send HostInterfaceCommand_e_BOOT command. HOST may set values of pre-boot page elements before sending boot command. One page element of particular interest in this regard is Sensor_Tuning_Control_ e_Flag_ReadConfigBeforeBoot _Byte0. Please refer to Sensor Tuning section of ISP FW user manual for details of this PE. Completion of boot sequence is notified to HOST by BOOT_COMPLETE event.

HOST can choose to enable SMIA++ power-up sequence by using SensorPowerManagement_Control_e_Flag_EnableSMIAPP_PowerUpSequence_By te0 PE. If this PE is set to Flag_e_TRUE, SMIA++ boot sequence is used. Default value of this PE Flag_e_FALSE.

6.1.2 Pages Exposed

SensorPowerManagement_Control

SensorPowerManagement_Status

SensorPowerManagement_Control		
Page Elements	Description	Range
e_PowerAction	Action PE to HOST. Host should set it to PowerAction_e_complete when requested command is complete before raising power change sequence event, ISP FW will always set this PE to PowerAction_e_requested and wait for its value to become PowerAction_e_complete. [DEFAULT] = PowerAction_e_idle	PowerAction_e_idle PowerAction_e_requested PowerAction_e_complete
e_Flag_Result	Result of operation. Host must program this variable before setting e_PowerAction [DEFAULT] = Flag_e_FALSE	Flag_e_FALSE Flag_e_TRUE
e_Flag_EnableSMIAPP_PowerUpSequence	Flag to enable/disable SMIA++ power-up sequence [DEFAULT] = Flag_e_FALSE	Flag_e_FALSE Flag_e_TRUE

SensorPowerManagement_Status		
Page Elements	Description	Range
u32_ExternalClockFreq_MHz_x100	External Clock frequency which HOST should apply. [DEFAULT] = 0	0 - 4294967295
u16_VoltageAnalog_x100	Result of operation. Host must program this variable before setting e_PowerAction [DEFAULT] = 0	0 - 65535
u16_VoltageDigital_x100	Flag to enable/disable SMIA++ power-up sequence [DEFAULT] = 0	0 - 65535
u16_VoltageIO_x100	I/O voltage [DEFAULT] = 0	0 - 65535
e_PowerCommand	command from isp fw to host. Host should read this variable on receiving power event notification from ISP FW [DEFAULT] = PowerCommand_e_voltage_on	PowerCommand_e_voltage_on PowerCommand_e_voltage_off PowerCommand_e_ext_clk_on PowerCommand_e_ext_clk_off PowerCommand_e_x_shutdown_on PowerCommand_e_x_shutdown_off

6.2 Overview

HOST can read sensor output mode information any time after successful BOOT of ISP FW. For reading sensor output mode information HOST has two options :-

- HOST should read size of sensor output mode data by using page element **g_ReadLLAConfig_Status.u32_sensor_Output_Mode_data_size**. Once HOST is aware about size of sensor output mode data, HOST should allocate a buffer of this size and pass its XP70 domain translated address to ISP FW in page element **g_ReadLLAConfig_Control.ptr32_Sensor_Output_Mode_Data_Address** (This PE is set to 0 by default). After that HOST should toggle coin **g_ReadLLAConfig_Control.e_Coin_ReadLLAConfigControl**. ISP FW will copy sensor output mode data to buffer provided by HOST and notify HOST using **Sensor_Output_Mode_Export_Notification()** event. For each mode supported by sensor, this buffer will contain information in following structured format :-

```
typedef struct
{
  isp_uint32_t      u32_woi_res_width;
  isp_uint32_t      u32_woi_res_height;
  isp_uint32_t      u32_output_res_width;
  isp_uint32_t      u32_output_res_height;
  isp_uint32_t      u32_data_format;
  isp_uint32_t      u32_usage_restriction_bitmask;
  isp_uint32_t      u32_max_frame_rate_x100;
} Sensor_Output_Mode_ts;
```

- If HOST is interested in reading output mode configuration corresponding to one mode at a time. HOST should set value of page element **g_ReadLLAConfig_Control.u16_SelectMode** to the index corresponding to the mode of interest, and toggle coin **g_ReadLLAConfig_Control.e_Coin_ReadLLAConfigControl**. ISP FW will report information corresponding to the mode specified by HOST in read only page **g_ReadLLAConfig_Status**, and notify HOST using **Sensor_Output_Mode_Export_Notification()** event.

6.2.1 Pre-requisites

Sensor output mode information can be accessed after ISP FW has booted successfully.

6.2.1.1 Pre BOOT Parameters

None

6.2.1.2 Pre RUN Parameters

None

6.2.2 Output Modes Page Elements

g_ReadLLAConfig_Control

g_ReadLLAConfig_Status

6.2.2.1 ReadLLAConfig_Control

Page Elements	Description	Range

ptr32_Sensor_Output_Mode_Data_Address	Pointer to memory location where all available sensor streaming configuration will be copied. Please note that pointer should be converted to xP70 memory space mapping. It cannot be in ARM logical space	Complete uint32 range (0 – 4294967295)
u16_SelectMode	Select whiich configuration to read. Value must be less than u16_number_of_modes. This option should be used only when host want to retrieve sensor modes one by one and they will be available in status page elements	0 - (u16_number_of_modes-1)
e_Coin_ReadLLAConfigControl	Control coin to use the feature. Host must program a value different than e_Coin_ReadLLAConfigStatus	Coin_e_Heads Coin_e_Tails

6.2.2.2 ReadLLAConfig_Status

Page Elements	Description	Range
u32_sensor_Output_Mode_data_size	size of sensor mode data	Complete uint32 range (0 – No of output modes * size of one mode)
u16_number_of_modes	Number of modes supported by LLD	
u16_woi_res_width	Width of the WOI of the selected mode	0 – 65535
u16_woi_res_height	Height of the WOI of the selected mode	0 – 65535
u16_output_res_width	Width of the output resolution of selected mode	0 – 65535
u16_output_res_height	Height of the output resolution of selected mode	0 – 65535
u16_max_frame_rate_x100	Max frame rate of the selected mode	0 – 65535
e_Coin_ReadLLAConfigStatus	Status coin ReadLLAConfig_Status_ts becomes equal to e_Coin_ReadLLAConfigControl when values of the requested mode are populated	Coin_e_Heads Coin_e_Tails–

e_DataFormat_SensorMode	Enumeration corresponding to data format, for sensor mode indicated in g_ReadLLAConfig_Control .u16_SelectMode	RAW8: 0x0808– RAW10: 0x0A0A DPCM10_08: 0x0A08 DPCM10_06: 0x0A06

6.3 Use case specification to ISP FW

6.3.1 Overview

Under LLCD model implementation, LLCD exports certain number of sensor modes. Every mode has a corresponding usage restriction bitmask associated with it which actually indicates that, for one specific usage this mode is invalid. Client of the LLCD (ISP FW) selects one of the modes based on the use-case specified by HOST. This selection of the best sensor mode also known by SMS is done by ISP FW with minimum input from HOST. Mode control mechanism is implemented in ISP FW to enable host to participate in SMS decision where it can directly select a mode according to the usage taking into account the usage restriction bitmask of sensor modes.

Host should specify use case through user restriction bitmask.

6.3.2 Pre-requisites

Sensor output mode information can be accessed after ISP FW has booted successfully.

6.3.2.1 Pre BOOT Parameters

None

6.3.2.2 Pre RUN Parameters

Value must be programmed before RUN command.

6.3.3 Run mode control page elements

6.3.3.1 RunMode_Control

Page Elements	Description	Range
e_StreamMode	Not used	
e_SensorModeVF	Not used	
e_SensorModeCapture	Not used	
e_Flag_MechanicalShutterUsed	Not used	
e_Coin_Ctrl	Not used	
e_LLD_USAGE_MODE_usagemode	The page element is used to select sensor modes from list of modes exported by low level camera driver. For example, in Video capture from HR pipe LLD_USAGE_MODE_VIDE	/// Restriction bitmask for VF LLD_USAGE_MODE_VF , /// Restriction bitmask for AF

	O_CAPTURE should be programmed. When only VF is running from LR pipe, LLD_USAGE_MODE_VF should be programmed. When doing BMS only from BMS2, LLD_USAGE_MODE_STILL_CAPTURE should be programmed. NOTE: Only LLD_USAGE_MODE_VF, LLD_USAGE_MODE_STILL_CAPTURE, LLD_USAGE_MODE_VIDEO_CAPTURE, LLD_USAGE_MODE_HQ_VIDEO_CAPTURE and LLD_USAGE_MODE_HS_VIDEO_CAPTURE are supported. If CAM_DRV_USAGE_MODE_UNDEFINED is selected, ISP FW is free to choose any mode.	LLD_USAGE_MODE_AF , /// Restriction bitmask for STILL_CAPTURE LLD_USAGE_MODE_STILL_CAPTURE, /// Restriction bitmask for NIGHT_STILL_CAPTURE LLD_USAGE_MODE_NIGHT_STILL_CAPTURE , /// Restriction bitmask for NIGHT_STILL_CAPTURE LLD_USAGE_MODE_STILL_SEQ_CAPTURE , /// Restriction bitmask for VIDEO_CAPTURE LLD_USAGE_MODE_VIDEO_CAPTURE , /// Restriction bitmask for NIGHT_VIDEO_CAPTURE LLD_USAGE_MODE_NIGHT_VIDEO_CAPTURE , /// Restriction bitmask for HQ_VIDEO_CAPTURE LLD_USAGE_MODE_HQ_VIDEO_CAPTURE , /// Restriction bitmask for HS_VIDEO_CAPTURE LLD_USAGE_MODE_HS_VIDEO_CAPTURE , // high frame rate video ///This value ndicates that restriction bitmask is not not used by SMS CAM_DRV_USAGE_MODE_UNDEFINED = 255,

6.4 Sensor Mode Selection

6.4.1 Overview

The SMS (Sensor Mode Selection) feature has been provided to the Host to decide which sensor mode it would like to use for still capture and then program it in the Firmware. In trial mode, the Firmware will return exposure parameters (min, max and step) and frame parameters (length and readout time) to the Host. Based on the exposure parameters, the Host can re-partition exposure as per new exposure steps.

6.4.2 Description

Normally, to attain a particular exposure (and white balance) configuration, the Host applies the same onto the firmware in quantization steps, until the same get converged. While switching from LR/HR to BMS datapath, the exposure parameters converged in the LR/HR datapath are not preserved, since the settings corresponding to the chosen BMS mode are selected. If the Host is aware beforehand (before switching to BMS) the mode it has to select, then it can save a number of parameter converge cycles. The Host will then request switching to that particular BMS mode.
The framework for SMS is generic in the sense that the Host may request for the parameters anytime after the Firmware boot.

6.4.3 Usage

6.4.3.1 Pre-Requisites

- The Host is aware of the various BMS modes supported by the sensor (by querying the same).
- The Host will program requested frame rate less than or same as the Max Framerate (for that mode).
- The Host will be aware of the sensor data format to program.
- The Firmware has been booted.

6.4.3.2 Pre Boot Parameters

None

6.4.3.3 Pre Run Parameters

None

6.4.3.4 Live Parameters

All the page elements are live parameters. SMS request will take effect only after the Host toggles e_Coin_Ctrl of SMSControl Page.

6.4.3.5 Modes of Operation

None

6.4.3.6 Interface between Host and Firmware

1.1.3.6.1 Host Programming (Control Page)

The Host will set the various parameters (exposure and frame properties related parameters) in the SMS control page. The Host shall finally toggle e_Coin_Ctrl in the SMS Control Page.

PS: The Host can program this Page anytime after Firmware Boot (whether or not the Firmware is streaming).

1.1.3.6.2 Firmware Feedback (Status Page)

The Firmware will execute the SMS command, fill up the Status Page, and reset e_Coin_Status (equal it to e_Coin_Ctrl).

The Firmware also shall send SMS done notification to the Host. The notification to MMDSP is sent by setting the 29th bit of the ITM_FW_EVENT0_STATUS register. Also, the u16_EVENT0_29_SMS_DONE element of the Event_Count Page is incremented on each notification.

6.4.3.7 Pages Exposed

SMSControl

SMSStatus

Control Page for SMS	SMSControl	
Page Elements	**Description**	**Range**
u32_FrameRate_x100	Frame rate to be used in calculation. [DEFAULT]: 0	>0 to any value <= Max Frame Supported by the sensor for the chosen mode.
u16_WOI_X_size	Window Of Interest X Size [DEFAULT]: 0	Window Of Interest X Size for the chosen mode
u16_WOI_Y_size	Window Of Interest Y Size [DEFAULT]: 0	Window Of Interest Y Size for the chosen mode
u16_X_size	Output size X [DEFAULT]: 0	Output X Size for the chosen mode
u16_Y_size	Output size Y [DEFAULT]: 0	Output Y Size for the chosen mode
u16_CsiRawFormat	Raw Data Format from Sensor [DEFAULT]: RAW8: 0x0808	Any of the formats in DataFormat_te. For example RAW8:0x0808, RAW10:0x0A0A etc.
e_Coin_Ctrl	Control to trigger SMS: (Head/Tails). Host should program a different value than status PE to activate SMS [DEFAULT]: Coin_e_Heads	Coin_e_Heads (0) Coin_e_Tails (1)

Status Page for SMS	SMSStatus	
Page Elements	**Description**	**Range**
u32_LineLength_pck	Line length of frame (in pixel clocks) [DEFAULT]: 0	>= u16_X_size
u32_FrameLength_lines	Length of frame (in lines) [DEFAULT]: 0	>= u16_Y_size
u32_Min_ExposureTime_us	Minimum exposure that can be configured and achieved (in micro secs) [DEFAULT]: 0	> 0

u32_Max_ExposureTime_us	Maximum exposure that can be configured and achieved (in micro secs) [DEFAULT]: 0	This value, divided by u32_ExposureQuantizationStep_us should be an integer.
u32_ExposureQuantizationStep_us	Exposure quantization step [DEFAULT]: 0	Should be multiple of u32_Min_ExposureTime_us.
u32_ActiveData_ReadoutTime_us	Active Frame Data Readout time (in micro secs) [DEFAULT]: 0	This value, divided by u32_ExposureQuantizationStep_us should be = u16_Y_size.
e_Coin_Status	Status Coin: (Head/Tails) [DEFAULT]: Coin_e_Heads	Coin_e_Heads (0) Coin_e_Tails (1)

7. Error Handler

7.1 Error Handler

7.1.1 Overview

The Error Handler subsystem is responsible for managing Error and EOF events generated by IPP Frontend. This module monitors the errors, and may be enabled to take appropriate response of aborting or recovering from the types of errors encountered.

7.1.2 Description

Error Handler subsystem is responsible for following:

- **ErrorEOF_ISR**: This is Line 27 ISR which is trigger for Error Handler module to process and update required workflow. The interrupt count for this ISR indicates total number of errors encountered.
- **Abort**: Abort streaming and go to Error state, based on the Error identified by IPP.
- **Recover**: Try to recover from the Error by resetting the Rx Pipe and resume streaming. This is transparent to the Host.
- **Counters**: The module maintains several counters which indicate which errors were encountered

7.1.3 Usage

7.1.3.1 Pre-Requisites

The mandatory requirement is that **e_Error_Control** is set to desired error recovery mode. This can be changed anytime, however by default it is disabled. First level of enabling doesn't abort or recover any error.

7.1.3.2 Pre Boot Parameters

7.1.3.3 Pre Run Parameters

7.1.3.4 Live Parameters

All the page elements are live parameters.

7.1.3.5 Modes of Operation

e_Error_Control sets the Mode of operation. There are four –

- **Disabled**: In this state Error Handler will not take any action. All page elements will be in default value of zero.
- **Enabled**: This will only report errors by incrementing counters, but not take any action on the same.
- **Abort**: This mode will abort the streaming and enter Error state, if Front End errors are encountered.
- **Recover**: This mode will attempt recovery by Smia Rx force-reset for certain class of errors. For non-recoverable errors, state machine will follow abort flow.

7.1.3.6 Interface between Host and Firmware

1.1.3.6.1 Host to Firmware (Mode)

The Host gathers information regarding Error Handler module by following elements

- **Mode** – The mode of error recovery. This can be changed runtime but not recommended.
- **Counter**: This is only to report errors. FW will reset them if recovery is attempted. Host can also modify them to trigger corresponding actions.
- **Flags**: These can be used to trigger soft-reset directly, however ErrorEOF_ISR manages these flags internally.

1.1.3.6.2 Firmware to Host (Abort Events and Counters)

Firmware will notify the host about abort event via **Grab_Abort** (**3_2**) event, and incrementing corresponding counters. If error is recoverable, Firmware will reset the error counts and will not notify the host.

7.1.3.7 ErrorHandler use case

The key use case for ErrorHandler module is to be aware of FrontEnd error's occurring during different modes in regression testing. The host can change the mode for Firmware to recover from these errors on its own, else host is responsible to recover the Firmware State machine, including resetting these errors.

The Grab Abort flow is shown in (Figure 1). Taken from "Error Handling for v1.4 MBL SAS.pdf" section 17.2.14, Figure 3

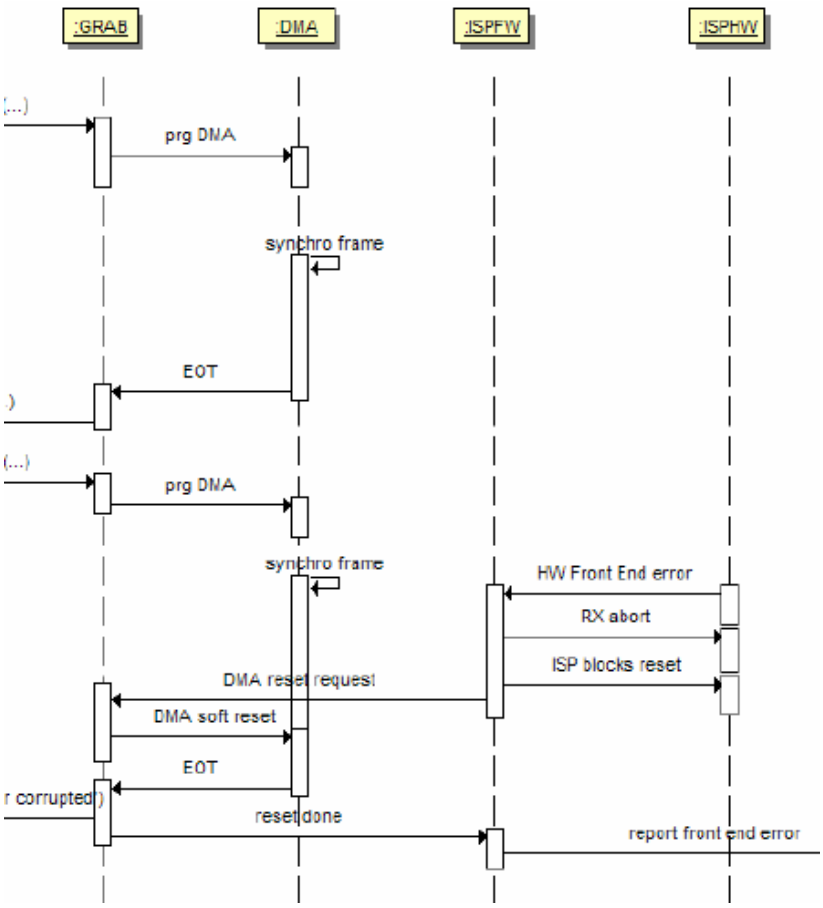


Figure (1) Front End Error Recovery

Following errors are handled by **ErrorEOF_ISR**

- **IPP_CSI2_PACKET_ERROR_ITS** – Bits **0:3**
- **IPP_SD_ERROR_ITS** – **Bit 10**, and **0:2**

In Abort mode all **SD_ERRORS** send the Device to Abort state, and in Recover mode a recovery is attempted without abort.

7.1.3.8 Error Management

In Enabled mode, only counters are incremented but no action is taken on errors. For other modes either Abort or Recovery path is followed.

7.1.3.9 References

Please refer to following documentation for details of types of error handled.

- **106750** - Front end error recovery follows specification mentioned in Error Handling for v1.4 MBL SAS.pdf, section 17.2.14
- Section STn8500V2_SIA_HW_specification_v1.1.pdf - Section 1.9.15 on Interrupts Table 10, Section 1.13.6 - IPP_CSI2_PACKET_ERROR*, IPP_CCP_SYNC_*_ERROR, IPP_SD_ERROR_*, IPP_EVT_ITM

7.1.3.10 Implementation

- Added INTR ISR ERROR_EOF 27, default Interrupt is Enabled (XINTR4)
- Added New Grab_Abort Event
- Modified StateMachine (HostInterface.c) to abort/recover streaming in case of different SD_ERROR.

- Modified StateMachine (Stream.c) For forcing RxAbort in case of Frontend Errors
- Added **Error Handler Module** – for ISR processing and New Page for Error events.

7.1.3.11 Pages Exposed

Error Handler

Control Page for Flash	FlashControl	
Page Elements	Description	Range
u16_CSI_EOF_Counter	Total Number of CSI End of Frames [DEFAULT]: 0	0 to 2^16-1
e_Error_Control	Primary Control to enable/disable the Error Handler [DEFAULT]: 0 Error_DISABLED 1: Enabled 2: Abort 3: Recover + Abort for different types of errors	0 to 3
e_Flag_Error_Abort	Flag to Abort streaming [DEFAULT]: 0 Flag_e_FALSE	0 and 1
e_Flag_Error_Abort_Recover	Flag to force Smia Rx re-start [DEFAULT]: 0 Flag_e_FALSE	0 and 1
u8_CSI_Error_Counter	Total Number of CSI Errors [DEFAULT]: 0	0 to 256
u8_CCP_Error_Counter	Total Number of CCP Errors [DEFAULT]: 0	0 to 256

8. CRM

8.1 CRM

8.1.1 Overview

The CRM firmware module is responsible for programming CRM (Clock and reset manager) hardware block. HOST can read configuration of CRM using g_CRM_Status page, it is a read only page.

8.1.2 Pages Exposed

g_CRM_Status

g_CRM_Status		
Page Elements	Description	Range
u16_Window	Window of detection	uint16_t range
u16_interrupt_0_CKC_OK_ITS	Count of occurrences of clock_ok interrupt	uint16_t range
u16_interrupt_0_CKC_LOST_ITS	Count of occurrences of clock_loss interrupt	uint16_t range
u16_interrupt_0_CKC_OK_3D_ITS	Count of occurrences of clock_ok_3d interrupt	uint16_t range
u16_interrupt_0_CKC_LOST_3D_ITS	Count of occurrences of clock_loss_3d interrupt	uint16_t range
u16_interrupt_0_CKC_OK_ITS_warning	Count of invalid occurrences of clock_ok interrupt.	uint16_t range
u16_interrupt_0_CKC_LOST_ITS_warning	Count of invalid occurrences of clock_loss interrupt.	uint16_t range
e_Flag_StartEnabled	This flag specifies whether a start sequence is enabled or not.	flag_e_TRUE, flag_e_FALSE
e_Flag_StopEnabled	This flag specifies whether a start sequence is enabled or not.	flag_e_TRUE, flag_e_FALSE
e_Flag_SensorClocksAvailable	Specifies if the sensor clocks are available	flag_e_TRUE, flag_e_FALSE

9. Data Rate Control

9.1 Video Timing module

9.1.1 Video Timing Overview

Video Timing module in ISP FW is used to control sensor data format and data rate setting from the sensor.

9.1.2 Video Timing Usage

The module is calculates clock multiplier and divider values to be programmed in the sensor and its inputs are:

1. Host Data rate supported at CCP interface

9.1.2.1 Pre-requisites

Host should know following parameters to access the module:

1. Host Data rate
2. Data clock / Data strobe mode
3. Data format

9.1.2.2 Pre BOOT parameters

1. Host data rate
2. Data clock/ data strobe mode
3. Data format

9.1.2.3 Pre RUN parameters

NA

9.1.2.4 Live parameters

NA

9.1.2.5 Video Timing Host Inputs

VideoTimingHostInputs	Video Timing control structure (Read/Write)	
Page Elements	**Description**	**Range**
f_HostRxMaxDataRate_Mbps	specifies the maximum host data rate (in Mbps)	
f_OutputClockDeratingFraction	Specifies the target derating factor as a fraction of the maximum possible de-rating i.e. target_de-rating_factor = maximum_possible_de-rating_factor * f_OutputClockDeratingFraction	Range [0.0 to Max possible] with 0.0-> No de-rating support
u16_CsiRawFormat	csi_raw_format's MSB represents sensors data format and LSB denote tranmit format. E.g. in RAW8 compressed format, the value is 0x0A08 means Sensor is giving 10 bit(0x0A) data in 8 bit (0x08) format	RAW8: 0x0808 RAW10: 0x0A0A For more details on data format, refer SMIA_Functional_specification_1.0.pdf
e_VideoTimingMode_Mode	Video Timing Mode: Manual or Automatic. [DEFAULT]: 'VideoTimingMode_e_VideoTimingMode_Automatic' The host has to provide a few inputs to the video timing block for it to automatically (automatic mode) compute the video timing parameters. These values may be programmed at any point before issuing a streaming command	'VideoTimingMode_e_VideoTimingMode_Manual'(0) 'VideoTimingMode_e_VideoTimingMode_Automatic'(1)
e_SensorBitsPerSystemClock_DataType	Number of data bits per sensor clock [DEFAULT]:	'SensorBitsPerSystemClock_e_DATA_CLOCK'(1)

	'SensorBitsPerSystemClock_e_DATA_STROBE'(2)	'SensorBitsPerSystemClock_e_DATA_STROBE'(2)
e_DeratingRoundingMode_RoundingMode	pecifies the output clock de-rating rounding mode. DEFAULT] ROUND_UP then FIFO requirements are low since theoutput clock is faster then the average scaler output data rate f derating rounding mode == ROUND_CLOSEST then the error between the target de-rating factor and actual de-rating factor is the minimum.	'DeratingRoundingMode_e_ROUND_UP'(0) 'DeratingRoundingMode_e_ROUND_CLOSEST'(1)
e_Flag_DerateVideoTimingClockForProfileZero	Specifies if the video timing clock de-rating is required for profile 0 sensor Any sub sampling can then be exploited to slow down the video timing clock to reduce power consumption. [DEFAULT]: 'Flag_e_FALSE'	'Flag_e_FALSE'(0) 'Flag_e_TRUE'(1)
e_Polarity_VsyncPolarity	Specifies the vsync polarity, Input only valid for parallel sensor, Don't care for serial sensor	Polarity_e_Polarity_ActiveLow(0) Polarity_e_Polarity_ActiveHigh(1)
e_Polarity_HsyncPolarity	Specifies the hsync polarity, Input only valid for parallel sensor, Don't care for serial sensor	'Flag_e_FALSE'(0) 'Flag_e_TRUE'(1)

9.2 Frame rate module

9.2.1 Frame rate Overview

Frame rate module in ISP firmware receives frame rate from host and apply it to the sensor. There are two different page element interfaces for controlling frame rate, each of which can be used independently for controlling frame rate. HOST can use VariableFrameRateControl.e_Flag page elemnt for selecting frame rate control interface to be used. These two interfaces are:-

9.2.1.1.1 VariableFrameRateControl page element interface

For using this interface VariableFrameRateControl.e_Flag must be set to TRUE before RUN command. In this case framerate is controlled by VariableFrameRateControl page. Host must set value of maximum framerate using VariableFrameRateControl.f_MaximumFrameRate_Hz PE before issuing RUN command. Actual framerate is controlled by g_VariableFrameRateControl.f_CurrentFrameRate_Hz PE, this PE can be set at any time after boot. If ISP FW is in streaming state HOST must toggle system coin after modifying value of g_VariableFrameRateControl.f_CurrentFrameRate_Hz. f_MaximumFrameRate_Hz puts upper bound on the value of framerate. if value of VariableFrameRateStatus.f_MaximumFrameRate_Hz set by host is higher than what ISP FW can achieve, ISP FW goes into ERROR state.

9.2.1.1.2 FrameRateControl page element interface

For using this interface VariableFrameRateControl.e_Flag must be set to FALSE before RUN command. In this case framerate is controlled by g_FrameRateControl page. After RUN command, ISP FW checks if requested frame rate can be achieved. If not, it returns ERROR and goes in error state. In a running system, if host try to increase frame rate that is not possible in the mode selected, ISP FW do not return error, it apply and report the max achievable in that mode.

9.2.2 Frame rate Usage

9.2.2.1 Pre-requisites

NA

9.2.2.2 Pre BOOT parameters

NA

9.2.2.3 Pre RUN parameters

VariableFrameRateControl.e_Flag must be set by HOST before issuing RUN command. If this flag is set to TRUE, VariableFrameRateControl will be used else FrameRateControl. f_UserMaximumFrameRate_Hz PE interface will be used for frame rate control. Other pre-run paremeters are :-

If VariableFrameRateControl.e_Flag is set to TRUE

1. VariableFrameRateControl.f_CurrentFrameRate_Hz
2. VariableFrameRateControl.f_MaximumFrameRate_Hz

else

1. FrameRateControl. f_UserMaximumFrameRate_Hz

9.2.2.4 Live parameters

If VariableFrameRateControl.e_Flag is set to TRUE, VariableFrameRateControl.f_CurrentFrameRate_Hz else FrameRateControl. f_UserMaximumFrameRate_Hz. System coin must be toggled after modifying live parameters.

9.2.3 Frame rate control page elements

Control Structure for Frame rate	**VariableFrameRateControl**	
Page Elements	**Description**	**Range**
f_CurrentFrameRate_Hz	Target frame rate desired by HOST	+ve float
f_MaximumFrameRate_Hz	Maximum framerate requested by HOST	+ve float
e_Flag	Control flag for enabling/disabling Variable frame rate control DEFAULT : Flag_e_FALSE	Flag_e_FALSE, Flag_e_TRUE

Control Structure for Frame rate	**VariableFrameRateStatus**	
Page Elements	**Description**	**Range**
f_CurrentFrameRate_Hz	Frame rate set by ISP FW	+ve float
f_MaximumFrameRate_Hz	Status of maximum framerate requested by HOST	+ve float
e_Flag	Staus flag showing enable/disable statis for Variable frame rate control DEFAULT : Flag_e_FALSE	Flag_e_FALSE, Flag_e_TRUE

Control Structure for Frame rate	FrameRateControl	
Page Elements	**Description**	**Range**
f_UserMaximumFrameRate_Hz	Target frame rate	

9.2.4 Default settings recommendation

NA

9.3 Frame Dimension Manager

9.3.1 Frame Dimension Manager Overview

The responsibility of Frame Dimension Manager is to provide the top level firmware modules a simplified view of the sensor frame dimension management. It is responsible for programming all the parameters into the sensor with respect to the sensor frame dimensions, sub-sampling factor and scaling factor. It also exposes these parameters to the host.

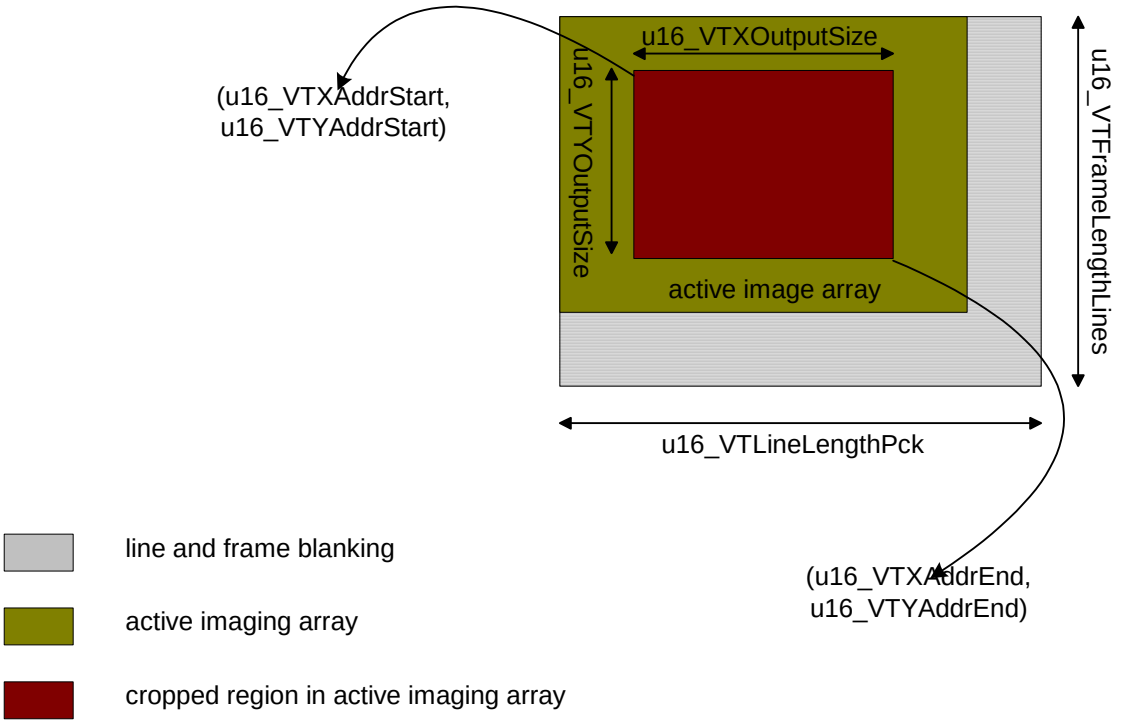
- Pre-requisites and recommendations for Using Frame Dimension Manager
  - Frame Dimension Manager provides negligible control to the user. It takes inputs from Zoom Manager and then produces the necessary and desired output.
  - Sensor registers are read from the sensor at BOOT time. These registers specify the maximum and minimum values of parameters related to frame dimension. It may be noted that Host can modify the parameters exposed by the sensor by programming the corresponding element in the SensorFrameConstraints page. This must be done only after Boot command has been issued. This ensures that when the host has overridden the default sensor frame constraints elements, they have already been fetched from the sensor and that they will not be subsequently overwritten by the device.

9.3.2 Frame Dimension Manager Description

The sensor timing domain can be divided in two parts:

- Video Timing Domain(VT): This deals with data read out from the pixel array of the sensor.
- Output Domain(OP): This deals with data transfer from the sensor to the ISP firmware via CCP interface.

The following diagram shows the frame dimension parameters in the Video Timing Domain of the sensor.



9.3.2.1 Video Timing Frame Length

This corresponds to actual video timing frame length parameter as defined by the SMIA specifications. The application of frame length must be re-timed internally to the start of the frame boundary.

9.3.2.2 Video Timing Line Length

This corresponds to actual video timing line length parameter as defined by the SMIA specifications. The application of line length must be re-timed internally to the start of the frame boundary.

9.3.2.3 Region Of Interest

The region of interest in the sensor imaging array is defined by the video-timing read out co-ordinates in the sensor. It is calculated by the frame dimension manager on the basis of a given reduction factor(*scaling_factor* * *sub-sampling_factor*) and target/requested output size.

9.3.2.4 Video Timing X Output Size

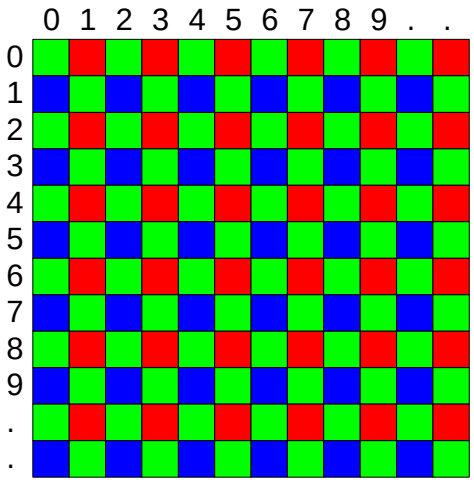
There is no mention of a video timing X output size in the SMIA specifications. The video timing X output size, here, refers to the number of pixels read out per line from *u16_VTXAddrStart* to *u16_VTXAddrEnd* at the current video timing sub-sampling factor.

9.3.2.5 Video Timing Y Output Size

There is no mention of a video timing Y output size in the SMIA specifications. The video timing Y output size, here, refers to the number of rows read out from *u16_VTYAddrStart to u16_VTYAddrEnd* at the current video timing sub-sampling factor.

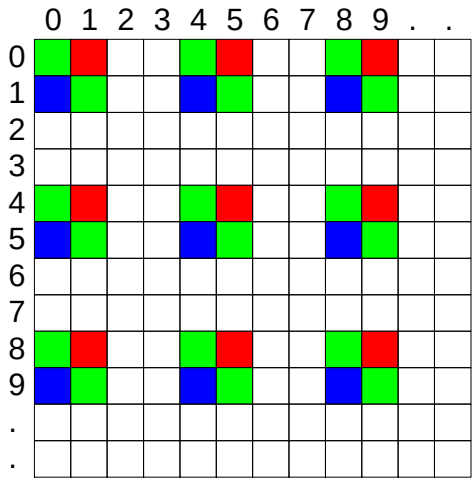
9.3.2.6 Sensor Sub Sampling

The device can use the sub-sampling capabilities of the sensor. Sub sampling factor in the sensor is specified in terms of number of locations to skip in the readout process along the horizontal and vertical directions. The following diagram depicts the sub- sampled read out logic.



x_even_inc = 1 y_even_inc = 1
x_odd_inc = 1 y_odd_inc = 1

readout with no sub sampling



x_even_inc = 1 y_even_inc = 1
x_odd_inc = 3 y_odd_inc = 3

x sub sampling factor = 2
y sub sampling factor = 2

Note: No colour at a grid location indicates that the corresponding pixel is not read out

9.3.2.7 Sensor Scaling

The device can utilize the sensor scaling capabilities. Scaling factor in the sensor is defined as the ratio of two integers:

Sensor_scale_factor = scale_m/scale_n

Where *scale_n* = 16

Please note that a basic assumption that is being made is that the device scales by the same factor in horizontal and vertical directions.

9.3.2.8 Output Timing X/Y Output Size

The sensor sub sampling operation happens in the video timing domain. Hence, in this case the output timing X/Y output size is equal to video timing X output size.

The sensor scaling operation happens in the output timing domain. Hence, in this case the output timing X/Y output size is different and is given by:

U16_OPXOutputSize = u16_VTXOutputSize/sensor_scale_factor

U16_OPYOutputSize = u16_VTYOutputSize/sensor_scale_factor

9.3.2.9 Impact of Anti Flicker

If the sensor integrates in multiples of flicker free period, then all the lines in the sensor will receive the same amount of light during their integration cycle. So it must be guaranteed that the video-timing frame length of the sensor is as close as possible to a flicker free period multiple.

9.3.3 Frame Dimension access usage

9.3.3.1 Pre-requisites

3. The Zoom Manager should be running so that it could give the requirements with regards to frame dimensions to the Frame Dimension Manager.
4. The Binning manager should provide details like whether binning is enabled and minimum video timing line length of the sensor when binning is enabled.
5. The Sensor Manager must provide all the sensor specific details.

6. Video Timing module should provide information like: sensor bits per system clock, ccp raw format, output timing clock derating and video timing pixel clock derating.
7. Information like: round up bunch fudge, coarse integration time max margin, digital gains floor should come from Exposure.

9.3.3.2 Pre BOOT parameters

None

9.3.3.3 Pre RUN parameters

Following parameters should be programmed before RUN command:

1. Sensor Constraints.
2. Data Clock/Strobe Mode
3. Data Format.
4. Pixel clock period from Video Timing.

9.3.3.4 Live parameters

None of the frame dimension pages should be changed while streaming.

9.3.4 Frame Dimension Manager page elements

9.3.4.1 HostFrameConstraints

HostFrameConstraints	Constraints to be applied in the output timing domain	
Page Elements	**Description**	**Range**
f_FOVMargin	Specifies the margin in field of view that will be maintained at each frame config request. Basically provisioned_fov = requested_fov * f_FOVMargin f_FOVMargin should be greater than 1.0	Should be greater than or equal to 1.0
u8_MinimumPostScalar0LineBlanking_pixels	Minimum interline pixel clocks required by the host beyond the GPS0.	
u8_MinimumPostScalar1LineBlanking_pixels	Minimum interline pixel clocks required by the host beyond the GPS1.	
u8_MinimumInterFrame_lines	Minimum interframe lines to be maintained at the sensor output.	Flag_e_TRUE(1) Flag_e_FALSE(0)
u8_MaximumPreScale	Maximum pre scale to be used in the sensor	
u8_MainsFrequency_Hz	AC Frequency - used for flicker free time period calculations. This mains frequency determines the flicker free time period.	
e_Flag_GuaranteeStaticFlickerFrameLength	If this flag is set to TRUE, then it is ensured that the frame length will always have an integer number of flicker free bunches.	
e_FrameDimensionProgMode	Specifies whether the host will program CurrentFrameDimension page, which will be used for Rx Test pattern streaming and also for BML operations, or the firmware will calculate the values.	

	Must be set to FrameDimensionProgMode_e_Manual for BML operations and to FrameDimensionProgMode_e_Auto for normal sensor streaming.	
e_Flag_AllowChangeOver	Specifies whether a sensor changeover (reprogram) is allowed or not	

9.3.4.2 SensorFrameConstraints

SensorFrameConstraints	**Sensor constraints are read at BOOT time (Read/Write)**	
Page Elements	**Description**	**Range**
u16_VTXAddrMin	Minimum value of X coordinate of the top left corner of Window Of Interest in Full Field of View.	Sensor dependant
u16_VTYAddrMin	Minimum value of Y coordinate of the top left corner of Window Of Interest in Full Field of View.	Sensor dependant
u16_VTXAddrMax	Maximum value of X coordinate of the bottom right corner of Window Of Interest in Full Field of View.	Sensor dependant
u16_VTYAddrMax	Maximum value of Y coordinate of the bottom right corner of Window Of Interest in Full Field of View.	Sensor dependant
u16_MinOPXOutputSize	Minimum OP X output size.	Sensor dependant
u16_MinOPYOutputSize	Minimum OP Y output size.	Sensor dependant
u16_MaxOPXOutputSize	Maximum OP X output size.	Sensor dependant
u16_MaxOPYOutputSize	Maximum OP Y output size.	Sensor dependant
u16_MinVTFrameLengthLines	Minimum value of video timing frame length (in lines).	Sensor dependant
u16_MaxVTFrameLengthLines	Maximum value of video timing frame length (in lines).	Sensor dependant
u16_MinVTLineLengthPck	Minimum value of video timing line length (in pixel clocks).	Sensor dependant
u16_MaxVTLineLengthPck	Maximum value of video timing line length (in pixel clocks).	Sensor dependant
u16_MinVTLineBlankingPck	Minimum value of video timing line blanking (in pixel clocks).	Sensor dependant
u16_MinVTFrameBlanking	Minimum value of video timing frame blanking (in	Sensor dependant

	lines).	
u16_ScalerMMin	Minimum value of scaler_m register.	Sensor dependant
u16_ScalerMMax	Maximum value of scaler_m register.	Sensor dependant
u16_MaxOddInc	Maximum value of odd inc value.	Sensor dependant
e_SensorProfile	Specifies the sensor scaling mode (None, Horizonatal only and Full).	SensorProfile_e_Profile0 SensorProfile_e_Profile1 SensorProfile_e_Profile1

9.3.4.3 CurrentFrameDimensions

CurrentFrameDimension	**Frame Dimension Read-Only structure**	
Page Elements	**Description**	**Range**
f_PreScaleFactor[7]	Since the sensor supports either scaling or sub-sampling at any point of time, this page element contains the scaling/sub-sampling factor depending upon whichever mode the sensor is in.	In case of sub-sampling, the valid values are: 1,2,3,4… In case of scaling it can range from [1.0 to 7.75]
u16_VTFrameLengthLines	Video Timing frame length in lines.	Sensor dependant
u16_VTLineLenthPck	Video timing line length in pixel clocks	Sensor dependant
u16_VTXAddrStart	X co-ordinate of the top left corner of Window of Interest in full Field Of View	Sensor dependant
u16_VTYAddrStart	Y co-ordinate of the top left corner of Window of Interest in full Field Of View	Sensor dependant

[7] Only sensor sub sampling supported at the moment

u16_VTXAddrEnd	X co-ordinate of the bottom right corner of Window of Interest in full Field Of View	Sensor dependant
u16_VTYAddrEnd	Y co-ordinate of the bottom right corner of Window of Interest in full Field Of View	Sensor dependant
u16_OPXOutputSize	Horizontal size of Window Of Interest in output timing domain	Sensor dependant
u16_OPYOutputSize	Vertical size of Window Of Interest in output timing domain	Sensor dependant
u16_VTXOutputSize	Horizontal size of Window Of Interest in video timing domain	
u16_VTYOutputSize	Vertical size of Window Of Interest in video timing domain	
u16_XOddInc	x_odd_inc value corresponding to the horizontal sub sampling factor	The valid values are: 1,3,5,7
u16_YOddInc	y_odd_inc value corresponding to the vertical sub sampling factor	The valid values are: 1,3,5,7
u16_Scaler_M	Specifies the scale_m factor with respect to the current f_ScaleFactor in case of sensor scaling.	Sensor dependant
u16_NumberOfNonActiveColumnsAtTheLeftEdge	Specifies the number of non active columns at the left edge of the frame.	Sensor dependant.
u16_NumberOfNonActiveColumnsAtTheRightEdge	Specifies the number of non active columns at the right edge of the frame.	Sensor dependant
u16_NumberofNonActiveLinesAtTopEdge	Specifies the number of non active lines at the top edge of the the frame.	Sensor dependant
u16_NumberofNonActiveLinesAtBottomEdge	Specifies the number of non active lines at the bottom edge of the frame.	Sensor dependant
u8_NumberOfStatusLines	Specifies the number of status lines in a frame.	Sensor dependant

e_SensorPrescaleType	Specifies whether sensor is scaling or sub sampling	SensorPrescaleType_e_SensorSubsample[0] SensorPrescaleType_e_SensorScale[1]
e_SensorScalingMode	Specifies the sensor scaling mode. No scaling support is required for SMIA profile 0 sensor. Profile 1 sensor supports only horizontal scaling. Profile 2 sensor supports both horizontal and vertical scaling	FDSensorScalingMode_e_SENSOR_SCALING_NONE [0] FDSensorScalingMode_e_VIDEOTIMING_SENSOR_SCALING_HORIZONTAL_ONLY[1] FDSensorScalingMode_e_VIDEOTIMING_SENSOR_SCALING_HORIZONTAL_AND_VERTICAL[2]

9.3.4.4 FrameDimensionStatus

FrameDimensionStatus	**Status page for frame timings**	
Page Elements	**Description**	**Range**
f_VTLineLength_us	Current video timing line length of the sensor is made available in this page element. The value is valid only when the device is streaming. The unit in which it is expressed is micro seconds.	
f_VTFrameLength_us	Current video timing frame length of the sensor is made available in this page element. This element is valid only when the device is streaming. The unit used is micro seconds.	
f_CurrentFrameRate	Current device frame rate is made available in this page element. f_CurrentFrameRate = 1/(f_VTFrameLength_us * 1000000)	
u16_VTFrameLengthPending_Lines	Specifies the video timing frame length (in number of lines) that has to be applied to the sensor.	
u16_MinVTLineLengthAtCurrentVTXSize_Pixels	While streaming, the minimum video timing line length at a given video timing X output size is made available in this page element.	
u16_MinVTFrameLengthAtCurrentVTYSize_Lines	While streaming, the minimum video timing frame length at a given video timing Y output size for a sensor is made available here.	
u16_MaximumUsableSensorFOVX	Specifies the maximum X FOV that the sensor can provide at the current streaming parameters.	
u16_MaximumUsableSensorFOVY	Specifies the maximum Y FOV that the sensor can provide at the current streaming parameters.	

e_Flag_IsFrameDimensionChangePending	Specified whether frame dimension change is pending. Frame dimension parameters are committed to the sensor only when this flag is TRUE.	Flag_e_TRUE(1) Flag_e_FALSE(0)
e_Flag_IsFrameLengthChangeInhibitedForCoarseExposure	This flag indicates whether a frame length change was inhibited due to insufficient frame length for coarse exposure.	Flag_e_TRUE(1) Flag_e_FALSE(0)

9.3.4.5 AntiFlicker_Status

AntiFlicker_Status	**Status page for Anti Flicker**	
Page Elements	**Description**	**Range**
f_FlickerFreePeriod_us	Specifies the flicker free period.	
f_GainedFlickerFreeTimePeriod_us	Specifies the gained flicker free period. It takes into account the minimum analog and digital gain values.	
u16_MaxFlickerFreeBunches	Specifies the maximum number of flicker free bunches possible at the current vt frame length.	

9.3.5 Default settings recommendation

None

10. Zoom

10.1 Digital Zoom

The zoom module is responsible for maintaining the user field of view and the center of the user field of view in the sensor's imaging array based on user inputs.

The following diagram shows the actual zoom methodology for a UXGA input image size (i.e. sensor output) and VGA device output.

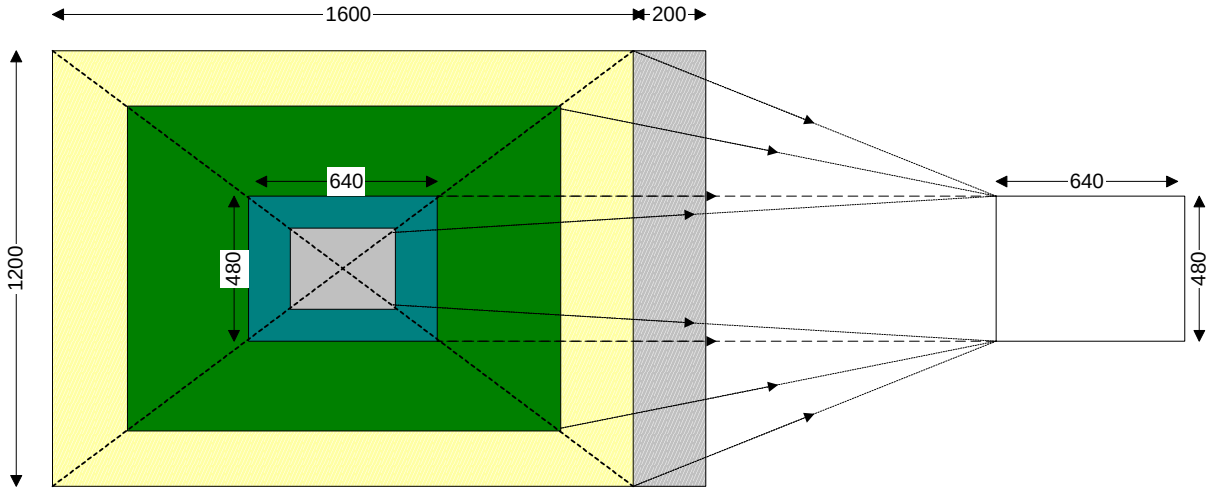


Figure 18 Zoom Methodology

For the sake of convenience only the active image areas have been considered (i.e. no SOF, Line Start and Line End codes are shown).

It may be noted from the above diagram, that the zoom operation consists of cropping into the full field of view to obtain the current window of interest. Depending upon various aspects, the crop operation may be done in the sensor or in the device. The cropped window of interest is then subsequently downscaled/ upscaled to the required output resolution.

10.1.1 Zoom Methodology

10.1.1.1 Zoom through Downscale

It is clear from the Figure 18 that a zoom in operation involves reduction in the window of interest. All zoom-in operations while the window of interest is larger than the required output image resolution is achieved by cropping into the image active area (to reduce the Window of Interest) and then performing a downscale of the cropped region to the required output resolution (VGA in our case). Converging arrows into the output image signify a downscale operation. The reader will note that any downscaling operation will result in a loss of pixels and hence the portion of the imaging pipeline after the scalar block will always have enough bandwidth to transmit the scalar output.

As a limiting case, when the zoom window of interest becomes equal to the output image resolution, no (up/down) scaling is performed. Only a crop operation is needed.

Both pipe0 and pipe1 hardware have downscaling capabilities and hence can perform zoom in through downscale.

10.1.1.2 Zoom through Upscale

Any zoom in operation beyond the limiting case explained in the previous section will result in a zoom window of interest that is smaller than the required pipe output resolution. This means that in order to achieve the required output resolution, the cropped window of interest will have to be up scaled to the required output image resolution.

Given the fact that the device design is based on a streaming architecture with no frame store, there are inherent limitations with respect to upscaling operations. In a streaming architecture, an upscale operation will introduce more pixels in the image stream. The scalar block uses interline blanking period to “insert” new pixels into the pixel stream. Hence it is necessary that there are enough interline blanking pixel clocks to accommodate new pixels.

10.1.2 Pipe Output While Updating Zoom Parameters

Whenever the device receives a zoom command, before updating the pipe with the corresponding zoom parameters, the pipe output is disabled and as soon as all the active pipes have been updated, the pipe output is enabled again. The pipe enable and disable is aligned to the start of frame. This ensures that the pipe programming will never be inconsistent. However, if the zoom command is for change of output image resolution of the active pipes, the pipe is only enabled when the host allows the device to do so.

10.1.3 Digital Zoom Notifications

The following table shows the notifications are issued by the zoom module.

Notification	Description
ZOOM_CONFIG_REQUEST_DENIED	Is raised in context of a zoom command for any of the following cases: - the required line length for upscaling > max sensor line length - max possible frame rate at the specified streaming params (i.e. sensor data rate, FOV and maximum allowable sub sampling factor) is less than the maximum desired fame rate. - a sensor change over is required for the zoom step, but has been inhibited by the host
ZOOM_CONFIG_REPROGRAM_REQUIRED	Is raised in context of a zoom command and signifies that a sensor change over was required for the zoom step and also allowed by the host
ZOOM_STEP_COMPLETE	Is raised in context of a zoom command and signifies that the last zoom step was completed by the device. After this point, it is safe for the host to issue another zoom command. However, it does not necessarily mean that the zoom step was successfully completed.
ZOOM_SET_OUT_OF_RANGE	It is raised in context of a setcenter command and signifies that the amount of horizontal or vertical offset applied to the array center was too big to be accommodated at the current field of view.

Table 9 Zoom Notifications

The following diagram shows the relative order in which the zoom notifications can be issued by the device:

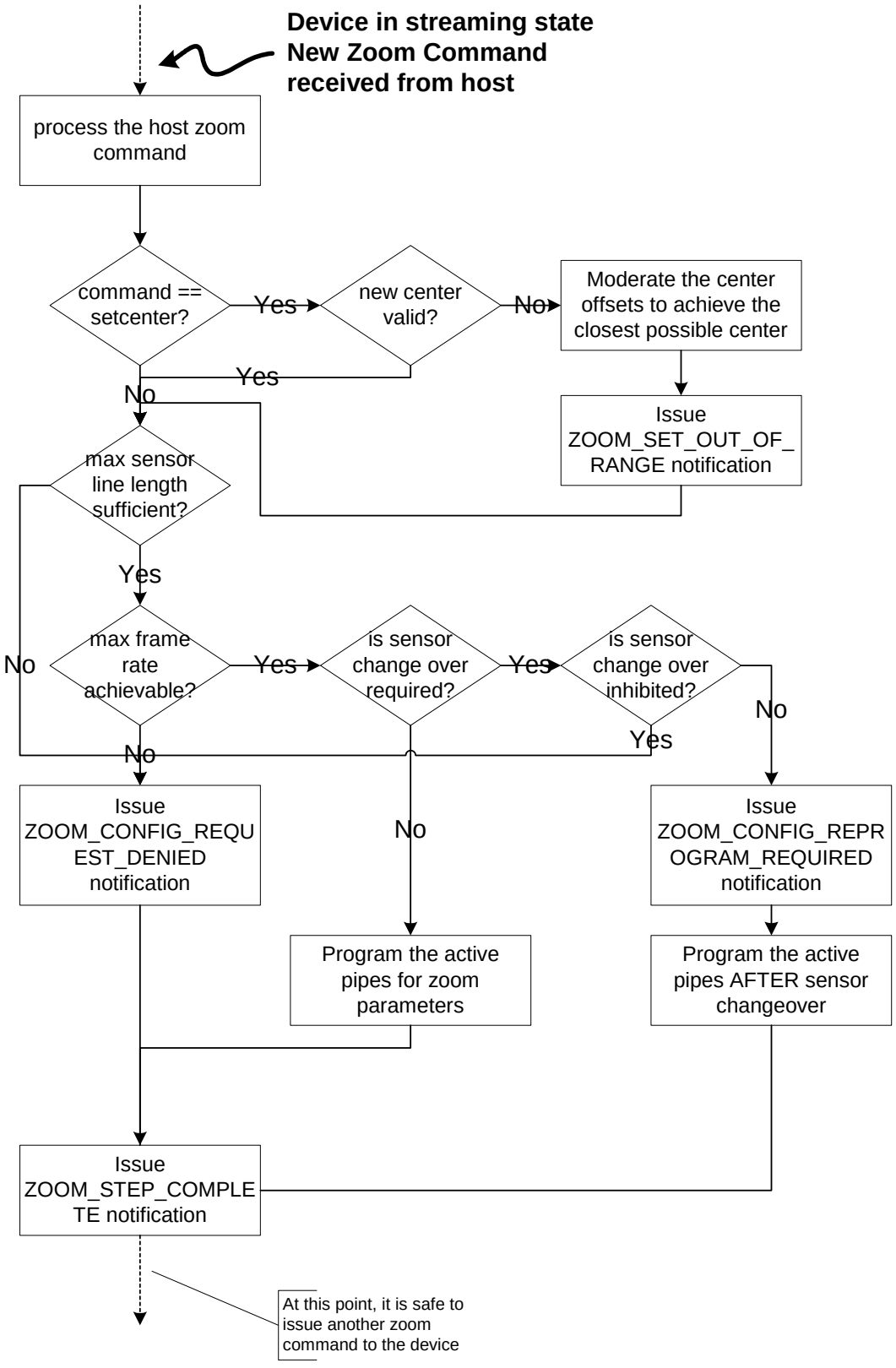


Figure 19 Zoom Notifications

10.1.4 Digital Zoom Commands

Currently the following digital zoom operations are possible:

10.1.4.1 ZoomCommand_e_SetCenter

It is used to set the center of the user field of view to be different from that of the center of the imaging array. This can be used to achieve a digital pan and tilt effect. The center of the user filed of view is specified as signed offset to the center of the imaging array. Figure 20 depicts the sequence of steps to be followed by the host to issue a command to set a new user field of view center. It is extremely important that whenever a set center command is issued, the host does not change the desired field of view in the device i.e. the device assumes the field of view to remain constant across set center command.

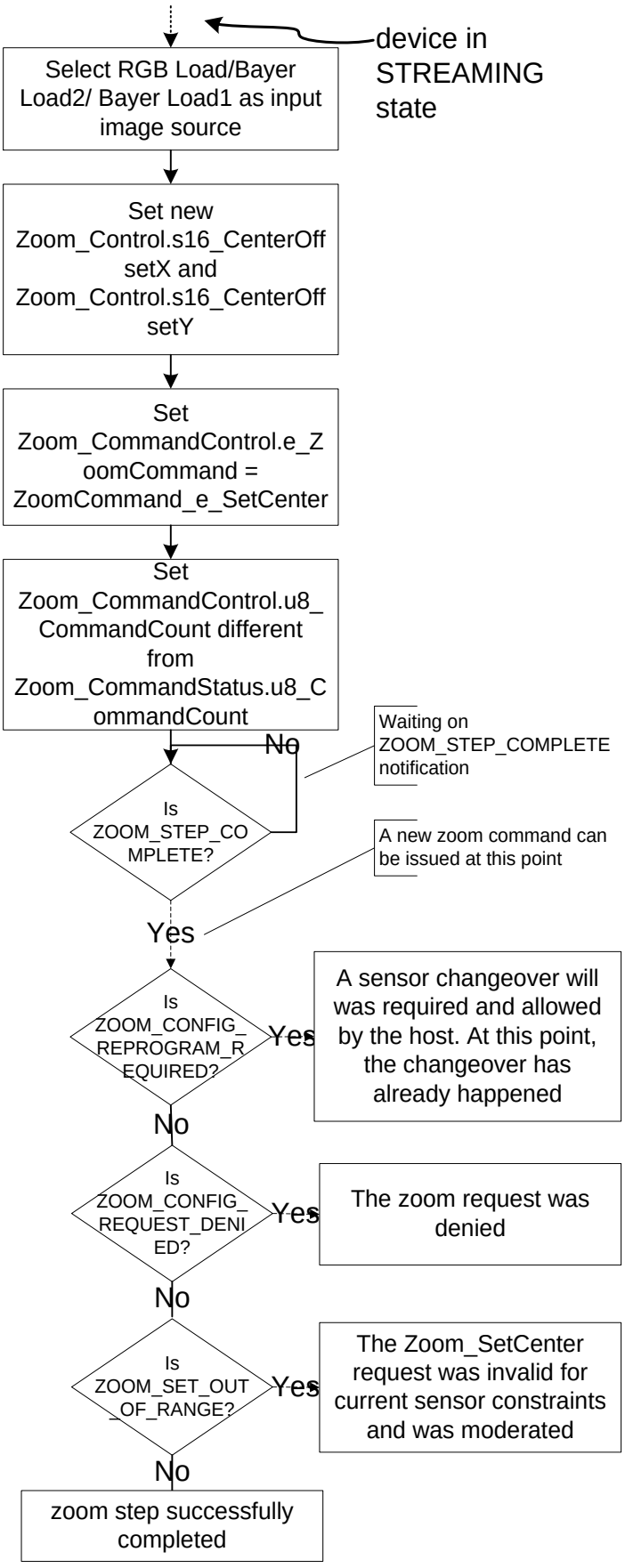


Figure 20 Zoom Command: Set Center

10.1.4.2 ZoomCommand_e_SetFOV

It is used to set a new user field of view. If the host specifies a field of view that is more than the maximum field of view available to the device then the desired field of view is clipped to the maximum field of view available to the device. Figure 21 depicts the sequence of steps to be followed by the host to issue a command to set a new field of view. It is extremely important that whenever a set field of view command is issued, the host does not change the center offsets in device i.e. the device assumes the center of the field of view to remain constant across set field of view command.

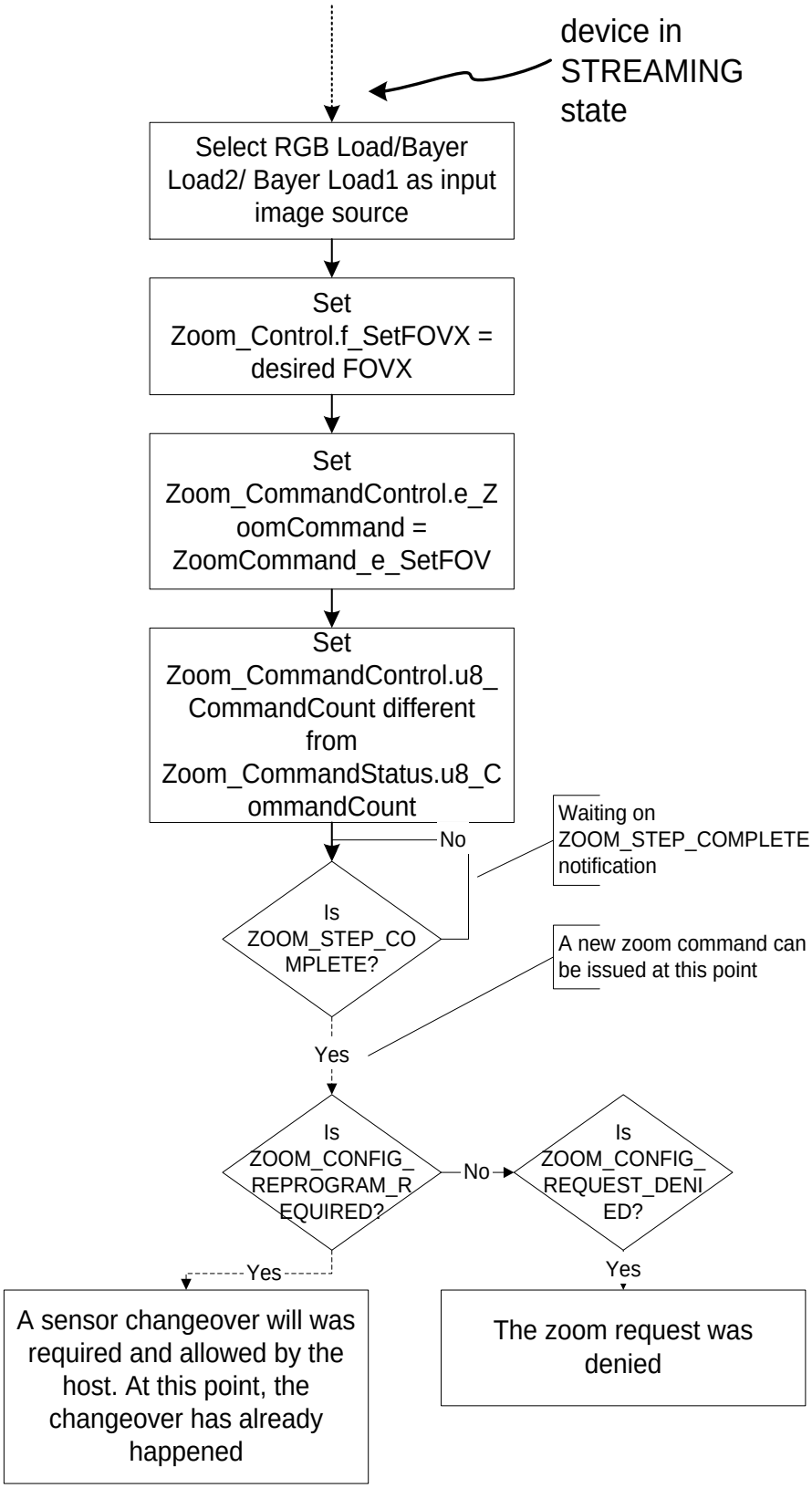


Figure 21 Zoom Command: SetFOV

10.1.4.3 ZoomCommand_e_RefreshOutputSize

This command can be used to dynamically change the output image resolution of the active pipes. It must be understood that any such change in the output image resolution may result in sensor change over due to changed scaling factors. Even though the device is able to update the pipe output resolution dynamically, the host may not be able to deal with a new output image resolution instantaneously. However, dealing with such cases is beyond the scope of zoom.

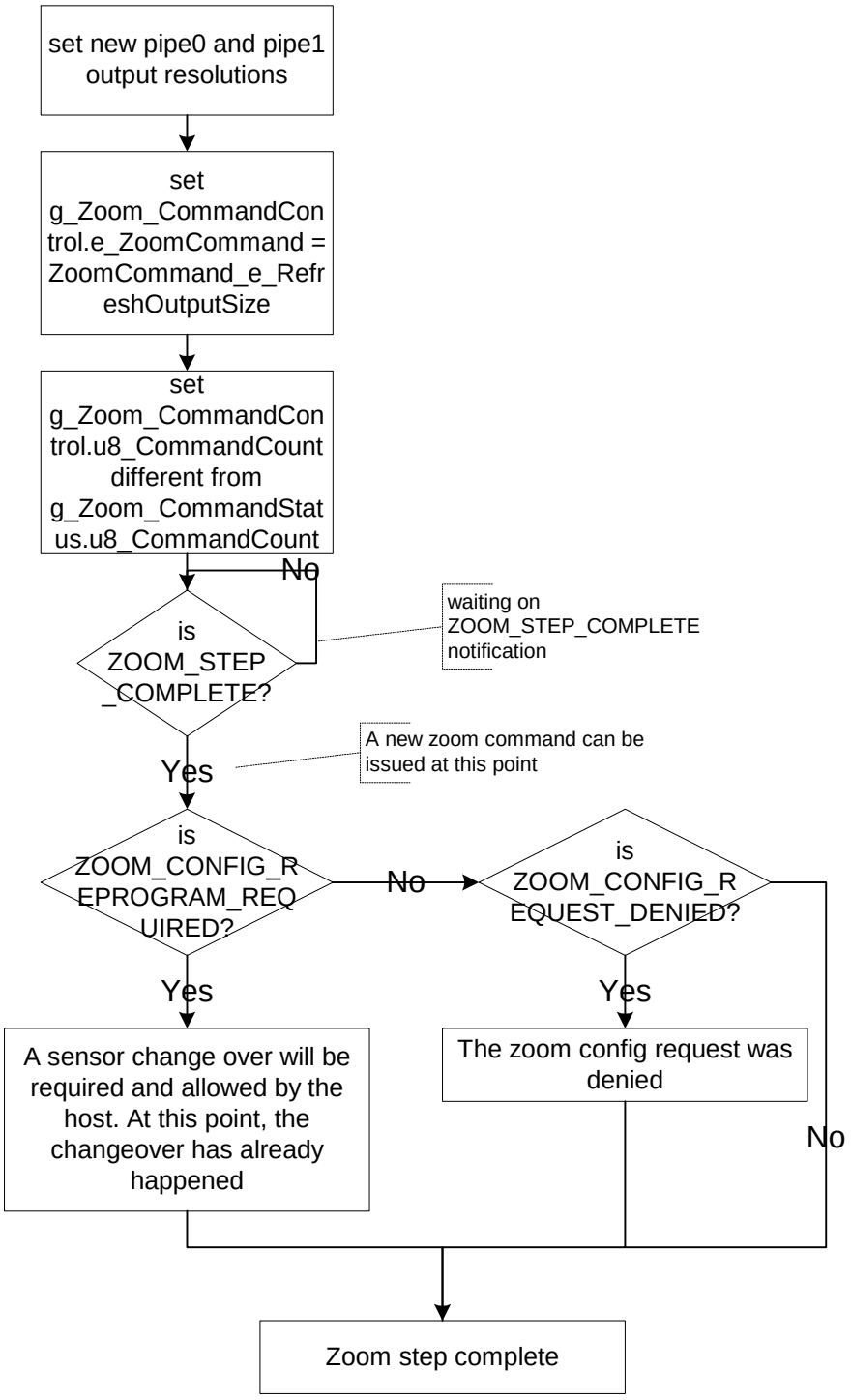


Figure 22 Zoom Command: RefreshOutputSize

10.1.5 Dynamic Change in Output Image Resolution

There are two ways in which the output image resolution of a pipe can be changed:

- Through a zoom command "`ZoomCommand_e_RefreshOutputSize`": Using this command, the nominal output image resolution of the active pipes can be changed. Refer to section 10.1.4.3 for details. It must be noted that such a command cannot be accompanied by another simultaneous zoom command i.e. only one zoom command can be issued by the host at a time. Hence in this case, a change in user field of view cannot be done along with a change in the output image resolution.
- Through a LUT at each zoom step. In this case, a change in output image resolution is done at each zoom step.

For the purpose of digital video stabilization, it is desirable to change the nominal output image resolution of the active pipes depending on the current user field of view. This is implemented using a LUT `Zoom_DynamicOutputResolutionUpdateLUT`. This LUT implements two tables of 8 entries each; `f_FOVRatio` and `f_OutputResolutionChangeFactor`.

To be able to specify the relative change in output image resolution across the complete range of user field of view, we must consider the ratio of user field of view to the maximum field of view at current center. Let us say that this ratio is `f_CurrentFOVRatio`. The complete zoom range is divided into eight zones for which different output image resolution change factors will be applied. Corresponding to these zones, the `f_CurrentFOVRatio` is computed by the host and programmed into `f_FOVRatio`. These values must be programmed in descending order i.e. `f_FOVRatio[0] > f_FOVRatio[1] > f_FOVRatio[2]` and so on. Corresponding to each `f_FOVRatio[n]`, the relative change in output image resolution must be programmed in `f_OutputResolutionChangeFactor[n]` by the host.

Consider Table 10 [8] which shows the possible values of `f_FOVRatio` and `f_OutputResolutionChangeFactor`:

Index	f_FOVRatio	f_OutputResolutionChange Factor	Interpretation
0	1.0	1.0	No change in output image resolution from FFOV to (not inclusive) 87.5% of FFOV
1	0.875	1.05	Output image resolution to increase by 5% for FOV from 87.5% of FFOV to (not inclusive) 75% of FFOV
2	0.75	1.10	Output image resolution to increase by 10% for FOV from 75% of FFOV to (not inclusive) 62.5% of FFOV
3	0.625	1.15	Output image resolution to increase by 15% for FOV from 62.5% of FFOV to (not inclusive) 50% of FFOV
4	0.50	1.20	Output image resolution to increase by 20% for FOV from 50% of FFOV to (not inclusive) 37.5% of FFOV
5	0.375	1.25	Output image resolution to increase by 25% for FOV from 37.5% of FFOV to (not inclusive) 25% of FFOV

[8] FFOV to be interpreted as the maximum field of view available at the current center in the table

6	0.25	1.30	Output image resolution to increase by 30% for FOV from 25% of FFOV to (not inclusive) 12.5% of FFOV
7	0.125	1.35	Output image resolution to increase by 35% for FOV smaller than 12.5% of FFOV

Table 10 Example and Interpretation of f_FOVRatio and f_OutputResolutionChangeFactor LUT

To enable the dynamic output image resolution change by any of the methods, the `Zoom_Control.e_Flag_PerformDynamicResolutionUpdate` element must be set to `Flag_e_TRUE`.

10.1.6 Pre-requisites

None

10.1.7 Pre BOOT Parameters

None

10.1.8 Pre RUN Parameters

- Maximum zoom range: This information specifies the maximum desired zoom range. It must be specified in `Zoom_Params.f_ZoomRange`. Using this information, the module computes the minimum field of view and makes it available in `Zoom_Status.f_MinFOVXAtArrayCenter`.
- User field of view: This information is used by the module to decide on the field of view at which the streaming will start. It is specified in `Zoom_Control.f_SetFOVX`. If this is set to a non zero value, then the device will start streaming at the specified field of view. If it is set to zero, then the device starts streaming at the maximum available field of view to the device.
- Center Offset: It is required for moving the center of the user field of view. It is specified as signed integers in `Zoom_Control.s16_CenterOffsetX` and `Zoom_Control.s16_CenterOffsetY`.
- Zoom_DynamicOutputResolutionUpdateLUT must be programmed before issuing a streaming command.

10.1.9 Live Parameters

The user field of view and the center offset can be changed at runtime through `ZoomCommand_e_SetFOV` and `ZoomCommand_e_SetCenter` respectively.

10.1.10 Digital Zoom Page Elements

10.1.10.1 Zoom_Params

Zoom_Params		
Page Elements	**Description**	**Range**
f_ZoomRange	Specifies the maximum zoom factor. This will relate directly to a minimum field of view. Must be programmed before starting a streaming operation	Should be greater than or equal to 1.0

Table 11 Zoom_Params Page

10.1.10.2 Zoom_Control

Zoom_Control		
Page Elements	**Description**	**Range**
f_SetFOVX	Specifies the FOV to be used if custom FOV is desired. If this is set to zero, then the maximum available FOV is used. When starting to stream after a mode static change, the device will start streaming from the FOV mentioned in this element.	
s16_CenterOffsetX	X offset to be applied to the center of the image. To be used for for pan and tilt operation. This offset can be negative as well as positive. The default center of the image is considered to be the center of the sensor imaging array. A negative offset will move the image center to the left while a positive offset will move the center to the right from its default offset. Only abosrbed in context of a zoom set command.	
s16_CenterOffsetY	Y offset to be applied to the center of the image. To be used for for pan and tilt operation. This offset can be negative as well as positive. The default center of the image is considered to be the center of the sensor imaging array. A negative offset will move the image center up while a positive offset will move the center down from its default offset. Only abosrbed in context of a zoom set command.	
e_Flag_PerformDynamicResolutionUpdate	Control to specify if the device must moderate the output image resolution of the active pipes based on the current zoom field of view. If set to Flag_e_TRUE, then at each zoom step, the device computes a factor by which the output image resolution is increased. This factor is extracted from the Zoom_DynamicOutputResolutionUpdateLUT	Flag_e_TRUE: Output image resolution of the active pipes is increased at each zoom step. Flag_e_FALSE: Output image resolutions of the active pipes remain constant at each zoom step.

Table 12 Zoom_Control Page

10.1.10.3 Zoom_Status

Zoom_Status		
Page Elements	**Description**	**Range**
f_XCenter	Specifies the zero based X coordinate of the current user FOV center	
f_YCenter	Specifies the zero based Y coordinate of the current user FOV center	
f_MaxFOVXAvailableToDevice	Specifies the maximum FOV X dimension that will be available to the user. It does not account for losses in the FOV due to differences in sensor aspect ratio and output image aspect ratio.	
f_MaxFOVYAvailableToDevice	Specifies the maximum FOV Y dimension that will be available to the user. It does not account for losses in the FOV due to differences in sensor aspect ratio and output	

	image aspect ratio.	
f_MinFOVXAtArrayCenter	Specifies the minimum FOV X dimension computed in basis of f_ZoomRange. It is computed as follows: f_MinFOVXAtArrayCenter = max_ar_corrected_fov_x/f_ZoomRange	
f_MaxAvailableFOVXAtCurrentCenter	Specifies the maximum FOV X dimension available at the currently applicable image center.	
f_FOVX	Specifies the current FOV X dimension	
f_FOVY	Specifies the current FOV Y dimension	
u16_MaximumAbsoluteXCenterShift	Specifies the maximum absolute X offset allowed for horizontal center shift. A shift of +-u16_MaximumAbsoluteXCenterShift is allowed to the center at the current FOV. If an offset greater than u16_MaximumAbsoluteXCenterShift is applied, it is moderated to u16_MaximumAbsoluteXCenterShift. Valid only when device is streaming	
u16_MaximumAbsoluteYCenterShift	Specifies the maximum absolute Y offset allowed for horizontal center shift. A shift of +-u16_MaximumAbsoluteYCenterShift is allowed to the center at the current FOV. If an offset greater than u16_MaximumAbsoluteYCenterShift is applied, it is moderated to u16_MaximumAbsoluteYCenterShift. Valid only when device is streaming	
s16_CenterOffsetX	Specifies the current horizontal center offset applied to the image. Valid only when the device is streaming	
s16_CenterOffsetY	Specifies the current vertical center offset applied to the image. Valid only when the device is streaming	

Table 13 Zoom_Status Page

10.1.10.4 Zoom_CommandControl

Zoom_CommandControl		
Page Elements	**Description**	**Range**
u8_CommandCount	Set the Zoom_CommandControl.u8_CommandCount value to be different from that of to Zoom_CommandStatus.u8_CommandCount to act as a trigger to absorb the zoom command (e_ZoomCommand).	
e_ZoomCommand	Zoom command is issued through this page element. The zoom command will be fetched only when control coin (Zoom_CommandControl.u8_CommandCount) is different from status coin (Zoom_CommandStatus.u8_CommandCount).	ZoomCommand_e_None ZoomCommand_e_SetCenter ZoomCommand_e_SetFOV ZoomCommand_e_RefreshOutputSize

Table 14 Zoom_CommandControl Page

10.1.10.5 Zoom_CommandStatus

Zoom_CommandStatus		
Page Elements	**Description**	**Range**
u8_CommandCount	When a zoom step has been computed and applied, the Zoom_CommandStatus.u8_CommandCount element becomes equal to Zoom_CommandControl.u8_CommandCount At this point, it is safe to issue another zoom command.	
e_ZoomCommand	Specifies the last zoom command executed	ZoomCommand_e_None ZoomCommand_e_SetCenter ZoomCommand_e_SetFOV ZoomCommand_e_RefreshOutputSize
e_ZoomCmdStatus	Specifies the status of the last zoom command.	ZoomCmdStatus_e_OK ZoomCmdStatus_e_FullyZoomedOut ZoomCmdStatus_e_FullyZoomedIn ZoomCmdStatus_e_SetOutOfRange
e_ZoomRequestStatus	Specifies the status of the last request made by the zoom module to the system to achieve a particular zoom configuration	ZoomRequestStatus_e_None ZoomRequestStatus_e_Accepted ZoomRequestStatus_e_Denied ZoomRequestStatus_e_ReProgramRequired

Table 15 Zoom_CommandStatus Page

10.1.10.6 Zoom_DynamicOutputResolutionUpdateLUT

Zoom_DynamicOutputResolutionUpdateLUT		
Page Elements	**Description**	**Range**
f_FOVRatio[0].. f_FOVRatio[7]	f_FOVRatio should be programmed in descending order i.e. f_FOVRatio[0] > f_FOVRatio[1] > f_FOVRatio[2]… At run time, if Zoom_Control. e_Flag_PerformDynamicResolutionUpdate is set to Flag_e_TRUE, then at each zoom step, the ratio of current_fov to f_MaxAvailableFOVXAtCurrentCenter is computed (lets say f_CurrentFOVRatio) Then the maximum f_FOVRatio <= f_CurrentFOVRatio is found. The index corresponding to this entry is used to index into f_OutputResolutionChangeFactor to find the relative change in output image resolution of the active pipes.	0.0 < f_FOVRatio[n]<=1.0
f_OutputResolutionChangeFactor[0].. f_OutputResolutionChangeFactor[7]	Both the tables must be programmed before issuing a streaming command.	OutputResolutionChangeFactor[n] >= 1.0

Table 16 Zoom_DynamicOutputResolutionUpdateLUT

10.2 Adaptive overscan

10.2.1 Adaptive over scanning Overview

For video stabilization, some margin should be added to the pipe output size and actual pipe output size should be greater than the required output by some percentage which depends on the zoom FOV.

10.2.2 Adaptive over scanning Usage

ISP FW and host share a common look-up table to calculate actual pipe output for a given FOV. ISP FW dynamically change this output size on a new setFOV() request according to the lookup table and GRAB also can adapt to this new output size and program DMA accordingly on the fly. Synchronization between ISP FW and GRAB is achieved through a 32 bit shared variable ***g_GrabNotify.u32_DMA_GRAB_Indicator_For_VideoStab*** which has a fixed memory address mentioned in section ***Grab Interface Overview***.

On receiving a new FOV, ISP FW calculates the new output size and necessary HW parameters (scalar/cropper etc), stop pipe output and notifies the GRAB about the output size change by setting the 1st bit of 3rd event register (0th bit of the same is used for flash). ISP FW waits for an acknowledgment from GRAB. This is done by setting the value of *g_GrabNotify.u32_DMA_GRAB_Indicator_For_VideoStab* to 1 by GRAB. On receiving the notification ISP FW enables the pipe output and frames with new resolution is streamed out.

10.2.2.1 Pre-requisites

None

10.2.2.2 Pre BOOT parameters

None

10.2.2.3 Pre RUN parameters

None

10.2.3 Page Elements

10.2.3.1 *Zoom_DynamicOutputResolutionUpdateLUT* page elements

	Zoom_DynamicOutputResolutionUpdateLUT	
Page Elements	Description	Range/ Possible Values
f_FOVRatio[0-7]	Specifies the FOV range as a fraction of g_Zoom_Status.f_MaxAvailableFOVX AtCurrentCenter	Must be setup in decreasing order i.e. f_FOVRange[0] should have the maximum value. At runtime, for each zoom step, the following is done: Consider f_FOVRange[n] = x and f_FOVRange[n+1] = y If x >= desired_fov / g_Zoom_Status.f_ MaxAvailableFOVXAtCurrentCenter > y then, the output image moderation factor used is f_OutputResolutionChangeFacto r[n]

f_OutputResolutionChangeFactor[0-7]	Specifies the factor by which the output image resolution has to be changed	

10.2.4 Default Settings recommendation

10.3 Best sensor mode selection

10.3.1 Best sensor mode selection Overview

ISP FW internally chooses a sensor mode which meets all the user port requirements (frame rate, output size, FOV), also taking into account aspect ratio of the ports and max digizoom requirement. Active port requirements are mandatory requirements and must be complied with. Inactive port requirements are indicative requirements and should also be complied with. However, if no single mode can satisfy both mandatory as well as indicative requirements, no error should be returned and sensor mode should be chosen to meet mandatory requirements alone.

10.3.2 Best sensor mode selection Usage

VPB0 port requirements are programmed with PE *Pipe[1]*. VPB2 port requirements are programmed with PE *Pipe[0]*. All these ports can have different frame rate (should be integer multiple of the port with least frame rate) at OMX level. But ISP FW will program the sensor for max frame rate and all the active pipes (including BMS path) should stream with same max frame rate. Frame rate can be programmed with PE *g_FrameRateControl.f_UserMaximumFrameRate_Hz*.

ISP FW will recalculate best sensor mode when host modifies the following,

a. Set a new DZ and

b. Activate any port

a is part of zoom functionality (details are available in zoom specific document). b is use case dependent and host can activate/deactivate any of the pipe on the fly using PE *DataPathControl* .

NB: When both pipes are disabled, this corresponds to still capture use case. As per SAS, for transition from VF to still capture, host needs to explicitly issue stop/start command. Again transition from still to VF requires explicit stop/start.

For other use cases, at least one of the pipes should remain active.

A coin mechanism has been implemented to support on the fly pipe (pipe0 & pipe1) enabling. Host needs to program the appropriate PE (*DataPathControl. e_Flag_Pipe[0/1]Enable*) and toggle the coin *DataPathControl .e_Coin_PipeEnable* to enable a pipe on the fly.

10.3.2.1 Pre-requisites

None

10.3.2.2 Pre BOOT parameters

None

10.3.2.3 Pre RUN parameters

None

10.3.3 Page Elements

10.3.3.1 *Pipe[0/1]*.page elements

I	*Pipe[0/1]*	
Page Elements	Description	Range/ Possible Values
u16_X_size	Output size X	
u16_Y_size	Output size Y	
e_OutputFormat_Pipe	Output image format from Pipe	
e_Flag_TogglePixValid	Set to e_TRUE if toggle_pix_valid required	
u8_PixValidLineTypes	decides for which pixels, the pixvalid signal would be generated	
e_Flag_Flip_Cb_Cr	Set to e_TRUE if Cb and Cr of the pipe have to be flipped	
e_Flag_Flip_Y_CbCr	Set to e_TRUE if Y and CbCr of the pipe have to be flipped	
e_Flag_Valid	Set to e_TRUE if above settings are valid for the specific pipe	

10.3.3.2 *DataPathControl/DataPathStatus* page elements

	DataPathControl/ DataPathStatus	
Page Elements	Description	Range/ Possible Values
e_Flag_Pipe0Enable	Pipe0 output is to be enabled	

e_Flag_Pipe1Enable	Pipe1 output is to be enabled	
e_Flag_BayerStore0Enable	Bayer store0 output is to be enabled	
e_Flag_BayerStore1Enable	Bayer store1 output is to be enabled	
e_Flag_BayerStore2Enable	Bayer store2 output is to be enabled	
e_BayerStore2Source	Bayer store2 source	
e_Flag_RGBStoreEnable	RGB store output is to be enabled	
e_RGBStoreSource	RGB store source	
e_Flag_PipeRAWEnable	Pipe0 output is to be enabled	Not used in this release
e_Coin_PipeEnable	Coin to update pipe dynamically.	

10.3.3.3 *Zoom_Control1* page elements

	Zoom_Control1	
Page Elements	Description	Range/ Possible Values
f_MaxDZ	Specifies max DZ factor	Default value is 10.0

Yellow indicates the new PEs added for best sensor mode selection.

10.3.4 Default Settings recommendation

11. Pipe Output Data Format and Sizes

11.1 Overview

The device has two output pipes which can generate image frames simultaneously or independently (based on user controls). The device allows the host to enable/disable the pipes, setup the output image resolution required from the pipes and the output image format required for the pipes independently. The following sections explain the different pipe output related functionalities and their controls.

11.2 Dynamic Change of Pipe Output Image Resolution

As specified in the zoom section, the device allows the host to dynamically update the output image resolution of the active pipes in the following two ways:

Through an explicit zoom command: In this case, a change in output image resolution cannot be accompanied by any change in the user field of view.

Through an automatic increment in output image resolution at each zoom step depending on the user field of view.

For details about programming the zoom to change the pipe output image resolution based on the above methods, please refer to section **Error! Reference source not found.** on digital zoom. The current section explains the support that the device provides to the host to allow it to cope with changed output image resolution asynchronously.

It is possible that the host is not able to cope with changed output image resolutions asynchronously. In such a case, it is essential to establish a handshaking mechanism between the host and the device to allow such a change to occur.

Figure 23 depicts the flow of control and the handshaking mechanism that has been established to allow the host to cope with change in output image resolution while streaming.

If the `Zoom_Control.e_Flag_PerformDynamicResolutionUpdate` element is set to `Flag_e_TRUE`, the device will issue a ZOOM_OUTPUT_IMAGE_RESOLUTION_READY event in context of each zoom command regardless of whether the output image resolution has to change or not. If the host wishes to establish a handshake with respect to output image resolution change, then it must set this element to `Flag_e_TRUE`. This notification is raised once the device has computed the complete scalar parameters and the corresponding pipe output image resolutions. In context of this event notification, the host can read off the resolution of the pipes from `PipeState[0].u16_OutputSizeX`, `PipeState[0].u16_OutputSizeY` and `PipeState[1].u16_OutputSizeX`, `PipeState[1].u16_OutputSizeY` for pipe0 and pipe1 respectively. In context of the vid complete interrupt of the corresponding pipe, the pipe output is disabled and the new scalar parameters applied. However if the `Zoom_Control.e_Flag_PerformDynamicResolutionUpdate` element is set to `Flag_e_TRUE`, the device will wait for an ACK from the host before enabling the pipe again. To implement such a mechanism, the elements `u8_RefreshOutputSizeControlCount` and `u8_RefreshOutputSizeStatusCount` in page `g_ZoomTop_ParamAppicationControl` are used. Whenever, the host is ready to absorb a changed output image from the device, it must set the `u8_RefreshOutputSizeControlCount` to be different from the `u8_RefreshOutputSizeStatusCount`. This acts as a signal to the device which then goes ahead to enable the pipe output if the relevant pipes.

The above handshake mechanism ensures that the host will always be able to know the resolution of the next frame from the ISP. It can prepare itself for this before it allows the device to enable the pipe again.

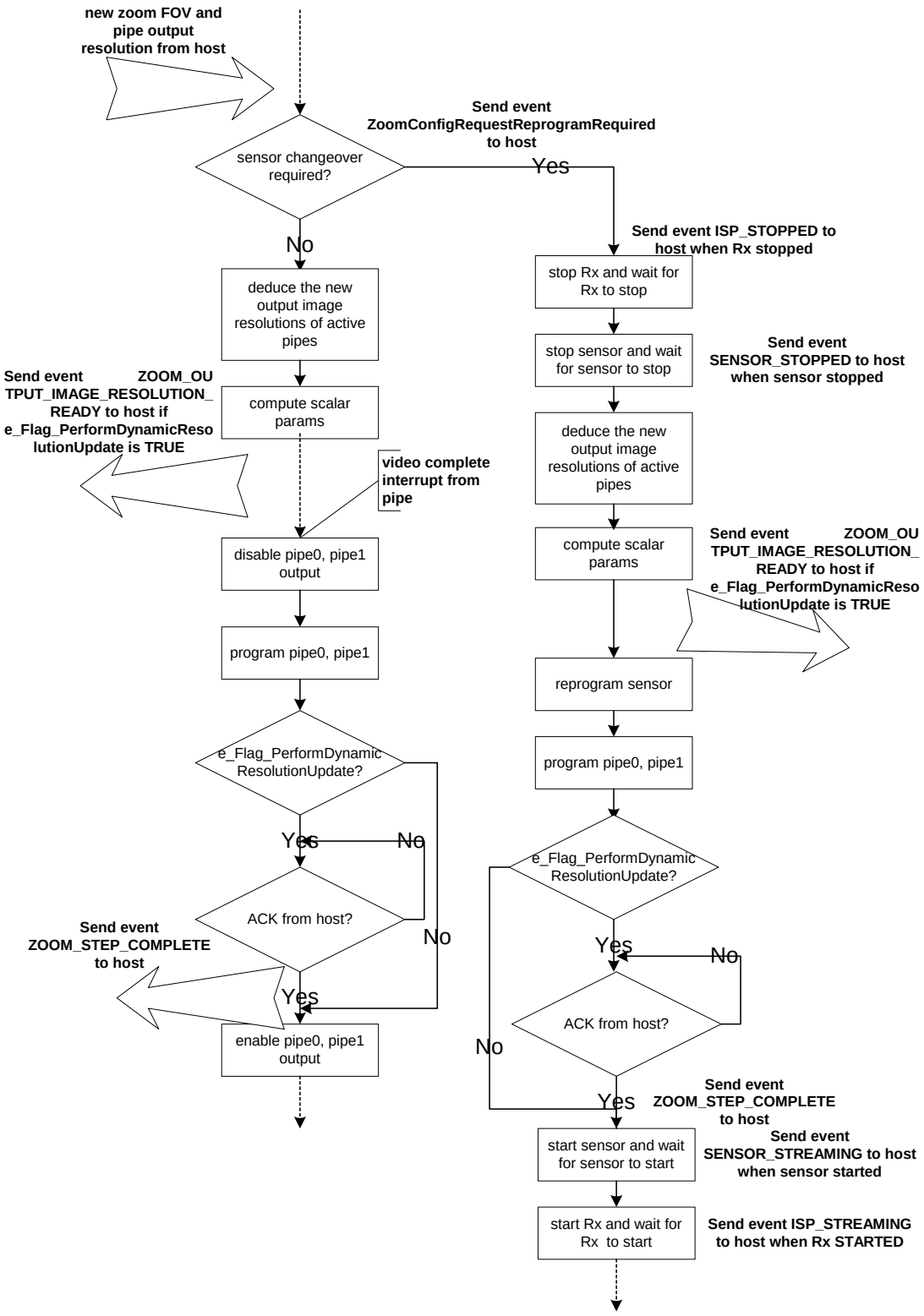


Figure 23 Flow of Control for Dynamic Update of Output Image Resolution

11.3 Usage

11.3.1 Pre-requisites

The host should be able to program the Pipe[0], Pipe[1], PipeStatus[0] and PipeStatus[1] pages

11.3.2 Pre BOOT parameters

None

11.3.3 Pre RUN parameters

All the parameters are pre run and must be programmed prior to issuing a streaming command.

11.3.4 Live parameters

None

11.4 Page Elements

11.4.1 Output image size, enable/ disable and format page elements

Pipe[0] Pipe[1]	Interface to program the pipe output size, output format, etc.	
Page Elements	Description	Range/ Possible Values
u16_X_size	Specifies the horizontal image resolution for the corresponding pipe	
u16_Y_size	Specifies the horizontal image resolution for the corresponding pipe	
e_OutputFormat_Pipe	Specifies the output image format for the corresponding pipe	OutputFormat_e_RGB101010_PEAKED, OutputFormat_e_RGB101010_UNPEAKED, OutputFormat_e_RGB888_PEAKED, OutputFormat_e_RGB888_UNPEAKED, OutputFormat_e_RGB565_PEAKED, OutputFormat_e_RGB565_UNPEAKED, OutputFormat_e_RGB555_PEAKED, OutputFormat_e_RGB555_UNPEAKED, OutputFormat_e_RGB444_UNPACKED_PEAKED, OutputFormat_e_RGB444_UNPACKED_UNPEAKED OutputFormat_e_RGB444_PACKED_PEAKED, OutputFormat_e_RGB444_PACKED_UNPEAKED, OutputFormat_e_YUV
e_Flag_TogglePixValid	Specifies if a toggle pix valid signal is required for the corresponding pipe	e_TRUE (if toggle pix valid is required), e_FALSE (if toggle pix valid is not required)
u8_PixValidLineTypes	Specifies the line types for which a pix valid signal is required for the corresponding pipe	
e_Flag_Flip_Cb_Cr	Specifies if a Cb and Cr order flip is required for the corresponding pipe	e_TRUE (if Cb and Cr order flip is required), e_FALSE (if Cb and Cr order flip is not required)
e_Flag_Flip_Y_CbCr	Specifies if a Y and Cb, Cr order flip is required for the corresponding pipe	e_TRUE (if Y and Cb, Cr order flip is required), e_FALSE (if Y and Cb, Cr order flip is not required)

11.4.2

11.4.3 Pipe status page elements

PipeStatus [0] PipeStatus [1]	Status page elements for the pipe	
Page Elements	Description	Range/ Possible Values
u8_FramesStreamedInPipeLastRun	Specifies the number of frames streamed from the sensor since the last streaming command. Only valid if the pipe is enabled	
u8_FramesStreamedOutOfPipeLastRun[9]	Specifies the number of frame streamed from the pipe since the last streaming command. Only valid if the pipe is enabled	
e_Flag_VideoCompleteInterruptPending	Specifies of the vid complete of the corresponding pipe is pending or not. It is set in context of the status line interrupt and is reset in context of the video complete of the corresponding pipe.	Flag_e_TRUE if vid complete is pending Flag_e_FALSE if vid complete is not pending.

11.4.4 Zoom Parameter Application Control Page

ZoomTop_ParamApplicationControl	Page to control the application of changed output image resolution	
Page Elements	Description	Range/ Possible Values
u8_RefreshOutputSizeControlCount	Using these elements, the host can specify when it wants the device to enable the pipe output for a pipe resolution change zoom command. To signal the device to enable the pipe output, the host must set the control count different from status count. Only used in case of the zoom command to change the output image resolutions. Not used for other zoom commands.	
u8_RefreshOutputSizeStatusCount		

[9] It is a good test to see that u8_FramesStreamedInPipeLastRun == u8_FramesStreamedOutOfPipeLastRun. This ensures that the number of frame that have entered the pipe have been output as well.

12. ISP Pipe Openings

12.1 Data path setup

12.1.1 Data path setup overview

The device provides various input data paths (data load) and output data paths (data store) to the host. The different output data paths supported by the device are specified in Table 17.

Output Point	Remarks
Colour Engine 0	High resolution Pipe0 output. Must be enabled before start of streaming. Cannot be enabled while live streaming.
Colour Engine 1	Low resolution Pipe1 output. Must be enabled before start of streaming. Cannot be enabled while live streaming.
RGB Store	Store point in RGB domain in the DMCE. This store point can source data from two different points in the DMCE (post Babylon or post HBarrel). The source of this data store point must be set prior to start of streaming. This data store point can be enabled during live streaming. The source of this data point must be programmed before enabling the data store point.
Bayer Store 2	Store point in bayer domain in the RE. This store point can source data from two different points in the RE (post Duster or post Bayer Crop). The source of this data store point must be set prior to start of streaming. This data store point can be enabled during live streaming. The source of this data point must be programmed before enabling the data store point.
Bayer Store 1	Store point in bayer domain in the SD Pipe post LBE. This data store point can be enabled during live streaming.
Bayer Store 0	Store point in bayer domain in the SD pipe pre LBE. Used for fast RAW bayer grab. This data store point can be enabled during live streaming.

Table 17 Data Store Points

The different input data paths supported by the device are specified in Table 18.

Input Point	Remarks
RGB Load	Load point in RGB domain in the DMCE post Babylon
Bayer Load 2	Load point in bayer domain in the DMCE pre Babylon
Bayer Load 1	Load point in bayer domain in RE pre RSO
Sensor 0	Input image source will be sensor0
Sensor 1	Input image source will be sensor1

Table 18 Data Load Points

The following figures show the different load and store points in the device.

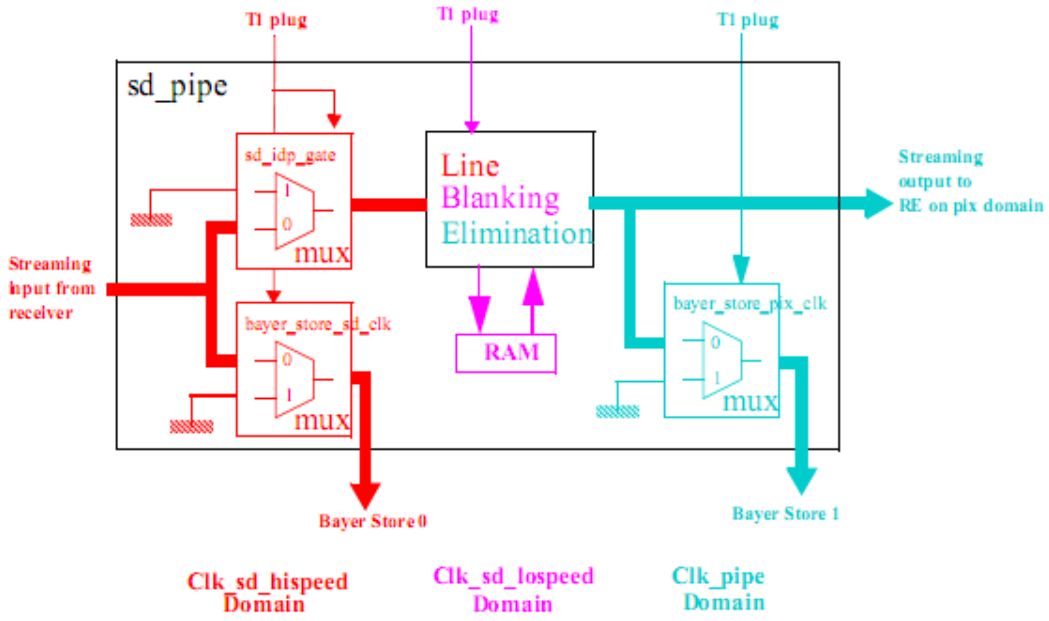


Figure 24 SD pipe block diagram showing bayer store 0, bayer store 1 and sensor input

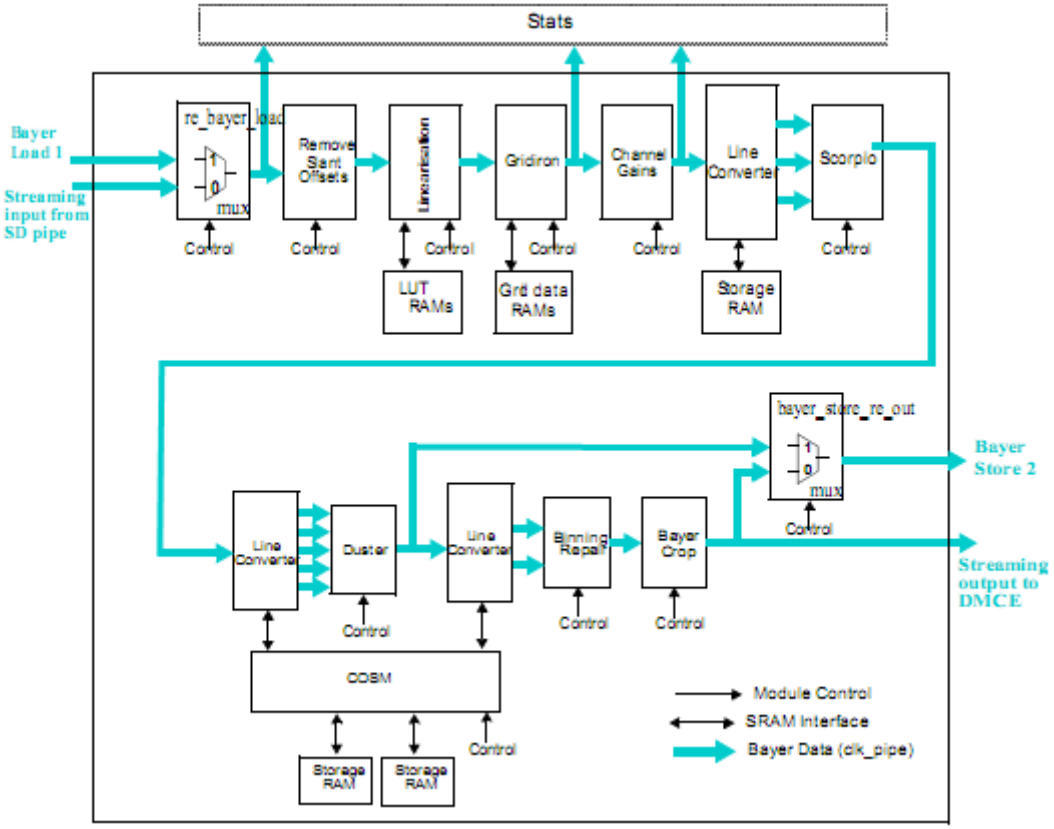


Figure 25 RE block diagram showing bayer load 1 and bayer store 2

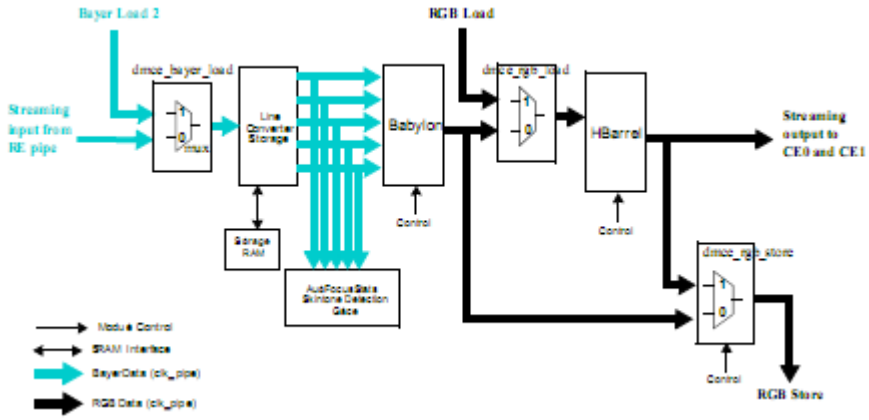


Figure 26 DMCE block diagram showing bayer load 2, RGB load and RGB store

12.1.2 Data path usage

The module can be used to save frames in memory using BMS, apply propriety algorithm and feed the frames back into pixel pipe using BML.

12.1.2.1 Pre-requisites

Only one data load point can be active at a given time for a given streaming operation.

Colour engine0/1 cannot be enabled during live streaming. They must be enabled before streaming start.

Source of RGB Store or bayer store 2 must be programmed before enabling the data points.

12.1.2.2 Pre BOOT parameters

None

12.1.2.3 Pre RUN parameters

Load point programming

Colour engine 0/1 enable

12.1.2.4 Live parameters

RGB Store

Bayer Store 2

Bayer Store 1

Bayer Store 0

12.1.3 Page Elements

12.1.3.1 Load Point Programming

SystemSetup	**System setup page used for programming load point**	
Page Elements	Description	Range/ Possible Values
e_InputImageSource	Specifies the data load point	InputImageSource_e_Sensor0, InputImageSource_e_Sensor1, InputImageSource_e_Rx[10], InputImageSource_e_BayerLoad1, InputImageSource_e_BayerLoad2, InputImageSource_e_RGBLoad

Table 19 Page Elements for Load Point Programming

12.1.3.2 Store Point Programming

DataPathSetup	**Page elements used for programming store points**	
Page Elements	Description	Range/ Possible Values
e_Flag_Pipe0Enable	Specifies that pipe0 should be enabled	Flag_e_TRUE, Flag_e_FALSE
e_Flag_Pipe1Enable	Specifies that pipe1 should be enabled	Flag_e_TRUE, Flag_e_FALSE
e_Flag_BayerStore0Enable	Specifies that bayer store 0 point should be enabled	Flag_e_TRUE, Flag_e_FALSE
e_Flag_BayerStore1Enable	Specifies that bayer store 1 point should be enabled	Flag_e_TRUE, Flag_e_FALSE
e_Flag_BayerStore2Enable	Specifies that bayer store 2 point should be enabled	Flag_e_TRUE, Flag_e_FALSE
e_BayerStore2Source	Specifies bayer store 2 source	BayerStore2Source_e_DusterOutput, BayerStore2Source_e_BayerCrop
e_Flag_RGBStoreEnable	Specifies that RGB store point should be enabled	Flag_e_TRUE, Flag_e_FALSE
e_RGBStoreSource	Specifies the RGB store source	RGBStoreSource_e_BabylonOutput, RGBStoreSource_e_HBarrelOutput

Table 20 Page Elements for Store Point Programming

[10] Not used at the moment

12.1.4 Default settings recommendation

All firmware page elements are set to use pixel pipe i.e. no BMS / BML is enabled.

13. Device Streaming Operations

As mentioned in section 12, the device has various data load and store points. Different modes of operation will need to enable/disable different store points in the device. This section deals with the host end programming that will be needed for common streaming operations.

13.1 Viewfinder or Movie Operation

A viewfinder or a movie operation will generally be done by on-the-fly image data being generated infinitely through pipe0 or pipe1 (or both) till the streaming operation is stopped. During this operation, a high frame rate may be desired and hence sub sampling in the sensor may have to be used. Please refer section **Error! Reference source not found.** regarding the significance of sub sampling and maximum frame rate requirements. Figure 27 depicts the host end programming required to setup the system for viewfinder or movie operation.

The device also supports enabling the different store points (except pipe0 and pipe1) on the fly while streaming. This facilitates the host to do an image grab during viewfinder or movie operation.

In viewfinder or movie operation, the sensor output may be cropped and/or sub sampled depending on maximum frame rate, sensor data rate and zoom parameters. If a store point is enabled dynamically in viewfinder or movie operation, the size of the frame that will be streamed through the store point will be available in `CurrentFrameDimension.u16_OPXOutputSize` and `CurrentFrameDimension.u16_OPYOutputSize` elements. The host must read off these elements and program the DMA accordingly before enabling the corresponding store points.

It must be noted that in the viewfinder or movie mode of operation, the device manages the frame dimension itself and hence the `HostFrameConstraints.e_FrameDimensionProgMode` element should be programmed to `FrameDimensionProgMode_e_Auto` before issuing a RUN command to the device.

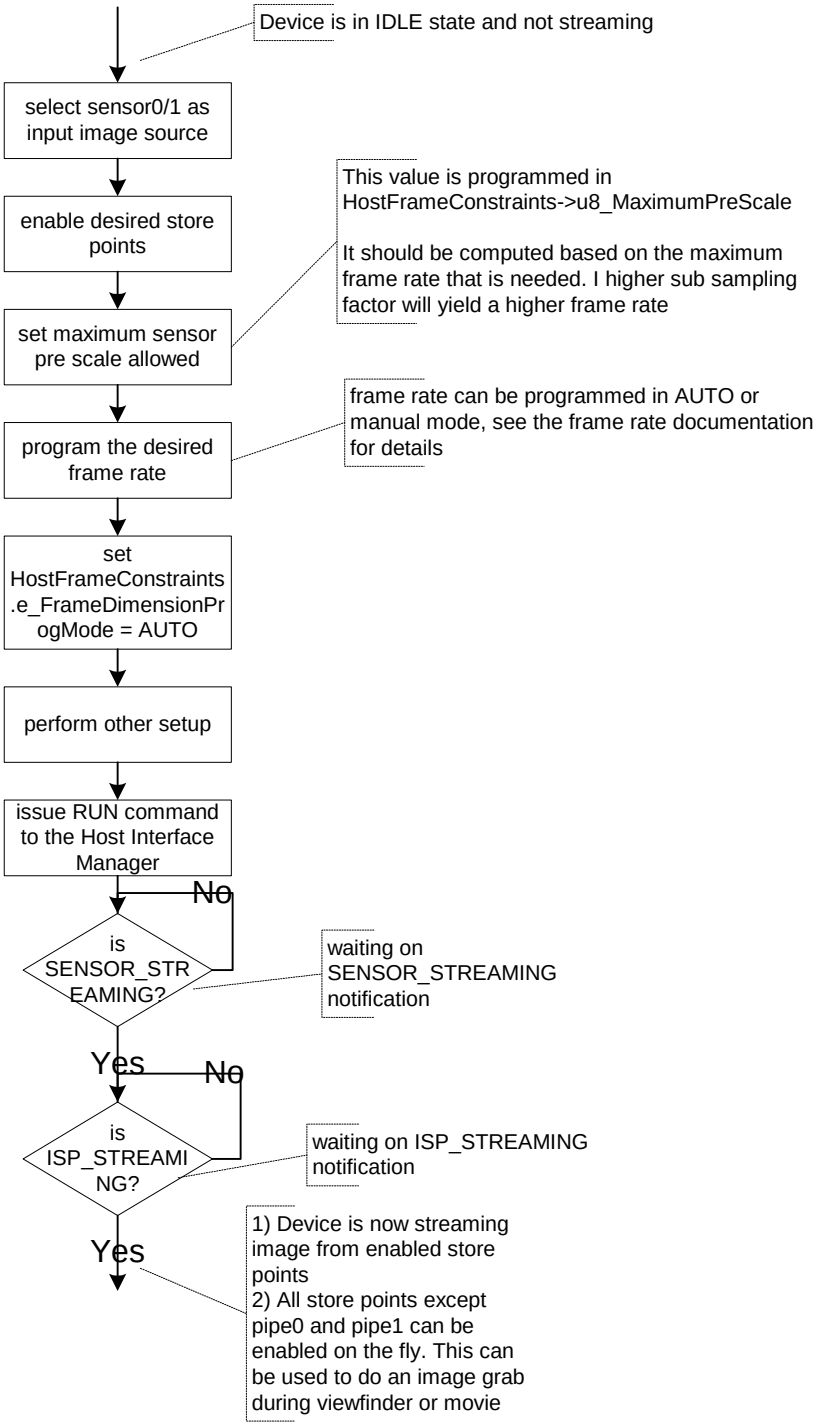


Figure 27 Viewfinder or Movie operation

13.2 Still Grab Operation

A still grab operation can be done as a sequence of image store and load operations or direct processing of on-the-fly frame from the sensor into the device. In both these cases, it is important to inhibit any sub sampling in the sensor for image quality purposes. This is done by programming the maximum sensor prescale to be 1 and the maximum sensor frame rate to be a small enough value to ensure that the maximum frame rate programmed is achievable at the current streaming parameters.

13.2.1 Single Stage Still Grab

In single stage still grab operation, the sensor image data is sent on-the-fly into the device and processed image is generated by the device. Figure 28 depicts the host end programming required to

setup the device for a single stage still grab. It may be noted that the host may enable pipe1 for this operation if it needs the thumbnail of the grabbed image to be processed concurrently.

It must be noted that in the single stage still grab, the device manages the frame dimension itself and hence the `HostFrameConstraints.e_FrameDimensionProgMode` element should be programmed to `FrameDimensionProgMode_e_Auto` before issuing a RUN command to the device.

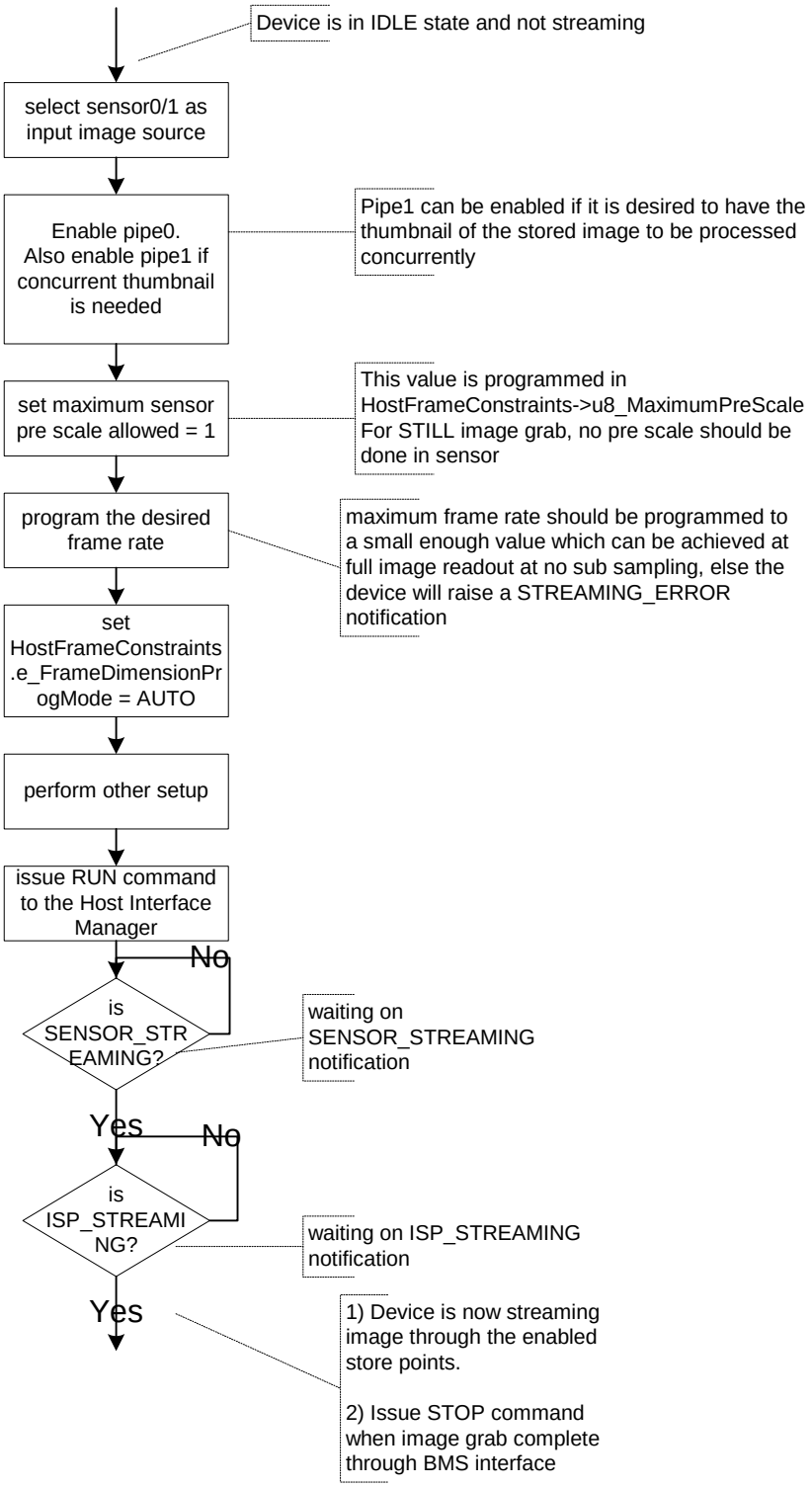


Figure 28 Single Stage Still Grab Operation

13.2.2 Multi Stage Still Grab

In multi stage still grab operation, a raw unprocessed image is stored in the memory in the first phase (through a BMS operation) and is subsequently loaded from memory and processed through the device in subsequent stages (BML operation).

The device makes no assumptions about the sequence of image store and load operations. However to be able to correctly process image data for a memory load operation, it is important that certain parameters that were applicable for the corresponding memory store operation be restored to the device before initiating the load operation.

The following parameters must be stored by the host at the time of a memory store operation and restored before memory load operation.

`Zoom_Control.s16_CenterOffsetX`

`Zoom_Control.s16_CenterOffsetY`

`Zoom_Control.f_SetFOVX`

`FrameDimensionStatus.u16_MaximumUsableSensorFOVX`

`FrameDimensionStatus.u16_MaximumUsableSensorFOVY`

`SensorFrameConstraints.u16_MaxOPXOutputSize`

`SensorFrameConstraints.u16_MaxOPYOutputSize`

`SensorFrameConstraints.u16_VTXAddrMin`

`SensorFrameConstraints.u16_VTYAddrMin`

`SensorFrameConstraints.u16_VTXAddrMax`

`SensorFrameConstraints.u16_VTYAddrMax`

`CurrentFrameDimension.f_PreScaleFactor`

`CurrentFrameDimension.u16_VTXAddrStart`

`CurrentFrameDimension.u16_VTYAddrStart`

`CurrentFrameDimension.u16_VTXAddrEnd`

`CurrentFrameDimension.u16_VTYAddrEnd`

It must be noted that during the memory load operation, the device does not manage the frame dimension itself and hence the `HostFrameConstraints.e_FrameDimensionProgMode` element should be programmed to `FrameDimensionProgMode_e_Manual` before issuing a RUN command to the device.

During memory store operation, the device manages the frame dimension itself and hence the `HostFrameConstraints.e_FrameDimensionProgMode` element should be programmed to `FrameDimensionProgMode_e_Auto` before issuing a RUN command to the device.

The host may decide to enable pipe1 if it is needed to generate the thumbnail of the stored image concurrently. Figure 29 and Figure 30 depict the host end programming required to setup the device for BMS and BML phase of multi stage still grab operation respectively. It must be noted that for BMS stage, the device will program the sensor to generate the biggest legal sensor frame. The host must program its DMA for store operation accordingly.

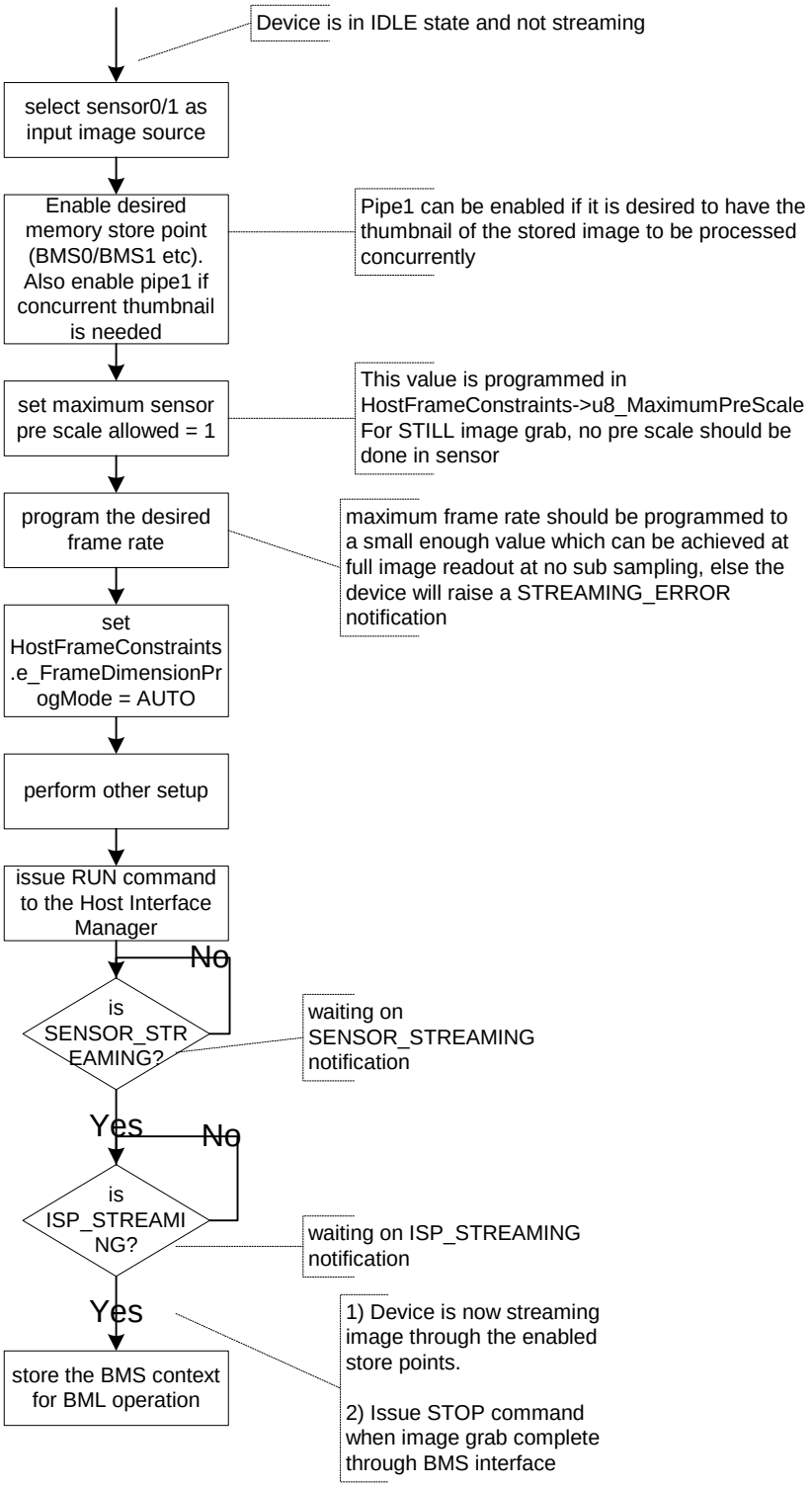


Figure 29 Multi Stage Still Grab Operation: BMS

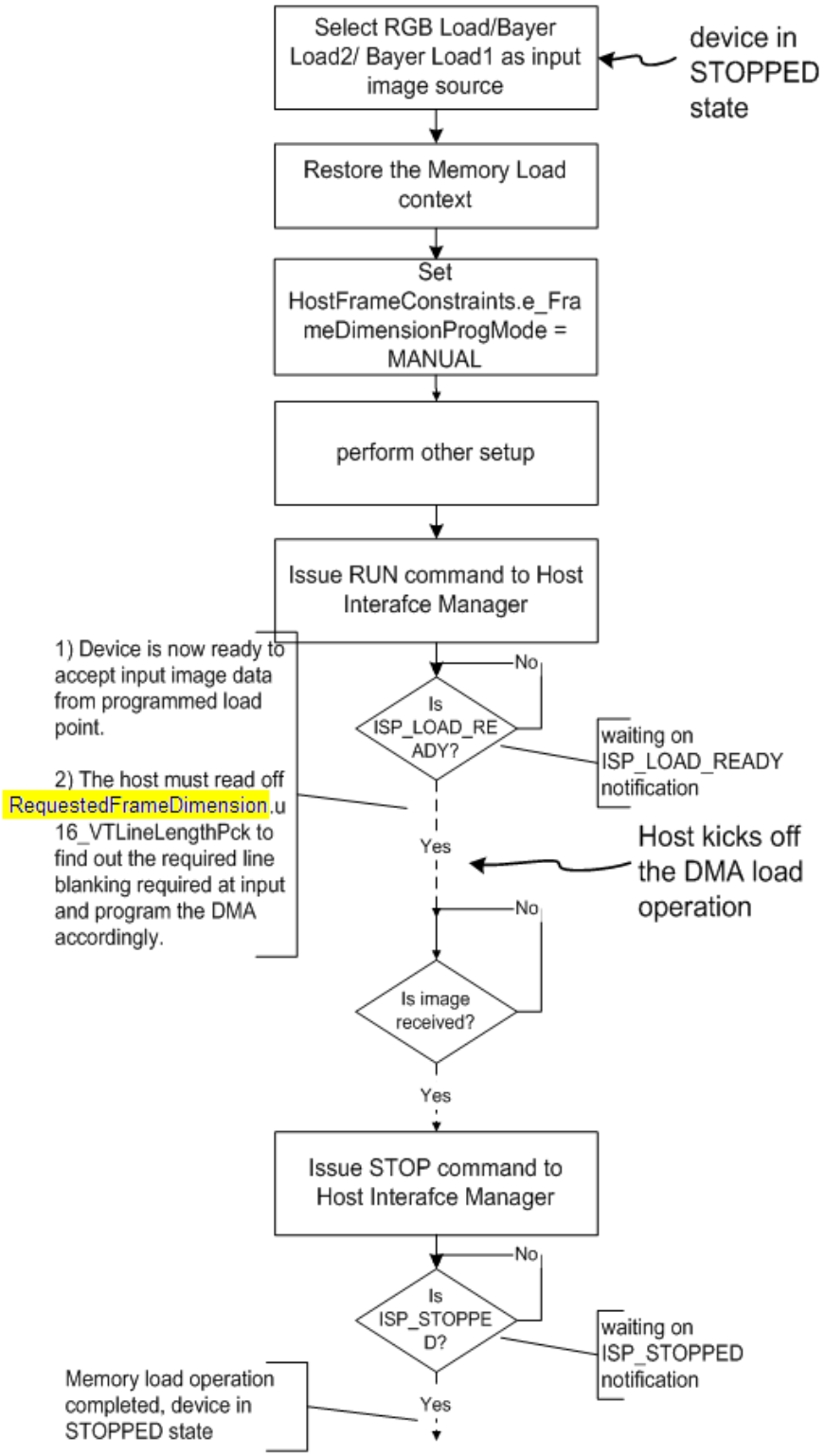


Figure 30 Multi Stage Still Grab Operation: BML

13.3 Scalar Stripe

13.3.1 Scalar with Striping overview

In a streaming ISP design, the horizontal width of the image that can be accommodated at its inputs points, intermediate blocks or output points can be limited due to limitations on line

memory requirements. Striping is a technique used to process images that have horizontal width greater than what can be accommodated in the pipe.

In striping, the image is vertically split into multiple stripes and processed through the ISP in multiple passes. These stripes have overlapping borders to ensure that the impact of image reconstruction at the borders can be negated. At the end of each pass, the processed output is stored in memory in such a way so as to reconstruct the final image correctly. Normally a striping operation is preceded by a store operation of the unprocessed image in memory.

In order to generate values which are needed by host to run striping mode, firmware needs certain inputs and according to those inputs it generates outputs which host programs later to provide correct data to ISP.

13.3.2 Scalar Stripe Module

The module reads input and output sizes from page elements which are exposed to HOST. After that it generates all the necessary outputs which are needed by HOST to run striping mode like output size, borders, minimum line length etc.

13.3.2.1 Pre-requisites

None

13.3.2.2 Pre BOOT parameters

None

13.3.2.3 Pre RUN parameters

Striping mode consists of two datapath, BMS + BML. To run this, HOST must perform following things in correct order:

- Program Input and Output Image resolution.
- Host will set datapath for Bayer Memory store operation. In this Pipe0 and Pipe1 are disabled and Bayer Store0 is enabled.
- Host will issue RUN Command to Host Interface Manager selecting sensor0/sensor1 as input image source.
- Host will wait for ISP Streaming notification to know whether streaming has started or not.
- Run command will enable Bayer Memory Store0 to store complete image grabbed from sensor into memory. For a store into memory when no other pipe is enabled, the complete frame is streamed from sensor is stored into memory. For a store operation when any pipe is enabled, the frame output from sensor is stored into memory. It may not be necessary that in this case, frame corresponds to full sensor frame.
- Host will wait for this store to be completed then it can give STOP command to Host Interface Manager to stop sensor streaming.
- Host will wait for ISP_STOP_STREAMING and SENSOR_STOP_STREAMING events to be notified; just to be sure that sensor has stopped its streaming operation.
- This is end of BMS datapath. This is explained in figure:1 as well. Now Host has to run Memory Load streaming Datapath.
- An image load from memory should be considered in association with previous image store operation. The device makes no assumptions about the sequences of image store and load operations. However to be able to correctly process image data for a memory load operation, it is important that certain parameters that were applicable for the corresponding memory store operation be restored to the device before initiating the load operation.

- To run BML datapath HOST will set datapath for this. Pipe0 and Pipe1 are enabled and Input Image Source is set to Bayer Memory Load1.
- Host will program No of Stripes and Current Stripe Count.
- Host will issue RUN Command to Host Interface Manager and input image source is already selected to Bayer Load0.
- Host will wait for ISP LOAD READY event. When this event has occurred, it means firmware has done all its computations for scalar outputs which are needed by HOST to program DMA to run Striping. Now Host must program DMA correctly and start the memory load. Memory Load operation is explained in figure:2 as well.

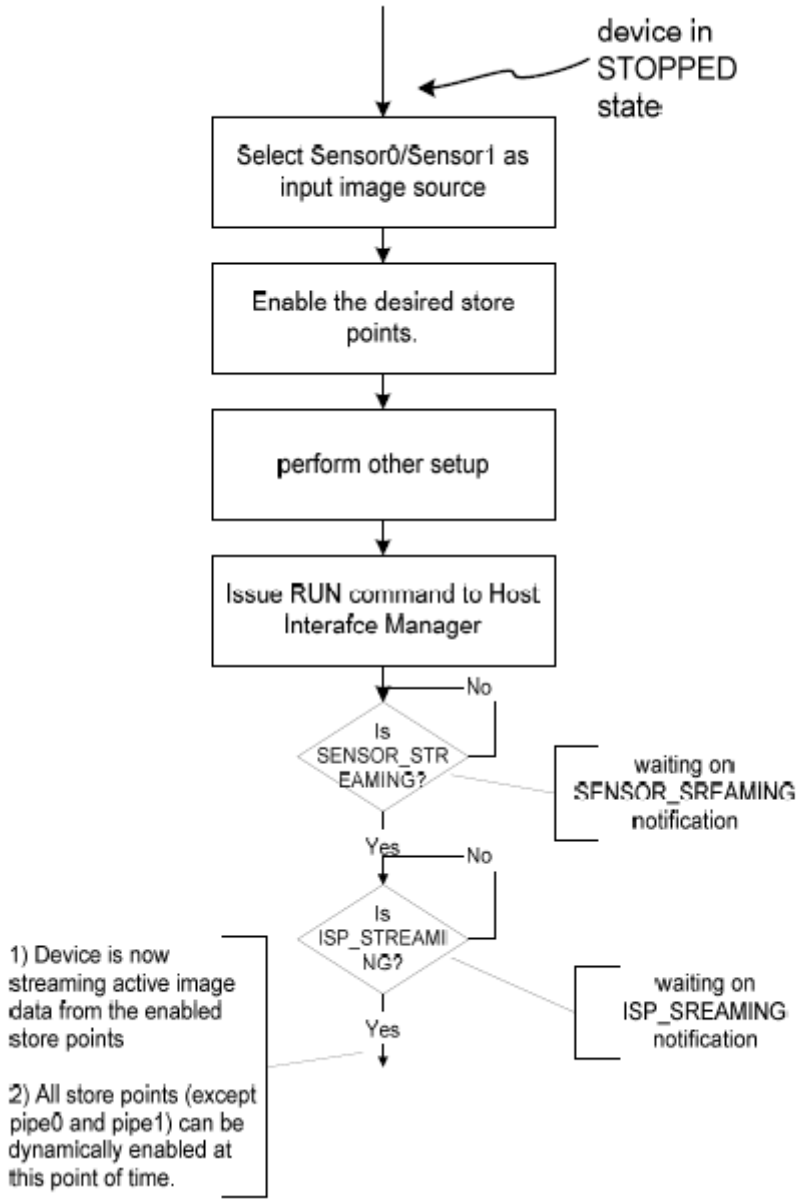


Figure 1 Bayer Memory Store

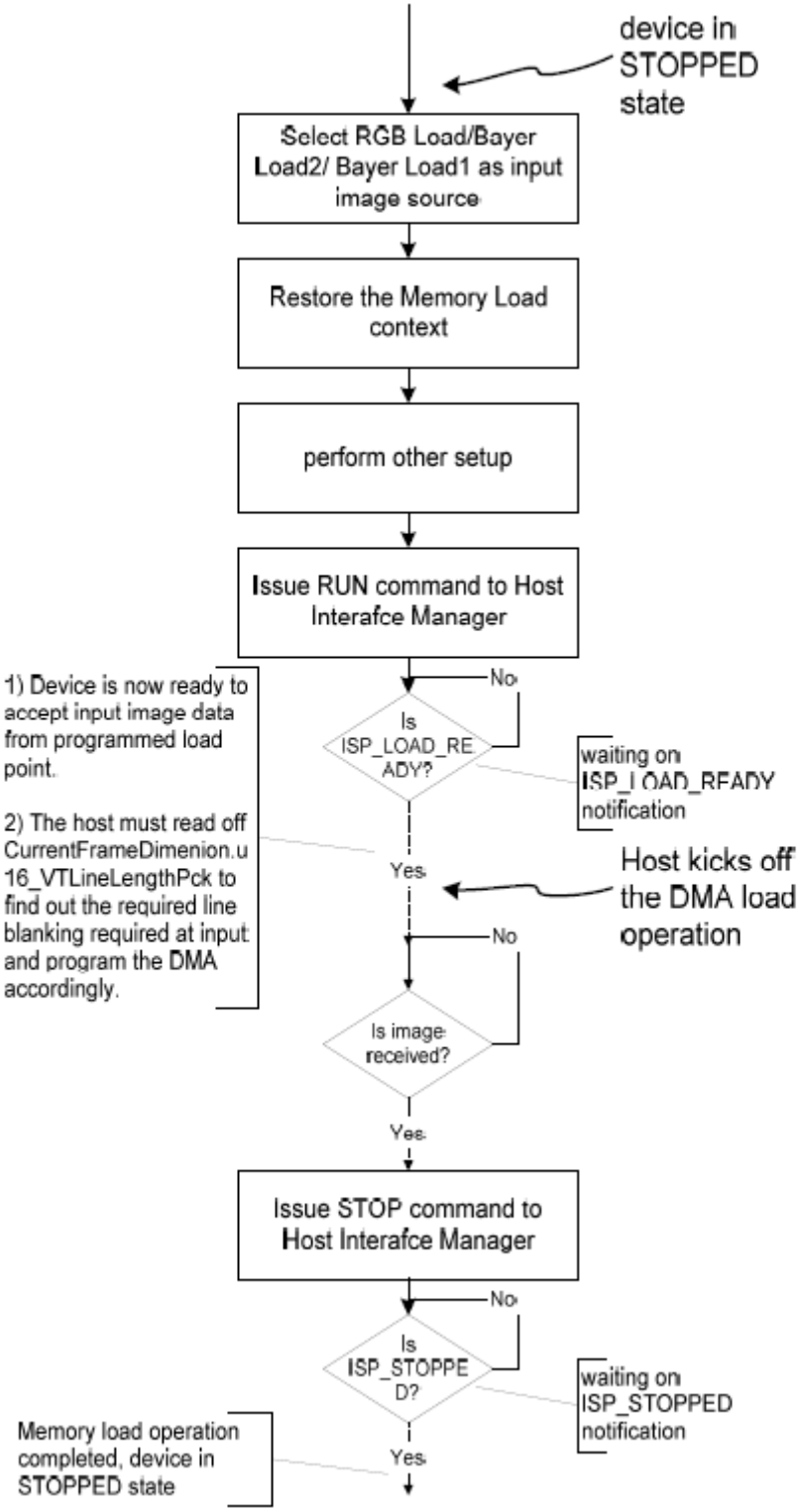


Figure 2 Bayer Memory Load

13.3.3 Scalar Stripe page elements

13.3.3.1 Scalar Stripe Input page elements

	g_Pipe_Scalar_Stripe_Inputs[0]:used for PIPE0 **g_Pipe_Scalar_Stripe_Inputs[1]:used for PIPE1**	
Page Elements	**Description**	**Range**
U8_StripeIndex	Current Stripe Index	
U8_StripeCount	No of Stripes	

13.3.3.2 Scalar user parameters

	g_Pipe_Scalar_UserParams[0]:used for PIPE0 **g_Pipe_Scalar_UserParams[1]:used for PIPE1**	
Page Elements	**Description**	**Range**
u16_PhysRegSize	Scalar coefficient array length	DEFAULT: 32
u8_PhysRegWidth	Scalar coefficient bitwidth	DEFAULT: 6
u8_BSize	Physical buffer level	DEFAULT: 2
u8_Crisp	Crispness level	DEFAULT: 8
e_Flag_AntiZipEnable	Anti zip control Enable: FLAG_e_TRUE Disable: Flag_e_FALSE	DEFAULT: FLAG_e_TRUE

13.3.3.3 Scalar Stripe Output Page Elements

	g_Pipe_Scalar_Stripe_Output[0]:used for PIPE0 **g_Pipe_Scalar_Stripe_Output[1]:used for PIPE1**	

Page Elements	**Description**	**Range**
S16_StripeInCropHStart	It specifies the horizontal offset into the unprocessed image from where the image should be read into the ISP.	
S16_StripeInCropVStart	It specifies the vertical offset into the unprocessed image from where the image should be read into the ISP.	
S16_StripeInCropHSize	It specifies the horizontal size of the unprocessed image that must be read into the ISP. Using in crop hstart and in crop hsize parameter, the host should be able to identify the portion of image that must be read into the ISP for the current stripe.	
S16_StripeInCropVSize	It specifies the vertical size of the unprocessed image that must be read into the ISP. Using in crop vstart and in crop vsize parameter, the host should be able to identify the portion of image that must be read into the ISP for the current stripe.	
S16_StripeOutCropHStart	The processed stripe generated by the ISP has some border which overlaps with the next stripe. This is required to negate the visible impact of scaling along borders. These borders are cropped by the ISP before the image is sent out.	
S16_StripeOutCropVStart	The processed stripe generated by the ISP has some border which overlaps with the next stripe. This is required to negate the visible impact of scaling along borders. These borders are cropped by the ISP before the image is sent out.	
S16_StripeOutCropHSize	Specifies the size of the processed stripe that will be generated by the ISP.	
s16_StripeOutCropVSize	Specifies the size of the processed stripe that will be generated by the ISP.	
s16_MinLineLength	For up scaling purposes, the host has to ensure certain amount of line blanking. It specifies the minimum line length required for the current stripe processing. The line blanking requirement must be deduced by the host using this parameter and “in crop size” parameter.	
S16_StripeScalarOutputHSize	Specifies the horizontal size of the processed stripe at the end of scalar	
S16_StripeScalarOutputVSize	Specifies the vertical size of the processed stripe at	

	the end of scalar	

13.3.4 Default settings recommendation

None

14. Image Quality

14.1 Color Matrix

14.1.1 Color matrix overview

In order to generate common image data from different SMIA sensors, a color space conversion is required to transform data from the color space of a particular SMIA sensor to some standard color space. As per SMIA standard, SMIA sensor must expose a static register map, a conversion matrix to transform pixel data from native sensor RGB color space to color space of sRGB.The individual color matrix elements are damped before they are actually applied onto the hardware.

14.1.2 Color matrix Usage

The module reads the color matrix programmed in the pages "`g_CE_ColourMatrixFloat[0]`" & "`g_CE_ColourMatrixCtrl[0]`". The coefficients are then converted to a 6.10 format and applied on to the hardware. No kind of sanity checks are done. The values are absorbed with new exposure or gain target in systemconfig coin.

14.1.2.1 Pre-requisites

None

14.1.2.2 Pre BOOT parameters

None

14.1.2.3 Pre RUN parameters

None

14.1.2.4 Live parameters

Any parameter can be changed but would be effective only after systemconfig control coin toggle.

14.1.2.5 Color matrix page elements

14.1.2.5.1 Color matrix control page CE_ColourMatrixFloat[]

Control Page element for color matrix (Both PIPE0 and PIPE1)	**g_CE_ColourMatrixFloat[0] store the values from Sensor and used for PIPE0 g_CE_ColourMatrixFloat[1] store the values from Sensor and used for PIPE1**	
Page Elements	**Description**	**Range**
f_RedInRed	ColourMatrix[0][0]: matrix_element_RedInRed	(-8) < Rcof < (+8)
f_GreenInRed	ColourMatrix[0][1]: matrix_element_GreenInRed	(-8) < Rcof < (+8)

f_BlueInRed	ColourMatrix[0][2]: matrix_element_BlueInRed	(-8) < Rcof < (+8)
f_RedInGreen	ColourMatrix[1][0]: matrix_element_RedInGreen	(-8) < Rcof < (+8)
f_GreenInGreen	ColourMatrix[1][1]: matrix_element_GreenInGreen	(-8) < Rcof < (+8)
f_BlueInGreen	ColourMatrix[1][2]: matrix_element_BlueInGreen	(-8) < Rcof < (+8)
f_RedInBlue	ColourMatrix[2][0]: matrix_element_RedInBlue	(-8) < Rcof < (+8)
f_GreenInBlue	ColourMatrix[2][1]: matrix_element_GreenInBlue	(-8) < Rcof < (+8)
f_BlueInBlue	ColourMatrix[2][2]: matrix_element_BlueInBlue	(-8) < Rcof < (+8)
e_SwapColour_Red	Swap Red channel with some other color DEFAULT]:e_RedSwap(0), No effect on image	/// Replace the channel with red e_RedSwap(0) /// Replace the channel with green e_GreenSwap(1) /// Replace the channel with blue e_BlueSwap(2)
e_SwapColour_Green	Swap Green channel with some other color [DEFAULT]:e_GreenSwap(1), No effect on image	/// Replace the channel with red e_RedSwap(0) /// Replace the channel with green e_GreenSwap(1) /// Replace the channel with blue e_BlueSwap(2)
e_SwapColour_Blue	Swap Blue channel with some other color [DEFAULT]:e_BlueSwap(2), No effect on image	/// Replace the channel with red e_RedSwap(0) /// Replace the channel with green e_GreenSwap(1) /// Replace the channel with blue e_BlueSwap(2)

14.1.2.6 Colour matrix control page CE_ColourMatrixCtrl []

Control Page element for color matrix (Both PIPE0 and PIPE1)	**g_CE_ColourMatrixCtrl [0] store the values from Sensor and used for PIPE0 g_CE_ColourMatrixCtrl [1] store the values from Sensor and used for PIPE1**	
Page Elements	**Description**	**Range**
s16_Offset_R	ColourMatrixOffset_R	[-1024,+1023]
s16_Offset_G	ColourMatrixOffset_G	[-1024,+1023]
s16_Offset_B	ColourMatrixOffset_B	[-1024,+1023]

14.1.2.7 Status page elements for color matrix

Status page element for Color matrix in fixed format, The values are programmed in Pixel Pipe	**g_CE_ColourMatrixDamped[0]: Damped matrix for PIPE0 g_CE_ColourMatrixDamped[1]: Damper matrix for PIPE1**	
Page Elements	**Description**	**Range**
s16_RedInRed	ColourMatrix[0][0]: matrix_element_RedInRed	[-1024 ; +1023]
s16_GreenInRed	ColourMatrix[0][1]: matrix_element_GreenInRed	[-1024 ; +1023]
s16_BlueInRed	ColourMatrix[0][2]: matrix_element_BlueInRed	[-1024 ; +1023]
s16_RedInGreen	ColourMatrix[1][0]: matrix_element_RedInGreen	[-1024 ; +1023]

s16_GreenInGreen	ColourMatrix[1][1]: matrix_element_GreenInGreen	[-1024 ; +1023]
s16_BlueInGreen	ColourMatrix[1][2]: matrix_element_BlueInGreen	[-1024 ; +1023]
s16_RedInBlue	ColourMatrix[2][0]: matrix_element_RedInBlue	[-1024 ; +1023]
s16_GreenInBlue	ColourMatrix[2][1]: matrix_element_GreenInBlue	[-1024 ; +1023]
s16_BlueInBlue	ColourMatrix[2][2]: matrix_element_BlueInBlue	[-1024 ; +1023]
s16_Offset_R	ColourMatrixOffset_R	[-1024,+1023]
s16_Offset_G	ColourMatrixOffset_G	[-1024,+1023]
s16_Offset_B	ColourMatrixOffset_B	[-1024,+1023]

14.1.3 Default settings recommendation

None

14.2 RGB to YUV Coder, contrast, saturation, Image formats

14.2.1 RGB to YUV Coder overview

The YUV-Coder module calculate 3x3 matrix coefficient and 3x1 offset to be programmed in YUV coder block. It also applies contrast and saturation programmed by user.

Image Contrast

Contrast is created by the difference in luminance, the amount of reflected light, reflected from two adjacent surfaces.

Image contrast can be altered by scaling data equally in all dimensions (whether RGB, R'G'B' or Y'CbCr).The device allows the host to control the image contrast of both the pipes individually.

Image Saturation

Color saturation control is similar to the image contrast control. The device allows the host to control the color saturation of both the pipes individually.

14.2.2 RGB to YUV Coder Usage

The module calculates Y'CbCr 3x3 matrix coefficients and offset to be programmed in ISP pipe in YUV coder hardware block. In calculating the matrix, the module applies contrast and saturation values programmed by user. The various output format and feature of YUV coder module are:

1. YUV / Y'CbCr output: Module can give following YUV format
    a. JFIF / Full range Y'CbCr data
    b. YCbCr REC601 format
    c. YCbCr REC709 format
    d. YCbCr REC709 full range format format
    e. Custom YCbCr
    f. Custom stock matrix and signal range format
2. RGB output
    g. Standard 3x10 RGB
    h. Custom 3x10 RGB
3. Contrast: contrast to be applied on RGB/YUV output
4. Color saturation: saturation to be applied on RGB/YUV output

14.2.2.1 Pre-requisites

Host should know following parameters to access the module:

1. Output required from pipe (RGB/YUV) as output format cannot be changed while streaming
2. For calculation REC601 standard is selected except REC709 is selected along with YUV as pipe output
3. In case host wants to use Transform_e_YCbCr_Custom mode than host should write corresponding page elements in custom signal range CE_CustomTransformOutputSignalRange and CustomStockMatrix page of ISP FW.

14.2.2.2 Pre BOOT parameters

None

14.2.2.3 Pre RUN parameters

Following parameters should be programmed before Run command

1. Output format requested from pipe (RGB / YUV)
2. If custom RGB / YUV mode is used, then range of Luma and chroma output should be specified. In custom YUV mode host should also specify custom stock matrix by using CustomStockMatrix page of ISP FW.

14.2.2.4 Live parameters

Following parameters can be changed while streaming

1. Any standard within the format i.e. if YUV format is selected, then JFIF, REC601, REC709 or YUV custom can be selected. For all cases other than REC709, REC601 is selected as standard for stock matrix. ISP FW supports custom stock matrix through Transform_e_YCbCr_Custom mode of Rgb_to_yuv_coder .
2. Contrast and saturation values.
3. Maximum / minimum contrast and saturation allowed by the system

14.2.3 YUV coder page elements

14.2.3.1 YUV coder control page elements

Control Structure for YUV Coder controls in Color engine	CE_YUVCoderControls[0] CE_YUVCoderControls[1]	
Page Elements	**Description**	**Range**
e_Transform_Type	Output generated by YUV coder YUV Coder can give both RGB and YUV output	/// Full Range BT Rec601 Y'CbCr output /// range for luma = 0-255 /// range for chroma = 0-255 Transform_e_YCbCr_JFIF, /// limited range BT Rec601 Y'CbCr output /// range for luma = 16-235 /// range for chroma = 16-240 Transform_e_YCbCr_Rec601, /// Full Range BT Rec709 Y'CbCr output /// range for luma = 0-255 /// range for chroma = 0-255 Transform_e_YCbCr_Rec709_FULL_RANGE, /// limited range BT Rec709 Y'CbCr output /// range for luma = 16-235 /// range for chroma = 16-240 Transform_e_YCbCr_Rec709, /// Custom requires manual update of stock matrix and signal range /// User should fill g_CE_CustomTransformOutputSignalRange[0/1] Transform_e_YCbCr_Custom, /// Standard 3x10 bit RGB output Transform_e_Rgb_Standard, /// Custom requires manual update of Luma and chroma parameters like excursion and MidPointTimes2 /// User should fill g_CE_CustomTransformOutputSignalRange[0/1] Transform_e_Rgb_Custom
u8_Contrast	Contrast Control in %. For example 100% will have value 100. 120% will apply have value 120 and will apply gain of 1.2 The value can not be less than 0 \n [DEFAULT]: 100% i.e. 100	1-255

u8_MaxContrast	Final contrast applied is clipped against u8_MaxContrast. e.g. final contrast will have values from u8_Contrast, Contrast from FadeToBlack [DEFAULT]: 200%	1-255
u8_ColorSaturation	Color Saturation Control in %. For example 100% will have value 100. 120% will apply have value 120 and will apply gain of 1.2. The value can not be less than 0. [DEFAULT]: 100% i.e. 100	1-255
u8_MaxColorSaturation	Final saturation applied is clipped against u8_MaxColorSaturation. e.g. final saturation will have values from u8_ColorSaturation only [DEFAULT]: 200%	1-255

14.2.3.2 Signal range for Custom transformation

The control page element will be used only for custom YUV or custom RGB trasform in e_Transform_Type.It is responsiblity of the host to program the Luma and chroma excursion as well as the midpoint*2 of the histogram.	**CE_CustomTransformOutputSignalRange[0] CE_CustomTransformOutputSignalRange[1]**	
Page Elements	**Description**	**Range**
u16_LumaExcursion	Difference between Maximum code and minimum Luma values YUV format can give maximum 255 as output, so difference between Max and min Luma or Y should be 255 RGB format can give maximum 1023 as output (ISP is 10 bit pipe), so difference between Max and min Luma or Y should be maximum 1023 [DEFAULT]: 255	1-1023
u16_LumaMidpointTimes2	Midpoint multiplied by 2 for Luma e.g. MaxY = 220, MinY = 20, So midpoint will be = (20 + (220 - 20)/2) ∗ 2 = 240 [DEFAULT]: 255	2-1023
u16_ChromaExcursion	Difference between Maximum code and minimum Chroma values YUV format can give maximum 255 as output, so difference between Max and min chroma or Cb/Cr will be 255 RGB format can give maximum 1023 as output (ISP is 10 bit pipe), so difference between Max and min should be maximum 1023 [DEFAULT]: 255	1-1023
u16_ChromaMidpointTimes2	Midpoint multiplied by 2 for Chroma e.g. Max = 220, Min = 20, So midpoint will be = (20 + (220 - 20)/2) ∗ 2 = 240 [DEFAULT]: 255	2-1023

14.2.3.3 Stock matrix for Custom transformation

The control page element will be used only for custom YUV transform in e_Transform_Type.It is responsiblity of the host to program custom stock matrix and signal range when Transform_e_YCbCr_Custom is selected.	**CE_ g_CustomStockMatrix [0] CE_ g_CustomStockMatrix [1]**	
Page Elements	**Description**	**Range**
f_StockMatrix[9]	Stock matrix to be programmed by host when Transform_e_YCbCr_Custom mode of rgb2yuvcoder mode is selected. f_StockMatrix[0] - f_StockMatrix[2] correspond to Y component, f_StockMatrix[3] - f_StockMatrix[5] correspond to U component stock matrix, and f_StockMatrix[6] - f_StockMatrix[8] correspond to V component.	Float. Following 3 constraints must be met by elements of array f_StockMatrix[9]. f_StockMatrix[0]+ f_StockMatrix[1]+ f_StockMatrix[2] = 1. f_StockMatrix[3]+ f_StockMatrix[4]+

		f_StockMatrix[5] = 0. And f_StockMatrix[6]+ f_StockMatrix[7]+ f_StockMatrix[8] = 0.

14.2.3.3.1 Fade to Black

Control Page element for Fade to Black feature. In a very dark scene the gains applied to the image data will increase to maintain the corRECt exposure. If the scene is too dark this will not be possible and a dark image with noise artefacts will be the output. The fade to black system ensures that in this situation a noise free black image is output by the system.	**CE_FadeToBlack[0] CE_FadeToBlack[1]**	
Page Elements	**Description**	**Range**
f_BlackValue	Minimum possible damper output. 0.0 fades to absolute black. 1.0 effectively disables fade to black	
f_DamperLowThreshold	Low Threshold (in Exposure compiler's gained mSecs) for calculating damper slope.	
f_DamperHighThreshold	High Threshold (in Exposure compiler's gained mSecs) for calculating scythe damper slope.	
f_DamperOutput	Status Page element to allow tracking of damper [NOTE]: Host should not modify this page element	
e_Flag_Disable	Fade to black control [DEFAULT]: Flag_e_TRUE (Fade to Black feature is disabled)	Flag_e_FALSE: Enable the Fade to Black feature Flag_e_TRUE: Disable the Fade to Black feature

14.2.3.3.2 3x3 Matrix coefficient calculated by module

The status pages has 3x3 matrix elements to be applied on the rgb2yuv coder (in color engine) on corresponding Pipe. The matrix is dependant on the selected transform type, contrast and color saturation values of the histogram	CE_OutputCoderMatrix[0] CE_OutputCoderMatrix[1]	
Page Elements	**Description**	**Range**
s16_w0_0	YUVCODER_Y_COF00: Conversion matrix coefficient 00	
s16_w0_1	YUVCODER_Y_COF00: Conversion matrix coefficient 01	
s16_w0_2	YUVCODER_Y_COF00: Conversion matrix coefficient 02	
s16_w1_0	YUVCODER_CB_COF10 Conversion matrix coefficient 10	
s16_w1_1	YUVCODER_CB_COF10 Conversion matrix coefficient 11	
s16_w1_2	YUVCODER_CB_COF10 Conversion matrix coefficient 12	
s16_w2_0	YUVCODER_CR_COF20 Conversion matrix coefficient 20	
s16_w2_1	YUVCODER_CR_COF20 Conversion matrix coefficient 21	
s16_w2_2	YUVCODER_CR_COF20 Conversion matrix coefficient 22	

14.2.3.3.3 3x1 offset matrix calculated by module

The status pages has 3x1 offset matrix elements to be applied on the rgb2yuv coder (in color engine) on corresponding Pipe	CE_OutputCoderOffsetVector[0] CE_OutputCoderOffsetVector[1]	
Page Elements	**Description**	**Range**
s16_i0	YUVCODER_YFLOOR Y data floor	
s16_i1	YUVCODER_CBFLOOR Cb data floor	
s16_i2	YUVCODER_CRFLOOR Cr data floor	

14.2.3.3.4 Applied contrast and saturation values

Status page element for applied contrast and saturation in PIPE	YUVCoderStatus[0] YUVCoderStatus[1]	
Page Elements	**Description**	**Range**
f_Contrast	Final contrast applied, the value will be clipped against maximum contrast	
f_Saturation	Final saturation applied, the value will be clipped against maximum saturation	

14.2.3.4 Default settings RECommendation

1. CE_YUVCoderControls[0/1]. e_Transform_Type = Transform_e_YCbCr_JFIF
2. CE_FadeToBlack[0/1]. e_Flag_Disable = Flag_e_TRUE

14.3 Exposure

14.3.1 Overview

The Exposure Algorithm is one of the most important pre-capture algorithms required for the better quality of the final image. The overall effect of this module is that the final image gets well exposed. Exposure of the image can be controlled through following 3 parameters:

a) Integration time: The duration for which each pixel is exposed to the light. The parameter is applied in sensor in terms of coarse integration lines and fine integration pixels.

b) Analogue Gain: Any gain factor applied by sensor in analogue domain.

c) Digital gain: The gain applied after converting any value in digital domain is termed as digital gain. It can be applied in sensor if it supports it. But in ISP architecture, digital gain is always applied in ISP HW by channel gains block.

14.3.2 Description

Exposure is the amount of light that reaches the image sensor. This amount of light determines the brightness of the image. So, for an image to be of good quality, the amount of light that reaches the sensor should be optimum.

14.3.3 Usage

14.3.3.1 Pre-Requisites

14.3.3.2 Pre Boot Parameters

None

14.3.3.3 Pre Run Parameters

None

14.3.3.4 Live Parameters

All the page elements are live parameters. Any change in values will only be absorbed after SystemConfig.e_coin_ctrl toggle.

14.3.4 Pages Exposed

Status Page 2 for Exposure	*Exposure_CompilerStatus*	
Page Elements	**Description**	**Range**

f_AnalogGainPending	Analog Gain calculated by the compiler - to be applied to the sensor.	Between Maximum & Minimum Values.
f_DigitalGainPending	Digital Gain calculated by the compiler - to be applied to the pixel pipe - this is multiplied with each of the channel gains as computed by the white balance module	Between Maximum & Minimum Values.
f_CompiledExposureTime_us	Exposure Time as cal. by the Exposure Compiler taking the present frame rate into account.	
u32_TotalIntegrationTimePending	Total Current Integration Time = composite of fine integration pixels and coarse integration lines.	
u16_CoarseIntegrationPending_lines	Coarse Integration calculated by Exposure Compiler in terms of number of lines	Between Maximum & Minimum Values.
u16_FineIntegrationPending_pixels	Fine Integration calculated by Exposure Compiler in terms of number of pixels	Between Maximum & Minimum Values.
u16_AnalogGainPending_x256	Analogue gain applied in the sensor and waiting to be absorbed	Between Maximum & Minimum Values.

Parameters Applied Status Page for Exposure	***Exposure_ParametersApplied***	
Page Elements	**Description**	**Range**
f_DigitalGain	Digital Gain in coherence with the above three values which would be used for the digital gain calculations; to be applied on the pixel pipe when a frame comes with the below mentioned integration time and analog gain.	Between Maximum & Minimum Values.
u32_TotalIntegrationTime_us	Total Integration time in micro seconds.	
u16_CoarseIntegration_lines	Coarse Integration Lines programmed on the sensor	Between Maximum & Minimum Values
u16_FineIntegration_pixels	Fine Integration Pixels programmed on the sensor	Between Maximum & Minimum Values
u16_AnalogGain_x256	Analog Gain programmed on the sensor	Between Maximum & Minimum Values

Error Control Page for Exposure	***Exposure_ErrorControl***	
Page Elements	**Description**	**Range**
u8_MaximumNumberOfFrames	Number of frames for which the system should wait for the applied analog gain and integration time to appear. If they do not appear within this number of frames, an error case is generated and e_Flag_ForceInputProcUpdation is set to TRUE forcibly.	

Exposure Values for the Sensor Driver	***Exposure_DriverControls***	
Page Elements	**Description**	**Range**

u32_TotalTargetExposureTime_us	Total Exposure Time to be applied on to the Sensor.	
u32_TargetExposureTime_us	Exposure Time to be applied on to the Sensor Driver.	
u16_TargetAnalogGain_x256	Analog Gain to be applied on to the Sensor Driver	
u16_Aperture	Aperture to be applied on to the Sensor.	
u8_FlashState	FlashState to be applied on to the Sensor.	
u8_DistanceFromConvergence	DistanceFromConvergence	
e_Flag_NDFilter	Flag depicting whether NDFilter should be used	
e_Flag_AECConverged	Flag for AEC Convergence	

Error Status Page for Exposure	***Exposure_ErrorStatus***	
Page Elements	**Description**	**Range**
u8_NumberOfForcedInputProcUpdates	Gives the number of times Input Proc has been forcibly updated. Integration Time and Analog Gain applied to the sensor have not appeared in a frame for a consecutive number of frames which is as specified by the Control Page.	
u8_NumberOfConsecutiveDelayedFrames	Gives the number of consecutive frames for which the analog gain and integration time have not appeared in a frame after they have been applied on the sensor. In an ideal scenario, they appear in the second frame after they are applied.	
u8_ExposureSyncErrorCount	Gives the total count of frames for which the analog gain and integration time were out of sync	
e_Flag_ForceInputProcUpdation	Flag which indicates to the SOF isr that it has to do the forced updating of Input Proc and let the exposure control start running again even though due to some error analog gain and integration time applied on the sensor have not appeared in the 'Maximum Number Of Frames' as specified by the control page	Flag_e_TRUE(1) Flag_e_FALSE(0)

14.4 White Balance

14.4.1 Overview

White Balance is one of the core pre-capture algorithms required for the better quality of the final image. The goal of White Balance is to make white parts of the image appear white.

An object is called White when 100% of the visible light falling on it is respectively diffused. White object do not appear White under all light sources. For example if one shines green light on the object, the object looks green if this is the only light source around. Our eyes are very good at judging what is white under different light sources so the human eye adapts itself to the ambient light and we can still tell the object is White. White Balance Algorithm tries to make objects which appear white in person to look white in camera too.

White balance is a name given to a system of color correction to deal with different illumination sources. Different Illumination sources have different light spectra. With coloured illumination sources it is difficult to say if a bright colored part of the image is a luminance - colored white part or a really colored part. The goal of the white balance is to make the neutral part appear neutral. The neutral part are the scene parts whose three color components are equal i.e. are neutral.

White balance in system is achieved by applying gains in ISP via Channel gains block. The calculation of RGB gains are done by ISP FW client.

14.4.2 Usage

14.4.2.1 Pre-Requisites

None

14.4.2.2 Pre Boot Parameters

None

14.4.2.3 Pre Run Parameters

None

14.4.2.4 Live Parameters

All the Pages are Non mode static and hence are live parameters.

14.4.2.5 Modes of Operation

The White Balance takes floating point gains from user and applies them to channel gains H/W. The gains are only applied when user toggle SystemSetup.e_CoinCtrl.

14.4.3 White Balance Page Elements

14.4.3.1 WhiteBalanceControl

Control Page for WhiteBalance	WhiteBalanceControl	
Page Elements	**Description**	**Range**
f_RedManualGain	Red Gain to be used	Cannot be less than 1.0
f_GreenManualGain	Green Gain to be used	Cannot be less than 1.0
f_BlueManualGain	Blue Gain to be used	Cannot be less than 1.0

14.4.3.2 WhiteBalanceStatus

Status Page 1 for WhiteBalance	WhiteBalanceStatus	
Page Elements	Description	Range
f_RedGain	Current red channel gain	
f_GreenGain	Current Green channel gain	
f_BlueGain	Current Blue channel gain	

14.5 Glace

The Glace firmware module is responsible for programming the Glace hardware block for accumulation and exporting the Glace statistics into the host address space once the statistics are available. It allows the host to control the window over which the accumulation happens and allows it to specify an external memory address where the Glace statistics have to be copied.

14.5.1 Glace Operation

If the Glace module has been enabled by the host, accumulation is performed over the user specified window of interest. As soon as the accumulation over the specified window of interest is completed, the Glace hardware block asserts an interrupt. In context of this interrupt, the Glace IP and its data source is disabled. This allows the Glace memories to be accessed and also ensures that the Glace memories retain their values. The device then copies the statistics into a host specified address. Subsequently, an event is raised to the host to signify the availability of Glace statistics in its address space and then the device enables the Glace IP and also enables the Glace IP data source.

The Glace IP enable is aligned to the frame start by the hardware. However, the Glace IP disable is immediate. Similarly, the enable/disable of the Glace IP data source is programmed such that it is aligned to the frame start by the hardware.

The Glace firmware module ensures that the Glace memories are accessed only when the IP has been disabled. Similarly, the IP is not enabled until all the statistics have been exported by the device.

The fact that Glace IP enable and the enable of Glace IP data source is aligned to the frame boundary ensures that if the device is able to export the complete statistics before the start of the next frame, then the Glace accumulation happens on all frames. However, if the device is not able to export the complete frame statistics before the start of the next frame, then the Glace accumulation happens over alternate frames in a safe manner.

14.5.2 Modes of Operation

Through the `Glace_Control.e_GlaceOperationMode_Control` element the device allows the host to configure Glace in the following modes of operation:

`GlaceOperationMode_e_Once`: The accumulation and exporting of the statistics happens only once. Subsequently the IP and its data source is disabled.

`GlaceOperationMode_e_Disable`: No accumulation happens. The IP and its data source are disabled.

To select a particular mode of operation, the host must first program the `Glace_Control.e_GlaceOperationMode_Control` element with one of the desired modes specified. Subsequently it must then set `Glace_Control.u8_ControlUpdateCount` to be different from `Glace_Status.u8_ControlUpdateCount`. This acts as a trigger for the device to consider a new mode of operation for the module.

For `GlaceOperationMode_e_Once` mode of operation, the device will generate statistics only once and subsequently it will be disabled. If the host wants to use the device again in

GlaceOperationMode_e_Once mode of operation, it simply needs to program Glace_Control.u8_ControlUpdateCount to be different from Glace_Status.u8_ControlUpdateCount.

Care must be taken that a change in mode of operation through programming of Glace_Control.u8_ControlUpdateCount should be requested only once any such previous operation has been completed by the device.

When attempting a GlaceOperationMode_e_Once mode of operation, the end of operation is signified by an event notification by the device.

When attempting to disable the module (GlaceOperationMode_e_Disable), the host must poll on Glace_Status.u8_ControlUpdateCount to become equal to Glace_Control.u8_ControlUpdateCount.

14.5.3 Glace IP Data Source

The Glace accumulation can happen over either channel gained image data post lens shading image data. The host can specify the accumulation data source using the Glace_Control.e_GlaceDataSource element. When starting to stream, the device takes into account the value of the page element and programs the hardware block accordingly. While streaming, the parameter can be updated using the u8_ParamUpdateCount element in the Glace_Control and Glace_Status pages. Refer to section 14.5.8 for details about updating the element while streaming.

14.5.4 Size of Window of Accumulation

The Glace IP divides the complete frame into grids. Each grid is made of blocks. Typically grid_size x block_size should cover the complete cross section of the sensor output size. The host can however specify the horizontal and vertical window of accumulation size as a fraction of the current sensor output size in the Glace_Control.f_HBlockSizeFraction and Glace_Control.f_VBlockSizeFraction elements respectively. E.g. if Glace_Control.f_HBlockSizeFraction = 0.8, then the device will consider 80% of the current sensor output while computing the horizontal Glace block size. This value must be less than or equal to 1.0

14.5.5 Block Size

Horizontal and vertical block sizes are computed as follows:

```
u8_HBlockSize = (uint8_t)((f_HBlockSizeFraction * u16_GlaceInputSizeX) / u8_HGridSize);

u8_VBlockSize = (uint8_t)((f_VBlockSizeFraction * u16_GlaceInputSizeY) / u8_VGridSize);
```

where:

u16_GlaceInputSizeX x u16_GlaceInputSizeY is the output size of the sensor,

f_HBlockSizeFraction and f_VBlockSizeFraction are user inputs taken from Glace_Control.f_HBlockSizeFraction and Glace_Control.f_VBlockSizeFraction elements respectively,

u8_HGridSize and u8_VGridSize are user inputs taken from Glace_Control.u8_HGridSize and Glace_Control.u8_VGridSize elements respectively.

If grid_size x block_size > sensor_output_size then the block size is clipped to the following value:

```
block_size = int(sensor_output_size/grid_size)
```

The above logic is applied to both the horizontal and vertical block size. This will ensure that the Glace accumulation will always be aligned to the sensor output size.

When starting to stream, the device takes into account the value of the page element and programs the hardware block accordingly. While streaming, the parameter can be updated using the `u8_ParamUpdateCount` element in the `Glace_Control` and `Glace_Status` pages. Refer to section 14.5.8 for details about updating the element while streaming.

14.5.6 Grid Size

The horizontal/vertical grid size specifies the number of blocks in the horizontal/vertical cross section of accumulation.

The host can specify the horizontal and vertical grid sizes in the `Glace_Control.u8_HGridSize` and `Glace_Control.u8_VGridSize` elements respectively.

The device assumes that the grid size will be even and changes the user input to the closest lower even number if the user input is not even.

When starting to stream, the device takes into account the value of the page element and programs the hardware block accordingly. While streaming, the parameter can be updated using the `u8_ParamUpdateCount` element in the `Glace_Control` and `Glace_Status` pages. Refer to section 14.5.8 for details about updating the element while streaming.

14.5.7 Region of Interest Start

The device allows the host to specify an offset to the start of the window of accumulation. This is expressed as a fraction of the current sensor output size. This horizontal and vertical offset (expressed as a fraction of the current sensor output size) can be programmed in `Glace_Control.f_HROIStartFraction` and `Glace_Control.f_VROIStartFraction` elements respectively.

If `(grid_size x block_size) + roi_start > glace_input_size` then the `roi_start` is clipped to the following value:

```
roi_start = glace_input_size – (grid_size x block_size)
```

The above logic is applied to both horizontal and vertical start of window of accumulation.

When starting to stream, the device takes into account the value of the page element and programs the hardware block accordingly. While streaming, the parameter can be updated using the `u8_ParamUpdateCount` element in the `Glace_Control` and `Glace_Status` pages. Refer to section 14.5.8 for details about updating the element while streaming.

14.5.8 Saturation Count

The Glace hardware IP provides a count of the saturated pixels within one block of accumulation. This count is provided in 12 bit domain. However for bandwidth optimisation purposes, the saturation count is exported in 8 bit mode as 4x8 packed item i.e. 4 saturation counts in one 32 bit word. The logic used for the conversion from 12 bit mode to 8 bit mode is as follows:

```
saturated_8 = (saturated_12 * 255)/(hblock_size * vblock_size)
```

14.5.9 Updating Glace Parameters Dynamically

The device allows the host to change the Glace parameters (Glace IP data source, block size, grid size and region of interest start) dynamically while streaming in an atomic manner. To change any combination of these parameters, the host must program the corresponding page elements and then program the `u8_ParamUpdateCount` element in the `Glace_Control` page to be different from the `u8_ParamUpdateCount` element in the `Glace_Status` page. This acts as a trigger for the module which will program the hardware block with the host inputs at an appropriate and safe time.

When starting to stream, the corresponding page elements are taken into account regardless of the value of the `u8_ParamUpdateCount` element.

14.5.10 External Memory Address for Statistics Export

The host must specify the address where the device will dump the mean statistics of the Red, Green, Blue channel per block and also the saturated pixels per block. For this, the host must program a

structure of type `Glace_Statistics_ts` in its address space. The description of this structure is shown below:

```
typedef struct
{
/// Memory address where the device will dump the mean pixel /// value per block for Red channel.
        uint32_t u32_TargetRedAddress;

/// Memory address where the device will dump the mean pixel /// value per block for Green channel.
        uint32_t u32_TargetGreenAddress;

/// Memory address where the device will dump the mean pixel /// value per block for Blue channel.
        uint32_t u32_TargetBlueAddress;

/// Memory address where the device will dump the number of  /// saturated pixels per block.
        uint32_t u32_TargetSaturationAddress;

/// The horizontal grid size applicable to the currents      /// statistics
        uint32_t u32_HGridSize;

/// The horizontal grid size applicable to the current       /// statistics
        uint32_t u32_VGridSize;

} Glace_Statistics_ts;
```

The base address of this structure must be programmed by the host in `Glace_Control.ptrGlace_Statistics` element. If this value is programmed to be 0, then the device will not export the Glace statistics.

14.5.11 Statistics Cancellation/Handling during Firmware STOP

Whenever there is a need to Abort/Stop the sensor and/or Rx during streaming, and if there is a pending Statistics request, following are the ways in which the request is handled:

14.5.11.1 In case of Rx abort

[g_SystemSetup.e_Flag_abortRx_OnStop = TRUE] Pending Statistics request would be cancelled.

14.5.11.2 In case of error

[g_ErrorHandler.e_ErrorStatus != ErrorStatus_e_NONE] Pending Statistics request would be cancelled.

14.5.11.3 In case Host requests cancellation

[g_SystemSetup.e_Flag_AecStatsCancel = TRUE] Pending Statistics request would be cancelled.

In all of the above three cases, the firmware will send a (dummy) Glace event, and a dummy DMA grab notification to the Host. The firmware, thus, does not wait for the requested synchronization, and immediately sends the notification.

Further, a new status is associated to the Statistics in the FrameParamStatus.u32_StatsInvalid element, telling the host the Statistics are INVALID. The firmware ensures that there is always one and only one event associated to a single request. The HOST shall ensure it does not send a new request before having received the event associated to the previous one.

14.5.11.4 In case Host does not request cancellation

[g_SystemSetup.e_Flag_AecStatsCancel = FALSE] This use case will be checked if all of the above three cases are not true. Hence, this use case will never be hit in case of Rx Abort or in Error case.

In this case, the firmware will not move to STOPPED state until it completes the pending Statistics request. Once it completes the request (without notifying the Host), it will move to STOPPED state. Following this, it notifies the Host of the request completion.

This model ensures that the Host won't be able to issue another Stats request after it gives STOP command, if there is an already pending request. After STOP, when the Host is notified, the Host may give the next command, which will, of course, be processed once the Host issues RUN command.

14.5.12 Statistics and Zoom simultaneous request

If the HOST gives Statistics (SystemConfig coin) and Zoom (Zoom coin) commands almost at the same time (in any sequence), the firmware shall be responsible for synchronising the two commands, and ensuring that both the commands complete. Internally, the firmware will always process the Zoom command, prior to the Statistics request. If the Zoom request requires a sensor changeover, then the Statistics request will be on hold till the changeover, and subsequently the Zoom command completes.

14.5.13 Pre-requisites

None

14.5.14 Pre BOOT Parameters

None

14.5.15 Pre RUN Parameters

Glace IP Data Source

Block size

Grid Size

Region of Interest Start

The address of the external memory where the Glace statistics have to be exported.

14.5.16 Live Parameters

Modes of Operation

14.5.17 Glace Page Elements

14.5.17.1 Glace_Control

Zoom_Params		
Page Elements	Description	Range
f_HBlockSizeFraction	Specifies the horizontal size of the window of accumulation specified as a fraction of the current sensor output size. This value must be less than or equal to 1. e.g. if f_HBlockSizeFraction == 0.8, then the device will only consider 80% of the current sensor output size while computing the horizontal Glace block size. Must be programmed once only before start of streaming.	0.0-1.0
f_VBlockSizeFraction	Specifies the vertical size of the window of accumulation specified as a fraction of the current sensor output size. This value must be less than or equal to 1. e.g. if f_VBlockSizeFraction == 0.8, then the device will only consider 80% of the current sensor output size while computing the vertical Glace block size. Must be programmed once only before start of streaming.	0.0-1.0
f_HROIStartFraction	Specifies the horizontal offset for the Glace window of accumulation expressed as a fraction of the current sensor output size. This value must be less than or equal to 1. e.g. if f_HROIStartFraction == 0.05, then the horizontal offset for window of accumulation is considered to be 5% of the current sensor output size. Must be programmed once only before start of streaming.	0.0-1.0
f_VROIStartFraction	Specifies the vertical offset for the Glace window of accumulation expressed as a fraction of current sensor output size. This value must be less than or equal to 1. e.g. if f_VROIStartFraction == 0.05, then the vertical offset for window of accumulation is considered to be 5% of the current sensor output size. Must be programmed once only before start of streaming.	0.0-1.0
ptrGlace_Statistics	Address of external memory where the device will dump the Glace Statistics The device assumes the memory	

	pointer to be of type Glace_Statistics_ts. No copy will be done if ptrGlace_Statistics == 0	
u8_RedSaturationLevel	Specifies the saturation level to be considered for the Red channel. Must be programmed once only before start of streaming.	
u8_GreenSaturationLevel	Specifies the saturation level to be considered for the Green channel. Must be programmed once only before start of streaming.	
u8_BlueSaturationLevel	Specifies the saturation level to be considered for the Blue channel. Must be programmed once only before start of streaming.	
u8_HGridSize	Number of blocks in the horizontal cross section of the Glace grid. Must be programmed once only before start of streaming.	
u8_VGridSize	Number of blocks in the vertical cross section of the Glace grid. Must be programmed once only before start of streaming.	
e_GlaceOperationMode_Control	Control to manage the operation mode.. In Once mode, the device performs Glace accumulation over one frame only. When disabled, no accumulation is performed.	GlaceOperationMode_e_Disable : Glace is disabledGlaceOperationMode_e_Once : Glace accumulation done only once
e_GlaceDataSource	Control to specify the source of data at Glace input. Must be programmed once only before start of streaming.	GlaceDataSource_e_PostLensShading : Source of input data to Glace is Gridiron output GlaceDataSource_e_PostChannelGain : Source of input data to Glace is Channel Gain output
u8_ParamUpdateCount	Control count to update the Glace parameters (like roi start, block size) Together with the Glace_Status.u8_ParamUpdateCount, it provides an atomic way for the host to update the Glace parameters. When Glace_Control.u8_ParamUpdateCount != Glace_Control.u8_ParamUpdateCount the device will absorb the new Glace	

	parameters	
u8_ControlUpdateCount_debug	Control count to update the Glace operation mode control Together with the Glace_Status.u8_ControlUpdateCount, it provides an atomic way for the host to update the Glace operation mode control. When Glace_Control.u8_ControlUpdateCount != Glace_Control.u8_ControlUpdateCount it will act as a trigger for the device to absorb Glace_Control.e_GlaceOperationMode_Control NOTE: To be used only for debugging	
e_StatisticsFov	specifies whether glace geometry is calculated on the basis of master pipe FOV or on the basis of sensor FOV DEFAULT value : StatisticsFov_e_Sensor	StatisticsFov_e_Sensor, StatisticsFov_e_Master_Pipe

Table 21 Glace Control Page

14.5.17.2 Glace_Status

Zoom_Control		
Page Elements	Description	Range
u32_GlaceMultiplier	Specifies the value of multiplier to be used for computation of pixel mean	
u16_HROIStart	Specifies the absolute value of start of horizontal window of accumulation in terms of number of pixels	
u16_VROIStart	Specifies the absolute value of start of vertical window of accumulation in terms of number of pixels	
u8_HGridSize	Specifies the horizontal size of the grid in terms of number of blocks	
u8_VGridSize	Specifies the vertical size of the grid in terms of number of blocks	
u8_HBlockSize	Specifies the absolute horizontal size of the block in terms of number of pixels	
u8_VBlockSize	Specifies the absolute vertical size of the block in terms of number of pixels	
u8_GlaceShift	Specifies the value of shift to be used to computation of pixel mean	
e_GlaceOperationMode_Status	Status of the glace operation	GlaceOperationMode_e_Disable : Glace is disabled GlaceOperationMode_e_Continuous : Glace accumulation to be done continuously over frames GlaceOperationMode_e_Once : Glace accumulation done only once
u8_ParamUpdateCount	Control count to update the Glace parameters (like roi start, block size) Together with the Glace_Status.u8_ParamUpdateCount, it provides an atomic way for the host to update the Glace parameters. When Glace_Control.u8_ParamUpdateCount != Glace_Control.u8_ParamUpdateCount the device will absorb the new Glace parameters	

u8_ControlUpdateCount	Status count to update the Glace operation mode control Together with the Glace_Control.u8_ControlUpdateCount , it provides an atomic way for the host to update the Glace operation mode control. When Glace_Control.u8_ControlUpdateCount != Glace_Control.u8_ControlUpdateCount it will act as a trigger for the device to absorb Glace_Control.e_GlaceOperationMode _Control	
e_StatisticsFov	specifies whether glace geometry is calculated on the basis of master pipe FOV or on the basis of sensor FOV DEFAULT value : StatisticsFov_e_Sensor	StatisticsFov_e_Sensor, StatisticsFov_e_Master_P ipe

Table 22 Glace Status Page

14.6 Histogram

The Histogram firmware module is responsible for programming the Histogram hardware block for accumulation and exporting the Histogram statistics into the host address space once the statistics are available. It allows the host to control the window over which the accumulation happens and allows it to specify an external memory address where the Histogram statistics have to be copied.

14.6.1 Histogram Operation

If the Histogram module has been enabled by the host, accumulation is performed over the user specified window of interest. As soon as the accumulation over the specified window of interest is completed, the Histogram hardware block asserts an interrupt. In context of this interrupt, the Histogram IP and its data source is disabled. This allows the Histogram memories to be accessed. The firmware then copies the statistics into a host specified address. Subsequently, an event is raised to the host to signify the availability of Histogram statistics in its address space and then the device enables the Histogram IP and its data source.

The Histogram IP enable is aligned to the frame start by the hardware. However, the Histogram IP disable is immediate. Similarly, the enable/disable of the Histogram IP data source is programmed such that it is aligned to the frame start by the hardware.

The Histogram firmware module ensures that the Histogram memories are accessed only when the IP has been disabled. Similarly, the IP is not enabled until all the statistics have been exported by the device. The Stats are exported only when they are valid.

The fact that Histogram IP enable and the enable of Histogram IP data source is aligned to the frame boundary ensures that if the device is able to export the complete statistics before the start of the next frame, then the Histogram accumulation happens on all frames. However, if the device is not able to export the complete frame statistics before the start of the next frame, then the Histogram accumulation happens over alternate frames in a safe manner.

14.6.2 Histogram Statistics

For improved AEC (Automatic Exposure Control) and GBCE (Global Brightness and Contrast Enhancement) algorithms, histograms are gathered from the pixels in the raw Bayer image. One histogram per channel is available but only pixels from selected Region Of Interest are taken into account. An additional flexibility allows the user to dynamically define the number of bins of the histogram. This can be useful in case of highly critical time constraints to decrease the amount of statistics to read from the block.

Pixels are gathered after Linearization and Lens Shading Correction processing.

14.6.3 Modes of Operation

Through the `HistStats_Ctrl.e_HistogramMode` element the device allows the host to configure Histogram in the following modes of operation:

- `HistogramMode_e_ONCE`: The accumulation and exporting of the statistics happens only once. Subsequently the IP and its data source are disabled. **To run the Histogram again, the control coin has to be toggled**. The coin mechanism ensures that all the parameters have been updated for the Histogram IP & accumulation can be performed.
- `HistogramMode_e_IDLE`: No accumulation happens. The IP and its data source are disabled.

14.6.4 IP Data Source

The Histogram accumulation can happen over either channel gained image data post lens shading image data. The host can specify the accumulation data source using the `HistStats_Ctrl.e_HistInputSrc` element. It must be programmed once before starting the streaming operation. The default value is ‘channel_gained_image_data’.

14.6.5 Programming of Region of Interest (ROI)

The Region of Interest over which the IP has to accumulate the statistics can be programmed through the control page. The IP requires these 5 parameters for starting of the accumulation.

a.) `X_SIZE ->` The size in X for the ROI.

b.) `Y_SIZE ->` The size in Y for the ROI.

c.) `X_OFFSET ->` The start offset for the size in X-direction.

d.) `Y_OFFSET ->` The start offset for the size in Y-direction.

e.) `PIXELIN_SHIFT ->` The shift in Input Pixel entering the Stats Block.

The page elements associated to these are in the control page `HistStats_Ctrl`:

a.) `f_HistSizeRelativeToFOV_X`

b.) `f_HistSizeRelativeToFOV_Y`

c.) `f_HistOffsetRelativeToFOV_X`

d.) `f_HistOffsetRelativeToFOV_Y`

e.) `u8_HistPixelInputShift`

The `X-Size, Y-Size, X-Offset & Y-Offset` are programmed relative to the present Field Of View.

The page element `u8_HistPixelInputShift` controls the value of the pixel used for the collection of statistics. Thus, the number of bins to be used for the accumulation of the statistics can be configured with this page element.

To align the window of accumulation to the user field of view, it is necessary to program in the hardware the starting point of accumulation in the incoming image stream. The device computes the starting point of accumulation automatically based on the user field of view and the dimensions of the incoming image stream.

However, it allows the host to specify an additional offset to the start of the window of accumulation. This is expressed as a fraction of the current user field of view. This additional horizontal and vertical offset (expressed as a fraction of the current user field of view) can be programmed in `HistStats_Ctrl.f_HistOffsetRelativeToFOV_X` and `HistStats_Ctrl.f_HistOffsetRelativeToFOV_Y` elements respectively.

Consider the following example to understand the application of this page element:

```
user_fov_x = 1600

histogram_input_image_size_x = 2052

user_fov_y = 1200

histogram_input_image_size_y = 1540
```

Assume that the center of the user field of view is aligned to the center of the sensor array.

Hence the user field of view spans the region [226..1825] horizontally and [170..1369] vertically.

Obviously, if no user offset is applied, then the start of accumulation should happen from (226,170)

Now assume:

```
HistStats_Ctrl.f_HistOffsetRelativeToFOV_X = 0.05

HistStats_Ctrl.f_HistOffsetRelativeToFOV_Y = 0.05
```

➔ h_roi_start = 226 + (0.05 * 1600) = 306

➔ v_ roi_start = 170 + (0.05 * 1200) = 230

The device always ensures that the `roi_start + size < histogram_input_size`

Hence if (`size + roi_start > histogram_input_size` then the `roi_start` is clipped to the following value:

```
roi_start = histogram_input_size – size
```

The above logic is applied to both horizontal and vertical start of window of accumulation.

14.6.6 External Memory Address for Statistics Export

The host must specify the address where the device will dump the mean statistics of the Red, Green, Blue channel per block and also the saturated pixels per block. For this, the host must program a structure of type `HistStats_Ctrl_ts` in its address space. The specific page elements have been described below:

a.) `ptru32_HistRAddr ->` Address of the R Memory to which the XP70 copies the stats so that they can be accessed by the HOST

b.) `ptru32_HistGAddr ->` Address of the G Memory to which the XP70 copies the stats so that they can be accessed by the HOST

c.) `ptru32_HistBAddr ->` Address of the B Memory to which the XP70 copies the stats so that they can be accessed by the HOST

If the value in these page elements is programmed to be 0, then the device will not export the Histogram statistics.

14.6.7 Statistics Cancellation

Statistics Cancellation would happen when the pipe is streaming (HostInterface_Control.e_HostInterfaceCommand_User = HostInterfaceCommand_e_RUN) and the Host gives a STOP command, provided there is any pending Statistics Request. In that case, the firmware will send a (dummy) event back, and a dummy DMA grab notification. The firmware, thus, does not wait for the requested synchronization, and immediately sends the notification.

Also, a new status is associated to the Statistics in the FrameParamStatus.u32_StatsInvalid element, telling the host the Statistics are INVALID. The firmware ensures that there is always one and only one event associated to a single request. The HOST shall ensure it does not send a new request before having received the event associated to the previous one.

14.6.8 Statistics and Zoom simultaneous request

If the HOST gives Statistics (SystemConfig coin) and Zoom (Zoom coin) commands almost at the same time (in any sequence), the firmware shall be responsible for synchronizing the two commands, and ensuring that both the commands complete. Internally, the firmware will always process the Zoom command, prior to the Statistics request. If the Zoom request requires a sensor changeover, then the Statistics request will be on hold till the changeover, and subsequently the Zoom command completes.

14.6.9 Pre-requisites

None

14.6.10 Pre BOOT Parameters

None

14.6.11 Pre RUN Parameters

- IP Data Source
- Region of Interest XSize & YSize
- Region of Interest Offset

14.6.12 Live Parameters

Modes of Operation

14.6.13 Glace Page Elements

14.6.13.1 `HistStats_Ctrl`

HistStats_Ctrl	

Page Elements	Description	Range
ptru32_HistRAddr	Address of external memory where the device will dump the Histogram Statistics	
ptru32_HistGAddr	Address of external memory where the device will dump the Histogram Statistics	0.0-1.0
ptru32_HistBAddr	Address of external memory where the device will dump the Histogram Statistics	0.0-1.0
f_HistSizeRelativeToFOV_X	Specifies the horizontal size of the individual Histogram block specified as a fraction of the horizontal field of view. This value must be less than or equal to 1	0.0-1.0
f_HistSizeRelativeToFOV_Y	Specifies the horizontal size of the individual Histogram block specified as a fraction of the vertical field of view. This value must be less than or equal to 1	
f_HistOffsetRelativeToFOV_X	Specifies the horizontal size of the individual Histogram block specified as a fraction of the horizontal field of view. This value must be less than or equal to 1	
f_HistOffsetRelativeToFOV_Y	Specifies the horizontal size of the individual Histogram block specified as a fraction of the vertical field of view. This value must be less than or equal to 1	
u8_HistPixelInputShift	The shift in Input Pixel entering the Stats Block	
e_HistInputSrc	The data input can be either channel gained image data post lens shading image data. Default value is channel gained image data	
e_CoinCtrl_debug	Ctrl Coin for the Host for running the IP In ONCE mode. The host must toggle the coin to indicate the firmware that it has programmed the ctrl registers & now that the Ip can be run in any of the modes available. The coin should only be used for debugging purpose	Coin_e_Heads, Coin_e_Tails.
e_HistogramMode	the mode of Histogram whether, it will be ONCE/CONTINUOUS	HistogramMode_e_ONCE, HistogramMode_e_IDLE
e_StatisticsFov	specifies whether histogram geometry is calculated on the basis of master pipe FOV or on the basis of sensor FOV DEFAULT value : StatisticsFov_e_Master_Pipe	StatisticsFov_e_Sensor, StatisticsFov_e_Master_Pipe

Table 23 Histogram Control Page

14.6.13.2 HistStats_Status

HistStats_Status		
Page Elements	**Description**	**Range**
u16_HistSizeX	Specifies the X size used for accumulation of stats	

u16_HistSizeY	Specifies the Y size used for accumulation of stats	
u16_HistOffsetX	Specifies the X offset used for accumulation of stats	
u16_HistOffsetY	Specifies the X offset used for accumulation of stats	
u16_DarkestBin_R	Lowest bin index for which value in the array is not zero	
u16_BrightestBin_R	Highest bin index for which value in the array is not zero	
u16_HighestBin_R	Bin index corresponding to the maximum value in the array	
u16_DarkestBin_G	Lowest bin index for which value in the array is not zero	
u16_BrightestBin_G	Highest bin index for which value in the array is not zero	
u16_HighestBin_G	Bin index corresponding to the maximum value in the array	
u16_DarkestBin_B	Lowest bin index for which value in the array is not zero	
u16_BrightestBin_B	Highest bin index for which value in the array is not zero	
u16_HighestBin_B	Bin index corresponding to the maximum value in the array	
e_CoinStatus	The status coin used in the Coin Mechanism for controlling the Histogram IP.	
e_ExportStatus	This page element specifies the status of the EXPORT of the statistics.	ExportStatus_COMPLETE, ExportStatus_INCOMPLETE
e_StatisticsFov	specifies whether histogram geometry is calculated on the basis of master pipe FOV or on the basis of sensor FOV DEFAULT value : StatisticsFov_e_Master_Pipe	StatisticsFov_e_Sensor, StatisticsFov_e_Master_Pipe

Table 24 Histogram Status Page

14.6.14 Gamma Overview

The Gamma firmware module is responsible for programming the Gamma hardware block for Gamma IP which is being used for Gamma Correction. Gamma IP is basically gain curve applied to each of the three input image components.

GammaPixel = Gca (InputPixel)

Where GammaPixel is the output value

InputPixel is the input value

Gca is the gamma curve approximation (user-defined LUT).

This firmware module allows Host to program any Gamma Curve in LUT Memories to achieve custom gamma curve apart from standard gamma curve programming by firmware in LUT Memories.

14.6.15 Gamma Usage

There are three modes available in Gamma Control.

GammaCurve_Standard: Firmware will program LUT Memories based on standard gamma curve. LUT table is fixed at the time of compilation.

GammaCurve_Custom: Host has flexibility to program LUT Memories according to any gamma curve. Firmware will only program hardware registers to enable flextf and last pixel values for each color.

GammaCurve_Disable: Firmware will disable flextf hardware IPs to bypass gamma correction.

14.6.15.1 Pre-requisites

None

14.6.15.2 Pre BOOT parameters

None

14.6.15.3 Pre RUN parameters

Host will program intended gamma curve in CE_Gamma page element. According to that Firmware will write to LUT memories. Incase Host wants custom gamma curve, it has to first disable flextf hardware IP, then write to LUT memories.

14.6.16 Gamma Page Elements

14.6.16.1 GammaControl page elements

	g_CE_GammaControl[0]:used for PIPE0 Gamma **g_CE_GammaControl[1]:used for PIPE1 Gamma**	
Page Elements	**Description**	**Range**
U8_GammaCurve	GammaCurve	0: GammaCurve_Standard 1: GammaCurve_Custom 2: GammaCurve_Disable

14.6.16.2 Gamma last pixel value

	CE_GammaLastPixelValueControl[0/1]:used for PIPE0 and PIPE1 Gamma	
Page Elements	**Description**	**Range**
u16_Sharp_Lst_GreenGIR	Sharp Last GreenGIR Pixel	
u16_Sharp_Lst_Red	Sharp Last RED Pixel	
u16_Sharp_Lst_Blue	Sharp Last BLUE Pixel	
u16_Sharp_Lst_GreenGIB	Sharp Last GreenGIB Pixel	

u16_UnSharp_Lst_GreenGIR	Sharp Last GreenGIR Pixel	
u16_UnSharp_Lst_Red	Sharp Last RED Pixel	
u16_UnSharp_Lst_Blue	Sharp Last BLUE Pixel	
u16_UnSharp_Lst_GreenGIB	Sharp Last GreenGIB Pixel	

14.6.17 Default Settings recommendation

1. Because of hardware constraints, firmware cannot access Ce1Sharp and Ce1Unsharp Gamma Memories so Host needs to program Custom Gamma Curve in Ce1. In Ce0 Host can program Standard or Custom gamma curve according to its need but in Ce1 only Custom Gamma Curve can be programmed and host has to write all the corresponding LUT Memories by first disabling Ce1Sharp and Ce1Unsharp Flextf hardware IP followed by memory writes.

In bridge context, gamma will always be configured in custom mode.

2. Workaround for H/W bug:

   [PictorBug #95083] Programming Order of LUT is incorrect for CE Flextf

   Corresponding ISP FW bug: [PictorBug #111096] Swapped colour channels in GammaLastPixelValueControl PEs

   Due to above bug in H/W, host should implement swapping on both Last pixel values as well as LUT.

   For Page elements:

   RED PE's = program values corresponding to Green channel
   GREEN PE's = program values corresponding to Blue channel
   BLUE PE's = program values corresponding to Red channel

   For example:

   u16_Sharp_Lst_GreenGIR = Program BLUE value

   u16_Sharp_Lst_Red = Program Green value

   u16_Sharp_Lst_Blue = Program Red value

   u16_Sharp_Lst_GreenGIB = Program Green in Blue value (To be ignored in RGB configuration)

14.7 Adsoc

14.7.1 Adsoc Overview

Adsoc stands for Adaptive Directional Sharpening & Overshoot Control. The ADSOC module function is required to add a certain amount of peaking components to the scaled RGB via a sharpness filter. ADSOC uses a combination of an “average + Laplacian directional filter” in order to well detect edges. Using an average filter in the overall computation permits to be less sensitive to noise and gives a better Laplacian analysis. The sharpness filter is only applied on the green components from the general-purpose scalar module. The output of the filter is re-injected into the R, G and B components via a process called coring. A block to avoid under/over sharpening is included to avoid ringing effects along edges.

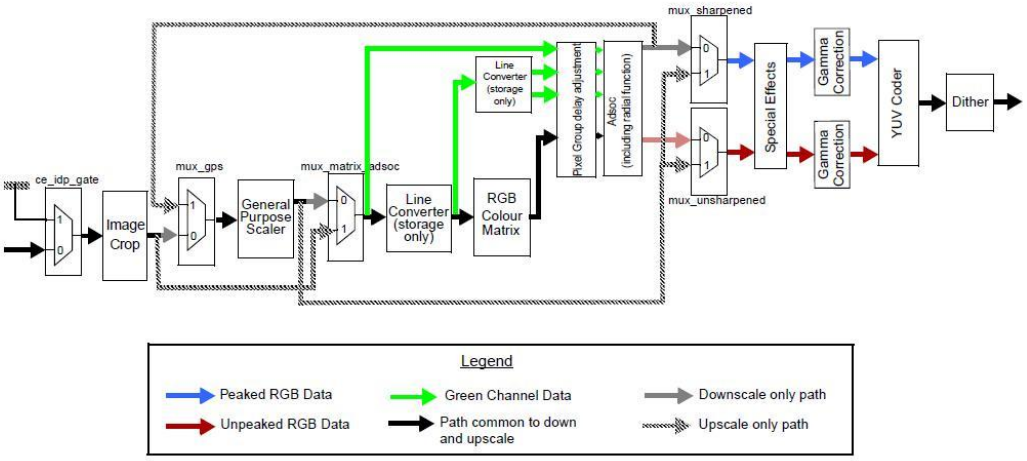


Fig 1: Colour Engine

14.7.2 Modes of Operation

The adsoc module can be operated in manual mode only. So values of all control parameters are set by HOST.

14.7.2.1 Pre-requisites

None

14.7.2.2 Pre BOOT Parameters

None

14.7.2.3 Pre RUN Parameters

14.7.2.4 Live Parameters

All Adsoc parameters are live parameters.

14.7.3 Adsoc Page Elements

g_Adsoc_PK_Ctrl

g_Adsoc_RP_Ctrl

g_Adsoc_RP_Status

14.7.3.1 g_Adsoc_PK_Ctrl

Page Elements	Description	Range
e_Flag_Adsoc_PK_Enable	Flag to enable/disable Adosc Peaking	Flag_e_TRUE Flag_e_FALSE
e_Flag_Adsoc_PK_AdaptiveSharpe	Flag to enable/disable	Flag_e_TRUE

ning_Enable	Adaptive sharpening	Flag_e_FALSE
u8_Adsoc_PK_Coring_Level	Adsoc Peaking Adaptive Coring Level	0-31
u8_Adsoc_PK_OverShoot_Gain_Bright	Adsoc Peaking Overshoot Gain Bright Ctrl	0-15
u8_Adsoc_PK_OverShoot_Gain_Dark	Adsoc Peaking Overshoot Gain dark Ctrl	0-255
u8_Adsoc_PK_Emboss_Effect_Ctrl	Peaking Emboss Effect Ctrl	0 – 7
u8_Adsoc_PK_Flipper_Ctrl	Peaking Flipper Ctrl	0 - 3
u8_Adsoc_PK_GrayBack_Ctrl	Peaking GreyBack Effect Ctrl	0 - 3
u8_Adsoc_PK_Gain	Adsoc Peaking Gain	0 - 255

14.7.3.2 g_Adsoc_RP_Ctrl

Page Elements	Description	Range
s16_Lens_Centre_HOffset	RP HOFFSET: signed relative horizontal coordinate of lens centre	0 - 65535
s16_Lens_Centre_VOffset	RP VOFFSET: signed relative vertical coordinate of lens centre	0 - 65535
e_Flag_Adsoc_RP_Enable	Adsoc Radial Peaking Enable	Flag_e_TRUE Flag_e_TRUE
u8_Radial_Adsoc_RP_Polycoef0	RP Polynomial coef 0	0 - 255
u8_Radial_Adsoc_RP_Polycoef1	RP Polynomial coef 1	0 - 255
u8_Radial_Adsoc_RP_COF_Shift	RP Polynomial cof shift	0 - 7
u8_Radial_Adsoc_RP_Out_COF_Shift	RP gain shift or out shift	0 - 7
u8_Radial_Adsoc_RP_Unity	RP Unity: Radial peaking modulation on lens center	0 - 255
u8_Adsoc_PK_Coring_Level	Adsoc Peaking Adaptive Coring Level	0 - 255

14.7.3.3 g_Adsoc_RP_Status

Page Elements	Description	Range
u16_Adsoc_RP_Scale_X	RP: radial peaking horizontal scaling factor	0 - 65535
u16_Adsoc_RP_Scale_Y	RP: radial peaking vertical scaling factor	0 - 65535
s16_Adsoc_RP_HOffset	RP HOFFSET: signed relative horizontal coordinate of lens centre	-32768 to 32767
s16_Lens_Centre_HOffset	RP VOFFSET: signed relative vertical coordinate of lens centre	-32768 to 32767

14.8 Babylon

14.8.1 Babylon Overview

Babylon also refers to the Demosaic Block. A demosaicing algorithm is a digital image process used to reconstruct a full color image from the incomplete color samples output from an image sensor overlaid with a Bayer filter. The Bayer filter has alternating red (R) and green (G) filters for odd rows and alternating green (G) and blue (B) filters for even rows. There are twice as many green filters as red or blue ones, exploiting the human eye's higher sensitivity to green light. Since each pixel of the sensor is behind a color filter, the output is an array of pixel values, each indicating a raw intensity of one of the three filter colors. Thus, an algorithm is needed to estimate for each pixel the color levels for all color components, rather than a single component. To reconstruct a full color image from the data collected by the color filtering array, a form of interpolation is performed to fill in the blanks. This process of reconstruction is called demosaicing, which is achieved by the Babylon block.

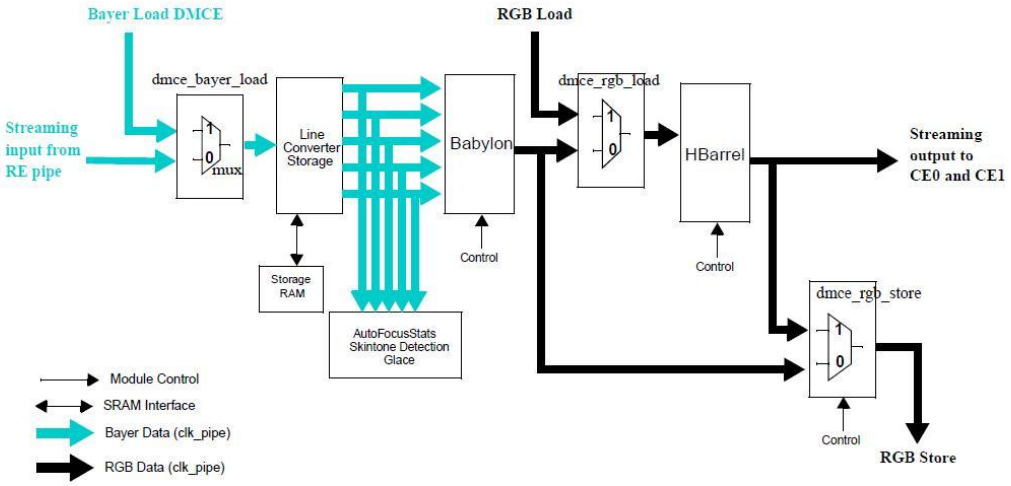


Fig 1: Demosaic Engine

The babylon module takes a 5x5 kernel of 12-bits Bayer data, and produces the corresponding RGB121212 values for the center pixel of the kernel by using adaptive interpolations with directional filtering. This algorithm ensures better quality and performances than Hamilton filter.

The behavior of the block is fine-tuned by two parameters - `flat_th` and `zipperkill`.

– Increasing the value of `flat_th` increases the probability of detecting flat regions, and thus increases the probability of uniform filtering (thus generating a smoother image). This may be useful for low-light operation, where directional filtering would be likely to add visible structures to the noise.

– Increasing the value of `zipperkill`, on the other hand, helps to reduce zipper artifacts in the output image. But, this also degrades high-contrast repeated structures like texts.

14.8.2 Usage

14.8.2.1 Pre-requisites

None

14.8.2.2 Pre BOOT Parameters

None

14.8.2.3 Pre RUN Parameters

The `Babylon_Ctrl` pages must be programmed by the host.

14.8.2.4 Live Parameters

None

14.8.3 Babylon Page Elements

14.8.3.1 Babylon_Ctrl

Page Elements	Description	Range
e_Flag_BabylonEnable	enable/disable the hw block	
u8_ZipperKill	zipper vs text trade-off control	0 - 15
u8_Flat_Threshold	flat region detect threshold	0 - 255

Table 25 Babylon_Ctrl

14.9 Binning Repair

14.9.1 Binning Repair Overview

Analog pixel binning introduces some phase errors from the image perspective as it clusterizes the output of each channel. Gir,R,B and Gib binned pixels are located around the same intersection point on the output lattice. They should ideally be spaced with the same inter pixel distances for the demosaic operation to behaves correctly. Analogue binning feature completely modifies the spatial organization of the pixels by changing the inter-pixel distance as described below:

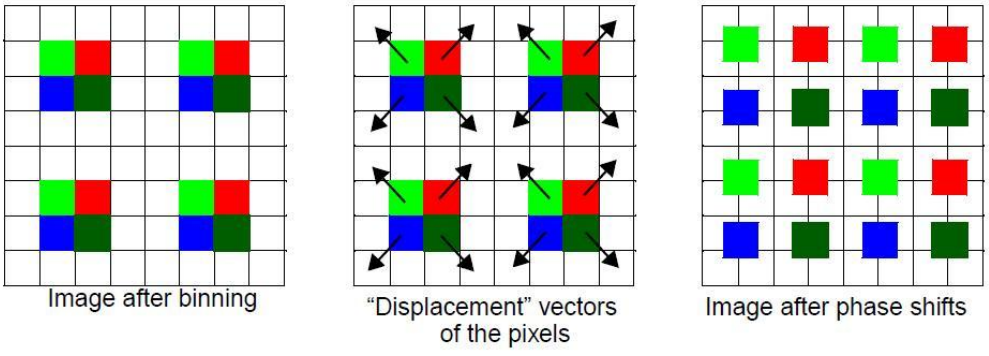


Fig 1: Binning Repair Spatial organization effect

Binning repair digitally corrects it by re-sampling the pixels at fixed frequency. Binning 2x2 and 4x4 are supported thanks to different coefficient sets. If binning is not done at sensor level, then this processing must be disabled.

Algorithm is able to process border pixels and does not change the size of the image.

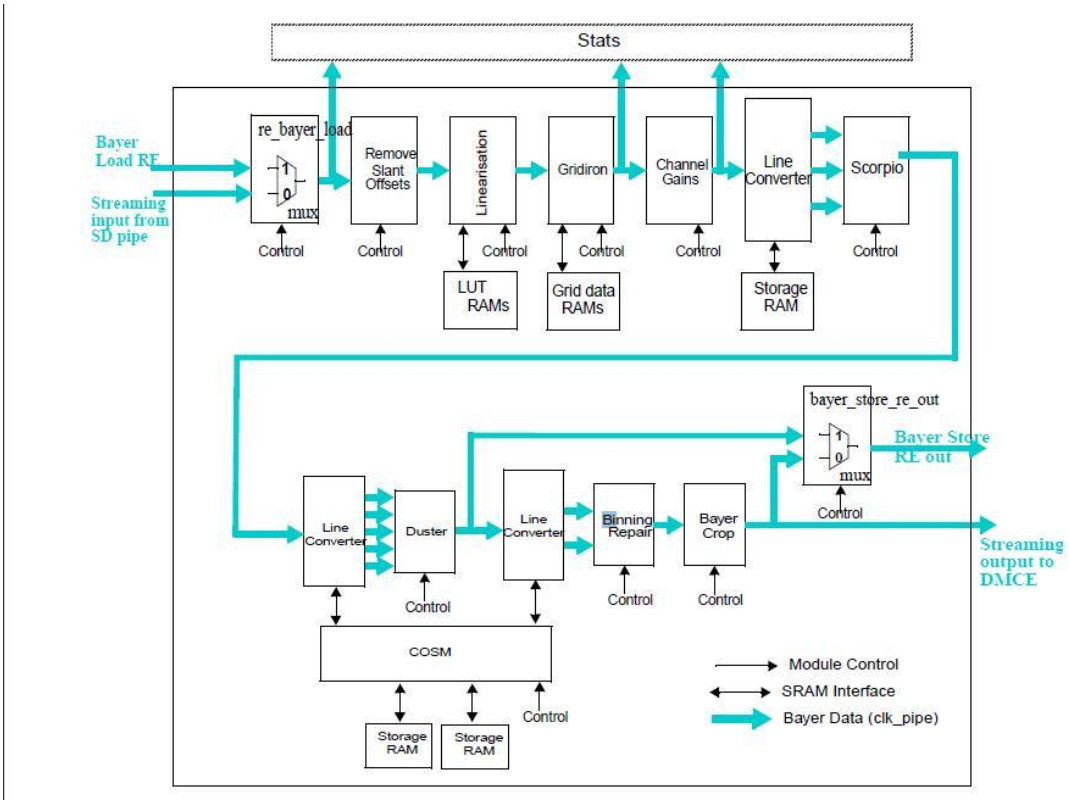


Fig 2: Recovery Engine

14.9.2 Modes of Operation

The Binning Repair block can be operated in one of the 2 different modes. The modes available are: `BinningRepairMode_e_Auto` & `BinningRepairMode_e_Custom`.

In the `BinningRepairMode_e_Auto` mode, binning factor is calculated by firmware, which is either 2 or 4. The coefficients are programmed by the firmware.

In the `BinningRepairMode_e_Custom` mode, coefficients are programmed by host.

However, a binning factor of 1 implies Binning repair block to be in disabled state.

When the binning factor is 2 or 4, the firmware automatically programs the coefficients to a fixed value. Below is the table specifying those values.

Coeffecient	Value when Binning Factor = 2	Value when Binning Factor = 4
u8_Coeff_00	1	9
u8_Coeff_01	9	39
u8_Coeff_10	7	39
u8_Coeff_11	49	169
u8_Coeff_shift	5	7

14.9.2.1 Pre-requisites

None

14.9.2.2 Pre BOOT Parameters

None

14.9.2.3 Pre RUN Parameters

The `BinningRepair_Ctrl` pages must be programmed by the host.

14.9.2.4 Live Parameters

NoneBinning Repair Page Elements

14.9.2.5 BinningRepair_Ctrl

Page Elements	Description	Range
e_Flag_BinningRepairEnable	enable/disable the hw block	
e_Flag_H_Jog_Enable	enable, disable the horizontal jog of the filter coefficients	
e_Flag_V_Jog_Enable	enable, disable the vertical jog of the filter coefficients	
e_BinningRepairMode	choose among the different modes available for the IP	BinningRepairMode_e_Auto(0), BinningRepairMode_e_Custom(1)
u8_Coeff_00	custom mode, coefficient top left	0 - 255
u8_Coeff_01	custom mode, coefficient top right	0 – 255
u8_Coeff_10	custom mode, coefficient bottom left	0 – 255
u8_Coeff_11	custom mode, coefficient bottom right	0 - 255
u8_Coeff_shift	custom mode, down shift value	0 - 7
u8_BinningRepair_factor	It is a status page, used internally by firmware. 1 indicates binning repair disabled.	

Table 26 BinningRepair_Ctrl

14.10 Duster

14.10.1 Duster Overview

The Duster firmware module is responsible for programming the Duster hardware block for Duster IP.

Duster IP is designed to remove defective pixels as well as to reduce noise from bayer pattern format image data. It works on a 5x5 pixel kernel, where nine pixels (one central and eight ring pixels) of the current bayer color are selected and manipulated. The algorithm operates in multiple stages described as:

1. The first determines if the central pixel is defective thanks to a defect map.

2. The second stage corrects the central pixel according to user choice and/or first stage result.

3. It computes the noise estimation.

4. Defect Correction is applied thanks to the ring corrector stage, which checks for a defective pixel within the working window border and correct it before correcting the defective central pixel thanks to the central corrector stage.

5. Lastly, an adaptative gaussian filter is applied to reduce the gaussian noise level within the image, which also keeps the edges sharp.

Defect Map contains the map of the known defective pixels. But, some pixels are “leaky” pixels (means they are functional, but in some conditions, may be lower than expected value) and will create defects in the image. Defect Correction stage is there to improve the overall image quality in those configurations and ensure the defect correction will be optimal any time.

The Duster block consists of the following modules:

-- Noise Level Estimator

-- Ring Corrector

-- Central Pixel Corrector

-- Gaussian Filter

-- Scythe

14.10.2 Duster Usage

Duster firmware module operates in manual mode only. So values of all Duster parameters are programmed by HOST.

14.10.2.1 Pre-requisites

None

14.10.2.2 Pre BOOT parameters

None

14.10.2.3 Pre RUN parameters:

For the very first frame after BOOT, Host has to program g_DusterOutput.u16_FrameSigma. Because firmware has no notion of frame sigma at this point of time.

14.10.3 Duster Page Elements

g_DusterControl

g_DusterStatus

14.10.3.1 DusterControl page elements

	g_DusterControl	
Page Elements	Description	Range
u16_FrameSigma	User Frame Sigma	t_uint16 range
u16_Duster_ByPass_Ctrl	Duster bypass control. This 16 bit value is mapped to DUSTER_BYPASS_CTRL hardware register.	00: (SCYTHEMAP) Scythe is used with defect pixel mapped only 01: (SCYTHEANDMAP) Scythe is used with defect pixel mapped and defect detected pixel 10: (BYPASSSCYTHE) Scythe is bypassed 11: (FORCESCYTHE) Scythe filter is always enable [6] duster_rc_enablegrad: enable gradiant correction [5] duster_rc_use_simplified: RC Threshold as it is program in register bank [4] duster_gaussian_bypass: Bypass gaussian filtering [3] duster_defcor_bypass: Bypass deftect correction [2] duster_detect_cc_bypass: Bypass Detection of defect Central Pixel [1] duster_cc_bypass: Bypass Central Corrector [0] duster_rc_bypass: Bypass Ring Corrector
e_Flag_DusterEnable	Flag to Enable/Disable Duster Block	Flag_e_TRUE Flag_e_FALSE
u8_GaussianWeight	Variable to hold Gaussian Weight for computing GaussianTh1,GaussianTh2,GaussianTh3	- 255
u8_SigmaWeight	Variable to hold Sigma Weight for computing Sigma_Gaussian	0 - 255
u8_ScytheRank_hi	Scythe Rank for High Population	0 - 255
u8_ScytheRank_lo	Scythe Rank for Low Population	0 - 255
u8_CenterCorrectionSigmaFactor		0 - 255
u8_RingCorrectionNormThr		0 - 255

14.10.3.2 g_DusterStatus page elements

	g_DusterStatus	
Page Elements	Description	Range
u16_FrameSigma	frame sigma	
u16_Gaussian_Th1	Gaussian Th1	
u16_Gaussian_Th2	Gaussian Th2	
u16_Gaussian_Th3	Gaussian Th3	
u16_S0	Sigma0 parameter	
u16_S1	Sigma1 parameter	
u16_S2	Sigma2 parameter	
u8_ScytheRank_hi	scythe rank promoted for high population	
u8_ScytheRank_lo	scythe rank promoted for lo population	
u8_ScytheSmoothControl_hi	scythe smooth control promoted for high population	
u8_ScytheSmoothControl_lo	scythe smooth control promoted for low population	
u8_CCLocalSigma_Th	Noise Level Control CC_LOCAL_SIGMA_TH	
u8_RCNorm_Th	Ring Correction Normalised Threshold RC_NORM_TH	
u8_SigmaGausssian	Gaussian filter control SIGMA_GAUSSIAN	
u8_GaussianWeight	Gaussian Sigma Weight promoted value	

14.10.4 Modes of Operation

Duster ISP FW module operates in manual mode only. In manual mode the device does not perform any computation on its own. It simply programs the inputs provided by the host into the Duster hardware.

14.10.5 User Frame Sigma

Frame sigma denotes the estimated noise standard deviation. Host can force the device to use a fixed frame sigma by programming g_DusterControl->u16_UserFrameSigma to a non-zero value. If a non-zero value has been programmed into g_DusterControl->u16_UserFrameSigma then the device will use the specified frame sigma instead of performing iterative frame sigma computations. If g_DusterControl->u16_UserFrameSigma has been programmed to zero, then the device performs iterative frame sigma computations.

Based on the value of frame sigma, the firmware computes the values of Duster_Gaussian_Th1, Duster_Gaussian_Th2, Duster_Gaussian_Th3, Duster_S0, Duster_S1, Duster_S2 and Duster_SigmaGaussian Duster_th2, Duster_th3, Duster_s0, Duster_s1 and Duster_s2.

14.10.6 GaussianWeight Parameter

GaussianWeight is applied through a promoter. Based on the promoted value of Gaussian weight and FrameSigma firmware computes a parameter called Duster_SigmaGaussian.

14.10.7 Center CorrectorWeight Parameter

Center corrector is applied through a promoter and then the final promoted value is programmed to the H/W register. The host may disable centre pixel correction by disabling the promoter.

14.10.8 RingWeight Parameter

Ring corrector is applied through a promoter and then the final promoted value is programmed to the H/W register. The host may disable ring correction by disabling the promoter.

14.10.9 GaussianWeight Parameter

Gaussian weight is applied through a promoter and then the final promoted value is programmed to the H/W register. The host may disable Gaussian weight by disabling the promoter. Based on the promoted value of GaussianWeight firmware computes the parameters Duster_Gaussian_Th1, Duster_Gaussian_Th2 and Duster_Gaussian_Th3.

14.10.10 Scythe Filter Lo/Hi

Scythe filter is applied through a promoter and then the final promoted value is programmed to H/W registers named as Scythe_Rank_Lo, Scythe_Rank_Hi, Scythe_SmoothControl_Lo and Scythe_SmoothControl_Hi. The host may disable Scythe Low/Hi filter by disabling the promoter.

14.10.11 Duster Dampers

There are following damper parameters in Duster :

Scythe_Damper : It drives Scythe_Rank_lo, Scythe_Rank_Hi, Scythe_Smooth_Lo and Scythe_Smooth_Hi.

CenterWeight Damper : It drives Duster_CC_LocalSigma_Th

RingWeight Damper : It drives Duster_RC_Norm_Th

SigmaWeight Damper : It drives Duster_SigmaGaussian

GaussianWeight Damper : It drives Duster_GaussianWeight which drives Duster_Gaussian_Th1, Duster_Gaussian_Th2 and Duster_Gaussian_Th3.

Host has to program these dampers.

14.10.12 Default Settings recommendation

NA

14.11 Gridiron

14.11.1 Gridiron Overview

The Gridiron firmware module is responsible for programming the Gridiron IP. Shading is the variation of pixel sensitivity with pixel x-y position, illuminant cast and Bayer channel, caused by a combination of optical and electrical factors in the sensor module. Gridiron is a multi-resolution technique which corrects this by using 2-D abstractions of compensating-gain surfaces which have been calibrated for 4 illuminants and the 4 Bayer channels.

Gridiron module is adaptive to the change in the live illuminant cast. It has four gridsets, one for each cast position on the black body locus. The four cast positions are chosen to cover the complete range of illumination. Each of the four gridsets has the gain surface abstraction for the four channels.

Gridiron does bilinear expansion between the gridsets to adapt to the illuminant cast. For every incoming pixel gridiron carries out the vertical and horizontal bilinear to determine the corrective gain and applies the same. In effect the gridiron does a cubic operation: between the casts, between two horizontal grid points and between two vertical grid points and applies the equalization on the fly.

14.11.2 Gridiron Usage

Gridiron module operates in manual mode only, so HOST is responsible for programming all control parameters of Gridiron.

Host should program Gridiron memories when ISP FW is in STOPPED state. At any time, either HOST or gridiron IP can access gridiron memories. In order to write to Gridiron memories, mem_init bit must be set to 1 in GRIDIRON_CONTROL register and once all memories are programmed, mem_init should be reset to 0. For writing to mem_init bit of GRIDIRON_CONTROL and Gridiron memories, host has to enable relevant clocks because by default firmware disables all the clocks before streaming. Once all the memories are downloaded correctly, host has to disable relevant clocks.

Host should program reference cast positions so as to cover complete range of illumination over black body locus. Maximum of 4 reference cast positions are supported by Gridiron module of ISP FW. Number of casts to be used can be controlled using PE GridironControl.u8_Active_Cast_Position_Count (valid value: 1 to 4).

For enabling gridiron block HOST should write Flag_e_TRUE to GridironControl .e_Flag_Enable PE.

14.11.3 Default Settings recommendation

14.11.3.1 Pre-requisites

None

14.11.3.2 Pre BOOT parameters

None

14.11.3.3 Pre RUN parameters

All control parameters except live cast are pre-run parameters. Gridiron memories must also be programmed prior to issuing RUN command.

14.11.3.4 Live parameters

Live parameters are those parameters which can be changed when pipe is streaming. Live cast (GridironControl.f_LiveCast) is the only live parameter supported by Gridiron module. After changing live cast host must toggle system coin.

14.11.4 Gridiron Page Elements

14.11.4.1 GridironControl page elements

g_GridironControl

g_GridironStatus_ts

g_GridironControl		
Page Elements	**Description**	**Range**
f_LiveCast	LiveCast Value Note: This PE can be modified when ISP FW is streaming. For change to take effect system coin must be toggled after modifying its value	+ve floating point number .
f_CastPosition0	Reference Cast Position0 Note: This PE should be modified only when ISP FW is in STOPPED state.	+ve floating point number
f_CastPosition1	Reference Cast Position1 Note: This PE should be modified only when ISP FW is in STOPPED state.	+ve floating point number
f_CastPosition2	Reference Cast Position2 Note: This PE should be modified only when ISP FW is in STOPPED state.	+ve floating point number
f_CastPosition3	Reference Cast Position3 Note: This PE should be modified only when ISP FW is in STOPPED state.	+ve floating point number
u16_GridWidth	Gridiron grid width Note: This PE should be modified only when ISP FW is in STOPPED state.	+ve integer. (grid memory size puts an upper bound on it).
u16_GridHeight	Gridiron grid height Note: This PE should be modified only when ISP FW is in STOPPED state.	+ve integer. (grid memory size puts an upper bound on it).
e_Flag_Enable	Flag to control Enable/Disable Gridiron Note: This PE should be modified only when ISP FW is in STOPPED state.	Flag_e_TRUE Flag_e_FALSE
e_Flag_VerFlip	Not being used	Flag_e_TRUE

		Flag_e_FALSE
Flag_HorFlip	Not being used	Flag_e_TRUE Flag_e_FALSE
u8_Active_Cast_Position_Count	No. of active reference cast positions. e.g. If value of this PE is 2 only first 2 reference casts (f_CastPosition0 and f_CastPosition1) will be used by Gridiron module. Note: This PE should be modified only when ISP FW is in STOPPED state.	1-4
e_PixelOrder	Characterization time bayer order. It is sensor dependent. Host should find default bayer order for the sensor being used, and program this page element accordingly. Default : PixelOrder_e_GrR_BGb Note: This PE should be modified only when ISP FW is in STOPPED state.	PixelOrder_e_GrR_BGb, PixelOrder_e_RGr_GbB, PixelOrder_e_BGb_GrR, PixelOrder_e_GbB_RGr

	g_GridironStatus_ts	
Page Elements	**Description**	**Range**
f_LiveCast	LiveCast Value being used by gridiron module	+ve floating point number
f_Sensor_HScale	Virtual scale factor. It is horizontal scale factor calculated by gridiron module for mapping virtual gridiron image to full FOV sensor array.	+ve floating point number
f_Sensor_VScale	Virtual scale factor. It is vertical scale factor calculated by gridiron module for mapping virtual gridiron image to full FOV sensor array.	+ve floating point number
u16_Sensor_HScale_x256	16 bit integer representation of f_Sensor_HScale	+ve integer, multiple of 16
u16_Sensor_VScale_x256	16 bit integer representation of f_Sensor_VScale	+ve integer, multiple of 16
u16_Crop_HStart	Horizontal offset in gridiron virtual image, beginning from where gridiron gains will be applied	+ve integer, multiple of 2

u16_Crop_VStart	Vertical offset in gridiron virtual image, beginning from where gridiron gains will be applied	+ve integer, multiple of 2
u16_Image_HSize	Image width	+ve integer
u16_Image_VSize	Image height	+ve integer
u16_Sensor_HSize	Gridiron virtual image width. Calculated on the basis of grid size and calculated grid pitch.	+ve integer
u16_Phase	Phase of live cast with respect to the reference casts set by HOST.	+ve integer
u8_LogGridPitch	Logarithm with base 2 of pitch selected by Gridiron module.	4 to 7 (for this HW revision)
e_Flag_Cast0	Enable/disable status 0f reference cast 0	Flag_e_TRUE Flag_e_FALSE
e_Flag_Cast1	Enable/disable status 0f reference cast 1	Flag_e_TRUE Flag_e_FALSE
e_Flag_Cast2	Enable/disable status 0f reference cast 2	Flag_e_TRUE Flag_e_FALSE
e_Flag_Cast3	Enable/disable status 0f reference cast 3	Flag_e_TRUE Flag_e_FALSE
e_Flag_Enable	Status flag to enable/disable Gridiron module of ISP FW.	Flag_e_TRUE Flag_e_FALSE
e_Flag_VerFlip	Not being used	Flag_e_TRUE Flag_e_FALSE
e_Flag_HorFlip	Not being used	Flag_e_TRUE Flag_e_FALSE

14.12 Remove Slant Offset

14.12.1 RSO Overview

To achieve the desired black level of Bayer pixel data it is often necessary to apply an offset to the data. This module provides the functionality to apply a colour channel dependent DC-level offset on active pixel data i.e. not dark or black data. The offset applied to any colour channel can be a linear “slant” defined by a term in X and Y which pivots the DC level at the image centre.

RSO corrects two different issues:

1.) **Black Offset**: Offset added at the sensor output, just before the optional SMIA compression. This offset is often set at 64 by sensor (in RAW10), and is a legacy feature as not any more used. The drawback of this offset is to reduce dynamic range of (1023-64)/1023 = 6.25%
2.) **Slant Offset**: Due to analog issues in the sensor, there is a small offset to be corrected dependant of the pixel position, “bilinearly” relative to horizontal and vertical positions. Current ST sensors don’t need any more this slant, but this trouble could be present again in the future, in our sensor or the one from the competition.

Below is the block diagram of Recovery Engine. The RSO block is the 1st block in the Recovery Engine.

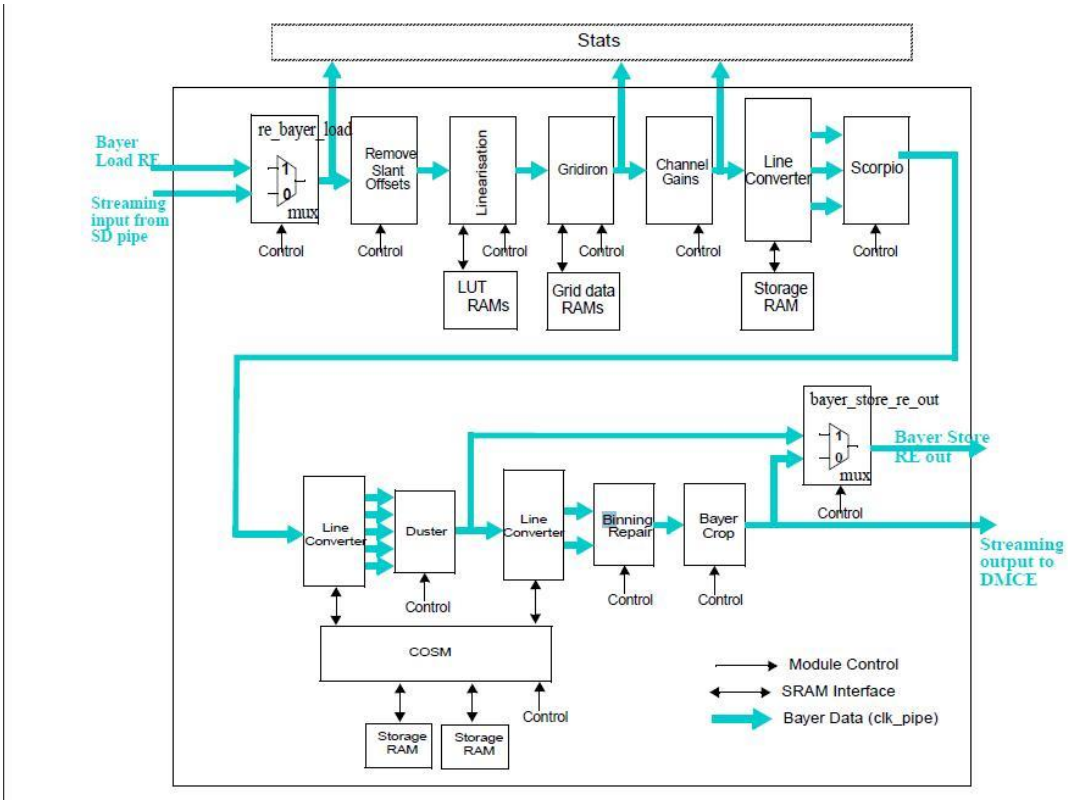


Fig 1: Recovery Engine

14.12.2 Modes of Operation

The RSO block can be operated in one of the 2 different modes. The modes available are: `RSO_Mode_e_Manual` & `RSO_Mode_e_Adaptive`.

In RSO`_Mode_e_Manual` mode, the values to be programmed in to the hardware are provided by the host in the `RSO_DataCtrl` page. These values are copied on to the `RSO_DataStatus` page.

In `RSO_Mode_e_Adaptive` mode, the firmware invokes damper on the `RSO_DataCtrl` page. The damper then damps the `RSO_DataCtrl` page & updates the `RSO_DataStatus` page.

At any point, the values in the `RSO_DataStatus` page reflect the values programmed in the hardware block.

Note: It must be ensured that the switch between the modes of operation does not happen on the fly. The system must be STOPPED before changing the mode of operation.

14.12.2.1 Pre-requisites

None

14.12.2.2 Pre BOOT Parameters

None

14.12.2.3 Pre RUN Parameters

The `RSO_DataCtrl` & `RSO_Control` pages must be programmed by the host. Before switching to `RSO_Mode_e_Adaptive` mode, the host must ensure that the dampers related to the RSO module have been updated.

14.12.2.4 Live Parameters

None

14.12.3 Working of Dampers

The RSO module has 12 parameters that are damped by the firmware. The pre-requisite for the above is that the memory containing the damper related settings be programmed before giving the RUN command to the system. The parameters damped in the RSO Module are mentioned in Table 27.

Serial No.	Parameter Name	Output Matrix
1	u32_XCoefGr	f_Output[0]
2	u32_YCoefGr	f_Output[1]
3	u16_DcTermGr	f_Output[2]
4	u32_XCoefGr	f_Output[3]
5	u32_YCoefGr	f_Output[4]
6	u16_DcTermGr	f_Output[5]
7	u32_XCoefGr	f_Output[6]
8	u32_YCoefGr	f_Output[7]
9	u16_DcTermGr	f_Output[8]
10	u32_XCoefGr	f_Output[9]
11	u32_YCoefGr	f_Output[10]
12	u16_DcTermGr	f_Output[11]

Table 1: RSO damped parameters

It must be noted that the slant origin parameters, `XSlantOrigin` & `YSlantOrigin` are not damped.

The damper structure associated to these are `RSO_Shared_Damper`.

The definition of shared damper is shown below:

```
typedef struct
{
float_t f_Output[u8_Parameters][u8_BasisCount0 * u8_BasisCount1];
float_t f_Base_A[u8_BasisCount0];
float_t f_Base_B[(u8_DimensionCount - 1) * u8_BasisCount1];
float_t f_DamperBase[u8_DimensionCount];
uint8_t u8_BasePoints[u8_DimensionCount];
}Shared_Damper_ts;
```

Below are the parameters for better understanding:

1.) u8_DimensionCount -> 1 for 1D damper, 2 for 2D damper
2.) u8_BasisCount0 -> number of points on base0
3.) u8_BasisCount1 -> number of points on base1
4.) u8_Parameters -> number of parameters that the shared damper will cater to. (12 for RSO)

The Output array (matrix) of the damper must be programmed in the **row-major format.**

Hypothetical Example:

Consider a damper with these parameters:

u8_DimensionCount -> 2

u8_BasisCount0 -> 3

u8_BasisCount1 -> 3

u8_Parameters -> 1

The Output matrix A of the damper is shown below:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}.$$

The programming of the output array must be in the row major format as shown below:

1	2	3	4	5	6	7	8	9

The elements 1, 4, 7, i.e. **the elements along the column represent the output values corresponding to Base A** values, and the elements 1, 2, 3, i.e. **the elements along the row, represent the output values corresponding to the Base B**.

B0 B1 B2 ← Values along Base B

A0 | 1 2 3

A1 | 4 5 6

A2 | 7 8 9

^

|

|

Values along Base A

IMOPRTANT NOTE: The RSO module has 12 parameters that are to be damped. So, the value of `u8_Parameters` is 12 here.

```
float_t f_Output[u8_Parameters][u8_BasisCount0 * u8_BasisCount1];
```

The above declaration in the damper definition also signifies that there are 12 arrays for storing the output matrices.

The `f_Output[0]` stores the output matrix values for the 1st parameter, which is `u32_XCoefGr` & the `f_Output[1]` stores the output matrix values for the 2nd parameter, which is `u32_YCoefGr`. **Table 1** lists all the parameters being damped in the RSO module along with the corresponding output matrix.

14.12.4 Remove Slant Offset Page Elements

14.12.4.1 RSO_Ctrl

Page Elements	Description	Range
e_Flag_EnableRSO	enable/disable the hw block	
e_RSO_Mode_Control	the mode of operation of IP	RSO_Mode_e_Manual(0), RSO_Mode_e_Adaptive(1)

Table 2 RSO_Ctrl

14.12.4.2 RSO_DataCtrl

Page Elements	Description	Range
u32_XCoefGr	X Coeff. for Gr Channel	(18 bits in hw)
u32_YCoefGr	Y Coeff. for Gr Channel	(18 bits in hw)
u32_XCoefR	X Coeff. for Rr Channel	(18 bits in hw)
u32_YCoefR	Y Coeff. for Rr Channel	(18 bits in hw)
u32_XCoefB	X Coeff. for Bb Channel	(18 bits in hw)
u32_YCoefB	Y Coeff. for Bb Channel	(18 bits in hw)
u32_XCoefGb	X Coeff. for Gb Channel	(18 bits in hw)
u32_YCoefGb	Y Coeff. for Gb Channel	(18 bits in hw)
u16_DcTermGr	DC Term for Gr Channel	(12 bits in hw)
u16_DcTermR	DC Term for Rr Channel	(12 bits in hw)
u16_DcTermB	DC Term for Bb Channel	(12 bits in hw)
u16_DcTermGb	DC Term for Gb Channel	(12 bits in hw)
u16_XSlantOrigin	X Slant Origin	(12 bits in hw)
u16_YSlantOrigin	Y Slant Origin	(12 bits in hw)

Table 3 RSO_DataCtrl

14.13 Scorpio

14.13.1 Scorpio Overview

Scorpio block refers to Green Imbalance reduction block. Green imbalance is manifest as unwanted energy at the Nyquist frequency on the green channel. Scorpio attenuates this energy by coring and only processes Green pixels. Coring is an operation on bipolar data which reduces its amplitude, in an additive fashion, by a certain amount towards, but not beyond, zero. The coring level is programmable.

The coring level controls the Nyquist Coring Level input to the Scorpio block. It cancels row-to-row variation in pixel response between odd/even lines. It can be seen as the maximum swing at Nyqusit which Scorpio can remove. A value of 8 is suitable for low-level RRV (row-row variation), and is like a threshold of visibility of Scorpio artifacts.

Below is the block diagram of Recovery Engine.

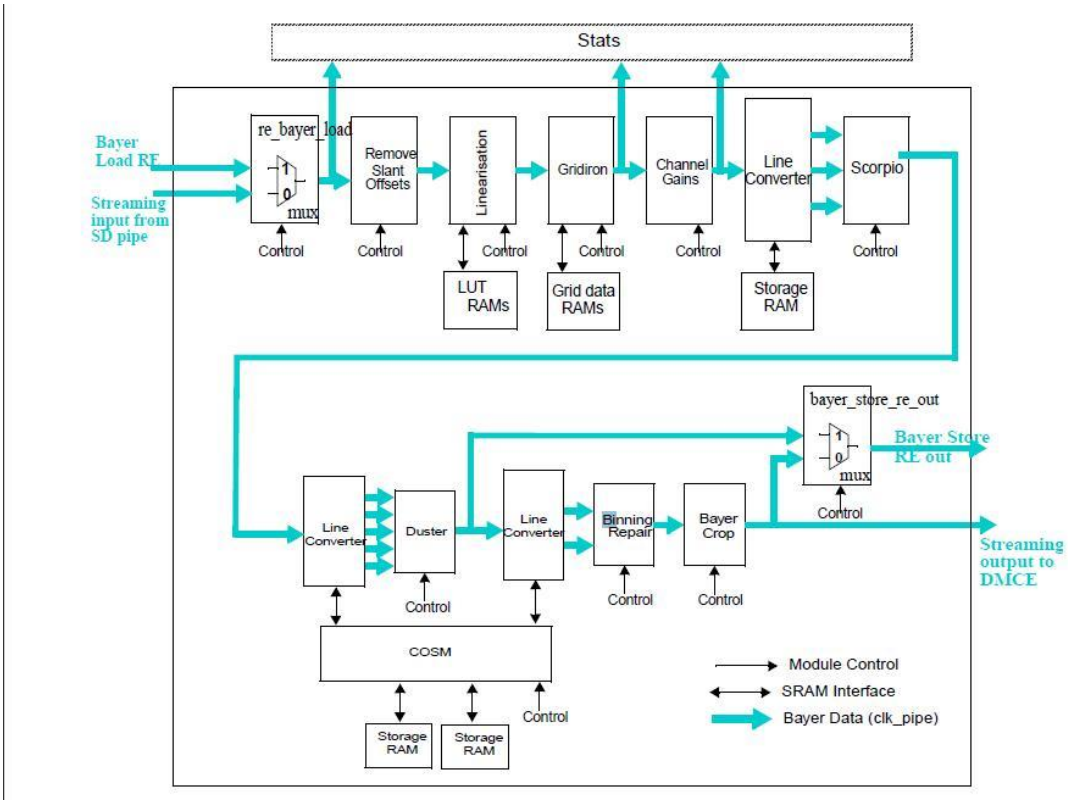


Fig 1: Recovery Engine

14.13.2 Modes of Operation

The Scorpio block can be operated in one of the 2 different modes. The modes available are: `ScorpioMode_e_Manual` & `ScorpioMode_e_Adaptive`.

In `ScorpioMode_e_Manual` mode, the values to be programmed in to the hardware are provided by the host in the `Scorpio_Ctrl.u8_CoringLevel_Ctrl` page element.

In `ScorpioMode_e_Adaptive` mode, the firmware invokes damper on the `Scorpio_Ctrl.u8_CoringLevel_Ctrl` page element. The damper then updates the `Scorpio_Ctrl.u8_CoringLevel_Status` page element with the damped value.

At any point of time, the `Scorpio_Ctrl.u8_CoringLevel_Status` specifies the value programmed in the hardware.

Note: It must be ensured that the switch between the modes of operation does not happen on the fly. The system must be STOPPED before changing the mode of operation.

14.13.2.1 Pre-requisites

None

14.13.2.2 Pre BOOT Parameters

None

14.13.2.3 Pre RUN Parameters

The `Scorpio_Ctrl` page must be programmed by the host. Before switching to `ScorpioMode_e_Adaptive` mode, the host must ensure that the dampers related to the Scorpio module have been updated.

14.13.2.4 Live Parameters

None

14.13.3 Working of Dampers

The Scorpio module has 1 parameter that is damped by the firmware. The pre-requisite for the above is that the memory containing the damper related settings be programmed before giving the RUN command to the system. The parameter damped in the Scorpio Module is `u8_ScorpioCoringLevel`. The damper structure associated to these are `SCORPIO_Shared_Damper`.

The definition of shared damper is shown below:

```
typedef struct
{
float_t f_Output[u8_Parameters][u8_BasisCount0 * u8_BasisCount1];
float_t f_Base_A[u8_BasisCount0];
float_t f_Base_B[(u8_DimensionCount - 1) * u8_BasisCount1];
float_t f_DamperBase[u8_DimensionCount];
uint8_t u8_BasePoints[u8_DimensionCount];
}Shared_Damper_ts;
```

Below are the parameters for better understanding:

5.) u8_DimensionCount -> 1 for 1D damper, 2 for 2D damper (2 for Scorpio)
6.) u8_BasisCount0 -> number of points on base0 (4 for Scorpio)
7.) u8_BasisCount1 -> number of points on base1 (2 for Scorpio)
8.) u8_Parameters -> number of parameters that the shared damper will cater to. (1 for Scorpio)

The Output array (matrix) of the damper must be programmed in the **row-major format**.

Hypothetical Example:

Consider a damper with these parameters:

u8_DimensionCount -> 2

u8_BasisCount0 -> 3

u8_BasisCount1 -> 3

u8_Parameters -> 1

The Output matrix A of the damper is shown below:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}.$$

The programming of the output array must be in the row major format as shown below:

1	2	3	4	5	6	7	8	9

The elements 1, 4, 7, i.e. **the elements along the column represent the output values corresponding to Base A** values, and the elements 1, 2, 3, i.e. **the elements along the row, represent the output values corresponding to the Base B**.

B0 B1 B2 ← Values along Base B

A0 | 1 2 3

A1 | 4 5 6

A2 | 7 8 9

^

|

| Values along Base A

14.13.4 Scorpio Page Elements

14.13.4.1 Scorpio_Ctrl

Page Elements	Description	Range
e_Flag_ScorpioEnable	enable/disable the hw block	

e_ScorpioMode	the mode of operation of IP	ScorpioMode_e_Manual(0), ScorpioMode_e_Adaptive (1)
u8_CoringLevel_Ctrl	the coring level requested by host	0 - 255
u8_CoringLevel_Status	the coring level programmed in the hardware block	0 - 255

Table 1 Scorpio_Ctrl

14.14 Sensor Data Linearization (SDL)

14.14.1 Overview

Linearization is the variation of pixel sensitivity with light intensity and Bayer channel, caused by a non-linear response of the phototransistor. It uses a down-sampled 256-input LUT to describe the sensor output vs. expected linear output. Intermediate values are linear-interpolated between 2 LUT values. Each of the 4 channels has a separate LUT to increase flexibility.

14.14.2 Usage

The device offers a simple linear ramp programming of the SDL LUT. This can be done by programming SDL_Control.e_SDLMode_Control to SDL_Linear. However the host may decide to implement a custom input to output pixel translation by programming SDL_Control.e_SDLMode_Control to SDL_Custom. In this case, it is the host responsibility to program the LUT with meaningful values before starting a streaming operation.

It is important to note that while the SDL is in enabled mode, the host cannot access the SDL LUT. Hence if the host wants to transition from Linear to Custom mode, it must first disable SDL, wait for event notification from device, reprogram the LUT for Custom mode and then enable SDL in Custom mode.

Similarly, while streaming, it is not possible to transition from Custom to Linear mode. For this, the host must disable SDL, wait for notification and then enable SDL again in linear mode.

14.14.3 Disabling SDL

SDL block can be disabled by programming SDL_Control.e_SDLMode_Control to SDL_Disable. This can be done either before starting to stream or while streaming. If SDL is disabled while streaming, then the device raises an event notification to signify that the host control for SDL has been absorbed by the device. At this point, the host may enable SDL again.

14.14.3.1 Pre-requisites

None

14.14.3.2 Pre BOOT parameters

None

14.14.3.3 Pre RUN parameters

None

14.14.3.4 Live parameters

SDL_Control.e_SDLMode_Control can be changed while streaming. Refer to section 14.14.2 and 14.14.3 for details about programming this element and the corresponding user model.

14.14.4 SDL page elements

14.14.4.1 SDL control page

SDL_Control		
Page Elements	**Description**	**Range**
e_SDLMode_Control	The control element for the SDL module	SDL_Disable: SDL module will be disabled SDL_Linear: Use linear ramps for SDL module SDL_Custom: Custom input pixel to output pixel translation, LUT will be programmed by the host. NOTE: While streaming only enable/disable of SDL can be done. ISP FW will not update any registers except enable/disable.

14.14.4.2 SDL status page

SDL_Status		
Page Elements	**Description**	**Range**
e_SDLMode_Status	The status element for the SDL module	SDL_Disable: SDL module will be disabled SDL_Linear: Use linear ramps for SDL module SDL_Custom: Custom input pixel to output pixel translation, LUT will be programmed by the host

14.14.4.3 SDL Last LUT entry

SDL_ELT		
Page Elements	**Description**	**Range**
u16_LastElementinLUT_GIR	ISP_FLEXTF_LINEAR_LAST_ELT_GREEN_GIR: Last Element for Green in Red channel	NOTE: Updated when programmed before RUN
u16_LastElementinLUT_RED	ISP_FLEXTF_LINEAR_LAST_ELT_RED: Last Element for Red channel	NOTE: Updated when programmed before RUN
u16_LastElementinLUT_BLUE	ISP_FLEXTF_LINEAR_LAST_ELT_BLUE: Last Element for Blue channel	NOTE: Updated when programmed before RUN
u16_LastElementinLUT_GIB	ISP_FLEXTF_LINEAR_LAST_ELT_GIB: Last Element for Green in Blue channel	NOTE: Updated when programmed before RUN
u16_PixelShift	ISP_FLEXTF_LINEAR_PIXELIN_SHIFT: Pixel Input Shift Index. [2:0] pixel_shift: index value of the right shift	0: 256 LUT entries to be used 1: 128 LUT entries to be used 2: 64 LUT entries to be used 3: 32 LUT entries to be used 4: 16 LUT entries to be used 5: 8 LUT entries to be used 6: 4 LUT entries to be used 7: 2 LUT entries to be used [DEFAULT]: 0, 256 LUT to be used. NOTE: Updated when programmed before RUN

15. Auto Focus

15.1 *Manual Focus Introduction*

With increasing resolution over 2 Mpixel and decreasing pixel size a fine control of the focus is required in order to get properly focused images since it becomes more and more difficult to get a depth of field from infinity till very short distances (macro, usually ~10cm).

Autofocus (AF) automatically determines the best focus condition for the current scene, using an average of the focus of the scene or automatically selecting the relevant zones accordingly to the operating mode selected. Moreover more advanced controls can let the user create more creative snapshots, taking pictures with the subject clear in focus, while less important details of the scene may be left out of focus. In view of the current scope of the Firmware. The Auto Focus Algorithm Part is not included in this document .

The purpose of this document is to describe the functionalities offered by Focus Control (FC) , AFStats , FLADriver Module ,AutoFocus Module will be added to this later.

Usually we can divide Focus related controls in the following categories:

Modes: The Scope of this Document is only at Manual Mode . There can be various Modes, but the current FW supports only the Manual Mode , where focus or Lens Movement is Manually Controlled .

Commands: The control for the Lens Movement is Configured in FW as a command interface between Host and the FC FW. In current implementation the FW requires to detect a command change in order to execute the command and a Toggle of Control coin so that the actual Movement of the Lens Start According to the Command Definition .Also if the user wants to issue two times the same command in a Set Command mode it has to only Toggle the coin and need not to again set the command (As already Set the Command Mode) , wait that it has been absorbed and the Notification of Lens Stop after the completion of the Lens Movement is received by the host .

15.1.1 Following are the steps to configure the command and complete a Lens Movement.

Configure the Lens Command : Set the e_FocusControl_LensCommand_Control of FocusControl_Controls to Command - **FocusControl_LensCommand_e_LA_CMD_XXX**.

Toggle the Coin : Toggle the command coin variable "e_Coin_Control" of FocusControl_Controls to **Coin_e_Heads/ Coin_e_Tails .**

As soon as the above toggling is done , the Command (FocusControl_LensCommand_e_LA_CMD_XXX) is absorbed by the FW and the Actual Lens Movement Starts here . The status of the command can be read from the "e_FocusControl_LensCommand_Status" of FocusControl_Status .

Now wait for the Notification of the Lens STOP .

Check for the coin status "e_Coin_Status" of FocusControl_Status.This should be equal to what has been set in step 2.

15.1.2 Focus Control Modes

Usually under FocusControl Two Types of Modes are Supported.

Manual Focus

Auto Focus

Auto focus is not targeted in this document. ISP FW only support manual control of lens, is also called as manual focus.

15.1.2.1 Manual Focus (M-F)

Allow the user/host to manually control the focus. In this mode the commands available are the ones for manual focus control . The stable flag – g_FLADriver_Status. e_Flag_LensIsMoving - informs the host about the current operation status: [Actually the status is also propagated to the host through the Notification properly, but there could be some delay in this propagation. Hence the most safe procedure is to check for the e_Flag_LensIsMoving flag .The process of the Lens Movement is explained above .]

If it reports e_Flag_FALSE operation is completed and a new command can be issued , Although the Notification is also issued to Host as soon as the Lens stop moving . So host Either poll for the g_FLADriver_Status. e_Flag_LensIsMoving flag or wait for the LENs Stop Notification in order to issue another command (if required.).

If Notification is not received the host has to wait for the current operation to complete.The g_FLADriver_LLLCtrlStatusParam. u16_CurrentPos shows the current position of the Lens according to the Low level range of the actuator described in g_FLADriver_LLLCtrlStatusParam. u16_MinPos and u16_MaxPos .Various other Control/Status of the Lens can be seen from the PE "g_FLADriver_LLLCtrlStatusParam".

Move Step Towards Infinity/Macro , host can use step u16_ManualStepSize of g_FLADriver_LLLCtrlStatusParam and then issue a manual command (through g_FocusControl_Controls.e_FocusControl_LensCommand_Control) to move towards infinity or macro By the step size "u16_ManualStepSize "

FocusControl_LensCommand_e_LA_CMD_MOVE_STEP_TO_INFINITY INFINITY <1> will move towards infinity end point .

FLADriver_LensCommand_e_LA_CMD_MOVE_STEP_TO_MACRO <2> will move towards macro end point .

The above command will set the mode for A Particular Manual operation , and the status of the modes set by host can be viewed from the "g_FocusControl_Status" PE .

The "e_FocusControl_LensCommand_Status" variable will show the current mode absorbed by the FW for the type of Manual processing .

The Actual Movement (the movement of the Lens) of the type (e_FocusControl_LensCommand_Control) will Happen when the host Toggles the coin . the Toggling of the coin can be done using control (FocusControl_Controls) PE's "e_Coin_Control". The value for this variable can be

Coin_e_Heads.

Coin_e_Tails .

So changing the value of "e_Coin_Control" to opposite of (e_Coin_Status of FocusControl_Status) Already set value .

The Status of the coin can be read from the 'e_Coin_Status ' Element of 'FocusControl_Status' Page.

15.1.3 Focus Control Settings at different Levels

Only the Manual Focus is used , so no need of a mode PE , which Set the Manual mode .

15.1.3.1 Pre-requisites

Host should know following parameters to access the module:

If NVM values are not Present then host should know the various Possible Parameters (Usually the NVM

values found from FLADriver_NVMStoredData PE) so that it can be programmed by the host Pre BOOT parameters.

The Orientation of the Module , especially when no NVM is present .

15.1.3.2 Pre BOOT parameters

[FLADriver_Controls] e_FLADriver_RangeDef_CtrlRange :FLADriver_RangeDef_e_NVM_LEVEL_RANGE (Default value is already set in FW .
Non of the Parameter is set under this .
:FLADriver_RangeDef_e_HOST_DEFINED_RANGE
All the parameters of FLADriver_LLLCtrlStatusParam has to be set by the host , if this range is selected .
[AFStats_Controls] e_Flag_AbsSquareEnabled : 1
[AFStats_Controls] u8_CoringValue : 1
[AFStats_Controls] e_Flag_AutoRefresh : 1

15.1.3.3 Pre RUN parameters

15.1.3.4 None

15.1.3.5 Live parameters

None

15.1.4 Toggling (coin) & Lens command mode

The coin Toggling has been explained earlier also , here various commands execution is also explained . If coin is toggled again in the same command mode , say if once toggled and received notification for the Lens stop , after that the coin is toggled again , then in this case Fw will again move the Lens to Same Target place (In This case again the LLA will be called , and the operation will happen in the same way as happens in First time , but the time of execution can be different.) Depending on the what command mode was selected.

15.1.4.1 The following Action will be performed in each Lens command Type when coin is toggled .

Command/Position PE	Action (on coin toggle)	Repeated coin toggling
Check if the command is same in both control "e_FocusControl_LensCommand_Control [FocusControl_Controls's]" and status "e_FocusControl_LensCommand_Status [FocusControl_Status's]". Check for the Control " **e_Coin_Control [FocusControl_Controls's]**" and Status Coin "**e_Coin_Status**[**FocusControl_Status's**]", if not equal it means the command is under process , wait for them to become equal . Program control page Element of FocusControl_Controls's ("e_FocusControl_LensCommand_Control") to FocusControl_LensCommand_e_LA_CMD_XXX . Program if any other Parameters Like **u16_ManualStepSize , u16_TarSetPos** or any other value from the **FLADriver_LLLCtrlStatusParam** structure in relevance to the above command . Once the value has been set Toggle the Command coin **e_Coin_Control** of **FocusControl_Controls's**. Wait for the Notification of the LENS stop , or can Poll on the **FLADriver_Status's**		

"**e_Flag_LensIsMoving**" PE.

If the Notification is received or the "**e_Flag_LensIsMoving**" becomes 0 , the command has finished its execution and now the Lens is at stop state after the Movement is Over .

Check for the Check for the Control " **e_Coin_Control [FocusControl_Controls's]**" and Status Coin "**e_Coin_Status**[**FocusControl_Status's**]", These must be equal if not then there is something wrong .

Read the value of the Lens from the u16_CurrentPos of FLADriver_LLLCtrlStatusParam PE .

NOTE : As soon as the Lens Movement stops the the Notification is send from the FW side to The host , in order to inform the host that movement of the Lens has finished and now host can issue any other command or can toggle the coin again depending on the current command mode scope [if its step to infinity/macro or the Target command then the Position can be set through the respective PE (**FLADriver_LLLCtrlStatusParam** 's Element)by the host and toggling the coin after that can make the Lens to perform the desire action].

pls make sure that once the notification has been received for the previous command issued from HOST side , then only the New command has to be issued .

Command : LA_CMD_MOVE_STEP_TO_INFINITY. Page Elements: CONTROL : FocusControl_Controls **Command[**Lens command**]**: e_FocusControl_LensCommand_Control. **Coin Cmd** [for Lens Movement] : e_Coin_Control PE : FLADriver_LLLCtrlStatusParam s16_ManualStepSize. STATUS: FocusControl_Status **Command[**Lens command**]**: e_FocusControl_LensCommand_Status **Coin Status**[for Lens Movement]: e_Coin_Status PE : FLADriver_LLLCtrlStatusParam **Position [**After Step size Movement**]** : u16_CurrentPos	Move towards infinity end point with the step size of **u16_ManualStepSize** , if lens was near to boundary and command issued then the step size can be either **u16_ManualStepSize** or [boundary pos - current lens position] whichever is less . Fw will issue the Notification irrespective of the Step size , even if the Lens is already at infinity boundary.	Move towards infinity end point with the step size of **u16_ManualStepSize** each time the coin toggles . if Already at boundary the FW will try to move at the same boundary position irrespective of the step size (the LLA will be called each time to move the Lens to same boundary position). FW will issue the Lens Notification each time the coin is toggled irrespective of the actual movement (movement or not movement)at the Lens Level.
Command : LA_CMD_MOVE_STEP_TO_MACRO Page Elements:	Move towards macro end point with the step size of **u16_ ManualStepSize** , if lens was near to macro boundary and command issued then the step size can be either	Move towards Macro end point with the step size of **u16_ManualStepSize** each time the coin toggles . if Already at Macro boundary the FW will try to move at the same boundary position

CONTROL : FocusControl_Controls **Command[**Lens command**]**: e_FocusControl_LensCommand_Control. **Coin Cmd** [for Lens Movement] : e_Coin_Control PE : FLADriver_LLLCtrlStatusParam u16_MacroNearEndPos. STATUS: FocusControl_Status **Command[**Lens command**]**: e_FocusControl_LensCommand_Status **Coin Status**[for Lens Movement]: e_Coin_Status PE : FLADriver_LLLCtrlStatusParam **Position [**After Step size Movement**]** : `u16_CurrentPos`	**u16_ManualStepSize** or [macro boundary pos - current lens position] whichever is less . Fw will issue the Notification irrespective of the Step size , even if the Lens is already at macro boundary.	irrespective of the step size (the LLA will be called each time to move the Lens to same boundary position). The result can be different in case of the current position value (**g_FLADriver_LensLowLevelParamStatus . `u16_CurrentPos`**) each time the coin is toggled , but the value would be near to the actual target position [**Note** : bug in case of Piezo Actuator the actual value sometime not reached correctly.] [**OBSERVATION** : the Lens move more nearer to target position (the difference between the target and the current position becomes less) with each subsequent toggle coin. same is the case with the **STEP_TO_INFINITY** command , but the error(not reaching to target with each subsequent toggling) is more in case of This command rather then move to infinity .This has been observed that the movement towards Macro is more error prone then the movement towards infinity .] FW will issue the Lens Notification each time the coin is toggled irrespective of the actual movement (movement or not movement)at the Lens Level.
Command : LA_CMD_GOTO_INFINITY_FAR_END Page Elements: CONTROL : FocusControl_Controls **Command[**Lens command**]**: e_FocusControl_LensCommand_Control. **Coin Cmd** [for Lens Movement] : e_Coin_Control PE : FLADriver_LLLCtrlStatusParam [Infinity far end pos]: u16_InfinityFarEndPos. STATUS: FocusControl_Status **Command[**Lens command**]**: e_FocusControl_LensCommand_Status **Coin Status**[for Lens Movement]: e_Coin_Status PE : FLADriver_LLLCtrlStatusParam **Position [**current position of Lens after	This command moves the lens to far end point , some time also called as optical infinity(worst case infinity , infinity looking down),the position in case of Piezo actuator is read from the NVM and this command makes the lens to move to this position , this position is less then the horizontal infinity position . **OBSERVATION**: [sometime reached to exact far end point position but sometime near to far end point , the difference of actual and target position is less as compared to **MACRO_NEAR_END** command .] NOTIFICATION is issued by FW to host when the command finishes its execution . **NOTE :**[if already at boundary and coin is toggled the position remains the same or near to it ,but notification is issued (even if the Lens is at boundary.)]	Toggling the coin again , will make the Lens to move the same target position . **OBSERVATION:** [it has been observed that during this command mode , toggling the coin again and again, actually makes the Lens to move to near Boundary position]. Notification is issued each time the coin is toggled no matter whatever the position is achieved .

command] : u16_CurrentPos		
Command : LA_CMD_GOTO_MACRO_NEAR_END Page Elements: CONTROL : FocusControl_Controls **Command[**Lens command**]**: e_FocusControl_LensCommand_Control. **Coin Cmd** [for Lens Movement] : e_Coin_Control PE : FLADriver_LLLCtrlStatusParam [Macro Near End]: u16_MacroNearEndPos. STATUS: FocusControl_Status **Command[**Lens command**]**: e_FocusControl_LensCommand_Status **Coin Status**[for Lens Movement]: e_Coin_Status PE : FLADriver_LLLCtrlStatusParam **Current POS [**current position of Lens after command**]**:: u16_CurrentPos	This command move the Lens to macro near end point or optical Macro position .(worst case Macro , macro looking up). This position in case of PIEZO read from the NVM and the Fw actually makes the Lens to move to this position when the command coin is toggled . **OBSERVATION**:[it has been observed that the Lens never moves to exact position , when the 1st time command coin is toggled , the the difference between the target and the actual position remains much as compared to the movement to FAR end point .] Notification is issued by the FW in any case , independent of the actual position is reached or not .	Toggling the coin again and again will make the Lens to move to position near the Macro end point . Each time the coin is toggled the difference in position between the actual and target becomes less and finally to the Macro end point . Notification is issued in every command coin toggling , irrespective of the Lens actual movement happens or not .
Command : LA_CMD_GOTO_REST Page Elements: CONTROL : FocusControl_Controls **Command[**Lens command**]**: e_FocusControl_LensCommand_Control. **Coin Cmd** [for Lens Movement] : e_Coin_Control PE : FLADriver_LLLCtrlStatusParam [Rest Position]: u16_RestPos. STATUS: FocusControl_Status **Command[**Lens command**]**:	This position is again read from the NVM in Piezo actuator case . the FW try to move the Lens to this position when the command is toggled . this is actually the position of the Lens when no Manual Action is provided to the Lens . this is the position usually called as the Low power state position (the position of the lens where the power consumed by it is very Less .)In case of the Piezo actuator , this is same as that of the Infinity position , this has been observed while testing the various module , but could be different for the different version of the same Piezo module. After the toggling of the coin "**e_Coin_Control**" the Host can check for the Position reach at **u16_CurrentPos of FLADriver_LLLCtrlStatusParam** .	Toggling coin again and again makes the Lens to move the same position or nearby position and notification is issued each time the coin is toggled irrespective of the actual movement happens at the lens Level or not .

e_FocusControl_LensCommand_Status **Coin Status**[for Lens Movement]: e_Coin_Status PE : FLADriver_LLLCtrlStatusParam **Current POS [**current position of Lens after command**]:** u16_CurrentPos		
Command : LA_CMD_GOTO_TARGET_POSITION Page Elements: **CONTROL : FocusControl_Controls** **Command[**Lens command**]**: e_FocusControl_LensCommand_Control. **Coin Cmd** [for Lens Movement] : e_Coin_Control. PE : FLADriver_LLLCtrlStatusParam [Target Pos]: u16_TarSetPos. STATUS: FocusControl_Status **Command[**Lens command**]**: e_FocusControl_LensCommand_Status **Coin Status**[for Lens Movement]: e_Coin_Status PE : FLADriver_LLLCtrlStatusParam **Current POS [**current position of Lens after command**]:** u16_CurrentPos	Target Position is the position actually set by the host to move the Lens to a absolute position (**FLADriver_LLLCtrlStatusParam's u16_TarSetPos** knowing that the host has the notion about the low level position of **Piezo actuator** and its orientation (host know the actual Macro and infinity direction for the **Piezo actuator** , it can be reversed or in standard direction (Macro towards lower side and infinity towards greater side))). In any case Notification is issued after this command is processed by the fw (independent of the Actual movement of the Lens happens or not). HOST TARGET POSITON settings : Host can set the target Position with the help of the PE control (**FLADriver_LLLCtrlStatusParam 's u16_TarSetPos**). After setting this PE the host has to toggle a coin "e_Coin_Control.", The target position reached can be seen from the "**u16_CurrentPos" of** FLADriver_LLLCtrlStatusParam.	Toggling coin again and gain for this command mode will make the lens to reach at the same position and notification is always received when the coin is toggled . Even if the Host remains in the same command mode , he can set the New target position as specified in the previous column and toggling the coin will make the lens to move to new position and the notification will be issued by the Fw to Host as soon as the Lens stops its movement .
Command : LA_CMD_GOTO_INFINITY_HOR Page Elements: CONTROL : FocusControl_Controls **Command[**Lens command**]**: e_FocusControl_LensCommand_Control. **Coin Cmd** [for Lens Movement] : e_Coin_Control. PE : FLADriver_LLLCtrlStatusParam	This command takes the Lens to the infinity horizontal position , Again this position in case of the Piezo actuator is read from the NVM .the Horizontal infinity position is the position which makes the object at infinity {greater (>) 1m distance }to perfect focus , provided that the camera or Lens is kept horizontal to the surface . Again notification is issued when the command coin is toggled irrespective of the current position	Again and again toggling the coin will make the Lens to reach to locality of the hor infinity position and notification for each command coin toggling is issued by the FW to host signifying that the Lens has been stopped .

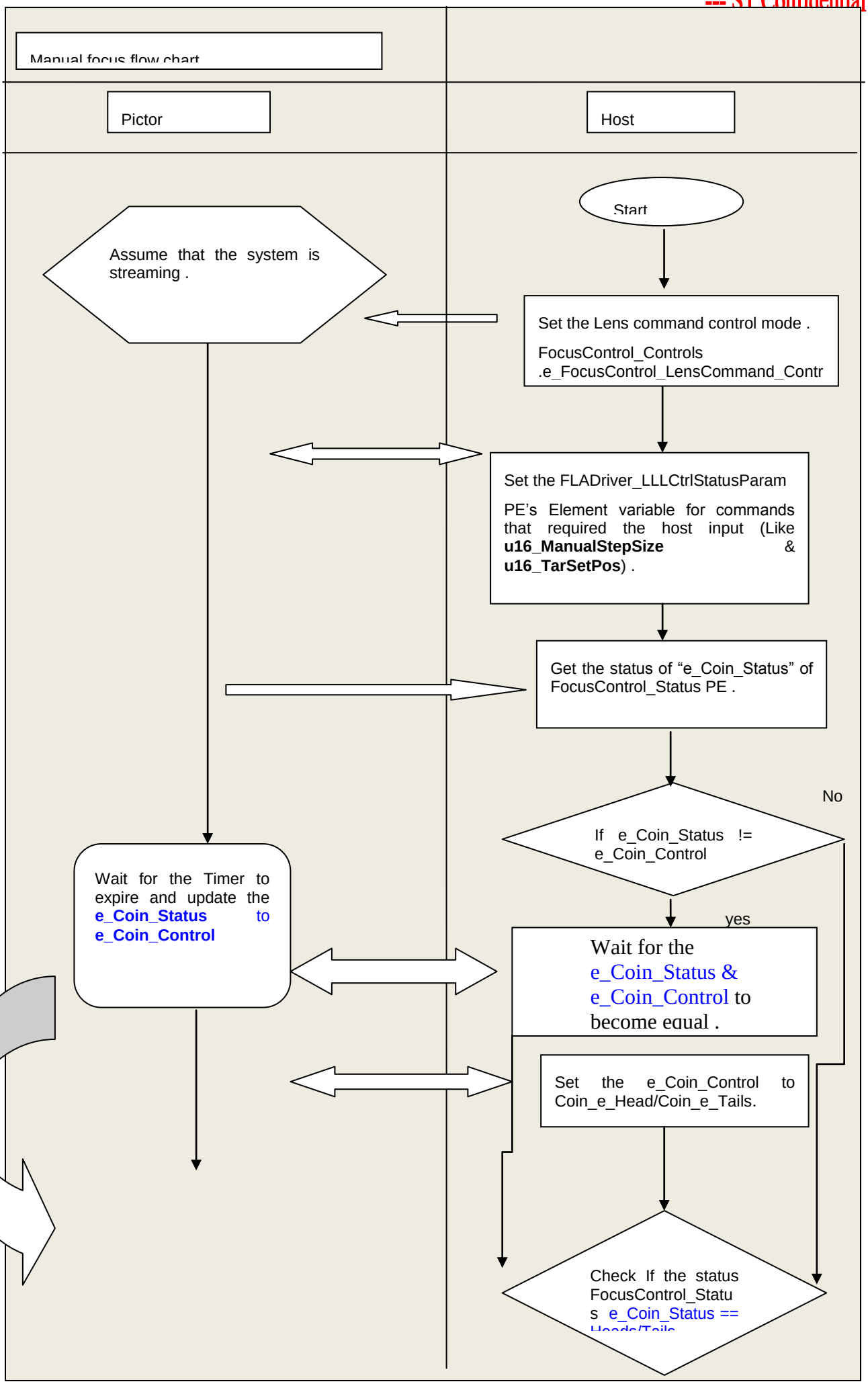
[Infinity hor Pos]: u16_InfinityHorPos. STATUS: FocusControl_Status **Command[**Lens command**]**: e_FocusControl_LensCommand_Status **Coin Status**[for Lens Movement]: e_Coin_Status PE : FLADriver_LLLCtrlStatusParam **Current POS [**current position of Lens after command**]:** u16_CurrentPos	of the Lens . After the toggling of the coin as in above case the Position can be seen from the **"u16_CurrentPos" of FLADriver_LLLCtrlStatusParam**	
Command : LA_CMD_GOTO_MACRO_HOR Page Elements: CONTROL : FocusControl_Controls **Command[**Lens command**]**: e_FocusControl_LensCommand_Control. **Coin Cmd** [for Lens Movement] : e_Coin_Control. PE : FLADriver_LLLCtrlStatusParam [Macro hor Pos]: u16_MacroHorPos. STATUS: FocusControl_Status **Command[**Lens command**]**: e_FocusControl_LensCommand_Status **Coin Status**[for Lens Movement]: e_Coin_Status PE : FLADriver_LLLCtrlStatusParam **Current POS [**current position of Lens after command**]:** u16_CurrentPos	This Command takes the Lens to the Macro Horizontal position . this position is the position when the object placed at the 10cm distance is in perfect focus . The Value for the Macro Hor position is read from the NVM . Notification is send to host as soon as the Lens Stop its movement independent of the Actual Lens Movement happens or not the Notification is always send if the Coin is toggled in this Command Mode . The Current Lens position can be read from the "u16_CurrentPos" of FLADriver_LLLCtrlStatusParam PE.	Same as above only the position is different .
Command : LA_CMD_GOTO_HYPERFOCAL	This command makes the lens to Move to the position from where till infinity , the Object Placed > 1m distance remains at Focus .	Same as above only the position is different .

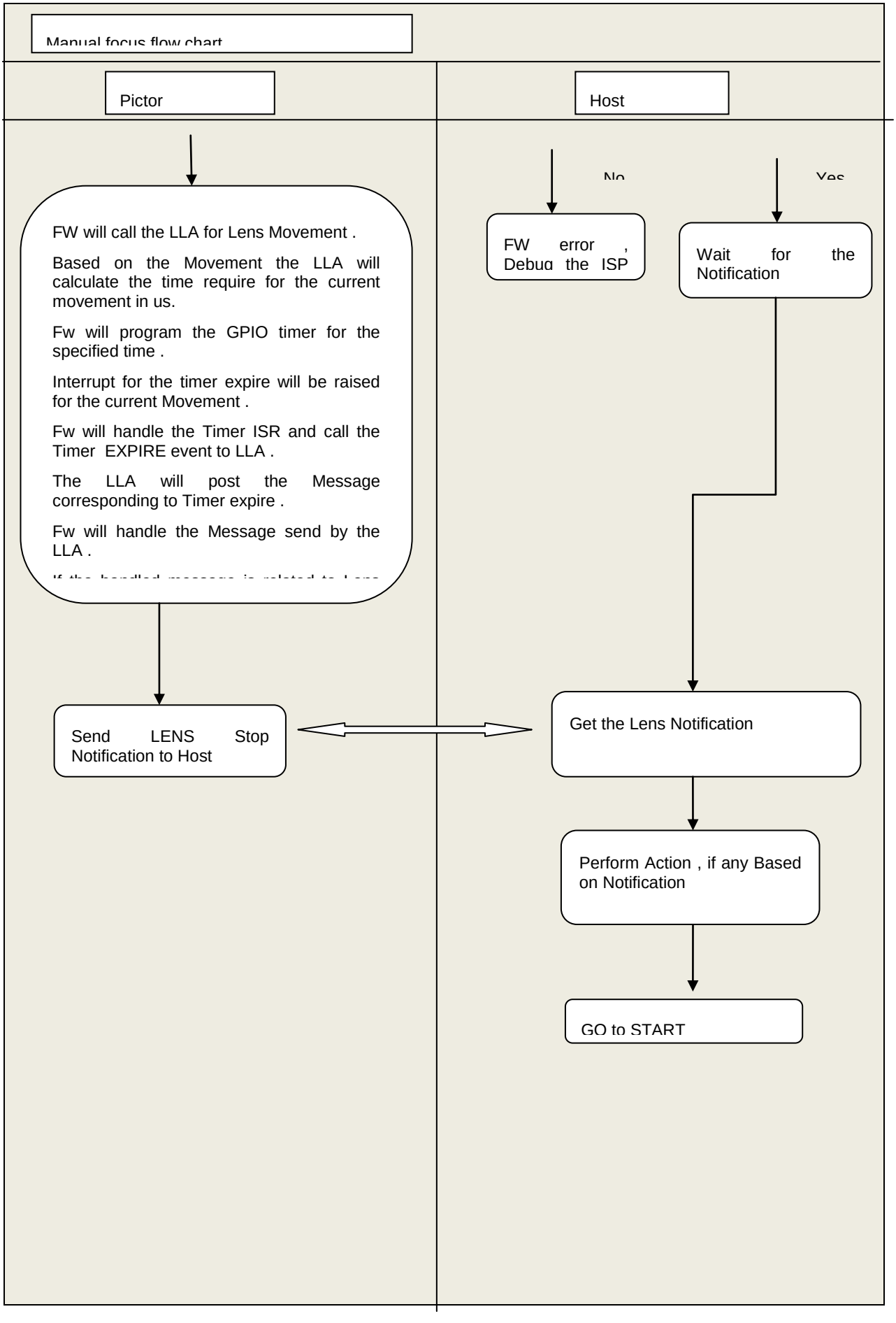
Page Elements: CONTROL : FocusControl_Controls **Command[**Lens command**]**: e_FocusControl_LensCommand_Control. **Coin Cmd** [for Lens Movement] : e_Coin_Control. PE : FLADriver_LLLCtrlStatusParam [Hyperfocal Pos]: u16_HyperfocalPos. STATUS: FocusControl_Status **Command[**Lens command**]**: e_FocusControl_LensCommand_Status **Coin Status**[for Lens Movement]: e_Coin_Status PE : FLADriver_LLLCtrlStatusParam **Current POS [**current position of Lens after command**]:** `u16_CurrentPos`	Again this Position for Piezo is Stored in NVM and Fw read it During the initialization . For Piezo This is same as Horizontal Infinity Position but can be different for Different Actuator . Notification is Also issued by theFW to the host, when the coin for this command has been toggled by the host . After the toggling of the coin as in above case the Position can be seen from the **"`u16_CurrentPos`" of FLADriver_LLLCtrlStatusParam**	

The movement will start soon after the command has been absorbed and the status flag **g_FLADriver_Status.`e_Flag_LensIsMoving`** will report the status of the movement.

[Because of the implementation there will be a few cycles for which the command has been absorbed but the movement hasn't started yet and the status flag reports stability. Usually this should be so small that the host will never notice it. Anyway if the communication with the host is so fast that he can detect this wrong state, a delay of a few ms should fix it. In the future the status update (command absorbed) must be done after the movement has been started]

15.1.4.2 Manual Focus Control chart is given below.




Manual focus flow chart
Pictor
Host
No
Yes
FW will call the LLA for Lens Movement .
Based on the Movement the LLA will calculate the time require for the current movement in us.
Fw will program the GPIO timer for the specified time .
Interrupt for the timer expire will be raised for the current Movement .
Fw will handle the Timer ISR and call the Timer EXPIRE event to LLA .
The LLA will post the Message corresponding to Timer expire .
Fw will handle the Message send by the LLA .
FW error , Debug the ISP
Wait for the Notification
Send LENS Stop Notification to Host
Get the Lens Notification
Perform Action , if any Based on Notification
GO to START

15.2 AF zones options

Autofocus algorithm uses focus measure – statistics - gathered from selected regions of interest (ROI). The regions number, positions and shape can be selected accordingly to final customer requirements.

These options take effect immediately, but it is advised to change them before starting a specific focus operation to allow the algorithm to use consistent data.

15.2.1 AF zones number, shape and size

Two preset configuration of the AF zones can be selected using the option g_**AFStats_Controls.e_AFStats_WindowsSystem_Control**.

[This control is marked ModeStatic. Actually the changes are applied as soon as the host changes the option. Anyway it is suggested to changes this option before enabling af, otherwise af will require a few frames before the new settings will be fully apllied, because of the previous history being related to a different setup.]

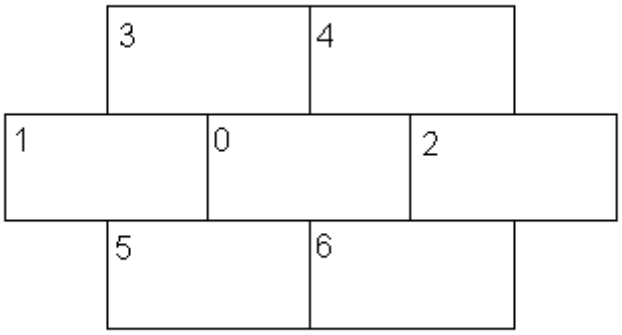


Figure 31. 7 zone eye shaped setup.

Moreover the Default size of the regions can be selected using the **g_AFStats_Controls bHRatio_Num**, **u8_HRatio_Den**, **u8_VRatio_Num**, **u8_VRatio_Den**. The applied values will be exposed in the corresponding g_AFStats_Status page. By default each zone size is set as 1/6 of the current **WOI width** and 1/9 of the current **WOI height**.

H and V ratios setting means that af ROI take roughly speaking 3*1/6 = ½ of the WOI width, and 3*1/9=1/3 of the WOI height.

For many reason – optical behavior of the lens, internal fw real time scheduling etc.- this is considered an optimal setting. Anyway the host could change them if required.

If the e_**Flag_AutoRefresh** control is set to VPIP_TRUE (default setting), then the af zones are automatically scaled when the WOI changes because of EZoom operation or because some changes in the input image size.

15.2.2 Host Zone Setup :

15.2.2.1 Configuring According to absolute value position

Host can configure the shape and size for each zone using **AFStats_HostZoneConfig** , and eventually design its own setup of zones . the interrupt for each zone comes according to the position it has acquired within the WOI running .

Host can program 5 Parameters in order to setup the zone shape and size .

The Parameters for host config page is given below in table .

For each zone host has to program the start co-ordinates and the size of the zone in terms of width and height . the host can enable/disable the zone . once disable

The interrupt for that zone doesn't comes and the AF stats are not taken for that zone. The interrupt comes according to the Y parameter of the zone . the End y near to initial y position of the WOI comes first and the interrupt for the Yend Parameters near to End of WOI Y comes last , rest accordingly .

[there may be the chance that the interrupt for the Last Zone of current Frame has not yet came , while interrupt for the 1st zone of the Next frame . a proper mechanism is implemented in order to make all the enable zone interrupts to come in sync with the stats of the current frame .AS the AFStats are start accumulating much before the next frame start s.]

15.2.2.2 Steps to configure the Zone Settings (Absolute Value)

Set the Value of all Elements from **AFStats_HostZoneConfig.**

Zone's start X position : **u16_HostAFZoneStartX**
Zone's start Y position : **u16_HostAFZoneStartY**
Zone's Width size : **u16_HostAFZoneWidth**
Zone's Height size : **u16_HostAFZoneHeight**
Enable the zone : e_Flag_Enabled
Set above elements for all the 10 zones

Set the e_Flag_HostZoneSetupInPercentage

[of AFStats_Controls] to **Flag_e_FALSE. This makes the configuration of in terms of Absolute value of the WOI.**

Read the status coin : *e_Coin_ZoneConfigStatus of* **AFStats_status.**

if Coin_e_Heads , set the e_Coin_ZoneConfigCmd to Coin_e_Tails otherwise set **Coin_e_Heads.**

Check for the Error/ Warning Message e*_AFStats_Error_Status of*

AFStats_status. If Error correct values can be set again , otherwise the Default 7 zone eye shape setup will be set **.**

NOTE : The above steps will set the zones in terms of absolute value . the host must take care of the over lapping , boundary , and correct programming of the zones start x, start y width and height of each zone , otherwise the ERROR will be shown . For good result , width and Height of each zone must be greater then *u16_AFZonesWidth & u16_AFZonesHeight.*In this case the ERROR will not be shown but FW will show the Warning Message .

15.2.2.3 Configuration according to Percentage (Wrt WOI)

Host can configure the zone start x , start y end x end y in % with respect to current FOV Width and Height used .

The typical programmable values are shown by the structure

```
typedef struct
{
        /// current Host Programmed Start X with in WOI X size .
        float_t f_HostAFZoneStartX_PER_wrt_WOIWidth;
        /// current Host Programmed Start Y with in WOI Y size .
        float_t f_HostAFZoneStartY_PER_wrt_WOIHeight;
        /// Width of the zone selected by the host .
        float_t f_HostAFZoneEndX_PER_wrt_WOIWidth;
        /// Height of the zone selected by the host .
        float_t f_HostAFZoneEndY_PER_wrt_WOIHeight;
        /// enable disable host zone
        uint8_t e_Flag_Enabled;
} AFStats_HostZoneConfigPercentage_ts;
```

All the above values are in float and can be programmed using current FOV , these values are automatically changed to the absolute start x , start y , end x end y , width & height of the zone, AFStats_HostZoneConfig_ts .

Now the host has both the choices , either it can program the percentage structure , or directly the absolute values through the AFStats_HostZoneConfig_ts .

The structure for this is shown below .

```
typedef struct
{
        /// current Host Programmed Start X with in WOI X size .
        uint16_t u16_HostAFZoneStartX;
        /// current Host Programmed Start Y with in WOI Y size .
        uint16_t u16_HostAFZoneStartY;
        /// Width of the zone selected by the host .
        uint16_t u16_HostAFZoneWidth;
        /// Height of the zone selected by the host .
        uint16_t u16_HostAFZoneHeight;
        /// enable disable host zone
        uint8_t e_Flag_Enabled;
} AFStats_HostZoneConfig_ts;
```

By default the Percentage Structure is selected . so the host can select the fields for configuring the zone in percentage of the FOV width and Height .

If host wants to set in terms of the absolute position then it can

Make the e_Flag_HostZoneSetupInPercentage element of AFStats_Controls to FALSE (Flag_e_FALSE), by default its value is TRUE (Flag_e_TRUE) in FW . the values for each structure will change its effect according to other depending on which is selected by the host .

The values of both the structure will mean the same .

15.2.2.4 Steps to configure the Zone Settings (In Percentage)

The values are configured in Terms of the

Percentage of the Current WOI width and Height , means Start x will be % factor of WOI width , similarly start y will be % factor of the Current WOI height . Same for the End x and End y Value (Both in terms of Percentage).

Set the Value of all Elements from **AFStats_HostZoneConfigPercentage.**

% WOI width : f_HostAFZoneStartX_PER_wrt_WOIWidth

% WOI Height : f_HostAFZoneStartY_PER_wrt_WOIHeight

% WOI width : f_HostAFZoneEndX_PER_wrt_WOIWidth

% WOI height : f_HostAFZoneEndY_PER_wrt_WOIHeight

Enable the zone : e_Flag_Enabled

(Note :Set above elements for all the 10 zones (only in percentage)

Set the e_Flag_HostZoneSetupInPercentage

[of AFStats_Controls] to **Flag_e_TRUE. This makes the configuration of in terms of Absolute value of the WOI.**

Read the status coin : *e_Coin_ZoneConfigStatus of* **AFStats_status.**

if Coin_e_Heads , set the e_Coin_ZoneConfigCmd to Coin_e_Tails otherwise set **Coin_e_Heads.**

Check for the Error/ Warning Message e*_AFStats_Error_Status of*

AFStats_status. If Error correct values can be set again , otherwise the Default 7 zone eye shape setup will be set **.**

NOTE : The above steps will set the zones in terms of % value of the WOI Width and Height . the host must take care of the over lapping , boundary , and correct programming of the zone's value in % of WOI Width and Height of each zone , otherwise the ERROR will be shown . the Per value Later converted to Absolute value by the FW .

15.2.3 Weighed AF stats

In weighed AF stats the weight are assigned to each region.

Usually this gives a good focus of the whole scene.

Then the regions weights should be set- **`AFStats_WeightControls`** page of elements. Each zone weight must be ≤32 to ensure overflow free calculations.

A flat average will require all the weights set to 1, otherwise more complex weighing schemes can be used.

15.2.4 Exporting AFStats To external Memory

15.2.4.1 Statistics Export Structure

```
typedef struct
{
    /// focus Measure for each zone when valid for AF .
    uint32_t u32_Focus;
    /// current Host Programmed Start X with in WOI X size .
    uint16_t u16_AFZoneStartX;
    /// current Host Programmed Start Y with in WOI Y size .
    uint16_t u16_AFZoneStartY;
    /// current Host Programmed End  X with in WOI X size .
    uint16_t u16_AFZoneEndX;
    /// current Host Programmed End Y with in WOI Y size .
    uint16_t u16_AFZoneEndY;
    /// Width of the zone
    uint16_t u16_AFZonesWidth;
    /// Height of the zone
    uint16_t u16_AFZonesHeight;
    /// Light Measure for each zone when valid for AF .
    uint8_t  u8_Light;
    /// weight assigned to zone .
    uint8_t  u8_WeightAssigned;
    /// enable disable host zone
    uint8_t e_Flag_Enabled;
} AFStats_HostZoneStatus_ts;
```

The AFStats Statistics values that is copied to external memory has following format .

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	V	LEN	FID

S0 to **S10** **:** Where are statistics of each zone as shown in above structure contains 19 Bytes for each zone .

The Statistics Structure for each zone is as follows :

STATISTICS

u32_Focus	Focus stats of zone	4bytes
u32_AFZoneStartX	Start x of zone	4bytes
u32_AFZoneStartY	Start y of zone	4bytes
u32_AFZoneEndX	End x of zone	4bytes
u32_AFZoneEndY	End y of zone	4bytes
u32_AFZonesWidth	Width of zone	4bytes
u32_AFZonesHeight	Height of zone	4bytes
u32_Light	Light stats of zone	4byte
u32_WeightAssigned	Weight assigned to zone	4byte
u32_Enabled	Enabled/Disabled Zone	4byte

Focus Measure shown by **u32_Focus (4 bytes)**

V(Lens Moving Status) : this is a Flag , if 1 it means the Valid stats , if 0 invalid stats .

This Flag can also be treated as Lens Moving Status .

Since the Valid and non valid stats is based on the Lens Movement , if the Lens is Moving the Stats will be invalid and if not the Stats will be Valid

So V = 1 Means **VALID**

V= 0 means INVALID stats **.**

AND

V = 1 Means Lens is not Moving **.**

V= 0 Means Lens is Moving **.**

LEN : this is Lens position in each frame .(DAC codes).

FID: SMIA Rx Frame ID: This is frame no for which focus statistics are exported to memory.

15.2.4.2 Host – to – AFStats communication .

Two type of interaction is performed by the host with the AF fw .

Host setup the shape and sizes for the zones and inform the Fw for the setting up the host window system .this has been explained in **host zone setup** heading above in the doc .

Also host issue the request to publish the AF stats to external memory .

These two interaction are handled through the command coin mechanism .

15.2.4.2.1 Zone configuaration Request .

Zone should be configured by the host through PE AFStats_HostZoneConfigPercentage

Explained in 1.4.2.2 Configuration according to Percentage above .

Pls refer to above heading for configuring the zone area .

After the zone configuration has been done by the host . The changed configuration will be taken into effect by the FW as soon as the host toggle the coin .

Toggling the command coin for AFStats_Controls 's e_Coin_ZoneConfigCmd element . As soon as toggling is done .(Make sure all the enabled zone parameters are programmed correctly, other wise the error will be reflected and the previous configuration would be taken by the ISP fw . New configuration will not be applied and the error is shown by the PE e_AFStats_Error_Status of AFStats_Status)the cmd to setup the zones is issued and for next zone onwards the modified setup will take place . the changed status can be seen from the PE

AFStats_HostZoneConfig (in absolute terms) and complete statistics can be seen from the AFStats_HostZoneStatus.

15.2.4.2.2 Request for AF stats

Host can issue a stats request to fw through cmd e_REQ_STATS of AFStats_HostCmd page element . once the request is encountered to fw from the Host , the Stats are copied to external memory (address specified by the host) The stats copied to external memory can be valid or in valid depending on the Lens movement at the time of gathering the stats . so stats are finally exported as the command is issued by the host with Valid and invalid Flag , representing the stats exported is valid or not .

There can be Four types of request configured in Fw .

1) when no request - ***AFStats_HostCmd_e_NO_REQ*** is programmed to

e_Coin_ZoneConfigCmd of FocusControl_Controls PE .

This is generally the default value for this PE . and there is no exporting of Stats is done , if this value is set . the Status for this command can be seen from the PE ***e_Coin_ZoneConfigStatus of AFStats_Status*** , the value for this PE is AFStats_StatusHostCmd_e_NO_REQ.

2) when requesting the stats ONCE - **AFStats_HostCmd_e_REQ_STATS_ONCE**

is programmed to *e_Coin_ZoneConfigCmd* of **AFStats_controls** Page . This value means that the stats are exported to external memory only once when this mode is set . the Status can be seen from the PE ***e_AFStats_StatusHostCmd_*Status of AFStats_Status ,** the value for this PE will be AFStats_StatusHostCmd_e_STATS_COPY_NOT_YET_DONE , This means the Mode is Set by the host to Once cmd mode but the Statistics are still not exported to Memory . Now once the Mode is set , host need to toggle the coin (coin mechanism in place in once mode only , toggling the coin from Heads to Tails and vice versa in this mode will always export the current statistics to external memory .) The Toggling of coin is done by the *AFStats_control PE e_Coin_AFStatsExportCmd* , toggling this variable to Heads and Tails and vice versa will export the stats to memory , once the stats are exported the notification for stats exporting completed is send to host , also the PE *e_AFStats_StatusHostCmd_Status* of *AFStats_Status* will now show the value AFStats_StatusHostCmd_e_STATS_COPY_DONE and the command coin status *in AFStats_Status* PE 's *e_Coin_AFStatsExportStatus* will be shown .

15.2.4.2.2.1 Steps to Export the Statistics to External Memory

Set the command to once mode[**e_AFStats_HostCmd_Ctrl of AFStats_Controls] :** AFStats_HostCmd_e_REQ_STATS_ONCE. Check for the command (e_Coin_AFStatsExportCmd) and status (e_Coin_AFStatsExportStatus) coin.

if both equal then e_AFStats_StatusHostCmd_Status of **AFStats_Status** will show AFStats_StatusHostCmd_e_STATS_COPY_NOT_YET_DONE. This means the stats are not yet copied to memory .

if both the unequal then the e_AFStats_StatusHostCmd_Status will show the
Processing is going on .
Specify the Host Address for Exporting the Statistics : through "**pu32_HostAssignedAddr**".

Toggle the control coin for exporting the memory in Once mode : "**e_Coin_AFStatsExportCmd" of AFStats_Control** to Coin_e_Heads/Coin_e_Tails depending on the e_Coin_AFStatsExportStatus value of AFStats_Status.

If "pu32_HostAssignedAddr" is 0 then Error is Shown in **AFStats_Status's e_AFStats_Error_Status.**

Check for error (if any) at AFStats_Status's e_AFStats_Error_Status.

Once the copy to memory has been done : check for the "**e_Coin_AFStatsExportCmd**" becomes equal to "e_Coin_AFStatsExportStatus".
Check for the Notification .

Check for then e_AFStats_StatusHostCmd_Status of **AFStats_Status,** will now show

AFStats_StatusHostCmd_e_STATS_COPY_ DONE.

3) AFStats_HostCmd_e_REQ_STATS_CONTINUOUS_AND_VALID

This is for continuous stats exporting to External Memory by the host , but only valid stats will be exported , with valid stats flag on .

AFStats_HostCmd_e_REQ_STATS_CONTINUOUS_AND_WITHOUT_VALID_CHECK

This request export the stats continuously to external memory ,irrespective of whether the Stats are valid or not .

The Status for the Command absorption can be seen by PE

e_AFStats_HostCmdStatus_Status of **AFStats_Status** .

Memory Address Configuration by Host

The Mamory can be configured by the host by PE pu32_HostAssignedAddr

Of AFStats_Controls . by default “0” is programmed . As long as it remains 0 The error is shown as “*AFStats_Error_e_AF_ERROR_HOST_ADDRESS_NOT_SPECIFIED_FOR_STATS_EXPORTING*” in e_AFStats_Error_Status of AFStats_Status . Host should specify some valid address here .

15.2.4.3 AF – to -- Host communication

Two type of interaction is performed by the by the AF fw with the Host . these are basically notification events to Host .

15.2.4.3.1 AF stats ready Notification .

This notification Event is done when the stats are exposed to external memory address .this event is triggered by the fw only when there is a request for stats from host side .

15.2.4.3.2 FLADriver Lens Stop Notification.

Lens Stop notification is sent to host as soon as the Lens stop moving after processing a lens movement cmd , in order to inform the host that the AF fw is ready to accept the command **.** If the Lens is at the same position and Host issue the same Manual Focus Command through PE *e_FocusControl_LensCommand_Control* of *Focus_Controls* .Then again a notification is send to the host .

Example :

e_FocusControl_LensCommand_Control ---------- *FocusControl_LensCommand_e_LA_CMD_GOTO_MACRO*. (Macro command issued from the host .)- As soon as the Lens arrive at Macro position , or the Lens Stop Moving the Note : NOTIFICATION for Lens Stop is send to Host .

e_FocusControl_LensCommand_Control ---------- *FocusControl_LensCommand_***e_LA_CMD_GOTO_MACRO**.(Again a Macro Command is issue by the Host Without issuing any other command before that), the FW will perform all the Action as performed in the first case , though it may not appear moving but Notification is send to host .

Note : NOTIFICATION is always send to Host whenever a command is processed with the Toggling of coin .

NOTE : So this must be ensured that , for each valid command issued from Host side to ISP FocusControl Fw there must be Valid Notification s given from the Host side .

15.2.5 Exporting AFStats (To memory) and Lens command Movement at the same Time.

Following are the steps to set the Lens Command and the AF Statistics Export at the same time .

Set the e_FocusControl_LensCommand_Control of **FocusControl_Controls** to Command – FocusControl_LensCommand_e_LA_CMD_XXX.

e_Flag_StatsWithLensMove_Control of **FocusControl_Controls** to **TRUE** , This variable is to inform the host about the stats gathering and Lens movement control at the same time .

Set the command to once mode[*e_AFStats_HostCmd_Ctrl* **of AFStats_Controls] :** AFStats_HostCmd_e_REQ_STATS_ONCE. Check for the command (e_Coin_AFStatsExportCmd) and status (e_Coin_AFStatsExportStatus) coin.

if both equal then *e_AFStats_StatusHostCmd_Status* of **AFStats_Status** will show AFStats_StatusHostCmd_e_STATS_COPY_NOT_YET_DONE. This means the stats are not yet copied to memory .

if both the unequal then the *e_AFStats_StatusHostCmd_Status* will show the Processing is going on .

Specify the Host Address for Exporting the Statistics : through "**pu32_HostAssignedAddr**". If not set then the error message is flaged from the FW side .

If "pu32_HostAssignedAddr" is 0 then Error is Shown in **AFStats_Status's** *e_AFStats_Error_Status***.** Check for the Error .

Toggle the Coin : Toggle the command coin variable "e_Coin_Control" of **FocusControl_Controls** *to Coin_e_Heads/ Coin_e_Tails* **.** The Lens Movement and the Export stats command is received by the fw at the same Time .

As soon as the Stats are Exported to memory the Notification is send to host , these Statistics must be invalid as during the stats gathering command lens is moving .

Get the value of the Valid Flag at (Host memory address+400) byte This must be INVALID (0) .

Wait for the FLADriver LENS STOP Notification .
Check for the coin status "*e_Coin_Status*" **of FocusControl_Status**. This should be equal to what has been set in *e_Coin_Controls*" ***of*** **FocusControl_Controls.**
Start a AF Statistics Export Request after the Lens STOP Notification in order to check the statistics exported is valid or not .
Get the value of the Valid Flag at (*Host memory address+400*) byte This must be VALID (1) now .

15.3 AFStats Cancellation/Handling during Firmware STOP

Whenever there is a need to Abort/Stop the sensor and/or Rx during streaming, and if there is a pending Statistics request, following are the ways in which the request is handled:

15.3.1 In case of Rx abort

[g_SystemSetup.e_Flag_abortRx_OnStop = TRUE] Pending Statistics request would be cancelled.

15.3.2 In case of error

[g_ErrorHandler.e_ErrorStatus != ErrorStatus_e_NONE] Pending Statistics request would be cancelled.

15.3.3 In case Host requests cancellation

[g_AFStats_Controls.e_Flag_AFStatsCancel = TRUE] Pending Statistics request would be cancelled.

In all of the above three cases, the firmware will send a (dummy) AFStats Ready event, and a dummy DMA grab notification to the Host. The firmware, thus, does not wait for the requested synchronization, and immediately sends the notification.

Further, a new status is associated to the Statistics in the FrameParamStatus_Af. u32_AfStatsValid element, telling the host the Statistics are INVALID. The firmware ensures that there is always one and only one event associated to a single request. The HOST shall ensure it does not send a new request before having received the event associated to the previous one.

15.3.4 In case Host does not request cancellation

[g_AFStats_Controls.e_Flag_AFStatsCancel = FALSE] This use case will be checked if all of the above three cases are not true. Hence, this use case will never be hit in case of Rx Abort or in Error case.

In this case, the firmware will not move to STOPPED state until it completes the pending Statistics request. Once it completes the request (without notifying the Host), it will move to STOPPED state. Following this, it notifies the Host of the request completion.

This model ensures that the Host won't be able to issue another Stats request after it gives STOP command, if there is an already pending request. After STOP, when the Host is notified, the Host may give the next command, which will, of course, be processed once the Host issues RUN command.

15.4 Pages Exposed

15.4.1 AFStats Module:

15.4.1.1 AFStats_Controls

15.4.1.2 AFStats_Controls	AFStats control information .	
Page Elements	Description	Value/Range (comments)
pu32_HostAssignedAddr	Pointer to host specified address , a 32 bit address can be assigned by the host , in order to dump the AFStats statistics . Address will contain the 408 bytes data . which can be accessed by host once the notification for the AF statistics is given by the fw .	
u8_CoringValue	Coring value. If coring is enabled, focus measure will be clipped with this value.	
u8_HRatioNum	Numerator value of the horizontal ratio of the WOI H size for each zone , if fixed zone system is used (generally if 7 or 9 zone system is used .)	
u8_HRatioDen	Denominator value of the horizontal ratio of the WOI Hsize for each zone , if fixed zone system is used (generally if 7 or 9 zone system is used .)	
u8_VRatioNum	Numerator value of the vertical ratio of the WOI V size for each zone , if fixed zone system is used (generally if 7 or 9 zone system is used .)	
u8_VRatioDen	Numerator value of the vertical ratio of the WOI V size for each zone , if fixed zone system is used (generally if 7 or 9 zone system is used .)	
u8_HostActiveZonesCounter	This counter contains the value for the no of zones enabled by the host .if all the zones enabled the count is 10 . if no zone is enabled the counter shows 0.	
e_AFStats_WindowsSystem_Control	There are 3 zone system supported , 7 , 9 & Host zone system	
e_Coin_ZoneConfigCmd	command coin to inform the host zone setup has been done.	
e_AFStats_HostCmd_Ctrl	Command issued by the host for the AF stats request .	
e_Flag_AutoRefresh	Control the automatic update of AF stats settings at the beginning of each frame. During zoom operation the FOV changes and there for the	

	zone ‘s reconstruction is required .	
e_Flag_AbsSquareEnabled	0 - Square value accumulated for autofocus stats. 1 - Absolute value accumulated for autofocus stats.	
e_Flag_ReducedZoneSetup	This variable is not currently used for the host zone system , but used for the 7 or 9 zone system , it reduces the zone size from the normal size .	
e_Flag_HostZoneSetupInPercentage	This element takes the input from the host in order to setup the zone configuration , either in absolute terms or in percentage with respect to current FOV .	
e_Coin_AFStatsExportCmd	This element is a command interface for the host to enable the export of the AF statistics into external memory . Toggling this coin flag to Heads and Tails and vice versa will enable the ISP fw to copy the Statistics into external memory whoes address is specified in pu32_HostAssignedAddr PE	

15.4.1.3 AFStats_Status

15.4.1.4 AFStats_Status	AFStats Status information .	
Page Elements	Description	Value/Range (comments)
u32_MaxFocusMeasurePerPixel	Theoretical max focus measure per pixel accumulated by the FSWM stats	Default is 0.
U16_StartingAFZoneLine	It contains the line from where the accumulation os afstats begins .	Default is 0.

U16_WOIWidth	WOI width available for AF stats zone system	Default 0 .
u16_WOIHeight	WOI Height available for Afstats zone system .	Default:0 .
u16_AFZonesWidth	This is used for the Fixed width zone system , like 7,9 zone system .contains the width of each zone .not used in case of host window system .	Default: 0
u16_AFZonesHeight	Contains the height of each zone , if 7 or 9 zone system is used .	Default :0
u8_CoringValue	Shows the status of the coring set in control .	By default coring is disabled .
u8_ActiveZonesCounter	Status of the no of zones active . for host zone system this is equivalent to u8_HostActiveZonesCounter. For 7 and 9 zone system it should be 7or 9 zones respectively .	By default for host system it should be 10.
U8_HratioNum	Shows status of the Numerator of H ratio set in the control field .	1
u8_HratioDen	Shows status of the denominator H ratio set in the control field	6
u8_VratioNum	Shows vertical ratio Numerator. .	1
u8_VratioDen	Shows vertical ratio Denomirator.	9
s8_ZoneIntCycles	This shows the incrementing interrupt cycle for the Frame , when all the interrupt of AF stats has come	Defsult is 0 , incremented to 1 each time when last interrupt of the zone has come .
u8_IndexMax	host configured zone no , for which the interrupt in a frame comes last .	Default is 9 .
u8_IndexMin	host configured zone no , for which the interrupt in a frame comes first .	Default 0 .
e_AFStats_WindowsSystem_Control	The status showing what window system has been used , 7 , 9 or host zone system .	The default status shows ,Host window system .
e_AFStats_Error_Status	Error shown , if the zone not configured correctly ,	Default 0 , showing no error

	overlapped zones etc .	(AFStats_Error_e_AF_ERROR_OK)
e_Coin_ZoneConfigStatus	Coin mechanism ,see the status of the coin when host flip the coin .the default value for this should be different in order to configure the zone shape and size .	Default : Coin_e_Heads
e_AFStats_HostCmd_Status	The status of the host cmd for requesting /not requesting can be shown by this element	The default value in this case is 0 . Showing No request from host .
e_Flag_ForcedAFStatsIrq	Flag indicating that the current int has been forced externally, and not by any af stats	Default is 0 .
e_Flag_AbsSquareEnabled	The status for the absSquareEnable/disable is shown here .	Default is 0.
e_Coin_AFStatsExportStatus	This variable tells the status of the cmd issued by the host for exporting the AF statistics into external memory . once the statistics is exported, the coin is toggled by the ISP fw to same as set by the host in e_Coin_AFStatsExportCmd	Default value : Coin_e_Heads

15.4.1.5 AFStats_AFZoneInterrupt

15.4.1.6 AFStats_AFZoneInterrupt **15.4.1.7**	AFStats Individual zone interrupt information.	
Page Elements	Description	Value/Range (comments)
u16_INT00_AUTOFOCUS	Total no of Interrupt count for the zone 0 .the count is incremented as soon as the interrupt for zone 0 comes .	Default : 0
u16_INT01_AUTOFOCUS	Total no of Interrupt count for the zone 1 .	Default : 0
u16_INT02_AUTOFOCUS	Total no of Interrupt count for the zone 2 .	Default : 0
u16_INT03_AUTOFOCUS	Total no of Interrupt count for the zone 3 .	Default : 0
u16_INT04_AUTOFOCUS	Total no of Interrupt count for the zone 4 .	Default : 0
u16_INT05_AUTOFOCUS	Total no of Interrupt count for the zone 5 .	Default : 0
u16_INT06_AUTOFOCUS	Total no of Interrupt count for the zone 6 .	Default : 0
u16_INT07_AUTOFOCUS	Total no of Interrupt count for the zone 7 .	Default : 0
u16_INT08_AUTOFOCUS	Total no of Interrupt count for the zone 8 .	Default : 0
u16_INT09_AUTOFOCUS	Total no of Interrupt count for the zone 9 .	Default : 0

15.4.1.8 AFStats_HostZoneConfig

15.4.1.9 AFStats_HostZoneConfig	AFStats Host zone configuration info in absolute values.	
Page Elements	Description	Value/Range (comments)
u16_HostAFZoneStartX	The x start of the zone can be configured by the host , this start is relative to the current fov .the host has to program this startX w.r.t current FOV programmed .the actual start will be programmed wrt to cropped WOI in the ISP , that would be currentFOV x + offset programmed by the host (u16_HostAFZoneStartX).	Default: xstart here is same as the 7 zone system has for each zone . Range : 0 to (current WOI start y + WOI H size -1).
U16_HostAFZoneStartY	The y start , should be programmed by the host using current Fov. The actual ystart for this would be w.r.t cropped woi Ystart.	Default : the default is same as for the 7 zone eye shaped system Range : 0 to (current WOI start y + WOI V size -1).
U16_HostAFZoneWidth	The width of each zone can be programmed by the host .this should be within current WOI H size .	Default : the Default value for this is current WOI H size *1/6(u8_HratioNum/ u8_HratioDen). Range : < current WOI H size . For better performance width programmed > 12 .
u16_HostAFZoneHeight	The height of each zone can be programmed by the host with in current WOI V size , if programmed incorrectly the error will be shown .	Default : the default value for this is current WOI V size * 1/9(u8_VratioNum/ u8_VratioDen) Range : < current WOI V size . For better performance height Programmed > 6
e_Flag_Enabled	Enable / disable the Zone . 0 : disable 1 :Enable	Default : using 7 zone eye shaped setup as default , oly zone 0-6 are enabled , and from 7-9 Disabled .
u16_HostAFZoneStartX	The x start of the zone can be configured by the host , this start is relative to the current fov .the host has to program this startX w.r.t current FOV programmed	Default: xstart here is same as the 7 zone system has for each zone .

	.the actual start will be programmed wrt to cropped WOI in the ISP , that would be currentFOV x + offset programmed by the host (u16_HostAFZoneStartX).	Range : 0 to (current WOI start y + WOI H size -1).
U16_HostAFZoneStartY	The y start , should be programmed by the host using current Fov. The actual ystart for this would be w.r.t cropped woi Ystart.	Default : the default is same as for the 7 zone eye shaped system Range : 0 to (current WOI start y + WOI V size -1).
U16_HostAFZoneWidth	The width of each zone can be programmed by the host .this should be within current WOI H size .	Default : the Default value for this is current WOI H size *1/6(u8_HratioNum/ u8_HratioDen). Range : < current WOI H size . For better performance width programmed > 12 .
u16_HostAFZoneHeight	The height of each zone can be programmed by the host with in current WOI V size , if programmed incorrectly the error will be shown .	Default : the default value for this is current WOI V size * 1/9(u8_VratioNum/ u8_VratioDen) Range : < current WOI V size . For better performance height Programmed > 6
e_Flag_Enabled	Enable / disable the Zone . 0 : disable 1 :Enable	Default : using 7 zone eye shaped setup as default , oly zone 0-6 are enabled , and from 7-9 Disabled .

15.4.1.10 AFStats_HostZoneStatus

AFStats_HostZoneStatus	AFStats Host Zone Status info.	
Page Elements	Description	Value/Range (comments)
U32_AFZoneStartX	The x start for each zone programmed by the host is shown here .if not programmed correctly then the error will be raised and the previous x start programmed will be	Default : 0 If HostAFZoneStartX programmed correctly (in case of no error) , this will show

	shown .	HostAFZoneStartX as a default value , other wise 0.
U32_Focus	Stats value	
U32_AFZoneStartY	The status of the host configured y start for each zone is shown here .	Default : 0 If HostAFZoneStartX programmed correctly (in case of no error) , this will show HostAFZoneStartX as a default value , other wise 0.
U32_AFZoneEndX	The End x for each zone is auto matically calculated by the fw based on host programmed start x + host programmed zone width .	Default : default for this is the same as the 7 zone eye shaped setup has , if programmed correctly by the host , otherwise 0 .
U32_AFZoneEndY	End y is calculated as the start y programmed by the host for the zone + height of the zone programmed .	Default: same as 7 zone eye shaped system has , otherwise will be 0 .
U32_AFZonesWidth	If programmed correctly by the host , the same width is shown here as programmed by the host for this zone .	Default : width of 7 zone system . shown by u16_AFZonesWidth. Otherwise 0 (if error).
U32_AFZonesHeight	If programmed correctly by the host , the same height is shown here as programmed by the host for this zone .	Default : height of 7 zone system . shown by u16_ u16_AFZonesHeight. Otherwise 0 (if error).
U32_light	Light stats	0
u32_WeightAssigned	This element sets the weight for each zone . the weight actually priorities the zone among the Enabled zones for having host Intreset of area . the weight can be given <=32 for each zone . For same priority the weight for each zone is fixed to 1 .	Default : 1 Range : <=32
U32_Enabled	Status of the current zone is shown by this element . 0 : disable , 1; enable .	Default : only zone 0-6 will show enable , 7-9 Disable . keeping in view the Default setup like 7 zone setup .

15.4.1.11 AFStats_ZoneHWStatus

15.4.1.12 AFStats_ZoneHWStatus	The Zone HW Dimension status.	
Page Elements	Description	Range
u16_AFStartX	The Actual start x programmed into the HW register is shown by this variable . this contain WOI start offset X + the offset of the zone start x .	Default : 0
u16_AFStartY	The actual start y programmed in the hw register , it has value (WOI start offset Y + the offset for the zone start_y)	Default : 0
u16_AFEndX	The Actual End X programmed for the zone contains (WOI start offset X + the offset of the zone start x programmed by the host +zone width)	Default : 0
u16_AFEndY	The Actual End Y programmed for the zone contains (WOI start offset Y + the offset of the zone start y programmed by the host +zone Height)	Default : 0

15.4.1.13 AFStats_FocusStats

15.4.1.14 AFStats_FocusStats	Focus stats of Each Zone info	
Page Elements	Description	Range
u32_StatsValue_0	Accumulated stats value for the zone 0 , of current frame .the value is in multiplicative factor of the weight assigned to zone .	Default : 0 Range : >= 0 Value : Zone 0 statsValue * Zone0 weight .
u32_StatsValue_1	Accumulated stats value for the zone 1	Default : 0 Range : >= 0 Value : Zone 1 statsValue * Zone1 weight .

u32_StatsValue_2	Accumulated stats value for the zone 2	Default : 0 Range : >= 0 Value : Zone 2 statsValue * Zone2 weight .
u32_StatsValue_3	Accumulated stats value for the zone 3	Default : 0 Range : >= 0 Value : Zone 3 statsValue * Zone3 weight .
u32_StatsValue_4	Accumulated stats value for the zone 4	Default : 0 Range : >= 0 Value : Zone 4 statsValue * Zone4 weight .
u32_StatsValue_5	Accumulated stats value for the zone 5	Default : 0 Range : >= 0 Value : Zone 5 statsValue * Zone5 weight .
u32_StatsValue_6	Accumulated stats value for the zone 6	Default : 0 Range : >= 0 Value : Zone 6 statsValue * Zone6 weight .
u32_StatsValue_7	Accumulated stats value for the zone 7	Default : 0 , initially the zone is disabled in view of 7 zone system , so value for this zone remains 0 , until the host set the dimension for this zone and enable it .
u32_StatsValue_8	Accumulated stats value for the zone 8	Same as above .
u32_StatsValue_9	Accumulated stats value for the zone 9.	Same as above .

15.4.1.15 AFStats_LightStats

15.4.1.16 AFStats_LightStats	Light Stats Structure info .	
Page Elements	Description	Range
u8_StatsValue_0	Accumulated Light stats(Brightness values) value read from hw register for zone 0 of current frame .	Default : 0 Range : 0<=light stats <=255
u8_StatsValue_1	Light stats value for zone 1.	Same as above .
u8_StatsValue_2	Light stats value for zone 2.	Same as above .
u8_StatsValue_3	Light stats value for zone 3.	Same as above .
u8_StatsValue_4	Light stats value for zone 4.	Same as above .
u8_StatsValue_5	Light stats value for zone 5.	Same as above .
u8_StatsValue_6	Light stats value for zone 6.	Same as above .
u8_StatsValue_7	Light stats value for zone 7.	Default : initially only 7 zones 0 to 6 are enabled anf zone 7 to 9 are diabled for 7 zone eye shaped setup . the value for this zone remains 0 untill host enable the zone with correct dimensions .
u8_StatsValue_8	Light stats value for zone 8.	Same as above
u8_StatsValue_9	Light stats value for zone 9.	Same as above .

15.4.1.17 AFStats_ZoneVectorBase

15.4.1.18 AFStats_ZoneVectorBase	Zone vector information .	
Page Elements	Description	Range
u8_Weight_0	Weight assigned by the host to zone 0 , this is a multiplication factor in the afstats calculated for zone 0 , if weight assigned is 0 the stats for this zone becomes 0 . This can also be treated to enable disable the stats for zone 0 .but the interrupt for the zone 0 will come as usual .	Default : 1 Range : <=32. to ensure overflow free calculations
u8_Weight_1	Weight for zone 1.	Default : 1 Same as above
u8_Weight_2	Weight for zone 2.	Default : 1 Same as above
u8_Weight_3	Weight for zone 3.	Default : 1 Same as above
u8_Weight_4	Weight for zone 4.	Default : 1 Same as above
u8_Weight_5	Weight for zone 5.	Default : 1 Same as above
u8_Weight_6	Weight for zone 6.	Default : 1 Same as above
u8_Weight_7	Weight for zone 7.	Default : 1 Same as above
u8_Weight_8	Weight for zone 8.	Default : 1 Same as above
u8_Weight_9	Weight for zone 9.	Default : 1 Same as above

15.4.2 FLADriver Module

15.4.2.1 FLADriver_LLLCtrlStatusParam Page Element

15.4.2.2 FLADriver_LLLCtrlStatusParam	Control and status page element for the actuator lens properties and its movements.	
Page Elements	Description	Range
u16_MinPos	Minimum position that can be achieved by the lens . Start position - initial start position at low level This value is recalculated at the time of initialization based on the actuator min value (usually the NVM value , otherwise the host set value), the smaller end point value.	Default : The infinity (worst case) is initialize to it . Infinity Looking Down at 1 meter distance .
u16_MaxPos	Maximum position that can be achieved by the lens .or End position . This value is recalculated at the time of initialization based on the actuator Max value (usually the NVM value , otherwise the host set value), the greater end point value.	Default: The Macro (worst case) is initialize to it. Macro looking up at 10 cm distance .
u16_RestPos	Rest position usually is that Lens position at which the power consumption is lowest , as it draw lowest current when placed at this value . Lens is forced to move at this position when no activity is performed .The ISP fw takes the REST position from the NVM .	Default : Read from NVM.
u16_InfinityFarEndPos	Low Level far End point position(In case of Piezo its on lower side of the range) . this value , generally measured during the characterization of the Focus Lens Module , when the camera is held looking up and focus at greater then 1m distance .	Default: Read From NVM.
u16_InfinityHorPos	Low level Horizontal infinity position. The Position when the Lens module is horizontal to the surface and Focus the object at infinity distance (greater then 1,1.5 m Distance)	Default: Read From NVM.
u16_HyperfocalPos	Focus Position , which is just before the infinity (not at infinity). The Position of the Lens when the Depth of Focus starts for the object at infinity (in other words ,from this position till infinity all the objects are at same focus) .Usually this is calculated as a fine step just before the infinity Position (Theoretically , where fine step is total range divided by 30).	Default: Read From NVM.
u16_MacroHorPos	low level Macro horizontal position . This Position is that Position of the Lens when the object at distance of 10 cm is in perfect focus . Keeping module horizontal to surface .	Default: Read From NVM.
u16_MacroNearEndPos	Low Level End point position of Lens when the camera is held upside and focus at 10cm distance .In case of Piezo Actuator its on higher side of the range .	Default: Read From NVM.
s16_ToleranceSize	Size in low level units(HW unit) ,by which the Lens movement +/- of this value around a perfect Focus Position will not make the object out of focus .Say , For a command movement to a Target	Default: 0 , No NVM value .

15.4.2.3 FLADriver_Controls

15.4.2.4 FLADriver_Controls	General Control structure for the FLADriver .	
Page Elements	Description	Range
u16_Ctrl_TimeLimit_ms	specifies the time factor (in ms), multiplied to the time read from the LLA . this can be configured by the host .	Default : 10
e_FLADriver_RangeDef_CtrlRange	This variable specify that the Low level parameters should be initialized with the NVM values or HOST specified values .	Default: FLADriver_RangeDef_e_NVM_LEVEL_RANGE

15.4.2.5

15.4.2.6 FLADriver_Status

15.4.2.7 FLADriver_Status	Status of the FLADriver PE.	
Page Elements	Description	Range
u8_Cycles	Number of times the Move to function is called or can say Number of I2C Grabs.	Default : 0
e_Flag_LensIsMoving	This shows lens movement . if 1 : lens moving 0 : lens not moving .	Default : 0
e_Flag_LimitsExceeded	specifies if the range limits have been reached, this variable is set by the fw if during the command operation Lens reached either boundary(at Macro or infinity)if 1: limit reached . 0: inside limit boundary.	Default : 0
e_Flag_LowLevelDriverInitialized	Flag to indicate if the Low Level initialization for the FLADriver has been done or not , if TRUE then the initialization has been done successfully otherwise not .	Default : 0 If initialization done set to 1.
e_FLADriver_ActuatorOrientation_Type	Tells the Actuator behaviour is aligned with the High level Standard , i.e . the Macro and infinity position for the current actuator is in same direction as thhe high level has (Macro towards the near end and infinity towards the far end).	Default : FLADriver_ActuatorOrientation_e_DIRECTION_REVERSED_WRT_HIGH_LEVEL
e_FLADriver_RangeDef_Status Range	This variable specify that status of the Low level parameters settings happens based on the NVM values or HOST specified values .	Default: FLADriver_RangeDef_e_NVM_LEVEL_RANGE

15.4.2.8 FLADriver_NvmStoredData

15.4.2.9 FLADriver_NvmStoredData	Data from the NVM and related info .	
Page Elements	Description	Default/Range
S32__NVMMacroPos	Position Read during the initialization from the camera driver Low level API .this position represent the near end read from the camera details about the lens position . near_end position from g_camera_details.p_lens_details->positions	Default : -1 Range : can be between low level min and max.
S32__NVMInfinityPos	Position read from the camera driver API and represent the far end point . Read from far_end position from g_camera_details.p_lens_details->positions	Default : -1 Range : can be between low level min and max.
S32_NVMInfinityHorPos	Read from the LLA . and represent the horizontal position of focus to infinity . it means when lens held horizontal and focus to infinity the position of the lens is given by this element. Infinity Position from g_camera_details.p_lens_details->positions	Default : -1 Range : can be between low level min and max. but should be greater then s16_NVMInfinityPos and Less then s16_NVMMacroPos
S32_NVMMacroHorPos	Read from the LLA . the position of the lens when the object is at 10 cm and camera is held horizontal . Macro Position from g_camera_details.p_lens_details->positions	Default : -1 Range : can be between low level min and max. but should be greater then s16_NVMInfinityHorPos and Less then s16_NVMMacroPos
S32_NVMHyperfocalPos	Hyperfocal position is the position at minimum infinity distance from where the object placed after that position till infinity remains in focus .	Default : -1 Range : can be between low level min and max. but should be

	Read from LLA during initialization , hyperfocal Position from g_camera_details.p_lens_details->positions	Less then or equal s16_NVMInfinityHorPos and greater then s16_NVMInfinityPos
S32_NVMRestPos	Position where the Lens can be placed when no work is performed . Read from the LLA . Rest Position from g_camera_details.p_lens_details->positions	Default : -1 Range : can be between low level min and max. Generally between 0 and infinity position .
e_Flag_NVMDataPresent	Tells if the nvm data is present or not .	Default : TRUE
e_Flag_NVMActuatorLensPresent	Tells if the actuator lens is present or not .	Default : TRUE
e_Flag _NVMPositionSensorPresent	Tells if the actuator lens is present then the position sensor capability is present or not .	Default : TRUE

15.4.2.10

15.4.2.11

15.4.2.12 FLADriver_LensLLDParam

15.4.2.13 FLADriver_LensLLDParam	Structure contains LLA and Lens information during process.	
Page Elements	Description	Range
u32_NVMLensUnitMovementTime_us	Total time in micro seconds consumed by the Lens during 1 unit movement of lens .	Default : 0
u32_TimeTakenByLensAPIs_us	Total time in micro seconds consumed by the Lens during a particular command movemnt set by the host .	Default : 0
u32_FLADIntTimer2Count	Count for Timer interrupt call for the Timer2 .	Default : 0
u32_FLADTimer2CallCount	The Count for the no of times the Timer2 is set for the Lens movement , total no of count for starting the lens Movement timer2.	Default : 0
u16_DiffFromTarget	the Actual difference between the target and the Current , after	Default : 0

	the movement command has been issued by the host . the difference is at the low level .	
e_FLADriver_APIError_Type	This shows the Error for the Timer APIs and Lens Movement at various Level .	Default : 0
e_FLADriver_Timer2Id_Type	This variable shows the State of the Timer2 at different level .shows information about Timer2 has NOT_STARTED, EXPIRED , WAITING FOR STOP , or STOPPED.	Default : 0

15.4.2.14

15.4.3 FOCUS CONTROL

15.4.3.1 FocusControl_Controls

15.4.3.2 FocusControl_Controls	Focus Control command related info .	
Page Elements	Description	Default/Range
e_FocusControl_LensCommand_Control	The various lens command discussed above in this document can be set by the host .	Default : FocusControl_LensCommand_e_LA_CMD_GOTO_REST Range : from 0 to 8.
e_Coin_Control	Focus control coin mechanism for the command issued by the host .	Default : Coin_e_Heads(0) Range : Coin_e_Heads(0) or Coin_e_Tails(1)
e_Flag_StatsWithLensMove_Control	this variable is set to TRUE if the host	Default : Flag_e_FALSE(0) Range : Flag_e_FALSE(0) or Flag_e_TRUE (1)

	want to export stats as well as the Lens Movement at the same time. A feature if True, says Move the Lens and export the AFStats .	

15.4.3.3 FocusControl_Status

15.4.3.4 FocusControl_Status **15.4.3.5**	Focus Status related info .	
Page Elements	Description	Default/Range
u16_Cycles	Focus Control Cycles, take the count of the AFStats Ready ISR execution.	Default : 0
e_FocusControl_LensCommand_Status	Lens command issued by the host is absorbed properly or not can be seen by this variable .	Default : FocusControl_LensCommand_e_LA_CMD_GOTO_REST Range : from 0 to 8.
e_FocusControl_FocusMsg_Status	This element shows the details about any error detected by the Focus Control module.	Default : AF_NO_ERROR(0) Range : 0 to 4

e_Coin_Status	Flag to behave as a Status Side of Coin .	Default : Coin_e_Heads(0) Range : : Coin_e_Heads(0) Or : Coin_e_Tails(1)
e_Flag_LensIsMovingAtTheSOF	This element tell to the Focus control that at the SOF the lens is still moving or not , so that the current frame probably isn't very good for Stats Exporting.	Default : Flag_e_FALSE (0) Range : Flag_e_FALSE or Flag_e_TRUE
e_Flag_IsStable	Signaling the the Focus System is correctly stable, and the command For the New Manual Focus can be anticipated.	Default : Flag_e_FALSE Range : Flag_e_FALSE or Flag_e_TRUE
e_Flag_Error	Signaling a serious error detected by the Focus Control module. The Focus Control module will halt until the host clears this flag.	Default : Flag_e_FALSE(0) Range : Flag_e_FALSE(0) or Flag_e_TRUE(1)
e_Flag_StatsWithLensMove_Status	status of the variable set for the AFstats Exporting as well as	Default : Flag_e_FALSE Rabge : Flag_e_FALSE or Flag_e_TRUE

	for the Command Mode at the same time.	

15.4.3.6 Focus parameters status

15.4.3.7 g_FrameParamStatus_Af	g_FrameParamStatus_Af	
Page Elements	Description	Default/Range
pu32_HostAssignedFrameStatusAddr	host specified address for FrameParamStatus_Af exporting to external memory .	0
u32_AfStatsValid	AF stats are valid/invalid Value = 1 indicates stats are valid Value = 0 indicates stats are invalid [DEFAULT]: 0 [INVALID]	
u32_AfStatsLensPos	Current Lens Position	
u32_AfStatsFrameId	Frame Id in which AF has exported valid statistics	
u32_SizeOfFrameParamStatus	sizeof(FrameParamStatus_Af_ts) Host should read this PE before allocating memory for the export	

Manual Focus Tests

These test are for Manual Focus on V1 Platform .

Setting the V1 Environment for Manual Focus Test

1) Select the Menu: **NMF test scripts Tosh LLD -----|**
|--debug--|
|--startenvPrimary
2) The environment will be configured for the Basic Focus Control test and a command prompt **MMTE>**
Will appear on the console .

3) Set command **MMTE> LR prepare .**
4) Set command **MMTE> LR start .**
5) Set command **MMTE> focushelp**
The test Environment will appear with the Test Name on Manual focus Tests on the environment console .
The Test are Describe below along with the command to run the test (Name in short for easy typing).

TEST NAME	COMMAND	DESCRIPTION
focusmanualtest	fmantest	Manual Focus Complete test , this test includes all the possible commands for the Manual focus .the following command are processed in a sequence given below . • MOVE TO MACRO • MOVE TO INFINITY • MOVE TO TARGET POSITION • MOVE TO REST POSITION • MOVE TO HOR INFINITY POSITION • MOVE SINGLE STEP TO MACRO • MOVE SINGLE STEP TO INFINITY • MOVE TO HOR MACRO POSITION • MOVE TO HYPERFOCAL POSITION Every Command has the detailed Log description for the following . ▪ Lens Position before the command given . ▪ Manual focus Command given . ▪ Parameter setting (if any) along with values . ▪ Error control , (if any) error occurs during the command execution . ▪ Command absorption . ▪ Coin set and absorb. ▪ Wait for notification status . ▪ Notification received . ▪ Target Lens Position (position to which the Lens should reach after the execution of command) ▪ Lens Position Actually Reached . ▪ Diff from the Target Position (difference of actual and target). ▪ Test Result as Fail and Pass . ▪ Dump of the Statistics (only in case of AFStats).
focusstatreadytest	fstattest <File Name>	This Test Basically Export the AF Statistics , when host issue the command/coin Toggle , the Statistics for all the 10 zones along with VALID and INVALID status and Lens Position is Copied from the FW structure to the Memory Address specified by the host .the Host receive the Notification as the Statistics Copy to Memory has been done . Please refer to Document for more Details .

TEST NAME	COMMAND	DESCRIPTION
focusmanualtest	fmantest	The meaning of command **fmantest** : ***focus Manual Tests .*** Manual Focus Complete test , this test includes all the possible commands for the Manual focus .the following command are processed in a sequence given below . • MOVE TO MACRO • MOVE TO INFINITY • MOVE TO TARGET POSITION • MOVE TO REST POSITION • MOVE TO HOR INFINITY POSITION • MOVE SINGLE STEP TO MACRO • MOVE SINGLE STEP TO INFINITY • MOVE TO HOR MACRO POSITION • MOVE TO HYPERFOCAL POSITION Every Command has the detailed Log description for the following . ▪ Lens Position before the command given . ▪ Manual focus Command given . ▪ Parameter setting (if any) along with values . ▪ Error control , (if any) error occurs during the command execution . ▪ Command absorption . ▪ Coin set and absorb. ▪ Wait for notification status . ▪ Notification received . ▪ Target Lens Position (position to which the Lens should reach after the execution of command) ▪ Lens Position Actually Reached . ▪ Diff from the Target Position (difference of actual and target). ▪ Test Result as Fail and Pass . ▪ Dump of the Statistics (only in case of AFStats).
focusstatreadytest	fstattest <File Name>	The meaning of command **fstattest** : ***focus statistics exporting Test .*** This Test Basically Export the AF Statistics , when host issue the command/coin Toggle , the Statistics for all the 10 zones along with VALID and INVALID status and Lens Position is Copied from the FW structure to the Memory Address specified by the host .the Host receive the Notification as the Statistics Copy to Memory has been done . Please refer to Document for more Details . The test Make sure the stats are same which are copied to memory and the dump of the stats can be taken by specifying the file name along with the command as given on left in command column .
focuszonesetupinper	fzsetper	The meaning of command **fzsetper** : ***focus Zone Set in Percentage*** This test Configure the Zone Dimension within the Current WOI . The Zone dimensions are set in terms of % of WOI Width and Height . Say For a Zone the start X and Start Y position are respectively set in Percentage(%) of Current WOI Width and Height . Similarly for the Endx and End y Are also in % of current WOI width and Height respectively. The Test Calculate the Percentage Randomly for each Zone , just to make sure the Dimension can be at any Place Within the WOI . these Dimensions are further converted to absolute values and FW verify that the dimension Programmed (As calculated Randomly) are correct and if

16. Exposure, White balance and Frame rate manual mode

16.1 Exposure, white balance and frame rate

16.1.1 Overview

The section target manual control for exposure, white balance and frame rate. The section is useful when auto exposure and auto white balance are running on ARM. ISP FW client is responsible for programming all the values in manual mode.

16.1.2 Description

All three values exposure, white balance and frame rate will be absorbed in one frame. After absorption, FW will raise event.

16.1.3 Usage

16.1.3.1 Pre-Requisites

None

16.1.3.2 Pre Boot Parameters

None

16.1.3.3 Pre Run Parameters

None

16.1.3.4 Live Parameters

All the page elements are live parameters.

16.1.3.5 Modes of Operation

Only manual mode is supported

16.1.3.6 Exposure & White Balance Link

Exposure and frame rate are related to sensor. White balance values are applied only when requested exposure is absorbed in the sensor

16.1.3.7 ND Filter Support

A positive value of the u32_NDFilter_Transparency_x_100 in FrameParamStatus Page indicates that ND Filter is supported (otherwise it is 0). Its value indicates the level of transparency while applying the filter. For example, a value of 1250 means a transparency of 12.5%.

To set (or reset) the filter usage, user has to set (or reset) the e_Flag_NDFilter element (in Exposure_DriverControls Page), and then toggle e_Coin_Ctrl of the SystemSetup Page. Once the command is absorbed, the firmware sets the u32_Flag_NDFilter element of the FrameParamStatus Page equal to the user flag.

16.1.3.8 Readout Time and Exposure Quantization

The Exposure Quantization Step and Active Pixel data readout time are exposed through the f_ExposureQuantizationStep_us and f_ActiveData_ReadoutTime_us elements in FrameParamStatus Page respectively.

16.1.3.9 Pages Exposed

The pages exposed by the module can be grouped in 3 categories.

1.) Control & Status Pages for exposure:

This group includes

a.) `Exposure_CompilerStatus`

b.) `Exposure_ParametersApplied`

c.) `Exposure_DriverControls`

d.) `Exposure_ErrorStatus`

Status Page 2 for Exposure	***Exposure_CompilerStatus***	
Page Elements	**Description**	**Range**
f_AnalogGainPending	Analog Gain calculated by the compiler - to be applied to the sensor.	Between Maximum & Minimum Values.
f_DigitalGainPending	Digital Gain calculated by the compiler - to be applied to the pixel pipe - this is multiplied with each of the channel gains as computed by the white balance module	Between Maximum & Minimum Values.
f_CompiledExposureTime_us	Exposure Time as cal. by the Exposure Compiler taking the present frame rate into account.	
u32_TotalIntegrationTimePending_us	Total Current Integration Time = composite of fine integration pixels and coarse integration lines.	
u16_CoarseIntegrationPending_lines	Coarse Integration calculated by Exposure Compiler in terms of number of lines	Between Maximum & Minimum Values.
u16_FineIntegrationPending_pixels	Fine Integration calculated by Exposure Compiler in terms of number of pixels	Between Maximum & Minimum Values.
u16_AnalogGainPending_x256	Coded Analog Gain	Between Maximum & Minimum Values.

Parameters Applied Status Page for Exposure	***Exposure_ParametersApplied***	
Page Elements	**Description**	**Range**
f_DigitalGain	Digital Gain in coherence with the above three values which would be used for the digital gain calculations; to be applied on the pixel pipe when a frame comes with the below mentioned integration time and analog gain.	Between Maximum & Minimum Values.
u32_TotalIntegrationTime_us	Total Integration time in micro seconds.	
u16_CoarseIntegration_lines	Coarse Integration Lines programmed on the sensor	Between Maximum & Minimum Values
u16_FineIntegration_pixels	Fine Integration Pixels programmed on the sensor	Between Maximum &

		Minimum Values
u16_AnalogGain_x256	Analog Gain programmed on the sensor	Between Maximum & Minimum Values

Error Control Page for Exposure	***Exposure_ErrorControl***	
Page Elements	**Description**	**Range**
u8_MaximumNumberOfFrames	Number of frames for which the system should wait for the applied analog gain and integration time to appear. If they do not appear within this number of frames, an error case is generated and e_Flag_ForceInputProcUpdation is set to TRUE forcibly.	(6 or 8)

Exposure Values for the Sensor Driver	***Exposure_DriverControls***	
Page Elements	**Description**	**Range**
u32_TotalTargetExposureTime_us	Total Exposure Time to be applied on to the Sensor.	
u32_TargetExposureTime_us	Exposure Time to be applied on to the Sensor Driver.	
u16_TargetAnalogGain_x256	Analog Gain to be applied on to the Sensor Driver	
u16_Aperture	Aperture to be applied on to the Sensor.	
u8_FlashState	FlashState to be applied on to the Sensor.	
u8_DistanceFromConvergence	DistanceFromConvergence	
e_Flag_NDFilter	Flag depicting whether NDFilter should be used	
e_Flag_AECConverged	Flag for AEC Convergence	

Error Status Page for Exposure	***Exposure_ErrorStatus***	
Page Elements	**Description**	**Range**
u8_NumberOfForcedInputProcUpdates	Gives the number of times Input Proc has been forcibly updated. Integration Time and Analog Gain applied to the sensor have not appeared in a frame for a consecutive number of frames which is as specified by the Control Page.	
u8_NumberOfConsecutiveDelayedFrames	Gives the number of consecutive frames for which the analog gain and integration time have not appeared in a frame after they have been applied on the sensor. In an ideal scenario, they appear in the second frame after they are applied.	

u8_ExposureSyncErrorCount	Gives the total count of frames for which the analog gain and integration time were out of sync	
e_Flag_ForceInputProcUpdation	Flag which indicates to the SOF isr that it has to do the forced updating of Input Proc and let the exposure control start running again even though due to some error analog gain and integration time applied on the sensor have not appeared in the 'Maximum Number Of Frames' as specified by the control page	Flag_e_TRUE(1) Flag_e_FALSE(0)

2.) Control & Status Pages for manual white balance:

Control Page for WhiteBalance	***WhiteBalanceControl***	
Page Elements	**Description**	**Range**
f_RedManualGain	Red Gain	
f_GreenManualGain	Green Gain	
f_BlueManualGain	Blue Gain	

16.1.3.10 WhiteBalanceStatus

Status Page 1 for WhiteBalance	***WhiteBalanceStatus***	
Page Elements	**Description**	**Range**
f_RedGain	Current red channel gain	
f_GreenGain	Current Green channel gain	
f_BlueGain	Current Blue channel gain	

3.) Control & Status Pages for frame rate:

Control Structure for Frame rate	**FrameRateControl**	
Page Elements	**Description**	**Range**
f_UserMaximumFrameRate_Hz	Maximum frame rate threshold [DEFAULT]: 30.0 fps	Anything greater than 0

4.) Consolidated status page element:

FrameParamStatus and FrameParamStatus_Extn pages are used as consolidated status page elements. HOST should allocate memory of appropriate size (i.e. size of FrameParamStatus + FrameParamStatus_Extn) and pass its XP70 domain translated address to FW in FrameParamStatus.ptru32_SensorParametersTargetAddress page element. FW will than copy FrameParamStatus and FrameParamStatus_Extn structure to this address. If this address is not passed by HOST FrameParamStatus and FrameParamStatus_Extn will not be copied.

Status Page	***FrameParamStatus***	

Page Elements	Description	Range
ptru32_SensorParametersTargetAddress	Memory address where the device will dump actual sensor exposure and pipe DG parameters	
u32_ExposureTime	Used exposure time in microseconds	
u32_AnalogGain_x256	Used analogue gain as multiplier (units: gain x 256, e.g. 1536)	
u32_RedGain_x1000	Current red channel gain	
u32_GreenGain_x1000	Current Green channel gain	
u32_BlueGain_x1000	Current Blue channel gain	
u32_frame_counter	count of frame (1-256)	
u32_frameRate_x100	frame rate	
u32_flash_fired	/// flash-lit frame indicator /// Value = 1 indicate it is flash lit frame /// Value = 0 indicate it is normal frame	
u32_NDFilter_Transparency_x_100	Signal Level with or without ND Filter. A provided value of 1250 means a transparency of 12.5%. A value of 0 means ND Filter is not supported [DEFAULT]: 0	
u32_Flag_NDFilter	Current status of ND Filter (whether applied or not), provided it is supported 1: ND filter applied [DEFAULT]: 0	
u32_ExposureQuantizationStep_us	exposure quantization step [DEFAULT]: 0	
u32_ActiveData_ReadoutTime_us	Active data readout time in microseconds [DEFAULT]: 0	
u32_SensorExposureTimeMin_us	Minimum exposure time corresponding to current sensor mode [DEFAULT]: 0	
u32_SensorExposureTimeMax_us	Maximum exposure time corresponding to current sensor mode [DEFAULT]: 0	
u32_applied_f_number_x_100	applied f_numberx100 in the sensor	

	[DEFAULT]: 0	

Status Page	*FrameParamStatus_Extn*	
Page Elements	**Description**	**Range**
u32_SensorParametersAnalogGainMin_x256	Minimum analog gain multiplied by 256 for sensor [DEFAULT]: 0	
u32_SensorParametersAnalogGainMax_x256	Maximum analog gain multiplied by 256 for sensor [DEFAULT]: 0	
u32_SensorParametersAnalogGainStep_x256	Analog gain step multiplied by 256 [DEFAULT]: 0	
u32_StatsInvalid	Whether stats exported are valid or not [DEFAULT]: 1 [INVALID]	Value = 0 indicates stats are valid Value = 1 indicates stats are invalid, and notification is to be ignored.
u32_SizeOfFrameParamStatus	Total size of FrameParamStatus + FrameParamStatus_Extn NOTE: Host must use this address to allocate buffer and programming address in PE ptru32_SensorParametersTargetAddress	>0 0 is invalid
u32_focal_length_x100	Sensor focal length	

5.) Coin toggling mechanism

a. Toggle coin SystemSetup. e_Coin_Ctrl

b. For event please refer Table 6

6.) Exported Statistics Validity

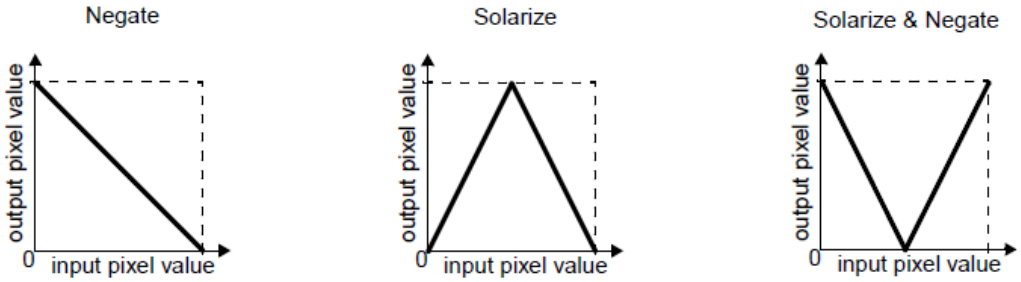
A new element u32_StatsInvalid indicates whether the exported Statistics are Valid or not. Statistics are invalid when they are cancelled. For more information, refer to the Glace and Histogram sections.

17. Special effects

17.1 Special effects

17.1.1 Special Effects Overview

The Special Effects (special_fx) module is used to apply simple transforms onto the RGB input data. The effect of this is to solarise and/or negate the original active data. Same transformation is applied on all channels.



17.1.2 Special Effects Usage

To Control all the special effects, a page element is provided for the HOST which allows host to enable/disable special effects like: Solaris, Negative, Black & White, Emboss and Sepia.

17.1.2.1 Pre-requisites

None

17.1.2.2 Pre BOOT parameters

None

17.1.2.3 Pre RUN parameters

Host will program intended special effect in g_SpecialEffects_Control page element. According to that Firmware will enable/disable that particular special effect.

17.1.3 SpecialEffects_Control Page Elements

17.1.3.1 SpecialEffects_Control page elements

	SpecialEffects_Control [0]:used for PIPE0 Special Effects **SpecialEffects_Control [1]:used for PIPE1 Special Effects**	
Page Elements	**Description**	**Range**

e_SFXSolarisControl	Solaris Control	0: SolariseControl_Enable 1: SolariseControl_Disable
e_SFXNegativeControl	Negative Control	0: NegativeControl_Enable 1: NegativeControl_Disable
~~e_BlackAndWhiteControl~~	~~Not supported~~	~~Not supported~~
~~e_SepiaControl~~	~~Not supported~~	~~Not supported~~

17.1.4 Default Settings recommendation

18. Aperture

18.1 Aperture

18.1.1 Aperture Overview

This provides an interface for host to set & get sensor aperture values.

18.1.2 Aperture Usage

To get the currently set aperture, host should read ApertureConfig_Status.u16_f_number_x_100 value anytime after BOOT command is given

To set a specific aperture, following sequence is to be performed after RUN command is issued

1. Set ApertureConfig_Control. u16_f_number_x_100 to the inended aperture value

2. set ApertureConfig_Control .e_ApertureCommand_Cmd to ApertureCommand_e_SET

3 .Toggle g_SystemSetup.e_Coin_Ctrl coin & wait for this to be equal to g_SystemConfig_Status.e_Coin_Status

18.1.2.1 Pre-requisites

None

18.1.2.2 Pre BOOT parameters

None

18.1.2.3 Pre RUN parameters

None

18.1.3 Page Elements

18.1.3.1 ApertureContro page elements

I	ApertureConfig_Control	
Page Elements	**Description**	**Range/ Possible Values**

u16_f_number_x_100	Select the aperture (format should be f_numberx100)	supported apertures can be read using g_ReadApertureConfig_ControlPE
e_ApertureCommand_Cmd	aperture value x 100 (normally defined as f number)	

18.1.3.2 ApertureConfig_Status page elements

	ApertureConfig_Status	
Page Elements	**Description**	**Range/ Possible Values**
u16_f_number_x_100	Currently applied f_number x 100	ApertureCommand_e_SET = 0, ApertureCommand_e_GET ,

18.1.4 Default Settings recommendation

NA

19. Flash

19.1 Flash

19.1.1 Overview

The flash sub-system is usually situated outside the camera/sensor module. The flash component is likely to be a separate IC. The firmware is responsible for sending strobe signal to this IC, based on flash configuration, managed by the Host. On a successful flash trigger, the firmware informs the Host about the arrival of the flash lit frame, and also reports statistics for the same.

19.1.2 Description

Flash is to affect the exposure of a scene. The factors which affect “exposure with a flash” are:

- Aperture: Since aperture determines the amount of light entering the lens, wider aperture (lowest F-number) enables to use flash more efficiently.
- Sensitivity: The higher the sensor’s sensitivity, again, more the efficiency of the flash.
- Working Distance: The distance between the camera and the subject is one of the most influential factors on exposure in flash photography. In the case of several subjects, it’s recommended to keep them not only within the flash's range, but also in the same plane from the flash, in order to guarantee equal lighting.

The shutter speed doesn’t influence the subject’s exposure when flash is used because the flash is faster than the fastest shutter speed. This is why the shutter speed only affects the background’s exposure and ambient light.

19.1.3 Usage

19.1.3.1 Pre-Requisites

- The mandatory requirement is that the device (sensor) should support flash strobe. This can be known anytime after the firmware boot, from the u8_MaxStrobesPerFrame element in FlashStatus Page (should be greater than zero). The firmware maintains this information throughout its lifetime. For more information, refer to the Flash Page elements description.
- A secondary requirement is that the stats manager should be running, so that stats are gathered for the flash lit frame.

19.1.3.2 Pre Boot Parameters

None

19.1.3.3 Pre Run Parameters

None

19.1.3.4 Live Parameters

All the page elements are live parameters.

19.1.3.5 Modes of Operation

Currently flash operates only in manual mode. Flash is triggered only when the Host requests for it (toggles the coin).

The Host may trigger flash either in VF or Still Capture mode. With respect to triggering of flash, the firmware does not distinguish between these two modes. Hence the method of requesting flash is same for both the modes. However, some other operations (like sending stats, notifications to the Host, etc) are done in BMS (still capture) mode, which is described in 1.1.3.7).

PS: The firmware also does not distinguish between pre-flash and main-flash.

19.1.3.6 Interface between Host and Firmware

1.1.3.6.1 Host to Firmware (Control Page)

The Host gathers information regarding the support of flash in the device and other properties (like strobe minimum and maximum lengths, max strobes per frame, strobe modulation support, etc) from the

FlashStatus Page. The Host can check the values of the elements in this page anytime after sensor boot.

Once the Host knows that flash is supported (as well as other limiting parameter info), he can configure flash by setting the various elements of FlashControl Page (taking care of the limitations). If the Host configures the values beyond the limits specified in FlashStatus, the firmware will crop the values within the range.

The Host issues Flash command by setting g_FlashControl.e_Flag_FlashMode to Flag_e_TRUE and toggling g_SystemSetup.e_Coin_Ctrl coin. If flash was not supported and still the Host toggles this coin, the firmware will return error in the e_FlashError_Info element of FlashStatus page (it will give FlashError_e_FLASH_ERROR_CONFIGURATION error)

A new feature allows Host to just enable Flash (by setting FlashControl.e_Flag_FlashMode to TRUE) in STOP state (before RUN command is given to the FW). At start of streaming, the first frame should typically be a flash lit frame.

Further to this, if the Host toggles System Coin and sets Flashmode to TRUE in STOP state, after giving the RUN command, again, the first frame should typically be a flash-lit frame, and the Host would also receive Glace Stats, with the applied values in FrameParamStatus and FrameParamStatus_Extn Structures.

1.1.3.6.2 Firmware to Host (Status Page)

In a normal case (flash supported by the device; pipe(s) enabled; BMS disabled), if flash strobe is successful, the u8_FlashFiredFrameCount element in FlashStatus Page is set (to 1, normally). This element retains its value till the Host makes another request for Flash configuration (then it is reset to 0). If the Host had requested for sequence flash (flash pulse to be triggered for multiple frames: this can be done by setting u8_FrameCount in FlashControl Page), then FlashFiredFrameCount will be incremented till the firmware is aware of the successful trigger of all the requested strobes. Ideally, when the Host configures for sequence flash (for say, n frames), and toggles the flash coin, he should expect the value of FlashFiredFrameCount to reach n (equal to u8_FrameCount).

The u32_flash_fired element of the FrameParamStatus Page is also set (to 1) when a flash lit frame arrives. This element is again reset to 0 whenever the next non-lit frame arrives. Also, another element in the same Page, u32_frame_counter, is also updated on arrival of a flash-lit frame. It will then give the no of frames streamed since device boot. These elements are useful for debugging purposes.

Once the firmware comes to know that Flash is successfully triggered, it toggles the e_Coin_Flash in FlashStatus and the Flash request is complete. If the Host somehow toggles the coin again (before the previous request is completed), no action will be taken. In case of sequence flash, the firmware will toggle the FlashStatus coin only when all the requested strobes have been fired.

PS: 1. Only single strobe per frame is at present supported by the firmware.

2. Though sequence flash is supported by the (page element) framework, as a feature it is not currently supported.

19.1.3.7 Error Management

1.1.3.7.1 Flash Error Prevention

The limitations of the flash device are exported through the FlashStatus Page, for example, the maximum and the minimum strobe lengths. While configuring flash, if the Host specifies values that exceed this range, the firmware crops the values within the range boundaries. This mechanism currently holds true for u32_StrobeLength_us, u8_StrobesPerFrame and u8_FrameCount elements of FlashControl Page.

1.1.3.7.2 Flash Error Information

Error information is exposed by the firmware in the following enumeration:

```
enum
{
   /// No error
   FlashError_e_FLASH_ERROR_NONE,
```

/// Error in Flash Configuration

FlashError_e_FLASH_ERROR_CONFIGURATION,

/// Error specifying Flash did not trigger

FlashError_e_FLASH_ERROR_STROBE_DID_NOT_TRIGGER,

/// Internal Error thrown outside any flash API call

FlashError_e_FLASH_ERROR_OUT_OF_FLASH_API_CONTEXT

} FlashError;

The only errors which could be of interest to the Host are FlashError_e_FLASH_ERROR_CONFIGURATION and FlashError_e_FLASH_ERROR_STROBE_DID_NOT_TRIGGER. The former will be set when the Host tries to configure the flash even though flash is not supported, or when the configuration fails due to other reasons (internal to the firmware). FlashError_e_FLASH_ERROR_STROBE_DID_NOT_TRIGGER will be set if Flash is not triggered, even after waiting for u8_MaximumNumberOfFrames. Remaining errors have been added for the ease of debugging the firmware and are not useful for the Host. Firmware does some basic error handling of these errors.

The firmware does some basic error handling of these errors, like toggling the FlashStatus coin, resetting some internal states, etc.

Also, in case if flash fire information does not arrive in the firmware (from the device) after several repeated attempts (u8_MaximumNumberOfFrames), then the firmware forcefully updates the status of the flash coin and also increments u8_NumberOfForcedInputProcUpdates. This is a standard procedure also followed by other modules of the firmware (e.g., Exposure).

1.1.3.7.3 Flash Error Control

Yet another level of Error Handling is implemented in case Flash lit frame does not arrive, even after waiting for u8_MaxFramesToWaitForFlashTrigger frames (which is an element in FlashControl Page). Its default value is 10 frames, which the Host can override. In that case, u8_NumberOfConsecutiveDelayedFrames and u8_FlashSyncErrorCount (in FlashStatus) keeps on incrementing, until it reaches u8_MaxFramesToWaitForFlashTrigger, or it receives a flash-lit frame. In both cases, only u8_NumberOfConsecutiveDelayedFrames is reset. On the other hand, u8_FlashSyncErrorCount maintains the history till the next flash configure request.

19.1.3.8 Pages Exposed

FlashControl

FlashStatus

Control Page for Flash	FlashControl	
Page Elements	Description	Range
s32_DelayFromStartPoint_lines	Strobe Delay from Start Point (e_StrobeStartPoint_Frame) in Number of lines [DEFAULT]: 0	-512 to +512
u32_StrobeLength_us	Length of Strobe in Micro sec [DEFAULT]: 0	1 to 10,000 (for Swordfish)
u8_StrobesPerFrame	Number of strobes per frame [DEFAULT]: 1	1

u8_FrameCount	Repeat set of pulses for these many frames [DEFAULT]: 1	>= 1
u8_MaxFramesToWaitForFlashTrigger	Number of frames for which the system should wait for the applied flash to appear. If it does not appear within this number of frames, an error case is generated and e_Flag_ForceInputProcUpdation is updated accordingly. [DEFAULT]: 10	> 0
e_StrobeStartPoint_Frame	For the host to specify the starting point of strobe [DEFAULT]: FLASH_STROBE_AT_EXPOSURE_START	StrobeStartPoint_e_READOUT_STARTi(1) StrobeStartPoint_e_EXPOSURE_START(0)
e_Flag_GlobalResetFrameOnly	Whether the frame is GBRST frame [DEFAULT]: Flag_e_FALSE	Flag_e_TRUE(1) Flag_e_FALSE(0)
e_Flag_DoStrobeModulation	Strobe Modulation support (At present not supported) [DEFAULT]: Flag_e_FALSE	Flag_e_TRUE(1) Flag_e_FALSE(0)
e_Flag_FlashMode	To indicate Flash Mode. Flash can be triggered only if this is TRUE. [DEFAULT]: Flag_e_FALSE	Flag_e_TRUE(1) Flag_e_FALSE(0)
e_Coin_Flash	Not relevant anymore. See g_SystemSetup.e_Coin_Ctrl for details. [DEFAULT]: Coin_e_Heads	Coin_e_Tails (1) Coin_e_Heads (0)

Status Page for Flash	**FlashStatus**	
Page Elements	Description	Range
u32_MinStrobeLength	Minimum Length of Strobe supported by the device [DEFAULT]: 0	
u32_MaxStrobeLength	Maximum Length of Strobe supported by the device [DEFAULT]: 1	
u32_StrobeLengthStep	Strobe Length step [DEFAULT]: 1	
u8_MaxStrobesPerFrame	Maximum Number of strobes per frame. A value of 0 means no flash support. [DEFAULT]: 1	1

u8_FlashFiredFrameCount	Number of frames for which flash is triggered (with respect to u8_FrameCount request in FlashControl_ts) [DEFAULT]: 0	
u8_NumberOfForcedInputProcUpdates	Gives the number of times Input Proc has been forcibly updated Flash request to the sensor has not appeared in a frame for a consecutive number of frames which is as specified by the Control Page. [DEFAULT]: 0	
u8_NumberOfConsecutiveDelayedFrames	Gives the number of consecutive frames for which flash has not appeared in a frame after it has been applied on the sensor. In an ideal scenario, it appears in the very next frame in which it is applied. [DEFAULT]: 0	
u8_FlashSyncErrorCount	Gives the total count of frames for which flash was out of sync [DEFAULT]: 0	
e_Flag_ForceInputProcUpdation	Flag which indicates to the SOF ISR that it has to do the forced updation of Input Proc and let flash be configured again, even though due to some error flash applied on the sensor has not appeared in the MaximumNumberOfFrames as specified by the control page [DEFAULT]: Flag_e_FALSE	Flag_e_TRUE(1) Flag_e_FALSE(0)
e_FlashError_Info	Flash Errors FlashError_e_FLASH_ERROR_NONE	Refer to section 1.1.3.7
e_Flag_StrobeModulationSupported	Strobe Modulation support (At present not supported)	Flag_e_TRUE(1) Flag_e_FALSE(0)
e_Coin_Flash	Not Relevant.	Coin_e_Tails (1) Coin_e_Heads (0)

20. Sensor Tuning & NVM

20.1 Sensor Tuning and NVM values

20.1.1 Overview

Sensor tuning are IQ Tuning parameters which the host may send to the firmware. These parameters are specific to various sensor modes (like Video/VF, Single and burst Still, Flash Modes, etc). Thus they determine and affect various tuning settings specific to these modes.

20.1.2 Sensor and module information

Sensor and module information is available after BOOT. The information is used to pick right Sensor Tuning file from the system. Following information is available. Please refer to sensorinformation table below.

MM=module manufacture id
III=Module id
RR = module revision.

20.1.3 Description

A typical Sensor Tuning parameter list may consist of the following:

Sensor Tuning format ID
Sensor Tuning size
Sensor Tuning name string
Saving year
Saving month
Saving day
Major revision string
zero byte
Minor revision string
zero byte
Number of blocks
Reserved
Block introduction table
Block data

The Block Introduction Table Header may consist of the Block Header and the number of bytes in the Block Data.

The sub-blocks present in the Block data correspond to the various sensor modes discussed above.

20.1.4 Usage

20.1.4.1 Pre-Requisites

For host to send Sensor Tuning configuration data to the firmware, Sensor Tuning must be supported by the firmware. This can be known by the u16_TotalSubBlockIdsCount element of the Sensor Tuning_Status Page. This element should be > 0, for Sensor Tuning to be supported by the device.

20.1.4.2 Pre Boot Parameters

None

20.1.4.3 Pre Run Parameters

All the page elements are Pre Run paramaters.

20.1.4.4 Live Parameters

None of the page elements are live parameters (all of them are Pre Run parameters).

20.1.4.5 Modes of Operation

None

20.1.4.6 Communication between Host and Firmware

1.1.3.6.1 Firmware to Host (Status Page)

Before giving HostInterfaceCommand_e_BOOT command HOST can chose to set Sensor_Tuning_Control_ e_Flag_ReadConfigBeforeBoot_Byte0 PE to Flag_e_TRUE. By default value of this PE is Flag_e_FALSE. If sensor supports Sensor Tuning data, ISP FW provides SensorTuning_Available notifiction. Depending on the value of Sensor_Tuning_Control_ e_Flag_ReadConfigBeforeBoot_Byte0 PE as set by HOST, following 2 cases are possible:-

Sensor_Tuning_Control_e_Flag_ReadConfigBeforeBoot_Byte0 PE set to Flag_e_FALSE :-

This is default case. In this case ISP FW will try to go to HostInterfaceLowLevelState_e_STOPPED state. Only after that Sensor Tuning configuration can be done.

Sensor_Tuning_Control_e_Flag_ReadConfigBeforeBoot_Byte0 PE set to Flag_e_TRUE :-

If HOST wants to configure Sensor Tuning data before getting BOOT_COMPLETE notification, HOST must set Sensor_Tuning_Control_e_Flag_ReadConfigBeforeBoot_Byte0 PE to Flag_e_TRUE. In this case ISP FW will remain in HostInterfaceLowLevelState_e_BOOTING state till HOST sets Sensor_Tuning_Control_e_Flag_ConfigurationDone_Byte0 PE to Flag_e_TRUE. Default value of Sensor_Tuning_Control_e_Flag_ConfigurationDone_Byte0 is Flag_e_FALSE. All Sensor Tuning configuration can be done before setting Sensor_Tuning_Control_e_Flag_ConfigurationDone_Byte0 PE to Flag_e_True, Thereafter ISP FW will try to go to HostInterfaceLowLevelState_e_STOPPED.

In both the above cases, following procedure should be used for doing Sensor Tuning configuration.

ISP FW specifies total number of configuration data sub-block ID count through the u16_TotalSubBlockIdsCount PE of the Sensor Tuning Status Page. If the value of this element is > 0, Sensor Tuning is supported.

The u16_CurrentSubBlockIdValue element gives the Value of index corresponding to the one set in the u16_SelectedSubBlockIdIndex element (in Sensor Tuning Control Page).

1.1.3.6.2 Host to Firmware (Control Page)

As the total number of sub-blocks supported by the device is known, the Host can query the value of Sub-block for any particular index. For this, the SelectedSubBlockIdIndex element is set, and the coin is toggled. The firmware updates the u16_CurrentSubBlockIdValue element and then resets the Status Coin.

Once the Host is aware of the indices supported, he can send Sensor Tuning buffer to the firmware (actually it is parsed Sensor Tuning content, but for the Firmware it is Sensor Tuning buffer only). The host actually has to specify the address of the buffer in the shared address space. The Host might need to allocate a contiguous memory chunk for the same in this shared area, and then mem-copy the buffer contents in that. Anyways, from the firmware's perspective, the buffer must be accessible and in contiguous memory space.

It should be noted that all of the reported sub-blocks may not be present in the Sensor Tuning file with the Host. That should not be treated as error; rather, the Host should feed whatever of the reported sub-blocks it finds in the file. On the other hand, if the Host feeds in some extra sub-blocks (not known to the Firmware), those would be simply ignored by the Firmware (without reporting an error).

20.2 NVM

NVM is Non Volatile Memory. HOST can read size of NVM data by using PE Sensor_Tuning_Status_u16_NVM_Data_Size_Byte0. If sensor supports NVM data, value of this PE should be non-zero after BOOT command has been sent and successfully acknowledged. If NVM data is supported HOST should allocate a buffer of relevant size (as indicated by PE Sensor_Tuning_Status_u16_NVM_Data_Size_Byte0) and pass its translated address (i.e. address translated to XP70 address space) to the FW. PE Sensor_Tuning_Control_u32_NVM_Data_Address_Byte0 is used for this address passing. Thereafter HOST should toggle coin Sensor_Tuning_Control_e_Coin_NVM__Control_Byte0. FW will copy NVM data to the address as indicated by HOST, and notify NVM data copy completion to user using event NVM_EXPORT_DONE (event 0, source 26).

20.2.1 Pasred Or Raw NVM Data

Size of the buffer exported by fw depends on what kind of NVM data is asked. Host sets value of PE Sensor_Tuning_Control.e_TypeNVMExport to either TypeNVMExport_e_Parsed or TypeNVMExport_e_Raw to get resp. output. In Raw export, firmware will not do any calculations and will fill shared buffer with data from EEPROM of the sensor as it is. Incase of Parsed data, firmware will interpret data from EEPROM of sensor, do some calculations if required and will populate the API's for storing NVM data. By default, firmware will export Parsed NVM data.

20.3 Pages Exposed

SensorInformation

Sensor_Tuning_Control

Sensor_Tuning_Status

SensorInformation		
Page Elements	Description	Range
u16_model_id	IIII = Module ID [DEFAULT]: 0x0	
u8_revision_number	RR = Module Revision [DEFAULT]: 0x0	
u8_manufacturer_id	MM = Module Manufacturer ID [DEFAULT]: 0x0	

u8_smia_version	Sensor SMIA version [DEFAULT]: 0x0	
e_Flag_Available	Status if firmware is able to contact sensor. Flag_e_TRUE: Communication successful. Flag_e_FALSE: Communication with sensor failed or some error [DEFAULT]: Flag_e_FALSE	

Control Page for Sensor Tuning		
Page Elements	Description	Range
u32_SubBlock_Data_Address	SDRAM (shared) Buffer Address, which holds the Sensor Tuning config data [DEFAULT]: 0	
u32_NVM_Data_Address	SDRAM (shared) Buffer Address, which is used for holding NVM data [DEFAULT]: 0	
u16_SelectedSubBlockIdIndex	Index of selected configuration data sub-block ID To be used only after firmware boot [DEFAULT]: 0	(0) to (u16_TotalSubBlockIdsCount - 1)
e_Coin_SubBlock_Control	Coin available to HOST for controlling the Sensor Tuning functionality. While querying a particular sub-block Id, the Host needs to toggle this. [DEFAULT]: Coin_e_Heads	Coin_e_Tails (1) Coin_e_Heads (0)
e_Coin_NVM__Control	Coin available to HOST for controlling the NVM data export functionality. NVM data will be exported only when this coin is toggled and u32_Nvm_Data_Address is set to a value other than zero. [DEFAULT]: Coin_e_Heads	Coin_e_Tails (1) Coin_e_Heads (0)
e_Flag_ConfigurationDone	Flag available to HOST for indicating to ISP FW whether Sensor Tuning configuration is complete or not. [DEFAULT]: Flag_e_FALSE	Flag_e_FALSE(0) Flag_e_TRUE(1)
e_Flag_ReadConfigBeforeBoot	Flag available to HOST for indicating to ISP FW if Sensor Tuning configuration will be done before getting BOOT_COMPLETE notification. In this case HOST must set e_Flag_ConfigurationDone flag to Flag_e_TRUE after doing Sensor Tuning configuration.	Flag_e_FALSE(0) Flag_e_TRUE(1)

	[DEFAULT]: Flag_e_FALSE	
e_TypeNVMExport	Set This enum value to TypeNVMExport_e_Raw to export raw NVM data. And to get parsed NVM data, set it to TypeNVMExport_e_Parsed. DEFAULT]: TypeNVMExport_e_Parsed	TypeNVMExport_e_Parsed = 0, TypeNVMExport_e_Raw = 1

Status Page for Sensor Tuning		
Page Elements	Description	Range
u16_TotalSubBlockIdsCount	Total number of configuration data sub-block IDs. Valid only after the firmware boot. [DEFAULT]: 0	> 0 (if firmware supports Sensor Tuning)
u16_CurrentSubBlockIdValue	Value of current configuration data sub-block ID [DEFAULT]: 0	
u16_NVM_Data_Size	Value of NVM data which sensor can export [DEFAULT]: 0	> 0 (if NVM is supported by sensor)
e_Coin_SubBlock_Status	Coin to inform HOST that Sensor Tuning command has completed.	Coin_e_Tails (1) Coin_e_Heads (0)
e_Coin_NVM_Status	Status coin for NVM functionality. NVM data is toggled only when control coin is toggled.	Coin_e_Tails (1) Coin_e_Heads (0)

21. Test pattern

21.1 Sensor Test pattern

21.1.1 Test pattern overview

ISP FW support different test pattern apart from normal streaming modes. In this section test patterns generated from sensor are being discussed. SMIA standard specify test patterns like:

- Solid color
- 100% color bars
- Fade to Grey color bar
- PN9

21.1.2 Test pattern usage

They are used to verify complete data flow and sensor with fixed image pattern.

21.1.2.1 Pre-requisites

None

21.1.2.2 Pre RUN parameters

Any test pattern must be requested before RUN

21.1.2.3 Live parameters

None

21.1.3 Features

- Solid color
- 100% color bars
- Fade to Grey color bar
- PN9

21.1.4 Test pattern page elements

21.1.4.1 Test pattern control

TestPattern_Ctrl	**Control for test pattern**	
Page Elements	**Description**	**Range**
u16_test_data_red	The test data used to replace red pixel data Valid only for TestPattern_SolidColour	
u16_test_data_greenR	The test data used to replace green pixel data on rows that also have red pixels Valid only for TestPattern_SolidColour	
u16_test_data_blue	The test data used to replace blue pixel data Valid only for TestPattern_SolidColour	
u16_test_data_greenB	The test data used to replace green pixel data on rows that also have blue pixels Valid only for TestPattern_SolidColour	
e_TestPattern	Test pattern from sensor [Note]: Set the parameter before RUN command [DEFAULT]: TestPattern_e_Normal	/// Normal streaming operation TestPattern_e_Normal, /// Solid colour TestPattern_e_SolidColour, /// 100% Solid colour bars TestPattern_e_SolidColourBars, /// Face to grey colour bars TestPattern_e_SolidColourBarsFade, /// PN9 pattern TestPattern_e_PN9, /// No supported TestPattern_e_NotSupported,

21.1.4.2 Test pattern Status

TestPattern_Status	Status for test pattern	
Page Elements	**Description**	**Range**
u16_test_data_red	The test data used to replace red pixel data Valid only for TestPattern_SolidColour	
u16_test_data_greenR	The test data used to replace green pixel data on rows that also have red pixels Valid only for TestPattern_SolidColour	
u16_test_data_blue	The test data used to replace blue pixel data Valid only for TestPattern_SolidColour	
u16_test_data_greenB	The test data used to replace green pixel data on rows that also have blue pixels Valid only for TestPattern_SolidColour	
e_TestPattern	Test pattern from sensor [Note]: Set the parameter before RUN command [DEFAULT]: TestPattern_e_Normal	/// Normal streaming operation TestPattern_e_Normal, /// Solid colour TestPattern_e_SolidColour, /// 100% Solid colour bars TestPattern_e_SolidColourBars, /// Face to grey colour bars TestPattern_e_SolidColourBarsFade, /// PN9 pattern TestPattern_e_PN9, /// No supported TestPattern_e_NotSupported,

21.1.5 Default settings recommendation

None

22. OST Traces

22.1.1 Trace Overview

OST Traces are used for debugging purpose. Host has a dedicated block known as System Trace Module (STM). This module provides a memory area to all STxP70, ARM0, ARM1, etc for writing trace data. STxP70 is programmed to write trace data to the memory area dedicated for it. To receive OST traces through STM IP on 8500, Trace collector devices like combiprobe is must. ISP FW also supports trace logging in memory area shared between host and ISP FW. Trace data is converted into an ASCII buffer and is dumped into the shared memory in circular buffer mode. To use traces in memory feature, the feature to read traces must be available on host side.

The following API implementations are available for writing trace data:

- OstTraceInt0(aGroupName, aTraceText)
- OstTraceInt1(aGroupName, aTraceText, aParam1)
- OstTraceInt2(aGroupName, aTraceText, aParam1, aParam2)
- OstTraceInt3(aGroupName, aTraceText, aParam1, aParam2, aParam3)
- OstTraceInt4(aGroupName, aTraceText, aParam1, aParam2, aParam3, aParam4)

Available group names along with their bit locations (separated by colon) are as under:

- TRACE_ERROR:0
- TRACE_WARNING:1
- TRACE_FLOW:2
- TRACE_DEBUG:3
- TRACE_API:4
- TRACE_OMX_API:5
- TRACE_OMX_BUFFER:6
- TRACE_RESERVED:7
- TRACE_USER1 :8
- TRACE_USER2 :9
- TRACE_USER3 :10
- TRACE_USER4:11
- TRACE_USER5:12
- TRACE_USER6:13
- TRACE_USER7:14
- TRACE_USER8:15

User should create a 16bit word selecting the trace group to be enabled and write that mask to TraceLogsControl. u32_LogLevels. Writing 1 at any trace group bit field will enable that trace group and writing 0 will disable that trace group. Standard printf format will be used to specify arguments while calling above mentioned API's. e.g.

OstTraceInt1 (TRACE_FLOW, "This is an example No: %d", aParam1);

22.1.2 PE TraceMechanismSelect

22.1.2.1 TraceMechanismSelect = 1 i.e. XTI

If user wants to use traces using XTI, then values in TraceLogsControl.u32_BufferAddr and TraceLogsControl.u32_BufferSize are ignored. And all the trace data is sent to STM's memory area. Now depending on the value of some trace related flags in Firmware, either binary or ASCII traces will be sent on output. In case if Binary is selected in firmware, then user needs to

- Run trace compiler on FW to generate dictionary
- In test environment, include the generated dictionary file in directory as required by trace decoder. Trace dictionary in form of STxP70_FW_TRACES_Dict.xml is part of release in ./DeviceParams directory

22.1.2.2 TraceMechanismSelect = 2 i.e. Memory Logging

If user wants to log traces in the memory area, then its responsibility of the user to create a buffer in Host memory space at boot time. User must share address and size of the buffer with STxP70 firmware using Page Element TraceLogsControl.u32_BufferAddr and TraceLogsControl.u32_BufferSize respectively.

22.1.2.3 TraceMechanismSelect =0 i.e. NoMesg Output

If user sets this PE value to 0 or any value other than 1 or 2, then there will be no trace output.

22.1.3 Usage

22.1.3.1 Pre-requisites

None

22.1.3.2 Pre BOOT Parameters

If using Trace logging in memory then create a memory buffer before booting and write buffer size and address in PE.

22.1.3.3 Pre RUN Parameters

If using OST Trace mechanism: Map IO space to STM's base address so that trace data can be sent to STM. Currently we are mapping IO space 1 for 8500v1 board and IO Space 2 for 8500v2 board.

22.1.3.4 Live Parameters

All the PE's can be changes any time and will be taken into effect.

22.1.4 OSTTrace Page Elements

22.1.4.1 TraceLogsControl

Page Elements	Description	Range
u32_BufferAddr	Address of the memory area shared by HOST	
u32_BufferSize	Size of the buffer shared by Host	
u32_LogLevels	Control to filter various trace groups	0x0 – 0xFFFF
u8_LogEnable	Enable/Disable the traces	0-1
u8_TraceMechanismSelect	Select among different trace mechanism available 0: No mesg 1: XTI 2: Memory Logging	0-2

Table 1 TraceLogsControl

23. Sensor and ISP Settings Interfaces

23.1 Sensor and ISP Settings

23.1.1 Overview

These interfaces have been provided to the Host to independently control the configuration of the Sensor and the Pipe. The objectives for implementing these interfaces were to segregate the sensor and pipe configuration, as well as to provide an interface to the Host independent of SystemCoin Control. One obvious point also to be noted is that these settings can be set anytime by the Host, but they shall be applied only when the Sensor and the Pipe are streaming.

23.1.2 Sensor Settings Interface

The Host can set Exposure, Analog Gain and Framerate values, and toggle g_SensorPipeSettings_Control.e_Coin_SensorSettings. In streaming state (after RUN command is given), the Host would receive SensorCommit Notification in ITM1 register, bit 1, confirming that the request has completed.

23.1.3 ISP Settings Interface

The Host can set various params for the various ISP blocks (for example, digital gain), and toggle g_SensorPipeSettings_Control. e_Coin_ISPSettings. In streaming state (after RUN command is given), the Host would receive PipeCommit Notification in ITM1 register, bit 2, confirming that the request has completed.

23.1.4 Pages Exposed

SensorPipeSettings_Control Page

Page Elements	Description	Range
e_Coin_SensorSettings	The Host will toggle this coin for requesting the FW to apply the various Sensor Parameters (Exposure, Analog Gain, etc.) DEFAULT VALUE: Coin_e_Heads	0-1 (Coin_e_Heads, Coin_e_Tails)
e_Coin_ISPSettings	The Host will toggle this coin for requesting the FW to apply the various ISP Parameters (Digital Gain, Gridiron, Duster, etc.) DEFAULT VALUE: Coin_e_Heads	0-1 (Coin_e_Heads, Coin_e_Tails)

SensorPipeSettings_Status Page

Page Elements	Description	Range
e_Coin_SensorSettings	ISP_FW will set this coin equal to the corresponding Control coin, once the request is complete (the Sensor parameters have been applied) DEFAULT VALUE: Coin_e_Heads	0-1 (Coin_e_Heads, Coin_e_Tails)
e_Coin_ISPSettings	ISP_FW will set this coin equal to the corresponding Control coin, once the request is complete (the various Pipe parameters have been applied) DEFAULT VALUE: Coin_e_Heads	0-1 (Coin_e_Heads, Coin_e_Tails)