SMIA 1.0 Part 1: Functional specification

DISCLAIMER

History

Version	Date	Author	Status	Notes
1.0	30-June-2004	Nokia and ST	Approved	

Table of contents

List of tables

List of figures

Acronyms Abbreviations and Definitions:

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
LE	Line End
LS	Line Start
LSB	Least Significant Byte
LVDS	Low Voltage Differential Signalling
Mbps	Megabits per second
MSB	Most Significant Byte
MIPI	Mobile Industry Processor Interface
MSP	Manufacturer Specific Pixels
OC	Open Collector
OD	Open Drain
PCK	Pixel Clock
PCM	Pulse Code Modulation
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 Signalling
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)
WOI	Window of Interest

Table 1: Acronyms

PREFACE

Specification Supersedes Earlier Documents

This document contains the draft SMIA Functional specification.

Following publication of the SMIA Standard, there may be future approved errata and/or approved changes to the standard prior to the issuance of another formal revision.

Incorporation of Engineering Change Requests (ECRs)

The following ECRs have been incorporated into this version of the specification:

ECR	DESCRIPTION

Table 2: ECR

1 Overview

1.1 Objective

The SMIA Functional Specification objective is to fully standardise the electrical, control and image data interfaces for raw Bayer sensor modules targeted at mobile applications. Knowing how the basic system will function regardless of camera attachment resolution and frame-rate by baseline compliancy

The main objective is driven by the requirement to be able to connect ANY SMIA compliant sensor to ANY SMIA compliant host system with matching capabilities and get a working system with acceptable performance.

This self configuration nature is a key feature of the Functional Specification and vital to the overall SMIA objectives

- Making a Raw Bayer Sensor Module easier to integrate in a system to create a 'camera'
- Creating a standard applicable to a wide range of sensor resolutions and frame-rates, to proliferate a family of 'like' modules with known upgrade paths.
- Improving design and verification cycles
- Enabling true second sourcing of sensor modules.

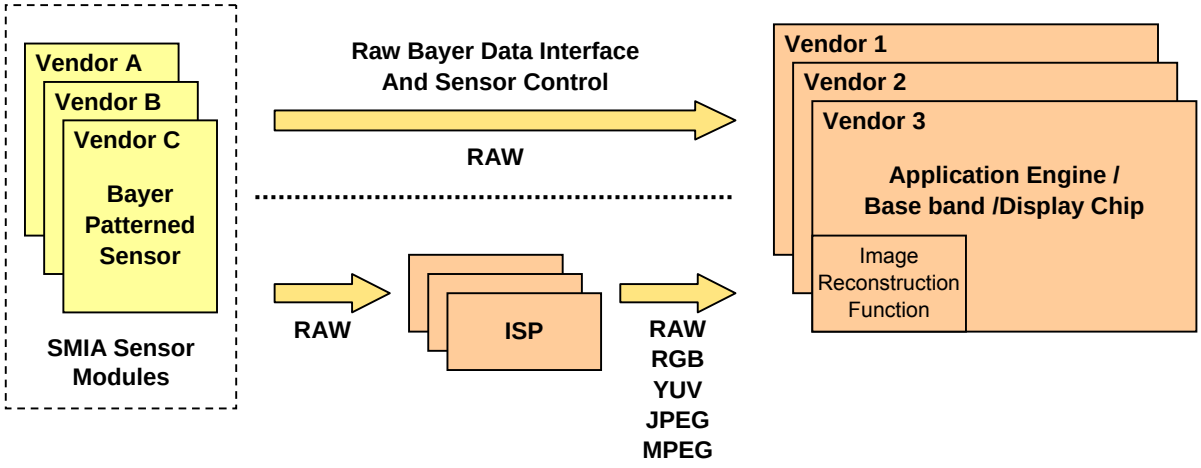


Figure 1: System Overview Example

The functional specification must be equally applicable to both software host implementations as well as hosts implemented with dedicated hardware machines. There can also be back channel from host system to the ISP that is not drawn in Figure 1 and in Figure 2.

In this context the host system may be either an Application Engine, a base band or a display chip.

The AEC, AWB and image reconstruction algorithms required to make a complete the 'Camera' system can be implemented is a variety of ways

- Dedicated parameterisable hardware.

- Software on a DSP/Processor.
- Hybrid approach of mixed software and programmable hardware.

The above implementations can be either part of the Application Engine or a separate Image Signal Processor device.

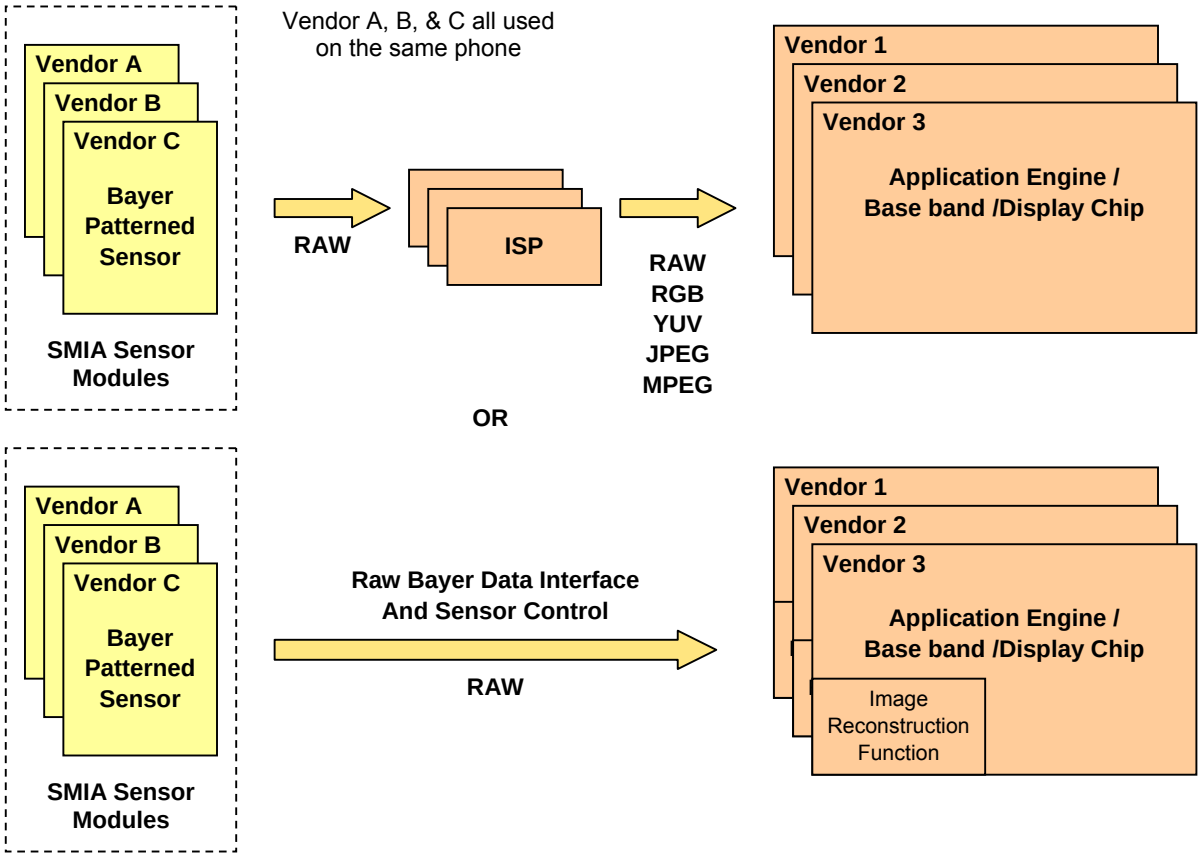


Figure 2:System Examples

The approach in the specification, must allow the required degree of functionality for a working system without the host having to be supplied with any additional fundamental information about the sensor module. This is achieved through the use of a 'baseline' profile and the description of basic capabilities, which the transmitter must comply with.

Beyond the 'baseline' profile there are 2 incremental profile levels which add image scaling and image compression features.

The host receiver must support all of the profile levels. Thus a host receiver must support the RAW10, RAW8 data formats and 10-bit to 8-bit DPCM/PCM decompression

In the application the sensor module set-up/configuration parameters are mastered by a 'personality file' when available in the software stack, this is also capable of setting up the parameterised reconstruction chain according to the sensor module attached in order to customise the look and feel of the camera, and allowing a route for specification evolution and manufacturer specific customisation without compromising the fundamental issue of compatibility

It is important that the specification can be read and implemented not only by Module builders, but also by those implementing the Host system including the software application. The benefits of a well-controlled and clear Functional Specification for both Module and Host implementers are:

- A single generic model describing the behaviour of all of image sensor modules
- Guaranteed sensor module / host inter-operability.
- A single host implementation can work with a wide range of sensor modules.
- A single Module implementation can work with a wide range of hosts.

1.2 The Specification

The Functional specification defines the following

- Electrical Interface - including pin out
- Operating Modes – how to power up and down the module
- Data format, and data arrangement
- Video Timing, cropping and decimation modes
- Integration Time and Gain Control
- Reporting of Sensor capabilities and key performances (e.g. Colour Matrix)
- Test Modes
- CCI Register Map
- A baseline profile
- Image Scaling profiles to aid software viewfinder implementations
- Image Compression
- Recommended Host Behaviour

The Functional specification adds a layer on top of the CCP2 specification in that it fully defines the sensor modules electrical, control and image data interfaces.

The physical and logical layers of the SMIA Functional specification is compliant with the CCP2 specification.

The electrical interface is as follows (Figure 4):

- CCP2

CCP2 is a 4-wire Low EMI SubLVDS data link. The maximum data rate is 650 Mbps.
This is sufficient to support readout of an image of up to 2 Mega Pixels in a 1/30th of a second using the RAW8 CCP2 mode.

- CCI

The 2-wire camera control interface is I^2C compatible. The maximum data rate is 400 kHz.

- Low frequency external clock (6 – 27MHz)
- Chip enable signal - XSHUTDOWN
- Analogue Power
- Digital Power

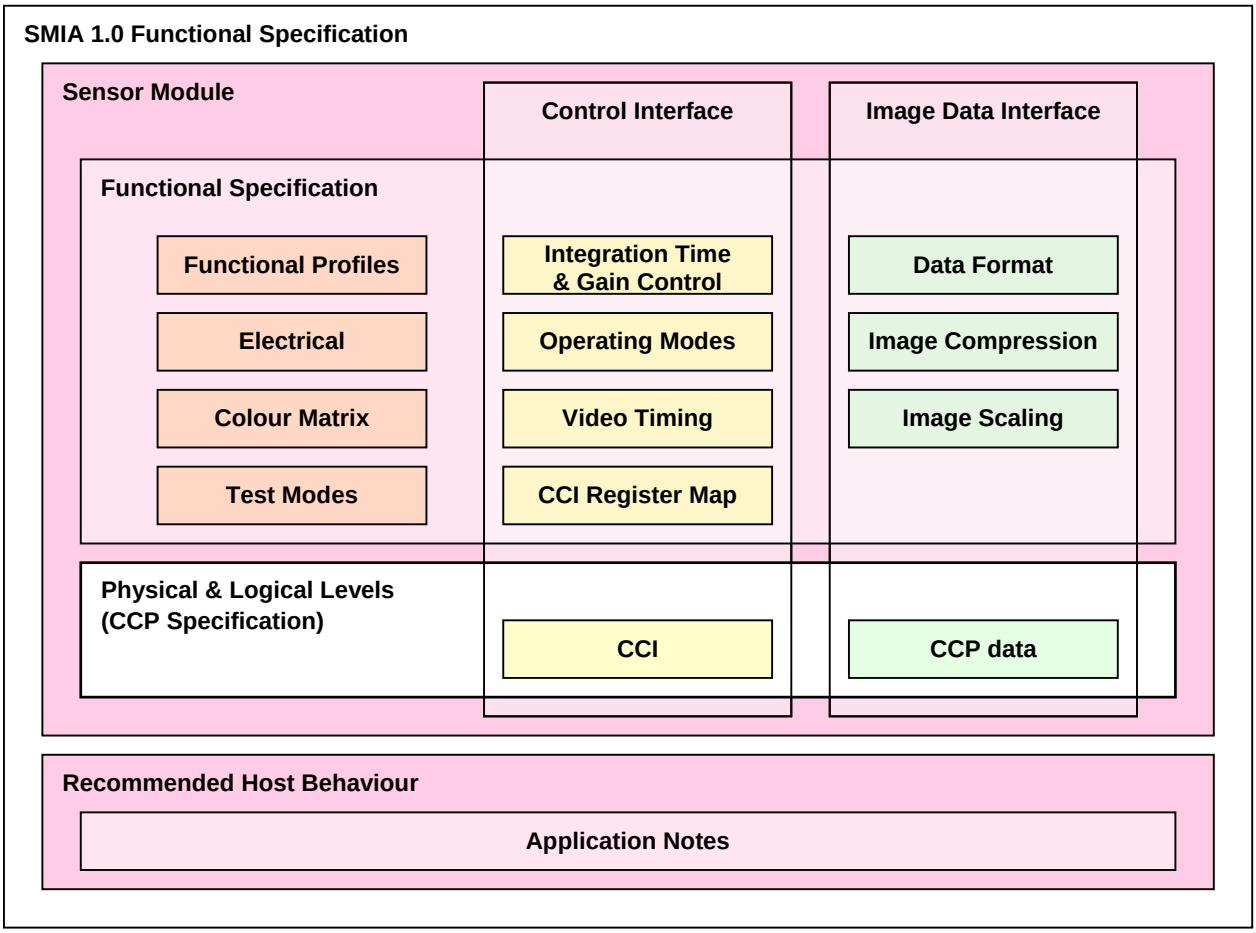


Figure 3: SMIA Functional Specification Overview

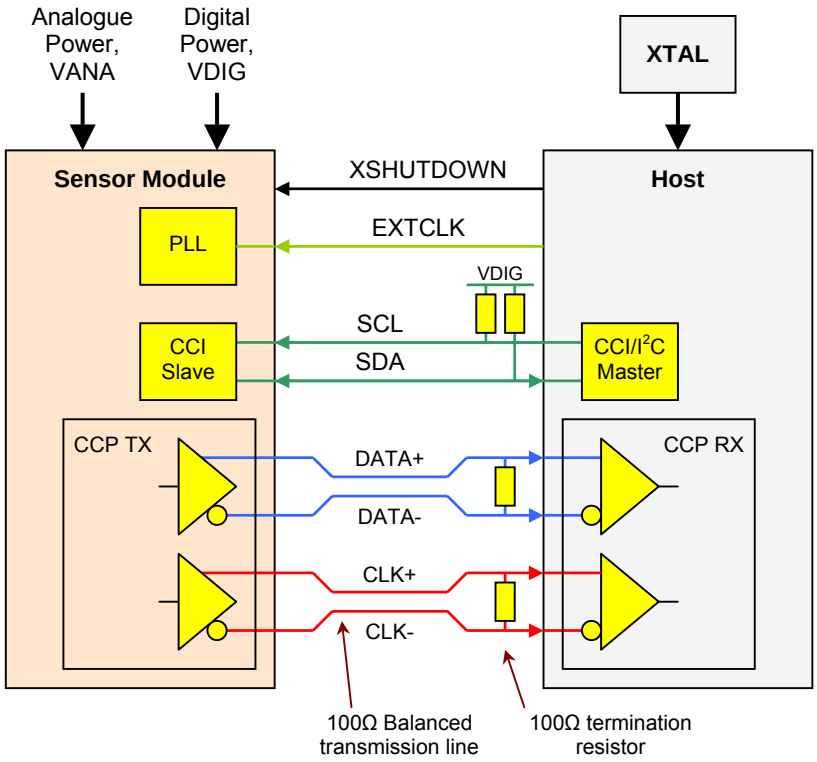


Figure 4: Sensor Module / Host Electrical Interface

1.3 Functional Profiles

Table 3 defines the features of the different sensor profile levels.

The incremental nature of the profiles allows level 1 or 2 devices to be specified initially and as the capabilities of the base-bands increase or with the integration of video DSP type hardware in the base-bands, profile level 0 devices can be specified instead without needing to change the SMIA specification.

Levels 1-2 represent an increasing level of data rate reduction for video applications i.e. viewfinder in high resolution sensor modules.

Profile Level	Base Line (Inc WOI)	Image Scaling	Compression
0	RAW10		
1	RAW10 + RAW8	Horizontal	10-b to 8-b DPCM/PCM
2	RAW10 + RAW8	Full	10-b to 8-b DPCM/PCM

Table 3: Functional Profiles

Output Data Rate	Signalling Type
<= 208 Mbps	Data/Clock
>208 Mbps	Data/Strobe

Table 4: Data Rates

All sensor modules must support the RAW10 data format while profile levels 1 and 2 must also support the RAW8 format.

There are 2 CCP2 RAW8 image data modes:

- 10-bit pixel data truncated to 8-bits (i.e. top 8-bits of the 10-bit data)
- 10-bit pixel data compressed to 8-bits via the DPCM/PCM specified in the compression chapter.

Baseline Features

- Electrical
- Operating Modes
- Data Format
- Video Timing (including programmable Window of Interest)
- Integration Time and Gain Control
- Colour Matrix
- Test Modes
- Camera Control Interface
- Register Map
- Recommended Host Behaviour

Additional Features:

- Image Scaling - 3 Levels None, Horizontal and Full (Horizontal and Vertical)
- 10-bit to 8-bit DPCM/PCM raw Bayer Compression

External References

- SMIA 1.0 Part 2: CCP2 Specification

Also defined within this specification are additional data format and image compressions options, which are extensions beyond the sensor profile levels.

It is important to emphasise that these extensions are NOT part of the sensor profiles.

1.4 Key Concepts

There are certain concepts (below) that combine together to give the Functional Specification desired self-configuration behaviour.

1.4.1 Baseline Compliancy

The baseline is the minimum specification a sensor module must be compliant with in order to achieve a functional system. Features such as image scaling and image compression are viewed as additions to the baseline specification.

For example the baseline compliancy for integration time and analogue gain is coarse integration time and a single global gain value. If a sensor module has per channel analogue gain capability, its default (baseline) behaviour must be as if it has a single global analogue gain. If the host receiver AEC/AWB supports analogue gain per channel the host can discover this via the gain capability register and then enable this feature. Similar baseline behaviours are also single scaling factor and scaling step in Image Scaling (Chapter 9) and only even divider values used in Video Timing (Chapter 5).

This is necessary to create a common baseline behaviour across all sensor modules and thus guaranteeing sensor/host interoperability.

All of the sensor module defaults must be baseline compliant i.e. RAW10 data format, no image scaling, not image compression, global analogue gain behaviour, coarse integration and only even divider values.

1.4.2 Intelligent Status Line

Within a frame of CCP2 data additional information about the set-up and configuration of the sensor module is embedded at the start and the end of the frame of data.

This embedded data contains the contents of the sensor modules CCI registers and thus fully describes the state of the sensor module

The embedded data at the start of the frame is termed the “Intelligent status line(s)”.

Examples contents of the “Intelligent Status Line”

- Generic frame format description describing the content and structure of the frame of CCP2 image data.
- Integration Time and Gain Parameters
- Video Timing Parameters

A good analogy is the EXIF information (header) within a JPEG image file.

The reasons behind the “Intelligent Status Line” are as follows:

- To enable the co-processor the receiver to process arbitrary sized images, via the generic frame format description
- It provides the co-processor with a fast method of receiving the contents of the image sensors internal CCI register values
- It lowers the overhead of CCI communications on the co-processor microprocessor by eliminating the need when the imager is streaming video data for the co-processor to read via the CCI serial interface.

This is of particular help when the viewfinder, AEC and AWB functions are being performed in software or in firmware running on a microprocessor with a limited number of MIPS.

Use of the “Intelligent Status Line” must be a central feature of the host receiver implementations.

1.4.3 Synchronisation

Sensor control commands, status data and the control loops that run on the host need to be synchronised, for example Automatic White Balance and Exposure.

This is a vital point for the use of the data with the “Intelligent Status Line”.

The embedded data must be correct for image data contained in that frame. For example the integration time and gain parameter values reported are those that are used by the subsequent image data output within that frame. Additionally the information for next frame image parameter values can be reported, if they are already available.

The host must use the values contained in the Intelligent Status line to synchronise the AEC and AWB control loops. For example after the host transmits a new set of integration time and gain parameters it waits for the new parameter values to appear in the Intelligent Status Line. Thus the host knows exactly when the first frame with the updated integration time and gains values appears.

This is a vital point for ensuring correct system behaviour for exposure control, as if the system thinks it has a different exposure/gain in the sensor than it really has, ‘because it is out of sync’, then it may apply the wrong digital gain for that frame causing image flashes to be observed.

1.4.4 Parameter Re-timing

The host must be able to send commands to the sensor at any point in time, in order that these registers do not corrupt the frame in progress in an ill-defined manner, they must be re-timed internally to the start of the next frame.

Integration time, gain and video timing parameters must be consistent within a frame of image data. To achieve this the sensor module control registers must be internal re-timed so that they are always applied at the start of the next frame.

In practise this means that there is an optimum period for dynamic sensor control commands to be sent to the sensor to minimise the loop time and improve the response of the system.

1.4.5 Generic Frame Format Description

The frame format description is a central part of enabling the development of software and hardware receivers, which can process arbitrary, sized images.

The frame format information is available to either software or hardware receiver via both CCI Registers and the embedded data within the intelligent status line.

1.4.6 Capability Registers/Parameter Limits

The capability registers provide a method of the host system to automatically discover the abilities of the sensor module.

The capability registers in reality describe the limits of parameters such as

- Integration Time
- Analogue Gain
- Line Length
- Frame Length

Thus for example the AEC/AWB control loops can be automatically configured without the host having to have prior knowledge of the sensor behaviour.

1.4.7 Generic Control Interface

The generic register map provides a common command interface for controlling the sensor module.

1.4.8 Personality file(s)

The personality file(s) while not required to achieve a working system, provide a way for the performance of the sensor module to be optimised and manufacturer specific features to be handled. The personality file can also contain additional information e.g. vignetting elimination parameters and colour component sensitivities. And the personality file can contain information from set of different sensors that can be used in products.

The capability information read form the sensor module is merged with any information contained in the personality file – the personality file values have priority over sensor module values – and merged set of information is used by the host system.

1.4.9 Recommended Host Behaviour

The host must implement the following functions to create a complete 'Camera' system.

- AEC and AWB Control
- Viewfinder and Still Image Reconstruction
- Colour matrixing
- Set up and control the Video timing
- Read the sensor parameter limits
- Personality information

Overriding sensor parameters, manufacturer specific registers and additional information

A host receiver implementation must go beyond the baseline specification in the features it supports. A host receiver must support the following data formats.

- RAW10 – top 10-bit of internal pixel data
- RAW8 – top 8-bit of internal pixel data
- RAW8 + 10-bit to 8-bit DPCM/PCM compression.

SMIA 1.0 Part 3.1: Software and Application specification defines how the host should use the features specified in the functional specification to achieve a sensor module / host interface which is self-configuring. Example areas covered

- Step by step to get image out
- Power up/down sequence
- Basic configuration
- Configuration for viewfinder
- Configuration for video
- Configuration for still
- Features for AEC/AWB control
- Smooth frame rate control
- Features for receiving arbitrary sized images
- Scaling
- Compression
- Working with Flash (LED and Xenon) and Shutters
- Digital Zoom
- Auto focus
- Video Timing

1.5 Baseline Feature Chapter Summaries

1.5.1 Electrical

- Signal List
- 12/14/16 Pin Out options
- Electrical characteristics of EXTCLK and XSHUTDOWN
- Operating Conditions
- Absolute Maximum Ratings
- Power Consumption

1.5.2 Operating Modes

- Definition of Operating Modes
- Power Up Sequence
- Power Down Sequence

1.5.3 Data Format

- High-level frame structure
- Generic Frame format description
- Embedded Data Format

1.5.4 Video Timing

- Programmable image Size
- Sub-Sampling
- EXTCLK to pixel clock relationship
- Variable Line Length and frame length
- Video Timing Requirements
- Parameter re-timing

1.5.5 Integration Time And Gain Control

- Fine and Coarse Integration Time
- Analogue Gain
- Digital Gain
- Synchronisation of AEC/AWB control loops
- Parameter Re-timing

1.5.6 Colour Matrix

- Sensor Bayer to sRGB Colour matrix

1.5.7 Camera Control Interface

- 16-bit index, 8-bit data variant of the CCI slave.

1.5.8 Register Map

- Register Locations

1.6 Additional Feature Chapter Summaries

1.6.1 Image Scaling

Scaling options

- None
- Horizontal
- Full (Horizontal & Vertical)

1.6.2 Compression

Image Compression option:

- 10-bit to 8-bit DPCM/PCM based image compression.

1.7 Extensions Beyond the Sensor Profile Levels

1.7.1 Data Formats

- CCP2 RAW6
- CCP2 RAW7
- CCP2 RAW12

1.7.2 DPCM/PCM Compression

- 10-bit to 6-bit compression transmitted via the CCP2 RAW6 format
- 10-bit to 7-bit compression transmitted via the CCP2 RAW7 format

2 Electrical

2.1 Introduction

This chapter defines the sensor module's electrical interface. The areas defined are:

- Module pin out
- Signal list
- Reference support circuit
- External Clock
- XSHUTDOWN Signal
- CCI SDA and SCL Signals
- Absolute Maximum Ratings
- Operating Conditions
- Average current consumption
- Average power consumption
- Peak current
- In rush current

For the full details of the CCP2 and CCI interfaces please refer to the SMIA 1.0 Part 2: CCP2 specification.

An important part of the SMIA electrical specification is providing the worst-case current consumption figures for the mobile phone's power management design.

Thus if the phone's power management is designed to supply the maximum currents for the chosen resolution, then it will be compatible with all SMIA compliant sensor modules with that resolution.

Without this guaranteed compatibility between a phone's power management design and the sensor module it is not possible to swap in and out different manufacturers sensor modules within the same phone platform.

2.2 Sensor Module Pin Order

2.2.1 16-Pin Sensor Module

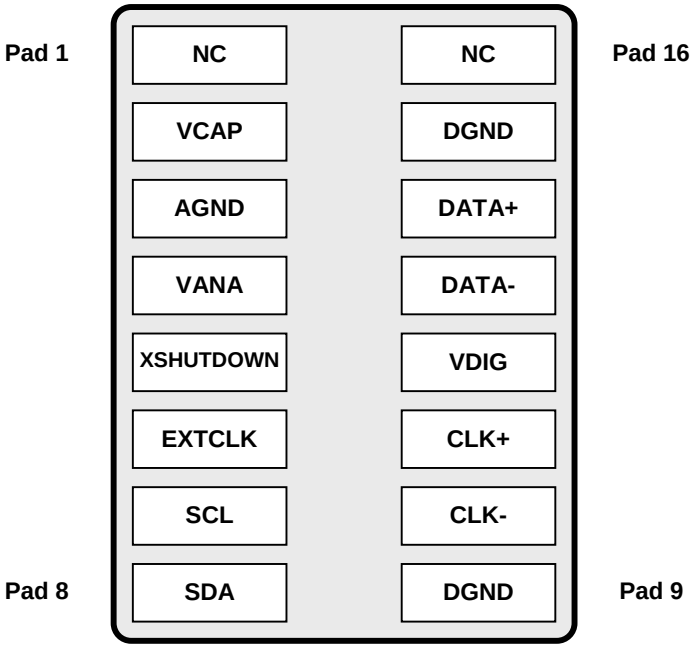


Figure 5: 16-Pin Out (Bottom View)

2.2.2 14-Pin Sensor Module

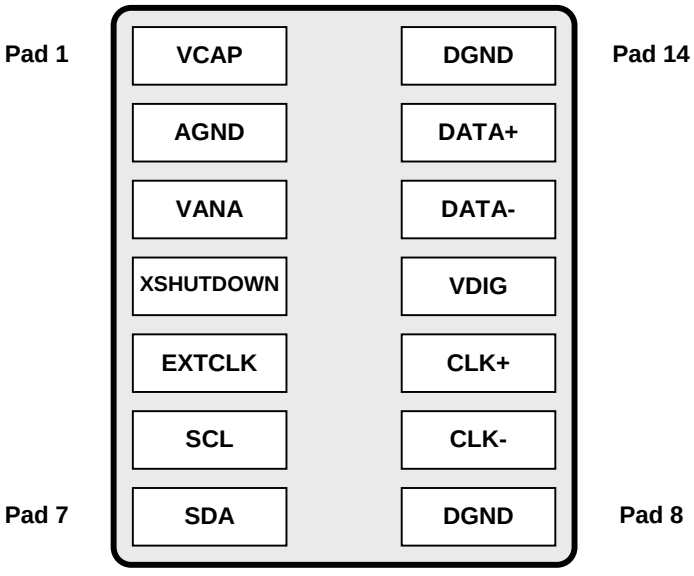


Figure 6: 14-Pin Out (Bottom View)

2.2.3 12-Pin Sensor Module

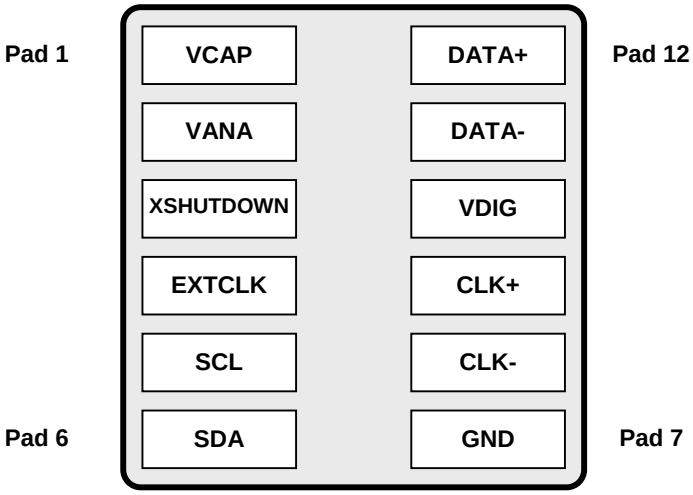


Figure 7: 12-Pin Order (Bottom View)

2.3 Connection Pads

2.3.1 16-Pin Sensor Module

Pad	Name	Direction	Function
1	NC	-	No Connect Reserved for Future Expansion
2	VCAP	ANA	External Capacitor connected to Ground 100nF for resolutions up to SVGA 200nF for resolutions greater than SVGA VCAP Maximum Voltage = 4.2V **Capacitor Specification:** 0402 Type, X5R dielectric, K tolerance (+/-10%), Rated Voltage 10V
3	AGND	POWER	Analogue power Local Decoupling 100nF capacitor to ground
4	VANA	POWER	Analogue power
5	XSHUTDOWN	IN	Active low shutdown signal
6	EXTCLK	IN	Input clock from host system
7	SCL	IN	CCI clock signal
8	SDA	IN/OUT	CCI data signal
9	DGND	POWER	Digital Ground
10	CLK-	OUT	Differential CCP2 clock signal (negative)
11	CLK+	OUT	Differential CCP2 Clock Signal (positive)
12	VDIG	POWER	Digital power Local Decoupling 100nF capacitor to ground
13	DATA-	OUT	Differential CCP2 data signal (negative)
14	DATA+	OUT	Differential CCP2 data signal (positive)
15	DGND	POWER	Digital Ground
16	NC	-	No Connect Reserved for Future Expansion

Table 5: Connection Pad List for 16-pin Sensor Modules

2.3.2 14-Pin Sensor Module

Pad	Name	Direction	Function
1	VCAP	ANA	External Capacitor connected to Ground 100nF for resolutions up to SVGA 200nF for resolutions greater than SVGA VCAP Maximum Voltage = 4.2V **Capacitor Specification:** 0402 Type, X5R dielectric, K tolerance (+/-10%), Rated Voltage 10V
2	AGND	POWER	Analogue power Local Decoupling 100nF capacitor to ground
3	VANA	POWER	Analogue power
4	XSHUTDOWN	IN	Active low shutdown signal
5	EXTCLK	IN	Input clock from host system
6	SCL	IN	CCI clock signal
7	SDA	IN/OUT	CCI data signal
8	DGND	POWER	Digital Ground
9	CLK-	OUT	Differential CCP2 clock signal (negative)
10	CLK+	OUT	Differential CCP2 Clock Signal (positive)
11	VDIG	POWER	Digital power Local Decoupling 100nF capacitor to ground
12	DATA-	OUT	Differential CCP2 data signal (negative)
13	DATA+	OUT	Differential CCP2 data signal (positive)
14	DGND	POWER	Digital Ground

Table 6: Connection Pad List for 14-pin Sensor Modules

2.3.3 12-Pin Sensor Module

Pad	Name	Direction	Function
1	VCAP	ANA	External Capacitor connected to Ground 100nF for resolutions up to SVGA 200nF for resolutions greater than SVGA VCAP Maximum Voltage = 4.2V **Capacitor Specification:** 0402 Type, X5R dielectric, K tolerance (+/-10%), Rated Voltage 10V
2	VANA	POWER	Analogue power Local Decoupling 100nF capacitor to ground
3	XSHUTDOWN	IN	Active low shutdown signal
4	EXTCLK	IN	Input clock from host system
5	SCL	IN	CCI clock signal
6	SDA	IN/OUT	CCI data signal
7	GND	POWER	Common Digital/Analogue Ground
8	CLK-	OUT	Differential CCP2 clock signal (negative)
9	CLK+	OUT	Differential CCP2 Clock Signal (positive)
10	VDIG	POWER	Digital power Local Decoupling 100nF capacitor to ground
11	DATA-	OUT	Differential CCP2 data signal (negative)
12	DATA+	OUT	Differential CCP2 data signal (positive)

Table 7: Connection Pad List for 12-pin Sensor Modules

2.4 Standard Support Circuit

This section defines a reference support circuit for SMIA sensor modules. The aim is to define a common reference platform so that sensor modules can be tested under exactly the same conditions.

Figure 8, Figure 9 and Figure 10 detail the standard support circuits for 16-pin, 14-pin and 12-pin modules respectively.
Table 8 defines the values for the reference components used in the support circuit.

Component	Symbol	Specification	
CCI Pull-Up Resistor	R_P	Value:	4.7kΩ
		Dissipation:	**TBC**
		Tolerance:	±5%
		Size:	**TBC**
CCP2 Termination Resistor	R_T	Value:	100Ω
		Dissipation:	**TBC**
		Tolerance:	±5%
		Size:	**TBC**
Local Capacitor	C_L	Value:	100nF
		Dielectric:	X5R
		Tolerance:	K (±10%)
		Voltage:	10V
		Size:	0402
Bulk Capacitor	C_B	Value:	**TBC**
		Dielectric:	**TBC**
		Tolerance:	**TBC**
		Voltage:	**TBC**
		Size:	**TBC**
Voltage Regulator	N	TBC	**TBC**

Table 8: Standard Support Circuit Reference Components

2.4.1 16-Pin Standard Support Circuit

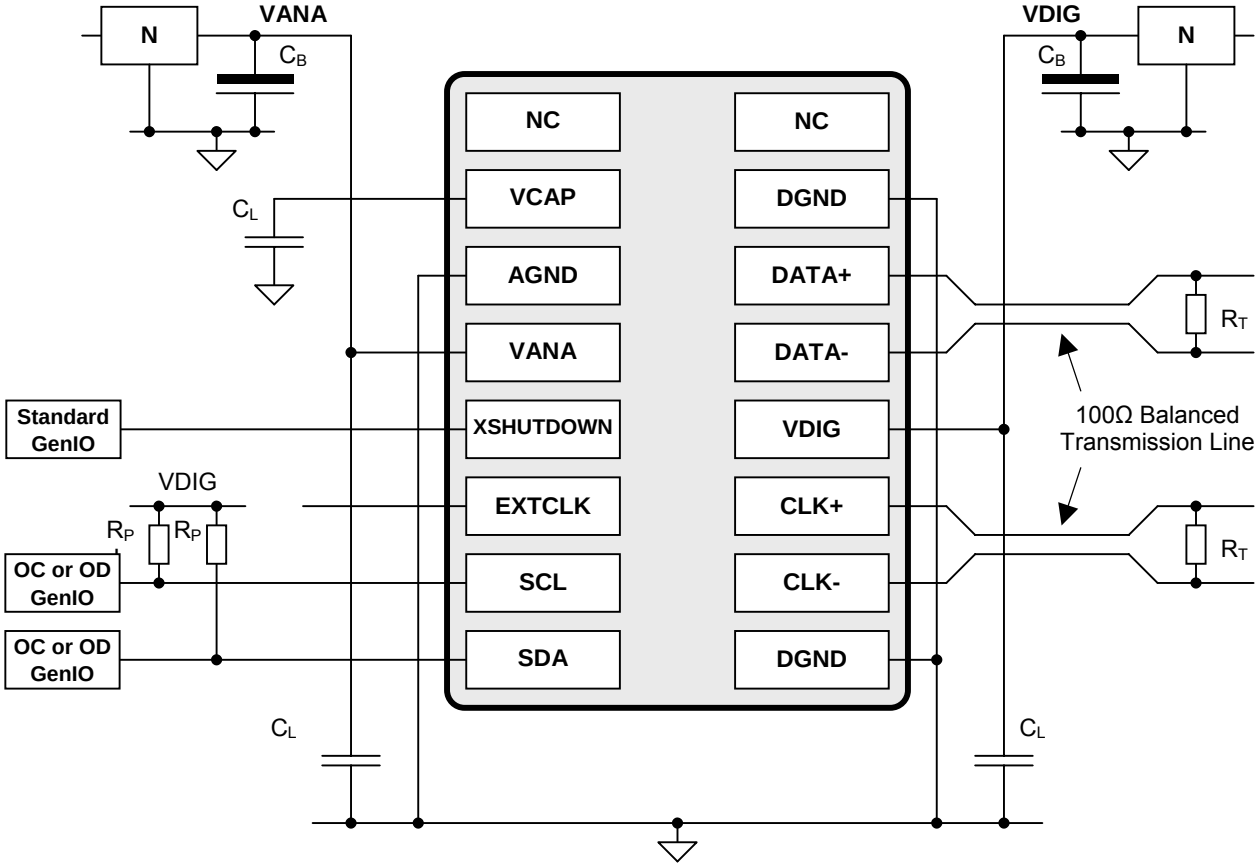


Figure 8: 16-Pin Standard Support Circuit

14-Pin Standard Support Circuit

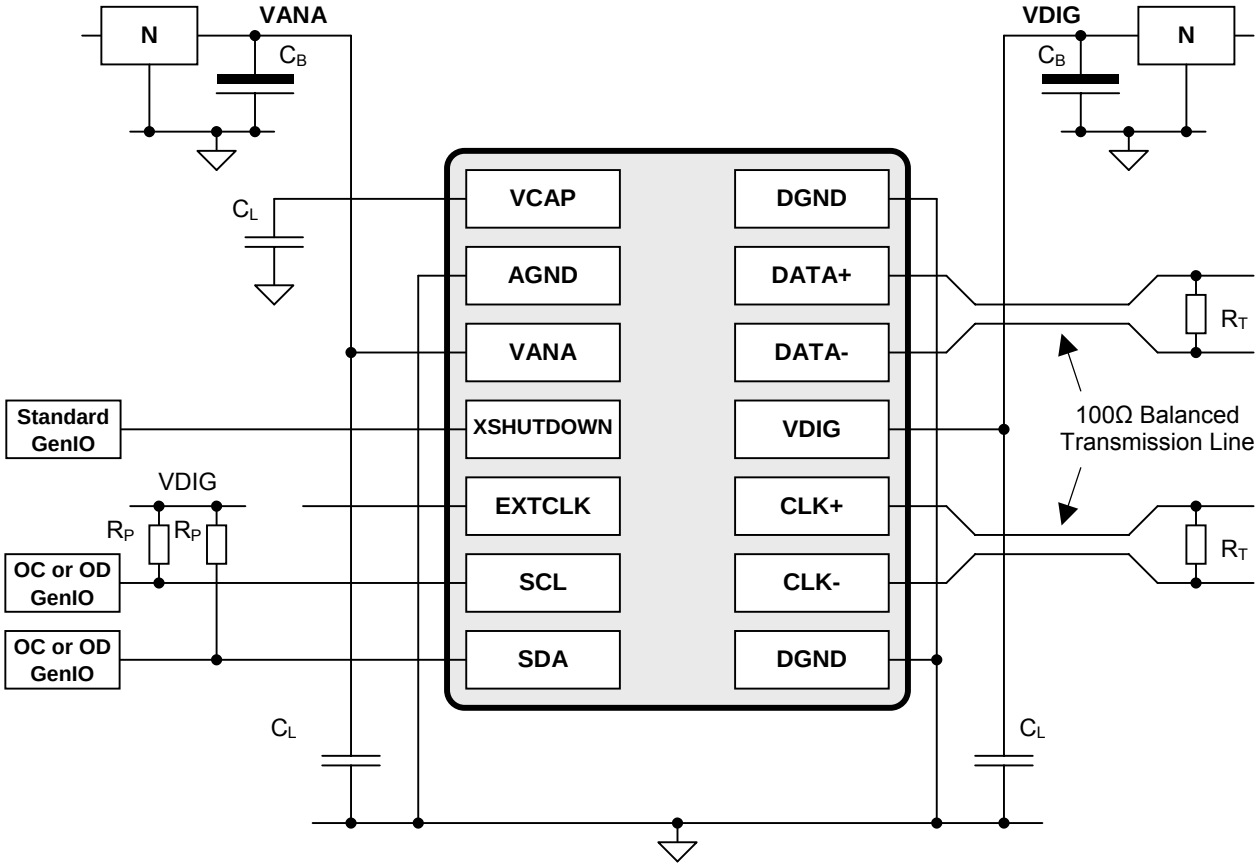


Figure 9: 14-Pin Standard Support Circuit

2.4.2 12-Pin Standard Support Circuit

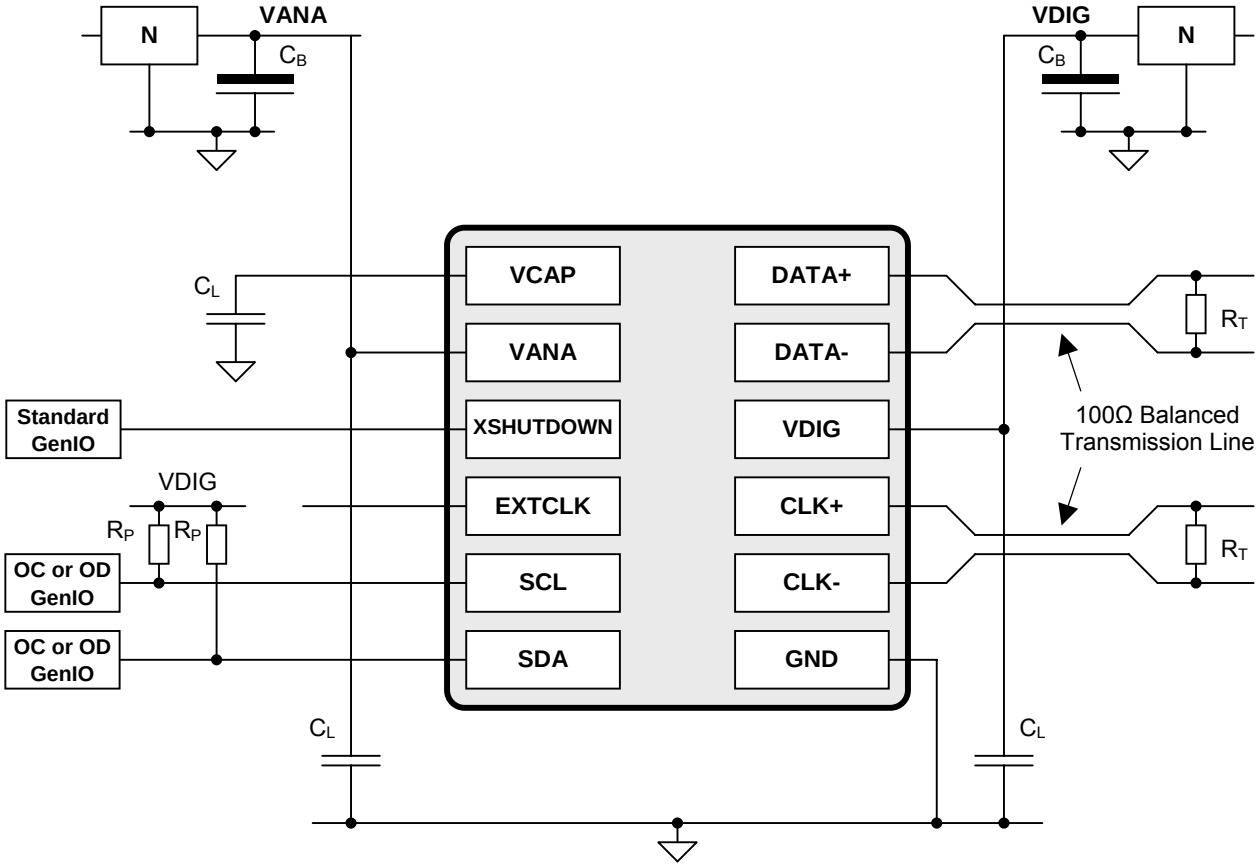


Figure 10: 12-Pin Standard Support Circuit

2.5 External Clock

The external reference clock is provided by the host system.

The external clock can either be either a DC coupled square wave, or an AC coupled sine wave. In either case, the clock may have been RC filtered

The external clock may be either a free-running system clock or a dedicated sensor module clock, which can enabled and disabled by the host.

The maximum input voltage of the external clock is 2.9V and nominal supply voltage for the sensor digital pads is 1.8V.

The ESD protection circuitry of the sensor modules clock input pad must not clip or distort the input clock shape even when XSHUTDOWN is low, the sensor module is powered down and/or the modules power supplies are not present.

The 4 possible clock configurations are shown in Figure 11.

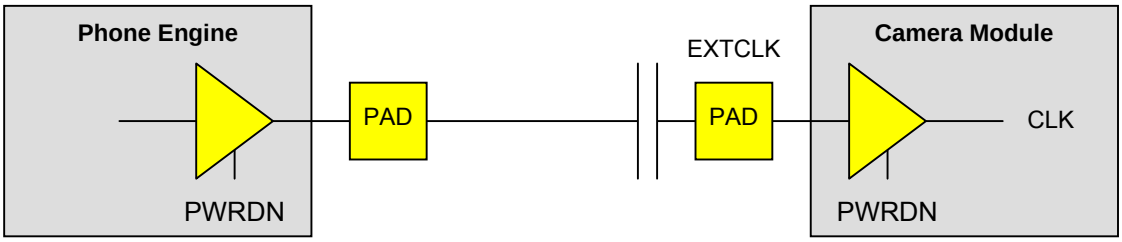
EXTCLK	Range			Unit
	Min	Typical	Max	
DC coupled Square Wave	1.0	1.8	2.9V	V
AC coupled Sine Wave	0.5	1	1.2	V p-p
Frequency	6.0		27.0	MHz
Duty Cycle	45		55	%
Input Leakage	-10		10	µA

Table 9: EXTCLK input clock

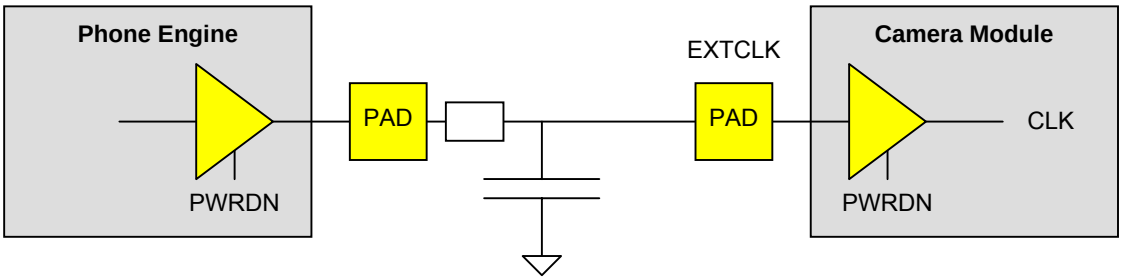
A PLL (Phase Locked Loop) block is embedded with the sensor module. The PLL generates all the necessary internal clocks from the external clock input.

Option 1: DC Coupled

Option 2: AC Coupled



Option 3: DC Coupled and filtered



Option 4: AC Coupled and filtered

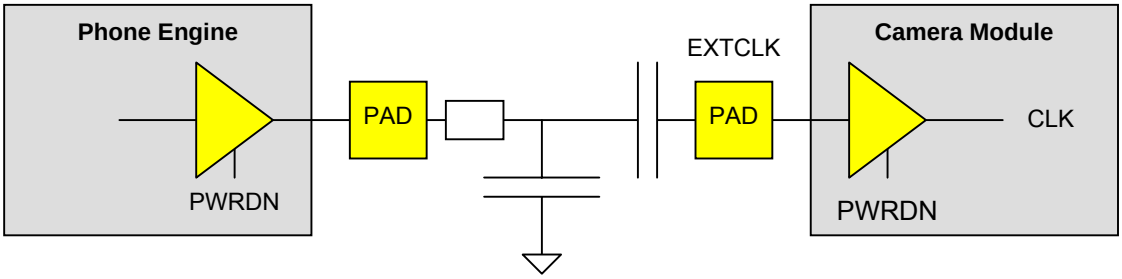


Figure 11: Clock Options

2.6 XSHUTDOWN

XSHUTDOWN is an asynchronous system reset signal. It is active low.

The maximum input voltage of the XSHUTDOWN signal is 2.9V and nominal supply voltage for the sensor digital pads is 1.8V.

The ESD protection circuitry of the sensor modules XSHUTDOWN pad must not clip the input voltage level even when the sensor module is powered down and/or the modules power supplies are not present.

Constraint	Label	Min	Typical	Max	Units
Low Level Input Voltage	V_{IL}			0.3 VDIG	V
High Level Input Voltage	V_{IH}	0.7 VDIG		2.9	V
Low Level Input Current (V_{IN} = GND)	I_{IL}	-10		10	µA
High Level Input Current (V_{IN} = V_{DIG})	I_{IH}	-10		10	µA

Table 10: XSHUTDOWN Specifications

2.7 CCI

The sensor control interface must comply with the CCI specification contained in the CCP2 specification with the following constraints:

The maximum input voltage of the SCL and SDA signals is 2.9V and nominal supply voltage for the sensor digital pads is 1.8V.

The ESD protection circuitry of the sensor modules SCL and SDA pads must not clip the input voltage level even when XSHUTDOWN is low, the sensor module is powered down and/or the modules power supplies are not present.

2.8 Absolute Maximum Ratings

Symbol	Description	Min	Typical	Max	Units
$V_{DIG(MAX)}$	Digital Absolute Max (1)	-0.3		2.2	V
$V_{ANA(MAX)}$	Analogue Absolute Max (2)	-0.3		3.2	V
$V_{IP(DIG)}$	Digital Input Voltages (3)	-0.3		V_{ANA}+0.3	V
VCAP	VCAP Analogue Voltage	-0.3		4.2	V
T_{STR}	Storage Temperature	-40		85	°C

Table 11: Absolute Maximum Rating

Notes:

1. Digital Supply 1V9 + 0.3 V
2. Analogue Supply 2V9+0.3V
3. Digital Inputs: EXTCLK, XSHUTDOWN, SCL, SDA

2.9 Operating Conditions

Symbol	Description	Min	Typical	Max	Units
V_{DIG}	Digital Power Supply (1)	1.7	1.8	1.9	V
V_{ANA}	Analogue Power Supply (2)	2.4	2.8	2.9	V
$V_{IP(DIG)}$	Digital Input Voltages (3)	0		V_{ANA}	V
VCAP	VCAP Analogue Voltage	0		4.2	V
T_{TEST}	Test Temperature (4)	21	23	25	°C
T_{OPT}	Optimum Operating Temperature (5)	5		30	°C
T_{OPR}	Normal Operating Temperature (6)	-25		55	°C
T_{FUNC}	Functional Operating Temperature (7)	-30		70	°C

Table 12:Operating Conditions

Notes:

1. Digital Supply tolerances: 1V8 +/- 100mV
2. Analogue Supply Tolerances: Lower limit 2V5-100mV, Upper Limit: 2V8+100mV
3. Digital Inputs: EXTCLK, XSHUTDOWN, SCL, SDA
4. Test Temperature – image quality test conditions
5. Optimum Operating Temperature – no visible degradation in image quality
6. Normal Operating Temperature – camera produces acceptable images
7. Functional Operating Temperature – camera fully functional

2.10 Average Current Consumption

The average current consumption of the sensor module is defined as the average current over 1 frame of streaming video at the sensor module's fastest readout rate.

Note: Peak currents are included in the average current consumption figure.

Measurement conditions:

- Standard Support Circuit
- Worst-case supplies
- Worst-case process
- Worst case temperature.

Mode	Format	Max Current Profiles 0 & 1	Max Current Profile 2	Units
Hardware Standby	-	10 (1)		µA
Software Standby	-	50 (2) 500 (3)		µA µA
Streaming (4)	VGA SVGA 1MP UXGA	20 22 28 40	22 26 38 68	mA mA mA mA

Table 13: Maximum Average Digital Supply Current Consumption vs. Operating Mode vs. Resolution

Notes:

1. At 25°C – external clock active or external clock not switching
2. Software Standby mode with external clock not switching
3. Software Standby mode with external clock active (External Clock Frequency = 6MHz)
4. Streaming Conditions: Readout of the full raw Bayer image at the maximum frame rate.

Mode	Max Current All Profiles		Units
Hardware Standby	5 (1)		µA
Software Standby	50 (2)		µA
Streaming (3)	VGA SVGA 1MP UXGA	15 18 26 40	mA mA mA mA

Table 14: Maximum Average Analogue Supply Current Consumption vs. Operating Mode vs. Resolution

Notes:

1. At 25°C – external clock active or external clock not switching
2. Software Standby mode with external clock either not switching or external clock active

3. Streaming Conditions: Readout of the full raw Bayer image at the maximum frame rate.

2.11 Average Power Consumption

In addition to specifications for the digital and analogue supplies current consumption there is a specification for the overall power consumption (Figure 12).

Measurement conditions:

- Standard Support Circuit
- Worst-case supplies
- Worst-case process
- Worst case temperature.

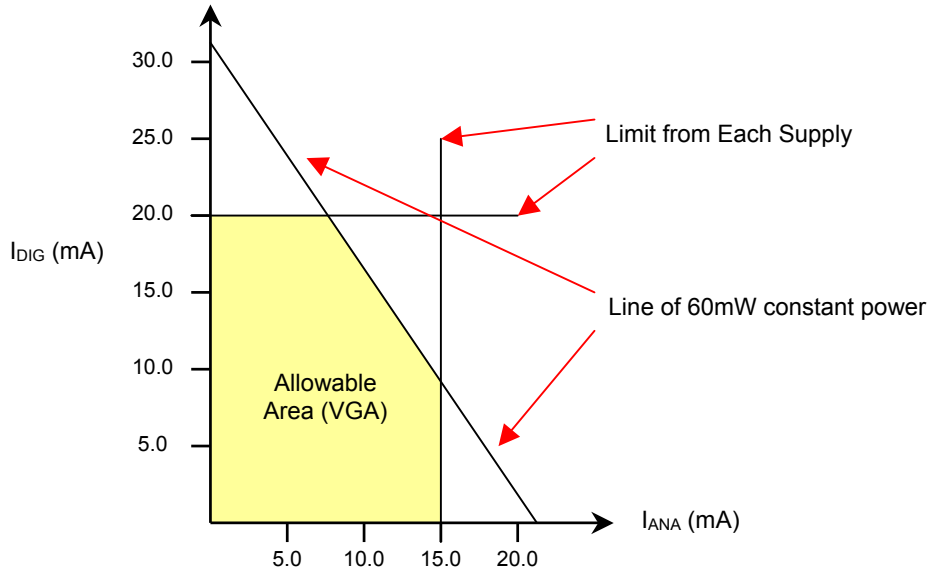


Figure 12: VGA Average Power Consumption Example

Mode	Format	Max Power Profiles 0 & 1	Max Power Profile 2	Units
Streaming (1)	VGA	60	65	mW
	SVGA	75	85	mW
	1MP	110	130	mW
	UXGA	175	235	mW

Table 15: Maximum Average Power Consumption vs. Operating Mode vs. Resolution

Notes:

1. Streaming Conditions: Readout of the full raw Bayer image at the maximum frame rate.

2.12 Peak Current Consumption

The peak current consumption of the sensor module is defined as any current pulse of greater than or equal to 10µs.

Measurement conditions:

- Standard Support Circuit
- Worst-case supplies
- Worst-case process
- Worst case temperature.

The peak current must be less than 1.333 times the maximum average current specification for that operating mode.

Maximum duty cycle of the high part to the low part of the current waveform is 33% with a worst-case period of 500µs

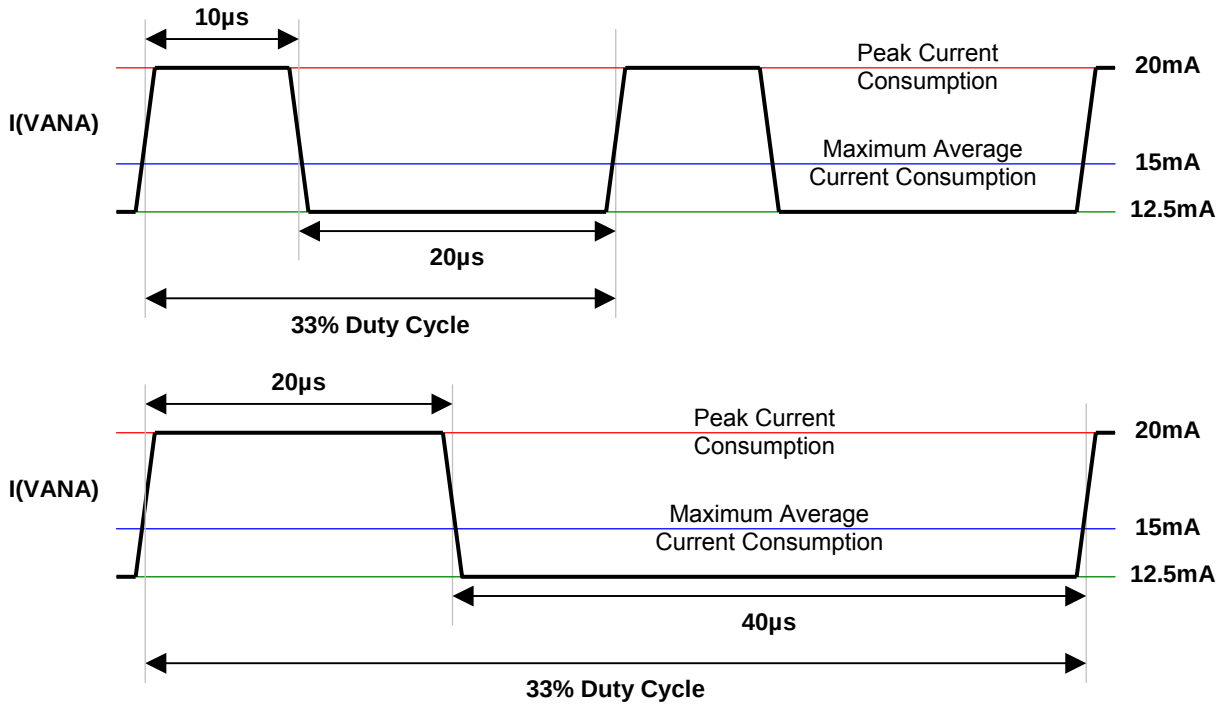


Figure 13: VGA Peak Current Consumption Example 1

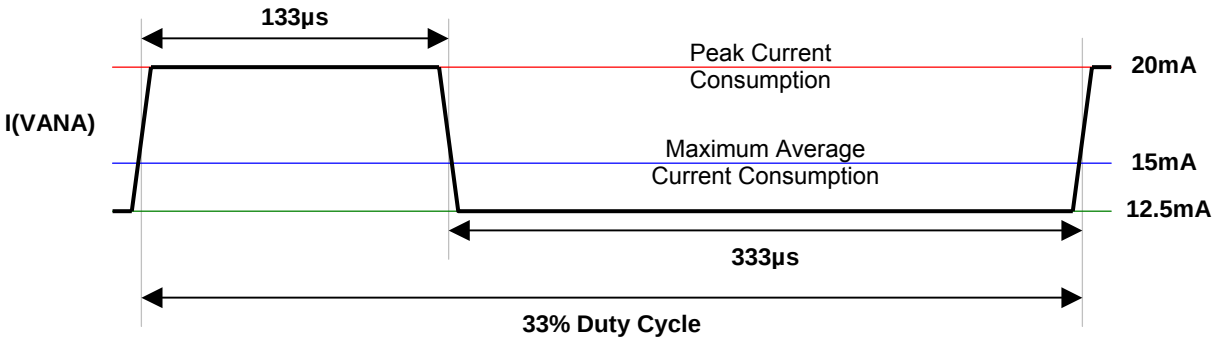


Figure 14: VGA Peak Current Consumption Example 2

2.13 Inrush Current

After power on of any rail or a mode change, the current drawn must not exceed the peak current specification of the mode that has the higher current specification.

The duration of the overshoot must not exceed 10µs

Measurement conditions:

- Standard Support Circuit
- Worst-case supplies
- Worst-case process
- Worst case temperature.

3 Operating Modes

3.1 Introduction

The sensor module has 4 operating modes (Table 16).

Power State	Description
Power-off	Power supplies are turned off.
Hardware Standby	No communication with the sensor is possible.
Software Standby	CCI communication with sensor is possible.
Streaming	The sensor module is fully powered and is streaming image data on the CCP2 bus.

Table 16: Operating Mode Summary

The host should configure and control the sensor module using only the above 4 operating modes.

Moving from one mode to another is achieved by issuing the appropriate mode command via the CCI serial control interface, the XSHUTDOWN signal changing state and the power supplies.

The sensor module also has “Soft-Reset” CCI command that resets of the registers back to their defaults and puts the sensor module into the software standby mode.

Figure 15 defines the valid mode changes for the sensor module.

A manufacturer can define additional operating modes, but the sensor module must be able to configured and controlled by a host without the using these extra operating modes.

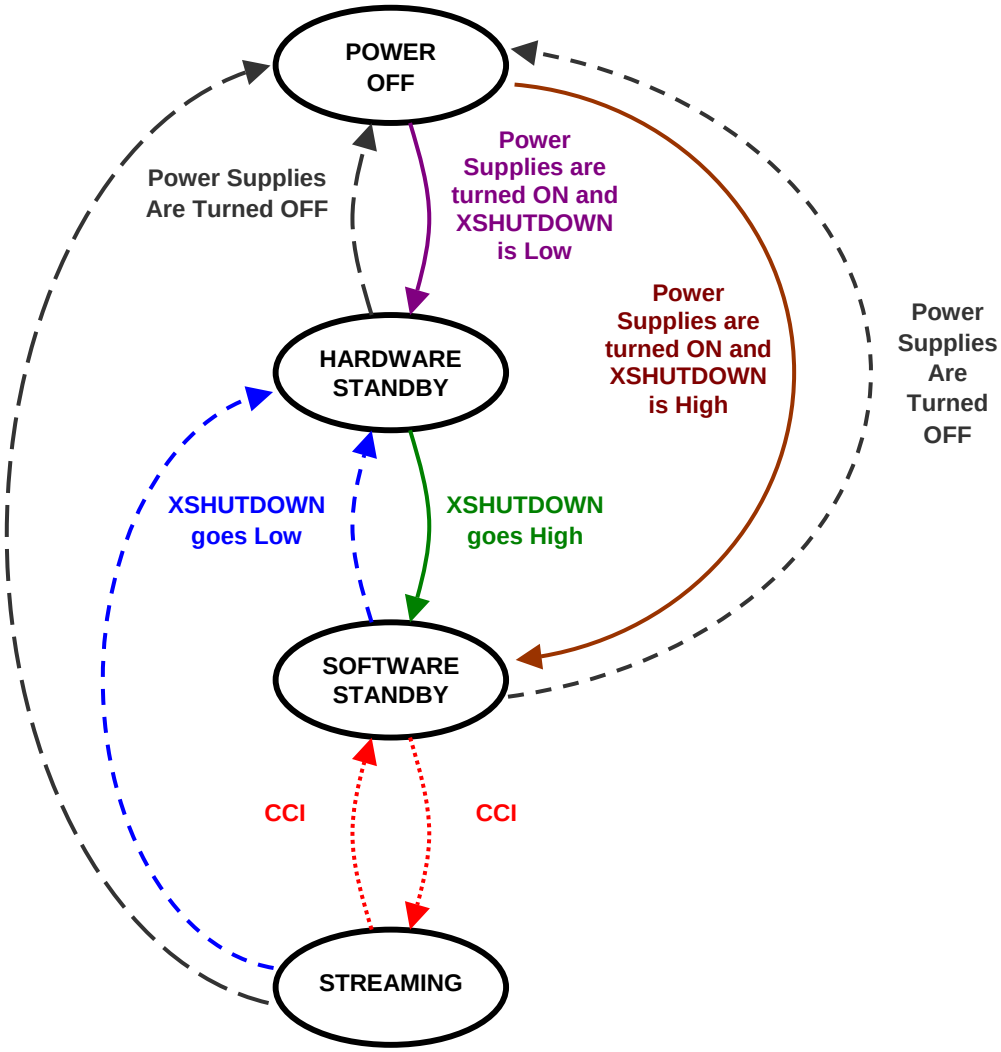


Figure 15: System State Diagram

3.2 Power Up Sequence

The digital and analogue supply voltages can be powered up in any order e.g. VDIG then VANA or VANA then VDIG.

On power up:

- If XSHUTDOWN is low when the power supplies are brought up then the sensor module will go into hardware standby mode.
- If XSHUTDOWN is high when the power supplies are brought up then the sensor module will go into software standby mode

In both cases the presence of an on-chip power-on reset cell ensures that the CCI register values are initialised correctly to their default values.

The EXTCLK clock can either be initially low and then enabled during software standby mode or EXTCLK can be a free running clock.

Constraint	Label	Min	Max	Units
VANA rising – VDIG rising	t0	VANA and VDIG may rise in any order.		ns
VDIG rising – VANA rising	t1	The rising separation can vary from 0ns to Indefinite		ns
VANA rising – XSHUTDOWN rising	t2	0.0		ns
XSHUTDOWN rising – First I^2C transaction	t3	15µs + 16 EXTCLK Cycles		
Minimum no of EXTCLK cycles prior to the first I^2C transaction	t4	16		EXTCLK cycles
PLL Start up/Lock Time	t5		1	ms
Entering Streaming Mode – First Frame Start Sequence (Fixed part)	t6		10	ms
Entering Streaming Mode – First Frame Start Sequence (Variable part)	t7	The delay is the coarse integration time value		lines

Table 17: Power-Up Sequence Timing Constraints

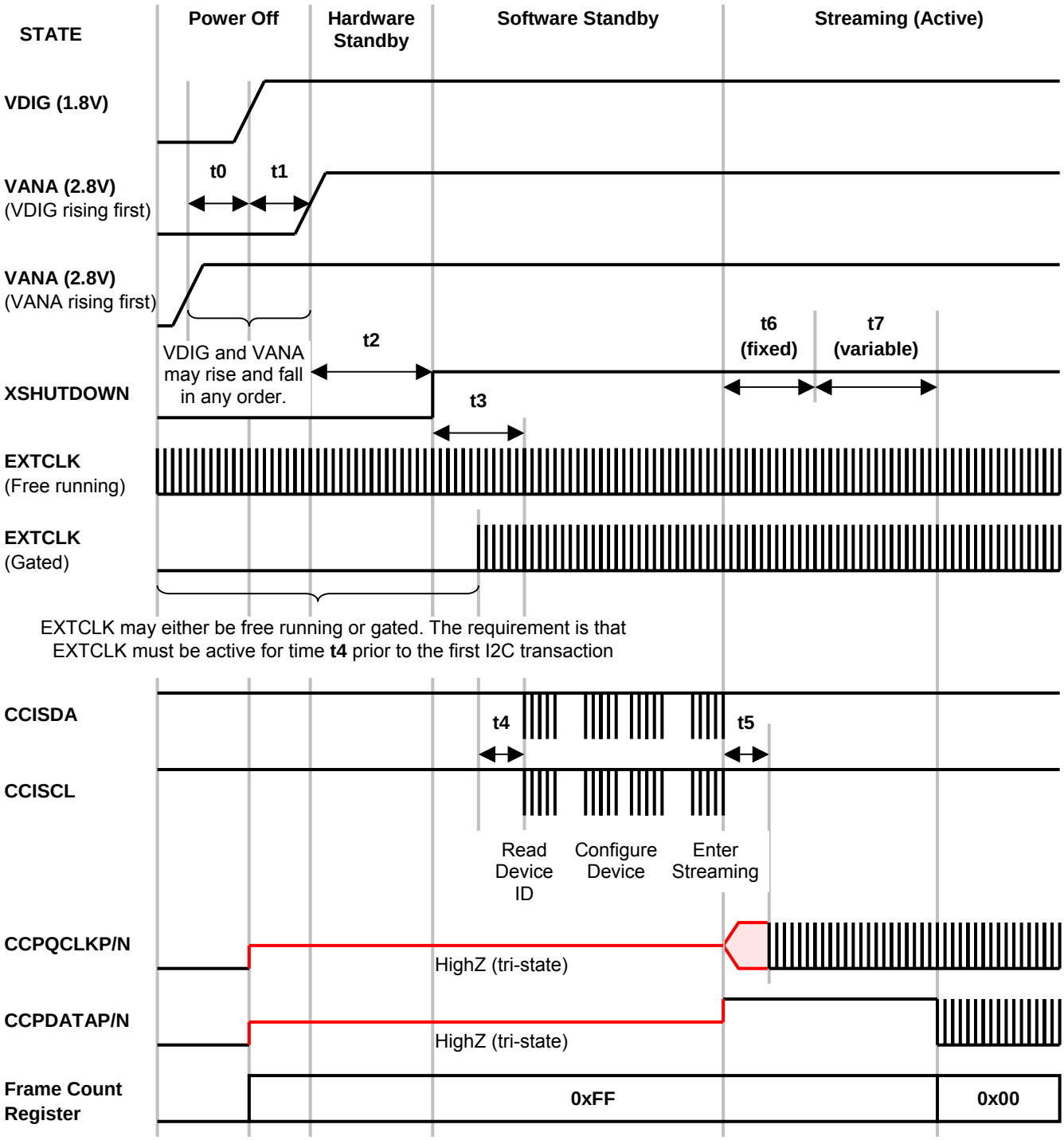


Figure 16: Power Up Sequence

3.3 Power Down Sequence

The digital and analogue supply voltages can be powered down in any order e.g. VDIG then VANA or VANA then VDIG.

Similarly to the power-up sequence the EXTCLK: input clock may be either gated or continuous.

If the CCI command to exit streaming is received while a frame of CCP2 data is being output then the sensor module must wait to the CCP2 frame end code before entering software standby mode.

If the CCI command to exit streaming mode is received during the inter frame time then the sensor module must enter software standby mode immediately.

Constraint	Label	Min	Max	Units
Enter Software Standby CCI command – Device in Software Standby mode	t0	If outputting a frame of CCP2 data wait to CCP2 frame end code before entering software standby, otherwise enter software standby mode immediately		
Minimum no of EXTCLK cycles after the last I^2C transaction or CCP2 frame end	t1	512		EXTCLK cycles
Last I^2C Transaction or CCP2 frame end – XSHUTDOWN falling	t2	512		EXTCLK cycles
XSHUTDOWN falling – VANA falling	t3	0.0		ns
VANA falling – VDIG falling	t4	VANA and VDIG may fall in any order. The falling separation can vary from 0ns to Indefinite		ns
VDIG falling – VANA falling	t5			ns

Table 18: Power Down Timing Constraints

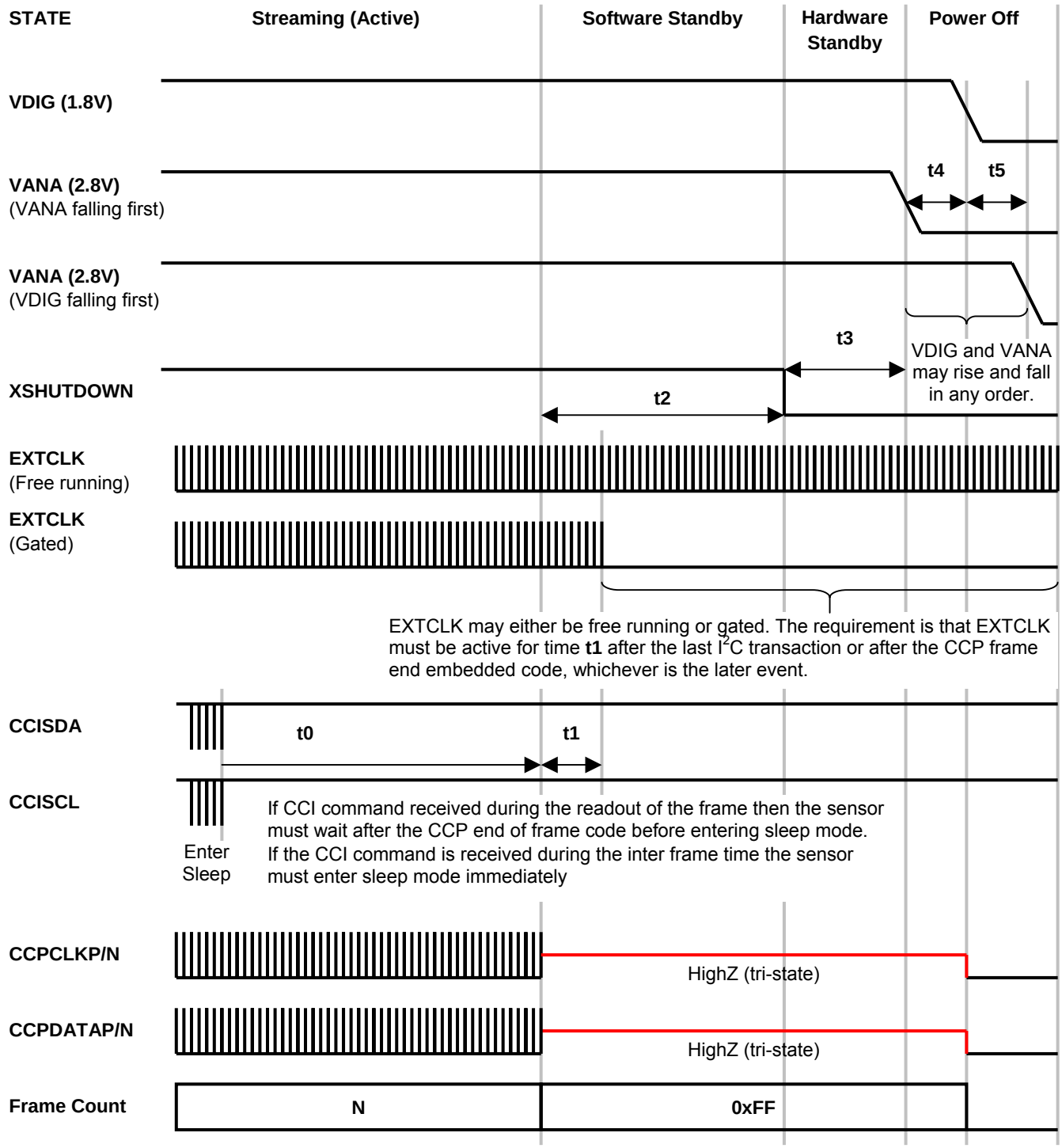


Figure 17: Power Down Sequence

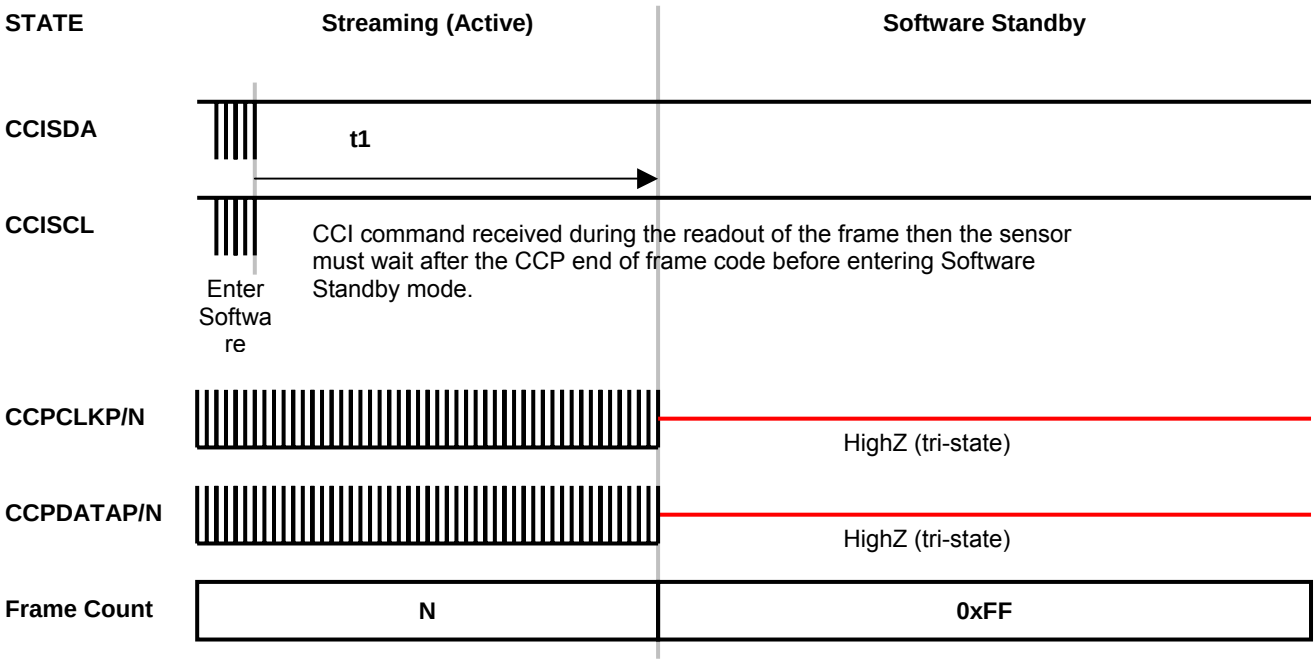


Figure 18: Exit from Streaming Mode when the CCI command is received during the output of a frame of CCP2 data

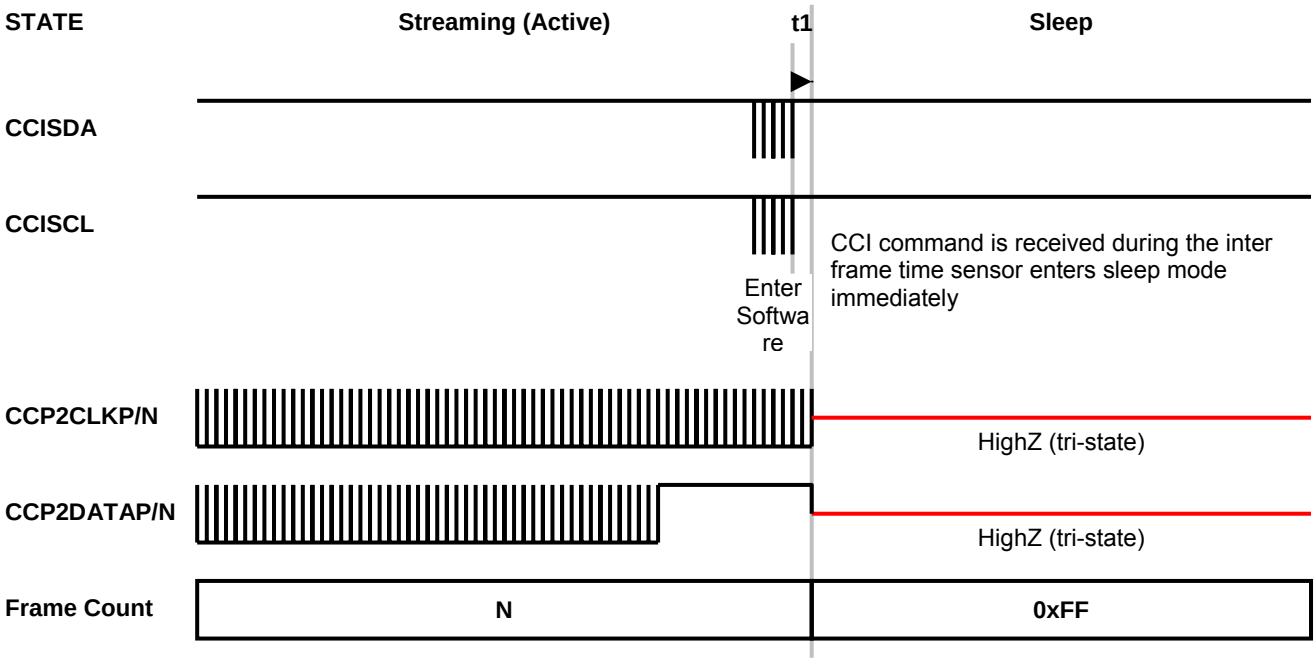


Figure 19: Exit from Streaming Mode when the CCI command is received during the inter frame period

3.4 Internal Power-on Reset (POR)

The sensor module should internally perform a power-on reset (POR) when the digital supply rises above a trigger level, V_{trig_rising}
Similarly is the digital power supply falls below the trigger level, $V_{trig\_falling}$, then the power-on reset should activate.

The host must be able to reset the sensor by turning the power supplies on and off.

Constraint	Label	Min	Typical	Max	Units
VDIG rising crossing V_{trig_rising} – Internal reset being released	t1	7	10	15	µs
VDIG falling crossing $V_{trig\_falling}$ - Internal reset active	t2		0.5	1	µs
Minimum VDIG spike width below $V_{trig\_falling}$ which is considered to be a reset when POR cell output high	t3		0.5		µs
Minimum VDIG spike width below $V_{trig\_falling}$ which is considered to be a reset when POR cell output low	t4		1.0		µs
Minimum VDIG spike width above V_{trig_rising} which is considered to be a supply is stable when POR cell output low While the POR cell output is low all VDIG spikes above V_{trig_rising} which are less than t5 must be ignored	t5		50		ns
VDIG rising trigger voltage	V_{trig_rising}	1.15	1.4	1.55	V
VDIG falling trigger voltage	$V_{trig\_falling}$	1.00	1.25	1.45	V

Table 19: Internal Power on Reset Cell Specifications

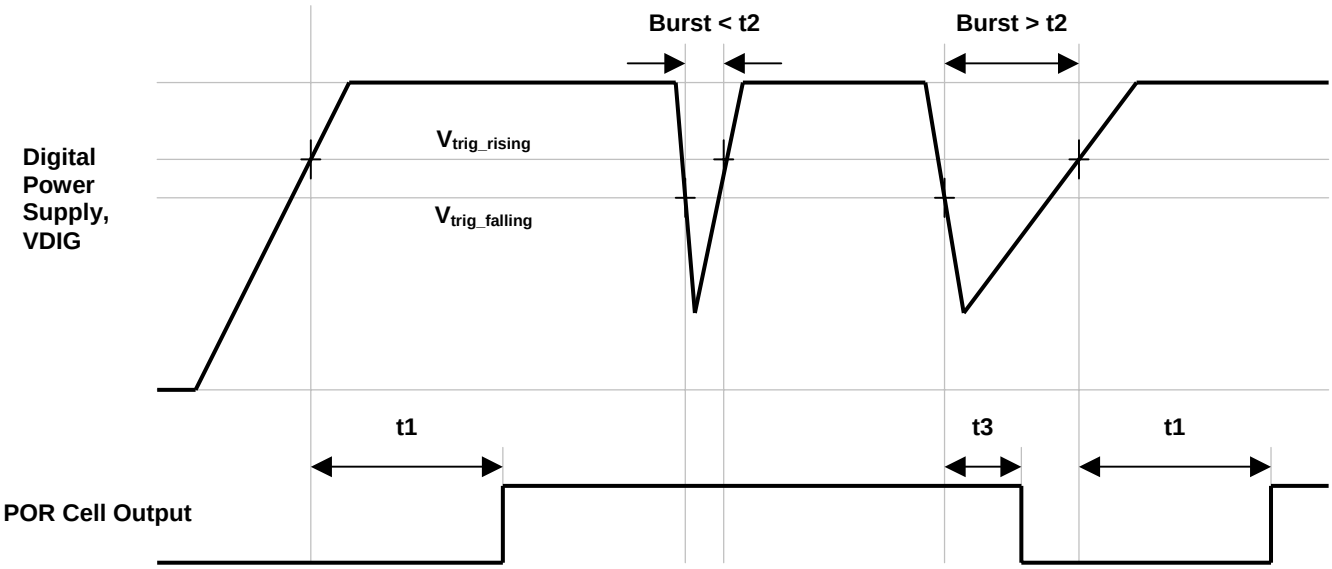


Figure 20: Power On Reset Power up Timing

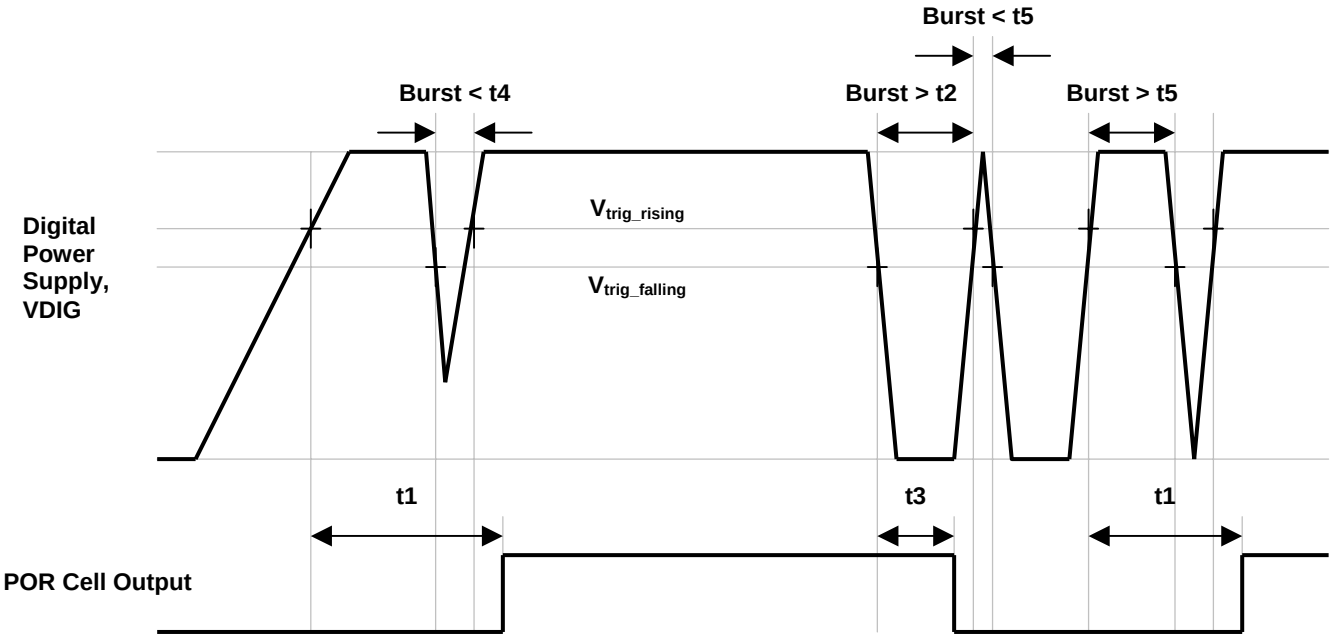


Figure 21: Power On Reset Supply "Glitch/Brown-out" Timing

3.5 Software-Reset Via CCI Interface

Setting the `software_reset` register value to 1 resets all of the serial interface registers to their default values.

The value of the `software_reset` register is also reset.

Register Name	Type	RW	Comment
`software_reset`	8-bit unsigned integer	RW	Setting this register to 1 resets the sensor to its power up defaults. The value of this bit is also reset 0: Off 1: On

Table 20: Soft Reset Register (Read/Write)

This “Soft Reset” puts the sensor module into Software Standby mode and then power-up sequence detailed earlier must be followed to generate streaming image data for the sensor module.

The purpose of this bit is to allow a CCI command to reset the sensor module to a known state, if the host system loses track of the state of the sensor module.

Constraint	Label	Min	Typical	Max	Units
Maximum time from the stop condition of CCI soft reset command – Reset of the sensor	t1			50	µs
Minimum time between successive CCI soft reset commands	t2			100	µs

Table 21: Soft Reset Specifications

3.6 XSHUTDOWN

XSHUTDOWN is an asynchronous system reset signal. It is active low.

If XSHUTDOWN goes low in any mode other than the power off state the sensor module must immediately move in to hardware standby mode. When XSHUTDOWN goes high and the power supplies are present the sensor moves into software standby mode.

XSHUTDOWN low resets all of the CCI registers to their default values.

Constraint	Label	Min	Typical	Max	Units
Minimum XSHUTDOWN low period to be considered to be a reset	t1	100			ns
Minimum time between successive XSHUTDOWN reset pulses	t2			100	µs

Table 22: XSHUTDOWN Specifications

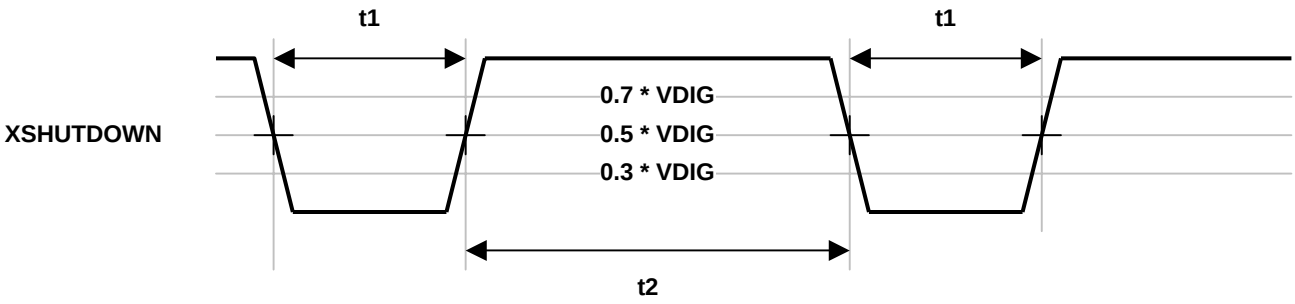


Figure 22: XSHUTDOWN Timing Specification

3.7 Mode Select Register

The operating mode of the sensor module is controlled via the `mode_select` register (Table 23).

- Mode 0 is software standby
- Mode 1 is streaming
- Modes 2 to 15 are reserved for future use.
- Modes 16 to 31 are manufacturer specific modes.

Register Name	Type	RW	Comment
`mode_select`	8-bit unsigned integer	RW	0: Software Standby (CCI communications active) 1: Streaming (Active Video) 2-15: Reserved 16 –31: Manufacturer Specific Modes

Table 23: Mode Select Register (Read/Write)

3.8 Frame Count Register

The `frame_count` register increments by 1 at the start of each frame.

When 255 is reached the counter rolls over to 0 and starts incrementing again

During Software Standby modes the frame counter is reset to 255 (FF_H)

Register Name	Type	RW	Comment
`frame_count`	8-bit unsigned integer	RO Dynamic	Increments by 1 at the start of each frame

Table 24: Frame Counter Register

3.9 Power-Off

The power-off state is defined as either or both of the digital and analogue supplies not present.

3.10 Hardware Standby

The hardware standby mode is defined as both the digital and analogue supplies present and the XSHUTDOWN signal low.

This mode is the lowest power consumption possible.

Requirements:

- No communication is possible over the CCI serial interface.
- The mode is entered asynchronously by forcing a low level on XSHUTDOWN pin.
- The video timing logic is then reset and the CCI serial interface is reset to its defaults. The clock input pad, PLL and the video blocks are powered down.
- SDA, SCL lines are not driven so that other devices are able to use the I2C or CCI bus.
- The CCP2 data and clock pads are high impedance.

3.11 Software Standby Mode

The software standby mode is defined as both the digital and analogue supplies present and the XSHUTDOWN signal high.

This mode maintains the minimum logic required to preserve the existing set-up and to communicate with the sensor using the CCI serial interface.

Requirements:

- The EXTCLK clock must be active for CCI communications with the sensor module.
- All read/write CCI registers must be able to read from or written to in software standby mode.
- All read only CCI registers must be able to read from in software standby mode.
- CCI commands must be able to be received at **any** point in time.
- The values of the serial interface registers, e.g. exposure and gain, are preserved.
- The CCP2 data and clock pads are high impedance.

During the power-up sequence then the host system or co-processor should read the sensor module's device identifier registers (Table 25)

From the values contained in the device identifier registers the host system can determine the model number, revision, manufacturer and SMIA version of the sensor module.

The clock divider and PLL multiplier registers must be configured during Software Standby mode while the PLL is powered down.

Register Name	Type	RW	Comment
model_id	16-bit unsigned integer	RO Static	Model Identification Number
revision_number	8-bit unsigned integer	RO Static	Device Revision Identifier
manufacturer_id	8-bit unsigned integer	RO Static	Manufacturers Code Please refer the SMIA Manufacturer Codes Document
smia_version	8-bit unsigned integer	RO Static	SMIA Version 9 – Version 0.9 10 – Version 1.0 11 – Version 1.1

Table 25: Device Identifier Registers

3.12 Streaming

In this mode the sensor module streams live video to the host system/co-processor.

It is mandatory that when a CCI command is received to exit the streaming mode the sensor module completes the current frame of CCP2 data before entering the software standby mode.

If the software standby mode command is received during the frame blanking time then the mode change is immediate.

CCI commands must be able to be received at **any** point in time.

This is the highest power consumption mode.

The sensor module PLL (phase locked loop) via the CCP2 Clock provides the system clock for co-processor devices.

4 Data Format

4.1 Introduction

Requirements for the image data interface

- The imager must be compliant with the CCP2 specification.
- All sensor profile levels must support the RAW10 data format.
- Sensor profile levels 1 and 2 must also support the RAW8 data format

Two types of 8-bit data must be supported.

- The 8-bit data is top 8-bits of the internal 10-bit data.
- The 8-bit data is the output of the 10-bit to 8-bit DPCM/PCM compression method.
- The continuous data qualification clock/strobe CCP2 option must be implemented.
- The CCP2 data stream must contain the checksum at the end of each line.
- For data rates less than 208Mbps, there must be the option to use CCP data/clock signalling instead of CCP2 data/strobe signalling, to provide backward compatibility with the CCP legacy receivers

4.2 Channel Identifier

The `CCP2_channel_identifier` register allows the DMA channel identifier within the CCP2 embedded synchronization codes to be programmed.

The default value for this register is 0x00 for backward compatibility with older CCP2 receivers.

Register Name	Type	RW	Comment
`CCP2_channel_identifier`	8-bit unsigned integer	RW	Valid Range: 0-7 Default Value: 0

Table 26: CCP2 Channel Identifier Register

4.3 Signalling Options

The CCP2 specification supports two signalling options for the image data interface

- Data/Clock Signalling: Positive edge of the data qualification clock qualifies the data
- Data/Strobe Signalling: The data qualification clock becomes a strobe signal which toggles state whenever the data does not change

The `CCP2_signalling_mode` register controls the signalling method used.

Register Name	Type	RW	Comment
`CCP2_signalling_mode`	8-bit unsigned integer	RW	0 – Data/Clock Signalling 1 – Data/Strobe Signalling

Table 27: CCP2 Signalling Mode Register

4.4 Output Format

The `CCP_data_format` register controls the output data format and the level of compression used.

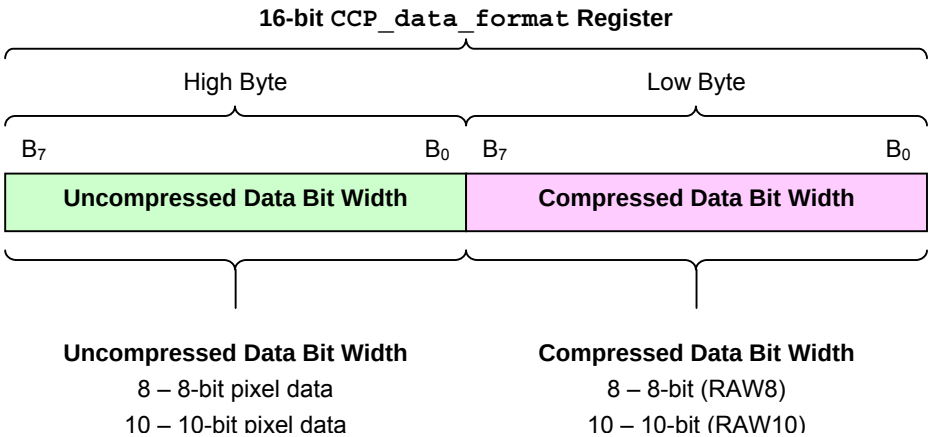


Figure 23: CCP Data Format Register

The MS byte of the `CCP_data_format` register contains the bit width of the uncompressed pixel data.
The LS byte of the `CCP_data_format` register contains the bit width of the compressed pixel data.

If the MS and LS bytes have the same value then the image data has not been compressed.

Examples

- MS byte = 8, LS byte = 8 – Top 8-bits of internal pixel data transmitted as RAW8
- MS byte = 10, LS byte = 8 – 10-bit pixel data compressed to 8-bits, transmitted as RAW8
- MS byte = 10, LS byte = 10 – Top 10-bits of internal pixel data transmitted as RAW10

Register Name	Type	RW	Comment
`ccp_data_format`	16-bit unsigned integer	RW	0x0808 – RAW8 (top 8-bits of internal data) 0x0A06 – RAW6 + 10-b to 6-b compression 0x0A07 – RAW7 + 10-b to 7-b compression 0x0A08 – RAW8 + 10-b to 8-b compression 0x0A0A – RAW10 (top 10-bits of internal data)

Table 28: CCP Data Format Register

Notes:

1. For Profile 0 – RAW10 is mandatory
2. For Profile 1 – RAW8, both top 8-bits and 10-b to 8-b compression, and RAW10 are mandatory
3. The RAW6 and RAW7 compressed readout modes are optional.

All data format mode changes must be performed in the software standby mode.

4.5 Data Format Description

The data format description lists the output data formats a sensor module supports. By reading this information the host will have a fully understanding of the data format capabilities of that sensor module, whether is RAW10 only or RAW10, RAW8 and RAW8 with 10-b to 8-b compression.

This description is made up of a data format model type byte, a data format model sub-type byte, followed by list of 2-byte descriptors (Table 29). The format of a descriptor is the same as for the CCP_data_format register (Figure 23).

- The data_format_model_type register value is always 0x01.
- The data_format_model_subtype register contains the number of descriptors used
- The minimum number of data format descriptors is 1.
- The maximum number of data format descriptors is 7.

In sensor module is required to automatically pad lines with dummy pixels to ensure that the number of pixels between the line start sync code for line m and the line end sync code for line m is a multiple of the appropriate number of pixels for the RAW format used (e.g. 4 pixels for RAW8, 16 pixels for RAW10)

The number of dummy pixels must be described by one of the frame format descriptors.

Figure 24 and Figure 25 contain examples of the data format descriptor for profile level 0 and profile level 1 sensor modules.

Register Name	Type	RW	Comment
data_format_model_type	8-bit unsigned integer	RO	0x01: 2-Byte Data Format
data_format_model_subtype	8-bit unsigned integer	RO	Contains the number of data format descriptors used
data_format_descriptor_0	16-bit unsigned integer	RO	
data_format_descriptor_1	16-bit unsigned integer	RO	
data_format_descriptor_2	16-bit unsigned integer	RO	
data_format_descriptor_3	16-bit unsigned integer	RO	
data_format_descriptor_4	16-bit unsigned integer	RO	
data_format_descriptor_5	16-bit unsigned integer	RO	
data_format_descriptor_6	16-bit unsigned integer	RO	
data_format_descriptor_7	16-bit unsigned integer	RO	

Table 29: Data Format Description Registers

Number of Data Format Descriptors: 1

	Uncompressed Data Bit Width	Compressed Data Bit Width	
Descriptor 0	$0A_H$	$0A_H$	Top 10-bits of internal pixel data. Transmitted as RAW10

Figure 24: Data Format Description – Profile Level 0 Example

Number of Data Format Descriptors: 3

	Uncompressed Data Bit Width	Compressed Data Bit Width	
Descriptor 0	08_H	08_H	Top 8-bits of internal pixel data. Transmitted as RAW8
Descriptor 1	$0A_H$	08_H	Top 10-bits of internal pixel data compressed to 8-bits. Transmitted as RAW8
Descriptor 2	$0A_H$	$0A_H$	Top 10-bits of internal pixel data. Transmitted as RAW10

Figure 25: Data Format Description - Profile Level 1 Example

4.6 Data Encoding for Uncompressed Image Data

For the baseline profile (level 0) the output digital video data is 10-bits per sample.
For profile levels 1 and 2 the output digital video data may be either 8-bit or 10-bits per sample.

Video data in which the 8 most significant bits are all set to 0's are illegal data values. This is necessary to prevent the generation of false CCP embedded escape sequences from occurring within the video data. RAW10 false sync code protection in addition to limiting pixel data to 4-1023 after data packing every 5th byte (containing LSBs) has to be examined and force the bit 4 (P3 bit 0) '1' if the byte is all zeros.

Consequently only 255 of the possible 256 8-bit values or 1020 of the possible 1024 10-bit values may be used to express a pixel value.

Thus

- For 8-bit pixel data, codes 0 is illegal value
- For 10-bit pixel data, codes 0- 3 (inc) are illegal values.

Read-out Order	Progressive Scan (Non-interlaced)		
Form of Encoding	Uniformly Quantised, PCM, 8/10 bits per sample		
Correspondence between video signal levels and quantisation levels	**CCP2 Format**	**Pixel Data**	**Pixel Values**
	RAW8	8-bit	1 to 255
	RAW10	10-bit	4 to 1023

Table 30: Video Encoding Parameters

4.6.1 Data Pedestal

The data pedestal is the pixel value the sensor module produces when there is no light incident on the sensor module.

The sensor module must have an internal calibration function, which ensures that data pedestal value remains constant with integration time, gain, and temperature and between different sensors.

The host system should always use the `data_pedestal` register value to determine the sensor output black level.

Register Name	Type	RW	Comment
`data_pedestal`	16-bit unsigned integer	RO Static	

Table 31: Data Pedestal Register

System	Typical Data Pedestal
8-bit	16
10-bit	64

Table 32: Typical Data Pedestal Values

4.7 Frame Format Overview

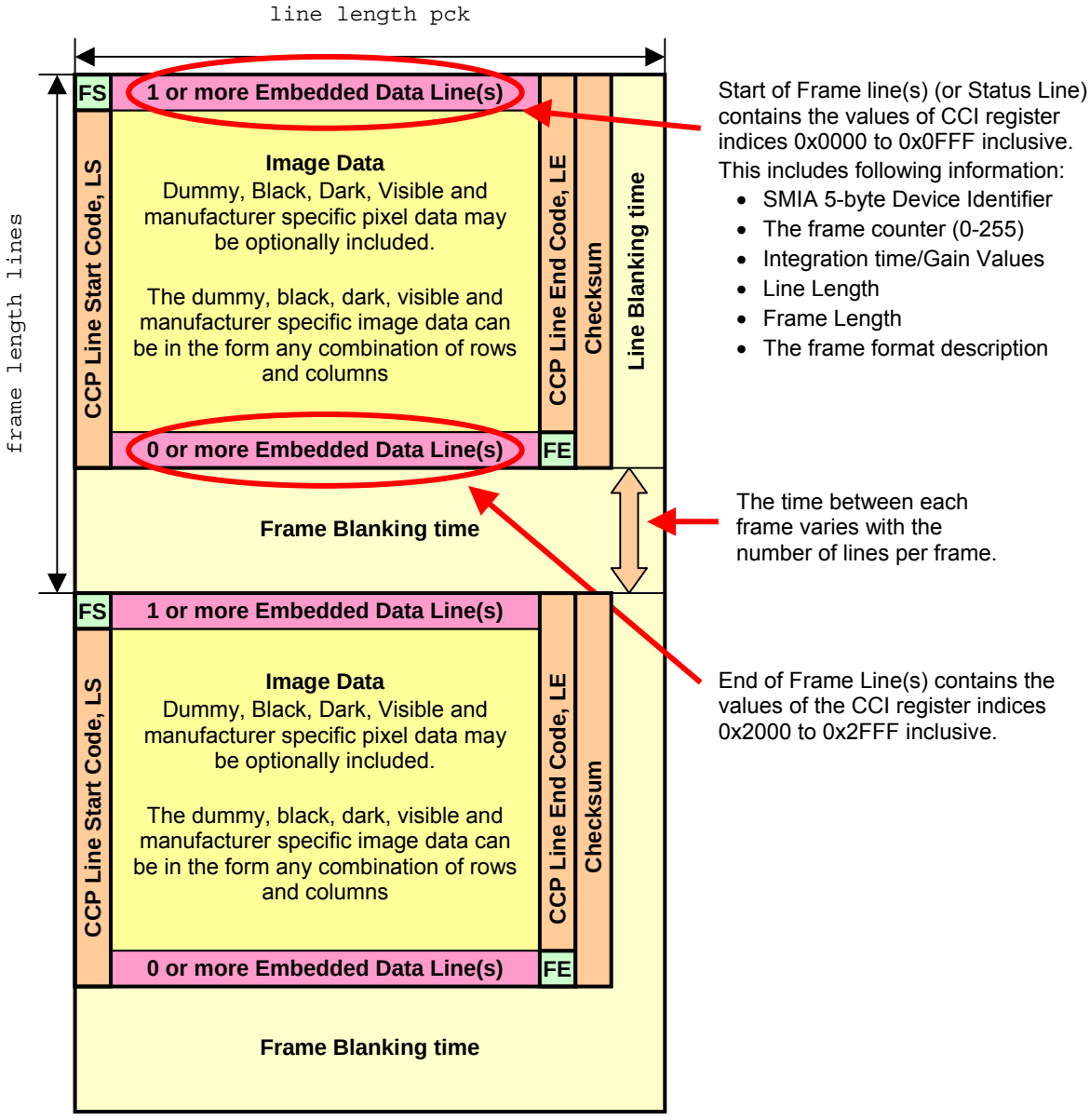


Figure 26 - Frame Format

A frame of CCP2 raw Bayer mage data has the following structure (Figure 26):

- 1 or more embedded data lines containing sensor module configuration data
- A block of image data which may be a mix of dummy, black, dark, visible and manufacturer specific pixel types.
- 0 or more embedded data lines containing sensor module image statistics data.

While the image data may be compressed via the 10-bit to 8-bit DPCM/PCM algorithm the embedded data is never compressed. Thus when the compression is used the frame of CCP2 data contains a mixture of uncompressed data – the embedded data lines – and compressed data – the image data.

The 1 or more embedded data lines at the beginning of the frame are termed Start of Frame Line(s) (SOF)

The start of frame line(s) or status line(s) must contain the values of the CCI registers that exist between indices 0x0000 to 0x0FFF. Additional manufacturer specific registers may also be included. The SOF line(s) includes the SMIA 5-byte device identifier, frame counter (0-255), the integration time and gain register values, line and frame length plus the frame format description.

The very first embedded data line at the start of a frame must contain the frame format description.

The presence of the SOF lines(s) enables the sensor set up and configuration data to be automatically extracted by either software or by a separate image-processing device. Thus minimising the CCI communication overhead.

The 0 or more blank lines at the end of the frame are termed End of Frame Line(s) (EOF)

The EOF lines(s) must contain the values of CCI registers that exist between indices 0x2000 to 0x2FFF. Additional manufacturer specific registers may also be included.

The EOF line(s) contain the statistics configuration data and the values of the statistics gathered.

The structure of the image data including the number of embedded data lines at the start and end of the field is fully described by the frame format description which is part of the data embedded within the first SOF line of the frame of CCP2 data.

The frame format description and the embedded data format are described later in Section 4.8 and Section 4.9 respectively.

4.7.1 Embedded Data Lines

The embedded data lines provide a mechanism to embedded non-image data such as sensor configuration details and image statistics values with a frame of CCP2 data.

The minimum requirement is to have 1 embedded data line at the start of the frame. If the number of CCI registers to be output is greater than the distance between the CCP2 embedded line start and line end codes allows then more embedded data lines can be added at either the start or the end of the frame as required.

The number of embedded data lines at the start and end of the frame is specified as part of the frame format description.

4.7.2 Dummy Pixel Data

This is invalid pixel data. The receiver should always ignore dummy pixel data.

4.7.3 Black Pixel Data (Zero Integration Time)

Black pixels contain pixel data with zero integration time and the same analogue gain value as visible pixels.

4.7.4 Dark Pixel Data (Light Shielded Pixels)

The dark pixels have the same integration time and analogue gain values as the visible pixels but are shielded from light by a metal layer.

The image sensor uses the data from these pixels to automatically calibrate the sensors output black level (data pedestal)

4.7.5 Visible Pixel Data

The visible pixels contain valid image data.
The correct integration time and analogue gain for the visible pixels is specified in the blank lines at the start of the frame.

4.7.6 Manufacturer Specific Pixel Data

To provide manufacturers a mechanism to include pixel data of types other than those already specified above, there are 7 manufacturer specific pixel (MSPs) data type codes available.

4.7.7 Frame Blanking Period

The user may choose to extend the frame-blanking period by increasing the frame length by writing to CCI registers. In this event, the appropriate additional padding is inserted between the Frame End (FE) code of frame “n” and the Frame Start (FS) code of frame “n+1”. This means that the distance between the Frame Start Code of frame “n” and the Frame End Code of frame “n+1” will remain constant.

4.7.8 Line Blanking Period

The user can extend the line length by writing to serial registers. The line length padding is inserted after the LE sequence. The distance between the LS and LE sequences will remain constant.

4.7.9 Simple Profile Receiver Behaviour

The recommended behaviour for a simple profile receiver is to only extract the embedded and visible data from the frame of CCP2 data. Any other type of pixel data present in the frame of CCP2 data can be ignored i.e. treated as dummy pixel data.

4.8 Frame Format Description

The frame format description registers fully describe the structure of the frame of CCP2 image data including the number of embedded data lines at the start and the end of the frame.

The objective of this information is to enable either software or a dedicated image processing device to be able to process arbitrary sized images without any prior knowledge of the image sensors image format.

The frame format model information is available to either software or a dedicated image-processing device via both CCI Registers and the embedded data line at the start of the frame.

The visible pixel frame format model information read via CCI Registers is not guaranteed to be up to date if the sensor module is in software standby mode. This is because new `x_output_size` and `y_output_size` values programmed during software standby mode will not be applied until the sensor module enters the streaming mode.

The intention is that the software or the dedicated image-processing device automatically configures itself by extracting the frame format details from the start of frame line(s) (Status Line).

The first embedded data line at the start of a frame must contain the frame format description.
This is very important for applications like for example digital zoom where the sensor module's image size may varying dynamically over time.

The `frame_format_model_type` register specifies which frame format description that particular sensor is using.

The aim of the generic frame model specified is to provide a flexible way of describing the format of an image frame that enables simple profile receivers to extract only the embedded data lines and visible pixel data from the frame of CCP2 frame but contains enough information that manufacturer specific software or co-processors can extract and use the additional pixel data which may be also included within the frame of CCP2 data transmitted by the sensor.

The presence of the `frame_format_model_type` register is important as it provides a mechanism for adding new frame format descriptions in a way that host systems or co-processors can easily detect.
There is currently only 1 frame format description defined.

Code	Description
00_H	*Illegal*
01_H	2-byte Generic Frame Format Description
FF_H	*Illegal*

Table 33: Frame Format Description Codes

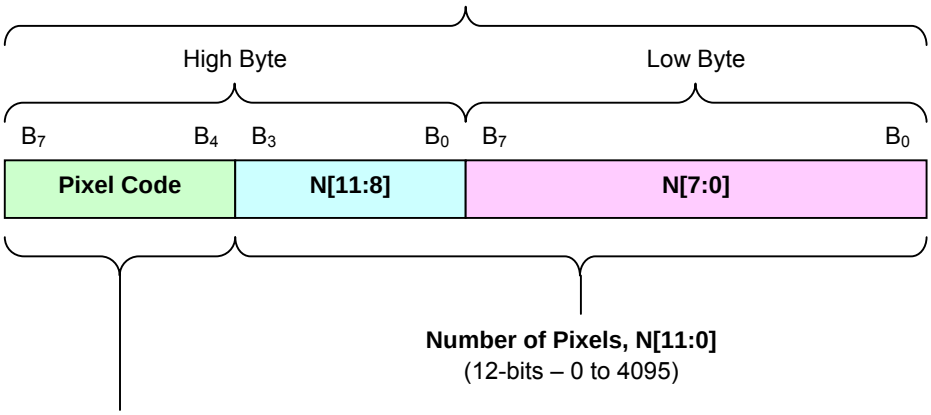
4.8.1 2-byte Generic Frame Format Description

The general rules are as follows:

- A CCP2 frame of raw Bayer data can include embedded, dummy pixel, black pixel, dark pixel, visible pixel and manufacturer specific pixel data.
- The dummy, black, dark, visible and manufacturer specific pixel data can be in the form of any combination of columns and/or rows.
- The embedded data must only be used in rows.
- There must be at least 1 embedded data line at the start and the end of the frame.

The basic component of this format is a 2-byte descriptor.

2-byte Generic Field Format Descriptor



4-bit Pixel Data Code

Code	Pixel Data	Code	Pixel Data
0	*Illegal*	8	Manufacturer Specific Pixel Type 0
1	Embedded Data	9	Manufacturer Specific Pixel Type 1
2	Dummy Pixel Data	10	Manufacturer Specific Pixel Type 2
3	Black Pixel Data	11	Manufacturer Specific Pixel Type 3
4	Dark Pixel Data	12	Manufacturer Specific Pixel Type 4
5	Visible Pixel Data	13	Manufacturer Specific Pixel Type 5
6	Reserved	14	Manufacturer Specific Pixel Type 6
7	Reserved	15	*Illegal*

Figure 27: 2-byte Generic Frame Format Descriptor

The format of the 2-byte descriptor is as follows:

1. Pixel Code (Top 4-bits in the MS Byte)
2. The pixel code defines the type of pixel data, embedded, dummy, black, dark, visible or manufacturer specific.
3. Number of Pixels (12-bit - Bottom 4-bits in the MS Byte + the LS Byte)
4. The number of pixels with the type defined in the pixel code.

5. The maximum number of pixels specified by a descriptor is 4095.

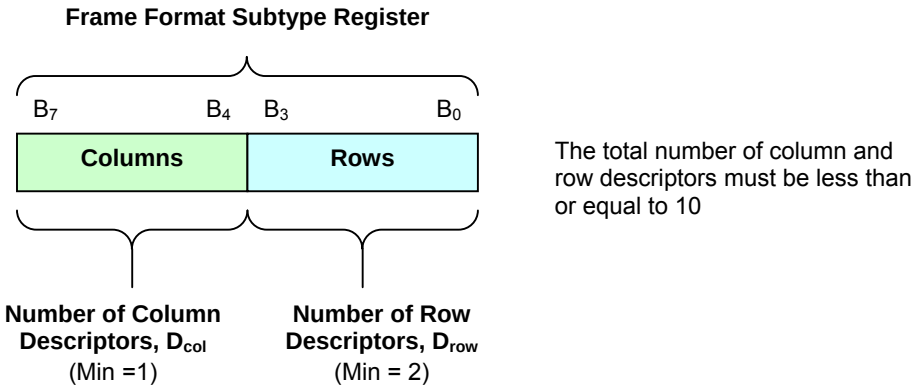


Figure 28: Frame Format Subtype register - Number of Column and Row descriptors used

The `frame_format_model_subtype` register specifies the number of descriptors used to describe the columns and the rows.

The top 4-bits of the register specifies the number of column descriptors, D_{col}.

The bottom 4-bit of the register specifies the number of row descriptors, D_{row}.

Rules for Descriptors:

1. The column descriptors must be specified first (x-direction – C-C section)
2. There must be block of D_{col} column descriptors followed by a block of D_{row} row descriptors. Column and row descriptors cannot be interleaved.
3. The column descriptors specify the left to right structure of a horizontal cross-section (C-C section in the examples) through the frame of image data. This cross-section must intersect the visible portion of the image data.
4. The total number of columns defined by the column descriptors must equal the number of pixels between the Line Start and Line End embedded codes.
5. The row descriptors specify the top to bottom structure of a vertical cross-section (R-R section in the examples) through the frame of image data. This cross-section must intersect the visible portion of the image data.
6. The total number of rows defined by the column descriptors must equal the number of rows in the whole frame of image data.
7. The embedded data type is only valid for rows and the embedded data type is valid for the whole line between the Line Start and Line End embedded codes.
8. For the non-embedded data types the width of the region defined by a row descriptor is the region's intersection with the visible pixel type i.e. the width of the visible region.
9. For the non-embedded data the height of the region defined by column descriptor is the region's intersection with the visible pixel type region i.e. the height of the visible region.
10. Any region of image data that is not defined by the descriptors must be treated as dummy pixel data.
11. The minimum number of descriptors is 3. One for the x-direction and two for the y-direction.
12. The total number of column and row descriptors must be less than or equal to 15.

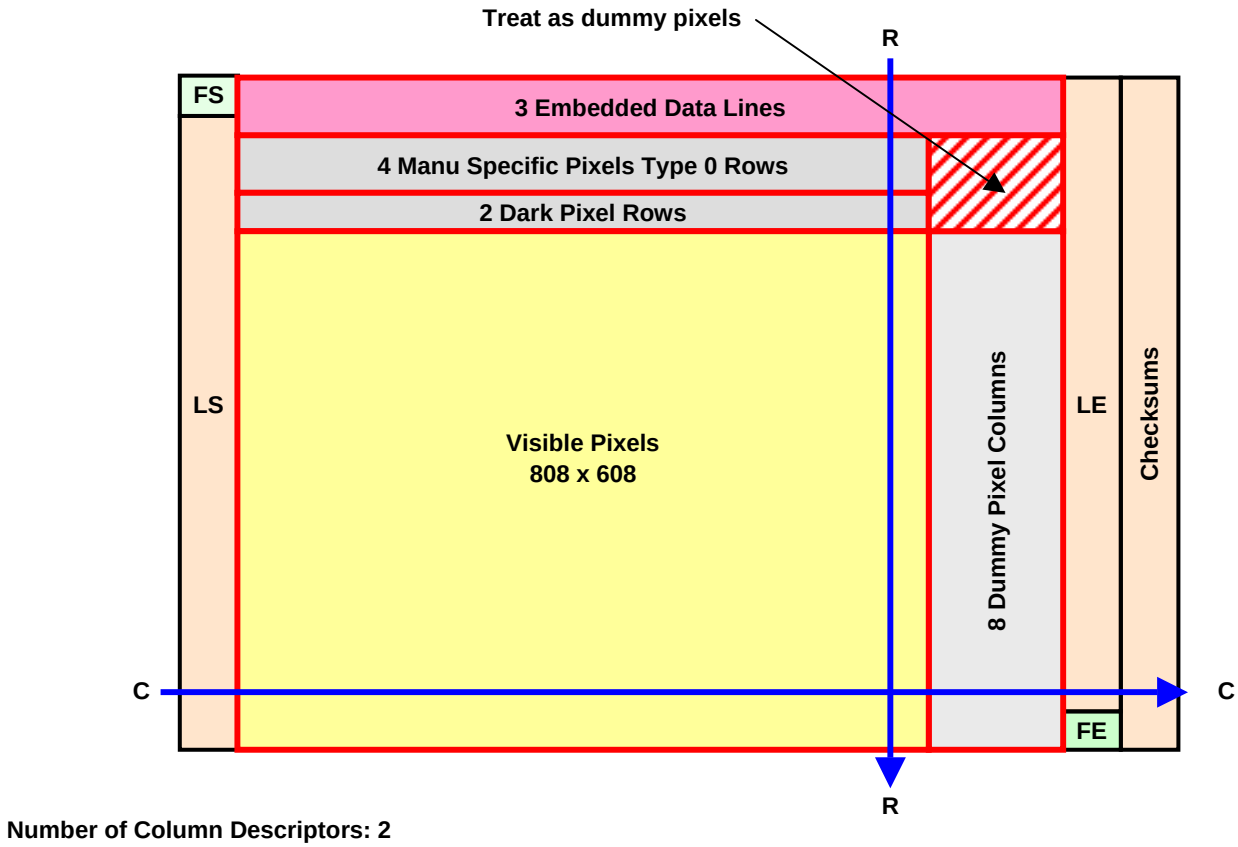
A simple profile receiver **must** be able extract the embedded data lines and the visible image pixel data from all of the examples of the generic frame format description are given in the following pages.

The first embedded data line at the start of a frame must contain the frame format description.

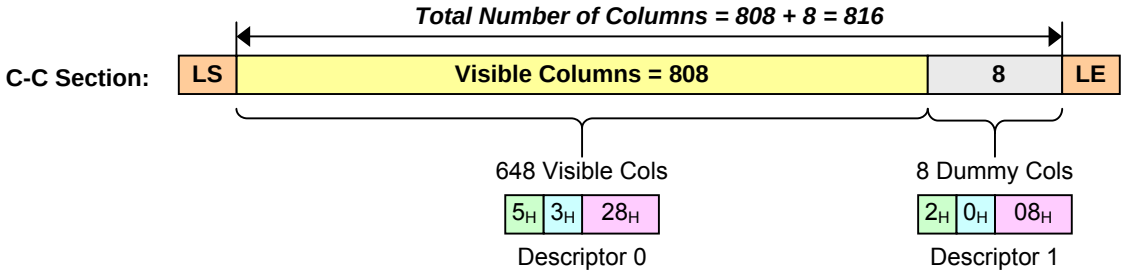
Register Name	Type	RW	Comment
frame_format_model_type	8-bit unsigned integer	RO	0x01: 2-Byte Generic Frame Format
frame_format_model_subtype	8-bit unsigned integer	RO	Contains the number of column and row of descriptors used to describe the frame format
frame_format_descriptor_0	16-bit unsigned integer	RO Dynamic	
frame_format_descriptor_1	16-bit unsigned integer	RO Dynamic	
frame_format_descriptor_2	16-bit unsigned integer	RO Dynamic	
frame_format_descriptor_3	16-bit unsigned integer	RO Dynamic	
frame_format_descriptor_4	16-bit unsigned integer	RO Dynamic	
frame_format_descriptor_5	16-bit unsigned integer	RO Dynamic	
frame_format_descriptor_6	16-bit unsigned integer	RO Dynamic	
frame_format_descriptor_7	16-bit unsigned integer	RO Dynamic	
frame_format_descriptor_8	16-bit unsigned integer	RO Dynamic	
frame_format_descriptor_9	16-bit unsigned integer	RO Dynamic	
frame_format_descriptor_10	16-bit unsigned integer	RO Dynamic	
frame_format_descriptor_11	16-bit unsigned integer	RO Dynamic	
frame_format_descriptor_12	16-bit unsigned integer	RO Dynamic	

Register Name	Type	RW	Comment
frame_format_descriptor_13	16-bit unsigned integer	RO Dynamic	
frame_format_descriptor_14	16-bit unsigned integer	RO Dynamic	

Table 34: 2-byte Generic Frame Format Description Registers



Number of Column Descriptors: 2



Number of Row Descriptors: 4

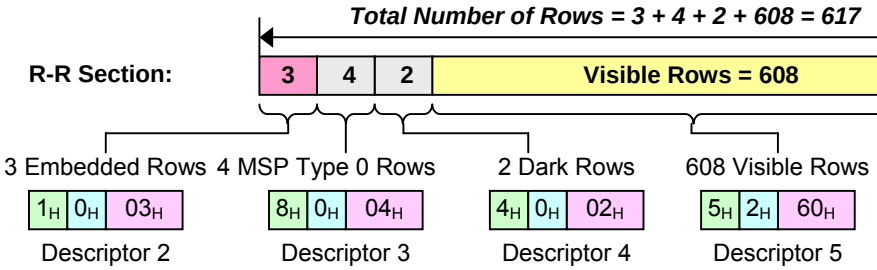


Figure 29: Generic Frame Format Description - SVGA Example

4.9 Embedded Data Line Formats

The purpose of the embedded data line formats is to allow the inclusion of extra information such as device set up information, within the output data stream.

An example of this is the frame format description, which must be included in the embedded data line at the start of frame

The high level format of embedded data lines is as follows

- The Frame Start/Line Start embedded synchronisation code is followed immediately by one pixel value which denotes the format used for the embedded data
- A sequence of embedded data values in the appropriate format.
- The remainder of the video portion of the line, up to the Frame End/Line End embedded synchronisation code is padded with 07_H characters.

The embedded data line format is fully specified in the following sections:

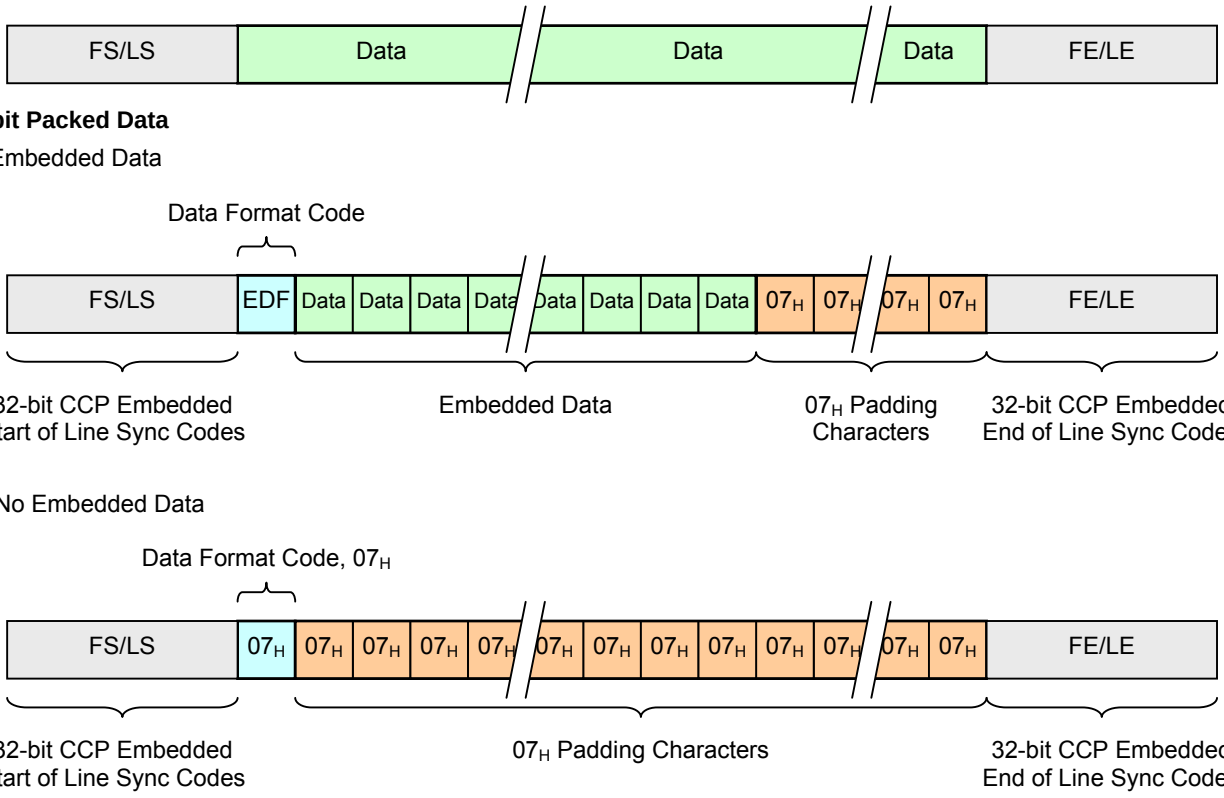


Figure 30: Embedded Data Line Format

Embedded Data Format Code	Descriptions
00_H	Illegal – If found treat as No Data
07_H	No Data – Remainder of line 07_H
$0A_H$	Simplified 2-Byte Tagged Data Format
$0B_H$-FE_H	Reserved for future use
FF_H	Illegal – If found treat as No Data

Table 35: Embedded Data Format Codes

4.9.1 Simplified 2-Byte Tagged Data Format

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

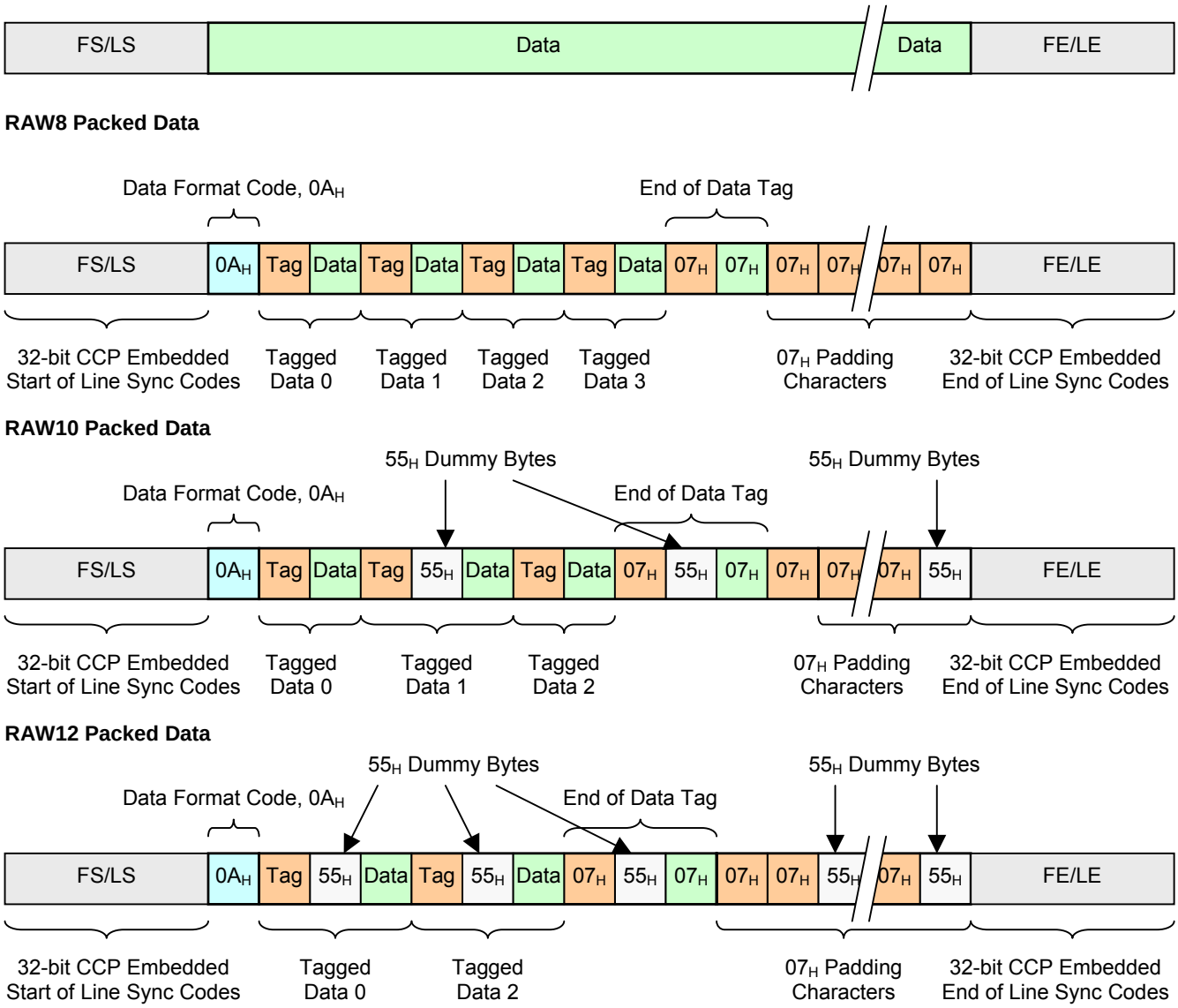


Figure 31: Tagged Data Format

Table 36: Tagged Data Format Tag Code Summary

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

There are 3 different types of tags

- Data Tags

The CCI register index is auto incremented AFTER the data byte.

The second byte is the data value for that tag.

There are two data tags one for valid data and a second for null data.

Null data is when a CCI a register value for the current CCI index does not exist. In this case 07_H is output as the data byte value.

For the Null data tag the receiver or host system must always ignore the data byte value.

- CCI Address Tags

The data byte contains either the MSB or the LSB of the CCI register index.

The CCI Register Index Tags can be used in two ways.

The first is using both the MSB and LSB address tags together to from a “long” jump over large areas of un-used CCI register space.

The second way is using only the LSB address tag, to implement a “short” jump over small areas of un-used CCI register space.

- Special Tags

End of Data Tag – 07_H

The tag values used are sufficient to prevent false CCP2 synchronisation codes from occurring with the embedded data as it is not possible to generate a sequence of 16 consecutive 0’s.

General Rules:

- The embedded data sequence of tags and the number of embedded data lines must not vary dynamically from frame to frame.

This is to speed up software-based systems by allowing the host to build up a positional map (pixel number N = the value for CCI register M) of from the embedded data lines in the first frame of image data received by the host system.

- The maximum number of tag-data pairs is 127. This is for compatibility with horizontal scaled output modes where the output image width may be only 256 pixels.

The rules for a Long Jump are:

- The MSB and LSB Address Tags must always be used together.
- Of the two tags the MS Register Index Byte tag must be specified first.
- The MSB and LSB Address Tags must be the first two tags at the start of the embedded data.
- The 4 byte MS/LS Register Tag sequence can occur anywhere in the embedded data, for example to jump over large sections of un-used register locations
- The jump may be either forward or backwards within the CCI register space.

The rules for a Short Jump are:

- Only the LSB Address Tag is used.
- The 2 byte LS Register Tag sequence can occur anywhere in the embedded data, after the initial “long” jump, for example to jump over small sections of un-used register locations
- The jump may be either forward or backwards within the CCI register space.

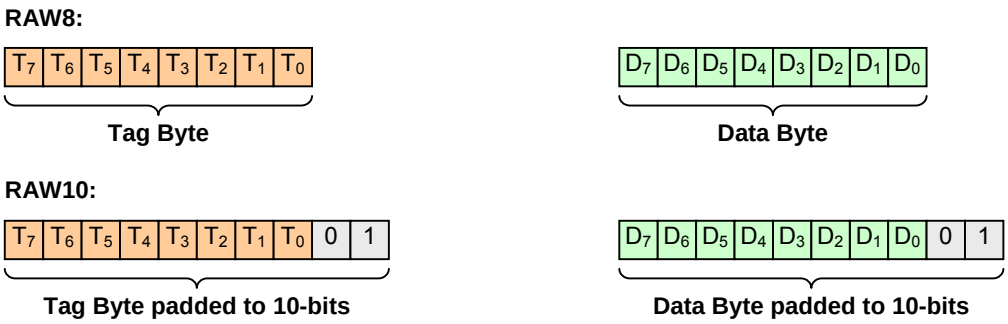


Figure 32: 2 Byte Tagged Data Packet - Data Packing for RAW8 and RAW10 Data Formats

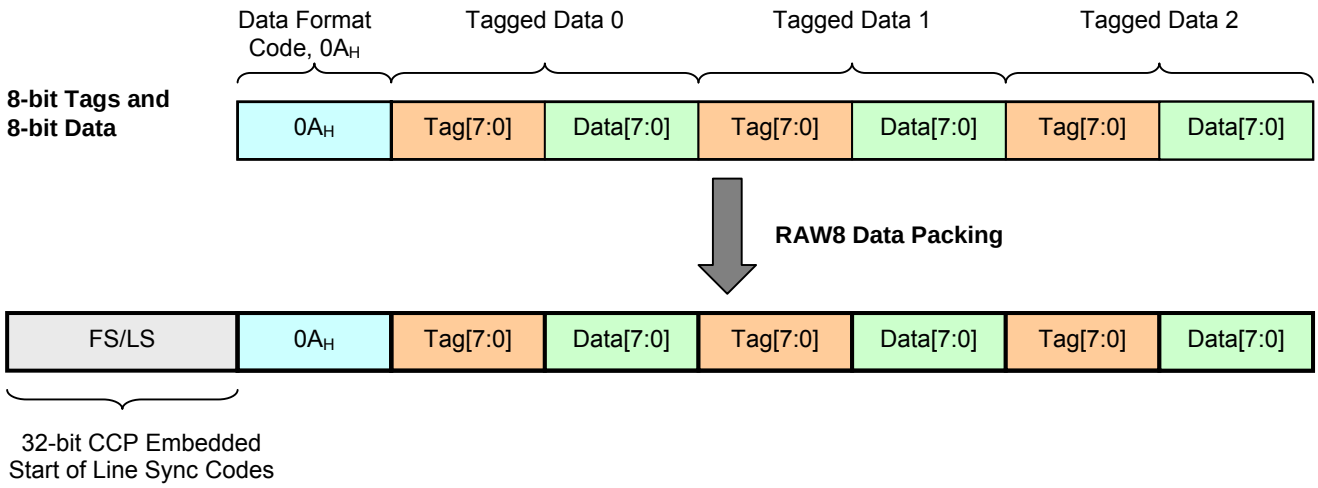


Figure 33: RAW8 Embedded Data Packing

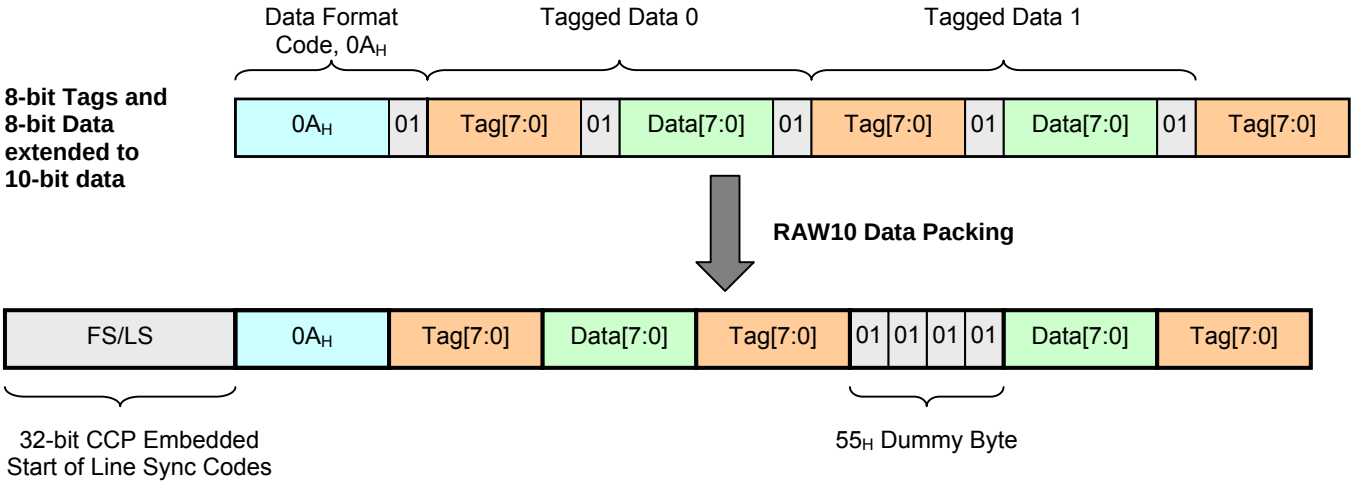


Figure 34: RAW10 Embedded Data Packing

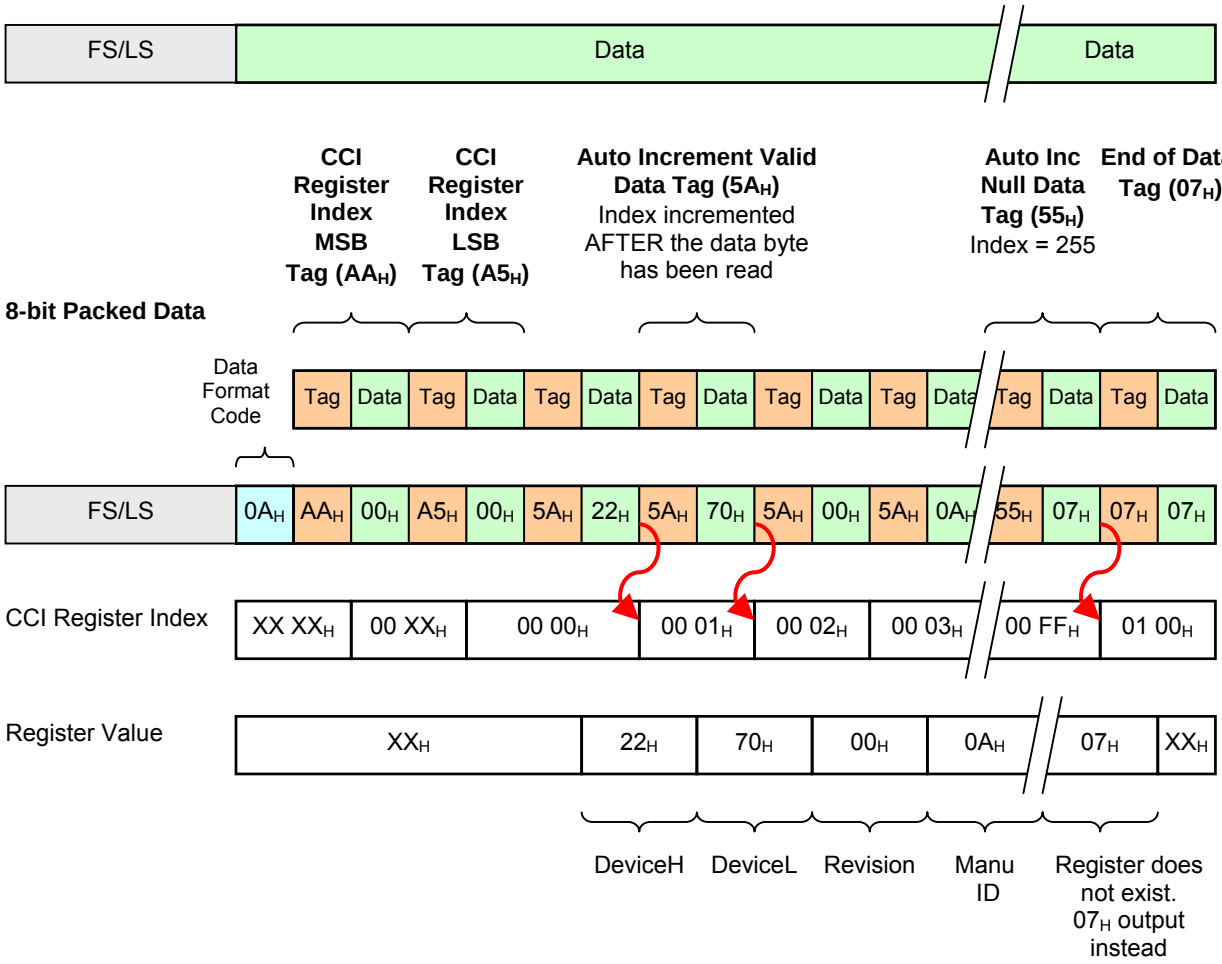


Figure 35: Embedded Data Example

5 Video Timing

5.1 Introduction

This chapter defines the following Video Timing functions:

- Programmable Image Size
- Image Readout order modes – Horizontal Mirror and Vertical Flip
- Sub-sampled readout modes
- Variable Line Length and Frame Length
- External Clock to Pixel Clock Relationship
- Video Timing Requirements
- Video Timing Capability Registers

This chapter specifies the video timing for the image data that is readout from the sensor modules pixel array. This is not necessarily the same size as the output image data.

Features such as programmable image size and sub-sampled modes all effect the size of image read from the pixel array. For these features the output image is the same size as the image read from the pixel array.

The programmable image size, digital image scaling and output size are independent functions. It is the responsibility of the host to ensure that these functions are programmed correctly for the intended application. These functions are used to reduce amount of data and also reduce the peak data rate of CCP2.

Integration time parameters are specified in terms of the video timing for the pixel array.

The application of all of the video timing read/write parameters must be re-timed to the start of frame boundary to ensure that the parameters are consistent within a frame.

For the CCI indices of the registers defined in this section please refer to the CCI Register Map Chapter.

The Video Timing Application Note presents an example method that a host system can use to configure and control the sensor's module's video timing and use the information in the video timing capability registers.

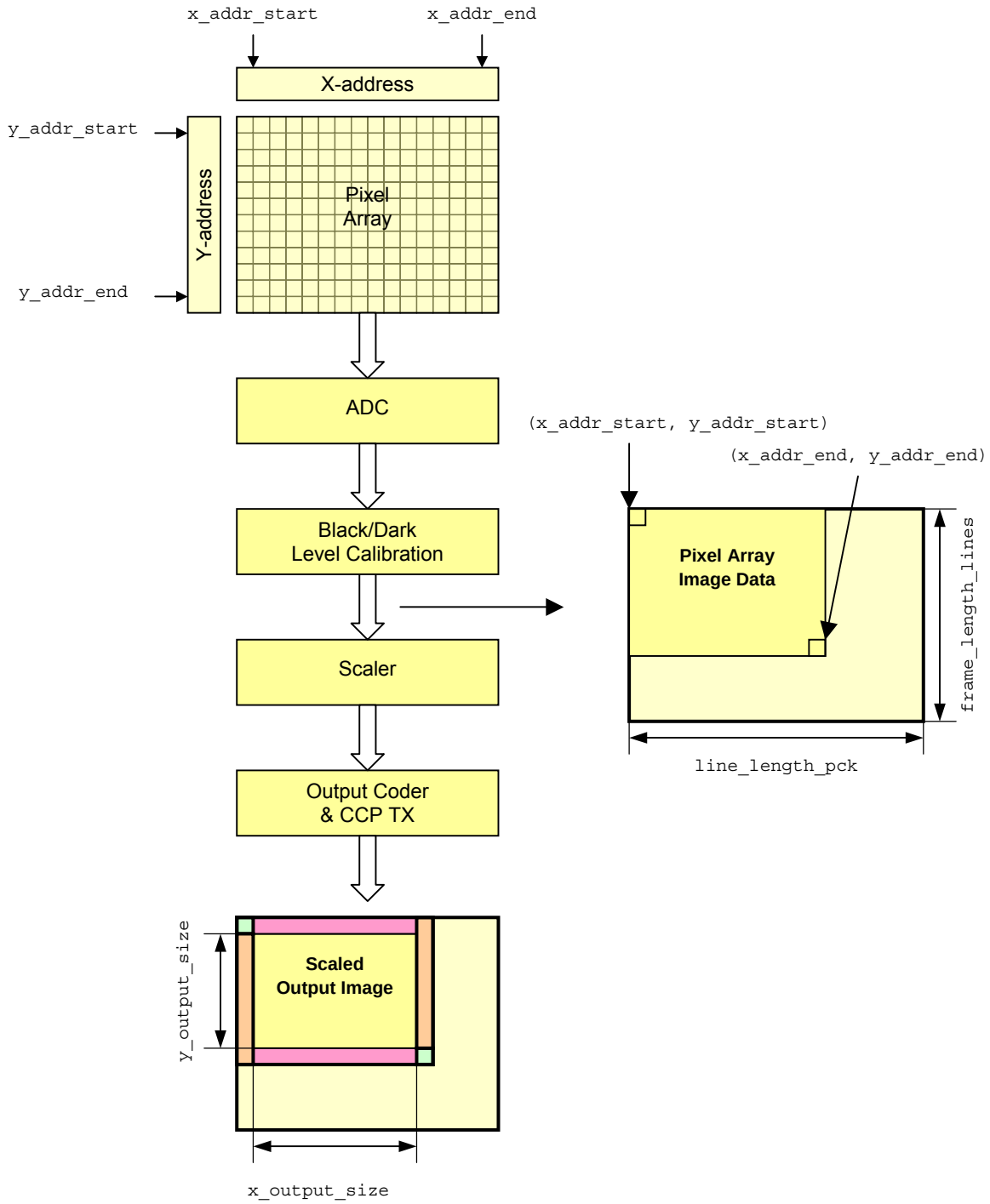


Figure 36: Video Timing Overview

5.2 Image readout

There are 3 mandatory features for the sensor modules image readout

- Programmable image size
- Horizontal Mirror and/or Vertical Flipped image readout
- Sub-sampled image readout

The following sections specify the above 3 features.

A user readout model is also defined

5.2.1 User Readout Model

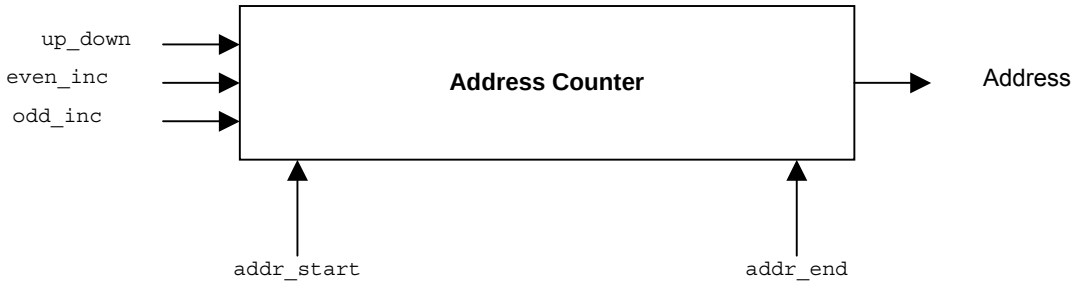


Figure 37: User Readout Model

The user readout model is based on an up/down address counter with programmable start and end values.

In non-sub-sampled readout modes the counter will count either up from the start address to the end address (inclusive) or down from the end address to the start address (inclusive).

The mirror and flip control bits determine the direction of the address count.

If the counter is counting up the termination condition is greater than the end address.
If the counter is counting down the termination condition is less than the start address.

To model sub-sampled readout modes the counter has two increment values.

- `even_inc`

The amount the address counter is incremented for EVEN pixels in the readout order– 0, 2, 4 etc.

- `odd_inc`

The amount the address counter is incremented for ODD pixels in the readout order – 1, 3, 5 etc.

If the counter is counting down then the above values are used to decrement the address counter.

For non-sub-sampled readout modes both `even_inc` and `odd_inc` are set to 1.

5.2.2 Programmable Image Size

There are 2 independent methods for programming image size.

- Programmable addressable region of the pixel array
- Programmable width and height for output image data

The output image size is a function of the size of the addressed region of the pixel array, the sub-sampling factor and image scaling downscale factor programmed.

It is the responsibility of the host system to calculate the output image size based on the above parameters and then program the sensor modules x and y output size registers.

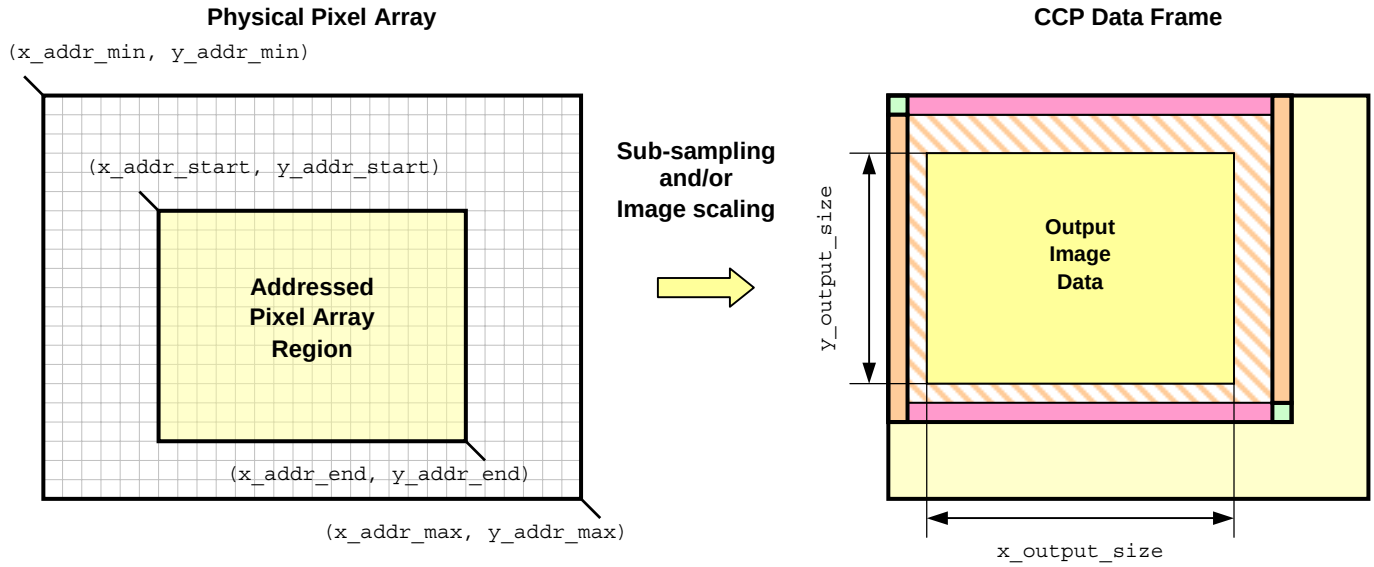


Figure 38: Programmable Image Size

5.2.2.1 Pixel Array Addresses

The addressable region of pixel array must be at least 8 columns and 8 rows larger than the width and height of basic image format of the sensor module (Table 37).

This gives a minimum border of 4 pixels around the whole pixel array for the colour reconstruction algorithms to use at the edges of the pixel array (Figure 39).

The addressed region of the pixel array is controlled by the `x_start_addr`, `y_start_addr`, `x_end_addr` and `y_end_addr` registers (Table 38).

The start and end addresses are limited to even and odd numbers respectively to ensure that there is always a even number of pixels readout in x and y.

The limits for the above parameters are given by the `x_addr_min`, `y_addr_min`, `x_addr_max` and `y_addr_max` registers (Table 39). From this information the host can automatically determine the sensor module's maximum resolution and thus its image format.

The image size programmed by the host should include any border pixels required for the subsequent image processing algorithms.

Format	Format Width	Format Height	Addressable Pixel Array Width	Addressable Pixel Array Height
VGA	640	480	648	488
SVGA	800	600	808	608
1.0MP	1152	864	1160	872
SXVGA	1280	960	1288	968
UXGA	1600	1200	1608	1208
QXGA	2048	1536	2056	1544

Table 37: Examples of Addressable Pixel Array Sizes

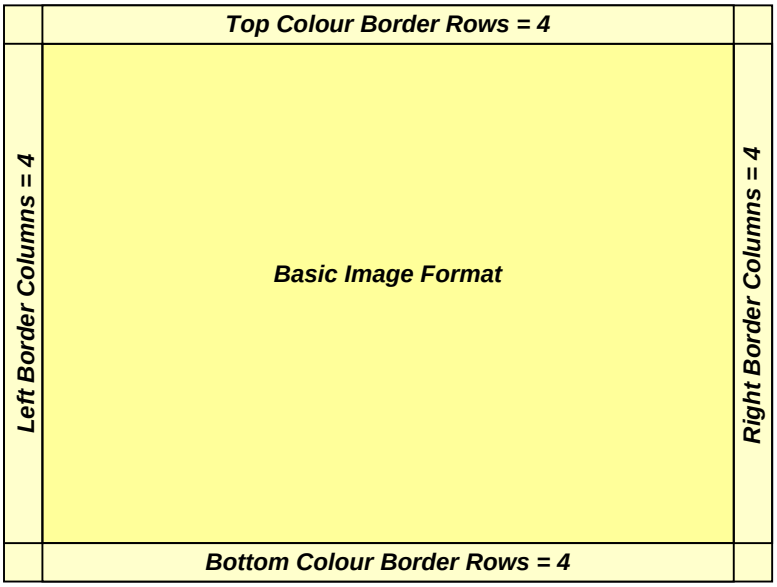


Figure 39: Basic Image Format plus Additional Border Rows

The rules for the programmable pixel array start and end address are listed below:

- The application of `x_addr_start`, `x_addr_end`, `y_addr_start` and `y_addr_end` must be re-timed internally to the start of frame boundary.
- The minimum limit for the x and y addresses are defined by the, `x_addr_min` and `y_addr_max` registers (Table 39). These values must always be 0.
- The maximum limit for the x and y addresses are defined by the, `x_addr_max` and `y_addr_min` registers (Table 39).
- The end address must be greater than the start address
- The x and y start addresses are restricted to even numbers only.

i.e. 0, 2, 4, 6, 8 etc

- The x and y end addresses are restricted to odd numbers only

i.e. 1, 3, 5, 7, 9, etc

- The minimum size in the y-direction is 2 lines.
- (`x_addr_max`–`x_addr_min`+1) > = Width of Basic Image Format+8
- (`y_addr_max`–`y_addr_min`+1) > = Height of Basic Image Format+8

Register Name	Type	RW	Comment
x_addr_start	16-bit unsigned integer	RW	X-address of the top left corner of the visible pixel data Values: Even numbers only, 0 2, 4, 6… Units: Pixels
y_addr_start	16-bit unsigned integer	RW	Y-address of the top left corner of the visible pixel data Values: Even numbers only, 0 2, 4, 6… Units: Lines
x_addr_end	16-bit unsigned integer	RW	X-address of the bottom right corner of the visible pixel data Values: Odd numbers only, 1, 3, 5, 7… Units: Pixels
y_addr_end	16-bit unsigned integer	RW	Y-address of the bottom right corner of the visible pixel data Values: Odd numbers only, 1, 3, 5, 7… Units: Lines

Table 38: Pixel Array Address Registers (Read/Write)

Register Name	Type	RW	Comment
x_addr_min	16-bit unsigned integer	RO Static	Minimum X-address of the addressable pixel array Value: Always 0
y_addr_min	16-bit unsigned integer	RO Static	Minimum Y-address of the addressable pixel array Value: Always 0
x_addr_max	16-bit unsigned integer	RO Static	Maximum X-address of the addressable pixel array
y_addr_max	16-bit unsigned integer	RO Static	Maximum Y-address of the addressable pixel array

Table 39: Pixel Array Address Capability Registers (Read Only and Static)

5.2.2.2 Output Size

The x_output_size and y_output_size registers (Table 40) control the width and height of the visible pixel data within the frame of data output form the sensor module.

- The application of x_output_size and y_output_size must be re-timed internally to the start of frame boundary.
- The x_output_size and y_output_size must be programmable in multiples of 2 pixels.
- The minimum output image size in the x-direction is 256 pixels.

This is necessary to ensure that there is always sufficient time within a line to output embedded CCI register data.

- The frame format description must track as the x_output_size and y_output_size values change.
- The host system when calculating the x_output_size value must ensure that the CCP2 Data transmitted per line length is a multiple of the correct number of pixels for the CCP2 RAW data format used (e.g. 4 pixels for RAW8 and 16-pixels for RAW10).

This is achieved by the host reading the sensor module's frame format description to determine the fixed number of columns and rows, which are readout in addition to the variable number of image data row and columns.

Figure 40 shows an example frame of image data. The information describing the number of additional columns and rows is contained in the frame format description, which output in the 1st line of embedded data.

- If the x_output_size and y_output_size values are smaller than actual image size, the sensor module will crop the image data to the size specified by the x_output_size and y_output_size.
- If the x_output_size and y_output_size values are larger than the actual image size, then output data between the actual image size and output size defined the x_output_size and y_output_size by is undefined.

Register Name	Type	RW	Comment
x_output_size	16-bit unsigned integer	RW	Width of image data output from the sensor module Units: Pixels
y_output_size	16-bit unsigned integer	RW	Height of image data output from the sensor module Units: Lines

Table 40: Output Image Size Registers (Read/Write)

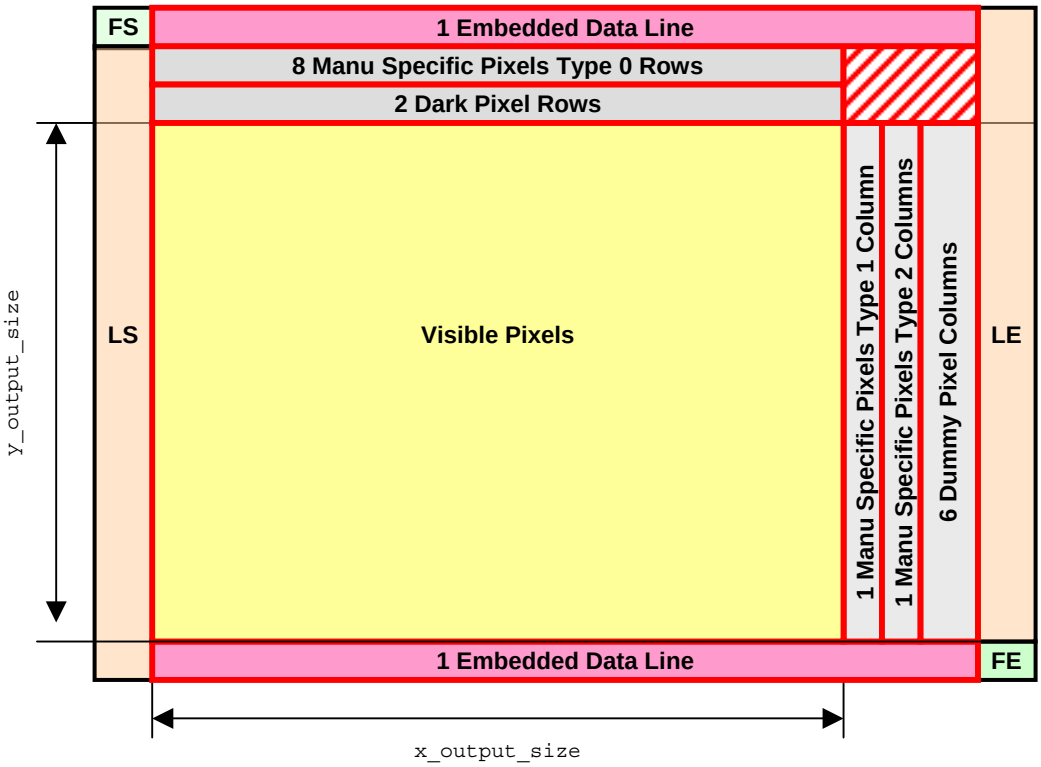


Figure 40: Output Image Size Example

5.2.3 Mirror/Flip

The sensor module has 2 independent options for altering the readout order of the image data.

- Horizontally Mirrored readout.

In this mode the columns in the pixel array are readout in reverse order.

- Vertically Flipped readout

In this mode the rows in the pixel array are readout in reverse order.

Thus the sensor module must support 4 possible pixel readout orders

1. Standard readout (Figure 41)
2. Horizontally Mirrored readout (Figure 42)
3. Vertically Flipped readout (Figure 43)
4. Horizontally Mirrored and Vertically Flipped readout (Figure 44)

The application of horizontal mirror and vertical flip readout modes must be re-timed internally to the start of frame boundary.

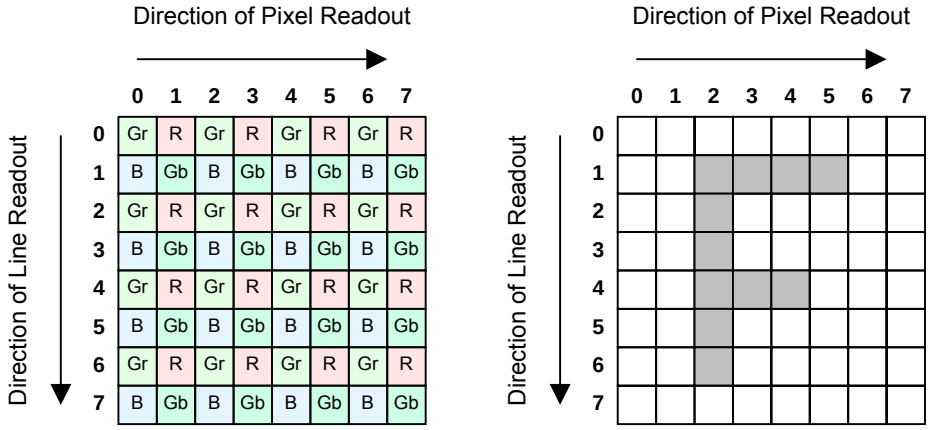


Figure 41: Standard Readout

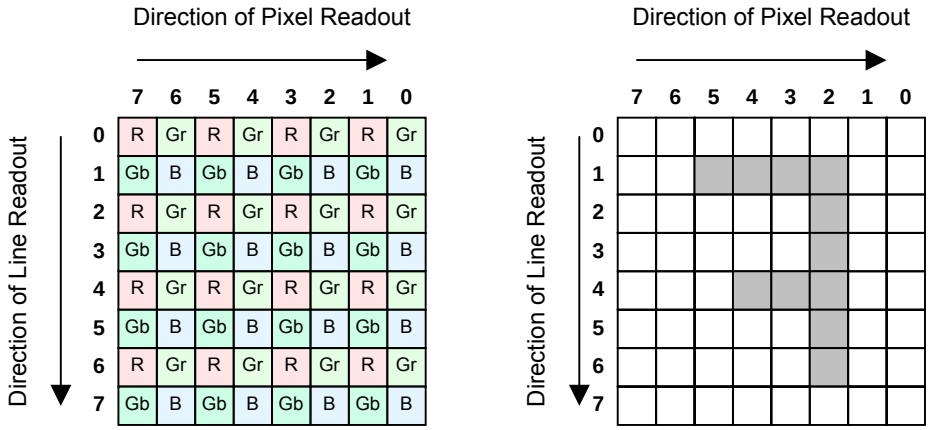


Figure 42: Horizontally Mirrored Readout

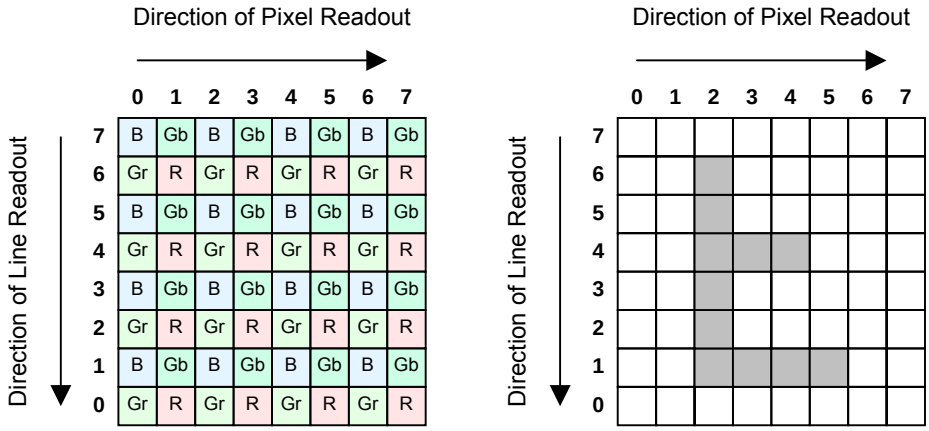


Figure 43: Vertically Flipped Readout

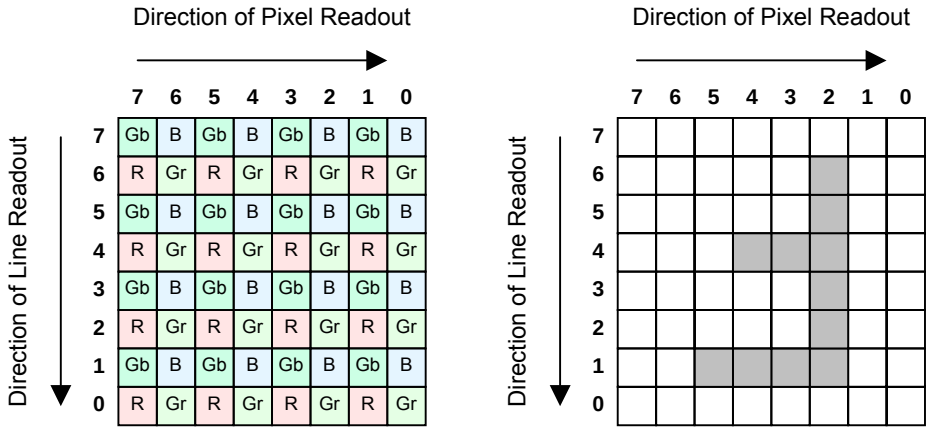


Figure 44: Horizontally Mirrored and Vertically Flipped Readout

The sensor module readout order is controlled via the `image_orientation` register (Table 41).

Register Name	Type	RW	Comment
`image_orientation`	8-bit unsigned integer	RW	Bit 0: Horizontal Mirror Enable Bit 1: Vertical Flip Enable

Table 41: Image Orientation Register (Read/Write)

Bit	Function	Default	Comment
0	Hmirror	0	0: no mirror 1: Horizontal Mirror
1	Vflip	0	0: no flip 1: Vertical Flip

Table 42: Contents of Image Orientation Register

The re-timed pixel readout order is reported by `pixel_order` register
The value of this register must also track with the programmable x and y start addresses.

Register Name	Type	RW	Comment
`Pixel_order`	8-bit unsigned integer	RO Dynamic	Defines the order of the colour pixel readout Changes with mirror and flip.

Table 43: Pixel Order Register (Read Only Dynamic)

Colour Pixel Order Code	Readout Order
0	GR BG
1	RG GB
2	BG GR
3	GB RG

Table 44 - Colour Pixel Order Code

5.2.4 Sub-Sampled Readout

By programming the x and y odd and even increment registers (Table 45) the sensor module can be configured to readout sub-sampled pixel data.

Figure 45 illustrates how the above register values can be programmed to sub-sample in x and y by a factor of 2.

Register Name	Type	RW	Comment
x_even_inc	16-bit unsigned integer	RW	Increment for even pixels in the readout order – 0, 2, 4 etc
x_odd_inc	16-bit unsigned integer	RW	Increment for odd pixels in the readout order – 1, 3, 5 etc
y_even_inc	16-bit unsigned integer	RW	Increment for even pixels in the readout order – 0, 2, 4 etc
y_odd_inc	16-bit unsigned integer	RW	Increment for odd pixels in the readout order – 1, 3, 5 etc

Table 45: X and Y-Address Increment Registers (Read/Write)

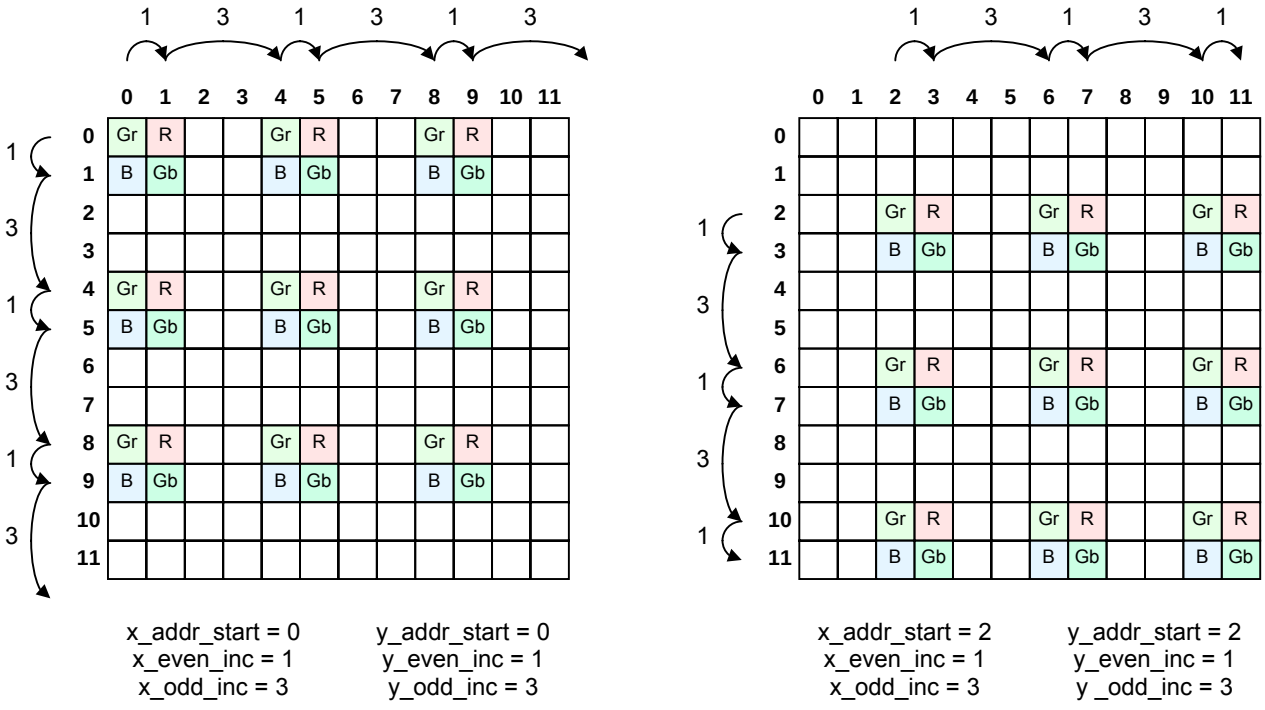


Figure 45: Sub-Sampled Readout Example

Table 46 lists the registers that define the minimum and maximum limits for the odd and even increment values.

Register Name	Type	RW	Comment
min_even_inc	16-bit unsigned integer	RO Static	Minimum Increment for even pixels
max_even_inc	16-bit unsigned integer	RO Static	Maximum increment for even pixels
min_odd_inc	16-bit unsigned integer	RO Static	Minimum Increment for odd pixels
max_odd_inc	16-bit unsigned integer	RO Static	Maximum Increment for odd pixels

Table 46: X and Y-Address Increment Parameter Limits (Read Only)

The rules for sub-sampled readout are

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

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

- The sensor modules pixel readout order register value must track as the start address, the end address, the odd address increment and the even address increment values change.
- The host system when calculating the start address, the end address, the odd address increment and the even address increment values must ensure that the CCP2 Data transmitted per line length is a multiple of the correct number of pixels for the CCP2 RAW data format used. (e.g. 4 pixels for RAW8 and 16-pixels for RAW10).

This is achieved by the host reading the sensor modules frame format description to determine the fixed number of columns and rows, which are readout in addition to the variable number of image data row and columns.

- It is the responsibility of the host to update the output image size registers according to the sub-sampling factors.

5.3 Line Length and Frame Length

The length of a line is specified as a number of pixel clocks, `line_length_pck`.

The length of a frame is specified as a number of lines, `frame_length_lines`.

The application the frame length and line length parameters must be re-timed internally to the start of frame boundary.

Register Name	Type	RW	Comment
`frame_length_lines`	16-bit unsigned integer	RW	Frame Length Units: Lines
`line_length_pck`	16-bit unsigned integer	RW	Line Length Units: Pixel Clocks

Table 47: Video Timing Registers (Read/Write)

To aid set-up by host or co-processor the read only and static frame timing parameter limits registers (Table 48) define the minimum and maximum frame and line lengths possible for that sensor module. These values are not dependent on x start, y start, x end and y end.

The sensor module does NOT internally clip the frame and line length register values.

It is the responsibility of the host or co-processor to read the minimum and maximum values and ensure that this range is not exceeded.

Register Name	Type	RW	Comment
`min_frame_length_lines`	16-bit unsigned integer	RO Static	Minimum Frame Length allowed. Value: sensor dependent Units: Lines
`max_frame_length_lines`	16-bit unsigned integer	RO Static	Maximum possible number of lines per Frame. Value: sensor dependent Units: Lines
`min_line_length_pck`	16-bit unsigned integer	RO Static	Minimum Line Length allowed. Value: sensor dependent Units: Pixel Clocks
`max_line_length_pck`	16-bit unsigned integer	RO Static	Maximum possible number of pixel clocks per line. Value: sensor dependent Units: Pixel Clocks
`min_line_blanking_pck`	16-bit unsigned integer	RO Static	Minimum line blanking time in pixel clocks Units: Pixel Clocks

Table 48: Frame Timing Parameter Limits (Read only and static)

The frame length and line length register values must be re-timed internally to the start of a frame.

5.3.1 Frame Length/Line Length CCI Message

To ensure that the 2 bytes that make up either register, are always consumed together, the `grouped_parameter_hold` register must be set to 1 prior to the transmission of the CCI message containing the frame length and line length values.

After the CCI message the `grouped_parameter_hold` register must be reset to 0 to allow the new frame length and line length values to be applied internally at the next frame boundary.

5.3.2 Frame Length/Line Length Requirements

The requirements for the frame length and line length are:

- The maximum frame length in lines must be sufficient to allow exposure times of 1 second to be achieved or 65535 lines if it is the lower value.

  The image readout time, the time from the CCP2 frame start code for frame n to the CCP2 frame end code for frame n, for this condition is the same as the image readout time for the sensor module reading out its maximum image size at its fastest frame rate e.g. $1/30^{th}$ sec.

- The maximum line length in pixel clocks must be sufficient for the line length for the sensor module reading out its maximum image size at the sensor's fastest readout rate, increased by a factor of 4.

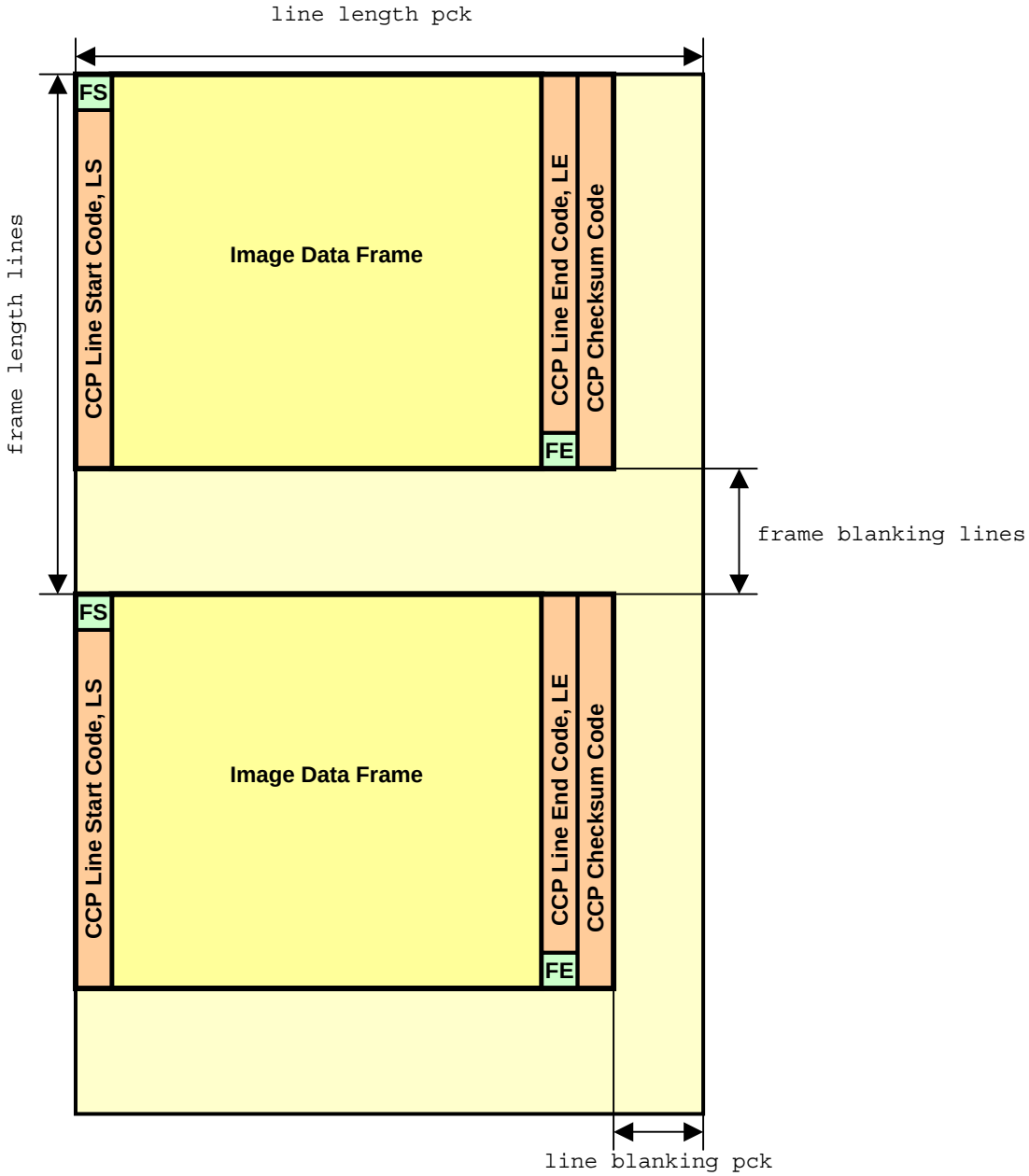


Figure 46: Frame Length/Line Length Definitions

5.4 Relationship between the External Clock Frequency and the Video Timing and Output Pixel Clock Frequencies

The sensor module contains a PLL (Phase Locked Loop) block, which generates all the necessary internal clocks from the external clock input.

Figure 49 shows the internal functional blocks, which define the relationship between the external input clock frequency and the video timing pixel clock frequencies for Profile level 0

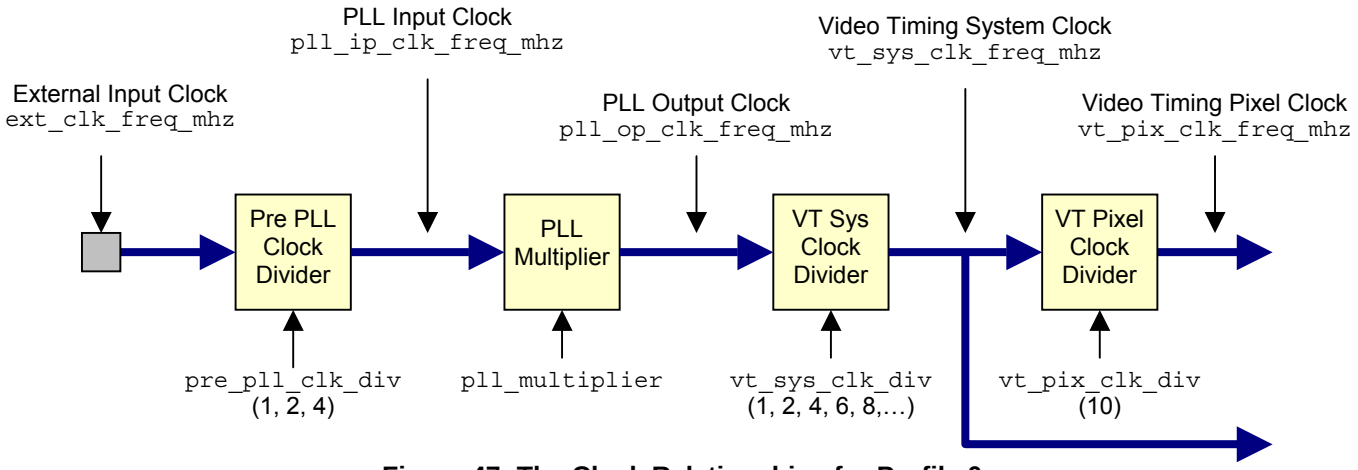
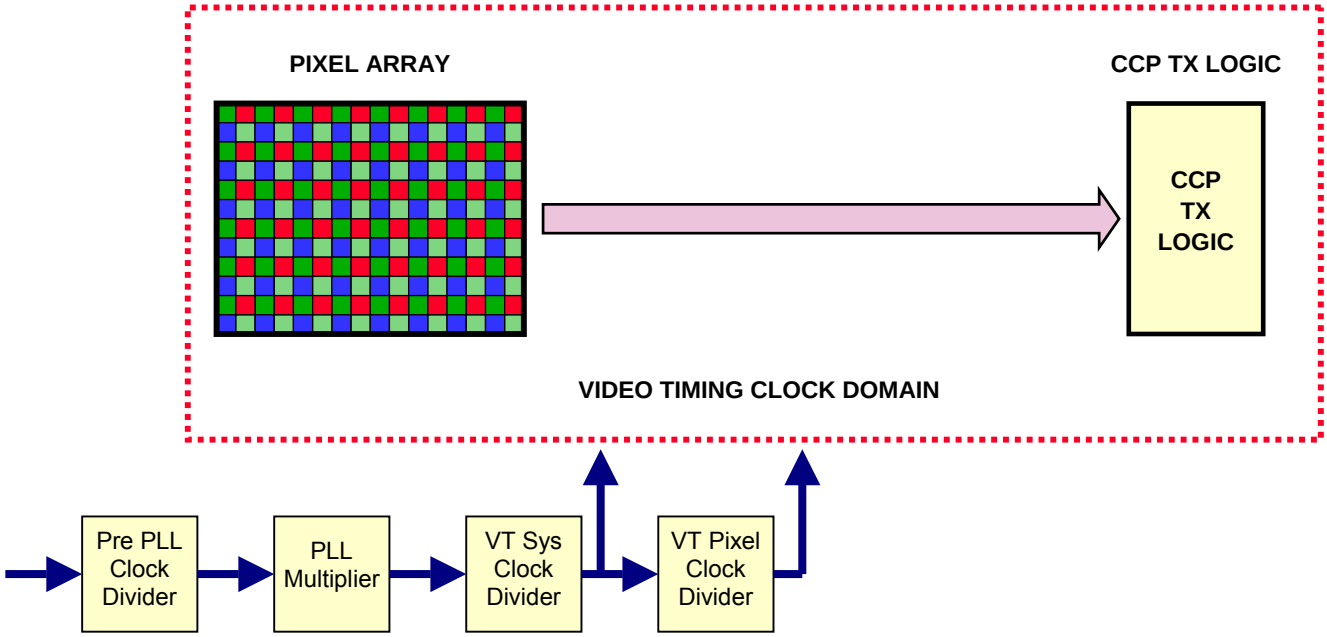


Figure 47: The Clock Relationships for Profile 0



Please note that this is a behavioural description only and does not imply any implementation

Figure 48: Profile 0 Clock Domains

Line length, min line blanking time, fine and coarse exposure values are all related to the video timing pixel clock.

In profile 0 the video timing system and pixel clock are also used for the CCP2 transmitter.

Profiles 1 and 2 have separate video timing and output clock domains. This provides a mechanism for reducing the output peak data rate when using the scaler functionality present in profiles 1 and 2. In this situation the full resolution of pixel array is still being readout internally at the target video rate e.g. 15fps or 30 fps. When programmable image size is used the output peak data rate can be reduced by using smaller line_length_pck and frame_length_lines values.

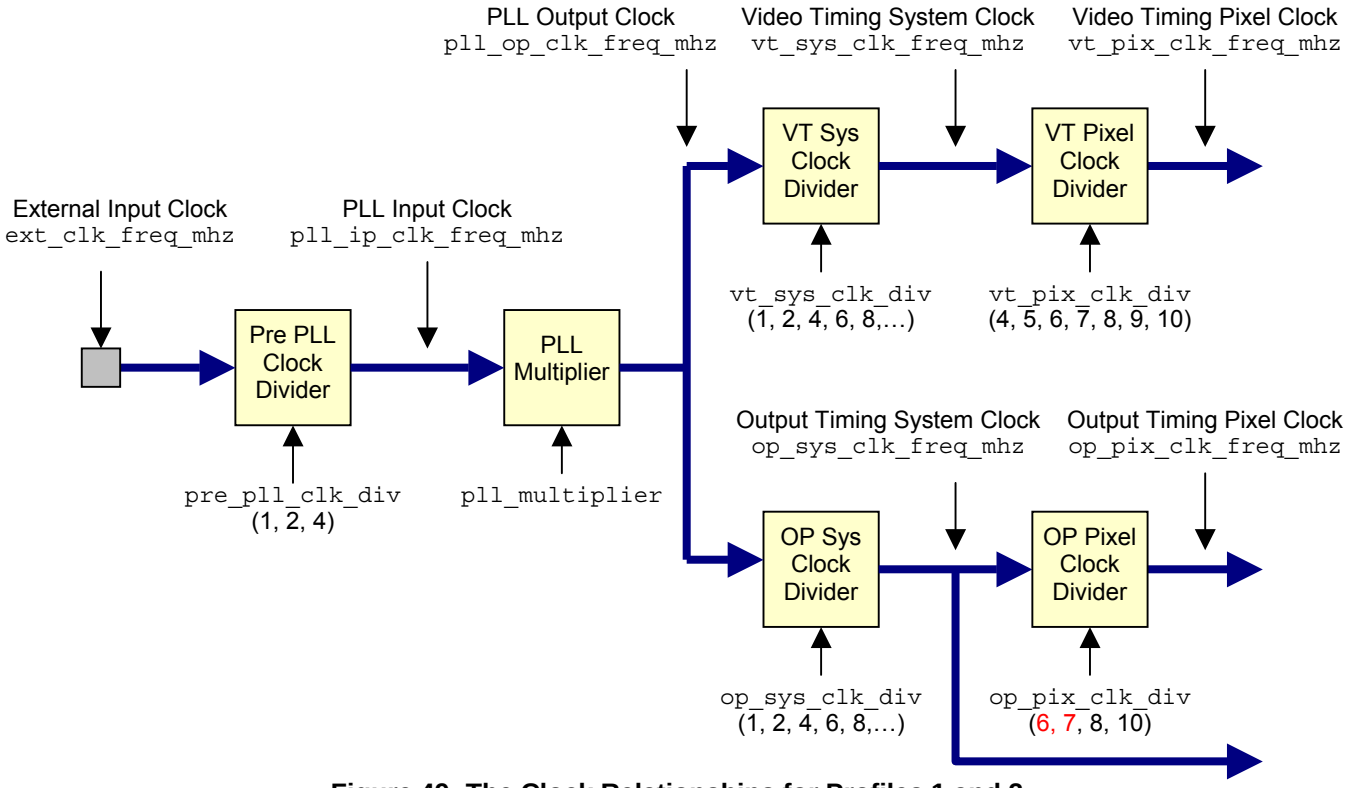
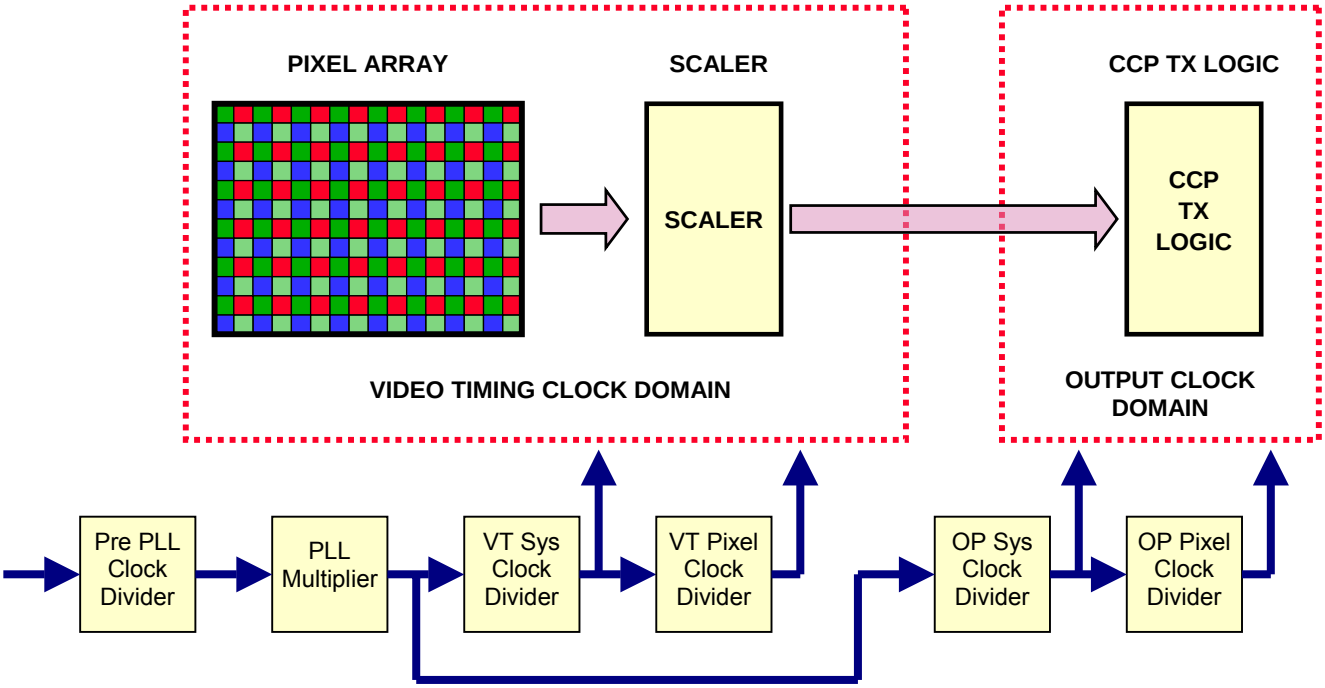


Figure 49: The Clock Relationships for Profiles 1 and 2

Values 6 and 7 for the op_pix_clk_div are not mandatory and are only required if the optional RAW6 and RAW7 modes are implemented. Also odd values can be used in op_sys_clk_div divider, but they are not mandatory. Similarly vt_pix_clk_div can use values smaller or bigger than shown in Figure 49, but these values from 4 to 10 are mandatory.



Please note that this is a behavioural description only and does not imply any implementation

Figure 50: Profile 1 & 2 Clock Domains

The equations relating the input clock frequency to the video timing and output pixel clock frequency are given below

$$vt\_pix\_clk\_freq\_mhz = \frac{ext\_clk\_freq\_mhz * pll\_multiplier}{pre\_pll\_clk\_div * vt\_sys\_clk\_div * vt\_pix\_clk\_div}$$

$$op\_pix\_clk\_freq\_mhz = \frac{ext\_clk\_freq\_mhz * pll\_multiplier}{pre\_pll\_clk\_div * op\_sys\_clk\_div * op\_pix\_clk\_div}$$

Table 49 lists the CCI registers, which control the relationship between the external input clock frequency and the pixel clock frequency.
+

Register Name	Type	RW	Comment
vt_pix_clk_div	16-bit unsigned integer	RW	Video Timing Pixel Clock Divider Value
vt_sys_clk_div	16-bit unsigned integer	RW	Video Timing System Clock Divider Value
pre_pll_clk_div	16-bit unsigned integer	RW	Pre PLL clock Divider Value
pll_multilpier	16-bit unsigned integer	RW	PLL multiplier Value
op_pix_clk_div	16-bit unsigned integer	RW	Output Pixel Clock Divider Value
op_sys_clk_div	16-bit unsigned integer	RW	Output System Clock Divider Value

Table 49: Clock Division and PLL multiplier registers

The AEC/AWB algorithms use the video timing clock division registers to calculate the absolute exposure/integration time.

The Pre-PLL Clock Divider enables sensor module to divide down an external clock frequency, which is too high for the PLL into one which lies with the PLL input range.

This greatly increases the range of input clocks the sensor module can work with as well as minimising the variation in output data rate across the full range of external clock frequencies.

The minimum and maximum limits for the clock frequencies, clock dividers and PLL multipliers in a sensor modules input clock are fully described by a bank of Read Only CCI registers

Sensor modules must be able to handle any input external clock frequency in the 6.0 MHz to 27 MHz range.

Register Name	Type	Default	Comment
vt_pix_clk_div	16-bit unsigned integer	10	The video timing pixel clock divider Mandatory values: Profile 0: vt_pix_clk_div = 10 Profiles 1 & 2: vt_pix_clk_div = 4, 5, 6, 7, 8, 9 and 10

Table 50: Video Timing Pixel Clock Divider Register (Read/Write)

Register Name	Type	Default	Comment
vt_sys_clk_div	16-bit unsigned integer	1	Only even vt_sys_clk_div values are supported with the exception of vt_sys_clk_div = 1 For example Code: 1: Divide by 1 Code 2: Divide by 2 Code 4: Divide by 4 Code 6: Divide by 6 Code 8: Divide by 8

Table 51: Video Timing System Clock Divider Register (Read/Write)

Register Name	Type	Default	Comment
Pre_pll_clk_div	16-bit unsigned integer	1	Only even pre_pll_clk_div values are supported with the exception of pre_pll_clk_div = 1 For example Code 1: Divide by 1 Code 2: Divide by 2 Code 4: Divide by 4 Code 6: Divide by 6 Code 8: Divide by 8

Table 52: Pre-PLL Clock Divider Register (Read/Write)

Register Name	Type	Default	Comment
Pll_multiplier	16-bit unsigned integer		PLL multiplier value e.g. Code 10 = multiply by 10

Table 53: PLL Multiplier (Read/Write)

Register Name	Type	Default	Comment
op_pix_clk_div	16-bit unsigned integer	10	The output pixel clock divider Profiles 1 & 2: op_pix_clk_div = 6, 7, 8 and 10 Values 6 and 7 are not mandatory and are only required if the optional RAW6 and RAW7 compressed modes are included.

Table 54: Output Pixel Clock Divider Register (Read/Write)

Register Name	Type	Default	Comment
op_sys_clk_div	16-bit unsigned integer	1	Only even op_sys_clk_div values are supported (mandatory) with the exception of op_sys_clk_div = 1 For example Code: 1: Divide by 1 Code 2: Divide by 2 Code 4: Divide by 4 Code 6: Divide by 6 Code 8: Divide by 8 (Profiles 1 & 2 Only)

Table 55: Output System Clock Divider Register (Read/Write)

5.4.1 Clock Examples

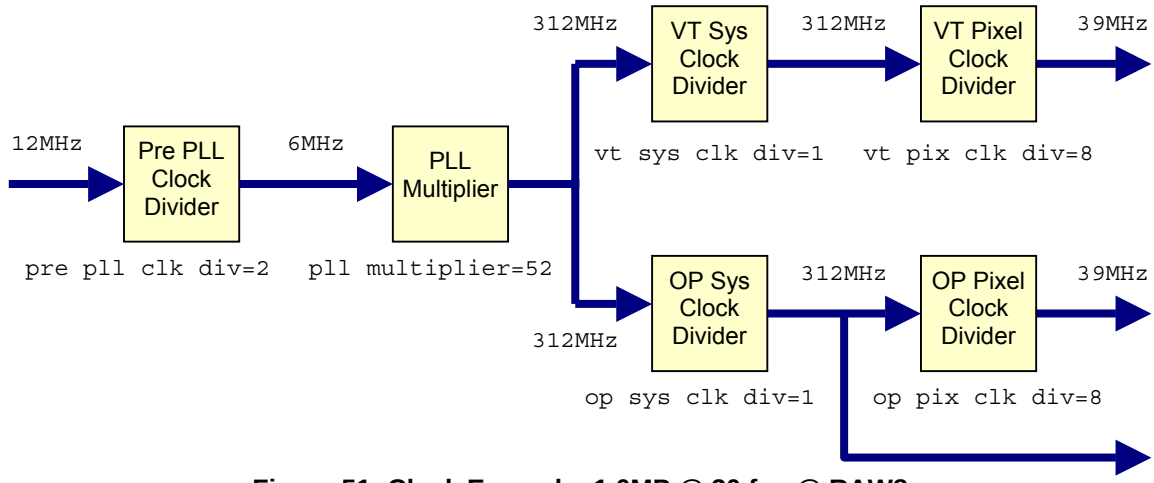


Figure 51: Clock Example: 1.0MP @ 30 fps @ RAW8

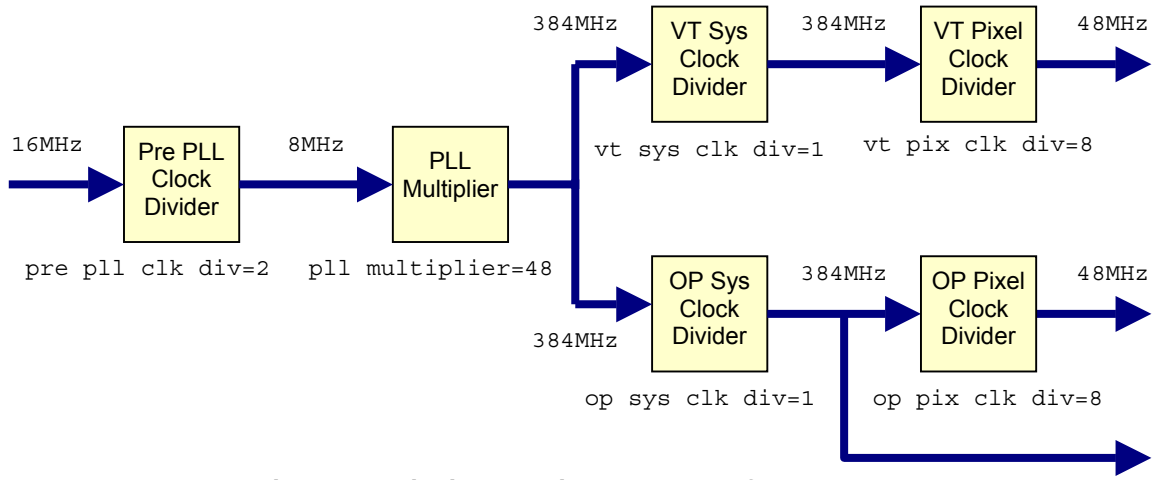


Figure 52: Clock Example: UXGA @ 20fps @ RAW8

5.4.2 Clock Set up Capability Read Only Registers

Register Name	Type	RW	Comment
min_ext_clk_freq_mhz	32-bit IEEE float	RO Static	Minimum external clock frequency Units: MHz
max_ext_clk_freq_mhz	32-bit IEEE float	RO Static	Maximum external clock frequency Units: MHz
min_pre_pll_clk_div	16-bit unsigned integer	RO Static	Minimum Pre PLL divider value
max_pre_pll_clk_div	16-bit unsigned integer	RO Static	Maximum Pre PLL divider value
min_pll_ip_freq_mhz	32-bit IEEE float	RO Static	Minimum PLL input clock frequency Units: MHz
max_pll_ip_freq_mhz	32-bit IEEE float	RO Static	Maximum PLL input clock frequency Units: MHz
min_pll_multiplier	16-bit unsigned integer	RO Static	Minimum PLL multiplier
max_pll_multiplier	16-bit unsigned integer	RO Static	Maximum PLL Multiplier
min_pll_op_freq_mhz	32-bit IEEE float	RO Static	Minimum PLL output clock frequency Units: MHz
max_pll_op_freq_mhz	32-bit IEEE float	RO Static	Maximum PLL output clock frequency Units: MHz

Table 56: Pre PLL and PLL Clock Set-up Capability Registers (Read Only)

Register Name	Type	RW	Comment
min_vt_sys_clk_div	16-bit unsigned integer	RO Static	Minimum video timing system clock divider value
max_vt_sys_clk_div	16-bit unsigned integer	RO Static	Maximum video timing system clock divider value
min_vt_sys_clk_freq_mhz	32-bit IEEE float	RO Static	Minimum video timing system clock frequency Units: MHz

Register Name	Type	RW	Comment
max_vt_sys_clk_freq_mhz	32-bit IEEE float	RO Static	Maximum video timing system clock frequency Units: MHz
min_vt_pix_clk_freq_mhz	32-bit IEEE float	RO Static	Minimum video timing pixel clock frequency Units: MHz
max_vt_pix_clk_freq_mhz	32-bit IEEE float	RO Static	Maximum video timing pixel clock frequency Units: MHz
min_vt_pix_clk_div	16-bit unsigned integer	RO Static	Minimum video timing pixel clock divider value
max_vt_pix_clk_div	16-bit unsigned integer	RO Static	Maximum video timing pixel clock divider value

Table 57: Video Timing Clock Setup Capability Registers (Read Only)

Register Name	Type	RW	Comment
min_op_sys_clk_div	16-bit unsigned integer	RO Static	Minimum output system clock divider value
max_op_sys_clk_div	16-bit unsigned integer	RO Static	Maximum output system clock divider value
min_op_sys_clk_freq_mhz	32-bit IEEE float	RO Static	Minimum output system clock frequency Units: MHz
max_op_sys_clk_freq_mhz	32-bit IEEE float	RO Static	Maximum output system clock frequency Units: MHz
min_op_pix_clk_div	16-bit unsigned integer	RO Static	Minimum output pixel clock divider value
max_op_pix_clk_div	16-bit unsigned integer	RO Static	Maximum output pixel clock divider value
min_op_pix_clk_freq_mhz	32-bit IEEE float	RO Static	Minimum output pixel clock frequency Units: MHz
max_op_pix_clk_freq_mhz	32-bit IEEE float	RO Static	Maximum output pixel clock frequency Units: MHz

Table 58: Output Clock Set up Capability Registers (Read Only)

5.4.3 Clock Frequency Requirements

The aims of the clock frequency requirements are:

- To allow the limited range of the pre PLL clock divider values (1, 2, 4, 6, etc) to keep the PLL input frequency within the **min_pll_ip_freq_mhz** to **2* min_pll_ip_freq_mhz** range.

This is necessary to minimise the variation in the pixel clock frequency across the full range of input clock frequencies.

- Allow the readout rate of the sensor module's full resolution to be smoothly reduced by at least a factor of 2 from the maximum readout rate e.g. from 30fps to at least 15fps.

Thus requirements for the sensor modules clocks are:

- The range for the external clock input frequency is 6.0 MHz to 27.0 MHz

```
min_ext_clk_freq_mhz = 6.0 MHz
max_ext_clk_freq_mhz = 27.0 MHz
```

- The maximum PLL input frequency must be at least twice the minimum PLL input frequency

```
max_pll_ip_freq_mhz >= 2* min_pll_ip_freq_mhz
```

- The maximum PLL output frequency must be at least twice the sum of minimum PLL output frequency and the variation of the PLL output frequency over the full range of external clock frequencies

```
max_pll_op_freq_mhz >= 2*(min_pll_op_freq_mhz + var_pll_op_freq_mhz)
```

or

```
min_pll_op_freq_mhz <= (max_pll_op_freq_mhz/2)-var_pll_op_freq_mhz
```

where the variation in the PLL output frequency is

```
var_pll_op_freq_mhz = 2*min_pll_ip_freq_mhz
```

- The maximum video timing system clock frequency must be at least twice the sum of minimum system clock frequency and the variation of the system clock over the full range of external clock frequencies

```
max_vt_sys_clk_freq_mhz >=
            2*(min_vt_sys_clk_freq_mhz + var_vt_sys_clk_freq_mhz)
```

or

```
min_vt_sys_clk_freq_mhz <= (max_vt_sys_clk_freq_mhz/2)-
                                        var_vt_sys_clk_freq_mhz
```

where the variation in the system clock is

```
var_vt_sys_clk_freq_mhz = (2*min_pll_ip_freq_mhz)/vt_sys_clk_div
```

- The maximum video timing pixel clock frequency must be at least twice the sum of the minimum pixel clock frequency and the variation of the pixel clock over the full range of external clock frequencies

```
max_vt_pix_clk_freq_mhz >=
            2*(min_vt_pix_clk_freq_mhz + var_vt_pix_clk_freq_mhz)
```

or

```
min_vt_pix_clk_freq_mhz <= (max_vt_pix_clk_freq_mhz/2)-
                                        var_vt_sys_clk_freq_mhz
```

where the variation in the pixel clock is

```
var_vt_pix_clk_freq_mhz = (2*min_pll_ip_freq_mhz)/
                                (vt_sys_clk_div*vt_pix_clk_div) )
```

5.5 Overall Video Timing Requirements

The requirements for the sensor modules video timing are:

- 30fps readout for the sensor module's full resolution shutter with a minimum frame banking period of 500us. Other readout rates can also be supported.

This 30fps readout is to minimise effects of motion distortion and image tearing in sensor modules using a rolling blade electronic shutter.

The 1/30th second must be achievable across the full range of external clock frequencies i.e. 6 MHz – 27 MHz

For sensor resolutions whose 30fps bandwidth requirement is beyond the 650Mbps data rate limit specified in the CCP2 specification, the maximum sensor readout rate may be de-rated.

In this case the sensor must also support the fastest frame rate from the following list which respects the 650Mbps maximum data rate of CCP2.

Frame Rate List: 30fps, 25fps, 24fps, 20fps, 15fps, 12.5fps, 12fps, 10fps, 7.5fps, 6.25fps, 6fps, 5fps.

- The readout rate of the maximum image size must be able to be smoothly reduced by a factor of 2 from the maximum readout rate by only changing the PLL multiplier and the video timing/output system clock dividers.
-

The mechanism for varying the readout rate is via the `pre_pll_clk_div`, `pll_multiplier`, `vt_sys_clk_div` and `op_sys_clk_div` registers.

This is necessary to give improved compatibility with the data rate capabilities of different host CCP2 receiver implementations.

- Sensor modules must be able to work down to a 0ms frame blanking time.

Ideally the host receiver should be able to work a 0ms frame blanking time. However it is recognised that in many systems this is may not be either practical or possible to achieve this target.

For compatibility with host receivers requiring more than a 0ms frame blanking time, the frame length of the sensor module is increased to provide the required frame blanking time.

Thus the sensor module's frame length parameter allows the frame blanking time to be 'tuned' to suit the requirements of the host receiver.

- The sensor video timings are based on the image size readout from the pixel array and not the image size within the frame of CCP2 output data.

Coarse and fine integration times are based on the timings for the image readout from the pixel array and not the digitally scaled output.

6 Integration Time and Gain Control

The generic control of SMIA sensors requires standardisation of the sensor interface. In terms of exposure control this affects control of integration time, analogue gain and digital gain.

For this reason the SMIA baseline specification includes a common model for control over sensor integration time, analogue gain and digital gain. The model is designed to be simple to use without imposing complicated or inconvenient restrictions on the design of a SMIA sensor.

6.1 Overview

This chapter defines sensor integration time and both analogue and digital gain control in the SMIA baseline specification. It is split into six sections that define the following aspects:

- Integration time control parameters
- Analogue gain control parameters
- Digital gain control parameters
- Determination of time and gain control parameter limits for a particular SMIA sensor
- Rules governing the ‘consumption’ of changes in time and/or gain settings
- The mechanism that allows synchronisation of time and/or gain parameter changes

6.2 Integration time control

Throughout any one image, all pixels of a compliant SMIA sensor must integrate light for exactly the same amount of time. This amount of time, the 'integration time', is defined using two integer control parameters `coarse_integration_time`, and `fine_integration_time`.

The `coarse_integration_time` parameter sets the number of complete sensor line periods of integration time, `fine_integration_time` sets an additional number of sensor pixel periods of integration time (although fine control is an optional feature).

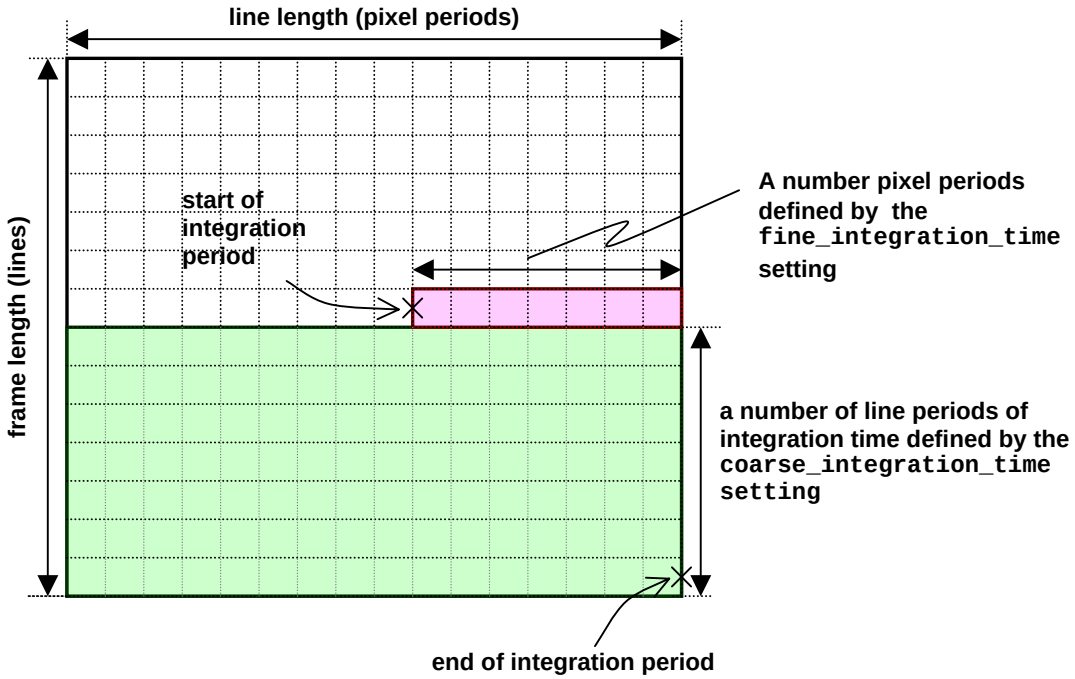
Neither `coarse_integration_time` or `fine_integration_time` parameter employ any form of coding, e.g. a value of 2 written to `coarse_integration_time` sets the number of complete line periods of integration time to 2.

Parameter Name	Type	R/W	Default	Units	Function
`fine_integration_time`	16-bit unsigned integer	read/write	Sensor dependent	sensor pixel periods	The ‘fine’ integration time control
`coarse_integration_time`	16-bit unsigned integer	read/write	Sensor dependent	Sensor line periods	The ‘coarse’ integration time control

Table 59:Integration Time Control Parameters

The total integration time of a SMIA sensor can be calculated using the following formula:

total_integration_time = [(coarse_integration_time * pixel_periods_per_line) + fine_integration_time] * pix_clk_period



The relationship between the exact timing of a change to either integration time control and the resultant effect on the image stream is covered in section 6.4.

Fine integration time control is an option in SMIA. All SMIA sensors must advertise whether they support it via the static `integration_time_capability` parameter.

6.2.1 Absolute Integration Time And Flicker Correction

Sampled imaging system can suffer from 'flicker' problems when imaging subjects illuminated by oscillating light sources (e.g. fluorescent lights). Such problems can be alleviated by use of integration times that are 'largely integer multiples' of the period of the lighting flicker.

To allow flicker avoidance techniques to be used with a SMIA sensor the exposure control system must be able to set absolute integration periods. It must, therefore, be possible to determine the absolute pixel period from details of the external clock frequency and the clock system set-up parameters.

6.3 Analogue gain control

Analogue gain control is an area where sensors from different manufactures often differ. With this in mind a general approach has been taken to gain control in the baseline SMIA specification with the aim of supporting the majority of analogue gain architectures.

Five analogue gain controls are defined for a SMIA sensor. One parameter offers control over analogue gain that is common to all pixels, the other four control possible Bayer-channel dependent gain.

Parameter Name	Type	R/W	Default	Function
analogue_gain_code_global	16-bit unsigned integer	RW	Sensor dependent	A control for analogue gain that is applied to all Bayer channels
analogue_gain_code_red	16-bit unsigned integer	RW	Sensor dependent	A control for analogue gain of red pixels
analogue_gain_code_greenR	16-bit unsigned integer	RW	Sensor dependent	A control for analogue gain of the green pixels on a row that also contains red pixels
analogue_gain_code_blue	16-bit unsigned integer	RW	Sensor dependent	A control for analogue gain of blue pixels
analogue_gain_code_greenB	16-bit unsigned integer	RW	Sensor dependent	A control for analogue gain of the green pixels on a row that also contains blue pixels

Table 60:Analogue Gain Control Parameters

If a sensor module has per channel analogue gain capability, its default (baseline) behaviour must be as if it has a single global analogue gain. If the host receiver AEC/AWB supports analogue gain per channel the host can discover this via the gain capability register and then enable this feature.

The gain models supported is advertised by a sensor via the analogue_gain_capability parameter

Parameter Name	Type	R/W	Default	Function
analogue_gain_capability parameter	16-bit unsigned integer	RO	Sensor dependent	Describes the sensor analogue gain capabilities 0 – global gain only 1 – separate Bayer gains only

Table 61:Analogue Gain Capability Parameters

The global and per channel analogue gain control parameter uses the same coding scheme. The relationship between the integer gain parameter and the resulting gain is given by the following equation:

$$gain = \frac{m_0 x + c_0}{m_1 x + c_1}$$

where ‘x’ is the integer analogue gain control parameter and m_0, c_0, m_1 & c_1 are sensor specific constants advertised by the sensor. These constants are static parameters and for any one sensor either m0 or m1 must be zero. The full gain equation therefore reduces to either

$$gain = \frac{c_0}{m_1 x + c_1} \text{ , or } gain = \frac{m_0 x + c_0}{c_1}$$

Parameter Name	Type	R/W	Default	Function
analogue_gain_type	16-bit signed integer	RO	0	The analogue gain coding type (which must default to 0 for baseline SMIA)
analogue_gain_constant_m0	16-bit signed integer	RO	Sensor dependent	The m0 constant used in the analogue gain control coding/decoding (either this or m1 must be zero)
analogue_gain_constant_c0	16-bit signed integer	RO	Sensor dependent	The c0 constant used in the analogue gain control coding/decoding
analogue_gain_constant_m1	16-bit signed integer	RO	Sensor dependent	The m1 constant used in the analogue gain control coding/decoding (either this or m0 must be zero)
analogue_gain_constant_c1	16-bit signed integer	RO	Sensor dependent	The c1 constant used in the analogue gain control coding/decoding

Table 62:Analogue Gain Coding/Decoding Constants

The relationship between the exact timing of a change to the analogue gain control and the resultant effect on the image stream is covered in Section 6.4.

Limits on the use of the analogue gain control are also advertised by the sensor, for details see Section 6.7.

6.4 Digital Gain control

Provision of a digital gain stage is not a requirement of the SMIA baseline specification, however, a SMIA sensor must advertise whether it includes digital gain capability via the static digital_gain_capability parameter.

6.4.1 Digital gain control parameters

Digital gain of the four Bayer channels is controlled separately using the four parameters shown in the following table.

Parameter Name	Type	R/W	Default	Function
digital_gain_red	unsigned 16-bit fixed point binary numbers (8 integer bits and 8 fractional)	RW	0x0100	The red channel digital gain control
digital_gain_greenR	unsigned 16-bit fixed point binary numbers (8 integer bits and 8 fractional)	RW	0x0100	The green channel digital gain control for the green pixels on rows that also contain red pixels

Parameter Name	Type	R/W	Default	Function
digital_gain_blue	unsigned 16-bit fixed point binary numbers (8 integer bits and 8 fractional)	RW	0x0100	The blue channel digital gain control
digital_gain_greenB	unsigned 16-bit fixed point binary numbers (8 integer bits and 8 fractional)	RW	0x0100	The green channel digital gain control for the green pixels on rows that also contain blue pixels

Table 63:Digital Gain Control Parameters

6.4.1.1 Control parameter change timing

The exact timing of the effect on the output pixel stream of changes in the digital gain parameters must be managed by the sensor.

If necessary the sensor will delay a gain change so as to affect the output pixel stream at a frame boundary and, if grouped with other changes, delay it so that all the grouped changes affect the output pixel stream at the same boundary. See Section 6.5 for more details.

6.4.2 Digital gain parameter limits

The range and precision of the digital gain controls is advertised by the sensor via the three static parameters: digital_gain_min, digital_gain_max and digital_gain_step_size.

The value advertised as digital_gain_step_size represents the smallest step in digital gain supported by the sensor. To be a valid SMIA sensor both digital_gain_min and digital_gain_max must be integer multiples of digital_gain_step_size.

Parameter Name	Type	R/W	Default	Function
digital_gain_min	unsigned 16-bit fixed point binary numbers (8 integer bits and 8 fractional)	RO	Sensor dependent	The minimum valid limit of the digital gain control parameters
digital_gain_max	unsigned 16-bit fixed point binary numbers (8 integer bits and 8 fractional)	RO	Sensor dependent	The maximum valid limit of the digital gain control parameters
digital_gain_step_size	unsigned 16-bit fixed point binary numbers (8 integer bits and 8 fractional)	RO	Sensor dependent	Defines the resolution of the digital gain control parameters

Table 64:Digital Gain Limit & Precision Parameters

As the same three parameters are used to define all four gain controls the range and precision of the four gain parameters must be the same.

The four gain control parameters and the static range/precision parameters all use the same unsigned 16-bit fixed-point binary coding scheme where the top eight bits are integer (i.e. 0x0100 = 1.0f, 0x0480 = 4.5f).

6.4.2.1 Null parameter limit set-up

Where a sensor does not include digital gain the `digital_gain_capability` parameter should be set accordingly and the static min/max parameters should be set to indicate unity (1.0f).

6.4.3 Digital gain & black data level

It is important that to note that the sensor is responsible for ensuring that use of digital gain does not affect the pixel data black level.

6.5 Re-timing of Integration Time and Gain Controls

The exact time at which changes to certain parameters take effect must be controlled both to ensure that each frame of image data produced has consistent settings and that changes in groups of related parameters can be synchronised. To this end, a compliant sensor must obey the following three 'parameter consumption' rules when reacting to changes in these 'retimed' parameters (see the register map document for details of which parameters are 'retimed').

6.5.1 Changes retimed to frame boundaries

Each change to one of these parameters must be effected so as to affect the output image stream on a frame boundary – regardless of the actual timing of the parameter change messages (i.e. change without causing a discontinuity within any image).

6.5.2 Grouped changes

The sensor must recognise a system by which a set of changes to these parameters can be grouped by the host system and apply these changes in such a way that not only do they effect an apparent change in the output pixel stream at a field boundary, but that the effect of each of the changes in the group effect the output pixel stream at the same field boundary.

A group of parameter changes is marked by the host using a dedicated Boolean control parameter, `grouped_parameter_hold`. Any changes made to 'retimed' parameters while the `grouped_parameter_hold` signal is in the 'hold' state should be considered part of the same group. Only when the `grouped_parameter_hold` control signal is moved back to the default 'no-hold' state should the group of changes be executed.

Parameter Name	Type	R/W	Default	Function
`grouped_parameter_hold`	8-bit unsigned integer	RW	0 (no hold)	Set to envelope a series of parameter changes as a group of changes that should be made so as to effect the output stream on the same frame boundary

Table 65:Grouped change control Parameter

6.5.3 Update speed requirement

To ensure a minimum response speed from the system a maximum time in which a sensor must implement grouped changes is defined. The constraint is that the maximum number of output field boundaries between the end of a grouped message transaction and the subsequent change in output image parameters must be less than or equal 6. The choice of 6 allows the host to consider a lack of update in 8 images to be a failure (allowing some margin to for loose control over exact message timing).

While this limit defines a minimum response speed, a sensor designed should, however, aim for a much faster turnaround time. A delay of 2 field boundaries should be considered more respectable.

6.6 Control synchronisation

Successful closed loop control of a SMIA sensor requires the timing relationship between changes in sensor control parameters and the effect on the output image stream to be taken into account.

Rather than defining a detailed model of exact delays between a parameter change and the corresponding change to the image stream, the SMIA specification takes a simpler approach that is designed to be both more flexible and more robust when used with asynchronous hosts (or a host with other real-time responsibilities).

Control system synchronisation with a SMIA sensor relies on the inclusion of status information within each image frame. Key to the success of the system is the requirement that in every image frame each piece of status information that defines the setting of a control parameter _must_ reflect the setting of that parameter that was used in the generation of that frame (irrespective of consumption delays).

Using this scheme a control system can always determine which parameter settings were used to generate any frame. When the host sends a message to the sensor requesting a change in a control parameter the host can detect at which point the change takes effect by monitoring the status information.

For details of parameters that are included in this scheme see **the CCI register map chapter and the data format chapter.**

6.7 Time & Gain Parameter Limit Discovery

Although a simple profile SMIA sensor must obey a model of integration time and analogue gain control there is flexibility in the range of control values that it must support.

In order that the host system can identify these limits automatically the sensor must make the necessary information available via the serial interface and also include it in the sensor status line.

The SMIA 'exposure control parameter limit discovery process' is designed so that the information that define the recommended parameter limits for a particular device are independent of all controls settings and therefore can be fetched early in the set-up process and considered static.

The only downside of this approach is that some of the limits are defined relative to other parameters (e.g. the maximum integration_time_line setting is defined relative to the number of lines in the field). These relative limits must therefore be re-evaluated on each change of the video timing set-up.

Parameters that define minimum and maximum recommended settings for each of the integration time controls and the gain control can be read from sensor serial interface registers. Details of these registers are given in the table below.

Parameter Name	Type	R/W	Default	Function
fine_integration_time_min	16-bit unsigned integer	RO	Sensor dependent	The minimum recommended setting for the fine_integration_time control
fine_integration_time_max_margin	16-bit unsigned integer	RO	Sensor dependent	A parameter that can be used to defined the maximum recommended setting for the fine integration time control
coarse_integration_time_min	16-bit unsigned integer	RO	Sensor dependent	The minimum recommended value for the coarse integration time control
coarse_integration_time_max_margin	16-bit unsigned integer	RO	Sensor dependent	A parameter that can be used to determine the maximum recommended setting for the coarse integration time control
analogue_gain_code_min	16-bit unsigned integer	RO	Sensor dependent	The minimum recommended setting for the analogue gain control
analogue_gain_code_max	16-bit unsigned integer	RO	Sensor dependent	The maximum recommended setting for the analogue gain control
analogue_gain_code_step	16-bit unsigned integer	RO	Sensor dependent	The precision of the analogue gain control

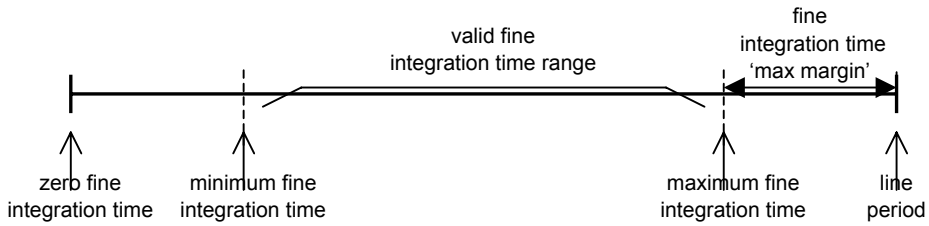
Table 66:Integration time and gain parameter limits

It is possible for a SMIA sensor to be compliant whilst offering no control over analogue gain. This is achieved by setting both minimum and maximum recommended limits for the analogue gain control to be the same as the default value of the register. (For details see section 5.)

Limits for the analogue gain controls are simply defined by absolute minimum and maximum recommended register settings: analogue_gain_code_min and analogue_gain_code_max

The minimum recommended settings for both integration time controls are also defined (registers fine_integration_time_min and coarse_integration_time_min).

The maximum settings are, however, defined relative to other control parameters. At any time the recommended maximum fine and coarse integration time limits can be determined using the following two equations:



coarse_integration_time_max = line_periods_per_field - coarse_integration_time_max_margin

fine_integration_time_max = pixel_periods_per_line - fine_integration_time_max_margin

The maximum time limits are described in this way so that the sensor dependent parameters can be static and thus only be read once by the host (if instead the sensor advertised the current maximum legal values the host would have to re-read them each time the line or field length were altered).

6.7.1.1 Operation outside limits

As the behaviour of a SMIA sensor when control parameters are set outside the recommended limits need not be defined, a SMIA host should not operate the sensor out with them.

7 Colour Matrix

In order to generate common image data from different SMIA sensors, the colour space conversion required to transform data from the colour space of a particular SMIA sensor to some standard colour space must be understood.

To this end, the SMIA sensor register map includes some static, read only parameters that should be used by sensor manufacturers to advertise a recommended colour space conversion matrix. As the parameters are static a host need only read them once.

The matrix should be derived so as to transform pixel data from the native sensor RGB colour space to the colour space of sRGB (ITU-R BT.709 reference primaries and a white point of D_{65}, see http://www.color.org/sRGB.html.

Although it is strongly advised that a recommended matrix should be advertised by a sensor, if for some reason no a manufacturer does not want to include one they should set the static values in the matrix parameter registers to represent an identity matrix. Use of this convention allows the simplest of sensor hosts to 'blindly' use the recommended matrix without corrupting the data.

The following equation shows how the matrix can be used and how the 9 elements of the 3x3 matrix are named.

$$\begin{bmatrix} r \\ g \\ b \end{bmatrix}_{sRGB} = \begin{bmatrix} RedInRed & GrnInRed & BluInRed \\ RedInGrn & GrnInGrn & BluInGrn \\ RedInBlu & GrnInBlu & BluInBlu \end{bmatrix} x \begin{bmatrix} r \\ g \\ b \end{bmatrix}_{sensor}$$

The matrix elements are included in the sensor register map as 16-bit fixed-point parameters (16-bit signed iReals) described more fully in the table below. For details of the 8-bit registers that these parameters are divided between see CCI register map chapter .

Parameter Name	Type	R/W	Default	Function
matrix_element_RedInRed	16-bit signed iReal	RO Static	Sensor dependent	The recommended amount of the sensor red signal to be included in the output red signal
matrix_element_GreenInRed	16-bit signed iReal	RO Static	Sensor dependent	The recommended amount of the sensor green signal to be included in the output red signal
matrix_element_BlueInRed	16-bit signed iReal	RO Static	Sensor dependent	The recommended amount of the sensor blue signal to be included in the output red signal
matrix_element_RedInGreen	16-bit signed iReal	RO Static	Sensor dependent	The recommended amount of the sensor red signal to be included in the output green signal
matrix_element_GreenInGreen	16-bit signed iReal	RO Static	Sensor dependent	The recommended amount of the sensor green signal to be included in the output green signal
matrix_element_BlueInGreen	16-bit signed iReal	RO Static	Sensor dependent	The recommended amount of the sensor blue signal to be included in the output green signal
matrix_element_RedInBlue	16-bit signed iReal	RO Static	Sensor dependent	The recommended amount of the sensor red signal to be included in the output blue signal
matrix_element_GreenInBlue	16-bit signed iReal	RO Static	Sensor dependent	The recommended amount of the sensor green signal to be included in the output blue signal
matrix_element_BlueInBlue	16-bit signed iReal	RO Static	Sensor dependent	The recommended amount of the sensor blue signal to be included in the output blue signal

Table 67: Suggested Colour Space Conversion Matrix Parameters

8 Test Modes

To aid the debugging of systems that include SMIA sensors the specification includes a number of test patterns that each sensor must be able to produce.

8.1 Full frame deterministic test patterns

Two types of full frame deterministic test patterns are defined. Most are Bayer test patterns more suitable for some tests than real image data and are injected early in the sensor data path. The only exception to this is a test pattern that is intended to test sensor-host link integrity, the data in this pattern is not Bayer data and it is injected just prior to CCP2 framing.

Use of these full frame test patterns is controlled by the `test_pattern_mode` parameter. The following table shows all the defined parameter settings.

Parameter Name	Type	R/W	Coding	Function
`test_pattern_mode`	16-bit unsigned integer	RW	0 – no pattern (default) 1 – solid colour 2 – 100% colour bars 3 – fade to grey' colour bars 4 - PN9 5 – 255 Reserved 256-65535 – Manufacturer Specific	Controls the output of the test pattern module

Table 68:The main full frame test pattern control parameter

In both the default parameter state and in any undefined parameter states normal array data should be output rather than a test pattern. The individual test patterns are described later in this chapter.

8.1.1 Scaled & cropped output

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

8.1.2 Solid colour mode

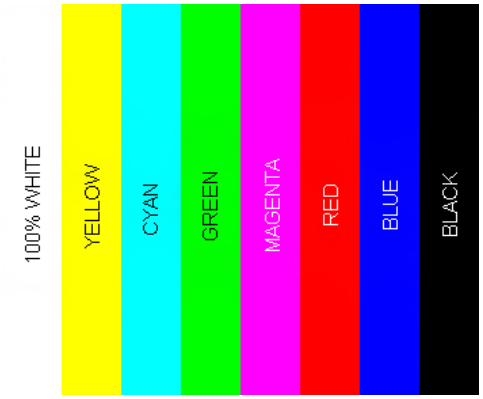
In the 'solid colour' test pattern mode all pixel data is replaced with fixed Bayer test data that is defined by the four 'test data colour' parameters, these are described in the following table.

Parameter Name	Type	R/W	Default	Function
test_data_red	16-bit unsigned integer	RW	0	The test data used to replace red pixel data
test_data_greenR	16-bit unsigned integer	RW	0	The test data used to replace green pixel data on rows that also have red pixels
test_data_blue	16-bit unsigned integer	RW	0	The test data used to replace blue pixel data
test_data_greenB	16-bit unsigned integer	RW	0	The test data used to replace green pixel data on rows that also have blue pixels

Table 69:Test data colour parameters

8.1.3 100% colour bars pattern mode

In the '100% colour bar' test pattern mode all pixel data is replaced with a Bayer version of an 8-bar colour bar pattern.



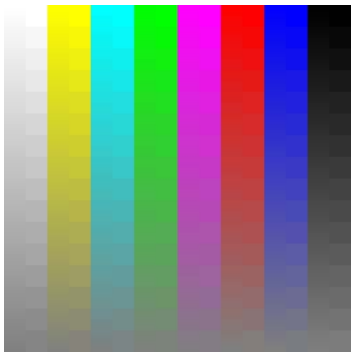
In each bar all pixels are either 0% or 100% full scale (e.g. 100/0/100/0 bars).

8.1.3.1 Pattern size

Where possible each bar should occupy exactly 1/8th of the output image width. Where the output image width is not a multiple of 8 pixels the bar size should be rounded down to the nearest integer number of pixels and the pattern allowed to repeat at the right hand side of the image.
The pattern should always occupy the full output image height.

8.1.4 'Fade to grey' colour bar mode

In the ‘fade to grey colour bar’ test pattern mode all pixel data is replaced with a colour bar that fades vertically from 100% colour bars to mid grey. The 'fade to grey' colour bar pattern is designed to exercise more of the colour space than 100% bars whilst still requiring minimal hardware overhead. The following figure gives an indication of the pattern (although the pattern is generated as Bayer data).



The pattern is made up of 8 vertical bars that fade vertically from one of the 100% colour bar colours towards a mid-grey at the bottom.

The bars follow the same order as standard colour bars.

Each of the bars is sub-divided vertically into a left hand side that contains a smooth gradient and a right hand size that contains a quantised version.

The aim of the quantised portion is to offer areas of flat-field Bayer data that should be large enough to result in known data values even after de-mosaic (independently of the de-mosaic algorithm).

To ensure maximum dynamic range in the quantised data, the LSBs of the quantised data should be generated by copying the MSBs of the unquantised data (rather than forcing them to 0).

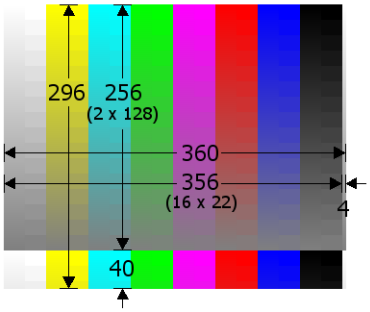
8.1.4.1 Pattern size

Ideally each of the 16 sub-bars will occupy 1/16th of the Bayer image width. Where the output width is not a multiple of 16 pixels the sub-bar size should be rounded down to the nearest integer number of pixels and the pattern allowed to repeat in the remaining pixels at the right-hand side of the image.

As the data values of the gradient are most efficiently generated directly from a row counter, arbitrary vertical size is not always convenient for the fade to grey bars.

As such, the height of each bar should always be a multiple of 128 pixels, the pattern should be allowed to repeat on any additional lines of image.

An example of a 360x296 pattern is shown in next page.



8.1.5 PN9 mode

In the ‘PN9’ test pattern mode all pixel data is replaced with data from an internally generated 512-bit pseudo-random ‘PN9’ sequence.

The PN9 test pattern is included to ease testing of sensor-link integrity (measurement of bit error rate etc). PN9 linear feedback shift register has the polynomial X^9+X^5+1 in Fibonacci type notation (Figure 53).

The PN9 sequence generator should be reset so as to start the sequence in a known state at the first replaced pixel of each frame.

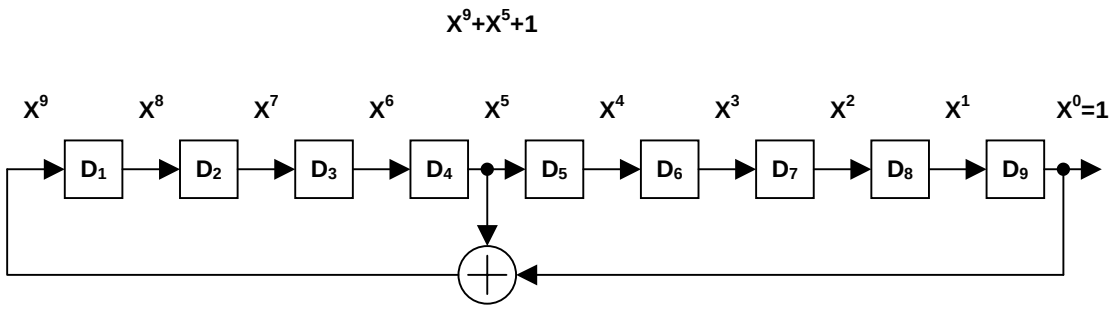


Figure 53: PN9 Linear Feedback Shift Registers

8.2 Test Cursors

In addition to the generation of full frame deterministic test patterns, a baseline SMIA sensor can superimpose simple ‘cursors’ on the image.

The cursors are generated by replacing Bayer pixel data with fixed Bayer data within narrow vertical and/or horizontal bands of the image. Injection of the test cursors must be arranged such that the cursors can be superimposed on top of the full frame test patterns as well as array image data.

Two cursors are defined, one vertical cursor and one horizontal. The four parameters described in the following table are used to control the cursors. The position and width of each cursor can be controlled manually. Each cursor can be inhibited by setting its width parameter to zero. The vertical cursor can be put into an ‘automatic position’ mode where it’s position reflects the current value of the sensor frame counter.

Parameter Name	Type	R/W	Default	Function
horizontal_cursor_width	16-bit unsigned integer	RW	0	Defines the width of the horizontal cursor (in pixels)
horizontal_cursor_position	16-bit unsigned integer	RW	0	Defines the top edge of the horizontal cursor
vertical_cursor_width	16-bit unsigned integer	RW	0	Defines the width of the vertical cursor (in pixels)
vertical_cursor_position	16-bit unsigned integer	RW	0	Defines the left hand edge of the vertical cursor. A value of 0xFFFFswitches the vertical cursor into automatic mode where it automatically advances every frame

Table 70: Test Cursor Control Registers

When in its automatic mode the position of the vertical cursor is incremented every frame – the initial position of the automatic cursor is undefined.
As the size of the frame counter and the image width are not closely coupled in the SMIA specification some rules must be set to define the exact nature of the relationship.

```
nFrameCounterBits = 8;
nBitsUsedForCursor = (int) log2(iArrayWidthInPixels);
iPixelsPerStep = 1;
if (nBitsUsedForCursor > nFrameCounterBits) {
        iPixelsPerStep = nBitsUsedForCursor – nFrameCounterBits;
        nBitsUsedForCursor = nFrameCounterBits;
}
```

The four registers used to define the output data in solid colour mode also define the Bayer data used for the image cursors.

9 Image Scaling

9.1 Introduction

Dependent on which profile level a sensor module is compliant with it many have one of 3 different levels of image scaling capability.

- Profile Level 0 - None
- Profile Level 1 - Horizontal Scaler
- Profile Level 2 - Full (Horizontal & Vertical) Scaler

The image scaling function within the sensor module provides a flexible way of generating lower resolution full field of view image data, at a reduced data rates, for viewfinder and video applications.

The scalers must be able to scale the full resolution of the sensor module down to within 10% of a the target image size (the smallest output size is 256x192). This flexibility means that SMIA sensor modules can support a wide range of LCD viewfinder sizes and different codec resolutions

A 10% crop is deemed to be the maximum acceptable error in the field of view of the sensor output image.

To for fill this is requirement the down scale factor must be programmable in steps of 1/16th.

The host should read `scaling_capability` register to determine the sensor module's image scaling capabilities.

Register Name	Type	RW	Comment
`scaling_capability`	16-bit unsigned integer	RO Static	0 – None 1 – Horizontal 2 – Full (Horizontal & Vertical)

Table 71: Image Scaling Capability Register (Read Only and Static)

Figure 54 illustrates some of the input pixel array sizes and output viewfinder sizes.

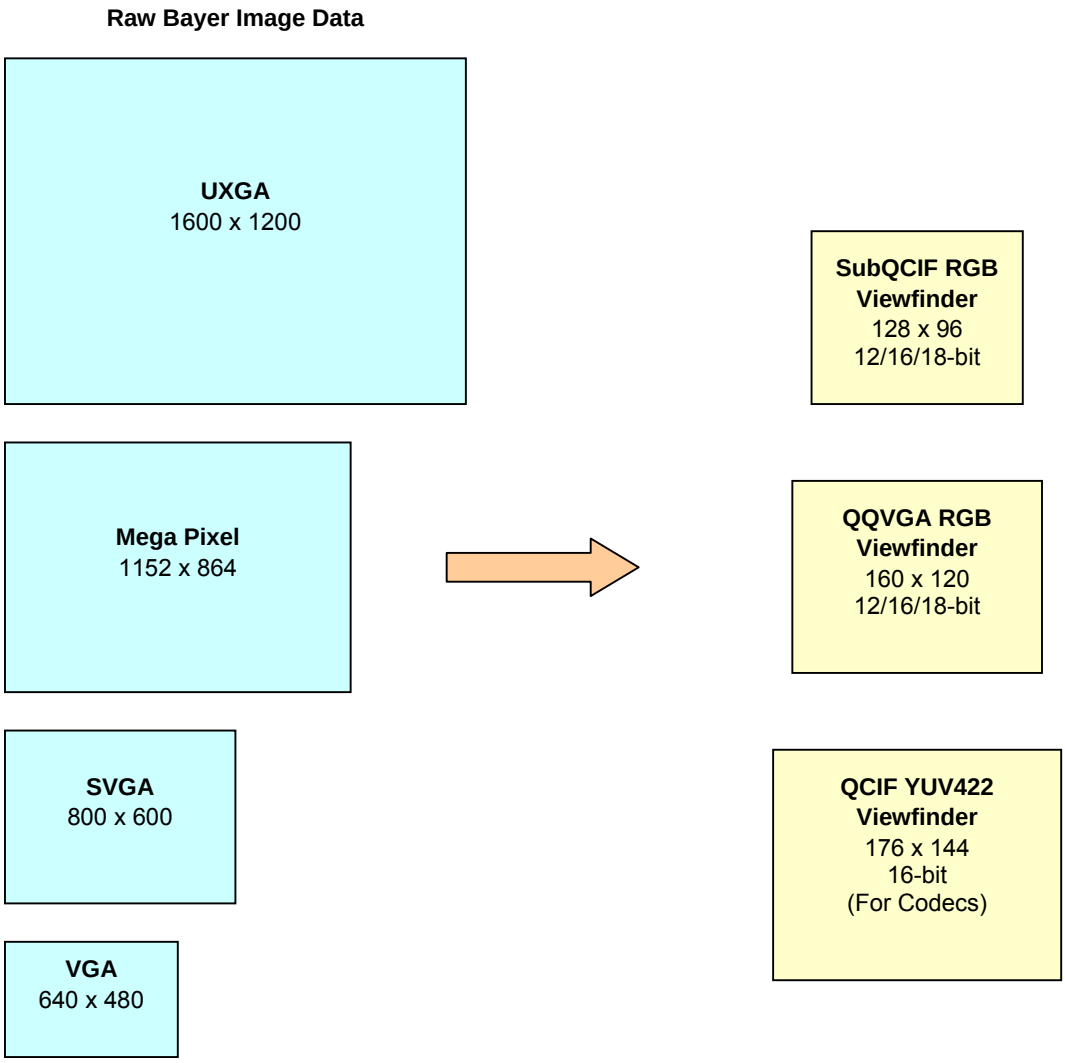


Figure 54: Raw Bayer Data Sizes vs. Example Output Sizes

Figure 55: Profile Level 0 – Both Software Horizontal and Vertical Scaling on Host system

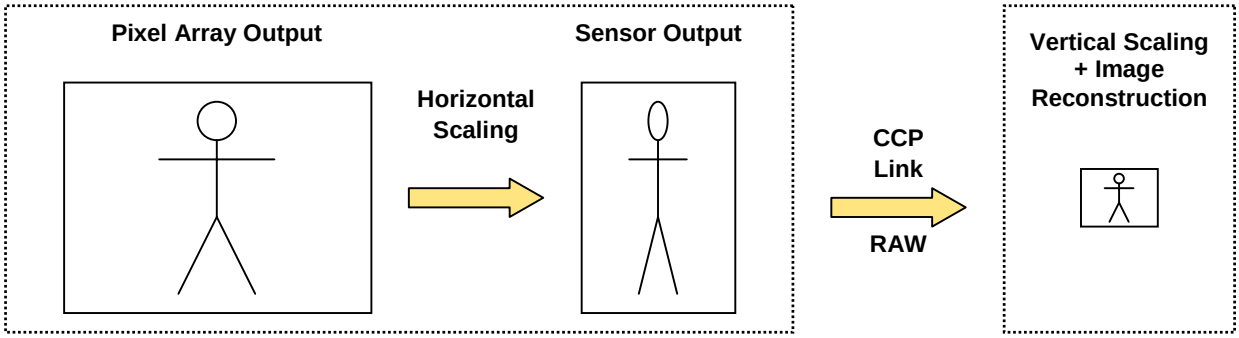


Figure 56: Profile Level 1 - Horizontal Scaling in Sensor Module, Software Vertical Scaling on Host System

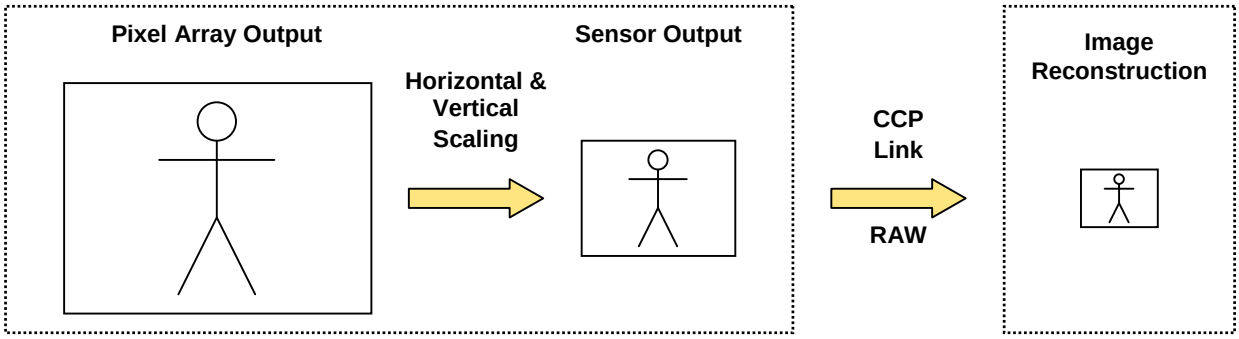


Figure 57: Profile Level 2 - Both Horizontal and Vertical Scaling on Sensor Module

9.2 Scaler Quality

Sub-sampling the image data to generate the scaled output is not acceptable.

The scaler must support 2 options for the spatial sampling of the scaled image data:

- Bayer Sampled scaled image data.

Bayer Sampled scaled image data is defined in Figure 58

- Co-sited scaled image data.

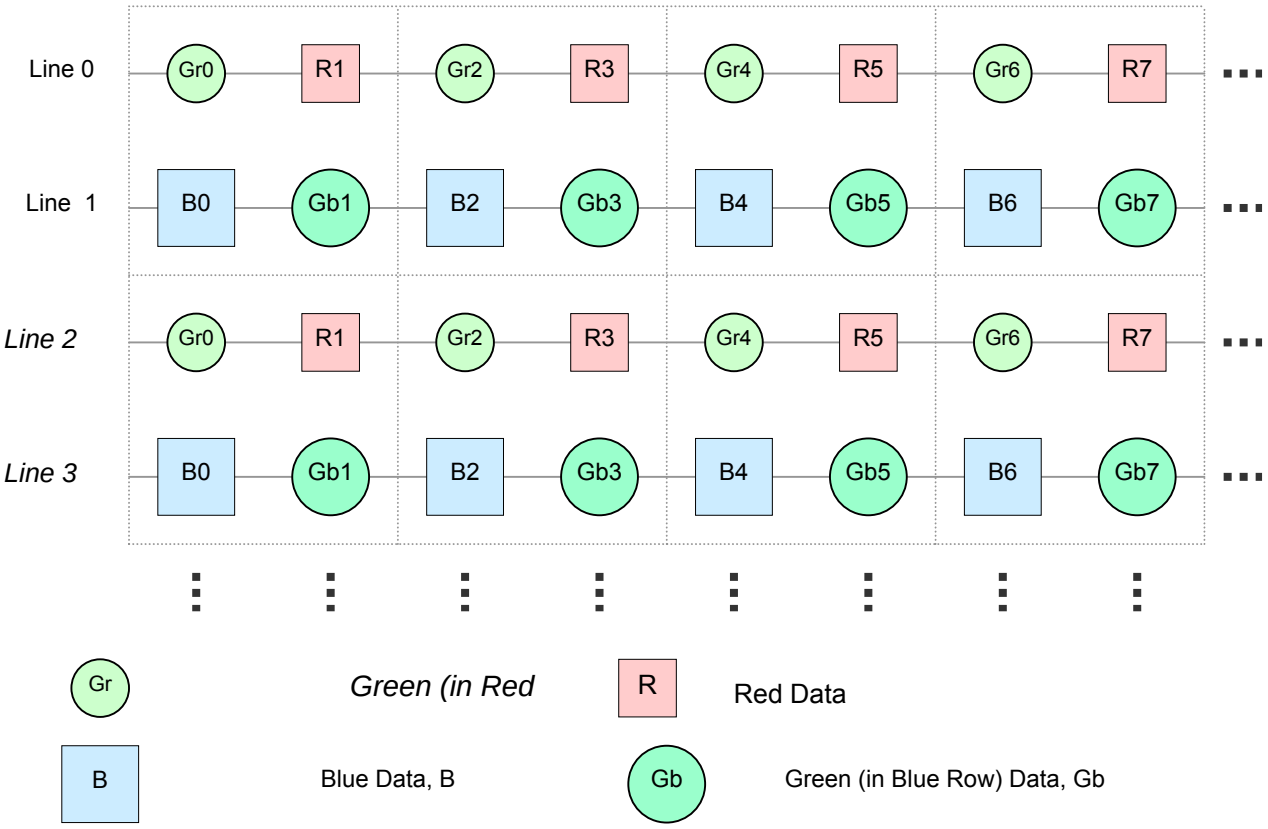
Co-sited horizontally scaled image data is defined in Figure 59
Co-sited fully scaled image data is defined in Figure 60

The spatial sampling mode is controlled by the `spatial_sampling` register (Table 73).

Readout Order:

Line 0	Gr0	R1	Gr2	R3	Gr4	R5	Gr6	R7
Line 1	B0	Gb1	B2	Gb3	B4	Gb5	B6	Gb7
Line 2	Gr0	R1	Gr2	R3	Gr4	R5	Gr6	R7
Line 3	B0	Gb1	B2	Gb3	B4	Gb5	B6	Gb7

Spatial Sampling:



Output Order:

Line 0	Gr0	R1	Gr2	R3	Gr4	R5	Gr6	R7
Line 1	B0	Gb1	B2	Gb3	B4	Gb5	B6	Gb7
Line 2	Gr0	R1	Gr2	R3	Gr4	R5	Gr6	R7
Line 3	B0	Gb1	B2	Gb3	B4	Gb5	B6	Gb7

Figure 58: Bayer Sampled Scaled Image Data

Readout Order:

Line 0	Gr0	R1	Gr2	R3	Gr4	R5	Gr6	R7
Line 1	B0	Gb1	B2	Gb3	B4	Gb5	B6	Gb7
Line 2	Gr0	R1	Gr2	R3	Gr4	R5	Gr6	R7
Line 3	B0	Gb1	B2	Gb3	B4	Gb5	B6	Gb7

Spatial Sampling:

Output Order:

Line 0	Gr0	R0	Gr1	R1	Gr2	R2	Gr3	R3
Line 1	B0	Gb0	B1	Gb1	B2	Gb2	B3	Gb3
Line 2	Gr0	R0	Gr1	R1	Gr2	R2	Gr3	R3
Line 3	B0	Gb0	B1	Gb1	B2	Gb2	B3	Gb3

Figure 59: Co-sited Horizontally Scaled Image Data

Readout Order:

Line 0	Gr0	R1	Gr2	R3	Gr4	R5	Gr6	R7
Line 1	B0	Gb1	B2	Gb3	B4	Gb5	B6	Gb7
Line 2	Gr0	R1	Gr2	R3	Gr4	R5	Gr6	R7
Line 3	B0	Gb1	B2	Gb3	B4	Gb5	B6	Gb7

Spatial Sampling:

Output Order:

Line 0	Gr0	R0	Gr1	R1	Gr2	R2	Gr3	R3
Line 1	B0	Gb0	B1	Gb1	B2	Gb2	B3	Gb3
Line 2	Gr0	R0	Gr1	R1	Gr2	R2	Gr3	R3
Line 3	B0	Gb0	B1	Gb1	B2	R2	B3	Gb3

Figure 60: Co-sited Fully (H&V) Scaled Image Data

9.3 Image Scaling for Software Viewfinder Implementations

For the software viewfinder implementation there are two scaling steps (Figure 61)

- Horizontal
- Vertical

The sensor module performs the horizontal scaling while the vertical scaling may either be performed in software or by the sensor module if it has full image scaling capability.

The readout order of the scaled output image data must be in the Bayer readout order e.g. Green, Red, Green, Red, etc or Blue, Green, Blue, Green, etc. As with the non -scaled output modes the Bayer readout order changes with the horizontal mirror and vertical flip options.

The scaler output is termed quasi Bayer data as while the data is output in a Bayer readout order, the scaling means that the spatial sampling of the pixel data is not the same as for “true” Bayer data.

The visible quasi Bayer data output by the horizontal scaler is twice the width of the target image size e.g. for a QQVGA viewfinder the horizontal scaler output is 320 pixels (= 2*160).

Similarly for the full scaler the output is twice the size of the target image size e.g. for a QQVGA viewfinder the full scaler output is 320 x 240 pixels (= 2*160 x 2*120).

This enables the use of simple viewfinder colour reconstruction algorithms to translate the scaled quasi Bayer data to display-formatted data.

For example in the case of down scaling a VGA image to a QQVGA viewfinder image the software takes the 320 x 488 horizontal scaled output image data and uses simple reconstruction algorithms to generate RGB pixel data (Figure 61). It have to be noticed, when programmable image size is used the software usually takes 320 x 480 horizontal scaled image data.

In this particular case due to the simple image reconstruction algorithms used for the software viewfinder implementations there is no requirement for additional border columns.

The horizontal scaler function **only** affects the horizontal readout of the sensor module. The vertical readout order of the sensor module does **not** change when the horizontal scaler function is enabled.

The host system must update the `x_output_size` and `y_output_size` parameters as the scaler parameters change to ensure that only valid scaled image data is output from the sensor module.

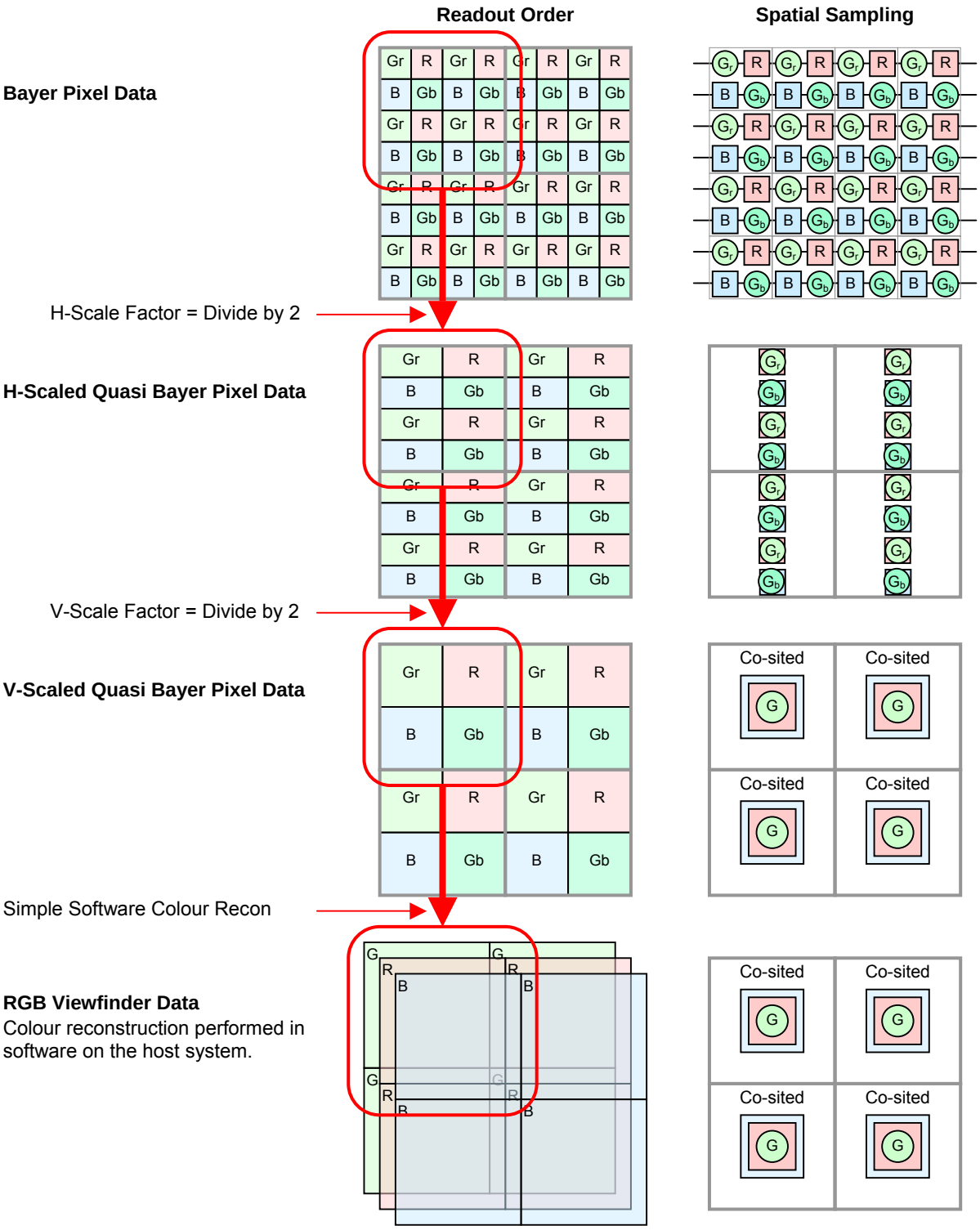


Figure 61: Overview of Viewfinder Colour Reconstruction Flow (Co-sited Example)

9.4 Down Scaler Factor

The scaler step size (downscale factor) is programmable from 1.0 up to the scaler step size required to downscale the maximum addressable width of the pixel array down to an output image width of 256 pixels.

The down scaler factor is controlled by an M/N ratio. M is >= 16 and N is fixed at 16.

$$down\_scale\_factor = \frac{scale\_m}{scale\_n} = \frac{scale\_m}{16}$$

This single down scale factor is used by both the horizontal and vertical scalars.

9.5 Full Image Scaler

The full scaler must contain a smooth horizontal scaler and a smooth vertical scaler.

For the full scaler there are 2 modes of operation

- Horizontal Scaler Only
- Full (Horizontal & Vertical) Scaler

This provides backward compatibility with hosts that are designed only to support sensor module with horizontal scalers.

To provide a wider range of data rate reduction options the full image scaler must be able to reduce the data rates in both the horizontal and vertical directions. This maybe achieved by the use of a FIFO between video timing and output clock domains (Figure 62).

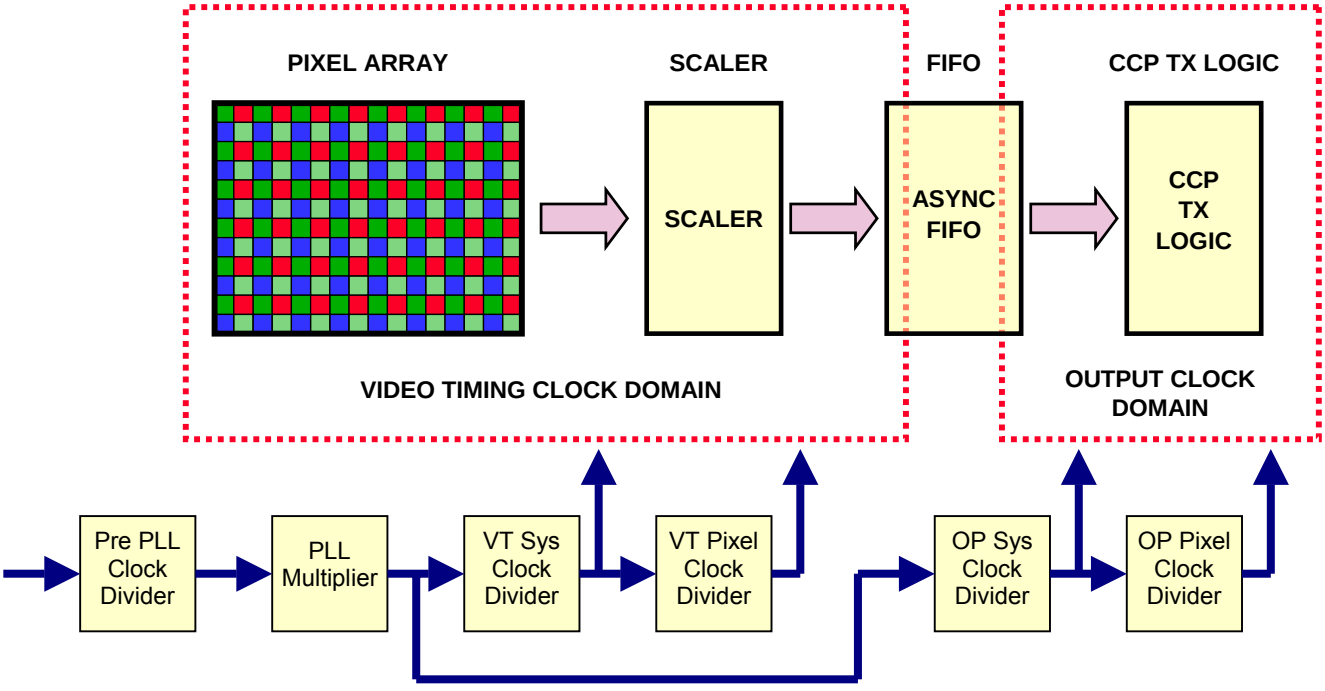


Figure 62: Full Scaler Block Diagram

9.6 Control Registers

The horizontal and vertical scalers are programmable via two parameters (Table 71 and Table 73).

- Scaler mode – Off / Horizontal Only/ Full
- Scaler down scale factor (Down-scale factor)

There is single scale factor for the both the horizontal and vertical scalers.

All of the scaler read/write registers (Table 73) must be re-timed within the sensor module to the start of frame boundary to ensure that the-scaler values are consistent within a frame of image data.

The minimum, maximum and resolution for the M, N components of the down scale and phase parameters are reported via the image scaling capability registers (Table 74).

Table 72 shows some example downscale scale factors.

Pixel Array Format	QQVGA (320x240)		SubQCIF (256x192)	
VGA (648x488)	÷ 2	M=32, N=16	÷ 2.5	M=40, N=16
SVGA (808x608)	÷ 2.5	M=40, N=16	÷ 3	M=48, N=16
1 MP (1160x872)	÷ 3.5	M=56, N=16	÷ 4.5	M=72, N=16
SXVGA (1288x968)	÷ 4	M=64, N=16	÷ 5	M=80, N=16
UXGA (1608x1208)	÷ 5	M=80, N=16	÷ 6	M=96, N=16
QXGA (2048x1536)	÷ 6	M=96, N=16	÷ 8	M=128, N=16

Table 72: Example Scale Factors

Register Name	Type	RW	Comment
scaling_mode	16-bit unsigned integer	RW	0 – None 1 – Horizontal 2 – Full (Horizontal & Vertical) - Smooth
spatial_sampling	16-bit unsigned integer	RW	0 – Bayer Sampling 1 – Co-sited 2 – reserved
scale_m	16-bit unsigned integer	RW	Down scale factor: M component Range: 1 to 16 upwards
scale_n	16-bit unsigned integer	RW	Down scale factor: N component Value: 16 (fixed)

Table 73: Image Scaling Registers (Read/Write)

Register Name	Type	RW	Comment
scaling_capability	16-bit unsigned integer	RO Static	0 – None 1 – Horizontal (Smooth) 2 – Full (Horizontal & Vertical) - Smooth
scaler_m_min	16-bit unsigned integer	RO Static	Down scale factor: Minimum M value
scaler_m_max	16-bit unsigned integer	RO Static	Down scale factor: Maximum M value
scaler_n_min	16-bit unsigned integer	RO Static	Down scale factor: Minimum N value Value is always 16
scaler_n_max	16-bit unsigned integer	RO Static	Down scale factor: Maximum N value Value is always 16

Table 74: Image Scaling Capability Registers (Read Only and Static)

9.7 Scaler Example

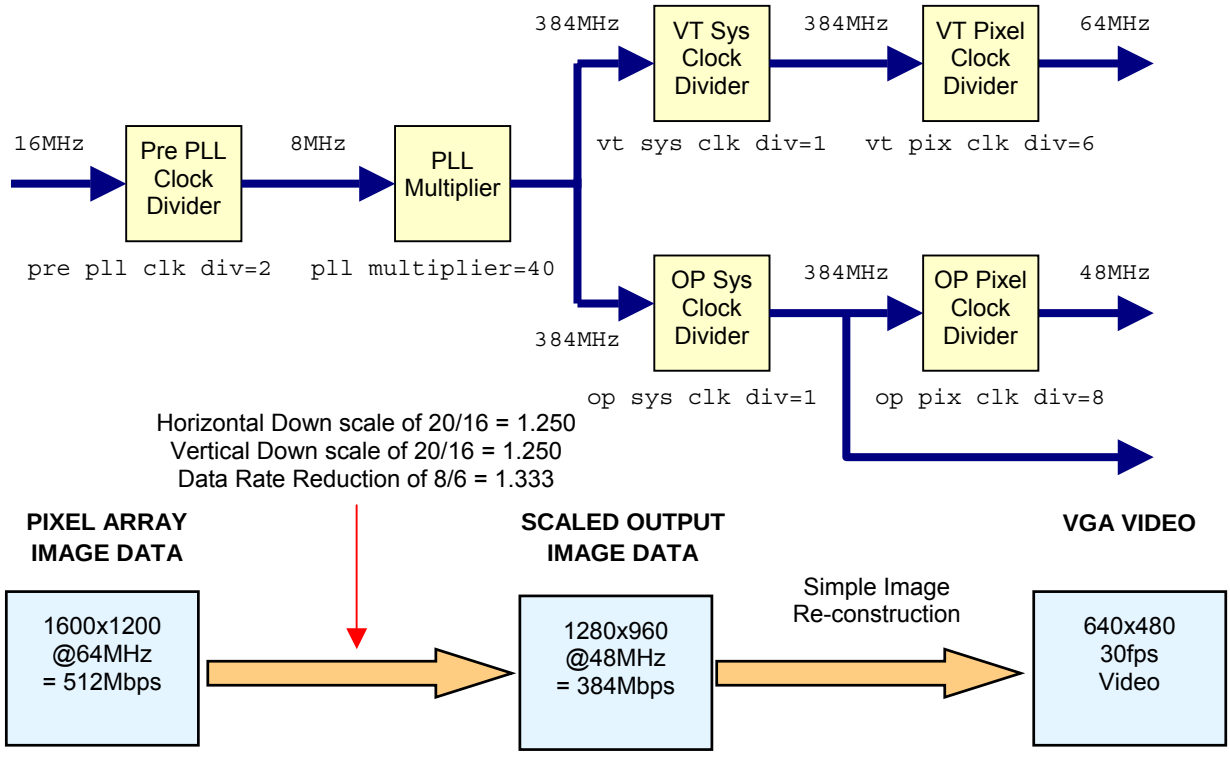


Figure 63: Full (H&V) Scaler Example

9.8 Scaler Frame Format Examples

The figures below show examples of a CCP2 frame of data with the horizontal and the full scaler enabled.

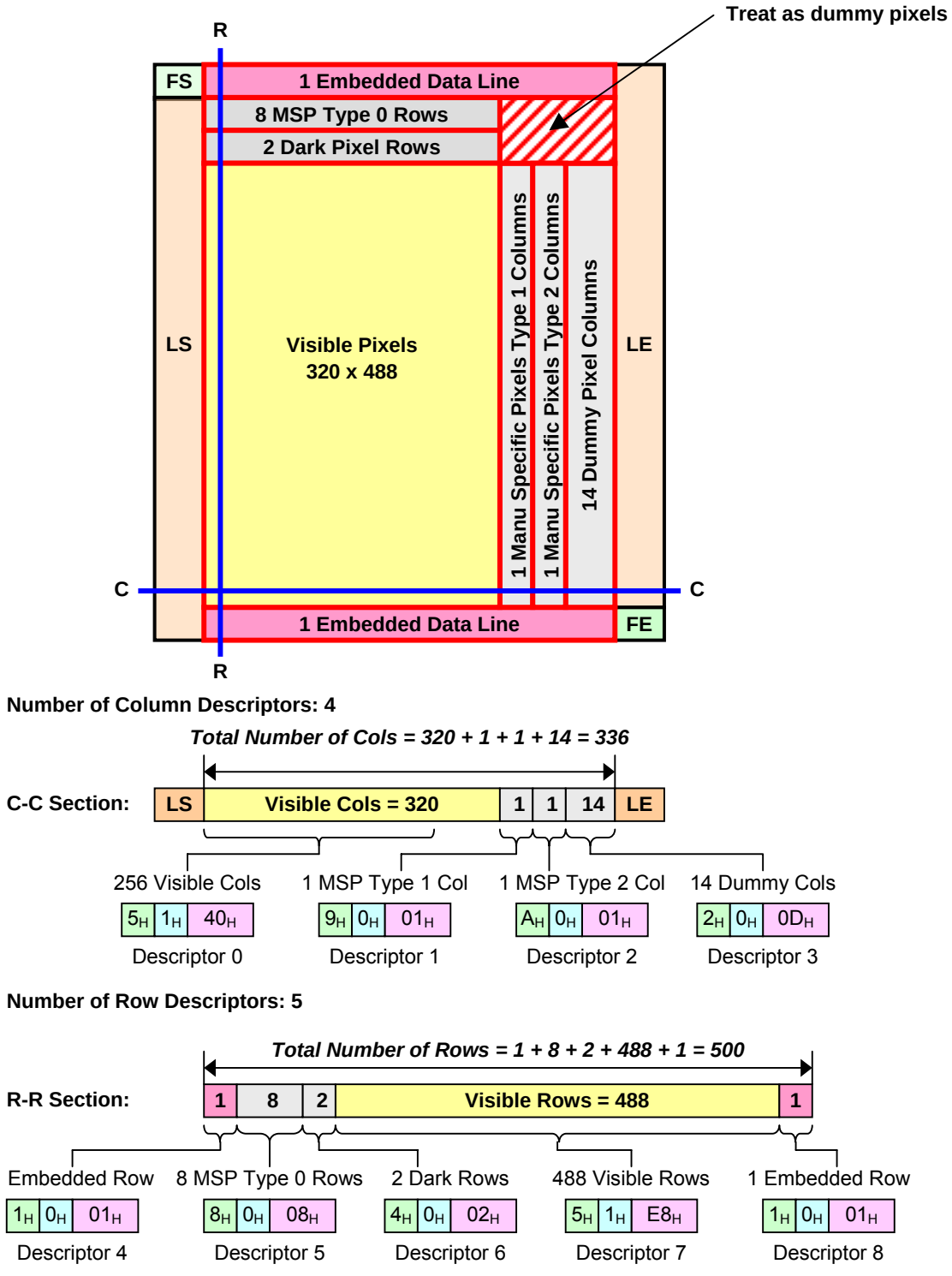


Figure 64: Horizontal Scaler Frame Format Example

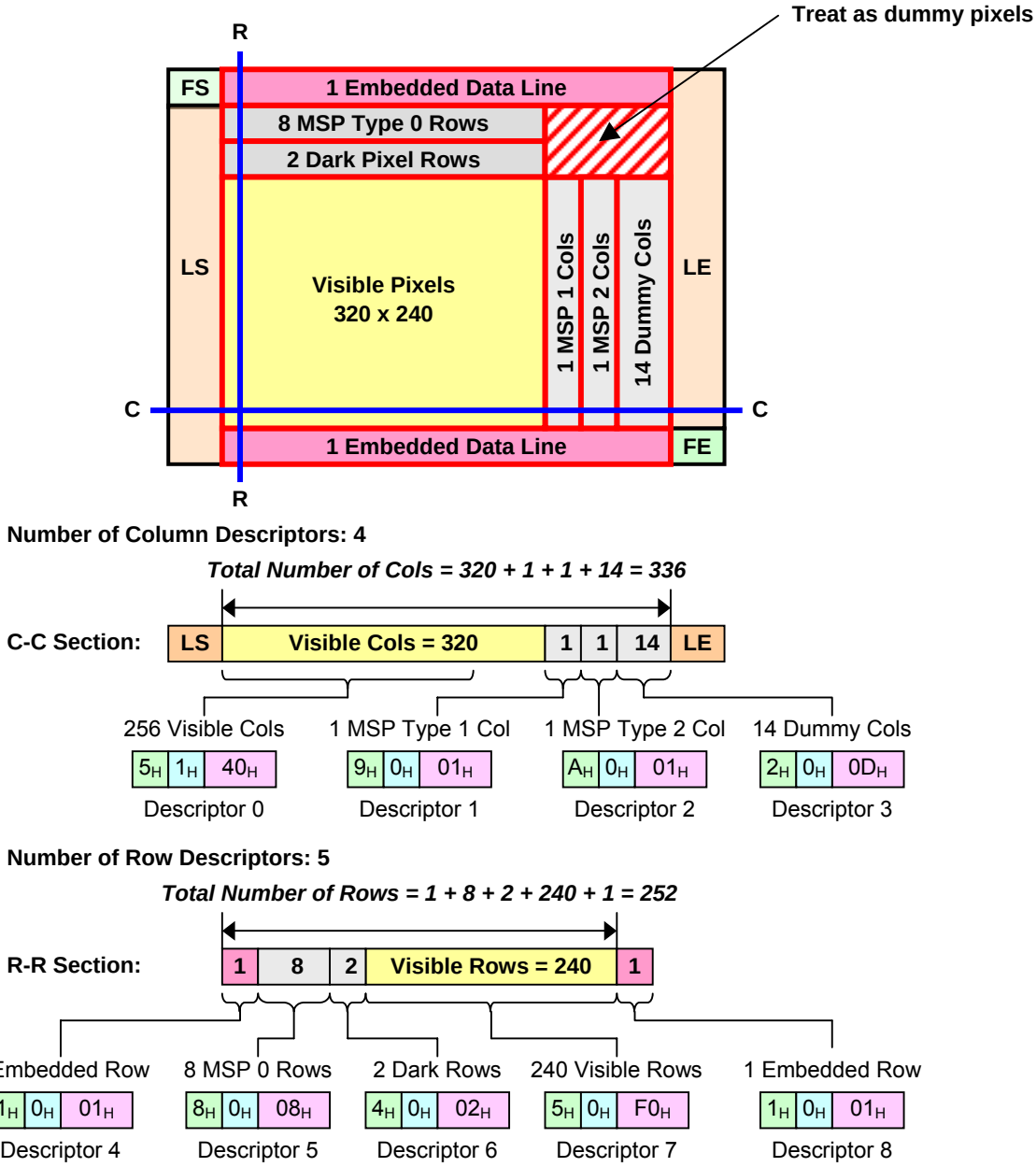


Figure 65: Full (H&V) Scaler Frame Format Example

10 Image Compression

10.1 Introduction

The objective of the image compression is to reduce the required bandwidth in transmission between the sensor and the host.

The key features of the DPCM/PCM compression algorithm are

- Visually lossless
- Low cost implementation (no line memories are required)
- Fixed Rate compression

Support of the 10-bit to 8-bit DPCM/PCM image compression algorithm specified in this chapter is mandatory for profile levels 1 and 2. 10-bit to 8-bit compression has the additional advantage that one pixel value equals one byte of data.

The level of compression is controlled via the `CCP_data_format` register. The same register is also used to enable and disable compression.

The `compression_mode` register is used to select which predictor, simple or advanced, the compression algorithm uses.

- The 10-bit to 8-bit DPCM/PCM compression only uses the simple predictor.
- The option of simple or advanced predictor is only valid for the optional 10-bit to 7-bit and 10-bit to 6-bit compression algorithms.

Register Name	Type	RW	Comment
`compression_mode`	16-bit unsigned integer	RW	0x01 – DPCM/PCM –Simple Predictor 0x02 – DPCM/PCM – Advanced Predictor

Table 75: Compression Mode Register

The `compression_capability` Read Only register tells the host whether a sensor module has does or does not have compression and if it has compression then what is the compression technique.

Again currently only the DPCM/PCM technique is supported.

Register Name	Type	RW	Comment
`compression_capability`	16-bit unsigned integer	RO	0x00 – None 0x01 – DPCM/PCM

Table 76: Compression Capability Register

10.2 10-bit to 8-bit DPCM/PCM Compression

The DPCM/PCM image compression method is lossy and it has designed so that it does not require any information outside the current encoded/decoded line. This means that all the image lines can be encoded/decoded separately.

The specified compression algorithm also avoids the false synchronization code generation of CCP2 bus. This property has been implemented so that the coded codeword cannot ever been full of zero bits and so there can never be more than 14 consequent zero bits in the stream.

Basic idea of the image compression system is:

- Xorig is the original pixel value (10 bits) INPUT
- Xpred is the predicted pixel value
- Xdiff is the difference value (Xorig – Xpred)
- Xenco is the encoded value (8 bits) CODED
- Xdeco is the decoded pixel value (10 bits) OUTPUT

When the prediction is good (abs(Xdiff) < Lim) its difference value is quantised and transmitted using DPCM codec. Otherwise the original value is quantised and transmitted using PCM codec. Of course, also the selection of the codec has to be transmitted. This selection information is combined together with DPCM/PCM code words.

The image compression system contains encoder and decoder blocks. In both blocks the similar prediction method has to be implemented. The block diagram of image compression system is shown in Figure 66.

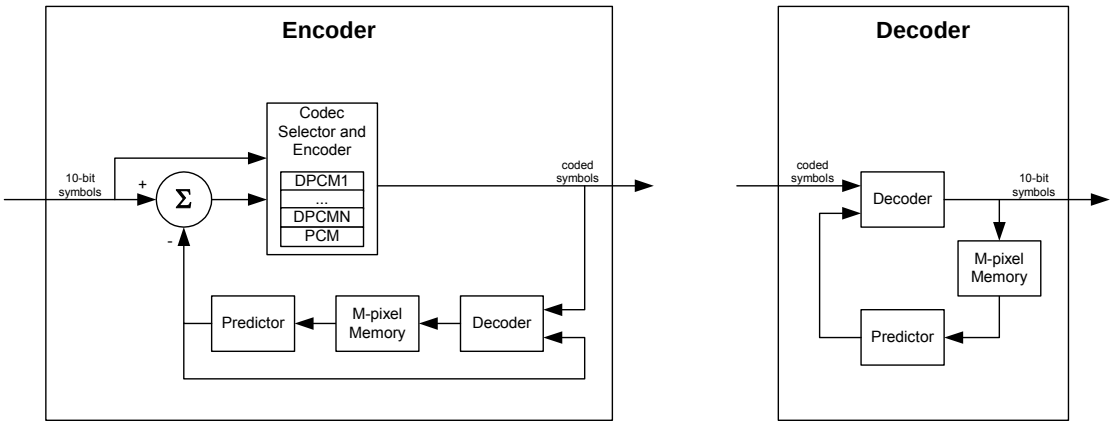


Figure 66:Block diagram of image compression system

10.2.1 Simple Predictor for 10-bit to 8-bit

The order of pixels in raw image has been depicted in Figure 67.

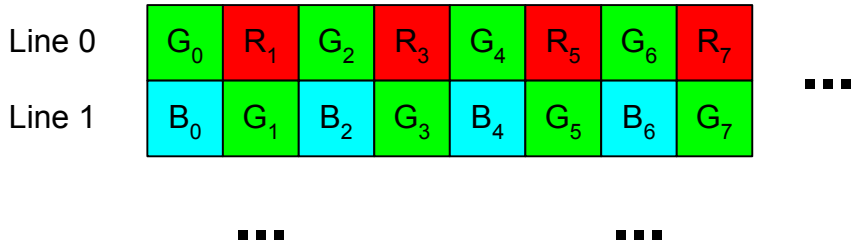


Figure 67:The order of pixels in raw image

This predictor uses only the previous same colour component value as a prediction value. Therefore, only two-pixel memory is required. The predictor equations can be written as below.

- First two pixels (G_0, R_1 / B_0, G_1) in all lines are coded without prediction.
- Other pixels in all lines are predicted using the previous same colour decoded value as a prediction value.

Xpred(n) = Xdeco(n-2)

10.2.2 Encoder for 10 bits to 8 bits (10 – 8 – 10)

The encoder for non-predicted pixels (beginning of lines) is different than the encoder for predicted pixels.

This coder offers 20 % bit rate reduction with very high image quality.

The pixels without prediction are encoded as below.

```
Xenco(n) = Xorig(n) / 4
```

And for avoiding full zero code the following check has been done.

```
If (Xenco(n) == 0) then Xenco(n) = 1
```

The pixels with prediction are encoded as below.

```
If (abs(Xdiff(n)) < 32) then              use DPCM1
Else if (abs(Xdiff(n)) < 64) then         use DPCM2
Else if (abs(Xdiff(n)) < 128) then        use DPCM3
Else                                      use PCM
```

10.2.2.1 DPCM1 (10 to 8)

```
Xenco(n) =   "00 s xxxxx"
             code word "00"
             sign "s"
             value with 5 bits "xxxxx"
value = abs(Xdiff(n))
if(Xdiff(n) <= 0) then
        sign = 1
else
        sign = 0
```

NOTE:
Zero code has been avoided (0 is sent as -0).

10.2.2.2 DPCM2 (10 to 8)

```
Xenco(n) =   "010 s xxxx"
             code word "010"
             sign "s"
             value with 4 bits "xxxx"
value = abs(Xdiff(n) - 32) / 2
if(Xdiff(n) < 0) then
```

```
            sign = 1
    else
            sign = 0
```

10.2.2.3 DPCM3 (10 to 8)

```
    Xenco(n) =   "011 s xxxx"
                 code word "011"
                 sign "s"
                 value with 4 bits "xxxx"
    value = abs(Xdiff(n) - 64) / 4
    if(Xdiff(n) < 0) then
            sign = 1
    else
            sign = 0
```

10.2.2.4 PCM (10 to 8)

```
    Xenco(n) =   "1 xxxxxxx"
                 code word "1"
                 value with 7 bits "xxxxxxx"
    value = Xorig(n) / 8
```

10.2.3 Decoder for 8 bits to 10 bits (10 – 8 – 10)

The decoder for non-predicted pixels is different than the decoder for predicted pixels.

The pixels without prediction are decoded as below.

```
    Xdeco(n) = 4 * Xenco(n) + 2
```

The pixels with prediction are decoded as below.

```
    If (Xenco(n) & 0xc0 == 0x00) then          use DPCM1
    Else if (Xenco(n) & 0xe0 == 0x40) then     use DPCM2
    Else if (Xenco(n) & 0xe0 == 0x60) then     use DPCM3
    Else                                       use PCM
```

10.2.3.1 DPCM1 (8 to 10)

```
    Xenco(n) = "00 s xxxxx"
            code word "00"
            sign "s"
            value with 5 bits "xxxxx"
    value = Xenco(n) & 0x1f
    sign = Xenco(n) & 0x20
    if(sign > 0) then
            Xdeco(n) = Xpred(n) - value
    Else
            Xdeco(n) = Xpred(n) + value
```

10.2.3.2 DPCM2 (8 to 10)

```
    Xenco(n) = "010 s xxxx"
                 code word "010"
                 sign "s"
                 value with 4 bits "xxxx"
```

```
value = 2 * (Xenco(n) & 0x0f) + 32
sign = Xenco(n) & 0x10
if(sign > 0) then
        Xdeco(n) = Xpred(n) – value
Else
        Xdeco(n) = Xpred(n) + value
```

10.2.3.3 DPCM3 (8 to 10)

```
Xenco(n) = "011 s xxxx"
        code word "011"
        sign "s"
        value with 4 bits "xxxx"
value = 4 * (Xenco(n) & 0x0f) + 64 + 1
sign = Xenco(n) & 0x10
if(sign > 0) then
        Xdeco(n) = Xpred(n) – value
        If (Xdeco(n) < 0) Xdeco(n) = 0
Else
        Xdeco(n) = Xpred(n) + value
        If (Xdeco(n) > 1023) Xdeco(n) = 1023
```

10.2.3.4 PCM (8 to 10)

```
Xenco(n) = "1 xxxxxxx"
        code word "1"
        value with 7 bits "xxxxxxx"
value = 8 * (Xenco(n) & 0x7f)
If (value > Xpred(n)) then
        Xdeco(n) = value + 3
Else
        Xdeco(n) = value + 4
```

11 Camera Control Interface (CCI)

SMIA sensor modules must use the CCI specification as defined in the CCP2 specification.

Figure 68 defines the sensor module's CCI slave address.

Figure 68: Sensor Module Slave Address - 0x20/0x21

Device	Write Address Byte	Read Address Byte
Image Sensor Module Slave Address	20_H	21_H

Table 77: Device Addresses

12 Register Map

12.1 Introduction

The registers are grouped according to function with each group occupying a pre-allocated region of the address space. This scheme is purely a conceptual feature and is not related to the actual hardware implementation.

The primary categories are given in Table:

CCI Indices	R/W	Description
Configuration Registers -[0x0000 – 0x0FFF]		
0x0000 - 0x00FF	Read Only	Status Registers (Dynamic Read Only Registers)
0x0100 - 0x01FF		Set-up Registers –Operating modes
0x0200 - 0x02FF		Integration time and Gain Parameters
0x0300 - 0x03FF		Video Timing Registers
0x0400 - 0x04FF		Image Scaling Registers
0x0500 - 0x05FF		Image Compression Registers
0x0600 - 0x06FF		Test Pattern Registers
Parameter Limit Registers – [0x1000 – 0x1FFF] (All registers are Read Only and Static)		
0x1000 - 0x10FF	Read Only	Integration time and Gain Parameters Limits
0x1100 - 0x11FF	Read Only	Video Timing Parameter Limits
0x1200 - 0x12FF	Read Only	Image Scaling Parameter Limits
0x1300 - 0x13FF	Read Only	Image Compression Capability Registers
0x1400 - 0x14FF	Read Only	Colour Matrix Registers
Image Statistics Registers -[0x2000 – 0x2FFF]		
0x2000 - 0x2FFF		Reserved for Image statistics Registers
Manufacturer Specific Registers - [0x3000 – 0x3FFF]		
0x3000 - 0x3FFF		Manufacturer specific registers

Table 78: CCI Register Groupings

Any internal register that can be written to can also be read from. There are also read only registers that contain device status information, (e.g. design revision details).

A read instruction from a reserved or un-used register location must yield the value 0x00.

A write instruction to an unused register location is illegal and the effect of such a write is undefined.

A read or write to an un-used register location **must** not cause the CCI read or write message to be aborted. Thus a CCI read or write message to an un-used register location must complete in the same way as if the read or write message had addressed a used register location.

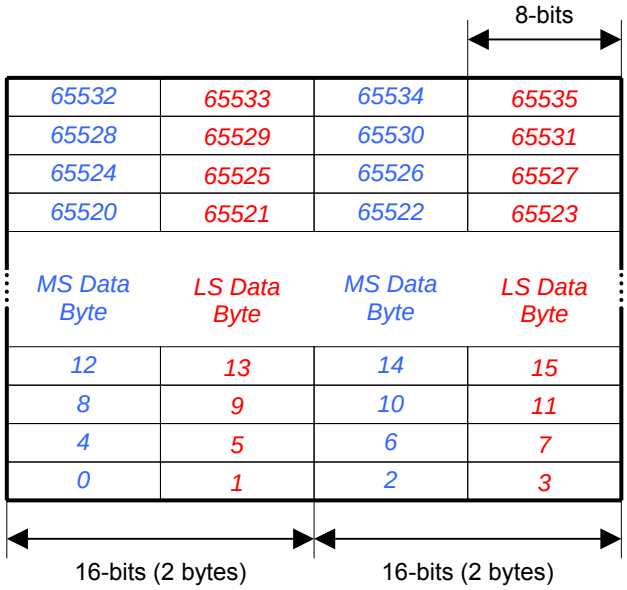
It is the responsibility of the host system to only write to register locations, which have been defined.

12.2 Multi-Byte Registers Index Space

The following sections define the valid locations for MS and LS Bytes of 16-bit and 32-bit register values.

12.2.1 Valid 16-bit Register Indices

For 16-bit wide registers the index of the MS byte must be a multiple of 2 i.e. the LS bit of the 16-bit index must be zero.

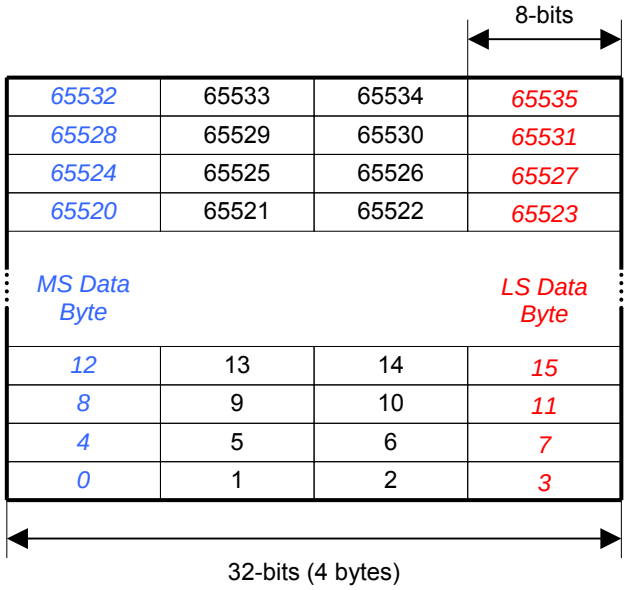


Blue – Valid 16-bit Indices for the MS Data Byte of a 16-bit Wide Register
Red – Valid 16-bit Indices for the LS Data Byte of a 16-bit Wide Register

Figure 69: Valid 16-bit Indices for the MS Data Byte of 16-bit wide Register

12.2.2 Valid 32-bit Register Indices

For 32-bit wide registers the index of the MS byte must be a multiple of 4 i.e. the 2 LS bits of the 16-bit index must be zero.



Blue – Valid 16-bit Indices for the MS Data Byte of a 32-bit Wide Register
Red – Valid 16-bit Indices for the LS Data Byte of a 32-bit Wide Register

Figure 70: Valid 16-bit Indices for the MS and LS Data Bytes of 32-bit wide Registers'

12.3 Data Alignment Within CCI Registers

If the width of an internal register is narrower than the CCI 8-bit, 16-bit or 32-bit register which reports it's value, then the register value is Right Aligned within the CCI register and the unused MS bits are padded with zeroes.

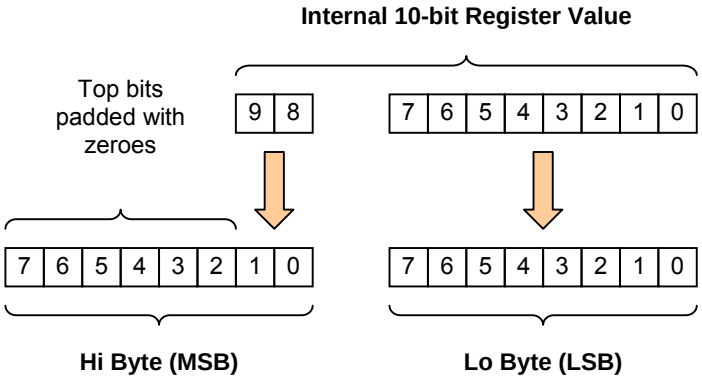


Figure 71: Right Alignment for Packing 10-bit Data into 2 8-bit Registers

12.4 Valid Register Formats

Table 79 lists the valid register formats.

Name	Range	Description
8-bit unsigned integer	0 to 255	
8-bit signed integer	-128 to 127	Two's complement notation
16-bit unsigned integer	0 to 65535	
16-bit signed integer	-32768 to 32767	Two's complement notation
16-bit unsigned iReal	0 to 255.99609375	8.8 fixed point number 8 integer bits (MS Byte), 8 fractional bits (LS Byte)
16-bit signed iReal	-128 to 127.9960375	Two's complement notation, 8 fractional bits
32-bit unsigned iReal	0 to 65535.99998474	16.16 fixed point number 16 integer bits (MS 2 Bytes), 16 fractional bits (LS 2 Bytes)
32-bit signed iReal	-32768 to 32767.99998474	Two's complement notation, 16 fractional bits
32-bit IEEE floating-point number	As per IEEE 754	As per IEEE 754 1 sign bit, 8 exponent bits, 23 fractional bits

Table 79- Valid Register Formats

12.5 Configuration Registers - [0x0000-0x0FFF]

12.5.1 Status Registers – [0x0000-0x00FF] (Read Only Dynamic Registers)

12.5.1.1 General Status Registers – [0x0000-0x000F] (Read Only and Dynamic)

Index	Byte	Register Name	RW	Comment
0x0000	Hi	model_id	RO	16-bit Sensor model number e.g. 552
0x0001	Lo			
0x0002		revision_number	RO	Silicon Revision Number
0x0003		manufacturer_id	RO	Manufacturer ID
0x0004		smia_version	RO	9 – SMIA V0.9 10 - SMIA V1.0 11 – SMIA V1.1
0x0005		frame_count	RO	8-bit (0-255) Frame counter value
0x0006		pixel_order	RO	Colour Pixel Order
0x0007		Reserved		
0x0008	Hi	data_pedestal	RO	Data pedestal – typically code 64 for 10-bit systems Refer to the Data Format Chapter
0x0009	Lo			
0x000A				
0x000B				
0x000C		pixel_depth	RO	8-bit, 10-bit or 12-bit pixel data
0x000D				
0x000E				
0x000F				

Table 80: General Status Registers (Read Only and Dynamic)

12.5.1.2 Frame Format Description – [0x0040-0x007F]

For a full description of the frame format description please refer to the Data Format Chapter.

Index	Byte	Register Name	R/W	Comment
0x0040		frame_format_model_type	RO	
0x0041		frame_format_model_subtype	RO	
0x0042	Hi	frame_format_descriptor_0	RO	
0x0043	Lo			
0x0044	Hi	frame_format_descriptor_1	RO	
0x0045	Lo			
0x0046	Hi	frame_format_descriptor_2	RO	
0x0047	Lo			
0x0048	Hi	frame_format_descriptor_3	RO	
0x0049	Lo			
0x004A	Hi	frame_format_descriptor_4	RO	
0x004B	Lo			
0x004C	Hi	frame_format_descriptor_5	RO	
0x004D	Lo			
0x004E	Hi	frame_format_descriptor_6	RO	
0x004F	Lo			
0x0050	Hi	frame_format_descriptor_7	RO	
0x0051	Lo			
0x0052	Hi	frame_format_descriptor_8	RO	
0x0053	Lo			
0x0054	Hi	frame_format_descriptor_9	RO	
0x0055	Lo			
0x0056	Hi	frame_format_descriptor_10	RO	
0x0057	Lo			
0x0058	Hi	frame_format_descriptor_11	RO	
0x0059	Lo			
0x005A	Hi	frame_format_descriptor_12	RO	
0x005B	Lo			
0x005C	Hi	frame_format_descriptor_13	RO	
0x005D	Lo			
0x005E	Hi	frame_format_descriptor_14	RO	
0x005F	Lo			

Table 81: Frame Format Description

12.5.1.3 Analogue Gain Description Registers – [0x0080-0x0097]

These registers are not dynamic but are required to be output on the status line so that it is possible to interpret the meaning of the analogue gain code(s).

For a full description of the analogue gain description please refer to the Integration Time and Gain Control Chapter.

Index	Byte	Register Name	RW	Comment
0x0080	Hi	analogue_gain_capabiltiy	RO	Analogue Gain Capability 0 – single global analogue gain only 1 – 4 separate channel analogue gains only
0x0081	Lo			
0x0082	Hi	Reserved	RO	
0x0083	Lo			
0x0084	Hi	analogue_gain_code_min	RO	Minimum recommended analogue gain code Format: 16-bit unsigned integer
0x0085	Lo			
0x0086	Hi	analogue_gain_code_max	RO	Maximum recommended analogue gain code Format: 16-bit unsigned integer
0x0087	Lo			
0x0088	Hi	analogue_gain_code_step	RO	Analogue gain code step size Format: 16-bit unsigned integer
0x0089	Lo			
0x008A	Hi	analogue_gain_type	RO	Analogue gain type Format: 16-bit unsigned integer
0x008B	Lo			
0x008C	Hi	analogue_gain_m0	RO	Analogue gain m0 constant Format: 16-bit signed integer
0x008D	Lo			
0x008E	Hi	analogue_gain_c0	RO	Analogue gain c0 constant Format: 16-bit signed integer
0x008F	Lo			
0x0090	Hi	analogue_gain_m1	RO	Analogue gain m1 constant Format: 16-bit signed integer
0x0091	Lo			
0x0092	Hi	analogue_gain_c1	RO	Analogue gain c1 constant Format: 16-bit signed integer
0x0093	Lo			

Table 82: Analogue Gain Description Registers

12.5.1.4 Data Format Description – [0x00C0-0x00FF]

For a full description of the data format description please refer to the Data Format Chapter.

Index	Byte	Register Name	R/W	Comment
0x00C0		data_format_model_type	RO	
0x00C1		data_format_model_subtype	RO	
0x00C2	Hi	data_format_descriptor_0	RO	
0x00C3	Lo			
0x00C4	Hi	data_format_descriptor_1	RO	
0x00C5	Lo			
0x00C6	Hi	data_format_descriptor_2	RO	
0x00C7	Lo			
0x00C8	Hi	data_format_descriptor_3	RO	
0x00C9	Lo			
0x00CA	Hi	data_format_descriptor_4	RO	
0x00CB	Lo			
0x00CC	Hi	data_format_descriptor_5	RO	
0x00CD	Lo			
0x00CE	Hi	data_format_descriptor_6	RO	
0x00CF	Lo			

Table 83: Data Format Description

12.5.2 Set-up Registers – [0x0100-0x01FF]

12.5.2.1 General Set-up Registers – [0x0100-0x010F]

Index	Byte	Register Name	RW	Comment
0x0100		mode_select	RW	Mode Select 0 – Software Standby 1 - Streaming Refer to Operating Modes Chapter
0x0101		image_orientation	RW	Image orientation i.e. horizontal mirror and vertical flip Refer to Video Timing Chapter
0x0102		Reserved		
0x0103		software_reset		Software reset Refer to Operating Modes Chapter
0x0104		grouped_parameter_hold	RW	The grouped parameter hold register disables the consumption of integration, gain and video timing parameters 0 – consume as normal 1 - hold Refer to Integration Time and Gain Control Chapter
0x0105		mask_corrupted_frames	RW	Refer to Integration Time and Gain Control Chapter

Table 84: General Set-up Registers

12.5.2.2 Output Set-up Registers – [0x0110-0x011F]

For a full description of the output set-up registers please refer to the Data Format Chapter.

Index	Byte	Register Name	RW	Comment
0x0110		CCP2_channel_identifier	RW	Channel Identifier Value for CCP2 embedded codes Valid Range: 0-7
0x0111		CCP2_signalling_mode	RW	0 – Data/Clock Signalling 1 - Data/Strobe Signalling
0x0112	Hi	CCP_data_format	RW	CCP Data Format 0x0808: Top 8-b of pixel data,RAW8 0x0A08: 10-b to 8-b compression, RAW8 0x0A0A: Top 10-b of pixel data, RAW10
0x0113	Lo			

Table 85: Output Set-up Registers

12.5.2.3 Integration Time and Gain Set-up Registers – [0x0120-0x012F]

For a full description of the integration time and gain set-up registers please refer to the Integration Time and Gain Control Chapter.

Index	Byte	Register Name	RW	Comment
0x0120		gain_mode	RW	0 – Global Analogue Gain (Default) 1 – Per Channel Analogue Gain (Only if sensor supports it) Refer to Integration Time and Gain Chapter

Table 86: Integration Time and Gain Set-up Registers

12.5.3 Integration Time and Gain Registers – [0x0200-0x02FF]

For a full description of the integration time and gain registers please refer to the Integration Time and Gain Control Chapter.

12.5.3.1 Integration Time Registers – [0x2000-0x2003]

Index	Byte	Register Name	RW	Comment
0x0200	Hi	fine_integration_time	RW	Fine integration time (pixels) Format: 16-bit unsigned integer
0x0201	Lo			
0x0202	Hi	coarse_integration_time	RW	Coarse integration time (lines) Format: 16-bit unsigned integer
0x0203	Lo			

Table 87: Integration Time Registers

12.5.3.2 Analogue Gain Registers – [0x0204-0x020D]

Index	Byte	Name	RW	Comment
0x0204	Hi	analogue_gain_code_global	RW	Global Analogue Gain Code Format: 16-bit unsigned integer
0x0205	Lo			
0x0206	Hi	analogue_gain_code_greenR	RW	Analogue Gain Code Format: 16-bit unsigned integer
0x0207	Lo			
0x0208	Hi	analogue_gain_code_red	RW	Red Channel Analogue Gain Code Format: 16-bit unsigned integer
0x0209	Lo			
0x020A	Hi	analogue_gain_code_blue	RW	Blue Channel Analogue Gain Code Format: 16-bit unsigned integer
0x020B	Lo			
0x020C	Hi	analogue_gain_code_greenB	RW	Green (Blue Row) Analogue Gain Code Format: 16-bit unsigned integer
0x020D	Lo			

Table 88: Analogue Gain Registers

12.5.3.3 Digital Gain Registers – [0x020E-0x0215]

Index	Byte	Name	RW	Comment
0x020E	Hi	digital_gain_greenR	RW	Green (Red Row) channel digital gain value Format: 16-bit unsigned iReal
0x020F	Lo			
0x0210	Hi	digital_gain_red	RW	Red channel digital gain value Format: 16-bit unsigned iReal
0x0211	Lo			
0x0212	Hi	digital_gain_blue	RW	Blue channel digital gain value Format: 16-bit unsigned iReal
0x0213	Lo			
0x0214	Hi	digital_gain_greenB	RW	Green (Blue Row) channel digital gain value Format: 16-bit unsigned iReal
0x0215	Lo			

Table 89: Digital Gain Registers

12.5.4 Video Timing Registers – [0x0300-0x03FF]

For a full description of the video timing registers please refer to the Video Timing Chapter.

12.5.4.1 Clock Set-up Registers – [0x0300-0x0307]

Index	Byte	Register Name	RW	Comment
0x0300	Hi	vt_pix_clk_div	RW	Video Timing Pixel Clock Divider Format: 16-bit unsigned integer
0x0301	Lo			
0x0302	Hi	vt_sys_clk_div	RW	Video Timing System Clock Divider Value Format: 16-bit unsigned integer
0x0303	Lo			
0x0304	Hi	pre_pll_clk_div	RW	Pre PLL clock Divider Value Format: 16-bit unsigned integer
0x0305	Lo			
0x0306	Hi	pll_multiplier	RW	PLL multiplier Value Format: 16-bit unsigned integer
0x0307	Lo			
0x0308	Hi	op_pix_clk_div	RO	Output Pixel Clock Divider Format: 16-bit unsigned integer
0x0309	Lo			
0x030A	Hi	op_sys_clk_div	RW	Output System Clock Divider Value Format: 16-bit unsigned integer
0x030B	Lo			

Table 90: Clock Set-up Registers

12.5.4.2 Frame Timing Registers – [0x0340-0x0343]

Index	Byte	Name	RW	Comment
0x0340	Hi	frame_length_lines	RW	Frame Length Format: 16-bit unsigned integer Units: Lines
0x0341	Lo			
0x0342	Hi	line_length_pck	RW	Line Length Format: 16-bit unsigned integer Units: Pixel Clocks
0x0343	Lo			

Table 91: Frame Timing Registers

12.5.4.3 Image Size Registers – [0x0344-0x034F]

Index	Byte	Register Name	RW	Comment
0x0344	Hi	x_addr_start	RW	X-address of the top left corner of the visible pixel data Format: 16-bit unsigned integer Units: Pixels
0x0345	Lo			
0x0346	Hi	y_addr_start	RW	Y-address of the top left corner of the visible pixel data Format: 16-bit unsigned integer Units: Lines
0x0347	Lo			
0x0348	Hi	x_addr_end	RW	X-address of the bottom right corner of the visible pixel data Format: 16-bit unsigned integer Units: Pixels
0x0349	Lo			
0x034A	Hi	y_addr_end	RW	Y-address of the bottom right corner of the visible pixel data Format: 16-bit unsigned integer Units: Lines
0x034B	Lo			
0x034C	Hi	x_output_size	RW	Width of image data output from the sensor module Format: 16-bit unsigned integer Units: Pixels
0x034D	Lo			
0x034E	Hi	y_output_size	RW	Height of image data output from the sensor module Format: 16-bit unsigned integer Units: Lines
0x034F	Lo			

Table 92: Image Size Registers

12.5.4.4 Sub-Sampling Registers – [0x0380-0x0387]

Index	Bybte	Register Name	RW	Comment
0x0380	Hi	x_even_inc	RW	Increment for even pixels – 0, 2, 4 etc Format: 16-bit unsigned integer
0x0381	Lo			
0x0382	Hi	x_odd_inc	RW	Increment for odd pixels – 1, 3, 5 etc Format: 16-bit unsigned integer
0x0383	Lo			
0x0384	Hi	y_even_inc	RW	Increment for even pixels – 0, 2, 4 etc Format: 16-bit unsigned integer
0x0385	Lo			
0x0386	Hi	y_odd_inc	RW	Increment for odd pixels – 1, 3, 5 etc Format: 16-bit unsigned integer
0x0387	Lo			

Table 93: Sub-Sampling Registers

12.5.5 Image Scaling Registers – [0x0400-0x04FF]

For a full description of the image scaling registers please refer to the Image Scaling Chapter.

Index	Byte	Register Name	RW	Comment
0x0400	Hi	Scaling_mode	RW	0 – No scaling 1 – Horizontal Scaling 2 – Full Scaling (both horizontal and vertical)
0x0401	Lo			
0x0402	Hi	Spatial_sampling	RW	0 – Bayer Sampling 1 – Co-sited 2 – Reserved
0x0403	Lo			
0x0404	Hi	scale_m	RW	Down scale factor: M component Range: 1 to 16 upwards Format: 16-bit unsigned integer
0x0405	Lo			
0x0406	Hi	scale_n	RW	Down scale factor: N component Value: 16 (fixed) Format: 16-bit unsigned integer
0x0407	Lo			
0x0408	Hi	reserved	RW	
0x0409	Lo			
0x040a	Hi	reserved	RW	
0x040b	Lo			

Table 94: Image Scaling Registers

12.5.6 Image Compression Registers – [0x0500-0x05FF]

For a full description of the image compression pattern registers please refer to the Image Compression Chapter.

Index	Byte	Register Name	RW	Comment
0x0500	Hi	compression_algorithim	RW	1 – DPCM/PCM Compression
0x0501	Lo			

Table 95: Image Compression Registers

12.5.7 Test Pattern Registers – [0x0600-0x06FF]

For a full description of the deterministic test pattern registers please refer to the Test Chapter.

Index	Byte	Register Name	RW	Comment
0x0600	Hi	test_pattern_mode	RW	
0x0601	Lo			
0x0602	Hi	test_data_red	RW	
0x0603	Lo			
0x0604	Hi	test_data_greenR	RW	
0x0605	Lo			
0x0606	Hi	test_data_blue	RW	
0x0607	Lo			
0x0608	Hi	test_data_greenB	RW	
0x0609	Lo			
0x060A	Hi	horizontal_cursor_width	RW	
0x060B	Lo			
0x060C	Hi	horizontal_cursor_position	RW	
0x060D	Lo			
0x060E	Hi	vertical_cursor_width	RW	
0x060F	Lo			
0x0610	Hi	vertical_cursor_position	RW	
0x0611	Lo			

Table 96: Test Pattern Registers

12.6 Parameter Limit Registers – [0x1000-0x1FFF] (Read Only and Static)

12.6.1 Integration Time and Gain Parameter Limit Registers – [0x1000-0x10FF]

For a full description of the integration time and gain parameter limits please refer to the Integration Time and Gain Control Chapter.

12.6.1.1 Integration Time Parameter Limit Registers – [0x1000-0x100B]

Index	Byte	Register Name	RW	Comment
0x1000	Hi	integration_time_capability	RO	0 – coarse integration but NO fine integration 1 – course and smooth (1 pixel) fine integration
0x1001	Lo			
0x1002	Hi	Reserved	RO	
0x1003	Lo			
0x1004	Hi	coarse_integration_time_min	RO	Lines Format: 16-bits unsigned integer
0x1005	Lo			
0x1006	Hi	coarse_integration_time_max_margin	RO	(Current frame length – current max coarse exp) Format: 16-bits unsigned integer
0x1007	Lo			
0x1008	Hi	fine_integration_time_min	RO	Pixels Format: 16-bits unsigned integer
0x1009	Lo			
0x100A	Hi	fine_integration_time_max_margin	RO	(Current line length – current max fine exp) Format: 16-bits unsigned integer
0x100B	Lo			

Table 97: Integration Time Capability Registers

12.6.1.2 Digital Gain Parameter Limit Registers – [0x1080-0x1089]

Index	Byte	Register Name	RW	Comment
0x1080	Hi	digital_gain_capability	RO	0 – none 1 – per channel digital gain
0x1081	Lo			
0x1082	Hi	Reserved	RO	
0x1083	Lo			
0x1084	Hi	digital_gain_min	RO	Minimum recommended digital gain value Format: 16-bit unsigned 8.8 fixed point number
0x1085	Lo			
0x1086	Hi	digital_gain_max	RO	Maximum recommended digital gain value Format: 16-bit unsigned 8.8 fixed point number
0x1087	Lo			
0x1088	Hi	digital_gain_step_size	RO	Digital gain step size Format: 16-bit unsigned 8.8 fixed point number
0x1089	Lo			

Table 98: Digital Gain Capability Registers

12.6.2 Video Timing Parameter Limit Registers – [0x1100-0x11FF]

For a full description of the video timing parameter limit registers please refer to the Video Timing Chapter.

12.6.2.1 Pre-PLL and PLL Clock Set-up Capability Registers - [0x1100-0x111F]

Index	Byte	Register Name	RW	Comment
0x1100	Hi	min_ext_clk_freq_mhz	RO	Minimum external clock frequency Format: IEEE 32-bit float Units: MHz
0x1101				
0x1102				
0x1103	Lo			
0x1104	Hi	max_ext_clk_freq_mhz	RO	Maximum external clock frequency Format: IEEE 32-bit float Units: MHz
0x1105				
0x1106				
0x1107	Lo			
0x1108	Hi	min_pre_pll_clk_div	RO	Minimum Pre PLL divider value Format: 16-bit unsigned integer
0x1109	Lo			
0x110A	Hi	max_pre_pll_clk_div	RO	Maximum Pre PLL divider value Format: 16-bit unsigned integer
0x110B	Lo			
0x110C	Hi	min_pll_ip_freq_mhz	RO	Minimum PLL input clock frequency Format: IEEE 32-bit float Units: MHz
0x110D				
0x110E				
0x110F	Lo			
0x1110	Hi	max_pll_ip_freq_mhz	RO	Maximum PLL input clock frequency Format: IEEE 32-bit float Units: MHz
0x1111				
0x1112				
0x1113	Lo			
0x1114	Hi	min_pll_multiplier	RO	Minimum PLL multiplier Format: 16-bit unsigned integer
0x1115	Lo			
0x1116	Hi	max_pll_multiplier	RO	Maximum PLL Multiplier Format: 16-bit unsigned integer
0x1117	Lo			
0x1118	Hi	min_pll_op_freq_mhz	RO	Minimum PLL output clock frequency Format: IEEE 32-bit float Units: MHz
0x1119				
0x111A				
0x111B	Lo			
0x111C	Hi	max_pll_op_freq_mhz	RO	Maximum PLL output clock frequency Format: IEEE 32-bit float Units: MHz
0x111D				
0x111E				

Index	Byte	Register Name	RW	Comment
0x111F	Lo			

Table 99: Pre-PLL and PLL Clock Set-up Capability Registers

12.6.2.2 Video Timing Clock Set-up Capability Registers - [0x1120-0x1137]

Index	Byte	Register Name	RW	Comment
0x1120	Hi	min_vt_sys_clk_div	RO	Minimum video timing system clock divider value Format: 16-bit unsigned integer
0x1121	Lo			
0x1122	Hi	max_vt_sys_clk_div	RO	Maximum video timing system clock divider value Format: 16-bit unsigned integer
0x1123	Lo			
0x1124	Hi	min_vt_sys_ck_clk_freq_mhz	RO	Minimum video timing system clock frequency Format: IEEE 32-bit float Units: MHz
0x1125				
0x1126				
0x1127	Lo			
0x1128	Hi	max_vt_sys_clk_freq_mhz	RO	Maximum video timing system clock frequency Format: IEEE 32-bit float Units: MHz
0x1129				
0x112A				
0x112B	Lo			
0x112C	Hi	min_vt_pix_clk_freq_mhz	RO	Minimum video timing pixel clock frequency Format: IEEE 32-bit float Units: MHz
0x112D				
0x112E				
0x112F	Lo			
0x1130	Hi	max_vt_pix_clk_freq_mhz	RO	Maximum video timing pixel clock frequency Format: IEEE 32-bit float Units: MHz
0x1131				
0x1132				
0x1133	Lo			
0x1134	Hi	min_vt_pix_clk_div	RO	Minimum video timing pixel clock divider value Format: 16-bit unsigned integer
0x1135	Lo			
0x1136	Hi	max_vt_pix_clk_div	RO	Maximum video timing pixel clock divider value Format: 16-bit unsigned integer
0x1137	Lo			

Table 100: Video Timing Clock Set-up Capability Registers

12.6.2.3 Frame Timing Parameter Limit Registers – [0x1140-0x1149]

Index	Byte	Register Name	RW	Comment
0x1140	Hi	min_frame_length_lines	RO	Minimum Frame Length allowed. Value both sensor dependent Format: 16-bit unsigned integer Units: Lines
0x1141	Lo			
0x1142	Hi	max_frame_length_lines	RO	Maximum possible number of lines per Frame. Value sensor dependent Format: 16-bit unsigned integer Units: Lines
0x1143	Lo			
0x1144	Hi	min_line_length_pck	RO	Minimum Line Length allowed. Value sensor dependent Format: 16-bit unsigned integer Units: Pixel Clocks
0x1145	Lo			
0x1146	Hi	max_line_length_pck	RO	Maximum possible number of pixel clocks per line. Value sensor dependent Format: 16-bit unsigned integer Units: Pixel Clocks
0x1147	Lo			
0x1148	Hi	min_line_blanking_pck	RO	Minimum line blanking time in pixel clocks Format: 16-bit unsigned integer Units: Pixel Clocks
0x1149	Lo			

Table 101: Frame Timing Capability Registers

12.6.2.4 Output Clock Set-up Capability Registers - [0x1160-0x1177]

Index	Byte	Register Name	RW	Comment
0x1160	Hi	min_op_sys_clk_div	RO	Minimum output system clock divider value Format: 16-bit unsigned integer
0x1161	Lo			
0x1161	Hi	max_op_sys_clk_div	RO	Maximum output system clock divider value Format: 16-bit unsigned integer
0x1163	Lo			
0x1164	Hi	min_op_sys_ck_clk_freq_mhz	RO	Minimum output system clock frequency Format: IEEE 32-bit float Units: MHz
0x1165				
0x1166				
0x1167	Lo			
0x1168	Hi	max_op_sys_clk_freq_mhz	RO	Maximum output system clock frequency Format: IEEE 32-bit float Units: MHz
0x1169				
0x116A				

Index	Byte	Register Name	RW	Comment
0x116B	Lo			
0x116C	Hi	min_op_pix_clk_div	RO	Minimum output pixel clock divider value Format: 16-bit unsigned integer
0x116D	Lo			
0x116E	Hi	max_op_pix_clk_div	RO	Maximum output pixel clock divider value Format: 16-bit unsigned integer
0x116F	Lo			
0x1170	Hi	min_op_pix_clk_freq_mhz	RO	Minimum output pixel clock frequency Format: IEEE 32-bit float Units: MHz
0x1171				
0x1172				
0x1173	Lo			
0x1174	Hi	max_op_pix_clk_freq_mhz	RO	Maximum output pixel clock frequency Format: IEEE 32-bit float Units: MHz
0x1175				
0x1176				
0x1177	Lo			

Table 102: Output Clock Set-up Capability Registers

12.6.2.5 Image Size Parameter Limit Registers – [0x1180-0x1187]

Index	Byte	Name	RW	Comment
0x1180	Hi	x_addr_min	RO	Minimum X-address of the addressable pixel array Format: 16-bit unsigned integer Value: Always 0
0x1181	Lo			
0x1182	Hi	y_addr_min	RO	Minimum Y-address of the addressable pixel array Format: 16-bit unsigned integer Value: Always 0
0x1183	Lo			
0x1184	Hi	x_addr_max	RO	Maximum X-address of the addressable pixel array Format: 16-bit unsigned integer
0x1185	Lo			
0x1186	Hi	y_addr_max	RO	Maximum Y-address of the addressable pixel array Format: 16-bit unsigned integer
0x1187	Lo			

Table 103: Image Size Capability Registers

12.6.2.6 Sub-Sampling Parameter Limit Registers – [0x11C0-0x011C7]

Index	Byte	Name	RW	Comment
0x11C0	Hi	min_even_inc	RO	Minimum Increment for even pixels Format: 16-bit unsigned integer (static)
0x11C1	Lo			
0x11C2	Hi	max_even_inc	RO	Maximum increment for even pixels Format: 16-bit unsigned integer (static)
0x11C3	Lo			
0x11C4	Hi	min_odd_inc	RO	Minimum Increment for odd pixels Format: 16-bit unsigned integer (static)
0x11C5	Lo			
0x11C6	Hi	max_odd_inc	RO	Maximum Increment for odd pixels Format: 16-bit unsigned integer (static)
0x11C7	Lo			

Table 104: Sub-Sampling Capability Registers

12.6.3 Image Scaling Parameter Limit Registers – [0x1200-0x12FF]

For a full description of the image scaling parameter limit registers please refer to the Image Scaling Chapter.

Index	Byte	Register Name	RW	Comment
0x1200	Hi	scaling_capability	RO	0 – None 1 – Horizontal 2 – Full (Horizontal & Vertical) Format: 16-bit unsigned integer
0x1201	Lo			
0x1202	Hi	reserved	RO	
0x1203	Lo			
0x1204	Hi	scaler_m_min	RO	Down scale factor: Minimum M value Value is always 16 Format: 16-bit unsigned integer
0x1205	Lo			
0x1206	Hi	scaler_m_max	RO	Down scale factor: Maximum M value Format: 16-bit unsigned integer
0x1207	Lo			
0x1208	Hi	scaler_n_min	RO	Down scale factor: Minimum N value Value is always 16 Format: 16-bit unsigned integer
0x1209	Lo			
0x120A	Hi	scaler_n_max	RO	Down scale factor: Maximum N value Value is always 16 Format: 16-bit unsigned integer
0x120B	Lo			
0x120C	Hi	Reserved	RO	
0x120D	Lo			
0x120E	Hi	Reserved	RO	
0x120F	Lo			

Table 105: Image Scaling Capability Registers

12.6.4 Image Compression Capability Registers – [0x1300-0x13FF]

For a full description of the image compression capability registers please refer to the Image Compression Chapter.

Index	Byte	Register Name	RW	Comment
0x1300	Hi	compression_capability	RW	0 – No Compression 1 – DPCM/PCM Compression
0x1301	Lo			

Table 106: Image Compression Capability Registers

12.6.5 Colour Matrix Registers – [0x1400-0x14FF]

For a full description of the colour matrix registers please refer to the Colour Matrix Chapter.

Index	Byte	Register Name	RW	Comment
0x1400	Hi	matrix_element_RedInRed	RO	Colour matrix parameter for Red in Red Format: 16-bit signed iReal
0x1401	Lo			
0x1402	Hi	matrix_element_GreenInRed	RO	Colour matrix parameter for Green in Red Format: 16-bit signed iReal
0x1403	Lo			
0x1404	Hi	matrix_element_BlueInRed	RO	Colour matrix parameter for Blue in Red Format: 16-bit signed iReal
0x1405	Lo			
0x1406	Hi	matrix_element_RedInGreen	RO	Colour matrix parameter for Red in Green Format: 16-bit signed iReal
0x1407	Lo			
0x1408	Hi	matrix_element_GreenInGreen	RO	Colour matrix parameter for Green in Green Format: 16-bit signed iReal
0x1409	Lo			
0x140A	Hi	matrix_element_BlueInGreen	RO	Colour matrix parameter for Blue in Green Format: 16-bit signed iReal
0x140B	Lo			
0x140C	Hi	matrix_element_RedInBlue	RO	Colour matrix parameter for Red in Blue Format: 16-bit signed iReal
0x140D	Lo			
0x140E	Hi	matrix_element_GreenInBlue	RO	Colour matrix parameter for Green in Blue Format: 16-bit signed iReal
0x140F	Lo			
0x1410	Hi	matrix_element_BlueInBlue	RO	Colour matrix parameter for Blue in Blue Format: 16-bit signed iReal
0x1411	Lo			

Table 107: Colour Matrix Registers

12.7 Image Statistics Registers - [0x2000-0x2FFF]

Registers indices 0x2000 to 0x2FFF are reserved for future use by image statistics features.

12.8 Manufacturer Specific Registers – [0x3000-0x3FFF]

Register Indices 0x3000 to 0x3FFF are for manufacturer specific features or set-up.

13 Extensions Beyond the Sensor Profiles

13.1 Introduction

The features defined in this chapter a extensions beyond the requirements for the sensor profiles defined in chapter 1. Thus none of these features are directly supported by host receivers.

They are included here for reference only.

13.2 Data Format

13.2.1 Embedded Data Packing for RAW6, RAW7 and RAW12 Data Formats

The section defines how embedded data is transmitted across the RAW6, RAW7 and RAW12 data formats. The RAW6, RAW7 and RAW12 embedded data uses the same tag codes as the RAW8 and RAW10 variants.

For the RAW6 and RAW7 embedded data formats each byte of data is split into 2 4-bit nibbles. padded to either 6 or 7-bits respectively and then transmitted over 2 pixels.

For the RAW12 embedded data each byte of data is padded to 12-bits and transmitted as a one pixel.

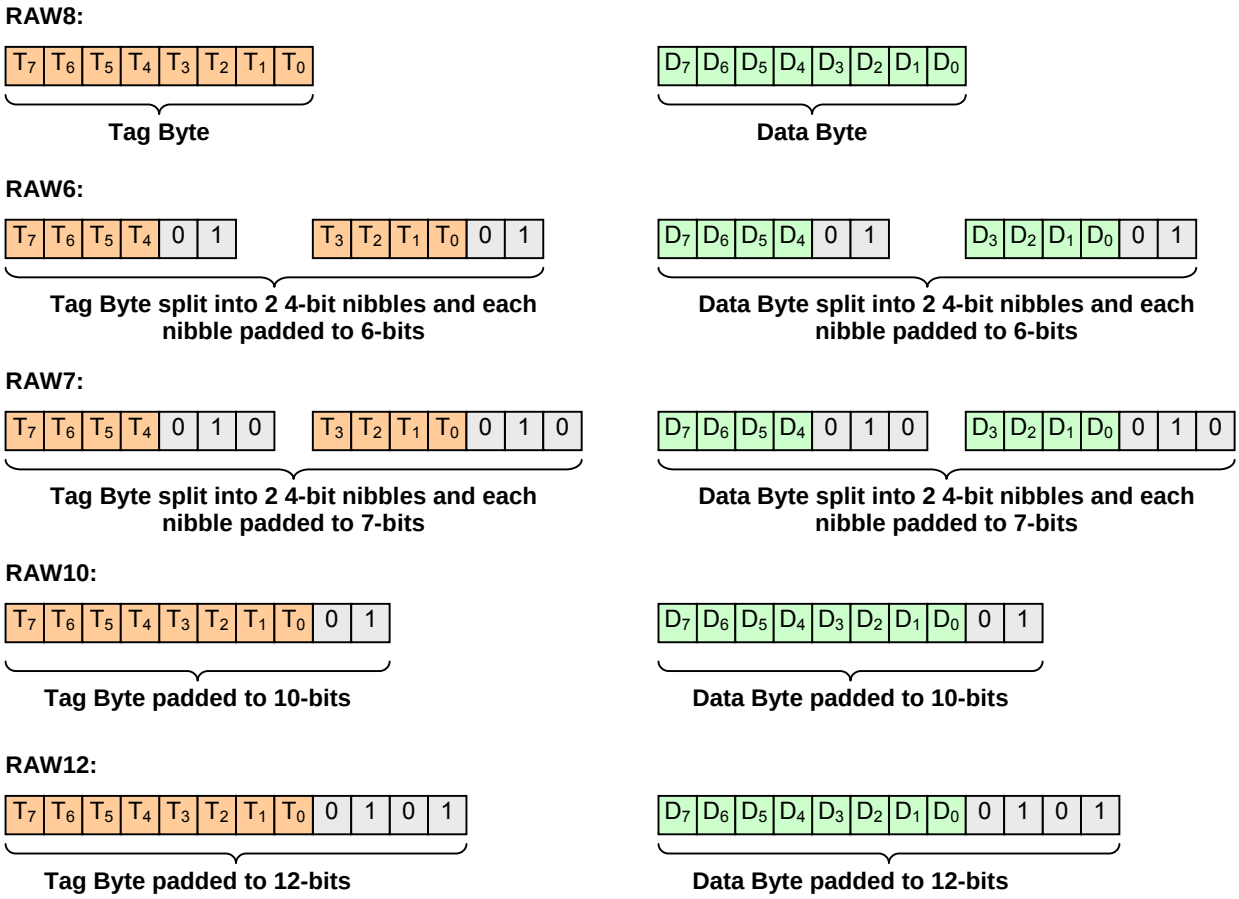


Figure 72: 2 Byte Tagged Data Packet - Data Packing for RAW6, RAW7, RAW8, RAW10 and RAW12 Data Formats

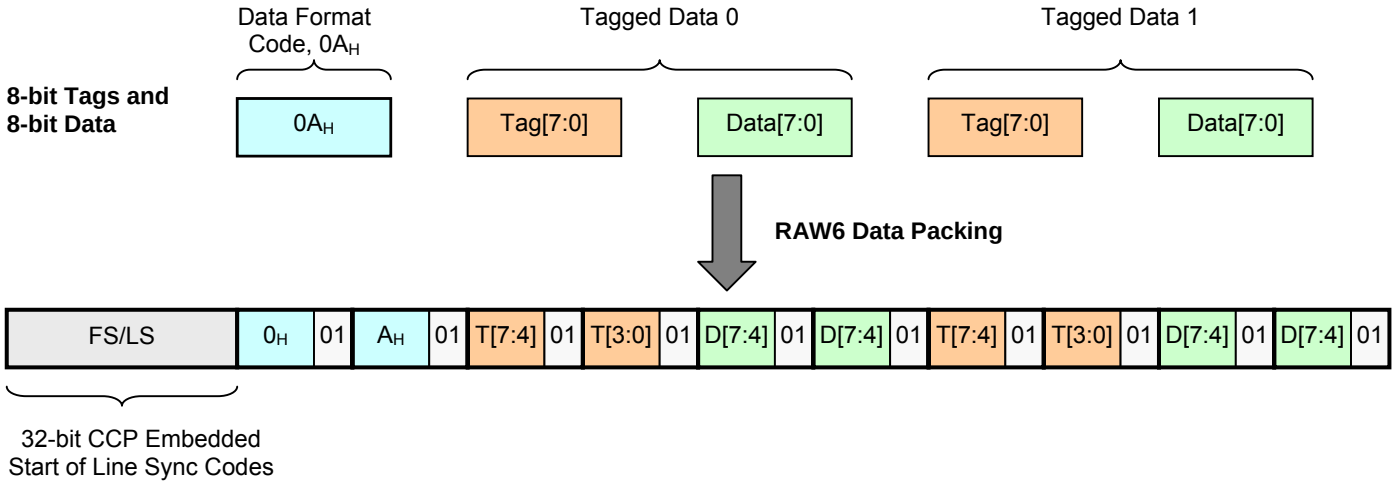


Figure 73: RAW6 Embedded Data Packing

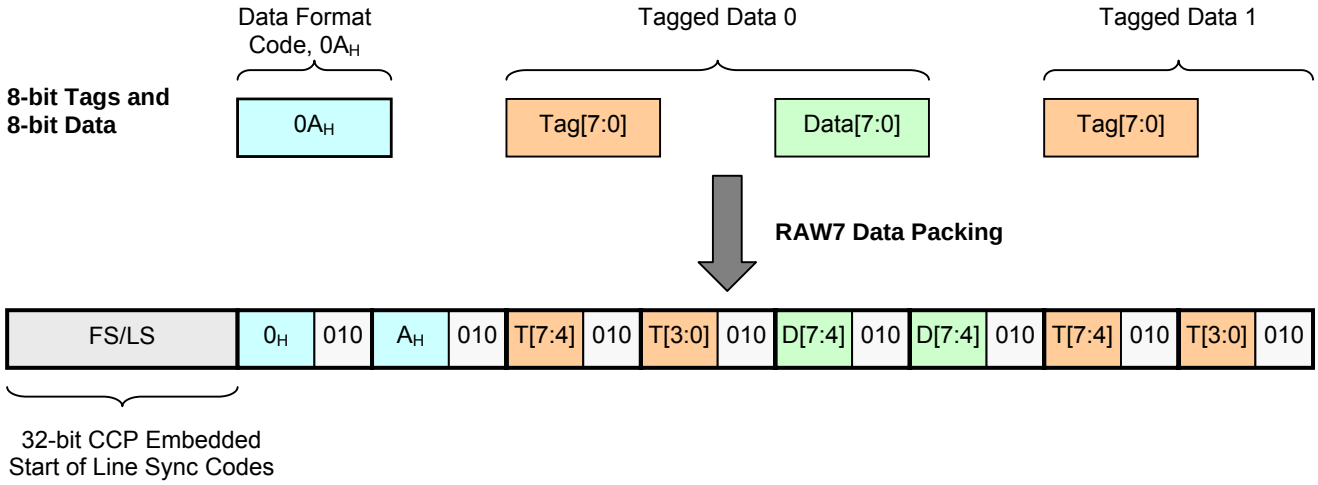


Figure 74: RAW7 Embedded Data Packing

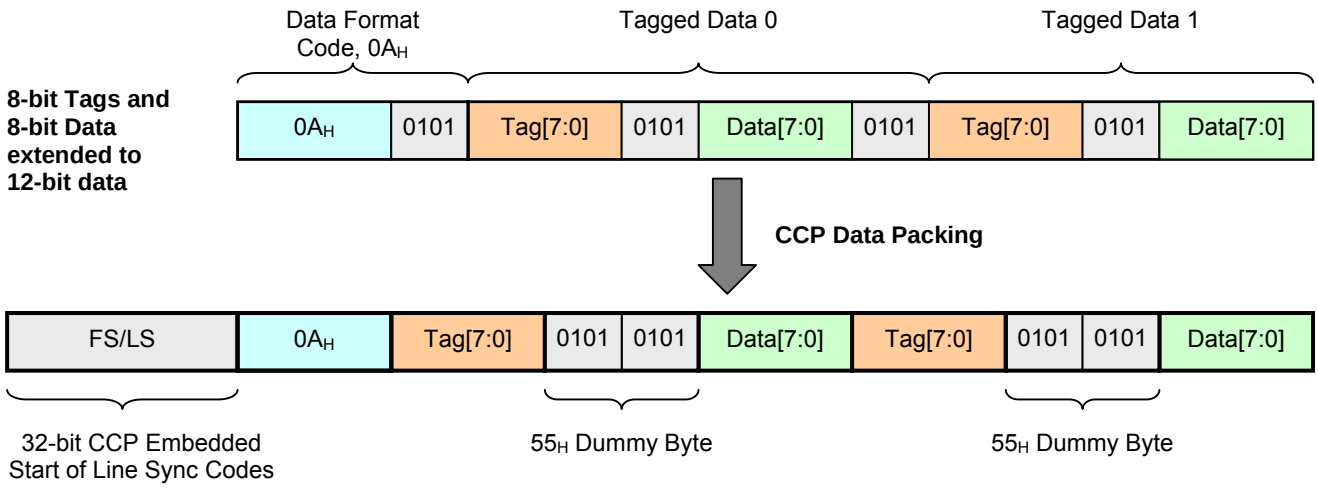


Figure 75: RAW12 Embedded Data Packing

13.3 10-bit to 7-bit DPCM/PCM Compression

13.3.1 Advanced Predictor for 10-bit to 7-bit

The 10-bit to 7-bit compression can either use the simple predictor used for 10-bit to 8-bit or the advanced predictor specified below.

The simple or advanced predictor is selected via the `compression_mode` register.

The advanced predictor uses four previous pixel values, when the prediction value is evaluated. This means that also the other colour component values are used, when the prediction value has been defined. The predictor equations can be written as below.

- 1st pixel (G_0 / B_0) in all lines is coded without prediction.
- 2nd pixel (R_1 / G_1) in all lines is predicted using the previous decoded different colour value as a prediction value.

Xpred(n) = Xdeco(n-1)

- 3rd pixel (G_2 / B_2) in all lines is predicted using the previous decoded same colour value as a prediction value.

Xpred(n) = Xdeco(n-2)

- 4th pixel (R_3 / G_3) in all lines is predicted using equation

```
if ((Xdeco(n-1) <= Xdeco(n-2) and Xdeco(n-2) <= Xdeco(n-3)) or
…….(Xdeco(n-1) >= Xdeco(n-2) and Xdeco(n-2) >= Xdeco(n-3))) then
        Xpred(n) = Xdeco(n-1)
else
        Xpred(n) = Xdeco(n-2)
```

Other pixels in all lines are predicted using equation

```
if ((Xdeco(n-1) <= Xdeco(n-2) and Xdeco(n-2) <= Xdeco(n-3)) or
        (Xdeco(n-1) >= Xdeco(n-2) and Xdeco(n-2) >= Xdeco(n-3))) then
            Xpred(n) = Xdeco(n-1)
else if ((Xdeco(n-1) <= Xdeco(n-3) and Xdeco(n-2) <= Xdeco(n-4)) or
        (Xdeco(n-1) >= Xdeco(n-3) and Xdeco(n-2) >= Xdeco(n-4))) then
            Xpred(n) = Xdeco(n-2)
else
            Xpred(n) = (Xdeco(n-2) + Xdeco(n-4) + 1) / 2
```

13.3.2 Encoder for 10 bits to 7 bits (10 – 7 – 10)

The encoder for non-predicted pixels (beginning of lines) is different than the encoder for predicted pixels.

This coder offers 30% bit rate reduction with high image quality.
The pixels without prediction are encoded as below.

```
Xenco(n) = Xorig(n) / 8
```

And for avoiding full zero code the following check has been done.

```
If (Xenco(n) == 0) then Xenco(n) = 1
```

The pixels with prediction are encoded as below.

```
If (abs(Xdiff(n)) < 8) then                 use DPCM1
Else if (abs(Xdiff(n)) < 16) then           use DPCM2
Else if (abs(Xdiff(n)) < 32) then           use DPCM3
Else if (abs(Xdiff(n)) < 160) then          use DPCM4
Else                                        use PCM
```

13.3.2.1 DPCM1 (10 to 7)

```
Xenco(n) =   "000 s xxx"
             code word "000"
             sign "s"
             value with 3 bits "xxx"
value = abs(Xdiff(n))
if(Xdiff(n) <= 0) then
        sign = 1
else
        sign = 0
```

NOTE:
Zero code has been avoided (0 is sent as -0).

13.3.2.2 DPCM2 (10 to 7)

```
Xenco(n) =   "0010 s xx"
             code word "0010"
             sign "s"
             value with 2 bits "xx"
```

```
value = abs(Xdiff(n) - 8) / 2
if(Xdiff(n) < 0) then
        sign = 1
else
        sign = 0
```

13.3.2.3 DPCM3 (10 to 7)

```
Xenco(n) =   "0011 s xx"
             code word "0011"
             sign "s"
             value with 2 bits "xx"
value = abs(Xdiff(n) - 16) / 4
if(Xdiff(n) < 0) then
        sign = 1
else
        sign = 0
```

13.3.2.4 DPCM4 (10 to 7)

```
Xenco(n) =   "01 s xxxx"
             code word "01"
             sign "s"
             value with 4 bits "xxxx"
value = abs(Xdiff(n) - 32) / 8
if(Xdiff(n) < 0) then
        sign = 1
else
        sign = 0
```

13.3.2.5 PCM (10 to 7)

```
Xenco(n) = "1 xxxxxx"
        code word "1"
        value with 6 bits "xxxxxx"

value = Xorig(n) / 16
```

13.3.3 Decoder for 7 bits to 10 bits (10 – 7 – 10)

The decoder for non-predicted pixels is different than the decoder for predicted pixels.

The pixels without prediction are decoded as below.

```
Xdeco(n) = 8 * Xenco(n) + 4
```

The pixels with prediction are decoded as below.

```
If (Xenco(n) & 0x70 == 0x00) then          use DPCM1
Else if (Xenco(n) & 0x78 == 0x10) then     use DPCM2
Else if (Xenco(n) & 0x78 == 0x18) then     use DPCM3
Else if (Xenco(n) & 0x60 == 0x20) then     use DPCM4
Else                                       use PCM
```

13.3.3.1 DPCM1 (7 to 10)

```
Xenco(n) =   "000 s xxx"
             code word "000"
             sign "s"
             value with 3 bits "xxx"
value = Xenco(n) & 0x07
sign = Xenco(n) & 0x08
if(sign > 0) then
        Xdeco(n) = Xpred(n) - value
Else
        Xdeco(n) = Xpred(n) + value
```

13.3.3.2 DPCM2 (7 to 10)

```
Xenco(n) =   "0010 s xx"
             code word "0010"
             sign "s"
             value with 2 bits "xx"
value = 2 * (Xenco(n) & 0x03) + 8
sign = Xenco(n) & 0x04
if(sign > 0) then
        Xdeco(n) = Xpred(n) - value
Else
        Xdeco(n) = Xpred(n) + value
```

13.3.3.3 DPCM3 (7 to 10)

```
Xenco(n) =   "0011 s xx"
             code word "0011"
             sign "s"
             value with 2 bits "xx"
value = 4 * (Xenco(n) & 0x03) + 16 + 1
sign = Xenco(n) & 0x04
if(sign > 0) then
        Xdeco(n) = Xpred(n) - value
        If (Xdeco(n) < 0) Xdeco(n) = 0
Else
        Xdeco(n) = Xpred(n) + value
        If (Xdeco(n) > 1023) Xdeco(n) = 1023
```

13.3.3.4 DPCM4 (7 to 10)

```
Xenco(n) =   "01 s xxxx"
             code word "01"
             sign "s"
             value with 4 bits "xxxx"
value = 8 * (Xenco(n) & 0x0f) + 32 + 3
sign = Xenco(n) & 0x10
if(sign > 0) then
        Xdeco(n) = Xpred(n) - value
        If (Xdeco(n) < 0) Xdeco(n) = 0
```

```
Else
        Xdeco(n) = Xpred(n) + value
        If (Xdeco(n) > 1023) Xdeco(n) = 1023
```

13.3.3.5 PCM (7 to 10)

```
Xenco(n) =   "1 xxxxxx"
             code word "1"
             value with 6 bits "xxxxxx"
value = 16 * (Xenco(n) & 0x3f)
If (value > Xpred(n)) then
        Xdeco(n) = value + 7
Else
        Xdeco(n) = value + 8
```

13.4 10-bit to 6-bit DPCM/PCM Compression

13.4.1 Advanced Predictor for 10-bit to 6-bit

The 10-bit to 6-bit compression can either use the simple predictor used for 10-bit to 8-bit or the advanced predictor specified below.

The simple or advanced predictor is selected via the `compression_mode` register.

The advanced predictor uses four previous pixel values, when the prediction value is evaluated. This means that also the other colour component values are used, when the prediction value has been defined. The predictor equations can be written as below.

- 1st pixel (G_0 / B_0) in all lines is coded without prediction.
- 2nd pixel (R_1 / G_1) in all lines is predicted using the previous decoded different colour value as a prediction value.

Xpred(n) = Xdeco(n-1)

- 3rd pixel (G_2 / B_2) in all lines is predicted using the previous decoded same colour value as a prediction value.

Xpred(n) = Xdeco(n-2)

- 4th pixel (R_3 / G_3) in all lines is predicted using equation

```
if ((Xdeco(n-1) <= Xdeco(n-2) and Xdeco(n-2) <= Xdeco(n-3)) or
…….(Xdeco(n-1) >= Xdeco(n-2) and Xdeco(n-2) >= Xdeco(n-3))) then
        Xpred(n) = Xdeco(n-1)
else
        Xpred(n) = Xdeco(n-2)
```

Other pixels in all lines are predicted using equation

```
if ((Xdeco(n-1) <= Xdeco(n-2) and Xdeco(n-2) <= Xdeco(n-3)) or
        (Xdeco(n-1) >= Xdeco(n-2) and Xdeco(n-2) >= Xdeco(n-3))) then
            Xpred(n) = Xdeco(n-1)
else if ((Xdeco(n-1) <= Xdeco(n-3) and Xdeco(n-2) <= Xdeco(n-4)) or
        (Xdeco(n-1) >= Xdeco(n-3) and Xdeco(n-2) >= Xdeco(n-4))) then
            Xpred(n) = Xdeco(n-2)
else
            Xpred(n) = (Xdeco(n-2) + Xdeco(n-4) + 1) / 2
```

13.4.2 Encoder for 10 bits to 6 bits (10 – 6 – 10)

The encoder for non-predicted pixels (beginning of lines) is different than the encoder for predicted pixels.

This coder offers 40% bit rate reduction with acceptable image quality.
The pixels without prediction are encoded as below.

```
Xenco(n) = Xorig(n) / 16
```

And for avoiding full zero code the following check has been done.

```
If (Xenco(n) == 0) then Xenco(n) = 1
```

The pixels with prediction are encoded as below.

```
If (abs(Xdiff(n)) < 1) then                use DPCM1
Else if (abs(Xdiff(n)) < 3) then           use DPCM2
Else if (abs(Xdiff(n)) < 11) then          use DPCM3
Else if (abs(Xdiff(n)) < 43) then          use DPCM4
Else if (abs(Xdiff(n)) < 171) then         use DPCM5
Else                                       use PCM
```

13.4.2.1 DPCM1 (10 to 6)

```
Xenco(n) = "00000 s"
           code word "00000"
           sign "s"
sign = 1
```

NOTE:
Zero code has been avoided (0 is sent as -0).

13.4.2.2 DPCM2 (10 to 6)

```
Xenco(n) =   "00001 s"
             code word "00001"
             sign "s"
if(Xdiff(n) < 0) then
           sign = 1
else
           sign = 0
```

13.4.2.3 DPCM3 (10 to 6)

```
Xenco(n) =   "0001 s x"
             code word "0001"
             sign "s"
             value with 1 bits "x"
value = abs(Xdiff(n) - 3) / 4
if(Xdiff(n) < 0) then
           sign = 1
else
           sign = 0
```

13.4.2.4 DPCM4 (10 to 6)

```
Xenco(n) =   "001 s xx"
             code word "001"
             sign "s"
             value with 2 bits "xx"
value = abs(Xdiff(n) - 11) / 8
if(Xdiff(n) < 0) then
           sign = 1
```

```
else
        sign = 0
```

13.4.2.5 DPCM5 (10 to 6)

```
Xenco(n) =   "01 s xxx"
             code word "01"
             sign "s"
             value with 3 bits "xxx"
value = abs(Xdiff(n) - 43) / 16
if(Xdiff(n) < 0) then
        sign = 1
else
        sign = 0
```

13.4.2.6 PCM (10 to 6)

```
Xenco(n) =   "1 xxxxx"
             code word "1"
             value with 5 bits "xxxxx"
value = Xorig(n) / 32
```

13.4.3 Decoder for 6 bits to 10 bits (10 – 6 – 10)

The decoder for non-predicted pixels is different than the decoder for predicted pixels.

The pixels without prediction are decoded as below.

```
Xdeco(n) = 16 * Xenco(n) + 8
```

The pixels with prediction are decoded as below.

```
If (Xenco(n) & 0x3e == 0x00) then          use DPCM1
Else if (Xenco(n) & 0x3e == 0x02) then     use DPCM2
Else if (Xenco(n) & 0x3c == 0x04) then     use DPCM3
Else if (Xenco(n) & 0x38 == 0x08) then     use DPCM4
Else if (Xenco(n) & 0x30 == 0x10) then     use DPCM5
Else                                       use PCM
```

13.4.3.1 DPCM1 (6 to 10)

```
Xenco(n) = "00000 s"
        code word "00000"
        sign "s"
Xdeco(n) = Xpred(n)
```

13.4.3.2 DPCM2 (6 to 10)

```
Xenco(n) =   "00001 s"
             code word "00001"
             sign "s"
value = 1
sign = Xenco(n) & 0x01
if(sign > 0) then
        Xdeco(n) = Xpred(n) – value
```

```
Else
        Xdeco(n) = Xpred(n) + value
```

13.4.3.3 DPCM3 (6 to 10)

```
Xenco(n) = "0001 s x"
        code word "0001"
        sign "s"
        value with 1 bits "x"
value = 4 * (Xenco(n) & 0x01) + 3 + 1
sign = Xenco(n) & 0x02
if(sign > 0) then
        Xdeco(n) = Xpred(n) - value
        If (Xdeco(n) < 0) Xdeco(n) = 0
Else
        Xdeco(n) = Xpred(n) + value
        If (Xdeco(n) > 1023) Xdeco(n) = 1023
```

13.4.3.4 DPCM4 (6 to 10)

```
Xenco(n) =   "001 s xx"
             code word "001"
             sign "s"
        value with 2 bits "xx"
value = 8 * (Xenco(n) & 0x03) + 11 + 3
sign = Xenco(n) & 0x04
if(sign > 0) then
        Xdeco(n) = Xpred(n) - value
        If (Xdeco(n) < 0) Xdeco(n) = 0
Else
        Xdeco(n) = Xpred(n) + value
        If (Xdeco(n) > 1023) Xdeco(n) = 1023
```

13.4.3.5 DPCM5 (6 to 10)

```
Xenco(n) =   "01 s xxx"
             code word "01"
             sign "s"
             value with 3 bits "xxx"
value = 16 * (Xenco(n) & 0x07) + 43 + 7
sign = Xenco(n) & 0x08
if(sign > 0) then
        Xdeco(n) = Xpred(n) - value
        If (Xdeco(n) < 0) Xdeco(n) = 0
Else
        Xdeco(n) = Xpred(n) + value
        If (Xdeco(n) > 1023) Xdeco(n) = 1023
```

13.4.3.6 PCM (6 to 10)

```
Xenco(n) = "1 xxxxx"
        code word "1"
```

```
        value with 5 bits "xxxxx"
value = 32 * (Xenco(n) & 0x1f)
If (value > Xpred(n)) then
        Xdeco(n) = value + 15
Else
        Xdeco(n) = value + 16
```

14 Additional Examples

14.1 Generic Frame Format Description

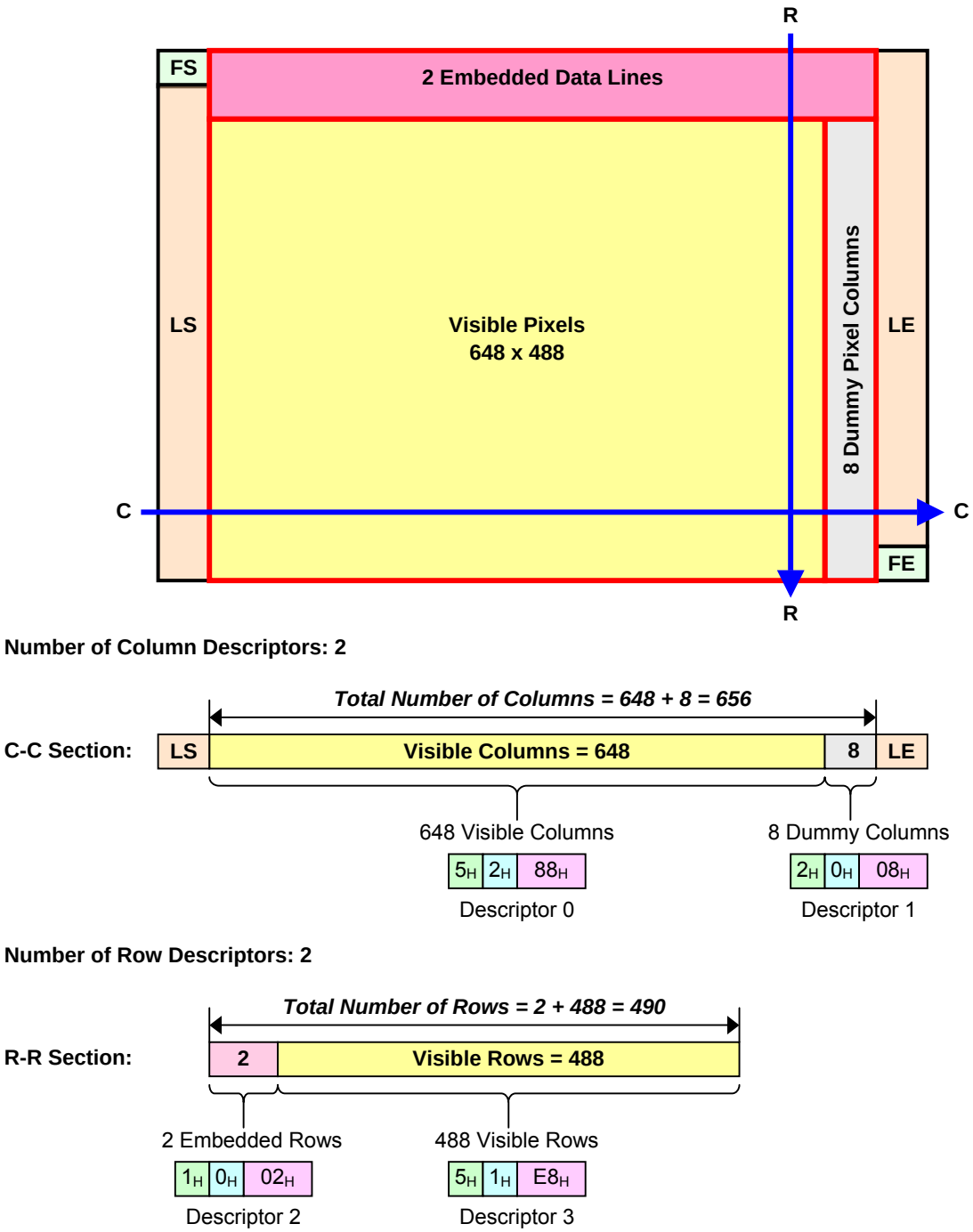
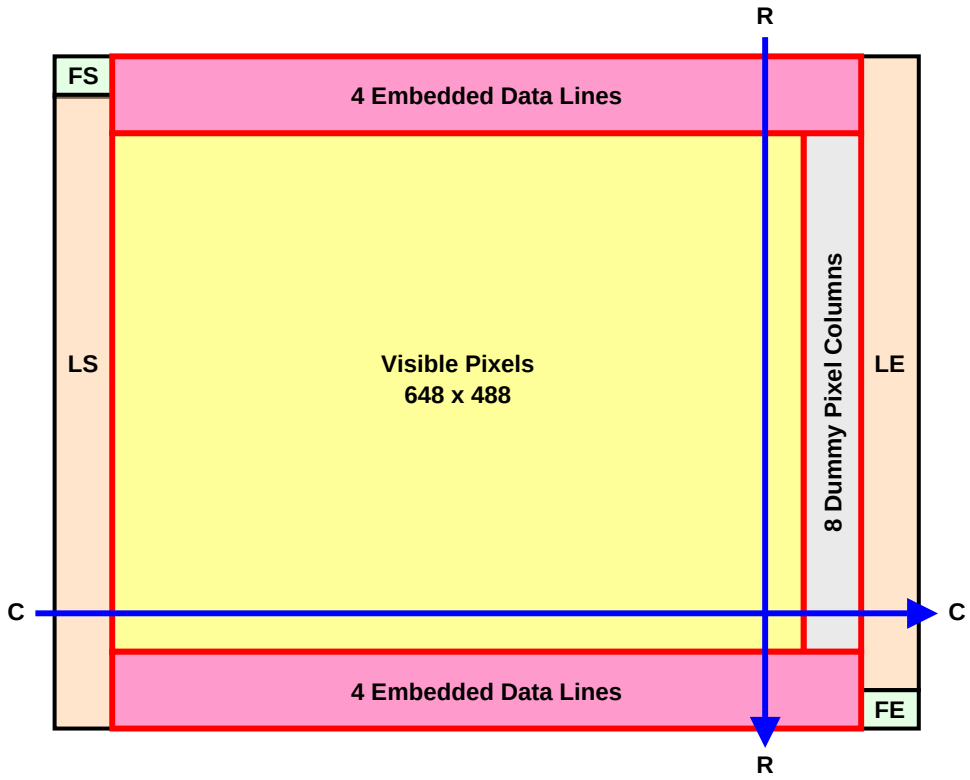
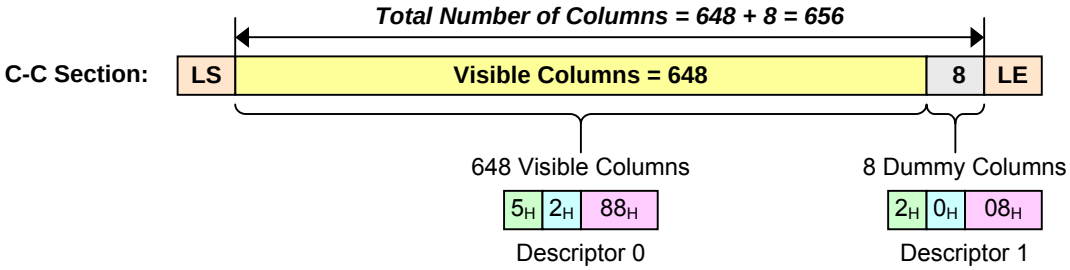


Figure 76: 2-Byte Generic Frame Format Description - Simple VGA Example



Number of Column Descriptors: 2



Number of Row Descriptors: 3

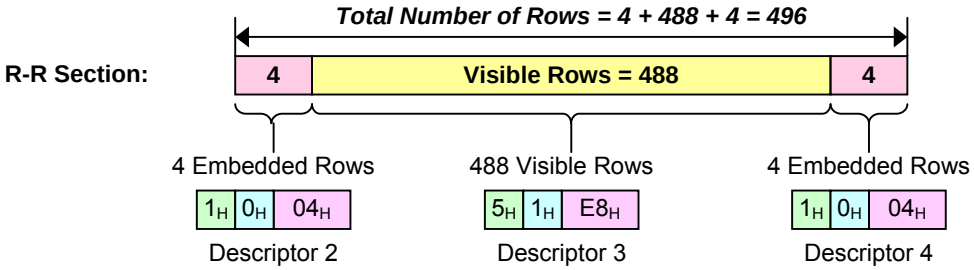
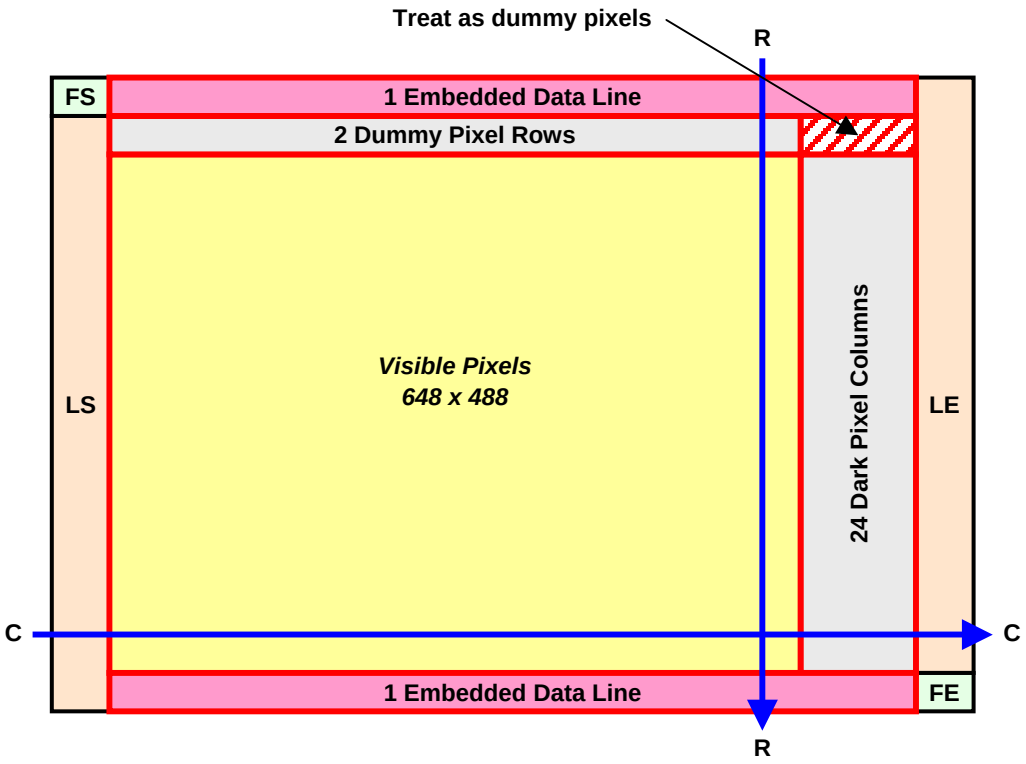
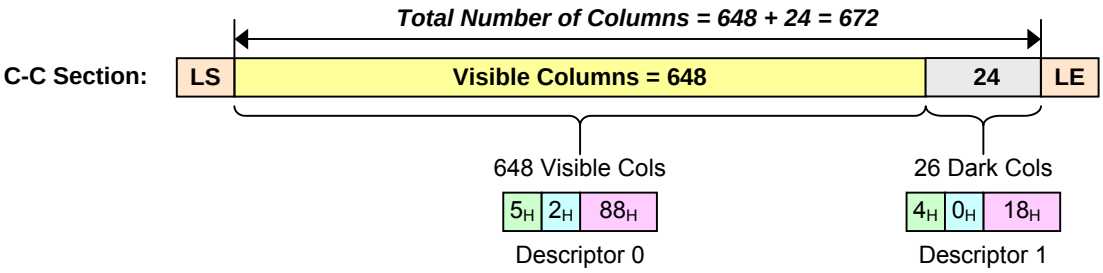


Figure 77: 2-Byte Generic Frame Format Description – VGA Example with extra SOF and EOF lines



Number of Column Descriptors: 2



Number of Row Descriptors: 4

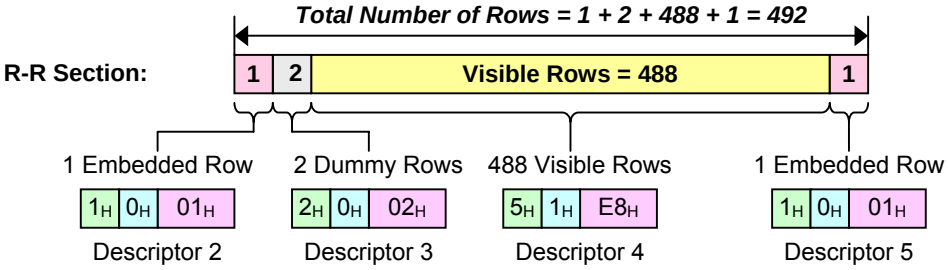
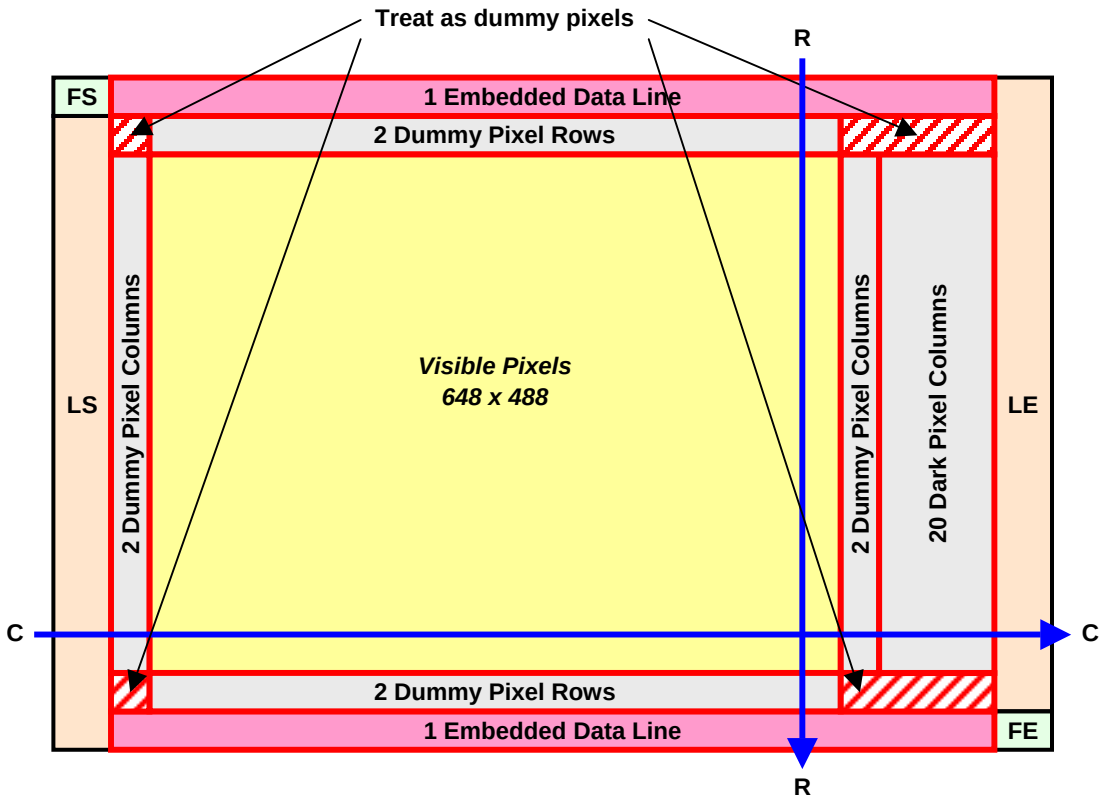
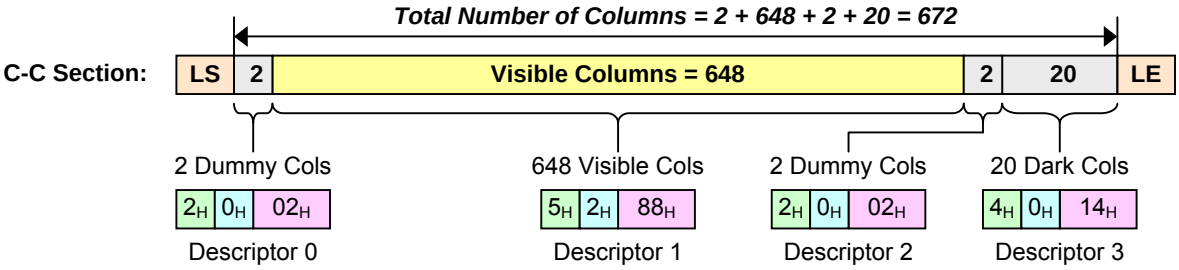


Figure 78: 2-Byte Generic Frame Format Description - Dark Column Based VGA Example 1



Number of Column Descriptors: 4



Number of Row Descriptors: 5

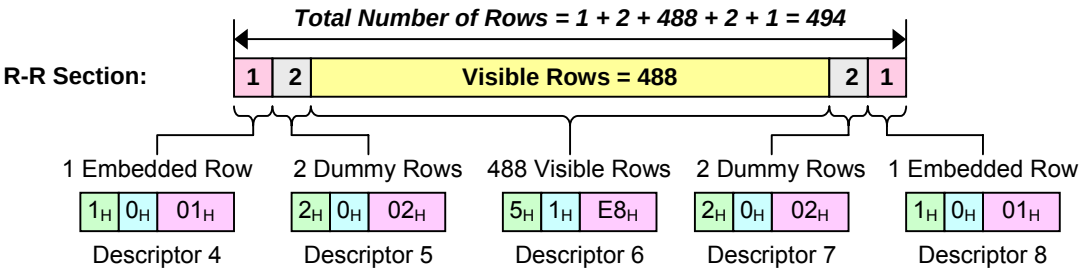
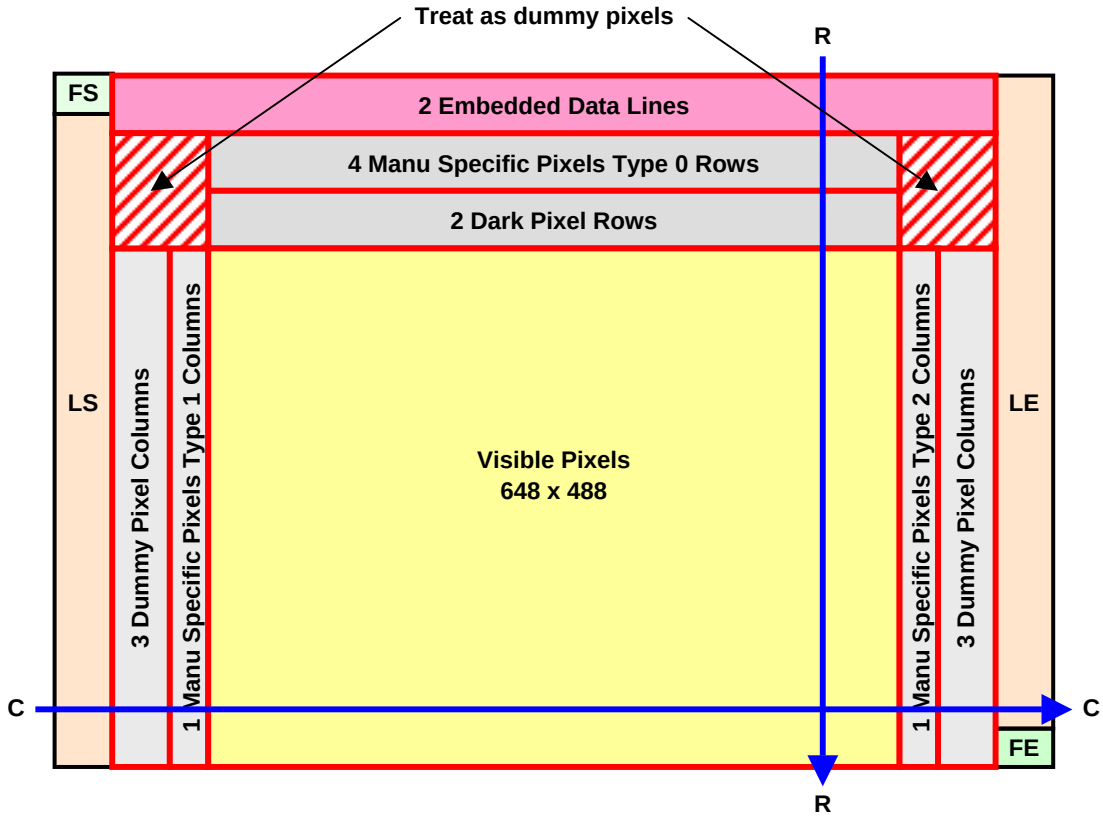
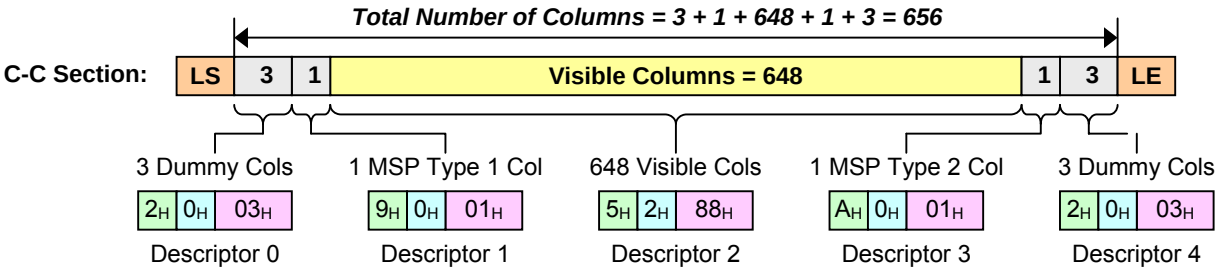


Figure 79: 2-Byte Generic Frame Format Description - Dark Column Based VGA Example 2



Number of Column Descriptors: 5



Number of Row Descriptors: 4

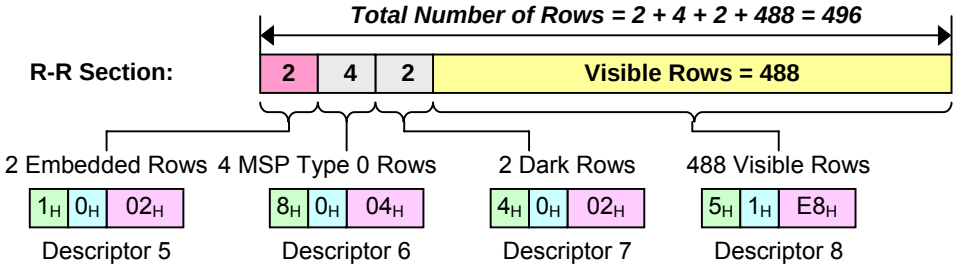
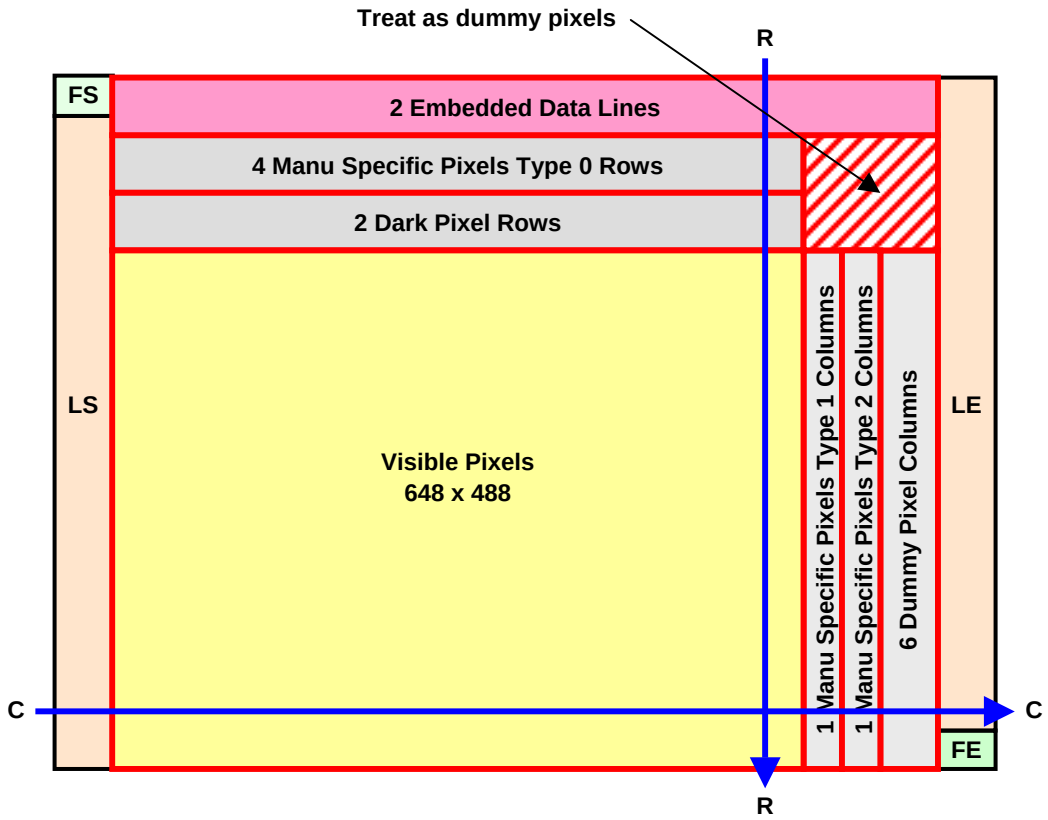
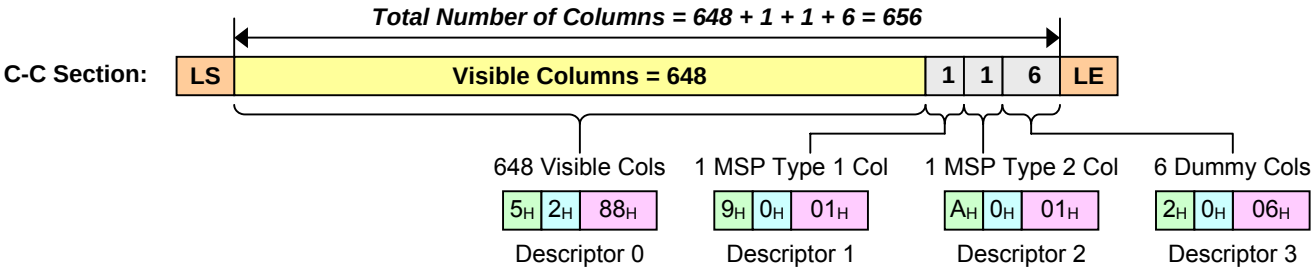


Figure 80: 2-Byte Generic Frame Format Description – Dark Row based VGA Example 1



Number of Column Descriptors: 4



Number of Row Descriptors: 4

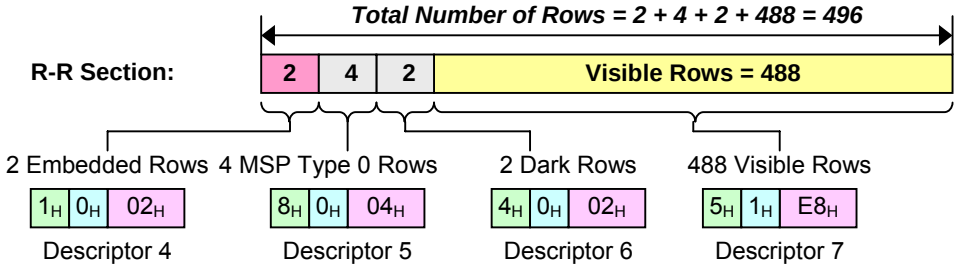


Figure 81: 2-Byte Generic Frame Format Description – Dark Row based VGA Example 2

14.2 Horizontal Scaling Frame Format Examples

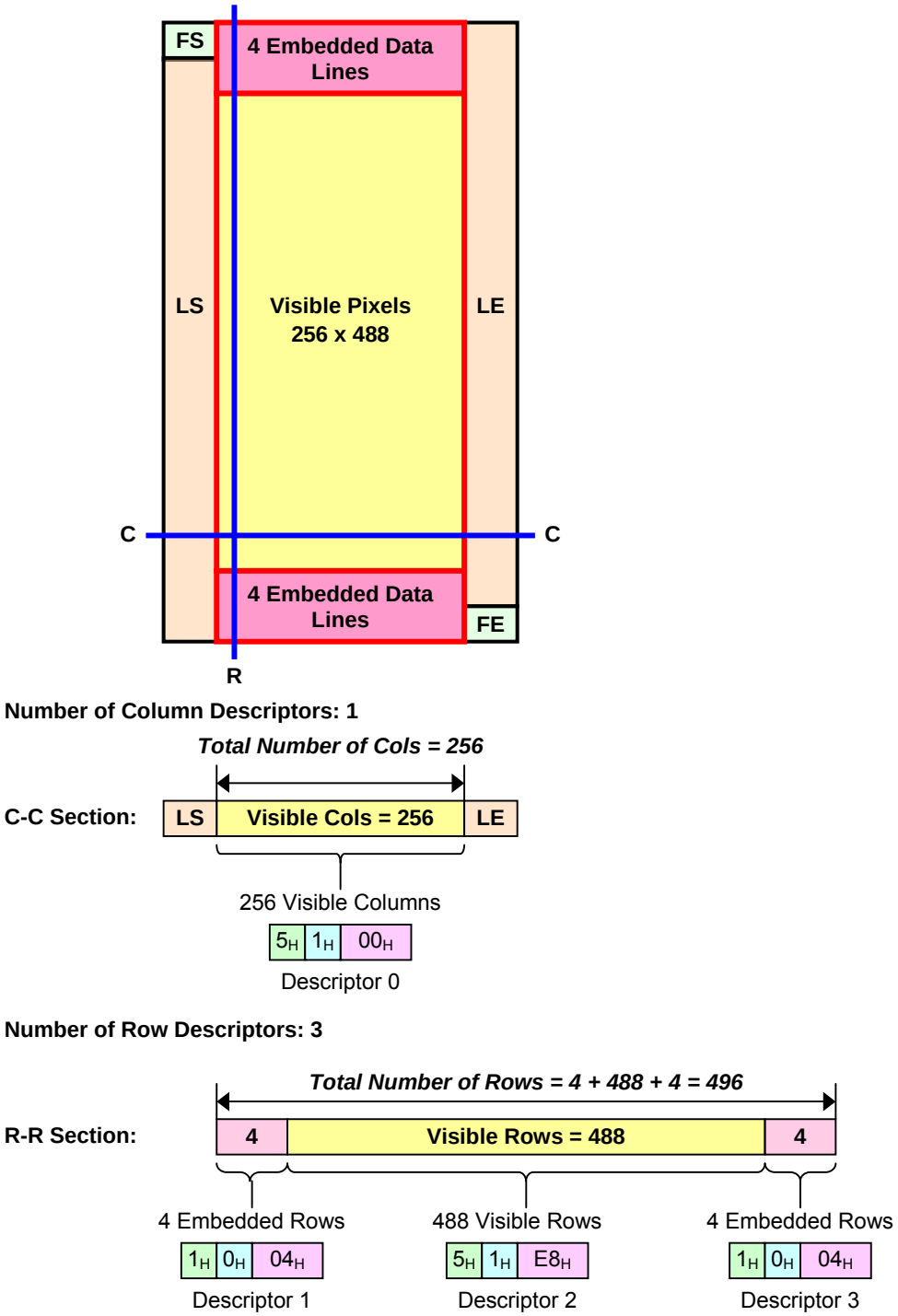
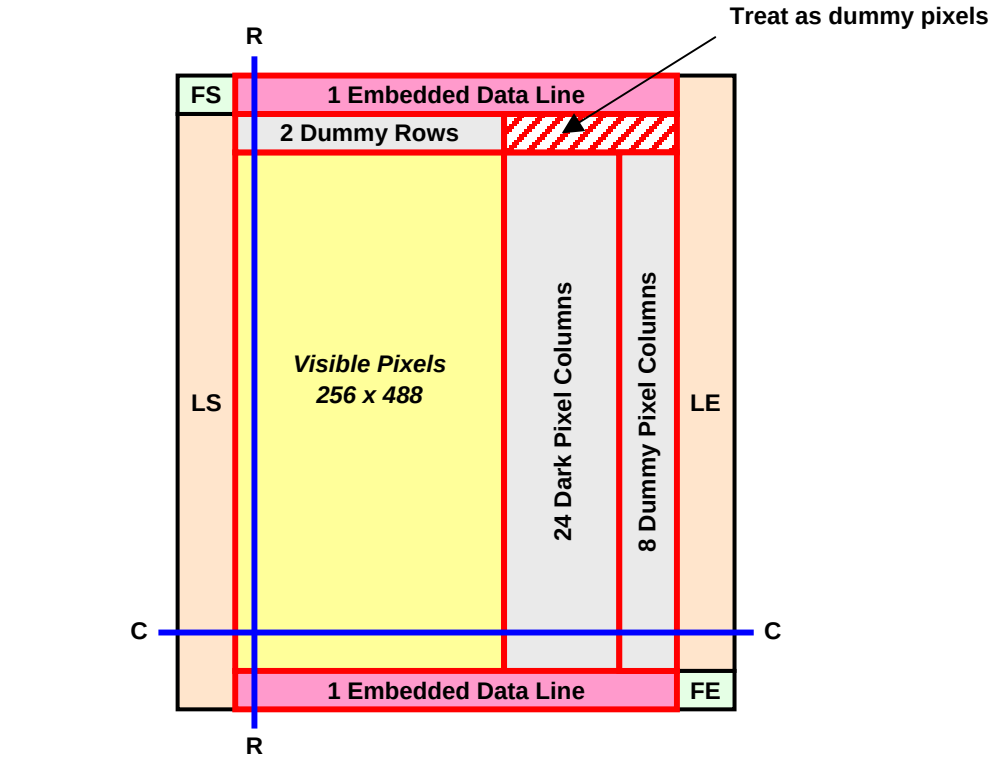
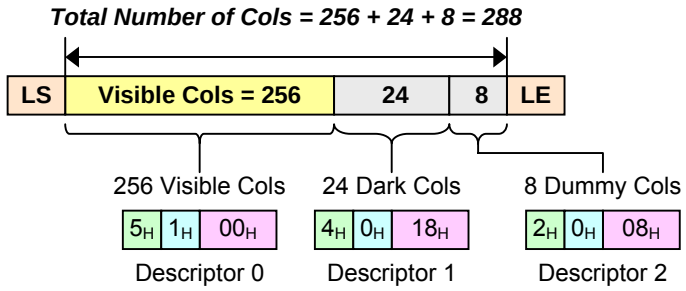


Figure 82 - SubQCIF H-scaler Example 1



Number of Column Descriptors: 3



Number of Row Descriptors: 4

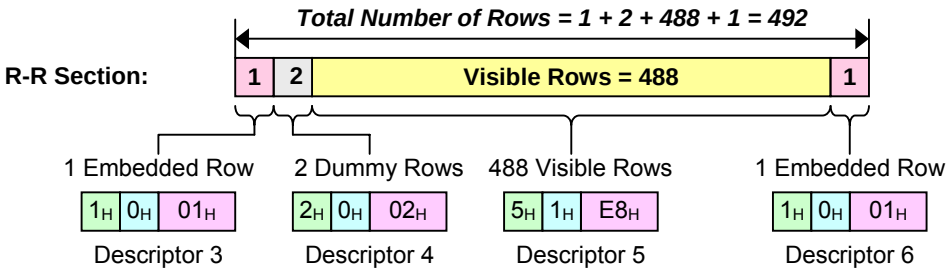


Figure 83 - SubQCIF H-scaler Example 2

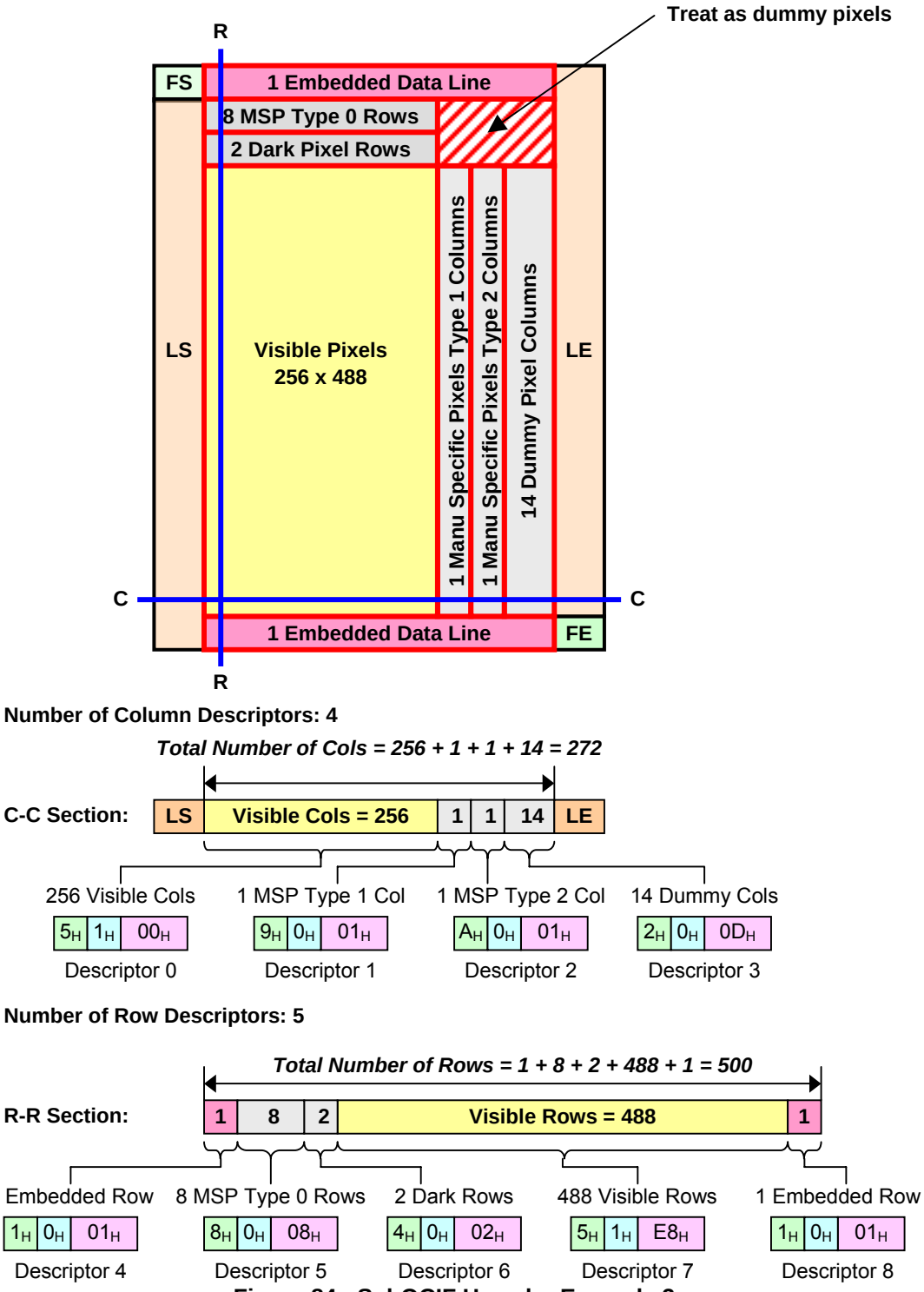
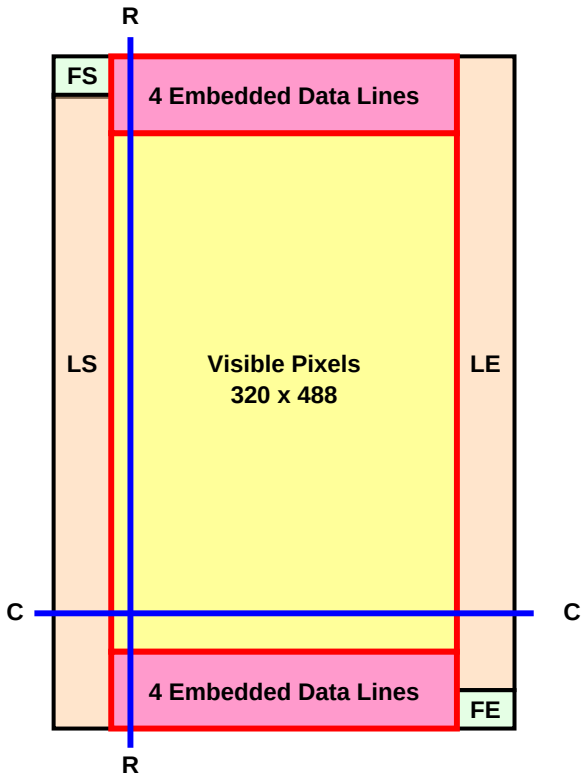
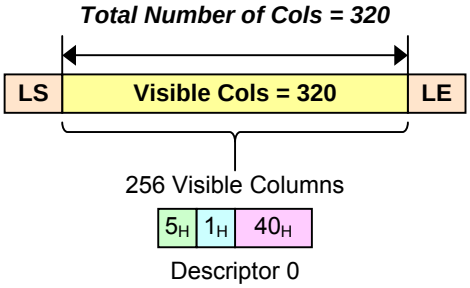


Figure 84 - SubQCIF H-scaler Example 3



Number of Column Descriptors: 1



Number of Row Descriptors: 3

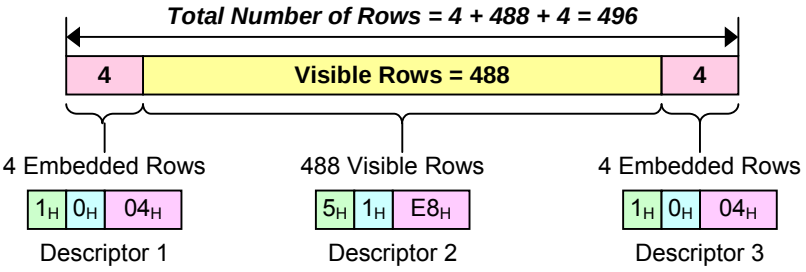
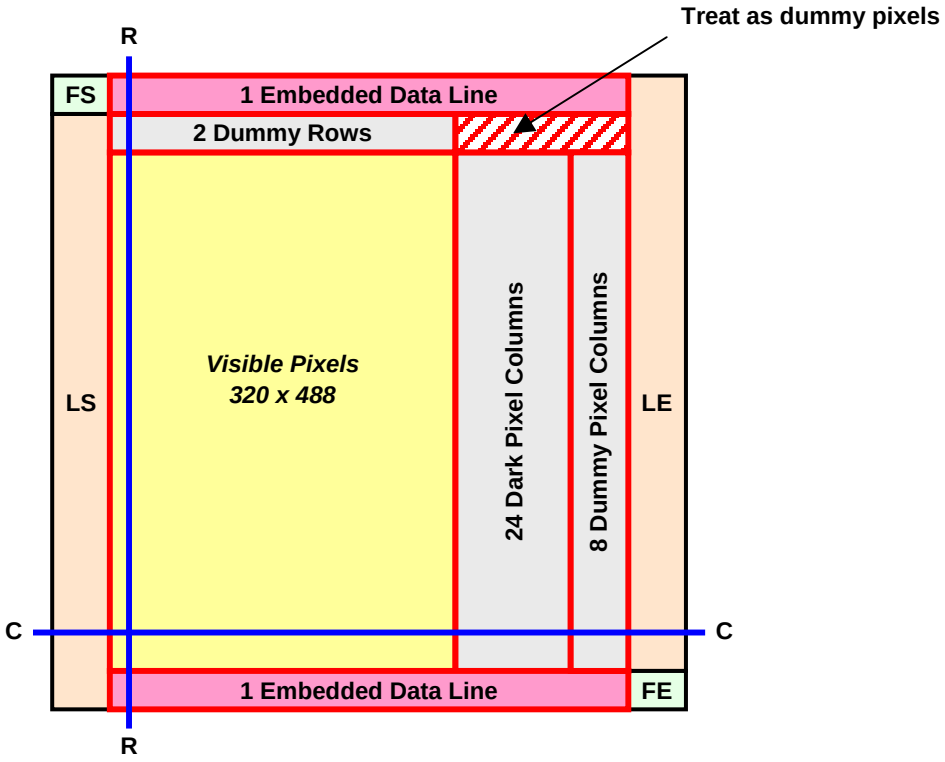
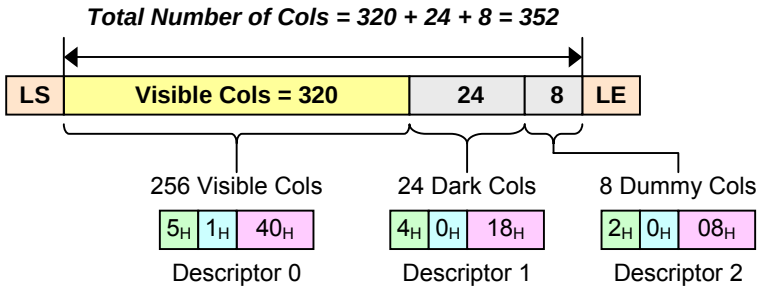


Figure 85 - QQVGA H-scaler Example 1



Number of Column Descriptors: 3



Number of Row Descriptors: 4

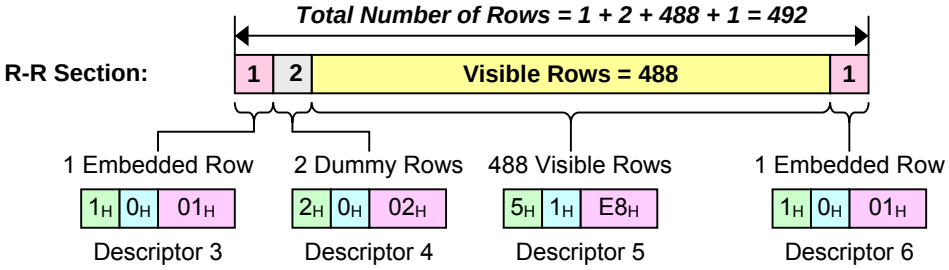


Figure 86 - QQVGA H-scaler Example 2

14.3 Full Scaler Frame Format Examples

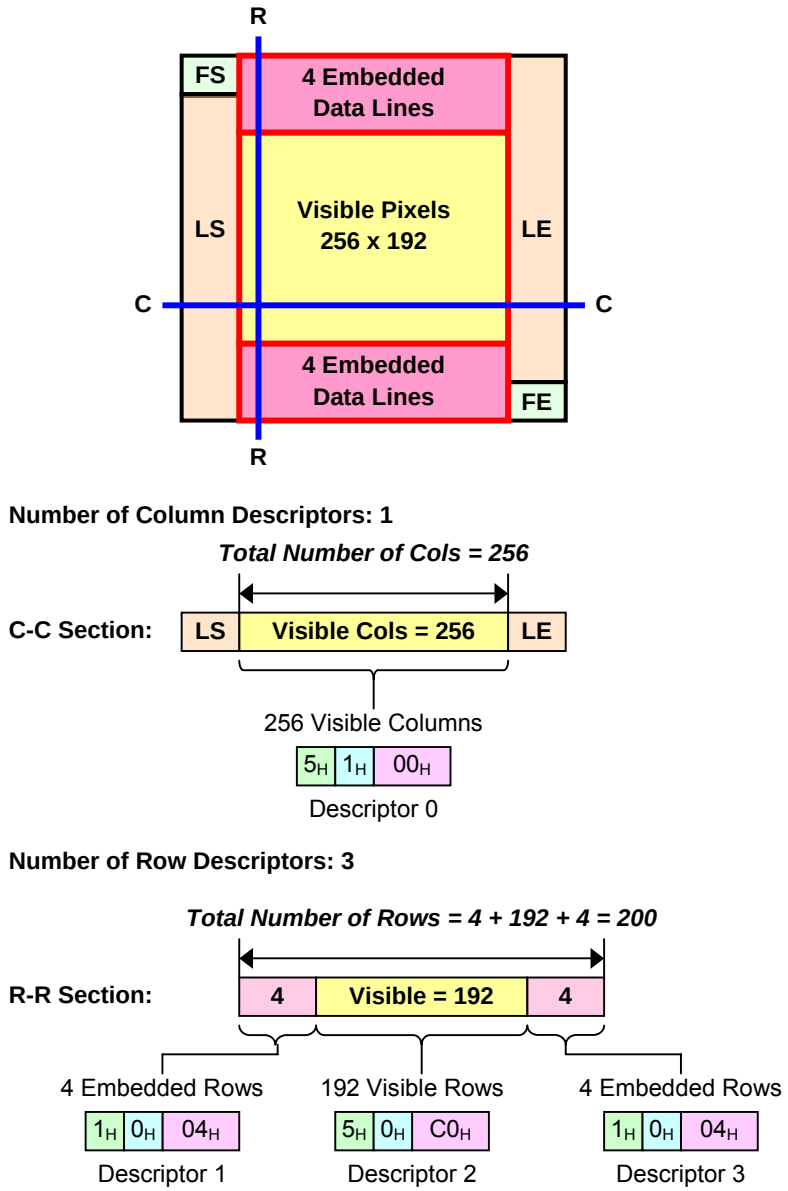


Figure 87 - SubQCIF Full-Scaler Example 1

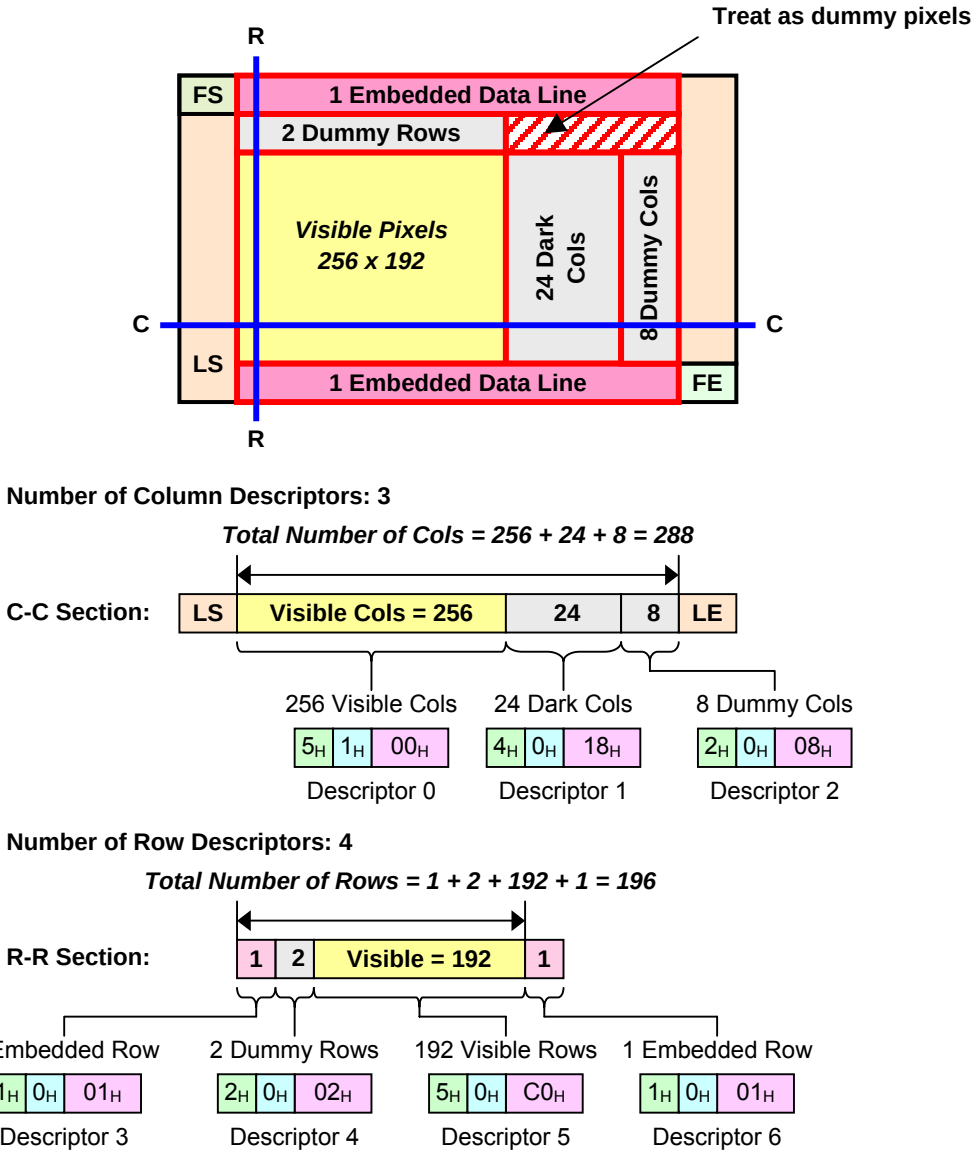
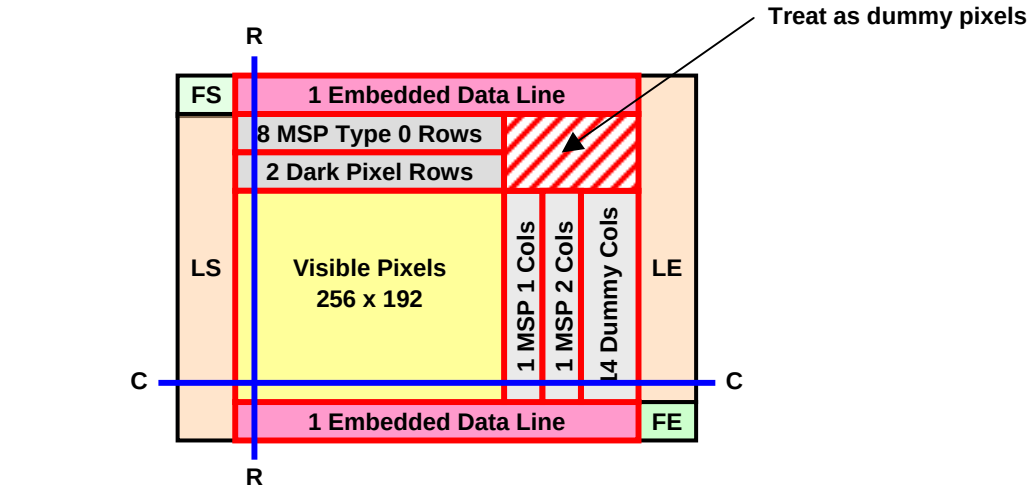
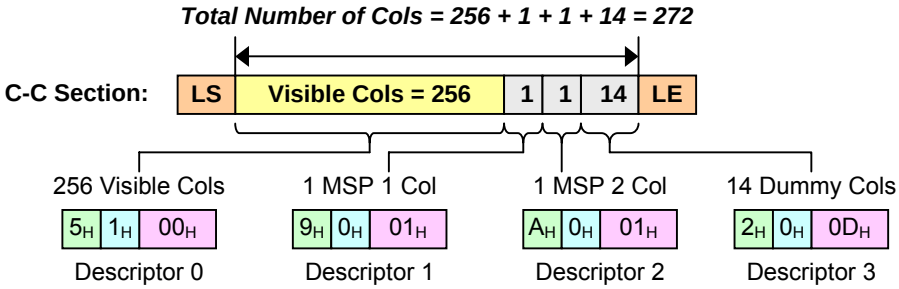


Figure 88: SubQCIF Full-Scaler Example 2



Number of Column Descriptors: 4



Number of Row Descriptors: 5

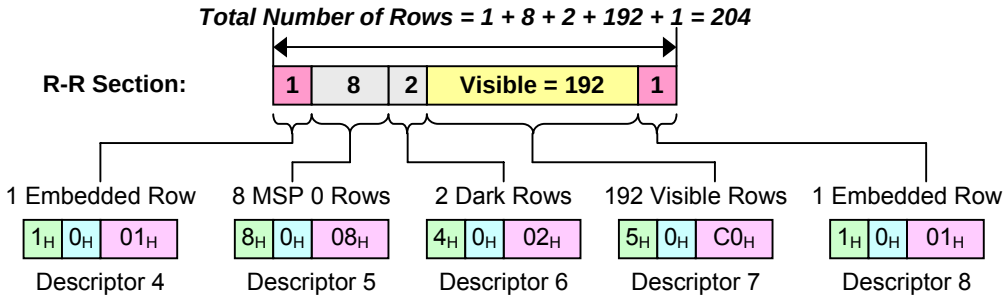
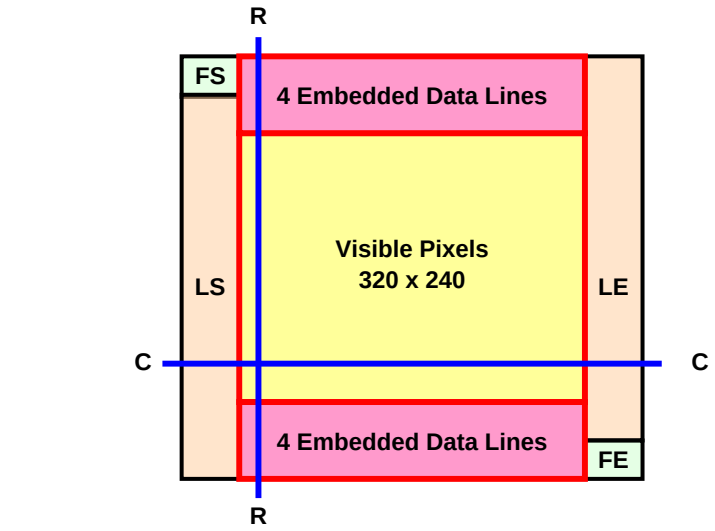
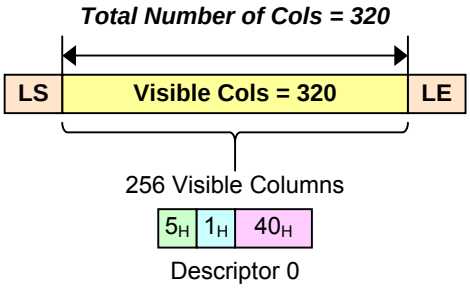


Figure 89: SubQCIF Full Scaler Example 3



Number of Column Descriptors: 1



Number of Row Descriptors: 3

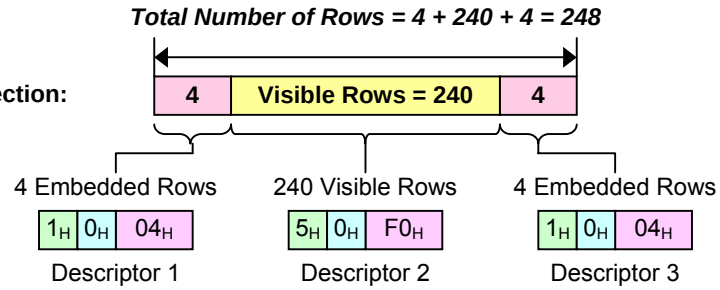


Figure 90: QQVGA Full-Scaler Example 1

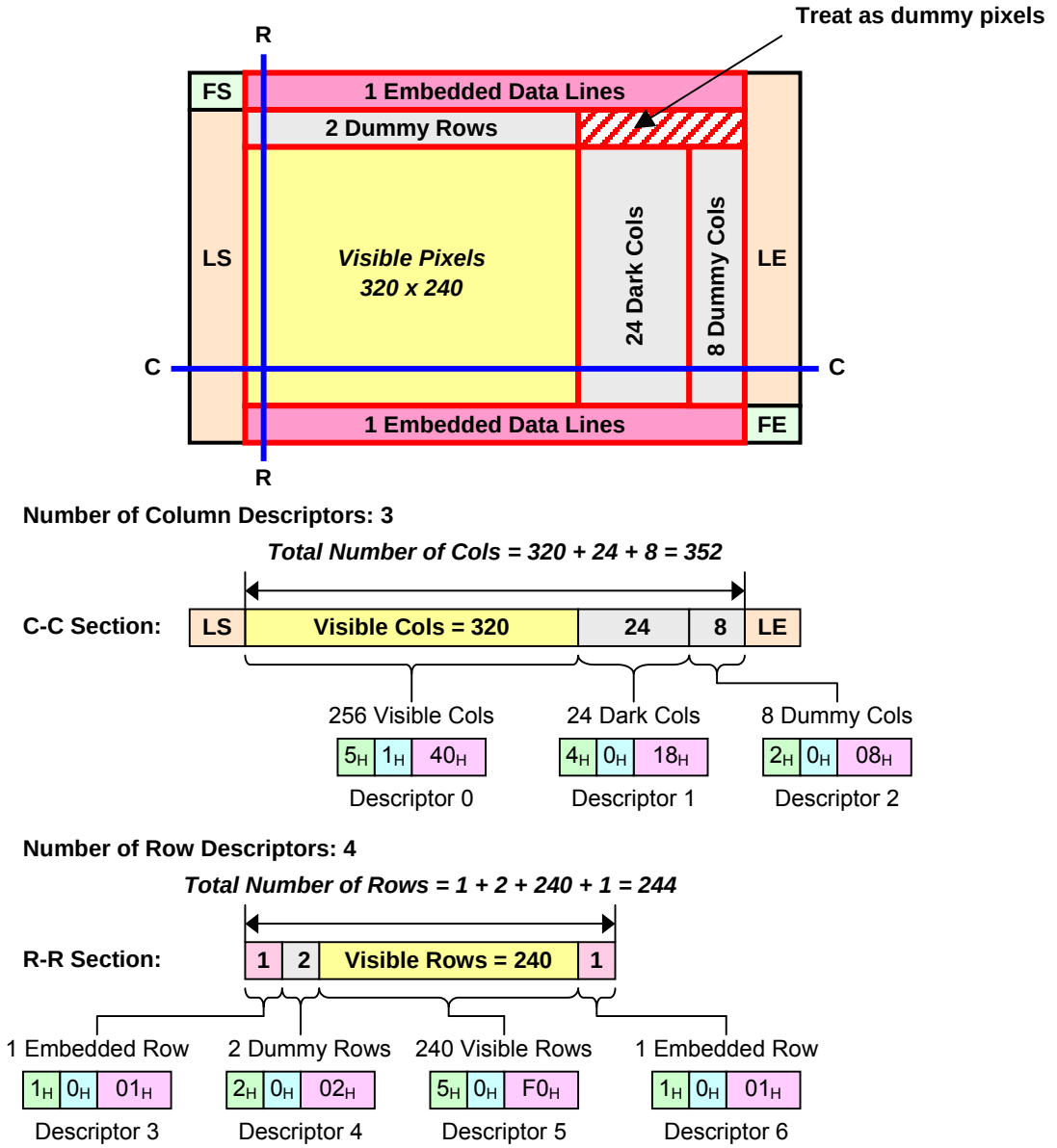


Figure 91 - QQVGA Full Scaler Example 2

ANNEX