



**Specification for  
Camera Command Set  
(MIPI CCS<sup>SM</sup>)**

**Version 1.1.1 – 4 January 2023**

MIPI Board Adopted 17 April 2023  
**Public Release Edition**

**\* NOTE TO IMPLEMENTERS \***

This document is a MIPI Specification. MIPI member companies' rights and obligations apply to this Specification as defined in the MIPI Membership Agreement and MIPI Bylaws.

This page intentionally left blank.



# Specification for Camera Command Set (MIPI CCS<sup>SM</sup>)

**Version 1.1.1  
4 January 2023**

MIPI Board Adopted 17 April 2023  
**Public Release Edition**

Further technical changes to this document are expected as work continues in the Camera Working Group.

**NOTICE OF DISCLAIMER**

The material contained herein is provided on an “AS IS” basis. To the maximum extent permitted by applicable law, this material is provided AS IS AND WITH ALL FAULTS, and the authors and developers of this material and MIPI Alliance Inc. (“MIPI”) hereby disclaim all other warranties and conditions, either express, implied or statutory, including, but not limited to, any (if any) implied warranties, duties or conditions of merchantability, of fitness for a particular purpose, of accuracy or completeness of responses, of results, of workmanlike effort, of lack of viruses, and of lack of negligence. ALSO, THERE IS NO WARRANTY OR CONDITION OF TITLE, QUIET ENJOYMENT, QUIET POSSESSION, CORRESPONDENCE TO DESCRIPTION OR NON-INFRINGEMENT WITH REGARD TO THIS MATERIAL.

IN NO EVENT WILL ANY AUTHOR OR DEVELOPER OF THIS MATERIAL OR MIPI BE LIABLE TO ANY OTHER PARTY FOR THE COST OF PROCURING SUBSTITUTE GOODS OR SERVICES, LOST PROFITS, LOSS OF USE, LOSS OF DATA, OR ANY INCIDENTAL, CONSEQUENTIAL, DIRECT, INDIRECT, OR SPECIAL DAMAGES WHETHER UNDER CONTRACT, TORT, WARRANTY, OR OTHERWISE, ARISING IN ANY WAY OUT OF THIS OR ANY OTHER AGREEMENT RELATING TO THIS MATERIAL, WHETHER OR NOT SUCH PARTY HAD ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

The material contained herein is not a license, either expressly or impliedly, to any IPR owned or controlled by any of the authors or developers of this material or MIPI. Any license to use this material is granted separately from this document. This material is protected by copyright laws, and may not be reproduced, republished, distributed, transmitted, displayed, broadcast or otherwise exploited in any manner without the express prior written permission of MIPI Alliance. MIPI, MIPI Alliance and the dotted rainbow arch and all related trademarks, service marks, tradenames, and other intellectual property are the exclusive property of MIPI Alliance Inc. and cannot be used without its express prior written permission. The use or implementation of this material may involve or require the use of intellectual property rights (“IPR”) including (but not limited to) patents, patent applications, or copyrights owned by one or more parties, whether or not members of MIPI. MIPI does not make any search or investigation for IPR, nor does MIPI require or request the disclosure of any IPR or claims of IPR as respects the contents of this material or otherwise.

Without limiting the generality of the disclaimers stated above, users of this material are further notified that MIPI: (a) does not evaluate, test or verify the accuracy, soundness or credibility of the contents of this material; (b) does not monitor or enforce compliance with the contents of this material; and (c) does not certify, test, or in any manner investigate products or services or any claims of compliance with MIPI specifications or related material.

Questions pertaining to this material, or the terms or conditions of its provision, should be addressed to:

MIPI Alliance, Inc.  
c/o IEEE-ISTO  
445 Hoes Lane, Piscataway New Jersey 08854, United States  
Attn: Executive Director

# Contents

<b>Figures.....</b>	<b>X</b>
<b>Tables.....</b>	<b>xiv</b>
<b>Release History.....</b>	<b>xxiv</b>
<b>1    Introduction .....</b>	<b>1</b>
1.1    Scope.....	1
1.2    Purpose.....	3
<b>2    Terminology .....</b>	<b>4</b>
2.1    Use of Special Terms .....	4
2.2    Definitions.....	4
2.3    Abbreviations .....	7
2.4    Acronyms.....	7
<b>3    References.....</b>	<b>9</b>
<b>4    Technical Overview .....</b>	<b>11</b>
4.1    System View.....	11
4.2    Key Concepts .....	12
4.2.1    Minimum Required Features .....	12
4.2.2    Intelligent Status Line.....	12
4.2.3    Synchronization .....	13
4.2.4    Parameter Re-Timing.....	13
4.2.5    Generic Frame Format Description.....	13
4.2.6    Capability Registers/Parameter Limits .....	13
4.2.7    Generic Control Interface .....	13
4.2.8    Configuration/Tuning Files.....	14
4.2.9    Recommended Host Behavior .....	14
<b>5    Image Sensor Operating Modes .....</b>	<b>15</b>
5.1    Introduction.....	15
5.1.1    Power-Off Mode .....	16
5.1.2    Hardware Standby Mode .....	17
5.1.3    Software Standby Mode.....	17
5.1.4    Streaming Mode.....	17
5.2    Changing Operating Modes .....	18
5.2.1    Power Up Sequence .....	18
5.2.2    Power Down Sequence .....	23
5.2.3    Mode Change Sequence.....	26
5.2.4    Rolling Shutter Mode Transitions.....	30
5.2.5    Mode Select Register .....	31
5.2.6    Software-Reset Via CCI Interface.....	31
5.2.7    Frame Count Register .....	33
5.2.8    XSHUTDOWN.....	33
5.2.9    EXTCLK.....	34
5.2.10    GPIO_TRIGGER.....	34
<b>6    Module and Sensor Version and Identification.....</b>	<b>35</b>
6.1    Module Version Identification Registers.....	35
6.2    Image Sensor Identification Registers .....	37

<b>7 Data Transmission and Data Formats .....</b>	<b>39</b>
7.1 Introduction .....	39
7.2 CSI-2 Introduction .....	39
7.3 Size Rules.....	41
7.3.1 RAW10 Example (Informative).....	41
7.3.2 RAW12 Example (Informative).....	42
7.4 Signaling Options.....	43
7.4.1 CSI Signaling Mode.....	43
7.4.2 CSI Lane Mode .....	43
7.4.3 CSI Signaling Mode Capability .....	44
7.4.4 CSI D-PHY Lane Mode .....	44
7.4.5 CSI C-PHY Lane Mode .....	45
7.5 Data Format, Data Type, and Virtual Channel Control .....	46
7.5.1 CSI Data Format .....	46
7.5.2 DPCM Frame Data Type.....	47
7.5.3 CSI Channel Identifier .....	47
7.5.4 Bottom Embedded Data Control.....	47
7.5.5 Data Type and Virtual Channel Programmability .....	48
7.6 ADC Controls .....	48
7.7 CSI-2 v2.0 Feature Controls .....	49
7.7.1 CSI-2 LRTE Feature .....	49
7.7.2 CSI-2 ALPS Feature .....	52
7.7.3 CSI-2 Data Scrambling Feature .....	53
7.8 PHY Calibration Controls .....	57
7.8.1 D-PHY Skew Calibration.....	57
7.8.2 D-PHY Alternate Calibration.....	57
7.8.3 C-PHY Calibration Controls.....	58
7.8.4 PHY Calibration Registers .....	59
7.9 PHY Control.....	62
7.9.1 PHY Timing Controls .....	62
7.9.2 D-PHY Timing Registers .....	67
7.9.3 C-PHY Timing Control Registers .....	74
7.9.4 Equalization .....	80
7.9.5 D-PHY Preamble .....	81
7.9.6 D-PHY Spread Spectrum Clocking (SSC).....	81
7.9.7 Manual LP Control.....	82
7.10 Data Format Description – Type 1 .....	83
7.11 Data Format Description – Type 2 .....	85
7.12 Data Encoding for Uncompressed Image Data.....	86
7.12.1 Data Pedestal.....	86
7.13 Frame Format Overview .....	88
7.13.1 CSI-2 Frame Format .....	88
7.13.2 Frame Format Elements .....	91
7.14 Frame Format Description .....	93
7.14.1 2-Byte Generic Frame Format Description.....	93
7.14.2 4-Byte Generic Frame Format Description.....	98
7.15 Embedded Data Line Formats.....	100
7.15.1 Simplified 2-Byte Tagged Data Format .....	101

<b>8    Video Timing.....</b>	<b>107</b>
8.1    Introduction.....	107
8.2    Image Readout .....	108
8.2.1    Programmable Image Size .....	108
8.2.2    Mirror/Flip .....	117
8.2.3    Sub-Sampled Readout.....	120
8.2.4    Sub-Sampling with Mirror and Flip.....	125
8.2.5    Binning Readout .....	127
8.2.6    Line Length and Frame Length.....	134
8.3    Video Timing and Output Pixel Clock Frequency Control.....	138
8.3.1    Introduction to Clock Trees .....	140
8.3.2    OP Domain Details and Additional Options .....	145
8.3.3    Frequencies .....	158
8.3.4    Clock Tree Control Registers.....	160
8.3.5    Clock Set-Up Capability Read Only Registers .....	165
8.3.6    Clock Frequency Requirements .....	171
8.4    Overall Video Timing Requirements.....	172
<b>9    Integration Time and Gain Control.....</b>	<b>173</b>
9.1    Overview.....	173
9.2    Integration Time Control.....	173
9.2.1    Calculation of Integration Time .....	174
9.3    Analog Gain Control.....	175
9.3.1    Global Analog Gain .....	176
9.3.2    Alternate Global Analog Gain.....	178
9.4    Digital Gain Control.....	180
9.4.1    Digital Gain Control Parameters.....	180
9.4.2    Digital Gain Parameter Limits .....	180
9.4.3    Digital Gain and Black Level.....	180
9.5    Re-Timing of the Gain and Integration Time Controls .....	181
9.5.1    Re-Timing Rule 1: Re-Time Changes to Frame Boundaries .....	181
9.5.2    Re-Timing Rule 2: Specify the Latching Moment.....	181
9.5.3    Re-Timing Rule 3: Grouped Parameter Hold .....	181
9.5.4    Re-Timing Rule 4: Masking of Corrupted Frames .....	182
9.5.5    Re-Timing Rule 5: Image Parameter Update Time Limit.....	182
9.5.6    Re-Timing Rule 6: Integration Time Registers .....	183
9.5.7    Re-Timing Rule 7: Gain Registers .....	183
9.6    Control Synchronization .....	183
9.6.1    Time Parameter Limit Discovery.....	184
9.6.2    Operation Outside Limits.....	185
9.7    Advanced Timing Modes .....	186
9.7.1    Automatic Frame Length Mode.....	187
9.7.2    Manual Readout Start Mode .....	188
9.7.3    Delayed Exposure Start Mode .....	191
9.8    Auto-Bracketing Function.....	192
9.9    HDR Timing Mode and HDR Synthesis.....	198
9.9.1    Impact of HDR Function on Other CCS Features .....	204

<b>10 Test Modes .....</b>	<b>205</b>
10.1 Full Frame Deterministic Test Patterns .....	205
10.1.1 Scaled & Cropped and Re-Orientated Output.....	206
10.1.2 Solid Color Mode.....	206
10.1.3 100% Color Bars Mode .....	207
10.1.4 Fade to Grey Color Bar Mode .....	209
10.1.5 100% Color Tile Mode.....	215
10.1.6 PN9 Mode .....	217
<b>11 Image Scaling .....</b>	<b>219</b>
11.1 Introduction.....	219
11.2 Digital Crop.....	220
11.3 Scaler Sampling .....	222
11.4 Down Scaler Factor.....	222
11.5 Control Registers.....	223
11.6 FIFO Control Registers.....	224
<b>12 Image Compression .....</b>	<b>225</b>
12.1 Introduction.....	225
12.2 Compression Algorithms.....	225
<b>13 Data Transfer Interface.....</b>	<b>227</b>
13.1 Introduction.....	227
13.2 Capabilities for Data Transfer Interface .....	227
13.3 Control Register and Usage of Data Transfer Interface .....	228
13.3.1 Read Sequence.....	229
13.3.2 Write Sequence .....	229
13.3.3 Error and Special Cases .....	230
<b>14 Image Processing and Sensor Corrections .....</b>	<b>231</b>
14.1 Introduction.....	231
14.2 Sensor Processing Control Interface .....	231
14.3 Shading Correction .....	233
14.4 Green Imbalance Correction .....	233
14.5 Defect Correction.....	234
14.5.1 Mapped Defect Correction.....	234
14.5.2 Dynamic Couplet Correction .....	234
14.5.3 Single Pixel Defect On-the-Fly Correction.....	234
14.5.4 Combined Dynamic Couplet and Single Pixel Defect Correction.....	235
14.5.5 Dynamic Triplet Defect Correction.....	235
14.6 Noise Filter.....	236
14.7 Module Specific Correction .....	236
14.8 Scene Color Temperature Feedback.....	237
14.9 Optical Black Pixel Readout .....	238
<b>15 NVM Memory Map .....</b>	<b>241</b>
15.1 Introduction.....	241
15.2 Usage of NVM Memory Map .....	241
15.3 Content of NVM Memory Map .....	241
<b>16 Timer Functions.....</b>	<b>243</b>
16.1 Introduction.....	243
16.2 Flash Strobe.....	243
16.3 Special Actuator Strobe.....	247

<b>17 PDAF .....</b>	<b>249</b>
17.1 Introduction .....	249
17.2 Impact of PDAF Function on Other CCS Features .....	251
17.2.1 PDAF Pixel Readout Limited to Visible Pixel Channel .....	251
17.2.2 Interleaved or Bottom Embedded Data Readout .....	252
17.3 Bottom Embedded PDAF Readout .....	253
17.4 Interleaved PDAF Readout .....	253
17.4.1 Virtual Channel PDAF Interleaving .....	253
17.4.2 Data Type PDAF Interleaving .....	254
<b>18 Temperature Sensor .....</b>	<b>255</b>
<b>19 Camera Control Interface (CCI).....</b>	<b>259</b>
19.1 Independent and Simultaneous Camera Usage Support with I <sup>2</sup> C-Based CCI .....	259
19.1.1 Software-Switchable CCI Address .....	261
19.1.2 Alternate CCI Address .....	262
<b>20 CCI Register Map.....</b>	<b>263</b>
20.1 Introduction .....	263
20.2 Responsibilities .....	265
20.2.1 Unused Registers .....	265
20.2.2 RW Register Default Values .....	265
20.2.3 Accessing Registers .....	265
20.2.4 Manufacturer Specific Functionality .....	266
20.3 Multi-Byte Registers Index Space .....	267
20.3.1 Valid 16-bit Register Indices .....	267
20.3.2 Valid 32-bit Register Indices .....	267
20.4 Data Alignment Within CCI Registers .....	268
20.5 Valid Register Formats .....	268
20.6 Configuration Registers: 0x0000–0x0FFF .....	269
20.6.1 Status Registers: 0x0000–0x00FF (RO, RO Dynamic) .....	269
20.6.2 Set-Up Registers: 0x0100–0x01FF .....	276
20.6.3 Integration Time and Gain Registers: 0x0200–0x02FF (RW) .....	279
20.6.4 Video Timing Registers: 0x0300–0x03FF (RW) .....	281
20.6.5 Image Scaling Registers: 0x0400–0x04FF (RW and RO) .....	284
20.6.6 Image Compression Registers: 0x0500–0x0501 (RO) .....	285
20.6.7 Test Pattern Registers: 0x0600–0x060B (RW) .....	285
20.6.8 PHY Configuration Registers: 0x0800–0x0819 (RW) .....	286
20.6.9 CSI-2 Link Rate Registers: 0x0820–0x0823 (RW) .....	287
20.6.10 Equalization Control Registers: 0x0824–0x0825 (RW) .....	287
20.6.11 D-PHY Preamble Control Registers: 0x0826–0x0827 (RW) .....	287
20.6.12 D-PHY Spread Spectrum Control Registers: 0x0828 (RW) .....	287
20.6.13 Manual LP Control Register: 0x0829 (RW) .....	287
20.6.14 Additional PHY Configuration Registers: 0x082A–0x082F (RW) .....	288
20.6.15 PHY Calibration Configuration Registers: 0x0830–0x083F (RW) .....	289
20.6.16 C-PHY Manual Control Registers: 0x0840–0x0851 (RW) .....	290
20.6.17 CSI-2 v2 Feature Control Registers: 0x085A–0x08B1 (RW) .....	291
20.6.18 USL Control Registers: 0x08C0–0x08DF (RW) .....	294
20.6.19 Binning Configuration Registers: 0x0900–0x0902 (RW) .....	295
20.6.20 Data Transfer Interface Configuration Registers: 0xA00–0xA43 (RW, RO Dynamic) .....	295

20.6.21	Image Processing and Sensor Correction Configuration Registers: 0x0B00–0x0BA0 (RW) .....	296
20.6.22	Timer Configuration Registers: 0x0C00–0x0cff (RW, RO Dynamic) .....	298
20.6.23	PDAF Control Registers: 0x0D00–0x0dff (RW) .....	299
20.6.24	Bracketing Interface Configuration Registers: 0x0E00–0x0eff (RW) .....	300
20.7	Parameter Limit Registers: 0x1000–0x1FFF (Static RO).....	301
20.7.1	Integration Time and Gain Parameter Limit Registers: 0x1000–0x10DF (RO) ...	301
20.7.2	Generic Parameter Limit Registers: 0x10E0–0x10EF (RO).....	302
20.7.3	ADC Parameter Limit Registers: 0x10F0–0x10FF (RO) .....	303
20.7.4	Video Timing Parameter Limit Registers: 0x1100–0x11FF (RO) .....	303
20.7.5	Image Scaling Parameter Limit Registers: 0x1200–0x120F (RO) .....	314
20.7.6	HDR Limit Registers: 0x1210–0x121F (RO).....	315
20.7.7	USL Capability Registers: 0x1230–0x126F (RO).....	315
20.7.8	Image Compression Capability Registers: 0x1300–0x1301 (RO).....	317
20.7.9	Test Mode Capability Registers: 0x1310–0x1318 (RO).....	317
20.7.10	FIFO Capability Registers: 0x1500–0x1502 (RO) .....	317
20.7.11	CSI-2 Capability Registers: 0x1600–0x16FF (RO).....	318
20.7.12	Binning Capability Registers: 0x1700–0x17FF (RO).....	324
20.7.13	Data Transfer Interface Capability Registers: 0x1800–18FF (RO) .....	325
20.7.14	Image Processing and Sensor Correction Capability Registers: 0x1900–0x19FF (RO).....	326
20.7.15	Timer Capability Registers: 0x1A00–0x1A03 (RO) .....	327
20.7.16	Soft Reset Capability Registers: 0x1A10–0x1A11 (RO).....	327
20.7.17	PDAF Capability Registers: 0x1B80–0x1B8F (RO) .....	327
20.7.18	Bracketing Interface Capability Registers: 0x1C00–0x1cff (RO) .....	327
20.8	Image Statistics Registers: 0x2000–0x2FFF.....	328
20.9	Manufacturer Specific Registers: 0x3000–0xFFFF .....	328
<b>Annex A</b>	<b>Additional Examples .....</b>	<b>329</b>
A.1	Data Format.....	329
A.1.1	Embedded Data Packing for RAW6, RAW7, RAW12, and RAW14.....	329
A.1.2	Embedded Data Packing for RAW16, RAW20, and RAW24.....	331
A.1.3	Embedded Data Example Using Simplified 2-Byte Tagged Data Format.....	332
A.2	Retiming Examples .....	349
A.2.1	Generic Retiming Rules .....	349
A.2.2	Exposure Parameter Retiming .....	349
A.2.3	Flash Retiming .....	349
A.2.4	Exposure Parameter Retiming with Flash.....	351
A.2.5	Flash Synchronization Using Software Standby Mode .....	354
<b>Annex B</b>	<b>CCS Static Data .....</b>	<b>355</b>
B.1	Introduction.....	355
B.2	CCS Static Data Format.....	355
B.2.1	Introduction.....	355
B.2.2	Generic Definitions .....	356
B.2.3	Common Register BLOB.....	356
B.2.4	Block Header .....	361
B.2.5	First Block Header .....	363
B.2.6	Dummy Block.....	363
B.2.7	CCS Static Data Version Block.....	363
B.2.8	Read Only Register Block .....	364

B.2.9	MSR Register Block .....	365
B.2.10	Generic Rule Based Block .....	367
B.2.11	FFD Record.....	372
B.2.12	PDAF Pixel Location Block .....	384
B.2.13	PDAF Readout Record .....	393
B.2.14	Complete FFD Usage.....	399
B.2.15	License Block.....	403
B.2.16	End of Data Block.....	403
B.3	CCS Static Data Sources.....	404
B.3.2	Sensor, Module, and Data Source Specific Information.....	405
B.3.3	CCS Static Data Block Priorities .....	405
B.3.4	Prioritization Granularity .....	405
B.3.5	Override Policy .....	405
B.3.6	Sensor and Module Identification.....	405
B.3.7	Terms.....	405
B.3.8	Example 1 .....	406
B.3.9	Example 2 .....	406
B.4	File Delivery .....	407
B.4.1	Licensing.....	407
B.4.2	License Block Validation .....	407
B.5	Additional Information .....	408
B.5.1	4-Byte Extended FFD .....	408
B.5.2	Checksum Calculation .....	411
<b>Annex C</b>	<b>Non-Bayer Support .....</b>	<b>413</b>
<b>Annex D</b>	<b>USL Support .....</b>	<b>415</b>
D.1.1	Power-Up and Initialization of USL Image Sensor.....	415
D.2	Clock Tree of USL Image Sensor .....	418
D.3	USL Image Sensor Receiver LRTE Capability and Control Registers .....	423

# Figures

Figure 1 System Overview Example .....	11
Figure 2 System Example.....	11
Figure 3 System State Diagram.....	16
Figure 4 Power Up Sequence with D-PHY, All Lanes In Use.....	21
Figure 5 Power Up Sequence with D-PHY, 2-Lane System with Only One-Lane In Use .....	22
Figure 6 Power Down Sequence with D-PHY .....	25
Figure 7 Mode Change with D-PHY .....	27
Figure 8 Mode Change, 2-Lane D-PHY to 1-Lane D-PHY .....	28
Figure 9 Mode Change, 1-Lane D-PHY to 2-Lane D-PHY .....	29
Figure 10 XSHUTDOWN Timing Specification .....	33
Figure 11 Example of Valid CSI-2 Frame .....	40
Figure 12 CSI Data Format Register .....	46
Figure 13 C-PHY Timing Details (Excerpt from C-PHY Specification) .....	76
Figure 14 Data Format Description – Simple Example .....	84
Figure 15 Data Format Description – More Complex Example.....	84
Figure 16 CSI-2 Frame Format .....	90
Figure 17 2-Byte Generic Frame Format Descriptor.....	94
Figure 18 Frame Format Subtype Register – Number of Column and Row Descriptors Used .....	95
Figure 19 Generic Frame Format Description - SVGA Example.....	97
Figure 20 4-Byte Frame Format Descriptor Format.....	98
Figure 21 Embedded Data Line Format .....	100
Figure 22 Embedded Data in CSI-2 .....	102
Figure 23 Tagged Data Format (Detail).....	103
Figure 24 RAW8 Embedded Data Packing .....	106
Figure 25 RAW10 Embedded Data Packing .....	106
Figure 26 Video Timing Overview .....	107
Figure 27 Programmable Image Size .....	108
Figure 28 Image Frame with Additional Border Pixels.....	110
Figure 29 Output Sizes Larger Than Available Image Data .....	115
Figure 30 Output Image Size Example .....	116
Figure 31 Standard Readout .....	117

Figure 32 Horizontally Mirrored Readout.....	117
Figure 33 Vertically Flipped Readout.....	118
Figure 34 Horizontally Mirrored and Vertically Flipped Readout .....	118
Figure 35 Sub-Sampled Bayer Pixel Readout Examples .....	120
Figure 36 Bayer Pixel Sub-Sampling Behavior with Hmirror .....	125
Figure 37 Monochrome Pixel Sub-Sampling Examples with Hmirror and Vflip .....	126
Figure 38 Model for Scaling Features (for binning_capability = 1).....	127
Figure 39 Example of Binning .....	131
Figure 40 Bayer Pixel Binning Examples for binning_capability = 2.....	132
Figure 41 Monochrome Pixel Binning with binning_capability = 2 .....	133
Figure 42 Frame Length/Line Length Definitions .....	135
Figure 43 Simple One-PLL System.....	140
Figure 44 One PLL with System Speed Model .....	141
Figure 45 One PLL with Lane Speed Model.....	141
Figure 46 Two PLL with System Speed Model.....	142
Figure 47 Two PLL with Lane Speed Model .....	142
Figure 48 Integration Time.....	174
Figure 49 Fine Integration Time Range.....	185
Figure 50 Coarse Integration Time Range.....	185
Figure 51 Manual Readout Start Mode .....	188
Figure 52 Single Long Exposure Frame.....	188
Figure 53 Multiple Long Exposure Frames.....	189
Figure 54 Single Long Exposure Frame Followed by Normal Frames.....	189
Figure 55 SA_strobe During Manual Readout Start Mode .....	190
Figure 56 GPIO_TRIG Used to Start Exposure .....	191
Figure 57 Loop Mode Update Behavior.....	197
Figure 58 100% Color Bar Pattern .....	207
Figure 59 Fade to Grey Color Bar Pattern.....	209
Figure 60 360x296 Fade to Grey Color Bar Pattern.....	214
Figure 61 360x296 Fade to Grey Color Bar Pattern with Dimensions.....	214
Figure 62 100% Color Tile Pattern.....	215
Figure 63 Example of 100% Color Tile Pattern .....	216
Figure 64 PN9 Linear Feedback Shift Registers .....	217

Figure 65 Data Flow in Scaling.....	221
Figure 66 Control for Image Area After Analog and Digital Crop.....	222
Figure 67 Example of Data Type Interleaved OB Pixel Readout.....	240
Figure 68 Example of Virtual Channel Interleaved OB Pixel Readout .....	240
Figure 69 Example of Short Flash Strobe Pulse in Rolling Shutter Use Case .....	246
Figure 70 Example of Long Flash Strobe Pulse in Rolling Shutter Use Case .....	246
Figure 71 PDAF Data as Bottom Embedded Data .....	253
Figure 72 PDAF Data in Different Virtual Channel .....	254
Figure 73 PDAF Data with Different Data Type.....	254
Figure 74 Camera Sensor 7-Bit Target Address and R/~W Control Bit.....	259
Figure 75 Valid 16-Bit Indices for the MS Data Byte of 16-bit-Wide Register .....	267
Figure 76 Valid 16-Bit Indices for the MS and LS Data Bytes of 32-bit-Wide Register .....	267
Figure 77 Right Alignment for Packing 10-bit Data into Two 8-bit Registers.....	268
Figure 78 2-Byte Tagged Data Packing for RAW6, 7, 8, 10, 12, and 14 .....	329
Figure 79 RAW6 Embedded Data Packing .....	330
Figure 80 RAW7 Embedded Data Packing .....	330
Figure 81 RAW12 Embedded Data Packing .....	330
Figure 82 RAW14 Embedded Data Packing .....	331
Figure 83 2-Byte Tagged Data Packing for RAW16, 20 and 24 .....	331
Figure 84 Embedded Data Example.....	332
Figure 85 Flash Strobe with Long Pulse .....	350
Figure 86 Flash Strobe with Short Pulse .....	350
Figure 87 Flash Example with Flash Trigger .....	351
Figure 88 Flash Example with GPH.....	353
Figure 89 Flash Example in Streaming Start.....	354
Figure 90 CCS Static Data Format.....	355
Figure 91 Example of Simple FFD Usage.....	373
Figure 92 Example 2 of Simple FFD Usage.....	375
Figure 93 Example of Advanced FFD Usage.....	377
Figure 94 Example 2 of Advanced FFD Usage.....	379
Figure 95 Top Left OB Pixel Example 1 .....	381
Figure 96 OB Pixel Example with Dummy Pixels.....	382
Figure 97 OB Pixel Example with Native Readout Order .....	382

Figure 98 OB Pixel Example with Different Readout Order .....	383
Figure 99 Example of Simple PDAF Block Pattern.....	388
Figure 100 Example of Simple PDAF Arrangement in Block .....	389
Figure 101 Example of Line PDAF Block Pattern.....	391
Figure 102 Example of Line PDAF Arrangement in Block .....	391
Figure 103 Simple PDAF Readout Example.....	395
Figure 104 Advanced PDAF Readout Example .....	397
Figure 105 4-Byte Extended Frame Format Descriptor .....	408
Figure 106 Example of USL Clock Tree .....	418

## Tables

Table 1 Operating Mode Summary .....	15
Table 2 Power Up Sequence Timing Constraints .....	20
Table 3 Power Down Timing Constraints.....	24
Table 4 Mode Change Sequence Timing Constraints.....	26
Table 5 Fast Standby Capability Register.....	30
Table 6 Fast Standby Control Register.....	30
Table 7 Mode Select Register.....	31
Table 8 Soft Reset Register .....	31
Table 9 Soft Reset Specifications .....	32
Table 10 Soft Reset Limit Registers.....	32
Table 11 Frame Counter Register.....	33
Table 12 XSHUTDOWN Specifications.....	33
Table 13 Extclk Frequency Register.....	34
Table 14 GPIO_TRIG Mode Control Registers .....	34
Table 15 GPIO_TRIG Specifications.....	34
Table 16 Camera Module Identifier Registers.....	36
Table 17 Image Sensor Identifier Registers.....	37
Table 18 CSI Signaling Mode Register .....	43
Table 19 CSI Lane Mode Register .....	43
Table 20 CSI Signaling Mode Capability Register .....	44
Table 21 CSI D-PHY Lane Mode Capability Register.....	44
Table 22 CSI C-PHY Lane Mode Capability Register.....	45
Table 23 CSI Data Format Register.....	46
Table 24 CSI Compression DT Register .....	47
Table 25 CSI Channel Identifier Register.....	47
Table 26 Bottom Embedded Data Control Registers.....	47
Table 27 Data Type Capability Register.....	48
Table 28 ADC Control and Capability Registers.....	48
Table 29 LRTE Registers.....	50
Table 30 ALPS Registers.....	52
Table 31 Data Scrambling Registers .....	53

Table 32 PHY Calibration Capability Registers.....	59
Table 33 PHY Calibration Registers.....	60
Table 34 Additional C-PHY Calibration Control Registers.....	61
Table 35 PHY Control Capability Register .....	63
Table 36 PHY Control Register.....	65
Table 37 CSI-2 Requested Link Rate .....	67
Table 38 D-PHY Manual Constant Parameters .....	68
Table 39 Parameters Containing Time and UI Values Register 1 .....	69
Table 40 Parameters Containing Time and UI Values Register 2.....	70
Table 41 Parameters Containing Only Time Values Register.....	71
Table 42 Timing Registers with Scale Factors .....	72
Table 43 D-PHY Timing Register Limits.....	73
Table 44 C-PHY Timing Parameters from MIPI C-PHY Specification .....	75
Table 45 C-PHY Manual Constant Parameters .....	77
Table 46 Additional C-PHY Timing Parameters .....	78
Table 47 C-PHY Timing Register Limits .....	79
Table 48 PHY Capability Register .....	80
Table 49 PHY Equalization Register.....	80
Table 50 PHY Capability Register .....	81
Table 51 D-PHY Preamble Registers .....	81
Table 52 PHY Capability Register .....	81
Table 53 D-PHY SSC Register.....	81
Table 54 Manual LP Control Register.....	82
Table 55 Data Format Description Registers.....	83
Table 56 Extended Data Format Description Registers.....	85
Table 57 Data Pedestal Register .....	87
Table 58 Typical Data Pedestal Value.....	87
Table 59 Frame Format Description Codes.....	93
Table 60 Pixel Code Definitions.....	94
Table 61 2-Byte Generic Frame Format Description Registers.....	96
Table 62 4-Byte Generic Frame Format Description Registers.....	99
Table 63 Embedded Data Format Codes .....	100
Table 64 Embedded Data Capability and Control Registers .....	101

Table 65 Tagged Data Format Tag Code Summary .....	104
Table 66 Example of Addressable Pixel Array Size .....	109
Table 67 Pixel Array Address Registers .....	111
Table 68 Pixel Array Address Capability Registers.....	112
Table 69 Output Image Size Registers .....	115
Table 70 Output Size Limits.....	116
Table 71 Image Orientation Register.....	118
Table 72 Contents of Image Orientation Register .....	118
Table 73 Pixel Order Register (Read Only Dynamic).....	119
Table 74 Color Pixel Order Code .....	119
Table 75 X and Y-Address Increment Registers.....	120
Table 76 X and Y-Address Increment Parameter Limits and Sub-sampling Capability for Bayer Pixel Readout .....	121
Table 77 Pixel Readout Capability Register.....	122
Table 78 Monochrome Readout Capability, Sub-sampling Limits, and Control Registers.....	123
Table 79 Binning Capability Register .....	128
Table 80 X and Y-Address Increment Parameter Limits for binning_capability = 2 .....	129
Table 81 Binning Mode Register.....	130
Table 82 Binning Control Registers .....	130
Table 83 Video Timing Registers .....	134
Table 84 Frame Timing Parameter Limits .....	136
Table 85 Frame Blanking Register .....	136
Table 86 Frame Timing Parameter Limits for Binning mode.....	137
Table 87 PLL Control and Capability Registers.....	139
Table 88 Clocking Example (Assuming op_bits_per_lane = 0).....	151
Table 89 Clocking Example (Assuming op_bit_per_lane = 0) .....	157
Table 90 Clock Division and PLL Multiplier Registers for Simple One-PLL System.....	160
Table 91 Clock Division and PLL Multiplier Registers for One-PLL System.....	160
Table 92 Clock Division and PLL Multiplier Registers for Two-PLL System.....	161
Table 93 Video Timing Pixel Clock Divider Register .....	162
Table 94 Video Timing System Clock Divider Register .....	162
Table 95 Pre-PLL Clock Divider Registers .....	163
Table 96 PLL Multiplier Registers .....	164

Table 97 Output Pixel Clock Divider Register.....	164
Table 98 Output System Clock Divider Register .....	164
Table 99 Clock Capability Type Register.....	165
Table 100 Pre PLL and PLL Clock Set-Up Capability Registers.....	165
Table 101 Video Timing Clock Set-Up Capability Registers .....	167
Table 102 Output Clock Set-Up Capability Registers.....	168
Table 103 D-PHY Per-Lane Bitrate Limit Registers .....	169
Table 104 C-PHY Per-Lane Symbol Rate Limit Registers .....	170
Table 105 Integration Time Control Parameters.....	173
Table 106 Analog Gain Capability Parameters.....	175
Table 107 Gain Mode Register.....	175
Table 108 Analog Gain Control Parameters .....	176
Table 109 Analog Gain Coding/Decoding Constants .....	177
Table 110 Gain Parameter Limits .....	177
Table 111 Alternate Analog Gain Control Parameters.....	178
Table 112 Alternate Gain Parameter Limits .....	178
Table 113 Digital Gain Control Parameters.....	180
Table 114 Digital Gain Capability .....	180
Table 115 Digital Gain Limit & Precision Parameters .....	180
Table 116 Grouped Change Control Parameter .....	181
Table 117 Mask Corrupted Frames Register .....	182
Table 118 Gain Delay Type Register.....	183
Table 119 Integration Time Parameter Limits .....	184
Table 120 Timing Mode Capability Registers .....	186
Table 121 Timing Mode Control Registers .....	186
Table 122 Automatic Frame Length Control Registers .....	187
Table 123 Manual Readout Control Registers.....	189
Table 124 Bracketing LUT Capability Registers.....	192
Table 125 Example 1 LUT Content.....	193
Table 126 Example 2 LUT Content.....	194
Table 127 Possible LUT Entry Types.....	194
Table 128 Bracketing Configuration Registers .....	195
Table 129 HDR_internal_bit_depth Value as a Function of Exposure Ratio .....	198

Table 130 HDR Capability Registers .....	201
Table 131 HDR Control Registers.....	202
Table 132 Main Full Frame Test Pattern Control Parameter .....	205
Table 133 Test Mode Capability.....	205
Table 134 Test Data Color Parameters .....	206
Table 135 100% Color Bar Values .....	207
Table 136 Test Pattern Capability Registers.....	208
Table 137 Example of Fade to Grey Values .....	212
Table 138 Fade to Grey Control Registers .....	212
Table 139 100% Color Tile Values .....	215
Table 140 PN9 Capability Registers.....	218
Table 141 Image Scaling Capability Register .....	219
Table 142 Digital Crop Capability Register .....	219
Table 143 Digital Crop Control Register.....	220
Table 144 Image Scaling Registers .....	223
Table 145 Image Scaling Capability Registers.....	223
Table 146 FIFO Control Registers.....	224
Table 147 Compression Mode Register.....	225
Table 148 Compression Capability Register .....	225
Table 149 Data Transfer Interface Capability Registers.....	227
Table 150 Data Transfer Interface 1 Registers .....	228
Table 151 Sensor Processing Capabilities .....	232
Table 152 Shading Correction Control.....	233
Table 153 Luminance Correction Control .....	233
Table 154 Green Imbalance Correction Control.....	233
Table 155 Mapped Defect Correction Control .....	234
Table 156 Dynamic Couplet Correction Control.....	234
Table 157 Single Pixel Correction Control.....	234
Table 158 Combined Defect Correction Control.....	235
Table 159 Dynamic Triplet Defect Correction Control .....	235
Table 160 Noise Filter Control .....	236
Table 161 Module Specific Correction Control .....	236
Table 162 Color Temperature Feedback Capability .....	237

Table 163 Color Temperature Feedback Locations .....	237
Table 164 Optical Black Readout Control.....	239
Table 165 Optical Black Readout Capability .....	239
Table 166 Flash Strobe Capability Register .....	243
Table 167 Flash Strobe Adjustment Register .....	243
Table 168 Achievable Flash Strobe Range.....	244
Table 169 Strobe Config Registers.....	244
Table 170 Flash Strobe Control Register.....	245
Table 171 Flash Status Register .....	245
Table 172 SA_strobe Capability Register .....	247
Table 173 Special Actuator Strobe Config Register.....	248
Table 174 Special Actuator Strobe Status Register .....	248
Table 175 PDAF Registers .....	250
Table 176 Temperature Sensor Registers .....	256
Table 177 Temperature Sensor Output Format.....	257
Table 178 CCI Address Control .....	260
Table 179 CCI Register Groupings .....	263
Table 180 Valid Register Formats .....	268
Table 181 General Status Registers .....	269
Table 182 Frame Format Description Registers.....	271
Table 183 Analog Gain Description Registers .....	273
Table 184 Data Format Description Registers.....	275
Table 185 General Set-Up Registers .....	276
Table 186 Output Set-Up Registers.....	277
Table 187 Gain Set-Up Registers .....	278
Table 188 ADC Set-up Registers.....	278
Table 189 Generic Control Registers.....	278
Table 190 GPIO Set-Up Registers.....	278
Table 191 Reference Clock Frequency Registers.....	278
Table 192 Temperature Sensor Registers .....	279
Table 193 Integration Time Registers.....	279
Table 194 Analog Gain Registers .....	279
Table 195 Digital Gain Registers .....	279

Table 196 HDR Control Registers.....	280
Table 197 Clock Set-Up Registers.....	281
Table 198 Frame Timing Registers.....	282
Table 199 Image Size Registers.....	282
Table 200 Timing Mode Registers.....	283
Table 201 Sub-Sampling Registers.....	283
Table 202 Monochrome Readout Registers.....	283
Table 203 Image Scaling Registers .....	284
Table 204 Image Compression Registers .....	285
Table 205 Test Pattern Registers.....	285
Table 206 PHY Configuration Registers .....	286
Table 207 CSI-2 Requested Link Rate Registers .....	287
Table 208 Equalization Control Registers .....	287
Table 209 D-PHY Preamble Control Registers .....	287
Table 210 D-PHY Spread Spectrum Control Register .....	287
Table 211 Manual LP Control Register .....	287
Table 212 Additional PHY Configuration Registers .....	288
Table 213 D-PHY 1.2 Related Configuration Registers.....	289
Table 214 C-PHY Manual Control Registers .....	290
Table 215 ALPS Feature Control Registers.....	291
Table 216 LRTE Feature Control Registers .....	291
Table 217 Data Scrambling Feature Control Registers .....	292
Table 218 USL Control Registers.....	294
Table 219 Binning Configuration Registers .....	295
Table 220 Data Transfer Interface Configuration Registers .....	295
Table 221 Sensor Correction Configuration Registers .....	296
Table 222 Optical Black Pixel Readout Registers.....	297
Table 223 Color Temperature Feedback Registers .....	297
Table 224 CFA Conversion Registers.....	297
Table 225 Flash and SA Strobe Configuration .....	298
Table 226 PDAF Control Registers .....	299
Table 227 Bracketing Interface Configuration Registers .....	300
Table 228 Integration Time Capability Registers .....	301

Table 229 Digital Gain Capability Registers .....	302
Table 230 Data Pedestal Capability Registers .....	302
Table 231 ADC Capability Registers.....	303
Table 232 Pre-PLL and PLL Clock Set-up Capability Registers.....	303
Table 233 Video Timing Clock Set-up Capability Registers.....	307
Table 234 Clock Calculation Capability Registers.....	308
Table 235 Frame Timing Capability Registers .....	309
Table 236 Timing Capability Registers .....	310
Table 237 Output Clock Set-Up Capability Registers.....	311
Table 238 Image Size Capability Registers.....	312
Table 239 Sub-Sampling Capability Registers.....	313
Table 240 Image Scaling Capability Registers.....	314
Table 241 HDR Capability Registers .....	315
Table 242 USL Capability Registers .....	315
Table 243 Image Compression Capability Registers.....	317
Table 244 Test Mode Capability Registers.....	317
Table 245 FIFO Capability Registers .....	317
Table 246 CSI-2 Capability Registers .....	318
Table 247 Binning Capability Registers.....	324
Table 248 Data Transfer Interface Capability Registers.....	325
Table 249 Sensor Correction Capability Registers.....	326
Table 250 Optical Black Readout Capability Registers .....	326
Table 251 Color Feedback Capability Registers .....	326
Table 252 CFA Pattern Capability Registers .....	327
Table 253 Timer Capability Registers .....	327
Table 254 Timer Capability Registers .....	327
Table 255 PDAF Capability Registers.....	327
Table 256 Bracketing Interface Capability Registers .....	327
Table 257 Simple Example of Embedded Data Content .....	333
Table 258 Complex Example of Embedded Data.....	340
Table 259 Length Specifier for 0 – 2 <sup>6</sup> -1 bytes.....	356
Table 260 Length Specifier for 2 <sup>6</sup> – 2 <sup>14</sup> -1 bytes.....	356
Table 261 Length Specifier for 2 <sup>14</sup> – 2 <sup>22</sup> -1 bytes .....	356

Table 262 Description of BLOB Fields.....	357
Table 263 BLOB with Format Selection Field Value 0 .....	358
Table 264 BLOB with Format Selection Field Value 1 .....	359
Table 265 BLOB with Format Selection Field Value 2 .....	360
Table 266 Generic Block Header.....	361
Table 267 Description of Block fields.....	361
Table 268 Block ID usage .....	361
Table 269 Block ID Allocations .....	362
Table 270 Block Header of First Block .....	363
Table 271 File Version Block .....	363
Table 272 Example of RO Register Block .....	365
Table 273 Description of a Record in the Generic Rule Based Block.....	367
Table 274 Generic Rule Based Block Record Types.....	367
Table 275 If Rule .....	368
Table 276 Example of Data in Type 2 .....	370
Table 277 Example of Generic Rule Based Block .....	371
Table 278 Example of Simple FFD Usage .....	374
Table 279 Example 2 of Simple FFD Usage .....	376
Table 280 Example of Advanced FFD Usage .....	378
Table 281 Example 2 of Advanced FFD Usage .....	380
Table 282 Structure of PDAF Location Block .....	385
Table 283 Content of Block Descriptor Group.....	386
Table 284 Content of Block ID Properties .....	387
Table 285 Content of PDAF Location Block in Simple Example.....	390
Table 286 Content of PDAF Location Block in Line Example.....	392
Table 287 Content of PDAF Readout Record .....	393
Table 288 Simple PDAF Readout Block Example.....	396
Table 289 Advanced PDAF Readout Example.....	398
Table 290 Summary of the FFD and PDAF Readout Records .....	400
Table 291 Summary of the FFD Records .....	401
Table 292 Summary of FFD Records .....	402
Table 293 End of Data Block .....	403
Table 294 Naming Abbreviations in File Name .....	404

Table 295 Pixel Code Definitions.....	409
Table 296 Checksum Block Content .....	411
Table 297 CFA Pattern Capability Register.....	413
Table 298 CFA Conversion Registers.....	413
Table 299 USL ALP Registers.....	416
Table 300 USL Support Capability Register .....	418
Table 301 USL_REV Clock Divider Registers .....	419
Table 302 USL Clock Set-Up Capability Registers.....	420
Table 303 USL Switching Registers.....	421
Table 304 Additional USL Registers .....	422
Table 305 USL Image Sensor CSI-2 Receiver LRTE Capability Registers .....	423
Table 306 USL LRTE Control Registers .....	425

## Release History

Date	Version	Description
24-Oct-2017	v1.0	Initial Board Adopted release.
12-Dec-2019	v1.1	Board Adopted release.
17-Apr-2023	v1.1.1	Board Adopted release.

## 1 Introduction

This document, the MIPI Camera Command Set (CCS) Specification, defines functionality for camera modules and image sensors. The use of standardized functionalities removes the need for tailored, vendor-specific solutions for basic functionalities, making the integration of CCS compatible cameras into different systems easy and fast.

### 1.1 Scope

This Specification primarily defines control and data interfaces for image sensor functionality. Additional camera module level functionalities are defined (mainly identification information). Separate, related MIPI Specifications define the electrical interfaces.

This Specification defines the following:

- Operating Modes: How to power-up and power-down the camera module
- Identification
- Data format and data arrangement
- Video timing, cropping, and subsampling and binning modes
- Integration Time and Gain control
- Single frame and multi frame Exposure modes
- HDR (High Dynamic Range) mode
- PDAF
- Data transfer interface between camera and Host for calibration and other data
- Sensor corrections
- Timer functionalities
- Reporting of camera module capabilities and key performances
- Test modes
- CCI Register map
- CCS Static Data

Key related MIPI Specifications:

- MIPI CSI-2 Specification ([\[MIPI01\]](#), [\[MIPI06\]](#), [\[MIPI14\]](#), [\[MIPI12\]](#)), including D-PHY ([\[MIPI02\]](#), [\[MIPI03\]](#), [\[MIPI07\]](#), [\[MIPI08\]](#), [\[MIPI16\]](#)) and C-PHY ([\[MIPI04\]](#), [\[MIPI09\]](#), [\[MIPI10\]](#), [\[MIPI15\]](#))

This Specification (MIPI CCS) is compatible with the following MIPI Specifications:

- Camera Serial Interface 2 (CSI-2), Version 1.3 [\[MIPI01\]](#)
- Camera Serial Interface 2 (CSI-2), Version 2.0 [\[MIPI06\]](#)
- Camera Serial Interface 2 (CSI-2), Version 2.1 [\[MIPI14\]](#)
- Camera Serial Interface 2 (CSI-2), Version 3.0 [\[MIPI12\]](#)
- D-PHY, Version 1.1 [\[MIPI02\]](#)
- D-PHY, Version 1.2 [\[MIPI03\]](#)
- D-PHY, Version 2.0 [\[MIPI07\]](#)
- D-PHY, Version 2.1 [\[MIPI16\]](#)
- D-PHY, Version 2.5 [\[MIPI08\]](#)
- C-PHY, Version 1.0 [\[MIPI04\]](#)
- C-PHY, Version 1.1 [\[MIPI09\]](#)
- C-PHY, Version 1.2 [\[MIPI15\]](#)
- C-PHY, Version 2.0 [\[MIPI10\]](#)

This page intentionally left blank.

## 1.2 Purpose

The objective of this Specification (MIPI CCS) is to standardize Bayer image sensors in such a way that a standard software driver can use them with minimal software changes. In order to achieve this level of abstraction, the standardized functionality includes interfaces for electrical, control, and image data. In addition to Bayer image sensors, this Specification can also be used with non-Bayer sensors, see *Annex C*.

This Specification does not require the use of any particular image sensor SW driver. However, image sensors that implement MIPI CCS should be easier to integrate into systems than ones that use vendor-specific functionality. This ease of integration comes from the use of standardized MIPI CCS features, functionality, and interfaces.

## 2 Terminology

### 2.1 Use of Special Terms

The MIPI Alliance has adopted Section 13.1 of the *IEEE Standards Style Manual*, which dictates use of the words “shall”, “should”, “may”, and “can” in the development of documentation, as follows:

The word *shall* is used to indicate mandatory requirements strictly to be followed in order to conform to the Specification and from which no deviation is permitted (*shall* equals *is required to*).

The use of the word *must* is deprecated and shall not be used when stating mandatory requirements; *must* is used only to describe unavoidable situations.

The use of the word *will* is deprecated and shall not be used when stating mandatory requirements; *will* is only used in statements of fact.

The word *should* is used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain course of action is deprecated but not prohibited (*should* equals *is recommended that*).

The word *may* is used to indicate a course of action permissible within the limits of the Specification (*may* equals *is permitted to*).

The word *can* is used for statements of possibility and capability, whether material, physical, or causal (*can* equals *is able to*).

All sections are normative, unless they are explicitly indicated to be informative.

### 2.2 Definitions

**3A:** Control algorithms for camera module control, typically consisting of automatic Exposure control, automatic focus control, and automatic White Balance. In certain cases the term 3A is still used, even though 2A might be more correct, e.g. if a fixed focus camera module is used and no auto-focus algorithm is used.

**Bayer:** A Bayer pattern for visible pixels in image sensors. For a 2x2 pixel area, depending on readout order the pattern can be either RGrGbB, GrRBGb, BGbGrR, or GbBRGr, where Gr means GreenRed and Gb GreenBlue pixels (i.e. green pixel in an image line with red or blue pixels).

**Camera Control Interface (CCI):** A control interface between Host and image sensor, carrying at least control data, and conforming to the MIPI CSI-2 Specification (e.g. [\[MIPI12\]](#)).

**CCI Register:** A register accessible via a CCI bus.

**CCS Register:** A subset of CCI Registers as defined in this Specification. CCS Registers occupy register indexes 0x0000 through 0x2FFF. CCS v1.1 introduces CCS Static Data, which may replace certain CCS Registers (see [Annex B](#)).

**CCS Static Data:** Set of data which may replace CCS Parameter Limit Register(s), but can also include other capability information and MSRs. CCS Static Data may be located in file, or in camera module’s non-volatile memory, or both.

**CIL-PCNN:** Control and Interface Logic of the PHY in use; formerly known as CIL-MCNN. See the CSI-2 (e.g. [\[MIPI12\]](#)), D-PHY (e.g. [\[MIPI08\]](#)), and C-PHY (e.g. [\[MIPI10\]](#)) Specifications.

**CIL-PFEN:** Control and Interface Logic of the PHY in use; formerly known as CIL-MFEN. See the CSI-2 (e.g. [\[MIPI12\]](#)), D-PHY (e.g. [\[MIPI08\]](#)), and C-PHY (e.g. [\[MIPI10\]](#)) Specifications.

**C-PHY:** One of the serial physical interface options used to transfer image data (e.g. [\[MIPI10\]](#)).

**Controller:** An I<sup>2</sup>C or I3C Device that is capable of controlling an I<sup>2</sup>C or I3C Bus, respectively.

89 **Note:**

90     *The normative term “Controller” has replaced the deprecated term “Master” when the latter is used*  
91     *to describe a Device on an I<sup>2</sup>C or I3C Bus. Please note that the technical definition of such a Device,*  
92     *and its role on an I<sup>2</sup>C or I3C Bus, are unchanged.*

93     **D-PHY:** One of the serial physical interface options used to transfer image data (e.g. [\[MIPI08\]](#)).

94     **Escape** (in escape mode): Low power mode for Data Lanes to transfer commands as per the CSI-2  
95     Specification (e.g. [\[MIPI12\]](#)). See also the D-PHY (e.g. [\[MIPI08\]](#)) and C-PHY (e.g. [\[MIPI10\]](#))  
96     Specifications.

97     **EXIF:** Exchangeable Image File Format, used to carry additional data related to a still image [\[CIPA01\]](#).

98     **Exposure:** Event during which the image sensor gathers light in order to form an image. Exposure is  
99     controlled by Exposure time control and by Gain control. Exposure parameters are Exposure time and Gain  
100     control.

101     **EXTCLK:** Reference clock signal sent to the image sensor.

102     **Gain:** Analog or digital Gain used to amplify the signal recorded by the image sensor pixels.

103     **Generic Short Packet:** Generic Short Packet as defined in the CSI-2 Specification (e.g. [\[MIPI12\]](#)), carrying  
104     user-defined data.

105     **Global Analog Gain:** Gain performed in the analog domain and impacting all color channels in the same  
106     way. See also Gain.

107     **High Dynamic Range (HDR):** HDR modes in which multiple pixels are combined, either at the sensor level  
108     or at the system level, to produce a higher dynamic range than what a single pixel can provide.

109     **Host:** The image sensor is connected to a Host, and that Host controls the image sensor.

110     **Integration Time:** The time period during which the image sensor gathers light in order to form an image.  
111     (Basically the same meaning as the Exposure time.)

112     **Intelligent Status Line:** Top and bottom embedded data carrying register information, conveying image  
113     sensor state.

114     **HS-TX:** High-Speed Transmitter as per the CSI-2 Specification (e.g. [\[MIPI12\]](#)).

115     **Line Synchronization Packet:** Per-image-frame packets conforming to the CSI-2 Specification (e.g.  
116     [\[MIPI12\]](#)) and containing line start and line end information.

117     **LP-00:** Low power state when using D-PHY (e.g. [\[MIPI08\]](#)).

118     **LP-000:** Low power state when using C-PHY (e.g. [\[MIPI10\]](#)).

119     **LP-TX:** Low-Power Transmitter. See the CSI-2 (e.g. [\[MIPI12\]](#)), D-PHY (e.g. [\[MIPI08\]](#)), and C-PHY (e.g.  
120     [\[MIPI10\]](#)) Specifications.

121     **NVM:** Non-volatile memory, holding calibration data or other data.

122     **OTP:** One-Time Programmable memory, holding calibration data or other data.

123     **Phase Difference:** Phase difference between certain pixels, for use in phase-difference auto-focus and other  
124     functions.

125     **Primary:** The Primary is the PHY on the side of a Link that is the main data source. The Primary transmits  
126     data in the Forward Direction and receives data in the Reverse Direction. In some Bi-directional systems the  
127     functions of the Primary and Secondary are nearly equivalent. The difference between being Primary or  
128     Secondary is simply that one side is identified as the Primary, and the other side is identified as the Secondary.  
129     The side that is defined as the Primary does not change after the system is initialized.

130     **Note:**

131     *In previous versions of the CCS Specification, a Primary PHY was called a “Master PHY”. MIPI  
132     Alliance has deprecated use of the word “Master” as a technical term, and as a result this version of  
133     the CCS Specification now uses the updated normative term “Primary.” Please note that the technical*

134           definition of such a PHY, and its role on a Link, are unchanged. The adjective “primary” when used  
135           in describing a CCI interface in Section 19 is distinct from the defined term “Primary” used to describe  
136           a PHY.

137           **RAW8:** CSI-2 (e.g. [\[MIPI12\]](#)) RAW image format with 8 bits per color channel per pixel.

138           **RAW10:** CSI-2 (e.g. [\[MIPI12\]](#)) RAW format with 10 bits per color channel per pixel.

139           **Re-Timing:** A technique in which the image sensor delays putting the values of changed register values into  
140           effect, rather than doing so immediately when the Host performs a register value write operation over the  
141           CCI. Re-Timing is typically done in order to coordinate the internal register update with other internal timings  
142           within the image sensor.

143           **ROI:** Region of Interest. When cropping an image, only a subset of the full image frame is used. This sub-  
144           frame is referred to the ROI.

145           **SDA:** Data line of the CCI interface, as per the CSI-2 Specification (e.g. [\[MIPI12\]](#)).

146           **SCL:** Clock line of the CCI interface, as per the CSI-2 Specification (e.g. [\[MIPI12\]](#)).

147           **Secondary:** The Secondary is the PHY on the side of a Link that is the main data sink. The Secondary  
148           receives data in the Forward Direction and transmits data in the Reverse Direction. In some Bi-directional  
149           systems the functions of the Primary and Secondary are nearly equivalent. The difference between being  
150           Primary or Secondary is simply that one side is identified as the Primary, and the other side is identified as  
151           the Secondary. The side that is defined as the Secondary does not change after the system is initialized.

152           **Note:**

153           *In previous versions of the CSI-2 Specification, a Secondary PHY was called a “Slave PHY”. MIPI  
154           Alliance has deprecated use of the word “Slave” as a technical term, and as a result this version of  
155           the CSI-2 Specification now uses the updated normative term “Secondary.” Please note that the  
156           technical definition of such a PHY, and its role on a Link, are unchanged.*

157           **Soft-Reset:** Reset of the image sensor via CCI command, as contrasted with a purely electrical reset.

158           **Symbol:** In C-PHY (e.g. [\[MIPI10\]](#)), data is transferred in the form of encoded ternary Symbols. The CSI-2  
159           Specification (e.g. [\[MIPI12\]](#)) also addresses C-PHY Symbols.

160           **Sync Word:** In C-PHY (e.g. [\[MIPI10\]](#)), the Sync Word allows Tx and Rx to synchronize state for a  
161           transmission. The CSI-2 Specification (e.g. [\[MIPI12\]](#)) also addresses C-PHY Sync Words.

162           **Synchronization:** Synchronization of data (such as state or settings) between the Host and the image sensor.  
163           For example, the Exposure setting in the image sensor can be synchronized with the Host’s Exposure control  
164           algorithm.

165           **Target:** A Device on an I<sup>2</sup>C or I3C Bus that can only respond to commands from a Controller.

166           **Note:**

167           *In previous versions of the CCS Specification, a Target Device was called a “Slave Device”. MIPI  
168           Alliance has deprecated use of the word “Slave” as a technical term, and as a result this version of  
169           the CCS Specification now uses the updated normative term “Target.” Please note that the technical  
170           definition of such a Device, and its role on an I<sup>2</sup>C or I3C Bus, are unchanged.*

171           **Tuning:** Optimization of image signal processing parameters in order to improve image quality.

172           **ULPS:** Ultra Lower Power State as per the D-PHY (e.g. [\[MIPI08\]](#)) or C-PHY Specification (e.g. [\[MIPI10\]](#)).  
173           The CSI-2 Specification (e.g. [\[MIPI12\]](#)) also addresses ULPS.

174           **VBAT:** Battery voltage, i.e. the first voltage made available to the camera module or image sensor.

175           **White Balance:** Image processing in which relative intensities of the color channels are adjusted in order to  
176           maximize color neutrality of the overall image.

177           **XSHUTDOWN:** Input signal to the image sensor, which in the active low state both keeps the image sensor  
178           in the low power state, and serves as a hardware reset signal.

## 2.3 Abbreviations

- 179 e.g. For example (Latin: exempli gratia)  
180 i.e. That is (Latin: id est)

## 2.4 Acronyms

- 181 AEC Automatic Exposure Control  
182 ADC Analog to digital converter  
183 AF Auto Focus  
184 AWB Automatic White Balance  
185 BLOB Binary Large Object  
186 CCI Camera Control Interface  
187 CS Commercial sample  
188 EOT End of Transmission  
189 ES Engineering sample  
190 FE Frame End  
191 FS Frame Start  
192 HDR High Dynamic Range  
193 HW Hardware  
194 ISP Image Signal Processing  
195 LRTE Latency Reduction Transport Efficiency  
196 LS Least Significant  
197 LSB Least Significant Bit  
198 MP Mass production  
199 MS Most Significant  
200 MSB Most Significant Bit  
201 MSR Manufacturer Specific Registers  
202 PDAF Phase Difference Auto Focus  
203 PHY Physical Interface i.e. D-PHY or C-PHY  
204 SOT Start of Transmission  
205 SW Software  
206 TS Technical sample  
207 UI Unit Interval  
208 ULPS Ultra Low Power State  
209 VANA Analog voltage  
210 VIO I/O Voltage

211 VCore Digital voltage

### 3 References

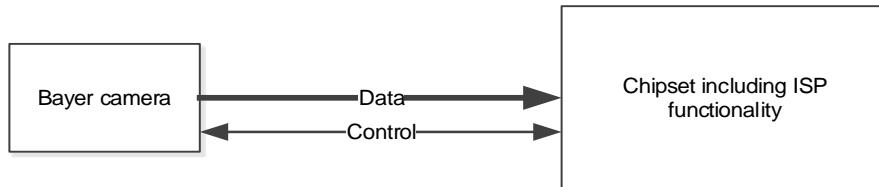
- 212 [MIP101] *MIPI Alliance Specification for Camera Serial Interface 2 (CSI-2), Version 1.3,*  
213 MIPI Alliance, Inc., 29 May 2014.
- 214 [MIP102] *MIPI Alliance Specification for D-PHY, Version 1.1,*  
215 MIPI Alliance, Inc., 7 November 2011.
- 216 [MIP103] *MIPI Alliance Specification for D-PHY, Version 1.2, MIPI Alliance, Inc., 1 August 2014.*
- 217 [MIP104] *MIPI Alliance Specification for C-PHY, Version 1.0, MIPI Alliance, Inc., 5 August 2014.*
- 218 [MIP105] *MIPI Alliance Manufacturer ID page, <http://mid.mipi.org/>,*  
219 MIPI Alliance, Inc., last accessed 17 April 2023.
- 220 [MIP106] *MIPI Alliance Specification for Camera Serial Interface 2 (CSI-2), Version 2.0,*  
221 MIPI Alliance, Inc., 28 March 2017.
- 222 [MIP107] *MIPI Alliance Specification for D-PHY, Version 2.0,*  
223 MIPI Alliance, Inc., 23 November 2015.
- 224 [MIP108] *MIPI Alliance Specification for D-PHY, Version 2.5,*  
225 MIPI Alliance, Inc., 17 October 2019.
- 226 [MIP109] *MIPI Alliance Specification for C-PHY, Version 1.1, MIPI Alliance, Inc., 7 October 2015.*
- 227 [MIP110] *MIPI Alliance Specification for C-PHY, Version 2.0, MIPI Alliance, Inc., 9 October 2019.*
- 228 [MIP111] *MIPI Alliance Specification for Improved Inter Integrated Circuit (I3C), Version 1.0,*  
229 MIPI Alliance, Inc., 31 December 2016.
- 230 [MIP112] *MIPI Alliance Specification for Camera Serial Interface 2 (CSI-2), Version 3.0,*  
231 MIPI Alliance, Inc., 31 May 2019.
- 232 [MIP113] *MIPI Alliance Camera Command Set page,*  
233 <https://mipi.org/specifications/camera-command-set>,  
234 MIPI Alliance, Inc., last accessed 17 April 2023.
- 235 [MIP114] *MIPI Alliance Specification for Camera Serial Interface 2 (CSI-2), Version 2.1,*  
236 MIPI Alliance, Inc., 9 April 2018.
- 237 [MIP115] *MIPI Alliance Specification for C-PHY, Version 1.2, MIPI Alliance, Inc., 28 March 2017.*
- 238 [MIP116] *MIPI Alliance Specification for D-PHY, Version 2.1, MIPI Alliance, Inc., 28 March 2017.*
- 239 [CIPA01] *Exchangeable image file format for digital still cameras: Exif Version 2.3,*  
240 [http://www.cipa.jp/std/documents/e/DC-008-2012\\_E.pdf](http://www.cipa.jp/std/documents/e/DC-008-2012_E.pdf),  
241 Camera & Imaging Products Association, Revised December 2012.
- 242 [ITUT01] ITU-T Recommendation V.42 (03/02), *Error-correcting procedures for DCEs using*  
243 *asynchronous-to-synchronous conversion, <https://www.itu.int/rec/T-REC-V/en>,*  
244 International Telecommunication Union, March 2002.

This page intentionally left blank.

## 4 Technical Overview

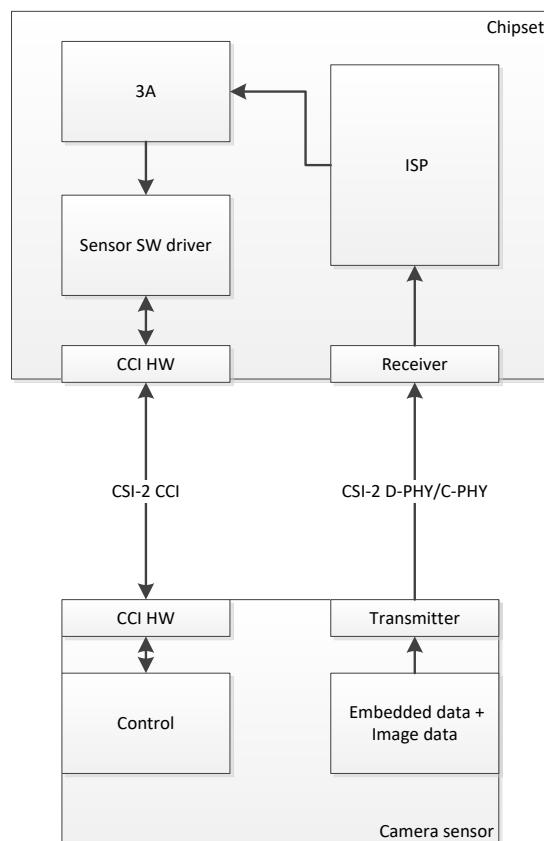
### 4.1 System View

A typical system using CCS consists of a camera module with a Bayer image sensor, and a chipset including ISP functionality. ISP functionality may be implemented as part of a main ASIC, using a separate IC, or as a combination of those. This Specification does not mandate any particular method of partitioning the ISP functionality among the ICs used in a given camera system.



**Figure 1 System Overview Example**

A complete camera system, as illustrated in **Figure 2**, will consist of a camera module including an image sensor, a chipset including ISP functionality, and a 3A and sensor SW driver. 3A includes Automatic White Balance (AWB), Automatic Exposure Control (AEC), and optionally Auto Focus (AF) algorithms. In addition to image sensors supporting I<sup>2</sup>C/I3C-based CCI, this Specification can also be used with USL image sensors (see **Annex D**).



**Figure 2 System Example**

In addition to a standard set of control registers described in this Specification, it is possible for sensors to have additional manufacturer specific registers (MSRs). The purpose of MSRs is to control additional features not described in this Specification, for example to fine-tune sensor internal operations.

MSRs should NOT be dependent upon:

- Reference clock frequency
- Pixel clock frequency
- CSI-2 speed
- Operating voltage level
- Any other standard CCS Registers.

Image sensor manufacturers shall note any deviation from these recommendations in the product datasheet, and the datasheet should show the formula for any CCS Registers upon which the image sensor depends.

## 4.2 Key Concepts

This section presents several key concepts that can be used together to build systems with self-configuration behavior.

### 4.2.1 Minimum Required Features

The camera module shall support the minimum set of required features needed to achieve a functional system. The camera module may additionally support optional features, such as image scaling, PDAF, and image compression.

The minimum set of required features creates a common baseline behavior across all camera modules, thus guaranteeing camera/Host interoperability.

All of the image sensor defaults shall be baseline compliant, i.e.:

- $\text{RAW}_n$  data format (in case of sensor with  $n$ -bit ADC)
- No image scaling
- No image compression
- Coarse Integration Time

### 4.2.2 Intelligent Status Line

Within a frame of CSI data, additional information about the set-up and configuration of the image sensor is embedded at the start and the end of the frame of data. This embedded data include the contents of many image sensor CCI Registers, thus describing the image sensor state. See [Section 7.13](#) for details.

The embedded data at the start of the frame is termed the Intelligent Status Line(s) and includes details such as Integration Time and Gain Parameters and Video Timing Parameters. A good analogy is the EXIF information (header) within a JPEG image file.

The Intelligent Status Line:

- Provides the Host with a fast method of receiving the contents of the image sensor's internal CCI Register values
- Lowers the overhead of CCI communications on the Host CPU by eliminating the need when the image sensor is streaming image data for the Host to read via the CCI bus
- Makes it possible to have test systems, which can open and process images even after image capture if captured raw image is stored with Intelligent Status Line and other information including for example image size and used Exposure information.

Use of the Intelligent Status Line should be a central feature of Host implementations.

#### 4.2.3 Synchronization

Synchronization of the Host and the image sensor is a vital part of any working camera system. For example, the Exposure used by the image sensor must be Synchronized with the Host's Exposure control algorithm.

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

The embedded data shall be correct for the image data contained in that frame. For example, the Integration Time and Gain parameter values reported shall be the values used for the image data output contained in that same frame.

The Host can use the values contained in the Intelligent Status Line to Synchronize the AEC and AWB control loops. For example, after the Host transmits a new set of Integration Time and Gain parameters it may wait for the new parameter values to appear in the Intelligent Status Line. This allows the Host to know the exact frame in which the updated Integration Time and Gains values first appear.

Synchronization is necessary for correct Exposure control behavior. If the system is not aware of the image sensor's correct Exposure/gain settings for an image frame, it could mistakenly apply erroneous ISP digital Gain to that frame, resulting in image flashes.

#### 4.2.4 Parameter Re-Timing

In order to allow the Host to send commands to the image sensor at any point in time without risk of corrupting the frame in progress, the image sensor shall internally Re-Time (delay) commands to occur at the start of the next image frame.

Integration Time, Gain, and video timing parameters shall be consistent within a frame of image data. To achieve this, many of the image sensor control registers shall be internally Re-Timed so that they are always applied at the start of the next frame. See **Section 9.5** for details.

In practice, this means that there is an optimum period for dynamic camera control commands to be sent to the image sensor to minimize the loop time and improve the response of the system.

#### 4.2.5 Generic Frame Format Description

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

The frame format information is available to the Host via CCI Registers or CCS Static Data, and optionally via Intelligent Status Line embedded data.

#### 4.2.6 Capability Registers/Parameter Limits

The capability registers allow the Host system to automatically discover image sensor capabilities. They describe the limits of parameters such as:

- Integration Time
- Analog Gain
- Line Length
- Frame Length

Thus, for example, the AEC control loop can be automatically configured even if the Host has no prior knowledge of the image sensor's characteristics. Capability information may be provided via CCI Registers and/or via CCS Static Data.

#### 4.2.7 Generic Control Interface

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

#### 4.2.8 Configuration/Tuning Files

329 Configuration/tuning and other settings files, while not required to achieve a working system, provide a way  
330 for the performance of the image sensor to be optimized, and for manufacturer-specific features to be handled.

331 Capability information is mostly read from the image sensor, and may be merged with any additional  
332 information contained in the configuration/tuning file (values read from the file having priority over values  
333 read from the image sensor). The Host then uses the merged set of information. CCS v1.1 adds support for  
334 the CCS Static Data concept, to standardize capability and configuration files. For example, CCS Static Data  
335 can be used to move capability information from CCI Registers to CCS Static Data. See *Annex B*.

#### 4.2.9 Recommended Host Behavior

336 A Host system supporting the majority of the optional CSI-2 features will function efficiently with MIPI  
337 CCS-compatible image sensors.

## 5 Image Sensor Operating Modes

### 5.1 Introduction

*Table 1* summarizes the four main image sensor Operating Modes. *Figure 3* illustrates the possible Operating Mode states and transitions among them.

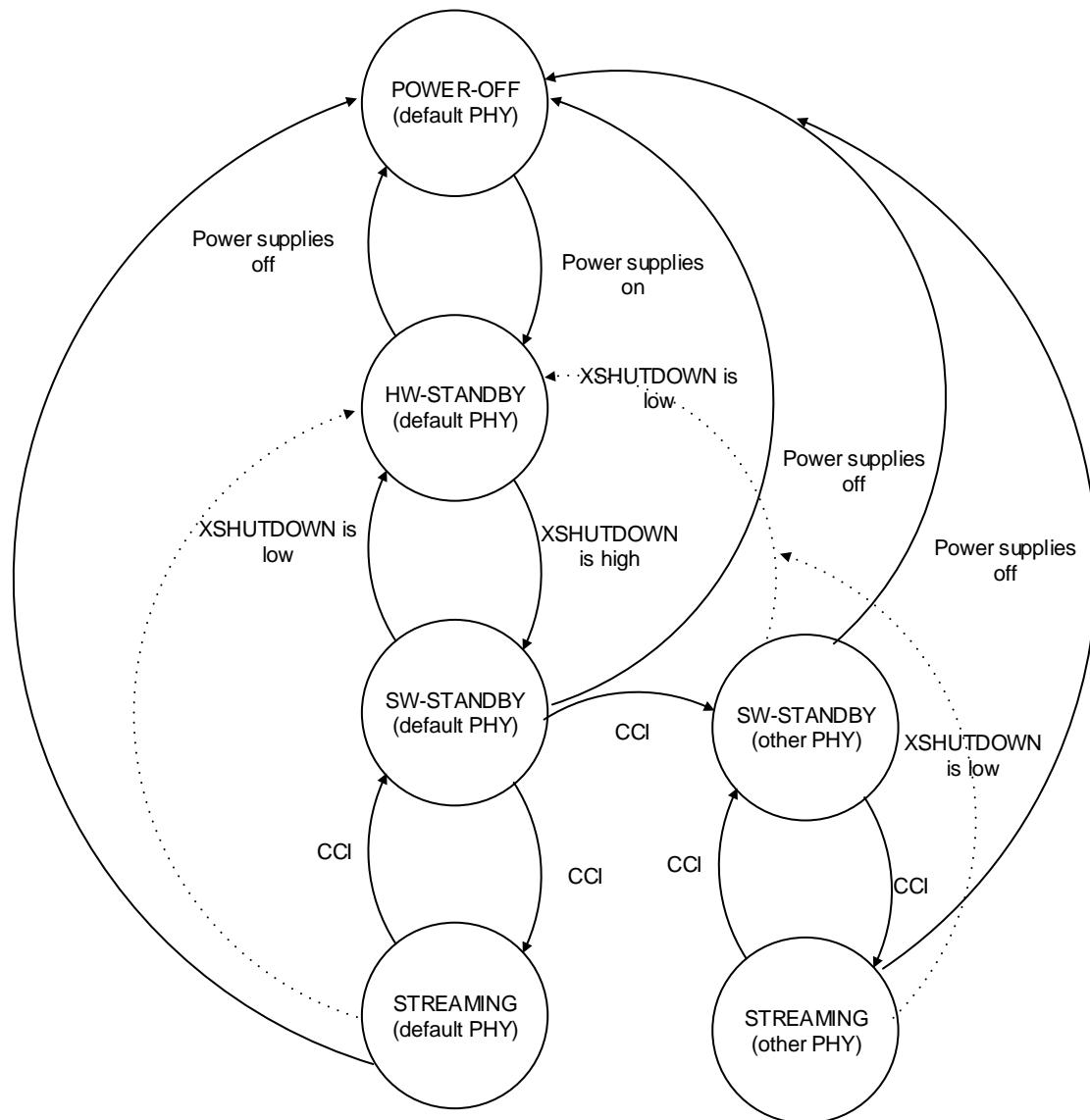
**Table 1 Operating Mode Summary**

Operating Mode (Power State)	Description
Power-Off	Power supplies are turned off
Hardware Standby	No communication with the image sensor is possible
Software Standby	CCI communication with image sensor is possible
Streaming	The image sensor is fully-powered, and is streaming image data on the CSI bus.

To configure and control the image sensor, the Host may use manufacturer-specific Operating Modes as defined in *Section 5.2.5* in addition to the four basic Operating Modes shown in *Table 1*.

Moving from one Operating Mode to another is achieved by issuing the appropriate Mode command via the CCI serial control interface, dependent upon the activity of EXTCLK, XSHUTDOWN state, and power supply states.

The image sensor also has a Soft-Reset CCI command that resets the registers back to their defaults and puts the image sensor into Software Standby Mode.



348

**Figure 3 System State Diagram**

### 5.1.1 Power-Off Mode

349 Power-Off Mode is defined as the absence of either the digital supply, the analog supply, or both.

### 5.1.2 Hardware Standby Mode

Hardware Standby Mode is defined as the presence of all Host-supplied supplies (one or more of VBAT, VANA, VIO, and/or VCore), plus the XSHUTDOWN signal being low.

Hardware Standby Mode is the Mode with the lowest possible power consumption.

#### Requirements:

1. No communication is possible over the CCI bus.
2. The Mode is entered asynchronously by forcing a low level on XSHUTDOWN pin.
3. The video timing logic is then reset, and the CCI Registers are reset to their defaults, unless they are instead set to their defaults upon exit from Hardware Standby Mode. The PLL and video blocks are powered down.
4. The SDA and SCL lines are not driven, so that other devices can use the CCI bus.
5. The CSI-2 data and clock pads are in the LP-00 state in case of D-PHY, and all CSI-2 pads are in the LP-000 state in case of C-PHY if the sensor has power supplied.

### 5.1.3 Software Standby Mode

Software Standby Mode is defined as both the digital and analog supplies being present, plus the XSHUTDOWN signal being high as shown in *Section 5.2.1*.

The Host uses Software Standby Mode:

- To initialize and setup the image sensor
- Between instances of Streaming Mode, where certain parameters need to be changed and a quick return to Streaming Mode will be required

#### Requirements

1. The EXTCLK clock shall be active for CCI communications with the image sensor.
2. All read/write CCI Registers shall be able to be read from, or written to, in Software Standby Mode.
3. All read only CCI Registers shall be able to be read from in Software Standby Mode.
4. The image sensor shall be able to receive CCI commands at **any** point in time (see exceptions related to first power-up in *Section 5.2.1*).
5. The values of the CCI Registers, e.g. Exposure and Gain, shall be preserved.
6. Relatively large power consumption is permitted, in order to achieve fast transition between Streaming Mode and Software Standby Modes without impacting image quality.

The Host configures the number of CSI-2 Lanes in Software Standby Mode.

The clock divider and PLL multiplier registers shall be configured during Software Standby Mode. The image sensor should not power down the PLL in order to allow quick changes between Software Standby Mode and Streaming Mode.

### 5.1.4 Streaming Mode

In Streaming Mode, the image sensor streams image frames to the Host system. In Streaming Mode, the image sensor shall be able to receive CCI commands at any point in time. However, the image sensor shall NACK any CCI command that it has received but is temporarily unable to process.

Streaming Mode is generally the Mode with the highest power consumption.

## 5.2 Changing Operating Modes

### 5.2.1 Power Up Sequence

While this Specification allows multiple power up sequences, the following example operation flow is recommended. Power up is further detailed later in this Section.

#### Recommended basic power up sequence:

1. Set/assert XSHUTDOWN low and then power-up the image sensor, if applicable
  - Externally supplied, switched sources for VANA and/or VIO and/or VCORE are enabled
2. Start EXTCLK
3. Release XSHUTDOWN
  - The image sensor changes to Software Standby Mode
4. Read the MIPI CCS version from image sensor's nominal CCI address
5. Read the module ID and related information through the image sensor's nominal CCI address
6. Write EXTCLK frequency info to register **extclk\_frequency\_mhz** (see *Section 5.2.9*)
7. Configure the image sensor, including the CSI interface

At this point power up and identification are complete, and the Link is initialized.

#### Note:

*CCS v1.1 introduces CCS Static Data which may include Camera module ID information, if the information is available inside the camera module (EEPROM or image sensor NVM). See Annex B.*

Power up has the following properties:

- The digital power supply may consist of either one voltage, or two voltages. This Specification describes digital voltages VIO and VCORE. VIO may have a different voltage level than VCORE. This Specification allows the use of both one-digital-voltage image sensors and two-digital-voltage image sensors, though the Figures show only the two-digital-voltage option.
- The camera module and the image sensor shall support the voltage supplies being enabled in any order.

**Examples:** All of the following orders are valid:

- VIO, VCORE, VANA
- VCORE, VIO, VANA
- VANA, VIO, VCORE
- VANA, VCORE, VIO
- VIO, VANA, VCORE
- VCORE, VANA, VIO

None, one, two, or all of the voltages may be generated by PMIC inside the camera module.

- The image sensor shall enter Software Standby Mode when XSHUTDOWN is inactive and power supplies are active.
- EXTCLK may be started either before or after XSHUTDOWN inactivation.
- After a delay of t3 or t4 (see *Table 2*), whichever is longer, both the camera module and the image sensor identification registers (see *Table 16* and *Table 17*) shall be ready to be accessed by the Host. Successful camera module identification depends upon this behavior.

During power up, an on-chip power-on reset cell shall ensure correct initialization of the CCI register values and Firmware register values to their default values.

EXTCLK can be either:

- Initially low and then enabled during Hardware Standby Mode, or
- A free running clock.

428 In Software Standby Mode it is the Host's responsibility to program and initialize the CSI interface, including  
429 both the transmitter and the receiver. The Host should program the number of CSI-2 Lanes in use. If the  
430 image sensor supports more than one PHY, then the Host should program the PHY mode (i.e., either D-PHY  
431 mode or C-PHY mode).

432 After receiving the Streaming Mode command, image sensor activity shall be as fast as possible while still  
433 conforming to the CSI-2/D-PHY/C-PHY Specifications. If the image sensor's internal initialization process  
434 has not yet completed (e.g. still reading data from NVM for sensor internal purposes, or any other  
435 initialization operations) at the time when the Host sends the Streaming Mode command, then the image  
436 sensor shall continue with its initialization process, and start streaming when ready.

437 The t7 and t8 start-up times are critical to latency. Any deviations shall be noted in the image sensor datasheet.  
438 The image sensor shall always follow valid CSI-2 D-PHY or C-PHY sequences when entering and exiting  
439 Streaming Mode.

440 **For D-PHY:** During first power-up, the image sensor may initialize D-PHY by going directly from the LP-  
441 00 state to the LP-11 state without going through the LP-10 state, as allowed by Section 6.11 of the D-PHY  
442 Specification [\[MIPI02\]](#), [\[MIPI03\]](#). The image sensor may also leave the unused Lanes in the LP-00 state,  
443 even though the Figures in this Section show a transition from the LP-00 state to the LP-11 state.

444 **For C-PHY:** The image sensor may initialize C-PHY by going directly from the LP-000 state to the LP-111  
445 state without going through the LP-100 state, as allowed by section 6.11 of the C-PHY Specification  
446 [\[MIPI04\]](#). The image sensor may also leave the unused Lanes in the LP-000 state.

447 CCS v1.1 introduces manual control when the transmitter moves enabled Lanes into the LP-11/LP-111 state.  
448 See **Section 7.9.7**.

449

**Table 2 Power Up Sequence Timing Constraints**

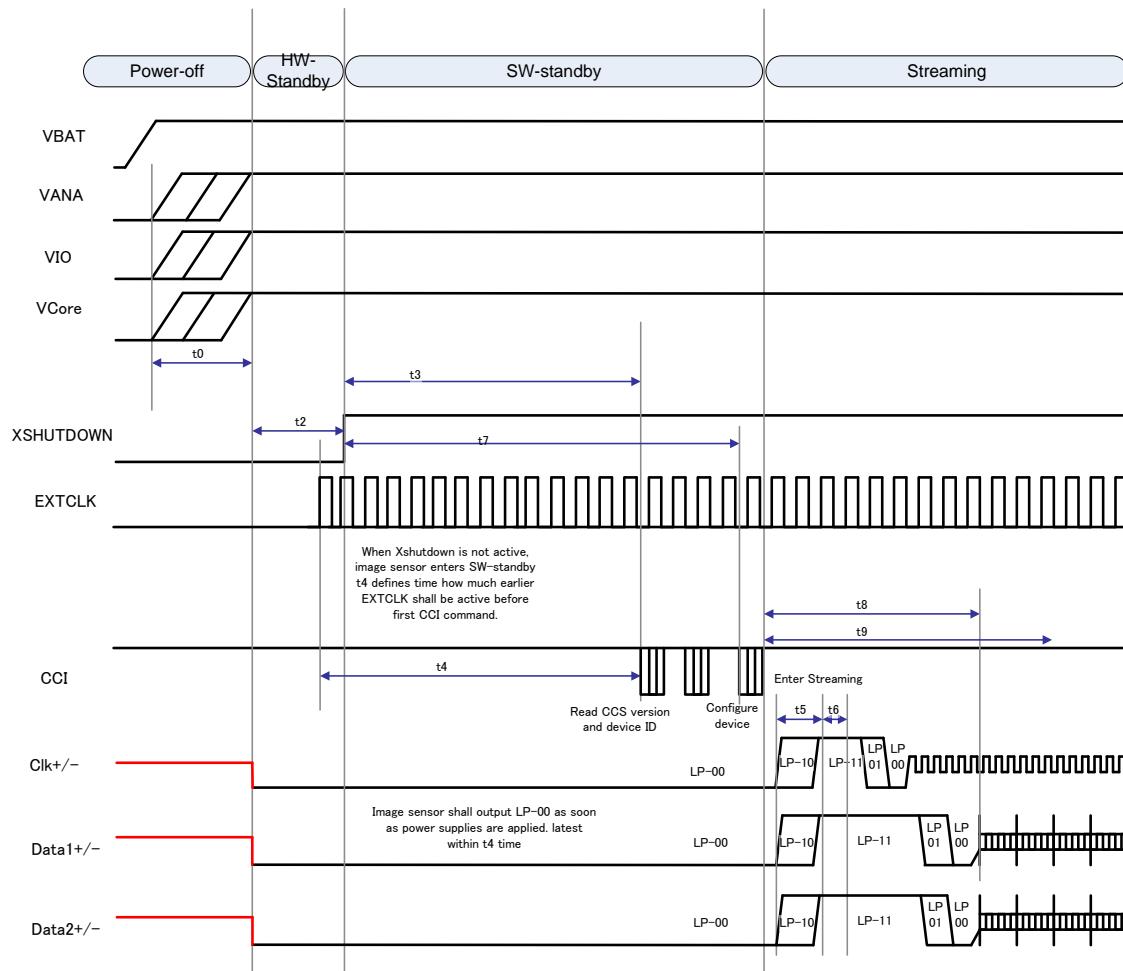
Constraint	Label	Min	Max	Units	Note
VANA, VCore, and VIO rising	t0	VANA, VCore, and VIO may rise in any order. The rising separation can vary from 0 ns to Indefinite		ns	1, 2
Last voltage of VANA, VCore, and VIO rising to XSHUTDOWN rising	t2		100	μs	1
Last voltage of VANA, VCore, and VIO rising to XSHUTDOWN rising	t2	100		μs	2
XSHUTDOWN rising to first CCI transaction (to read identification and version control information)	t3		5	ms	1
XSHUTDOWN rising to first CCI transaction (to read identification and version control information)	t3	5		ms	2, 3
EXTCLK running prior to the first CCI transaction (to read identification and version control information)	t4		5	ms	1
EXTCLK running prior to the first CCI transaction (to read identification and version control information)	t4	5		ms	2, 3
PHY power-up	t5	1	1.1	ms	1
PHY init	t6	100	110	μs	1
Minimum delay after clock is running and XSHUTDOWN is inactive to <b>mode_select = 1</b> command.	t7		20	ms	1, 3, 4
Minimum delay after clock is running and XSHUTDOWN is inactive to <b>mode_select = 1</b> command.	t7	20		ms	2, 3, 4, 5
Entering Streaming Mode to first Frame Start sequence in first power-up (assumes initialization completed before CCI command)	t8		2 ms + The delay of the coarse Integration Time value	ms	1, 3, 6
Entering Streaming Mode to first Frame Start sequence of good quality frame in first power-up	t9		t8 + 1 frame	ms	1

**Note:**

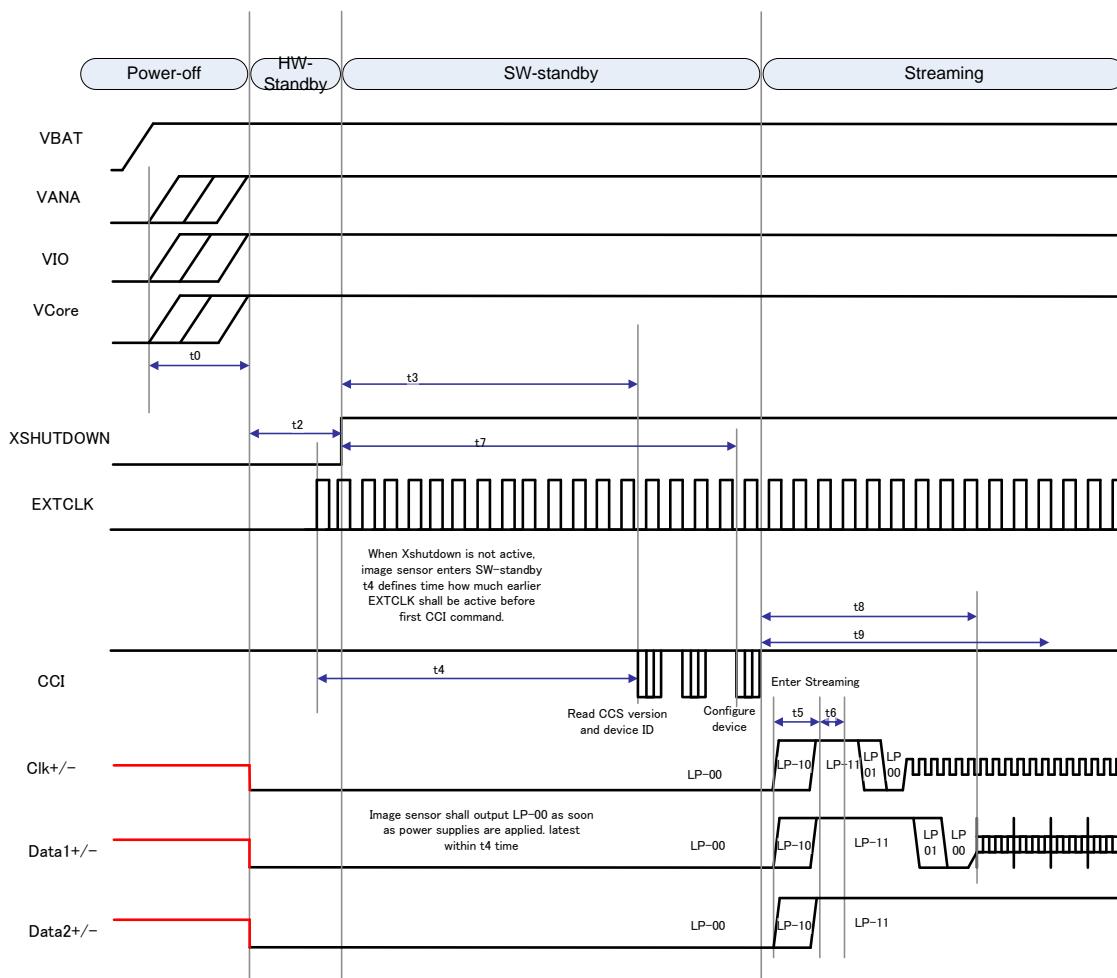
- 450 1. *Image-sensor-specific requirement*  
 451 2. *Host specific requirement*  
 452 3. *After t3 and t4, the Host can detect what camera module is connected to the Host. Following such*  
 453 *detection, the Host may want to use image-sensor-specific delay values, or long enough delay val-*  
 454 *ues for t7 and t8.*  
 455 4. *Target image sensor may require longer time to finish its internal initialization.*  
 456 5. *The Host should wait for this interval before starting sensor initialization.*  
 457 6. *Target image sensor may require longer time.*

To further detail the image sensor requirements shown in **Table 2, Figure 4, and Figure 5**:

- **t0:** The image sensor shall support VANA, VCORE, and VIO rising in any order and with any separation in their rising times
- **t2:** The image sensor shall be ready to accept rising XSHUTDOWN after last the voltage rising within this interval
- **t3 and t4:** The image sensor shall be ready to accept its first CCI transaction after XSHUTDOWN rising and EXTCLK running within these intervals
- **t5:** The image sensor shall power up the PHY within this interval
- **t6:** The image sensor shall initialize the PHY within this interval
- **t7:** The image sensor should be ready to accept a Start Streaming command within this interval after XSHUTDOWN rising and EXTCLK running. If the image sensor is unable to meet this interval, then the image sensor datasheet shall show the required timing.
- **t8:** The image sensor should output the first image frame within this interval. If the image sensor is unable to meet this interval, then the image sensor datasheet shall show the required interval.
- **t9:** The image sensor shall output the first good quality image frame within this interval.



**Figure 4 Power Up Sequence with D-PHY, All Lanes In Use**



475

**Figure 5 Power Up Sequence with D-PHY, 2-Lane System with Only One-Lane In Use**

### 5.2.2 Power Down Sequence

While this Specification allows multiple power down sequences, the following example operation flow is recommended. Power down is further detailed later in the Section.

#### Recommended basic power down sequence:

1. Activate XSHUTDOWN
2. Stop EXTCLK
3. Power down the image sensor, if applicable
  - Externally supplied, switched sources for VANA and/or VIO and/or VCORE are disabled
4. The image sensor is in either Power-Off Mode or Hardware Standby Mode.

Power down has the following properties:

- The digital power supply may consist of either one voltage, or two voltages. This Specification describes digital voltages VIO and VCORE. VIO may have a different voltage level than VCORE. This Specification allows the use of both one-digital-voltage image sensors and two-digital-voltage image sensors, though the Figures show only the two-digital-voltage option.
- The digital and analog supply voltages can be powered down in any order. For example, all of the following orders are valid:
  - VIO, VCORE, VANA
  - VCORE, VIO, VANA
  - VANA, VIO, VCORE
  - VANA, VCORE, VIO
  - VIO, VANA, VCORE
  - VCORE, VANA, VIO

None, one, two, or all of the voltages may be generated by PMIC inside the camera module.

- Hardware Standby Mode is activated when XSHUTDOWN is active. In Power Off Mode, one or more power supplies are turned off.
- When XSHUTDOWN is activated, the CSI interface is disabled.

501

**Table 3 Power Down Timing Constraints**

Constraint	Label	Min	Max	Units	Note
Enter Software Standby CCI command to device in Software Standby Mode	t0		See Section of rolling shutter mode transitions		1, 2
Number of EXTCLK cycles after LP-11 or LP-111 stop state in CSI-2 and last CCI command	t1		512	EXTCLK cycles	1
Number of EXTCLK cycles after LP-11 or LP-111 stop state in CSI-2 and last CCI command	t1	512		EXTCLK cycles	2
XSHUTDOWN falling after LP-11 or LP-111 stop state in CSI-2 and last CCI command	t2		512	EXTCLK cycles	1
XSHUTDOWN falling after LP-11 or LP-111 stop state in CSI-2 and last CCI command	t2	512		EXTCLK cycles	2
Time that sensor shall be able to support between XSHUTDOWN falling to VANA falling	t3	0.0	infinite	ns	1
VANA, VCore, and VIO falling	t4	VANA, VCore, and VIO may fall in any order. The falling separation can vary from 0ns to Indefinite		ns	1, 2, 3

502

**Note:**

503

1. *Image-sensor-specific requirement*
2. *Host specific requirement*
3. *The image sensor shall support any order and any separation*

504

505

506

507

To further detail the image sensor requirements shown in **Table 3** and **Figure 6**:

508

509

510

511

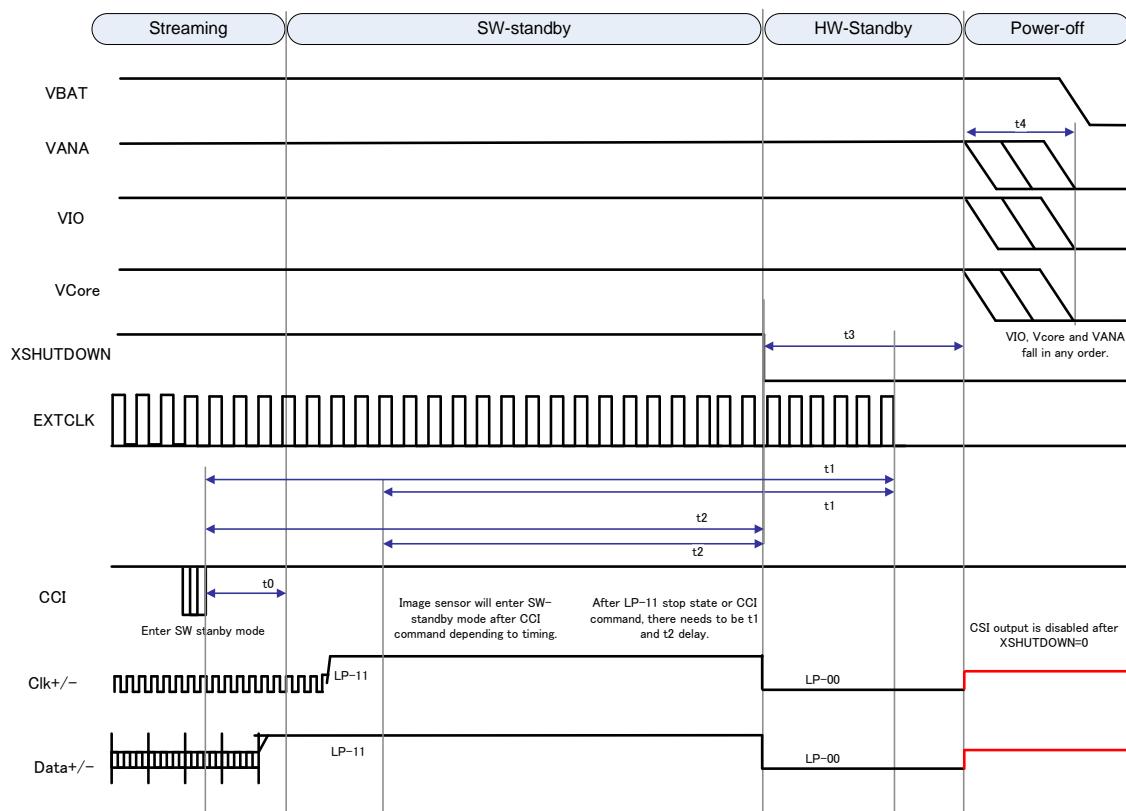
512

513

514

515

- **t0:** See **Section 5.2.4**.
- **t1:** The image sensor shall be ready to accept stopping EXTCLK after the stop state in CSI-2 and the last CCI command within this interval
- **t2:** The image sensor shall be ready to accept falling XSHUTDOWN after the stop state in CSI-2 and the last CCI command within this interval
- **t3:** The image sensor shall support falling VANA after falling XSHUTDOWN within this interval
- **t4:** The image sensor shall support VANA, VCore, and VIO falling in any order and with any separation in their falling times



516

Figure 6 Power Down Sequence with D-PHY

### 5.2.3 Mode Change Sequence

In normal camera operation, changing between Streaming Mode and Software Standby Mode is a very common and vital event which must execute reliably and with low latency. For example, image quality shall be high even in the first output frames, and Mode change latencies must be short.

As specified elsewhere in this document, the only times when the clock divider and PLL multiplier registers can be configured are during Software Standby Mode. In order to allow fast changes between Software Standby Mode and Streaming Mode, the powering down of internal blocks within the image sensor (e.g. the PLL) should be minimized. The image sensor shall ensure that if the PLL is running and the PLL control registers are changed, then any loss and re-acquisition of PLL lock is handled internally by the image sensor and not visible to the Host.

CSI-2 Lane configurations may be changed during Software Standby Mode. The Host may send a command to change Lane configuration, and then send a command to enter Streaming Mode. The image sensor shall ensure that no timings required by D-PHY or C-PHY are violated, regardless of the delay between the Lane configuration command and the streaming command.

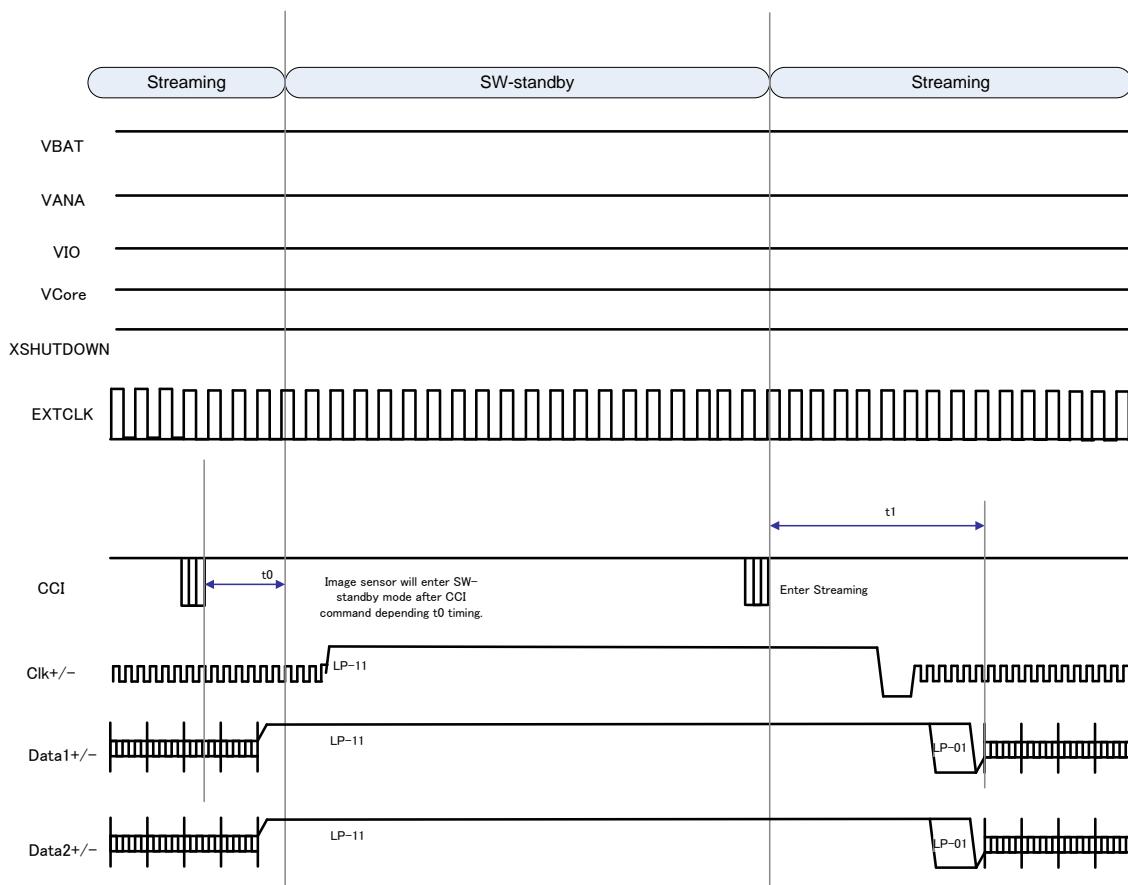
When changing Lane configurations, if a Lane previously in use is taken out of use then the ULPS entry sequence shall be followed. If a Lane previously not in use is put into use, then the ULPS exit sequence shall be followed if applicable (after first power up, the unused Lane may already be in the LP-11 state or the LP-111 state). It is also permissible for all Lanes to automatically enter ULPS when Software Standby Mode is entered and then automatically exit ULPS when Streaming Mode is entered. Note that if register **manual\_LP\_ctrl** is supported, it may be used to move Lanes from ULPS state to the LP-11/LP-111 state; see [Section 7.9.7](#).

t1 is a performance figure. Any deviation from it shall be visible in the image sensor datasheet.

The image sensor shall always follow valid CSI-2 D-PHY or C-PHY sequences when entering and exiting Streaming Mode.

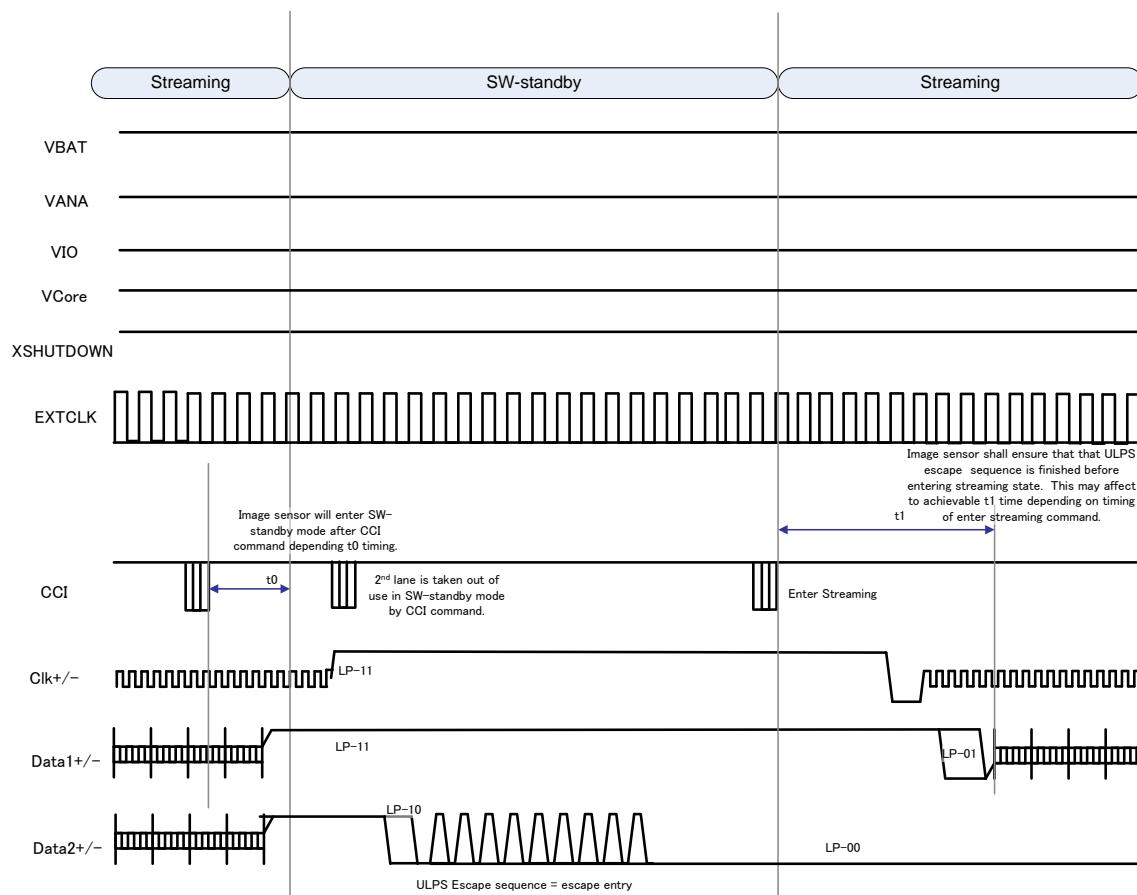
**Table 4 Mode Change Sequence Timing Constraints**

Constraint	Label	Min	Max	Units
Enter Software Standby CCI command – Device in Software Standby Mode	t0	See Section of rolling shutter mode transitions		
Entering Streaming Mode – First Frame Start Sequence with good quality	t1		1 ms + coarse Integration Time	
PHY power-up	t5	1	1.1	ms
PHY init	t6	100	110	μs



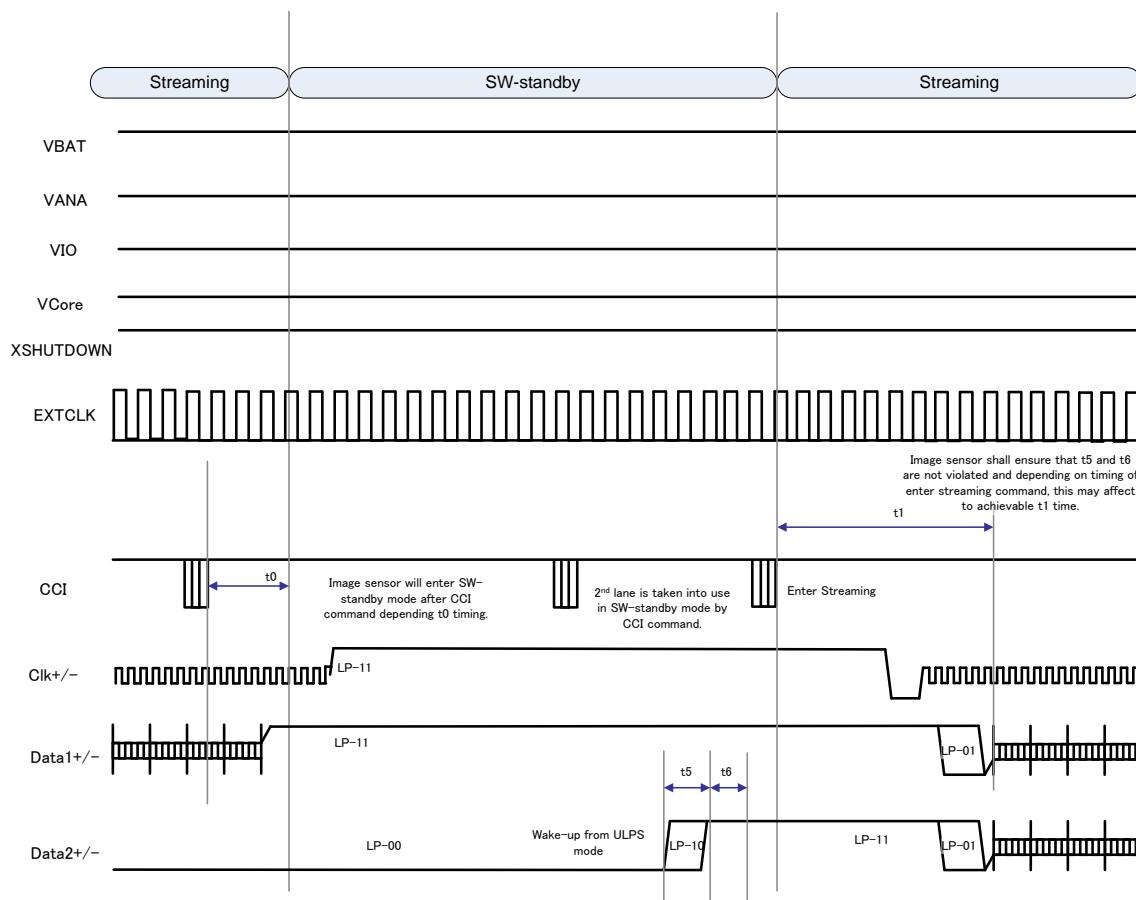
541

**Figure 7 Mode Change with D-PHY**



542

**Figure 8 Mode Change, 2-Lane D-PHY to 1-Lane D-PHY**



543

**Figure 9 Mode Change, 1-Lane D-PHY to 2-Lane D-PHY**

### 5.2.4 Rolling Shutter Mode Transitions

544 An image sensor should support an additional way of terminating frames before entering Software Standby  
 545 Mode, in addition to the standard method of allowing the existing frame to complete before Mode entry. The  
 546 image sensor shall indicate which methods are supported by the following capability register:

547 **Table 5 Fast Standby Capability Register**

Register Name	Type	RW	Comment
<b>fast_standby_capability</b>	8-bit unsigned integer	RO	0: Frame truncation not supported for rolling shutter 1: Frame truncation supported for rolling shutter

548 This function shall be selected by the **fast\_standby\_ctrl** register. The Host may change the value of that  
 549 register during Software Standby Mode, or even during Streaming Mode, and the image sensor shall take the  
 550 new value into use immediately. The image sensor shall not set any timing restrictions between the  
 551 **fast\_standby\_ctrl** command and the **mode\_select** command.

552 **Table 6 Fast Standby Control Register**

Register Name	Type	RW	Comment
<b>fast_standby_ctrl</b>	8-bit unsigned integer	RW	0: Frame completes before Mode entry 1: Frame may be truncated before Mode entry <b>Default: 0</b>

553 The image sensor may be in one of three places in the frame being sent at the time the command to enter  
 554 Software Standby Mode is received:

- 555 • **Command Received During Frame Blanking**

556 The image sensor shall immediately enter Software Standby Mode (same behavior as standard  
 557 Mode). It is assumed the Lanes will already be in an LP idle state, and that Software Standby  
 558 Mode can be entered immediately.

- 559 • **Command Received During the Active Line**

560 The image sensor shall complete the existing packet, terminate the frame with a frame end packet,  
 561 and then immediately enter Software Standby Mode. The image sensor shall ensure that this  
 562 termination does not produce consecutive frame end packets.

- 563 • **Command Received During Line Blanking**

564 The image sensor shall terminate the frame with a frame end packet, and then immediately enter  
 565 Software Standby Mode. It is assumed that the Lanes will already be in an LP idle state. The  
 566 image sensor shall ensure that this termination does not produce consecutive frame end packets.

567 **Note:**

568 *If an image sensor is outputting more than one overlapping frame using virtual channel  
 569 interleaving, then both the visible image data stream and other data streams should be  
 570 terminated. In addition, the image sensor shall ensure that the proper number of frame end  
 571 packets is always generated (i.e. one for each virtual channel stream).*

572 *If an image sensor uses interleaved PDAF readout (as described in **Section 17.4.1**), then both  
 573 the visible image data stream and PDAF data stream shall be terminated. In addition, the image  
 574 sensor shall ensure that the proper number of frame end packets is always generated (e.g. two  
 575 for the PDAF use case in **Section 17.4.1**).*

### 5.2.5 Mode Select Register

The image sensor's Operating Mode is controlled via the **mode\_select** register (see *Table 7*).

The register value selects the Operating Mode:

- Mode 0: Software Standby Mode
- Mode 1: Streaming Mode
- Modes 2 to 15: Reserved for future definition by MIPI Alliance
- Modes 16 to 31: Manufacturer-specific Modes.

**Table 7 Mode Select Register**

Register Name	Type	RW	Comment
<b>mode_select</b>	8-bit unsigned integer	RW	0: Software Standby Mode (CCI communications active) 1: Streaming Mode (Active Video) 2–15: Reserved for future definition by MIPI Alliance 16–31: Manufacturer-Specific Modes <b>Default:</b> 0

### 5.2.6 Software-Reset Via CCI Interface

Setting the **software\_reset** register value to 1 resets all of the CCI Registers and firmware registers in the particular CCI address to their default values. The value of the **software\_reset** register is also reset.

**Table 8 Soft Reset Register**

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

A write to this “Soft Reset” shall put the image sensor into Software Standby Mode, and then cause the power up sequence (see *Section 5.2.1*) to be followed, in order to generate streaming image data from the image sensor. If the image sensor is in the Streaming state, then the Host should ensure that the image sensor is returned to the Software Standby state prior to setting register **software\_reset** to 1, because otherwise the image sensor may exhibit output behavior that violates the CSI-2 specification. If the image sensor is in the Streaming state and also supports “frame truncation” (as indicated by **fast\_standby\_capability** = 1 in *Section 5.2.4*), then the Host shall first return the image sensor to the Software Standby state with frame truncation enabled (**fast\_standby\_ctrl** set to 1) prior to setting **software\_reset** to 1.

The purpose of this control is to allow the image sensor to be reset to a known state via a CCI command, in case the Host system loses track of the image sensor’s state. The image sensor may NACK additional CCI commands while executing the **software\_reset** command, and shall set **software\_reset** to 0 upon command completion. The image sensor should support values for Soft Reset as defined in *Table 9*. The image sensor shall report Soft Reset Specification maximum values by registers **reset\_max\_delay** and **reset\_min\_time**, as defined in

*Table 10*.

601

**Table 9 Soft Reset Specifications**

Constraint	Label	Min	Typical	Max	Units
Maximum time from the stop condition of CCI soft reset command until sensor is reset	t1	–	–	400	μs
Minimum time between successive CCI soft reset commands	t2	–	–	2	ms

602

**Table 10 Soft Reset Limit Registers**

Register Name	Type	RW	Comment
<b>reset_max_delay</b>	8-bit unsigned integer	RO	Maximum time for t1 (see <b>Table 9</b> ) Max time = register value * 100μs + 400μs <b>Example:</b> 0: 400μs 1: 500μs <b>Recommended Value:</b> 0
<b>reset_min_time</b>	8-bit unsigned integer	RO	Maximum time for t2 (see <b>Table 9</b> ) Max time = register value * 500μs + 2000μs For example 0: 2000μs 1: 2500μs <b>Recommended Value:</b> 0

### 5.2.7 Frame Count Register

The value of the **frame\_count** register increments by 1 at the start of each frame. When the value 254 (0xFE) is reached, the frame counter value rolls over to 1 and continues to increment.

During Software Standby Mode the frame counter value is reset to 255 (0xFF). The frame counter value for the first output frame following reset shall be 1.

**Table 11 Frame Counter Register**

Register Name	Type	RW	Comment
<b>frame_count</b>	8-bit unsigned integer	RO Dynamic	Increments by 1 at the start of each frame (unless frame is masked as described in <b>Section 9.5.4</b> ) <b>Default:</b> 255 (0xFF)

The value of the frame count register shall be also included in the frame start (FS) and frame end (FE) packets as the LS Byte of the frame number. See the CSI-2 Specification **[MIPI01]** for details.

### 5.2.8 XSHUTDOWN

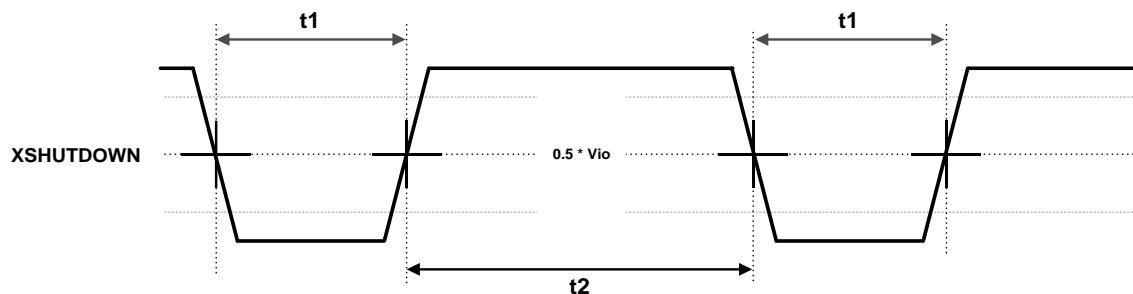
XSHUTDOWN is an asynchronous system reset signal. It is active low.

If XSHUTDOWN goes low in any Operating Mode other than Power-Off, the image sensor shall immediately move in to Hardware Standby Mode. When XSHUTDOWN goes high and the power supplies are present, the image sensor goes into Software Standby Mode. See details in the Power Up Sequence (**Section 5.2.1**).

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

**Table 12 XSHUTDOWN Specifications**

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



**Figure 10 XSHUTDOWN Timing Specification**

### 5.2.9 EXTCLK

The image sensor may use the information in the **extclk\_frequency\_mhz** register in order to adapt its internal operation. The Host shall write EXTCLK frequency information after reading identification information before initializing the image sensor, as shown in *Section 5.2.1*. The image sensor may omit the **extclk\_frequency\_mhz** register if it does not use the frequency information.

**Table 13 Extclk Frequency Register**

Register Name	Type	RW	Comment
<b>extclk_frequency_mhz</b>	16-bit unsigned iReal	RW	Nominal frequency (8.8 fixed point number) <b>Default:</b> Image-sensor-specific

### 5.2.10 GPIO\_TRIG

The GPIO\_TRIG signal is used in this version of this Specification as an external input trigger signal. It is possible that future versions of this Specification might use the GPIO\_TRIG signal for additional functionalities. Those would be controlled by the register **GPIO\_TRIG\_mode**, which is a Read-Only register in this version of this Specification. See *Section 9.7.3* for the description of the delayed Exposure start functionality.

The minimum pulse width is 1  $\mu$ s.

**Table 14 GPIO\_TRIG Mode Control Registers**

Register Name	Type	RW	Comment
<b>GPIO_TRIG_mode</b>	8-bit unsigned integer	RO	0: Input used as an external trigger to start Exposure. Other values: reserved <b>Default:</b> 0

**Table 15 GPIO\_TRIG Specifications**

Constraint	Min	Typical	Max	Units
Pulse width	1	—	—	$\mu$ s

## 6 Module and Sensor Version and Identification

This Section specifies the Module Version Identification Registers and the Image Sensor Identification Registers.

### 6.1 Module Version Identification Registers

The camera module shall be identified by the Camera Module Identifier Registers (see *Table 16*). The camera module shall implement all Camera Module Identifier Registers listed in *Table 16*. The information may be provided as part of CCS Static Data, if the information is available inside the camera module (EEPROM or image sensor NVM). See *Annex B*.

The model identification shall be reflected in the **module\_model\_id** register. This contains a unique, arbitrary number chosen by the manufacturer for a particular device type.

Module version identification shall be reflected in the two registers **module\_revision\_number\_major** and **module\_revision\_number\_minor**. When module properties change such that new configuration/tuning parameters are needed, then the **module\_revision\_number\_major** shall be changed. For example, if actuator properties or image quality tuning related properties change so that new ISP tunings are required, then **module\_revision\_number\_major** needs to be updated.

It is possible that changes in following items might cause a need for a change in **module\_revision\_number\_major**:

- Actuator
- Lens
- Firmware
- Chip versions

Other changes in the module that do not require configuration/tuning parameter changes should be identified by **module\_revision\_number\_minor**. This increases the traceability of camera module changes.

The registers **module\_date** and **module\_serial\_number** can be used to identify an exact module for R&D purposes. In such cases, these values are captured during module production.

Most of the register values are expected to be programmed at the camera module manufacturing stage. As a result they must be implemented in some form of NVM/OTP, and then copied to MIPI CCS Register space during camera module start-up. From the Host's perspective, the data is visible as normal read-only registers.

656

**Table 16 Camera Module Identifier Registers**

Register Name	Type	RW	Comment
<b>module_model_id</b>	16-bit unsigned integer	RO	Model Identification Number of the camera module
<b>module_revision_number_major</b>	8-bit unsigned integer	RO	Device Revision Identifier of the camera module for configuration/tuning change
<b>module_revision_number_minor</b>	8-bit unsigned integer	RO	Device Revision Identifier of the camera module for minor changes
<b>module_manufacturer_id</b>	16-bit unsigned integer	RO	Module manufacturer Code Please refer to the MIPI Alliance Manufacturer ID page [ <a href="#">MIPI05</a> ]
<b>MIPI_CCS_version</b>	8-bit unsigned integer	RO	CCS Version <b>Bits 7–4:</b> Major <b>Bits 3–0:</b> Minor <b>Examples:</b> 0x10 means version 1.0 0x11 means version 1.1
<b>module_date_year</b>	8-bit unsigned integer	RO	<b>Bits 6–0:</b> YEAR, Last two decimal digits of manufacturing year, range 0 to 99. For example year 2016 is saved as 0x10 i.e. '16' DEC.
<b>module_date_month</b>	8-bit unsigned integer	RO	<b>Bits 3–0:</b> MONTH: Manufacturing month, range 1 to 12
<b>module_date_day</b>	8-bit unsigned integer	RO	<b>Bits 4–0:</b> DAY, Manufacturing day, range 1 to 31
<b>module_date_phase</b>	8-bit unsigned integer	RO	<b>Bits 2–0:</b> PHASE, Phase 0: TS (Technical Sample) 1: ES (Engineering Sample) 2: CS (Commercial Sample) 3: MP (Mass production)
<b>module_serial_number</b>	32-bit unsigned integer	RO	Running number

## 6.2 Image Sensor Identification Registers

Additional registers are defined for Image Sensor Identification. The camera module shall implement all Image Sensor Identifier Registers listed in *Table 17*, and shall use them to report image sensor changes. Note that CCS v1.1 introduced support for the 16-bit **sensor\_revision\_number\_16** register. The image sensor shall support either the 8-bit **sensor\_revision\_number** or the 16-bit **sensor\_revision\_number\_16** register, and the other, unsupported register shall have value 0. The information may be provided as part of CCS Static Data, if the information is available inside the camera module (EEPROM or image sensor NVM). See *Annex B*.

**Table 17 Image Sensor Identifier Registers**

Register Name	Type	RW	Comment
<b>sensor_model_id</b>	16-bit unsigned integer	RO	Model Identification Number of the image sensor
<b>sensor_revision_number</b>	8-bit unsigned integer	RO	Device Revision Identifier of image sensor silicon (8-bit version)
<b>sensor_revision_number_16</b>	16-bit unsigned integer	RO	Device Revision Identifier of image sensor silicon (16-bit version)
<b>sensor_manufacturer_id</b>	16-bit unsigned integer	RO	Manufacturers Code Please refer to the MIPI Alliance Manufacturer ID page [ <a href="#">MIPI05</a> ]
<b>sensor_firmware_version</b>	8-bit unsigned integer	RO	Sensor firmware version

This page intentionally left blank.

## 7 Data Transmission and Data Formats

### 7.1 Introduction

This Section specifies data transmission, its control, frame formats, and data format control.

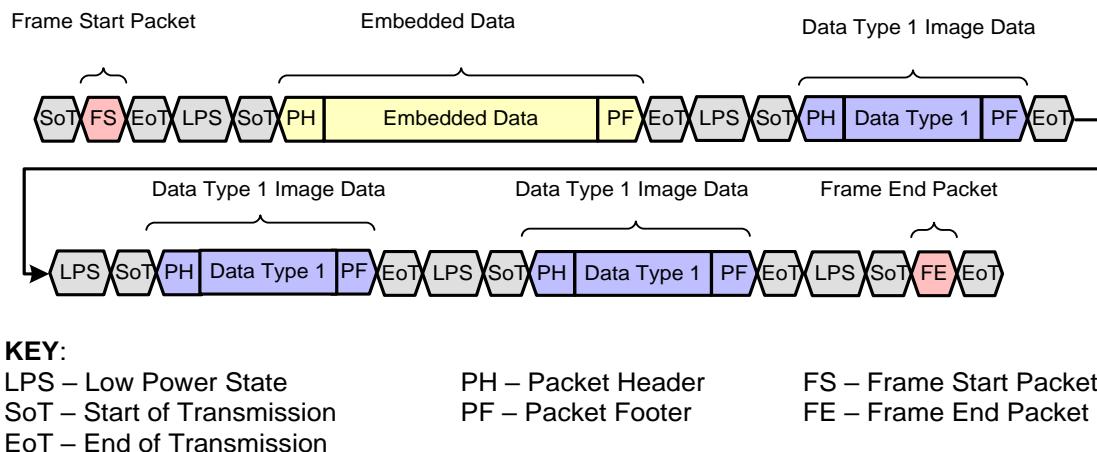
### 7.2 CSI-2 Introduction

A CSI-2 image sensor supports either C-PHY or D-PHY. This CCS Specification supports multiple PHY versions, for example for D-PHY the version can be v1.2 or earlier, v2.0, v2.1, or v2.5. D-PHY v1.2 and later can be used if the calibration sequence is needed. For C-PHY the version can be v1.0, v1.1, v1.2, or v2.0.

The image sensor shall support all requirements that are mandatory in the selected version(s) of the MIPI CSI-2 Specification, and all requirements that are mandatory in the chosen version of the MIPI D-PHY Specification and/or the MIPI C-PHY Specification (whichever is supported).

The following list highlights just some of these requirements:

- Visible pixel data shall follow the CSI-2 data type definitions. For example, RAW8 shall use data type 0x2A.
- User defined data types shall be used to transfer DPCM compressed raw data.
- The image sensor shall support payload data rules as specified in CSI-2 (see the relevant Section for each data type in the CSI-2 Specification).
- Support for Generic Short Packet data types is not required.
- Support for embedded data with image data frames is required. Top embedded data shall use data type 0x12, as specified in CSI-2.
- For bottom embedded data, it shall be possible to use data type 0x12 and user defined data type. The selection shall be configurable by register.
- Support for Line Synchronization Packets is not required.
- ULPS shall be supported in SW Standby and for unused Lanes in Streaming mode. See *Section 5* for details and possible exceptions related to Streaming mode.
- For D-PHY:
  - Some of the D-PHY timings may be Host controllable.
  - The continuous CSI-2 clock shall be implemented, and it shall be used as a default.
  - Support for non-continuous CSI-2 clock operation is not required.
- For a CSI-2 transmitter:
  - The Data Lane module shall support: Unidirectional Primary, HS-TX, LP-TX, and a CIL-PFEN function.
  - The Clock Lane module shall support: Unidirectional Primary, HS-TX, LP-TX, and a CIL-PCNN function.
  - All CSI-2 implementations shall support forward escape ULPS on all Data Lanes. Other forward escape modes are optional.
- For C-PHY:
  - Some of the C-PHY timings may be Host controllable.
  - For a CSI-2 transmitter, the Lane module shall support: Unidirectional Primary, HS-TX, LP-TX, a CIL-PFEN function and Sync Word insertion during data payload transmission.
  - All CSI-2 implementations shall support forward escape ULPS on all C-PHY Lanes. Other forward escape modes are optional.

**Figure 11 Example of Valid CSI-2 Frame**

703 Recommendations for frame start and end packet spacing:

- 704
- 705 • The spacing between the Frame Start packet and the first data packet should be as close as  
706 possible to the minimum packet spacing.
  - 707 • The spacing between the last data packet and the Frame End packet should be as close as possible  
708 to the minimum packet spacing.

### 7.3 Size Rules

The following rules clarify responsibilities for ensuring correct output formatting:

1. The Host shall determine the region of interest (analog ROI) from the pixel array, binning, and subsampling settings, such that Bayer elements (i.e. the 4-pixel group) are not split.
2. Although it is recognised that digital scaling settings cannot be restricted as per the above rules, the Host shall nonetheless recognize that the maximum number of valid pixels that can be recovered from the scaled output shall be determined by the above rules.
3. Only values of (**x\_output\_size** + additional columns) which fulfil CSI-2 RAW format packet data size constraints are valid. The Host is responsible for setting appropriate values, determined by the CSI-2 payload specification; see also examples below.
4. In cases where the **x\_output\_size** and **y\_output\_size** define a ‘window’ that is larger than the amount of valid pixel data after analog ROI, subsampling, binning, and digital scaling operations, and bit 2 of the **non\_flexible\_resolution\_support** capability register (*Table 68*) is set to 0, the image sensor shall pad with undefined data between the end of valid data and the actual output size. I.e., the valid data shall always be ‘top left justified’ within the output size ‘window’.
5. In cases where the **x\_output\_size** or **y\_output\_size** define a ‘window’ that is smaller than the amount of valid pixel data after analog ROI, subsampling, binning, and digital scaling operations, the image sensor shall crop to the actual output size. I.e., the valid data shall always be ‘top left justified’ within the output size ‘window’.

#### 7.3.1 RAW10 Example (Informative)

**Note:**

*The following example is informative only. If there is conflict with the CSI-2 Specification, then the CSI-2 Specification takes precedence. See also Section 8.2.1 and Section 8.2.1.2.*

For a 2592x1944 array, analog binned to 1296x972 and digitally horizontally scaled by 29/16, there will be 715 complete columns output from the scaler. Of these, the Host can only use 714 to avoid splitting Bayer elements. However, to recover all 714 columns the Host shall set the **x\_output\_size** to 716 (assuming bit 2 of **non\_flexible\_resolution\_support** is set to 0 and there are no additional columns in this image sensor). This is because RAW10 over CSI-2 requires the line length per packet to be a multiple of 8 bits, and the requirement not to split a packed pixel also imposes an integer multiple of 4 pixels. These two rules are both satisfied on 4-pixel boundaries. The image sensor pads the remainder of the line between the last complete pixel and the actual **x\_output\_size** with undefined data. Note that if bit 2 of **non\_flexible\_resolution\_support** is set to 1, then the image sensor doesn’t support padding, but the Host can still satisfy CSI-2 RAW10 packet size rules by setting **x\_output\_size** to 712.

In the same case, as there is no vertical scaler in the y-axis, there would be 972 complete lines output from the scaler and the **y\_output\_size** would be set to 972.

However, if there were a vertical scaler, then there would be 536 complete lines output from the scaler. Of these, the Host can use all 536 and avoid splitting Bayer elements. To recover all 536 lines (assuming that there are no additional rows in this image sensor), the **y\_output\_size** can be set to 536 (because the CSI-2 data size rule only applies in the x direction). Valid data extends right up to the output size here, so no padding or cropping is required.

### 7.3.2 RAW12 Example (Informative)

747 **Note:**

748     *The following example is informative only. If there is conflict with the CSI-2 Specification, then the*  
749     *CSI-2 Specification takes precedence. See also Section 8.2.1 and Section 8.2.1.2.*

750     For a 3464x2468 array, analog binned to 1732x1234 and digitally horizontally scaled by 18/16, there will be  
751     1539 complete columns output from the scaler. Of these, the Host can only use 1538 to avoid splitting Bayer  
752     elements. RAW12 transmission over CSI-2 adds no further restrictions, since the packing requires only that  
753     even pixel boundaries are selected. The Host can set the **x\_output\_size** (assuming no additional columns) to  
754     1538 and receive only the desired image data.

755     In the same case, as there is no vertical scaling in the y-axis, there would be 1234 complete lines output from  
756     the scaler and the **y\_output\_size** would be set to 1234.

757     However, if there were a vertical scaler, then there would be 1096 complete lines output from the scaler. Of  
758     these, the Host can use all 1096 and avoid splitting Bayer elements. To recover all 1096 lines (assuming that  
759     there are no additional rows in this image sensor), the **y\_output\_size** can be set to 1096. Valid data extends  
760     right up to the output size here, so no padding or cropping is required.

## 7.4 Signaling Options

### 7.4.1 CSI Signaling Mode

This MIPI CCS Specification supports two main signaling modes for the image data interface:

- CSI-2 using D-PHY
- CSI-2 using C-PHY

The default signaling mode after power-up may be either D-PHY or C-PHY. The Host can determine which by reading the default value of the register **CSI\_signaling\_mode**. Mode switching from D-PHY to C-PHY, or vice versa, is allowed in the first SW Standby mode after power up, as shown in *Figure 3*. The **CSI\_signaling\_mode** register can be used to control this.

**Table 18 CSI Signaling Mode Register**

Register Name	Type	RW	Comment
<b>CSI_signaling_mode</b>	8-bit unsigned integer	RW	0: Reserved 1: Reserved 2: CSI-2 with D-PHY 3: CSI-2 with C-PHY <b>Default:</b> Image-sensor-specific (2 or 3)

### 7.4.2 CSI Lane Mode

The **CSI\_lane\_mode** register controls the Lane configuration used with either D-PHY or with C-PHY. The Host shall configure the number of used Lanes while in SW Standby mode. D-PHY uses two-wire Data Lanes, whereas C-PHY uses three-wire Lanes carrying both Data and Lane-specific embedded Clock information.

**Table 19 CSI Lane Mode Register**

Register Name	Type	RW	Comment
<b>CSI_lane_mode</b>	8-bit unsigned integer	RW	0: 1-Lane 1: 2-Lane 2: 3-Lane 3: 4-Lane 4: 5-Lane 5: 6-Lane 6: 7-Lane 7: 8-Lane <b>Default:</b> Image-sensor-specific (0 to 7).

### 7.4.3 CSI Signaling Mode Capability

The **CSI\_signaling\_mode\_capability** register indicates which signaling modes (D-PHY, C-PHY, or both) the image sensor supports.

**Table 20 CSI Signaling Mode Capability Register**

Register Name	Type	RW	Comment
<b>CSI_signaling_mode_capability</b>	8-bit unsigned integer	RO	<b>Bits 0 &amp; 1</b> Reserved <b>Bit 2</b> 1: CSI-2 with D-PHY supported 0: Not supported <b>Bit 3</b> 1: CSI-2 with C-PHY supported 0: Not supported

### 7.4.4 CSI D-PHY Lane Mode

The **CSI\_DPHY\_lane\_mode\_capability** register indicates which Data Lane usages the image sensor supports when D-PHY is used.

**Table 21 CSI D-PHY Lane Mode Capability Register**

Register Name	Type	RW	Comment
<b>CSI_DPHY_lane_mode_capability</b>	8-bit unsigned integer	RO	<b>Bit 0</b> 1: 1-Lane supported 0: Not supported <b>Bit 1</b> 1: 2-Lane supported 0: Not supported <b>Bit 2</b> 1: 3-Lane supported 0: Not supported <b>Bit 3</b> 1: 4-Lane supported 0: Not supported <b>Bit 4</b> 1: 5-Lane supported 0: Not supported <b>Bit 5</b> 1: 6-Lane supported 0: Not supported <b>Bit 6</b> 1: 7-Lane supported 0: Not supported <b>Bit 7</b> 1: 8-Lane supported 0: Not supported

#### 7.4.5 CSI C-PHY Lane Mode

The **CSI\_CPHY\_lane\_mode\_capability** register indicates which Data Lane usages the image sensor supports when C-PHY is used.

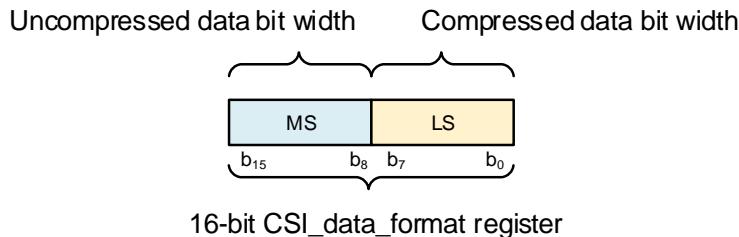
**Table 22 CSI C-PHY Lane Mode Capability Register**

Register Name	Type	RW	Comment
<b>CSI_CPHY_lane_mode_capability</b>	8-bit unsigned integer	RO	<p><b>Bit 0</b> 1: 1-Lane supported 0: Not supported</p> <p><b>Bit 1</b> 1: 2-Lane supported 0: Not supported</p> <p><b>Bit 2</b> 1: 3-Lane supported 0: Not supported</p> <p><b>Bit 3</b> 1: 4-Lane supported 0: Not supported</p> <p><b>Bit 4</b> 1: 5-Lane supported 0: Not supported</p> <p><b>Bit 5</b> 1: 6-Lane supported 0: Not supported</p> <p><b>Bit 6</b> 1: 7-Lane supported 0: Not supported</p> <p><b>Bit 7</b> 1: 8-Lane supported 0: Not supported</p>

## 7.5 Data Format, Data Type, and Virtual Channel Control

### 7.5.1 CSI Data Format

The **CSI\_data\_format** register controls the output data format and the level of compression used.



**Figure 12 CSI Data Format Register**

The MS byte of the **CSI\_data\_format** register contains the bit width of the uncompressed pixel data.

The LS byte of the **CSI\_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 and LS byte = 8: Top 8 bits of internal pixel data are transmitted as RAW8
- MS byte = 10 and LS byte = 8: 10-bit pixel data is compressed to 8-bits and transmitted as RAW8 (via user defined data type)
- MS byte = 10 and LS byte = 10: Top 10 bits of internal pixel data are transmitted as RAW10

**Table 23 CSI Data Format Register**

Register Name	Type	RW	Comment
<b>CSI_data_format</b>	16-bit unsigned integer	RW	<p><b>0x0808:</b> RAW8 (top 8-bits of internal data)  <b>0x0A06:</b> 10-b to 6-b compression  <b>0x0A07:</b> 10-b to 7-b compression  <b>0x0A08:</b> 10-b to 8-b compression  <b>0x0A0A:</b> RAW10 (top 10 bits of internal data)  <b>0x0C06:</b> 12-b to 6-b compression  <b>0x0C07:</b> 12-b to 7-b compression  <b>0x0C08:</b> 12-b to 8-b compression  <b>0x0C0A:</b> 12-b to 10-b compression  <b>0x0C0C:</b> RAW12 (top 12 bits of internal data)  <b>0x0E0E:</b> RAW14 (top 14 bits of internal data)  <b>0x1010:</b> RAW16 (top 16 bits of internal data)  <b>0x1414:</b> RAW20 (top 20 bits of internal data)  <b>0x1818:</b> RAW24 (top 24 bits of internal data)  <b>Default:</b> 0xA0A if supported by the image sensor. Otherwise the default value is vendor-specific standard RAW format.</p>

All data format mode changes shall be performed while in SW Standby mode.

#### Note:

*Writing to the **CSI\_data\_format** register shall also select the appropriate CSI-2 mode RAW data type values, as defined in the CSI-2 Specification. Example: When the Host selects RAW8 (0x0808 in the **CSI\_data\_format** register), the image sensor automatically uses data type 0x2A.*

### 7.5.2 DPCM Frame Data Type

When compressed data formats are used with CSI-2, a data type must be defined. This should be controlled through the register **DPCM\_frame\_DT**. The Host can determine whether this register is supported by inspecting bit 0 of the **data\_type\_capability** register. It is the Host's responsibility to program a suitable value. The image sensor shall not restrict the used values. Typical values might be 0x30 through 0x37.

**Table 24 CSI Compression DT Register**

Register Name	Type	RW	Comment
<b>DPCM_Frame_DT</b>	8-bit unsigned integer	RW	<b>Default:</b> 0x30

All data type changes shall be performed while in SW Standby mode.

### 7.5.3 CSI Channel Identifier

The **CSI\_channel\_identifier** register is used to distinguish different data channels from each other. It will select the virtual channel for top embedded data and visible pixel data. See the CSI-2 Specification [\[MIP101\]](#) for the definition of Virtual Channel identifier.

For normal virtual channel control, the valid range shall be from 0 to 3. The Host can determine whether the image sensor supports the extended virtual channel range by inspecting bit 3 of the **data\_type\_capability** register. When the extended virtual channel range is supported, the valid range for shall be from 0 to 15 for D-PHY, and from 0 to 31 for C-PHY. It is the responsibility of the Host to ensure that the register is programmed to a value within the valid range.

The default value for the **CSI\_channel\_identifier** register is 0x00.

**Table 25 CSI Channel Identifier Register**

Register Name	Type	RW	Comment
<b>CSI_channel_identifier</b>	8-bit unsigned integer	RW	<b>Valid Range:</b> 0-3, or 0-15, or 0-31 <b>Default:</b> 0

All virtual channel changes shall be performed while in SW Standby mode.

### 7.5.4 Bottom Embedded Data Control

The image sensor may support bottom embedded data. The data type and virtual channel of bottom embedded data may be programmable. Bottom embedded data may be used with PDAF, see [Section 17.3](#).

**Table 26 Bottom Embedded Data Control Registers**

Register Name	Type	RW	Comment
<b>bottom_embedded_data_DT</b>	8-bit unsigned integer	RW	Programmable data type for bottom embedded data. Image sensor shall not set limits to the programmable values. Typical values may be values between 0x30-0x37 and 0x12 <b>Default:</b> 35h
<b>bottom_embedded_data_VC</b>	8-bit unsigned integer	RW	Programmable virtual channel for bottom embedded data. <b>Default:</b> 00h

All data type and virtual channel changes shall be performed while in SW Standby mode.

### 7.5.5 Data Type and Virtual Channel Programmability

Image sensor capabilities related to data type and virtual channel programmability are defined in the **data\_type\_capability** register.

**Table 27 Data Type Capability Register**

Register Name	Type	RW	Comment
<b>data_type_capability</b>	8-bit unsigned integer	RO	<b>Bit 0</b> 1: DPCM data type is programmable <b>Bit 1</b> 1: Bottom embedded data type is programmable <b>Bit 2</b> 1: Bottom embedded data virtual channel is programmable <b>Bit 3</b> 1: Extended virtual channel range is supported

## 7.6 ADC Controls

The image sensor may support changeable ADC bit depth.

Image sensor capabilities related to ADC capabilities are defined in the register **ADC\_capability**.

The bit depth is controllable via the register **ADC\_bit\_depth**. The default value shall be the nominal ADC bit depth.

The image sensor may support register **ADC\_bit\_depth\_capability**, which specifies which bit depths the ADC supports. The purpose of this register is to identify which ADC bit depth(s) the image sensor supports. The **ADC\_bit\_depth\_capability** register uses bit mask coding, meaning that each bit indicates whether a particular bit depth is supported.

**Table 28 ADC Control and Capability Registers**

Register Name	Type	RW	Comment
<b>ADC_capability</b>	8-bit unsigned integer	RO	<b>Bit 0</b> 1: Supports controllable ADC bit depth <b>Bit 1</b> 1: Supports register <b>ADC_bit_depth_capability</b>
<b>ADC_bit_depth_capability</b>	32-bit unsigned integer	RO	For each bit from 0 to 31: <b>Bit <i>n</i></b> 1: Supports ADC bit depth <i>n</i> + 1 <b>Example:</b> If Bit[7] = 1, then 8-bit ADC is supported.
<b>ADC_bit_depth</b>	8-bit unsigned integer	RW	0: Not controlled by Host Other values: ADC bit depth <b>Default:</b> Nominal ADC bit depth, e.g. 10-bit ADC has value 0x0A.

## 7.7 CSI-2 v2.0 Feature Controls

CSI-2 v2.0 introduces a few new features, which require control registers. This Section describes these features from the CCS perspective. For details, refer to the version of the CSI-2 Specification that the image sensor supports.

### Note:

*It is recommended to follow CSI-2 v2.1 Specification [MIPI14] or newer. Compared to earlier versions, v2.1 clarified certain features described in this Section.*

### 7.7.1 CSI-2 LRTE Feature

CSI-2 v2.0 introduced the Latency Reduction and Transport Efficiency (LRTE) feature, and CSI-2 v3.0 adds additional LRTE capabilities. If an image sensor supports the LRTE feature, then it shall support the control registers defined in this Section.

The Host can detect whether an image sensor supports LRTE for C-PHY by inspecting the register **LRTE\_cphy\_capability**. The Host can detect whether an image sensor supports LRTE for D-PHY by inspecting the register **LRTE\_dphy\_capability**.

- For C-PHY, the LRTE feature consists of short and long packet PDQs, and short and long packet spacers (see [MIPI12] for definitions). The registers **TX\_REG\_CSI\_EPD\_EN\_SSP\_cphy**, **TX\_REG\_CSI\_EPD\_OP\_SLP\_cphy**, and **TX\_REG\_CSI\_EPD\_MISC\_OPTIONS\_cphy** are used to control LRTE.
- For D-PHY, there are two LRTE options (see [MIPI12] for details):
  - **PHY-generated LRTE (Option 1):** The LRTE feature consists of short and long packet PDQs, and short and long packet spacers.
  - **Protocol-generated LRTE (Option 2):** The LRTE feature consists of short and long packet spacers.

The registers **TX\_REG\_CSI\_EPD\_EN\_SSP\_dphy**, **TX\_REG\_CSI\_EPD\_OP\_SLP\_dphy**, and **TX\_REG\_CSI\_EPD\_MISC\_OPTIONS\_dphy** are used to control LRTE for D-PHY.

The Host shall ensure that the LRTE control registers are programmed correctly in SW Standby Mode before starting streaming. The LRTE control registers shall only be programmed while SW Standby Mode.

Please refer to the CSI-2 Specification [MIPI12] for how to program the registers **TX\_REG\_CSI\_EPD\_EN\_SSP**, **TX\_REG\_CSI\_EPD\_OP\_SLP** and **TX\_REG\_CSI\_EPD\_MISC\_OPTIONS**.

**Table 29 LRTE Registers**

Register Name	Type	RW	Comment
<b>LRTE_cphy_capability</b>	8-bit unsigned integer	RO	<p><b>Bit 0</b> 1: Supports PDQ (short) for C-PHY</p> <p><b>Bit 1</b> 1: Supports spacer (short) for C-PHY</p> <p><b>Bit 2</b> 1: Supports PDQ (long) for C-PHY</p> <p><b>Bit 3</b> 1: Supports spacer (long) for C-PHY</p> <p><b>Bit 4</b> 1: Supports spacers without PDQ for C-PHY. Common for both Short and Long Packets.</p> <p><b>Other Bits:</b> Reserved for future use</p>
<b>LRTE_dphy_capability</b>	8-bit unsigned integer	RO	<p><b>Bit 0</b> 1: Supports PDQ (short) for D-PHY option 1</p> <p><b>Bit 1</b> 1: Supports spacer (short) for D-PHY option 1</p> <p><b>Bit 2</b> 1: Supports PDQ (long) for D-PHY option 1</p> <p><b>Bit 3</b> 1: Supports spacer (long) for D-PHY option 1</p> <p><b>Bit 4</b> 1: Supports spacer (short) for D-PHY option 2</p> <p><b>Bit 5</b> 1: Supports spacer (long) for D-PHY option 2</p> <p><b>Bit 6</b> 1: Supports spacers without PDQ for D-PHY EPD Option 1. Common for both Short and Long packets.</p> <p><b>Bit 7</b> 1: Supports spacers with variable length for D-PHY EPD Option 2. Common for both short and long packets.</p>

Register Name	Type	RW	Comment
<b>CSI2_interface_capability_misc</b>	8-bit unsigned integer	RO	<b>Bit 0</b> 1: Supports EoTp Short Packets for D-PHY EPD Option 2. <b>Other Bits:</b> Reserved for future use
<b>TX_REG_CSI_EPD_EN_SSP_cphy</b>	16-bit unsigned integer	RW	Short Packet control register for C-PHY <b>Default:</b> Image-sensor-specific
<b>TX_REG_CSI_EPD_OP_SLP_cphy</b>	16-bit unsigned integer	RW	Long Packet control register for C-PHY <b>Default:</b> Image-sensor-specific
<b>TX_REG_CSI_EPD_MISC_OPTIONS_cphy</b>	8-bit unsigned integer	RW	Control register for C-PHY <b>Default:</b> 0
<b>TX_REG_CSI_EPD_EN_SSP_dphy</b>	16-bit unsigned integer	RW	Short Packet control register for D-PHY <b>Default:</b> Image-sensor-specific
<b>TX_REG_CSI_EPD_OP_SLP_dphy</b>	16-bit unsigned integer	RW	Long Packet control register for D-PHY <b>Default:</b> Image-sensor-specific
<b>TX_REG_CSI_EPD_MISC_OPTIONS_dphy</b>	8-bit unsigned integer	RW	Control register for D-PHY <b>Default:</b> 0

### 7.7.2 CSI-2 ALPS Feature

CSI-2 v2.0 [[MIPI06](#)] introduces the Alternate Low Power State (ALPS) feature. An image sensor may support the ALPS feature for C-PHY and/or for D-PHY:

- For D-PHY, LVLP is used for the ALPS feature. The Host can detect whether LVLP is supported by inspecting register **ALPS\_dphy\_capability**.
- For C-PHY, LVLP and/or ALP Mode is used for the ALPS feature. The Host can detect the supported features by inspecting register **ALPS\_cphy\_capability**.

If the image sensor supports LVLP, then it may support either controllable LVLP or non-controllable LVLP.

If the image sensor supports the ALP Mode and/or controllable LVLP, then it shall support the control registers defined in this Section.

The Host shall ensure that both the receiver and the transmitter use the same signaling. If the image sensor supports controllable ALPS (i.e., if the image sensor supports both LVLP and legacy LPS, and/or if the image sensor supports ALP Mode), then the Host shall configure the correct mode in the Link (i.e. Transmitter and/or Receiver) during the first power-up in SW Standby Mode.

**Table 30 ALPS Registers**

Register Name	Type	RW	Comment
<b>ALPS_dphy_capability</b>	8-bit unsigned integer	RO	<b>Bits 0–1</b> 0: LVLP not supported 1: LVLP supported 2: Controllable LVLP supported
<b>ALPS_cphy_capability</b>	8-bit unsigned integer	RO	<b>Bits 0–1</b> 0: LVLP not supported 1: LVLP supported 2: Controllable LVLP supported <b>Bits 2–3</b> 0: ALP Mode not supported 1: ALP Mode supported 2: Controllable ALP Mode supported
<b>ALPS_ctrl</b>	8-bit unsigned integer	RW	<b>Bit 0</b> 0: LVLP disabled for D-PHY 1: LVLP enabled for D-PHY <b>Bit 1</b> 0: LVLP disabled for C-PHY 1: LVLP enabled for C-PHY <b>Bit 2</b> 0: ALP Mode disabled for C-PHY 1: ALP Mode enabled for C-PHY <b>Other Bits</b> Reserved for future use <b>Note:</b> <i>This register is only supported if the image sensor supports one or more controllable ALPS features.</i> <b>Default:</b> Image-sensor-specific

### 7.7.3 CSI-2 Data Scrambling Feature

CSI-2 v2.0 introduces the Data Scrambling feature, in which data is randomized on a per-Lane basis (see the MIPI CSI-2 v2.0 Specification [[MIPI06](#)] or later for details). If an image sensor supports the Data Scrambling feature, it shall support the control registers defined in this Section.

The Host shall ensure that both the receiver and the transmitter use same method, i.e., that data scrambling is either off or on, and if on, that 4-seed operation mode is only used if both transmitter and receiver support it. The Host shall ensure that the data scrambling control registers and optional seed value programming registers are programmed correctly in SW Standby Mode before starting streaming. The data scrambling control registers and the seed value programming registers shall only be programmed during SW Standby Mode. For valid data scrambling seed values, see the CSI-2 v2.0 Specification [[MIPI06](#)] or later. In 1-seed operation mode, Seed 3 (i.e. the `lane_X_seed_value3` register[s]) shall be used, where X is the Lane number.

**Table 31 Data Scrambling Registers**

Register Name	Type	RW	Comment
<code>scrambling_capability</code>	8-bit unsigned integer	RO	<b>Bit 0</b> 1: Data scrambling supported <b>Bits 1-2</b> Maximum number of seeds per Lane in scrambling with C-PHY. Valid if scrambling is supported. 0: 1 seed 1: Reserved 2: Reserved 3: 4 seeds <b>Bit 3-5</b> Number of scrambling seed value registers per lane with C-PHY 0: No seed value registers 1: 1 seed value register (Seed 3) 4: 4 seed value registers Other values: Reserved <b>Bit 6</b> Number of scrambling seed value registers per lane with D-PHY 0: No seed value registers 1: 1 seed value register (Seed 3) <b>Other Bits</b> Reserved for future use

Register Name	Type	RW	Comment
<b>scrambling_ctrl</b>	8-bit unsigned integer	RW	<p><b>Bit 0</b> 0: Scrambling disabled 1: Scrambling enabled</p> <p><b>Bits 1-2</b> Valid with C-PHY (D-PHY uses single seed) 0: Use only 1 seed (Seed 3) 1: Reserved 2: Reserved 3: Use 4 seeds</p> <p><b>Other Bits</b> Reserved for future use</p> <p><b>Default:</b> Image-Sensor-specific</p>
<b>lane_1_seed_value0</b>	16-bit unsigned integer	RW	Seed value for seed index 0 for Lane 1 <b>Default:</b> See CSI-2 Specification
<b>lane_2_seed_value0</b>	16-bit unsigned integer	RW	Seed value for seed index 0 for Lane 2 <b>Default:</b> See CSI-2 Specification
<b>lane_3_seed_value0</b>	16-bit unsigned integer	RW	Seed value for seed index 0 for Lane 3 <b>Default:</b> See CSI-2 Specification
<b>lane_4_seed_value0</b>	16-bit unsigned integer	RW	Seed value for seed index 0 for Lane 4 <b>Default:</b> See CSI-2 Specification
<b>lane_5_seed_value0</b>	16-bit unsigned integer	RW	Seed value for seed index 0 for Lane 5 <b>Default:</b> See CSI-2 Specification
<b>lane_6_seed_value0</b>	16-bit unsigned integer	RW	Seed value for seed index 0 for Lane 6 <b>Default:</b> See CSI-2 Specification
<b>lane_7_seed_value0</b>	16-bit unsigned integer	RW	Seed value for seed index 0 for Lane 7 <b>Default:</b> See CSI-2 Specification
<b>lane_8_seed_value0</b>	16-bit unsigned integer	RW	Seed value for seed index 0 for Lane 8 <b>Default:</b> See CSI-2 Specification
<b>lane_1_seed_value1</b>	16-bit unsigned integer	RW	Seed value for seed index 1 for Lane 1 <b>Default:</b> See CSI-2 Specification
<b>lane_2_seed_value1</b>	16-bit unsigned integer	RW	Seed value for seed index 1 for Lane 2 <b>Default:</b> See CSI-2 Specification
<b>lane_3_seed_value1</b>	16-bit unsigned integer	RW	Seed value for seed index 1 for Lane 3 <b>Default:</b> See CSI-2 Specification

Register Name	Type	RW	Comment
<b>lane_4_seed_value1</b>	16-bit unsigned integer	RW	Seed value for seed index 1 for Lane 4 <b>Default:</b> See CSI-2 Specification
<b>lane_5_seed_value1</b>	16-bit unsigned integer	RW	Seed value for seed index 1 for Lane 5 <b>Default:</b> See CSI-2 Specification
<b>lane_6_seed_value1</b>	16-bit unsigned integer	RW	Seed value for seed index 1 for Lane 6 <b>Default:</b> See CSI-2 Specification
<b>lane_7_seed_value1</b>	16-bit unsigned integer	RW	Seed value for seed index 1 for Lane 7 <b>Default:</b> See CSI-2 Specification
<b>lane_8_seed_value1</b>	16-bit unsigned integer	RW	Seed value for seed index 1 for Lane 8 <b>Default:</b> See CSI-2 Specification
<b>lane_1_seed_value2</b>	16-bit unsigned integer	RW	Seed value for seed index 2 for Lane 1 <b>Default:</b> See CSI-2 Specification
<b>lane_2_seed_value2</b>	16-bit unsigned integer	RW	Seed value for seed index 2 for Lane 2 <b>Default:</b> See CSI-2 Specification
<b>lane_3_seed_value2</b>	16-bit unsigned integer	RW	Seed value for seed index 2 for Lane 3 <b>Default:</b> See CSI-2 Specification
<b>lane_4_seed_value2</b>	16-bit unsigned integer	RW	Seed value for seed index 2 for Lane 4 <b>Default:</b> See CSI-2 Specification
<b>lane_5_seed_value2</b>	16-bit unsigned integer	RW	Seed value for seed index 2 for Lane 5 <b>Default:</b> See CSI-2 Specification
<b>lane_6_seed_value2</b>	16-bit unsigned integer	RW	Seed value for seed index 2 for Lane 6 <b>Default:</b> See CSI-2 Specification
<b>lane_7_seed_value2</b>	16-bit unsigned integer	RW	Seed value for seed index 2 for Lane 7 <b>Default:</b> See CSI-2 Specification
<b>lane_8_seed_value2</b>	16-bit unsigned integer	RW	Seed value for seed index 2 for Lane 8 <b>Default:</b> See CSI-2 Specification
<b>lane_1_seed_value3</b>	16-bit unsigned integer	RW	Seed value for seed index 3 for Lane 1 <b>Default:</b> See CSI-2 Specification
<b>lane_2_seed_value3</b>	16-bit unsigned integer	RW	Seed value for seed index 3 for Lane 2 <b>Default:</b> See CSI-2 Specification
<b>lane_3_seed_value3</b>	16-bit unsigned integer	RW	Seed value for seed index 3 for Lane 3 <b>Default:</b> See CSI-2 Specification

Register Name	Type	RW	Comment
lane_4_seed_value3	16-bit unsigned integer	RW	Seed value for seed index 3 for Lane 4 <b>Default:</b> See CSI-2 Specification
lane_5_seed_value3	16-bit unsigned integer	RW	Seed value for seed index 3 for Lane 5 <b>Default:</b> See CSI-2 Specification
lane_6_seed_value3	16-bit unsigned integer	RW	Seed value for seed index 3 for Lane 6 <b>Default:</b> See CSI-2 Specification
lane_7_seed_value3	16-bit unsigned integer	RW	Seed value for seed index 3 for Lane 7 <b>Default:</b> See CSI-2 Specification
lane_8_seed_value3	16-bit unsigned integer	RW	Seed value for seed index 3 for Lane 8 <b>Default:</b> See CSI-2 Specification

## 7.8 PHY Calibration Controls

### 7.8.1 D-PHY Skew Calibration

D-PHY v1.2 and later requires the transmitter to send a calibration sequence if speeds higher than 1.5Gbit/s Lane are used. See the MIPI D-PHY Specification v1.2 or later for details regarding the calibration sequence. If the image sensor supports a Lane speed higher than 1.5Gbit/s, then it shall support the CCS register **PHY\_init\_calibration\_ctrl**, in order to control the initial calibration sequence.

The Host shall enable the initial skew calibration sequence:

- Whenever the CSI-2 Lane speed changes from a previous streaming state, to a new Lane speed higher than 1.5Gbit/s, and
- If the CSI-2 Lane speed for the first streaming after power-up is higher than 1.5Gbit/s

The Host shall enable the calibration sequence only while the image sensor is in SW Standby Mode. When the calibration sequence feature is enabled, the image sensor shall send the calibration sequence when streaming is started, before HS transmission.

When changing the Lane speed from a speed higher than 1.5Gbit/s to a speed of 1.5Gbit/s or less, the Host may disable the calibration sequence while in SW Standby Mode (see the D-PHY specification for details).

In addition, the image sensor may optionally support periodically sending the calibration sequence during streaming. The Host can use the registers **PHY\_periodic\_calibration\_ctrl** and **PHY\_periodic\_calibration\_interval** to control whether the image sensor performs such periodic calibration. Those registers shall only be used during SW Standby Mode.

Image sensor capabilities related to D-PHY calibration are defined in the register **DPHY\_calibration\_capability**.

### 7.8.2 D-PHY Alternate Calibration

Starting in D-PHY v2.1, the transmitter shall send, in addition to the initial skew calibration, an Alternate Calibration sequence if speeds higher than 2.5Gbit/s per Lane are used and if the receiver supports and requires it. See [\[MIP108\]](#) for details regarding the alternate calibration sequence. If the image sensor supports a Lane speed higher than 2.5Gbit/s, then it should support the CCS registers **PHY\_init\_calibration\_ctrl** and **DPHY\_calibration\_mode** in order to control the Alternate Calibration Sequence.

The Host shall enable the calibration sequences only while the image sensor is in SW Standby Mode. When the calibration sequence feature is enabled, the image sensor shall send the calibration sequences (i.e. both the initial skew calibration sequence and the alternate calibration sequence per [\[MIP108\]](#)) when streaming is started, before HS transmission.

When changing the Lane speed from a speed higher than 2.5Gbit/s to one of 2.5Gbit/s or less, the Host may disable the Alternate Calibration Sequence while in SW Standby Mode (see [\[MIP108\]](#)).

Image sensor capabilities related to D-PHY calibration are defined in the register **DPHY\_calibration\_capability**.

### 7.8.3 C-PHY Calibration Controls

Starting in C-PHY v1.2, the transmitter shall support a calibration sequence if speeds higher than 3.0 Gsps per Lane are supported. Calibration may be used at 3.0 Gsps or lower if supported by both the Transmitter and the Receiver. See [*MIPI10*] for details regarding the calibration sequence. If the image sensor supports a Lane speed higher than 3.0Gsps, then it should support the CCS registers **PHY\_init\_calibration\_ctrl** and **CPHY\_calibration\_mode** in order to control the initial calibration sequence. The calibration shall be performed simultaneously on all Lanes of a Link.

The Host should enable the initial calibration sequence:

- Whenever the CSI-2 Lane speed changes from a previous Streaming state to a new Lane speed higher than 3.0 Gsps, and
- If the CSI-2 Lane speed for the first streaming after power-up is higher than 3.0 Gsps

In addition, the Host might need to enable the initial calibration sequence whenever the CSI-2 Lane speed is increased after a previous calibration.

The Host shall enable the calibration sequence only while the image sensor is in SW Standby Mode. When the calibration sequence feature is enabled, the image sensor shall send the calibration sequence when streaming is started, before HS transmission.

When changing the Lane speed from a speed higher than 3.0 Gsps to one of 3.0 Gsps or less, the Host may disable the calibration sequence while in SW Standby Mode per [*MIPI10*].

In addition, the image sensor may optionally support periodically sending the calibration sequence during streaming. The Host can use the registers **PHY\_periodic\_calibration\_ctrl** and **PHY\_periodic\_calibration\_interval** to control whether the image sensor performs such periodic calibration. These registers shall only be used during SW Standby Mode.

The Host shall ensure that both the receiver and the transmitter use the same method, i.e. that the same calibration format is used and that a sufficiently long calibration sequence length is used. The calibration format should be selected by the register **CPHY\_calibration\_mode**, and calibration sequence lengths for different formats should be programmed via registers **t3\_calpreamble\_length**, **t3\_calaltseq\_length**, and **t3\_caludefseq\_length** for initial calibration, and via registers **t3\_calpreamble\_length\_per**, **t3\_calaltseq\_length\_per**, and **t3\_caludefseq\_length\_per** for periodic calibration. Register **FM2\_init\_seed** should be used for selecting the seed value for all Lanes in the Link when Format 2 is used. All these registers shall only be used during SW Standby Mode. The minimum length required by the receiver may be higher than the length provided by the minimum value of the control register.

Image sensor capabilities related to C-PHY calibration are defined in the register **CPHY\_calibration\_capability**.

#### 7.8.4 PHY Calibration Registers

949

Table 32 PHY Calibration Capability Registers

Register Name	Type	RW	Comment
<b>DPHY_calibration_capability</b>	8-bit unsigned integer	RO	<b>Bit 0</b> 1: Supports manual calibration sequence in start of streaming (required for D-PHY 1.2 or newer) <b>Bit 1</b> 1: Supports manual calibration sequence during streaming (optional) <b>Bit 2</b> 1: Supports CCS controls for alternate calibration sequence in start of streaming (required for D-PHY 2.1) <b>Other Bits</b> Reserved for future use
<b>CPHY_calibration_capability</b>	8-bit unsigned integer	RO	<b>Bit 0</b> 1: Supports manual calibration sequence in start of streaming <b>Bit 1</b> 1: Supports manual calibration sequence during streaming (optional) <b>Bit 2</b> 1: Supports format 1 CCS controls <b>Bit 3</b> 1: Supports format 2 CCS controls <b>Bit 4</b> 1: Supports format 3 CCS controls (optional) <b>Other Bits</b> Reserved for future use

950

**Table 33 PHY Calibration Registers**

Register Name	Type	RW	Comment
<b>PHY_init_calibration_ctrl</b>	8-bit unsigned integer	RW	0: Do not send calibration sequence 1: Send calibration sequence during start of streaming <b>Default:</b> 0
<b>PHY_periodic_calibration_ctrl</b>	8-bit unsigned integer	RW	0: Do not send calibration sequence 1: Send calibration sequence during frame blanking <b>Default:</b> 0
<b>PHY_periodic_calibration_interval</b>	8-bit unsigned integer	RW	Output frame interval of period skew calibration 0: No output 1: Output with all frames A: Output once by A frames <b>Example:</b> 2 = every other frame <b>Default:</b> 0
<b>DPHY_calibration_mode</b>	8-bit unsigned integer	RW	Calibration selection for initialization calibration 0: Only Basic Calibration Sequence 1: Basic and Alternate Calibration Sequence <b>Default:</b> 0

951

**Table 34 Additional C-PHY Calibration Control Registers**

Register Name	Type	RW	Comment
<b>CPHY_calibration_mode</b>	8-bit unsigned integer	RW	Calibration preamble selection 0: Format 1 1: Format 2 2: Format 3 (optional) <b>Default:</b> Sensor specific
<b>t3_calpreamble_length</b>	8-bit unsigned integer	RW	Used with format 1, format 2, and format 3 in initial calibration. The length of t3-CALPREAMBLE is controlled in steps of 7 UI, between 7 UIs and 1792 UIs. Valid register value: 0 to 255 Length = ( register value + 1 ) * 7 <b>Default:</b> Sensor specific
<b>t3_calpreamble_length_per</b>	8-bit unsigned integer	RW	Used with format 1, format 2, and format 3 in periodic calibration. The length of t3-CALPREAMBLE is controlled in steps of 7 UI, between 7 UIs and 1792 UIs. Valid register value: 0 to 255 Length = ( register value + 1 ) * 7 <b>Default:</b> Sensor specific
<b>t3_calaltseq_length</b>	8-bit unsigned integer	RW	Used with format 2 in initial calibration. The length of t3-CALALTSEQ is controlled in steps of 7 UI, between 7 UIs and 1792 UIs. Valid register value: 0 to 255 Length = ( register value + 1 ) * 7 <b>Default:</b> Sensor specific
<b>t3_calaltseq_length_per</b>	8-bit unsigned integer	RW	Used with format 2 in periodic calibration. The length of t3-CALALTSEQ is controlled in steps of 7 UI, between 7 UIs and 1792 UIs. Valid register value: 0 to 255 Length = ( register value + 1 ) * 7 <b>Default:</b> Sensor specific
<b>FM2_init_seed</b>	16-bit unsigned integer	RW	Initial seed value for all Lanes used with format 2 <b>Default:</b> Sensor specific
<b>t3_caludefseq_length</b>	16-bit unsigned integer	RW	Used with format 3 in initial calibration. The length of t3-CALUDEFSEQ is controlled in steps of 7 UI, between 7 UIs and 14336 UIs. Valid register value: 0 to 2047 Length = ( register value + 1 ) * 7 <b>Default:</b> Sensor specific
<b>t3_caludefseq_length_per</b>	16-bit unsigned integer	RW	Used with format 3 in periodic calibration. The length of t3-CALUDEFSEQ is controlled in steps of 7 UI, between 7 UIs and 14336 UIs. Valid register value: 0 to 2047 Length = ( register value + 1 ) * 7 <b>Default:</b> Sensor specific

## 7.9 PHY Control

This Section describes the following PHY features:

- PHY Timing Controls for D-PHY and C-PHY
- Equalization for D-PHY and C-PHY
- Preamble for D-PHY
- Spread Spectrum Clocking for D-PHY

### 7.9.1 PHY Timing Controls

The image sensor PHY interface can support different level of controls for PHY timings. Some image sensors might be almost self-controlling.

CCS supports three levels of PHY timing controls:

1. Fully automatic
2. Only UI-information-based control
3. Manual register controls for different parameter sets

The Host shall change PHY register settings only while in SW Standby Mode. Registers related to PHY Timing Control are defined in this Section, *Section 7.9.2*, and *Section 7.9.2.5*.

965

**Table 35 PHY Control Capability Register**

Register Name	Type	RW	Comment
<b>PHY_ctrl_capability</b>	8-bit unsigned integer	RO	<p><b>Bit 0</b> 0: Not supported 1: Automatic PHY control supported</p> <p><b>Bit 1</b> 0: Not supported 1: UI-based, non-manual PHY control supported</p> <p><b>Bit 2</b> 0: Not supported 1: D-PHY control by the non-extended “Time and UI Values Register 1” is supported</p> <p><b>Bit 3</b> 0: Not supported 1: D-PHY control by the non-extended “Time and UI Values Register 2” is supported</p> <p><b>Bit 4</b> 0: Not supported 1: D-PHY control by the non-extended “Time values” register is supported</p> <p><b>Bit 5</b> 0: Not supported 1: D-PHY control by the registers for extended “Time and UI Values Register 1” is supported</p> <p><b>Bit 6</b> 0: Not supported 1: D-PHY control by the registers for extended “Time and UI Values Register 2” is supported</p> <p><b>Bit 7</b> 0: Not supported 1: D-PHY control by the registers for extended “Time value” is supported</p>
Note that in register <b>PHY_ctrl_capability</b> :			<ul style="list-style-type: none"> <li>• Only one of bits 2 or 5 may be set to 1</li> <li>• Only one of bits 3 or 6 may be set to 1</li> <li>• Only one of bits 4 or 7 may be set to 1</li> </ul>

Register Name	Type	RW	Comment
<b>PHY_ctrl_capability_2</b>	8-bit unsigned integer	RO	<p><b>Bit 0</b> 0: Not supported 1: <b>TGR_Preamble_Length</b> and <b>TGR_Post_Length</b> is supported</p> <p><b>Bit 1</b> 0: Not supported 1: <b>TGR_Preamble_Prog_Sequence_n</b>, <b>n+1</b> is supported</p> <p><b>Bit 2</b> 0: Not supported 1: Additional manual C-PHY timing parameter is supported</p> <p><b>Bit 3</b> 0: UI clock based manual control for both C-PHY and D-PHY 1: Generic clock based manual control for both C-PHY and D-PHY</p> <p><b>Bit 4</b> 0: UI clock based manual control for D-PHY 1: Generic clock based manual control for D-PHY</p> <p><b>Bit 5</b> 0: UI clock based manual control for C-PHY 1: Generic clock based manual control for C-PHY</p> <p><b>Bit 6</b> 0: Not supported 1: Manual LP control supported for D-PHY (see <b>Section 7.9.7</b>)</p> <p><b>Bit 7</b> 0: Not supported 1: Manual LP control supported for C-PHY (see <b>Section 7.9.7</b>)</p>

Register Name	Type	RW	Comment
<b>PHY_ctrl_capability_3</b>	8-bit unsigned integer	RO	<p><b>Bit 0</b> 0: D-PHY timing register values are multiples of bits-per-pixel or a constant 1: D-PHY timing register values are not multiples of bits-per-pixel or a constant</p> <p><b>Bit 1</b> 0: Min D-PHY timing register value is 0 1: Min D-PHY timing register value is 1</p> <p><b>Bit 2</b> 0: <b>t wakeup</b> timing register is not supported 1: <b>t wakeup</b> timing register is supported for both C-PHY and D-PHY</p> <p><b>Bit 3</b> 0: <b>t init</b> timing register is not supported 1: <b>t init</b> timing register is supported for both C-PHY and D-PHY</p> <p><b>Bit 4</b> 0: <b>ths_exit</b> and <b>ths_exit_ex</b> timing registers are not supported 1: <b>ths_exit</b> or <b>ths_exit_ex</b> timing register is supported for both C-PHY and D-PHY</p> <p><b>Bit 5</b> 0: C-PHY timing register values are multiples of a constant 1: C-PHY timing register values are not multiples of a constant</p> <p><b>Bit 6</b> 0: Min C-PHY timing register value is 0 1: Min C-PHY timing register value is 1</p> <p><b>Bit 7</b> Reserved</p>

966 The image sensor should support Automatic or UI-based non-manual PHY control. The same **PHY\_ctrl**  
 967 register is used to control both the D-PHY interface and the C-PHY interface.

968 The image sensor shall observe the following rule regarding the default value of register **PHY\_ctrl**:

- 969   • If the image sensor supports Automatic control, then the default value shall be 0,  
 970   • Otherwise, if the image sensor supports UI control, then the default value shall be 1,  
 971   • Otherwise, the default value shall be 2.

972              **Table 36 PHY Control Register**

Register Name	Type	RW	Comment
<b>PHY_ctrl</b>	8-bit unsigned integer	RW	<p>0: Use Automatic control 1: Use UI control 2: Use Manual Mode register control</p> <p><b>Default:</b> See text immediately above.</p>

### 7.9.1.1 Use Cases for D-PHY (Informative)

These examples clarify the procedure for setting the link while in SW Standby Mode, before the Host commands the image sensor to start streaming. Note that support for the register **tclk\_post** is required in all three use cases.

#### 976 Use Case 1: Automatic D-PHY Control

- 977 • The Host writes extclk frequency information to the image sensor
- 978 • The Host controls video timing normally, as per this Specification (e.g. sends the image sensor  
979 settings suitable for a 30 fps 2x2 binned image stream, or e.g. a 15 fps 5 Mpix image stream)
- 980 • The image sensor automatically calculates all D-PHY values. Because the image sensor has  
981 knowledge of the extclk frequency, the clock tree settings, and the number of data Lanes in use  
982 (though some image sensors may not require this information), it can calculate the correct internal  
983 timing parameters for D-PHY. In setting these registers, the image sensor is not permitted to  
984 populate the registers described in *Section 7.9.2*.

#### 985 Use Case 2: UI-Based D-PHY Control

- 986 • The Host writes extclk frequency information to the image sensor
- 987 • The Host controls video timing normally, as per this Specification (e.g. sends the image sensor  
988 settings suitable for a 30 fps 2x2 binned image stream, or e.g. a 15fps 5Mpix image stream)
- 989 • The Host writes link speed information to the register **requested\_link\_rate** (e.g. 800 Mbit/s)
- 990 • The image sensor calculates all D-PHY values. Because the image sensor has knowledge of the  
991 link speed and the number of used data Lanes in use (though some image sensors may not require  
992 this information), it can calculate the correct internal timing parameters for D-PHY. In setting  
993 these registers, the image sensor is not permitted to populate the registers described in *Section  
994 7.9.2*.

#### 995 Use Case 3: Manual D-PHY Control

- 996 • The Host writes extclk frequency information to the image sensor
- 997 • The Host controls video timing normally, as per this Specification (e.g. sends the image sensor  
998 settings suitable for a 30fps 2x2 binned video, or e.g. a 15fps 5Mpix image stream)
- 999 • The Host controls the manual D-PHY control registers described in *Section 7.9.2* to adjust D-PHY  
1000 timing parameters

### 7.9.1.2 UI-Based PHY Control

For non-manual, UI-Based PHY control, the Host shall write the requested value to the register **requested\_link\_rate**. The Host shall program the clock tree and the **requested\_link\_rate** register to operate the link with the same speed. A small difference between the link speed programmed via the clock tree registers and the link speed programmed via the **requested\_link\_rate** register is permitted if such difference is caused solely by different bit depths in the related registers.

**Table 37 CSI-2 Requested Link Rate**

Register Name	Type	RW	Comment
<b>requested_link_rate</b>	32-bit unsigned iReal	RW	Target bitrate for CSI-2 transmission <ul style="list-style-type: none"> <li>• For D-PHY this is the bitrate (Mbit/s)</li> <li>• For C-PHY this is the Symbol rate (Msym/s)</li> </ul> <b>Default:</b> Image-sensor-specific

### 7.9.2 D-PHY Timing Registers

For manual control there are two options based on Bit 3 and Bit 4 of register **PHY\_ctrl\_capability\_2**. If Bit 3 and Bit 4 have the same value, then any option can be used. If they have different values, then the Bit 4 option shall be used. As described below, there are also two additional options based on Bit 0 and Bit 1 of register **PHY\_ctrl\_capability\_3**.

In UI clock based manual control, D-PHY timing values are given by the following equation:

$$PHY\_time\_value = \frac{M * (register\_value + N) * (csi\_lane\_mode + 1)}{requested\_link\_rate}$$

**Note:**

For D-PHY, 'register\_value' refers to the CCS registers described in this Section: 'Time and UI Values Register 1', 'Time and UI Values Register 2', and 'Time Values'.

For the value M in the above equation, see **Section 7.9.2.1**. The value of N is 1 if Bit 1 of register **PHY\_ctrl\_capability\_3** is 0; otherwise, the value of N is 0.

In the generic clock based manual control (i.e. Bit 4 of register **PHY\_ctrl\_capability\_2** is 1), D-PHY timing values controlled by the registers **ths\_prepare**, **ths\_prepare\_ex**, **tclk\_prepare**, **tclk\_prepare\_ex**, **tlpx**, **tlpx\_ex**, **ths\_exit**, **ths\_exit\_ex**, **twakeup**, and **tinit** are given by the following equation. Other timing values are given by the above UI clock based manual control formula.

$$PHY\_time\_value = \frac{(register\_value + N)}{generic\_clock\_frequency}$$

In the above equation, **generic\_clock\_frequency** is defined by the image sensor datasheet, for example twice the D/C-PHY PPI escape clock frequency. The value of N above is the same as the value of N in the UI clock based manual control equation.

### 7.9.2.1 D-PHY Manual Constant

In the above UI clock based manual control equation, if bit 0 of register **PHY\_ctrl\_capability\_3** is 0, then the value of M is 1, and the value of ‘register\_value + N’ shall be either an integer multiple of a constant (i.e. must be evenly divisible by the constant) or an integer multiple of a value based on the output bit depth.

Otherwise, if bit 0 of register **PHY\_ctrl\_capability\_3** is 1, then the value of M is either a constant or a value based on the output bit depth, and the value of ‘register\_value + N’ is independent of M.

If bit 7 of register **dphy\_manual\_constant** is 0, then a constant shall be used; a typical value for this constant is 8. If the value of this constant is not 8, then the value shall be reported in bits 6:0 of capability register **dphy\_manual\_constant**; otherwise, if the value of this constant is 8, then the value in bits 6:0 shall be either 0 or 8.

If bit 7 of register **dphy\_manual\_constant** is 1, then a value based on the output bit depth shall be used which is either the output bit depth itself (if the value in register **op\_bits\_per\_lane** [*Table 87*] is either 0 or greater than or equal to the output bit depth), or the output bit depth divided by 2 (if the value in register **op\_bits\_per\_lane** is nonzero and less than the output bit depth). For example, if the value in register **op\_bits\_per\_lane** is 16, then the value of M or the divisor of ‘register\_value + N’ (whichever is appropriate) is 16 for RAW16 output pixels and 20/2 = 10 for RAW20 output pixels.

**Table 38 D-PHY Manual Constant Parameters**

Register Name	Type	RW	Comment
<b>dphy_manual_constant</b>	8-bit unsigned integer	RO	<b>Bits 6-0</b> Value of the constant used in D-PHY time value calculations <b>Bit 7</b> 0: Use above constant 1: Use value based on output bit depth

### 7.9.2.2 D-PHY Time and UI Values Register 1

In UI-Based and Automatic modes, if the Host writes the value 0x00 or 0xFF to the register **tclk\_post**, or writes the value 0x0000 or 0xFFFF to the register **tclk\_post\_ex**, then the effect of the **tclk\_post** register shall be suspended as the internal timing parameter is calculated by the sensor. All other **tclk\_post** and **tclk\_post\_ex** values shall have their normal effect in all modes.

This register has both an 8-bit version, and a 16-bit version with ‘\_ex’ appended to the same Register Name. The 16-bit version shall be implemented if the image sensor supports high speed D-PHY and bit 0 of register **PHY\_ctrl\_capability\_3** is 0. New designs having bit 0 of register **PHY\_ctrl\_capability\_3** set to 0 should support the 16-bit version.

Table 39 Parameters Containing Time and UI Values Register 1

Register Name	Type	RW	Comment
<b>tclk_post</b>	8-bit unsigned integer	RW	Time that the transmitter shall continue sending HS clock after the last associated Data Lane has transitioned to LP mode. Host shall use a suitable value. <b>Default:</b> Image-sensor-specific
<b>tclk_post_ex</b>	16-bit unsigned integer	RW	Time that the transmitter shall continue sending HS clock after the last associated Data Lane has transitioned to LP mode. Host shall use a suitable value. <b>Default:</b> Image-sensor-specific

### 7.9.2.3 D-PHY Time and UI Values Register 2

1051 Each of these registers has both an 8-bit version, and a 16-bit version with ‘\_ex’ appended to the same  
 1052 Register Name. The 16-bit version shall be implemented if the image sensor supports high speed D-PHY and  
 1053 bit 0 of register **PHY\_ctrl\_capability\_3** is 0. New designs having bit 0 of register **PHY\_ctrl\_capability\_3** set  
 1054 to 0 should support the 16-bit version.

1055 However, these registers are optional if all of the following are true:

- 1056 1. The D-PHY Specification is completely fulfilled  
 1057 2. Interoperability has been verified with small latencies, especially SOT and EOT times  
 1058 3. The image sensor uses Automatic or UI-Based non-manual control

1059 **Table 40 Parameters Containing Time and UI Values Register 2**

Register Name	Type	RW	Comment
<b>ths_prepare</b>	8-bit unsigned integer	RW	Time to drive LP-00 before starting the HS transmission on a Data Lane. <b>Default:</b> Image-sensor-specific
<b>ths_zero_min</b>	8-bit unsigned integer	RW	Time to send HS-0, i.e. turn on the line termination and drive the interconnect with the HS driver, prior to sending the SoT Sync sequence. <b>Default:</b> Image-sensor-specific
<b>ths_trail</b>	8-bit unsigned integer	RW	Time the transmitter must drive the flipped last data bit after sending the last payload data bit of a HS transmission burst. This time is required by the receiver to determine EoT. <b>Default:</b> Image-sensor-specific
<b>ths_prepare_ex</b>	16-bit unsigned integer	RW	Time to drive LP-00 before starting the HS transmission on a Data Lane. <b>Default:</b> Image-sensor-specific
<b>ths_zero_min_ex</b>	16-bit unsigned integer	RW	Time to send HS-0, i.e. turn on the line termination and drive the interconnect with the HS driver, prior to sending the SoT Sync sequence. <b>Default:</b> Image-sensor-specific
<b>ths_trail_ex</b>	16-bit unsigned integer	RW	Time the transmitter must drive the flipped last data bit after sending the last payload data bit of a HS transmission burst. This time is required by the receiver to determine EoT. <b>Default:</b> Image-sensor-specific

### 7.9.2.4 D-PHY Time Values

Each of these registers has both an 8-bit version, and a 16-bit version with ‘\_ex’ appended to the same Register Name. The 16-bit version shall be implemented if the image sensor supports high speed D-PHY and bit 0 of register **PHY\_ctrl\_capability\_3** is 0. New designs having bit 0 of register **PHY\_ctrl\_capability\_3** set to 0 should support the 16-bit version. Support for **ths\_exit** or **ths\_exit\_ex** is indicated if bit 4 of **PHY\_ctrl\_capability\_3** is 1.

However, these registers are optional if all of the following are true:

1. The D-PHY Specification is completely fulfilled
2. Interoperability has been verified with small latencies, especially SOT and EOT times
3. The image sensor uses Automatic or UI-Based non-manual control

**Table 41 Parameters Containing Only Time Values Register**

Register Name	Type	RW	Comment
<b>tclk_trail_min</b>	8-bit unsigned integer	RW	Time to drive HS differential state after last payload clock bit of a HS transmission burst <b>Default:</b> Image-sensor-specific
<b>tclk_prepare</b>	8-bit unsigned integer	RW	Time to drive LP-00 to prepare for HS clock transmission <b>Default:</b> Image-sensor-specific
<b>tclk_zero</b>	8-bit unsigned integer	RW	Time for lead HS-0 drive period before starting Clock. <b>Default:</b> Image-sensor-specific
<b>tclk_trail_min_ex</b>	16-bit unsigned integer	RW	Time to drive HS differential state after last payload clock bit of a HS transmission burst <b>Default:</b> Image-sensor-specific
<b>tclk_prepare_ex</b>	16-bit unsigned integer	RW	Time to drive LP-00 to prepare for HS clock transmission <b>Default:</b> Image-sensor-specific
<b>tclk_zero_ex</b>	16-bit unsigned integer	RW	Time for lead HS-0 drive period before starting Clock <b>Default:</b> Image- sensor-specific
<b>tlpx</b>	8-bit unsigned integer	RW	Length of any Low-Power state period <b>Default:</b> Image-sensor-specific
<b>tlpx_ex</b>	16-bit unsigned integer	RW	Length of any Low-Power state period <b>Default:</b> Image-sensor-specific
<b>ths_exit</b>	8-bit unsigned integer	RW	Time that the transmitter drives the Stop state following a high-speed burst <b>Default:</b> Image-sensor-specific <i>This register is shared with C-PHY.</i>
<b>ths_exit_ex</b>	16-bit unsigned integer	RW	Time that the transmitter drives the Stop state following a high-speed burst <b>Default:</b> Image-sensor-specific <i>This register is shared with C-PHY.</i>

### 7.9.2.5 D-PHY Timing Registers with Additional Scale Factors

Image sensor support for the registers shown in *Table 42* is indicated when bit 2 and/or bit 3 of register **PHY\_ctrl\_capability\_3** is set to 1 and D-PHY is the selected PHY. For D-PHY the time values corresponding to the contents of registers **t wakeup** and **t init** follow the equations described in *Section 7.9.2*, but require the application of the additional power-of-two scale factors stored in read-only register **dphy\_sf**; i.e. the time values corresponding to the 8-bit contents of **t wakeup** and **t init** are multiplied by  $2^{\wedge} \text{dphy\_sf}[3:0]$  and  $2^{\wedge} \text{dphy\_sf}[7:4]$ , respectively. The scale factors are intended to facilitate the programming of the relatively large time values corresponding to the PHY T<sub>WAKEUP</sub> and T<sub>INIT</sub> timing parameters, and shall be large enough to ensure that the minimum required values of these parameters can be satisfied.

**Table 42 Timing Registers with Scale Factors**

Register Name	Type	RW	Comment
<b>t wakeup</b>	8-bit unsigned integer	RW	Time that the PHY drives a Mark-1 state prior to a Stop state in order to initiate an exit from ULPS. <b>Default:</b> Image-sensor-specific <i>This register is shared with C-PHY.</i>
<b>t init</b>	8-bit unsigned integer	RW	Minimum time that the PHY drives the Stop state during PHY initialization prior to starting any high-speed transmission. <b>Default:</b> Image-sensor-specific <i>This register is shared with C-PHY.</i>
<b>dphy_sf</b>	8-bit unsigned integer	RO	<b>Bits 3-0</b> Power-of-two scale factor for <b>t wakeup</b> register values when D-PHY is selected and bit 2 of <b>PHY_ctrl_capability_3</b> is set to 1. <b>Bits 7-4</b> Power-of-two scale factor for <b>t init</b> register values when D-PHY is selected and bit 3 of <b>PHY_ctrl_capability_3</b> is set to 1. <b>Default:</b> Image-sensor-specific

**7.9.2.6 D-PHY Timing Register Limit Values**

The limit registers shown in *Table 43* enable image sensors to specify maximum allowed values for the various D-PHY timing registers defined in *Section 7.9.2*. The maximum value of each timing register is given by  $[2^{(L+1)}] - 1$ , where L is the corresponding 4-bit field value shown in *Table 43*. If L = 0, then no maximum value is specified other than that implied by the bit width of each timing register; however, note that each register may be programmed only with values supporting D-PHY timing within the min-max limits defined in the D-PHY specification.

**Table 43 D-PHY Timing Register Limits**

Register Name	Type	RW	Comment
dphy_limits_1	8-bit unsigned integer	RO	<b>Bits 3-0</b> Indicates max supported value of <b>ths_prepare</b> or <b>ths_prepare_ex</b> , whichever is appropriate. <b>Bits 7-4</b> Indicates max supported value of <b>ths_zero_min</b> or <b>ths_zero_min_ex</b> , whichever is appropriate.
dphy_limits_2	8-bit unsigned integer	RO	<b>Bits 3-0</b> Indicates max supported value of <b>ths_trail</b> or <b>ths_trail_ex</b> , whichever is appropriate. <b>Bits 7-4</b> Indicates max supported value of <b>tclk_trail_min</b> or <b>tclk_trail_min_ex</b> , whichever is appropriate.
dphy_limits_3	8-bit unsigned integer	RO	<b>Bits 3-0</b> Indicates max supported value of <b>tclk_prepare</b> or <b>tclk_prepare_ex</b> , whichever is appropriate. <b>Bits 7-4</b> Indicates max supported value of <b>tclk_zero</b> or <b>tclk_zero_ex</b> , whichever is appropriate.
dphy_limits_4	8-bit unsigned integer	RO	<b>Bits 3-0</b> Indicates max supported value of <b>tclk_post</b> or <b>tclk_post_ex</b> , whichever is appropriate. <b>Bits 7-4</b> Indicates max supported value of <b>tlpx</b> or <b>tlpx_ex</b> , whichever is appropriate.
dphy_limits_5	8-bit unsigned integer	RO	<b>Bits 3-0</b> Indicates max supported value of <b>ths_exit</b> or <b>ths_exit_ex</b> , whichever is appropriate, when D-PHY is selected <b>Bits 7-4</b> Indicates max supported value of <b>twakeup</b> when D-PHY is selected.
dphy_limits_6	8-bit unsigned integer	RO	<b>Bits 3-0</b> Indicates max supported value of <b>tinit</b> when D-PHY is selected. <b>Bits 7-4</b> Reserved

### 7.9.3 C-PHY Timing Control Registers

1086 Two sets of C-PHY timing control registers are defined in this Section:

- 1087 • Set 1: Recommended and optional timing control registers corresponding to the C-PHY  
1088 Specification
- 1089 • Set 2: Optional alternate manual timing control registers

#### 7.9.3.1 C-PHY Timing Control Register Set 1

1090 The first set of CCS C-PHY Timing Control Registers (see *Table 44*) closely follows the C-PHY v1.0 timing  
1091 control specifications. Note that although the C-PHY v1.0 specification describes these parameters, it does  
1092 not mandate them; this CCS Specification recommends some of them.

1093 The Host can determine whether the **TGR\_Preamble\_Length** and **TGR\_Post\_Length** registers are supported  
1094 by inspecting bit 0 of the register **PHY\_ctrl\_capability\_2**. The Host can determine whether the  
1095 **TGR\_Preamble\_Prog\_Sequence\_n, n+1** registers are supported by inspecting bit 1 of the register  
1096 **PHY\_ctrl\_capability\_2**. These registers can be used when the register **PHY\_ctrl** has the value 2 (manual  
1097 control).

**Table 44 C-PHY Timing Parameters from MIPI C-PHY Specification**

Register Name	RW	Comment
<b>TGR_Preamble_Length</b> (Recommended)	RW	<p><b>Bit 7: Enable/Disable the Preamble Programmable Sequence</b> 0: Disable the Preamble Programmable Sequence, the lower waveforms of Figure 13 1: Enable the Preamble Programmable Sequence, the upper waveforms of Figure 13</p> <p><b>Default:</b> 0 (Disables the Preamble Programmable Sequence on system reset)</p> <p><b>Bit 6:</b> Reserved for future use</p> <p><b>Bits 5–0: Begin_Preamble_Length</b> The number of Symbols in the PreBegin section of the preamble is: (<b>Begin_Preamble_Length</b> + 1) *7</p> <p><b>Default:</b> 0x3f (63 decimal), which sets the length of the PreBegin part of the Preamble to 64 Words on system reset.</p> <p>The PreBegin part of the Preamble may range from 1 to 64 Words, or 7 to 448 Symbols</p> <p>Refer to <b>Section 12.5.3</b> in the MIPI C-PHY v1.0 Specification [<b>MIPI04</b>]</p>
<b>TGR_Post_Length</b> (Recommended)	RW	<p><b>Bits 7–5:</b> Reserved for future use</p> <p><b>Bits 4–0: Post_Length</b> The number of Symbols in the Post field is: (Post_Length + 1) *7</p> <p><b>Default:</b> 0x1f (31 decimal) (Sets the length of the Post to 32 Words on system reset)</p> <p>The Post field may range from 1 to 32 Words, or 7 to 224 Symbols</p> <p>Refer to <b>Section 12.5.4</b> in the MIPI C-PHY v1.0 Specification [<b>MIPI04</b>]</p>
<b>TGR_Preamble_Prog_Sequence_n, n+1</b> i.e. registers (Optional)  i.e.: <b>TGR_Preamble_Prog_Sequence_0,1</b> <b>TGR_Preamble_Prog_Sequence_2,3</b> <b>TGR_Preamble_Prog_Sequence_4,5</b> <b>TGR_Preamble_Prog_Sequence_6,7</b> <b>TGR_Preamble_Prog_Sequence_8,9</b> <b>TGR_Preamble_Prog_Sequence_10,11</b> <b>TGR_Preamble_Prog_Sequence_12,13</b>	RW	<p><b>Bits 7–6:</b> Reserved for future use.</p> <p><b>Bits 5–3: Symbol n+1 of the Preamble Programmable Sequence</b> 0–4: Symbol value 5–7: Invalid value, not a valid Symbol value. The default value is 3 on system reset.</p> <p><b>Bits 2–0: Symbol n of the Preamble Programmable Sequence</b> 0–4: Symbol value 5–7: Invalid value, not a valid Symbol value.</p> <p><b>Default on system reset:</b> 3</p> <p>Refer to Sections 12.5.5 -12.5.11 in the MIPI C-PHY 1.0 Specification [<b>MIPI04</b>]</p>

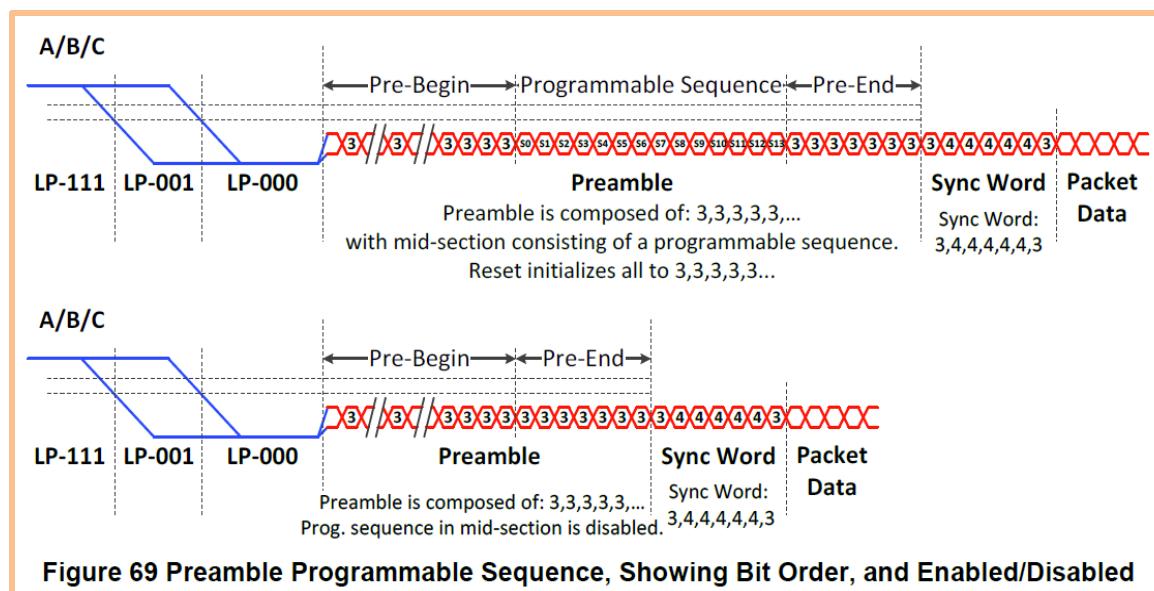


Figure 69 Preamble Programmable Sequence, Showing Bit Order, and Enabled/Disabled

1099

Figure 13 C-PHY Timing Details (Excerpt from C-PHY Specification)

### 7.9.3.2 C-PHY Timing Control Register Set 2

The second set of C-PHY timing control registers can be used if manual mode is set via register **PHY\_ctrl**, and if bit 2 of the register **PHY\_ctrl\_capability\_2** indicates that manual C-PHY mode is supported. These registers are not defined in the MIPI C-PHY Specification, and are optional.

For manual control there are two options based on Bit 3 and Bit 5 of register **PHY\_ctrl\_capability\_2**. If Bit 3 and Bit 5 have the same value, any option can be used. If they have different values, then Bit 5 option shall be used.

In UI clock based manual control, the PHY timing values are given by the following equation:

$$PHY\_time\_value = \frac{M * (register\_value + N) * (csi\_lane\_mode + 1)}{requested\_link\_rate}$$

#### Note:

For the value of M in the above equation, see **Section 7.9.3.2.1**. The value of N is 1 if Bit 6 of register **PHY\_ctrl\_capability\_3** is 0; otherwise, the value of N is 0.

In the generic clock based manual control, the PHY timing values controlled by registers **t3\_prepare**, **t3\_lpx**, **ths\_exit**, **ths\_exit\_ex**, **twakeup**, and **tinit** are given by the following equation:

$$PHY\_time\_value = \frac{(register\_value + N)}{generic\_clock\_frequency}$$

In the above equation, **generic\_clock\_frequency** is defined by the image sensor datasheet, for example twice the D/C-PHY PPI escape clock frequency. The value of N above is the same as the value of N in the UI clock based manual control equation.

### 7.9.3.2.1 C-PHY Manual Constant

In the above UI clock based manual control equation, if bit 5 of register **PHY\_ctrl\_capability\_3** is 0, the value of ‘register\_value + N’ shall be an integer multiple of a constant (i.e. must be evenly divisible by the constant); a typical value for this constant is 7. If the value of this constant is not 7, then the value shall be reported in the capability register **cphy\_manual\_constant**. If the value of this constant is 7, then the capability register value shall be 0 or 7. Otherwise, if bit 5 of register **PHY\_ctrl\_capability\_3** is 1, then the value of M in the above UI clock based manual control equation shall equal the latter constant, and the value of ‘register value + N’ is independent of M.

Table 45 C-PHY Manual Constant Parameters

Register Name	Type	RW	Comment
<b>cphy_manual_constant</b>	8-bit unsigned integer	RO	Specifies constant used in PHY time value calculation

### 7.9.3.2.2 C-PHY Additional Timing Parameters

As shown in *Table 46*, CCS specifies additional optional parameters for the C-PHY transmitter, which may be used to adjust transmitter timing parameters; four of them are shared with D-PHY.

1127

**Table 46 Additional C-PHY Timing Parameters**

Register Name	Type	RW	Comment
<b>t3_prepare</b>	16-bit unsigned integer	RW	Time that the transmitter drives the 3-wire LP-000 line state immediately before starting the HS transmission. <b>Default:</b> Image-sensor-specific
<b>t3_lpx</b>	16-bit unsigned integer	RW	Length of any Low-Power state period. <b>Default:</b> Image-sensor-specific
<b>twakeup</b>	8-bit unsigned integer	RW	Time that the PHY drives a Mark-1 state prior to a Stop state in order to initiate an exit from ULPS. <b>Default:</b> Image-sensor-specific <i>This register is shared with D-PHY.</i>
<b>tinit</b>	8-bit unsigned integer	RW	Minimum time that the PHY drives the Stop state during PHY initialization prior to starting any high-speed transmission. <b>Default:</b> Image-sensor-specific <i>This register is shared with D-PHY.</i>
<b>cphy_sf</b>	8-bit unsigned integer	RO	<b>Bits 3-0</b> Power-of-two scale factor for <b>twakeup</b> register values when C-PHY is selected and bit 2 of <b>PHY_ctrl_capability_3</b> is set to 1. <b>Bits 7-4</b> Power-of-two scale factor for <b>tinit</b> register values when C-PHY is selected and bit 3 of <b>PHY_ctrl_capability_3</b> is set to 1. <b>Default:</b> Image-sensor-specific
<b>ths_exit</b>	8-bit unsigned integer	RW	Time (identified as t3-HS-EXIT in the C-PHY spec) that the transmitter drives the Stop state following a high-speed burst. <b>Default:</b> Image-sensor-specific <i>This register is shared with D-PHY.</i>
<b>ths_exit_ex</b>	16-bit unsigned integer	RW	Time (identified as t3-HS-EXIT in the C-PHY spec) that the transmitter drives the Stop state following a high-speed burst. <b>Default:</b> Image-sensor-specific <i>This register is shared with D-PHY.</i>

1128 Image sensor support for **twakeup** and/or **tinit**, along with **cphy\_sf**, is indicated when bit 2 and/or bit 3,  
 1129 respectively, of the **PHY\_ctrl\_capability\_3** register is 1 and C-PHY is the selected PHY. For C-PHY the time  
 1130 values corresponding to the contents of the **twakeup** and **tinit** registers follow the equations described in  
 1131 **Section 7.9.3.2**, but require the application of the additional power-of-two scale factors stored in the read-  
 1132 only **cphy\_sf** register; i.e. the time values corresponding to the 8-bit contents of **twakeup** and **tinit** are  
 1133 multiplied by  $2^{\wedge} \text{cphy\_sf}[3:0]$  and  $2^{\wedge} \text{cphy\_sf}[7:4]$ , respectively. The scale factors are intended to  
 1134 facilitate the programming of the relatively large time values corresponding to the PHY T<sub>WAKEUP</sub> and T<sub>INIT</sub>  
 1135 timing parameters and shall be large enough to ensure that the minimum required values of these parameters  
 1136 can be satisfied.

1137 Support for **ths\_exit** or **ths\_exit\_ex** is indicated if bit 4 of **PHY\_ctrl\_capability\_3** is 1. Note that **ths\_exit\_ex**  
 1138 shall be supported if bits 4 and 5 of **PHY\_ctrl\_capability\_3** are 1 and 0, respectively.

### 7.9.3.2.3 C-PHY Timing Register Limit Values

The limit registers shown in *Table 47* enable image sensors to specify maximum allowed values for the various C-PHY timing registers defined in *Section 7.9.3.2.2*. The maximum value of each timing register is given by  $[2 ^ (L + 1)] - 1$ , where L is the corresponding 4-bit field value shown in *Table 47*. If L = 0, then no maximum value is specified other than that implied by the bit width of each timing register; however, note that each register may be programmed only with values supporting C-PHY timing within the min-max limits defined in the C-PHY specification.

**Table 47 C-PHY Timing Register Limits**

Register Name	Type	RW	Comment
cphy_limits_1	8-bit unsigned integer	RO	<b>Bits 3-0</b> Indicates max supported value of <b>t3_prepare</b> . <b>Bits 7-4</b> Indicates max supported value of <b>t3_ipx</b> .
cphy_limits_2	8-bit unsigned integer	RO	<b>Bits 3-0</b> Indicates max supported value of <b>ths_exit</b> or <b>ths_exit_ex</b> , whichever is appropriate, when C-PHY is selected. <b>Bits 7-4</b> Indicates max supported value of <b>t wakeup</b> when C-PHY is selected.
cphy_limits_3	8-bit unsigned integer	RO	<b>Bits 3-0</b> Indicates max supported value of <b>t init</b> when C-PHY is selected. <b>Bits 7-4</b> Reserved

## 7.9.4 Equalization

### 7.9.4.1 D-PHY Equalization

Starting in D-PHY v2.1, the transmitter shall support equalization if speeds higher than 2.5Gbit/s per Lane are used. See [\[MIPI08\]](#) for details regarding the equalization. If the image sensor supports a Lane speed higher than 2.5Gbit/s, then it should support CCS registers **PHY\_equalization\_ctrl** and **DPHY\_equalization\_mode** in order to control the equalization.

Image sensor capabilities related to D-PHY equalization are defined in register **DPHY\_equalization\_capability**.

### 7.9.4.2 C-PHY Equalization

The transmitter may support Advanced Tx Equalization. See [\[MIPI10\]](#) for details regarding Advanced Tx Equalization. If the image sensor supports Advanced Tx Equalization, then it should support CCS register **PHY\_equalization\_ctrl** in order to control the Advanced Tx Equalization.

Image sensor capabilities related to C-PHY equalization are defined in the register **CPHY\_equalization\_capability**.

### 7.9.4.3 PHY Equalization Registers

Capability and control registers for PHY Equalization are defined in this Section.

**Table 48 PHY Capability Register**

Register Name	Type	RW	Comment
<b>DPHY_equalization_capability</b>	8-bit unsigned integer	RO	<b>Bit 0</b> 1: Supports CCS equalization control <b>Bit 1</b> 1: Supports EQ1 <b>Bit 2</b> 1: Supports EQ2 <b>Other Bits</b> Reserved for future use
<b>CPHY_equalization_capability</b>	8-bit unsigned integer	RO	<b>Bit 0</b> 1: Supports CCS equalization control <b>Other Bits</b> Reserved for future use

**Table 49 PHY Equalization Register**

Register Name	Type	RW	Comment
<b>DPHY_equalization_mode</b>	8-bit unsigned integer	RW	<b>Bit 0</b> 0: Use EQ1 1: Use EQ2 <b>Other Bits</b> Reserved for future use <b>Default:</b> Sensor specific
<b>PHY_equalization_ctrl</b>	8-bit unsigned integer	RW	0: Do not use equalization 1: Use equalization <b>Default:</b> 0

### 7.9.5 D-PHY Preamble

Starting with D-PHY v2.1, the transmitter shall support the Preamble Sequence if speeds higher than 2.5Gbit/s per Lane are used. See [\[MIPI08J\]](#) for details regarding the preamble sequence. If the image sensor supports a Lane speed higher than 2.5Gbit/s, then it should support CCS registers **DPHY\_preamble\_ctrl** and **DPHY\_preamble\_length** in order to control the Preamble.

Image sensor capabilities related to the D-PHY preamble are defined in register **DPHY\_preamble\_capability**.

**Table 50 PHY Capability Register**

Register Name	Type	RW	Comment
<b>DPHY_preamble_capability</b>	8-bit unsigned integer	RO	<b>Bit 0</b> 1: Supports CCS preamble sequence control <b>Other Bits</b> Reserved for future use

**Table 51 D-PHY Preamble Registers**

Register Name	Type	RW	Comment
<b>DPHY_preamble_ctrl</b>	8-bit unsigned integer	RW	0: Do not send preamble sequence 1: Send preamble sequence for every HS burst <b>Default:</b> 0
<b>DPHY_preamble_length</b>	8-bit unsigned integer	RW	The length of preamble is controlled in steps of 32 UI, between 32 UI and 512 UI. Valid register value: 0 to 15 Length = ( register value + 1 ) * 32 <b>Default:</b> 0

### 7.9.6 D-PHY Spread Spectrum Clocking (SSC)

Starting with D-PHY v2.0, the transmitter shall support Spread Spectrum Clocking (SSC) if speeds higher than 2.5Gbit/s per Lane are used. See [\[MIPI08J\]](#) for details regarding SSC. If the image sensor supports a Lane speed higher than 2.5Gbit/s, then it should support CCS register **PHY\_SSC\_ctrl** in order to control the SSC.

**Table 52 PHY Capability Register**

Register Name	Type	RW	Comment
<b>DPHY_SSC_capability</b>	8-bit unsigned integer	RO	<b>Bit 0</b> 1: Supports CCS SSC control <b>Other Bits</b> Reserved for future use

**Table 53 D-PHY SSC Register**

Register Name	Type	RW	Comment
<b>PHY_SSC_ctrl</b>	8-bit unsigned integer	RW	0: Disable 1: Enable <b>Default:</b> 0

### 7.9.7 Manual LP Control

CCS v1.1 introduces manual control when the image sensor's CSI-2 transmitter moves enabled Lanes into the LP-11 or LP-111 states. This transition is controlled by register **manual\_LP\_ctrl**. This feature is recommended. The host can detect whether the feature is supported from bit 6 and bit 7 of register **PHY\_ctrl\_capability\_2** (see *Section 7.9.1*).

**Table 54 Manual LP Control Register**

Register Name	Type	RW	Comment
<b>manual_LP_ctrl</b>	8-bit unsigned integer	RW	<p><b>Bit 0</b> 0: Disabled 1: Enable immediate transition of the transmitter to the LP-11 or LP-111 state (and auto-clear after enabled Lanes have been transitioned to LP-11/LP-111)</p> <p><b>Other Bits</b> Reserved for future use</p> <p><b>Default:</b> 0</p>

Before enabling this register, the host shall ensure that the proper PHY has been selected (**CSI\_signalling\_mode**) and that correct number of Lanes have been configured (**CSI\_lane\_mode**). The image sensor shall move the enabled Lanes to the LP-11/LP-111 state immediately when the register is enabled, and shall auto-clear the register after the transition has happened. The register change shall be performed while in Software Standby Mode. If the number of Lanes is changed in mode changes, the Host shall first change number of Lanes, and then enable the register to transition the enabled Lanes to the LP-11/LP-111 state immediately.

## 7.10 Data Format Description – Type 1

The Data Format Description Registers list all output data formats that the image sensor supports. The Host can gain a complete understanding of the image sensor's data format capabilities by reading these registers.

Each supported output data format is described in one Format Description, consisting of several registers (see **Table 55**):

- Data Format Model Type (1 byte)

This byte contains the value 0x01 when normal data is used, or 0x02 if more than 8 Data Format Descriptors are needed (see **Section 7.11**)

- Data Format Model Sub-type (1 byte)

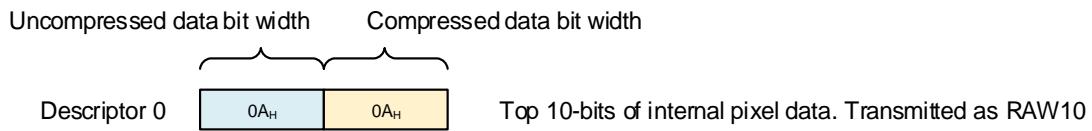
This byte contains the number of Data Format Descriptors that follow (from 1 to 8)

- A list of from 1 to 8 Data Format Descriptors, using the same 2-byte format as the register **CSI\_data\_format** (see **Figure 12**).

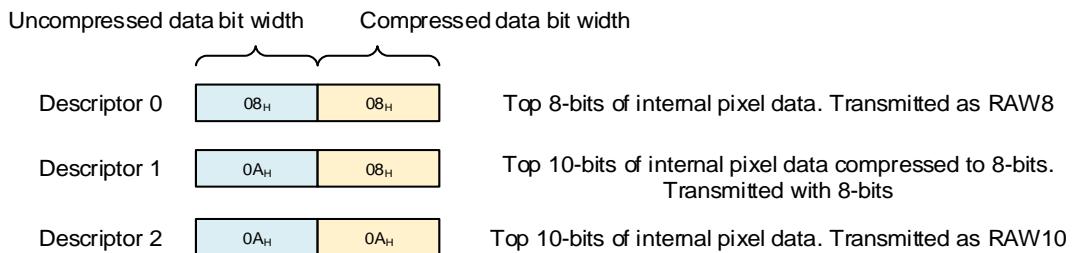
**Figure 14** shows example Data Format Descriptors for a simple image sensor supporting just RAW10, and **Figure 15** shows example Data Format Descriptors for a more complex image sensor supporting three different formats.

**Table 55 Data Format Description Registers**

Register Name	Type	RW	Comment
<b>data_format_model_type</b>	8-bit unsigned integer	RO	0x01: 2-byte data format (normal) 0x02: 2-byte data format (extended)
<b>data_format_model_subtype</b>	8-bit unsigned integer	RO	Contains the number of data format descriptors used
<b>data_format_descriptor_0</b>	16-bit unsigned integer	RO	–
<b>data_format_descriptor_1</b>	16-bit unsigned integer	RO	–
<b>data_format_descriptor_2</b>	16-bit unsigned integer	RO	–
<b>data_format_descriptor_3</b>	16-bit unsigned integer	RO	–
<b>data_format_descriptor_4</b>	16-bit unsigned integer	RO	–
<b>data_format_descriptor_5</b>	16-bit unsigned integer	RO	–
<b>data_format_descriptor_6</b>	16-bit unsigned integer	RO	–
<b>data_format_descriptor_7</b>	16-bit unsigned integer	RO	–

**Number of data format descriptors: 1**

1201

**Figure 14 Data Format Description – Simple Example****Number of data format descriptors: 3**

1202

**Figure 15 Data Format Description – More Complex Example**

## 7.11 Data Format Description – Type 2

If the register **data\_format\_model\_type** contains the value 0x02, then the image sensor requires more than 8 Data Format Descriptors to describe its supported output data formats. This may be needed depending on the number of different data formats the image sensor is required to support.

All the rules for Data Format Description - Type 1 registers (see *Section 7.10*) also apply to the following Type 2 registers, except that the maximum number of Data Format Descriptors is 16 instead of 8.

The Extended data Format Description Register names immediately follow the first 8 (normal) descriptors listed in *Table 55*.

**Table 56 Extended Data Format Description Registers**

Register Name	Type	RW	Comment
<b>data_format_descriptor_8</b>	16-bit unsigned integer	RO	–
<b>data_format_descriptor_9</b>	16-bit unsigned integer	RO	–
<b>data_format_descriptor_10</b>	16-bit unsigned integer	RO	–
<b>data_format_descriptor_11</b>	16-bit unsigned integer	RO	–
<b>data_format_descriptor_12</b>	16-bit unsigned integer	RO	–
<b>data_format_descriptor_13</b>	16-bit unsigned integer	RO	–
<b>data_format_descriptor_14</b>	16-bit unsigned integer	RO	–
<b>data_format_descriptor_15</b>	16-bit unsigned integer	RO	–

## 7.12 Data Encoding for Uncompressed Image Data

### 7.12.1 Data Pedestal

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

The image sensor shall have an internal function ensuring that the data pedestal value remains constant with Integration Time, Gain, and temperature, and between different image sensor configurations under nominal operation conditions. It is recommended that under extreme operation conditions, the pedestal value should stay close to its nominal value.

The Host system should always use the **data\_pedestal** register value to determine the image sensor's output black level. CCS v1.0 specifies 10-bit reporting mode, and CCS v1.1 introduces  $n$ -bit reporting mode. The image sensor shall support either of those reporting modes. The Host can detect which is supported from register **pedestal\_capability**.

In 10-bit reporting mode:

The **data\_pedestal** register shall always reflect the pedestal value of a 10-bit output. It is the Host's responsibility to understand that the actual pedestal value in the data changes according to the image sensor's output bit depth. The black level value may be 64 codes in RAW10.

#### Example 1: Decimal Values

A 10-bit image sensor has a pedestal value of 64. Therefore the **data\_pedestal** register has a static value of 64. The Host sets the image sensor to output RAW8 data. The actual pedestal value changes from 64 to 16 and the Host should understand that but the register value shall not change.

#### Example 2: Decimal Values

A 12-bit image sensor has a pedestal value of 240. Therefore the **data\_pedestal** register has a static value of 60, because  $60 \times 4 = 240$ . The Host should understand that the register value of 60 needs to be multiplied by 4 (i.e.,  $2^{12} / 2^{10} = 4$ ), because the image sensor outputs 12-bit data instead of 10-bit data.

In  $n$ -bit reporting mode:

The **data\_pedestal** register shall always reflect the pedestal value of a  $n$ -bit output, where " $n$ " is the image sensor's maximum ADC output resolution. It is the Host's responsibility to understand that the actual pedestal value in the data changes according to the image sensor's output bit depth. The black level value may be one of  $2^{(n-4)}$  codes for RAW $n$  output.

#### Example 3: Decimal Values

A 12-bit image sensor has a pedestal value of 240. Therefore the **data\_pedestal** register has a static value of 60, because  $60 \times 4 = 240$ . The Host should understand that when the same sensor outputs RAW10 data, the register value of 240 needs to be divided by 4 (i.e.,  $2^{12} / 2^{10} = 4$ ), because the image sensor outputs 10-bit data instead of 12-bit data.

The image sensor datasheet should indicate the data pedestal values for the supported output data formats, especially if the image sensor supports HDR synthesis operation.

1246

**Table 57 Data Pedestal Register**

Register Name	Type	RW	Comment
<b>data_pedestal</b>	16-bit unsigned integer	RO	–
<b>pedestal_capability</b>	8-bit unsigned integer	RO	0: Legacy 10-bit reporting 1: $n$ -bit reporting

1247

**Table 58 Typical Data Pedestal Value**

System	Typical Data Pedestal
10 bit	64

## 7.13 Frame Format Overview

### 7.13.1 CSI-2 Frame Format

The format of frames shall be as per the CSI-2 Specification, with additional CCS-specific requirements as shown in italic text in *Figure 16*.

A frame of raw image data has the following structure:

- 1 or more embedded data lines containing image sensor 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 image sensor image statistics data or e.g. PDAF data.  
See **Section 17.3**.

While the image data might be compressed via the 10-bit to 8-bit DPCM/PCM algorithm, the embedded data shall be never compressed. Thus when compression is used, a CSI-2 data frame contains a mixture of uncompressed data (the embedded data lines) and compressed data (the image data).

**Note:**

*Additional data using different data types or virtual channels may also be uncompressed, see Section 14.9 and Section 17.2 for details.*

The embedded data line (or lines) at the beginning of the frame shall contain most of the values of the CCS registers, as detailed below. To summarize, the values of all CCI Registers with indices 0x0000 to 0x0FFF are required to appear, with specific exceptions which may be left out in order to reduce the amount of top embedded data.

#### Generic Rules

The following registers are allowed, but should be omitted:

- Reserved and un-named registers
- Registers marked as OPTIONAL or REDUNDANT in the register map
- Registers that are unused (e.g. **op\_pix\_clk\_div** and/or **pll\_mode** in some devices)
- Unused frame format descriptors
- Registers associated with the data transfer interface(s)

#### Specific Rules

The following registers are allowed, but should be omitted:

- **General Status Registers [0x0000-0x0021]**

However, the following registers shall always be included:

- **frame\_count**
- **pixel\_order**
- **data\_pedestal**

- **Frame Format Description Registers [0x0040-0x007F]**

- **Analog Gain Description Registers [0x0080-0x009F]**

- **Data Format Description Registers [0x00C0-0x00E1]**

- **General Set-up Registers [0x0100-0x0109]**

However, the following register shall always be included:

- **image\_orientation**

- **Output Set-up Registers [0x0110-0x011F]**

1287 However, the following register shall always be included:

- 1288 • **CSI\_data\_format**
- 1289 • **Gain Set-up Registers [0x0120]**
- 1290 • **Generic Control Registers [0x0122]**
- 1291 • **GPIO Set-up Registers [0x0130]**
- 1292 • **Reference clock frequency registers [0x0136-0x0137]**
- 1293 • **USL Reverse Mode Clocking Registers [0x0312-0x0315]**
- 1294 • **start\_readout\_rs**
- 1295 • **compression\_mode**
- 1296 • **Test Pattern Registers**
- 1297 • The following Link-related registers:
  - 1298 • **PHY Configuration Register [0x0800-0x0819]**
  - 1299 • **CSI-2 Bitrate Negotiation Registers [0x0820-0x0823]**
  - 1300 • **Equalization Control Registers [0x0824-0x0825]**
  - 1301 • **D-PHY Preamble Control Registers [0x0826-0x0827]**
  - 1302 • **D-PHY Spread Spectrum Control Register [0x0828]**
  - 1303 • **Manual LP Control Register [0x0829]**
  - 1304 • **Additional PHY Configuration Registers [0x082A-0x082F]**
  - 1305 • **PHY Calibration Configuration Registers [0x0830-0x083F]**
  - 1306 • **C-PHY Manual Control Registers [0x0840-0x0851]**
  - 1307 • **CSI-2 v2 Feature Control Registers [0x085A – 0x08B1]**
  - 1308 • **USL Control Registers [0x08C0-0x08CF]**
- 1309 • **Optical Black Pixel Readout Registers [0x0B30-0x00B33]**

1310 However, the following register shall always be included:

- 1311 • **OB\_readout\_ctrl**
- 1312 • **flash\_strobe\_adjustment**
- 1313 • **tsa\_strobe\_width\_ctrl**
- 1314 • **PDAF Control Registers [0x0D00-0x0DFF]**

1315 However, the following registers shall always be included:

- 1316 • **PDAF\_ctrl**
- 1317 • **pdaf\_x\_addr\_start, pdaf\_y\_addr\_start, pdaf\_x\_addr\_end, and pdaf\_y\_addr\_end** (if PDAF ROI supported)
- 1318 • **Bracketing Interface Configuration Registers [0x0E00-0x0EFF]**

1319 However the following registers shall always be included:

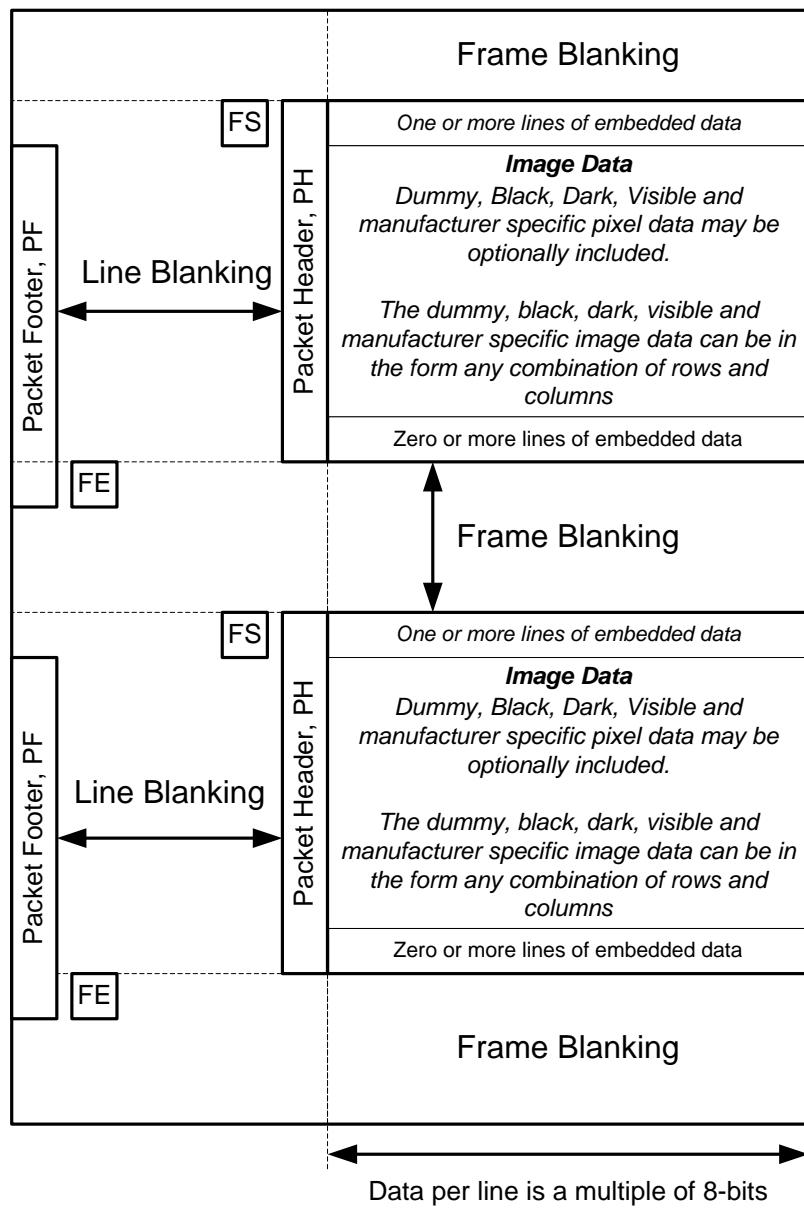
- 1320 • **bracketing\_LUT\_ctrl**
- 1321 • **bracketing\_LUT\_mode**

1322 Additional manufacturer-specific registers may also be included in the embedded data lines at the beginning of the frame.

1323 The very first embedded data line at the start of a frame may contain the frame format description.

1324 The optional bottom embedded data lines shall contain the values of all CCI Registers with indices 0x2000 to 0x2FFF. Additional manufacturer-specific registers may also be included.

1325 The structure of the image data, including the number of embedded data lines at the start and end of the frame, is specified in *Section 7.14*. The frame format description may be part of the data embedded in the first embedded data line of the CSI-2 data frame (see *Section 7.15*).

**KEY:**

PH – Packet Header  
 FS – Frame Start  
 LS – Line Start

PF – Packet Footer  
 FE – Frame End  
 LE – Line End

1331

**Figure 16 CSI-2 Frame Format**

### 7.13.2 Frame Format Elements

#### 7.13.2.1 Embedded Data Lines

The embedded data lines provide a mechanism to embed non-image data, such as image sensor configuration details and image statistics values, with a frame of CSI-2 data.

The minimum requirement is to have one embedded data line at the start of the frame. The length of embedded data line may be shorter than image data line (see e.g. **Section 7.15.1** and **Section 7.15**). To fulfil these rules, more than one embedded data line can be used if needed, depending on the number of CCI Register values to be added to the embedded data.

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

#### 7.13.2.2 Dummy Pixel Data

This is invalid pixel data. The receiver shall handle dummy pixel data like any other data, such as visible pixel data.

#### 7.13.2.3 Black Pixel Data

Black pixels contain pixel data with zero Integration Time, and the same analog Gain value as visible pixels.

#### 7.13.2.4 Dark Pixel Data

Dark pixels have the same Integration Time and analog 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 sensor's output black level (data pedestal).

**Note:**

*Dark Pixels are also known as 'Optical Black pixels' or 'Light Shielded pixels'.*

#### 7.13.2.5 Visible Pixel Data

Visible pixels contain valid image data. The correct Integration Time and analog Gain for the visible pixels is specified in the embedded lines at the start of the frame.

#### 7.13.2.6 Manufacturer Specific Pixel Data

To provide manufacturers a mechanism to include pixel data of types other than those already specified above, seven Manufacturer-Specific Pixel (MSP) data type codes are available.

#### 7.13.2.7 Frame Blanking Period

The user may choose to extend the frame-blanking period by increasing the frame length by writing to CCI Registers. In a CSI-2 frame there is no concept of frame blanking being transmitted. If the Host commands the image sensor to use an increased frame blanking period, then the bus will simply spend a longer time in the LP state at the end of the active data for each frame.

**Note:**

*The image sensor may also support a state in which the LP state is replaced by another, similar state.  
See **Section 7.7** for details.*

### 7.13.2.8 Line Blanking Period

The user can extend the line length by writing to CCI Registers. In a CSI-2 frame there is no concept of line blanking being transmitted. If the Host commands the image sensor to use an increased line blanking period, then the bus will simply spend a longer time in the LP state between active data lines.

**Note:**

*The image sensor may also support a state in which the LP state is replaced by another, similar state.  
See Section 7.7 for details.*

## 7.14 Frame Format Description

The frame format description registers fully describe the structure of the CSI-2 image data frame, including the number of embedded data lines at the start of the frame and the number of embedded data lines at the end of the frame. The frame format description defined in this Section is optional if CCS Static Data is used to provide similar information (see **Section B.2.11**).

This information is exposed in the form of CCI registers (and optionally via the embedded data line at the start of the frame) in order to enable software (or a dedicated image processing device) to be able to process arbitrary sized images without any prior knowledge of the image sensor's image frame format(s).

The visible pixel frame format description information read via the CCI Registers shall always be current, even if the image sensor is in Software Standby Mode. New **x\_output\_size** and **y\_output\_size** values programmed during software standby mode shall always be taken into account in frame format descriptors, even if the image sensor is not in Streaming Mode.

The intention is to allow the software (or dedicated image-processing device) to automatically configure itself by reading all needed frame format details directly from the embedded data lines.

The first embedded data line at the start of a frame may contain the frame format description. This is very important for applications such as digital zoom, where the image sensor's image size may vary dynamically over time. The **frame\_format\_model\_type** register specifies what frame format description the image sensor is currently using. Its presence is important because it provides a mechanism for adding new frame format descriptions in a manner that Host systems can easily detect.

Two frame format description codes are currently defined (see **Table 59**), and are detailed in the following Sub-Sections.

**Table 59 Frame Format Description Codes**

Code	Description
00h	<i>Invalid</i>
01h	2-byte Generic Frame Format Description
02h	4-byte Generic Frame Format Description
FFh	<i>Invalid</i>

### 7.14.1 2-Byte Generic Frame Format Description

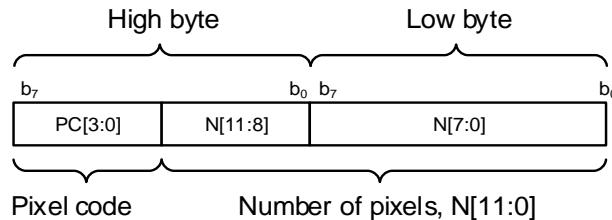
The general rules are as follows:

- A CSI frame of raw 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 have any combination of columns and/or rows
- The embedded data shall only be used in rows
- There shall be at least 1 embedded data line at the start of the frame

### 7.14.1.1 2-Byte Generic Frame Format Descriptor

1393

The basic component of this format is a 2-byte descriptor as illustrated in *Figure 17*.



1394

**Figure 17 2-Byte Generic Frame Format Descriptor**

1395

**Table 60 Pixel Code Definitions**

Pixel Code	Definition
0	<i>Invalid</i>
1	Embedded data
2	Dummy Pixel data
3	Black Pixel data
4	Dark Pixel data
5	Visible Pixel data
6	<i>Reserved</i>
7	<i>Reserved</i>
8	Manufacturer-Specific Pixel type 0
9	Manufacturer-Specific Pixel type 1
10	Manufacturer-Specific Pixel type 2
11	Manufacturer-Specific Pixel type 3
12	Manufacturer-Specific Pixel type 4
13	Manufacturer-Specific Pixel type 5
14	Manufacturer-Specific Pixel type 6
15	<i>Invalid</i>

1396

The format of the 2-Byte Generic Frame Format Descriptor is as follows:

1397

- **Pixel Code:** The top 4 bits of the High byte

1398

This indicates the pixel data type (see *Table 60*)

1399

- **Number of Pixels (12 bits):** The low 4 bits of the High byte, plus the entire Low byte

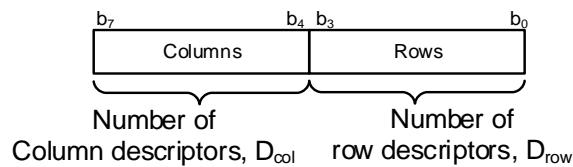
1400

This is the number of pixels with the indicated type. The maximum number of pixels specified by a descriptor is 4095.

1401

**Example:** A Pixel Code of 4 and a Number of Pixels of 1024 indicates 1024 Dark Pixels.

### 7.14.1.2 Frame Format Subtype Register



**Figure 18 Frame Format Subtype Register – Number of Column and Row Descriptors Used**

The **frame\_format\_model\_subtype** register (see *Figure 18*) specifies the number of column and row descriptors:

- **Number of Column Descriptors:** Top 4 bits

This specifies the number of column descriptors, D<sub>col</sub>

- **Number of Row Descriptors:** Bottom 4 bits

This specifies the number of row descriptors, D<sub>row</sub>

### 7.14.1.3 Descriptor Rules

The image sensor shall observe all of the following rules in its use of the descriptors:

1. The column descriptors shall be specified first (x-direction – C-C section)
2. There shall be a block of D<sub>col</sub> column descriptors followed by a block of D<sub>row</sub> row descriptors. Column and row descriptors shall not 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 image data frame. This cross-section shall intersect the visible portion of the image data.
4. The total number of columns defined by the column descriptors shall equal the number of pixels in each long packet in CSI-2, except that:
  - A. An embedded data line may be shorter (see **Section 7.15**)
  - B. This requirement is not relevant for interleaved manner readout modes of PDAF or OB pixels.
5. The row descriptors specify the top-to-bottom structure of a vertical cross-section (R-R section in the examples) through the image data frame. This cross-section shall intersect the visible portion of the image data.
6. The total number of rows defined by the row descriptors shall 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, in each CSI-2 embedded data long packet.
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 a 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 shall 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 shall not exceed 15.

The Host should be able extract the embedded data lines and the visible image pixel data from all of the examples of the generic frame format description given in the following pages.

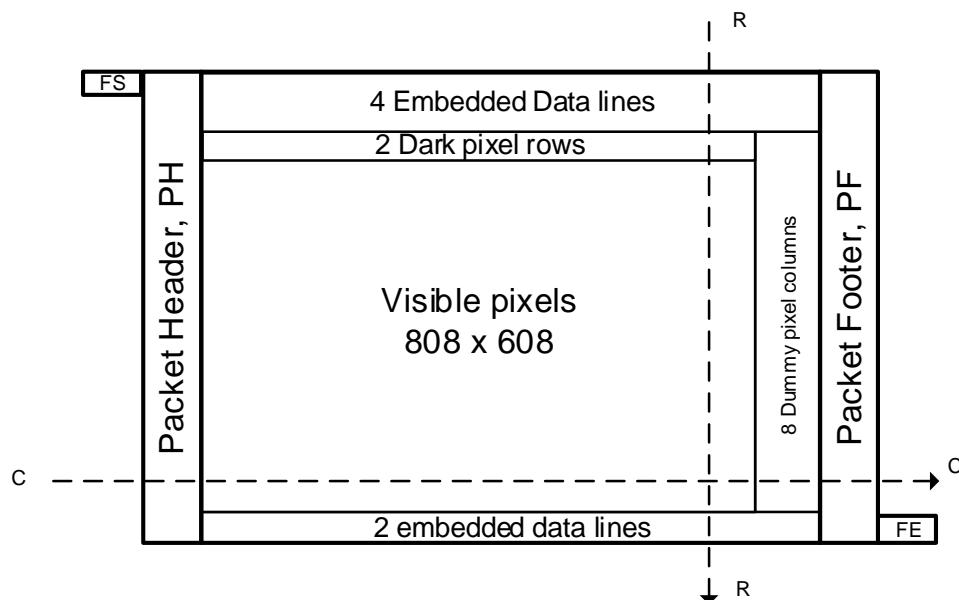
The first embedded data line at the start of a frame may contain the frame format description.

#### 7.14.1.4 2-Byte Generic Frame Format Description Registers

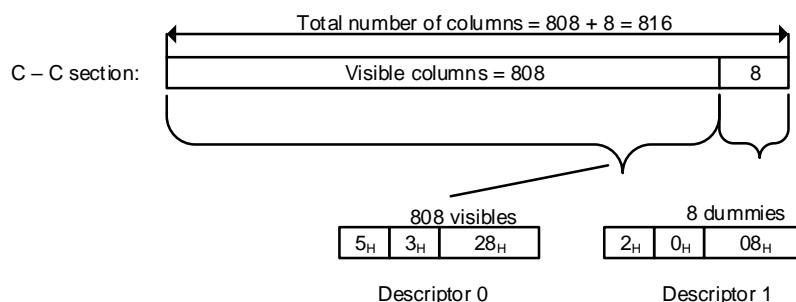
1439

**Table 61 2-Byte Generic Frame Format Description Registers**

Register Name	Type	RW	Comment
<b>frame_format_model_type</b>	8-bit unsigned integer	RO Dynamic	0x01: 2-Byte Generic Frame Format 0x02: 4-Byte Generic Frame Format
<b>frame_format_model_subtype</b>	8-bit unsigned integer	RO Dynamic	The number of column and row descriptors used to describe the frame format
<b>frame_format_descriptor_0</b>	16-bit unsigned integer	RO Dynamic	–
<b>frame_format_descriptor_1</b>	16-bit unsigned integer	RO Dynamic	–
<b>frame_format_descriptor_2</b>	16-bit unsigned integer	RO Dynamic	–
<b>frame_format_descriptor_3</b>	16-bit unsigned integer	RO Dynamic	–
<b>frame_format_descriptor_4</b>	16-bit unsigned integer	RO Dynamic	–
<b>frame_format_descriptor_5</b>	16-bit unsigned integer	RO Dynamic	–
<b>frame_format_descriptor_6</b>	16-bit unsigned integer	RO Dynamic	–
<b>frame_format_descriptor_7</b>	16-bit unsigned integer	RO Dynamic	–
<b>frame_format_descriptor_8</b>	16-bit unsigned integer	RO Dynamic	–
<b>frame_format_descriptor_9</b>	16-bit unsigned integer	RO Dynamic	–
<b>frame_format_descriptor_10</b>	16-bit unsigned integer	RO Dynamic	–
<b>frame_format_descriptor_11</b>	16-bit unsigned integer	RO Dynamic	–
<b>frame_format_descriptor_12</b>	16-bit unsigned integer	RO Dynamic	–
<b>frame_format_descriptor_13</b>	16-bit unsigned integer	RO Dynamic	–
<b>frame_format_descriptor_14</b>	16-bit unsigned integer	RO Dynamic	–



Number of Column Descriptors: 2



Number of Row Descriptors: 4

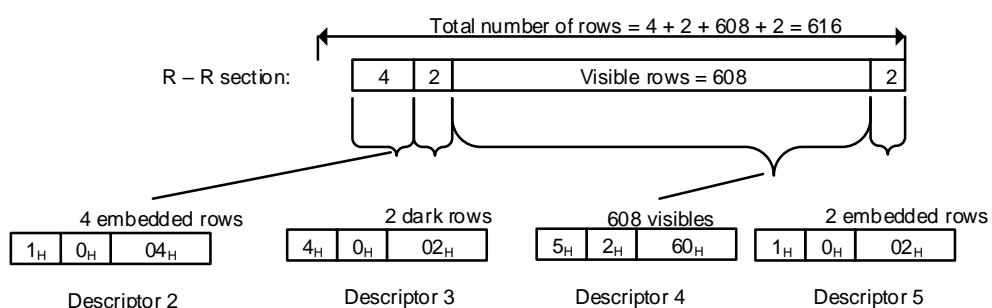


Figure 19 Generic Frame Format Description - SVGA Example

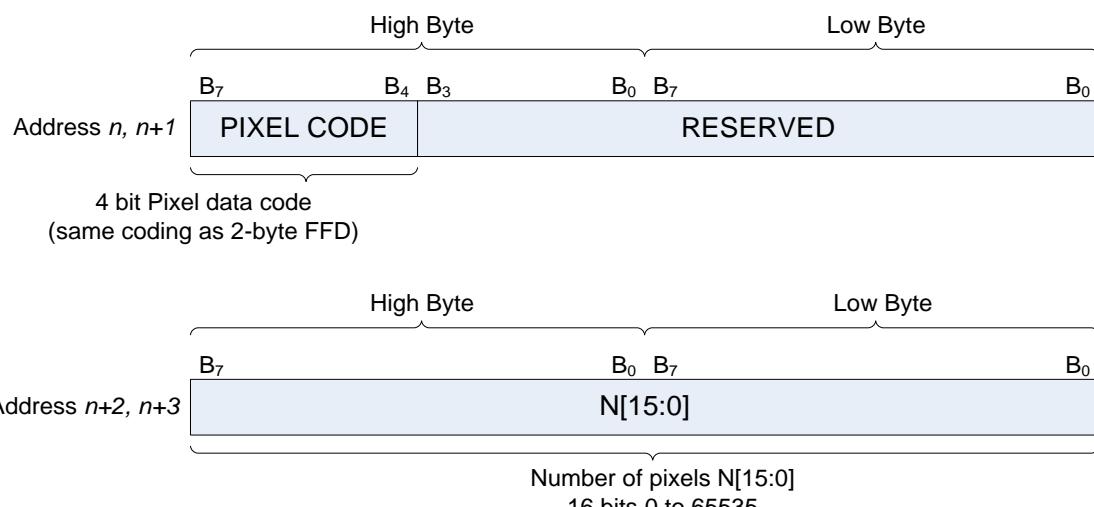
### 7.14.2 4-Byte Generic Frame Format Description

In image sensors where the horizontal or vertical resolution exceeds 4095 pixels, the 4-byte Generic Frame Format Descriptor (FFD) shall be used. Where 4-byte FFDs are used, the locations reserved for the 2-byte FFDs shall not be used.

Register **frame\_format\_model\_subtype** (see *Section 7.14.1.2*) shall be used to indicate the numbers of FFDs for both 2-byte and 4-byte types, and all Descriptor Rules for the 2-byte Generic Frame Format Description type (see *Section 7.14.1.3*) shall apply for both 2-byte and 4-byte types.

#### 7.14.2.1 4-Byte Generic Frame Format Descriptor

The 4-byte FFD shall be coded as shown in *Figure 20*. The Pixel Code has the same interpretation as in the 2-byte FFD (see *Section 7.14.1.1*), but the remainder of the High byte and the entire Low byte are ignored. Instead, the entire second 16-bit word is used as the Number of Pixels, for a range of up to 65,535 pixels.



**Figure 20 4-Byte Frame Format Descriptor Format**

### 7.14.2.2 4-Byte Generic Frame Format Description Registers

1451

The 4-byte FFDs shall occupy a separate space from the 2-byte FFDs, as shown in *Table 62*:

**Table 62 4-Byte Generic Frame Format Description Registers**

Register Name	Type	RW	Comment
frame_format_descriptor_4_0	32-bit unsigned integer	RO Dynamic	–
frame_format_descriptor_4_1	32-bit unsigned integer	RO Dynamic	–
frame_format_descriptor_4_2	32-bit unsigned integer	RO Dynamic	–
frame_format_descriptor_4_3	32-bit unsigned integer	RO Dynamic	–
frame_format_descriptor_4_4	32-bit unsigned integer	RO Dynamic	–
frame_format_descriptor_4_5	32-bit unsigned integer	RO Dynamic	–
frame_format_descriptor_4_6	32-bit unsigned integer	RO Dynamic	–
frame_format_descriptor_4_7	32-bit unsigned integer	RO Dynamic	–

## 7.15 Embedded Data Line Formats

This Section specifies the formats of the embedded data lines.

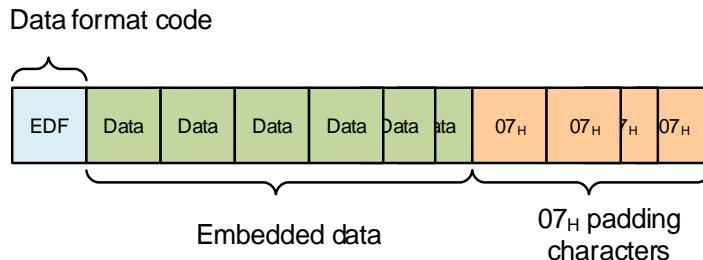
The purpose of the embedded data line formats is to allow the inclusion of extra information, such as device set up information, within the output data stream. An example of this is the frame format description, which may be included in the embedded data line at the start of frame.

Also bottom-embedded data shall contain embedded data format code, when bottom-embedded data is used. For example, PDAF or statistical information may be included in bottom-embedded data.

The length of the embedded data line shall not exceed the length of the image data line. The embedded data line should have the same length as the image data line.

The high level format of embedded data lines is as follows:

- The first pixel value in the beginning of embedded data line shall denote the format used for the embedded data
- A sequence of embedded data values in the appropriate format.
- The remainder of the line is padded with 07h characters.



**Figure 21 Embedded Data Line Format**

**Table 63 Embedded Data Format Codes**

Embedded Data Format Code	Description
00h	Invalid – If found, treat as No Data
07h	No Data – Remainder of line 07H
0Ah	Simplified 2-Byte Tagged Data Format
0Bh	PDAF data
0Ch	Statistics data
0Dh	CSI-2 SROI, introduced in <a href="#">[MIPI12]</a>
0Eh-1Fh	Reserved for future use
20h-2Fh	Vendor specific
30h-FEh	Reserved for future use
FFh	Invalid – If found, treat as No Data

### 7.15.1 Simplified 2-Byte Tagged Data Format

The simplified 2-byte tagged data format includes RAW-format dependent packing. This format shall be used with top-embedded data.

By default, top-embedded data shall use the same data packing as the Visible pixel data with one embedded data byte per pixel. For example, if the Visible pixels use RAW10, then the top-embedded data shall also be packed as RAW10.

However, for RAW16, RAW20, and/or RAW24 Visible pixels, top-embedded data may instead be more optimally packed using RAW8, RAW10, and/or RAW12 pixels, respectively, effectively enabling two embedded data bytes to be transported per RAW16, RAW20, and/or RAW24 pixel instead of the normal single byte per pixel; see *Annex A* for examples. In this context, even numbers of RAW8, RAW10, and/or RAW12 pixels shall be used to transport an equal number of bytes on each embedded data line while at the same time also satisfying applicable CSI-2 pixel granularity rules; for example, the latter rules require the number of RAW10 pixels to be a multiple of four. See *Table 64* for the capability and control registers associated with this option. The image sensor may support either of the packings, and if both are supported then the **emb\_data\_ctrl** control register shall be supported.

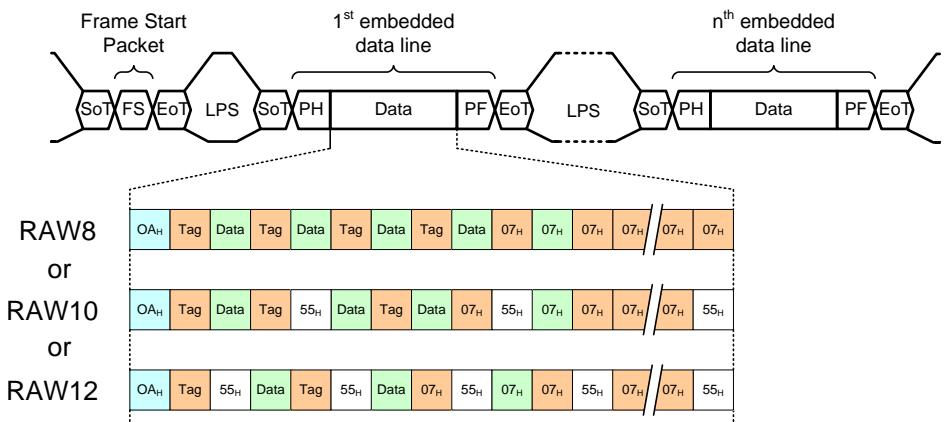
**Table 64 Embedded Data Capability and Control Registers**

Register Name	Type	RW	Comment
<b>emb_data_capability</b>	8-bit unsigned integer	RO	<b>Bit 0</b> 1: Supports two embedded data bytes per RAW16 pixel <b>Bit 1</b> 1: Supports two embedded data bytes per RAW20 pixel <b>Bit 2</b> 1: Supports two embedded data bytes per RAW24 pixel <b>Bit 3</b> 1: Does not support one embedded data byte per RAW16 pixel (legacy) <b>Bit 4</b> 1: Does not support one embedded data byte per RAW20 pixel (legacy) <b>Bit 5</b> 1: Does not support one embedded data byte per RAW24 pixel (legacy) <b>Other Bits</b> Reserved for future use

Register Name	Type	RW	Comment
<b>emb_data_ctrl</b>	8-bit unsigned integer	RW	<p><b>Bit 0</b> 1: Enable RAW8 packing for RAW16 Visible pixels 0: One embedded data byte per RAW16 pixel (legacy)</p> <p><b>Bit 1</b> 1: Enable RAW10 packing for RAW20 Visible pixels 0: One embedded data byte per RAW20 pixel (legacy)</p> <p><b>Bit 2</b> 1: Enable RAW12 packing for RAW24 Visible pixels 0: One embedded data byte per RAW24 pixel (legacy)</p> <p><b>Other Bits</b> Reserved for future use</p> <p><b>Default:</b> Image-sensor-specific</p>

If the Visible pixels are compressed, then the top-embedded data shall not be compressed. Instead, the top-embedded data shall use the same packing method as the compressed pixel data. Example: With 10-bit to 6-bit compression, RAW6 packing shall be used.

The general arrangement is shown in *Figure 22*.

**KEY:**

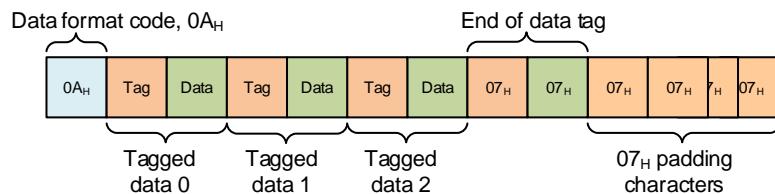
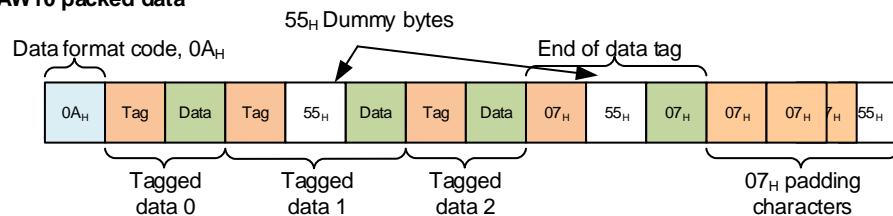
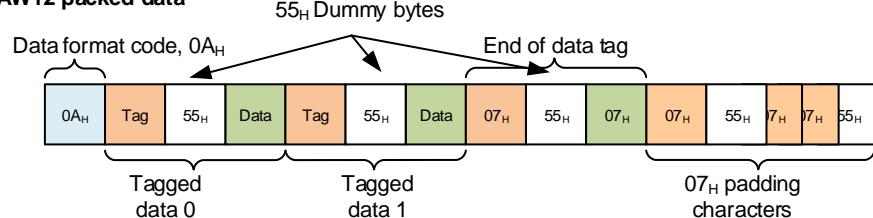
SoT – Start of Transmission  
PH – Packet Header  
FS – Frame Start

EoT – End of Transmission LPS – Low Power State  
PF – Packet Footer

1486

**Figure 22 Embedded Data in CSI-2**

1487 The first byte of the 2-byte packet is the tag byte. The tag byte value defines the meaning of the second byte,  
 1488 the data byte. The embedded data comprises a sequence of the above tagged data packets, and is terminated  
 1489 by a data packet with the special end-of-data tag, 07H.

**RAW8 packed data****RAW10 packed data****RAW12 packed data**

1490

**Figure 23 Tagged Data Format (Detail)**

### 7.15.1.1 Tag Codes

1491 The 2-byte tagged data format uses 5 different tag codes as shown in **Table 65**.

1492 **Table 65 Tagged Data Format Tag Code Summary**

Tag Code	Data Byte Description
00h	Invalid Tag. If found, treat as end of Data
07h	End of Data (Data Byte Value = 07h)
AAh	CCI Register Index MSB [15:8]
A5h	CCI Register Index LSB [7:0]
5Ah	Auto increment the CCI index after the data byte – valid data Data byte contains valid CCI Register data
55h	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 07h
FFh	Invalid Tag. If found, treat as end of Data

1493 Three tag types are defined:

#### 1. 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 register value for the current CCI index does not exist. In this case 07h shall be used as the data byte value.
- For the Null data tag the receiver or Host system shall always ignore the data byte value.

#### 2. 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:
  - First Way: Use both the MSB and LSB address tags together, to from a “long” jump over large areas of un-used CCI Register space.
  - Second Way: Use only the LSB address tag, to implement a “short” jump over small areas of un-used CCI Register space.

#### 3. Special Tags

- End-of-Data: 07h

### 7.15.1.2 Tagged Format Rules

#### 7.15.1.2.1 General Rules

- The embedded data sequence of tags and the number of embedded data lines shall 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) from the embedded data lines in the first frame of image data received by the Host system.
- The maximum number of tag-data pairs per line is 159. This is for compatibility with horizontal scaled output modes where the output image width may be only 320 pixels.
- Each embedded data line shall start with a “long” jump.

#### 7.15.1.2.2 Long Jump Rules

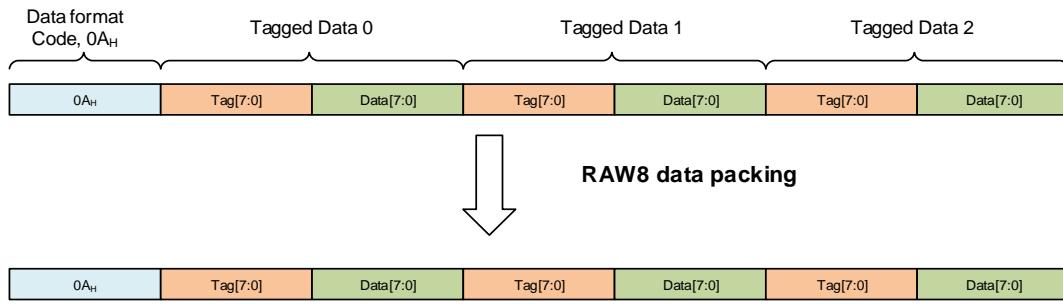
- The MSB and LSB Address Tags shall always be used together.
- Of the two tags, the MS Register Index Byte tag shall be specified first.
- The MSB and LSB Address Tags shall be the first two tags at the start of the embedded data.
- The 4 byte MS/LS Register Tag sequence can occur anywhere in the embedded 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.

#### 7.15.1.2.3 Short Jump Rules

- 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. This allows jumping over small sections of un-used register locations.
- The jump may be either forward or backwards within the CCI Register space.

### 7.15.1.3 Tagged Format Embedded Data Packing

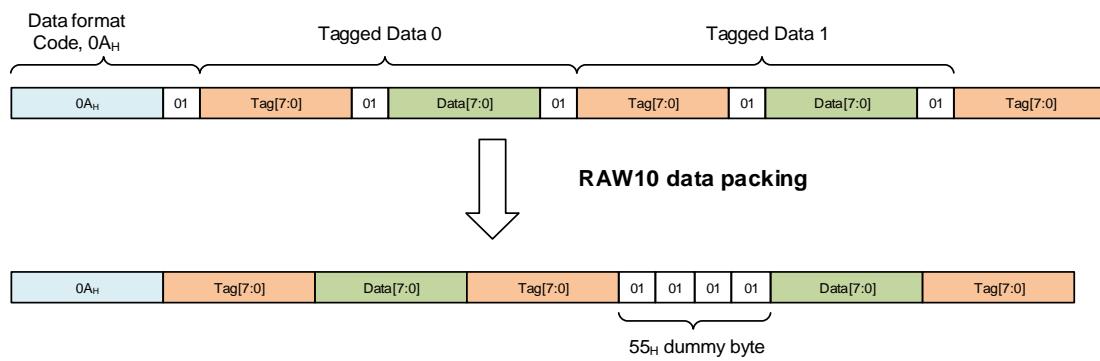
#### 8-bit tags and 8-bit data



1527

**Figure 24 RAW8 Embedded Data Packing**

#### 8-bit tags and 8-bit data extended to 10bits



1528

**Figure 25 RAW10 Embedded Data Packing**

1529  
1530

The embedded data packing for other RAW formats is described in *Annex A.1.1*. An example of embedded data content is shown in *Annex A.1.3*.

## 8 Video Timing

### 8.1 Introduction

This Section defines the following Video Timing functions:

- Programmable Image Size
- Image Readout order modes – Horizontal Mirror and Vertical Flip
- Binning modes
- Sub-sampled readout modes
- Variable Line Length and Frame Length
- Control of Pixel Clock Frequencies
- Video Timing Requirements
- Video Timing Capability Registers

This Section specifies the video timing for the image data that is read out from the image sensor's pixel array. This is not necessarily the same size as the output image data.

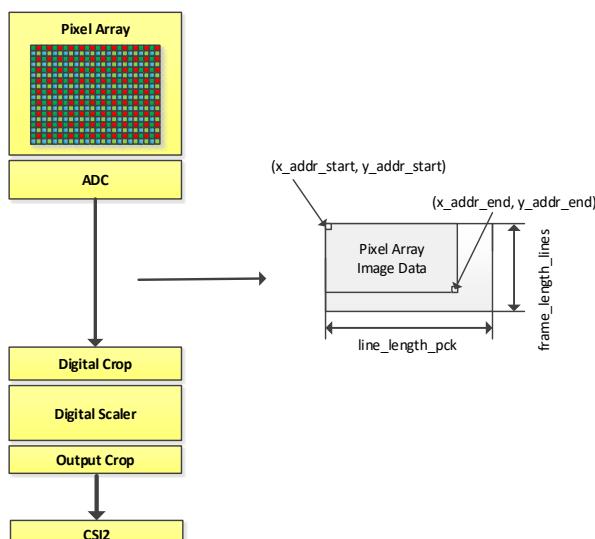
Features such as programmable image size, sub-sampling, and binning modes all affect the size of the image read from the pixel array. For these features, the output image is the same size as the image read from the pixel array.

The programmable image size, digital image scaling, and output size are independent functions. The Host shall ensure that these functions are programmed correctly for the intended application. These functions are used to reduce the amount of data, and to reduce the peak data rate on the CSI interface.

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

The Programmable Image Size, Variable Line Length, and Frame Length control registers shall be re-timed to the start of frame boundary, in order to ensure that these parameters are consistent within each image frame.

For the CCI indexes of the registers defined in this Section, please refer to the CCI Register Map (**Section 20**).



**Figure 26 Video Timing Overview**

## 8.2 Image Readout

This Section specifies the required and optional features for the image sensor image readout.

The two required features are:

- Programmable image size
- Horizontal Mirror and Vertical Flipped image readout

The two optional features are:

- Subsampled image readout
- Binned Image readout

### 8.2.1 Programmable Image Size

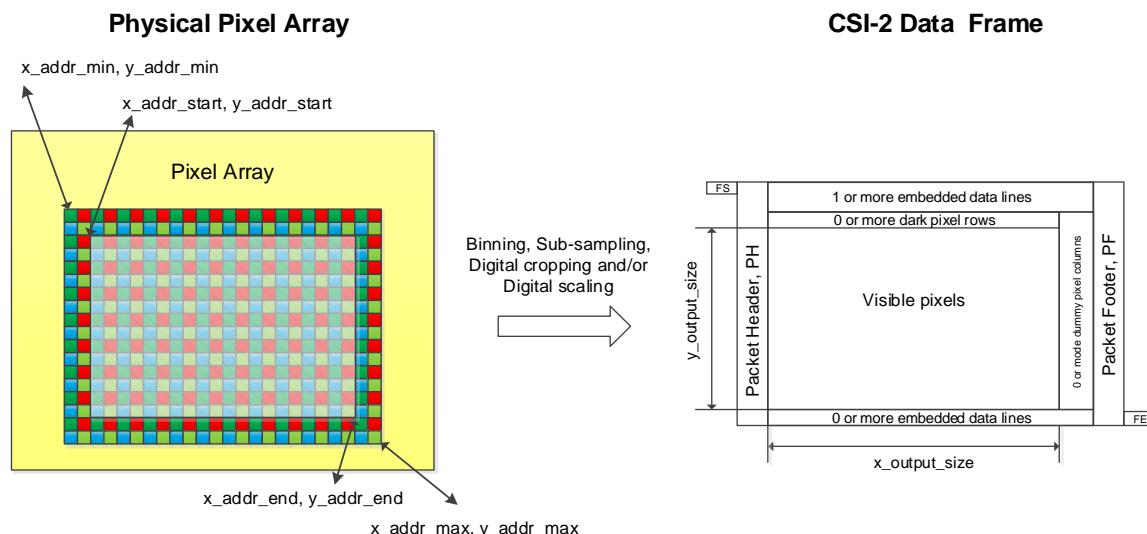
The image sensor shall have two independent methods for programming the image size.

- Method 1: Programmable addressable region of the pixel array
- Method 2: 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 binning factor, the sub-sampling factor, the digital crop, and the programmed image scaling downscale factor.

The Host system shall calculate the output image size based on the above parameters, and shall then program the image sensor's x and y output size registers.

Registers **x\_output\_size** and **y\_output\_size** are not intended as the primary cropping controls. Instead, they define a window for the output image data, so that the sensor does not need to calculate this based on the region of interest, sub-sampling, digital crop, and scaling settings. The Host should set the output sizes to exactly enclose the output image data. If the Host does not do so, then the image sensor shall treat the output sizes as being calculated from the top left hand corner of the output array. So in the case where output sizes are smaller than the output data, the data shall be cropped from its right hand and lower limits. In the case where output sizes are larger than the output data, and bit 2 of the **non\_flexible\_resolution\_support** capability register (*Table 68*) is set to 0, the lines shall be padded out to the defined output size with undefined data (for example, the last pixel value can be repeated). See *Section 8.2.1.2* for further details.



**Figure 27 Programmable Image Size**

### 8.2.1.1 Pixel Array Addresses

The addressable region of the pixel array should be at least 8 columns and 8 rows larger than the width and height of the image sensor's basic image format (*Table 66*). This gives a minimum border of 4 pixels around the whole pixel array which the color reconstruction algorithms can use at the edges of the pixel array (*Figure 28*).

The addressed region of the pixel array is controlled by the registers **x\_addr\_start**, **y\_addr\_start**, **x\_addr\_end**, and **y\_addr\_end** (*Table 67*). For Bayer pixels, the start and end addresses are limited to even and odd numbers, respectively, in order to ensure that an even number of pixels is always read out in both the x dimension and the y dimension.

The limits for the above parameters are given by the registers **x\_addr\_min**, **y\_addr\_min**, **x\_addr\_max**, and **y\_addr\_max** (*Table 68*). From this information the Host can automatically determine the image sensor's maximum resolution, and thus its image format.

The image sensor may have additional restrictions on the start and end addresses, if so indicated by bit 0 of register **non\_flexible\_resolution\_support**. For example, the start address might be required to be a multiple of 4. The Host can determine additional division rules for start addresses from registers **x\_addr\_start\_div\_constant** and **y\_addr\_start\_div\_constant** (i.e., if **x\_addr\_start** must be a multiple of **x\_addr\_start\_div\_constant**, or if **y\_addr\_start** must be a multiple of **y\_addr\_start\_div\_constant**). In addition, registers **x\_addr\_end\_div\_constant** and **y\_addr\_end\_div\_constant** specify the division rule constant for end address +1 (i.e., if **x\_addr\_end** +1 must be a multiple of **x\_addr\_end\_div\_constant**, or if **y\_addr\_end** +1 must be a multiple of **y\_addr\_end\_div\_constant**). The registers are mandatory with values greater than 4, and recommended with smaller values.

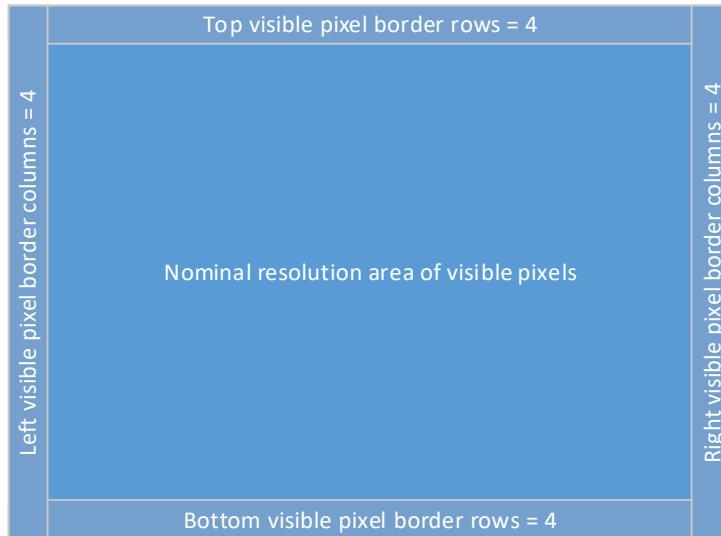
In addition, registers **x\_size\_div** and **y\_size\_div** are provided to describe any addition size limitations of the sensor, i.e., if  $(x_{addr\_end} + 1 - x_{addr\_start})$  or  $(y_{addr\_end} + 1 - y_{addr\_start})$  needs to be a multiple of **x\_size\_div** or **y\_size\_div**. If the value of these registers is zero, then it is assumed that end address division rules are enough to describe sensor restrictions. If the value of register **x\_size\_div** or register **y\_size\_div** is non-zero, then the Host can determine the valid values for pixel array addressing from the rules for the size, start, and end values. The registers are mandatory with values greater than 4, and recommended with smaller values.

Any additional restrictions related to binning, subsampling, or digital scaling should be visible in CCS Static Data by using Generic Rule Based Block (*Section B.2.8*).

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

Table 66 Example of Addressable Pixel Array Size

Format	Format Width	Format Height	Addressable Pixel Array Width	Addressable Pixel Array Height
VGA	640	480	648	488



1611

**Figure 28 Image Frame with Additional Border Pixels**

1612 The rules for the programmable pixel array start and end address are listed below:

- 1613 • The application of registers **x\_addr\_start**, **x\_addr\_end**, **y\_addr\_start**, and **y\_addr\_end** shall be  
1614 re-timed internally to the start of frame boundary.
- 1615 • The minimum limit for the x and y addresses are defined by the registers **x\_addr\_min** and  
1616 **y\_addr\_min** (*Table 68*). These values shall always be 0.
- 1617 • The maximum limit for the x and y addresses are defined by registers **x\_addr\_max** and  
1618 **y\_addr\_max** (*Table 68*).
- 1619 • The end address shall be greater than the start address
- 1620 • The x and y start addresses shall be set to even numbers only (0, 2, 4, 6, 8, etc.)

1621 **Note:**

1622     *The image sensor may set additional restrictions as indicated by registers*  
1623     ***x\_addr\_start\_div\_constant** and **y\_addr\_start\_div\_constant***

- 1624 • For Bayer pixel readout, the x and y end addresses shall be set to odd numbers only (1, 3, 5, 7, 9,  
1625 etc.). For monochrome pixel readout, the x and y end addresses may be either even or odd.

1626 **Note:**

1627     *The image sensor may set additional restrictions as indicated by registers*  
1628     ***x\_addr\_end\_div\_constant**, **y\_addr\_end\_div\_constant**, **x\_size\_div**, and **y\_size\_div***

- 1629 • The minimum size in lines in the y-direction shall be the largest minimum size specified by  
1630     registers **y\_addr\_start\_div\_constant**, **y\_addr\_end\_div\_constant**, and **y\_size\_div** for the non-  
1631     flexible resolution option, or 2 lines for the legacy flexible resolution option.
- 1632 • (**x\_addr\_max** – **x\_addr\_min**+1) shall be  $\geq$  Width of Basic Image Format + border pixels

1633 **Note:**

1634     *The image sensor should have 8 border pixels, consisting of a 4-pixel border around the image*  
1635     *sensor array.*

- 1636 • (**y\_addr\_max** – **y\_addr\_min**+1) shall be  $\geq$  Height of Basic Image Format + border pixels

1637 **Note:**

1638     *The image sensor should have 8 border pixels, consisting of a 4-pixel border around the image*  
1639     *sensor array.*

1640

**Table 67 Pixel Array Address Registers**

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 <b>Values:</b> Even numbers only: 0, 2, 4, 6... <b>Units:</b> Pixels <b>Default:</b> Image-sensor-specific
y_addr_start	16-bit unsigned integer	RW	Y-address of the top left corner of the visible pixel data <b>Values:</b> Even numbers only: 0, 2, 4, 6... <b>Units:</b> Lines <b>Default:</b> Image-sensor-specific
x_addr_end	16-bit unsigned integer	RW	X-address of the bottom right corner of the visible pixel data <b>Values:</b> For Bayer pixels, odd numbers only: 1, 3, 5, 7... <b>Units:</b> Pixels <b>Default:</b> Image-sensor-specific
y_addr_end	16-bit unsigned integer	RW	Y-address of the bottom right corner of the visible pixel data <b>Values:</b> For Bayer pixels, odd numbers only: 1, 3, 5, 7... <b>Units:</b> Lines <b>Default:</b> Image-sensor-specific

1641

**Table 68 Pixel Array Address Capability Registers**

Register Name	Type	RW	Comment
x_addr_min	16-bit unsigned integer	RO	Minimum X-address of the addressable pixel array <b>Value:</b> Always 0
y_addr_min	16-bit unsigned integer	RO	Minimum Y-address of the addressable pixel array <b>Value:</b> Always 0
x_addr_max	16-bit unsigned integer	RO	Maximum X-address of the addressable pixel array
y_addr_max	16-bit unsigned integer	RO	Maximum Y-address of the addressable pixel array
x_addr_start_div_constant	8-bit unsigned integer	RO	Divisible by constant for start address
y_addr_start_div_constant	8-bit unsigned integer	RO	Divisible by constant for start address
x_addr_end_div_constant	8-bit unsigned integer	RO	Divisible by constant for end address + 1
y_addr_end_div_constant	8-bit unsigned integer	RO	Divisible by constant for end address + 1
x_size_div	8-bit unsigned integer	RO	Divisible by constant for x size
y_size_div	8-bit unsigned integer	RO	Divisible by constant for y size

Register Name	Type	RW	Comment
<b>non_flexible_resolution_support</b>	8-bit unsigned integer	RO	<b>Bit 0</b> 1: New non-flexible definition supported for pixel array addressing 0: Legacy flexible definition supported <b>Bit 1</b> 1: New non-flexible definition supported for output size 0: Legacy flexible definition supported <b>Bit 2</b> 1: Output crop function doesn't support padding (i.e. X/Y output size needs to be set same or smaller than actual image) 0: Legacy flexible definition supported <b>Bit 3</b> 1: <b>x_output_size</b> is required to be a multiple of the number of CSI-2 output Lanes (i.e. <b>CSI_lane_mode</b> + 1) for the Coupled OP model or a multiple of the number of OP Lanes (i.e. <b>num_of_op_lanes</b> ) for the Decoupled OP model 0: No constraints on <b>x_output_size</b> due to <b>CSI_lane_mode</b> or <b>num_of_op_lanes</b> <b>Other Bits</b> Reserved for future use

### 8.2.1.2 Output Size

Registers **x\_output\_size** and **y\_output\_size** (*Table 69*) control the width and height of the Visible Pixel data in the image sensor's output data frame.

- The application of registers **x\_output\_size** and **y\_output\_size** shall be re-timed internally to the start of frame boundary.
- Registers **x\_output\_size** and **y\_output\_size** shall be programmable in multiples of 4 pixels. The image sensor may have additional restrictions on output image size if indicated by bits 1 and/or 3 of register **non\_flexible\_resolution\_support**. If bit 1 is set to 1, then the Host can determine additional division rules from registers **x\_output\_div** and **y\_output\_div** which indicate whether **x\_output\_size** must be a multiple of **x\_output\_div**, and whether **y\_output\_size** must be a multiple of **y\_output\_div**. If bit 3 is set to 1, then the Host shall ensure that **x\_output\_size** is a multiple of the CSI-2 output Lane count (for the Coupled OP model, i.e. bit 1 of register **clock\_calculation** in *Table 87* is 0) or a multiple of the OP Lane count (for the Decoupled OP model, i.e. bit 1 of register **clock\_calculation** is 1) regardless of any additional size restrictions due to **x\_output\_div**. In addition, the Host shall follow the image width rules of the CSI-2 data format [*MIPI01*]. The **x\_output\_div** and **y\_output\_div** registers are mandatory with values greater than 4, and recommended with smaller values.

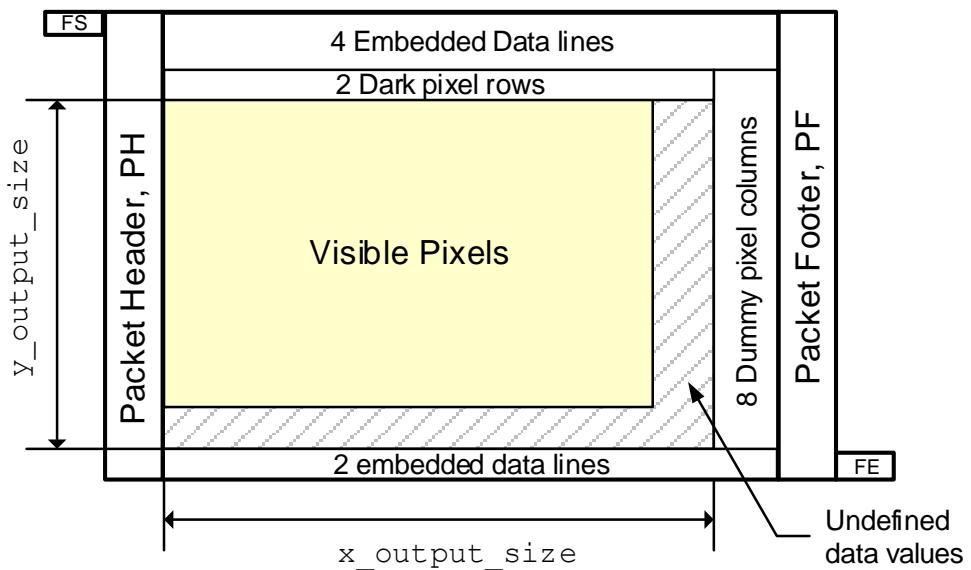
- The minimum output image size in the x-direction shall be 320 pixels.

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

- The frame format description registers shall be immediately updated whenever the **x\_output\_size** and **y\_output\_size** values change.
- When calculating the **x\_output\_size** value, the Host shall observe the size rules defined in *Section 7.3*.

*Figure 29* and *Figure 30* show example frames of image data. The information describing the number of additional columns and rows may be contained in the frame format description which is included in the first line of embedded data.

- If the **x\_output\_size** and **y\_output\_size** values are smaller than actual image size, then the image sensor shall 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, and bit 2 of **non\_flexible\_resolution\_support** is set to 0, then the output data for pixels between the actual image size and the output size (as defined by registers **x\_output\_size** and **y\_output\_size**) shall contain data with undefined values.



1674

**Figure 29 Output Sizes Larger Than Available Image Data**

1675

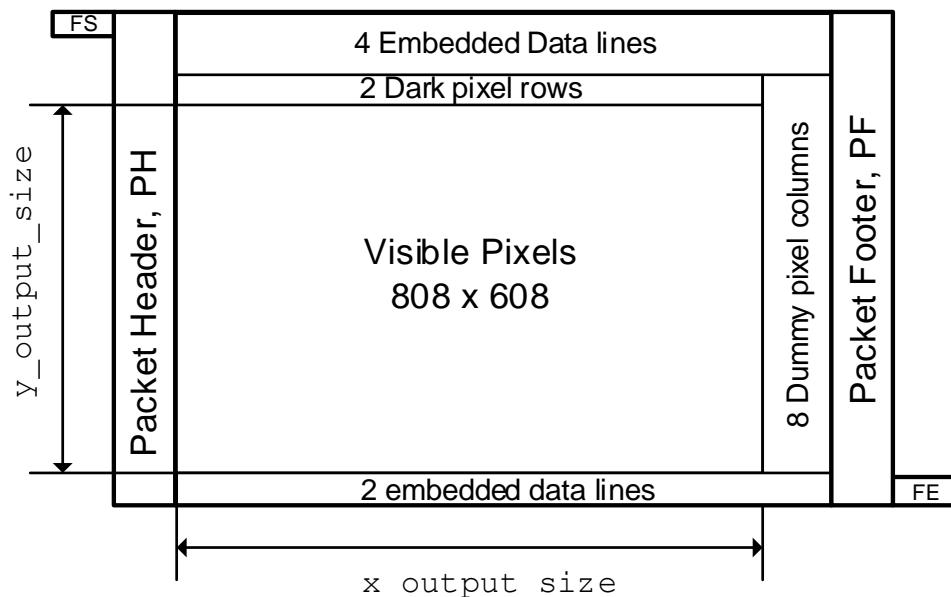
**Table 69 Output Image Size Registers**

Register Name	Type	RW	Comment
<b>x_output_size</b>	16-bit unsigned integer	RW	Width of image data output from the image sensor <b>Units:</b> Pixels <b>Default:</b> Image-sensor-specific
<b>y_output_size</b>	16-bit unsigned integer	RW	Height of image data output from the image sensor Units: Lines <b>Default:</b> Image-sensor-specific

1676

**Table 70 Output Size Limits**

Register Name	Type	RW	Comment
<b>min_x_output_size</b>	16-bit unsigned integer	RO	Minimum <b>x_output_size</b> <b>Units:</b> Pixels
<b>min_y_output_size</b>	16-bit unsigned integer	RO	Minimum <b>y_output_size</b> <b>Units:</b> Lines
<b>max_x_output_size</b>	16-bit unsigned integer	RO	Maximum <b>x_output_size</b> <b>Units:</b> Pixels
<b>max_y_output_size</b>	16-bit unsigned integer	RO	Maximum <b>y_output_size</b> <b>Units:</b> Lines
<b>x_output_div</b>	8-bit unsigned integer	RO	Additional image width rule
<b>y_output_div</b>	8-bit unsigned integer	RO	Additional image width rule



1677

**Figure 30 Output Image Size Example**

### 8.2.2 Mirror/Flip

The image sensor shall have two independent means for altering the readout order of the image data:

- Horizontally Mirrored readout.

In this mode, the columns in the pixel array are read out in reverse order.

- Vertically Flipped readout

In this mode, the rows in the pixel array are read out in reverse order.

Thus the image sensor will have four possible pixel readout orders:

1. Standard readout (**Figure 31**)
2. Horizontally Mirrored readout (**Figure 32**)
3. Vertically Flipped readout (**Figure 33**)
4. Horizontally Mirrored and Vertically Flipped readout (**Figure 34**)

The Host shall set the mirror and flip register bits correctly while in SW Standby Mode.

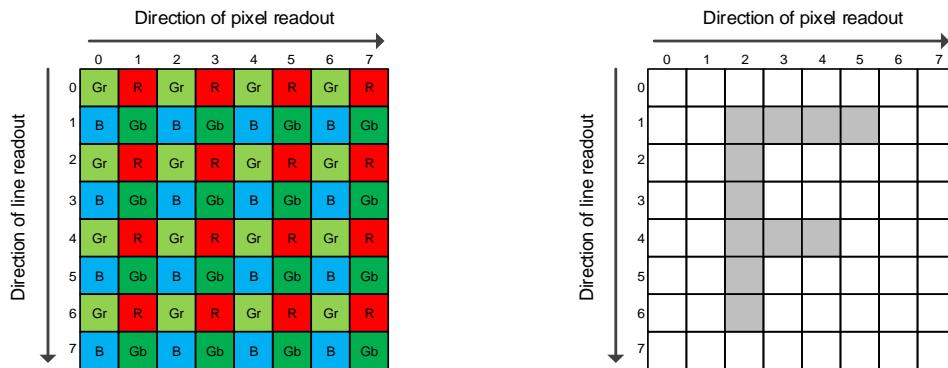


Figure 31 Standard Readout

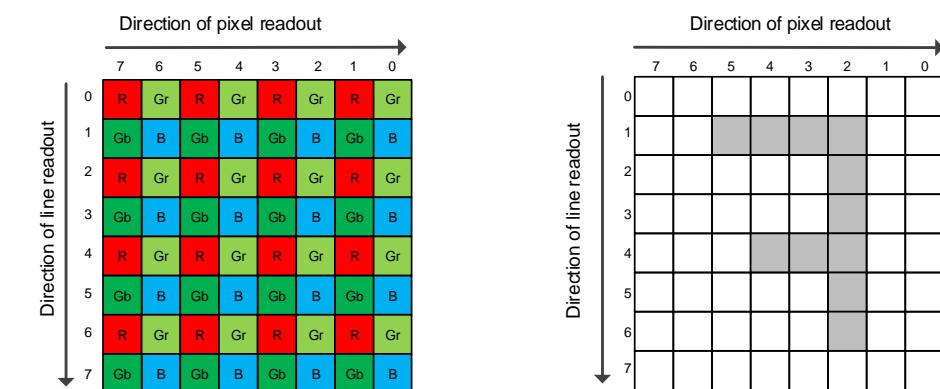
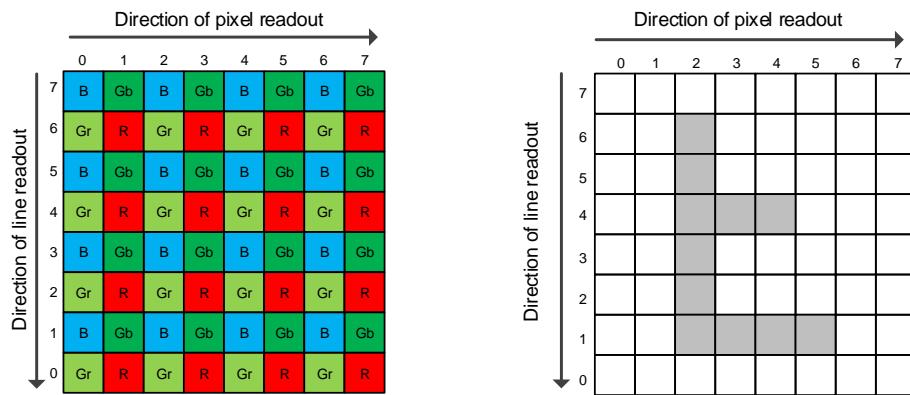
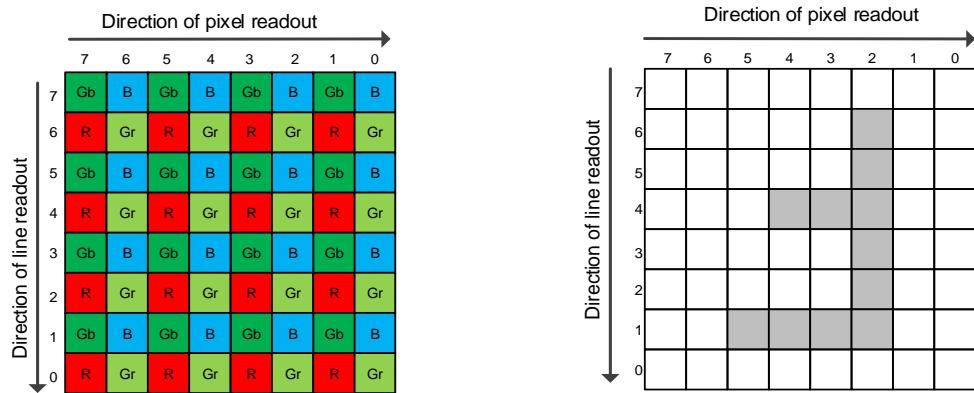


Figure 32 Horizontally Mirrored Readout



1691

**Figure 33 Vertically Flipped Readout**

1692

**Figure 34 Horizontally Mirrored and Vertically Flipped Readout**

1693

The image sensor readout order shall be controlled via register **image\_orientation** (*Table 71*).

1694

**Table 71 Image Orientation Register**

Register Name	Type	RW	Comment
<b>image_orientation</b>	8-bit unsigned integer	RW	<b>Bit 0:</b> Horizontal Mirror Enable <b>Bit 1:</b> Vertical Flip Enable <b>Default:</b> 0x00

1695

**Table 72 Contents of Image Orientation Register**

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

1696 The re-timed pixel readout order is reported by register **pixel\_order**. For Bayer pixel readout, the image sensor shall automatically update this register whenever the setting of register **image\_orientation** changes.  
1697

1698 **Table 73 Pixel Order Register (Read Only Dynamic)**

Register Name	Type	RW	Comment
<b>pixel_order</b>	8-bit unsigned integer	RO Dynamic	Defines the order of the color pixel readout Changes with mirror and flip.

1699 **Table 74 Color Pixel Order Code**

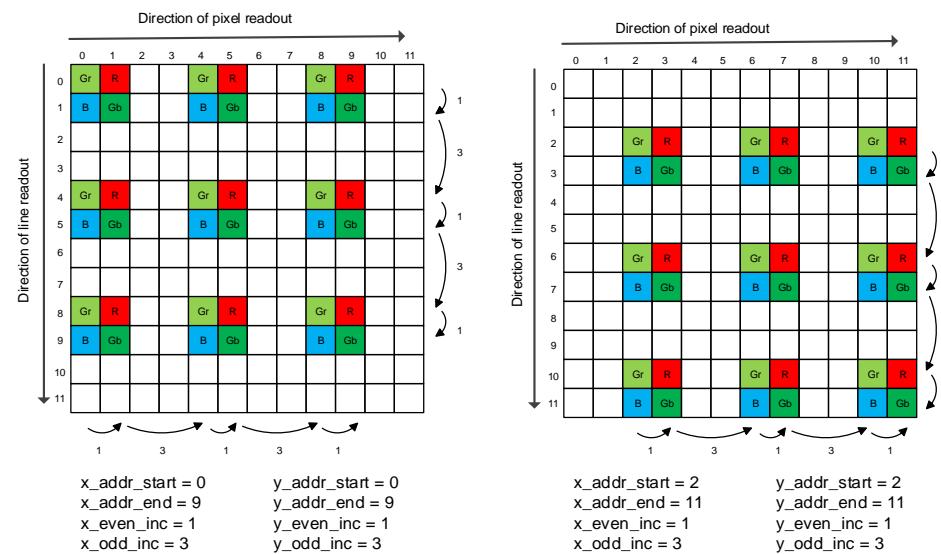
Color Pixel Order Code	Readout Order
0	GR BG
1	RG GB
2	BG GR
3	GB RG

### 8.2.3 Sub-Sampled Readout

By programming the x and y odd and even increment registers (**Table 75**), the Host can configure the image sensor to read out sub-sampled pixel data. **Figure 35** illustrates how these register's values can be programmed in order to sub-sample in the x dimension and the y dimension by a factor of 2.

**Table 75 X and Y-Address Increment Registers**

Register Name	Type	RW	Comment
<b>x_even_inc</b>	16-bit unsigned integer	RW	Increment for even pixels in the readout order: 0, 2, 4, etc. <b>Default:</b> 0x01
<b>x_odd_inc</b>	16-bit unsigned integer	RW	Increment for odd pixels in the readout order: 1, 3, 5, etc. <b>Default:</b> 0x01
<b>y_even_inc</b>	16-bit unsigned integer	RW	Increment for even pixels in the readout order: 0, 2, 4, etc. <b>Default:</b> 0x01
<b>y_odd_inc</b>	16-bit unsigned integer	RW	Increment for odd pixels in the readout order: 1, 3, 5, etc. <b>Default:</b> 0x01



**Figure 35 Sub-Sampled Bayer Pixel Readout Examples**

1705  
1706 **Table 76** lists the registers that define the minimum and maximum limits for the X and Y- address odd and even increment values used for Bayer pixel readout.

1707  
1708 **Table 76 X and Y-Address Increment Parameter Limits and Sub-sampling Capability for  
Bayer Pixel Readout**

Register Name	Type	RW	Comment
<b>min_even_inc</b>	16-bit unsigned integer	RO	Minimum Increment for even pixels
<b>max_even_inc</b>	16-bit unsigned integer	RO	Maximum increment for even pixels Required value is 1
<b>min_odd_inc</b>	16-bit unsigned integer	RO	Minimum Increment for odd pixels
<b>max_odd_inc</b>	16-bit unsigned integer	RO	Maximum Increment for odd pixels Recommended value is 3
<b>aux_subsamp_capability</b>	8-bit unsigned integer	RO	<b>Bit 0</b> 1: Bayer sub-sampling factor must be a power-of-two (i.e., 1, 2, 4, etc.) <b>Bits 1-7</b> Reserved

1709 The rules for sub-sampled Bayer pixel readout are:

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

$$sub\_sampling\_factor = (even\_inc + odd\_inc) / 2$$

1721 If bit 0 of the **aux\_subsamp\_capability** register in **Table 76** is set to 1, then the horizontal and  
1722 vertical sub-sampling factors computed using the formula above shall also be powers of two.

- The image sensor pixel readout order register value shall be immediately updated whenever any of the following values change: start address, end address, odd address increment, and/or even address increment.
- When calculating the start address, the end address, the odd address increment, and the even address increment values, the Host shall observe the CSI Data size rules in **Section 7.3**.
- The Host shall ensure that **x\_addr\_end** and **y\_addr\_end** coincide with a selected pixel (rather than a skipped one), otherwise the output data will be implementation-defined.
- The Host shall update the output image size registers according to the sub-sampling factors.

1731 Image sensors may optionally support monochrome pixel readout, either in addition to or instead of  
 1732 Bayer pixel readout, as indicated by the **pixel\_readout\_capability** register shown in **Table 77**. Note,  
 1733 for example, that it is possible for an image sensor with a Bayer CFA to support both Bayer and  
 1734 monochrome pixel readout. Monochrome readout may include monochrome sub-sampling and/or  
 1735 binning (binning is described in **Section 8.2.5**).

**Table 77 Pixel Readout Capability Register**

Register Name	Type	RW	Comment
<b>pixel_readout_capability</b>	8-bit unsigned integer	RO	Defines the supported pixel readout methods: 0: Bayer pixel readout (default) 1: Monochrome pixel readout 2: Both Bayer and Monochrome pixel readout Other values: Reserved

1736 If an image sensor supports monochrome readout, then additional details on how monochrome sub-  
 1737 sampling is supported are given by the **monochrome\_capability** register shown in **Table 78**. The Host  
 1738 enables monochrome readout by setting the **monochrome\_en** register to 1.

1739 The X- and Y-address increment control registers shown in **Table 75** for Bayer pixel readout are also  
 1740 used for monochrome readout, but parameter limits different from those shown in **Table 76** may be  
 1741 applied. When **monochrome\_en** = 1 and either **binning\_mode** = 0 or **binning\_capability** = 1 (see  
 1742 **Section 8.2.5**), the parameter limits for these registers are indicated by the **min\_even\_inc\_mono**,  
 1743 **max\_even\_inc\_mono**, **min\_odd\_inc\_mono**, and **max\_odd\_inc\_mono** registers shown in **Table 78**. In  
 1744 addition, the contents of the **monochrome\_capability** register further restrict the control register values  
 1745 permitted between the latter minimum and maximum parameter limits.

**Table 78 Monochrome Readout Capability, Sub-sampling Limits, and Control Registers**

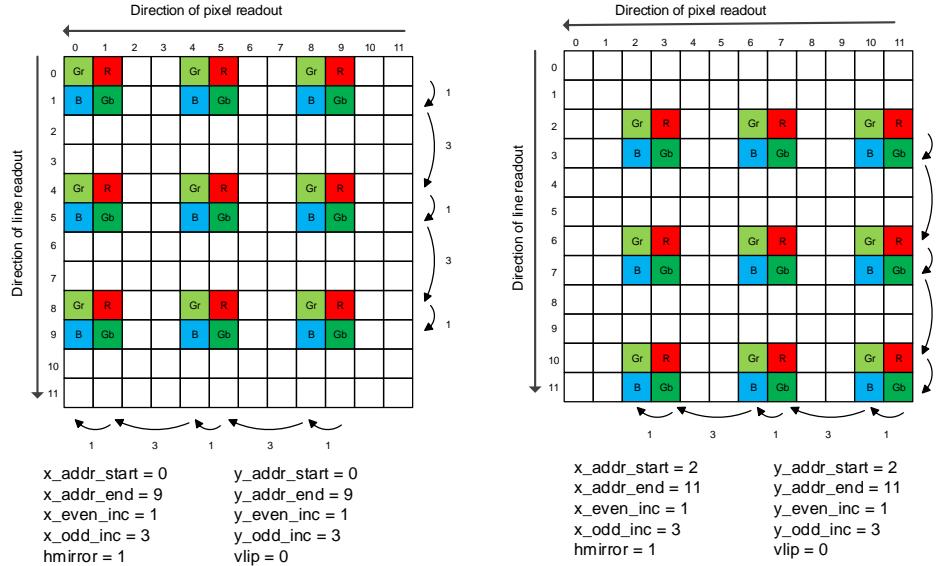
Register Name	Type	RW	Comment
<b>monochrome_capability</b>	8-bit unsigned integer	RO	Monochrome sub-sampling method: 0: All X/Y-address increment registers shall have odd values 1: All X/Y-address increment registers shall have even values (except for the required minimum value of 1); it is also required that <b>x_odd_inc_mono = x_even_inc_mono</b> and <b>y_odd_inc_mono = y_even_inc_mono</b> Other Values: Reserved
<b>min_even_inc_mono</b>	16-bit unsigned integer	RO	Minimum value of <b>x_even_inc</b> and <b>y_even_inc</b> Required value is 1
<b>max_even_inc_mono</b>	16-bit unsigned integer	RO	Maximum value of <b>x_even_inc</b> and <b>y_even_inc</b> Required value is 1 for <b>monochrome_capability = 0</b>
<b>min_odd_inc_mono</b>	16-bit unsigned integer	RO	Minimum value of <b>x_odd_inc</b> and <b>y_odd_inc</b> Required value is 1
<b>max_odd_inc_mono</b>	16-bit unsigned integer	RO	Maximum value of <b>x_odd_inc</b> and <b>y_odd_inc</b>
<b>aux_subsamp_mono_capability</b>	8-bit unsigned integer	RO	<b>Bit 0</b> 1: Monochrome sub-sampling factor must be a power-of-two (i.e., 1, 2, 4, etc.) <b>Bits 1-7</b> Reserved
<b>monochrome_en</b>	8-bit unsigned integer	RW	Monochrome readout enable 0: Disabled (default) 1: Enabled

1746 The rules listed above for sub-sampled Bayer pixel readout also apply to monochrome pixel readout, but with  
1747 the following exceptions:

- 1748 • The **x\_even\_inc** and **x\_odd\_inc** registers shall both be set to either even values or odd values, as  
1749 indicated by the contents of **monochrome\_capability**; similarly, the **y\_even\_inc** and **y\_odd\_inc**  
1750 registers shall both be set to either even values or odd values. For example, setting **x\_even\_inc** =  
1751 1 and **x\_odd\_inc** = 2 is not permitted because one value is odd and the other is even.
- 1752 • During monochrome readout, X-addresses are incremented by the horizontal sub-sampling factor  
1753 given by  $(x_{even\_inc} + x_{odd\_inc}) / 2$  and Y-addresses are incremented by the vertical sub-  
1754 sampling factor given by  $(y_{even\_inc} + y_{odd\_inc}) / 2$ . Therefore, setting  
1755 **x\_even\_inc** = **x\_odd\_inc** = 1 disables horizontal sub-sampling, and setting  
1756 **y\_even\_inc** = **y\_odd\_inc** = 1 disables vertical sub-sampling. If bit 0 of the  
1757 **aux\_subsamp\_mono\_capability** register in *Table 78* is set to 1, then the horizontal and vertical  
1758 sub-sampling factors shall also be powers of two.
- 1759 • Permissible values of the X- and Y-address increment control registers during monochrome  
1760 readout are indicated by the capability and parameter limits registers shown in *Table 78*. However,  
1761 unlike Bayer pixel readout, the contents of the even and odd increment registers aren't required to  
1762 be actual address increment values for even and odd pixel addresses.
- 1763 • The image sensor pixel readout order register may or may not be updated as determined by the  
1764 image sensor datasheet.
- 1765 • As with Bayer pixel readout, the Host shall ensure that **x\_addr\_end** and **y\_addr\_end** coincide  
1766 with a selected pixel (rather than a skipped one), otherwise the output data will be implementation-  
1767 defined. However, **x\_addr\_end** and **y\_addr\_end** may be either even or odd numbers.

### 8.2.4 Sub-Sampling with Mirror and Flip

Bayer and monochrome pixel sub-sampling can also be used with mirror and flip. The *Figure 35* represents non-mirrored and non-flipped Bayer pixel readout, and *Figure 36* represents the mirrored readout versions of those. The image sensor shall interpret the values as illustrated below:

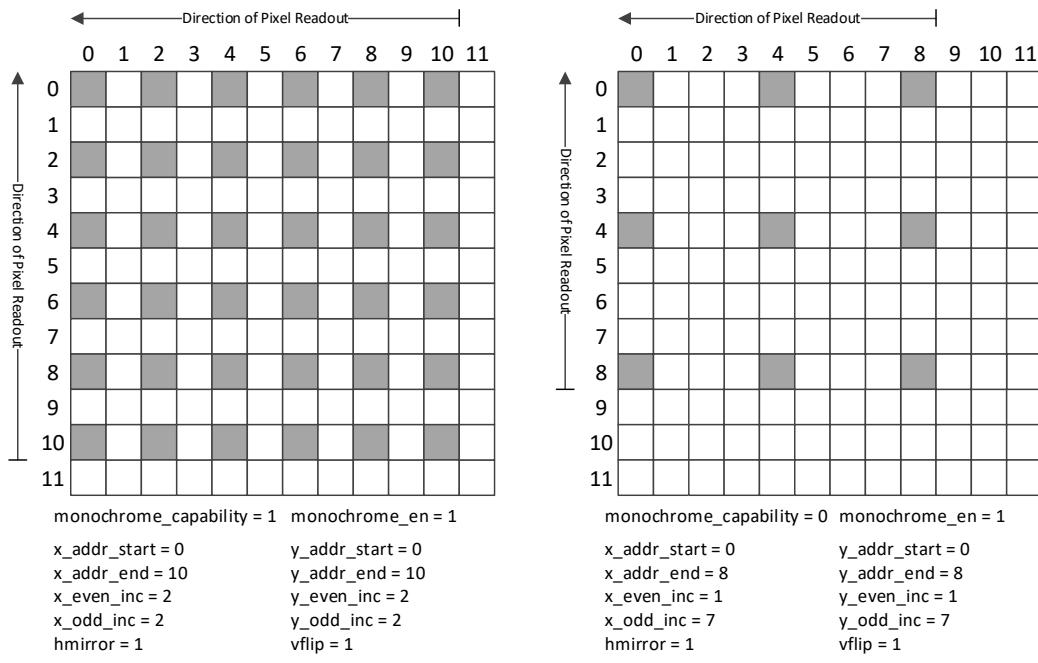


**Figure 36** Bayer Pixel Sub-Sampling Behavior with Hmirror

The same principle applies for Vflip and combined Hmirror/Vflip. The Host shall ensure that **x\_addr\_start** and **y\_addr\_start** coincide with a selected pixel (rather than a skipped one) when mirror and/or flip is used, otherwise the output data will be implementation-defined. Note that *Section 8.2.3* defines that the Host shall ensure that **x\_addr\_end** and **y\_addr\_end** coincide with a selected pixel, thus the Host shall ensure that all **x\_addr\_start**, **y\_addr\_start**, **x\_addr\_end**, and **y\_addr\_end** coincide with a selected pixel.

1777 Monochrome pixel sub-sampling examples with mirror and flip are illustrated in **Figure 37**. In example (a)  
 1778 2x sub-sampling is performed, so the output resolution is one half of the input resolution in both the horizontal  
 1779 and vertical directions. Even numbers are used for the X- and Y-address odd- and even-pixel increment values  
 1780 (i.e., **monochrome\_capability** = 2; see **Table 78**). The horizontal and vertical sub-sampling factor is  
 1781  $(2+2) / 2 = 2$ , so every other pixel is output in both dimensions.

1782 Example (b) illustrates 4x sub-sampling. In this example, odd numbers are used for the X- and Y-address  
 1783 odd- and even-pixel increment values (i.e., **monochrome\_capability** = 1; see **Table 78**). The horizontal and  
 1784 vertical sub-sampling factor is  $(1+7) / 2 = 4$ , so every fourth pixel is output in both dimensions.



1785

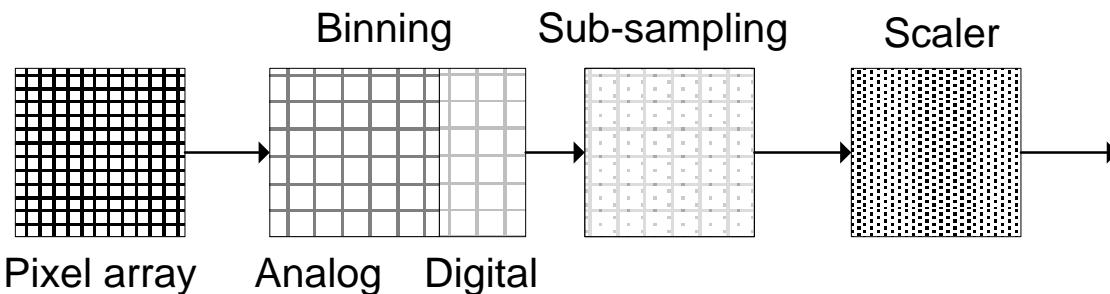
**Figure 37 Monochrome Pixel Sub-Sampling Examples with Hmirror and Vflip**

### 8.2.5 Binning Readout

Binning reduces image sensor resolution by combining adjacent pixels that have the same color. This has the advantage of being able to produce a full field of view output image at a higher frame rate than using a scaler, and at a higher image quality than using subsampling. The binned output may be a combination of analog binning and digital binning.

Instead of a separate control and configuration for the analog binning and the digital binning, they will be treated as a single combined binning function.

Binning shall be able to work in conjunction with sub-sampling, or with a scaler (if present), to achieve the desired output image size.



**Figure 38 Model for Scaling Features (for binning\_capability = 1)**

(Some May Not be Present)

#### 8.2.5.1 Binning Capability Registers

The presence of the binning function and the binning subtypes shall be indicated in capability registers (see **Table 79**). Bayer and/or monochrome pixel binning may be supported using one of two basic binning schemes, as indicated by the **binning\_capability** register. Registers **binning\_sub\_types** and **binning\_sub\_types\_mono** shall indicate the number of binning subtypes available in the image sensor for Bayer and monochrome binning, respectively.

Registers **binning\_type\_(n)** and **binning\_type\_(n)\_mono** for Bayer and monochrome binning, respectively, shall indicate the integer downsampling factors for binning type *n*, where the MS 4 bits specify the column binning factor and the LS 4 bits specify the row binning factor.

Registers **binning\_weighting\_capability** and **binning\_weighting\_mono\_capability** for Bayer and monochrome binning, respectively, specify the possible binning options. For example, a **binning\_weighting\_capability** value of 0x07 (0b00000111) indicates that the image sensor is capable of averaged, summed, and Bayer corrected output. The image sensor datasheet should provide additional details on how the supported binning weighting options are implemented.

1809

**Table 79 Binning Capability Register**

Register Name	Type	RW	Comment
<b>binning_capability</b>	8-bit unsigned integer	RO	0: Binning not supported 1: Binning precedes sub-sampling and is functionally independent of sub-sampling 2: Sub-sampling precedes binning and is functionally integrated with binning
<b>Applicable to Bayer Pixel Binning (for pixel_readout_capability = 0 or 2)</b>			
<b>binning_sub_types</b>	8-bit unsigned integer	RO	N: Number of binning subtypes available 0 implies Bayer binning is not supported
<b>binning_type_1</b>	8-bit unsigned integer	RO	<b>Example</b> 0x21 = Column x 2, Row x 1
<b>binning_type_2</b>	8-bit unsigned integer	RO	<b>Example</b> 0x22 = Column x 2, Row x 2
...	(Use as many types as supported, up to a limit of 64)		
<b>binning_type_63</b>	8-bit unsigned integer	RO	<b>Example</b> 0x31 = Column x 3, Row x 1
<b>binning_type_64</b>	8-bit unsigned integer	RO	<b>Example</b> 0x44 = Column x 4, Row x 4
<b>binning_weighting_capability</b>	8-bit unsigned integer	RO	<b>Bit 0</b> 1: Averaged <b>Bit 1</b> 1: Summed (for low light) <b>Bit 2</b> 1: Bayer-corrected <b>Bit 3</b> 1: Module specific weighting
<b>Applicable to Monochrome Pixel Binning (for pixel_readout_capability = 1 or 2)</b>			
<b>binning_sub_types_mono</b>	8-bit unsigned integer	RO	N: Number of binning subtypes available 0 implies monochrome binning is not supported
<b>binning_type_1_mono</b>	8-bit unsigned integer	RO	<b>Example</b> 0x21 = Column x 2, Row x 1
<b>binning_type_2_mono</b>	8-bit unsigned integer	RO	<b>Example</b> 0x22 = Column x 2, Row x 2
...	(Use as many types as supported, up to a limit of 64)		
<b>binning_type_63_mono</b>	8-bit unsigned integer	RO	<b>Example</b> 0x31 = Column x 3, Row x 1

Register Name	Type	RW	Comment
<b>binning_type_64_mono</b>	8-bit unsigned integer	RO	<b>Example</b> 0x44 = Column x 4, Row x 4
<b>binning_weighting_mono_capability</b>	8-bit unsigned integer	RO	<b>Bit 0</b> 1: Averaged <b>Bit 1</b> 1: Summed (for low light) <b>Bit 2</b> 1: Bayer-corrected <b>Bit 3</b> 1: Module specific weighting

When binning is enabled (i.e. **binning\_mode** = 1; see *Section 8.2.5.2*) with **binning\_capability** = 2, the contents of the X- and Y-address increment control registers are constrained by the min/max parameter limit registers in *Table 80* rather than the corresponding registers in *Table 76* and *Table 78*. When **binning\_mode** = 0 or **binning\_capability** = 1, the parameter limits in *Table 76* and *Table 78* apply instead. The **aux\_subsamp\_capability** and **aux\_subsamp\_mono\_capability** registers in *Table 76* and *Table 78*, respectively, also apply to the Bayer and monochrome X- and Y-address increment parameter limits in *Table 80*.

**Table 80 X and Y-Address Increment Parameter Limits for binning\_capability = 2**

Register Name	Type	RW	Comment
<b>Applicable to Bayer Pixel Binning:</b>			
<b>min_even_inc_bc2</b>	16-bit unsigned integer	RO	Minimum Increment for even pixels
<b>max_even_inc_bc2</b>	16-bit unsigned integer	RO	Maximum increment for even pixels Required value is 1
<b>min_odd_inc_bc2</b>	16-bit unsigned integer	RO	Minimum Increment for odd pixels
<b>max_odd_inc_bc2</b>	16-bit unsigned integer	RO	Maximum Increment for odd pixels Recommended value is 3
<b>Applicable to Monochrome Pixel Binning:</b>			
<b>min_even_inc_mono_bc2</b>	16-bit unsigned integer	RO	Minimum value of <b>x_even_inc</b> and <b>y_even_inc</b> Required value is 1
<b>max_even_inc_mono_bc2</b>	16-bit unsigned integer	RO	Maximum value of <b>x_even_inc</b> and <b>y_even_inc</b> Required value is 1 for <b>monochrome_capability</b> = 1
<b>min_odd_inc_mono_bc2</b>	16-bit unsigned integer	RO	Minimum value of <b>x_odd_inc</b> and <b>y_odd_inc</b> Required value is 1
<b>max_odd_inc_mono_bc2</b>	16-bit unsigned integer	RO	Maximum value of <b>x_odd_inc</b> and <b>y_odd_inc</b>

### 8.2.5.2 Binning Enable Control

The Binning function shall be enabled with control register **binning\_mode** (*Table 81*). This register shall be re-timed to take effect at a predictable point. Monochrome binning is selected if the **monochrome\_en** register (*Table 81*) is set to 1.

**Table 81 Binning Mode Register**

Register Name	Type	RW	Comment
<b>binning_mode</b>	8-bit unsigned integer	RW	0: None 1: Enabled <b>Default:</b> 0

### 8.2.5.3 Binning Configuration Registers

The binning type and weighting shall be selected for the supported types (see *Section 8.2.5.1*) by registers **binning\_type** and **binning\_weighting** (*Table 82*).

**Table 82 Binning Control Registers**

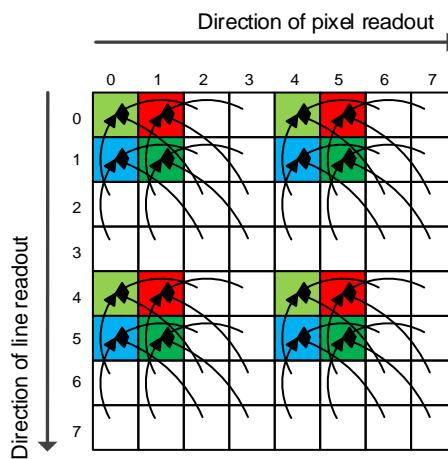
Register Name	Type	RW	Comment
<b>binning_type</b>	8-bit unsigned integer	RW	<b>Bits 0-3:</b> Row Binning factor <b>Bits 4-7:</b> Column Binning factor <b>Default:</b> 0x11 (1 Row, 1 Column)
<b>binning_weighting</b>	8-bit unsigned integer	RW	0x00: Averaged weighting 0x01: Summed weighting 0x02: Bayer weighting 0x03: Image sensor-specific weighting <b>Default:</b> Image-sensor-specific, depending on capabilities (0x00, 0x01, 0x02, or 0x03)

### 1825 8.2.5.4 Bayer Binning Examples

1826 *Figure 39* generically illustrates 2x2 Bayer pixel binning, where 4 pixels are combined together. For  
 1827 **binning\_capability** = 1, this example corresponds to the following register settings:

```
1828 binning_type = 0x22
1829 x_addr_start = 0
1830 x_addr_end = 7
1831 y_addr_start = 0
1832 y_addr_end = 7
1833 hmirror = vflip = 0
1834 x_even_inc = x_odd_inc = y_even_inc = y_odd_inc = 1
```

1835 The latter four registers are all set to 1 based on the assumption that the output of the binning function is not  
 1836 further sub-sampled.

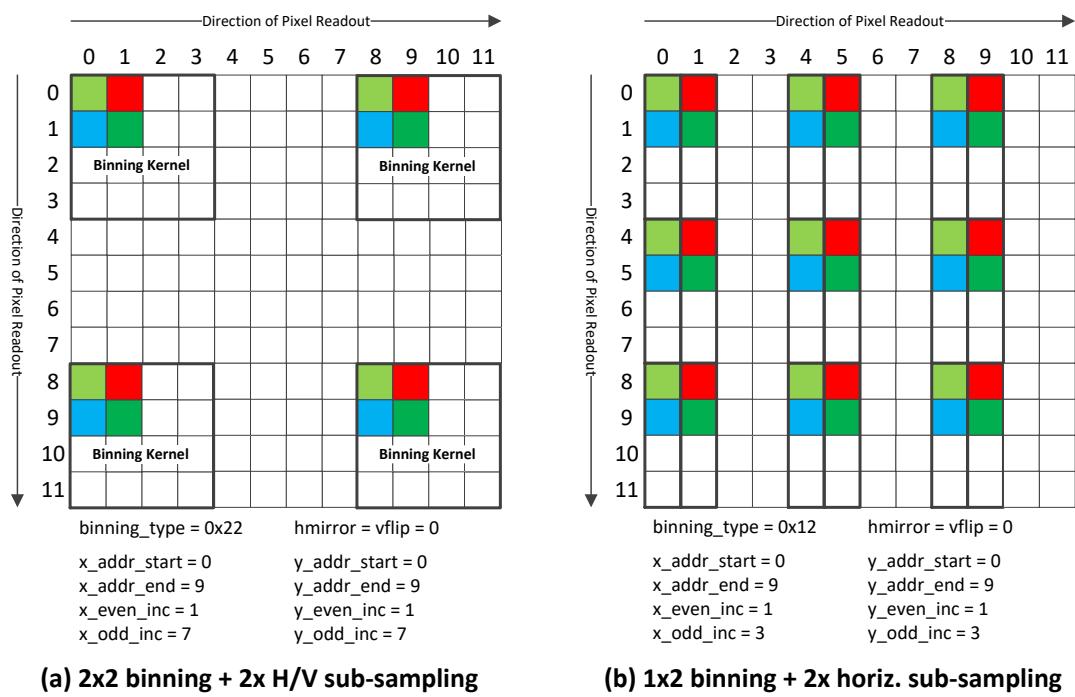


1837 **Figure 39 Example of Binning**

1838 For **binning\_capability** = 2, this example also corresponds to **binning\_type** = 0x22, **hmirror** = **vflip** = 0, and  
 1839 **x\_addr\_start** = **y\_addr\_start** = 0; however, **x\_addr\_end** = **y\_addr\_end** = 5 because binning is integrated  
 1840 with the sub-sampling function, and **x\_addr\_end** and **y\_addr\_end** must both coincide with a selected rather  
 1841 than skipped pixel. Also, the X- and Y-address increment registers are set to **x\_even\_inc** = 1, **x\_odd\_inc** =  
 1842 3, **y\_even\_inc** = 1, and **y\_odd\_inc** = 3 because for **binning\_capability** = 2, the X- and Y-address increment  
 1843 registers serve to position the upper left pixel of the complete 4x4 binning kernel as it is stepped across the  
 1844 image.

1845 Two additional Bayer pixel binning examples are shown below in *Figure 40* for **binning\_capability** = 2. In  
 1846 example (a), 2x2 binning is performed on every other 4x4 pixel group. This is equivalent to 2x2 binning for  
 1847 **binning\_capability** = 1 in which the binning function output is further sub-sampled by setting  
 1848 **x\_even\_inc** = 1, **x\_odd\_inc** = 3, **y\_even\_inc** = 1, and **y\_odd\_inc** = 3. Example (b) illustrates 1x2 vertical  
 1849 binning performed in conjunction with 2x horizontal sub-sampling.

1850 Note that the address start/end and increment registers in both examples of *Figure 40* remain unchanged if  
 1851 **hmirror** or **vflip** is 1; i.e., each binning kernel processes the same pixels regardless of the horizontal or vertical  
 1852 readout direction.



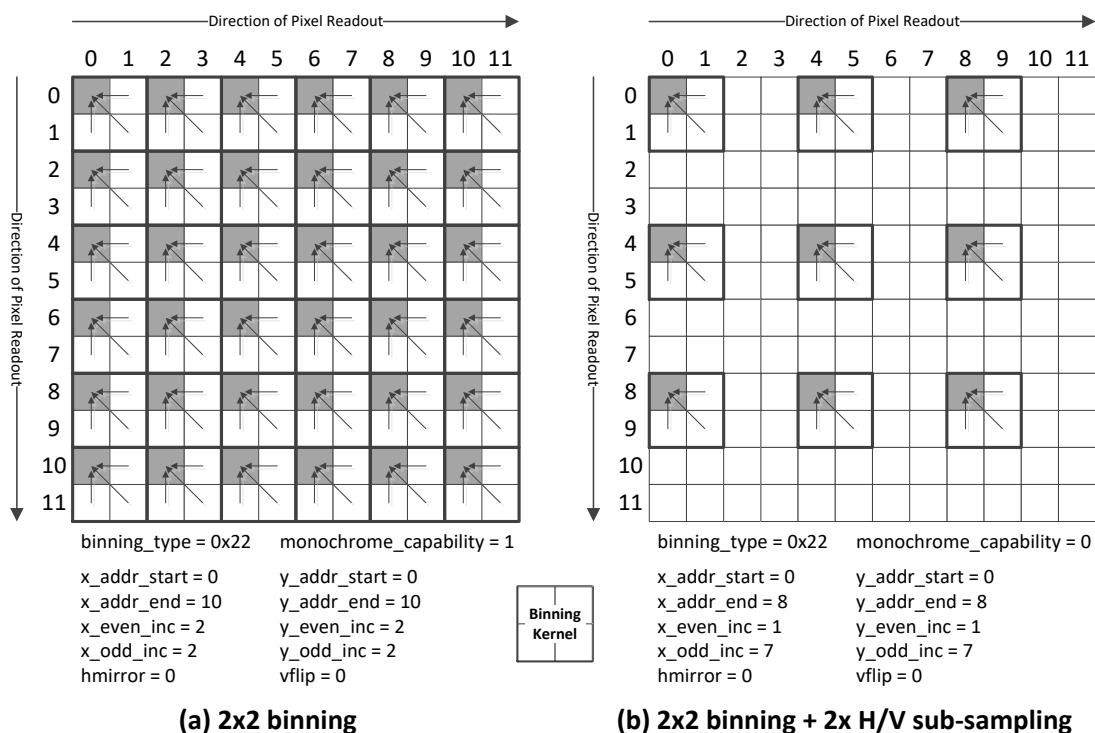
1853

**Figure 40** Bayer Pixel Binning Examples for **binning\_capability = 2**

### 8.2.5.5 Monochrome Binning Examples

Two monochrome pixel 2x2 binning examples are shown below in **Figure 41** for **binning\_capability** = 2. As shown by the arrows, 2x2 groups of adjacent pixels are combined into single pixels. In example (a), binning is performed on every 2x2 pixel group, so the output resolution is one half of the input resolution both horizontally and vertically. Even numbers are used for the X- and Y-address odd- and even-pixel increment values (i.e. **monochrome\_capability** = 1; see **Table 78**). The horizontal and vertical sub-sampling factor is  $(2+2) / 2 = 2$ , so the 2x2 monochrome binning kernel is stepped across the image in increments of two pixels in both dimensions.

Example (b) illustrates 2x2 binning performed in conjunction with 2x horizontal and vertical sub-sampling. In this example, odd numbers are used for the X- and Y-address odd- and even-pixel increment values (i.e. **monochrome\_capability** = 0; see **Table 78**). The horizontal and vertical sub-sampling factor is  $(1+7) / 2 = 4$ , so the 2x2 monochrome binning kernel is stepped across the image in increments of four pixels in both dimensions.



**Figure 41 Monochrome Pixel Binning with binning\_capability = 2**

The examples in **Figure 41** may also be supported with **binning\_capability** = 1 if a few modifications are made. With example (a), one must set **x\_addr\_end** = **y\_addr\_end** = 11 and **x\_even\_inc** = **x\_odd\_inc** = **y\_even\_inc** = **y\_odd\_inc** = 1, and with example (b), one must set **x\_addr\_end** = **y\_addr\_end** = 9, **x\_even\_inc** = **y\_even\_inc** = 1, and **x\_odd\_inc** = **y\_odd\_inc** = 3. The other register settings in **Figure 41** remain the same.

### 8.2.6 Line Length and Frame Length

The length of a line is specified in terms of a number of pixel clocks: **line\_length\_pck**. The length of a frame is specified as a number of lines: **frame\_length\_lines**. **Line\_length\_pck** indicates the overall number of pixels per line.

The Host shall program the values so that if the system speed model is used, then **line\_length\_pck** shall be based on **vt\_pix\_clk\_freq\_mhz**; and if the Lane speed model is used, then **line\_length\_pck** shall be based on **vt\_pix\_clk\_freq\_mhz \* num\_of\_vt\_lanes**. See *Section 8.3* for additional information on clock calculation models.

Register **min\_line\_length\_pck\_step\_size** defines the minimum increment by which the Host shall adjust **line\_length\_pck**. The origin for this adjustment is the value of **min\_line\_length\_pck**. The valid values for **line\_length\_pck** are given by the following formula, where n is an integer:

$$\text{min\_line\_length\_pck} + (n * \text{min\_line\_length\_pck\_step\_size})$$

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

**Table 83 Video Timing Registers**

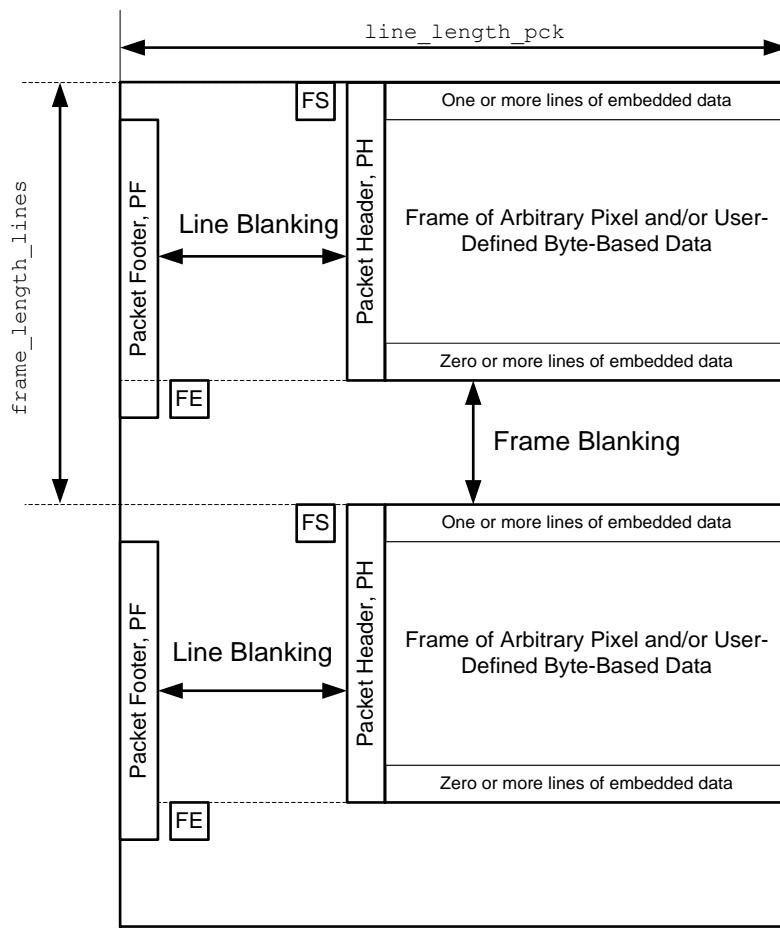
Register Name	Type	RW	Comment
<b>frame_length_lines</b>	16-bit unsigned integer	RW	Frame Length <b>Units:</b> Lines <b>Default:</b> Image-sensor-specific
<b>line_length_pck</b>	16-bit unsigned integer	RW	Line Length <b>Units:</b> VT Pixel Clocks <b>Default:</b> Image-sensor-specific

In order to aid Host set-up, the read-only and static frame timing parameter limit registers (*Table 84*) define the image sensor's minimum and maximum possible frame length and line length. These values are not dependent on x start, y start, x end and y end.

Basic operation mode has following rules; note also the exceptions in *Section 9.7*:

- The image sensor shall NOT internally clip the frame and line length register values.
- The Host shall read the minimum and maximum values, and shall not exceed this range.

*Figure 26* illustrates the relationship between these controls (**frame\_length\_lines** and **line\_length\_pck**) and the image sensor functions. Note that **frame\_length\_lines** and **line\_length\_pck** apply in the VT clock domain, and may not necessarily have direct impact on OP timings, depending on the clock tree and PLL configurations being used. *Figure 42* illustrates **frame\_length\_lines** and **line\_length\_pck** for cases where the VT pixel clock and the OP pixel clock have the same frequency.

**KEY:**

PH – Packet Header

FS – Frame Start

PF – Packet Footer

FE – Frame End

1896

**Figure 42 Frame Length/Line Length Definitions**

1897

**Table 84 Frame Timing Parameter Limits**

Register Name	Type	RW	Comment
<b>min_frame_length_lines</b>	16-bit unsigned integer	RO	Minimum Frame Length allowed. <b>Units:</b> Lines
<b>max_frame_length_lines</b>	16-bit unsigned integer	RO	Maximum possible number of lines per Frame. <b>Units:</b> Lines
<b>min_line_length_pck</b>	16-bit unsigned integer	RO	Minimum Line Length allowed. <b>Units:</b> Pixel Clocks
<b>max_line_length_pck</b>	16-bit unsigned integer	RO	Maximum possible number of pixel clocks per line. <b>Units:</b> Pixel Clocks
<b>min_line_length_pck_step_size</b>	8-bit unsigned integer	RO	Minimum step size of line length pck. Typical value may be 1, 2, or 4. <b>Units:</b> Pixel Clocks
<b>min_line_blanketing_pck</b>	16-bit unsigned integer	RO	Minimum line blanking time in pixel clocks <b>Units:</b> Pixel Clocks

1898

Some internal operations (e.g. dark calibration) may require a minimum frame blanking time to be specified. This limit is exposed in the register **min\_frame\_blanketing\_lines** register (*Table 85*).

1899

**Table 85 Frame Blanking Register**

Parameter Name	Type	RW	Function
<b>min_frame_blanketing_lines</b>	16-bit unsigned integer	RO	Minimum number of VT domain frame blanking lines required.

1900

In binning mode, the image sensor may have different frame timing limits for the parameters identified in *Table 86*. No other parameters have variable limits in binning mode.

1901

An image sensor whose binning mode limits (as listed in *Table 86*) all have the same values as for non-binning mode (see *Table 84*) may report the value 0 for all of the binning limit registers listed in *Table 86*. The Host can detect this scenario by checking whether the value of register **max\_frame\_length\_lines\_bin** is zero (same limits), vs. non-zero (binning limits are different).

1902

1903

1904

1905

1906

1907

**Table 86 Frame Timing Parameter Limits for Binning mode**

Register Name	Type	RW	Comment
<b>min_frame_length_lines_bin</b>	16-bit unsigned integer	RO	Minimum Frame Length allowed in binning mode <b>Units:</b> Lines
<b>max_frame_length_lines_bin</b>	16-bit unsigned integer	RO	Maximum possible number of lines per Frame in binning mode <b>Units:</b> Lines
<b>min_line_length_pck_bin</b>	16-bit unsigned integer	RO	Minimum Line Length allowed in binning mode <b>Units:</b> Pixel Clocks
<b>max_line_length_pck_bin</b>	16-bit unsigned integer	RO	Maximum possible number of pixel clocks per line in binning mode <b>Units:</b> Pixel Clocks
<b>min_line_blanketing_pck_bin</b>	16-bit unsigned integer	RO	Minimum line blanking time in pixel clocks in binning mode <b>Units:</b> Pixel Clocks
<b>fine_integration_time_min_bin</b>	16-bit unsigned integer	RO	Minimum fine Integration Time allowed in binning mode <b>Units:</b> Pixels
<b>fine_integration_time_max_margin_bin</b>	16-bit unsigned integer	RO	Margin used to determine the maximum fine Integration Time allowed in binning mode. <b>Units:</b> Pixels

1908 The frame length and line length register values shall be re-timed internally to the start of a frame.

### 8.3 Video Timing and Output Pixel Clock Frequency Control

This Specification supports a few clock tree options. The image sensor shall support at least one of these options.

It is important that the Host can precisely calculate and control the VT pixel clock frequency (and thus the actual Exposure time used) from the clock tree registers.

The clock trees have following properties:

- The image sensor shall support at least either a 1-PLL based clock tree, or a 2-PLL based clock tree. The clock trees, including clock dividers, are specified in this Section.
  - The image sensor shall support one of two clock calculation models, either:
    - System Speed Model, or
    - Lane Speed Model (recommended for image sensors supporting multiple CSI-2 Lanes)
- The related capability registers are defined in this Section.
- The image sensor shall support either Coupled or Decoupled OP domain calculation:
    - In Coupled mode, OP domain in Lane speed model assumes to have as many OP Lanes as there are CSI-2 Lanes.
    - In Decoupled mode, there is no tight coupling between OP Lanes and CSI-2 Lanes; both must be taken into account in calculations.
  - The Host system can detect which clock tree is supported by inspecting capability registers **clock\_tree\_pll\_capability** and **clock\_calculation**. The Host can detect image sensor support of simple 1-PLL mode by finding the default value of 0 in register **op\_pix\_clk\_div**, and that register **clock\_tree\_pll\_capability** reports 1-PLL.
  - If the image sensor supports both 1-PLL Mode and 2-PLL Mode, then the Host shall select the desired option with register **pll\_mode** after power-up, while in SW Standby Mode. If only one of the modes is supported, then implementing the register **pll\_mode** is optional.
  - If the image sensor supports the Lane Speed Model, then the Host system can determine the number of supported VT Lanes from register **num\_of\_vt\_lanes**.
  - If the image sensor supports the Decoupled OP Model, then the Host system can determine the number of supported OP Lanes from the register **num\_of\_op\_lanes**.
  - If the image sensor supports output pixel bit depths which are greater than the number of bits per OP Lane, then the Host can read the number of bits per OP Lane from register **op\_bits\_per\_lane** and use it to determine which pixel bit depths require two OP pixel clocks per pixel. This information is useful for ensuring that the proper value is programmed in register **op\_pix\_clk\_div**.

**Table 87 PLL Control and Capability Registers**

Register Name	Type	RW	Comment
<b>clock_calculation</b>	8-bit unsigned integer	RO	<p><b>Bit 0</b> 0: Clock Calculation based on System speed 1: Clock Calculation based on Lane speed</p> <p><b>Bit 1</b> 0: OP Clock calculation Coupled with link 1: OP Clock calculation Decoupled from link</p> <p><b>Bit 2</b> with 2-PLL clock tree for OP sys domain 0: Single rate bitrate / Symbol rate 1: Double rate bitrate / Symbol rate</p> <p><b>Bit 3</b> with 2-PLL clock tree for OP pixel domain 0: Single rate 1: Double rate</p>
<b>clock_tree_pll_capability</b>	8-bit unsigned integer	RO	<p><b>Bit 0</b> 0: 2-PLL clock tree not supported 1: 2-PLL clock tree supported</p> <p><b>Bit 1</b> 0: 1-PLL clock tree not supported 1: 1-PLL clock tree supported</p> <p><b>Bit 2</b> 0: Extended PLL input divider values not supported 1: Extended PLL input divider values supported</p> <p><b>Bit 3</b> 0: Legacy <b>OP_pix_clk_div</b> values supported 1: Flexible <b>OP_pix_clk_div</b> values supported</p>
<b>num_of_vt_lanes</b>	8-bit unsigned integer	RO	Number of VT Lanes Effective if <b>clock_calculation</b> bit 0 has value 1
<b>num_of_op_lanes</b>	8-bit unsigned integer	RO	Number of OP Lanes Effective if <b>clock_calculation</b> bit 1 has value 1
<b>op_bits_per_lane</b>	8-bit unsigned integer	RO	Number of bits per OP Lane Required if the number of bits per OP Lane is less than the maximum number of bits per pixel
<b>pll_mode</b>	8-bit unsigned integer	RW	<p><b>Bit 0</b> 0: 1 PLL in use 1: 2 PLLs in use</p> <p><b>Default:</b> Image-sensor-specific</p>

### 8.3.1 Introduction to Clock Trees

The image sensor shall support one of the clock trees. The properties of clock trees are described in this Section. An image sensor contains one or two PLL (Phase Locked Loop) blocks, which generate all the necessary internal clocks from the external clock input.

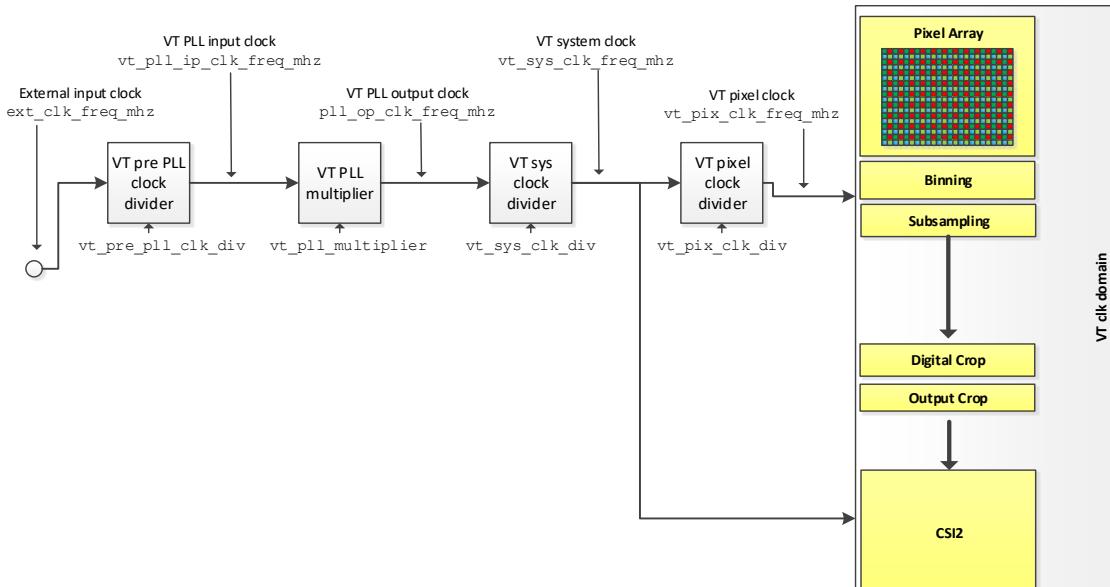
Different clock trees are described in following figures. They show the internal functional blocks, which define the relationship between the external input clock frequency and the pixel clock frequencies.

- **Figure 43** describes a simple 1-PLL system. There CSI-2 speed is coupled to VT domain speed. OP domain does not exist. This model is typically used in simplest image sensors often using only single CSI-2 lane.
- **Figure 44** describes a 1-PLL system with the System Speed Model. The clock tree consists of a PLL, and separate VT and OP clock domains.
- **Figure 46** describes a 2-PLL system with the System Speed Model. The clock tree consists of 2 PLLs, one controlling the VT clock domain and the other controlling the OP clock domain.
- **Figure 47** describes a 2-PLL system with the Lane Speed Model. The clock tree consists of 2 PLLs, one controlling the VT clock domain and the other controlling the OP clock domain.
- **Figure 45** describes a 1-PLL system with the Lane Speed Model. The clock tree consists of a PLL, and separate VT and OP clock domains.

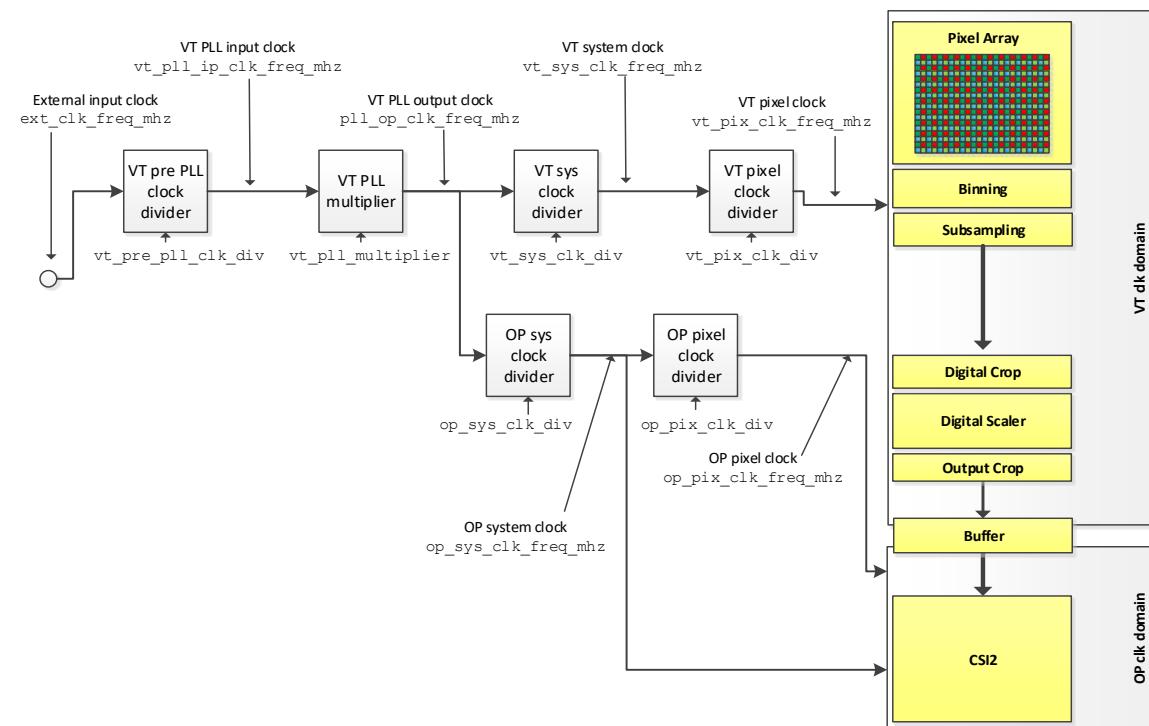
Line length, min line blanking time, and fine and coarse Exposure values are all related to the video timing pixel clock.

In a simple 1-PLL system, the video timing system and pixel clock are also used for the CSI-2 transmitter.

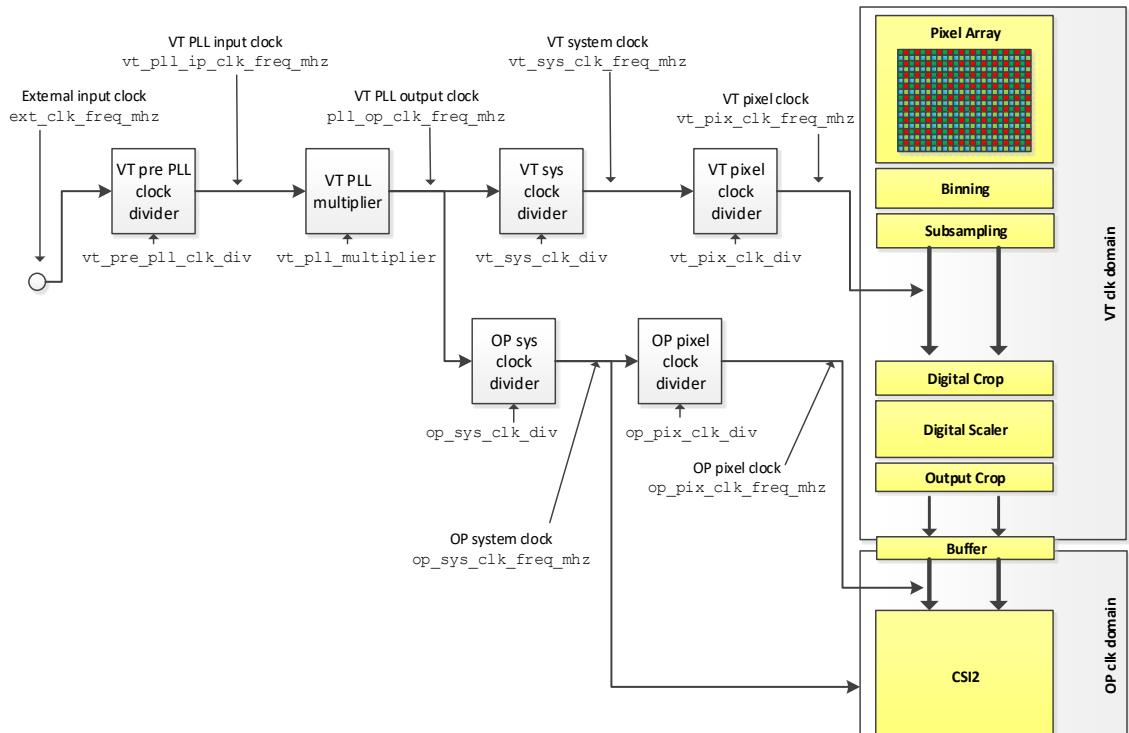
A normal 1-PLL system has separate domains for video timing and the output clock. This provides a mechanism for reducing the peak output data rate when using the optional scaler functionality, or when using an internal FIFO to reduce line blanking. In this situation, the full resolution of the pixel array is still being read out internally at the target video rate (e.g. 15fps or 30 fps), while the peak output data rate can be reduced by using smaller values for **line\_length\_pck** and **frame\_length\_lines**. The FIFO is described in **Section 11.6**.



**Figure 43 Simple One-PLL System**

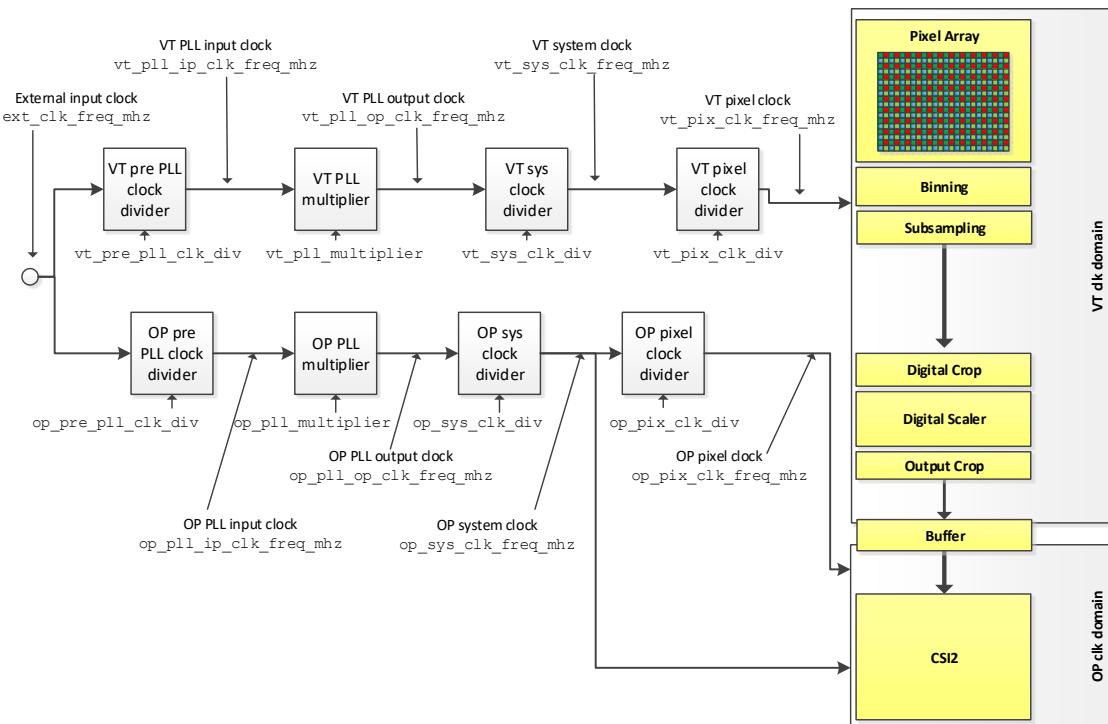


1966

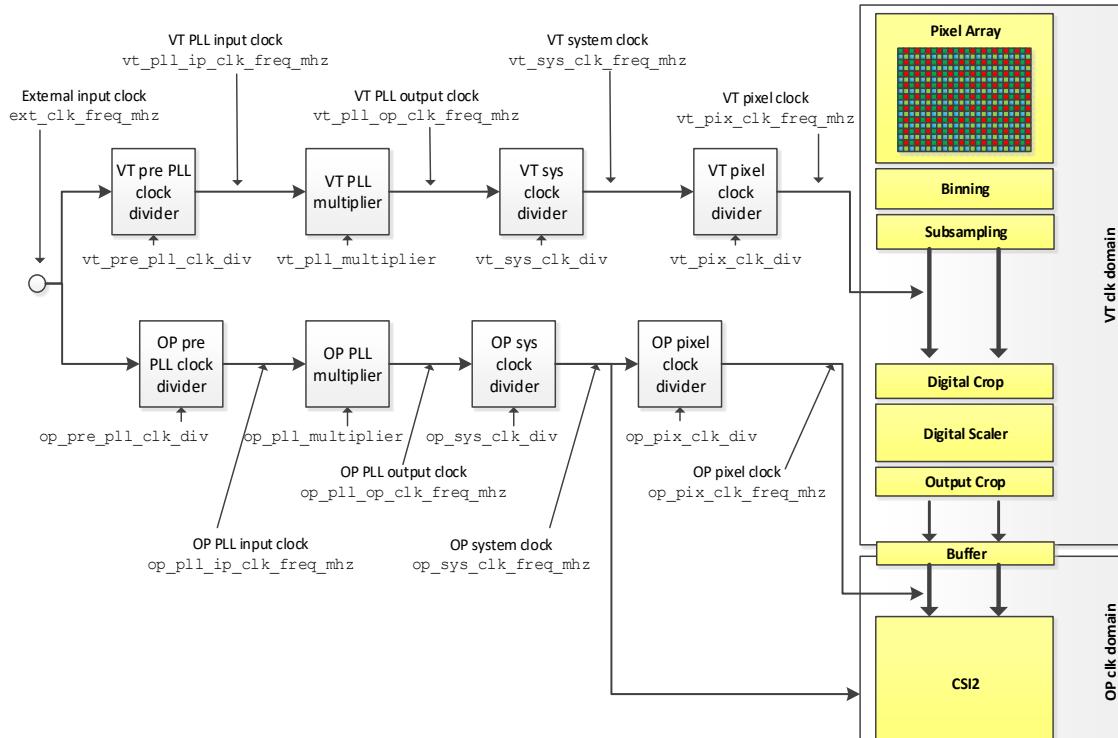
**Figure 44 One PLL with System Speed Model**

1967

**Figure 45 One PLL with Lane Speed Model**



1968

**Figure 46 Two PLL with System Speed Model**

1969

**Figure 47 Two PLL with Lane Speed Model**

1970      In the Lane Speed Model all VT and OP domain speeds are per-Lane, including the limit capability registers  
1971      in ***Table 100***,

1972      **Table 101**, and **Table 102**.

1973      **Example:** With an image sensor running 2-Lane CSI-2 with 1 Gbit/s per Lane, and a 2-Lane VT domain with  
1974      100 Mpix/s per Lane, the speeds within the clock tree would be 1 GHz and 100 MHz. For the System Speed  
1975      Model these speeds would be 2 GHz and 200 MHz, respectively. (This example assumes either the Coupled  
1976      OP Model, or else the Decoupled OP Model with 2 OP Lanes in use.)

### 8.3.2 OP Domain Details and Additional Options

This section further specifies the OP domain that was introduced in *Section 8.3.1*, including recommended control of the OP domain clock tree and related calculations.

#### 8.3.2.1 OP Domain Data Rate Options

The image sensor shall support one of the OP domain frequency options:

- Single Data Rate (for both sys and pixel domain), which is the typical option, or
- Double Data Rate (for both sys and pixel domain), which is an alternative option, or
- Double Data Rate for OP sys domain and Single Data Rate for OP pixel domain, which is an alternative option.

Bit 2 of register **clock\_calculation** determines the bitrate/Symbol rate used by the **op\_sys\_clk\_freq\_mhz**:

- 0: Single Data Rate, where 1 GHz means a bitrate of 1 Gbit/s or a Symbol rate of 1 Gsym/s, or
- 1: Double Data Rate (e.g. 500 MHz means 1 Gbit/s or 1 Gsym/s).

Note that this bit is only valid for a 2-PLL clock tree, and a 1-PLL clock tree always operates at Single Data Rate. Note that **op\_pll\_op\_clk\_freq\_mhz** also follows the SDR vs. DDR setting.

**Note:**

*When op\_sys\_clk\_freq\_mhz is in Double Data Rate mode, with a 2-PLL system, op\_pll\_op\_clk\_freq\_mhz is also in DDR mode. The limit registers (min\_op\_sys\_clk\_freq\_mhz, max\_op\_sys\_clk\_freq\_mhz, min\_op\_pll\_op\_freq\_mhz and max\_op\_pll\_op\_freq\_mhz) will report their values as real double-rate frequencies, not as single-rate frequencies.*

Bit 3 of register **clock\_calculation** determines the rate used by the **op\_pix\_clk\_freq\_mhz**:

- 0: Single Data Rate, where 1 GHz means 1 GHz, or
- 1: Double Data Rate (e.g. 500 MHz means 1 GHz).

**Note:**

*When op\_pix\_clk\_freq\_mhz is in Double Data Rate mode, the limit registers (min\_op\_pix\_clk\_freq\_mhz and max\_op\_pix\_clk\_freq\_mhz) will report their values as real double-rate frequencies, not as single-rate Mpix/s.*

#### 8.3.2.2 Bitrate and Symbol Rate

With D-PHY, the image sensor shall use bitrate.

With C-PHY, the image sensor shall use Symbol rate. The C-PHY bitrate is given by the formula:

$$\text{bitrate [bps]} = \text{Symbol rate [sps]} * 16/7$$

**Note:**

*The value of the 16/7 multiplier is approximately 2.28.*

### 8.3.2.3 OP Domain Formulas

This Section specifies how OP Domain Frequencies are calculated.

There are two Mode options for **op\_pix\_clk\_div** programming rules:

- **Flexible op\_pix\_clk\_div Mode**

In this Mode, **op\_pix\_clk\_div** values are not fixed.

- **Legacy op\_pix\_clk\_div Mode**

In this Mode, **op\_pix\_clk\_div** values are fixed.

For C-PHY, the image sensor shall support Flexible **op\_pix\_clk\_div** Mode.

For D-PHY, the image sensor shall support either Flexible **op\_pix\_clk\_div** Mode or Legacy **op\_pix\_clk\_div** Mode. The Host can determine which Mode is supported by inspecting bit 3 of register **clock\_tree\_pll\_capability**.

As detailed in the following Sections, each Mode uses a separate formula to calculate OP Domain Frequencies. The Host shall ensure that:

**For Flexible op\_pix\_clk\_div Mode:**

$$\text{op\_sys\_clk\_freq\_mhz} * \text{number of CSI-2 data lanes} * (\text{op\_sys\_ddr} + 1) \leq (\text{number of OP lanes} * (\text{op\_pix\_ddr} + 1) * \text{bits per pixel} * \text{op\_pix\_clk\_freq\_mhz} * K/16) / L$$

where K = 16 for D-PHY, and K = 7 for C-PHY.

**Note:**

*If the two sides of the above formula are unequal, then the image sensor shall ensure proper operation despite the aggregate bit rate on the OP lanes being greater than the aggregate bit rate on the CSI-2 data lanes. One possible method for ensuring this is by periodically interrupting the pixel stream on the OP lanes.*

**For Legacy op\_pix\_clk\_div Mode:**

$$\text{op\_sys\_clk\_freq\_mhz} * \text{number of CSI-2 data lanes} * (\text{op\_sys\_ddr} + 1) = (\text{number of OP lanes} * (\text{op\_pix\_ddr} + 1) * \text{bits per pixel} * \text{op\_pix\_clk\_freq\_mhz} * K/16) / L$$

where K=16 for D-PHY.

**Note:**

*In above formulas:*

**op\_sys\_ddr** = 1 if the OP sys domain is in Double Data Rate mode and 0 if in Single Data Rate mode  
**op\_pix\_ddr** = 1 if the OP pixel domain is in Double Data Rate mode and 0 if in Single Data Rate mode

*'bits per pixel' is equal to the output pixel resolution*

*L = 1 if op\_bits\_per\_lane (Table 87) is 0, or greater than or equal to 'bits per pixel'; otherwise, L = 2*

### 8.3.2.4 Coupled OP Mode

This Section shows how the OP Domain Frequency formulas given in *Section 8.3.2.3* are used in Coupled OP Mode. In Coupled OP mode, the number of active internal OP Lanes is the same as the number of active CSI-2 data Lanes, simplifying the formulas.

#### 8.3.2.4.1 Op\_sys\_clk\_freq\_mhz Calculation

For D-PHY:

In Flexible op\_pix\_clk\_div Mode:

$$\text{op\_sys\_clk\_freq\_mhz} * (\text{op\_sys\_ddr} + 1) \leq \text{bits per pixel} * \text{op\_pix\_clk\_freq\_mhz} * (\text{op\_pix\_ddr} + 1) / L$$

In Legacy op\_pix\_clk\_div Mode:

$$\text{op\_sys\_clk\_freq\_mhz} * (\text{op\_sys\_ddr} + 1) = \text{bits per pixel} * \text{op\_pix\_clk\_freq\_mhz} * (\text{op\_pix\_ddr} + 1) / L$$

For C-PHY in Flexible op\_pix\_clk\_div Mode:

$$\text{op\_sys\_clk\_freq\_mhz} * (\text{op\_sys\_ddr} + 1) \leq (\text{bits per pixel} * \text{op\_pix\_clk\_freq\_mhz} * (\text{op\_pix\_ddr} + 1) * 7/16) / L$$

Note:

In above formulas:

**op\_sys\_ddr** = 1 if the OP sys domain is in Double Data Rate mode and 0 if in Single Data Rate mode

**op\_pix\_ddr** = 1 if the OP pixel domain is in Double Data Rate mode and 0 if in Single Data Rate mode

'bits per pixel' is equal to the output pixel resolution

*L* = 1 if **op\_bits\_per\_lane** (*Table 87*) is 0 or greater than or equal to 'bits per pixel'; otherwise, *L* = 2

### 8.3.2.4.2 op\_pix\_clk\_div Calculation

2056 For D-PHY:

2057 In Flexible op\_pix\_clk\_div Mode:

$$2058 \quad \text{op\_pix\_clk\_div} = \text{op\_sys\_clk\_freq\_mhz} / \text{op\_pix\_clk\_freq\_mhz}$$

2059 and

$$2060 \quad \text{op\_pix\_clk\_div} \leq (\text{bits per pixel} * (\text{op\_pix\_ddr} + 1)) / (\text{op\_sys\_ddr} + 1) / L$$

2061 In Legacy op\_pix\_clk\_div Mode:

$$2062 \quad \text{op\_pix\_clk\_div} = \text{op\_sys\_clk\_freq\_mhz} / \text{op\_pix\_clk\_freq\_mhz}$$

2063 and

$$2064 \quad \text{op\_pix\_clk\_div} = (\text{bits per pixel} * (\text{op\_pix\_ddr} + 1)) / (\text{op\_sys\_ddr} + 1) / L$$

2065 For C-PHY in Flexible op\_pix\_clk\_div Mode:

$$2066 \quad \text{op\_pix\_clk\_div} = \text{op\_sys\_clk\_freq\_mhz} / \text{op\_pix\_clk\_freq\_mhz}$$

2067 and

$$2068 \quad \text{op\_pix\_clk\_div} \leq (\text{bits per pixel} * (\text{op\_pix\_ddr} + 1)) / (\text{op\_sys\_ddr} + 1) * 7/16 / L$$

2069 Note:

2070 If '(bits per pixel \* (op\_pix\_ddr + 1) / (op\_sys\_ddr + 1) \* 7/16) / L' is not an integer value, then  
2071 op\_pix\_clk\_div should be rounded down to the next lowest integer value, in order to guarantee that  
2072 'op\_pix\_clk\_div'  $\leq$  (bits per pixel \* (op\_pix\_ddr + 1) / (op\_sys\_ddr + 1) \* 7/16) / L.

2073 Note:

2074 In above formulas:

2075 **op\_sys\_ddr** = 1 if the OP sys domain is in Double Data Rate mode and 0 if in Single Data Rate mode;

2076 **op\_pix\_ddr** = 1 if the OP pixel domain is in Double Data Rate mode and 0 if in Single Data Rate mode

2077 'bits per pixel' is equal to the output pixel resolution

2078 *L* = 1 if **op\_bits\_per\_lane** (Table 87) is 0, or greater than or equal to 'bits per pixel'; otherwise, *L* = 2

### 8.3.2.4.3 Examples

2079 In per-Lane mode, the Host programs the image sensor Lane frequencies, resulting in higher total Link speeds  
2080 (depending on number of CSI-2 Lanes being used).

2081 For Single Data Rate:

2082 In this example, both **op\_sys\_clk\_freq\_mhz** and **op\_pix\_clk\_freq\_mhz** are in Single Data Rate mode.

- 2083 • For D-PHY with 2 CSI-2 data Lanes both running at 2Gbit/s, the **op\_sys\_clk\_freq\_mhz** is  
2084 programmed to be 2 GHz. The toggling frequency of the CSI-2 clock Lane is then 1 GHz, giving  
2085 an overall CSI-2 Link speed of 4 Gbit/s.

2086 Assuming two OP Lanes with **op\_bits\_per\_lane** = 16, **op\_pix\_clk\_freq\_mhz** is programmed to  
2087 be **op\_sys\_clk\_freq\_mhz** / (bits per pixel / L). For RAW10 (i.e., 10 bits per pixel), this is  
2088 **op\_sys\_clk\_freq\_mhz** / 10 (i.e., L = 1) resulting in an **op\_pix\_clk\_freq\_mhz** value of 200 MHz  
2089 and 200 Mpix/s per Lane. For RAW20 (20 bits per pixel), **op\_pix\_clk\_freq\_mhz** is also  
2090 **op\_sys\_clk\_freq\_mhz** / 10 (i.e., L = 2), but there are two **op\_pix\_clk\_freq\_mhz** clock cycles per  
2091 RAW20 pixel, resulting in an **op\_pix\_clk\_freq\_mhz** value of 200 MHz and 100 Mpix/s per Lane.

- 2092 • For C-PHY with 2 CSI-2 Lanes both running at 2 Gbit/s (and therefore a Symbol rate of  
2093 2000 Mbit/s / 16\*7 = 875 Msym/s), the **op\_sys\_clk\_freq\_mhz** is programmed to be 875 MHz.  
2094 The Link speed is then 1750 Msym/s, giving an overall CSI-2 Link speed of 4 Gbit/s.

Assuming two OP Lanes with **op\_bits\_per\_lane** = 0, **op\_pix\_clk\_freq\_mhz** is programmed to be **op\_sys\_clk\_freq\_mhz** / (bits per pixel \* 7/16). For RAW10 the denominator is then (10 \* 7/16). Because this is not an integer value, we use a value of 4 (not 4.375) for **op\_pix\_clk\_div**, in order to satisfy the formula requirement that **op\_pix\_clk\_div**  $\leq$  (bits per pixel \* 7/16). This gives an **op\_pix\_clk\_freq\_mhz** value of 875/4 or 218.75 MHz, meaning 218.75 Mpix/s per Lane.

#### For Double Data Rate:

In this example, both **op\_sys\_clk\_freq\_mhz** and **op\_pix\_clk\_freq\_mhz** are in Double Data Rate mode.

- For D-PHY with 2 CSI-2 data Lanes both running at 2 Gbit/s, the **op\_sys\_clk\_freq\_mhz** is programmed to be 1 GHz. The toggling frequency of the CSI-2 clock Lane is then 1 GHz, giving an overall CSI-2 Link speed of 4 Gbit/s.

Assuming two OP Lanes with **op\_bits\_per\_lane** = 0, **op\_pix\_clk\_freq\_mhz** is programmed to be **op\_sys\_clk\_freq\_mhz** / bits per pixel. For RAW10 (with 10 bits per pixel) this is **op\_sys\_clk\_freq\_mhz** / 10. This gives an **op\_pix\_clk\_freq\_mhz** 100 MHz, but because this is Double Data Rate the actual speed is 200 Mpix/s per Lane.

- For C-PHY with 2 CSI-2 Lanes both running at 2 Gbit/s (and therefore a Symbol rate of 2000 Mbit/s / 16\*7 = 875 Msym/s), the **op\_sys\_clk\_freq\_mhz** is programmed to be 875/2 = 437.5 MHz. The Link speed is then 1750 Msym/s, i.e., 4 Gbit/s.

Assuming two OP Lanes with **op\_bits\_per\_lane** = 0, **op\_pix\_clk\_freq\_mhz** is programmed to be **op\_sys\_clk\_freq\_mhz** / (bits per pixel \* 7/16). For RAW10 the denominator is then (10 \* 7/16). Because this is not an integer value, we use a value of 4 (not 4.375) for **op\_pix\_clk\_div**, in order to satisfy the formula requirement that **op\_pix\_clk\_div**  $\leq$  (bits per pixel \* 7/16). This gives an **op\_pix\_clk\_freq\_mhz** value of 437.5/4 or 109.375 MHz, but because this is Double Data Rate the actual speed is 218.75 Mpix/s per Lane.

#### For **op\_sys\_clk\_freq\_mhz** Double Data Rate and **op\_pix\_clk\_freq\_mhz** Single Data Rate:

- For D-PHY with 2 CSI-2 data Lanes both running at 2 Gbit/s, the **op\_sys\_clk\_freq\_mhz** is programmed to be 1 GHz. The toggling frequency of the CSI-2 clock Lane is then 1 GHz, giving an overall CSI-2 Link speed of 4 Gbit/s.

Assuming two OP Lanes with **op\_bits\_per\_lane** = 16, **op\_pix\_clk\_freq\_mhz** is programmed to be **op\_sys\_clk\_freq\_mhz** / (bits per pixel/(2\*L)). For RAW10 (10 bits per pixel), this is **op\_sys\_clk\_freq\_mhz** / 5 (i.e., L = 1) resulting in an **op\_pix\_clk\_freq\_mhz** value of 200 MHz and 200 Mpix/s per Lane. For RAW20 (20 bits per pixel), **op\_pix\_clk\_freq\_mhz** is also **op\_sys\_clk\_freq\_mhz** / 5 (i.e., L = 2), but there are two **op\_pix\_clk\_freq\_mhz** clock cycles per RAW20 pixel resulting in an **op\_pix\_clk\_freq\_mhz** value of 200 MHz and 100 Mpix/s per Lane.

- For C-PHY with 2 CSI-2 Lanes both running at 2 Gbit/s (and therefore a Symbol rate of 2000 Mbit/s / 16\*7 = 875 Msym/s), the **op\_sys\_clk\_freq\_mhz** is programmed to be 875/2 = 437.5 MHz. The Link speed is then 1750 Msym/s, i.e. 4 Gbit/s.

Assuming two OP Lanes with **op\_bits\_per\_lane** = 0, **op\_pix\_clk\_freq\_mhz** is programmed to be **op\_sys\_clk\_freq\_mhz** / (bits per pixel/2 \* 7/16). For RAW10 the denominator is then (10/2 \* 7/16). Because this is not an integer value, we use a value of 2 (not 2.1875) for **op\_pix\_clk\_div**, in order to satisfy the formula requirement that **op\_pix\_clk\_div**  $\leq$  (bits per pixel/2 \* 7/16). This gives an **op\_pix\_clk\_freq\_mhz** value of 437.5/2 or 218.75 MHz, meaning 218.75 Mpix/s per Lane.

### 8.3.2.4.4 OP Domain Pixel Clock Calculations

When needed, the Host can calculate the average OP domain pixel rate (in pixels per second) as follows:

**For D-PHY:**

**For a Single Data Rate image sensor:**

**op pixel clock frequency mpix = op\_sys\_clk\_freq\_mhz \* number of CSI-2 lanes / bits per pixel**

**For a Double Data Rate image sensor:**

**op pixel clock frequency mpix = op\_sys\_clk\_freq\_mhz \* number of CSI-2 lanes \* 2 / bits per pixel**

**For a op\_sys\_clk\_freq\_mhz Double Data Rate and op\_pix\_clk\_freq\_mhz Single Data Rate image sensor:**

**op pixel clock frequency mpix = op\_sys\_clk\_freq\_mhz \* number of CSI-2 lanes \* 2 / bits per pixel**

**For C-PHY:**

**For a Single Data Rate image sensor:**

**op pixel clock frequency mpix = op\_sys\_clk\_freq\_mhz \* number of CSI-2 lanes \* 16 / (7 \* bits per pixel)**

**For a Double Data Rate image sensor:**

**op pixel clock frequency mpix = op\_sys\_clk\_freq\_mhz \* number of CSI-2 lanes \* 32 / (7 \* bits per pixel)**

**For a op\_sys\_clk\_freq\_mhz Double Data Rate and op\_pix\_clk\_freq\_mhz Single Data Rate image sensor:**

**op pixel clock frequency mpix = op\_sys\_clk\_freq\_mhz \* number of CSI-2 lanes \* 32 / (7 \* bits per pixel)**

2157  
2158 **Table 88** shows example values of **op\_pix\_clk\_freq\_mhz** for 10-bit and 8-bit data in MHz (not in pixels per second) and the peak Mpix/s value.

2159

**Table 88 Clocking Example (Assuming op\_bits\_per\_lane = 0)**

PHY	Data Rate	op_sys_clk_freq_mhz	CSI_data_format(7:0)	op_pix_clk_freq_mhz	Number of Lanes		Total Link Speed
					CSI-2	OP	
D-PHY	Single	2000	10	200	2 (data)	2	400 Mpix/s, 4 Gbit/s
					2 (data)	2	500 Mpix/s, 4 Gbit/s
	Double	1000	10	100	2 (data)	2	400 Mpix/s, 4 Gbit/s
					2 (data)	2	500 Mpix/s, 4 Gbit/s
	Sys DDR and Pix SDR	1000	10	200	2 (data)	2	400 Mpix/s, 4 Gbit/s
					2 (data)	2	500 Mpix/s, 4 Gbit/s
C-PHY	Single	875	10	218.75	2	2	437.5 Mpix/s, 4 Gbit/s
					2	2	583.3 Mpix/s, 4 Gbit/s
	Double	437.5	10	109.375	2	2	437.5 Mpix/s, 4 Gbit/s
					2	2	583.3 Mpix/s, 4 Gbit/s
	Sys DDR and Pix SDR	437.5	10	218.75	2	2	437.5 Mpix/s, 4 Gbit/s
					1	1	437.5 Mpix/s, 2 Gbit/s

### 8.3.2.5 Decoupled OP Mode Examples

This Section shows how the formulas given in *Section 8.3.2.3* are used in Decoupled OP Mode. In Decoupled OP Mode, the number of OP Lanes can be different from the number of CSI-2 data Lanes.

#### 8.3.2.5.1 Op\_sys\_clk\_freq\_mhz Calculation

For D-PHY:

In Flexible op\_pix\_clk\_div Mode:

$\text{op\_sys\_clk\_freq\_mhz} * \text{number of CSI-2 data lanes} * (\text{op\_sys\_ddr} + 1) \leq \text{number of OP lanes} * (\text{op\_pix\_ddr} + 1) * \text{bits per pixel} * \text{op\_pix\_clk\_freq\_mhz} / L$

In Legacy op\_pix\_clk\_div Mode:

$\text{op\_sys\_clk\_freq\_mhz} * \text{number of CSI-2 data lanes} * (\text{op\_sys\_ddr} + 1) = \text{number of OP lanes} * (\text{op\_pix\_ddr} + 1) * \text{bits per pixel} * \text{op\_pix\_clk\_freq\_mhz} / L$

For C-PHY in Flexible op\_pix\_clk\_div Mode:

$\text{op\_sys\_clk\_freq\_mhz} * \text{number of CSI-2 data lanes} * (\text{op\_sys\_ddr} + 1) \leq (\text{number of OP lanes} * (\text{op\_pix\_ddr} + 1) * \text{bits per pixel} * \text{op\_pix\_clk\_freq\_mhz} * 7/16) / L$

Note:

In above formulas:

$\text{op\_sys\_ddr} = 1$  if the OP sys domain is in Double Data Rate mode and 0 if in Single Data Rate mode;

$\text{op\_pix\_ddr} = 1$  if the OP pixel domain is in Double Data Rate mode and 0 if in Single Data Rate mode

'bits per pixel' is equal to the output pixel resolution

$L = 1$  if  $\text{op\_bits\_per\_lane}$  (*Table 87*) is 0, or greater than or equal to 'bits per pixel'; otherwise,  $L = 2$

### 8.3.2.5.2 op\_pix\_clk\_div Calculation

For D-PHY:

In Flexible op\_pix\_clk\_div Mode:

$$\text{op\_pix\_clk\_div} = \text{op\_sys\_clk\_freq\_mhz} / \text{op\_pix\_clk\_freq\_mhz}$$

and

$$\text{op\_pix\_clk\_div} \leq (\text{number of OP lanes} * \text{bits per pixel} / \text{number of CSI-2 data lanes} * (\text{op\_pix\_ddr} + 1) / (\text{op\_sys\_ddr} + 1)) / L$$

**Note:**

If '(number of OP lanes \* bits per pixel / number of CSI-2 data lanes \* (op\_pix\_ddr + 1) / (op\_sys\_ddr + 1)) / L' is not an integer value, then op\_pix\_clk\_div should be rounded down to the next lowest integer value.

In Legacy op\_pix\_clk\_div Mode:

$$\text{op\_pix\_clk\_div} = \text{op\_sys\_clk\_freq\_mhz} / \text{op\_pix\_clk\_freq\_mhz}$$

and

$$\text{op\_pix\_clk\_div} = (\text{number of OP lanes} * \text{bits per pixel} / \text{number of CSI-2 data lanes} * (\text{op\_pix\_ddr} + 1) / (\text{op\_sys\_ddr} + 1)) / L$$

For C-PHY in Flexible op\_pix\_clk\_div Mode:

$$\text{op\_pix\_clk\_div} = \text{op\_sys\_clk\_freq\_mhz} / \text{op\_pix\_clk\_freq\_mhz}$$

and

$$\text{op\_pix\_clk\_div} \leq (\text{number of OP lanes} * \text{bits per pixel} / \text{number of CSI-2 data lanes} * (\text{op\_pix\_ddr} + 1) / (\text{op\_sys\_ddr} + 1) * 7/16) / L$$

**Note:**

If '(number of OP lanes \* bits per pixel / number of CSI-2 data lanes \* (op\_pix\_ddr + 1) / (op\_sys\_ddr + 1) \* 7/16) / L' is not an integer value, then op\_pix\_clk\_div should be rounded down to the next lowest integer value.

**Note:**

In above formulas:

**op\_sys\_ddr** = 1 if the OP sys domain is in Double Data Rate mode and 0 if in Single Data Rate mode;

**op\_pix\_ddr** = 1 if the OP pixel domain is in Double Data Rate mode and 0 if in Single Data Rate mode

'bits per pixel' is equal to the output pixel resolution

$L = 1$  if **op\_bits\_per\_lane** (Table 87) is 0, or greater than or equal to 'bits per pixel'; otherwise,  $L = 2$

### 8.3.2.5.3 Examples

In per-Lane mode, the Host programs the image sensor Lane frequencies, resulting in higher total Link speeds (depending on number of CSI-2 Lanes being used).

#### For Single Data Rate:

In this example, both **op\_sys\_clk\_freq\_mhz** and **op\_pix\_clk\_freq\_mhz** are in Single Data Rate mode.

- For D-PHY with 2 CSI-2 data Lanes each running at 2 Gbit/s, and an image sensor supporting 4 OP Lanes (i.e., **num\_of\_op\_lanes** = 4), the **op\_sys\_clk\_freq\_mhz** is programmed to be 2 GHz. The CSI-2 clock Lane toggling frequency is then 1 GHz, giving an overall Link speed of 4 Gbit/s.

The **op\_pix\_clk\_freq\_mhz** is programmed to be  $\text{op\_sys\_clk\_freq\_mhz} / ((\text{number of OP Lanes} * \text{bits per pixel} / \text{number of CSI-2 data Lanes}) / L)$ . Assume **op\_bits\_per\_lane** = 16. Therefore, for RAW10 (i.e., 10 bits per pixel) L = 1, and the latter expression reduces to  $\text{op\_sys\_clk\_freq\_mhz} / (4 * 10/2) = \text{op\_sys\_clk\_freq\_mhz} / 20$ , yielding an **op\_pix\_clk\_freq\_mhz** of 100 MHz, and 100 Mpix/s per OP Lane. For RAW20 (i.e., 20 bits per pixel), **op\_pix\_clk\_freq\_mhz** is also  $\text{op\_sys\_clk\_freq\_mhz} / 20$  (i.e., L = 2), but there are two **op\_pix\_clk\_freq\_mhz** clock cycles per RAW20 pixel resulting in an **op\_pix\_clk\_freq\_mhz** value of 100 MHz and 50 Mpix/s per OP Lane.

- For C-PHY with 2 CSI-2 Lanes each running at 2 Gbit/s (i.e., having a Symbol rate of 2000 Mbit/s / 16\*7 = 875 Msym/s), and an image sensor supporting 4 OP Lanes (i.e., **num\_of\_op\_lanes** = 4), the **op\_sys\_clk\_freq\_mhz** is programmed to be 875 MHz. The 2-Lane CSI-2 Link speed is then 1750 Msym/s, giving an overall Link speed of 4 Gbit/s.

Assuming **op\_bits\_per\_lane** = 0, **op\_pix\_clk\_freq\_mhz** is programmed to be  $\text{op\_sys\_clk\_freq\_mhz} / (\text{number of OP lanes} * \text{bits per pixel} / \text{number of CSI-2 data Lanes} * 7/16)$ . For RAW10 (10 bits per pixel) the denominator is  $(4 * 10/2 * 7/16)$ . Because this is not an integer value, we use a value of 8 (not 8.75) for **op\_pix\_clk\_div**, in order to satisfy the formula requirement that  $\text{op\_pix\_clk\_div} \leq (\text{number of OP Lanes} * \text{bits per pixel} / \text{number of CSI-2 data Lanes} * 7/16)$ . This gives an **op\_pix\_clk\_freq\_mhz** value of 875/8 or 109.375 MHz, meaning 109.75 Mpix/s per OP Lane.

#### For Double Data Rate:

In this example, both **op\_sys\_clk\_freq\_mhz** and **op\_pix\_clk\_freq\_mhz** are in Double Data Rate mode.

- For D-PHY with 2 CSI-2 data Lanes each running at 2 Gbit/s, and an image sensor supporting 4 OP lanes (i.e., **num\_of\_op\_lanes** = 4), the **op\_sys\_clk\_freq\_mhz** is programmed to be 1 GHz. The toggling frequency of CSI-2 clock Lane is then 1 GHz, giving an overall CSI-2 Link speed of 4 Gbit/s.

Assuming **op\_bits\_per\_lane** = 0, **op\_pix\_clk\_freq\_mhz** is programmed to be  $\text{op\_sys\_clk\_freq\_mhz} / (\text{number of OP Lanes} * \text{bits per pixel} / \text{number of CSI-2 data Lanes})$ . For RAW10 (i.e., 10 bits per pixel) this is  $\text{op\_sys\_clk\_freq\_mhz} / (4 * 10/2)$  which gives an **op\_pix\_clk\_freq\_mhz** of 50 MHz. Because this is Double Data Rate, the actual data rate is twice that, or 100 Mpix/s per OP Lane.

- For C-PHY with 2 CSI-2 Lanes each running at 2 Gbit/s (i.e., having a Symbol rate of 2000 Mbit/s / 16\*7 = 875Msym/s), the **op\_sys\_clk\_freq\_mhz** is programmed to be  $875/2 = 437.5$  MHz. The 2-Lane CSI-2 Link speed is then 1750 Msym/s, giving an overall Link speed of 4 Gbit/s.

2249 Assuming **op\_bits\_per\_lane** = 0, **op\_pix\_clk\_freq\_mhz** is programmed to be  
 2250 **op\_sys\_clk\_freq\_mhz** / (number of OP Lanes \* bits per pixel / number of CSI-2 data Lanes \*  
 2251 7/16). For RAW10 (i.e., 10 bits per pixel) the denominator is (4 \* 10/2 \* 7/16). Because this is not  
 2252 an integer value, we use a value of 8 (not 8.75) for **op\_pix\_clk\_div**, in order to satisfy the formula  
 2253 requirement that **op\_pix\_clk\_div**  $\leq$  (number of OP Lanes \* bits per pixel / number of CSI-2 data  
 2254 Lanes \* 7/16). This gives an **op\_pix\_clk\_freq\_mhz** value of 437.5/8 or 54.69 MHz. Because this  
 2255 is Double Data Rate, the actual data rate is twice that, or 109.375 Mpix/s per OP Lane.

2256 **For op\_sys\_clk\_freq\_mhz Double Data Rate and op\_pix\_clk\_freq\_mhz Single Data Rate:**

- For D-PHY with 2 CSI-2 data Lanes each running at 2 Gbit/s, and an image sensor supporting 4 OP lanes (i.e., **num\_of\_op\_lanes** = 4), the **op\_sys\_clk\_freq\_mhz** is programmed to be 1 GHz. The toggling frequency of CSI-2 clock Lane is then 1 GHz, giving an overall CSI-2 Link speed of 4 Gbit/s.

2261 The **op\_pix\_clk\_freq\_mhz** is programmed to be **op\_sys\_clk\_freq\_mhz** / ((number of OP Lanes \*  
 2262 (bits per pixel / 2) / number of CSI-2 data Lanes) / L). Assume **op\_bits\_per\_lane** = 16. Therefore,  
 2263 for RAW10 (i.e., 10 bits per pixel), L = 1 and the latter expression reduces to  
 2264 **op\_sys\_clk\_freq\_mhz** / (4 \* (10/2)/2) = **op\_sys\_clk\_freq\_mhz** / 10, resulting in an  
 2265 **op\_pix\_clk\_freq\_mhz** value of 100 MHz or 100 Mpix/s per OP Lane. For RAW20 (i.e., 20 bits  
 2266 per pixel), **op\_pix\_clk\_freq\_mhz** is also **op\_sys\_clk\_freq\_mhz** / 10 (i.e., L = 2), but there are  
 2267 two **op\_pix\_clk\_freq\_mhz** clock cycles per RAW20 pixel resulting in an **op\_pix\_clk\_freq\_mhz**  
 2268 value of 100 MHz and 50 Mpix/s per OP Lane.

- For C-PHY with 2 CSI-2 Lanes each running at 2 Gbit/s (i.e. having a Symbol rate of 2000 Mbit/s/16\*7 = 875Msym/s), the **op\_sys\_clk\_freq\_mhz** is programmed to be 875/2 = 437.5 MHz. The 2-Lane CSI-2 Link speed is then 1750 Msym/s, giving an overall Link speed of 4 Gbit/s.

2269 Assuming **op\_bits\_per\_lane** = 0, **op\_pix\_clk\_freq\_mhz** is programmed to be  
 2270 **op\_sys\_clk\_freq\_mhz** / (number of OP Lanes \* (bits per pixel / 2) / number of CSI-2 data Lanes \*  
 2271 7/16). For RAW10 (10 bits per pixel) the denominator is (4 \* (10/2)/2 \* 7/16). Because this is not  
 2272 an integer value, we use a value of 4 (not 4.375) for **op\_pix\_clk\_div**, in order to fulfill the formula  
 2273 requirement that **op\_pix\_clk\_div**  $\leq$  (number of OP Lanes \* bits per pixel/2 / number of CSI-2 data  
 2274 Lanes \* 7/16). This gives an **op\_pix\_clk\_freq\_mhz** value of 437.5/4 or 109.375 MHz, meaning  
 2275 109.375 Mpix/s per OP Lane.

### 8.3.2.5.4 OP Domain Pixel Clock Calculations

When needed, the Host can calculate the average OP domain pixel rate (in pixels per second) as follows:

**For D-PHY:**

**For a Single Data Rate image sensor:**

**op pixel clock frequency mpix = op\_sys\_clk\_freq\_mhz \* number of CSI-2 lanes / bits per pixel**

**For a Double Data Rate image sensor:**

**op pixel clock frequency mpix = op\_sys\_clk\_freq\_mhz \* number of CSI-2 lanes \* 2 / bits per pixel**

**For a op\_sys\_clk\_freq\_mhz Double Data Rate and op\_pix\_clk\_freq\_mhz Single Data Rate image sensor:**

**op pixel clock frequency mpix = op\_sys\_clk\_freq\_mhz \* number of CSI-2 lanes \* 2 / bits per pixel**

**For C-PHY:**

**For a Single Data Rate image sensor:**

**op pixel clock frequency mpix = op\_sys\_clk\_freq\_mhz \* number of CSI-2 lanes \* 16 / (7 \* bits per pixel)**

**For a Double Data Rate image sensor:**

**op pixel clock frequency mpix = op\_sys\_clk\_freq\_mhz \* number of CSI-2 lanes \* 32 / (7 \* bits per pixel)**

**For a op\_sys\_clk\_freq\_mhz Double Data Rate and op\_pix\_clk\_freq\_mhz Single Data Rate image sensor:**

**op pixel clock frequency mpix = op\_sys\_clk\_freq\_mhz \* number of CSI-2 lanes \* 32 / (7 \* bits per pixel)**

2299  
2300 **Table 89** shows example values of **op\_pix\_clk\_freq\_mhz** for 10-bit and 8-bit data in MHz (not in pixels per second) and the peak Mpix/s value.

2301 **Table 89 Clocking Example (Assuming op\_bit\_per\_lane = 0)**

PHY	Data Rate	op_sys_clk_freq_mhz	CSI_data_format(7:0)	op_pix_clk_freq_mhz	Number of Lanes		Total Link Speed
					CSI-2	OP	
D-PHY	Single	2000	10	100	2 (data)	4	400 Mpix/s, 4 Gbit/s
			8	125	2 (data)	4	500 Mpix/s, 4 Gbit/s
	Double	1000	10	50	2 (data)	4	400 Mpix/s, 4 Gbit/s
			8	62.5	2 (data)	4	500 Mpix/s, 4 Gbit/s
	Sys DDR and Pix SDR	1000	10	100	2 (data)	4	400 Mpix/s, 4 Gbit/s
			8	125	2 (data)	4	500 Mpix/s, 4 Gbit/s
C-PHY	Single	875	10	109.375	2	4	437.5 Mpix/s, 4 Gbit/s
			8	125	2	4	500 Mpix/s, 4 Gbit/s
	Double	437.5	10	54.69	2	4	437.5 Mpix/s, 4 Gbit/s
			8	62.5	2	4	500 Mpix/s, 4 Gbit/s
	Sys DDR and Pix SDR	437.5	10	109.375	2	4	437.5 Mpix/s, 4 Gbit/s
			8	145.83	2	4	583.3 Mpix/s, 4 Gbit/s

### 8.3.3 Frequencies

This Section specifies how clock tree frequencies can be calculated, based on the Clock Tree control registers. All examples and formulas assume Single Data Rate operation for the OP Domain (*Section 8.3.2.1*) and Legacy **op\_pix\_clk\_div** values (*Section 8.3.2.3*). See *Section 8.3.2* for further OP Domain options.

#### 8.3.3.1 System Speed Model

In system speed models, the relationships among clocks are defined as follows.

##### For a 1-PLL System

The OP domain is not relevant with the simple 1-PLL model.

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

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

##### For a 2-PLL System

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

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

### 8.3.3.2 Lane Speed Model

In Lane speed models, the relationships among clocks are defined as follows.

#### 8.3.3.2.1 External Frequencies

The external VT and OP pixel clock frequencies are calculated as follows.

The following formula is useful for determining the pixel rate (e.g., 8 Mpix at 30 fps = 240 Mpix/s), or in Exposure time calculations.

$$ext\_vt\_pix\_clk\_freq\_mhz = \frac{ext\_clk\_freq\_mhz * vt\_pll\_multiplier * num\_of\_vt\_lanes}{vt\_pre\_pll\_clk\_div * vt\_sys\_clk\_div * vt\_pix\_clk\_div}$$

The following formula assumes that the image sensor supports coupled calculation for the OP domain, i.e., that the number of active internal OP domain Lanes is the same as the number of active CSI-2 Lanes. In the formula, the number of used CSI-2 Lanes is specified by register **csi\_lane\_mode**.

$$ext\_op\_pix\_clk\_freq\_mhz = \frac{ext\_clk\_freq\_mhz * op\_pll\_multiplier * (csi\_lane\_mode + 1)}{op\_pre\_pll\_clk\_div * op\_sys\_clk\_div * op\_pix\_clk\_div}$$

The following formula assumes that the image sensor supports decoupled calculation for the OP domain. In the formula, the number of internal OP domain Lanes is specified by register **num\_of\_op\_lanes**.

$$ext\_op\_pix\_clk\_freq\_mhz = \frac{ext\_clk\_freq\_mhz * op\_pll\_multiplier * num\_of\_op\_lanes}{op\_pre\_pll\_clk\_div * op\_sys\_clk\_div * op\_pix\_clk\_div}$$

### 8.3.3.2.2 Internal Frequencies

The internal VT and OP pixel clock frequencies are calculated as follows. These formulas are useful for determining the pixel rate per Lane, for example when detecting the image sensor per-Lane speed capabilities. See also *Section 9.2.1* for an explanation of how **vt\_pix\_clk\_freq\_mhz** is used in exposure time calculation.

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

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

### 8.3.4 Clock Tree Control Registers

2330 This Section defines the Clock Tree control registers, which control clock frequencies within the Clock Tree.

#### 8.3.4.1 Register Usage

2331 Depending on the particular Clock Tree option being used, different Clock Tree control registers can be used.  
 2332 Image sensor behaviour is undefined if any of the registers in *Table 90*, *Table 91*, or *Table 92* are changed  
 2333 during Streaming mode.

2334 *Table 90* lists the CCI Registers that control the relationship between the external input clock frequency and  
 2335 the pixel clock frequency, for a simple One-PLL system.

2336 **Table 90 Clock Division and PLL Multiplier Registers for Simple One-PLL System**

Register Name	Type	RW	Comment
<b>vt_pix_clk_div</b>	16-bit unsigned integer	RW	Video Timing Pixel Clock Divider Value
<b>vt_sys_clk_div</b>	16-bit unsigned integer	RW	Video Timing System Clock Divider Value
<b>pre_pll_clk_div</b>	16-bit unsigned integer	RW	Pre PLL clock Divider Value
<b>pll_multiplier</b>	16-bit unsigned integer	RW	PLL multiplier Value

2337 *Table 91* lists the CCI Registers that control the relationship between the external input clock frequency and  
 2338 the pixel clock frequency, for a One-PLL system.

2339 **Table 91 Clock Division and PLL Multiplier Registers for One-PLL System**

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

2340  
2341     Table 92 lists the CCI Registers that control the relationship between the external input clock frequency and  
the pixel clock frequency, for a Two-PLL system.

2342     **Table 92 Clock Division and PLL Multiplier Registers for Two-PLL System**

Register Name	Type	RW	Comment
<b>vt_pix_clk_div</b>	16-bit unsigned integer	RW	Video Timing Pixel Clock Divider Value
<b>vt_sys_clk_div</b>	16-bit unsigned integer	RW	Video Timing System Clock Divider Value
<b>vt_pre_pll_clk_div</b>	16-bit unsigned integer	RW	Pre PLL clock Divider Value for VT domain
<b>op_pre_pll_clk_div</b>	16-bit unsigned integer	RW	Pre PLL clock Divider Value for OP domain
<b>vt_pll_multiplier</b>	16-bit unsigned integer	RW	PLL multiplier Value for VT domain
<b>op_pll_multiplier</b>	16-bit unsigned integer	RW	PLL multiplier Value for OP domain
<b>op_pix_clk_div</b>	16-bit unsigned integer	RW	Output Pixel Clock Divider Value
<b>op_sys_clk_div</b>	16-bit unsigned integer	RW	Output System Clock Divider Value

### 8.3.4.2 Register Definitions

2343 This Section specifies the set of CCI Registers defined for Clock Tree control.

2344 **Table 93 Video Timing Pixel Clock Divider Register**

Register Name	Type	RW	Comment
<b>vt_pix_clk_div</b>	16-bit unsigned integer	RW	<p>The video timing pixel clock divider</p> <p><b>Mandatory Values:</b></p> <ul style="list-style-type: none"> <li>For simple PLL system: <code>vt_pix_clk_div = 10</code></li> <li>For others: <code>vt_pix_clk_div = 4, 5, 6, 7, 8, 9, and 10</code></li> </ul> <p>In addition, smaller and bigger values may be supported by the image sensor if so indicated by the capability registers.</p> <p><b>Default:</b> 10</p>

2345 **Table 94 Video Timing System Clock Divider Register**

Register Name	Type	RW	Comment
<b>vt_sys_clk_div</b>	16-bit unsigned integer	RW	<p>Only even <b>vt_sys_clk_div</b> values are supported, with the exception of <b>vt_sys_clk_div = 1</b></p> <p><b>Examples:</b></p> <ul style="list-style-type: none"> <li>Code 1: Divide by 1</li> <li>Code 2: Divide by 2</li> <li>Code 4: Divide by 4</li> <li>Code 6: Divide by 6</li> </ul> <p><b>Default:</b> 1</p>

2346

**Table 95 Pre-PLL Clock Divider Registers**

Register Name	Type	RW	Comment
<b>pre_pll_clk_div</b>	16-bit unsigned integer	RW	<p>If bit 2 of register <b>clock_tree_pll_capability</b> is 0, then only even <b>pre_pll_clk_div</b> values are supported, with the exception of <b>pre_pll_clk_div</b> = 1.</p> <p>If bit 2 of register <b>clock_tree_pll_capability</b> is 1, then all integer values including and between the reported minimum and maximum values are supported.</p> <p><b>Examples:</b></p> <ul style="list-style-type: none"> <li>Code 1: Divide by 1</li> <li>Code 2: Divide by 2</li> <li>Code 4: Divide by 4</li> <li>Code 6: Divide by 6</li> </ul> <p><b>Default:</b> 1</p>
<b>vt_pre_pll_clk_div</b>	16-bit unsigned integer	RW	<p>If bit 2 of register <b>clock_tree_pll_capability</b> is 0, then only even <b>vt_pre_pll_clk_div</b> values are supported, with the exception of <b>vt_pre_pll_clk_div</b> = 1.</p> <p>If bit 2 of register <b>clock_tree_pll_capability</b> is 1, then all integer values including and between the reported minimum and maximum values are supported.</p> <p><b>Examples:</b></p> <ul style="list-style-type: none"> <li>Code 1: Divide by 1</li> <li>Code 2: Divide by 2</li> <li>Code 4: Divide by 4</li> <li>Code 6: Divide by 6</li> </ul> <p><b>Default:</b> 1</p>
<b>op_pre_pll_clk_div</b>	16-bit unsigned integer	RW	<p>If bit 2 of register <b>clock_tree_pll_capability</b> is 0, then only even <b>op_pre_pll_clk_div</b> values are supported, with the exception of <b>op_pre_pll_clk_div</b> = 1.</p> <p>If bit 2 of register <b>clock_tree_pll_capability</b> is 1, then all integer values including and between the reported minimum and maximum values are supported.</p> <p><b>Examples:</b></p> <ul style="list-style-type: none"> <li>Code 1: Divide by 1</li> <li>Code 2: Divide by 2</li> <li>Code 4: Divide by 4</li> <li>Code 6: Divide by 6</li> </ul> <p><b>Default:</b> 1</p>

2347

**Table 96 PLL Multiplier Registers**

Register Name	Type	RW	Comment
<b>pll_multiplier</b>	16-bit unsigned integer	RW	PLL multiplier value <b>Example:</b> Code 10: Multiply by 10 <b>Default:</b> Image-sensor-specific
<b>vt_pll_multiplier</b>	16-bit unsigned integer	RW	PLL multiplier value <b>Example:</b> Code 10: Multiply by 10 <b>Default:</b> Image-sensor-specific
<b>op_pll_multiplier</b>	16-bit unsigned integer	RW	PLL multiplier value <b>Example:</b> Code 10: Multiply by 10 <b>Default:</b> Image-sensor-specific

2348

**Table 97 Output Pixel Clock Divider Register**

Register Name	Type	RW	Comment
<b>op_pix_clk_div</b>	16-bit unsigned integer	RW	If bit 3 of the register <b>clock_tree_pll_capability</b> has the value 0, then the output pixel clock divider <b>op_pix_clk_div</b> = 6, 7, 8, 10, 12 and 14 <b>Default:</b> 10 Note that values 6 and 7 are not mandatory, and are only required if the optional RAW6 and RAW7 compressed modes are included. Also values 12 and 14 are optional, and are only required if such values are needed, for example due to HDR functionality supporting 12 bits or 14 bits output, or if the ADC output has more than 10 bits. If bit 3 of the register <b>clock_tree_pll_capability</b> has the value 1, then valid <b>op_pix_clk_div</b> values range from <b>min_op_pix_clk_div</b> to <b>max_op_pix_clk_div</b> (inclusive). See also <b>Section 8.3.2</b> for details. In case of simple PLL mode, this register is not supported, and shall have the value 0.

2349

**Table 98 Output System Clock Divider Register**

Register Name	Type	RW	Comment
<b>op_sys_clk_div</b>	16-bit unsigned integer	RW	Only even <b>op_sys_clk_div</b> values are supported (mandatory), with the exception of <b>op_sys_clk_div</b> = 1 <b>Examples:</b> 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 <b>Default:</b> 1 In case of simple PLL mode, this register is not supported, and shall have the value 0.

### 8.3.5 Clock Set-Up Capability Read Only Registers

A bank of Read Only CCI Registers fully describes the minimum and maximum limits for the clock frequencies, clock dividers, and PLL multipliers in an image sensor input clock. If the image sensor supports both One-PLL Mode and Two-PLL Mode, then the values of these registers are dynamic, and depend upon the value of the register **pll\_mode**. The register(s) may be omitted from the sensor if CCS Static Data provides the same information. See *Section B.2.8*.

In Lane speed mode all VT and OP domain speeds are per-Lane, including the limit capability registers in *Table 100*,

*Table 101*, and *Table 102*. Note that some registers have either 32-bit IEEE float or 32-bit unsigned iReal as a Type, depending upon the **clock\_capa\_type\_capability** capability register.

Example: For CSI-2 with 2 Lanes at 1Gbit/s per Lane, and a VT domain with 2 Lanes each at 100Mpix/s per Lane, the speeds within the Clock Tree are 1GHz and 100MHz. With System Speed model, these would be 2GHz and 200MHz. This example assumed coupled OP mode or decoupled OP mode with using 2 OP lanes. See Section 8.3.2 for details.

For Double Data Rate values (i.e., in the OP domain), the limit values represent real frequencies (see *Section 8.3.2.1*).

**Table 99 Clock Capability Type Register**

Register Name	Type	RW	Comment
<b>clock_capa_type_capability</b>	8-bit unsigned integer	RO	<b>Bit 0</b> 0: 32-bit IEEE float Type used 1: 32-bit unsigned iReal Type used <b>Other bits</b> Reserved for future use

**Table 100 Pre PLL and PLL Clock Set-Up Capability Registers**

Register Name	Type	RW	Comment
<b>min_ext_clk_freq_mhz</b>	32-bit IEEE float or 32-bit unsigned iReal	RO	Minimum external clock frequency <b>Units:</b> MHz
<b>max_ext_clk_freq_mhz</b>	32-bit IEEE float or 32-bit unsigned iReal	RO	Maximum external clock frequency <b>Units:</b> MHz
<b>min_pre_pll_clk_div</b>	16-bit unsigned integer	RO	Minimum Pre PLL divider value
<b>max_pre_pll_clk_div</b>	16-bit unsigned integer	RO	Maximum Pre PLL divider value
<b>min_vt_pre_pll_clk_div</b>	16-bit unsigned integer	RO	Minimum Pre PLL divider value
<b>max_vt_pre_pll_clk_div</b>	16-bit unsigned integer	RO	Maximum Pre PLL divider value
<b>min_op_pre_pll_clk_div</b>	16-bit unsigned integer	RO	Minimum Pre PLL divider value

Register Name	Type	RW	Comment
<b>max_op_pre_pll_clk_div</b>	16-bit unsigned integer	RO	Maximum Pre PLL divider value
<b>min_pll_ip_freq_mhz</b>	32-bit IEEE float or 32-bit unsigned iReal	RO	Minimum PLL input clock frequency <b>Units:</b> MHz
<b>max_pll_ip_freq_mhz</b>	32-bit IEEE float or 32-bit unsigned iReal	RO	Maximum PLL input clock frequency <b>Units:</b> MHz
<b>min_vt_pll_ip_freq_mhz</b>	32-bit IEEE float or 32-bit unsigned iReal	RO	Minimum PLL input clock frequency <b>Units:</b> MHz
<b>max_vt_pll_ip_freq_mhz</b>	32-bit IEEE float or 32-bit unsigned iReal	RO	Maximum PLL input clock frequency <b>Units:</b> MHz
<b>min_op_pll_ip_freq_mhz</b>	32-bit IEEE float or 32-bit unsigned iReal	RO	Minimum PLL input clock frequency <b>Units:</b> MHz
<b>max_op_pll_ip_freq_mhz</b>	32-bit IEEE float or 32-bit unsigned iReal	RO	Maximum PLL input clock frequency <b>Units:</b> MHz
<b>min_pll_multiplier</b>	16-bit unsigned integer	RO	Minimum PLL multiplier
<b>max_pll_multiplier</b>	16-bit unsigned integer	RO	Maximum PLL Multiplier
<b>min_vt_pll_multiplier</b>	16-bit unsigned integer	RO	Minimum PLL multiplier
<b>max_vt_pll_multiplier</b>	16-bit unsigned integer	RO	Maximum PLL Multiplier
<b>min_op_pll_multiplier</b>	16-bit unsigned integer	RO	Minimum PLL multiplier
<b>max_op_pll_multiplier</b>	16-bit unsigned integer	RO	Maximum PLL Multiplier
<b>min_pll_op_freq_mhz</b>	32-bit IEEE float or 32-bit unsigned iReal	RO	Minimum PLL output clock frequency <b>Units:</b> MHz
<b>max_pll_op_freq_mhz</b>	32-bit IEEE float or 32-bit unsigned iReal	RO	Maximum PLL output clock frequency <b>Units:</b> MHz
<b>min_vt_pll_op_freq_mhz</b>	32-bit IEEE float or 32-bit unsigned iReal	RO	Minimum PLL output clock frequency <b>Units:</b> MHz
<b>max_vt_pll_op_freq_mhz</b>	32-bit IEEE float or 32-bit unsigned iReal	RO	Maximum PLL output clock frequency <b>Units:</b> MHz

Register Name	Type	RW	Comment
<b>min_op_pll_op_freq_mhz</b>	32-bit IEEE float or 32-bit unsigned iReal	RO	Minimum PLL output clock frequency <b>Units:</b> MHz
<b>max_op_pll_op_freq_mhz</b>	32-bit IEEE float or 32-bit unsigned iReal	RO	Maximum PLL output clock frequency <b>Units:</b> MHz

2367

**Table 101 Video Timing Clock Set-Up Capability Registers**

Register Name	Type	RW	Comment
<b>min_vt_sys_clk_div</b>	16-bit unsigned integer	RO	Minimum video timing system clock divider value
<b>max_vt_sys_clk_div</b>	16-bit unsigned integer	RO	Maximum video timing system clock divider value
<b>min_vt_sys_clk_freq_mhz</b>	32-bit IEEE float or 32-bit unsigned iReal	RO	Minimum video timing system clock frequency <b>Units:</b> MHz
<b>max_vt_sys_clk_freq_mhz</b>	32-bit IEEE float or 32-bit unsigned iReal	RO	Maximum video timing system clock frequency <b>Units:</b> MHz
<b>min_vt_pix_clk_freq_mhz</b>	32-bit IEEE float or 32-bit unsigned iReal	RO	Minimum video timing pixel clock frequency <b>Units:</b> MHz
<b>max_vt_pix_clk_freq_mhz</b>	32-bit IEEE float or 32-bit unsigned iReal	RO	Maximum video timing pixel clock frequency <b>Units:</b> MHz
<b>min_vt_pix_clk_div</b>	16-bit unsigned integer	RO	Minimum video timing pixel clock divider value
<b>max_vt_pix_clk_div</b>	16-bit unsigned integer	RO	Maximum video timing pixel clock divider value

2368

**Table 102 Output Clock Set-Up Capability Registers**

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

For D-PHY, the bitrate limits for all Lane configurations that the image sensor uses shall be given in the following set of registers. The Host shall ensure that the values in these registers are not violated with the used PHY, especially for combination devices (i.e., where a single image sensor supports both D-PHY and C-PHY).

If all of these registers would contain the same value, then the image sensor may report that value only in the first register (**max\_per\_lane\_bitrate\_1\_lane\_d\_mode\_mbps**), and report the value 0 in all the other registers.

2376

**Table 103 D-PHY Per-Lane Bitrate Limit Registers**

Register Name	Type	RW	Comment
<b>max_per_lane_bitrate_1_lane_d_mode_mbps</b>	32-bit unsigned iReal	RO	Maximum bitrate per Lane in 1-Lane D-PHY mode <b>Units:</b> Mbps
<b>max_per_lane_bitrate_2_lane_d_mode_mbps</b>	32-bit unsigned iReal	RO	Maximum bitrate per Lane in 2-Lane D-PHY mode <b>Units:</b> Mbps
<b>max_per_lane_bitrate_3_lane_d_mode_mbps</b>	32-bit unsigned iReal	RO	Maximum bitrate per Lane in 3-Lane D-PHY mode <b>Units:</b> Mbps
<b>max_per_lane_bitrate_4_lane_d_mode_mbps</b>	32-bit unsigned iReal	RO	Maximum bitrate per Lane in 4-Lane D-PHY mode <b>Units:</b> Mbps
<b>max_per_lane_bitrate_5_lane_d_mode_mbps</b>	32-bit unsigned iReal	RO	Maximum bitrate per Lane in 5-Lane D-PHY mode <b>Units:</b> Mbps
<b>max_per_lane_bitrate_6_lane_d_mode_mbps</b>	32-bit unsigned iReal	RO	Maximum bitrate per Lane in 6-Lane D-PHY mode <b>Units:</b> Mbps
<b>max_per_lane_bitrate_7_lane_d_mode_mbps</b>	32-bit unsigned iReal	RO	Maximum bitrate per Lane in 7-Lane D-PHY mode <b>Units:</b> Mbps
<b>max_per_lane_bitrate_8_lane_d_mode_mbps</b>	32-bit unsigned iReal	RO	Maximum bitrate per Lane in 8-Lane D-PHY mode <b>Units:</b> Mbps

For C-PHY, the Symbol rate limits for all Lane configurations that the image sensor uses shall be given in the following set of registers. The Host shall ensure that the values in these registers are not violated with the used PHY, especially for combination devices (i.e., where a single image sensor supports both D-PHY and C-PHY). See *Section 8.3.2.2* for details of Symbol rate and bitrate.

If all of these registers would contain the same value, then the image sensor may report that value only in the first register (**max\_per\_lane\_symbolrate\_1\_lane\_c\_mode\_mbps**), and report the value 0 in all the other registers.

**Table 104 C-PHY Per-Lane Symbol Rate Limit Registers**

Register Name	Type	RW	Comment
<b>max_per_lane_symbolrate_1_lane_c_mode_msp</b>	32-bit unsigned iReal	RO	Maximum Symbol rate per Lane in 1-Lane C-PHY mode <b>Units:</b> Msym/s
<b>max_per_lane_symbolrate_2_lane_c_mode_msp</b>	32-bit unsigned iReal	RO	Maximum Symbol rate per Lane in 2-Lane C-PHY mode <b>Units:</b> Msym/s
<b>max_per_lane_symbolrate_3_lane_c_mode_msp</b>	32-bit unsigned iReal	RO	Maximum Symbol rate per lane in 3-Lane C-PHY mode <b>Units:</b> Msym/s
<b>max_per_lane_symbolrate_4_lane_c_mode_msp</b>	32-bit unsigned iReal	RO	Maximum Symbol rate per Lane in 4-Lane C-PHY mode <b>Units:</b> Msym/s
<b>max_per_lane_symbolrate_5_lane_c_mode_msp</b>	32-bit unsigned iReal	RO	Maximum Symbol rate per Lane in 5-Lane C-PHY mode <b>Units:</b> Msym/s
<b>max_per_lane_symbolrate_6_lane_c_mode_msp</b>	32-bit unsigned iReal	RO	Maximum Symbol rate per Lane in 6-Lane C-PHY mode <b>Units:</b> Msym/s
<b>max_per_lane_symbolrate_7_lane_c_mode_msp</b>	32-bit unsigned iReal	RO	Maximum Symbol rate per lane in 7-Lane C-PHY mode <b>Units:</b> Msym/s
<b>max_per_lane_symbolrate_8_lane_c_mode_msp</b>	32-bit unsigned iReal	RO	Maximum Symbol rate per Lane in 8-Lane C-PHY mode <b>Units:</b> Msym/s

### 8.3.6 Clock Frequency Requirements

This Section defines requirements governing image sensor clock frequencies.

The purposes of these requirements are:

- To limit the range of the pre-PLL clock divider values (1, 2, 4, 6, etc.), in order to keep the PLL input frequency within the range from `min_pll_ip_freq_mhz` through  $2 * \text{min\_pll\_ip\_freq\_mhz}$ . This limitation necessary in order to minimize variation in the pixel clock frequency, across the full range of input clock frequencies.
- To allow the readout rate of the image sensor's full resolution to be smoothly reduced by at least a factor of 2 from the maximum readout rate

For example, from 30fps to at least 15fps

The requirements for the image sensor clock frequencies are:

1. The external clock input frequency range should be from 6.0 MHz through 27.0 MHz.

$$\text{min\_ext\_clk\_freq\_mhz} = 6.0 \text{ MHz}$$

$$\text{max\_ext\_clk\_freq\_mhz} = 27.0 \text{ MHz}$$

2. The maximum PLL input frequency shall be at least twice the minimum PLL input frequency.

$$\text{max\_pll\_ip\_freq\_mhz} \geq 2 * \text{min\_pll\_ip\_freq\_mhz}$$

3. The maximum PLL output frequency shall 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.

$$\text{max\_pll\_op\_freq\_mhz} \geq 2 * (\text{min\_pll\_op\_freq\_mhz} + \text{var\_pll\_op\_freq\_mhz})$$

or

$$\text{min\_pll\_op\_freq\_mhz} \leq (\text{max\_pll\_op\_freq\_mhz}/2) - \text{var\_pll\_op\_freq\_mhz}$$

where the variation in the PLL output frequency is:

$$\text{var\_pll\_op\_freq\_mhz} = 2 * \text{min\_pll\_ip\_freq\_mhz}$$

4. The maximum video timing system clock frequency shall 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:

$$\text{max\_vt\_sys\_clk\_freq\_mhz} \geq 2 * (\text{min\_vt\_sys\_clk\_freq\_mhz} + \text{var\_vt\_sys\_clk\_freq\_mhz})$$

or

$$\text{min\_vt\_sys\_clk\_freq\_mhz} \leq (\text{max\_vt\_sys\_clk\_freq\_mhz}/2) - \text{var\_vt\_sys\_clk\_freq\_mhz}$$

where the variation in the system clock is

$$\text{var\_vt\_sys\_clk\_freq\_mhz} = (2 * \text{min\_pll\_ip\_freq\_mhz}) / \text{vt\_sys\_clk\_div}$$

5. The maximum video timing pixel clock frequency shall 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

$$\text{max\_vt\_pix\_clk\_freq\_mhz} \geq 2 * (\text{min\_vt\_pix\_clk\_freq\_mhz} + \text{var\_vt\_pix\_clk\_freq\_mhz})$$

or

$$\text{min\_vt\_pix\_clk\_freq\_mhz} \leq (\text{max\_vt\_pix\_clk\_freq\_mhz}/2) - \text{var\_vt\_pix\_clk\_freq\_mhz}$$

where the variation in the pixel clock is

$$\text{var\_vt\_pix\_clk\_freq\_mhz} = (2 * \text{min\_pll\_ip\_freq\_mhz}) / (\text{vt\_sys\_clk\_div} * \text{vt\_pix\_clk\_div})$$

## 8.4 Overall Video Timing Requirements

The typical targets for the readout are:

- It is recommended to support full resolution image readout at 30 fps as a maximum readout speed with a minimum frame banking period of 500  $\mu$ s.
- This 30 fps readout is recommended to minimize effects of motion distortion and image tearing in image sensor using a rolling blade electronic shutter.
- The maximum nominal readout speed (e.g. 1/30 s) shall be achievable across the full range of external clock frequencies
- For image sensor resolutions whose bandwidth requirement is beyond the data rate limit of CSI-2 capacity, the maximum camera module readout rate may be de-rated. Image sensor shall always support de-rating if image sensor supports scaler without dependency whether scaler is used or not.
- Recommended supportable frames rates are:

30 fps, 25 fps, 24 fps, 20 fps, 15 fps, 12.5 fps, 12 fps, 10 fps, 7.5 fps, 6.25 fps, 6 fps, 5 fps.

In addition to those, special high frame rate modes like 240 fps, 120 fps, 90 fps and 60 fps for video use cases are often seen beneficial.

The requirements for the image sensor video timing are:

- The readout rate shall 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.

This is necessary to give improved compatibility with the data rate capabilities of different Host CSI receiver implementations, system performance issues and e.g. due EMC issues.

- Image sensor shall be able to work down to a frame blanking time of 0.5 ms

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 image sensor is increased to provide the required frame blanking time.

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

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

Coarse and fine Integration Times shall be based on the timings for the image readout from the pixel array i.e. based on video timings and not based on the output pixel clock timings.

## 9 Integration Time and Gain Control

### 9.1 Overview

This Section defines image sensor Integration Time, analog Gain control, and digital Gain control, including:

- Integration Time control parameters
- Analog Gain control parameters
- Digital Gain control parameters
- Determining time and Gain control parameter limits for a particular image sensor
- Rules governing the ‘consumption’ of changes in time and/or Gain settings
- Mechanism for Synchronizing time and/or Gain parameter changes
- Advanced timing modes
- Auto-bracketing function
- HDR timing and synthesis

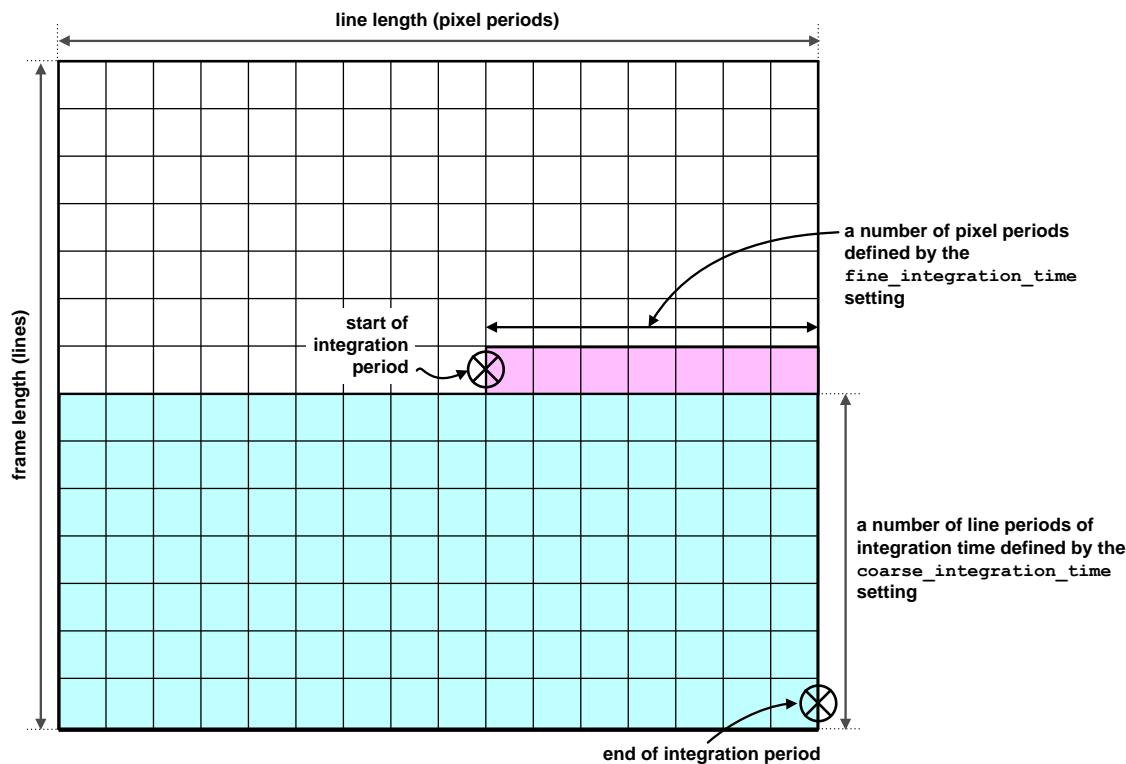
### 9.2 Integration Time Control

With rolling-shutter image sensors, Exposure time is controlled by the concept of Integration Time. CCS implements Integration Time as the two integer control parameters **coarse\_integration\_time** and **fine\_integration\_time** (see *Table 105* and *Figure 48*). The parameter **coarse\_integration\_time** sets the integer number of (complete) image sensor line periods in the Integration Time. The optional **fine\_integration\_time** parameter add additional time to the Integration Time, measured as an integer number of image sensor pixel periods. The Host can determine whether the image sensor supports **fine\_integration\_time** by reading the register **integration\_time\_capability**.

Timing rules for Integration Time, and its impact upon the image stream, are covered in *Section 9.5*.

**Table 105 Integration Time Control Parameters**

Parameter Name	Type	RW	Default	Units	Function
<b>fine_integration_time</b>	16-bit unsigned integer	RW	Image sensor dependent	Image sensor pixel periods	The ‘fine’ Integration Time control <b>Optional</b>
<b>coarse_integration_time</b>	16-bit unsigned integer	RW	Image sensor dependent	Image sensor line periods in VT time domain	The ‘coarse’ Integration Time control



2477

2478

**Figure 48 Integration Time**

### 9.2.1 Calculation of Integration Time

Calculation of the total Integration Time depends upon which Speed Model the image sensor is using. (See [Section 8.3](#) for definitions of System Speed Model and Lane Speed Model.)

For an image sensor using the System Speed Model, the total Integration Time  $t_{TI}$  is given by:

$$t_{TI} = [(\text{coarse\_integration\_time} * \text{pixel\_periods\_per\_line}) + \text{fine\_integration\_time}] * \text{pix\_clk\_period}$$

Where

$$\text{pix\_clk\_period} = 1 / \text{vt\_pix\_clk\_freq\_mhz}$$

and

$$\text{pixel\_periods\_per\_line} = \text{line\_length\_pck}$$

For an image sensor using the Lane Speed Model, the total Integration Time  $t_{TI}$  is given by:

$$t_{TI} = [(\text{coarse\_integration\_time} * \text{pixel\_periods\_per\_line}) + \text{fine\_integration\_time}] * \text{pix\_clk\_period}$$

Where

$$\text{pix\_clk\_period} = 1 / (\text{vt\_pix\_clk\_freq\_mhz} * \text{num\_of\_vt\_lanes})$$

and

$$\text{pixel\_periods\_per\_line} = \text{line\_length\_pck}$$

### 9.3 Analog Gain Control

This specification supports global analog Gain, i.e. an analog Gain factor that is applied equally to all channels.

Timing rules for analog Gain control, and its impact upon the image stream, are covered in [Section 9.5](#).

CCS supports two Gain modes for global analog Gain, as CCS v1.1 introduces Alternate Global Gain. The image sensor shall support one or the other of these Gain modes (i.e., either Global Gain mode or Alternate Global Gain mode), and shall report which one is supported via register **analog\_gain\_capability** ([Table 106](#)).

The image sensor exposes which Gain type it supports via the register **analog\_gain\_type** ([Table 106](#)).

**Table 106 Analog Gain Capability Parameters**

Parameter Name	Type	RW	Function
<b>analog_gain_capability</b>	16-bit unsigned integer	RO	Describes image sensor analog Gain capabilities 0: Global Gain only 1: Reserved 2: Alternate Global Gain
<b>analog_gain_type</b>	16-bit unsigned integer	RO	Analog Gain coding type 0: <ul style="list-style-type: none"><li>• If <b>analog_gain_capability</b> = 0, then coding scheme is as defined in <a href="#">Section 9.3.1</a></li><li>• If <b>analog_gain_capability</b> = 2, then coding scheme is as defined in <a href="#">Section 9.3.2</a></li></ul> Other Values: Reserved

The selection between the Gain modes is done by register **gain\_mode** ([Table 107](#)).

**Table 107 Gain Mode Register**

Parameter Name	Type	RW	Function
<b>gain_mode</b>	8-bit unsigned integer	RW	Gain model to be used by Host. 0: Global analog Gain 1: Reserved <b>Default: 0</b>

Image sensors may optionally implement modes requiring additional and/or different Gain characteristics or controls (for example, a coarse global Gain preceded or followed by multiple fine, per-channel Gains), however any such controls shall be controlled via Manufacturer-Specific Registers (see [Section 20.9](#)).

### 9.3.1 Global Analog Gain

Starting with CCS v1.0, global analog Gain can be controlled by register **analog\_gain\_code\_global** if the image sensor supports it.

**Table 108 Analog Gain Control Parameters**

Parameter Name	Type	RW	Default	Function
<b>analog_gain_code_global</b>	16-bit unsigned integer	RW	Image sensor dependent	A control for analog Gain that is applied to all channels (e.g. Bayer channels)

The relationship between the integer Gain parameter and the resulting Gain multiplier is given by the following equation:

$$gain = \frac{m_0x + c_0}{m_1x + c_1}$$

where 'x' is the integer analog Gain control parameter, and  $m_0$ ,  $c_0$ ,  $m_1$ , and  $c_1$  are image-sensor-specific constants exposed by the sensor. These constants are static parameters, and for any given image sensor either  $m_0$  or  $m_1$  shall be zero.

The full Gain equation therefore reduces to either:

$$gain = \frac{c_0}{m_1x + c_1}$$

Or:

$$gain = \frac{m_0x + c_0}{c_1}$$

2519

**Table 109 Analog Gain Coding/Decoding Constants**

Parameter Name	Type	RW	Function
<b>analog_gain_type</b>	16-bit unsigned integer	RO	Analog Gain coding type 0: Coding scheme as defined in this Section Other values: Reserved <b>Default:</b> 0 (required)
<b>analog_gain_m0</b>	16-bit signed integer	RO	m0, a constant used in the analog Gain control coding/decoding. <b>Note:</b> Either m0 or m1 shall = zero
<b>analog_gain_c0</b>	16-bit signed integer	RO	c0, a constant used in the analog Gain control coding/decoding.
<b>analog_gain_m1</b>	16-bit signed integer	RO	m1, a constant used in the analog Gain control coding/decoding <b>Note:</b> Either m0 or m1 shall = zero
<b>analog_gain_c1</b>	16-bit signed integer	RO	c1, a constant used in the analog Gain control coding/decoding.

2520 The image sensor exposes limits on the use of the analog Gain control. The Host can read the parameters  
 2521 defining the minimum and maximum recommended settings for the Gain control, from the registers listed in  
 2522 *Table 110*.

2523

**Table 110 Gain Parameter Limits**

Parameter Name	Type	RW	Function
<b>analog_gain_code_min</b>	16-bit unsigned integer	RO	The minimum recommended setting for the analog Gain control
<b>analog_gain_code_max</b>	16-bit unsigned integer	RO	The maximum recommended setting for the analog Gain control
<b>analog_gain_code_step_size</b>	16-bit unsigned integer	RO	The precision of the analog Gain control

2524 Limits for the analog Gain controls are given by the absolute minimum and absolute maximum recommended  
 2525 registers: **analog\_gain\_code\_min** and **analog\_gain\_code\_max**. All CCS image sensors should support  
 2526 analog Gain control with, for example 1 - 8x Gain, even if it is possible to set the minimum and maximum  
 2527 settings to the same value (effectively not supporting a variable Gain).

### 9.3.2 Alternate Global Analog Gain

Starting with CCS v1.1, Alternate Global Analog Gain is also available. If the image sensor supports it, then the global analog Gain can be controlled by linear and exponential gain formula:

$$gain = analog\_linear\_gain\_global * 2^{analog\_exponential\_gain\_global}$$

where *analog\_linear\_gain\_global* and *analog\_exponential\_gain\_global* are as defined in **Table 111**.

**Table 111 Alternate Analog Gain Control Parameters**

Parameter Name	Type	RW	Default	Function
<b>analog_linear_gain_global</b>	16-bit unsigned iReal	RW	Image sensor dependent	A control for analog Gain that is applied to all channels
<b>analog_exponential_gain_global</b>	16-bit signed iReal	RW	Image sensor dependent	A control for analog Gain that is applied to all channels

The image sensor exposes limits on the use of the analog Gain control. The Host can read the parameters defining the minimum and maximum recommended settings for the Gain control from the registers listed in **Table 112**.

**Table 112 Alternate Gain Parameter Limits**

Parameter Name	Type	RW	Function
<b>analog_linear_gain_min</b>	16-bit unsigned iReal	RO	The minimum recommended setting for the linear Gain control
<b>analog_linear_gain_max</b>	16-bit unsigned iReal	RO	The maximum recommended setting for the linear Gain control
<b>analog_linear_gain_step_size</b>	16-bit unsigned iReal	RO	The precision of the linear Gain control
<b>analog_exponential_gain_min</b>	16-bit signed iReal	RO	The minimum recommended setting for the exponential Gain control
<b>analog_exponential_gain_max</b>	16-bit signed iReal	RO	The maximum recommended setting for the exponential Gain control
<b>analog_exponential_gain_step_size</b>	16-bit signed iReal	RO	The precision of the exponential Gain control

2537 The exponential gain parameters use the signed 16-bit iReal format.

2538 **Examples:**

2539    1.5 = 0x0180

2540    -1.5 = 0xFE80 (2's complement)

2541 A simple exponential gain mode,  $2^x$ , where  $x = \{-1, 0, 1, 2, 3, 4\}$ , could have the following as limit  
2542 parameters:

- 2543    • **analog\_linear\_gain\_min** = 1.0 = 0x0100
- 2544    • **analog\_linear\_gain\_max** = 1.0 = 0x0100
- 2545    • **analog\_linear\_gain\_step\_size** = 0
- 2546    • **analog\_exponential\_gain\_min** = -1.0 = 0xFF00
- 2547    • **analog\_exponential\_gain\_max** = 4.0 = 0x0400
- 2548    • **analog\_exponential\_gain\_step\_size** = 1.0 = 0x0100

2549 Limits for the analog Gain controls are given by the absolute minimum and absolute maximum recommended  
2550 registers as defined in *Table 112*. All CCS image sensors should support analog Gain control with, for  
2551 example 1–8x Gain, even if it is possible to set the minimum and maximum settings to the same value  
2552 (effectively not supporting a variable Gain).

## 9.4 Digital Gain Control

The Digital Gain control feature is optional, and recommended. The Host can detect whether the image sensor supports Digital Gain control by reading register **`digital_gain_capability`**.

### 9.4.1 Digital Gain Control Parameters

Digital Gain of the channels is controlled by a single, global digital Gain control (*Table 113*). When digital Gain is applied, the LSB(s) of the resulting data shall be padded with zeros.

**Table 113 Digital Gain Control Parameters**

Parameter Name	Type	RW	Default	Function
<b><code>digital_gain_global</code></b>	16-bit unsigned iReal	RW	0x0100	A control for digital Gain that is applied to all channels (e.g. Bayer channels)

**Table 114 Digital Gain Capability**

Parameter Name	Type	RW	Comment
<b><code>digital_gain_capability</code></b>	8-bit unsigned integer	RO	0: None 1: Reserved 2: Global gains

#### 9.4.1.1 Control Parameter Change Timing

The image sensor shall manage the exact timing of the effect of digital Gain parameter changes on the output pixel stream. If necessary, the image sensor shall delay a Gain parameter change in order to put the change into effect at a frame boundary. If grouped together with other parameter changes, the image sensor shall delay the changes such that all of them take effect simultaneously at the same boundary. See *Section 9.5* for more details.

#### 9.4.2 Digital Gain Parameter Limits

The image sensor shall expose the range and precision of the digital Gain controls via three static parameters: **`digital_gain_min`**, **`digital_gain_max`**, and **`digital_gain_step_size`** (*Table 115*). If the image sensor does not support digital Gain, then these registers are optional.

Register **`digital_gain_step_size`** represents the smallest step in digital Gain supported by the image sensor. Both **`digital_gain_min`** and **`digital_gain_max`** shall be integer multiples of **`digital_gain_step_size`**.

**Table 115 Digital Gain Limit & Precision Parameters**

Parameter Name	Type	RW	Function
<b><code>digital_gain_min</code></b>	16-bit unsigned iReal	RO	The minimum valid limit of the digital Gain control parameters
<b><code>digital_gain_max</code></b>	16-bit unsigned iReal	RO	The maximum valid limit of the digital Gain control parameters
<b><code>digital_gain_step_size</code></b>	16-bit unsigned iReal	RO	Defines the resolution of the digital Gain control parameters

The digital Gain control parameter 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).

#### 9.4.3 Digital Gain and Black Level

**Important:** Use of the digital Gain setting shall not affect the pixel data Black Level.

## 9.5 Re-Timing of the Gain and Integration Time Controls

In order to ensure consistent settings within each image data frame, and that groups of related parameters can be changed in a synchronized manner, the image sensor shall control the exact time at which changes to certain ‘Re-timed’ parameters (i.e., those registers designated as ‘re-timed’ in the *Section 20* register map) take effect. When handling changes in these ‘re-timed’ parameters, the image sensor shall follow the re-timing rules specified in this Section. The image sensor shall also support the *Annex A.2* re-timing examples.

### 9.5.1 Re-Timing Rule 1: Re-Time Changes to Frame Boundaries

Effective changes to all re-timed parameters shall occur (i.e., shall first affect the output image data stream) on a frame boundary, irrespective of when the corresponding parameter change message(s) arrive. Re-timed parameter changes shall not cause image discontinuities. See Rule 4.

### 9.5.2 Re-Timing Rule 2: Specify the Latching Moment

Image sensor internal operations inevitably require a finite amount of latching time between setting a register and putting the corresponding image parameter into effect in the output image data stream. It also takes time to release group parameter hold. This means that there is some interval immediately before every frame boundary, during which it is too late for a register change to be re-timed to the next image frame. That is, there is insufficient time for the image parameter corresponding to the register to take effect at the next frame boundary. The start of this interval can be considered the ‘latching moment’. It can be advantageous for the Host to be aware of this re-timing constraint.

The timing of the latching moment is image-sensor-specific, however the latching moment typically occurs at a specific interval before a frame start packet. The image sensor vendor should specify (perhaps via a formula) the time interval between the latching moment and the frame start packet in the image sensor datasheet.

### 9.5.3 Re-Timing Rule 3: Grouped Parameter Hold

The image sensor shall support the grouped parameter hold mechanism, which synchronizes the moment of effect of any number of changes to the re-timed registers. The result is that all of the changes in the group of changes take effect simultaneously, at the same image frame.

To group multiple changes to re-timed registers together, the Host can:

1. Write 0x01 (the ‘Hold’ state) to register **grouped\_parameter\_hold** (*Table 116*)

This will postpone all subsequent changes to re-timed registers from taking effect until the Hold state is removed.

2. Make desired changes to re-timed registers

No parameter changes will take effect at this time. There is no limit on how long the image sensor can be left in the Hold state, or how many re-timed registers can be changed while in the Hold state.

3. Write 0x00 (the ‘No hold’ state) to register **grouped\_parameter\_hold**

This removes the Hold state. All changes to re-timed registers that were made in step 2 will take effect simultaneously, on the same frame boundary.

**Table 116 Grouped Change Control Parameter**

Parameter Name	Type	RW	Default	Function
<b>grouped_parameter_hold</b>	8-bit unsigned integer	RW	0: No hold	Set to envelope a series of parameter changes as a group of changes that shall be made so as to effect the output stream on the same frame boundary

### 9.5.4 Re-Timing Rule 4: Masking of Corrupted Frames

In some circumstances when changing parameters while streaming, even re-timing changes to a frame boundary might not avoid corruption of subsequent image frames. For example, Integration Time might not be constant across all lines in the frame. In particular, changing **line\_length\_pck** or **y\_addr\_start** and **y\_addr\_end** might produce corruption. The control parameter **mask\_corrupted\_frames** (*Table 117*) allows the Host to tell the image sensor whether to send such corrupted frames, or to mask them.

The image sensor shall support the register **mask\_corrupted\_frames**. The Host shall only program the register **mask\_corrupted\_frames** while in SW Standby Mode. If the parameter is set to 0, the image sensor shall output the corrupt frames. If the parameter is set to 1, the image sensor shall blank all corrupt frames as specified in *Section 9.5.4.1*. The image sensor shall mask only frames that are actually corrupt.

**Example:** If the mask bit is set, and then **line\_length\_pck** is written in mid-frame, the remainder of that frame won't be masked because it's not corrupt. This is because **line\_length\_pck** is a re-timed register, and therefore its change won't take effect during this frame.

Changing configuration of the image sensor shall result in a maximum of one corrupted frame.

When a masking condition occurs and frames are masked, the frame counter shall skip the masked frame(s) (i.e. the frame counter shall increment for masked frames). The Host shall interpret a skipped frame counter number as a corrupt frame.

**Example:** If the frame before a masked corrupt frame has number N, then the number N+1 will not be used and the frame after the masked frame will have number N+2.

The image sensor shall not produce masking as a result of changes to the following AEC-related registers:

- **fine\_integration\_time**
- **coarse\_integration\_time**
- **analog\_gain\_code\_global**
- **analog\_linear\_gain\_global**
- **analog\_exponential\_gain\_global**
- **digital\_gain\_global**
- **frame\_length\_lines**

**Table 117 Mask Corrupted Frames Register**

Parameter Name	Type	RW	Default	Function
<b>mask_corrupted_frames</b>	8-bit unsigned integer	RW	1	0: Allow corrupted frames 1: Mask corrupted frames

#### 9.5.4.1 Blanking of Corrupt Frames

When blanking corrupt frames, the image sensor shall use the following methods:

- For D-PHY: By putting the CSI-2 data Lanes into the LP-11 state
- For C-PHY: By putting the CSI-2 Lanes into the LP-111 state, in inter-frame when applicable

### 9.5.5 Re-Timing Rule 5: Image Parameter Update Time Limit

In order to limit system response delays:

- The image sensor's delay between a write to a re-timed register and the resulting change to the corresponding output image parameter shall not exceed two frames.
- The image sensor's delay between a write of 'No hold' to register **grouped\_parameter\_hold** and the resulting synchronized change to all of the accumulated output image parameters shall not exceed two frames.

### 9.5.6 Re-Timing Rule 6: Integration Time Registers

Longer delays apply for the Integration Time and related registers:

- The image sensor's delay between a write to the Integration Time registers and the resulting change to the output image parameters shall be exactly two frames.
- The image sensor's delay between a write to registers **frame\_length\_lines** and **line\_length\_pck** and the resulting change to the output image parameters shall be exactly two frames.
- If the image sensor supports registers **exposure\_ratio** and/or **short\_coarse\_integration\_time**, then the delay between a write to the register and the resulting change to the output image parameters shall be exactly two frames.

### 9.5.7 Re-Timing Rule 7: Gain Registers

There are two different possible image sensor behaviors when changing Gain:

- When both Gain and Integration Time are changed with **grouped\_parameter\_hold**, there will be a delay of two frames. When only Integration Time is changed, there will also be a delay of two frames.
- When only analog and/or digital Gain settings are changed, the delay is either:
  - Two frames (i.e., the same delay as for Integration Time) (fixed Gain delay), or
  - One frame (variable Gain delay).

Both of these options are compatible with CCS, and the image sensor shall support either option. The Host can determine which Gain delay type the image sensor supports by inspecting register **gain\_delay\_type**.

**Table 118 Gain Delay Type Register**

Register Name	Type	RW	Comment
<b>gain_delay_type</b>	8-bit unsigned integer	RO	0: Fixed Gain delay 1: Variable Gain delay

## 9.6 Control Synchronization

For closed loop control of an image sensor to succeed, the timing relationship between changes in image sensor control parameters and the effect on the output image stream must 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, this Specification takes a simpler approach that is designed to be both more flexible and more robust when used with asynchronous Hosts (or Hosts with other real-time responsibilities).

Control system Synchronization with a MIPI CCS compatible sensor should rely on the inclusion of status information within each image frame. In every image frame, each piece of status information that defines the setting of a control parameter shall reflect the setting of that parameter that was used in the generation of that frame (irrespective of consumption delays). This requirement is key to the success of the system.

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 image sensor requesting a change in a control parameter, the Host can detect the point at which the change takes effect by monitoring the status information.

For details of which parameters are included in the control synchronization scheme, see the CCI Register map in *Section 20*, and *Section 7.13*.

### 9.6.1 Time Parameter Limit Discovery

Although a MIPI CCS image sensor shall obey a model of Integration Time, there is flexibility in the range of control values that it shall support.

In order for the Host system to be able to automatically identify these limits, the image sensor shall make the necessary information available via the CCI Registers. The register(s) may be omitted from the sensor if CCS Static Data provides the same information. See *Section B.2.8*.

The CCS process for discovering the Exposure control parameter limits 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 disadvantage of this approach is that some of the limits are defined relative to other parameters. For example, the maximum **coarse\_integration\_time** setting is defined relative to the number of lines in the frame. For this reason, the Host shall re-evaluate these relative limits every time it changes the video timing set-up.

The Host can read the parameters defining the minimum and maximum recommended settings for each of the Integration Time controls from the registers listed in *Table 119*.

**Table 119 Integration Time Parameter Limits**

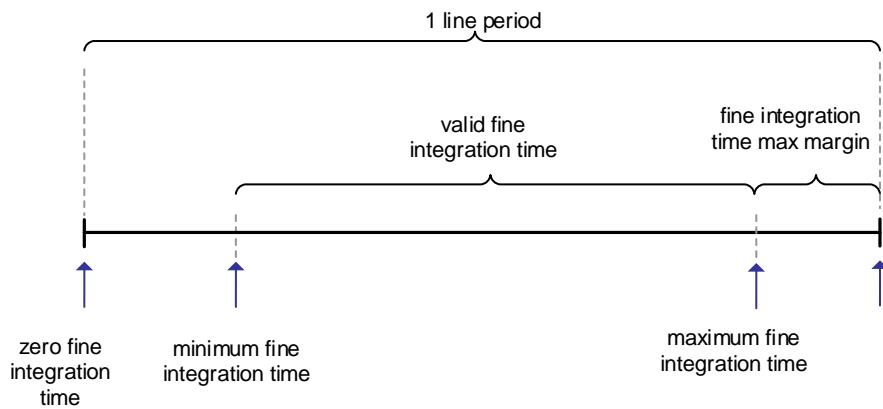
Parameter Name	Type	RW	Function
<b>fine_integration_time_min</b>	16-bit unsigned integer	RO	The minimum recommended setting for the fine Integration Time control
<b>fine_integration_time_max_margin</b>	16-bit unsigned integer	RO	A parameter that can be used to defined the maximum recommended setting for the fine Integration Time control
<b>coarse_integration_time_min</b>	16-bit unsigned integer	RO	The minimum recommended value for the coarse Integration Time control
<b>coarse_integration_time_max_margin</b>	16-bit unsigned integer	RO	A parameter that can be used to determine the maximum recommended setting for the coarse Integration Time control

For Integration Time, the minimum recommended settings for both controls are defined by registers **fine\_integration\_time\_min** and **coarse\_integration\_time\_min**. The maximum settings are 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:

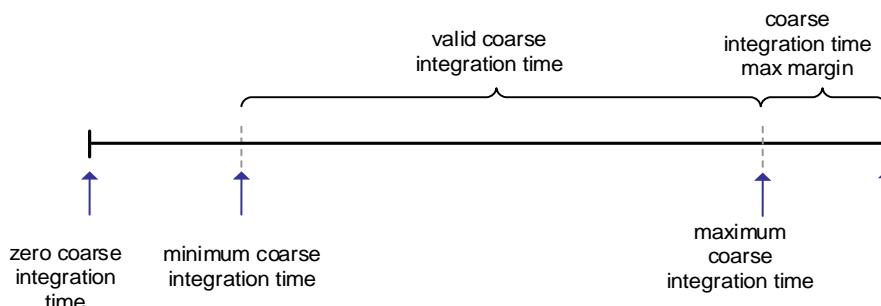
$$\text{coarse\_integration\_time\_max} = \text{current frame length lines} - \text{coarse\_integration\_time\_max\_margin}$$

$$\text{fine\_integration\_time\_max} = \text{current line length pck} - \text{fine\_integration\_time\_max\_margin}$$

CCS defines the maximum time limits in this way in order to allow the image-sensor-dependent parameters to be static, so that the Host can read them just one time. If the image sensor instead continually updated the current maximum legal values, then the Host would have to re-read them each time the line or frame length changed.



**Figure 49 Fine Integration Time Range**



**Figure 50 Coarse Integration Time Range**

### 9.6.2 Operation Outside Limits

The behavior of a CCS image sensor operating in Baseline Exposure Mode (i.e., when optional CCS or manufacturer-specific exposure mode is not taken into use by a specific control register) is not defined when control parameters are set to values outside of the recommended limits. As a result, a CCS-compatible Host should not operate the image sensor with control parameters set to values outside the recommended limits while operating in Baseline Exposure Mode.

Additional behaviors are defined for the Advanced Timing Modes, see [Section 9.7](#).

## 9.7 Advanced Timing Modes

In addition to Baseline Exposure Mode, the image sensor may also support any or all of the following three optional Advanced Timing Modes (*Table 120*):

- Automatic Frame Length Mode
- Manual Readout in Rolling Shutter Mode
- Delayed Exposure Start Mode

**Table 120 Timing Mode Capability Registers**

Register Name	Type	RW	Comment
<b>timing_mode_capability</b>	8-bit unsigned integer	RO	<p>Supported Timing Modes: Set bit to 1 to indicate support for that Mode</p> <p><b>Bit 0</b> 1: Automatic Frame Length Mode</p> <p><b>Bit 1</b> Reserved</p> <p><b>Bit 2</b> 1: Manual Readout in Rolling Shutter Mode</p> <p><b>Bit 3</b> 1: Delayed Exposure Start</p> <p><b>Bit 4</b> 1: For Manual Readout Mode: Store Exposure time in embedded data</p>

The **timing\_mode\_ctrl** register (*Table 121*) is used to select Manual Readout Mode and Delayed Exposure Mode, which are described in the following Sub-Sections.

**Table 121 Timing Mode Control Registers**

Register Name	Type	RW	Comment
<b>timing_mode_ctrl</b>	8-bit unsigned integer	RW	<p><b>Bit 0</b> 1: Manual readout mode</p> <p><b>Bit 1</b> 1: Delayed Exposure mode Default: 0x00</p>

### 9.7.1 Automatic Frame Length Mode

If Bit 0 of register **timing\_mode\_capability** is set to 1, then the image sensor shall support Automatic Frame Length Mode (*Table 122*). This Mode provides variable frame rate support in bracketing and in normal streaming, without the Host needing to manually control the frame length.

When Bit 0 of register **frame\_length\_ctrl** is set to 1, then the image sensor shall enable automatic frame length, i.e., shall automatically increase the value of **frame\_length\_lines** if the Exposure is longer than expected. Additional margin can be added with the register **frame\_margin** as detailed in the following pseudocode.

```

2724 If (auto_frame_length = 1)
2725   If (coarse_integration_time > (frame_length_lines - coarse_integration_time_max_margin))
2726     frame_length_lines =
2727       coarse_integration_time + coarse_integration_time_max_margin + frame_margin;
2728   Else
2729     frame_length_lines = frame_length_lines;
2730 Else // (auto_frame_length = 0)
2731   If (coarse_integration_time > (frame_length_lines - coarse_integration_time_max_margin))
2732     Don't care, not defined. In Baseline Exposure Mode, Host shall not use settings entering
2733     here.
2734     See Section 9.6.2
2735   Else
2736     Normal Baseline Exposure Mode. Host controls frame length and coarse Integration Time
2737     directly within sensor limits.

```

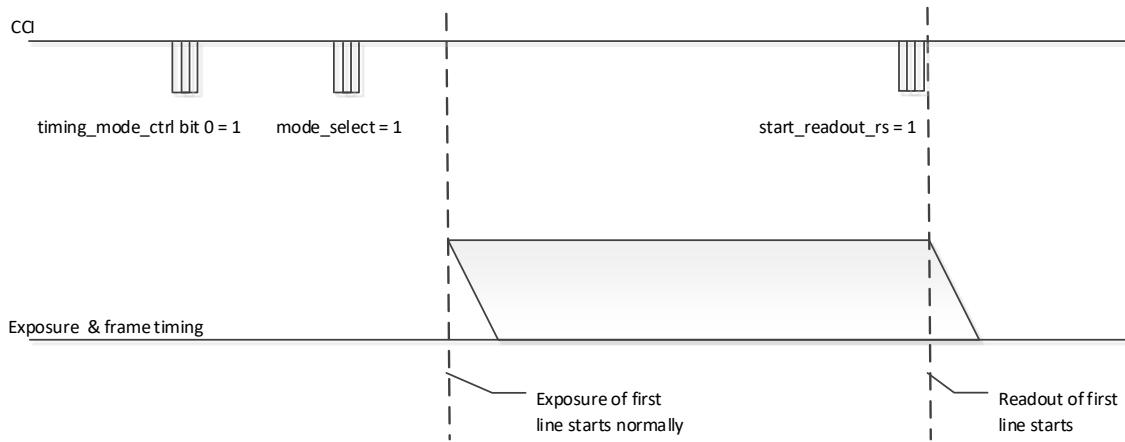
The additional margin, **frame\_margin**, can have minimum and maximum values. If the maximum value is 0, then the **frame\_margin** register is considered to not be supported. The separate minimum value register **frame\_margin\_min\_value** is provided for greater flexibility in image sensor design.

**Table 122 Automatic Frame Length Control Registers**

Register Name	Type	RW	Comment
<b>frame_length_ctrl</b>	8-bit unsigned integer	RW	<b>Bit 0</b> 1: Auto Frame Length in use 0: Auto Frame Length not in use <b>Default:</b> Image-sensor-specific
<b>frame_margin_min_value</b>	8-bit unsigned integer	RO	–
<b>frame_margin_max_value</b>	16-bit unsigned integer	RO	–
<b>frame_margin</b>	16-bit unsigned integer	RW	Number of extra VT lines to add <b>Default:</b> 0x00

### 9.7.2 Manual Readout Start Mode

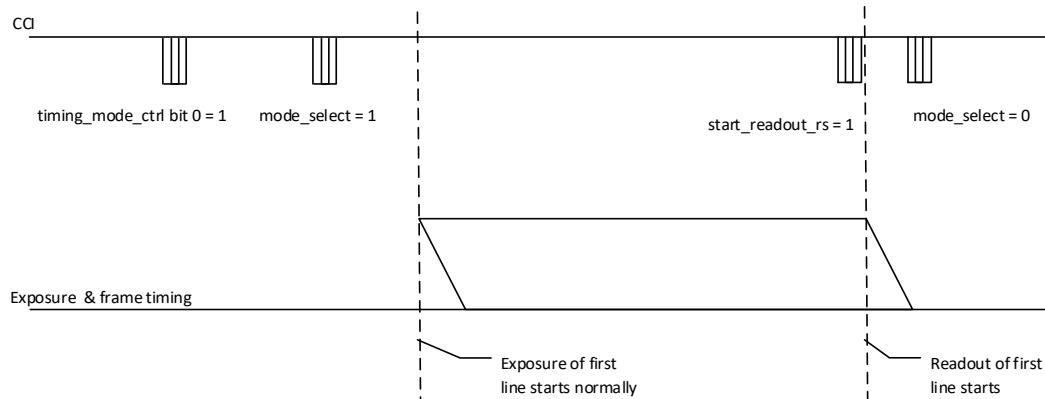
If Bit 2 of register **timing\_mode\_capability** is set to 1, then the image sensor supports Manual Readout Start Mode. This Mode allows for long Exposure times, and is illustrated in *Figure 51*.



**Figure 51 Manual Readout Start Mode**

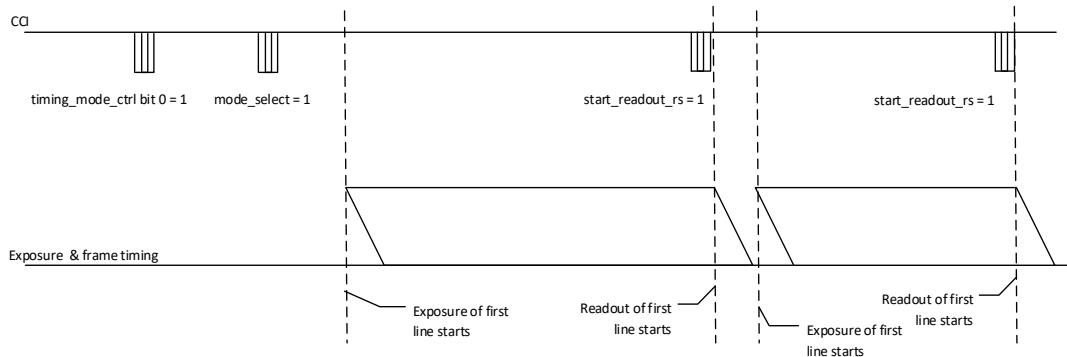
Manual Readout Start Mode is enabled by setting Bit 0 of register **timing\_mode\_ctrl** to 1. The Host shall only enable this Mode while in SW Standby Mode, and then enter Streaming Mode to start the Exposure. The image sensor shall continue the Exposure until the Host initiates read-out by setting Bit 0 of register **start\_readout\_rs** (*Table 123*) to 1. After image is read out, the image sensor shall automatically clear register **start\_readout\_rs**.

As *Figure 52* illustrates, the Host can capture a single frame with a long Exposure time by putting the image sensor into SW Standby Mode (by setting register **mode\_select** to 0) after sending the **start\_readout\_rs** command.



**Figure 52 Single Long Exposure Frame**

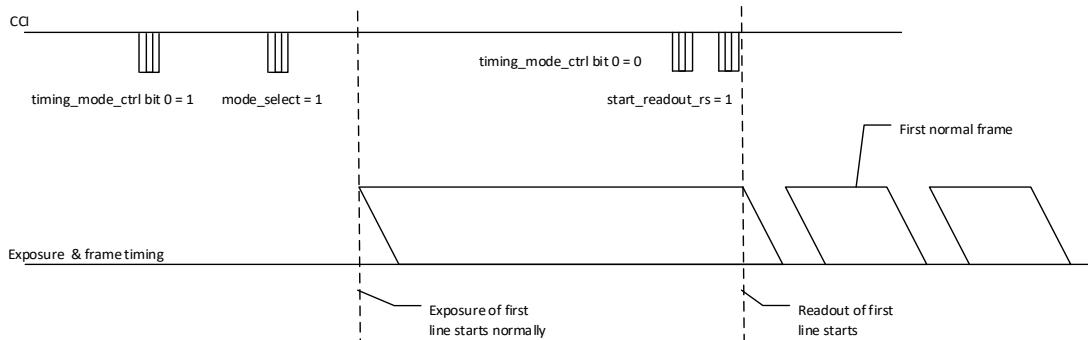
If the Host does not disable manual readout after sending the **start\_readout\_rs** command, and also does not put the image sensor into SW Standby mode, then the next frame will also operate in Manual Readout Mode as shown in **Figure 53**. In this situation, the Host shall initiate manual readout (by setting Bit 0 of register **start\_readout\_rs** to 1) when needed for each frame. This allows the Host to capture multiple long Exposure frames. The image sensor shall start Exposure of next frame after the first line of the previous frame has been read out.



2760

**Figure 53 Multiple Long Exposure Frames**

If the Host disables Manual Readout Mode (by setting Bit 0 of register **timing\_mode\_ctrl** to 0) before sending the **start\_readout\_rs** command, then the image sensor shall continue normal streaming after the long Exposure frame as shown in Figure 54, with Exposure controlled normally (i.e., by register **coarse\_integration\_time**).



2765

**Figure 54 Single Long Exposure Frame Followed by Normal Frames**

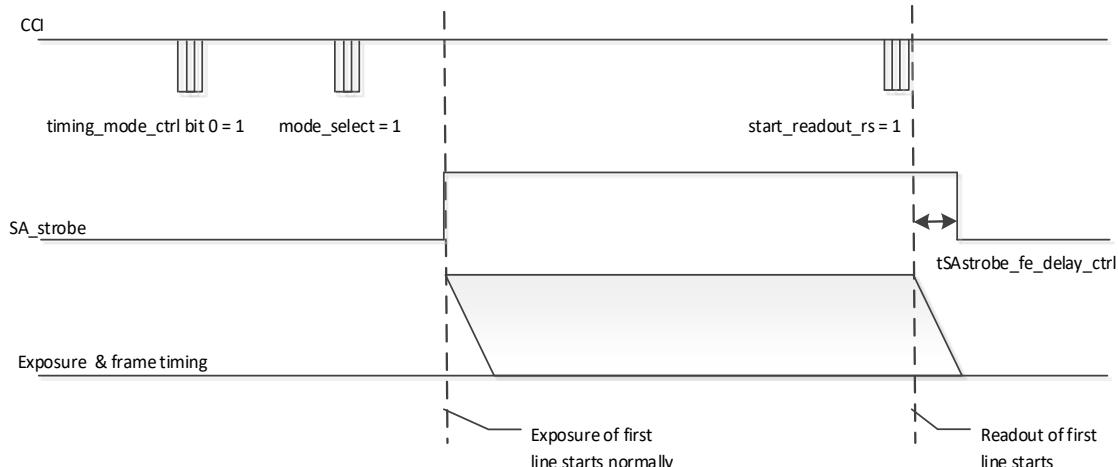
The image sensor should include the Exposure time used for the frame (register **coarse\_integration\_time**) in the top embedded data. If the Exposure time is longer than the maximum value for the **coarse\_integration\_time** register, then the **coarse\_integration\_time** value used in the embedded data shall be capped ( $2^{16}-1$ , or 65,535). The Host can detect whether the image sensor supports storing Exposure information in the top embedded data by inspecting Bit 4 of register **timing\_mode\_capability**.

2771

**Table 123 Manual Readout Control Registers**

Register Name	Type	RW	Comment
<b>start_readout_rs</b>	8-bit unsigned integer	RW	<b>Bit 0</b> 1: Manual readout start (auto clear) <b>Default:</b> 0x00

If SA\_strobe is used in Edge Mode during Manual Readout Mode, then the behavior is different from normal Exposure mode (see **Figure 55**). SA\_strobe can be enabled by multiple means (e.g. with asynchronous or synchronous mode), see **Section 16.3**. When the image sensor receives a **start\_readout\_rs** command, the the image sensor shall delay the falling edge of SA\_strobe by the value of register **tSA\_strobe\_fe\_delay\_ctrl**. That is, the rising edge of SA\_strobe is controlled normally but the **tSA\_strobe\_fe\_delay\_ctrl** reference point changes to the **start\_readout\_rs** command.



2778

**Figure 55 SA\_strobe During Manual Readout Start Mode**

### 9.7.3 Delayed Exposure Start Mode

If Bit 3 of register **timing\_mode\_capability** is 1, then the image sensor supports Delayed Exposure Start Mode, which supports accurate timing of the moment of Exposure. In this Mode, the start time of the first frame after SW Standby Mode is based on the external trigger signal **GPIO\_TRIGGER**.

The Host shall enable Delayed Exposure Start Mode only while in SW Standby Mode, by setting Bit 1 of register **timing\_mode\_ctrl** to 1. The image sensor shall start listening to the **GPIO\_TRIGGER** pin after the Host has put it into Streaming Mode (register **mode\_select** = 1).

The image sensor should start the Exposure as soon as possible after detecting a rising edge on the **GPIO\_TRIGGER** pin. The image sensor datasheet shall describe this delay, and possible formula if the delay changes (for example, varying as a function of the image sensor clocking frequency).

The image sensor datasheet shall make visible any deviation from the following requirements and recommendations:

- The image sensor vendor shall do its best to ensure that the delay is fixed
- The delay shall not vary by more than 500ns
- The delay should be smaller than 1ms
- The delay may be different for the first power-up

If the image sensor has some minimum time delay between the **mode\_select** command and the rising edge of the **GPIO\_TRIGGER** signal, then the image sensor datasheet shall make that minimum time visible.

The Host shall only change Bit 1 of register **timing\_mode\_ctrl** while in SW Standby Mode.

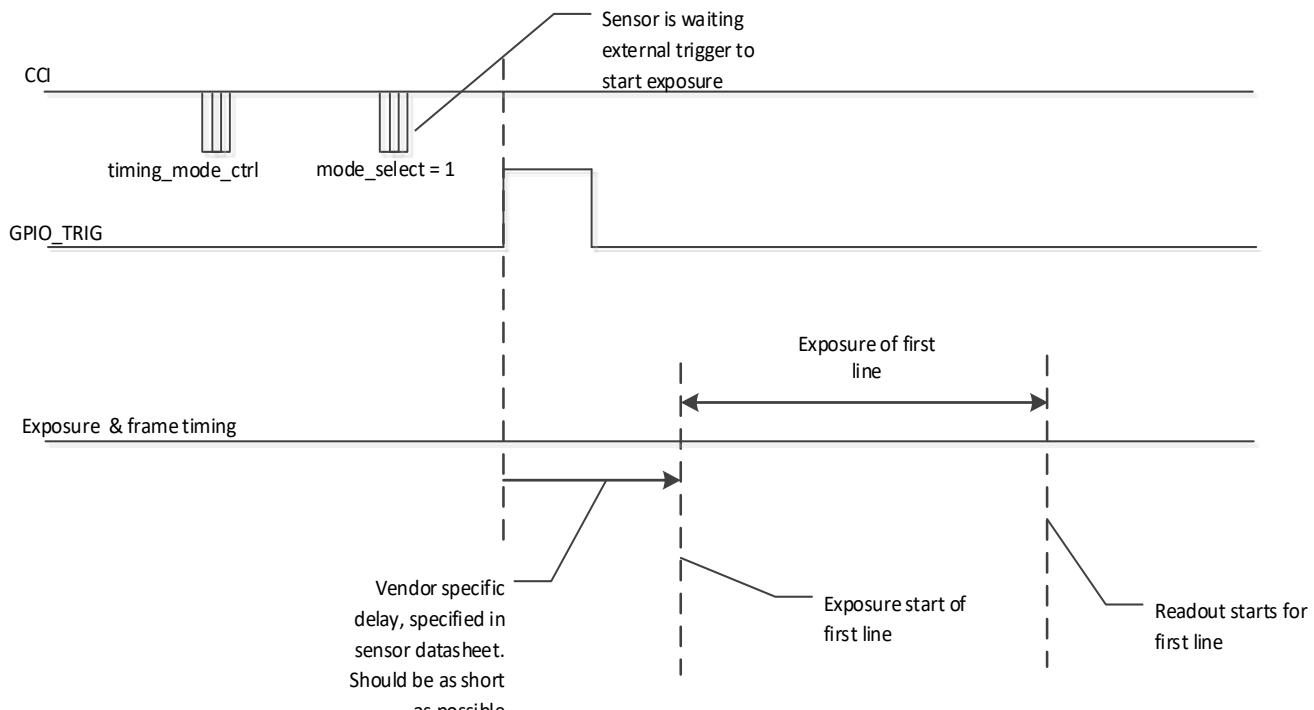


Figure 56 **GPIO\_TRIGGER** Used to Start Exposure

## 9.8 Auto-Bracketing Function

Where indicated as supported, the image sensor shall have the facility to output multiple consecutive frames using settings derived from a look up table (LUT). This is intended for use in auto-bracketing functions, i.e., rapidly taking multiple images at a number of different settings. The advantages compared to separately writing all of the different settings are smaller size and a more determinate latency.

The bracketing LUT occupies a range in the image sensor's standard CCS Register map, and will contain the settings for several frames. To use the function, the Host shall program the LUT values (e.g. coarse Integration Time, analog Gain) for the required number of frames only while in SW Standby Mode. Then, by changing to Streaming mode with the appropriate control register set, the Host commands the image sensor to output the same number of frames using the programmed settings.

The image sensor shall change its internal parameters (including updating the embedded data) for each consecutive frame, based on the LUT content. This is necessary for proper functioning of the auto-bracketing operation.

The bracketing LUT may support different content, depending on the image sensor's specific capabilities. The Host can discover the LUT capabilities, and calculate the available LUT size, by reading the registers listed in *Table 124*.

**Table 124 Bracketing LUT Capability Registers**

Register Name	Type	RW	Comment
<b>bracketing_LUT_capability_1</b>	8-bit unsigned integer	RO	Supported Capabilities 1 <b>Bit 0</b> Coarse Integration Time LUT entry <b>Bit 1</b> Global analog Gain LUT entry <b>Bits 2 &amp; 3</b> Reserved <b>Bit 4</b> Flash LUT entry <b>Bit 5</b> Global digital Gain LUT entry <b>Bit 6</b> Alternate Global analog Gain LUT entry <b>Bit 7</b> Reserved for future use <b>All bits</b> 0: Not supported 1: Supported
<b>bracketing_LUT_capability_2</b>	8-bit unsigned integer	RO	Supported Capabilities 2 <b>Bit 0</b> Single Bracketing Mode <b>Bit 1</b> Looped Bracketing Mode <b>Bits 2–7</b> Reserved for future use <b>All bits</b> 0: Not supported 1: Supported
<b>bracketing_LUT_size</b>	8-bit unsigned integer	RO	Maximum number of LUT frames <b>Example:</b> 5 = LUT can contain settings for up to 5 frames

2814 The content of the bracketing LUT depends on the values of registers **bracketing\_LUT\_size** (the maximum  
 2815 number of frames for which there can be settings) and **bracketing\_LUT\_capability\_1**, which determines the  
 2816 number of entries per frame. Register **bracketing\_LUT\_capability\_2** indicates which LUT modes the image  
 2817 sensor supports.

### 2818 Example 1

2819 The following register settings define a 3-frame LUT, with entries for global analog Gain and coarse  
 2820 Integration Time. The resulting LUT format is shown in *Table 125*.

- 2821 • **bracketing\_LUT\_size** = 0x03: Three frames in the LUT
- 2822 • **bracketing\_LUT\_capability\_1** = 0x03: Each frame includes entries for Coarse Integration Time  
 2823 (0x01) and Global analog Gain (0x02)
- 2824 • **bracketing\_LUT\_capability\_2** = 0x01: Use Single Bracketing Mode

2825 **Table 125 Example 1 LUT Content**

Name of LUT field	LUT field purpose/value	RW	LUT field format (depends on LUT entry format)
bracketing_LUT_frame_A, first entry	coarse_integration_time (A)	RW	16-bit unsigned integer
bracketing_LUT_frame_A, 2nd entry	analog_gain_code_global (A)	RW	16-bit unsigned integer
bracketing_LUT_frame_B, first entry	coarse_integration_time (B)	RW	16-bit unsigned integer
bracketing_LUT_frame_B, 2nd entry	analog_gain_code_global (B)	RW	16-bit unsigned integer
bracketing_LUT_frame_C, first entry	coarse_integration_time (C)	RW	16-bit unsigned integer
bracketing_LUT_frame_C, 2nd entry	analog_gain_code_global (C)	RW	16-bit unsigned integer

### 2826 Example 2

2827 The following register settings define a 2-frame LUT, with one entry for global analog Gain. The resulting  
 2828 LUT format is shown in *Table 126*.

- 2829 • **bracketing\_LUT\_size** = 0x02: Two frames in the LUT
- 2830 • **bracketing\_LUT\_capability\_1** = 0x02: Each frame includes entry for Global analog Gain entry  
 2831 (0x02)
- 2832 • **bracketing\_LUT\_capability\_2** = 0x01: Use Single Bracketing Mode

2833 This defines a 2-frame LUT in which entries for only global analog Gain are supported.

2834

**Table 126 Example 2 LUT Content**

Name of LUT field	LUT field purpose/value	RW	LUT field format
<b>bracketing_LUT_frame_A</b> , first entry	<b>analog_gain_code_global</b> (A)	RW	16-bit unsigned integer
<b>bracketing_LUT_frame_B</b> , first entry	<b>analog_gain_code_global</b> (B)	RW	16-bit unsigned integer

2835 The LUT content is static. If the Host wants to bracket fewer parameters than the LUT supports (e.g. only  
 2836 analog Gain, but not coarse Integration Time), then the Host shall ensure that each **coarse\_integration\_time**  
 2837 entry in the LUT is programmed with the same value.

2838 All possible LUT parameter entries are shown in **Table 127**. Each LUT parameter entry shall use the same  
 2839 data format (e.g. 16-bit unsigned integer) that that parameter uses in the standard CCS Register map (**Section**  
 2840 **20**), with the following exception.

2841 The flash entry has no exact counterpart in the Register map. If Bit 4 of the **bracketing\_LUT\_entry** for any  
 2842 frame is set to 1, then the flash strobe shall be re-timed to this frame. This has the same meaning as sending  
 2843 the **flash\_trigger\_rs** command while in normal operation, so that re-timing happens to the same frame.

2844

**Table 127 Possible LUT Entry Types**

Register Name	Type	RW	Comment
<b>coarse_integration_time</b>	16-bit unsigned integer	RW	Coarse Integration Time (lines) <b>Default:</b> Image-sensor-specific
<b>analog_gain_code_global</b>	16-bit unsigned integer	RW	Global analog Gain code <b>Default:</b> Image-sensor-specific
<b>digital_gain_global</b>	16-bit unsigned iReal	RW	Global digital Gain value <b>Default:</b> Image-sensor-specific
<b>bracketing_LUT_entry</b>	16-bit unsigned integer	RW	<b>Bit 4</b> 1: Flash strobe is retimed to this frame <b>Bits 0-3 &amp; 5-15</b> Reserved for future use <b>Default:</b> Bit 4 image-sensor-specific, other bits 0.
<b>analog_linear_gain_global</b>	16-bit unsigned iReal	RW	Alternate Global analog gain control register <b>Default:</b> Image-sensor-specific
<b>analog_exponential_gain_global</b>	16-bit signed iReal	RW	Alternate Global analog gain control register <b>Default:</b> Image-sensor-specific

2845 Bracketing is controlled by the configuration registers in *Table 128*.

2846 **Table 128 Bracketing Configuration Registers**

Register Name	Type	RW	Comment
<b>bracketing_LUT_ctrl</b>	8-bit unsigned integer	RW	<p>0: Disabled 1...N: Bracket over N frames <b>Maximum Value:</b> bracketing_LUT_size If N &lt; bracketing_LUT_size, then only N values from the LUT are used E.g. With bracketing_LUT_size of 5, set this register to 3 to bracket over just 3 frames <b>Default:</b> 0x00</p>
<b>bracketing_LUT_mode</b>	8-bit unsigned integer	RW	<p><b>Bit 0</b> 0: Return to SW Standby Mode after bracketing 1: Continue in Streaming Mode after bracketing <b>Bit 1</b> 0: Single Bracketing Mode 1: Loop Bracketing Mode <b>Bit 2–7</b> Reserved for future use <b>Default:</b> 0x00</p>
<b>bracketing_LUT_entry_ctrl</b>	8-bit unsigned integer	RW	Reserved for future use

**Operation Rules**

- The Host shall update the LUT only while in SW Standby Mode.
- The image sensor shall start bracketing after SW Standby Mode only if bracketing has been enabled and the image sensor is then put into Streaming Mode (via register **mode\_select**).
- During bracketing, the image sensor shall automatically load settings from the LUT to the image sensor's internal registers (though not to the corresponding CCI control registers). These values shall also be included in embedded data for all frames required by the Host.

**Example:** For a bracketing LUT that changes the **coarse\_integration\_time** value from frame to frame, the per-frame value is also written into the embedded data lines as though they were the contents of the **coarse\_integration\_time** registers for that frame. Note that the bracketed value changes must not be written to the corresponding actual CCI **coarse\_integration\_time** registers.

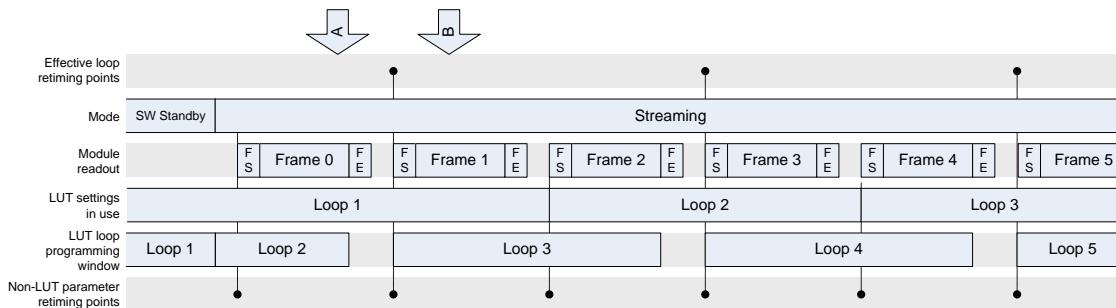
- All registers listed in *Table 128* shall be included in the embedded data. The LUT itself need not appear.
- The **bracketing\_LUT\_mode** register (*Table 128*) gives two options for the operation after the bracketing sequence end: the image sensor either automatically returns to SW Standby Mode, or it continues to stream with the original settings that were in use before the bracketing sequence started.
- The Host can reprogram or disable bracketing LUT and bracketing LUT control while in SW Standby Mode.
- The Host can end the bracketing operation before the image sensor completes the bracketing sequence by putting the image sensor into SW Standby Mode (via register **mode\_select**). In executing this mode change, the image sensor might terminate the frame, depending on the value of register **fast\_standby\_ctrl**.
- Where settings with different latencies (e.g. Gain and Integration Time) are changed together in the same LUT frame, the image sensor shall output no intermediate frames. That is, frame A shall always be consecutive with frame B, etc.

## 2873 Specific Operation Rules for Loop Bracketing Mode

- 2874 • Loop Bracketing Mode is entered by setting Bit 1 of register **bracketing\_LUT\_mode** to 1. In Loop  
2875 Mode, the image sensor shall continuously repeat the bracketing sequence, always applying the  
2876 LUT parameter changes. For example, with a LUT size of two, the image sensor would output  
2877 images with frame\_A settings, frame\_B settings, frame\_A settings, frame\_B settings, etc.
- 2878 • During Loop Bracketing Mode, the Host may update the bracketing LUT content during  
2879 streaming. This applies only to Loop Bracketing Mode, not Single Bracketing Mode.
- 2880 • If the LUT is updated in Loop Bracketing Mode, then the image sensor shall re-time the changes  
2881 to take effect only at the next start of the bracketing sequence (i.e., when the bracketing loop starts  
2882 again from the first LUT entry).
- 2883 • The Host can terminate Loop Bracketing Mode in the following ways:
  - 2884 • By setting register **bracketing\_LUT\_ctrl** to 0. In this case:
    - 2885 • If register **bracketing\_LUT\_mode** = 0, then the image sensor shall return to SW Standby  
2886 Mode after completing the current LUT loop.
    - 2887 • If register **bracketing\_LUT\_mode** = 1, then the image sensor shall continue to stream with  
2888 original values after completing the current LUT loop.
  - 2889 • By register setting **mode\_select** to 0. In this case:
    - 2890 • If register **fast\_standby\_ctrl** is set to 1, then the image sensor shall terminate the current  
2891 frame immediately and return to SW Standby Mode as described in *Section 5.2.4*.
    - 2892 • If register **fast\_standby\_ctrl** is set to 0, then the image sensor shall complete the current  
2893 frame and then return to SW Standby Mode.
  - 2894 • Other than the above use to terminate a loop sequence, registers **bracketing\_LUT\_ctrl** and  
2895 **bracketing\_LUT\_mode** shall only be changed while in SW Standby Mode.

2896 **Figure 57** illustrates loop mode LUT update behavior for a two-frame LUT example (**bracketing\_LUT\_ctrl**  
2897 = 2). The programming window for the next loop ends at the start of the penultimate frame of the current  
2898 loop, in order to accommodate the parameters that have two-frame latency.

2899 Since all LUT updates are made using **grouped\_parameter\_hold**, updates released by  
2900 **grouped\_parameter\_hold** at point A will become effective in loop 2. By contrast, updates released at point  
2901 B will only be effective in loop 3. The Host should avoid programming the LUT during the frame end code  
2902 and the frame blanking between adjacent loop programming windows, because in some image sensors the  
2903 exact retime point may be slightly offset from the frame start code.



2904 **Figure 57 Loop Mode Update Behavior**

2905 Where a particular **grouped\_parameter\_hold** also contains CCS parameters other than the LUT entries, the  
2906 image sensor shall re-time those parameters normally. For example, if non-LUT parameters were in the  
2907 **grouped\_parameter\_hold** released at point B, then they would all become effective either from frame 2 (for  
2908 any one-frame latency parameters), or from frame 3 (for any two-frame latency parameters).

## 9.9 HDR Timing Mode and HDR Synthesis

An image sensor may support an on-module single frame HDR function. There are two flavors of HDR function:

- **HDR Timing Mode:** The image sensor outputs image data with differently exposed areas, namely short Exposure and long Exposure areas. The Host can determine whether the image sensor supports HDR Timing Mode from Bit 3 of register **HDR\_capability\_2**.
- **HDR Synthesis Mode:** The image sensor internally uses differently exposed image areas to produce high-bit-depth linear output. The Host can determine whether the image sensor supports HDR Synthesis Mode from Bit 4 of register **HDR\_capability\_2**.

The image sensor HDR function shall operate in HDR Synthesis Mode if all of the following are true:

- HDR Synthesis Mode is selected via Bit 4 of register **HDR\_mode**
- HDR is enabled via Bit 0 of register **HDR\_mode**
- The internal HDR operating bit depth is set properly via register **HDR\_internal\_bit\_depth**. The Host shall set the bit depth to a value between the minimum value (register **min\_HDR\_bit\_depth**) and the maximum value (register **max\_HDR\_bit\_depth**).
- Register **CSI\_data\_format** is set properly (see below)

Register **CSI\_data\_format** controls which bits per pixel are sent to the CSI-2 bus, and whether or not compression is used. The Host shall set **CSI\_data\_format** as desired.

### Examples:

- Assuming that HDR processing uses 14 bits (i.e., that register **HDR\_internal\_bit\_depth** has value 0x0E), and that register **CSI\_data\_format** has value 0x0C0C (12 bits), then the top 12 bits of the 14 bits data produced by the HDR function will be sent to the CSI-2 bus.
- If register **CSI\_data\_format** instead had the value 0x0C08, then the top 12 bits of the 14-bit data produced by the HDR function would be first compressed to 8 bits (via 12b-to-8b compression) and then sent to the CSI-2.
- If the HDR processing uses 13 bits (i.e., **HDR\_internal\_bit\_depth** has value 0x0D) and register **CSI\_data\_format** has value 0x0E0E (14 bits), then the 13-bit data produced by the HDR function (i.e., the real data) will be used as the LS bits of the data and the MS bit of the data is set to 0. This 14-bit data is sent to the CSI-2 bus.

The Host shall ensure that register **HDR\_internal\_bit\_depth** has sufficient range to allow the HDR effect to be applied based on registers **exposure\_ratio** or **short\_coarse\_integration\_time**, per *Table 129*.

**Table 129** **HDR\_internal\_bit\_depth** Value as a Function of Exposure Ratio

Exposure ratio	Needed bit depth	Comment
max 1:16	14 bits	Assuming 10-bit ADC.
max 1:8	13 bits	Exposure ratio is ratio between short and long Exposure times controlled either by ratio or direct controls.
max 1:4	12 bits	
1:1	10 bits	

The image sensor HDR function shall operate in HDR Timing Mode if all of the following are true:

- HDR Timing Mode is selected via Bit 4 of register **HDR\_mode**
- HDR is enabled via Bit 0 of **HDR\_mode** register.
- Register **CSI\_data\_format** is set normally

**Example:** If the image sensor supports a 10 bit ADC, then Bits 15:8 of register **CSI\_data\_format** will need to be set to 10 (0x0A) for all 10 bits data bits to be transferred.

2946 **Note:**

2947     *The Host need not set register **HDR\_internal\_bit\_depth** for HDR Timing Mode.*

2948 The image sensor may place limitations on the output resolution of the HDR Synthesis function. The Host  
2949 can read such limitations from registers **HDR\_resolution\_sub\_types** and **HDR\_resolution\_type\_n** (*Table  
2950 130*).

- 2951     • **HDR\_resolution\_sub\_types** indicates how many **HDR\_resolution\_type\_n** registers exist  
2952     • Each **HDR\_resolution\_type\_n** register defines an allowed HDR resolution

2953 If the maximum image size defined by register **HDR\_resolution\_type\_n** is smaller than 1x1 (i.e., if all  
2954 **HDR\_resolution\_type\_n** registers are not 1x1), then reduced resolution must be used in order to apply HDR  
2955 processing. If reduced resolution is not used, then the output frame may have different Exposure time areas  
2956 visible.

2957 Image sensor HDR functions may have different capabilities. The Host can determine the image sensor's  
2958 HDR capabilities from Registers **HDR\_capability\_1** and **HDR\_capability\_2** (*Table 130*).

2959 Bit 0 of register **HDR\_capability\_1** indicates whether the image sensor HDR function can operate with  
2960 binning:

- 2961     • The HDR function should operate without binning.  
2962     • Additionally, 2x2 binning may be applied either before the HDR functionality, or during HDR if  
2963       so indicated via Bit 0 of register **HDR\_capability\_1**. If additional restrictions exist, they shall be  
2964       visible in the image sensor datasheet.

2965 Bits 1 and 2 of register **HDR\_capability\_1** indicate whether the image sensor HDR function supports  
2966 Combined Analog Gain Mode, Separate Analog Gain Mode, or both:

- 2967     • Combined Analog Gain Mode is required. In this Mode, both the short Exposure areas and the  
2968       long Exposure areas shall use the normal analog Gain control (*Section 9.3*).  
2969     • Separate Analog Gain Mode is optional. In this Mode, Gain for the short Exposure areas is set  
2970       with register **short\_analog\_gain\_global** or **short\_analog\_linear\_gain\_global** and  
2971       **short\_analog\_exponential\_gain\_global**, and the long Exposure areas use the normal analog  
2972       Gain control. Difference between Global analog Gain and Alternate Global analog gain is  
2973       described in *Section 9.3*.  
2974     • The limit register values for the analog Gain shall be the same for short and long Exposure areas.

2975 Bits 0 and 1 of register **HDR\_capability\_2** indicate whether the image sensor HDR function supports  
2976 Combined Digital Gain, Separate Digital Gain, or both:

- 2977     • Combined Digital Gain Mode is optional, but if digital Gain is supported for non-HDR image, it  
2978       shall also be supported for the HDR function. In this Mode, both the short Exposure areas and the  
2979       long Exposure areas use the normal digital Gain control (*Section 9.4*).  
2980     • Separate Digital Gain Mode is optional. In this Mode, Gain for the short Exposure areas is set with  
2981       register **short\_digital\_gain\_global**, and the long Exposure areas use the normal digital Gain  
2982       control.  
2983     • The limit register values for the digital Gain shall be the same for short and long Exposure areas.

2984 Bit 3 of register **HDR\_capability\_1** indicates whether the image sensor HDR function can maintain resolution  
2985 during HDR synthesis processing requiring resolution reduction.

- 2986 • One possible use case is that at least one of the **HDR\_resolution\_type\_n** registers is not 1x1 (e.g.  
2987 the list of values is { 1x1, 1x2 }, so reduced resolution is required for 1x2 mode). If register  
2988 **HDR\_resolution\_reduction** is set to the value requiring the reduction (1x2 in the example), then  
2989 that resolution reduction will be used.
- 2990 • If upscaling functionality is also used, then the image sensor can maintain the same resolution  
2991 (i.e., the resolution of the input to the HDR function is the same as the resolution of the output  
2992 resolution of HDR function) during HDR synthesis. But if upscaling is not used, then the output  
2993 resolution will be reduced.
- 2994 • The image sensor HDR synthesis function should maintain resolution either by not requiring  
2995 resolution reduction, or by supporting upscaling.

2996 Bit 4 of register **HDR\_capability\_1** indicates whether the image sensor HDR function supports only Readout  
2997 Sync Mode, or both Readout Sync Mode and Reset Sync Mode.

- 2998 • **Readout Sync Mode** is required. In this Mode, the Exposures of the short Exposure area and of  
2999 the long Exposure areas are started at different times and a single image sensor frame is generated.  
3000 However the read-outs of both areas occur simultaneously (but within the limits set by rolling  
3001 shutter).
- 3002 • **Reset Sync Mode** is optional and intended for use with HDR Timing Mode (i.e., the Mode is not  
3003 intended for use with HDR Synthesis Mode, and might not operate with it). In this Mode, the  
3004 Exposures of the short Exposure area and of the long Exposure areas are started at the same time.  
3005 During readout, the image sensor outputs first a frame with the short Exposure time, and then a  
3006 frame with the long Exposure time. The CCS frame counter and the CSI-2 frame counter shall  
3007 calculate all of these frames.

3008 Bits 5 and 6 of register **HDR\_capability\_1** indicate whether the image sensor HDR function supports Direct  
3009 control for short Integration Time.

- 3010 • Ratio control is required. In this Mode, short Integration Time is controlled by the register  
3011 **exposure\_ratio**.
- 3012 • Direct control is optional. If Direct control is supported, then short Integration Time can be  
3013 controlled via register **short\_coarse\_integration\_time**.
- 3014 • In both cases, long Exposure time is controlled by the standard Exposure control (register  
3015 **coarse\_integration\_time**).

3016 Group parameter hold functionality can be used when changing registers **exposure\_ratio**,  
3017 **short\_coarse\_integration\_time**, **short\_digital\_gain\_global**, **short\_analog\_gain\_global**,  
3018 **short\_analog\_linear\_gain\_global** and **short\_analog\_exponential\_gain\_global**.

3019

**Table 130 HDR Capability Registers**

Register Name	Type	RW	Comment
<b>min_HDR_bit_depth</b>	8-bit unsigned integer	RO	The minimum bit depth for HDR internal processing during HDR Synthesis Mode
<b>max_HDR_bit_depth</b>	8-bit unsigned integer	RO	The maximum bit depth for HDR internal processing during HDR Synthesis Mode
<b>HDR_resolution_sub_types</b>	8-bit unsigned integer	RO	N: Number of HDR resolution subtypes available
<b>HDR_resolution_type_1</b>	8-bit unsigned integer	RO	<b>Example:</b> 0x12 = Column x 1, Row x 2
<b>HDR_resolution_type_2</b>	8-bit unsigned integer	RO	<b>Example:</b> 0x22 = Column x 2, Row x 2
<b>HDR_capability_1</b>	8-bit unsigned integer	RO	<p>Supported Capabilities 1</p> <p><b>Bit 0:</b> 2x2 Binning with HDR</p> <p><b>Bit 1:</b> Combined Analog Gain Mode</p> <p><b>Bit 2:</b> Separate Analog Gain Mode</p> <p><b>Bit 3:</b> HDR Upscaling</p> <p><b>Bit 4:</b> Reset Sync Mode</p> <p><b>Bit 5:</b> Direct Short Exposure Time Control for HDR Timing</p> <p><b>Bit 6:</b> Direct Short Exposure Time Control for HDR Synthesis Mode</p> <p><b>Bit 7:</b> Reserved</p> <p><b>All bits</b></p> <p>0: Not supported 1: Supported</p>
<b>HDR_capability_2</b>	8-bit unsigned integer	RO	<p>Supported Capabilities 2</p> <p><b>Bit 0:</b> Combined Digital Gain</p> <p><b>Bit 1:</b> Separate Digital Gain</p> <p><b>Bit 2, 5-7:</b> Reserved</p> <p><b>Bit 3:</b> HDR Timing Mode</p> <p><b>Bit 4:</b> HDR Synthesis Mode</p> <p><b>All bits</b></p> <p>0: Not supported 1: Supported</p>

**Table 131 HDR Control Registers**

Register Name	Type	RW	Comment
<b>HDR_mode</b>	8-bit unsigned integer	RW	<p><b>Bit 0</b> 0: HDR function off 1: HDR function on</p> <p><b>Bit 1</b> 0: Combined Analog Gain used 1: Separate Analog Gain used</p> <p><b>Bit 2</b> 0: HDR Upscaling off 1: HDR Upscaling on</p> <p><b>Bit 3</b> 0: Read Sync Mode 1: Reset Sync Mode</p> <p><b>Bit 4</b> 0: HDR Synthesis Mode 1: HDR Timing Mode</p> <p><b>Bit 5</b> 0: Short Exposure determined by Ratio 1: Short Exposure controlled by Direct Control</p> <p><b>Bit 6</b> 0: Combined Digital Gain used 1: Separate Digital Gain used</p> <p><b>Bit 7</b> Reserved</p> <p><b>Default</b> Bit 4 = Image-sensor-specific, other bits = 0.</p>
<b>exposure_ratio</b>	8-bit unsigned integer	RW	<p>Defines Exposure ratio between short and long Exposure. Short Exposure value = <b>coarse_integration_time / exposure_ratio</b></p> <p><b>Default:</b> Image-sensor-specific</p>
<b>HDR_resolution_reduction</b>	8-bit unsigned integer	RW	<p>High nibble specifies Column scaling factor Low nibble specifies Row scaling factor</p> <p><b>Example:</b> 0x12 = Column x 1, Row x 2</p> <p><b>Default:</b> Image-sensor-specific</p>
<b>short_analog_gain_global</b>	16-bit unsigned integer	RW	<p>Same format as in <b>analog_gain_code_global</b></p> <p><b>Default:</b> Image-sensor-specific</p>
<b>short_digital_gain_global</b>	16-bit unsigned iReal	RW	<p>Same format as in <b>digital_gain_global</b></p> <p><b>Default:</b> Image-sensor-specific</p>
<b>short_coarse_integration_time</b>	16-bit unsigned integer	RW	<p>Same format as in <b>coarse_integration_time</b></p> <p><b>Default:</b> Image-sensor-specific</p>

<b>HDR_internal_bit_depth</b>	8-bit unsigned integer	RW	Controls bit depth of HDR processing in case of HDR synthesis mode <b>Default:</b> Image-sensor-specific
<b>short_analog_linear_gain_global</b>	16-bit unsigned iReal	RW	Same format as in <b>analog_linear_gain_global</b> <b>Default:</b> Image-sensor-specific
<b>short_analog_exponential_gain_global</b>	16-bit signed iReal	RW	Same format as in <b>analog_exponential_gain_global</b> <b>Default:</b> Image-sensor-specific

## 9.9.1 Impact of HDR Function on Other CCS Features

### 9.9.1.1 Flash Strobe and SA Strobe

3021 Both Flash Strobe and SA Strobe should be triggered by a long Exposure area, and should not be triggered  
3022 by a short Exposure area.

### 9.9.1.2 Binning

3023 The image sensor may support binning with HDR. 2x2 binning during HDR is specified in **Section 9.9**. Other  
3024 binnings are outside the scope of this Specification. If the image sensor supports the use of additional binning  
3025 modes with HDR, this should be described in the image sensor datasheet.

### 9.9.1.3 Exposure Time

3026 The Minimum Exposure time for HDR is larger than in Baseline Exposure Mode, because the short Exposure  
3027 value minimum value cannot be violated. As a result, during actual operation the value of HDR function  
3028 register **coarse\_integration\_time** must be greater than the **coarse\_integration\_time\_min** value multiplied  
3029 by the **exposure\_ratio**. The Host shall ensure that the minimum value of short Exposure is not violated, by  
3030 limiting the long Exposure value (**coarse\_integration\_time**).

3031 **Example 1:** In normal mode, for an image sensor with a **coarse\_integration\_time\_min** of 1 line, the  
3032 minimum value that register **coarse\_integration\_time** can have is 1 line.

3033 **Example 2:** In HDR mode, the minimum value that register **coarse\_integration\_time** can have is given by  
3034 the formula **coarse\_integration\_time\_min \* exposure\_ratio**.

### 9.9.1.4 Other Use Cases

3035 A value of 255x1 in register **HDR\_resolution\_type\_n** indicates that the horizontal scaler may also be used  
3036 with the HDR function.

## 10 Test Modes

3037 This Section defines several required and recommended image sensor test patterns.

### 10.1 Full Frame Deterministic Test Patterns

3038 Two types of full-frame deterministic test patterns are defined. Most are artificial Bayer test patterns more  
 3039 suitable for some tests than real image data. One non-Bayer test pattern is PN9, which is intended to test  
 3040 integrity of the link between the image sensor and the Host.

3041 Use of these full-frame test patterns is controlled by register **test\_pattern\_mode** (*Table 132*).

3042 **Table 132 Main Full Frame Test Pattern Control Parameter**

Parameter Name	Type	RW	Comment
<b>test_pattern_mode</b>	16-bit unsigned integer	RW	Controls the output of the test pattern module 0: No pattern (default) 1: Solid color 2: 100% color bars 3: 'Fade to grey' color bars 4: PN9 5: 100% color tile 6–255: Reserved 256–65535: Manufacturer Specific

3043 **Table 133 Test Mode Capability**

Parameter Name	Type	RW	Comment
<b>test_mode_capability</b>	16-bit unsigned integer	RO	<b>Bit 0</b> 1: Solid color supported <b>Bit 1</b> 1: 100% color bars supported <b>Bit 2</b> 1: 'Fade to grey' color bars supported <b>Bit 3</b> 1: PN9 supported <b>Bit 4</b> Reserved <b>Bit 5</b> 1: 100% color tile supported <b>Bits 6–7</b> Reserved <b>Bits 8–15</b> Manufacturer Specific

3044 Both in the default parameter state and in any undefined parameter states, normal pixel array data shall be  
 3045 output instead of a test pattern. The individual test patterns are described later in this Section.

3046 If the image sensor is operated with non-mandatory features enabled (e.g. digital Gain), or with manufacturer-  
 3047 specific features, then the test patterns' operations might be undefined.

### 10.1.1 Scaled & Cropped and Re-Orientated Output

If the image sensor includes image-scaling options, then the Bayer test patterns need only be generated according to the specification when the scaling hardware is disabled. The test patterns are undefined if cropping or binning is selected, or if register **image\_orientation** is set to a non-zero value.

### 10.1.2 Solid Color Mode

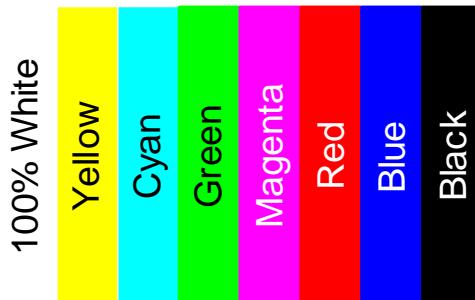
The image sensor shall support Solid Color Mode. In this Mode, all pixel data is replaced with fixed Bayer test data defined by the four ‘test data color’ parameters described in *Table 134*.

**Table 134 Test Data Color Parameters**

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

### 10.1.3 100% Color Bars Mode

The image sensor shall support 100% Color Bars Mode. In this Mode, all pixel data is replaced with a Bayer version of an 8-bar color bar pattern as shown in **Figure 58**.



**Figure 58 100% Color Bar Pattern**

In each color bar, all pixels in the same color plane shall have the same value. The values shall be either 0% or 100% of full scale. That is, for RAW8 100% represents value 255 (0xFF), and for RAW10 it represents value 1023 (0x3FF). For all data formats 0% represents value 0. Hence the colors are composed as shown in **Table 135**. The image sensor shall support at least RAW format with the same bit depth as the ADC.

**Table 135 100% Color Bar Values**

Bar Color	R	Gr, Gb	B
White	100%	100%	100%
Yellow	100%	100%	0%
Cyan	0%	100%	100%
Green	0%	100%	0%
Magenta	100%	0%	100%
Red	100%	0%	0%
Blue	0%	0%	100%
Black	0%	0%	0%

#### 10.1.3.1 Pattern Size

Pattern size rules are determined by the **pattern\_size\_div\_m1** and **test\_pattern\_capability** registers shown in **Table 136**.

When the output image width is a multiple of  $8 * (\text{pattern\_size\_div\_m1} + 1)$  pixels, each bar shall occupy exactly 1/8th of the output image width.

When the output image width is not a multiple of  $8 * (\text{pattern\_size\_div\_m1} + 1)$  pixels, the bar size shall be rounded down to the next lowest integer number of pixels that is a multiple of **pattern\_size\_div\_m1** + 1. Furthermore:

- If bit 0 of **test\_pattern\_capability** is 0, then the test pattern shall start to repeat at the right hand side of the image.
- Otherwise, the test pattern doesn't repeat and the width of the eighth color bar is extended as needed to fill the output image width. Note that the width of this extension is guaranteed to be a multiple of **pattern\_size\_div\_m1** + 1 because the output image width itself is required to be a multiple of **pattern\_size\_div\_m1** + 1.

The pattern shall always occupy the full output image height.

**Table 136 Test Pattern Capability Registers**

Parameter Name	Type	RW	Comment
<b>pattern_size_div_m1</b>	8-bit unsigned integer	RO	The widths of all full or repeated portions of sub-patterns (i.e., bars, sub-bars, and tiles) making up 100% color bar, fade to grey color bar, and 100% color tile test patterns are integer multiples of <b>pattern_size_div_m1 + 1</b> . The output image width must also be a multiple of <b>pattern_size_div_m1 + 1</b> . <b>Default:</b> 0
<b>test_pattern_capability</b>	8-bit unsigned integer	RO	<b>Bit 0 (Sub-pattern repetition)</b> 0: All eight 100% color bars, sixteen fade to grey sub-bars, and/or eight 100% color tiles have equal width within their respective test patterns and start repeating as necessary to fill the output image width. 1: Sub-patterns don't repeat. The first seven 100% color bars, fifteen fade to grey sub-bars, and/or seven 100% color tiles have equal width within their respective test patterns. The width of the eighth 100% color bar, sixteenth fade to grey sub-bar, and/or eighth 100% color tile is extended as necessary to fill the output image width. <b>Bits 1–7</b> Reserved for future use <b>Default:</b> 0

#### 10.1.4 Fade to Grey Color Bar Mode

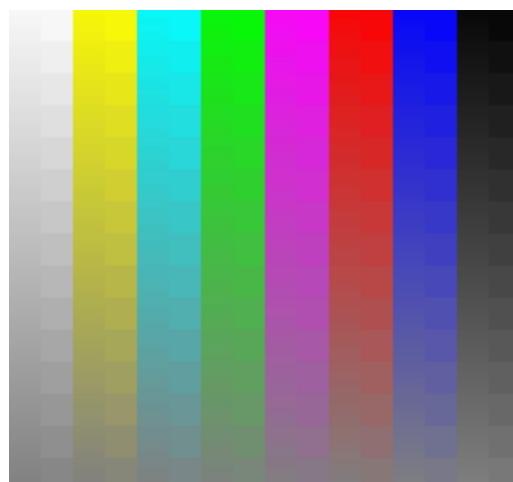
Image sensors should support Fade to Grey Color Bar Mode.

**Note:**

*It is expected that this Mode will become mandatory in a future version of the CCS Specification.*

In this Mode, all pixel data is replaced with a color bar that fades vertically from 100% color bars at the top, towards a mid-grey, based on control registers. This pattern is designed to exercise more of the color space than the 100% Color Bars Mode while still requiring only minimal hardware overhead.

While **Figure 59** illustrates the pattern's appearance, the test pattern itself is generated as Bayer data.



**Figure 59 Fade to Grey Color Bar Pattern**

- The pattern consists of 8 vertical bars that fade vertically from one of the 100% Color Bars colors at the top towards a mid-grey at the bottom
- The bar colors follow the same order as the 100% Color Bars test pattern
- Each of the bars is sub-divided vertically into a left half with a smooth color gradient, and a right half with a quantized version. See **Section 10.1.4.1** for definition of the pattern size.

The smooth gradient sub-bar values are defined as follows:

- As the color bar is in Bayer pattern, every two lines of the smooth gradient sub-bar have the same values.
- The first two lines (lines 0 and 1) shall have the same values as described in **Section 10.1.3**, i.e. either 100% or 0% (e.g., 100% is 255 for RAW8 format, or 1023 for RAW10 format).
- The value of the next two lines (lines 2 and 3) shall increment or decrement by the value of register **value\_step\_size\_smooth**, depending upon the value of the first two lines:
  - If the value of first two lines was 100%, then the value of the next two lines shall be the value of the first two lines minus the value of register **value\_step\_size\_smooth**.

**Examples:**

- For a **value\_step\_size\_smooth** of 1 and RAW8 format, the first two lines' value is 255 and the second two lines' value is 254.
- For a **value\_step\_size\_smooth** of 2 and RAW8 format, the first two lines' value is 255 and the second two lines' value is 253.
- For a **value\_step\_size\_smooth** of 1 and RAW10 format, the first two lines' value is 1023 and the second two lines' value is 1022.

- 3106     • For a **value\_step\_size\_smooth** of 2 and RAW10 format, the first two lines' value is 1023  
3107       and the second two lines' value is 1021.  
3108     • If the value of the first two lines was 0% (i.e. value 0), then the value of the next two lines shall  
3109       be the value of register **value\_step\_size\_smooth**.

3110       **Examples:**

- 3111       • For a **value\_step\_size\_smooth** of 1, the second two lines' value is 1.  
3112       • For a **value\_step\_size\_smooth** of 2, the second two lines' value is 2.  
3113     • The values of the remaining lines shall change with the same value step size  
3114       (**value\_step\_size\_smooth**) for every two lines (i.e. line step size = 2), for all data formats.  
3115     • The value of the last two lines shall depend upon the value on first two lines (though see *Section*  
3116       *10.1.4.1* for exceptions):  
3117       • If the value of the first two lines was 100%:

$$(2^{\text{number of bits}} - 1) - \text{value step size} * (2^{\text{(number of bits)}} / (\text{value step size}*2) - 1)$$

3119       **Examples:**

- 3120       • For a **value\_step\_size\_smooth** of 1 and RAW8 format, the value for the last two lines is 128  
3121       • For a **value\_step\_size\_smooth** of 1 and RAW10 format, the value for the last two lines is  
3122       512.  
3123       • If the value of the first two lines was 0%:

$$\text{value step size} * (2^{\text{(number of bits)}} / (\text{value step size}*2) - 1)$$

3125       **Examples:**

- 3126       • For a **value\_step\_size\_smooth** of 1 and RAW8 format, the value for the last two lines is 127  
3127       • For a **value\_step\_size\_smooth** of 1 and RAW10, the value for the last two lines is 511.

3128 The aim of the quantized portion is to offer areas of flat-field Bayer data large enough to result in known data  
3129 values, even after de-mosaic processing (independently of the particular de-mosaic algorithm being used).

3130 The quantized sub-bar values are defined as follows:

- 3131     • As the color bar is in Bayer pattern, every 16 lines of the quantized sub-bars have the same values.  
3132     • The first 16 lines (lines 0 through 15) shall have the same values as described in *Section 10.1.3*,  
3133       i.e. either 100% or 0% (e.g., 100% is 255 for RAW8 format, or 1023 for RAW10 format).  
3134     • The value of the next 16 lines (lines 16 through 31) shall increment or decrement by the value of  
3135       register **value\_step\_size\_quantised**, depending upon the value of the first 16 lines:  
3136       • If the value of first 16 lines was 100%, then the value of the next 16 lines shall be the value of  
3137       the first 16 lines minus the value of register **value\_step\_size\_quantised**.

3138       **Examples:**

- 3139       • For a **value\_step\_size\_quantised** of 8 and RAW8 format, lines 0 through 15 have value 255  
3140       and lines 16 through 31 have value 247.  
3141       • For a **value\_step\_size\_quantised** of 16 and RAW8 format, lines 0 through 15 have value  
3142       255 and lines 16 through 31 have value 239.  
3143       • For a **value\_step\_size\_quantised** of 8 and RAW10 format, lines 0 through 15 have value  
3144       1023 and lines 16 through 31 have value 1015.  
3145       • For a **value\_step\_size\_quantised** of 16 and RAW10 format, lines 0 through 15 have value  
3146       1023 and lines 16 through 31 have value 1007.

- 3147     • If the value of the first 16 lines was 0% (i.e. value 0), then the value of the next 16 lines shall be  
3148       the value of register **value\_step\_size\_quantised**.

3149       **Examples:**

- 3150       • For a **value\_step\_size\_quantised** of 8, lines 16 through 31 have value 8.  
3151       • For a **value\_step\_size\_quantised** of 16, lines 16 through 31 have value 16.

- 3152     • The values of the remaining lines shall change with the same value step size  
3153       (**value\_step\_size\_quantised**) for every 16 lines (i.e. line step size = 16), for all data formats.  
3154     • The value of the last 16 lines shall depend upon the value on first two lines (though see *Section*  
3155       *10.1.4.1* for exceptions):  
3156       • If the value of the first two lines was 100%:

3157               $(2^{\text{number of bits}} - 1) - \text{value step size} * (2^{\text{(number of bits)}} / (\text{value step size} * 2) - 1)$

3158       **Examples:**

- 3159       • For a **value\_step\_size\_quantised** of 8 and RAW8 format, the value for the last 16 lines is  
3160        135.  
3161       • For a **value\_step\_size\_quantised** of 8 and RAW10 format, the value for the last 16 lines is  
3162        519.  
3163       • If the value of the first two lines was 0%:

3164               $\text{value step size} * (2^{\text{(number of bits)}} / (\text{value step size} * 2) - 1)$

3165       **Examples:**

- 3166       • For a **value\_step\_size\_quantised** of 8 and RAW8 format, the value for the last 16 lines is  
3167        120.  
3168       • For a **value\_step\_size\_quantised** of 8 and RAW10, the value for the last 16 lines is 504.

3169     The image sensor shall support color bars generated by following rules, and may optionally support additional  
3170     color bars:

- 3171     • Rule 1: For all data formats:  
3172       • **value\_step\_size\_smooth** = 1 and  
3173        **value\_step\_size\_quantised** = 8  
3174     • Rule 2: For RAW10 and other data formats with more than 8 bits:  
3175       • **value\_step\_size\_smooth** =  $2^{\text{(number of bits} - 8)}$  and  
3176        **value\_step\_size\_quantised** =  $8 * 2^{\text{(number of bits} - 8)}$

3177     The image sensor shall support Fade to Grey color bars with at least a RAW format of the same number of  
3178     bits as the ADC bit depth.

3179       **Examples:**

- 3180       • An 8-bit ADC requires RAW8, and only one color bar format is required: RAW8.  
3181       • A 10-bit ADC requires RAW10, so two color bar formats are required: RAW8 (per Rule 1) and  
3182        RAW10 (per Rule 2).

3183     The advantage of having multiple Rules is always being able to verify a large data range with 1 LSB accuracy  
3184     using Rule 1. The drawback of Rule 1 is a large bar height.

3185     Rule 2 has a smaller bar size (i.e., 2 x 128 pixels height), at the cost of larger value steps. It is important to  
3186     notice both Rules use the same line step values: 2 for the smooth bar, and 16 for the quantized bar. According  
3187     to Rule 2, for RAW10 the value of **value\_step\_size\_smooth** is 4, and the value of the last two lines of the  
3188     smooth sub-bars is either 508 if starting from 0%, or 515 if starting from 100%. On the quantized side, for a  
3189     **value\_step\_size\_quantised** of 32, the value of the last 16 lines of the quantized sub-bar is either 480 if  
3190     starting from 0%, or 543 if starting from 100%.

3191  
3192 **Table 137** further illustrates Fade to Grey's Smooth and Quantized sub-bars with example values for various combinations of bit depth and value step size.

3193

**Table 137 Example of Fade to Grey Values**

Bits	First Line Value	Value Step Size	Last Line Value	Sub-Bar Height in Pixels	Sub-Bar Type
8	255	1	128	256	Smooth
	0	1	127	256	Smooth
	255	8	135	256	Quantized
	0	8	120	256	Quantized
10	1023	1	512	1024	Smooth
	0	1	511	1024	Smooth
	1023	8	519	1024	Quantized
	0	8	504	1024	Quantized
	1023	4	515	256	Smooth
	0	4	508	256	Smooth
	1023	32	543	256	Quantized
	0	32	480	256	Quantized

3194

**Table 138 Fade to Grey Control Registers**

Register Name	Type	RW	Comment
<b>value_step_size_smooth</b>	8-bit unsigned integer	RW	Controls value step for the Smooth gradient sub-bar <b>Default:</b> Image-sensor-specific
<b>value_step_size_quantised</b>	8-bit unsigned integer	RW	Controls value step size for the Quantized sub-bar <b>Default:</b> Image-sensor-specific

#### 10.1.4.1 Pattern Size

Pattern size rules are determined by the **pattern\_size\_div\_m1** and **test\_pattern\_capability** registers shown in *Table 136*.

When the output image width is a multiple of  $16 * (\text{pattern\_size\_div\_m1} + 1)$  pixels, each fade to grey color sub-bar shall occupy exactly 1/16th of the output image width.

When the output image width is not a multiple of  $16 * (\text{pattern\_size\_div\_m1} + 1)$  pixels, the sub-bar size shall be rounded down to the next lowest integer number of pixels that is a multiple of **pattern\_size\_div\_m1 + 1**.

Furthermore:

- If bit 0 of **test\_pattern\_capability** is 0, then the test pattern shall start to repeat at the right hand side of the image.
- Otherwise, the test pattern doesn't repeat and the width of the sixteenth fade to grey color sub-bar is extended as needed to fill the output image width. Note that the width of this extension is guaranteed to be a multiple of **pattern\_size\_div\_m1 + 1** because the output image width itself is required to be a multiple of **pattern\_size\_div\_m1 + 1**.

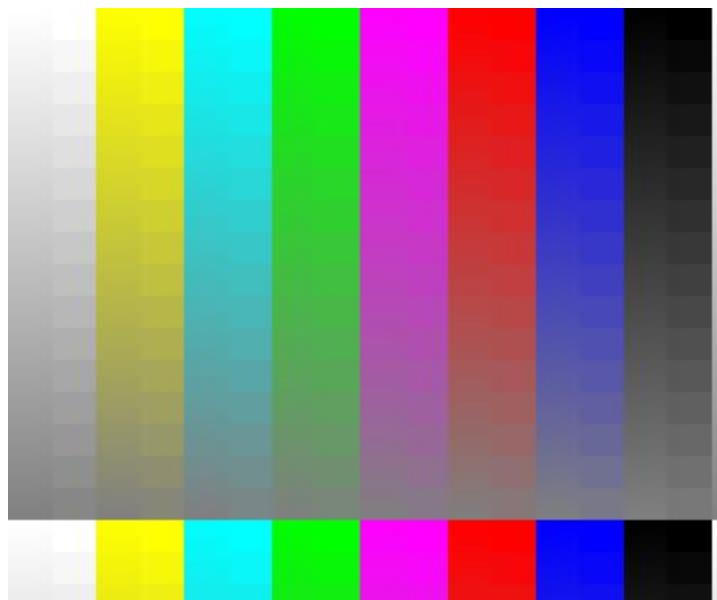
Because gradient data values are most efficiently generated directly from a row counter, an arbitrary vertical size is not always convenient for Fade to Grey color bars. The height of each color bar shall therefore follow the formula:

$$\lceil 2^{(\text{number of bits}) / (\text{value step size})} \rceil * (\text{line step size}) / 2$$

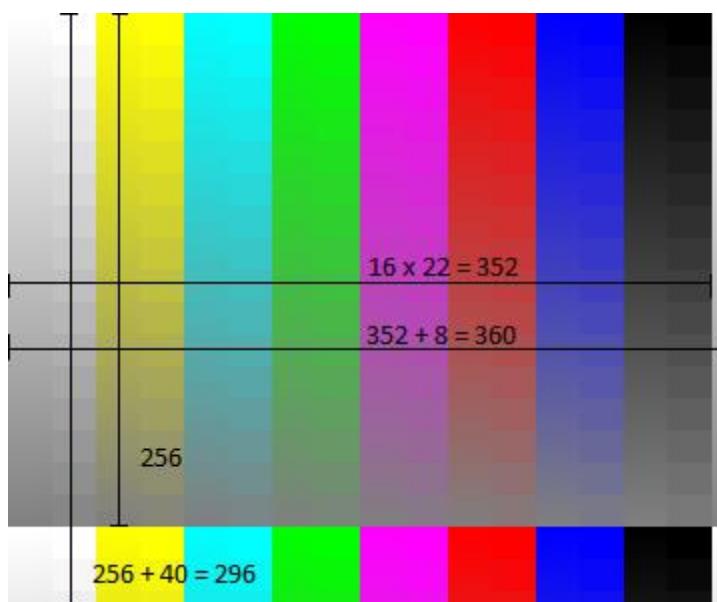
The only exception to this formula is that if the actual image height is less than the calculated bar height, then the bars shall omit the bottom lines.

If the actual image height is greater than the calculated bar height, then the color bar pattern shall repeat on any remaining image lines.

*Figure 60* (with **test\_pattern\_capability** = 0, **pattern\_size\_div\_m1** = 0) shows an example RAW8 360x296 pattern where **value\_step\_size\_smooth** value is 1 and **value\_step\_size\_quantised** is 8. A similar-looking pattern could be generated for RAW10 with **value\_step\_size\_smooth** of 4 and **value\_step\_size\_quantised** of 32. See *Table 137* for actual sub-bar values *Figure 61* shows exactly the same color bar pattern as *Figure 60*, adding only the pixel dimension information.



3222

**Figure 60 360x296 Fade to Grey Color Bar Pattern**

3223

**Figure 61 360x296 Fade to Grey Color Bar Pattern with Dimensions**

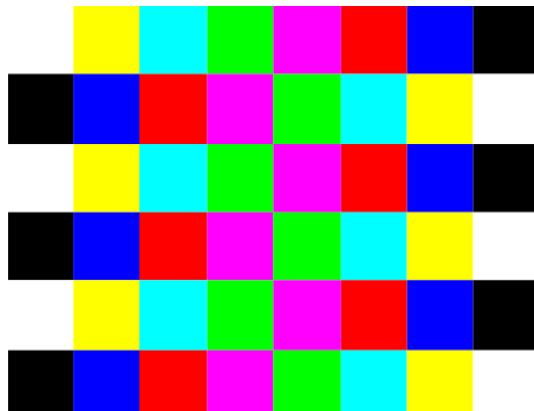
### 10.1.5 100% Color Tile Mode

Image sensors should support 100% Color Tile Mode.

**Note:**

*It is expected that this Mode will become mandatory in a future version of the CCS Specification.*

In this Mode, all pixel data is replaced with a Bayer version of an 8-bar color tile pattern, as illustrated in **Figure 62**.



**Figure 62 100% Color Tile Pattern**

The 100% Color Tile pattern shall consist of N lines of the 100% Color Bars Pattern (per **Section 10.1.3**), followed by N lines of the same pattern in horizontally reversed order, and so forth, repeating for the full image height. The height and the width per tile shall be the same except as noted by the pattern size specifications in **Section 10.1.5.1**.

As the color tile is in Bayer pattern, every  $2^*N$  lines have the same values. The first  $2^*N$  lines shall have the same values as the 100% Color Bars Pattern (per **Section 10.1.3**), and the next  $2^*N$  lines shall have the same values, but in horizontally reversed order.

**Table 139** illustrates the color tile values, which shall be either 100% of full scale, or 0% of full scale (e.g., 100% is 255 for RAW8 format, or 1023 for RAW10 format). 0% represents value 0 for all data formats.

**Table 139 100% Color Tile Values**

Tile Color	R	Gr, Gb	B
White	100%	100%	100%
Yellow	100%	100%	0%
Cyan	0%	100%	100%
Green	0%	100%	0%
Magenta	100%	0%	100%
Red	100%	0%	0%
Blue	0%	0%	100%
Black	0%	0%	0%

The image sensor shall support the 100% Color Tile pattern with at least a RAW format of the same number of bits as the ADC bit depth.

### 10.1.5.1 Pattern Size

Pattern size rules are determined by the **pattern\_size\_div\_m1** and **test\_pattern\_capability** registers shown in *Table 136*.

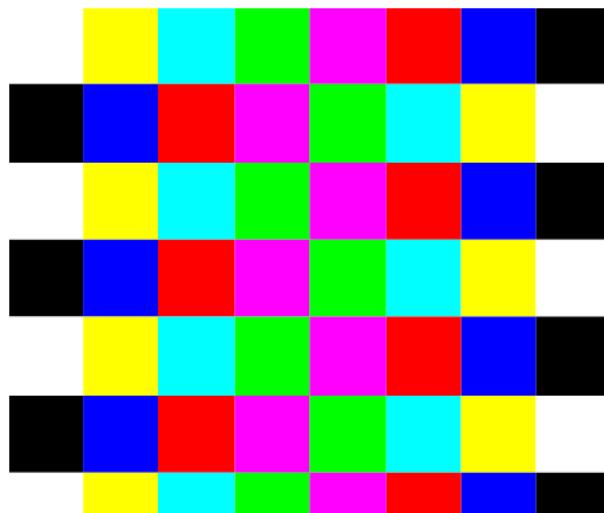
When the output image width is a multiple of  $8 * (\text{pattern\_size\_div\_m1} + 1)$  pixels, each color tile shall occupy exactly 1/8th of the output image width.

When the output image width is not a multiple of  $8 * (\text{pattern\_size\_div\_m1} + 1)$  pixels, the tile size shall be rounded down to the next lowest integer number of pixels that is a multiple of **pattern\_size\_div\_m1 + 1**. Furthermore:

- If bit 0 of **test\_pattern\_capability** is 0, then the test pattern shall start to repeat at the right hand side of the image.
- Otherwise, the test pattern doesn't repeat and the width of the eighth color tile is extended as needed to fill the output image width. Note that the width of this extension is guaranteed to be a multiple of **pattern\_size\_div\_m1 + 1** because the output image width itself is required to be a multiple of **pattern\_size\_div\_m1 + 1**.

The height of each tile shall be the same as the width of each tile, except when (a) the output image height is not divisible by the tile width or (b) the width of every eighth tile in a row is extended to fill the output image width. If (b) is true, then the height of every eighth tile shall be the same as the height of the other tiles in the same row. If (a) is true, then the height of the last row of tiles in the output image shall be reduced as necessary in order to fill the full output image height (as shown by the example in *Figure 63*). The pattern of normal and reversed tile lines shall repeat for the full image height.

An example of a 128x104 pattern with 16 x 16 tiles is shown in *Figure 63*.



**Figure 63 Example of 100% Color Tile Pattern**

### 10.1.6 PN9 Mode

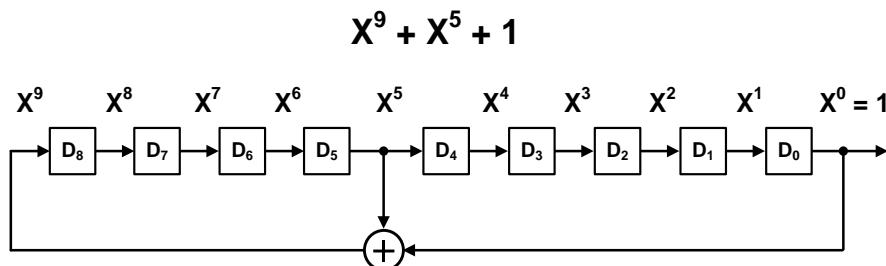
Image sensors should support PN9 Mode.

**Note:**

*It is expected that this Mode will become mandatory in a future version of the CCS Specification.*

In this Mode, all pixel data is replaced with data from an internally generated 512-bit pseudo-random PN9 sequence. This test pattern is included in order to ease testing the integrity of the link between the image sensor and the host, in terms of measuring bit error rate, etc.

The image sensor PN9 linear feedback shift register shall have the polynomial  $X^9 + X^5 + 1$  in Fibonacci type notation, as per **Figure 64**.



**Figure 64 PN9 Linear Feedback Shift Registers**

The image sensor shall reset the PN9 sequence generator so as to start the sequence in a known state for the first replaced pixel of each frame. The PN9 reset state shall be 0x1FF.

The PN9 sequence generator shall replace the real pixel data of the CSI-2 payload without any impact upon the embedded data.

Any impact of the PN9 sequence generator upon any other Virtual Channels or Data Types, for example upon interleaved PDAF data, is undefined.

After reset at the beginning of each image frame, the serial PN9 generator depicted in **Figure 64** shall output one bit for every pixel bit position, beginning with the LSB, shifting a total of n bits for an n-bit pixel.

The preferred behavior is for the PN9 generator to output a total of m\*n bits to replace all the n-bit pixels in an m-pixel line (i.e. a unique PN9 pixel replaces every original pixel).

**Note:**

*Uniqueness is subject to the 511 cycle periodicity inherent in PN9 sequences.*

However, an image sensor may also choose to generate only one unique PN9 pixel for every kth pixel in an m-pixel line, where k > 1, generating a total of ceiling(m/k) PN9 pixels for the entire line.

**Note:**

*The ceiling function returns the greatest integer value less than or equal to its argument.*

In this case, pixels (j-1) \* k+1 through min(j\*k, m) are all replaced with PN9 pixel j, for j = 1 to ceiling(m/k). The value of k shall be less than or equal to eight; note that k = 1 indicates that a unique PN9 pixel replaces every original pixel.

Note that if m/k is not an integer, then there will be k - (m mod k) unused copies of PN9 pixel j, for j = ceiling(m/k), at the end of each line; these copies are discarded and not carried over to the next line. A new PN9 pixel is always generated at the beginning of each line.

The value of k, the total number of supported PN9 pixel widths, and the pixel widths themselves shall all be discoverable using the image sensor PN9 Capability Registers shown in **Table 141**. Up to four different pixel widths may be supported. If the image sensor supports the PN9 test pattern mode, then it shall support at least one PN9 pixel width.

When the PN9 test pattern mode is used, the Host shall program register **CSI\_data\_format**(15:8) with a value that is supported by PN9. Image sensors supporting compression of normal image pixels are not required to support compression of PN9 pixels having the same width; for such image sensors, **CSI\_data\_format**(15:8) and **CSI\_data\_format**(7:0) shall both be programmed to the same PN9-supported value. For PN9 calculations, image sensor shall use the bit depth according to register **CSI\_data\_format**(15:8).

If the PN9 test pattern mode is selected, and **CSI\_data\_format**(15:8) is programmed to a value that is not supported by the image sensor for PN9, then the image sensor may operate normally and not replace image pixels with PN9 pixels.

**Table 140 PN9 Capability Registers**

Register Name	Type	RW	Comment
<b>PN9_data_format1</b>	8-bit unsigned integer	RO	First supported PN9 pixel width. Shall be a non-zero value if PN9 is supported.
<b>PN9_data_format2</b>	8-bit unsigned integer	RO	Second supported PN9 pixel width. 0 indicates a null value.
<b>PN9_data_format3</b>	8-bit unsigned integer	RO	Third supported PN9 pixel width. 0 indicates a null value.
<b>PN9_data_format4</b>	8-bit unsigned integer	RO	Fourth supported PN9 pixel width. 0 indicates a null value.
<b>PN9_misc_capability</b>	8-bit unsigned integer	RO	<b>Bits 2-0</b> The number of consecutive pixels (minus 1) having the same unique PN9 value (i.e., k – 1, where 1 ≤ k ≤ 8). <b>Bit 3</b> 0: Compression is not supported for PN9 pixels (default) 1: Compression is supported for PN9 pixels <b>Other Bits:</b> Reserved for future use

## 11 Image Scaling

### 11.1 Introduction

An image sensor may scale its output image size by the following three ‘digital’ functions. These functions can be used in addition to, or independently of, the ‘analog’ functions (region of interest, binning and sub-sampling) described in earlier Sections:

- Digital crop at the input of the scaler (optional)
- Scaler (optional)
- Output crop (mandatory)

This Specification defines controls for two types of image sensors:

- Image sensors with no image scaler
- Image sensors that support no scaling and horizontal scaling

CCS does not specify any controls for vertical scaling. Vertical scaling may be supported via manufacturer-specific registers.

Digital crop before the scaler can be used instead of analog cropping by the analog ROI function (see [Section 8.2.1.1](#) for details). This may be required in order to implement smooth digital zoom, where analog ROI might cause frame corruption and hence an uneven frame rate. Digital crop is specified in [Section 11.2](#).

Output crop can be used to adjust output sizes after the scaler. However, due to the restrictions on its use, output crop should not be considered the primary cropping control. Some examples are described in [Section 7.3](#). Output crop control functionality is specified in [Section 8.2.1.2](#).

The following rules apply for the scaler:

- The scaler should be capable of scaling the full image sensor resolution down to within 10% of the target image size (the smallest image width is 320). If the image sensor does not implement this recommendation, then this should be noted in the image sensor datasheet.
- A 10% crop is the maximum acceptable error in the field of view of the image sensor output image.

In order to fulfil the rules, the down-scale factor shall be programmable in steps of 1/16<sup>th</sup> or more.

The Host can determine the image sensor’s image scaling capabilities by inspecting the registers **scaling\_capability** and **digital\_crop\_capability**.

**Table 141 Image Scaling Capability Register**

Register Name	Type	RW	Comment
<b>scaling_capability</b>	16-bit unsigned integer	RO	0: None 1: Horizontal 2: Reserved

**Table 142 Digital Crop Capability Register**

Register Name	Type	RW	Comment
<b>digital_crop_capability</b>	8-bit unsigned integer	RO	0: None 1: input digital crop supported

## 11.2 Digital Crop

3335 *Figure 65* shows the location of the digital crop function at the scaler input.

3336 The image area after digital crop depends on:

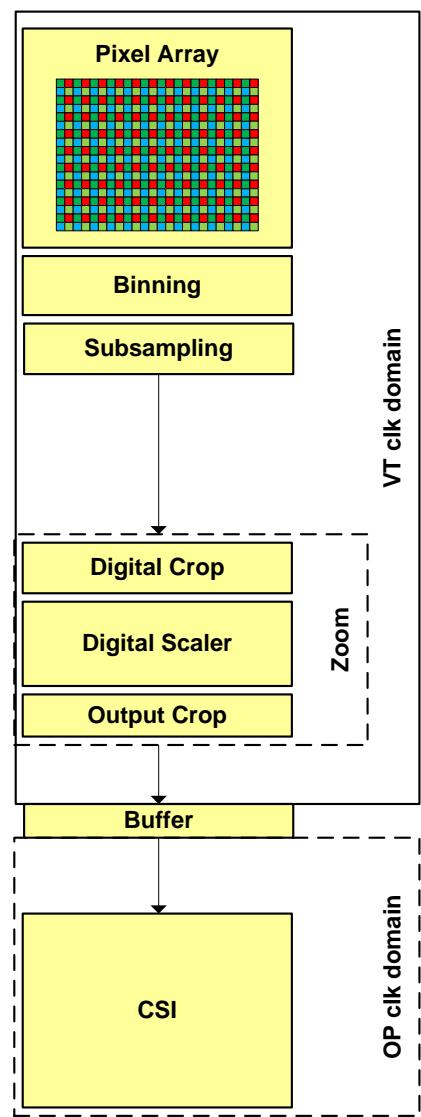
- 3337 1. The physical pixel array size (delimited by registers **x\_addr\_min**, **y\_addr\_min**, **x\_addr\_max** and  
**y\_addr\_max**)
- 3339 2. The region of interest (or analog cropping), determined by **x\_addr\_start**, **y\_addr\_start** and  
**x\_addr\_end**, **y\_addr\_end**
- 3341 3. The amount of binning selected by **binning\_mode**
- 3342 4. The amount of subsampling, selected by **x\_even\_inc**, **x\_odd\_inc**, **y\_even\_inc**, and **y\_odd\_inc**
- 3343 5. The offset of digital crop by **digital\_crop\_x\_offset** and **digital\_crop\_y\_offset**
- 3344 6. The image width and height of the digital crop by **digital\_crop\_image\_width** and  
**digital\_crop\_image\_height**.

3346 Note that the array size that the digital crop controls will operate is affected by the amount of binning and  
3347 subsampling, as well as by the analog ROI. The Host shall take these factors into account when calculating  
3348 the correct control values to use for digital crop. Since the input to the digital crop block is variable, no limit  
3349 registers are associated with digital crop.

3350 *Figure 66* shows which pixel array pixels are visible after analog and digital crop, and which commands are  
3351 used for control.

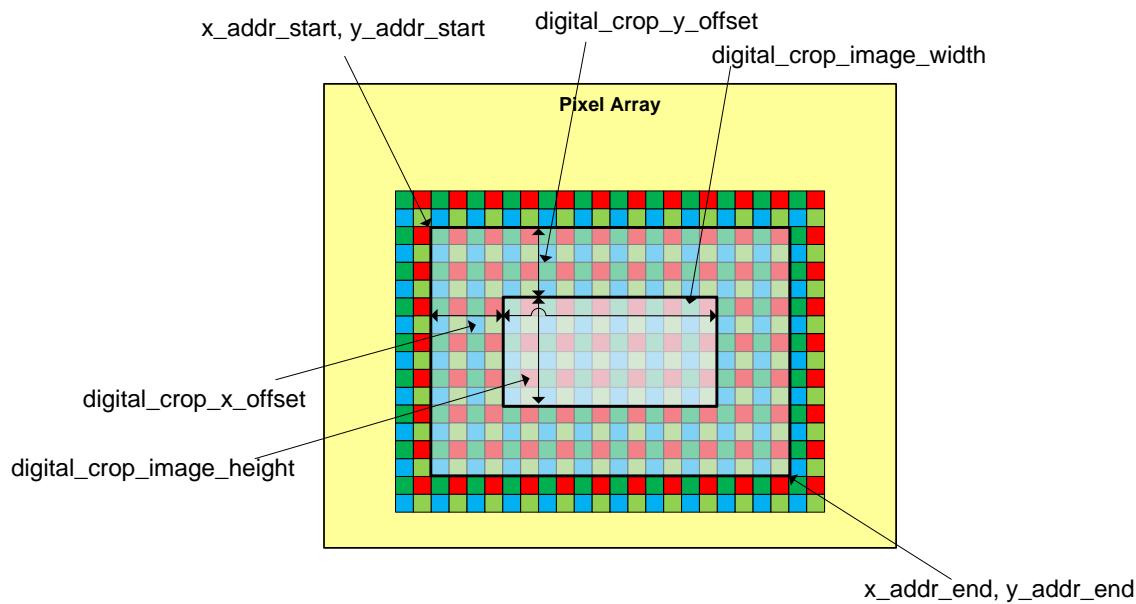
3352 **Table 143 Digital Crop Control Register**

Register Name	Type	RW	Comment
<b>digital_crop_x_offset</b>	16-bit unsigned integer	RW	Offset from X-address of the top left corner of the visible pixel data after analog crop, bin and subsample <b>Values:</b> Even numbers only, i.e. 0, 2, 4, 6... <b>Units:</b> Pixels <b>Default:</b> 0x00
<b>digital_crop_y_offset</b>	16-bit unsigned integer	RW	Offset from Y-address of the top left corner of the visible pixel data after analog crop, bin and subsample <b>Values:</b> Even numbers only, i.e. 0, 2, 4, 6... <b>Units:</b> Lines <b>Default:</b> 0x00
<b>digital_crop_image_width</b>	16-bit unsigned integer	RW	Image width after digital crop. <b>Values:</b> Even numbers only, i.e. 0, 2, 4, 6... <b>Units:</b> Pixels <b>Default:</b> Image-sensor-specific
<b>digital_crop_image_height</b>	16-bit unsigned integer	RW	Image height after digital crop. <b>Values:</b> Even numbers only, i.e. 0, 2, 4, 6... <b>Units:</b> Lines <b>Default:</b> Image-sensor-specific



3353

**Figure 65 Data Flow in Scaling**



3354

**Figure 66 Control for Image Area After Analog and Digital Crop**

### 11.3 Scaler Sampling

3355 The scaler in a Bayer image sensor shall support Bayer sampling for the scaled image data. With a non-Bayer  
3356 image sensor the sampling is implementation-specific.

### 11.4 Down Scaler Factor

3357 The scaler step size (downscale factor) should be programmable from 1.0 up to the scaler step size required  
3358 to downscale the maximum addressable width of the pixel array down to an output image width of 320 pixels.

3359 The down scaler factor is controlled by an M/N ratio. M shall be  $\geq N$ , and N is a fixed, vendor specific value.  
3360 Typically N has value 16 and M is  $\geq 16$ .

$$3361 \quad \text{down\_scale\_factor} = \frac{\text{scale\_m}}{\text{scale\_n}} = \frac{\text{scale\_m}}{16}$$

3362 For example, if an image from a 1280 x 720 sensor is vertically binned to 1280 x 360, and then input to a  
3363 horizontal scaler with M=32 and N=16, then the output will be 640 x 360 pixels.

## 11.5 Control Registers

The horizontal and vertical scalers are programmable via two parameters, see **Table 141** and **Table 144**.

- Scaler mode: Off / Horizontal
- Scaler down scale factor (Down-scale factor)

All of the scaler read/write registers (**Table 144**) shall be re-timed within the image sensor to the start of a frame boundary, in order to ensure that the scaler values are consistent within each frame of image data.

The minimum and maximum values for M and N are reported via the image scaling capability registers (**Table 145**).

**Table 144 Image Scaling Registers**

Register Name	Type	RW	Comment
<b>scaling_mode</b>	16-bit unsigned integer	RW	0: None 1: Horizontal 2: Reserved <b>Default:</b> 0
<b>scale_m</b>	16-bit unsigned integer	RW	Down scale factor: M component <b>Range:</b> From N upwards in increments of 1, until the image size is reduced to 320 pixels wide. <b>Default:</b> Image-sensor-specific
<b>scale_n</b>	16-bit unsigned integer	RO	Down scale factor: N component <b>Value:</b> Vendor-specific (fixed). Typical value is 16.

**Table 145 Image Scaling Capability Registers**

Register Name	Type	RW	Comment
<b>scale_m_min</b>	16-bit unsigned integer	RO	Down scale factor: Minimum M value Value shall be the same as <b>scale_n_max</b>
<b>scale_m_max</b>	16-bit unsigned integer	RO	Down scale factor: Maximum M value
<b>scale_n_min</b>	16-bit unsigned integer	RO	Down scale factor: Minimum N value Vendor specific value, typically 16
<b>scale_n_max</b>	16-bit unsigned integer	RO	Down scale factor: Maximum N value Vendor specific value, typically 16

## 11.6 FIFO Control Registers

Some image sensors support a clock tree with separate VT and OP domains, as described in *Section 8.3.1*. In order to implement decoupling of the VT domain from the OP domain, the image sensor must use a FIFO. A scaler might also require a FIFO.

If a FIFO is supported, then the image sensor shall manage the FIFO in a manner that is transparent to the Host. Entities other than the image sensor shall not manage the FIFO.

Decoupling of the VT and OP domains means two typical use cases:

- The VT domain is running faster than OP domain, so the FIFO is used for derating
- The VT domain is running slower than OP domain, so the FIFO is used for overrating

Decoupling is important, as it gives the Host the ability to set appropriate OP domain frequencies as needed to suit its available bandwidth, and in order to avoid any EMC-sensitive frequencies without restricting control of the VT domain.

All image sensors with a scaler shall support derating, and should support overrating.

**Table 146 FIFO Control Registers**

Register Name	Type	RW	Function
<b>fifo_support_capability</b>	8-bit unsigned integer	RO	0: Not supported 1: Supports derating 2: Supports both derating and overrating

## 12 Image Compression

### 12.1 Introduction

The objective of the image compression is to reduce the required bandwidth in transmission between the image 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

All image sensors should support the 10-bit to 8-bit DPCM/PCM image compression algorithm.

The level of compression is controlled via the **CSI\_data\_format** register. The same register is also used to enable and disable compression. All compressions shall support Simple Predictor.

The **compression\_mode** register is used to select simple predictor for the compression algorithm.

**Table 147 Compression Mode Register**

Register Name	Type	RW	Comment
<b>compression_mode</b>	16-bit unsigned integer	RO	0x01: DPCM/PCM – Simple Predictor Other Values: Reserved <b>Default:</b> 0x01

The **compression\_capability** register tells the Host whether an image sensor does have or does not have compression and if it has compression then what is the compression technique.

This Specification supports only the DPCM/PCM technique.

**Table 148 Compression Capability Register**

Register Name	Type	RW	Comment
<b>compression_capability</b>	16-bit unsigned integer	RO	0x00: None 0x01: DPCM/PCM

### 12.2 Compression Algorithms

The following compression algorithms are specified in the MIPI CSI-2 Specification, Version 1.3 [**MIPI01**]:

- 10-8-10
- 10-7-10
- 10-6-10
- 12-8-12
- 12-7-12
- 12-6-12

The following compression algorithm is introduced in the MIPI CSI-2 Specification, Version 2.0 [**MIPI06**]:

- 12-10-12

This page intentionally left blank.

## 13 Data Transfer Interface

### 13.1 Introduction

Present and future camera modules are likely to contain increasing levels of built-in functionality that may require the upload or download of significant amounts of data. The transferred data could be, for example, calibration data upload to the ISP from the image sensor OTP/EEPROM, or a firmware patch download from the SW driver to the image sensor.

This Section presents a standardized method for such data upload and download. The data transfer mechanism uses a ‘paged’ technique.

Both read and write interfaces, if supported, shall be full accessible during both SW Standby Mode and Streaming Mode.

If the image sensor has NVM for calibration data usage, the image sensor should support this data transfer interface.

### 13.2 Capabilities for Data Transfer Interface

The Host can determine whether the image sensor supports the data transfer interface by reading bit 0 of the register **data\_transfer\_if\_capability**. The Host can determine whether polling is required for reading by inspecting bit 2 of the register **data\_transfer\_if\_capability**.

Table 149 Data Transfer Interface Capability Registers

Register Name	Type	RW	Comment
<b>data_transfer_if_capability</b>	8-bit unsigned integer	RO	<b>Bit 0</b> 1: Interface 1 supported <b>Bit 1</b> Reserved, report value 0. <b>Bit 2</b> 0: Polling not needed in reading 1: Polling is needed in reading <b>Bit 3</b> Reserved <b>Other Bits</b> Reserved for future use

### 13.3 Control Register and Usage of Data Transfer Interface

3424

Table 150 Data Transfer Interface 1 Registers

Register Name	Type	RW	Comment
<b>data_transfer_if_1_ctrl</b>	8-bit unsigned integer	RW	<p><b>Bit 0</b> 1: Enable 0: Disable</p> <p><b>Bit 1</b> 1: Write enable 0: Read enable</p> <p><b>Bit 2</b> 1: Clear error bits.</p> <p><b>Default Value:</b> 0x00</p>
<b>data_transfer_if_1_status</b>	8-bit unsigned integer	RO dynamic	<p><b>Bit 0</b> 1: <b>read_if_ready</b></p> <p><b>Bit 1</b> 1: <b>write_if_ready</b></p> <p><b>Bit 2</b> 1: <b>Data_corrupted</b></p> <p><b>Bit 3</b> 1: <b>Improper_if_usage</b></p>
<b>data_transfer_if_1_page_select</b>	8-bit unsigned integer	RW	Pages from 0–255 <b>Default Value:</b> 0x00
<b>data_transfer_if_1_data</b>	64 x 8-bit	RW	64 x 8-bit register for read or write accesses

### 13.3.1 Read Sequence

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

1. The Host shall select the appropriate page by writing the required value to register **data\_transfer\_if\_1\_page\_select**.
2. The Host shall write the value 0x01 to the register **data\_transfer\_if\_1\_ctrl**. The image sensor shall then start preparing data in the read interface.
3. The Host shall wait until data is ready. If polling is needed, the Host shall poll bit **read\_if\_ready** in register **data\_transfer\_if\_1\_status**; otherwise, skip this step.
4. The Host shall read data from the register **data\_transfer\_if\_1\_data**. The image sensor presents the selected page as 64 x 8-bit words (512 bits total).
5. If access to a further page is needed, the Host shall first select a new page via register **data\_transfer\_if\_1\_page\_select**, and then perform steps 3 and 4 again.
6. Once the Host has read all the needed data, it shall disable the read interface by writing the value 0x00 to the register **data\_transfer\_if\_1\_ctrl**.
7. After the read interface is disabled, the Host may then start a new read sequence, or a new write sequence. The Host shall not switch between read sequences and write sequences without first disabling the data transfer interface.

### 13.3.2 Write Sequence

The data transfer interface can also be used for writing data. A typical write sequence would be as follows:

1. The Host shall select the appropriate page by writing the required value to register **data\_transfer\_if\_1\_page\_select**.
2. The Host shall write the value 0x03 to the register **data\_transfer\_if\_1\_ctrl**, enabling writing for data transfer interface 1. The image sensor shall then start preparing the write interface.
3. The Host shall write data to register **data\_transfer\_if\_1\_data**. This data represents the data in the selected page. The image sensor shall shadow the data, as the same interface can also be used for accessing different pages, or as a data read interface.

**Note:**

*The image sensor can wait to start shadowing until the Host polls the **write\_if\_ready** bit.*

4. When writing to new page is needed:
  - A. The Host shall poll bit **write\_if\_ready** in register **data\_transfer\_if\_1\_status**, to verify that shadowing has ended.
  - B. Once shadowing has ended, the Host shall select a new page via register **data\_transfer\_if\_1\_page\_select**, and then continue writing as per step 3.
5. Repeat steps 3 and 4 until the Host has written all needed data.
6. Once the Host has written all the needed data, it shall poll bit **write\_if\_ready** in register **data\_transfer\_if\_1\_status** to ensure that shadowing has ended
7. Once shadowing has ended, the Host shall write the value 0x00 to register **data\_transfer\_if\_1\_ctrl**, disabling the write interface.
8. After the write interface is disabled, the Host may then start a new read sequence or a new write sequence. The Host shall not switch between write sequences and read sequences without first disabling the data transfer interface.

### 13.3.3 Error and Special Cases

If the Host selects a page that doesn't exist, in the read case the image sensor may return undefined data values, but shall respond to any commands normally in all other respects. The image sensor shall also set the **improper\_if\_usage** bit in the **data\_transfer\_if\_1\_status** register.

A Host write to a page that doesn't exist shall have no effect on the image sensor's operation. The image sensor shall set the **improper\_if\_usage** bit in register **data\_transfer\_if\_1\_status**.

If the image sensor uses checksums for the data, and if the checksums fails to match, then the image sensor shall set the **data\_corrupted** bit in register **data\_transfer\_if\_1\_status**.

The Host can clear error bits by writing to bit 2 of register **data\_transfer\_if\_1\_ctrl**.

## 14 Image Processing and Sensor Corrections

### 14.1 Introduction

Although image signal processing is typically done outside of the image sensor, an image sensor might support some amount of image data pre-processing; for example, defect pixel correction. This Section specifies control mechanisms for such functionalities.

### 14.2 Sensor Processing Control Interface

This control interface standardizes the interface between the Host and functions in the image sensor that correct for imperfections in the raw data from the pixel array.

Functions considered here are:

- Color shading correction
- Luminance shading correction
- Green imbalance correction
- Defect correction
- Noise filter

3483 The image sensor shall indicate capabilities for the above functions via the registers specified in **Table 151**.

3484

**Table 151 Sensor Processing Capabilities**

Register Name	Type	RW	Comment
<b>shading_correction_capability</b>	8-bit unsigned integer	RO	<b>Bit 0</b> 1: Color shading correction supported <b>Bit 1</b> 1: Luminance correction adjust supported
<b>green_imbalance_capability</b>	8-bit unsigned integer	RO	<b>Bit 0</b> 1: Green imbalance correction supported
<b>NF_capability</b>	8-bit unsigned integer	RO	<b>Bit 0</b> 1: Luma noise filter supported <b>Bit 1</b> 1: Chroma noise filter supported. <b>Bit 2</b> 1: Combined noise filter supported. <b>Bit 3</b> Reserved If the image sensor only supports a generic noise filter that can't separate luma and chroma noise, then report that a combined noise filter is supported.
<b>defect_correction_capability</b>	16-bit unsigned integer	RO	<b>Bit 0</b> 1: Mapped defect correction supported <b>Bit 2</b> 1: Dynamic couplet correction supported <b>Bit 5</b> 1: Dynamic single pixel defect correction supported <b>Bit 8</b> 1: Combined dynamic couplet & single pixel defect correction supported <b>Other Bits</b> Reserved for future extensions
<b>defect_correction_capability_2</b>	16-bit unsigned integer	RO	<b>Bit 3</b> 1: Dynamic triplet correction supported <b>Other Bits</b> Reserved for future extensions
<b>module_specific_correction_capability</b>	8-bit unsigned integer	RO	<b>Bit 0</b> 0: Not supported 1: Supported

3485  
3486

Note that from the Host perspective it assumed that non-radial, radial, and luminance shading are all correctable by a single functional block.

Some of the functions above have an associated slider. The slider is intended to be a simple Host control for the strength of the correction applied, without the Host having to know the detailed manufacturer-specific registers that also control the function. When a slider is supported, the full slider range of 0-255 shall always be available in all processing modules; however if there is no benefit from having a finer step, this range may map to a smaller number of actual correction steps (e.g., 8 or 16). If the Host knows how to control the function via MSRs then it may use this method instead, however this is outside the scope of this Specification.

### 14.3 Shading Correction

If indicated as present, shading correction shall be available in all image sensor output modes (sub-sampled, binned, digital scaled). The shading correction function shall be enabled by the following register:

**Table 152 Shading Correction Control**

Register Name	Type	RW	Comment
<b>shading_correction_en</b>	8-bit unsigned integer	RW	0: Shading correction disabled 1: Shading correction enabled <b>Default:</b> 0x00

If indicated as present, the level of luminance correction shall be adjustable by a slider with a range to be determined on a case-by-case basis. The default position of this slider shall be 128, corresponding to the luminance at a corner being 10% less than that at the centre. A value of 0 means that shading correction is not used, and a value of 255 means that over correction is done.

**Note:**

*The resolution of the actual correction adjustment may be less than the resolution of the control.*

If shading correction is implemented, it shall automatically follow the setting of register **image\_orientation**. The level of luminance correction can optionally be used to adjust the level of shading correction, depending upon the use cases or desired tuning settings. For example, in daylight conditions typically more correction is used than in dim conditions.

**Table 153 Luminance Correction Control**

Register Name	Type	RW	Comment
<b>luminance_correction_level</b>	8-bit unsigned integer	RW	Range 0–255 <b>Default:</b> 0x80

### 14.4 Green Imbalance Correction

Static green imbalance (an imbalance which can be related to the pixel's physical array position) is assumed to be corrected by lens shading correction. If indicated as present, green imbalance correction shall be available in all image sensor output modes (sub-sampled, binned, digital scaled). The green imbalance correction function shall be enabled by the following register:

**Table 154 Green Imbalance Correction Control**

Register Name	Type	RW	Comment
<b>green_imbalance_filter_en</b>	8-bit unsigned integer	RW	0: Green imbalance correction disabled 1: Green imbalance correction enabled <b>Default:</b> 0x00

## 14.5 Defect Correction

### 14.5.1 Mapped Defect Correction

If indicated as present, mapped defect correction shall be available in all image sensor output modes (sub-sampled, binned, digital scaled). The mapped defect correction function shall be enabled by the following register:

**Table 155 Mapped Defect Correction Control**

Register Name	Type	RW	Comment
<b>mapped_defect_correct_en</b>	8-bit unsigned integer	RW	0: Mapped defect correction disabled 1: Mapped defect correction enabled <b>Default:</b> Image-sensor-specific

### 14.5.2 Dynamic Coupleot Correction

If indicated as present, dynamic couplet correction shall be available in all image sensor output modes (sub-sampled, binned, digital scaled). The dynamic couplet correction function shall be enabled by the following register:

**Table 156 Dynamic Couplet Correction Control**

Register Name	Type	RW	Comment
<b>dynamic_couplet_correct_en</b>	8-bit unsigned integer	RW	0: Dynamic couplet correction disabled 1: Dynamic couplet correction enabled <b>Default:</b> Image-sensor-specific

### 14.5.3 Single Pixel Defect On-the-Fly Correction

If indicated as present, single pixel defect correction shall be available in all image sensor output modes (sub-sampled, binned, digital scaled). The single pixel defect on-the-fly correction function shall be enabled by the following register:

**Table 157 Single Pixel Correction Control**

Register Name	Type	RW	Comment
<b>single_defect_correct_en</b>	8-bit unsigned integer	RW	0: Single pixel defect correction disabled 1: Single pixel defect correction enabled <b>Default:</b> Image-sensor-specific

#### 14.5.4 Combined Dynamic Couple and Single Pixel Defect Correction

If indicated as present, combined dynamic couplet and single pixel defect correction shall be available in all image sensor output modes (sub-sampled, binned, digital scaled). The combined dynamic couplet and single pixel defect correction function shall be enabled by the following register:

**Table 158 Combined Defect Correction Control**

Register Name	Type	RW	Comment
<b>combined_defect_correct_en</b>	8-bit unsigned integer	RW	0: Combined defect correction disabled 1: Combined defect correction enabled <b>Default:</b> Image-sensor-specific

#### 14.5.5 Dynamic Triplet Defect Correction

If indicated as present, dynamic triplet defect correction shall be available in all image sensor output modes (sub-sampled, binned, digital scaled). The dynamic triplet defect correction function shall be enabled by the following register:

**Table 159 Dynamic Triplet Defect Correction Control**

Register Name	Type	RW	Comment
<b>dynamic_triplet_defect_correct_en</b>	8-bit unsigned integer	RW	0: Dynamic triplet defect correction disabled 1: Dynamic triplet defect correction enabled <b>Default:</b> Image-sensor-specific

## 14.6 Noise Filter

The image sensor may support a noise filter. This Specification supports two noise filter types:

- **Separate noise filter** which can filter luma noise and chroma noise separately
- **Combined noise filter** which do not have separate controls for luma and chroma noise.

An image sensor may support either of these types, but not both. If indicated as present, the noise filter shall be available in all image sensor output modes (sub-sampled, binned, digital scaled). The noise filter function shall be enabled by the following register:

**Table 160 Noise Filter Control**

Register Name	Type	RW	Comment
<b>NF_ctrl</b>	8-bit unsigned integer	RW	<p><b>Bit 0</b> 1 = Luma noise filter enabled</p> <p><b>Bit 1</b> 1 = Chroma noise filter enabled</p> <p><b>Bit 2</b> 1 = Combined noise filter enabled</p> <p><b>Default:</b> 0x00</p>

## 14.7 Module Specific Correction

The image sensor may have additional correction blocks which can be controlled by module specific correction control registers. By default, module specific correction is disabled.

**Table 161 Module Specific Correction Control**

Register Name	Type	RW	Comment
<b>module_specific_correction_en</b>	8-bit unsigned integer	RW	<p>0: Correction disabled 1: Correction enabled</p> <p><b>Default:</b> 0</p>

## 14.8 Scene Color Temperature Feedback

Some processing blocks inside an image sensor may require the Host to provide the image sensor with scene color temperature information. The Host can provide this in two forms:

- An integer representation of the color temperature, in degrees Kelvin
- The color channel gains being used by the Host for AWB

The image sensor can use this color temperature information for lens shading correction or other functions. All such functions shall make use of the same register locations. Color temperature capability is independent of the capabilities of other functions.

The image sensor shall indicate whether it requires scene color temperature feedback by a capability register (see **Table 162**). The image sensor supplier shall advise in the product datasheet what kind of feedback information the image sensor needs, if any.

**Table 162 Color Temperature Feedback Capability**

Register Name	Type	RW	Comment
<b>color_feedback_capability</b>	8-bit unsigned integer	RO	<b>Bit 0</b> Module requires Kelvin feedback <b>Bit 1</b> Module requires AWB Gain feedback

If the image sensor indicates that it requires scene color temperature feedback, then the Host should write the values into the following locations, depending on the value of the **color\_feedback\_capability** register:

**Table 163 Color Temperature Feedback Locations**

Register Name	Type	RW	Comment
<b>color_temperature</b>	16-bit unsigned integer	RW	Scene color temperature in Kelvins <b>Default:</b> 6500
<b>absolute_gain_greennr</b>	16-bit unsigned iReal	RW	Calculated Gain for green-red channel <b>Default:</b> Value for 6500 K
<b>absolute_gain_red</b>	16-bit unsigned iReal	RW	Calculated Gain for red channel <b>Default:</b> Value for 6500 K
<b>absolute_gain_blue</b>	16-bit unsigned iReal	RW	Calculated Gain for blue channel <b>Default:</b> Value for 6500 K
<b>absolute_gain_greenb</b>	16-bit unsigned iReal	RW	Calculated Gain for greenb channel <b>Default:</b> Value for 6500 K

Although the 16-bit iReal format has a range of 0-255.996, the range used for absolute Gain feedback should not exceed 0 to 16.0x Gain (i.e. code 0x0000 to 0x1000). The step size is determined by the 16b unsigned iReal format.

The image sensor can assume that all differing per-color-channel gains that the Host uses, up to and including the AWB operation, will be reflected in the gains written to these locations, even if the Gain is actually implemented in the image sensor.

## 14.9 Optical Black Pixel Readout

The image sensor may support readout of optical black (OB) pixel data.

There are two operating modes:

- Reading the OB pixel data out as part of the visible pixel data channel (see *Figure 19* example)
- Reading the OB pixel data out in an interleaved manner (optional)

If the image sensor supports interleaved readout mode, then the data may be read out either (a) by using virtual channel interleaving, or (b) by using data type interleaving that depends upon image sensor capabilities. If the image sensor supports interleaved readout mode, then it should support both virtual channel interleaving and data type interleaving for OB pixel readout.

OB pixel readout is enabled when bit 0 of register **OB\_readout\_ctrl** has the value 1.

If the image sensor supports readout as part of visible pixel data, then the image sensor shall use visible pixel readout when bit 1 of register **OB\_readout\_ctrl** has value the 0. In this use case, the OB pixels shall use the same data type as the visible pixel data, and the data format shall be the same as for the visible pixel data (i.e., if visible pixels use compression, then so shall the OB pixels).

If the image sensor supports virtual channel interleaving, then the image sensor shall use virtual channel interleaving if register **OB\_virtual\_channel** has a different value than register **CSI\_channel\_identifier**, and bit 1 of register **OB\_readout\_ctrl** has the value 1. In this use case, the **OB\_DT** register controls the data type. See *Figure 68* as an example (though OB pixel lines can have different lengths).

If the image sensor supports data type interleaving, then the image sensor shall use data type interleaving if register **OB\_virtual\_channel** has the same value as register **CSI\_channel\_identifier**, and bit 1 of register **OB\_readout\_ctrl** has the value 1. In this use case, register **OB\_DT** controls the data type. See *Figure 67* as an example (though OB pixel lines can have different lengths).

In addition, the image sensor may support a programmable data format in interleaved readout mode. Register **OB\_data\_format** specifies the format for data transfer. For example, if normal visible pixel data use RAW10 format, then it may be sufficient and beneficial to send only the bottom bits of the optical black level pixels, instead of all 10 bits. This is controlled by register **OB\_data\_format**, i.e. that register controls how many of the bottom bits per pixel of the internal OB pixel data are sent out.

When OB pixel readout uses interleaving the image sensor shall not compress the OB pixels, even if the visible pixels do use compression.

If OB pixels are included in the visible pixel data channel, the image sensor may report OB pixels in generic frame format descriptors as a function of bit 0 of register **OB\_readout\_ctrl**.

If OB pixel readout uses interleaving, then the image sensor should not report OB pixels in generic frame format descriptors as a function of bit 0 of register **OB\_readout\_ctrl**.

Analog cropping may impact OB pixels. For example, if a 640x480 image sensor only reads 320x240 visible pixels out from its sensor array, then OB pixels may also be read out from the same columns and/or rows.

If digital crop is used together with interleaved readout, then the digital crop function should not affect the number of OB pixels the image sensor reads out. If digital crop is used together with visible pixel data channel readout, then only OB pixels from same columns and/or rows should be read out.

The readout order of OB pixels may be different from OB pixel physical locations (*Annex B.2.11*). In addition, the readout order of the OB pixel group may differ from the visible pixel readout order. For example, Left OB pixel readout order may be, e.g., either left-to-right or right-to-left, and may be independent from the visible pixel readout order.

3603

**Table 164 Optical Black Readout Control**

Register Name	Type	RW	Comment
<b>OB_readout_ctrl</b>	8-bit unsigned integer	RW	<b>Bit 0</b> 0: Disable OB pixel readout 1: Enable OB pixel readout <b>Bit 1</b> 0: Use visible pixel data channel. 1: Use interleaving <b>Default:</b> 0x00 if bit 1 supports value 0, otherwise 0x02.
<b>OB_virtual_channel</b>	8-bit unsigned integer	RW	<b>Default:</b> 0
<b>OB_DT</b>	8-bit unsigned integer	RW	<b>Default:</b> Image-sensor-specific
<b>OB_data_format</b>	8-bit unsigned integer	RW	<b>Default:</b> Image-sensor-specific

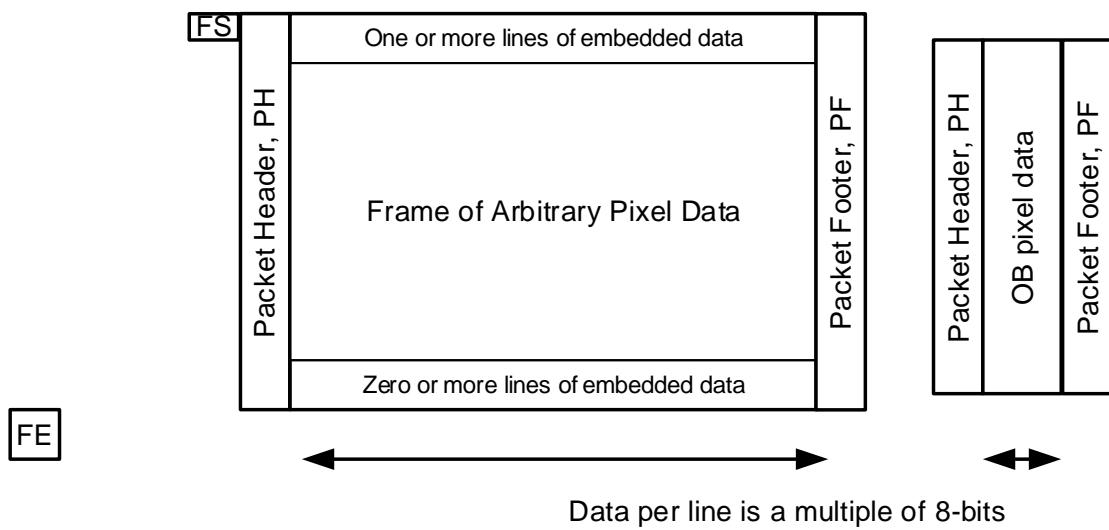
3604

The image sensor capabilities are visible in register **OB\_readout\_capability**.

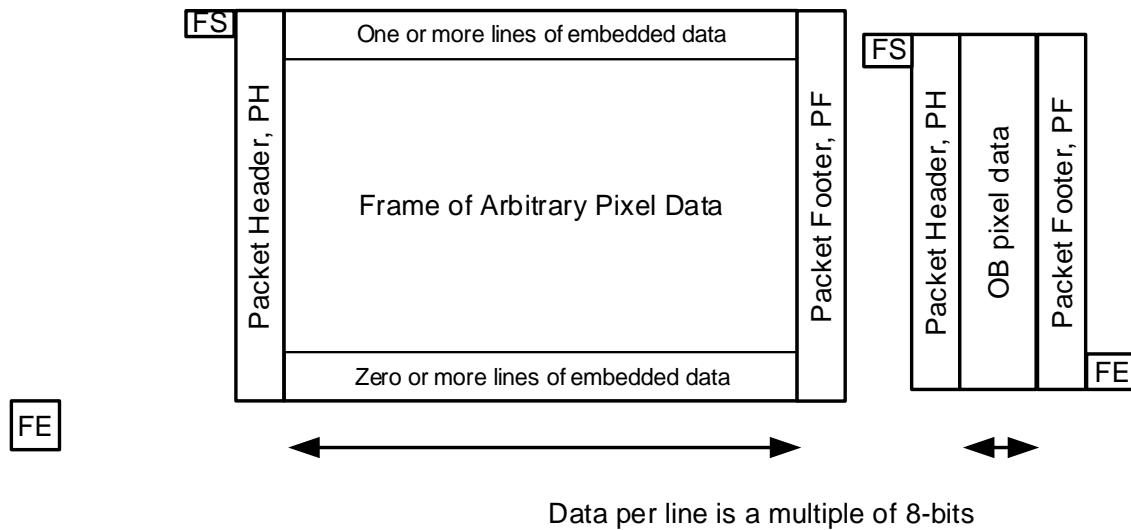
3605

**Table 165 Optical Black Readout Capability**

Register Name	Type	RW	Comment
<b>OB_readout_capability</b>	8-bit unsigned integer	RO	<b>Bit 0</b> 1: Controllable OB pixel readout is supported <b>Bit 1</b> 1: Readout in visible pixel data channel <b>Bit 2</b> 1: Readout in different virtual channel <b>Bit 3</b> 1: Readout with different data type <b>Bit 4</b> 1: Data format for OB pixels in interleaved readout mode is programmable <b>Other Bits</b> Reserved



3606

**Figure 67 Example of Data Type Interleaved OB Pixel Readout**

3607

**Figure 68 Example of Virtual Channel Interleaved OB Pixel Readout**

## 15 NVM Memory Map

### 15.1 Introduction

A camera module is typically calibrated to achieve high image quality, requiring calibration data to be saved in NVM. This Section defines basic rules for the memory map for such NVM.

### 15.2 Usage of NVM Memory Map

For an image sensor with a data transfer interface (see *Section 13*), the Host should access the NVM memory map through the data transfer interface. The first page of the data transfer interface shall always include the first page of the NVM memory map, plus as many additional pages after that as are needed. For example, if the image sensor supports a 1.5kbit NVM, then there will be three pages of 512 bits each.

For a camera module with external EEPROM, that access interface is vendor-specific and beyond the scope of this Specification.

### 15.3 Content of NVM Memory Map

The content of the NVM memory map may consist of two areas: a user area, and a vendor-specific area.

The following rules apply to the content of the NVM memory map:

1. For example, if the image sensor has 4k bits of NVM, with a user area of 3k bits and a vendor-specific area of 1kbits, then the first 3k bits of the NVM shall be the user area, and the latter 1kbits shall be the vendor-specific area.
2. The user area of the NVM shall be accessible starting from the very beginning (i.e. lowest address value) of the NVM memory map. If the image sensor supports a data transfer interface, the NVM memory map shall start at byte 0 of page 0.
3. The content of user area of the NVM memory map is user-specific and beyond the scope of this Specification.
4. The content of the vendor-specific area of the NVM memory map is vendor-specific and beyond the scope of this Specification. For example, the vendor-specific area may contain such items as identification data, or vendor-specific calibration data used by on-module functions.

This page intentionally left blank.

## 16 Timer Functions

### 16.1 Introduction

This Section specifies several timer functions that the image sensor may include. From the perspective of Host software, these timer functions bring several benefits:

- There are no strict timing requirements with respect to Host SW
- Same functionality can be achieved without dependency on Host properties
- The achieved accuracy may be better (or at least can be achieved more easily)

### 16.2 Flash Strobe

The Host can determine whether the image sensor supports flash strobe functionality by inspecting register **flash\_mode\_capability**. All image sensors that can be used as a main camera should support flash strobe.

**Table 166 Flash Strobe Capability Register**

Register Name	Type	RW	Comment
<b>flash_mode_capability</b>	8-bit unsigned integer	RO	<b>Bit 0</b> 1: Single flash strobe supported 0: Single flash strobe not supported <b>Bit 1</b> Reserved

For xenon strobes the accuracy of the flash strobe timer function is very critical as  $1\mu s$  accuracy needs to be achieved with short pulses, but with longer pulses the requirements are not so tight. In some systems, the length of the FSTROBE can be used to control flash output. For this reason, accuracy and range shall be adjusted via register **flash\_strobe\_adjustment** which defines step size for register **tflash\_strobe\_width\_high\_rs\_ctrl** from EXTCLK. Register **flash\_strobe\_adjustment** shall only be programmed during SW Standby Mode.

**Table 167 Flash Strobe Adjustment Register**

Register Name	Type	RW	Comment
<b>flash_strobe_adjustment</b>	8-bit unsigned integer	RW	Register to control pre-divider for <b>flash_strobe_width_high_rs</b> counter <b>Default:</b> Image-sensor-specific

**Table 168** shows the different values that can be achieved by minimum accuracy (**flash\_strobe\_adjustment** = 255) and maximum accuracy (**flash\_strobe\_adjustment** = 1). The maximum value for register **tflash\_strobe\_width\_high\_rs\_ctrl** is 65535.

The actual duration of the flash strobe is determined by the following equations. The Host shall program registers **tflash\_strobe\_width\_high\_rs\_ctrl** and **flash\_strobe\_adjustment** as needed to achieve the required accuracy and minimum and maximum values.

$$\text{Flash_strobe_length_high} = (\text{tFlash_strobe_width_high_rs_ctrl} / \text{EXTCLK frequency}) * \text{flash_strobe_adjustment}$$

3651      **Table 168** shows what kind of flash strobe length can be achieved, depending upon the frequency of EXTCLK  
 3652      and the values of the control registers.

3653      **Table 168 Achievable Flash Strobe Range**

EXTCLK Frequency	Max Value at Min Accuracy	Max Value at Max Accuracy	Accuracy Range	Comment
6 MHz	2785 ms	10 ms	0.16 µs to 43 µs	Min EXTCLK frequency
19.2 MHz	870 ms	3 ms	0.05 µs to 13 µs	Typical EXTCLK frequency
24 MHz	696 ms	2 ms	0.04 µs to 10 µs	Typical EXTCLK frequency
27 MHz	618 ms	2 ms	0.04 µs to 9 µs	Max EXTCLK frequency

3654      Flash strobe can be used in rolling shutter mode (see *Figure 69*). In rolling shutter mode, the flash strobe can  
 3655      be synchronized to the start of any horizontal line, as controlled by register **flash\_strobe\_start\_point**. The  
 3656      reference point is the start of Exposure for that line. The rising edge of flash strobe can be delayed in rolling  
 3657      shutter mode via register **tflash\_strobe\_delay\_rs\_ctrl**, and the width of the high-level pulse is defined by  
 3658      register **tflash\_strobe\_width\_high\_rs\_ctrl**.

3659      **Table 169 Strobe Config Registers**

Register Name	Type	RW	Comment
<b>flash_strobe_start_point</b>	16-bit unsigned integer	RW	Register to select reference point for flash strobe. Adjustable in steps of one line. <b>Range:</b> 0 – last line <b>Default:</b> Image-sensor-specific
<b>tflash_strobe_delay_rs_ctrl</b>	16-bit unsigned integer	RW	1H step (VT clock domain) <b>Range:</b> 0–65535 <b>Default:</b> Image-sensor-specific
<b>tflash_strobe_width_high_rs_ctrl</b>	16-bit unsigned integer	RW	Used to control flash strobe width for high period in rolling shutter mode <b>Range:</b> 1–65535 <b>Default:</b> Image-sensor-specific

3660      In rolling shutter mode, there are three ways to use flash strobe:

1. **Single Flash:** To select single flash strobe mode, the Host sets bit 0 of register **flash\_mode\_rs** to 0. Triggering is handled via bit 0 of register **flash\_trigger\_rs**. Once single trigger mode has been selected, the Host can start triggering by writing 1 to **flash\_trigger\_rs**. If bit 3 of register **flash\_mode\_rs** register is set to synchronous mode, then the image sensor shall re-time the **flash\_trigger\_rs** command to the start of the Exposure. While in single trigger mode, the image sensor shall auto-clear register **flash\_trigger\_rs**.
2. **Single Flash Asynchronous:** A Single Flash strobe as per case 1 can also be used in an asynchronous manner. This is achieved with register **flash\_mode\_rs** (bit 0 = 0 and bit 3 = 1). Triggering is handled via register **flash\_trigger\_rs**, which starts the delay and width timers immediately.
3. **Continuous Triggering:** Continuous triggering can be selected by setting bit 0 of register **flash\_mode\_rs** to 1. Once continuous mode has been selected, the Host can start triggering by writing 1 to register **flash\_trigger\_rs**. If bit 3 of register **flash\_mode\_rs** is set to synchronous mode, then the image sensor shall re-time the **flash\_trigger\_rs** command to the start of the Exposure. The Host can stop triggering by writing 0 to register **flash\_trigger\_rs**. While in continuous mode, the image sensor shall not auto-clear register **flash\_trigger\_rs**. The image

3677 sensor shall end continuous mode either immediately (if bit 1 of register **flash\_mode\_rs** has value  
 3678 1), or after a complete strobe sequence (if bit 1 of register **flash\_mode\_rs** has value 0).

3679 The Host shall not set Continuous Asynchronous mode (i.e. register **flash\_mode\_rs** bit 0 = 1 and bit 3 = 1),  
 3680 because its effect might be undefined.

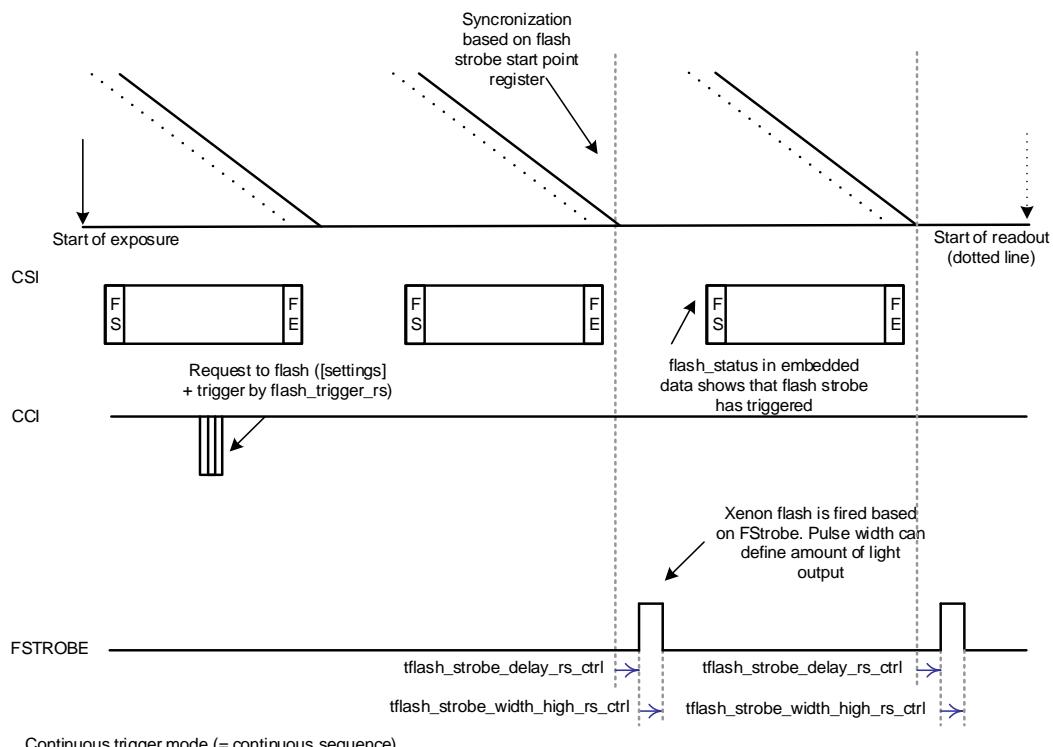
3681 **Table 170 Flash Strobe Control Register**

Register Name	Type	RW	Comment
<b>flash_mode_rs</b>	8-bit unsigned integer	RW	<b>Bit 0: Flash Strobe Triggering Mode</b> 0: Single Flash 1: Continuous Triggering <b>Bit 1: Flash Strobe Stop Mode</b> 0: Complete strobe 1: Truncated strobe <b>Bit 2</b> Reserved <b>Bit 3: Flash Strobe Synch Mode</b> 0: Synchronous 1: Asynchronous <b>Bits 4 &amp; 5</b> Reserved <b>Default:</b> 0x00
<b>flash_trigger_rs</b>	8-bit unsigned integer	RW	<b>Bit 0</b> 1: Trigger flash (auto clear in single mode) 0: Disable strobe generation <b>Default:</b> 0x00

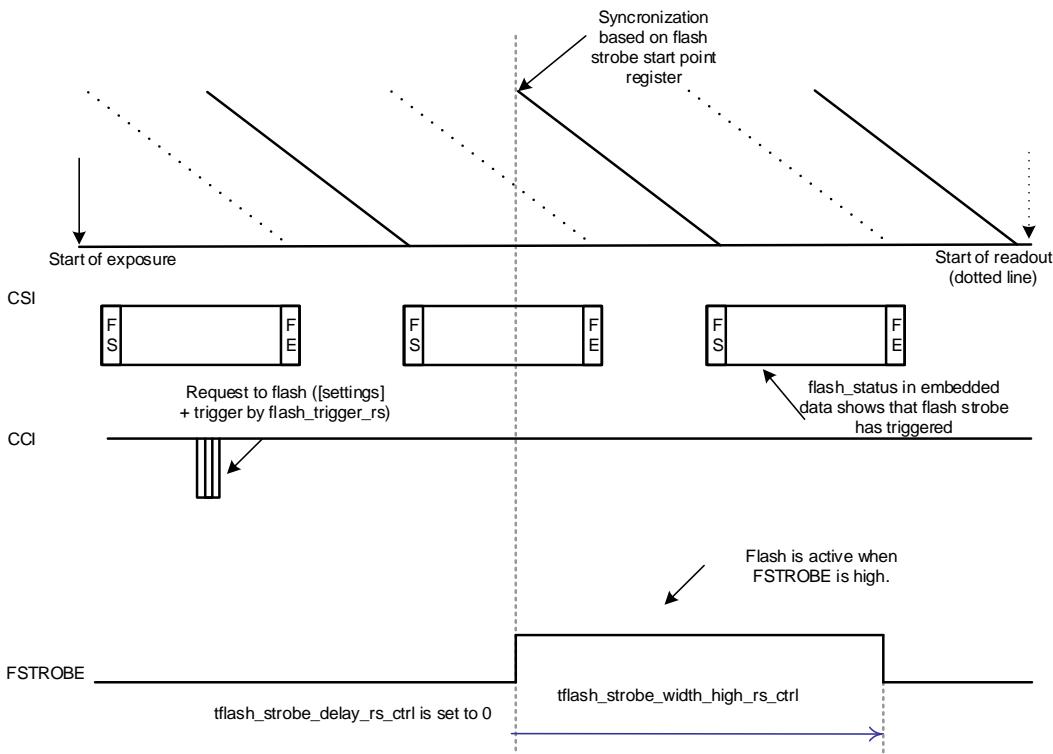
3682 It shall be possible to read the register **flash\_status** from the embedded data lines, so that the Host can detect  
 3683 whether the **flash\_trigger\_rs** command has been retimed, and if so, to which frame. Based on this  
 3684 information and the strobe config registers, the Host can determine which frame receives the proper  
 3685 Exposure.

3686 **Table 171 Flash Status Register**

Register Name	Type	RW	Comment
<b>flash_status</b>	8-bit unsigned integer	RO dynamic	<b>Bit 0</b> 0: Flash strobe is not retimed to this frame 1: Flash strobe is retimed to this frame <b>Bit 1</b> Reserved



3687

**Figure 69 Example of Short Flash Strobe Pulse in Rolling Shutter Use Case**

3688

**Figure 70 Example of Long Flash Strobe Pulse in Rolling Shutter Use Case**

### 16.3 Special Actuator Strobe

The Host can determine whether the image sensor supports special actuator (SA) strobe functionality by inspecting register **SA\_strobe\_mode\_capability**.

**Table 172 SA\_strobe Capability Register**

Register Name	Type	RW	Comment
<b>SA_strobe_mode_capability</b>	8-bit unsigned integer	RO	<b>Bit 0</b> 0: Fixed width sa_strobe is not supported 1: Fixed width sa_strobe is supported <b>Bit 1</b> 0: sa_strobe edge control is not supported 1: sa_strobe edge control is supported

There are three different triggering modes for SA\_strobe:

1. **Single Strobe:** To select single strobe mode, the Host sets bit 0 of register **SA\_strobe\_mode** to 0. Triggering is handled via bit 0 of register **SA\_strobe\_trigger**. Once single trigger mode has been selected, the Host can start triggering by writing 1 to **SA\_strobe\_trigger**. If bit 3 of register **SA\_strobe\_mode** is set to synchronous mode, then the image sensor shall re-time the **SA\_strobe\_trigger** command to the start of the Exposure. While in single strobe mode, the image sensor shall auto-clear register **SA\_strobe\_trigger**.
2. **Single Strobe Asynchronous:** A Single Strobe as per case 1 can also be used in an asynchronous manner. This is achieved with register **SA\_strobe\_mode** (bit 0 = 0 and bit 3 = 1). Triggering is handled via register **SA\_strobe\_trigger**, which starts the triggering.
3. **Continuous Triggering:** Continuous triggering can be selected by setting bit 0 of register **SA\_strobe\_mode** to 1. Once continuous mode has been selected, the Host can start triggering by writing 1 to register **SA\_strobe\_trigger**. If bit 3 of register **SA\_strobe\_mode** is set to synchronous mode, then the image sensor shall re-time the **SA\_strobe\_trigger** command to the start of the Exposure. The Host can stop triggering by writing 0 to register **SA\_strobe\_trigger**. While in continuous mode, the image sensor shall not auto-clear register **SA\_strobe\_trigger**. The image sensor shall end continuous mode either immediately (if bit 1 of register **SA\_strobe\_mode** has value 1), or after a complete strobe sequence (if bit 1 of register **SA\_strobe\_mode** has value 0).

Host shall not set continuous asynchronous mode (i.e. register **SA\_strobe\_mode** bit 0 = 1 and bit 3 = 1), because its effect might be undefined.

The three SA strobe triggering modes can operate in either of two pulse width modes. The Host selects one of these modes via bit 4 of register **SA\_strobe\_mode**:

- **Fixed Width Mode:** The rising edge of SA\_strobe is controlled by register **tSA\_strobe\_delay\_ctrl**, and the SA strobe pulse width is fixed to 1  $\mu$ s.
- **Adjustable Edge Mode:** Pulse width is controlled by setting two independent delay periods: (1) delay between the trigger command and the SA strobe rising edge (register **tSA\_strobe\_re\_delay\_ctrl**), and (2) delay between the trigger command and the SA strobe falling edge (**tSA\_strobe\_fe\_delay\_ctrl**). For Adjustable Edge Mode the SA strobe pulse width is given by the formula:

$$\text{SA strobe pulse width} = \text{tSA_strobe_fe_delay_ctrl} - \text{tSA_strobe_re_delay_ctrl}$$

3723

**Table 173 Special Actuator Strobe Config Register**

Register Name	Type	RW	Comment
<b>SA_strobe_mode</b>	8-bit unsigned integer	RW	<p><b>Bit 0: SA Strobe Triggering Mode</b>            0: Single Strobe            1: Continuous Triggering</p> <p><b>Bit 1: SA Strobe Stop Mode</b>            0: Complete strobe            1: Truncated strobe mode</p> <p><b>Bit 2</b>            Reserved</p> <p><b>Bit 3: SA Strobe Synch Mode</b>            0: Synchronous            1: Asynchronous</p> <p><b>Bit 4: SA Strobe Pulse Width Mode</b>            0: Fixed Width            1: Adjustable Edge</p> <p><b>Default:</b> 0x00</p>
<b>SA_strobe_start_point</b>	16-bit unsigned integer	RW	Register to select reference point for special actuator strobe. Adjustable in one line steps. Range 0 – last line <p><b>Default:</b> Image-sensor-specific</p>
<b>tSA_strobe_delay_ctrl</b>	16-bit unsigned integer	RW	1H step (in VT domain) <p><b>Range:</b> 0–65535</p> <p><b>Default:</b> Image-sensor-specific</p>
<b>tSA_strobe_width_ctrl</b>	16-bit unsigned integer	RO	Strobe width is fixed to 1us. It cannot be adjusted. Thus this register is reserved for future use.
<b>tSA_strobe_re_delay_ctrl</b>	16-bit unsigned integer	RW	1H step (in VT domain) <p><b>Range:</b> 0–65535</p> <p><b>Default:</b> Image-sensor-specific</p>
<b>tSA_strobe_fe_delay_ctrl</b>	16-bit unsigned integer	RW	1H step (in VT domain) <p><b>Range:</b> 0–65535</p> <p><b>Default:</b> Image-sensor-specific</p>
<b>SA_strobe_trigger</b>	8-bit unsigned integer	RW	<p><b>Bit 0</b>            1: Trigger strobe (auto clear in single mode)            0: Disable strobe generation.</p> <p><b>Default:</b> 0x00</p>

3724

**Table 174 Special Actuator Strobe Status Register**

Register Name	Type	RW	Comment
<b>SA_strobe_status</b>	8-bit unsigned integer	RO dynamic	<p><b>Bit 0</b>            0: Special actuator strobe is not retimed to this frame            1: Special actuator strobe is retimed to this frame</p>

## 17 PDAF

### 17.1 Introduction

When an image sensor supports PDAF functionality, it typically means that the image sensor supports an N x M sized PDAF data grid, where N and M are vendor-specific. In the image sensor datasheet, the supplier shall describe the location of PDAF pixels in the pixel array, so that the relationship between the PDAF and the camera's field-of-view (FOV) can be understood. The Host can detect whether the image sensor supports PDAF functionality by inspecting bit 0 of register **PDAF\_capability**.

The image sensor should support at least one of the following PDAF readout modes: bottom embedded data readout, interleaved readout. The image sensor may support PDAF by having PDAF pixels only within the visible pixel area.

When PDAF data is read out of the image sensor in an interleaved manner, in practice the PDAF data is read out after each relevant image line. By contrast, with bottom-embedded data readout the PDAF data is read out at the end of the frame.

The image sensor may support some amount of processing for the read-out PDAF data. The image sensor should always support the readout of RAW PDAF data, even if processed PDAF readout is also supported. The image sensor should support interleaved RAW PDAF readout. The image sensor should not support bottom-embedded RAW PDAF data readout.

The image sensor may support visible pixel area PDAF pixel-specific defect correction. If bit 5 of register **PDAF\_capability** has the value 1, then the Host can enable or disable visible pixel area PDAF pixel-specific defect correction via bit 3 of register **PDAF\_ctrl** (even if PDAF readout is disabled via bit 0 of register **PDAF\_ctrl**). If the Host wants to receive unprocessed PDAF pixels within the visible pixel area data, then the Host must turn off additional standard sensor corrections (for example, dynamic defect correction) as specified in *Section 14.5*.

Some image sensors supporting interleaved or bottom-embedded data PDAF readout might support configurable PDAF ROI. If the value of bit 0 of register **PDAF\_capability\_2** is 1, then the Host can program the PDAF ROI via registers **pdaf\_x\_addr\_start**, **pdaf\_y\_addr\_start**, **pdaf\_x\_addr\_end**, and **pdaf\_y\_addr\_end**. If the configured PDAF ROI is smaller than the FOV before PDAF, then the image sensor may do some amount of processing only on the PDAF-ROI-specific PDAF data, and only the PDAF-ROI-specific PDAF data shall be read out.

The Host can control PDAF readout as follows:

- PDAF data readout is enabled via bit 0 of register **PDAF\_ctrl**. If at least one PDAF readout mode is supported (i.e. one or more of bits 1 - 4 in register **PDAF\_capability** has the value 1), then the Host can enable PDAF readout.
- PDAF readout mode is controlled by bit 2 in register **PDAF\_ctrl**. The PDAF data readout modes are: bottom-embedded (see *Section 17.3*), and interleaved (see *Section 17.4*).
- PDAF processing for different readout modes is controlled by bit 1 in register **PDAF\_ctrl**. If bit 1 or bit 2 in register **PDAF\_capability** have the value 1, then the Host can select processed PDAF readout mode. If bit 3 or bit 4 in register **PDAF\_capability** have the value 1, then the Host can select RAW PDAF readout mode.

An image sensor capable of outputting PDAF pixels only as part of the visible pixel area shall set bit 0 in register **PDAF\_capability** to the value 1, and shall set bits 1 through 4 to 0 in order to indicate that none of the PDAF readout modes are supported.

3765

**Table 175 PDAF Registers**

Register Name	Type	RW	Comment
<b>PDAF_capability</b>	8-bit unsigned integer	RO	<p><b>Bit 0</b> 1: PDAF is supported</p> <p><b>Bit 1</b> 1: Processed PDAF data readout in bottom embedded data</p> <p><b>Bit 2</b> 1: Processed PDAF data readout is interleaved</p> <p><b>Bit 3</b> 1: RAW PDAF data readout in bottom embedded data</p> <p><b>Bit 4</b> 1: RAW PDAF data readout is interleaved</p> <p><b>Bit 5</b> 1: Visible pixel area PDAF pixel specific defect correction is supported</p> <p><b>Bit 6</b> 1: Virtual channel interleaving is supported</p> <p><b>Bit 7</b> 1: Data type interleaving is supported</p>
<b>PDAF_capability_2</b>	8-bit unsigned integer	RO	<p><b>Bit 0</b> 1: PDAF ROI is supported</p> <p><b>Bit 1</b> 0: Separation before digital crop 1: Separation after digital crop</p> <p><b>Bit 2</b> 0: <b>PDAF_ctrl</b> is not retimed 1: <b>PDAF_ctrl</b> is retimed</p> <p><b>Other Bits</b> Reserved for future use</p>
<b>PDAF_ctrl</b>	8-bit unsigned integer	RW	<p><b>Bit 0</b> 1: Enable PDAF readout 0: Disable PDAF readout</p> <p><b>Bit 1</b> 1: Processed PDAF data 0: RAW PDAF data</p> <p><b>Bit 2</b> 1: Interleaved readout 0: Bottom embedded data readout</p> <p><b>Bit 3</b> 1: PDAF defect correction on for visible pixel area 0: PDAF defect correction off for visible pixel area</p> <p><b>Bit 4</b> Reserved</p> <p><b>Default:</b> Bit 0 = 0</p>
<b>PDAF_VC</b>	8-bit unsigned integer	RW	Virtual channel for PDAF interleaved readout <b>Default:</b> Image-sensor-specific

Register Name	Type	RW	Comment
<b>PDAF_DT</b>	8-bit unsigned integer	RW	Data type for PDAF interleaved readout <b>Default:</b> Image-sensor-specific
<b>pdaf_x_addr_start</b>	16-bit unsigned integer	RW	PDAF ROI top left corner X-coordinate <b>Values:</b> Even numbers only: 0, 2, 4, 6... <b>Units:</b> Pixels <b>Default:</b> Image-sensor-specific
<b>pdaf_y_addr_start</b>	16-bit unsigned integer	RW	PDAF ROI top left corner Y-coordinate <b>Values:</b> Even numbers only: 0, 2, 4, 6... <b>Units:</b> Lines <b>Default:</b> Image-sensor-specific
<b>pdaf_x_addr_end</b>	16-bit unsigned integer	RW	PDAF ROI bottom right corner X-address <b>Values:</b> Odd numbers only: 1, 3, 5, 7... <b>Units:</b> Pixels <b>Default:</b> Image-sensor-specific
<b>pdaf_y_addr_end</b>	16-bit unsigned integer	RW	PDAF ROI bottom right corner Y-address <b>Values:</b> Odd numbers only: 1, 3, 5, 7... <b>Units:</b> Lines <b>Default:</b> Image-sensor-specific

## 17.2 Impact of PDAF Function on Other CCS Features

3766

This Section describes the impact of the PDAF function upon other CCS features.

### 17.2.1 PDAF Pixel Readout Limited to Visible Pixel Channel

3767  
3768  
3769

An image sensor supporting PDAF pixel readout only as part of the visible pixel data is not capable of separating the PDAF pixels from the visible pixels, and therefore also not able to read out PDAF-specific data separately.

3770  
3771  
3772  
3773

When controlling such image sensors, Hosts should take into account that some image array pixels are PDAF pixels. For example, sensor processing (e.g., dynamic defect correction) and image scaling (if supported by the image sensor) might be applied to the PDAF pixels as well as the visible pixels, and as a result such sensor processing might be less usable.

#### 17.2.1.1 Binning and Subsampling

3774  
3775

Binning may have an impact upon PDAF pixels. For example, PDAF pixels could be combined with non-PDAF pixels.

3776  
3777  
3778

Subsampling may have an impact upon PDAF pixels. For example, some PDAF pixels could be skipped. The image sensor datasheet should describe any limitations regarding PDAF pixel usage with the supported binning and sub-sampling modes.

#### 17.2.1.2 Cropping and Scaling

3779  
3780  
3781  
3782  
3783

Image sensors that have only PDAF pixels should not support digital scaling. Scaling might have an impact upon PDAF pixels similar to the effect that binning has.  
Analog ROI functionality (which is controlled by registers **x\_addr\_start**, **y\_addr\_start**, **x\_addr\_end**, and **y\_addr\_end**) and digital crop functionality (which is controlled by registers **digital\_crop\_image\_width**, **digital\_crop\_image\_height**, **digital\_crop\_x\_offset**, and **digital\_crop\_y\_offset**) do not have any specific

3784 dependency on PDAF pixels. However, when setting image sensor output resolutions the Host should take  
3785 care that the desired PDAF pixels will be included in the image sensor readout the Host will receive.

### 17.2.1.3 Compression

3786 Because PDAF pixels are not separated from the visible pixels, compression will have an impact upon the  
3787 PDAF pixels.

### 17.2.2 Interleaved or Bottom Embedded Data Readout

3788 Image sensors supporting at least one of the PDAF readout modes (interleaved readout and/or bottom-  
3789 embedded data readout) may separate the PDAF pixels out from the visible pixels, in order to implement a  
3790 separate PDAF-specific readout. Doing so has the advantage of reducing the dependencies between PDAF  
3791 and other image sensor features.

### 17.2.2.1 Binning and Subsampling

3792 Binning may have an impact upon PDAF pixels. For example, PDAF pixels could be combined with non-  
3793 PDAF pixels.

3794 Subsampling may have an impact upon PDAF pixels. For example, some PDAF pixels could be skipped.  
3795 The image sensor datasheet should describe any limitations regarding PDAF pixel usage with the supported  
3796 binning and sub-sampling modes.

### 17.2.2.2 Cropping and Scaling

3797 Image sensors that support separation of PDAF pixel data from the visible pixels shall support exactly one  
3798 of two possible PDAF separation options:

- 3799 1. PDAF pixels are separated after analog crop and before digital crop (if digital crop is supported)
- 3800 2. PDAF pixels are separated after analog crop and after digital crop (if digital crop is supported)

3801 Image sensors should support option 1.

3802 The Host can determine which option is supported by inspecting bit 1 of register **PDAF\_capability\_2**. This  
3803 bit is valid for image sensors that support interleaved PDAF readout or bottom embedded data PDAF readout.  
3804 An image sensor does not support digital crop shall report the value 0 in bit 1 of register **PDAF\_capability\_2**.

3805 Analog ROI and digital crop functionality do not have any specific dependency on PDAF pixels. However,  
3806 when setting image sensor output resolutions the Host should take care that the desired PDAF pixels will be  
3807 included in the image sensor readout the Host will receive, taking into account the supported PDAF  
3808 separation option.

3809 Digital scaling shall not have an impact upon PDAF pixels.

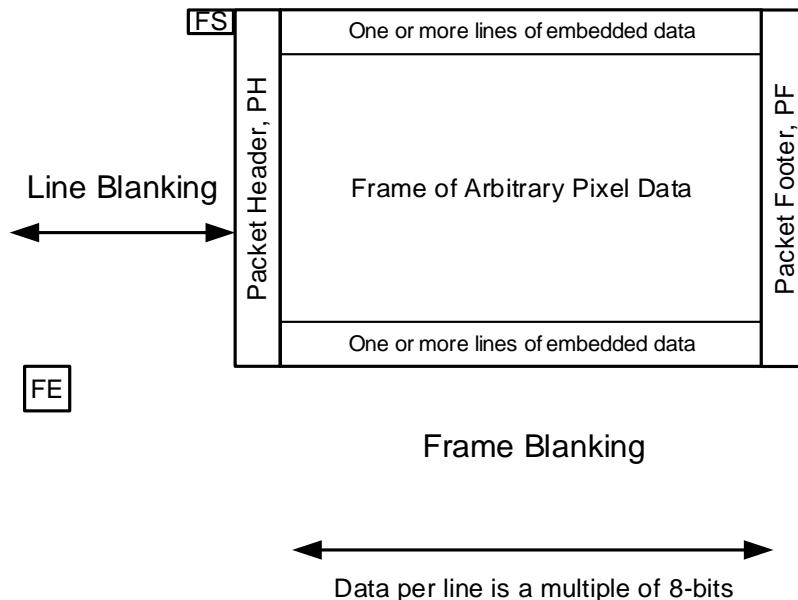
### 17.2.2.3 Compression

3810 Compression shall not have an impact upon PDAF pixels, as PDAF separation occurs before compression.

### 17.3 Bottom Embedded PDAF Readout

If the image sensor supports PDAF data readout with bottom-embedded data, and if bit 2 of register **PDAF\_ctrl** has the value 0, then the image sensor shall output PDAF data in bottom-embedded data.

When PDAF data is transferred within bottom-embedded data, the frame is transferred as shown in *Figure 71* and as described below.



**Figure 71 PDAF Data as Bottom Embedded Data**

Top embedded data and bottom embedded data should use different data types, so that the receiver can easily distinguish between top-embedded data and bottom-embedded data. However because the bottom-embedded data type should be programmable, it ought to possible to use the same data type in both top-embedded data and bottom-embedded data (i.e. DT = 0x12). If the image sensor supports programmable data type for bottom-embedded data, then the bottom-embedded data shall be programmable by register **bottom\_embedded\_data\_DT**. See *Section 7.5* for additional details.

### 17.4 Interleaved PDAF Readout

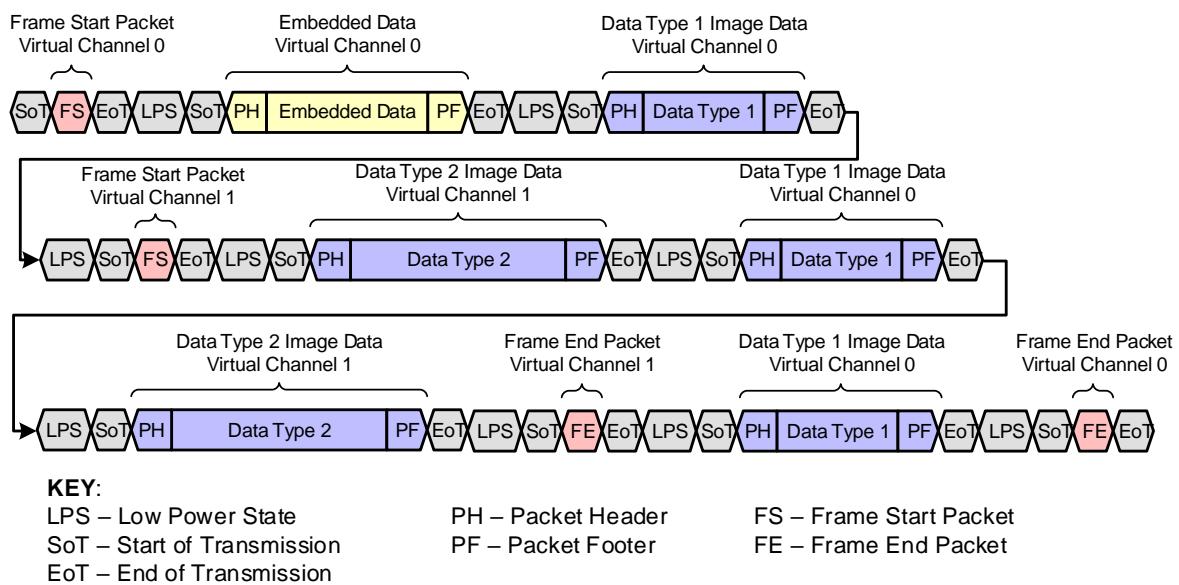
If the image sensor supports interleaved PDAF data readout, then it may support virtual channel interleaving and/or data type interleaving; it should support both.

#### 17.4.1 Virtual Channel PDAF Interleaving

If the image sensor supports virtual channel PDAF interleaving, then it shall use virtual channel interleaving if register **PDAF\_VC** contains a different value than register **CSI\_channel\_identifier**, and if bit 2 of register **PDAF\_ctrl** contains the value 1.

When PDAF data is transferred via virtual channel interleaving, the PDAF frame is transferred on a different virtual channel than the visible pixels frame, as shown in *Figure 72* and as described below.

The main virtual channel (for example, Channel 0) carries the top-embedded data and the Image Data, using a data type. A separate virtual channel carries the PDAF data (Data type 2 Image Data, Virtual channel 1 in the Figure). The Host can control the PDAF data type via register **PDAF\_DT**.



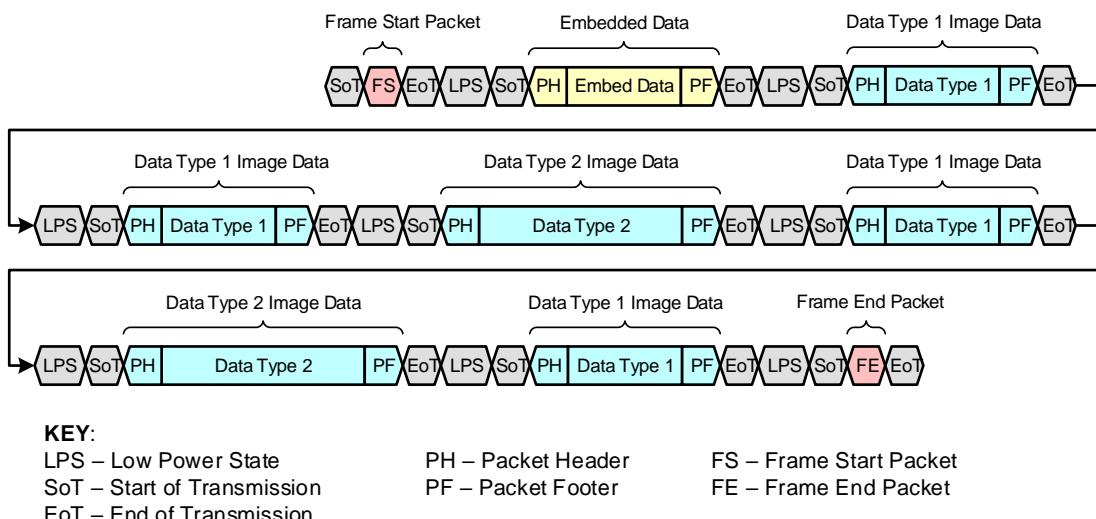
3832

**Figure 72 PDAF Data in Different Virtual Channel****17.4.2 Data Type PDAF Interleaving**

3833 If the image sensor supports data type PDAF interleaving, then it shall use data type interleaving if register **PDAF\_VC** contains the same value as register **CSI\_channel\_identifier**, and if bit 2 of register **PDAF\_ctrl** contains the value 1. The Host can control the PDAF data type via register **PDAF\_DT**.

3834

3835



3836

**Figure 73 PDAF Data with Different Data Type**

## 18 Temperature Sensor

The image sensor may support an on-module temperature sensor. The Host can determine whether the image sensor supports a temperature sensor by inspecting bit 0 of register **temp\_sensor\_capability**.

If the image sensor does support a temperature sensor, then the Host can determine by inspecting register **temp\_sensor\_capability** whether the temperature sensor is a legacy solution (bit 1 has value 0), or conforms to the interface given below in this Section (bit 1 has value 1).

A compliant temperature sensor shall have following properties:

- Measurement range is from  $-20^{\circ}\text{ C}$  to  $+120^{\circ}\text{ C}$ , unless otherwise specified in the image sensor datasheet.
- Accuracy target is  $\pm 5^{\circ}$  over an operating range of  $+20^{\circ}\text{ C}$  to  $+60^{\circ}\text{ C}$ , unless otherwise specified in the image sensor datasheet.
- Calibration shall be used if the accuracy is not otherwise achieved. The image sensor shall automatically adjust temperature sensor operation based on internal calibration information.
- The temperature sensor output values should reflect the actual temperature of the pixel area, so that pixel performance can be estimated as a function of temperature.
- The temperature sensor should update its output at least once per frame while in Streaming Mode. In SW Standby mode the temperature sensor output shall update at least once for each entry into SW Standby mode.
- The temperature sensor shall output valid values both in SW Standby mode and in Streaming mode.
- The temperature sensor output should have a valid value within 15ms after being enabled in SW Standby mode, or within one frame period after being enabled in Streaming mode.
- The temperature sensor output shall be nominally zero for  $0^{\circ}\text{ C}$ , and shall increase or decrease by a nominal 1 LSB for each degree C change in temperature. Where an image sensor needs calibration to achieve this, such calibration shall occur internally to the image sensor and shall be invisible to the Host. The range of values supported by the “MIPI CCS output format” referred to in **Table 176** is nominally between -128 and +127 (i.e., 8 bits, 2’s complement).

The Host can enable and disable the temperature via bit 0 of register **temp\_sensor\_ctrl**. If the image sensor uses the temperature sensor data internally for calibration or adjustment, then such use shall not be affected by the setting of register **temp\_sensor\_ctrl**.

3866

**Table 176 Temperature Sensor Registers**

Register Name	Type	RW	Comment
<b>temp_sensor_ctrl</b>	8-bit unsigned integer	RW	<b>Bit 0</b> 0: Disable 1: Enable <b>Default:</b> 0
<b>temp_sensor_mode</b>	8-bit unsigned integer	RO	Reserved for future use
<b>temp_sensor_output</b>	8-bit signed integer	RO dynamic	Temperature sensor output
<b>temp_sensor_capability</b>	8-bit unsigned integer	RO	<b>Bit 0</b> 0: Temperature sensor not supported 1: Temperature sensor is supported <b>Bit 1</b> 0: Custom output format 1: MIPI CCS output format <b>Bit 2</b> 0: <b>temp_sensor_output</b> reset value is undefined 1: <b>temp_sensor_output</b> is set to 0x80 whenever the temperature sensor output is invalid, and also once following each time the Host reads a valid temperature value.

3867      **Table 177** shows example values for register **temp\_sensor\_output**, with corresponding temperatures in  
 3868      degrees C. Although **Table 177** shows allowed capping, the image sensor should report temperature sensor  
 3869      output values without capping for the full temperature measurement range (i.e., between the minimum and  
 3870      maximum values).

3871

**Table 177 Temperature Sensor Output Format**

<b>Output Value</b>	<b>Temperature</b>
0x81 – 0xEC	-20°C
0xED	-19°C
–	–
0x00	0°C
–	–
0x50	80°C
–	–
0x77	119°C
0x78 – 0x7F	120°C

3872

A possible operation sequence for a temperature sensor would be:

3873

1. Start up the camera module, i.e. camera module enters SW Standby mode
2. The Host reads the MIPI CCS version, the camera module ID, and other version information
3. The Host writes the EXTCLK frequency info to register **extclk\_frequency\_mhz**
4. The Host reads register **temp\_sensor\_capability** to detect whether a CCS-compliant temperature sensor is supported
5. The Host enables the temperature sensor by writing 1 to register **temp\_sensor\_ctrl**
6. The Host may send new settings to the image sensor, or may read NVM content
7. The Host reads the temperature sensor output from register **temp\_sensor\_output**
8. The Host sends ‘start streaming’ command, then the image sensor starts streaming.
9. The temperature sensor output is then updated at least once per frame

3874

The Host may read the temperature sensor output during streaming via CCI (register **temp\_sensor\_output**), or it may monitor temperature output values via embedded data. If bit 2 of **temp\_sensor\_capability** is 1, then a value of 0x80 (-128) in register **temp\_sensor\_output** indicates that the temperature sensor output is invalid (e.g., because the temperature sensor is disabled, or because the first reading isn’t ready yet), or that the temperature hasn’t been updated since the last time it was read by the Host.

3875

3876

3877

3878

3879

3880

3881

3882

3883

3884

3885

3886

3887

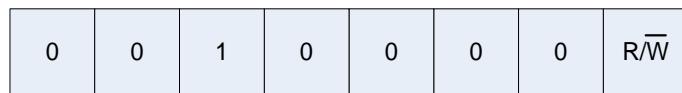
This page intentionally left blank.

## 19 Camera Control Interface (CCI)

The image sensor control interface shall in all respects comply either with the I<sup>2</sup>C-based or I3C-based CCI Specification described in [\[MIPI12\]](#), with the following constraints:

- The CCI variant to be used shall be 7-bit Target address, 16-bit index, and 8-bit data.
- Where parameters written on the CCI bus are described as ‘re-timed’, this shall only mean that their use as settings and their inclusion in the embedded data is re-timed. If these parameters are read from the CCI, then it shall return the data most recently written. Where bits in the registers are unused, a ‘hard-wired’ value of zero shall be returned, e.g. if the value written is 0xF5 but the 4 MSBs are unused, then the device shall return 0x05.

If the image sensor supports I<sup>2</sup>C-based CCI, then it should use the default image sensor 7-bit CCI Target address shown in [Figure 74](#) (i.e., 7'h10). If the image sensor supports I3C-based CCI, then it should use a Static Address (as the term ‘Static Address’ is defined and used in [\[MIPI11\]](#) for the purpose of facilitating dynamic address assignment), using the same address value (i.e., 7'h10).



**Figure 74 Camera Sensor 7-Bit Target Address and R/~W Control Bit**

### 19.1 Independent and Simultaneous Camera Usage Support with I<sup>2</sup>C-Based CCI

In order to allow more than one active MIPI CCS compatible image sensor to be used on a single CCI bus, an image sensor may support the following functionality:

- A software-switchable CCI address, if bit 0 of register **CCI\_address\_ctrl\_capability** has the value 1
- An alternate CCI address, if bit 1 of register **CCI\_address\_ctrl\_capability** has the value 1
- A software-switchable alternate CCI address, if bit 2 of register **CCI\_address\_ctrl\_capability** has the value 1

The software-switchable CCI address may also be used to solve device address clashes with other devices sharing the same bus.

3910

**Table 178 CCI Address Control**

Register Name	Type	RW	Comment
<b>CCI_address_ctrl_capability</b>	8-bit unsigned integer	RO	<p><b>Bit 0</b> 1: Software-switchable CCI address is supported 0: Not supported</p> <p><b>Bit 1</b> 1: Alternate CCI address is supported 0: Not supported</p> <p><b>Bit 2</b> 1: Software-switchable alternate CCI address is supported 0: Not supported</p> <p><b>Bits 3-7</b> Reserved for future use</p>
<b>CCI_address_ctrl</b>	8-bit unsigned integer	RW	<p>7-bit address for image sensor. 7-bit addresses shall be written MSB-aligned, with the LSB being set to 0 (e.g., address 7'h10 is written as 8'h20). The image sensor shall not set restrictions on usable addresses. Typical register values are: 0x20, 0x6C, 0x6E.</p> <p><b>Default:</b> If supported, default value shall be the nominal CCI address</p>
<b>2nd_CCI_address_ctrl</b>	8-bit unsigned integer	RW	<p>7-bit address for alternate CCI interface of image sensor. 7-bit addresses shall be written MSB-aligned, with the LSB being set to 0 (e.g., address 7'h10 is written as 8'h20). Image sensor shall not set restrictions on usable addresses. Typical register values are: 0x20, 0x6C, 0x6E.</p>
<b>2nd_CCI_if_ctrl</b>	8-bit unsigned integer	RW	<p><b>Bit 0</b> 1: Enable alternate CCI interface 0: Disable alternate CCI interface</p> <p><b>Bit 1</b> 1: Enable ACK sending for alternate CCI interface 0: Disable ACK sending for alternate CCI interface</p> <p><b>Default:</b> 0x00</p>

### 19.1.1 Software-Switchable CCI Address

An image sensor that supports a Software-switchable CCI address shall comply with the following two-camera example use cases.

#### Assumptions

- Both image sensors share the same CCI bus.
- Both image sensors share the same power supplies.
- The image sensors have separate XSHUTDOWN signals.

#### Sequence

1. Both image sensors are powered up.
2. EXTCLK clock is provided to both of the image sensors (by individual or shared clock line)
3. Only one image sensor is activated, by driving its XSHUTDOWN to 1 (following the MIPI CCS power-up sequence).
4. The Host verifies that the CCI address of the active image sensor can be changed, by reading register **CCI\_address\_ctrl\_capability**.
5. The Host writes a new CCI address to register **CCI\_address\_ctrl**.
6. The image sensor shall take the new address into use immediately (having completed the preceding CCI transaction correctly, e.g. with a valid ACK).
7. The Host can read register **CCI\_address\_ctrl** to verify that the new address has been taken into use.
8. Another image sensor can now be activated and operated with the default CCI address.

If a camera modules includes other identical devices (e.g., a separate lens driver), then the Host may also change the lens driver's CCI address, providing that the lens driver supports a changeable CCI address.

### 19.1.2 Alternate CCI Address

An image sensor may support an alternate CCI interface, in order to allow the Host to simultaneously send a given command or collection of settings to multiple image sensors.

An image sensor that supports the alternate CCI interface shall have following properties:

1. The alternate CCI interface shall have a CCI address that is different from the primary CCI address. The primary CCI interface should have a programmable CCI address (register **CCI\_address\_ctrl**)
2. The alternate CCI interface should have a software-programmable CCI address (register **2nd\_CCI\_address\_ctrl**)
3. The alternate CCI interface shall support control over ACK sending (bit 1 in register **2nd\_CCI\_if\_ctrl**)
  - When ACK sending is disabled and the alternate CCI interface is enabled, the image sensor shall use that interface for receiving commands from the CCI bus, so that the image sensor is not externally visible to the bus.
  - When ACK sending is enabled and the alternate CCI interface is enabled, the image sensor shall use that interface normally.

To use the image sensor's alternate CCI interface, the Host:

- Shall ensure that the alternate CCI interface, in devices wanting to receive command or settings for the alternate CCI interface, shall have the same CCI address on that interface.
- Shall ensure that at least one of those devices will send ACKs when receiving CCI messages for that CCI address.
- Shall use the alternate CCI interface only for writing data to the devices, and shall not read data from them.
- May use the alternate CCI interface for purposes such as simultaneously sending the same configuration data to multiple image sensors, or simultaneously sending the **grouped\_parameter\_hold** command to multiple image sensors in order to put (for example) image-sensor-specific Exposure parameters into use at the same, synchronized moment.

## 20 CCI Register Map

### 20.1 Introduction

This Section describes the CCI Register Map. CCI Registers are the registers that can be accessed via the CCI bus.

The CCI Registers are grouped by function, with each group occupying a pre-allocated region of the address space. The primary CCI Register categories are given in *Table 179*.

The CCS Registers are the set of CCI Registers that are defined in this Specification. They are located within the CCS Register area, i.e. register indexes 0x0000 through 0x2FFF. Certain RO Register(s) may be omitted from the sensor, if CCS Static Data provides the same or similar information. See details in *Section 20.6*, *Section 20.7*, and *Annex B*.

**Table 179 CCI Register Groupings**

CCI Indices	RW	Description
<b>CCS Registers – [0x0000 – 0x1FFF]</b>		
<b>Configuration Registers – [0x0000 – 0x0FFF]</b>		
RO, RO Dynamic, and RW Registers		
0x0000 – 0x00FF	Read Only	Status Registers (Static and 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
0x0700 – 0x07FF	–	Not used
0x0800 – 0x08FF	–	CSI-2 and PHY Configuration Registers
0x0900 – 0x09FF	–	Binning Configuration Registers
0xA00 – 0xAFF	–	Data Transfer Interface Configuration Registers
0xB00 – 0xBFF	–	Image Processing and Sensor Correction Configuration Registers
0xC00 – 0xCFF	–	Timer Configuration Registers
0xD00 – 0xDFF	–	PDAF Control Registers
0xE00 – 0xEFF	–	Bracketing Interface Configuration Registers
0xFC0 – 0xFFFF	–	Not used
<b>Parameter Limit Registers – [0x1000 – 0x1FFF]</b>		
Static Read Only Registers		
0x1000 – 0x10DF	Read Only	Integration Time and Gain Parameters Limits
0x10E0 – 0x10EF	Read Only	Generic Parameters Limits
0x10F0 – 0x10FF	Read Only	ADC Parameter Limits
0x1100 – 0x11FF	Read Only	Video Timing Parameter Limits
0x1200 – 0x120F	Read Only	Image Scaling Parameter Limits
0x1210 – 0x121F	Read Only	HDR Limit Registers

CCI Indices	RW	Description
0x1230 – 0x126F	Read Only	USL Capability Registers
	Read Only	Image Compression Capability Registers
	Read Only	Test Mode Capability Registers
	Read Only	Not used
	Read Only	FIFO Capability Registers
	Read Only	CSI-2 Capability Registers
	Read Only	Binning Capability Registers
	Read Only	Data Transfer Interface Capability Registers
	Read Only	Image Processing and Sensor Correction Capability Registers
	Read Only	Timer and Soft Reset Capability Registers
	Read Only	Not used
	Read Only	PDAF Capability Registers
	Read Only	Bracketing Interface Capability Registers
	–	Not used
<b>Image Statistics Registers – [0x2000 – 0x2FFF]</b>		
0x2000 – 0x2FFF	–	Reserved for Image statistics Registers
<b>Manufacturer Specific Registers – [0x3000 – 0xFFFF]</b>		
0x3000 – 0xFFFF	–	Manufacturer Specific Registers

## 20.2 Responsibilities

This Section clarifies certain register-related responsibilities, in order to ensure interoperability of CCS-compatible image sensors and Hosts.

### 20.2.1 Unused Registers

The image sensor may omit implementing some registers. Such registers are referred to as ‘unused registers’. If the Host reads an ‘unused register’ after writing, then the image sensor may return an undefined value (for example, either 0 or the written value).

### 20.2.2 RW Register Default Values

For RW registers, default values are defined as follows:

- If this Specification defines a default value for a RW register, then the image sensor shall implement that default value.
- For all reserved RW register indexes, the image sensor should implement a default value of zero.
- For all undefined RW register indexes, the image sensor shall implement a default value of zero. An undefined CCS register index is one that is located within the CCS Register area, but not defined in this Specification.

### 20.2.3 Accessing Registers

#### 20.2.3.1 Complete Registers

This Section clarifies Host and image sensor responsibilities when accessing 8-bit, 16-bit, and 32-bit registers where all register bits and register values are defined.

**Note:**

*In this Section, the term ‘undefined register’ means a CCI register that is not defined in this Specification, and is not defined by the image sensor manufacturer.*

Host Responsibilities:

- The Host may write to any RW register(s)
- The Host may read from any RW register(s)
- The Host may read from any RO or RO Dynamic register(s)
- The Host should not write to any RO or RO Dynamic register(s)
- The Host shall not write to any undefined register(s)
- The Host should not write to any reserved register(s) because doing so might have undefined effects
- The Host should not read from any undefined or reserved register(s)

Image sensor responsibilities:

- The Image sensor shall respond to both read and write commands to any RW register(s)
- The Image sensor shall respond to read commands to any RO or RO Dynamic register(s)
- If the Host writes to any RO or RO Dynamic register(s), then the image sensor shall ignore the value written
- If the Host writes to any undefined or reserved register(s), then the image sensor may ignore the value written
- If the Host reads from any undefined register(s), then the image sensor shall report the value 0x00
- If the Host reads from any reserved register(s), then the image sensor should report the value 0x00

### 20.2.3.2 Registers with Reserved or Undefined Bits/Values

4002 **Note:**

4003     *In this section, the terms ‘undefined bits’ and ‘undefined values’ mean register bits or values that are  
4004       not defined in this Specification.*

4005 This Section clarifies Host and image sensor responsibilities when accessing registers defined in this  
4006 Specification where some register bits or register values are either reserved, or undefined.

4007     **Example 1:** An 8-bit register where only bits [5:0] are defined, and the other two bits are either  
4008       reserved or undefined.

4009     **Example 2:** An 8-bit register with an 8-bit value, where only values 0, 1, 2, and 3 are defined and  
4010       the remaining 252 values are either reserved or undefined.

4011 Host responsibilities for writes:

- 4012     • The Host should not write the value 1 to any reserved or undefined bit, because doing so might  
4013       have undefined effects
- 4014     • The Host should not write any reserved or undefined value to any register field, because doing so  
4015       might have undefined effect.

4016 Image sensor responsibilities:

- 4017     • If the Host writes a non-zero value to any reserved or undefined bits, then the image sensor may  
4018       ignore the written value
- 4019     • If the Host writes the value zero to any reserved or undefined bits, then the image sensor shall  
4020       behave as specified in this Specification
- 4021     • If the Host writes any reserved or undefined value to any defined register field, then the image  
4022       sensor may ignore the written value
- 4023     • If the Host writes a defined value to any defined register field, then the image sensor shall behave  
4024       as specified in this Specification

### 20.2.4 Manufacturer Specific Functionality

4025 Control of manufacturer specific functionalities should be implemented exclusively in the Manufacturer  
4026 Specific Registers area. The purpose of this recommendation is to ensure compatibility with future version  
4027 of the CCS Specification, which are expected to use additional registers in the other register areas. Image  
4028 sensors designers should note that not following this recommendation for a given image sensor product could  
4029 cause it to become non-compliant with future CCS versions.

## 20.3 Multi-Byte Registers Index Space

This Section defines the valid locations (register indexes) for the MS Bytes and LS Bytes for 16-bit registers and 32-bit register.

### 20.3.1 Valid 16-bit Register Indices

For 16-bit registers, the MS Byte's index shall be a multiple of 2. As a result, the LS bit of the 16-bit index for the MS Byte shall be zero.

Blue – Valid 16-bit indexes for the MS Data Byte of a 16-bit register

Red – Valid 16-bit indexes for the LS Data Byte of a 16-bit register

**Figure 75 Valid 16-Bit Indices for the MS Data Byte of 16-bit-Wide Register**

### 20.3.2 Valid 32-bit Register Indices

For 32-bit registers, the MS Byte's index shall be a multiple of 4. As a result, the 2 LS bits of the 16-bit index for the MS Byte shall both be zero.

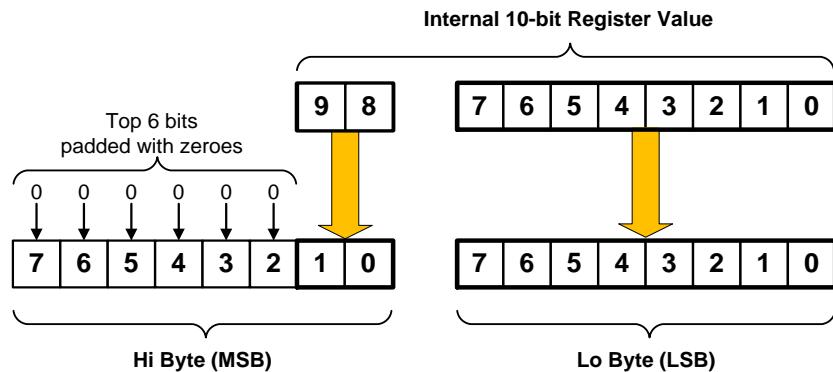
Blue – Valid 16-bit indexes for the MS Data Byte of a 32-bit register

Red – Valid 16-bit indexes for the LS Data Byte of a 32-bit register

**Figure 76 Valid 16-Bit Indices for the MS and LS Data Bytes of 32-bit-Wide Register**

## 20.4 Data Alignment Within CCI Registers

If the width of an image sensor's internal data register is narrower than the CCI 8-bit, 16-bit, or 32-bit register used to report its value to the Host, then the internal register value shall be right-aligned within the CCI Register, and the unused MS bits shall be padded with zeroes. *Figure 77* illustrates an example with a 10-bit internal register right-aligning within a 16-bit CCI register.



**Figure 77 Right Alignment for Packing 10-bit Data into Two 8-bit Registers**

## 20.5 Valid Register Formats

*Table 180* lists all of the valid formats for CCI registers.

**Table 180 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 255.999609375	8.8 fixed point number 8 integer bits (MS Byte), 8 fractional bits (LS Byte)
16-bit signed iReal	-128 to 127.999609375	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
32-bit unsigned integer	0 to 4,294,967,295	–

## 20.6 Configuration Registers: 0x0000–0xFFFF

### 20.6.1 Status Registers: 0x0000–0x00FF (RO, RO Dynamic)

#### 20.6.1.1 General Status Registers: 0x0000–0x0021 (RO, RO Dynamic)

4045

Table 181 General Status Registers

Index	Byte	Register Name	RW	Comment	Retime
0x0000	Hi	<b>module_model_id</b>	RO	16-bit camera module model number e.g. 552	NA
0x0001	Lo				
0x0002	–	<b>module_revision_number_major</b>	RO	Revision identifier of camera module for tuning/settings change	NA
0x0003	–	Reserved	–	–	NA
0x0004	–	Reserved	–	–	NA
0x0005	–	<b>frame_count</b>	RO dynamic	8-bit (0-254) Frame counter value	NA
0x0006	–	<b>pixel_order</b>	RO dynamic	Color Pixel Order	NA
0x0007	–	<b>MIPI_CCS_version</b>	RO	CCS Version <b>Bits 7–4:</b> Major <b>Bits 3–0:</b> Minor Examples: 0x10 means version 1.0 0x11 means version 1.1	NA
0x0008	Hi	<b>data_pedestal</b>	RO	Data pedestal – typically code 64 for 10-bit systems Refer to the Data Format Section	NA
0x0009	Lo				
0x000A	–	Reserved	–	–	NA
0x000B	–	Reserved	–	–	NA
0x000C	–	Reserved	–	–	NA
0x000D	–	Reserved	–	–	NA
0x000E	Hi	<b>module_manufacturer_id</b>	RO	Module manufacturer ID	NA
0x000F	Lo				
0x0010	–	<b>module_revision_number_minor</b>	RO	Device Revision Identifier of the camera module for minor changes	NA
0x0011	–	Reserved	–	Reserved	NA
0x0012	–	<b>module_date_year</b>	RO	<b>Bits 6–0</b> Last two digits of manufacturing year	NA
0x0013	–	<b>module_date_month</b>	RO	<b>Bits 3–0</b> Manufacturing month	NA
0x0014	–	<b>module_date_day</b>	RO	<b>Bits 4–0</b> Manufacturing day	NA

Index	Byte	Register Name	RW	Comment	Retime
0x0015	–	<b>module_date_phase</b>	RO	<b>Bits 2–0</b> Manufacturing phase 0: TS 1: ES 2: CS 3: MP	NA
0x0016	Hi	<b>sensor_model_id</b>	RO	–	NA
0x0017	Lo				NA
0x0018	–	<b>sensor_revision_number</b>	RO	<b>Note:</b> CCS v1.1 introduced support for the 16-bit register <b>sensor_revision_number_16</b> . The image sensor shall support either the 8-bit register or the 16-bit register, not both. See 0x0022. The unsupported register shall have value 0.	NA
0x0019	–	Reserved	–	–	NA
0x001A	–	<b>sensor_firmware_version</b>	RO	–	NA
0x001B	–	Reserved	–	–	NA
0x001C	Hi	<b>module_serial_number</b>	RO	Running serial number	NA
0x001D	–				
0x001E	–				
0x001F	Lo				
0x0020	Hi	<b>sensor_manufacturer_id</b>	RO	–	NA
0x0021	Lo				
0x0022	Hi	<b>sensor_revision_number_16</b>	RO	<b>Note:</b> CCS v1.1 introduced support for the 16-bit register <b>sensor_revision_number_16</b> . The image sensor shall support either the 8-bit register or the 16-bit register, not both. See 0x0018. The unsupported register shall have value 0.	NA
0x0023	Lo				

### 20.6.1.2 Frame Format Description Registers: 0x0040–0x007F (RO Dynamic)

The Frame Format Description is fully specified in *Section 7.14*. The Frame Format Description Register(s) may be omitted from the image sensor if the Generic Rule Based Block in CCS Static Data includes one or more FFD Records. See *Section B.2.11*.

**Table 182 Frame Format Description Registers**

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

4050

<b>Index</b>	<b>Byte</b>	<b>Register Name</b>	<b>RW</b>	<b>Comment</b>	<b>Retime</b>
0x0060	Hi	<b>frame_format_descriptor_4_0</b>	RO dynamic	Valid only for frame_format_model_type 0x02	NA
0x0061	–				
0x0062	–				
0x0063	Lo				
0x0064	Hi	<b>frame_format_descriptor_4_1</b>	RO dynamic	Valid only for frame_format_model_type 0x02	NA
0x0065	–				
0x0066	–				
0x0067	Lo				
0x0068	Hi	<b>frame_format_descriptor_4_2</b>	RO dynamic	Valid only for frame_format_model_type 0x02	NA
0x0069	–				
0x006A	–				
0x006B	Lo				
0x006C	Hi	<b>frame_format_descriptor_4_3</b>	RO dynamic	Valid only for frame_format_model_type 0x02	NA
0x006D	–				
0x006E	–				
0x006F	Lo				
0x0070	Hi	<b>frame_format_descriptor_4_4</b>	RO dynamic	Valid only for frame_format_model_type 0x02	NA
0x0071	–				
0x0072	–				
0x0073	Lo				
0x0074	Hi	<b>frame_format_descriptor_4_5</b>	RO dynamic	Valid only for frame_format_model_type 0x02	NA
0x0075	–				
0x0076	–				
0x0077	Lo				
0x0078	Hi	<b>frame_format_descriptor_4_6</b>	RO dynamic	Valid only for frame_format_model_type 0x02	NA
0x0079	–				
0x007A	–				
0x007B	Lo				
0x007C	Hi	<b>frame_format_descriptor_4_7</b>	RO dynamic	Valid only for frame_format_model_type 0x02	NA
0x007D	–				
0x007E	–				
0x007F	Lo				

### 20.6.1.3 Analog Gain Description Registers: 0x0080–0x009F (RO)

These registers are not dynamic, but may be output on the status line in order that the analog Gain code(s) can be interpreted. Analog Gain is fully described in *Section 9*. The Analog Gain Description Register(s) may be omitted from the image sensor if CCS Static Data provides the same information. See *Section B.2.8*.

**Table 183 Analog Gain Description Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x0080	Hi	analog_gain_capability	RO	Analog Gain Capability	NA
0x0081	Lo				
0x0082	Hi	Reserved	RO		NA
0x0083	Lo				
0x0084	Hi	analog_gain_code_min	RO	Minimum recommended analog Gain code <b>Format:</b> 16-bit unsigned integer	NA
0x0085	Lo				
0x0086	Hi	analog_gain_code_max	RO	Maximum recommended analog Gain code <b>Format:</b> 16-bit unsigned integer	NA
0x0087	Lo				
0x0088	Hi	analog_gain_code_step_size	RO	Analog Gain code step size <b>Format:</b> 16-bit unsigned integer	NA
0x0089	Lo				
0x008A	Hi	analog_gain_type	RO	Analog Gain type <b>Format:</b> 16-bit unsigned integer	NA
0x008B	Lo				
0x008C	Hi	analog_gain_m0	RO	Analog Gain m0 constant <b>Format:</b> 16-bit signed integer	NA
0x008D	Lo				
0x008E	Hi	analog_gain_c0	RO	Analog Gain c0 constant <b>Format:</b> 16-bit signed integer	NA
0x008F	Lo				
0x0090	Hi	analog_gain_m1	RO	Analog Gain m1 constant <b>Format:</b> 16-bit signed integer	NA
0x0091	Lo				
0x0092	Hi	analog_gain_c1	RO	Analog Gain c1 constant <b>Format:</b> 16-bit signed integer	NA
0x0093	Lo				
0x0094	Hi	analog_linear_gain_min	RO	Added in CCS v1.1	NA
0x0095	Lo				
0x0096	Hi	analog_linear_gain_max	RO	Added in CCS v1.1	NA
0x0097	Lo				
0x0098	Hi	analog_linear_gain_step_size	RO	Added in CCS v1.1	NA
0x0099	Lo				
0x009A	Hi	analog_exponential_gain_min	RO	Added in CCS v1.1	NA
0x009B	Lo				

Index	Byte	Register Name	RW	Comment	Retime
0x009C	Hi	analog_exponential_gain_max	RO	Added in CCS v1.1	NA
0x009D	Lo				
0x009E	Hi	analog_exponential_gain_step_size	RO	Added in CCS v1.1	NA
0x009F	Lo				

#### 20.6.1.4 Data Format Description Registers: 0x00C0–0x00E1 (RO)

Data Format Descriptors are fully specified in *Section 7.10* and *Section 7.11*. The Data Format Description Register(s) may be omitted from the image sensor if CCS Static Data provides the same information. See *Section B.2.8*.

**Table 184 Data Format Description Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x00C0	–	data_format_model_type	RO	–	NA
0x00C1	–	data_format_model_subtype	RO	–	NA
0x00C2	Hi	data_format_descriptor_0	RO	–	NA
0x00C3	Lo				
0x00C4	Hi	data_format_descriptor_1	RO	–	NA
0x00C5	Lo				
0x00C6	Hi	data_format_descriptor_2	RO	–	NA
0x00C7	Lo				
0x00C8	Hi	data_format_descriptor_3	RO	–	NA
0x00C9	Lo				
0x00CA	Hi	data_format_descriptor_4	RO	–	NA
0x00CB	Lo				
0x00CC	Hi	data_format_descriptor_5	RO	–	NA
0x00CD	Lo				
0x00CE	Hi	data_format_descriptor_6	RO	–	NA
0x00CF	Lo				
0x00D0	Hi	data_format_descriptor_7	RO	–	NA
0x00D1	Lo				
0x00D2	Hi	data_format_descriptor_8	RO	Available in data format model type 0x02 only	NA
0x00D3	Lo				
0x00D4	Hi	data_format_descriptor_9	RO	Available in data format model type 0x02 only	NA
0x00D5	Lo				
0x00D6	Hi	data_format_descriptor_10	RO	Available in data format model type 0x02 only	NA
0x00D7	Lo				
0x00D8	Hi	data_format_descriptor_11	RO	Available in data format model type 0x02 only	NA
0x00D9	Lo				
0x00DA	Hi	data_format_descriptor_12	RO	Available in data format model type 0x02 only	NA
0x00DB	Lo				
0x00DC	Hi	data_format_descriptor_13	RO	Available in data format model type 0x02 only	NA
0x00DD	Lo				
0x00DE	Hi	data_format_descriptor_14	RO	Available in data format model type 0x02 only	NA
0x00DF	Lo				
0x00E0	Hi	data_format_descriptor_15	RO	Available in data format model type 0x02 only	NA
0x00E1	Lo				

4059

## 20.6.2 Set-Up Registers: 0x0100–0x01FF

### 20.6.2.1 General Set-Up Registers: 0x0100–0x0109 (RW)

4060

**Table 185 General Set-Up Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x0100	–	<b>mode_select</b>	RW	Mode Select 0: Software Standby 1: Streaming Refer to Operating Modes Section	No
0x0101	–	<b>image_orientation</b>	RW	Image orientation i.e. horizontal mirror and vertical flip Refer to Video Timing Section	No
0x0102	–	Reserved	–	–	NA
0x0103	–	<b>software_reset</b>	RW	Software reset Refer to Operating Modes Section	No
0x0104	–	<b>grouped_parameter_hold</b>	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 Section	Yes
0x0105	–	<b>mask_corrupted_frames</b>	RW	Refer to Integration Time and Gain Control Section	Yes
0x0106	–	<b>fast_standby_ctrl</b>	RW	0: Frame completes before mode entry 1: Frame may be truncated before mode entry	No
0x0107	–	<b>CCI_address_ctrl</b>	RW	Expressed as 8 bit (e.g. 0x20 write, 0x21 read is entered as 0x20)	No
0x0108	–	<b>2nd_CCI_if_ctrl</b>	RW	<b>Bit 0</b> 1: enable 2 <sup>nd</sup> CCI interface 0: disable 2 <sup>nd</sup> CCI interface <b>Bit 1</b> 1: enable ACK sending for 2 <sup>nd</sup> CCI interface 0: disable ACK sending for 2 <sup>nd</sup> CCI interface	No
0x0109	–	<b>2nd_CCI_address_ctrl</b>	RW	Expressed as 8 bit (e.g. 0x20 write, 0x21 read is entered as 0x20)	No

### 20.6.2.2 Output Set-Up Registers: 0x0110–0x011F (RW)

4061 The Output Set-Up registers are fully described in *Section 7.5*.

4062 **Table 186 Output Set-Up Registers**

<b>Index</b>	<b>Byte</b>	<b>Register Name</b>	<b>RW</b>	<b>Comment</b>	<b>Retime</b>
0x0110	–	<b>CSI_channel_identifier</b>	RW	Virtual Channel Identifier <b>Range:</b> 0–3, or 0–15, or 0–31	No
0x0111	–	<b>CSI_signaling_mode</b>	RW	–	No
0x0112	Hi	<b>CSI_data_format</b>	RW	<b>CSI Data Format</b> <b>0x0808:</b> RAW8 (top 8-bits of internal data) <b>0xA06:</b> 10-b to 6-b compression <b>0xA07:</b> 10-b to 7-b compression <b>0xA08:</b> 10-b to 8-b compression <b>0xA0A:</b> RAW10 (top 10-bits of internal data) <b>0xC06:</b> 12-b to 6-b compression <b>0xC07:</b> 12-b to 7-b compression <b>0xC08:</b> 12-b to 8-b compression <b>0xC0A:</b> 12-b to 10-b compression <b>0xC0C:</b> RAW12 (top 12-bits of internal data) <b>0xE0E:</b> RAW14 (top 14-bits of internal data) <b>0x1010:</b> RAW16 (top 16-bits of internal data) <b>0x1414:</b> RAW20 (top 20-bits of internal data) <b>0x1818:</b> RAW24 (top 24-bits of internal data)	No
0x0113	Lo				
0x0114	–	<b>CSI_lane_mode</b>	RW	Number of CSI-2 lanes to be used <b>Range:</b> 0 (1-Lane) to 7 (8-Lane)	No
0x0115	–	Reserved	RW	Reserved	No
0x0116	–	Reserved	RW	Reserved	No
0x0117	–	Reserved	RW	Reserved	No
0x0118	–	Reserved	RW	Reserved	No
0x0119	–	Reserved	RW	Reserved	No
0x011A	–	Reserved	RW	Reserved	No
0x011B	–	Reserved	RW	Reserved	No
0x011C	–	Reserved	RW	Reserved	No
0x011D	–	<b>DPCM_Frame_DT</b>	RW	–	No
0x011E	–	<b>bottom_embedded_data_DT</b>	RW	–	No
0x011F	–	<b>bottom_embedded_data_VC</b>	RW	–	No

### 20.6.2.3 Gain Set-Up Register: 0x0120 (RW)

4063 The Gain Set-Up registers are fully described in *Section 9*.

4064 **Table 187 Gain Set-Up Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x0120		gain_mode	RW	0: Global Analog Gain (Default) 1: Reserved 2: Alternate Global analog Gain <b>Default:</b> 0 if supported, else 2. Refer to Integration Time and Gain Section	No

### 20.6.2.4 ADC Set-Up Registers: 0x0121 (RW)

4065 The ADC Set-Up registers are fully described in *Section 7.6*.

4066 **Table 188 ADC Set-up Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x0121		ADC_bit_depth	RW	0: Not controlled by Host Other values: ADC bit depth	No

### 20.6.2.5 Generic Control Registers: 0x0122 (RW)

4067 **Table 189 Generic Control Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x0122		emb_data_ctrl	RW	Added in CCS v1.1	No

4068

### 20.6.2.6 GPIO Set-Up Registers: 0x0130 (RO)

4069 The GPIO Set-Up register is fully described in *Section 5.2.10*.

4070 In this specification version, this register is defined as Read-Only.

4071 **Table 190 GPIO Set-Up Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x0130	–	GPIO_TRIGGER_mode	RO	0: Input used as an external trigger to start Exposure. Other values: reserved	No

### 20.6.2.7 Reference Clock Frequency Registers: 0x0136–0x0137 (RW)

4072 **Table 191 Reference Clock Frequency Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x0136	Hi	extclk_frequency_mhz	RW	Nominal Extclk frequency in MHz	No
0x0137	Lo				

### 20.6.2.8 Temperature Sensor Registers: 0x0138–0x013A (RW, RO Dynamic)

4073

**Table 192 Temperature Sensor Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x0138	–	temp_sensor_ctrl	RW	–	No
0x0139	–	temp_sensor_mode	RO	Reserved for future use	NA
0x013A	–	temp_sensor_output	RO dynamic	–	NA

### 20.6.3 Integration Time and Gain Registers: 0x0200–0x02FF (RW)

4074

The Integration Time and Gain registers are fully described in *Section 9 Integration Time and Gain Control*.

#### 20.6.3.1 Integration Time Registers: 0x0200–0x0203 (RW)

4075

**Table 193 Integration Time Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x0200	Hi	fine_integration_time	RW	Fine Integration Time (pixels) <b>Format:</b> 16-bit unsigned integer	Yes
0x0201	Lo			Coarse Integration Time (lines) <b>Format:</b> 16-bit unsigned integer	
0x0202	Hi	coarse_integration_time	RW	Coarse Integration Time (lines) <b>Format:</b> 16-bit unsigned integer	Yes
0x0203	Lo			Coarse Integration Time (lines) <b>Format:</b> 16-bit unsigned integer	

#### 20.6.3.2 Analog Gain Registers: 0x0204–0x020D (RW)

4076

**Table 194 Analog Gain Registers**

Index	Byte	Name	RW	Comment	Retime
0x0204	Hi	analog_gain_code_global	RW	Global Analog Gain Code <b>Format:</b> 16-bit unsigned integer	Yes
0x0205	Lo			Added in CCS v1.1	
0x0206	Hi	analog_linear_gain_global	RW	Alternate Global Analog Gain <b>Format:</b> 16-bit unsigned iReal	Yes
0x0207	Lo			Added in CCS v1.1	
0x0208	Hi	analog_exponential_gain_global	RW	Alternate Global Analog Gain <b>Format:</b> 16-bit signed iReal	Yes
0x0209	Lo			Added in CCS v1.1	

#### 20.6.3.3 Digital Gain Registers: 0x020E–0x020F (RW)

4077

**Table 195 Digital Gain Registers**

Index	Byte	Name	RW	Comment	Retime
0x020E	Hi	digital_gain_global	RW	Global digital Gain for all channels <b>Format:</b> 16-bit unsigned iReal	Yes
0x020F	Lo			Added in CCS v1.1	

#### 20.6.3.4 HDR Control Registers: 0x0216–0x0225 (RW)

4078

The HDR Control registers are fully described in *Section 9.9*.

4079

**Table 196 HDR Control Registers**

Index	Byte	Name	RW	Comment	Retime
0x0216	Hi	<b>short_analog_gain_global</b>	RW	—	Yes
0x0217	Lo				
0x0218	Hi	<b>short_digital_gain_global</b>	RW	—	Yes
0x0219	Lo				
0x021A	—	Reserved	NA	Reserved	NA
0x021B	—				
0x021C	—				
0x021D	—				
0x021E	—				
0x021F	—				
0x0220	—	<b>HDR_mode</b>	RW	—	No
0x0221	—	<b>HDR_resolution_reduction</b>	RW	—	No
0x0222	—	<b>exposure_ratio</b>	RW	—	Yes
0x0223	—	<b>HDR_internal_bit_depth</b>	RW	—	Yes
0x0224	Hi	<b>short_coarse_integration_time</b>	RW	—	Yes
0x0225	Lo				
0x0226	Hi	<b>short_analog_linear_gain_global</b>	RW	Added in CCS v1.1	Yes
0x0227	Lo				
0x0228	Hi	<b>short_analog_exponential_gain_global</b>	RW	Added in CCS v1.1	Yes
0x0229	Lo				

## 20.6.4 Video Timing Registers: 0x0300–0x03FF (RW)

4080 The Video Timing Registers are fully described in *Section 8*.

### 20.6.4.1 Clock Set-Up Registers: 0x0300–0x0315 (RW)

4081 **Note:**

4082 A USL sensor has different retiming rules for some of these registers (marked with \* in Table 197),  
4083 depending on the clock tree used by the sensor. See *Annex D*.

4084 **Table 197 Clock Set-Up Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x0300	Hi	<b>vt_pix_clk_div</b>	RW	Video Timing Pixel Clock Divider <b>Format:</b> 16-bit unsigned integer	No*
0x0301	Lo				
0x0302	Hi	<b>vt_sys_clk_div</b>	RW	Video Timing System Clock Divider Value <b>Format:</b> 16-bit unsigned integer	No*
0x0303	Lo				
0x0304	Hi	<b>pre_pll_clk_div or vt_pre_pll_clk_div</b>	RW	This register address may support two different functionalities. If the image sensor supports both 1-PLL systems and 2-PLL systems, then the register <b>pll_mode</b> controls which will be in effect. Pre PLL clock Divider Value <b>Format:</b> 16-bit unsigned integer	No*
0x0305	Lo				
0x0306	Hi	<b>pll_multiplier</b> or <b>vt_pll_multiplier</b>	RW	This register address may support two different functionalities. If the image sensor supports both 1-PLL systems and 2-PLL systems, then the register <b>pll_mode</b> controls which will be in effect. PLL multiplier Value <b>Format:</b> 16-bit unsigned integer	No*
0x0307	Lo				
0x0308	Hi	<b>op_pix_clk_div</b>	RW	Output Pixel Clock Divider <b>Format:</b> 16-bit unsigned integer	No*
0x0309	Lo				
0x030A	Hi	<b>op_sys_clk_div</b>	RW	Output System Clock Divider Value <b>Format:</b> 16-bit unsigned integer	No*
0x030B	Lo				
0x030C	Hi	<b>op_pre_pll_clk_div</b>	RW	This register is used in 2-PLL mode. See register <b>pll_mode</b> . Pre PLL clock Divider Value <b>Format:</b> 16-bit unsigned integer	No*
0x030D	Lo				
0x030E	Hi	<b>op_pll_multiplier</b>	RW	This register is used in 2-PLL mode. See register <b>pll_mode</b> . PLL multiplier Value <b>Format:</b> 16-bit unsigned integer	No*
0x030F	Lo				
0x0310	–	<b>pll_mode</b>	RW	<b>Bit 0</b> 0: 1-PLL mode 1: 2-PLL mode	No*
0x0311	–	Reserved	NA	Reserved for future use	NA

Index	Byte	Register Name	RW	Comment	Retime
0x0312	Hi	<b>op_pix_clk_div_rev</b>		Added in CCS v1.1	Special *
0x0313	Lo				
0x0314	Hi	<b>op_sys_clk_div_rev</b>		Added in CCS v1.1	Special *
0x0315	Lo				

#### 20.6.4.2 Frame Timing Registers: 0x0340–0x0343 (RW)

4085

Table 198 Frame Timing Registers

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

#### 20.6.4.3 Image Size Registers: 0x0344–0x034F (RW)

4086

Table 199 Image Size Registers

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

#### 20.6.4.4 Timing Mode Registers: 0x0350–0x0355 (RW)

4087

**Table 200 Timing Mode Registers**

<b>Index</b>	<b>Byte</b>	<b>Register Name</b>	<b>RW</b>	<b>Comment</b>	<b>Retime</b>
0x0350	–	<b>frame_length_ctrl</b>	RW	<b>Bit 0</b> 1: Auto frame length in use 0: Auto frame length not in use	No
0x0351	–	Reserved	–	Reserved	NA
0x0352	–	<b>timing_mode_ctrl</b>	RW	<b>Bit 0</b> 1: Manual readout mode <b>Bit 1</b> 1: Delayed Exposure mode	Yes
0x0353	–	<b>start_readout_rs</b>	RW	<b>Bit 0</b> 1: Manual readout start (auto clear)	No
0x0354	Hi	<b>frame_margin</b>	RW	–	Yes
0x0355	Lo				

#### 20.6.4.5 Sub-Sampling Registers: 0x0380–0x0387 (RW)

4088

**Table 201 Sub-Sampling Registers**

<b>Index</b>	<b>Byte</b>	<b>Register Name</b>	<b>RW</b>	<b>Comment</b>	<b>Retime</b>
0x0380	Hi	<b>x_even_inc</b>	RW	Increment for even pixels: 0, 2, 4, etc. <b>Format:</b> 16-bit unsigned integer	Yes
0x0381	Lo				
0x0382	Hi	<b>x_odd_inc</b>	RW	Increment for odd pixels: 1, 3, 5, etc. <b>Format:</b> 16-bit unsigned integer	Yes
0x0383	Lo				
0x0384	Hi	<b>y_even_inc</b>	RW	Increment for even pixels: 0, 2, 4, etc. <b>Format:</b> 16-bit unsigned integer	Yes
0x0385	Lo				
0x0386	Hi	<b>y_odd_inc</b>	RW	Increment for odd pixels: 1, 3, 5, etc. <b>Format:</b> 16-bit unsigned integer	Yes
0x0387	Lo				

#### 20.6.4.6 Monochrome Readout Registers: 0x0390–0x0390 (RW)

4089

**Table 202 Monochrome Readout Registers**

<b>Index</b>	<b>Byte</b>	<b>Register Name</b>	<b>RW</b>	<b>Comment</b>	<b>Retime</b>
0x0390	-	<b>monochrome_en</b>	RW	Added in CCS v1.1	Yes

4090

### 20.6.5 Image Scaling Registers: 0x0400–0x04FF (RW and RO)

4091

The Image Scaling registers are fully described in *Section 11*.

4092

**Table 203 Image Scaling Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x0400	Hi	<b>scaling_mode</b>	RW	0: No scaling 1: Horizontal Scaling 2: Reserved	Yes
0x0401	Lo				
0x0402	Hi	Reserved	RO	–	NA
0x0403	Lo				
0x0404	Hi	<b>scale_m</b>	RW	Down scale factor: M component <b>Range:</b> From N upwards <b>Format:</b> 16-bit unsigned integer	Yes
0x0405	Lo				
0x0406	Hi	<b>scale_n</b>	RO	Down scale factor: N component <b>Value:</b> Vendor-specific, typically 16 (fixed) <b>Format:</b> 16-bit unsigned integer	NA
0x0407	Lo				
0x0408	Hi	<b>digital_crop_x_offset</b>	RW	Offset from X-address of the top left corner of the visible pixel data after analog crop, bin and subsample	Yes
0x0409	Lo				
0x040A	Hi	<b>digital_crop_y_offset</b>	RW	Offset from Y-address of the top left corner of the visible pixel data after analog crop, bin and subsample	Yes
0x040B	Lo				
0x040C	Hi	<b>digital_crop_image_width</b>	RW	Image width after digital crop	Yes
0x040D	Lo				
0x040E	Hi	<b>digital_crop_image_height</b>	RW	Image height after digital crop	Yes
0x040F	Lo				

### 20.6.6 Image Compression Registers: 0x0500–0x0501 (RO)

4093 The Image Compression registers are fully described in *Section 12*.

4094 **Table 204 Image Compression Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x0500	Hi	<b>compression_mode</b>	RO	1: DPCM/PCM Compression – Simple Predictor <b>Other Values:</b> Reserved <b>Default:</b> 0x01	No
0x0501	Lo				

### 20.6.7 Test Pattern Registers: 0x0600–0x060B (RW)

4095 The Test Pattern registers are fully described in *Section 10*.

4096 **Table 205 Test Pattern Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x0600	Hi	<b>test_pattern_mode</b>	RW	–	Yes
0x0601	Lo				
0x0602	Hi	<b>test_data_red</b>	RW	–	Yes
0x0603	Lo				
0x0604	Hi	<b>test_data_greenR</b>	RW	–	Yes
0x0605	Lo				
0x0606	Hi	<b>test_data_blue</b>	RW	–	Yes
0x0607	Lo				
0x0608	Hi	<b>test_data_greenB</b>	RW	–	Yes
0x0609	Lo				
0x060A	–	<b>value_step_size_smooth</b>	RW	–	No
0x060B	–	<b>value_step_size_quantised</b>	RW	–	No

### 20.6.8 PHY Configuration Registers: 0x0800–0x0819 (RW)

4097

The PHY Configuration registers are fully described in *Section 7.9*.

4098

**Table 206 PHY Configuration Registers**

<b>Index</b>	<b>Byte</b>	<b>Register Name</b>	<b>RW</b>	<b>Comment</b>	<b>Retime</b>	
0x0800	–	<b>tclk_post</b>	RW	For D-PHY control Not recommended for new designs.	No	
0x0801	–	<b>ths_prepare</b>	RW		No	
0x0802	–	<b>ths_zero_min</b>	RW		No	
0x0803	–	<b>ths_trail</b>	RW		No	
0x0804	–	<b>tclk_trail_min</b>	RW		No	
0x0805	–	<b>tclk_prepare</b>	RW		No	
0x0806	–	<b>tclk_zero</b>	RW		No	
0x0807	–	<b>tlpx</b>	RW		No	
0x0808	–	<b>dphy_ctrl</b>	RW	0: Use automatic control 1: Use UI control 2: Use register control For D-PHY and C-PHY control	No	
0x0809	–	Reserved	–	Reserved	–	
0x080A	Hi	<b>tclk_post_ex</b>	RW	For D-PHY control	No	
0x080B	Lo				No	
0x080C	Hi	<b>ths_prepare_ex</b>	RW		No	
0x080D	Lo				No	
0x080E	Hi	<b>ths_zero_min_ex</b>	RW		No	
0x080F	Lo				No	
0x0810	Hi	<b>ths_trail_ex</b>	RW		No	
0x0811	Lo				No	
0x0812	Hi	<b>tclk_trail_min_ex</b>	RW		No	
0x0813	Lo				No	
0x0814	Hi	<b>tclk_prepare_ex</b>	RW		No	
0x0815	Lo				No	
0x0816	Hi	<b>tclk_zero_ex</b>	RW		No	
0x0817	Lo				No	
0x0818	Hi	<b>tlpx_ex</b>	RW		No	
0x0819	Lo				No	

### 20.6.9 CSI-2 Link Rate Registers: 0x0820–0x0823 (RW)

The CSI-2 Link Rate registers are fully described in *Section 7.9*.

**Table 207 CSI-2 Requested Link Rate Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x0820	Hi	<b>requested_link_rate</b>	RW	Target bitrate for CSI-2 transmission: <ul style="list-style-type: none"> <li>For D-PHY, this is bitrate (Mbit/s)</li> <li>For C-PHY, this is Symbol rate (Msym/s)</li> </ul>	No
0x0821	–				
0x0822	–				
0x0823	Lo				

### 20.6.10 Equalization Control Registers: 0x0824–0x0825 (RW)

The Equalization Control registers are fully described in *Section 7.9*.

**Table 208 Equalization Control Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x0824	–	<b>DPHY_equalization_mode</b>	RW	Added in CCS 1.1	No
0x0825	–	<b>PHY_equalization_ctrl</b>	RW	Added in CCS 1.1	No

### 20.6.11 D-PHY Preamble Control Registers: 0x0826–0x0827 (RW)

The D-PHY Preamble Control registers are fully described in *Section 7.9*.

**Table 209 D-PHY Preamble Control Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x0826	–	<b>DPHY_preamble_ctrl</b>	RW	Added in CCS 1.1	No
0x0827	–	<b>DPHY_preamble_length</b>	RW	Added in CCS 1.1	No

### 20.6.12 D-PHY Spread Spectrum Control Registers: 0x0828 (RW)

The D-PHY Spread Spectrum Control register is fully described in *Section 7.9*.

**Table 210 D-PHY Spread Spectrum Control Register**

Index	Byte	Register Name	RW	Comment	Retime
0x0828	–	<b>PHY_SSC_ctrl</b>	RW	Added in CCS 1.1	No

### 20.6.13 Manual LP Control Register: 0x0829 (RW)

The Manual LP control register is fully described in *Section 7.9*.

**Table 211 Manual LP Control Register**

Index	Byte	Register Name	RW	Comment	Retime
0x0829	–	<b>manual_LP_ctrl</b>	RW	Added in CCS v1.1	No

**20.6.14 Additional PHY Configuration Registers: 0x082A-0x082F (RW)**

4109 The PHY Configuration registers are fully described in *Section 7.9*.

4110

**Table 212 Additional PHY Configuration Registers**

<b>Index</b>	<b>Byte</b>	<b>Register Name</b>	<b>RW</b>	<b>Comment</b>	<b>Retime</b>
0x082A	–	<b>t wakeup</b>	RW	Added in CCS v1.1	No
0x082B	–	<b>t init</b>	RW	Added in CCS v1.1	No
0x082C	–	<b>ths_exit</b>	RW	Added in CCS v1.1	No
0x082D	–	Reserved	–	Reserved for future use	–
0x082E	Hi	<b>ths_exit_ex</b>	RW	Added in CCS v1.1	No
0x082F	Lo				

### 20.6.15 PHY Calibration Configuration Registers: 0x0830–0x083F (RW)

The PHY Calibration Configuration registers are fully described in *Section 7.7*.

**Table 213 D-PHY 1.2 Related Configuration Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x0830	–	<b>PHY_periodic_calibration_ctrl</b>	RW	0: Do not send calibration sequence 1: Send calibration sequence during frame blanking <b>Default:</b> 0	No
0x0831	–	<b>PHY_periodic_calibration_interval</b>	RW	Output frame interval of period skew calibration 0: No output 1: Output with all frames A: Output once by A frames (for example, number 2 means every 2 <sup>nd</sup> frame) <b>Default:</b> 0	No
0x0832	–	<b>PHY_init_calibration_ctrl</b>	RW	0: Do not send calibration sequence 1: Send calibration sequence during start of streaming <b>Default:</b> 0	No
0x0833	–	<b>DPHY_calibration_mode</b>	RW	Added in CCS 1.1	No
0x0834	–	<b>CPHY_calibration_mode</b>	RW	Added in CCS 1.1	No
0x0835	–	<b>t3_calpreamble_length</b>	RW	Added in CCS 1.1	No
0x0836	–	<b>t3_calpreamble_length_per</b>	RW	Added in CCS 1.1	No
0x0837	–	<b>t3_calaltseq_length</b>	RW	Added in CCS 1.1	No
0x0838	–	<b>t3_calaltseq_length_per</b>	RW	Added in CCS 1.1	No
0x0839	–	Reserved	–	Reserved	NA
0x083A	Hi	<b>FM2_init_seed</b>	RW	Added in CCS 1.1	No
0x083B	Lo				
0x083C	Hi	<b>t3_caludefseq_length</b>	RW	Added in CCS 1.1	No
0x083D	Lo				
0x083E	Hi	<b>t3_caludefseq_length_per</b>	RW	Added in CCS 1.1	No
0x083F	Lo				

### 20.6.16 C-PHY Manual Control Registers: 0x0840–0x0851 (RW)

4113 The C-PHY Manual Control registers are fully described in *Section 7.9*.

4114 **Table 214 C-PHY Manual Control Registers**

<b>Index</b>	<b>Byte</b>	<b>Register Name</b>	<b>RW</b>	<b>Comment</b>	<b>Retime</b>
0x0840	–	Reserved	–	–	NA
0x0841	–	<b>TGR_Preamble_Length</b>	RW	For C-PHY control	No
0x0842	–	<b>TGR_Post_Length</b>	RW	For C-PHY control	No
0x0843	–	<b>TGR_Preamble_Prog_Sequence_0,1</b>	RW	For C-PHY control	No
0x0844	–	<b>TGR_Preamble_Prog_Sequence_2,3</b>	RW	For C-PHY control	No
0x0845	–	<b>TGR_Preamble_Prog_Sequence_4,5</b>	RW	For C-PHY control	No
0x0846	–	<b>TGR_Preamble_Prog_Sequence_6,7</b>	RW	For C-PHY control	No
0x0847	–	<b>TGR_Preamble_Prog_Sequence_8,9</b>	RW	For C-PHY control	No
0x0848	–	<b>TGR_Preamble_Prog_Sequence_10,11</b>	RW	For C-PHY control	No
0x0849	–	<b>TGR_Preamble_Prog_Sequence_12,13</b>	RW	For C-PHY control	No
0x084A	–	Reserved	–	–	NA
0x084B	–	Reserved	–	–	NA
0x084C	–	Reserved	–	–	NA
0x084D	–	Reserved	–	–	NA
0x084E	Hi	<b>t3_prepare</b>	RW	For C-PHY control	No
0x084F	Lo				
0x0850	Hi	<b>t3_ipx</b>	RW	For C-PHY control	No
0x0851	Lo				

### 20.6.17 CSI-2 v2 Feature Control Registers: 0x085A–0x08B1 (RW)

4115 The CSI-2 v2 Feature Control registers are fully described in *Section 7.7*.

#### 20.6.17.1 CSI-2 ALPS Feature Control Registers: 0x85A–0x85F (RW)

4116 **Table 215 ALPS Feature Control Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x085A	–	ALPS_ctrl	RW	–	No

#### 20.6.17.2 CSI-2 LRTE Feature Control Registers: 0x860–0x86F (RW)

4117 **Table 216 LRTE Feature Control Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x0860	Hi	TX_REG_CSI_EPD_EN_SSP_cphy	RW	Short Packet control register for C-PHY	No
0x0861	Lo				
0x0862	Hi	TX_REG_CSI_EPD_OP_SLP_cphy	RW	Long Packet control register for C-PHY	No
0x0863	Lo				
0x0864	Hi	TX_REG_CSI_EPD_EN_SSP_dphy	RW	Short Packet control register for D-PHY	No
0x0865	Lo				
0x0866	Hi	TX_REG_CSI_EPD_OP_SLP_dphy	RW	Long Packet control register for D-PHY	No
0x0867	Lo				
0x0868	-	TX_REG_CSI_EPD_MISC_OPTIONS_cphy	RW	Added in CCS v1.1	No
0x0869	-	TX_REG_CSI_EPD_MISC_OPTIONS_dphy	RW	Added in CCS v1.1	No

### 20.6.17.3 CSI-2 Data Scrambling Feature Control Registers: 0x870–0x8FF (RW)

4118

**Table 217 Data Scrambling Feature Control Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x0870	–	<b>scrambling_ctrl</b>	RW	–	No
0x0871	–	Reserved	–	Reserved for future use	NA
0x0872	Hi	<b>lane_1_seed_value0</b>	RW	Value 0	No
0x0873	Lo	<b>lane_2_seed_value0</b>	RW		No
0x0874	Hi		RW		No
0x0875	Lo	<b>lane_3_seed_value0</b>	RW		No
0x0876	Hi		RW		No
0x0877	Lo	<b>lane_4_seed_value0</b>	RW		No
0x0878	Hi		RW		No
0x0879	Lo	<b>lane_5_seed_value0</b>	RW	Value 1	No
0x087A	Hi		RW		No
0x087B	Lo	<b>lane_6_seed_value0</b>	RW		No
0x087C	Hi		RW		No
0x087D	Lo	<b>lane_7_seed_value0</b>	RW		No
0x087E	Hi		RW		No
0x087F	Lo	<b>lane_8_seed_value0</b>	RW		No
0x0880	Hi		RW		No
0x0881	Lo	<b>lane_1_seed_value1</b>	RW	Value 1	No
0x0882	Hi		RW		No
0x0883	Lo	<b>lane_2_seed_value1</b>	RW		No
0x0884	Hi		RW		No
0x0885	Lo	<b>lane_3_seed_value1</b>	RW		No
0x0886	Hi		RW		No
0x0887	Lo	<b>lane_4_seed_value1</b>	RW		No
0x0888	Hi		RW		No
0x0889	Lo	<b>lane_5_seed_value1</b>	RW		No
0x088A	Hi		RW		No
0x088B	Lo	<b>lane_6_seed_value1</b>	RW		No
0x088C	Hi		RW		No
0x088D	Lo	<b>lane_7_seed_value1</b>	RW		No
0x088E	Hi		RW		No
0x088F	Lo	<b>lane_8_seed_value1</b>	RW		No
0x0890	Hi		RW	Value 2	No
0x0891	Lo	<b>lane_1_seed_value2</b>	RW		No
0x0892	Hi		RW	Value 2	No
0x0893	Lo		RW		No

<b>Index</b>	<b>Byte</b>	<b>Register Name</b>	<b>RW</b>	<b>Comment</b>	<b>Retime</b>
0x0894	Hi	<b>lane_2_seed_value2</b>	<b>RW</b>		No
0x0895	Lo				No
0x0896	Hi	<b>lane_3_seed_value2</b>	<b>RW</b>		No
0x0897	Lo				No
0x0898	Hi	<b>lane_4_seed_value2</b>	<b>RW</b>		No
0x0899	Lo				No
0x089A	Hi	<b>lane_5_seed_value2</b>	<b>RW</b>		No
0x089B	Lo				No
0x089C	Hi	<b>lane_6_seed_value2</b>	<b>RW</b>		No
0x089D	Lo				No
0x089E	Hi	<b>lane_7_seed_value2</b>	<b>RW</b>		No
0x089F	Lo				No
0x08A0	Hi	<b>lane_8_seed_value2</b>	<b>RW</b>		No
0x08A1	Lo				No
0x08A2	Hi	<b>lane_1_seed_value3</b>	<b>RW</b>	Value 3	No
0x08A3	Lo				No
0x08A4	Hi	<b>lane_2_seed_value3</b>	<b>RW</b>		No
0x08A5	Lo				No
0x08A6	Hi	<b>lane_3_seed_value3</b>	<b>RW</b>		No
0x08A7	Lo				No
0x08A8	Hi	<b>lane_4_seed_value3</b>	<b>RW</b>		No
0x08A9	Lo				No
0x08AA	Hi	<b>lane_5_seed_value3</b>	<b>RW</b>		No
0x08AB	Lo				No
0x08AC	Hi	<b>lane_6_seed_value3</b>	<b>RW</b>		No
0x08AD	Lo				No
0x08AE	Hi	<b>lane_7_seed_value3</b>	<b>RW</b>		No
0x08AF	Lo				No
0x08B0	Hi	<b>lane_8_seed_value3</b>	<b>RW</b>		No
0x08B1	Lo				No

### 20.6.18 USL Control Registers: 0x08C0-0x08DF (RW)

4119 The USL Control registers are fully described in *Annex D*.

4120 **Table 218 USL Control Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x08C0	Hi	TX_USL_REV_ENTRY	RW	Added in CCS v1.1	No
0x08C1	Lo				
0x08C2	Hi	TX_USL_REV_Clock_Counter	RW	Added in CCS v1.1	No
0x08C3	Lo				
0x08C4	Hi	TX_USL_REV_LP_Counter	RW	Added in CCS v1.1	No
0x08C5	Lo				
0x08C6	Hi	TX_USL_REV_Frame_Counter	RW	Added in CCS v1.1	No
0x08C7	Lo				
0x08C8	Hi	TX_USL_REV_Chronological_Timetr	RW	Added in CCS v1.1	No
0x08C9	Lo				
0x08CA	Hi	TX_USL_FWD_ENTRY	RW	Added in CCS v1.1	No
0x08CB	Lo				
0x08CC	Hi	TX_USL_GPIO	RW	Added in CCS v1.1	No
0x08CD	Lo				
0x08CE	Hi	TX_USL_Operation	RW	Added in CCS v1.1	No
0x08CF	Lo				
0x08D0	Hi	TX_USL_ALP_CTRL	RW	Added in CCS v1.1	No
0x08D1	Lo				
0x08D2	Hi	TX_USL_APP_BTA_ACK_TIMEOUT	RW	Added in CCS v1.1	No
0x08D3	Lo				
0x08D4	Hi	TX_USL_SNS_BTA_ACK_TIMEOUT	RO / RO Dynamic	Added in CCS v.1.1	NA
0x08D5	Lo				
0x08D6	—	USL_Clock_Mode_D_ctrl	RW	Added in CCS v1.1	No
0x08D7	—	Reserved	—	Future use	NA
0x08D8	—	SNS_USL_LPDT_LRTE_cphy	RW	Added in CCS v1.1	No
0x08D9	—	SNS_USL_LPDT_LRTE_dphy	RW	Added in CCS v1.1	No
0x08DA	—	LRTE_usl_rx_enable_cphy	RW	Added in CCS v1.1	No
0x08DB	—	LRTE_usl_rx_enable_dphy	RW	Added in CCS v1.1	No

### 20.6.19 Binning Configuration Registers: 0x0900–0x0902 (RW)

The Binning Configuration registers are fully described in *Section 8.2.5*.

**Table 219 Binning Configuration Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x0900	–	<b>binning_mode</b>	RW	–	Yes
0x0901	–	<b>binning_type</b>	RW	–	Yes
0x0902	–	<b>binning_weighting</b>	RW	–	Yes

### 20.6.20 Data Transfer Interface Configuration Registers: 0x0A00–0x0A43 (RW, RO Dynamic)

The Data Transfer Interface Configuration registers are fully described in *Section 13*.

**Table 220 Data Transfer Interface Configuration Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x0A00	–	<b>data_transfer_if_1_ctrl</b>	RW	–	No
0x0A01	–	<b>data_transfer_if_1_status</b>	RO dynamic	–	No
0x0A02	–	<b>data_transfer_if_1_page_select</b>	RW	–	No
0x0A03	–	Reserved	–	For future use	NA
0x0A04	–	<b>data_transfer_if_1_data_0</b>	RW	–	No
...	...	...	...	...	...
0x0A43	–	<b>data_transfer_if_1_data_63</b>	RW	–	No

### 20.6.21 Image Processing and Sensor Correction Configuration Registers: 0x0B00–0x0BA0 (RW)

#### 20.6.21.1 Sensor Correction Configuration: 0x0B00–0x0B18 (RW)

4125

The Sensor Correction Configuration registers are fully described in *Section 14*.

4126

**Table 221 Sensor Correction Configuration Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x0B00	–	<b>shading_correction_en</b>	RW	–	Yes
0x0B01	–	<b>luminance_correction_level</b>	RW	–	Yes
0x0B02	–	<b>green_imbalance_filter_en</b>	RW	–	Yes
0x0B03	–	Reserved	NA	–	NA
0x0B04	–	Reserved	NA	–	NA
0x0B05	–	<b>mapped_defect_correct_en</b>	RW	–	Yes
0x0B06	–	<b>single_defect_correct_en</b>	RW	–	Yes
0x0B07	–	Reserved	NA	–	NA
0x0B08	–	<b>dynamic_couplet_correct_en</b>	RW	–	Yes
0x0B09	–	Reserved	NA	–	NA
0x0B0A	–	<b>combined_defect_correct_en</b>	RW	–	Yes
0x0B0B	–	Reserved	NA	–	NA
0x0B0C	–	<b>module_specific_correction_en</b>	RW	–	Yes
0x0B0D	–	Reserved	NA	–	NA
0x0B0E	–	Reserved	NA	–	NA
0x0B0F	–	Reserved	NA	–	NA
0x0B10	–	Reserved	NA	–	NA
0x0B11	–	Reserved	NA	–	NA
0x0B12	–	Reserved	NA	–	NA
0x0B13	–	<b>dynamic_triplet_defect_correct_en</b>	RW	–	Yes
0x0B14	–	Reserved	NA	–	NA
0x0B15	–	<b>NF_ctrl</b>	RW	–	Yes
0x0B16	–	Reserved	NA	–	NA
0x0B17	–	Reserved	NA	–	NA
0x0B18	–	Reserved	NA	–	NA

#### 20.6.21.2 Optical Black Pixel Readout Registers: 0x0B30–0x0B33 (RW)

The Optical Black Pixel Readout registers are fully described in *Section 14.9*.

Table 222 Optical Black Pixel Readout Registers

Index	Byte	Register Name	RW	Comment	Retime
0x0B30	–	OB_readout_ctrl	RW	–	No
0x0B31	–	OB_virtual_channel	RW	–	No
0x0B32	–	OB_DT	RW	–	No
0x0B33	–	OB_data_format	RW	–	No

#### 20.6.21.3 Color Temperature Feedback Registers: 0x0B8C–0x0B95 (RW)

The Color Temperature Feedback registers are fully described in *Section 14.8*.

Table 223 Color Temperature Feedback Registers

Index	Byte	Register Name	RW	Comment	Retime
0x0B8C	Hi	color_temperature	RW	–	Yes
0x0B8D	Lo				
0x0B8E	Hi	absolute_gain_greennr	RW	–	Yes
0x0B8F	Lo				
0x0B90	Hi	absolute_gain_red	RW	–	Yes
0x0B91	Lo				
0x0B92	Hi	absolute_gain_blue	RW	–	Yes
0x0B93	Lo				
0x0B94	Hi	absolute_gain_greenb	RW	–	Yes
0x0B95	Lo				

#### 20.6.21.4 CFA Conversion Registers: 0x0BA0–0x0BA0 (RW)

The CFA Conversion registers are fully described in *Annex C*.

Table 224 CFA Conversion Registers

Index	Byte	Register Name	RW	Comment	Retime
0x0BA0	-	CFA_conversion_ctrl	RW	Added in CCS v1.1	Yes

## 20.6.22 Timer Configuration Registers: 0x0C00–0x0CFF (RW, RO Dynamic)

### 20.6.22.1 Flash Strobe and SA Strobe Control Registers: 0x0C00–0x0C33 (RW, RO Dynamic)

4133 The Flash Strobe and Special Actuator (SA) Strobe registers are fully described in *Section 16.2* and *Section 16.3*, respectively.

4134

**Table 225 Flash and SA Strobe Configuration**

Index	Byte	Register Name	RW	Comment	Retime
0x0C00 – 0x0C11	–	Reserved	–	–	NA
0x0C12	–	<b>flash_strobe_adjustment</b>	RW	–	No
0x0C13	–	Reserved	–	–	NA
0x0C14	Hi	<b>flash_strobe_start_point</b>	RW	–	Yes
0x0C15	Lo				
0x0C16	Hi	<b>tflash_strobe_delay_rs_ctrl</b>	RW	–	Yes
0x0C17	Lo				
0x0C18	Hi	<b>tflash_strobe_width_high_rs_ctrl</b>	RW	–	Yes
0x0C19	Lo				
0x0C1A	–	<b>flash_mode_rs</b>	RW	–	No
0x0C1B	–	<b>flash_trigger_rs</b>	RW	See <b>Section 16.2</b> for specific retiming rules	Yes
0x0C1C	–	<b>flash_status</b>	RO dynamic	RO dynamic This register appears in the Config section because it's needed in embedded line	NA
0x0C1D	–	<b>SA_strobe_mode</b>	RW	–	No
0x0C1E	Hi	<b>SA_strobe_start_point</b>	RW	–	No
0x0C1F	Lo				
0x0C20	Hi	<b>tSA_strobe_delay_ctrl</b>	RW	–	No
0x0C21	Lo				
0x0C22	Hi	<b>tSA_strobe_width_ctrl</b>	RO	1µs fixed pulse width is used	NA
0x0C23	Lo				
0x0C24	–	<b>SA_strobe_trigger</b>	RW	See <b>Section 16.3</b> for specific retiming rules	Yes
0x0C25	–	<b>SA_strobe_status</b>	RO dynamic	RO dynamic This register appears in the Config section because it's needed in embedded line	NA
0x0C26	Hi	Reserved	–	–	NA
0x0C27	Lo				
0x0C28	Hi	Reserved	–	–	NA
0x0C29	Lo				

Index	Byte	Register Name	RW	Comment	Retime
0x0C2A	–	Reserved	–	–	NA
0x0C2B – 0x0C2F	–	Reserved	–	–	NA
0x0C30	Hi	<b>tSA_strobe_re_delay_ctrl</b>	RW	–	No
0x0C31	Lo				
0x0C32	Hi	<b>tSA_strobe_fe_delay_ctrl</b>	RW	–	No
0x0C33	Lo				

### 20.6.23 PDAF Control Registers: 0x0D00–0x0DFF (RW)

4136

The PDAF Control Registers are fully described in *Section 17*.

4137

**Table 226 PDAF Control Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x0D00	–	<b>PDAF_ctrl</b>	RW	–	No or Yes depending on bit 2 of <b>PDAF_capability_2</b> register
0x0D01	–	Reserved	–	–	NA
0x0D02	–	<b>PDAF_VC</b>	RW	–	No
0x0D03	–	<b>PDAF_DT</b>	RW	–	No
0x0D04	Hi	<b>pdaf_x_addr_start</b>	RW	–	Yes
0x0D05	Lo				
0x0D06	Hi	<b>pdaf_y_addr_start</b>	RW	–	Yes
0x0D07	Lo				
0x0D08	Hi	<b>pdaf_x_addr_end</b>	RW	–	Yes
0x0D09	Lo				
0x0D0A	Hi	<b>pdaf_y_addr_end</b>	RW	–	Yes
0x0D0B	Lo				

### 20.6.24 Bracketing Interface Configuration Registers: 0x0E00–0x0EFF (RW)

4138

The Bracketing Interface Control registers are fully described in *Section 9.8*.

4139

**Table 227 Bracketing Interface Configuration Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x0E00	–	<b>bracketing_LUT_ctrl</b>	RW	See <i>Section 9.8</i> for details	Special
0x0E01	–	<b>bracketing_LUT_mode</b>	RW	See <i>Section 9.8</i> for details	No
0x0E02	–	<b>bracketing_LUT_entry_ctrl</b>	RW	Reserved for future use	No
0x0E03 – 0x0E0F	–	Reserved	–	–	NA
0x0E10 – 0x0EFF	–	<b>bracketing_LUT_frame_A</b> (first entry) ... <b>bracketing_LUT_frame_n</b> (last entry)	RW	Dynamic usage depending on bracketing LUT capabilities and controls. See <i>Section 9.8</i> for details	Yes

## 20.7 Parameter Limit Registers: 0x1000–0x1FFF (Static RO)

Capability information may be provided via CCI Registers and/or via CCS Static Data. The Parameter Limit Register(s) may be omitted from the image sensor if CCS Static Data provides the same information. See [Section B.2.8](#).

### 20.7.1 Integration Time and Gain Parameter Limit Registers: 0x1000–0x10DF (RO)

The Integration Time and Gain Parameter Limit registers are fully described in [Section 9](#).

#### 20.7.1.1 Integration Time Parameter Limit Registers: 0x1000–0x100B (RO)

**Table 228 Integration Time Capability Registers**

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

### 20.7.1.2 Digital Gain Parameter Limit Registers: 0x1080–0x1089 (RO)

4145

**Table 229 Digital Gain Capability Registers**

<b>Index</b>	<b>Byte</b>	<b>Register Name</b>	<b>RW</b>	<b>Comment</b>	<b>Retime</b>
0x1080	-	Reserved	RO	-	NA
0x1081	-	<b>digital_gain_capability</b>	RO	0: None 1: Reserved 2: Global digital Gain	
0x1082	Hi	Reserved	RO	-	NA
0x1083	Lo				
0x1084	Hi	<b>digital_gain_min</b>	RO	Minimum recommended digital Gain value <b>Format:</b> 16-bit unsigned 8.8 fixed point number	NA
0x1085	Lo				
0x1086	Hi	<b>digital_gain_max</b>	RO	Maximum recommended digital Gain value <b>Format:</b> 16-bit unsigned 8.8 fixed point number	NA
0x1087	Lo				
0x1088	Hi	<b>digital_gain_step_size</b>	RO	Digital Gain step size <b>Format:</b> 16-bit unsigned 8.8 fixed point number	NA
0x1089	Lo				

### 20.7.2 Generic Parameter Limit Registers: 0x10E0–0x10EF (RO)

4146

This section may include multiple types capability registers.

#### 20.7.2.1 Data Pedestal Capability Registers: 0x10E0 (RO)

4147

The Data Pedestal Capability Registers are fully described in *Section 7.12.1*.

4148

**Table 230 Data Pedestal Capability Registers**

<b>Index</b>	<b>Byte</b>	<b>Register Name</b>	<b>RW</b>	<b>Comment</b>	<b>Retime</b>
0x10E0	-	<b>pedestal_capability</b>	RO	Added in CCS v1.1	NA

### 20.7.3 ADC Parameter Limit Registers: 0x10F0–0x10FF (RO)

4149

The ADC Parameter Limit Registers are fully described in *Section 7.6*.

#### 20.7.3.1 ADC Capability Registers: 0x10F0–0x10FF (RO)

4150

**Table 231 ADC Capability Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x10F0	–	<b>ADC_capability</b>	RO	<b>Bit 0</b> 1: Supports controllable ADC bit depth	NA
0x10F1	–	Reserved		–	NA
0x10F2	–	Reserved		–	NA
0x10F3	–	Reserved		–	NA
0x10F4	Hi	<b>ADC_bit_depth_capability</b>	RO	Added in CCS v1.1	NA
0x10F5	–				
0x10F6	–				
0x10F7	Lo				

### 20.7.4 Video Timing Parameter Limit Registers: 0x1100–0x11FF (RO)

4151

The Video Timing Parameter Limit registers are fully described in *Section 8* and *Section 9*.

#### 20.7.4.1 Pre-PLL and PLL Clock Set-up Capability Registers: 0x1100–0x11B9 (RO)

4152

**Table 232 Pre-PLL and PLL Clock Set-up Capability Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x1100	Hi	<b>min_ext_clk_freq_mhz</b>	RO	Minimum external clock frequency <b>Format:</b> IEEE 32-bit float or 32-bit unsigned iReal <b>Units:</b> MHz	NA
0x1101	–				
0x1102	–				
0x1103	Lo				
0x1104	Hi	<b>max_ext_clk_freq_mhz</b>	RO	Maximum external clock frequency <b>Format:</b> IEEE 32-bit float or 32-bit unsigned iReal <b>Units:</b> MHz	NA
0x1105	–				
0x1106	–				
0x1107	Lo				
0x1108	Hi	<b>min_pre_pll_clk_div</b> or <b>min_vt_pre_pll_clk_div</b>	RO	This register address may support two different functionalities. If the image sensor supports both 1-PLL systems and 2-PLL systems, then register <b>pll_mode</b> will control which is in effect. Minimum Pre PLL divider value <b>Format:</b> 16-bit unsigned integer	NA
0x1109	Lo				

Index	Byte	Register Name	RW	Comment	Retime
0x110A	Hi	<b>max_pre_pll_clk_div</b> or <b>max_vt_pll_clk_div</b>	RO	This register address may support two different functionalities. If the image sensor supports both 1-PLL systems and 2-PLL systems, then register <b>pll_mode</b> will control which is in effect.  Maximum Pre PLL divider value <b>Format:</b> 16-bit unsigned integer	NA
0x110B	Lo				
0x110C	Hi	<b>min_pll_ip_freq_mhz</b> or <b>min_vt_pll_ip_freq_mhz</b>	RO	This register address may support two different functionalities. If the image sensor supports both 1-PLL systems and 2-PLL systems, then register <b>pll_mode</b> will control which is in effect.  Minimum PLL input clock frequency <b>Format:</b> IEEE 32-bit float or 32-bit unsigned iReal <b>Units:</b> MHz	NA
0x110D	—				
0x110E	—				
0x110F	Lo				
0x1110	Hi	<b>max_pll_ip_freq_mhz</b> or <b>max_vt_pll_ip_freq_mhz</b>	RO	This register address may support two different functionalities. If the image sensor supports both 1-PLL systems and 2-PLL systems, then register <b>pll_mode</b> will control which is in effect.  Maximum PLL input clock frequency <b>Format:</b> IEEE 32-bit float or 32-bit unsigned iReal <b>Units:</b> MHz	NA
0x1111	—				
0x1112	—				
0x1113	Lo				
0x1114	Hi	<b>min_pll_multiplier</b> or <b>min_vt_pll_multiplier</b>	RO	This register address may support two different functionalities. If the image sensor supports both 1-PLL systems and 2-PLL systems, then register <b>pll_mode</b> will control which is in effect.  Minimum PLL multiplier <b>Format:</b> 16-bit unsigned integer	NA
0x1115	Lo				
0x1116	Hi	<b>max_pll_multiplier</b> or <b>max_vt_pll_multiplier</b>	RO	This register address may support two different functionalities. If the image sensor supports both 1-PLL systems and 2-PLL systems, then register <b>pll_mode</b> will control which is in effect.  Maximum PLL Multiplier <b>Format:</b> 16-bit unsigned integer	NA
0x1117	Lo				
0x1118	Hi	<b>min_pll_op_freq_mhz</b>	RO	Minimum PLL output clock frequency  <b>Format:</b> IEEE 32-bit float or 32-bit unsigned iReal <b>Units:</b> MHz	NA
0x1119	—				
0x111A	—				
0x111B	Lo				

Index	Byte	Register Name	RW	Comment	Retime
0x111C	Hi	<b>max_pll_op_freq_mhz</b>	RO	Maximum PLL output clock frequency <b>Format:</b> IEEE 32-bit float or 32-bit unsigned iReal <b>Units:</b> MHz	NA
0x111D	—				
0x111E	—				
0x111F	Lo				
0x11A0	Hi	<b>min_op_pre_pll_clk_div</b>	RO	This register address is used in 2-PLL mode. See register <b>pll_mode</b> . Minimum Pre PLL clock Divider Value <b>Format:</b> 16-bit unsigned integer	NA
0x11A1	Lo				
0x11A2	Hi	<b>max_op_pre_pll_clk_div</b>	RO	This register address is used in 2-PLL mode. See register <b>pll_mode</b> . Maximum Pre PLL clock Divider Value <b>Format:</b> 16-bit unsigned integer	NA
0x11A3	Lo				
0x11A4	Hi	<b>min_op_pll_ip_freq_mhz</b>	RO	This register address is used in 2-PLL mode. See register <b>pll_mode</b> . Minimum PLL input clock frequency <b>Format:</b> IEEE 32-bit float or 32-bit unsigned iReal <b>Units:</b> MHz	NA
0x11A5	—				
0x11A6	—				
0x11A7	Lo				
0x11A8	Hi	<b>max_op_pll_ip_freq_mhz</b>	RO	This register address is used in 2-PLL mode. See register <b>pll_mode</b> . Maximum PLL input clock frequency <b>Format:</b> IEEE 32-bit float or 32-bit unsigned iReal <b>Units:</b> MHz	NA
0x11A9	—				
0x11AA	—				
0x11AB	Lo				
0x11AC	Hi	<b>min_op_pll_multiplier</b>	RO	This register address is used in 2-PLL mode. See register <b>pll_mode</b> . Minimum PLL Multiplier <b>Format:</b> 16-bit unsigned integer	NA
0x11AD	Lo				
0x11AE	Hi	<b>max_op_pll_multiplier</b>	RO	This register address is used in 2-PLL mode. See register <b>pll_mode</b> . Maximum PLL Multiplier <b>Format:</b> 16-bit unsigned integer	NA
0x11AF	Lo				
0x11B0	Hi	<b>min_op_pll_op_freq_mhz</b>	RO	This register address is used in 2-PLL mode. See register <b>pll_mode</b> . Minimum PLL output clock frequency <b>Format:</b> IEEE 32-bit float or 32-bit unsigned iReal <b>Units:</b> MHz	NA
0x11B1	—				
0x11B2	—				
0x11B3	Lo				
0x11B4	Hi	<b>max_op_pll_op_freq_mhz</b>	RO	This register address is used in 2-PLL mode. See register <b>pll_mode</b> .	NA
0x11B5	—				
0x11B6	—				

<b>Index</b>	<b>Byte</b>	<b>Register Name</b>	<b>RW</b>	<b>Comment</b>	<b>Retime</b>
0x11B7	Lo			Maximum PLL output clock frequency <b>Format:</b> IEEE 32-bit float or 32-bit unsigned iReal <b>Units:</b> MHz	
0x11B8	-	<b>clock_tree_pll_capability</b>	RO	<b>Bit 0</b> 0: 2-PLL clock tree not supported 1: 2-PLL clock tree supported <b>Bit 1</b> 0: 1-PLL clock tree not supported 1: 1-PLL clock tree supported <b>Bit 2</b> 0: Extended PLL input divider values not supported 1: Extended PLL input divider values supported <b>Bit 3</b> 0: Legacy <b>op_pix_clk_div</b> values supported 1: Flexible <b>op_pix_clk_div</b> values supported	NA
0x11B9	-	<b>clock_capa_type_capability</b>	RO	added in CCS v1.1	NA

### 20.7.4.2 Video Timing Clock Set-up Capability Registers: 0x1120–0x1137 (RO)

4153

**Table 233 Video Timing Clock Set-up Capability Registers**

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

### 20.7.4.3 Clock Calculation Capability Registers: 0x1138–0x113B (RO)

4154

**Table 234 Clock Calculation Capability Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x1138	–	<b>clock_calculation</b>	RO	<b>Bit 0</b> 0: System speed clock calculation 1: Lane number definition <b>Bit 1</b> 0: OP clock calculation coupled with link 1: OP clock calculation decoupled from link <b>Bit 2</b> with 2-PLL clock tree for OP sys domain 0: single rate bitrate / Symbol rate 1: double rate bitrate / Symbol rate <b>Bit 3</b> with 2-PLL clock tree for OP pixel domain 0: single rate 1: double rate	NA
0x1139	–	<b>num_of_vt_lanes</b>	RO	Number of VT lanes (effective if <b>clock_calculation</b> bit 0 has value 1)	NA
0x113A	–	<b>num_of_op_lanes</b>	RO	Number of OP lanes (effective if <b>clock_calculation</b> bit 1 has value 1)	NA
0x113B	–	<b>op_bits_per_lane</b>	RO	Added in CCS v1.1	NA

#### 20.7.4.4 Frame Timing Parameter Limit Registers: 0x1140–0x114C (RO)

4155

Table 235 Frame Timing Capability Registers

Index	Byte	Register Name	RW	Comment	Retime
0x1140	Hi	<b>min_frame_length_lines</b>	RO	Minimum Frame Length allowed. <b>Format:</b> 16-bit unsigned integer <b>Units:</b> Lines	NA
0x1141	Lo				
0x1142	Hi	<b>max_frame_length_lines</b>	RO	Maximum possible number of lines per Frame. <b>Format:</b> 16-bit unsigned integer <b>Units:</b> Lines	NA
0x1143	Lo				
0x1144	Hi	<b>min_line_length_pck</b>	RO	Minimum Line Length allowed. <b>Format:</b> 16-bit unsigned integer <b>Units:</b> Pixel Clocks	NA
0x1145	Lo				
0x1146	Hi	<b>max_line_length_pck</b>	RO	Maximum possible number of pixel clocks per line. <b>Format:</b> 16-bit unsigned integer <b>Units:</b> Pixel Clocks	NA
0x1147	Lo				
0x1148	Hi	<b>min_line_blanketing_pck</b>	RO	Minimum line blanking time in pixel clocks <b>Format:</b> 16-bit unsigned integer <b>Units:</b> Pixel Clocks	NA
0x1149	Lo				
0x114A	Hi	<b>min_frame_blanketing_lines</b>	RO	Minimum frame blanking in video timing lines <b>Format:</b> 16-bit unsigned integer <b>OPTIONAL</b>	NA
0x114B	Lo				
0x114C	–	<b>min_line_length_pck_step_size</b>	RO	Minimum step size of line length pck. Typical value may be 1, 2, or 4. <b>Units:</b> Pixel Clocks	NA

### 20.7.4.5 Timing Capability Registers: 0x114D–0x1151 (RO)

4156

See *Section 9*.

4157

**Table 236 Timing Capability Registers**

<b>Index</b>	<b>Byte</b>	<b>Register Name</b>	<b>RW</b>	<b>Comment</b>	<b>Retime</b>
0x114D	–	<b>timing_mode_capability</b>	RO	<b>Bit 0</b> 1: Supports automatic frame length <b>Bit 1</b> Reserved <b>Bit 2</b> 1: Supports manual readout in rolling shutter <b>Bit 3</b> 1: Supports delayed Exposure start <b>Bit 4</b> 1: Supports storing Exposure time into embedded data in manual readout mode	NA
0x114E	Hi	<b>frame_margin_max_value</b>	RO	–	NA
0x114F	Lo				
0x1150	–	<b>frame_margin_min_value</b>	RO	–	NA
0x1151	–	<b>gain_delay_type</b>	RO	0: Fixed Gain delay 1: Variable Gain delay	NA

### 20.7.4.6 Output Clock Set-Up Capability Registers: 0x1160–0x1177 (RO)

4158

**Table 237 Output Clock Set-Up Capability Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x1160	Hi	<b>min_op_sys_clk_div</b>	RO	Minimum output system clock divider value <b>Format:</b> 16-bit unsigned integer	NA
0x1161	Lo				
0x1162	Hi	<b>max_op_sys_clk_div</b>	RO	Maximum output system clock divider value <b>Format:</b> 16-bit unsigned integer	NA
0x1163	Lo				
0x1164	Hi	<b>min_op_sys_clk_freq_mhz</b>	RO	Minimum output system clock frequency <b>Format:</b> IEEE 32-bit float or 32-bit unsigned iReal <b>Units:</b> MHz	NA
0x1165	–				
0x1166	–				
0x1167	Lo				
0x1168	Hi	<b>max_op_sys_clk_freq_mhz</b>	RO	Maximum output system clock frequency <b>Format:</b> IEEE 32-bit float or 32-bit unsigned iReal <b>Units:</b> MHz	NA
0x1169	–				
0x116A	–				
0x116B	Lo				
0x116C	Hi	<b>min_op_pix_clk_div</b>	RO	Minimum output pixel clock divider value <b>Format:</b> 16-bit unsigned integer	NA
0x116D	Lo				
0x116E	Hi	<b>max_op_pix_clk_div</b>	RO	Maximum output pixel clock divider value <b>Format:</b> 16-bit unsigned integer	NA
0x116F	Lo				
0x1170	Hi	<b>min_op_pix_clk_freq_mhz</b>	RO	Minimum output pixel clock frequency <b>Format:</b> IEEE 32-bit float or 32-bit unsigned iReal <b>Units:</b> MHz	NA
0x1171	–				
0x1172	–				
0x1173	Lo				
0x1174	Hi	<b>max_op_pix_clk_freq_mhz</b>	RO	Maximum output pixel clock frequency <b>Format:</b> IEEE 32-bit float or 32-bit unsigned iReal <b>Units:</b> MHz	NA
0x1175	–				
0x1176	–				
0x1177	Lo				

### 20.7.4.7 Image Size Parameter Limit Registers: 0x1180–0x119F (RO)

4159

Table 238 Image Size Capability Registers

Index	Byte	Name	RW	Comment	Retime
0x1180	Hi	<b>x_addr_min</b>	RO	Minimum X-address of the addressable pixel array <b>Format:</b> 16-bit unsigned integer <b>Value:</b> Always 0	NA
0x1181	Lo				
0x1182	Hi	<b>y_addr_min</b>	RO	Minimum Y-address of the addressable pixel array <b>Format:</b> 16-bit unsigned integer <b>Value:</b> Always 0	NA
0x1183	Lo				
0x1184	Hi	<b>x_addr_max</b>	RO	Maximum X-address of the addressable pixel array <b>Format:</b> 16-bit unsigned integer	NA
0x1185	Lo				
0x1186	Hi	<b>y_addr_max</b>	RO	Maximum Y-address of the addressable pixel array <b>Format:</b> 16-bit unsigned integer	NA
0x1187	Lo				
0x1188	Hi	<b>min_x_output_size</b>	RO	Minimum x output size in pixels <b>Format:</b> 16-bit unsigned integer	NA
0x1189	Lo				
0x118A	Hi	<b>min_y_output_size</b>	RO	Minimum y output size in pixels <b>Format:</b> 16-bit unsigned integer	NA
0x118B	Lo				
0x118C	Hi	<b>max_x_output_size</b>	RO	Maximum x output size in pixels <b>Format:</b> 16-bit unsigned integer	NA
0x118D	Lo				
0x118E	Hi	<b>max_y_output_size</b>	RO	Maximum y output size in pixels <b>Format:</b> 16-bit unsigned integer	NA
0x118F	Lo				
0x1190	—	<b>x_addr_start_div_constant</b>	RO	Added in CCS v1.1	NA
0x1191	—	<b>y_addr_start_div_constant</b>	RO	Added in CCS v1.1	NA
0x1192	—	<b>x_addr_end_div_constant</b>	RO	Added in CCS v1.1	NA
0x1193	—	<b>y_addr_end_div_constant</b>	RO	Added in CCS v1.1	NA
0x1194	—	<b>x_size_div</b>	RO	Added in CCS v1.1	NA
0x1195	—	<b>y_size_div</b>	RO	Added in CCS v1.1	NA
0x1196	—	<b>x_output_div</b>	RO	Added in CCS v1.1	NA
0x1197	—	<b>y_output_div</b>	RO	Added in CCS v1.1	NA
0x1198	—	<b>non_flexible_resolution_support</b>	RO	Added in CCS v1.1	NA

#### 20.7.4.8 Sub-Sampling Parameter and Readout Limit Registers: 0x11C0–0x11F3 (RO)

4160

Table 239 Sub-Sampling Capability Registers

Index	Byte	Name	RW	Comment	Retime
0x11C0	Hi	<b>min_even_inc</b>	RO	Minimum Increment for even pixels <b>Format:</b> 16-bit unsigned integer	NA
0x11C1	Lo				
0x11C2	Hi	<b>max_even_inc</b>	RO	Maximum increment for even pixels <b>Format:</b> 16-bit unsigned integer	NA
0x11C3	Lo				
0x11C4	Hi	<b>min_odd_inc</b>	RO	Minimum Increment for odd pixels <b>Format:</b> 16-bit unsigned integer	NA
0x11C5	Lo				
0x11C6	Hi	<b>max_odd_inc</b>	RO	Maximum Increment for odd pixels <b>Format:</b> 16-bit unsigned integer	NA
0x11C7	Lo				
0x11C8	–	<b>aux_subsamp_capability</b>	RO	Added in CCS v1.1	NA
0x11C9	–	<b>aux_subsamp_mono_capability</b>	RO	Added in CCS v1.1	NA
0x11CA	–	<b>monochrome_capability</b>	RO	Added in CCS v1.1	NA
0x11CB	–	<b>pixel_readout_capability</b>	RO	Added in CCS v1.1	NA
0x11CC	Hi	<b>min_even_inc_mono</b>	RO	Added in CCS v1.1	NA
0x11CD	Lo				
0x11CE	Hi	<b>max_even_inc_mono</b>	RO	Added in CCS v1.1	NA
0x11CF	Lo				
0x11D0	Hi	<b>min_odd_inc_mono</b>	RO	Added in CCS v1.1	NA
0x11D1	Lo				
0x11D2	Hi	<b>max_odd_inc_mono</b>	RO	Added in CCS v1.1	NA
0x11D3	Lo				
0x11D4	Hi	<b>min_even_inc_bc2</b>	RO	Added in CCS v1.1	NA
0x11D5	Lo				
0x11D6	Hi	<b>max_even_inc_bc2</b>	RO	Added in CCS v1.1	NA
0x11D7	Lo				
0x11D8	Hi	<b>min_odd_inc_bc2</b>	RO	Added in CCS v1.1	NA
0x11D9	Lo				
0x11DA	Hi	<b>max_odd_inc_bc2</b>	RO	Added in CCS v1.1	NA
0x11DB	Lo				
0x11DC	Hi	<b>min_even_inc_mono_bc2</b>	RO	Added in CCS v1.1	NA
0x11DD	Lo				

Index	Byte	Name	RW	Comment	Retime
0x11DE	Hi	<b>max_even_inc_mono_bc2</b>	RO	Added in CCS v1.1	NA
0x11DF	Lo				
0x11F0	Hi	<b>min_odd_inc_mono_bc2</b>	RO	Added in CCS v1.1	NA
0x11F1	Lo				
0x11F2	Hi	<b>max_odd_inc_mono_bc2</b>	RO	Added in CCS v1.1	NA
0x11F3	Lo				

### 20.7.5 Image Scaling Parameter Limit Registers: 0x1200–0x120F (RO)

4161

The Image Scaling Parameter Limit registers are fully described in *Section 11*.

4162

**Table 240 Image Scaling Capability Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x1200	Hi	<b>scaling_capability</b>	RO	0: None 1: Horizontal 2: Reserved <b>Format:</b> 16-bit unsigned integer	NA
0x1201	Lo				
0x1202	Hi	Reserved	RO	–	NA
0x1203	Lo				
0x1204	Hi	<b>scale_m_min</b>	RO	Down scale factor: Minimum M value. Value shall be the same as <b>scale_n_max</b> . <b>Format:</b> 16-bit unsigned integer	NA
0x1205	Lo				
0x1206	Hi	<b>scale_m_max</b>	RO	Down scale factor: Maximum M value <b>Format:</b> 16-bit unsigned integer	NA
0x1207	Lo				
0x1208	Hi	<b>scale_n_min</b>	RO	Down scale factor: Minimum N value. Vendor specific value. Value is typically 16. <b>Format:</b> 16-bit unsigned integer	NA
0x1209	Lo				
0x120A	Hi	<b>scale_n_max</b>	RO	Down scale factor: Maximum N value. Vendor specific value. Value is typically 16. <b>Format:</b> 16-bit unsigned integer	NA
0x120B	Lo				
0x120C	Hi	Reserved	RO	–	NA
0x120D	Lo				
0x120E		<b>digital_crop_capability</b>	RO	0: None 1: Input digital crop supported	NA

### 20.7.6 HDR Limit Registers: 0x1210–0x121F (RO)

4163

The HDR Limit registers are fully defined in *Section 9.9*.

4164

**Table 241 HDR Capability Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x1210	–	<b>HDR_capability_1</b>	RO	–	NA
0x1211	–	<b>min_HDR_bit_depth</b>	RO	–	NA
0x1212	–	<b>HDR_resolution_sub_types</b>	RO	–	NA
0x1213	–	<b>HDR_resolution_type_1</b>	RO	–	NA
0x1214	–	<b>HDR_resolution_type_2</b>	RO	–	NA
0x1215	–	<b>HDR_resolution_type_3</b>	RO	–	NA
0x1216	–	<b>HDR_resolution_type_4</b>	RO	–	NA
0x1217	–	<b>HDR_resolution_type_5</b>	RO	–	NA
0x1218	–	<b>HDR_resolution_type_6</b>	RO	–	NA
0x1219	–	<b>HDR_resolution_type_7</b>	RO	–	NA
0x121A	–	<b>HDR_resolution_type_8</b>	RO	–	NA
0x121B	–	<b>HDR_capability_2</b>	RO	–	NA
0x121C	–	<b>max_HDR_bit_depth</b>	RO	–	NA

### 20.7.7 USL Capability Registers: 0x1230–0x126F (RO)

4165

The USL capability registers are fully defined in *Annex D*.

4166

**Table 242 USL Capability Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x1230	–	<b>USL_support_capability</b>	RO	Added in CCS v.1.1	NA
0x1231	–	<b>USL_Clock_mode_D_capability</b>	RO	Added in CCS v.1.1	NA
0x1232	–	Reserved	NA	Reserved for future use	NA
0x1233	–	Reserved	NA	Reserved for future use	NA
0x1234	Hi	<b>min_op_sys_clk_div_rev</b>	RO	Added in CCS v.1.1	NA
0x1235	Lo				
0x1236	Hi	<b>max_op_sys_clk_div_rev</b>	RO	Added in CCS v.1.1	NA
0x1237	Lo				
0x1238	Hi	<b>min_op_pix_clk_div_rev</b>	RO	Added in CCS v.1.1	NA
0x1239	Lo				
0x123A	Hi	<b>max_op_pix_clk_div_rev</b>	RO	Added in CCS v.1.1	NA
0x123B	Lo				
0x123C	Hi	<b>min_op_sys_clk_freq_rev_mhz</b>	RO	Added in CCS v.1.1	NA
0x123D	–				
0x123E	–				
0x123F	Lo				

<b>Index</b>	<b>Byte</b>	<b>Register Name</b>	<b>RW</b>	<b>Comment</b>	<b>Retime</b>
0x1240	Hi	<b>max_op_sys_clk_freq_rev_mhz</b>	RO	Added in CCS v.1.1	NA
0x1241	–				
0x1242	–				
0x1243	Lo				
0x1244	Hi	<b>min_op_pix_clk_freq_rev_mhz</b>	RO	Added in CCS v.1.1	NA
0x1245	–				
0x1246	–				
0x1247	Lo				
0x1248	Hi	<b>max_op_pix_clk_freq_rev_mhz</b>	RO	Added in CCS v.1.1	NA
0x1249	–				
0x124A	–				
0x124B	Lo				
0x124C	Hi	<b>max_bitrate_rev_d_mode_mbps</b>	RO	Added in CCS v.1.1	NA
0x124D	–				
0x124E	–				
0x124F	Lo				
0x1250	Hi	<b>max_symbolrate_rev_c_mode_msps</b>	RO	Added in CCS v.1.1	NA
0x1251	–				
0x1252	–				
0x1253	Lo				
0x1254	–	Reserved	NA	Reserved for future	NA
0x1255	–	Reserved	NA	Reserved for future	NA
0x1256	–	Reserved	NA	Reserved for future	NA
0x1257	–	Reserved	NA	Reserved for future	NA
0x1258	Hi	<b>LRTE_usl_cphy_hs_rx_capability</b>	RO	Added in CCS v1.1	NA
0x1259	Lo				
0x125A	Hi	<b>LRTE_usl_cphy_lp_rx_capability</b>	RO	Added in CCS v1.1	NA
0x125B	Lo				
0x125C	Hi	<b>LRTE_usl_dphy_hs_rx_capability_1</b>	RO	Added in CCS v1.1	NA
0x125D	Lo				
0x125E	Hi	<b>LRTE_usl_dphy_hs_rx_capability_2</b>	RO	Added in CCS v1.1	NA
0x125F	Lo				
0x1260	Hi	<b>LRTE_usl_dphy_lp_rx_capability</b>	RO	Added in CCS v1.1	NA
0x1261	Lo				
0x1262	Hi	<b>LRTE_usl_rx_max_burst_size</b>	RO	Added in CCS v1.1	NA
0x1263	Lo				

### 20.7.8 Image Compression Capability Registers: 0x1300–0x1301 (RO)

4167 The Image Compression Capability registers are fully described in *Section 12*.

**Table 243 Image Compression Capability Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x1300	Hi	compression_capability	RO	0: No Compression 1: DPCM/PCM Compression	NA
0x1301	Lo				

### 20.7.9 Test Mode Capability Registers: 0x1310–0x1318 (RO)

4168 The Test Mode Capability registers are fully defined in *Section 10*.

**Table 244 Test Mode Capability Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x1310	Hi	test_mode_capability	RO	–	NA
0x1311	Lo				
0x1312	–	PN9_data_format1	RO	–	NA
0x1313	–	PN9_data_format2	RO	–	NA
0x1314	–	PN9_data_format3	RO	–	NA
0x1315	–	PN9_data_format4	RO	–	NA
0x1316	–	PN9_misc_capability	RO	–	NA
0x1317	–	test_pattern_capability	RO	Added in CCS v1.1	NA
0x1318	–	pattern_size_div_m1	RO	Added in CCS v1.1	NA

### 20.7.10 FIFO Capability Registers: 0x1500–0x1502 (RO)

4170 The FIFO Capability registers are fully described in *Section 11.6*.

**Table 245 FIFO Capability Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x1500	–	Reserved	–	Reserved	NA
0x1501	–	Reserved	–	Reserved	NA
0x1502	–	fifo_support_capability	RO	0: Not supported 1: Supports derating 2: Supports both derating and overrating	NA

### 20.7.11 CSI-2 Capability Registers: 0x1600–0x16FF (RO)

4172 The CSI-2 Capability registers are fully described in *Section 7.9*.

4173

**Table 246 CSI-2 Capability Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x1600	–	<b>PHY_ctrl_capability</b>	RO	<b>Bit 0</b> 1: Automatic PHY control supported <b>Bit 1</b> 1: UI based PHY control supported <b>Bit 2</b> 1: D-PHY control by the non-extended “Time and UI Values Register 1”) is supported <b>Bit 3</b> 1: D-PHY control by the non-extended “Time and UI Values Register 2”) is supported <b>Bit 4</b> 1: D-PHY control by the non-extended “Time values” register is supported <b>Bit 5</b> 1: D-PHY control by the registers for extended “Time and UI Values Register 1” is supported <b>Bit 6</b> 1: D-PHY control by the registers for extended “Time and UI Values Register 2” is supported <b>Bit 7</b> 1: D-PHY control by the registers for extended “Time value” is supported <b>All Bits:</b> 0: Not supported	NA

Index	Byte	Register Name	RW	Comment	Retime
0x1601	–	<b>CSI_DPHY_lane_mode_capability</b>	RO	<b>Bit 0</b> 1 : 1-Lane supported <b>Bit 1</b> 1 : 2-Lane supported <b>Bit 2</b> 1 : 3-Lane supported <b>Bit 3</b> 1 : 4-Lane supported <b>Bit 4</b> 1 : 5-Lane supported <b>Bit 5</b> 1 : 6-Lane supported <b>Bit 6</b> 1 : 7-Lane supported <b>Bit 7</b> 1 : 8-Lane supported <b>All Bits:</b> 0: Not supported	NA
0x1602	–	<b>CSI_signaling_mode_capability</b>	RO	<b>Bits 0 &amp; 1</b> Reserved <b>Bit 2</b> 1: CSI-2 with D-PHY supported 0: Not supported. <b>Bit 3</b> 1: CSI-2 with C-PHY supported 0: Not supported.	NA
0x1603	–	<b>fast_standby_capability</b>	RO	0: Frame truncation not supported for rolling shutter 1: Frame truncation supported for rolling shutter	NA
0x1604	–	<b>CCI_address_ctrl_capability</b>	RO	<b>Bit 0</b> 1: SW changeable CCI address supported 0: Not supported <b>Bit 1</b> 1: 2 <sup>nd</sup> CCI address supported 0: Not supported <b>Bit 2</b> 1: SW changeable 2 <sup>nd</sup> CCI address supported 0: Not supported <b>Bits 3-7</b> Reserved for future use	NA

Index	Byte	Register Name	RW	Comment	Retime
0x1605	–	<b>data_type_capability</b>	RO	<b>Bit 0</b> 1: DPCM data type is programmable <b>Bit 1</b> 1: Bottom embedded data type is programmable <b>Bit 2</b> 1: Bottom embedded data virtual channel is programmable. <b>Bit 3</b> 1: Extended virtual channel range is supported.	NA
0x1606	–	<b>CSI_CPHY_lane_mode_capability</b>	RO	<b>Bit 0</b> 1: 1-Lane supported <b>Bit 1</b> 1: 2-Lane supported <b>Bit 2</b> 1: 3-Lane supported <b>Bit 3</b> 1: 4-Lane supported <b>Bit 4</b> 1: 5-Lane supported <b>Bit 5</b> 1: 6-Lane supported <b>Bit 6</b> 1: 7-Lane supported <b>Bit 7</b> 1: 8-Lane supported <b>All Bits:</b> 0: Not supported	NA
0x1607	–	<b>emb_data_capability</b>	RO	Added in CCS v1.1	NA
0x1608	Hi	<b>max_per_lane_bitrate_1_lane_d_mode_mbps</b>	RO	–	NA
0x1609	–				
0x160A	–				
0x160B	Lo				
0x160C	Hi	<b>max_per_lane_bitrate_2_lane_d_mode_mbps</b>	RO	–	NA
0x160D	–				
0x160E	–				
0x160F	Lo				
0x1610	Hi	<b>max_per_lane_bitrate_3_lane_d_mode_mbps</b>	RO	–	NA
0x1611	–				
0x1612	–				
0x1613	Lo				

Index	Byte	Register Name	RW	Comment	Retime
0x1614	Hi	<b>max_per_lane_bitrate_4_lane_d_mode_mbps</b>	RO	–	NA
0x1615	–				
0x1616	–				
0x1617	Lo				
0x1618	–	<b>temp_sensor_capability</b>	RO	–	NA
0x1619	–	Reserved	RO	–	NA
0x161A	Hi	<b>max_per_lane_symbolrate_1_lane_c_mode_msps</b>	RO	–	NA
0x161B	–				
0x161C	–				
0x161D	Lo				
0x161E	Hi	<b>max_per_lane_symbolrate_2_lane_c_mode_msps</b>	RO	–	NA
0x161F	–				
0x1620	–				
0x1621	Lo				
0x1622	Hi	<b>max_per_lane_symbolrate_3_lane_c_mode_msps</b>	RO	–	NA
0x1623	–				
0x1624	–				
0x1625	Lo				
0x1626	Hi	<b>max_per_lane_symbolrate_4_lane_c_mode_msps</b>	RO	–	NA
0x1627	–				
0x1628	–				
0x1629	Lo				
0x162A	–	Reserved	–	–	NA
0x162B	–	<b>DPHY_equalization_capability</b>	RO	Added in CCS v1.1	NA
0x162C	–	<b>CPHY_equalization_capability</b>	RO	Added in CCS v1.1	NA
0x162D	–	<b>DPHY_preamble_capability</b>	RO	Added in CCS v1.1	NA
0x162E	–	<b>DPHY_SSC_capability</b>	RO	Added in CCS v1.1	NA
0x162F	–	<b>CPHY_calibration_capability</b>	RO	Added in CCS v1.1	NA
0x1630	–	<b>DPHY_calibration_capability</b>	RO	–	NA
0x1631	–	<b>PHY_ctrl_capability_2</b>	RO	–	NA
0x1632	–	<b>LRTE_cphy_capability</b>	RO	–	NA
0x1633	–	<b>LRTE_dphy_capability</b>	RO	–	NA
0x1634	–	<b>ALPS_dphy_capability</b>	RO	–	NA
0x1635	–	<b>ALPS_cphy_capability</b>	RO	–	NA
0x1636	–	<b>scrambling_capability</b>	RO	–	NA
0x1637	–	<b>dphy_manual_constant</b>	RO	–	NA
0x1638	–	<b>cphy_manual_constant</b>	RO	–	NA
0x1639	–	<b>CSI2_interface_capability_misc</b>	RO	Added in CCS v1.1	NA

<b>Index</b>	<b>Byte</b>	<b>Register Name</b>	<b>RW</b>	<b>Comment</b>	<b>Retime</b>
0x163A	Hi	<b>max_per_lane_bitrate_5_lane_d_mode_mbps</b>	RO	–	NA
0x163B	–				
0x163C	–				
0x163D	Lo				
0x163E	Hi	<b>max_per_lane_bitrate_6_lane_d_mode_mbps</b>	RO	–	NA
0x163F	–				
0x1640	–				
0x1641	Lo				
0x1642	Hi	<b>max_per_lane_bitrate_7_lane_d_mode_mbps</b>	RO	–	NA
0x1643	–				
0x1644	–				
0x1645	Lo				
0x1646	Hi	<b>max_per_lane_bitrate_8_lane_d_mode_mbps</b>	RO	–	NA
0x1647	–				
0x1648	–				
0x1649	Lo				
0x164A	Hi	<b>max_per_lane_symbolrate_5_lane_c_mode_msps</b>	RO	–	NA
0x164B	–				
0x164C	–				
0x164D	Lo				
0x1650	Hi	<b>max_per_lane_symbolrate_6_lane_c_mode_msps</b>	RO	–	NA
0x1651	–				
0x1652	–				
0x1653	Lo				
0x1654	Hi	<b>max_per_lane_symbolrate_7_lane_c_mode_msps</b>	RO	–	NA
0x1655	–				
0x1656	–				
0x1657	Lo				
0x1658	Hi	<b>max_per_lane_symbolrate_8_lane_c_mode_msps</b>	RO	–	NA
0x1659	–				
0x165A	–				
0x165B	Lo				
0x165C	–	<b>PHY_ctrl_capability_3</b>	RO	Added in CCS v1.1	NA
0x165D	–	<b>dphy_sf</b>	RO	Added in CCS v1.1	NA
0x165E	–	<b>cphy_sf</b>	RO	Added in CCS v1.1	NA
0x165F	–	<b>dphy_limits_1</b>	RO	Added in CCS v1.1	NA
0x1660	–	<b>dphy_limits_2</b>	RO	Added in CCS v1.1	NA
0x1661	–	<b>dphy_limits_3</b>	RO	Added in CCS v1.1	NA
0x1662	–	<b>dphy_limits_4</b>	RO	Added in CCS v1.1	NA

Index	Byte	Register Name	RW	Comment	Retime
0x1663	–	<b>dphy_limits_5</b>	RO	Added in CCS v1.1	NA
0x1664	–	<b>dphy_limits_6</b>	RO	Added in CCS v1.1	NA
0x1665	–	<b>cphy_limits_1</b>	RO	Added in CCS v1.1	NA
0x1666	–	<b>cphy_limits_2</b>	RO	Added in CCS v1.1	NA
0x1667	–	<b>cphy_limits_3</b>	RO	Added in CCS v1.1	NA

### 20.7.12 Binning Capability Registers: 0x1700–0x17FF (RO)

4174

The Binning Capability registers are fully described in *Section 8.2.5*.

4175

**Table 247 Binning Capability Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x1700	Hi	<b>min_frame_length_lines_bin</b>	RO	–	NA
0x1701	Lo				
0x1702	Hi	<b>max_frame_length_lines_bin</b>	RO	–	NA
0x1703	Lo				
0x1704	Hi	<b>min_line_length_pck_bin</b>	RO	–	NA
0x1705	Lo				
0x1706	Hi	<b>max_line_length_pck_bin</b>	RO	–	NA
0x1707	Lo				
0x1708	Hi	<b>min_line_blanketing_pck_bin</b>	RO	–	NA
0x1709	Lo				
0x170A	Hi	<b>fine_integration_time_min_bin</b>	RO	–	NA
0x170B	Lo				
0x170C	Hi	<b>fine_integration_time_max_margin_bin</b>	RO	–	NA
0x170D	Lo				
0x170E	Hi	Reserved	–	–	NA
0x170F	Lo				
0x1710	–	<b>binning_capability</b>	RO	–	NA
0x1711	–	<b>binning_weighting_capability</b>	RO	–	NA
0x1712	–	<b>binning_sub_types</b>	RO	–	NA
0x1713	–	<b>binning_type_1</b>	RO	–	NA
0x1714	–	<b>binning_type_2</b>	RO	–	NA
...		(Use as many types as supported, up to a limit of 64)	RO		–
0x1751	–	<b>binning_type_63</b>	RO	–	NA
0x1752	–	<b>binning_type_64</b>	RO	–	NA
0x1753-0x1770	-	Reserved for future use	–	–	NA
0x1771	–	<b>binning_weighting_mono_capability</b>	RO	Added in CCS v1.1	NA
0x1772	–	<b>binning_sub_types_mono</b>	RO	Added in CCS v1.1	NA
0x1773	–	<b>binning_type_1_mono</b>	RO	Added in CCS v1.1	NA
0x1774	–	<b>binning_type_2_mono</b>	RO	Added in CCS v1.1	NA
...		(Use as many types as supported, up to a limit of 64)	RO	Added in CCS v1.1	

Index	Byte	Register Name	RW	Comment	Retime
0x17B1	–	<b>binning_type_63_mono</b>	RO	Added in CCS v1.1	NA
0x17B2	–	<b>binning_type_64_mono</b>	RO	Added in CCS v1.1	NA

### 20.7.13 Data Transfer Interface Capability Registers: 0x1800–18FF (RO)

4176

The Data Transfer Interface Capability registers are fully described in *Section 13*.

4177

**Table 248 Data Transfer Interface Capability Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x1800	–	<b>data_transfer_if_capability</b>	RO	–	NA

## 20.7.14 Image Processing and Sensor Correction Capability Registers: 0x1900–0x19FF (RO)

### 20.7.14.1 Sensor Correction Capability Registers: 0x1900–0x1908 (RO)

4178 The Sensor Correction Capability registers are fully described in *Section 14*.

4179 **Table 249 Sensor Correction Capability Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x1900	–	shading_correction_capability	RO	–	NA
0x1901	–	green_imbalance_capability	RO	–	NA
0x1902	–	Reserved	RO	–	NA
0x1903	–	module_specific_correction_capability	RO	–	NA
0x1904	Hi	defect_correction_capability	RO	–	NA
0x1905	Lo				
0x1906	Hi	defect_correction_capability_2	RO	–	NA
0x1907	Lo				
0x1908	–	NF_capability	RO	–	NA

### 20.7.14.2 Optical Black Readout Capability Registers: 0x1980–0x1981 (RO)

4180 The Optical Black Readout Capability registers are fully described in *Section 14.9*

4181 **Table 250 Optical Black Readout Capability Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x1980	–	OB_readout_capability	RO	<b>Bit 0</b> 1: Controllable OB pixel readout is supported <b>Bit 1</b> 1: Readout as part of visible pixel data channel. <b>Bit 2</b> 1: Readout with different virtual channel <b>Bit 3</b> 1: Readout with different data type <b>Bit 4</b> 1: Data format for OB pixels is programmable in interleaved readout mode	NA

### 20.7.14.3 Color Feedback Capability Registers: 0x1987 (RO)

4182 The Color Feedback Capability registers are described in *Section 14.8*.

4183 **Table 251 Color Feedback Capability Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x1987	–	color_feedback_capability	RO	–	NA

#### 20.7.14.4 CFA Pattern Capability Registers: 0x1990–0x1991 (RO)

4184 The CFA Pattern Capability registers are described in *Annex C*.

4185 **Table 252 CFA Pattern Capability Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x1990	–	<b>CFA_pattern_capability</b>	RO	Added in CCS v1.1	NA
0x1991	–	<b>CFA_pattern_conversion_capability</b>	RO	Added in CCS v1.1	NA

#### 20.7.15 Timer Capability Registers: 0x1A00–0x1A03 (RO)

4186 The Timer Capability registers are fully described in *Section 16*.

4187 **Table 253 Timer Capability Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x1A00	Hi	Reserved	RO	–	NA
0x1A01	Lo				
0x1A02	–	<b>flash_mode_capability</b>	RO	–	NA
0x1A03	–	<b>SA_strobe_mode_capability</b>	RO	–	NA

#### 20.7.16 Soft Reset Capability Registers: 0x1A10–0x1A11 (RO)

4188 The Soft Reset Capability registers are fully described in *Section 5.2.6*.

4189 **Table 254 Timer Capability Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x1A10	–	<b>reset_max_delay</b>	RO	Added in CCS v1.1	NA
0x1A11	–	<b>reset_min_time</b>	RO	Added in CCS v1.1	NA

4190

#### 20.7.17 PDAF Capability Registers: 0x1B80–0x1B8F (RO)

4191 The PDAF Capability registers are described in *Section 17*.

4192 **Table 255 PDAF Capability Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x1B80	–	<b>PDAF_capability</b>	RO	–	NA
0x1B81	–	<b>PDAF_capability_2</b>	RO	–	NA

#### 20.7.18 Bracketing Interface Capability Registers: 0x1C00–0x1CFF (RO)

4193 The Bracketing Interface Capability registers are fully described in *Section 9.8*.

4194 **Table 256 Bracketing Interface Capability Registers**

Index	Byte	Register Name	RW	Comment	Retime
0x1C00	–	<b>bracketing_LUT_capability_1</b>	RO	–	NA
0x1C01	–	<b>bracketing_LUT_capability_2</b>	RO	–	NA
0x1C02	–	<b>bracketing_LUT_size</b>	RO	–	NA

## 20.8 Image Statistics Registers: 0x2000–0x2FFF

4195 Register indices 0x2000 to 0x2FFF are reserved for future use by image statistics features.

## 20.9 Manufacturer Specific Registers: 0x3000–0xFFFF

4196 Register indices 0x3000 to 0xFFFF are reserved for manufacturer specific features or set-up.

## Annex A Additional Examples

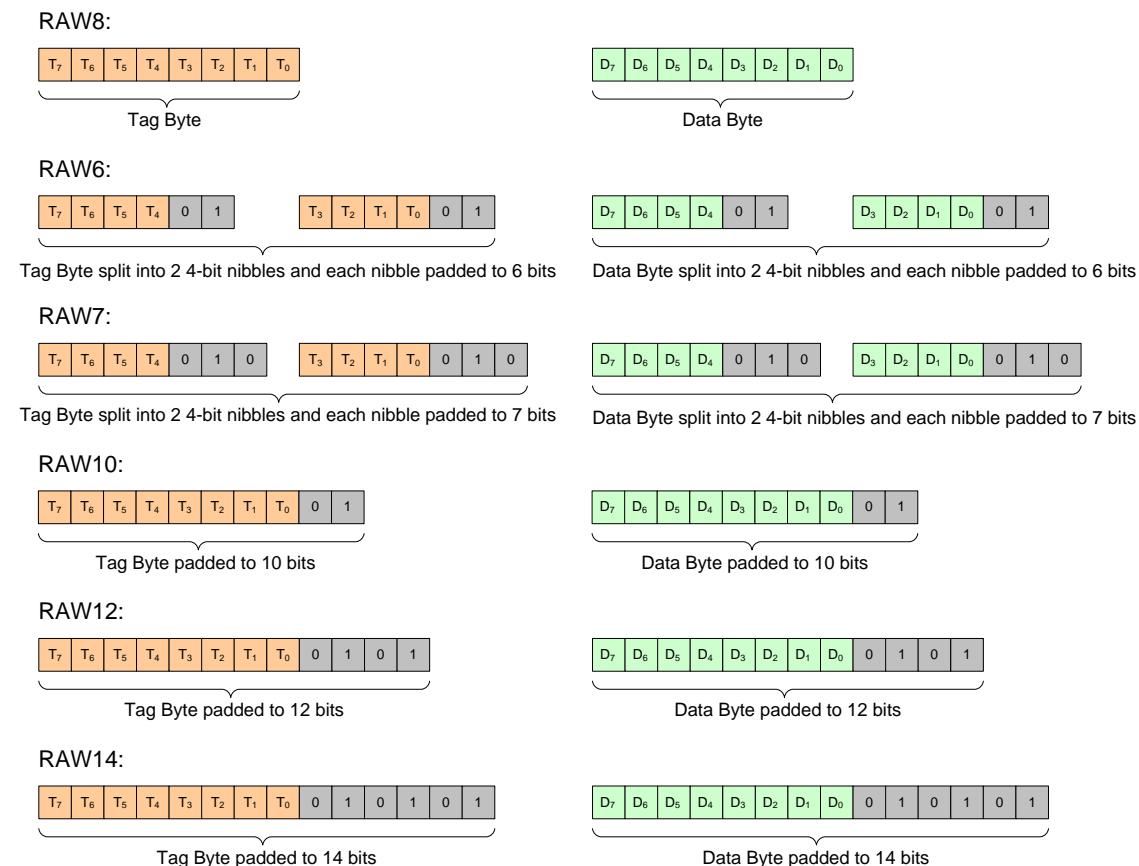
### A.1 Data Format

#### A.1.1 Embedded Data Packing for RAW6, RAW7, RAW12, and RAW14

The Section defines how embedded data is transmitted across the RAW6, RAW7, RAW12, and RAW14 data formats. The RAW6, RAW7, RAW12, and RAW14 embedded data uses the same tag codes as the RAW8 and RAW10 variants.

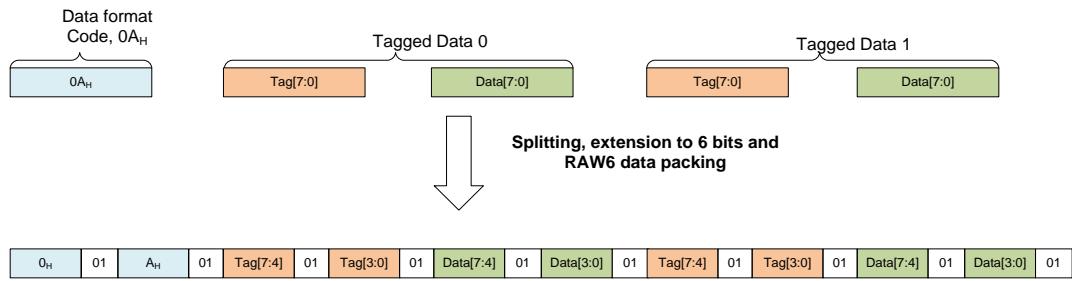
For the RAW6 and RAW7 embedded data formats, each byte of data is split into two 4-bit nibbles, padded to either 6 bits or 7 bits, respectively, and then transmitted over 2 pixels. Since this packing means that 4 pixel equivalents are needed to transmit one tag-data pair (whereas 2 pixel equivalents are used in RAW8, for example), the camera module can assume that the Host will not set a payload line length less than 640 pixels when using RAW6 or RAW7 formats.

For the RAW12 embedded data, each byte of data is padded to 12 bits and then transmitted as one pixel. For the RAW14 embedded data, each byte of data is padded to 14 bits and then transmitted as a one pixel.



**Figure 78 2-Byte Tagged Data Packing for RAW6, 7, 8, 10, 12, and 14**

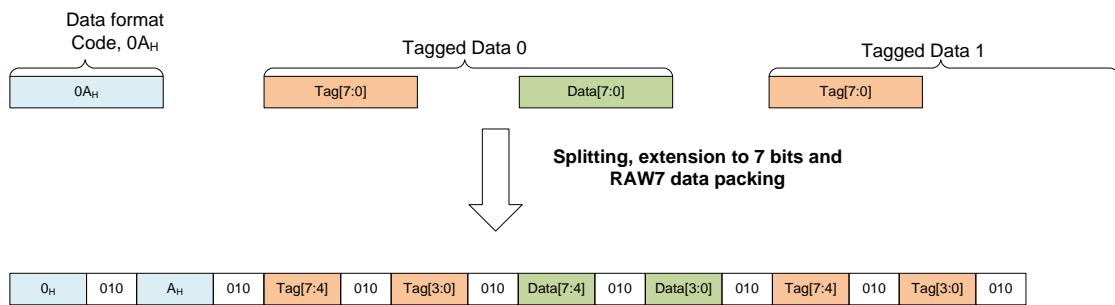
**3-bit tags and 8-bit data are split to 4bits and then extended to 6bits**



4208

**Figure 79 RAW6 Embedded Data Packing**

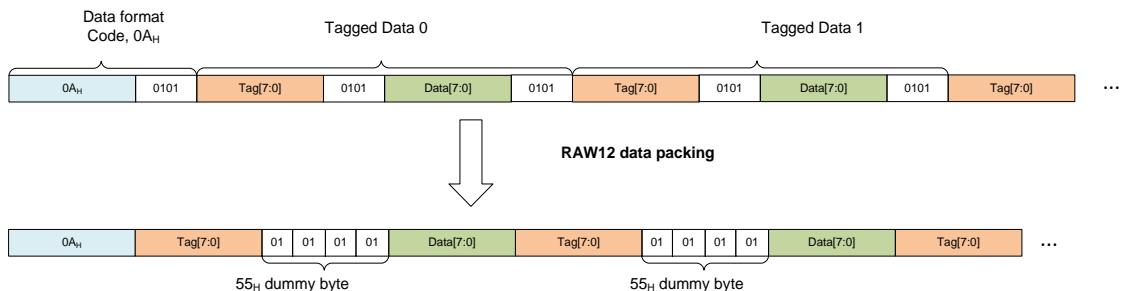
**3-bit tags and 8-bit data are split to 4bits and then extended to 7bits**



4209

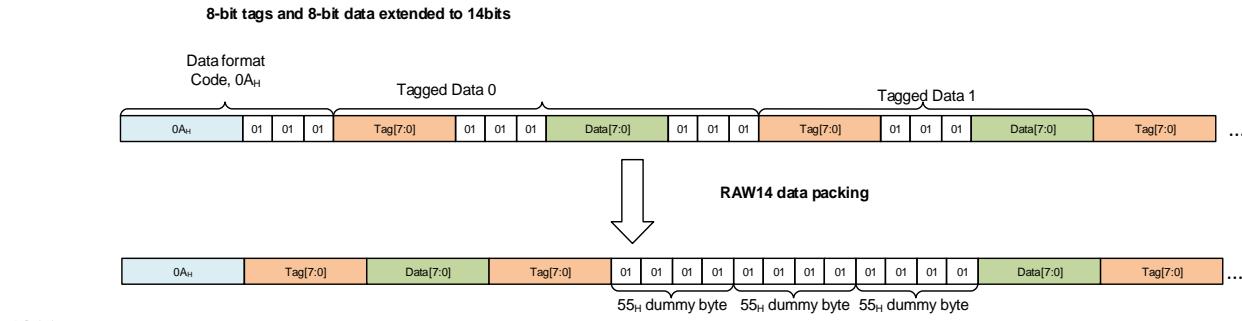
**Figure 80 RAW7 Embedded Data Packing**

**8-bit tags and 8-bit data extended to 12bits**



4210

**Figure 81 RAW12 Embedded Data Packing**

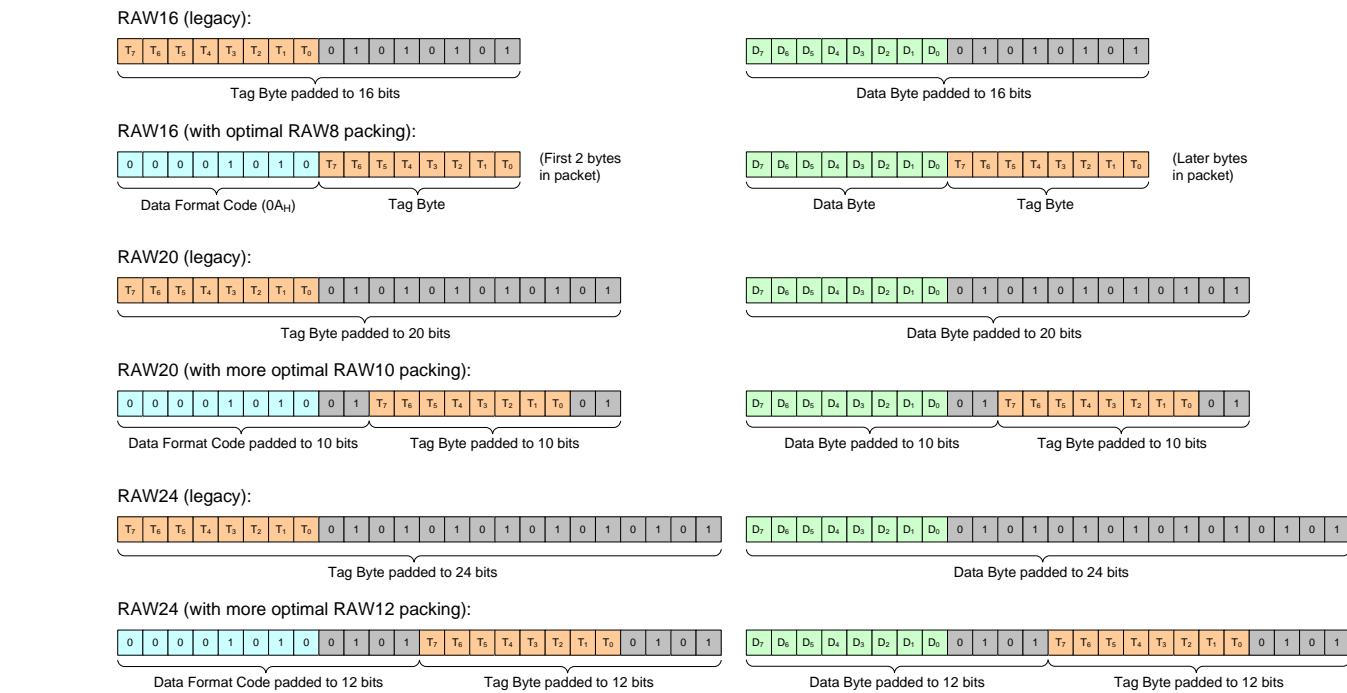


4211

**Figure 82 RAW14 Embedded Data Packing****A.1.2 Embedded Data Packing for RAW16, RAW20, and RAW24**

4212 **Figure 83** illustrates how the two-byte tagged data format may be transported using the RAW16, RAW20, and RAW24 pixel data formats with either legacy (i.e., one byte per pixel) or optimized (i.e., two bytes per pixel) packing conventions.

4213 With legacy packing, 50-67% of the bits in each pixel contain padding which for some use cases may severely  
4214 limit the number of bytes which may be transported within a single embedded data line. The more optimal  
4215 packing options introduced in **Section 7.15.1** reduce padding overhead to 0%, 20%, and 33% for RAW16,  
4216 RAW20, and RAW24, respectively.



4219

**Figure 83 2-Byte Tagged Data Packing for RAW16, 20 and 24**

### A.1.3 Embedded Data Example Using Simplified 2-Byte Tagged Data Format

This Section shows examples of embedded data content using the simplified 2-byte tagged data format. This format uses tag+data formatting, where the tag indicates the purpose of the data byte. Simplified 2-byte tagged data format is specified in [Section 7.15.1](#).

In this Section, two different examples are shown.

- [Table 257](#) specifies an example image sensor with a typical feature set. Because the feature set impacts upon the embedded data content, [Table 257](#) is valid only for image sensors with the same feature set. The Comment column indicates any required features.
- A second, more complex example is included in [Table 258](#) in order to show what additional fields must be added to the embedded data when the image sensor includes additional features.

In [Table 257](#) and [Table 258](#), the columns are:

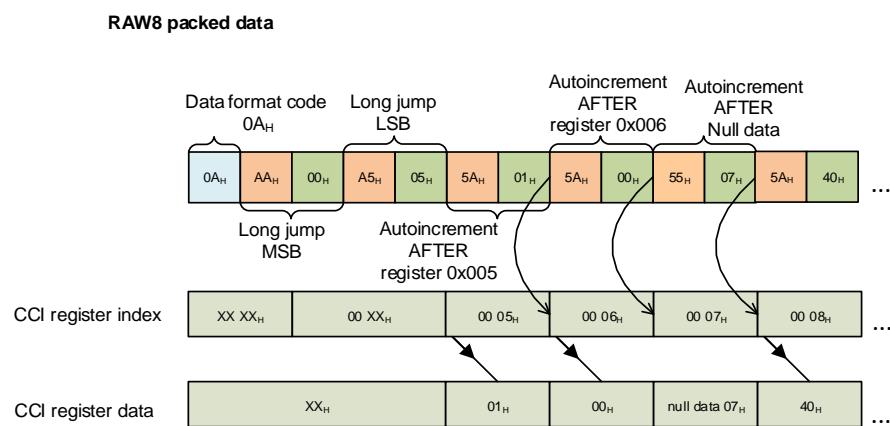
- **Embedded Data Byte Number:** The byte number (index) within the embedded data
- **Content of Embedded Data Bytes:** Each embedded byte is either Embedded data format code, tag, register address, data of certain register, or null data. For embedded data format codes, see [Table 63](#).
- **Value of Embedded Data Bytes:** column shows values of tag, register name in case of the data comes from certain register or null data.
- **Tag Code:** Tag code for data bytes. For tag codes for simplified 2-byte tagged data, see [Table 65](#).

Note that both [Table 257](#) and [Table 258](#) show the data before data packing occurs. Embedded data packing examples are shown in [Section A.1.1](#).

Top-embedded data shall use the same data packing as the visible pixel data; e.g. if the visible pixels use RAW10, then the top-embedded data shall also be packed as RAW10.

If the visible pixels are compressed, then the top-embedded data shall not be compressed; however the top-embedded data shall be sent with the same packing that the compressed data uses; e.g., with 10-bit-to-6-bit compression, RAW6 packing shall be used.

[Figure 84](#) shows the first bytes of embedded data for the [Table 257](#) example if the RAW8 data format is used.



**Figure 84 Embedded Data Example**

4247

**Table 257 Simple Example of Embedded Data Content**

Embedded Data Byte Number	Content of Embedded Data Bytes	Value of Embedded Data Bytes	Tag Code	Comment
0	Embedded data format code	0a	—	—
1	Tag	aa	INDEX_MSB	—
2	Address	00	—	—
3	Tag	a5	INDEX_LSB	—
4	Address	05	—	—
5	Tag	5a	AutoIncrement	—
6	Register 0x0005	<b>frame_count</b>	—	—
7	Tag	5a	AutoIncrement	—
8	Register 0x0006	<b>pixel_order</b>	—	—
9	Tag	55	AutoInc. DataNull	—
10	Null data	07	—	—
11	Tag	5a	AutoIncrement	—
12	Register 0x0008	<b>data_pedestal</b>	—	—
13	Tag	5a	AutoIncrement	—
14	Register 0x0009	<b>data_pedestal</b>	—	—
15	Tag	aa	INDEX_MSB	—
16	Address	01	—	—
17	Tag	a5	INDEX_LSB	—
18	Address	01	—	—
19	Tag	5a	AutoIncrement	—
20	Register 0x0101	<b>image_orientation</b>	—	—
21	Tag	a5	INDEX_LSB	—
22	Address	12	—	—
23	Tag	5a	AutoIncrement	—
24	Register 0x0112	<b>CSI_data_format</b>	—	—
25	Tag	5a	AutoIncrement	—
26	Register 0x0113	<b>CSI_data_format</b>	—	—
27	Tag	a5	INDEX_LSB	—
28	Address	38	—	—
29	Tag	5a	AutoIncrement	—
30	Register 0x0138	<b>temp_sensor_ctrl</b>	—	Required if temperature sensor supported
31	Tag	55	AutoInc. DataNull	—
32	Null data	07	—	—
33	Tag	5a	AutoIncrement	—

Embedded Data Byte Number	Content of Embedded Data Bytes	Value of Embedded Data Bytes	Tag Code	Comment
34	Register 0x013A	<b>temp_sensor_output</b>	—	Required if temperature sensor supported
35	Tag	aa	INDEX_MSB	—
36	Address	02	—	—
37	Tag	a5	INDEX_LSB	—
38	Address	00	—	—
39	Tag	5a	AutoIncrement	—
40	Register 0x0200	<b>fine_integration_time</b>	—	Required if fine Integration Time control supported
41	Tag	5a	AutoIncrement	—
42	Register 0x0201	<b>fine_integration_time</b>	—	Required if fine Integration Time control supported
43	Tag	5a	AutoIncrement	—
44	Register 0x0202	<b>coarse_integration_time</b>	—	—
45	Tag	5a	AutoIncrement	—
46	Register 0x0203	<b>coarse_integration_time</b>	—	—
47	Tag	5a	AutoIncrement	—
48	Register 0x0204	<b>analog_gain_code_global</b>	—	—
49	Tag	5a	AutoIncrement	—
50	Register 0x0205	<b>analog_gain_code_global</b>	—	—
51	Tag	a5	INDEX_LSB	—
52	Address	0E	—	—
53	Tag	5a	AutoIncrement	—
54	Register 0x020E	<b>digital_gain_global</b>	—	Required if digital Gain global supported
55	Tag	5a	AutoIncrement	—
56	Register 0x020F	<b>digital_gain_global</b>	—	Required if digital Gain global supported
57	Tag	aa	INDEX_MSB	—
58	Address	03	—	—
59	Tag	a5	INDEX_LSB	—
60	Address	02	—	—
61	Tag	5a	AutoIncrement	—
62	Register 0x0302	<b>vt_sys_clk_div</b>	—	This clock tree assumes a 2-PLL clock tree
63	Tag	5a	AutoIncrement	
64	Register 0x0303	<b>vt_sys_clk_div</b>	—	

Embedded Data Byte Number	Content of Embedded Data Bytes	Value of Embedded Data Bytes	Tag Code	Comment
65	Tag	5a	AutoIncrement	
66	Register 0x0304	<b>vt_pre_pll_clk_div</b>	—	
67	Tag	5a	AutoIncrement	
68	Register 0x0305	<b>vt_pre_pll_clk_div</b>	—	
69	Tag	5a	AutoIncrement	
70	Register 0x0306	<b>vt_pll_multiplier</b>	—	
71	Tag	5a	AutoIncrement	
72	Register 0x0307	<b>vt_pll_multiplier</b>	—	
73	Tag	5a	AutoIncrement	
74	Register 0x0308	<b>op_pix_clk_div</b>	—	
75	Tag	5a	AutoIncrement	
76	Register 0x0309	<b>op_pix_clk_div</b>	—	
77	Tag	5a	AutoIncrement	
78	Register 0x030A	<b>op_sys_clk_div</b>	—	
79	Tag	5a	AutoIncrement	
80	Register 0x030B	<b>op_sys_clk_div</b>	—	
81	Tag	5a	AutoIncrement	
82	Register 0x030C	<b>op_pre_pll_clk_div</b>	—	
83	Tag	5a	AutoIncrement	
84	Register 0x030D	<b>op_pre_pll_clk_div</b>	—	
85	Tag	5a	AutoIncrement	
86	Register 0x030E	<b>op_pll_multiplier</b>	—	
87	Tag	5a	AutoIncrement	
88	Register 0x030F	<b>op_pll_multiplier</b>	—	
89	Tag	5a	AutoIncrement	
90	Register 0x0310	<b>pll_mode</b>	—	
91	Tag	a5	INDEX_LSB	—
92	Address	40	—	—
93	Tag	5a	AutoIncrement	—
94	Register 0x0340	<b>frame_length_lines</b>	—	—
95	Tag	5a	AutoIncrement	—
96	Register 0x0341	<b>frame_length_lines</b>	—	—
97	Tag	5a	AutoIncrement	—
98	Register 0x0342	<b>line_length_pck</b>	—	—
99	Tag	5a	AutoIncrement	—
100	Register 0x0343	<b>line_length_pck</b>	—	—
101	Tag	5a	AutoIncrement	—
102	Register 0x0344	<b>x_addr_start</b>	—	—

Embedded Data Byte Number	Content of Embedded Data Bytes	Value of Embedded Data Bytes	Tag Code	Comment
103	Tag	5a	AutoIncrement	—
104	Register 0x0345	<b>x_addr_start</b>	—	—
105	Tag	5a	AutoIncrement	—
106	Register 0x0346	<b>y_addr_start</b>	—	—
107	Tag	5a	AutoIncrement	—
108	Register 0x0347	<b>y_addr_start</b>	—	—
109	Tag	5a	AutoIncrement	—
110	Register 0x0348	<b>x_addr_end</b>	—	—
111	Tag	5a	AutoIncrement	—
112	Register 0x0349	<b>x_addr_end</b>	—	—
113	Tag	5a	AutoIncrement	—
114	Register 0x034A	<b>y_addr_end</b>	—	—
115	Tag	5a	AutoIncrement	—
116	Register 0x034B	<b>y_addr_end</b>	—	—
117	Tag	5a	AutoIncrement	—
118	Register 0x034C	<b>x_output_size</b>	—	—
119	Tag	5a	AutoIncrement	—
120	Register 0x034D	<b>x_output_size</b>	—	—
121	Tag	5a	AutoIncrement	—
122	Register 0x034E	<b>y_output_size</b>	—	—
123	Tag	5a	AutoIncrement	—
124	Register 0x034F	<b>y_output_size</b>	—	—
125	Tag	a5	INDEX_LSB	—
126	Address	80	—	—
127	Tag	5a	AutoIncrement	—
128	Register 0x0380	<b>x_even_inc</b>	—	—
129	Tag	5a	AutoIncrement	—
130	Register 0x0381	<b>x_even_inc</b>	—	—
131	Tag	5a	AutoIncrement	—
132	Register 0x0382	<b>x_odd_inc</b>	—	—
133	Tag	5a	AutoIncrement	—
134	Register 0x0383	<b>x_odd_inc</b>	—	—
135	Tag	5a	AutoIncrement	—
136	Register 0x0384	<b>y_even_inc</b>	—	—
137	Tag	5a	AutoIncrement	—
138	Register 0x0385	<b>y_even_inc</b>	—	—
139	Tag	5a	AutoIncrement	—
140	Register 0x0386	<b>y_odd_inc</b>	—	—

Embedded Data Byte Number	Content of Embedded Data Bytes	Value of Embedded Data Bytes	Tag Code	Comment
141	Tag	5a	AutoIncrement	–
142	Register 0x0387	<b>y_odd_inc</b>	–	–
143	Tag	aa	INDEX_MSB	–
144	Address	09	–	–
145	Tag	a5	INDEX_LSB	–
146	Address	00	–	–
147	Tag	5a	AutoIncrement	–
148	Register 0x0900	<b>binning_mode</b>	–	Required if binning supported
149	Tag	5a	AutoIncrement	
150	Register 0x0901	<b>binning_type</b>	–	
151	Tag	5a	AutoIncrement	
152	Register 0x0902	<b>binning_weighting</b>	–	
153	Tag	aa	INDEX_MSB	–
154	Address	0B	–	–
155	Tag	a5	INDEX_LSB	–
156	Address	05	–	–
157	Tag	5a	AutoIncrement	–
158	Register 0xB05	<b>mapped_defect_correct_en</b>	–	It is required to include sensor corrections if those are supported. This example supports enable controls for mapped defect correction, single defect correction, and combined couplet single defect correction.
159	Tag	5a	AutoIncrement	
160	Register 0xB06	<b>single_defect_correct_en</b>	–	
161	Tag	a5	INDEX_LSB	
162	Address	0a	–	
163	Tag	5a	AutoIncrement	–
164	Register 0xB0A	<b>combined_defect_correct_en</b>	–	
165	Tag	a5	INDEX_LSB	–
166	Address	30	–	–
167	Tag	5a	AutoIncrement	–
168	Register 0xB30	<b>OB_readout_ctrl</b>	–	Required if supported
169	Tag	aa	INDEX_MSB	–
170	Address	0C	–	–
171	Tag	a5	INDEX_LSB	–
172	Address	14	–	–
173	Tag	5a	AutoIncrement	–
174	Register 0xC14	<b>flash_strobe_start_point</b>	–	–

Embedded Data Byte Number	Content of Embedded Data Bytes	Value of Embedded Data Bytes	Tag Code	Comment
175	Tag	5a	AutoIncrement	Required if flash supported
176	Register 0x0C15	<b>flash_strobe_start_point</b>	—	
177	Tag	5a	AutoIncrement	
178	Register 0x0C16	<b>tflash_strobe_delay_rs_ctrl</b>	—	
179	Tag	5a	AutoIncrement	
180	Register 0x0C17	<b>tflash_strobe_delay_rs_ctrl</b>	—	
181	Tag	5a	AutoIncrement	
182	Register 0x0C18	<b>tflash_strobe_width_high_rs_ctrl</b>	—	
183	Tag	5a	AutoIncrement	
184	Register 0x0C19	<b>tflash_strobe_width_high_rs_ctrl</b>	—	
185	Tag	5a	AutoIncrement	
186	Register 0x0C1A	<b>flash_mode_rs</b>	—	
187	Tag	5a	AutoIncrement	
188	Register 0x0C1B	<b>flash_trigger_rs</b>	—	
189	Tag	5a	AutoIncrement	
190	Register 0x0C1C	<b>flash_status</b>	—	
191	Tag	aa	INDEX__MSB	—
192	Address	0D	—	—
193	Tag	a5	INDEX__LSB	—
194	Address	00	—	—
195	Tag	5a	AutoIncrement	—
196	Register 0xD00	<b>PDAF_ctrl</b>	—	Required if interleaved PDAF readout is supported
197	Tag	a5	INDEX__LSB	—
198	Address	04	—	—
199	Tag	5a	AutoIncrement	—
200	Register 0xD04	<b>pd_x_addr_start</b>	—	Required if PDAF ROI is supported
201	Tag	5a	AutoIncrement	
202	Register 0xD05	<b>pd_x_addr_start</b>	—	
203	Tag	5a	AutoIncrement	
204	Register 0xD06	<b>pd_y_addr_start</b>	—	
205	Tag	5a	AutoIncrement	
206	Register 0xD07	<b>pd_y_addr_start</b>	—	
207	Tag	5a	AutoIncrement	
208	Register 0xD08	<b>pd_x_addr_end</b>	—	
209	Tag	5a	AutoIncrement	
210	Register 0xD09	<b>pd_x_addr_end</b>	—	

Embedded Data Byte Number	Content of Embedded Data Bytes	Value of Embedded Data Bytes	Tag Code	Comment
211	Tag	5a	AutoIncrement	
212	Register 0x0D0A	<b>pd_y_addr_end</b>	—	
213	Tag	5a	AutoIncrement	
214	Register 0x0D0B	<b>pd_y_addr_end</b>	—	
215	Tag	aa	INDEX__MSB	—
216	Address	0e	—	—
217	Tag	a5	INDEX__LSB	—
218	Address	00	—	—
219	Tag	5a	AutoIncrement	—
220	Register 0xE00	<b>bracketing_LUT_ctrl</b>	—	Required if LUT bracketing is supported.
221	Tag	5a	AutoIncrement	
222	Register 0xE01	<b>bracketing_LUT_mode</b>	—	
223	Tag	07	end of data	Note that the end-of-data tags are used to select a size for the embedded data that can support all data packings irrespective of the selected data format.
224-227	Data	07	end of data	
228-255	Tag	07	end of data	It may be beneficial to send at least 256 pixels of embedded data. However, it is recommended that top embedded data lines have the same length as image data lines.

4248

**Table 258 Complex Example of Embedded Data**

<b>Embedded data byte number</b>	<b>Content of embedded data bytes</b>	<b>Value of embedded data bytes</b>	<b>Tag code</b>	<b>Comment</b>
0	Embedded data format code	0a	—	—
1	Tag	aa	INDEX_MSB	—
2	Address	00	—	—
3	Tag	a5	INDEX_LSB	—
4	Address	05	—	—
5	Tag	5a	AutoIncrement	—
6	Register 0x0005	<b>frame_count</b>	—	—
7	Tag	5a	AutoIncrement	—
8	Register 0x0006	<b>pixel_order</b>	—	—
9	Tag	55	AutoInc. DataNull	—
10	Null data	07	—	—
11	Tag	5a	AutoIncrement	—
12	Register 0x0008	<b>data_pedestal</b>	—	—
13	Tag	5a	AutoIncrement	—
14	Register 0x0009	<b>data_pedestal</b>	—	—
15	Tag	aa	INDEX_MSB	—
16	Address	01	—	—
17	Tag	a5	INDEX_LSB	—
18	Address	01	—	—
19	Tag	5a	AutoIncrement	—
20	Register 0x0101	<b>image_orientation</b>	—	—
21	Tag	a5	INDEX_LSB	—
22	Address	12	—	—
23	Tag	5a	AutoIncrement	—
24	Register 0x0112	<b>CSI_data_format</b>	—	—
25	Tag	5a	AutoIncrement	—
26	Register 0x0113	<b>CSI_data_format</b>	—	—
27	Tag	a5	INDEX_LSB	—
28	Address	21	—	—
29	Tag	5a	AutoIncrement	—
30	Register 0x0121	<b>ADC_bit_depth</b>	—	Required if controllable ADC bit depth is supported.
31	Tag	a5	INDEX_LSB	—
32	Address	38	—	—
33	Tag	5a	AutoIncrement	—

Embedded data byte number	Content of embedded data bytes	Value of embedded data bytes	Tag code	Comment
34	Register 0x0138	<b>temp_sensor_ctrl</b>	—	Required if temperature sensor is supported
35	Tag	55	AutoInc. DataNull	—
36	Null data	07	—	—
37	Tag	5a	AutoIncrement	—
38	Register 0x013A	<b>temp_sensor_output</b>	—	Required if temperature sensor is supported
39	Tag	aa	INDEX_MSB	—
40	Address	02	—	—
41	Tag	a5	INDEX_LSB	—
42	Address	00	—	—
43	Tag	5a	AutoIncrement	—
44	Register 0x0200	<b>fine_integration_time</b>	—	Required if fine Integration Time control is supported
45	Tag	5a	AutoIncrement	—
46	Register 0x0201	<b>fine_integration_time</b>	—	Required if fine Integration Time control is supported
47	Tag	5a	AutoIncrement	—
48	Register 0x0202	<b>coarse_integration_time</b>	—	—
49	Tag	5a	AutoIncrement	—
50	Register 0x0203	<b>coarse_integration_time</b>	—	—
51	Tag	5a	AutoIncrement	—
52	Register 0x0204	<b>analog_gain_code_global</b>	—	—
53	Tag	5a	AutoIncrement	—
54	Register 0x0205	<b>analog_gain_code_global</b>	—	—
55	Tag	a5	INDEX_LSB	—
56	Address	0E	—	—
57	Tag	5a	AutoIncrement	—
58	Register 0x020E	<b>digital_gain_global</b>	—	Required if digital Gain global is supported
59	Tag	5a	AutoIncrement	—

Embedded data byte number	Content of embedded data bytes	Value of embedded data bytes	Tag code	Comment
60	Register 0x020F	<b>digital_gain_global</b>	—	Required if digital Gain global is supported
61	Tag	a5	INDEX_LSB	—
62	Address	16	—	—
63	Tag	5a	AutoIncrement	—
64	Register 0x0216	<b>short_analog_gain_global</b>	—	In this example, the image sensor also supports HDR, with a certain feature set. Not all of the features are required with HDR. If HDR is supported, then <b>HDR_mode</b> and <b>exposure_ratio</b> are the required control registers. Other control registers are required if additional optional HDR features are supported. See <b>Section 9.9</b> for details.
65	Tag	5a	AutoIncrement	
66	Register 0x0217	<b>short_analog_gain_global</b>	—	
67	Tag	5a	AutoIncrement	
68	Register 0x0218	<b>short_digital_gain_global</b>	—	
69	Tag	5a	AutoIncrement	
70	Register 0x0219	<b>short_digital_gain_global</b>	—	
71	Tag	a5	INDEX_LSB	
72	Address	20	—	
73	Tag	5a	AutoIncrement	
74	Register 0x0220	<b>HDR_mode</b>	—	
75	Tag	5a	AutoIncrement	
76	Register 0x0221	<b>HDR_resolution_reduction</b>	—	
77	Tag	5a	AutoIncrement	
78	Register 0x0222	<b>exposure_ratio</b>	—	
79	Tag	5a	AutoIncrement	
80	Register 0x0223	<b>HDR_internal_bit_depth</b>	—	
81	Tag	5a	AutoIncrement	
82	Register 0x0224	<b>short_coarse_integration_time</b>	—	This clock tree assumes a 2-PLL clock tree
83	Tag	5a	AutoIncrement	
84	Register 0x0225	<b>short_coarse_integration_time</b>	—	
85	Tag	aa	INDEX_MSB	
86	Address	03	—	
87	Tag	a5	INDEX_LSB	
88	Address	02	—	
89	Tag	5a	AutoIncrement	
90	Register 0x0302	<b>vt_sys_clk_div</b>	—	
91	Tag	5a	AutoIncrement	
92	Register 0x0303	<b>vt_sys_clk_div</b>	—	
93	Tag	5a	AutoIncrement	
94	Register 0x0304	<b>vt_pre_pll_clk_div</b>	—	
95	Tag	5a	AutoIncrement	

Embedded data byte number	Content of embedded data bytes	Value of embedded data bytes	Tag code	Comment
96	Register 0x0305	<b>vt_pre_pll_clk_div</b>	—	AutoIncrement
97	Tag	5a	AutoIncrement	
98	Register 0x0306	<b>vt_pll_multiplier</b>	—	
99	Tag	5a	AutoIncrement	
100	Register 0x0307	<b>vt_pll_multiplier</b>	—	
101	Tag	5a	AutoIncrement	
102	Register 0x0308	<b>op_pix_clk_div</b>	—	
103	Tag	5a	AutoIncrement	
104	Register 0x0309	<b>op_pix_clk_div</b>	—	
105	Tag	5a	AutoIncrement	
106	Register 0x030A	<b>op_sys_clk_div</b>	—	
107	Tag	5a	AutoIncrement	
108	Register 0x030B	<b>op_sys_clk_div</b>	—	
109	Tag	5a	AutoIncrement	
110	Register 0x030C	<b>op_pre_pll_clk_div</b>	—	
111	Tag	5a	AutoIncrement	
112	Register 0x030D	<b>op_pre_pll_clk_div</b>	—	
113	Tag	5a	AutoIncrement	
114	Register 0x030E	<b>op_pll_multiplier</b>	—	
115	Tag	5a	AutoIncrement	
116	Register 0x030F	<b>op_pll_multiplier</b>	—	
117	Tag	5a	AutoIncrement	
118	Register 0x0310	<b>pll_mode</b>	—	
119	Tag	a5	INDEX_LSB	—
120	Address	40	—	—
121	Tag	5a	AutoIncrement	—
122	Register 0x0340	<b>frame_length_lines</b>	—	—
123	Tag	5a	AutoIncrement	—
124	Register 0x0341	<b>frame_length_lines</b>	—	—
125	Tag	5a	AutoIncrement	—
126	Register 0x0342	<b>line_length_pck</b>	—	—
127	Tag	5a	AutoIncrement	—
128	Register 0x0343	<b>line_length_pck</b>	—	—
129	Tag	5a	AutoIncrement	—
130	Register 0x0344	<b>x_addr_start</b>	—	—
131	Tag	5a	AutoIncrement	—
132	Register 0x0345	<b>x_addr_start</b>	—	—
133	Tag	5a	AutoIncrement	—

Embedded data byte number	Content of embedded data bytes	Value of embedded data bytes	Tag code	Comment
134	Register 0x0346	<b>y_addr_start</b>	—	—
135	Tag	5a	AutoIncrement	—
136	Register 0x0347	<b>y_addr_start</b>	—	—
137	Tag	5a	AutoIncrement	—
138	Register 0x0348	<b>x_addr_end</b>	—	—
139	Tag	5a	AutoIncrement	—
140	Register 0x0349	<b>x_addr_end</b>	—	—
141	Tag	5a	AutoIncrement	—
142	Register 0x034A	<b>y_addr_end</b>	—	—
143	Tag	5a	AutoIncrement	—
144	Register 0x034B	<b>y_addr_end</b>	—	—
145	Tag	5a	AutoIncrement	—
146	Register 0x034C	<b>x_output_size</b>	—	—
147	Tag	5a	AutoIncrement	—
148	Register 0x034D	<b>x_output_size</b>	—	—
149	Tag	5a	AutoIncrement	—
150	Register 0x034E	<b>y_output_size</b>	—	—
151	Tag	5a	AutoIncrement	—
152	Register 0x034F	<b>y_output_size</b>	—	—
153	Tag	5a	AutoIncrement	—
154	Register 0x0350	<b>frame_length_ctrl</b>	—	Required if automatic frame length is supported
155	Tag	55	AutoInc. DataNull	—
156	Null data	07	—	—
157	Tag	5a	AutoIncrement	—
158	Register 0x0352	<b>timing_mode_ctrl</b>	—	Required if manual readout or delayed Exposure start is supported
159	Tag	55	AutoInc. DataNull	—
160	Null data	07	—	—
161	Tag	5a	AutoIncrement	—
162	Register 0x0354	<b>frame_margin</b>	—	Required if automatic frame length is supported
163	Tag	5a	AutoIncrement	
164	Register 0x0355	<b>frame_margin</b>	—	
165	Tag	a5	INDEX_LSB	—
166	Address	80	—	—

Embedded data byte number	Content of embedded data bytes	Value of embedded data bytes	Tag code	Comment
167	Tag	5a	AutoIncrement	–
168	Register 0x0380	<b>x_even_inc</b>	–	–
169	Tag	5a	AutoIncrement	–
170	Register 0x0381	<b>x_even_inc</b>	–	–
171	Tag	5a	AutoIncrement	–
172	Register 0x0382	<b>x_odd_inc</b>	–	–
173	Tag	5a	AutoIncrement	–
174	Register 0x0383	<b>x_odd_inc</b>	–	–
175	Tag	5a	AutoIncrement	–
176	Register 0x0384	<b>y_even_inc</b>	–	–
177	Tag	5a	AutoIncrement	–
178	Register 0x0385	<b>y_even_inc</b>	–	–
179	Tag	5a	AutoIncrement	–
180	Register 0x0386	<b>y_odd_inc</b>	–	–
181	Tag	5a	AutoIncrement	–
182	Register 0x0387	<b>y_odd_inc</b>	–	–
183	Tag	aa	INDEX_MSB	–
184	Address	09	–	–
185	Tag	a5	INDEX_LSB	–
186	Address	00	–	–
187	Tag	5a	AutoIncrement	–
188	Register 0x0900	<b>binning_mode</b>	–	Required if binning is supported
189	Tag	5a	AutoIncrement	
190	Register 0x0901	<b>binning_type</b>	–	
191	Tag	5a	AutoIncrement	
192	Register 0x0902	<b>binning_weighting</b>	–	
193	Tag	aa	INDEX_MSB	–
194	Address	0B	–	–
195	Tag	a5	INDEX_LSB	–
196	Address	05	–	–
197	Tag	5a	AutoIncrement	–
198	Register 0xB05	<b>mapped_defect_correct_en</b>	–	It is required to include sensor corrections if they are supported. This example supports enable controls for mapped defect
199	Tag	5a	AutoIncrement	
200	Register 0xB06	<b>single_defect_correct_en</b>	–	
201	Tag	a5	INDEX_LSB	
202	Address	0a	–	
203	Tag	5a	AutoIncrement	–

Embedded data byte number	Content of embedded data bytes	Value of embedded data bytes	Tag code	Comment
204	Register 0x0B0A	<b>combined_defect_correct_en</b>	—	correction, single defect correction, and combined couplet single defect correction.
205	Tag	a5	INDEX_LSB	—
206	Address	30	—	—
207	Tag	5a	AutoIncrement	—
208	Register 0x0B30	<b>OB_readout_ctrl</b>	—	Required if supported
209	Tag	aa	INDEX_MSB	—
210	Address	0C	—	—
211	Tag	a5	INDEX_LSB	—
212	Address	14	—	—
213	Tag	5a	AutoIncrement	—
214	Register 0x0C14	<b>flash_strobe_start_point</b>	—	Required if flash supported
215	Tag	5a	AutoIncrement	
216	Register 0x0C15	<b>flash_strobe_start_point</b>	—	
217	Tag	5a	AutoIncrement	
218	Register 0x0C16	<b>tflash_strobe_delay_rs_ctrl</b>	—	
219	Tag	5a	AutoIncrement	
220	Register 0x0C17	<b>tflash_strobe_delay_rs_ctrl</b>	—	
221	Tag	5a	AutoIncrement	
222	Register 0x0C18	<b>tflash_strobe_width_high_rs_ctrl</b>	—	
223	Tag	5a	AutoIncrement	
224	Register 0x0C19	<b>tflash_strobe_width_high_rs_ctrl</b>	—	
225	Tag	5a	AutoIncrement	
226	Register 0x0C1A	<b>flash_mode_rs</b>	—	
227	Tag	5a	AutoIncrement	
228	Register 0x0C1B	<b>flash_trigger_rs</b>	—	
229	Tag	5a	AutoIncrement	
230	Register 0x0C1C	<b>flash_status</b>	—	
231	Tag	aa	INDEX_MSB	—
232	Address	0D	—	—
233	Tag	a5	INDEX_LSB	—
234	Address	00	—	—
235	Tag	5a	AutoIncrement	—

Embedded data byte number	Content of embedded data bytes	Value of embedded data bytes	Tag code	Comment
236	Register 0x0D00	<b>PDAF_ctrl</b>		Required if interleaved PDAF readout is supported
237	Tag	a5	INDEX_LSB	–
238	Tag	5a	AutoIncrement	–
239	Address	04	–	–
240	Register 0x0D04	<b>pd_x_addr_start</b>	–	Required if PDAF ROI is supported
241	Tag	5a	AutoIncrement	
242	Register 0x0D05	<b>pd_x_addr_start</b>	–	
243	Tag	5a	AutoIncrement	
244	Register 0x0D06	<b>pd_y_addr_start</b>	–	
245	Tag	5a	AutoIncrement	
246	Register 0x0D07	<b>pd_y_addr_start</b>	–	
247	Tag	5a	AutoIncrement	
248	Register 0x0D08	<b>pd_x_addr_end</b>	–	
249	Tag	5a	AutoIncrement	
250	Register 0x0D09	<b>pd_x_addr_end</b>	–	
251	Tag	5a	AutoIncrement	
252	Register 0x0D0A	<b>pd_y_addr_end</b>	–	
253	Tag	5a	AutoIncrement	
254	Register 0x0D0B	<b>pd_y_addr_end</b>	–	
255	Tag	aa	INDEX_MSB	–
256	Address	0e	–	–
257	Tag	a5	INDEX_LSB	–
258	Address	00	–	–
259	Tag	5a	AutoIncrement	–
260	Register 0xE00	<b>bracketing_LUT_ctrl</b>	–	Mandatory if LUT bracketing supported.
261	Tag	5a	AutoIncrement	
262	Register 0xE01	<b>bracketing_LUT_mode</b>	–	
263	Tag	07	end of data	Note that the end-of-data tags are used to select a size for the embedded data that can support all data packings irrespective of the selected data format.
264	Data	07	end of data	

Embedded data byte number	Content of embedded data bytes	Value of embedded data bytes	Tag code	Comment
265-319	Tag	07	end of data	This example shows how a single line can include tag+data pairs within a 320pixel image width (which is the minimum). However, it may be beneficial to use multiple embedded lines with less tag+data pairs per line. Notice that in all cases, that top embedded data lines should have the same length as the image data lines.

## A.2 Retiming Examples

4249 This Section illustrates group parameter hold and other retiming rules through examples.

### A.2.1 Generic Retiming Rules

4250 As defined in *Section 9.5*, **grouped\_parameter\_hold** can be used to group ‘retimed’ parameters together.  
4251 For a list of ‘retimed’ parameters, see the CCI Register map in *Section 20*. The image sensor shall support  
4252 the use of **grouped\_parameter\_hold** with ‘retimed’ parameters. The following Sections clarify some of the  
4253 typical usages of retiming and/or **grouped\_parameter\_hold**.

### A.2.2 Exposure Parameter Retiming

4254 When **grouped\_parameter\_hold** is used to synchronize the new Exposure parameters together, the image  
4255 sensor shall internally retime the usage of the settings. Exposure parameters can consist of two or more of  
4256 registers: **coarse\_integration\_time**, **fine\_integration\_time**, **analog\_gain\_code\_global**, and/or  
4257 **digital\_gain\_global**.

4258 If the Host is not using **grouped\_parameter\_hold**, then it must take care to ensure that Exposure parameters  
4259 take effect on the desired frame(s). For example, the Host want to take into account whether the image sensor  
4260 supports variable Gain delay or fixed Gain delay (based on register **gain\_delay\_type**). If the image sensor  
4261 supports variable Gain delay, then that means that the delay controlling the time at which new Gain settings  
4262 are taken into use is shorter than the **coarse\_integration\_time** period. In such a scenario, the Host must first  
4263 program register **coarse\_integration\_time**, and then during the next frame program Gain if the Host wants  
4264 those parameters to be applied to the same frame.

### A.2.3 Flash Retiming

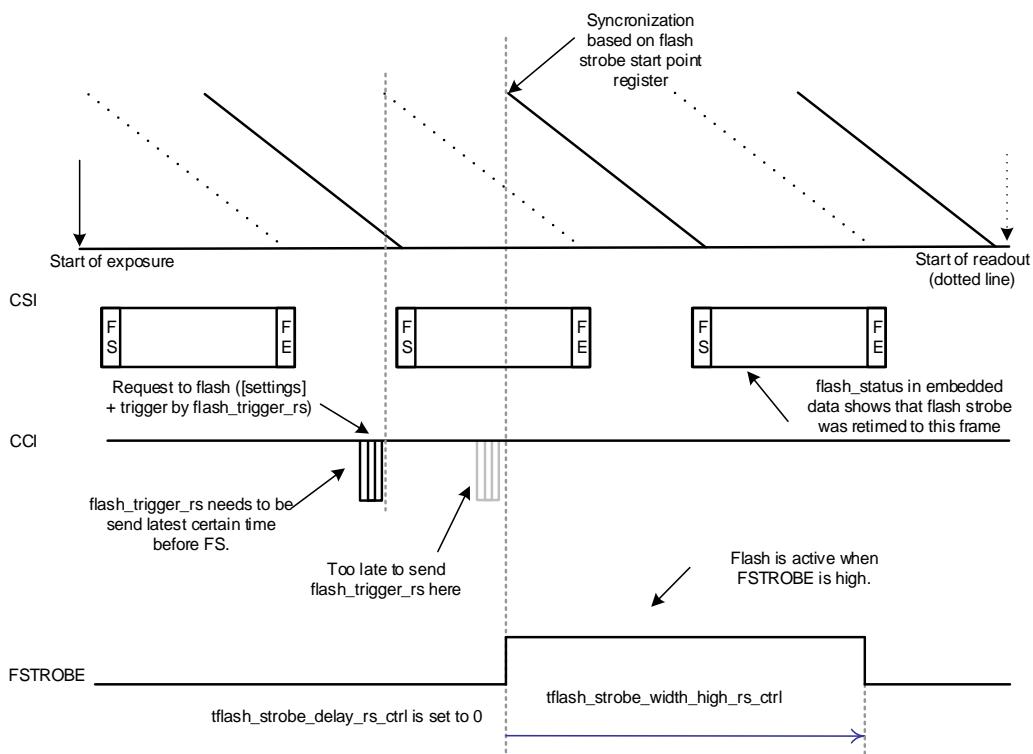
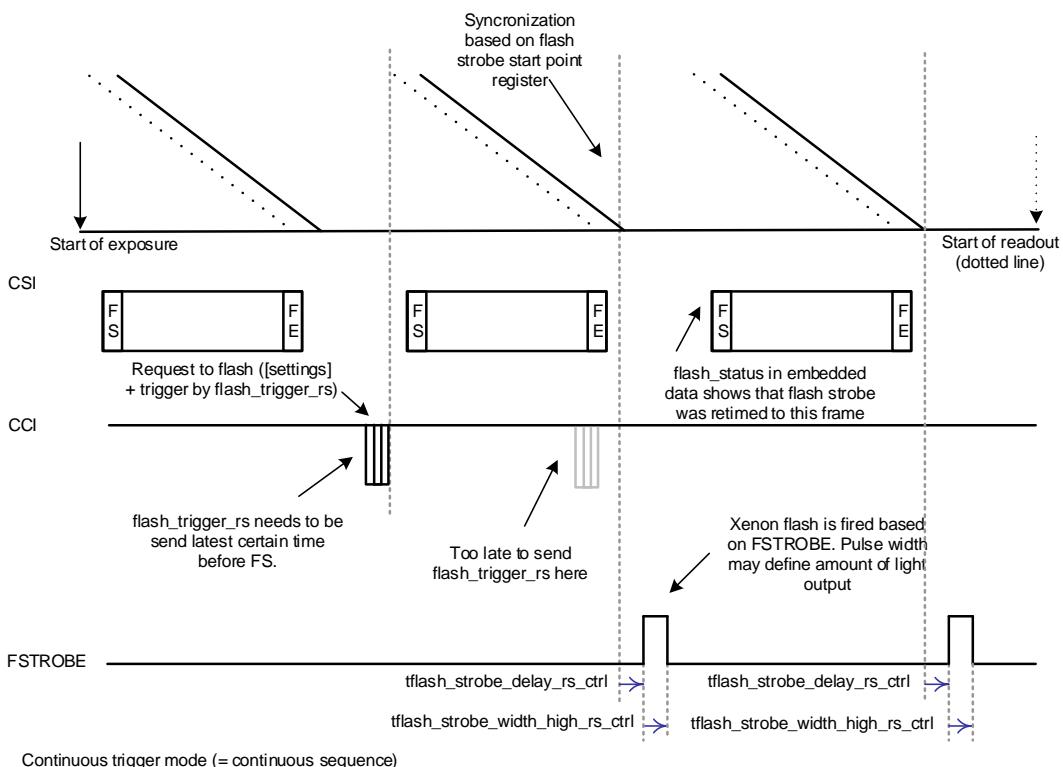
4265 A camera system may include both an image sensor and a flash component. The image sensor can control  
4266 the flash pulse via the flash strobe signal. This Section clarifies flash strobe control when the Exposure  
4267 parameters are not changed for the same frame. The Exposure parameters would consist of one or more of  
4268 registers: **coarse\_integration\_time**, **fine\_integration\_time**, **analog\_gain\_code\_global**, and/or  
4269 **digital\_gain\_global**.

4270 If the image sensor supports flash strobe, then it shall support the use cases described in this Section. The  
4271 following examples clarify required image sensor behavior in certain use cases.

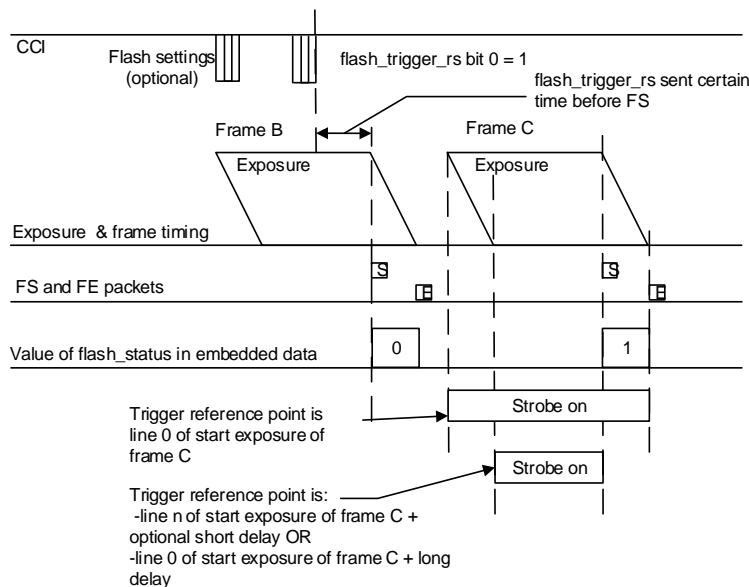
4272 As described in *Section 16.2*, in synchronous mode the flash strobe can be synchronized to the start of the  
4273 Exposure. The Synchronization point is controlled by register **flash\_strobe\_start\_point** (i.e., the line  
4274 number selection). In *Figure 85* and *Figure 86*, two different examples show how flash control can be used  
4275 with different Exposure times, in order to have the flash active at a different moment. Both Figures show the  
4276 last moment when register **flash\_trigger\_rs** needs to be sent in order to get the flash strobe activated as  
4277 illustrated. This last moment is the particular time before the previous frame’s frame start packet. The Figures  
4278 also show that the embedded data includes register **flash\_status**, which indicates whether or not the flash  
4279 was retimed to that frame.

4280 Typically the following flash settings will be updated before sending register **flash\_trigger\_rs**:

- **tflash\_strobe\_width\_high\_rs\_ctrl**
- **flash\_strobe\_start\_point**
- **tflash\_strobe\_delay\_rs\_ctrl**

**Figure 85 Flash Strobe with Long Pulse****Figure 86 Flash Strobe with Short Pulse**

4286  
4287 **Figure 85 and Figure 86** could be summarized into **Figure 87**. Note that the Exposure time is different than in **Figure 85**, and the **tflash\_strobe\_delay\_rs\_ctrl** delay from **Figure 86** is not shown.



4288

**Figure 87 Flash Example with Flash Trigger**

#### A.2.4 Exposure Parameter Retiming with Flash

4289 The previous Section described how the flash is retimed when the Exposure parameters do not change.  
4290 However, it is a typical use case for the Exposure parameters to be different between the frame before the  
4291 flash and the frame with the flash. This Section clarifies the use of **grouped\_parameter\_hold** for flash strobe  
4292 retiming, in order to synchronize flash firing with the desired Exposure parameters in the image sensor.

4293 When **grouped\_parameter\_hold** is used to synchronize the flash and the new Exposure parameters, the  
4294 image sensor shall internally retime the usage of the settings in such a way that the flash strobe activates for  
4295 the frame in which the new Exposure parameters first become valid. The Exposure parameters can consist of  
4296 one or more of registers: **coarse\_integration\_time**, **fine\_integration\_time**, **analog\_gain\_code\_global**,  
4297 and/or **digital\_gain\_global**.

4298 The following examples clarify required image sensor behavior in certain use cases.

4299 The following flash setting registers are ‘retimed’ registers, which can be used either with or without  
4300 **grouped\_parameter\_hold** to achieve retiming. These settings may be updated before releasing the  
4301 **grouped\_parameter\_hold**.

- 4302 • **tflash\_strobe\_width\_high\_rs\_ctrl**
- 4303 • **flash\_strobe\_start\_point**
- 4304 • **tflash\_strobe\_delay\_rs\_ctrl**

4305 Synchronizing register **flash\_trigger\_rs** with the Exposure parameters via register  
4306 **grouped\_parameter\_hold** achieves the desired retiming. The following two use cases illustrate this.

**4307 Use Case 1**

4308 During streaming, image sensor Exposure parameters are synchronized with the flash. The Host may need to  
4309 change **frame\_length\_lines** in order to accommodate new Integration Time values.

**4310 Initial State:**

- 4311 • The image sensor is streaming (e.g. is in viewfinder/preview configuration)  
4312 • The flash is off  
4313 • The old Exposure parameters are used

**4314 Sequence:**

- 4315 1. The Host sets register **grouped\_parameter\_hold** to the value 1, starting the hold
  - 4316 • The Host sends new Exposure settings, including one or more of the following registers:  
**coarse\_integration\_time**, **fine\_integration\_time**, **analog\_gain\_code\_global**, and/or  
**digital\_gain\_global**
  - 4319 • The Host can also optionally set a new value for register **frame\_length\_lines**, in order to  
4320 accommodate any new Integration Time values
- 4321 2. The Host set register **flash\_trigger\_rs** bit 0 to the value 1 , triggering the flash
- 4322 3. The Host sets register **grouped\_parameter\_hold** to value 0, releasing the hold
- 4323 4. Then image sensor then internally retimes the above parameters, so that the flash will trigger with  
4324 the first frame for which the new Exposure settings are used.

**4325 Use Case 2**

4326 As per Use Case 1, during streaming, image sensor Exposure parameters are synchronized with flash. The  
4327 Host may need to change **frame\_length\_lines** in order to accommodate new Integration Time values. In  
4328 addition, flash parameters are changed.

**4329 Initial State:**

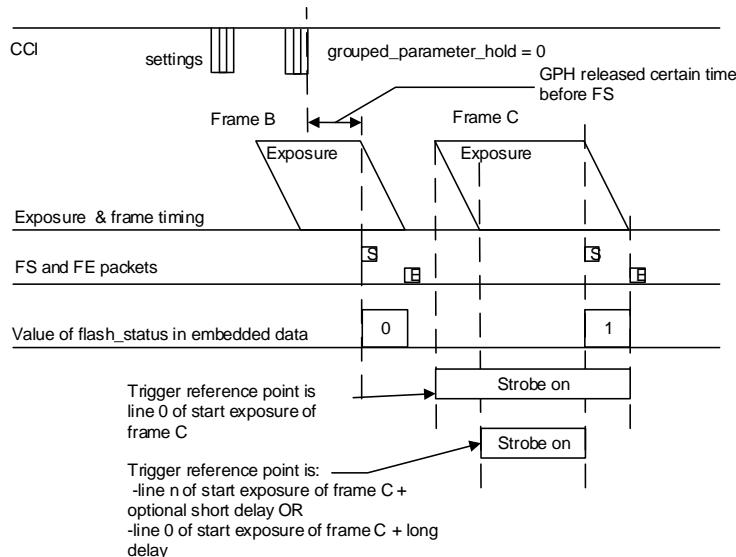
- 4330 • The image sensor is streaming (e.g. is in viewfinder/preview configuration)  
4331 • The flash is off  
4332 • The old Exposure parameters are used

**4333 Sequence:**

- 4334 1. Host sets register **grouped\_parameter\_hold** to the value 1, starting the hold
  - 4335 • The Host sends new Exposure settings, including one or more of following registers:  
**coarse\_integration\_time**, **fine\_integration\_time**, **analog\_gain\_code\_global**, and/or  
**digital\_gain\_global**
  - 4338 • The Host can also optionally set a new value for register **frame\_length\_lines**, in order to  
4339 accommodate any new Integration Time values
  - 4340 • The Host can also set new flash parameter values. Typically, at least register  
4341 **tflash\_strobe\_width\_high\_rs\_ctrl** would be set, but in theory at least one of following would  
4342 be set:
    - 4343 • **tflash\_strobe\_width\_high\_rs\_ctrl**
    - 4344 • optionally **tflash\_strobe\_delay\_rs\_ctrl**
    - 4345 • optionally **flash\_strobe\_start\_point**
- 4346     The purpose of setting new flash parameters could be to change the flash pulse width, or  
4347     to change flash trigger reference points. Alternatively, new flash parameters could be sent  
4348     to the image sensor earlier; for example, if Single Flash mode is used they could be sent  
4349     before starting the hold (i.e., before setting register **grouped\_parameter\_hold** to the  
4350     value 1).
- 4351 2. The Host sets register **flash\_trigger\_rs** bit 0 to the value 1, triggering the flash

- 4352 3. The Host sets register **grouped\_parameter\_hold** to the value 0, releasing the hold  
 4353 4. Image sensor shall internally retime above parameters so that flash will trigger with frame that has  
 4354 new Exposure settings in use.

4355 **Figure 88** illustrates how the flash strobe and embedded data behave, depending upon the settings, following  
 4356 the release of the grouped parameter hold.



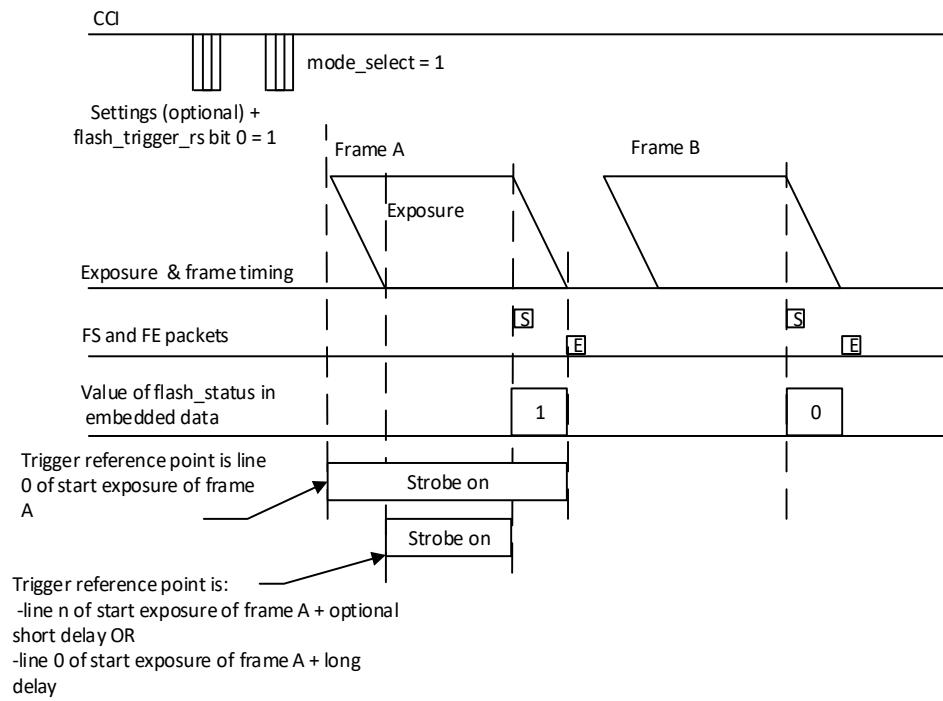
4357

**Figure 88 Flash Example with GPH**

### A.2.5 Flash Synchronization Using Software Standby Mode

4358 This Section presents an additional flash example.

4359 In **Figure 89**, the flash strobe is activated for the first frame after SW Standby mode.  
 4360 First, the image sensor  
 4361 is programmed while in SW Standby mode, and bit 0 of register **flash\_trigger\_rs** is set to the value 1 to  
 4362 trigger the flash. After that, streaming is started by setting register **mode\_select** to the value 1. Because all  
 4363 of the settings are programmed while in SW Standby mode, it is not necessary to use register  
**grouped\_parameter\_hold** to synchronize the parameter changes and the flash firing.



4364

**Figure 89 Flash Example in Streaming Start**

## 4365 Annex B CCS Static Data

### 4366 B.1 Introduction

4367 CCS v1.1 adds an option for the Host to obtain an image sensor's capability and parameter limit information  
 4368 from a file, instead of reading the information from CCS Capability registers in the image sensor. It is also  
 4369 possible for the information to be stored not in a file, but in an external EEPROM. This data is called CCS  
 4370 Static Data.

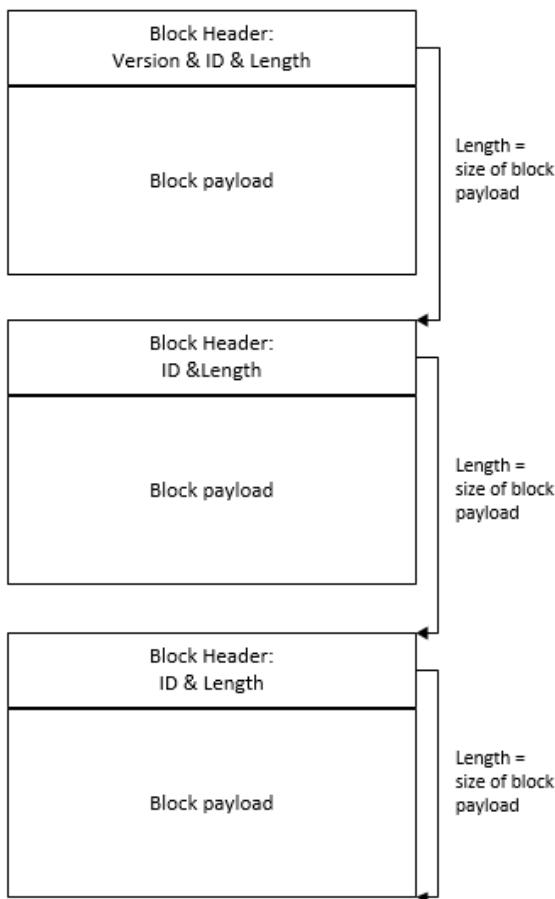
4371 The purpose of this non-register based capability and limit information (CCS Static Data) is to bring more  
 4372 flexibility to reporting capabilities, and also to make it possible to report image sensor limits, some of which  
 4373 may be known only after sensor characterization.

4374 The CCS Static Data format is generic and flexible. CCS Static Data can be used for MIPI-defined data,  
 4375 and/or for vendor-defined data. Vendor data may be, for example, calibration data or any other data.

### B.2 CCS Static Data Format

#### B.2.1 Introduction

4376 CCS Static Data consists of consecutive, independent blocks. Parsing any block in CCS Static Data is  
 4377 possible without parsing any other block apart from the block headers. In general it is possible to override  
 4378 and update information available from different data sources, through use of version and date information  
 4379 specific to the data source.



4380 **Figure 90 CCS Static Data Format**

## B.2.2 Generic Definitions

4381 These definitions are used in multiple different contexts in the CCS Static Data format.

### B.2.2.1 Length Specifier

4382 The length specifier is used for space-efficient encoding of length where the exact amount of data is generally  
 4383 smallish but may be significantly larger as well. The size of the length specifier is variable depending on the  
 4384 length it specifies, from 1 to 3 bytes. Always use the smallest length specifier applicable.

4385 **Table 259 Length Specifier for 0 – 2<sup>6</sup>-1 bytes**

Byte	Offset + 0
0	Bits 7-6: 0x00 Bits 5-0: Length(5:0)

4386 **Table 260 Length Specifier for 2<sup>6</sup>– 2<sup>14</sup> -1 bytes**

Byte	Offset + 0	Offset + 1
0	Bits 7-6: 0x01 Bits 5-0: Length(13:8)	Length(7:0)

4387 **Table 261 Length Specifier for 2<sup>14</sup>– 2<sup>22</sup> -1 bytes**

Byte	Offset + 0	Offset + 1	Offset + 2
0	Bits 7-6: 0x02 Bits 5-0: Length(21:16)	Length(15:8)	Length(7:0)

## B.2.3 Common Register BLOB

4388 The Common Register BLOB format is used in Read Only Register Block, in MSR Block, and also inside  
 4389 Generic Rule based Block. See description of each block to understand the specific rules for each. Note that  
 4390 BLOB shall describe complete registers: specifying only a part of the register is not allowed. For example, if  
 4391 a 16-bit register is described, then both the Hi byte and the Lo byte shall be described.  
 4392

4393 The initial index (e.g., the register index) is 0x0000 for each block, unless otherwise specified. The index  
 4394 incrementing or index overriding is controlled by the BLOB.

4395 The content of the BLOB depends on values of Format Selection field, which is located in the first byte of  
 4396 the BLOB. All the BLOB fields are described in **Table 262**. Valid BLOB structures are shown in **Table 263**,  
 4397 **Table 264**, and **Table 265**, depending on the value of Format selection field.

4398

**Table 262 Description of BLOB Fields**

Field	Type	Description
<b>Control field</b>	8-bit unsigned integer	<p>The first byte in the BLOB.</p> <p><b>Bits 6-7</b> Format selection field Defines the details of other bits in Control field and in Index field</p> <p><b>0:</b> Index field not used, Bit 5-3: Data size field Bit 2-0: Index Increment field</p> <p><b>1:</b> 1-byte Index field used Bit 5-1: Data size field Bit 0: Index Increment field combined with Index field</p> <p><b>2:</b> 2-Byte Index field used Bit 5-0: Data size field</p> <p><b>3: Reserved</b></p> <p><b>Bits 3-5, or 1-5 or 0-5:</b> Data size field</p> <p><b>Bits 0-2, 0, none:</b> Index increment field</p>
<b>Data size</b>	Part of the Control field	Specifies the length of the Data field Value = length -1
<b>Index increment</b>	Part of the Control field	<p>The field specifies how much the index is incremented before the data. The previous index is the index after the previous data BLOB, i.e., the index of last byte +1.</p> <p><b>Examples:</b></p> <p>0: index = previous index 1: index = prev. index +1 ... n: index = prev. index +n</p> <p><b>When format selection field has value 0:</b> index = prev. index + index increment field value (n =0 to 7)</p> <p><b>When format selection field has value 1:</b> index = prev. index + (index increment field value as top bit and index field as LSB, n =0 to 511)</p> <p><b>When format selection field has value 2:</b> The field does not exist in the BLOB</p>
<b>Index</b>	Variable size	<p>Included in the BLOB, after the Control field, if bit 6-7 of control field has value 1 or 2, otherwise excluded from the BLOB.</p> <p><b>When format selection field has value 1:</b> Index field is used as LSB when incrementing the index, the field size is 1 byte</p> <p><b>When format selection field has value 2:</b> Index field is used to override both MSB and LSB of the index, the field size is two bytes</p>
<b>Data field</b>	Variable size	<p>Included in the BLOB as the last field.</p> <p>The data, size based on Data size field. Index is auto-incremented by 1 after each byte.</p>

4399

**Table 263 BLOB with Format Selection Field Value 0**

Field	Type	Description
<b>Control field</b>	8-bit unsigned integer	<p>The first byte in the BLOB.</p> <p><b>Bits 6-7:</b> Format selection field</p> <p>Defines the details of other bits in Control field and in Index field</p> <p><b>0:</b></p> <ul style="list-style-type: none"> <li>Index field not used,</li> <li>Bit 5-3: Data size field</li> <li>Bit 2-0: Index Increment field</li> </ul> <p><b>Bits 3-5:</b> Data size field</p> <p>Specifies the length of the Data field</p> <p>Value = length -1</p> <p><b>Bits 0-2:</b> Index increment field</p> <p>The field specifies how much index is incremented before the data. The previous index is the index after previous data BLOB i.e. the index of last byte +1.</p> <p><b>Examples:</b></p> <ul style="list-style-type: none"> <li>0: index = previous index</li> <li>1: index = prev. index + 1</li> <li>...</li> <li>n: index = prev. index + n</li> </ul> <p><b>When format selection field has value 0:</b></p> <p>index = prev. index + index increment field value (n = 0 to 7)</p>
<b>Data field</b>	Variable size	Included in the BLOB as the last field. The data, size based on Data size field. Index is auto-incremented by 1 after each byte.

4400

**Table 264 BLOB with Format Selection Field Value 1**

Field	Type	Description
<b>Control field</b>	8-bit unsigned integer	<p>The first byte in the BLOB.</p> <p><b>Bits 6-7:</b> Format selection field          Defines the details of other bits in Control field and in Index field</p> <p><b>1:</b>          1-byte Index field used          Bit 5-1: Data size field          Bit 0: Index Increment field combined with Index field</p> <p><b>Bits 1-5:</b> Data size field          Specifies the length of the Data field          Value = length -1</p> <p><b>Bit 0:</b> Index increment field          The field specifies how much the index is incremented before the data. The previous index is the index after the previous data BLOB i.e. the index of last byte +1.</p> <p><b>Examples:</b>          0: index = previous index          1: index = prev. index + 1          ...          n: index = prev. index + n</p> <p><b>When format selection field has value 1:</b>          index = prev. index + (index increment field value as top bit and index field as LSB, n = 0 to 511)</p>
<b>Index field</b>	8-bit unsigned integer	<p>Included in the BLOB after the Control field if bit 6-7 of the Control field has value 1 or 2, otherwise excluded from the BLOB.</p> <p><b>When format selection field has value 1:</b>          Index field is used as LSB when incrementing the index, the field size is 1 byte.</p>
<b>Data field</b>	Variable size	<p>Included in the BLOB as the last field.</p> <p>The data, size based on Data size field. Index is auto-incremented by 1 after each byte.</p>

4401

**Table 265 BLOB with Format Selection Field Value 2**

Field	Type	Description
<b>Control field</b>	8-bit unsigned integer	<p>The first byte in the BLOB.</p> <p><b>Bits 6-7:</b> Format selection field Defines the details of other bits in Control field and in Index field</p> <p><b>2:</b> 2-Byte Index field used Bit 5-0: Data size field</p> <p><b>Bits 0-5:</b> Data size field Specifies the length of the Data field Value = length -1</p>
<b>Index field</b>	16-bit unsigned integer	<p>Included in the BLOB after the Control field if bit 6-7 of the Control field has value 1 or 2, otherwise excluded from the BLOB.</p> <p><b>When format selection field has value 2:</b> Index field is used to override both MSB and LSB of the index, the field size is two bytes.</p>
<b>Data field</b>	Variable size	<p>Included in the BLOB as the last field.</p> <p>The data, size based on Data size field. Index is auto-incremented by 1 after each byte.</p>

### B.2.4 Block Header

Each block header shall follow the format specified in **Table 266**, except the first block. See **Section B.2.5** for this exception. Any given block ID may be present in a CCS Static Data source only once.

**Table 266 Generic Block Header**

Byte	Offset + 0	Offset + 1 ... 3
0	Block ID (bits 7:0)	Length specifier, see <b>Section B.2.2.1</b>

**Table 267** specifies the fields used in the Generic Block Format.

**Table 267 Description of Block fields**

Field name	Type	Comment
<b>Block ID</b>	8-bit unsigned integer	Identifier of the Block
<b>Length specifier</b>	Variable size	Length specifier: used to denote block size between 0 and $2^{22} - 1$ bytes. See <b>Section B.2.1</b> .
<b>Block payload</b>	Variable size	Field after Length specifier field. Block payload. The data is byte based, i.e. the data size shall be divisible by 8 bits.

Block ID areas have been allocated for different purposes, as shown in **Table 268**.

**Table 268 Block ID usage**

Block ID	Usage	Comment
0	Reserved	—
1-23	MIPI	Note: Only these IDs can be used as a first block
24-31	Reserved	Note: Only these IDs can be used as a first block
32-127	MIPI	—
128-191	Vendor	—
192-255	Reserved	—

The following specific Block IDs have been specified (**Table 269**). For most of the blocks, a separate block ID exists for camera module vs. image sensor. The purpose of this is to simplify priority rules when using CCS Static Data (see **Section B.3**). Both the module block and the sensor block shall use the same format for the block content. **Table 269** also specifies how the information is provided by each block if same block is provided by multiple sources (i.e., there is both a module block and a sensor block). The prioritization granularity defines whether the block is prioritized as an entire block, or whether each item will be prioritized separately based on independent per-data-source priorities. The override policy defines whether the block is overridden by another block, independently of its data source's priority. See the details for each block in the relevant Section, and see **Section B.3** for definitions of the rules.

4418

**Table 269 Block ID Allocations**

Block name	Block ID	Prioritisation Granularity	Override Policy	Notes
<b>Reserved</b>	0	NA	NA	Reserved block ID, not used.
<b>Dummy Block</b>	1	NA	NA	Can be used as a dummy block
<b>CCS Static Data Version Block</b>	2	NA	NA	–
<b>Read Only Register Block – Sensor</b>	3	Item	Read Only Register Block – Module	–
<b>Read Only Register Block – Module</b>	4	Item	None	–
<b>MSR Block – Sensor</b>	5	Block	See MSR block documentation	–
<b>MSR Block – Module</b>	6	Block	See MSR block documentation	–
<b>Generic Rule Based Block – Sensor</b>	32	Block	Generic Rule Based Block – Module	(1)
<b>Generic Rule Based Block – Module</b>	33	Block	None	(1)
<b>PDAF Pixel Location Block – Sensor</b>	36	Block	PDAF Pixel Location Block – Module	(1)
<b>PDAF Pixel Location Block – Module</b>	37	Block	None	(1), Typically not used
<b>License Block</b>	40	NA	NA	Used if CCS Static Data is in file
<b>End of Data Block</b>	127	NA	NA	Describes the end of CCS Static Data

4419

**Note:**

1. The information in PDAF Readout Record, PDAF Pixel Location Block, and FFD Record shall not conflict with each other. For example, the number of visible pixels needs to be the same, and they should be available either from the same data source, or from data sources with equal priority (e.g., having the same version); see **Section B.3**. Note that in absence of an FFD Record, Frame Format Descriptors in CCS registers are used instead.

### B.2.5 First Block Header

The first block in this Specification version includes a block header, whose format is different from the Generic File Format Header. The Static Data format version shall have value 4b'0. If the Static Data format version is non-zero, then a CCS Static Data parser based on this Specification version shall stop.

Note that *Table 270* specifies only the block header, whereas in other Sections only the block payload is described.

Note that the first block can be any of the blocks for which the header ID has zeroes in bits 7-4. For example, the first block may be CCS Static Data Version, or Read Only Register block.

**Table 270 Block Header of First Block**

Byte	Offset + 0	Offset + 1 ... 3
0	Static Data format version (bits 7:4) & Block ID (bits 3:0)	Length specifier, see <b>Section B.2.2.1</b> .

### B.2.6 Dummy Block

If CCS Static Data would include only Blocks whose header IDs are not zero in bits 7-4, then those blocks cannot be added as a first block. In that case, the dummy block can be used instead to fulfill the rules for the first Block's header as stated in **Section B.2.5**. The dummy block may be used, e.g., when the CCS Static Data includes only vendor data.

The payload size of this block shall be zero.

### B.2.7 CCS Static Data Version Block

This block is mandatory if the CCS Static Data is in a file, and recommended if the CCS Static Data is in EEPROM. *Table 271* specifies the block payload.

**Table 271 File Version Block**

Byte	Offset + 0	Offset + 1	Offset + 2	Offset + 3
0	Static data version major (bits 15:8)	Static data version major (bits 7:0)	Static data version minor (bits 15:8)	Static data version minor (bits 7:0)
4	Year (bits 15:8)	Year (bits 7:0)	Month (7:0)	Day (7:0)

### 4437 B.2.8 Read Only Register Block

4438 The main purpose of this block is to provide RO information outside of image sensor registers, to gain  
4439 flexibility, e.g., in adjusting values after image sensor characterization.

4440 Using this block only for certain RO registers makes it possible to, in future, reallocate real register addresses  
4441 for other use, but still use the addresses as identification tags for existing RO register. This allows expansion  
4442 of the CCS register set both for RW registers and for Parameter information.

4443 If the value of a read-only register is available to the CCS device driver software through CCS Static Data  
4444 sources, then the CCS device driver shall use the value available in the CCS Static Data sources in the same  
4445 manner as though it had been read from the image sensor register.

4446 This block should only be used to describe the content of CCS RO Parameter Limit Registers (**Section 20.7**),  
4447 Frame Format Description Registers (**Section 20.6.1.2**), Analog Gain Description Registers  
4448 (**Section 20.6.1.3**), and Data Format Description Registers (**Section 20.6.1.4**). This block may also include  
4449 certain General Status Registers (**Section 20.6.1.1**), for example **data\_pedestal** and **MIPI\_CCS\_version**.

4450 The payload of this block shall use the Common Register BLOB format described in **Section B.2.3**.

4451 **Note:**

4452 In CCS Static Data, certain registers are not allowed. CCS Static Data provides a mechanism to  
4453 report limit values as a function of CCS register(s). Such mechanism shall be used to report limits  
4454 that depend on binning, see **Section B.2.10**.

4455 The following legacy registers specific to binning shall not be included in CCS Static Data:

- 4456 • **min\_frame\_length\_lines\_bin**
- 4457 • **max\_frame\_length\_lines\_bin**
- 4458 • **min\_line\_length\_pck\_bin**
- 4459 • **max\_line\_length\_pck\_bin**
- 4460 • **min\_line\_blanking\_pck\_bin**
- 4461 • **fine\_integration\_time\_min\_bin**
- 4462 • **fine\_integration\_time\_max\_margin\_bin**

4463 In this specification version, Read Only Register Block values are used to override image sensor register  
4464 values, i.e., the CCS Static Data block values are used for the registers included in the block, and image  
4465 sensor register values are used for other registers.

4466 This approach allows two main use cases:

- 4467 • Example 1: The image sensor may include capability registers, but a particular register has the wrong  
4468 value. In that case, the Block may include the value only for that register. That value may be either  
4469 zero or non-zero. This allows for reporting the correct image sensor capabilities to the Host without  
4470 having to include all of the capability register values in the Block.
- 4471 • Example 2: The image sensor may use the Block to report all of the capability registers. In this case,  
4472 the Block typically includes only the non-zero register values, since typically all CCS capabilities  
4473 have zero as the default value for non-supported features. This allows for a compact block size when  
4474 reporting image sensor capabilities to the Host.

4475 Note that block shall include complete register values. For example, if a 16-bit register is overridden, then  
4476 both the Hi byte and the Lo byte shall be included in the block.

4477 If both a Read Only Register Block and a Generic Rule Based Block are available, then they should both be  
4478 available either from the same data source, or from data sources with equal priority (e.g., that have the same  
4479 version); see **Section B.3**.

4480 **Table 272** describes payload data for the RO Register Block; the Data field registers are from **Table 228**.

4481

**Table 272 Example of RO Register Block**

BLOB	Byte	Field	Bits	Value	Content	
0	0	Control field	7:6	2	Format selection field	
			5:0	1	Data size field, value = size -1	
	1	Index field	15:8	0x10	Index of integration_time_capability	
			7:0	0x00		
	3	Data field	15:8	0x00	Data of index 0x1000 and 0x1001	
			7:0	0x01		
1	5	Control field	7:6	0	Format selection field	
			5:3	7	Data size field, value = size -1	
			2:0	2	Index increment field, new index = prev index +1 +2	
	6	Data field	15:8	e.g. 0x00	Data of index 0x1004 and 0x1005, coarse_integration_time_min Hi and Lo	
			7:0	e.g. 0x08		
	8		15:8	e.g. 0x00	Data of index 0x1006 and 0x1007, coarse_integration_time_max_margin Hi and Lo	
			7:0	e.g. 0x0A		
	10		15:8	e.g. 0x00	Data of index 0x1008 and 0x1009, fine_integration_time_min Hi and Lo	
			7:0	e.g. 0x08		
	12		15:8	e.g. 0x00	Data of index 0x100A and 0x100B, fine_integration_time_max_margin Hi and Lo	
			7:0	e.g. 0x0A		

4482

### B.2.9 MSR Register Block

4483

The MSR Register Block can be used to describe static MSR registers that the Host will write to the image sensor's MSR registers in each power-up during Software Standby Mode.

4485

During power-up the **extclk\_frequency\_mhz** register is written first, then the MSR Register Block registers, and then the standard CCS registers (e.g., image sensor settings for full resolution or binning). The MSR Register Block shall describe the values for all relevant MSR registers.

4488

If the CCS Static Data also includes a Generic Rule Based Block, then a value (or values) provided by the MSR Register Block may be overridden by the Generic Rule Based Block. For example, a Generic Rule Based Block provides the value for certain MSR register(s) if a certain rule is applicable, meaning that such MSR registers are written to the image sensor next. If the rule is not applicable, then the register values are instead provided by the MSR Register Block that was previously written to the image sensor. Finally in the power-up sequence, the image sensor is commanded to start streaming. The Generic Rule Based Block is described in **Section B.2.10**. If both an MSR Block and a Generic Rule Based Block are available, then they should both be available from the same data source, or from data sources with equal priority (i.e., that have the same version); see **Section B.3**.

4497

The block payload shall use the Common Register BLOB format as described in **Section B.2.5**. This format is the same as the one used in a Read Only Register Block (**Section B.2.8**).

**B.2.9.1     Override Policy**

4499 Same-priority specific MSR Register blocks will be written to the image sensor in the same order in which  
4500 they are introduced in the platform firmware. This applies both to the image sensor MSR block and the  
4501 camera module MSR block.

4502 If both an image sensor MSR block and a camera module MSR block are available, then the image-sensor-  
4503 specific MSR block contents will be written to the image sensor first, followed by the camera-module-  
4504 specific MSR block contents.

4505 In other words, the order is:

- 4506 1. All image-sensor-specific, same-priority MSR block contents are written to the image sensor in  
4507 the same order in which they are specified in the system firmware.
- 4508 2. All camera-module-specific, same-priority MSR block contents are written to the image sensor, in  
4509 the same order in which they are specified in the system firmware.

4510 MSR blocks with lower priority will be ignored; this is determined separately for the image sensor MSR  
4511 block and for the camera module MSR block.

4512      **B.2.10 Generic Rule Based Block**

4513      The purpose of the Generic Rule Based Block is to describe the values of capability and/or limit information  
 4514      based on value of some other register(s).

4515      *Table 273* specifies the fields used in records of the Generic Rule Based Block. A Generic Rule Based Block  
 4516      contains one or more consecutive records in this format.

4517      **Table 273 Description of a Record in the Generic Rule Based Block**

Field Name	Type	Comment
<b>Length Specifier</b>	Variable size	Length of Data field in bytes, see <b>Section B.2.2.1</b>
<b>Type</b>	8-bit unsigned integer	See <b>Table 274</b>
<b>Data</b>	Variable size	Content format determined by the Type field

4518      The defined record types are defined in *Table 274*.

4519      **Table 274 Generic Rule Based Block Record Types**

Name	Value	Consistency granularity	Comment
If Rule	1	N/A	Condition for the records following this record
Read-Only Register	2	Item	Conditional read-only registers
Frame Format Descriptor (FFD)	3	Record	Frame layout information, see <b>Section B.2.11</b>
Manufacturer Specific Register	4	Item	Conditional manufacturer specific registers
PDAF Readout	5	Record	PDAF Readout, see <b>Section B.2.13</b>

4520      Any condition in an If Rule (type 1) record applies to all records following that rule, until either the end of  
 4521      the block or the next If Rule record. A non-If-Rule record that is not preceded by an If Rule record is  
 4522      considered to be associated with a condition whose value is always True.

### B.2.10.1 Type 1: If Rule

In type 1, the Data field shall consist of:

- Number of rules
- Index
- If value
- Bitmask

**Table 275** specifies the fields used in Data field for Record Type 1 (If Rule). The fields describe the If Rule condition.

If rule == TRUE:

Action (at least one of the actions)

More precisely:

If AND( register\_value\_from\_index, bit\_mask ) == if\_value :

Action (at least one of the actions)

If more than one rule is used, then the rules are ANDed together:

If AND( rule 1, rule2, rule N ) == TRUE:

Action (at least one of the actions)

Note that all valid values of sensor runtime configurable registers shall have their specific If Rules to convey the required information to software whenever it deviates from the defaults in what is assumed in CCS Static Data or in CCS Read-only Registers. Valid values in this case means values that the register may have, as advertised in the sensor's read-only capability and limit registers or CCS Static Data. The same also applies to rules with multiple registers.

**Example:** The Rule Based Block contains an If Rule Record with rules **R & M = 0x01** where **M = 0x03**, which is followed by a Read-only Register Record depending on the rule. If the same (as well as other) read-only registers in the aforementioned record also have values deviating from the defaults in cases **R & M = 0x02** and **R & M = 0x03**, then If Rules for the two cases (i.e., case **R & M = 0x02** and case **R & M = 0x03**) shall be also included in the same Rule Based Block, on the condition that **0x02** and **0x03** are valid values of **R & M**. While is not necessary to add a specific rule for **R & M = 0x00** (assuming **0x00** is a valid value of **R & M**) if that is assumed elsewhere in CCS Static Data, it may still be the least error-prone way to express that information.

The If Rule Record payload consists of one or more consecutive rules. The data layout of a single rule is shown in **Table 275**.

**Table 275 If Rule**

Field Name	Type	Comment
<b>Index</b>	16-bit unsigned integer	Index of the register the value of which is compared against
<b>If value</b>	8-bit unsigned integer	Value that is used in the condition
<b>Bitmask</b>	8-bit unsigned integer	Bitmask to identify which bits of the data are used in the condition

Only standard CCS registers in the range [0x0000, 0x2FFF] shall be used as the value of the Index field in the rules. There may also be further restrictions in the records that the rules are applied to.

### B.2.10.2 Type 2: Read-Only Registers

Type 2 defines conditional read-only registers that may be specific to either the image sensor or the camera module, depending on the Generic Rule Based Block that they are contained in. In Type 2, the Common Register BLOB format is used to describe the content (same as Read Only Register Block). The purpose of Generic Rule Based Block Type 2 is to report image sensor capabilities as a function of CCS register(s). For example, certain limit parameter values may change as a function of binning or ADC bit depth. To avoid complexity on software side, Type 2 can only report the following specific parameters:

- `min_frame_length_lines`
- `max_frame_length_lines`
- `min_line_length_pck`
- `max_line_length_pck`
- `min_line_blanking_pck`
- `min_frame_blanking_lines`
- `min_line_length_pck_step_size`
- `x_addr_start_div_constant`
- `y_addr_start_div_constant`
- `x_addr_end_div_constant`
- `y_addr_end_div_constant`
- `x_size_div`
- `y_size_div`
- `x_output_div`
- `y_output_div`
- `analog_gain_code_min`
- `analog_gain_code_max`
- `analog_linear_gain_min`
- `analog_linear_gain_max`
- `analog_exponential_gain_min`
- `analog_exponential_gain_max`
- `digital_gain_min`
- `digital_gain_max`
- `coarse_integration_time_min`
- `coarse_integration_time_max_margin`
- `fine_integration_time_min`
- `fine_integration_time_max_margin`

The example shown in *Table 276* illustrates a conditional read-only register record using the Common Register BLOB.

4591

**Table 276 Example of Data in Type 2**

<b>Byte</b>	<b>Field</b>	<b>Bits</b>	<b>Value</b>	<b>Content</b>
0	Control field	7:6	2	Format selection field
		5:0	3	Data size field, value = size -1
1	Index field	15:8	0x11	Index of min_frame_length_lines
2		7:0	0x40	
3	Data field	15:8	e.g. 0x00	Data of index 0x1140 and 0x1141
4		7:0	e.g. 0xf0	
5		15:8	e.g. 0x3f	Data of index 0x1142 and 0x1143, i.e., max_frame_length_lines Hi and Lo
6		7:0	e.g. 0x7f	

**B.2.10.3 Type 3: Frame Format Descriptors**

4592

A Type 3 record contains a Frame Format Descriptor (FFD) as described in *Section B.2.11*.

**B.2.10.4 Type 4: Manufacturer Specific Registers**

4593

Type 4 defines conditional manufacturer specific registers. In this type, Common Register BLOB format is used to describe the content (same as MSR Block). The purpose of Generic Rule Based Block Type 4 is to describe MSR registers as a function of CCS register(s). See also *Section B.2.9*.

4594

4595

**B.2.10.5 Type 5: Frame Format Descriptors for PDAF**

4596

A Type 5 record contains Frame Format Descriptors for PDAF as described in *Section B.2.13*.

### B.2.10.6 Type 1 and Type 2 Examples

This example shows how certain parameter limits can take different values, based on a CCS register value. The purpose is to clarify how Type 1 and Type 2 (and/or Type 4) are used together, and the format of the Block.

The pseudocode for this example is:

```
if binning_type == 0x44, then
    use the Record content (new values for min_frame_length_lines and
    max_frame_length_lines)
```

The actual formula is:

```
if AND[(AND(register_value_of_0x0901, bitmask1) == 0x44),
       (AND(register_value_of_0x0901, bitmask2) == 0x00)]:
    #use the Read-Only Register Record content
```

**Table 277 Example of Generic Rule Based Block**

	Field Name	Content		Data Type	Value	Comment
Record 1	<b>Length Specifier</b>		Length Specifier	0x08	Record 1 Length of Data in bytes	
	<b>Type</b>		8-bit unsigned integer	1	Record Type 1: If Rule	
Data	First Rule	index	16-bit unsigned integer	0x0901	Index of CCS Register <b>binning_type</b>	
		if_value	8-bit unsigned integer	44	if_value for first rule	
		bitmask	8-bit unsigned integer	Bit string 0100 0100	If bitwise AND of register <b>binning_type</b> 's value with this bitmask equals the if_value, then first rule will be true	
	Second Rule	index	16-bit unsigned integer	0x0901	Second rule also uses same CCS Register index	
		if_value	8-bit unsigned integer	0	if_value for second rule	
		bitmask	8-bit unsigned integer	Bit string 1011 1011	If bitwise AND of register <b>binning_type</b> 's value with this bitmask equals the if_value, then second rule will be true	
Record 2	<b>Length Specifier</b>		Length Specifier	7	Record 2 Length of Data in bytes	
	<b>Type</b>		8-bit unsigned integer	2	Record Type 2: Read-Only Register	
Data	Control field		8-bit unsigned integer	0x83	Format selection field = 2 Data size field = 3	
	Index field		16-bit unsigned integer	0x1140	Index of CCS Register <b>min_frame_length_lines</b>	
	Data field		16-bit unsigned integer	e.g. 0x00	Value for CCS Register <b>min_frame_length_lines_Hi</b>	
				e.g. 0xf0	Value for CCS Register <b>min_frame_length_lines_Lo</b>	
			16-bit unsigned integer	e.g. 0x3f	Value for CCS Register <b>max_frame_length_lines_Hi</b>	
				e.g. 0x7f	Value for CCS Register <b>max_frame_length_lines_Lo</b>	

**4609      B.2.11 FFD Record**

4610 FFD Records may be used to, for example:

- 4611 • Convey CSI-2 frame layout to the Host if the image sensor is sending visible pixels and OB pixels  
4612 using same VC/DT, or
- 4613 • Configure the Host system, taking into account CSI-2 packet sizes

4614 FFD Records are contained in the Generic Rule Based Block, as defined in **Section B.2.10**.

4615 If an image sensor's FFDs are not affected by disabling and enabling its features, then the FFDs may be put  
4616 into an FFD Record in the Generic Rule Based Block without a related If Rule.

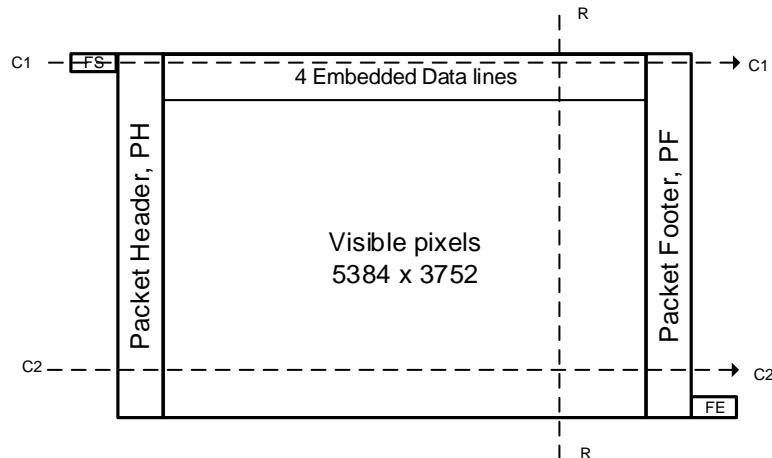
4617 On the other hand, if the sensor's FFDs are affected by features that can be enabled, disabled, or otherwise  
4618 configured at runtime, then an FFD Record shall be present for each combination of the configuration of such  
4619 features in the Generic Rule Based Block. Such features may include, for example, OB pixel readout and  
4620 bottom-embedded data PDAF readout. **Section B.2.14** clarifies FFD Record usage when a Generic Rule  
4621 Based Block includes more than one FFD Record.

4622 FFDs include multiple column descriptor groups. One column descriptor group shall be used for each line  
4623 that intersects with a row descriptor. For example, if a row descriptor includes 4 FFDs, then there shall be 4  
4624 column descriptor groups. Groups are ordered from the top to the bottom. The first column descriptor is  
4625 described starting from the top left corner, as shown in the **Figure 94** example.

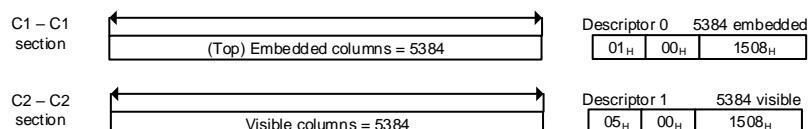
4626 In FFDs, it is assumed that the visible pixels use the maximum resolution defined by **x\_addr\_min**,  
4627 **y\_addr\_min**, **x\_addr\_end**, and **y\_addr\_end**.

4628 Four examples are shown: two examples with simple FFD, and two examples with advanced FFD usage  
4629 where FFDs depends on the value of a CCS Register. FFDs shall use the 4-Byte Extended Frame Format  
4630 Description specific to CCS Static Data (see **Section B.5.1**). When using FFDs, the number of visible pixels  
4631 is assumed to be static. See **Section B.2.13** for rules regarding how PDAF is reported.

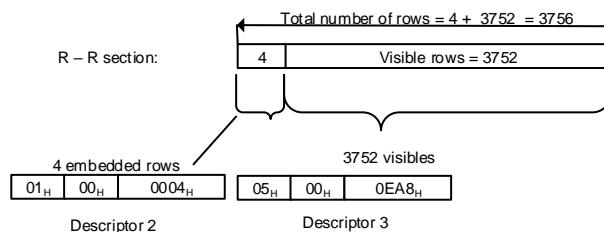
4632  
4633  
4634 **Table 278** and **Figure 91** show a simple, one-record example of FFD usage where the image frame contains neither OB pixels nor bottom-embedded data. This can happen if OB pixels are not supported, or if OB pixel readout has not been enabled.



Number of Column Descriptors: 2



Number of Row Descriptors: 2



4635

4636

**Figure 91 Example of Simple FFD Usage**

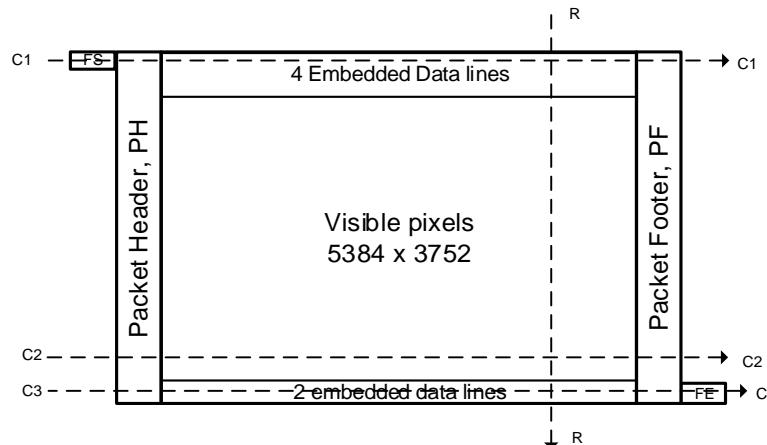
4637

**Table 278 Example of Simple FFD Usage**

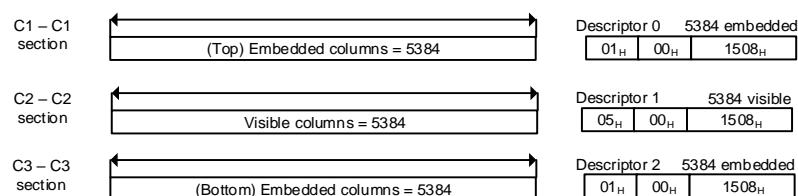
Field Name	Content	Data Type	Value	Comment
<b>Length Specifier</b>	Length specifier	0x12	Length of Data in bytes	
<b>Type</b>	8-bit unsigned integer	3	Record Type 3: FFD	
<b>Data</b>	frame_format_model_subtype	16-bit unsigned integer	0x02	Number of Column Descriptors
			0x02	Number of Row Descriptors
FFD_0		32-bit unsigned integer	0x01	PixelCode: Embedded Data
			0x00	Reserved
			0x15	Size: 5384
			0x08	
FFD_1		32-bit unsigned integer	0x05	PixelCode: Visible
			0x00	Reserved
			0x15	Size: 5384
			0x08	
FFD_2		32-bit unsigned integer	0x01	PixelCode: Embedded Data
			0x00	Reserved
			0x00	Size: 4
			0x04	
FFD_3		32-bit unsigned integer	0x05	PixelCode: Visible
			0x00	Reserved
			0x0E	Size: 3752
			0xA8	

4638  
4639  
4640  
**Table 279** and **Figure 92** show a second simple FFD usage example in which the image frame does not contain OB pixels, but does contain bottom-embedded data. For complete usage of FFD Records in this use case, see **Section B.2.14**, e.g. **Section B.2.14.4**.

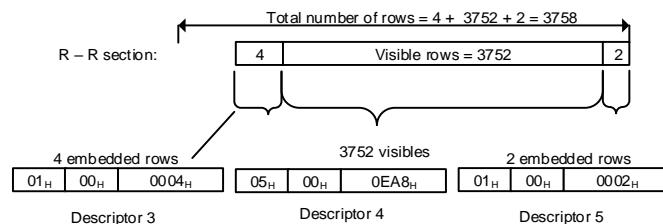
4641 Note that the image sensor in this example does not support interleaved PDAF Readout. For this reason, that option is not covered by the If Rules in the table.  
4642



Number of Column Descriptors: 3



Number of Row Descriptors: 3



4643

4644

**Figure 92 Example 2 of Simple FFD Usage**

4645

**Table 279 Example 2 of Simple FFD Usage**

	Field Name	Content	Data Type	Value	Comment
Record 1	Length Specifier	Length specifier	0x04	Length of Data in bytes	
	Type	8-bit unsigned integer	1	Record Type 1: If Rule	
	Data	index	16-bit unsigned integer	0x0D00	Index of CCS Register <b>PDAF_ctrl</b>
		if_value	8-bit unsigned integer	1	if_value for this rule
		bitmask	8-bit unsigned integer	Bit string 0000 0001	If bitwise AND of register <b>PDAF_ctrl</b> 's value with this bitmask equals the if_value, then this rule will be true
Record 2	Length Specifier	Length specifier	0x1A	Length of Data in bytes	
	Type	8-bit unsigned integer	3	Record Type 3: FFD	
	Data	frame_format_model_subtype	16-bit unsigned integer	0x03	Number of Column Descriptors
				0x03	Number of Row Descriptors
		FFD_0	32-bit unsigned integer	0x01	PixelCode: Embedded Data
				0x00	Reserved
				0x15	Size: 5384
				0x08	
		FFD_1	32-bit unsigned integer	0x05	PixelCode: Visible
				0x00	Reserved
				0x15	Size: 5384
				0x08	
	FFD_2	32-bit unsigned integer	0x01	PixelCode: Embedded	
			0x00	Reserved	
			0x15	Size: 5384	
			0x08		
	FFD_3	32-bit unsigned integer	0x01	PixelCode: Embedded Data	
			0x00	Reserved	
			0x00	Size: 4	
			0x04		
	FFD_4	32-bit unsigned integer	0x05	PixelCode: Visible	
			0x00	Reserved	
			0x0E	Size: 3752	
			0xA8		
	FFD_5	32-bit unsigned integer	0x01	PixelCode: Embedded data	
			0x00	Reserved	
			0x00	Size: 2	
			0x02		

4646  
4647  
4648  
4649 **Table 281** and **Figure 94** show a more advanced FFD usage example. In this example, the image frame includes OB pixels when OB pixel readout has been enabled, and bottom-embedded data is not included. For complete information regarding the usage of FFD Records in this use case, see **Section B.2.14**, e.g. **Section B.2.14.3**.

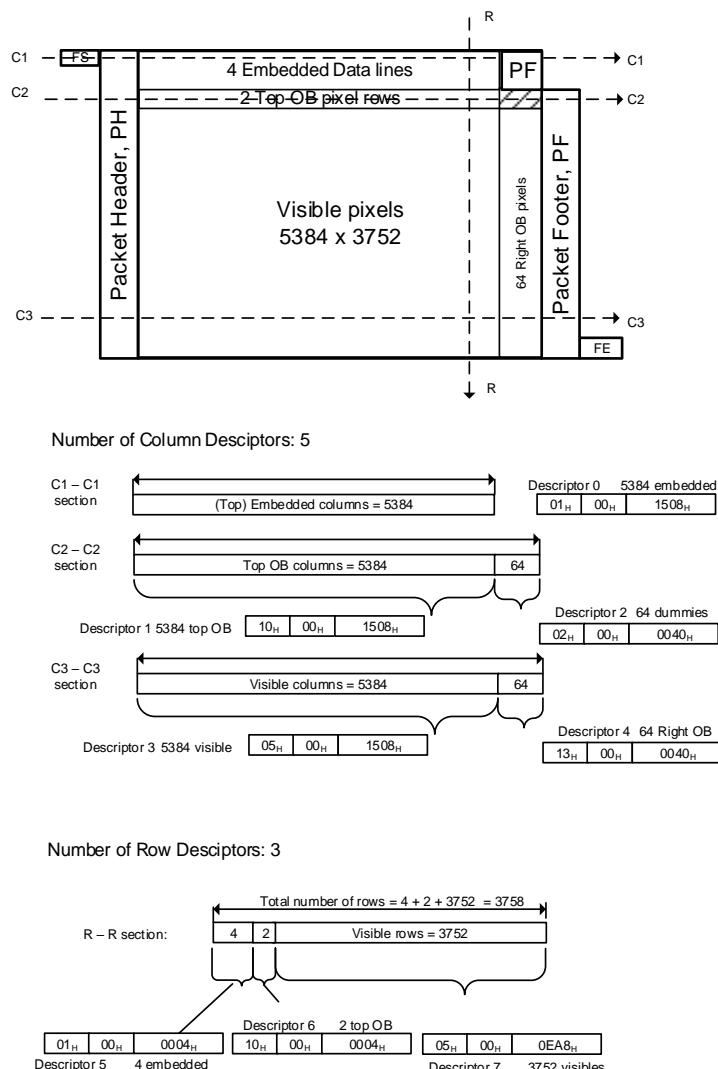
4650 The pseudocode for the advanced example is:

```
4651 if OB_readout_ctrl(0) == 1, then
4652     use the Record content (FFDs)
```

4653 The actual formula is:

```
4654 if and(register_value_of_0x0B30, bitmask) == 0x01:
4655     #use the Record content (FFDs)
```

4656 Note that the image sensor may send shorter embedded data lines instead of other data lines, as shown in  
4657 **Figure 94**. In this example it is assumed that OB pixels and visible pixels are readout using the same data  
4658 types, i.e., using the same CSI-2 Long Packets, which must have equal length. The embedded data lines are  
4659 using the same length as the simple FFD example presented in **Figure 92**.



4660  
4661 **Figure 93 Example of Advanced FFD Usage**

4662

**Table 280 Example of Advanced FFD Usage**

Field Name	Content	Data Type	Value	Comment
Record 1	Length Specifier	Length Specifier	0x04	Length of Data in bytes
	Type	8-bit unsigned integer	1	Record Type 1: If Rule
	Data	index	0x0B30	Index of CCS Register <b>OB_readout_ctrl</b>
		if_value	1	If_value for this rule
Record 2	Length Specifier	8-bit unsigned integer	0x22	Length of Data in bytes
		Type	3	Record Type 3: FFD
	Data	frame_format_model_subtype	0x06	Number of Column Descriptors
			0x04	Number of Row Descriptors
		FFD_0	0x01	PixelCode: Embedded Data
			0x00	Reserved
			0x15	Size: 5384
			0x08	
		FFD_1	0x10	PixelCode: Top Optical Black
			0x00	Reserved
			0x15	Size: 5384
			0x08	
		FFD_2	0x02	PixelCode: Dummy
			0x00	Reserved
			0x00	Size: 64
			0x40	
		FFD_3	0x05	PixelCode: Visible
			0x00	Reserved
			0x15	Size: 5384
			0x08	
		FFD_4	0x13	PixelCode: Right Optical Black
			0x00	Reserved
			0x00	Size: 64
			0x40	
		FFD_5	0x01	PixelCode: Embedded Data
			0x00	Reserved
			0x00	Size: 4
			0x04	
		FFD_6	0x10	PixelCode: Top Optical Black
			0x00	
			0x00	Size: 2
			0x02	
		FFD_7	0x05	PixelCode: Visible
			0x00	Reserved
			0x0E	Size: 3752
			0xA8	

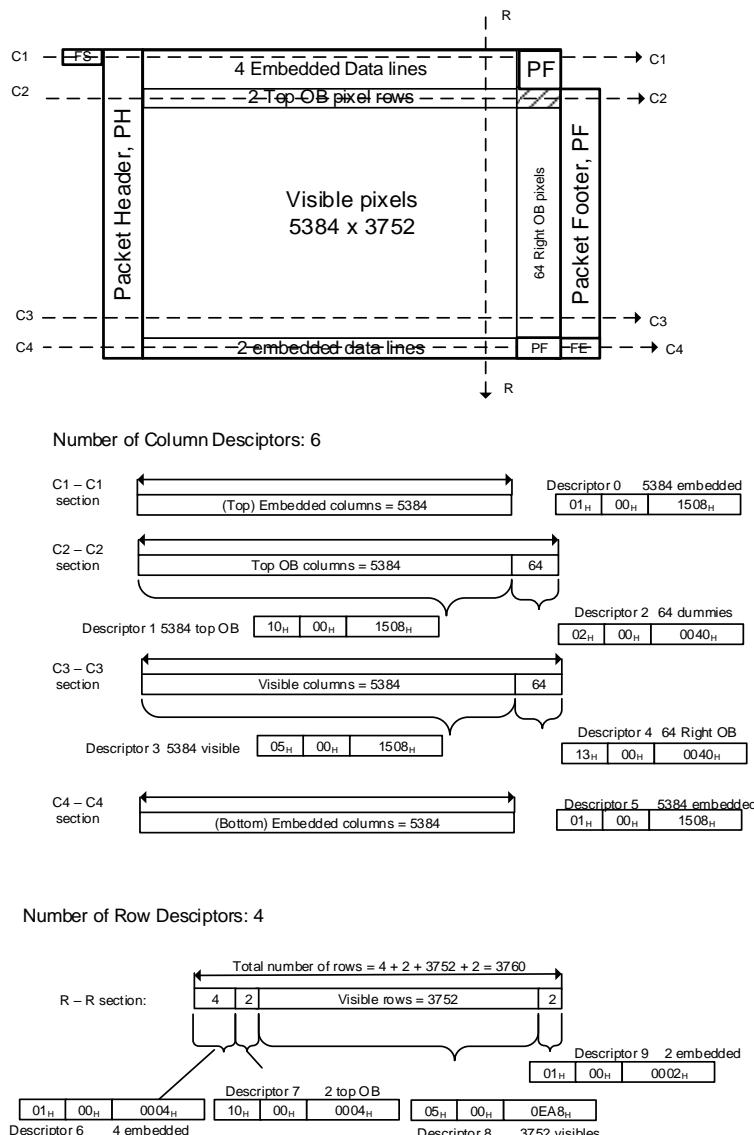
In **Table 281** and **Figure 94**, a second advanced FFD usage example is shown. In this example, the image frame includes both OB pixels when OB pixel readout has been enabled, and bottom-embedded data. For complete information regarding the usage of FFD Records in this use case, see **Section B.2.14**, e.g. **Section B.2.14.5**.

The pseudocode for the advanced example is:

```
if OB_readout_ctrl(0) == 1 and PDAF_ctrl(0) == 1, then
    use the Record content (FFDs)
```

Note that the image sensor may send shorter embedded data lines instead of other data lines, as shown in **Figure 94**. In this example it is assumed that OB pixels and visible pixels are readout using the same data types, i.e., using the same CSI-2 Long Packets, which must have equal length. The embedded data lines are using the same length as in the simple FFD example presented in **Figure 92**.

Also note that the image sensor in this example does not support interleaved PDAF Readout. For this reason, that option is not covered by the If Rules in the table.



**Figure 94 Example 2 of Advanced FFD Usage**

4677

**Table 281 Example 2 of Advanced FFD Usage**

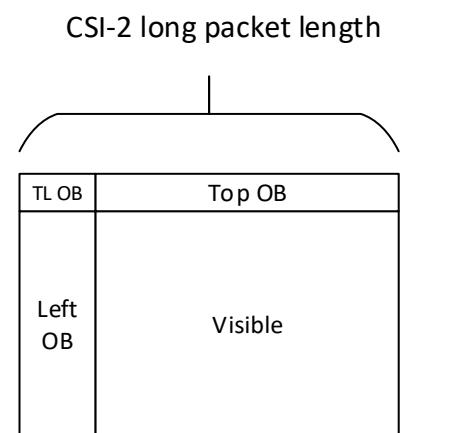
Field Name	Content		Data Type	Value	Comment	
Record 1	Length Specifier		Length Specifier	0x08	Length of Data in bytes	
	Type		8-bit unsigned integer	1	Record Type 1: If Rule	
	Data	First Rule	index	16-bit unsigned integer	0x0B30 Index of CCS Register <b>OB_readout_ctrl</b>	
			if_value	8-bit unsigned integer	1 If_value for first rule	
			bitmask	8-bit unsigned integer	Bit string 0000 0001 If bitwise AND of register <b>OB_readout_ctrl</b> 's value with this bitmask equals the first rule if_value, then the first rule will be true	
	Data	Second Rule	index	16-bit unsigned integer	0x0D00 Index of CCS Register <b>PDAF_ctrl</b>	
			if_value	8-bit unsigned integer	1 If_value for second rule	
			bitmask	8-bit unsigned integer	Bit string 0000 0001 If bitwise AND of register <b>PDAF_ctrl</b> 's value with this bitmask equals the second rule if_value, then the second rule will be true	
Record 2	Length Specifier		Length Specifier	0x2A	Length of Data in bytes	
	Type		8-bit unsigned integer	3	Record Type 3: FFD	
	Data	frame_format_model_subtype		16-bit unsigned integer	0x06 Number of Column Descriptors	
				0x04	Number of Row Descriptors	
		FFD_0		32-bit unsigned integer	0x01 PixelCode: Embedded Data	
				0x00	Reserved	
				0x15	Size: 5384	
				0x08		
		FFD_1		32-bit unsigned integer	0x10 PixelCode: Top Optical Black	
				0x00	Reserved	
				0x15	Size: 5384	
				0x08		
		FFD_2		32-bit unsigned integer	0x02 PixelCode: Dummy	
				0x00	Reserved	
				0x00	Size: 64	
				0x40		
		FFD_3		32-bit unsigned integer	0x05 PixelCode: Visible	
				0x00	Reserved	
				0x15	Size: 5384	
				0x08		
		FFD_4		32-bit unsigned integer	0x13 PixelCode: Right Optical Black	
				0x00	Reserved	
				0x00	Size: 64	
				0x40		
		FFD_5		32-bit unsigned integer	0x01 PixelCode: Embedded data	
				0x00	Reserved	
				0x15	Size: 5384	
				0x08		

Field Name	Content	Data Type	Value	Comment
	FFD_6	32-bit unsigned integer	0x01	PixelCode: Embedded Data
			0x00	Reserved
			0x00	Size: 4
			0x04	
	FFD_7	32-bit unsigned integer	0x10	PixelCode: Top Optical Black
			0x00	
			0x00	Size: 2
			0x02	
	FFD_8	32-bit unsigned integer	0x05	PixelCode: Visible
			0x00	Reserved
			0x0E	Size: 3752
			0xA8	
	FFD_9	32-bit unsigned integer	0x01	PixelCode: Embedded data
			0x00	Reserved
			0x00	Size: 2
			0x02	

4678 One purpose of extended FFDs is to support description of OB pixel readout, e.g., when OB pixels are located  
 4679 in two or more sides of visible pixels. An alternative and/or additional purpose is to support description of  
 4680 OB pixel readout order, which may be different from OB pixel physical location. The following examples  
 4681 show OB pixel readout location compared to visible pixels in certain examples.

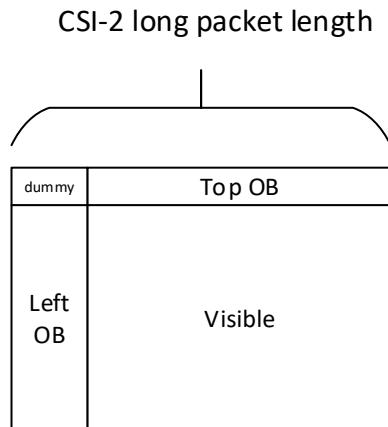
4682 If OB readout uses the same DT as visible pixels, then the CSI-2 Long Packets shall have the same size (i.e.,  
 4683 as when using RAW data format).

4684 For example, if top OB pixels, left OB pixels, and top left OB pixels are supported, then the top left OB  
 4685 pixels shall be labeled in FFD Records as TL OB pixels, even if they are readout in the same CSI-2 packet  
 4686 as top OB pixels. In that case, the height of the TL OB shall be the same as as the height of the top OB, and  
 4687 the height of the left OB shall be the same as the height of the visible pixels. This is shown in **Figure 95**.



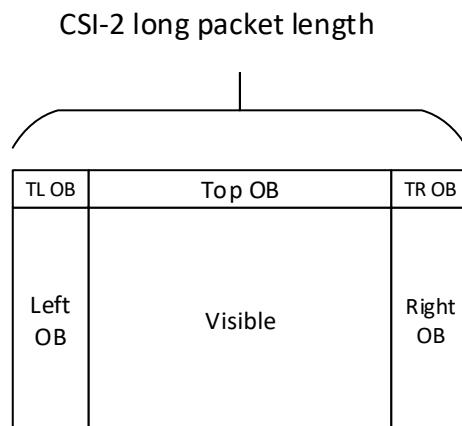
4688 **Figure 95 Top Left OB Pixel Example 1**

4689 Dummy pixels can be used to match the Long Packet lengths. For example, if only top OB pixels and left  
4690 OB pixels are supported, then dummy pixels shall be used to match the Long Packet length. The top length  
4691 will be the same as the visible height. The dummy pixels should be located so that the top OB pixels match  
4692 the relevant visible pixels; i.e., typically the width of the dummy pixels is the same as the width of the left  
4693 OB pixels.



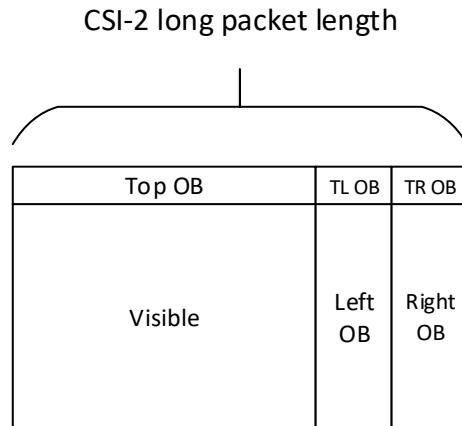
4695 **Figure 96 OB Pixel Example with Dummy Pixels**

4696 The main purpose of labeling the corner OB pixels is to be able to separate physical OB pixel location from  
4697 OB readout order. For example, in **Figure 97** the Left OB pixels are readout before the Visible pixels, i.e., as  
4698 shown in examples of **Figure 95** and **Figure 96**.



4700 **Figure 97 OB Pixel Example with Native Readout Order**

4701 In the **Figure 98** example, both the Left OB pixels and the right OB pixels are readout after the Visible pixels.  
4702 As the FFD Records use specific Pixel Codes for OB pixels, both the physical location and the readout order  
4703 can be detected from the FFDs.



**Figure 98 OB Pixel Example with Different Readout Order**

4706 If the OB pixels readout uses interleaving then it should follow the same rules as above, so that the Host can  
4707 easily identify OB pixel and visible pixel alignment.

4708      **B.2.12 PDAF Pixel Location Block**

4709      The purpose of the PDAF Pixel Location Block is to describe PDAF pixel locations in the physical pixel  
4710      array. This information may be used, e.g., to understand PDAF pixel locations compared to e.g. AF ROI. In  
4711      addition, the Host may want to use sensor-side cropping that does not split PDAF pixels in an unwanted way  
4712      (e.g., in case of sparsely-located half-shielded PDAF pixels).

4713      PDAF pixel locations are described by a flexible mechanism. This mechanism includes a global part, one or  
4714      more Block Descriptor Groups, and one or more Block ID properties. Block Descriptor Groups are ordered  
4715      from the top down to the bottom of the pixel array.

4716      The global part includes:

- 4717      • Main offset (x,y)
- 4718      • Global PDAF type
- 4719      • Block size (x, y)
- 4720      • Number of block descriptor groups

4721      For each Block Descriptor Group:

- 4722      • N\_desc\_count: The number of Block Descriptors in the X-direction
- 4723      • Repeat\_Y: How many times the Block Descriptor Group is repeated in the Y-direction
- 4724      • For each Block Descriptor in the X-direction:
  - 4725          • Block Type ID
  - 4726          • Repeat\_x: How many times this Block Descriptor is repeated in the X-direction

4727      After the Block Descriptor Groups, a list of Block ID properties is provided.

4728      For each Block ID:

- 4729      • Number of PDAF Pixel Descriptors within the block
- 4730      • For each PDAF Pixel Descriptor:
  - 4731          • PDAF Pixel Type
  - 4732          • Offset coordinates (x,y)

4733

**Table 282 Structure of PDAF Location Block**

Field Name	Type	Comment
<b>Main Offset</b>	32-bit unsigned integer	Main_offset_x Main_offset_y Calculated from x_addr_min and y_addr_min
<b>Global PDAF Type</b>	8-bit unsigned integer	Describe the type of PDAF Pattern. In the unlikely event that the sensor includes a combination of multiple types, report one Global type and then report the others individually. <b>0: Separated PDAF Pixels</b> E.g. half shielded. All pixels must be reported. <b>1: Side-by-Side PDAF Pixels</b> When PDAF pixels are side-by-side (whether left-right or top-bottom), only one is reported. If the same image sensor includes both left-right and top-bottom, then report the left and top ones. <b>2: Multiple Side-by-Side PDAF Pixels</b> When more than 2 PDAF pixels are side-by-side, 2x2 pattern is supported, report e.g. top left.
<b>Block Size</b>	16-bit unsigned integer	Width Height Unit: pixels
<b>Number of Block Descriptor Groups</b>	16-bit unsigned integer	–
<b>Block Descriptor Group(s)</b>	Variable size	At least one Block Descriptor Group
<b>Block ID Properties</b>	Variable size	At least one Block ID Property

4734 One Block Descriptor Group specifies one line of Blocks. That line may be repeated in the Y-direction (from top to bottom)  $n$  times. Depending on the value of the **Number of Block Descriptor Groups** field, additional Block Descriptor Groups may follow. **Table 283** defines the content of a Block Descriptor Group.

4737

**Table 283 Content of Block Descriptor Group**

Field Name	Type	Comment
<b>N_desc_count</b>	16-bit unsigned integer	Number of Block Descriptors in this Block Descriptor Group. <b>Minimum Value:</b> 1
<b>Repeat_Y</b>	8-bit unsigned integer	Number of times the Block Descriptor Group is repeated in the Y-direction
<b>Block Descriptor(s)</b>	Variable size	Each Block Descriptor includes the fields Block Type ID and Repeat_X. The first Block Descriptor describes the leftmost Block in a given line, and the last Block Descriptor describes the rightmost Block in that same line.
<i>Block Type ID</i>	8-bit unsigned integer	Subfield of the Block Descriptor This is an ID number using continuous numbering, starting from 0 and running until $n$ : 0, 1, 2, ... $n-1$ , $n$ .
<i>Repeat_X</i>	16-bit unsigned integer	Subfield of the Block Descriptor Number of times this Block is repeated to the right. 0: Not repeated. $n$ : Repeated $n$ times

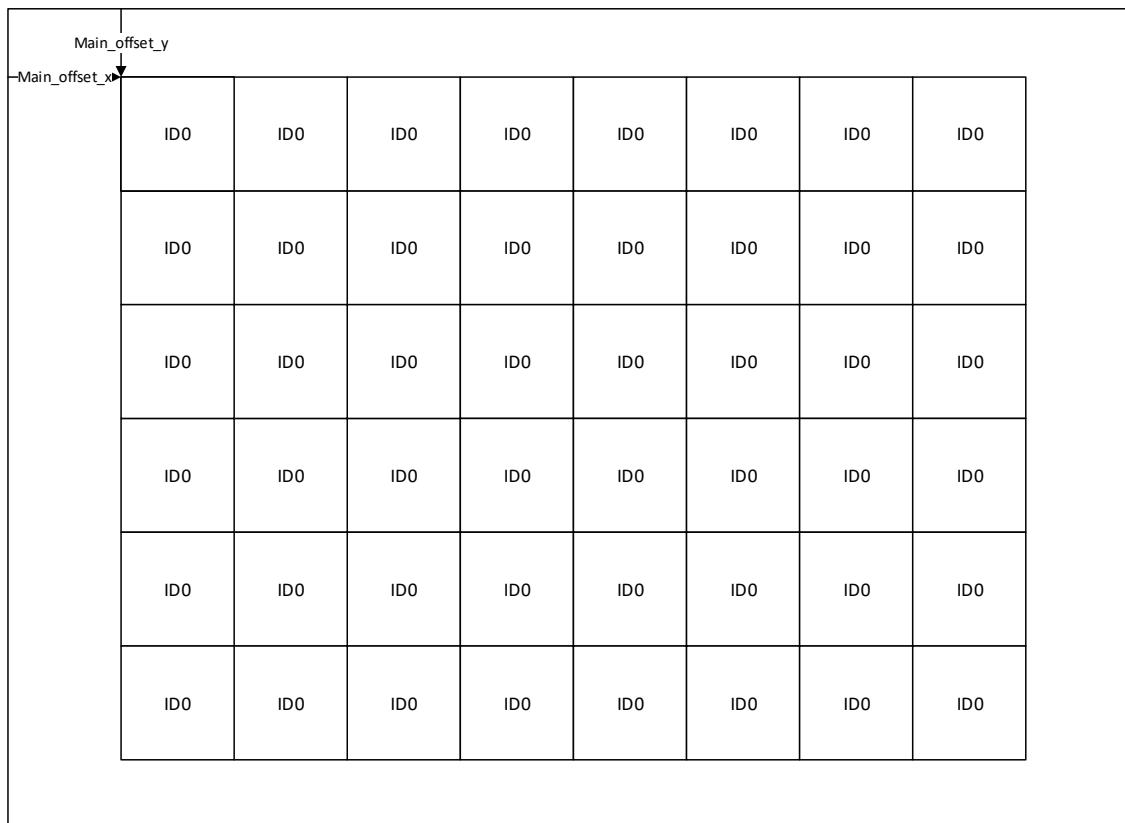
4738  
4739  
4740 After the Block Descriptor Groups, a list of Block ID properties is provided. Note that Block IDs shall be ordered from smallest to largest, i.e.: 0, 1, 2, ...  $n-1$ ,  $n$ . **Table 284** specifies the content of Block ID Properties.

4741

**Table 284 Content of Block ID Properties**

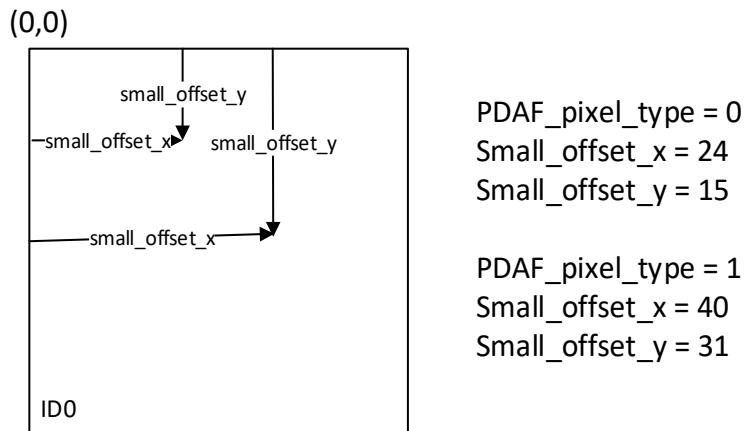
Field Name	Type	Comment
<b>Number of PDAF Pixel Descriptors</b>	8-bit unsigned integer	Number of PDAF pixel descriptors within Block Type ID $n$ . For each Block Type ID, a Number of PDAF Pixel Descriptors and one or more PDAF Pixel Descriptor(s) are provided.
<b>PDAF Pixel Descriptor(s)</b>	Variable size	Each PDAF Pixel Descriptor includes a PDAF Pixel Type and an Offset Coordinates fields.
<i>PDAF pixel type</i>	8-bit unsigned integer	Part of a PDAF Pixel Descriptor: 0: Left (separated) 1: Right (separated) 2: Top (separated) 3: Bottom (separated) 4: Left (side-by-side) 5: Right (side-by-side) 6: Top (side-by-side) 7: Bottom (side-by-side) 8: Top-left (TL) 9: Top-right (TR) 10: Bottom-left (BL) 11: Bottom-right (BR)
<i>Offset Coordinates (x,y)</i>	16-bit unsigned integer	Part of PDAF pixel descriptor small_offset_x small_offset_y

4742      **Table 285, Figure 99, and Figure 100** show an example of how the PDAF Block can be used to describe a  
4743      simple PDAF pattern.



4744      **Figure 99 Example of Simple PDAF Block Pattern**

4745 In **Figure 100**, only one Block Type is used. Each block will include the same PDAF pixel arrangement.  
4746 **Figure 100** shows an example of a Block, including one left half-shielded pixel and one right half-shielded  
4747 pixel. The local offsets are calculated from the local origin (0,0) at the top left corner. One block can include  
4748 several of the same type of PDAF pixels, even though this example shows only one. **Table 285** describes the  
4749 content of the PDAF Location Block used in this example.



4750

4751

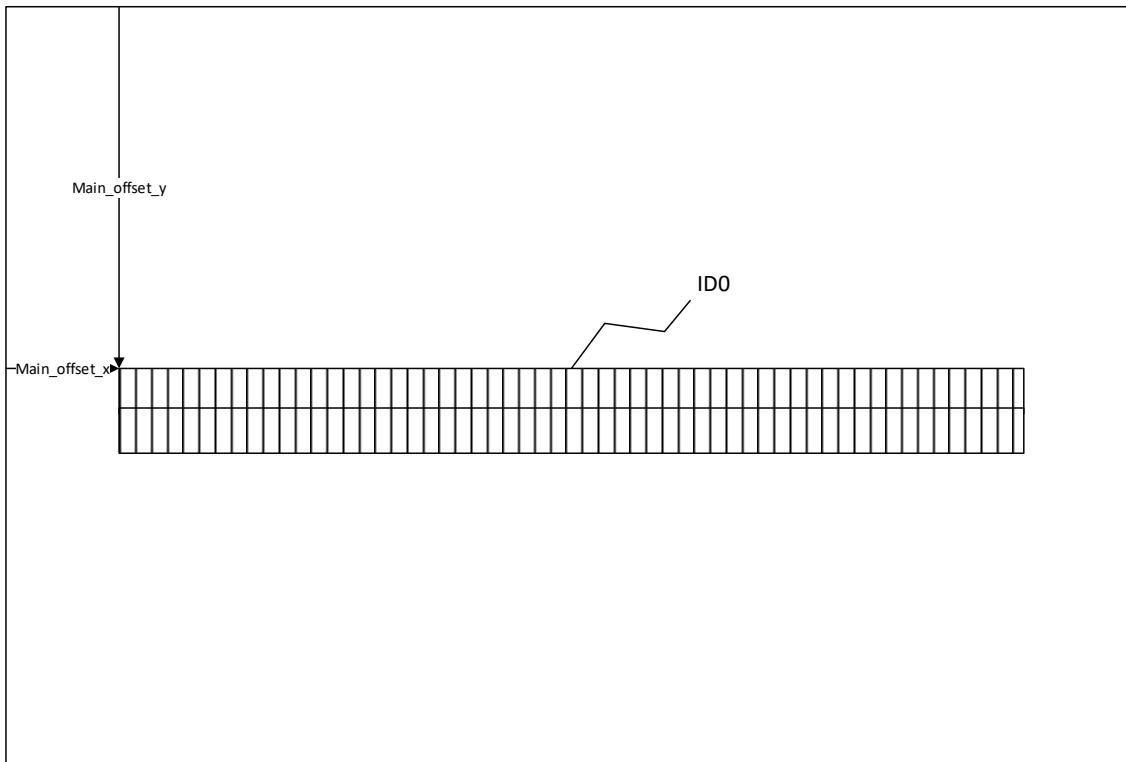
**Figure 100 Example of Simple PDAF Arrangement in Block**

4752

**Table 285 Content of PDAF Location Block in Simple Example**

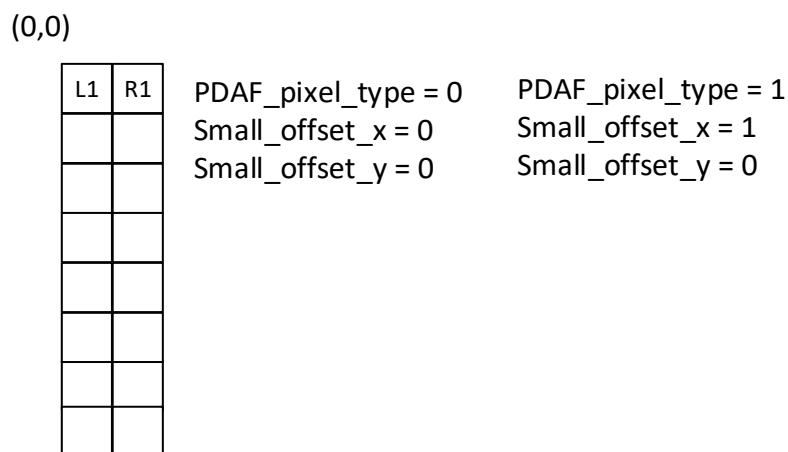
Field Name	Content	Data Type	Value	Comment
<b>Main Offset</b>	Main_offset_x	16-bit unsigned integer	64	—
	Main_offset_y	16-bit unsigned integer	48	—
<b>Global PDAF Type</b>		8-bit unsigned integer	0	Describes the type of PDAF pattern. In unlikely event that the image sensor includes a combination of multiple types, report one Global type and then report the others individually. 0: Separated PDAF pixels e.g. half shielded, all pixels must be reported
<b>Block Size</b>	Block_width	8-bit unsigned integer	64	Unit: pixels
	Block_height	8-bit unsigned integer	64	Unit: pixels
<b>Number of Block Descriptor Groups</b>		16-bit unsigned integer	1	At least one Block Descriptor Group
<b>Block Descriptor Group(s)</b>	<b>N_desc_count</b>	16-bit unsigned integer	1	Number of Block Descriptors in this Block Descriptor Group Minimum Value: 1
	<b>Repeat_Y</b>	8-bit unsigned integer	5	There is a total of 6 rows of Blocks. 1 + repeat 5 times = 6.
	<i>Block Type ID</i>	8-bit unsigned integer	0	Block ID number. Continuous numbering is used, starting from 0 and running until <i>n</i> : 0, 1, 2, ... <i>n</i> -1, <i>n</i> .
	<b>Repeat_X</b>	16-bit unsigned integer	7	Number of times this Block is repeated to the right. 0: Not repeated <i>n</i> : Repeated <i>n</i> times There is a total of 8 Blocks per row. 1 + repeat 7 times = 8.
<b>Block ID Properties</b>	Number of PDAF Pixel Descriptors for Block Type ID 0	8-bit unsigned integer	2	—
<i>PDAF Pixel Descriptor 1</i>	<i>PDAF Pixel Type</i>	8-bit unsigned integer	0	0: Left (separated)
	<i>Offset Coordinates (x,y)</i> small_offset_x	8-bit unsigned integer	24	—
	<i>Offset Coordinates (x,y)</i> small_offset_y	8-bit unsigned integer	15	—
<i>PDAF Pixel Descriptor 2</i>	<i>PDAF Pixel Type</i>	8-bit unsigned integer	1	1: Right (separated)
	<i>Offset coordinates (x,y)</i> small_offset_x	8-bit unsigned integer	40	—
	<i>Offset coordinates (x,y)</i> small_offset_y	8-bit unsigned integer	31	—

4753 PDAF pixels may also be located in a line or lines, as shown in the example of **Figure 103**, **Figure 102**, and  
4754 **Table 286**.



**Figure 101 Example of Line PDAF Block Pattern**

4755 In **Figure 103**, only one Block Type is used. Each block will include the same PDAF pixel arrangement.  
4756 **Figure 102** shows an example of a Block, including one left half-shielded pixel and one right half-shielded  
4757 pixel. The local offsets are calculated from the local origin (0,0) from the top left corner. In this example,  
4758 PDAF pixels are located in two lines. The block width is 2 pixels and the block height is 8 pixels. By repeating  
4759 the block line once, two PDAF pixel lines are covered. **Table 286** describes the content of the PDAF Location  
4760 Block for this example.  
4761



**Figure 102 Example of Line PDAF Arrangement in Block**

4764

**Table 286 Content of PDAF Location Block in Line Example**

Field Name	Content	Data Type	Value	Comment
Main Offset	Main_offset_x	16-bit unsigned integer	64	—
	Main_offset_y	16-bit unsigned integer	48	—
Global PDAF type		8-bit unsigned integer	0	Describes the type of PDAF Pattern. In the unlikely event that the image sensor includes a combination of multiple types, report one Global type and then report the others individually. 0: Separated PDAF pixels, e.g. half shielded. All pixels must be reported.
Block size	Block_width	8-bit unsigned integer	2	Unit: pixels
	Block_height	8-bit unsigned integer	8	Unit: pixels
Number of Block Descriptor Groups		16-bit unsigned integer	1	At least one Block Descriptor Group
Block Descriptor Group(s)	N_desc_count	16-bit unsigned integer	1	Number of Block Descriptors in this Block Descriptor Group Minimum value: 1
	Repeat_Y	8-bit unsigned integer	1	There is a total of 2 rows of Blocks. 1 + repeat 1 times = 2
	Block type ID	8-bit unsigned integer	0	Block ID number. Use continuous numbering, starting from 0 and running until $n$ : 0, 1, 2, ..., $n-1$ , $n$
	Repeat_X	16-bit unsigned integer	255	Number of times this Block is repeated to the right. 0: Not repeated $n$ : Repeated $n$ times There is a total of 256 Blocks per row. 1 + repeat 255 times = 256
Block ID properties	Number of PDAF pixel descriptors for Block type ID 0	8-bit unsigned integer	2	—
	PDAF Pixel Descriptor 1	PDAF pixel type	0	0: Left (separated)
		Offset coordinates (x,y) small_offset_x	0	—
		Offset coordinates (x,y) small_offset_y	0	—
	PDAF Pixel Descriptor 2	PDAF pixel type	1	1: Right (separated)
		Offset coordinates (x,y) small_offset_x	1	—
		Offset coordinates (x,y) small_offset_y	0	—

4765

### B.2.13 PDAF Readout Record

4766

The purpose of the PDAF Readout Record in the Generic Rule Based Block is to describe the details of how PDAF pixels are readout if PDAF readout uses the interleaved readout option. I.e., data is send out after each relevant image line during line blanking using DT/VC interleaving. If PDAF readout uses the bottom-embedded data readout option, then it shall be described with FFD Records. If the image sensor is not capable of reading out PDAF separately from visible pixels, then neither FFD Records nor PDAF Readout Records shall be used to describe PDAF.

4772

Note that the PDAF lines may exist only after certain image data lines. In addition, if PDAF ROI or sensor-side cropping is used, then it may reduce the number of PDAF lines sent out (see [Section 17.2.2.2](#)).

4774

PDAF Readout Records use a format similar to FFD Records, with the exception that each PDAF Readout Record contains two fields, PDAF\_readout\_info and FFD(s), as shown in [Table 287](#). As the PDAF Readout Record is only relevant if PDAF with interleaved readout is enabled, no If-Rule is required to make this record depend on register **PDAF\_ctrl** where PDAF with interleaved readout is configured. The FFD Record shall be used to describe the default frame format, i.e. the frame format used when PDAF readout is disabled (see [Section B.2.11](#) and [Section B.2.14](#)).

4780

Both Column FFDs and Row FFDs are used.

4781

- **Column FFDs** describe:

4782

- The location of PDAF data compared to other data (left-to-right intersection) in lines for which PDAF data exists, and
- The width of PDAF data lines.

4783

- **Row FFDs** describe:

4784

- The number of PDAF data lines (top-to-bottom intersection)

4785

Additional information is provided to describe different PDAF readout types. Three different readout options are supported:

4786

- Original order, i.e. in order of physical location
- Separate types within line order
- Separate types to separate type lines

4787

**Table 287 Content of PDAF Readout Record**

Field name	Type	Comment
<b>PDAF_readout_info</b>	16-bit unsigned integer	<b>Bits 0-7</b> 0: Reserved 1: Original order 2: Separate types within line order 3: Separate types to separate type lines <b>Bits 8-15</b> Reserved
<b>FFDs</b>	variable	4-byte extended FFDs, with PDAF Readout Record specific rules

4793 PDAF FFDs have certain specific rules, compared to generic FFD rules. The FFD rules for each readout  
4794 option are described separately.

#### 4795 **FFD Rules for Original Order**

- 4796 1. Pixel Code “Original order PDAF” shall be used for both Row and Column Descriptors
- 4797 2. Each row shall include the PDAF pixels of that line, in original order.  
4798 I.e., if the same row includes e.g. both Left PDAF pixels and Right PDAF pixels, then their  
4799 readout order corresponds to the physical order.
- 4800 3. There shall be as many Column Descriptor Groups as there are different PDAF data rows.
  - 4801 A. Rows shall be considered different if their lengths are different.
  - 4802 B. Rows shall not be considered different if the number of PDAF pixels stays the same, even if  
4803 the order changes
  - 4804 C. Note that some rows may include dummy pixels if a number of PDAF pixels in a row are  
4805 different compared to other lines; i.e., PDAF and dummy column descriptor may be included  
4806 for a shorter line.
- 4807 4. There shall be one Row Descriptor Group intersecting with the beginning of PDAF lines.
- 4808 5. Each Row Descriptor shall report the height of the intersecting Column Descriptor Group.
- 4809 6. If multiple Row Descriptors are used, then an additional descriptor with Pixel code “Total” shall  
4810 also be used to describe the complete height (which consists of repeating Column Descriptor  
4811 Groups, in order, up to the total height reported by the Total Descriptor).

#### 4812 **FFD Rules for Separate Types Within Line Order**

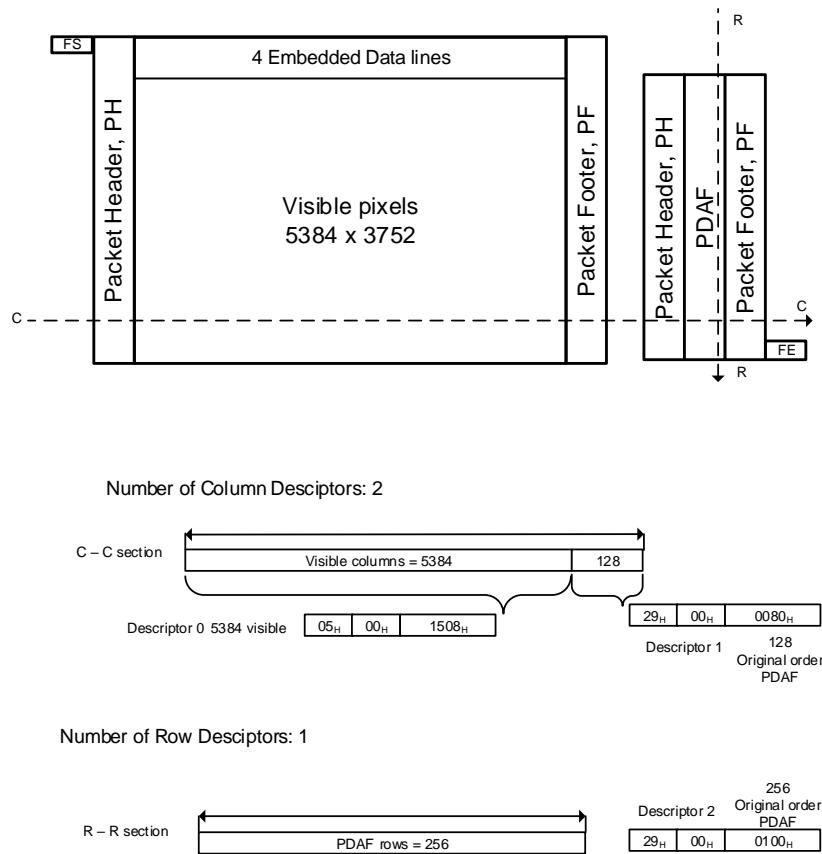
- 4813 1. Relevant PDAF Pixel Codes shall be used for Column Descriptors, and “Separated PDAF pixels”  
4814 shall be used for Row Descriptors.
- 4815 2. There shall be as many Column Descriptor Groups as there are different PDAF data rows.
- 4816 3. Different PDAF types in the same row shall be readout in the following sequence: pixels of the  
4817 first type are readout first, followed by pixels of second type, and so forth. Each Pixel Type Group  
4818 shall be described by a Column Descriptor. A complete row is described by a group of Column  
4819 Descriptors.
- 4820 4. There shall be as many Row Descriptor Groups as there are Column Descriptor Groups.
- 4821 5. Each Row Descriptor shall report the height of the intersecting Column Descriptor Group.
- 4822 6. If multiple Row Descriptors are used, then an additional descriptor with Pixel code “Total” shall  
4823 also be used to describe the complete height (which consists of repeating Column Descriptor  
4824 Groups, in order, up to the complete height reported by Total Descriptor).

#### 4825 **FFD Rules for Separate Types to Separate Type Lines Order**

- 4826 1. Relevant PDAF Pixel Codes shall be used for Column Descriptors, and “Separated PDAF pixels”  
4827 shall be used for Row Descriptors.
- 4828 2. There shall be as many Column Descriptor Groups as there are different PDAF data rows.
- 4829 3. Each Column Descriptor Group shall include only one PDAF Column Descriptor.
- 4830 4. There shall be as many Row Descriptor Groups as there are Column Descriptor Groups.
- 4831 5. Each Row Descriptor shall report the height of the intersecting Column Descriptor Group.
- 4832 6. Pixel code “Total” shall be used to describe the complete height, which consists of repeating  
4833 Column Descriptor Groups, in order, up to the complete height reported by Total Descriptor.

4834 Two examples are provided: Simple FFD with PDAF, and Advanced FFD with PDAF and OB pixel readout  
 4835 (i.e., the same examples as shown in **Section B.2.11**, but extended with PDAF data).

4836 **Figure 103** and **Table 288** show a simple example of PDAF readout: PDAF data is readout after each relevant  
 4837 line, using separate data type. This example assumes the PDAF pixel arrangement shown in **Figure 100**, and  
 4838 that Left PDAF pixels and Right PDAF pixels are readout in separate lines based on physical location order.



**Figure 103 Simple PDAF Readout Example**

4840

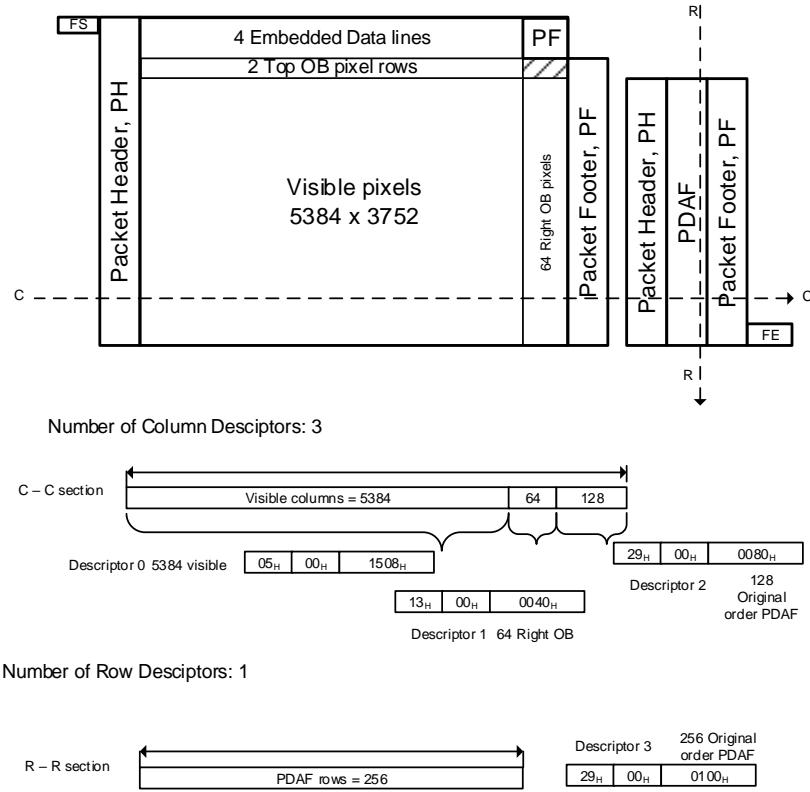
4841

4842

**Table 288 Simple PDAF Readout Block Example**

Field Name	Content	Data Type	Value	Comment
Length Specifier		Length specifier	0x10	Length of Data in bytes
Type		8-bit unsigned integer	5	Record Type 5: PDAF Readout
Data	PDAF_readout_info	16-bit unsigned integer	0x0001	1: Original Readout Order
	frame_format_model_subtype	16-bit unsigned integer	0x02	Number of Column Descriptors
			0x01	Number of Row Descriptors
	FFD_0 (First Column FFD)	32-bit unsigned integer	0x05	PixelCode: Visible
			0x00	Reserved
			0x15	Size: 5384
			0x08	
	FFD_1 (Second Column FFD)	32-bit unsigned integer	0x29	PixelCode: Original Order PDAF
			0x00	Reserved
			0x00	Size: 128
			0x80	
	FFD_2 (Row FFD)	32-bit unsigned integer	0x29	PixelCode: Original Order PDAF
			0x00	Reserved
			0x01	Size: 256
			0x00	

4843  
4844  
4845  
4846 **Figure 104** and **Table 289** shows an advanced example of PDAF readout. In this example, PDAF data is readout after each relevant line, as the last item on a line, using separate data type. This example assumes the PDAF pixel arrangement shown in **Figure 100**, and that Left PDAF pixels and Right PDAF pixels are readout in separate lines based on physical location order.



4847  
4848 **Figure 104 Advanced PDAF Readout Example**

4849

**Table 289 Advanced PDAF Readout Example**

Field Name	Content	Data Type	Value	Comment
Record 1	Length Specifier	Length specifier	0x04	Length of Data in bytes
	Type	8-bit unsigned integer	1	Record Type 1: If Rule
	Data (One If-Rule)	index if_value bitmask	0x0B30 1 Bit string 0000 0001	Index of CCS Register <b>OB_readout_ctrl</b> If_value for this rule If bitwise AND of register <b>OB_readout_ctrl</b> 's value with this bitmask equals the if_value, then this rule will be true
	Length Specifier	Length specifier	0x14	Length of Data in bytes
Record 2	Type	8-bit unsigned integer	5	Record Type 3: PDAF Readout
	Data	PDAF_readout_info	0x0001	1: Original Readout Order
		frame_format_model_subtype	0x03	Number of Column Descriptors
			0x01	Number of Row Descriptors
	FFD_0 (First Column FFD)	32-bit unsigned integer	0x05	PixelCode: Visible
			0x00	Reserved
			0x15	Size: 5384
			0x08	
	FFD_1 (Second Column FFD)	32-bit unsigned integer	0x13	PixelCode: Right Optical Black
			0x00	Reserved
			0x00	Size: 64
			0x40	
	FFD_2 (Third Column FFD)	32-bit unsigned integer	0x29	PixelCode: Original order PDAF
			0x00	Reserved
			0x00	Size: 128
			0x80	
	FFD_3 (Row FFD)	32-bit unsigned integer	0x29	PixelCode: Original order PDAF
			0x00	Reserved
			0x01	Size: 256
			0x00	

4850  
4851  
4852  
4853  
4854  
4855

An image sensor may reduce required line blanking during OB pixel DT/VC readout and PDAF DT/VC readout by sending OB pixel data in a different line than the PDAF data. For example, if the PDAF line length is 256 pixels, but the OB pixel length is shorter (e.g., 128 pixels), then it is possible to reduce the required minimum line blanking by sending two OB pixel lines (e.g., 128 + 128 pixels) during one line, instead of increasing the needed line blanking to 256 + 128 pixels. In such a scenario the PDAF Readout Record does not include OB pixels, even if the OB pixel readout is enabled and included into the PDAF Readout Record.

4856    **B.2.14    Complete FFD Usage**

4857    **B.2.14.1    Introduction**

4858    In this Section, different examples are shown to clarify how the FFD Record and the PDAF Readout Record  
4859    are used to describe readout details. In these examples, image sensors supporting the interleaved PDAF  
4860    readout option and image sensors supporting the bottom-embedded data PDAF readout option are shown,  
4861    both with and without the capability to readout OB pixels. Bottom-embedded data may be used to carry non-  
PDAF data, either alone or in addition to PDAF data, as described in *Section B.2.14.6*.

4862    **B.2.14.2    Image Sensor with Interleaved PDAF Readout Option**

4863    PDAF Readout Records define how PDAF pixels are readout. In a PDAF Readout Record, it is assumed that  
4864    PDAF pixels are readout when the CCS PDAF readout control register is enabled, even if the PDAF Readout  
Record does not describe an If-rule for CCS PDAF readout control (see *Table 288* for a detailed example).  
4865    FFD Records are used to describe basic FFD, assuming that PDAF readout is turned off (which is the default  
4866    case). See *Table 278* for a detailed example.

### B.2.14.3 Image Sensor with OB Pixel and with Interleaved PDAF Readout Option

4867 Five Records are provided:

- 4868 • One PDAF Readout Record defining how PDAF pixels are read out, e.g. as with Record 1 in Table  
4869 288
- 4870 • Basic FFD Record, assuming that OB pixel readout is turned off (which is the default case), e.g. as  
4871 with Record 1 in Table 278
- 4872 • A Condition Record depending on having the value 1 in register **OB\_readout\_ctrl** (i.e., on having  
4873 optical black pixel readout enabled), e.g. as with Record 1 in Table 289. This condition is applied  
4874 to the Records that follow it.
- 4875 • One PDAF Readout Record defining how PDAF pixels are read out when OB pixel readout is  
4876 enabled, e.g. as with Record 2 in Table 289.
- 4877 • One FFD Record when OB pixel readout is enabled, e.g. as with Record 2 in Table 280.

4878 **Table 290** summarizes the relevant examples for PDAF Readout and FFD Records in a Generic Rule Based  
4879 Block.

4880 **Table 290 Summary of the FFD and PDAF Readout Records**

Record	Field Name	Comment
<b>Record 1</b>	<b>Length Specifier</b>	As in <b>Table 288</b> (Record 1)
	<b>Type</b>	
	<b>Data</b>	
<b>Record 2</b>	<b>Length Specifier</b>	As in <b>Table 278</b> (Record 1)
	<b>Type</b>	
	<b>Data</b>	
<b>Record 3</b>	<b>Length Specifier</b>	If-Rule: <b>OB_readout_ctrl</b> bit 0 = 1 As in <b>Table 289</b> (Record 1)
	<b>Type</b>	
	<b>Data</b>	
<b>Record 4</b>	<b>Length Specifier</b>	FFDs as in <b>Table 289</b> (Record 2)
	<b>Type</b>	
	<b>Data</b>	
<b>Record 5</b>	<b>Length Specifier</b>	FFDs as in <b>Table 280</b> (Record 2)
	<b>Type</b>	
	<b>Data</b>	

#### B.2.14.4 Image Sensor with Bottom Embedded Data PDAF Readout Option

4881 Two FFD Records are provided:

- 4882 • Basic FFD Record assuming that PDAF readout is turned off (which is the default case), e.g. as  
 4883 with Record 1 in ***Table 278***
- 4884 • The case when PDAF readout is enabled (by CCS control register, defined by If-rule), e.g. as with  
 4885 Record 1 and Record 2 in ***Table 279***.

4886 ***Table 290*** summarizes the relevant examples for FFD Records in a Generic Rule Based Block.

4887 **Table 291 Summary of the FFD Records**

Record	Field Name	Comment
Record 1	Length Specifier	As in <b><i>Table 278</i></b> (Record 1)
	Type	
	Data	
Record 2	Length Specifier	If-Rule: PDAF_ctrl bit 0 =1. As in <b><i>Table 279</i></b> (Record 1)
	Type	
	Data	
Record 3	Length Specifier	FFDs as in <b><i>Table 279</i></b> (Record 2)
	Type	
	Data	

### B.2.14.5 Image Sensor with OB Pixels with Bottom Embedded Data PDAF Readout Option

Four FFD Records are provided:

- Basic FFD Record assuming that PDAF and OB pixel readout is turned off (which is the default case), e.g. as with Record 1 in *Table 278*
- The case when PDAF readout is enabled (by CCS control register, defined by If-rule), but assuming that OB pixel readout is disabled
- The case when OB pixel readout is enabled (by CCS control register, defined by If-rule), but assuming that PDAF readout is disabled
- The case when PDAF and OB pixel readout is enabled (by CCS control register, defined by If-rule), e.g. as with Record 1 and Record 2 in *Table 281*.

*Table 290* summarizes the relevant examples for FFD Records in a Generic Rule Based Block.

**Table 292 Summary of FFD Records**

Record	Field Name	Comment
Record 1	Length Specifier	As in <i>Table 278</i> (Record 1)
	Type	
	Data	
Record 2	Length Specifier	If-Rule: <b>PDAF_ctrl</b> bit 0 =1 and <b>OB_readout_ctrl</b> bit 0 = 0. Not in any table.
	Type	
	Data	
Record 3	Length Specifier	FFDs as in <i>Table 279</i> (Record 2)
	Type	
	Data	
Record 4	Length Specifier	If-Rule: <b>PDAF_ctrl</b> bit 0 =0 and <b>OB_readout_ctrl</b> bit 0 = 1 Not in any table.
	Type	
	Data	
Record 5	Length Specifier	FFDs as in <i>Table 280</i> (Record 2)
	Type	
	Data	
Record 6	Length Specifier	If -Rule: <b>PDAF_ctrl</b> bit 0 =1, and <b>OB_readout_ctrl</b> bit 0 = 1 As in <i>Table 281</i> (Record 1)
	Type	
	Data	
Record 7	Length Specifier	FFDs as in <i>Table 281</i> (Record 2)
	Type	
	Data	

#### B.2.14.6 Image Sensor with Bottom Embedded Data

An image sensor may also use bottom-embedded data, even if it would not support the bottom-embedded data PDAF readout option. Also, an image sensor may use bottom-embedded data to carry both PDAF data and non-PDAF data.

In such cases, the default FFD Record includes bottom-embedded data as shown in *Table 279* if the bottom-embedded data is non-controllable, or is on by default. Otherwise the default FFD Record does not include bottom-embedded data (e.g., as shown in *Table 278*), and the Generic Rule Based Block includes an additional If-Rule and FFD Records if bottom-embedded data is enabled. If the image sensor also supports OB pixel readout, then an additional FFD Record is described in the Generic Rule Based Block, e.g. similarly to *Figure 94*, but not requiring an If-Rule for the bottom-embedded PDAF data readout option as shown in *Table 281*.

An image sensor supporting bottom-embedded data may also support the interleaved PDAF readout option, instead of the bottom-embedded PDAF data readout option. In such a case, the FFD Record includes bottom-embedded data as shown in *Table 279*. The PDAF Readout Record would also include bottom-embedded data.

#### B.2.15 License Block

This block is mandatory if CCS Static Data is delivered as a file, see *Section B.4.1*. The content of this block is the license under which the entire file is made available. The content of the block shall be one of the approved licenses in the “licenses” directory in the CCS Static Data repository, using the Unicode character set in UTF-8 encoding.

#### B.2.16 End of Data Block

This block is mandatory if CCS Static Data is supported. The block is used to indicate the end of data, and includes a checksum. The checksum shall be calculated as specified in *Section B.5.2*. This block shall be the last block in the CCS Static Data.

Table 293 End of Data Block

Byte	Index	Content
0	Hi	Checksum
1	–	
2	–	
3	Lo	

### 4923 B.3 CCS Static Data Sources

4924 This Section defines the supported CCS Static Data sources. Currently sensor NVM, external EEPROM, and  
4925 file sources are supported.

#### 4926 B.3.1.1 Sensor NVM, Module EEPROM, and External EEPROM

4927 Sensor NVM and an external EEPROM which may or may not be a part of the camera module are supported  
4928 sources of CCS Static Data. It is specific to the platform firmware interface how to convey to the Operating  
4929 System the NVM or the EEPROM, and whether the entire content of the chip or a part of it conforms to the  
CCS Static Data format.

#### 4930 B.3.1.2 File on a File System

4931 One way to convey the CCS Static Data to software is through a file on a file system. The file system shall  
4932 be assumed to contain any number of CCS Static Data files. The identification of the correct file is based on  
4933 image sensor and camera module identification; therefore the identification registers (see *Section 6*) may not  
be placed in CCS Static Data if its source is a file.

4934 The static data files may be specific to either the image sensor or the camera module. Both may be used for  
4935 a camera module at the same time. In this case, the camera-module-specific file will override information in  
4936 the image-sensor-specific file. See additional information in *Section B.3.3*. The image-sensor-specific file  
4937 shall not include information outside the scope of the image sensor itself.

4938 For example, the image sensor might support 4 CSI-2 Data Lanes, but the camera module might only support  
4939 2 CSI-2 Data Lanes. The camera-module-specific CCS file information has higher priority, and thus the Host  
4940 can deduce from the camera module information that the module only supports 2 CSI-2 Data Lanes, even  
4941 when the image sensor CCS Static Data file is reporting 4 Data Lanes.

4942 The camera-module-specific file is not required to operate the image sensor.

4943 The file names shall follow the following formats:

- 4944 • For image sensor: **ccs-sensor-svid-sid-sver.fw**
- 4945 • For camera module: **ccs-module-mvid-mid-mver.fw**

4946

**Table 294 Naming Abbreviations in File Name**

Name	Comment
<b>svid</b>	Sensor vendor ID
<b>sid</b>	Sensor device ID
<b>sver</b>	Sensor revision 16 bits
<b>mvid</b>	Module vendor ID
<b>mid</b>	Module device ID
<b>mver</b>	Module revision major and minor (top 8 bits and bottom 8 bits, respectively)

4947 The version numbers are in lower case hexadecimal notation without any additional common postfixes or  
4948 prefixes (such as 0x or h).

4949 The CCS device driver shall attempt to load the image-sensor-specific and camera-module-specific data files  
4950 using the exact sensor and module version. If such a file is found, then the driver shall use it.

### B.3.2 Sensor, Module, and Data Source Specific Information

Individual items of information in CCS data sources may be specific to either the image sensor or the camera module, as well as to the data source itself. Most data block types not describing the data source itself have an ID for image-sensor-specific and camera-module-specific blocks. The distinction is that the image-sensor-specific block may describe the image sensor and the image sensor only. It is not allowed to describe any properties of the camera module in the sensor-specific blocks. The camera-module-specific block should contain information which is specific to the camera module, and only to the camera module. An example of the image-sensor- and camera-module-specific blocks is MSR Blocks, which are specific to an image sensor or to the camera module.

### B.3.3 CCS Static Data Block Priorities

The data sources are prioritized according to the version block's date information. If two sources have the same date, then the version number is considered. The newer the date and the higher the version number, the higher the priority. If a data source has no version block, then it is considered to have the lowest priority.

For the data sources that share the exactly the same version block content, these data sources have the same priority. Multiple data sources are applicable in that case, and they are used in the order in which they are declared in the platform firmware.

Each block from each data source is separately subject to selection, based on the prioritization granularity and override policy below.

### B.3.4 Prioritization Granularity

A data source may contain multiple blocks, and some blocks in turn consist of itemized data. Depending on the block, data from multiple sources may be combined together if the block contains itemized data. If the granularity of the block is an item (and not the entire block), then the data source priority is considered for each such item separately. In that case, information from data sources of lesser priority may be applicable as well.

### B.3.5 Override Policy

Override policy defines how different blocks and data sources may be overridden, as defined by *Table 269*. Override policy may be “None”, in which case the block is not overridden by any other block, or another block, in which case the block is entirely or partially (as defined by prioritization granularity) overridden by another block if that block is available.

### B.3.6 Sensor and Module Identification

Sensor and module identification information may well be available through the sensor's NVM interface or the EEPROM in the read-only register block. In that case the information will override the information specified in sensor's registers, just as it would override any other read-only register value. Further on, this information is still used to choose the image-sensor- and camera-module-specific data files that are subject to processing according to the above rules.

### B.3.7 Terms

**CCS data source:** Data source containing information conforming to the CCS Static Data format (e.g., a file, EEPROM, or NVM).

**data block:** A block in CCS Static Data (file).

**data item:** An indivisible unit of data that is a part of a CCS Static Data block.

### B.3.8 Example 1

In this example there are three EEPROM (or NVM) data sources S1, S2, and S3, declared in this order in platform firmware, all of which contain an image-sensor-specific read-only register block. S1 has the highest priority, S2 the next highest, and S3 the lowest.

The image-sensor-specific read-only register block of S3 contains registers that are not present in the corresponding blocks in the higher-priority sources S2 and S1. These S3 registers will be applicable. Similarly the registers that are present in the image-sensor-specific read-only register block of S2, but which are not present in the image-sensor-specific read-only register block of S1, will prevail. Naturally all registers from the image-sensor-specific read-only register block will be applicable.

The image-sensor-specific read-only registers from all three sources will be further subject to being overridden by camera-module-specific read-only register blocks from the same data sources.

### B.3.9 Example 2

In this example, both image sensor and camera module data files are available. A sensor's identification information is placed on registers, while the camera module identification is placed in EEPROM.

To identify the image sensor and the camera module, the CCS device driver shall first read the image sensor and camera module identification registers. Then it shall proceed to read the EEPROM for the CCS Static Data, either using the sensor data transfer interface (see *Section 13*) or from an external EEPROM chip, as conveyed to the CCS device driver from the platform firmware. The module identification is contained in the EEPROM content (`module_model_id`, `module_revision_number_major`, `module_revision_number_minor`, `module_manufacturer_ID`, `module_date_year`, `module_date_month`, `module_date_day`, `module_date_phase`, and `module_serial_number`). The image sensor and camera module identification information is used to determine the respective CCS Static Data files for both the camera module and the image sensor, and the files are loaded by the driver software.

The version block in the image-sensor- and camera-module-specific files are then compared to the version block in the CCS Static Data from the EEPROM. If the CCS Static Data in EEPROM has no version block, or if the version block in EEPROM has an older date than the camera-module-specific file, then the information in the camera-module-specific file will be used and the camera-module-specific blocks in the EEPROM will be ignored. Similarly, the CCS camera module data file will be ignored if the CCS Static Data version block in the data file is older than the one in EEPROM. The same applies to image-sensor-specific blocks in the EEPROM vs. the image-sensor-specific CCS Static Data file.

## 5015 B.4 File Delivery

5016 From a system perspective, the CCS Static Data is considered device firmware when its source is a file (vs.  
5017 NVM or EEPROM). This means that CCS Static Data files must be handled as device firmware files are. The  
5018 practices are somewhat specific to different Operating Systems, and the most common existing practices of  
5019 handling device firmware shall be supported. This leads to logistical requirements. The question is: how to  
5020 ensure that the CCS Static Data files will be available where they are needed without too much hassle?

5021 In the case of general-purpose Operating Systems the Operating System itself, possibly including drivers for  
5022 the hardware devices, are not tailored for the exact computer the Operating System is running on. Depending  
5023 on the Operating System, the device firmware, including CCS Static Data files, are part of either the  
5024 Operating System or the drivers additionally installed on it.

5025 MIPI cannot handle delivery of the CCS Static Data files to the general purpose Operating Systems, but it  
5026 could facilitate the process by placing the CCS Static Data files in a common repository where Operating  
5027 System vendors, driver developers, or system integrators can find them. The nature of the data is such that it  
5028 will require updates from time to time for various reasons, such as fixing bugs or refining image sensor tuning  
5029 values. Therefore, for correct and optimal hardware operation it is vital that up-to-date information is found  
5030 in this database.

5031 Image sensor and camera module vendors should ensure that the CCS Static Data files find their way to this  
5032 common repository.

5033 The details of the process for doing so shall be documented in a file called “PROCESS” in the root of the  
5034 CCS Static Data repository, see [\[MIPI13\]](#).

### 5035 B.4.1 Licensing

5036 In order to facilitate third parties’ (such as Operating System vendors) use of the CCS Static Data files, it is  
5037 required that **the license of the CCS Static Data files allows redistributing the CCS Static Data files**. The  
5038 exact license under which a CCS Static Data files are distributed shall be part of the CCS Static Data file  
5039 itself, as specified in the License Block (see [Section B.2.15](#)).

5040 The License Block is mandatory for all CCS Static Data files. The contents of the License Block shall  
5041 correspond to an approved license in the CCS Static Data file database. Approved licenses can be found in  
5042 the “licenses” directory in the CCS Static Data repository. In order to approve a new license, submit it for  
5043 review in order to merge it to the CCS Static Data repository.

5044 The License Block is optional if the CCS Static Data is stored in NVM or EEPROM.

5045 The CCS Static Data repository shall be managed as a MIPI Open Source project.

### 5046 B.4.2 License Block Validation

5047 Validating the License Block in a CCS Static Data file consists of implementing the following requirements:

- 5048 1. The CCS device driver looks up the license text corresponding to the License ID block contents  
5049 obtained from the CCS Static Data file. If no license text corresponding to such license ID block  
5050 content is found, then the result is a validation failure.
- 5051 2. The CCS device driver compares the License Block contents obtained from the CCS Static Data  
5052 file with the license. If the content matches, character by character, then the validation is  
5053 successful. Otherwise the result is a validation failure.

5054 The CCS device driver may only use a CCS Static Data file if the file passes the License Block validation  
5055 step.

5056 The License Block validation is only performed on the CCS Static Data if its source is a file. The validation  
5057 step is not performed to any other data sources.

5056 **B.5 Additional Information**

5057 **B.5.1 4-Byte Extended FFD**

5058 4-byte Extended Frame Format Descriptor (FFD) is the format for the FFDs used in CCS Static Data. This  
5059 format shall not be used as normal FFDs described in *Section 7.14*.

5060 The concept uses a generic part and FFDs. In the generic part the number of FFDs is described by the  
5061 extended Frame Format Subtype register.

5062 The **frame\_format\_model\_subtype** register specifies the number of Column Descriptors and Row  
5063 Descriptors:

- **Number of Column Descriptors:** Top 8 bits

5065 This specifies the number of Column Descriptors,  $D_{col}$

- **Number of Row Descriptors:** Bottom 8 bits

5067 This specifies the number of Row Descriptors,  $D_{row}$

5068 The format of the 4-Byte Extended Frame Format Descriptor is as follows:

- **Pixel Code:** The High byte

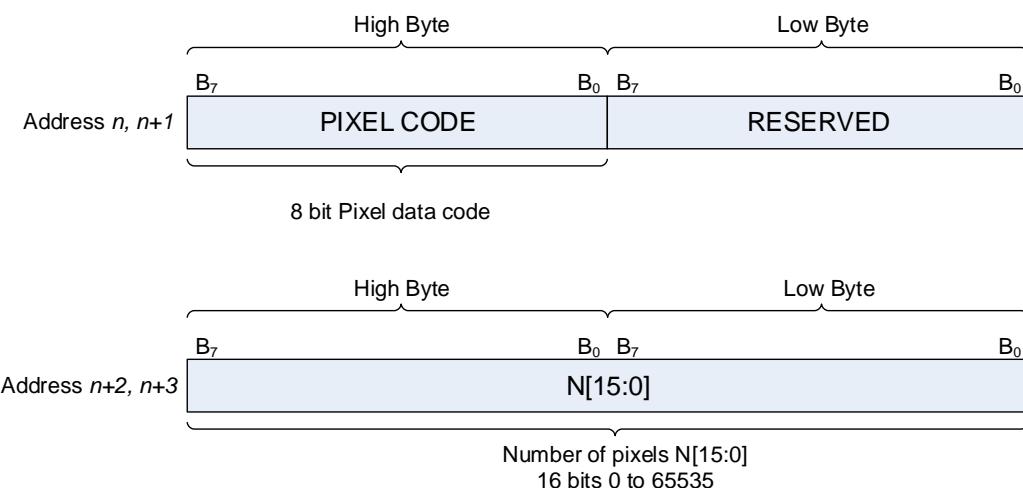
5070 This indicates the pixel data type (see *Table 295*)

- **Number of Pixels (16 bits):**

5072 This is the number of pixels of the indicated type.

5073 **Example:** A Pixel Code of 4 and a Number of Pixels of 1024 indicates 1024 Dark Pixels.

5074 The 4-byte descriptor is illustrated in *Figure 105*.



5075 **Figure 105 4-Byte Extended Frame Format Descriptor**

5076

**Table 295 Pixel Code Definitions**

<b>Pixel Code</b>	<b>Definition</b>
0	<i>Invalid</i>
1	Embedded data
2	Dummy Pixel data
3	Black Pixel data
4	(Legacy) Dark Pixel data
5	Visible Pixel data
6	<i>Reserved</i>
7	<i>Reserved</i>
8	Manufacturer-Specific Pixel type 0
9	Manufacturer-Specific Pixel type 1
10	Manufacturer-Specific Pixel type 2
11	Manufacturer-Specific Pixel type 3
12	Manufacturer-Specific Pixel type 4
13	Manufacturer-Specific Pixel type 5
14	Manufacturer-Specific Pixel type 6
15	<i>Invalid</i>
16	Top OB Pixel Data
17	Bottom OB Pixel Data
18	Left OB Pixel Data
19	Right OB Pixel Data
20	Top left OB Pixel Data (TL)
21	Top right OB Pixel Data (TR)
22	Bottom left OB Pixel Data (BL)
23	Bottom right OB Pixel Data (BR)
24	Total (used with PDAF block)
25-31	<i>Reserved</i>
32	Top PDAF pixel (Column Descriptor)
33	Bottom PDAF pixel (Column Descriptor)
34	Left PDAF pixel (Column Descriptor)
35	Right PDAF pixel (Column Descriptor)
36	Top left PDAF pixel (Column Descriptor)
37	Top right PDAF pixel (Column Descriptor)
38	Bottom left PDAF pixel (Column Descriptor)
39	Bottom right PDAF pixel (Column Descriptor)
40	Separated PDAF pixels (Row and Column Descriptor)
41	Original order PDAF (Row and Column Descriptor)
42	Vendor PDAF (Row and Column Descriptor)

5077

### B.5.1.1 Descriptor Rules

The following rules apply to 4-Byte Extended FFDs:

1. The Column Descriptors shall be specified first (X-direction – C-C section(s))
2. There shall be a block of  $D_{\text{col}}$  Column Descriptors followed by a block of  $D_{\text{row}}$  Row Descriptors. Column and Row Descriptors shall not be interleaved.
3. The Column Descriptors specify the left-to-right structure of a horizontal cross-section (C-C section(s) in the examples) through the image data frame.
4. The Row Descriptors specify the top-to-bottom structure of a vertical cross-section (R-R section in the examples) through the image data frame.
5. There shall be as many Column Descriptor Groups as Row Descriptors. A group shall have at least one Column Descriptor, and a group shall be used for each line that intersects with a row of a Row Descriptor.
  - This is a specific rule for FFD Records; PDAF Readout Records have their own rule.
6. The total number of columns defined by the Column Descriptors shall equal the number of pixels in each CSI-2 Long Packet, except that:
  - An embedded data line may be shorter (see **Section 7.15**)
  - This requirement is not relevant for interleaved readout modes of PDAF or OB pixels.
7. The total number of rows defined by the Row Descriptors shall equal the number of rows in the whole frame of image data.
  - See the PDAF Readout Record for an exception.
8. The Embedded Data Type is only valid for rows, and the Embedded Data Type is valid for the whole line, in each CSI-2 embedded data Long Packet.
9. 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), except that:
  - TL, TR, BL, and BR OB pixels do not intersect with visible pixels.
  - See the PDAF Readout Record for an exception
10. For the non-embedded data, the height of the region defined by a Column Descriptor is the region's intersection with the Visible pixel type region (i.e., the height of the visible region), except that:
  - TL, TR, BL, and BR OB pixels do not intersect with visible pixels.
  - See the PDAF Readout Record for an exception
11. Any region of image data that is not defined by the descriptors shall be treated as dummy pixel data.
  - See the PDAF Readout Record for an exception
12. The minimum number of descriptors is 4: Two for the X-direction, and two for the Y-direction.
  - See the PDAF Readout Record for an exception

5113      **B.5.2     Checksum Calculation**

5114    As defined in **Section B.2.16**, CCS Static Data may include a checksum. The checksum shall be calculated  
5115    as defined in this Section.

5116    Checksum algorithm is the CRC-32 algorithm [*ITU-T01*]. The checksum is calculated from first byte to the  
5117    end of data excluding the checksum.

5118    The CRC-32 has the following polynomial:

5119    
$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

5120    This Specification does not require any particular implementation of the CRC-32 algorithm. If the  
5121    implementer chooses to implement the algorithm in a linear feedback shift register, then the initial contents  
5122    of the shift register shall be initialized to all ones. In addition, the first bit of the CRC-32 calculation shall be  
5123    the most significant bit (bit 7) of the first byte in the CCS Static Data. Finally, the calculated bit sequence  
5124    shall be complemented (one's complement) to obtain the CRC-32 checksum.

- 5125    • The checksum is stored to the Checksum Block payload as specified in **Table 296**. The 32-bit  
5126    value generated by the CRC-32 calculator shall be stored such that bit 31 of the result shall  
5127    become the MSB of the first byte and bit 0 shall be the LSB of the last byte of the Checksum  
5128    Block.

5129      **Table 296 Checksum Block Content**

Byte	Index	Content
0	Hi	CRC-32 checksum
1	—	
2	—	
3	Lo	

This page intentionally left blank.

## 5130 Annex C Non-Bayer Support

5131 The CCS Specification can also be used with an image sensor that does not have a Bayer pattern, if certain  
5132 rules are followed.

5133 The image sensor shall support register **CFA\_pattern\_capability**, which indicates the CFA pattern.

**Table 297 CFA Pattern Capability Register**

Field Name	Type	RW	Comment
<b>CFA_pattern_capability</b>	8-bit unsigned integer	RO	0: Bayer 1: Monochrome 2: 4x4 Quad Bayer 3: vendor specific Other values: reserved for future use

5134 Specifically, the following CCS features must be considered carefully with non-Bayer image sensors:

- 5135 • The **Pixel\_order** register reports pixels in Bayer pattern. With a 2x2 non-Bayer pattern the nominal  
5136 pixel order shall use value 0, and the other values report the pixel order similarly as Bayer pattern  
5137 follows Mirror and Flip functionality, i.e. indicating the order of the first pixel and other pixels. If a  
5138 4x4 pattern is supported, then the **pixel\_order** register shall behave as with Bayer, i.e. indicating the  
5139 order of first pixel and other colored pixels. If monochrome (i.e., 1x1 pattern) is supported, then the  
5140 **pixel\_order** register value may or may not change when changing Mirror and Flip functionality.  
5141 The color of the first pixel reported by register **pixel\_order** with a non-Bayer image sensor may be  
5142 different than color with a Bayer image sensor.
- 5143 • Other features and requirements in the CCS specification can be understood easily by assuming that  
5144 Bayer format is a synonym for “2x2 pattern”. For example, size rules in **Section 7.3** are directly  
5145 applicable for Bayer and any other 2x2 pattern. If a 4x4 pattern is used in non-binned mode, then  
5146 the rules shall also be followed so that the pattern is not split.
- 5147 • Test patterns shall be generated assuming that the definitions are for 2x2 Bayer pattern.

5148 An image sensor with non-Bayer pattern may support a function to convert non-Bayer raw format to Bayer  
5149 format. Such a function can be controlled by register **CFA\_conversion** if so indicated by bit 0 of register  
5150 **CFA\_pattern\_conversion\_capability**.

**Table 298 CFA Conversion Registers**

Field Name	Type	RW	Comment
<b>CFA_pattern_conversion_capability</b>	8-bit unsigned integer	RO	<b>Bit 0</b> 1: Supports conversion to Bayer pattern <b>Other Bits</b> Reserved for future use
<b>CFA_conversion_ctrl</b>	8-bit unsigned integer	RW	<b>Bit 0</b> 1: Enable conversion to Bayer pattern <b>Other Bits</b> Reserved for future use

This page intentionally left blank.

## 5151 Annex D USL Support

5152 The CCS Specification also supports an alternative method for CCI communication, other than the normal  
5153 I<sup>2</sup>C/I3C based CCI. Such a link configuration is referred to as Unified Serial Link (USL). This Annex clarifies  
5154 USL's impact upon CCS features.

5155 The main differences are:

- 5156 • **Power-Up:** During power-up, the CCI bus cannot be used for image sensor configuration. This  
5157 requires different behavior during power-up.
- 5158 • **PHY Selection:** The default PHY must be correct, otherwise communication between an image  
5159 sensor and a Host is not possible.
- 5160 • **EXTCLK Frequency:** An image sensor cannot get the frequency info of EXTCLK until  
5161 communication between the image sensor and the Host has been established.
- 5162 • **Clock Tree:** The clock tree has additions to support clock generation for USL reverse mode.
- 5163 • **USL Control:** Controlling the USL link requires additional controls.
- 5164 • **Interoperability:** If an image sensor supports both USL and I<sup>2</sup>C/I3C based CCI, then the selection  
5165 should be done based on GPIO/NVM method in power-up. In addition, if a USL image sensor  
5166 supports both ALP Mode and LP/LVLP Mode signaling, then the selection of which mode to use  
5167 should also be done based on GPIO/NVM method in power-up.

### 5168 D.1.1 Power-Up and Initialization of USL Image Sensor

5169 USL uses the CSI-2 link in forward (USL\_FWD) mode and reverse (USL\_REV) mode. USL\_REV Mode is  
5170 used by data Lane 1 only. The PHY-level signaling used for reverse mode USL packet transmissions depends  
5171 upon whether the image sensor is configured in ALP Mode or LP/LVLP Mode at power-up; high-speed  
5172 signaling is always used for the former, whereas LPDT is used for the latter. D-PHY does not require a  
5173 running clock Lane during LPDT-based communications.

5174 In power-up, an image sensor must operate the link in USL\_REV Mode with default settings even if the  
5175 image sensor cannot know the EXTCLK frequency. An image sensor may not start the PLL automatically,  
5176 meaning that the image sensor requires slow USL\_REV Mode operation until the clock tree is configured.  
5177 Alternatively, an image sensor may start the PLL automatically, meaning that the image sensor may be able  
5178 to support a higher speed USL\_REV Mode. However, such an image sensor may also place a limit upon the  
5179 used link speed before the clock tree is programmed by the Host. The EXTCLK frequency may have an  
5180 impact on the supported initial link speed. For this reason, the link should be started with a low speed in  
5181 C-PHY ALP mode or D-PHY ALP mode. The maximum allowed initial USL\_REV Mode speed and its  
5182 dependency upon the EXTCLK frequency should be described in the image sensor datasheet.

5183 During ALP Mode power up with D-PHY, the image sensor should start the D-PHY clock Lane no later than  
5184 10 ms after the power-up sequence, and should be able to receive commands immediately. With C-PHY, the  
5185 image sensor should be able to receive commands no later than 10 ms after the power-up. This 10 ms value  
5186 is calculated from releasing XSHUTDOWN tied to image sensor power rail or XSHUTDOWN released by  
5187 the Host. The image sensor shall silently ignore any commands sent to it from the Host before the image  
5188 sensor is ready to process them.

5189 During the initial communication with an image sensor, the Host may program the PLL and clock tree to  
5190 change the USL\_REV and/or USL\_FWD speed. The initial communication before changing the link speed  
5191 may include read and/or write operations, and the sensor shall support both USL\_REV and USL\_FWD modes  
5192 before the Host configures the clock tree.

5193 During ALP Mode power-up, the image sensor shall keep the D-PHY clock Lane running until otherwise  
 5194 commanded by the Host. In Software Standby Mode, if it is supported by the image sensor, the Host may use  
 5195 the non-continuous D-PHY clock Lane feature. When that feature is supported, a USL image sensor has the  
 5196 capability to stop/restart the D-PHY clock Lane as specified in register **USL\_Clock\_Mode\_D\_capability**. D-  
 5197 PHY USL clock mode shall be selected by register **USL\_Clock\_Mode\_D\_ctrl**. If non-continuous D-PHY  
 5198 clock Lane mode is selected, then the Host may use register **TX\_USL\_ALP\_CTRL** to trigger stopping the  
 5199 clock [*MIPII2*].

5200

**Table 299 USL ALP Registers**

Field Name	Type	RW	Comment
<b>USL_Clock_Mode_D_capability</b>	8-bit unsigned integer	RO	<p><b>Bit 0</b>            1: Continuous clock in SW Standby Mode supported            0: Not supported</p> <p><b>Bit 1</b>            1: Continuous clock during Vertical blanking supported            0: Not supported</p> <p><b>Bit 2</b>            1: Continuous clock during horizontal blanking supported            0: Not supported</p> <p><b>Bit 3</b>            1: Non-continuous clock in SW Standby Mode supported            0: Not supported</p> <p><b>Bit 4</b>            1: Non-continuous clock during vertical blanking supported            0: Not supported</p> <p><b>Bit 5</b>            1: Non-continuous clock during horizontal blanking supported            0: Not supported</p> <p><b>Other Bits</b>            Reserved for future use</p>
<b>USL_Clock_Mode_D_ctrl</b>	8-bit unsigned integer	RW	<p><b>Bit 0</b>            0: Continuous clock in SW Standby Mode            1: Non-continuous clock in SW Standby Mode</p> <p><b>Bit 1</b>            0: Continuous clock during vertical blanking            1: Non-continuous clock during vertical blanking</p> <p><b>Bit 2</b>            0: Continuous clock during horizontal blanking            1: Non-continuous clock during horizontal blanking</p> <p><b>Other Bits</b>            Reserved for future use</p> <p><b>Default:</b> 0 if supported</p>

Field Name	Type	RW	Comment
TX_USL_ALP_CTRL	16-bit unsigned integer	RW	<p>Control register, introduced in <a href="#">[MIPI12]</a>.</p> <p><b>Bit 0</b></p> <p>1: Shall trigger image sensor to pause the D-PHY clock Lane during ALP mode. Image sensor shall auto-clear the register after stopping the clock <a href="#">[MIPI12]</a>.</p> <p><b>Other Bits</b></p> <p>Reserved for future use</p>

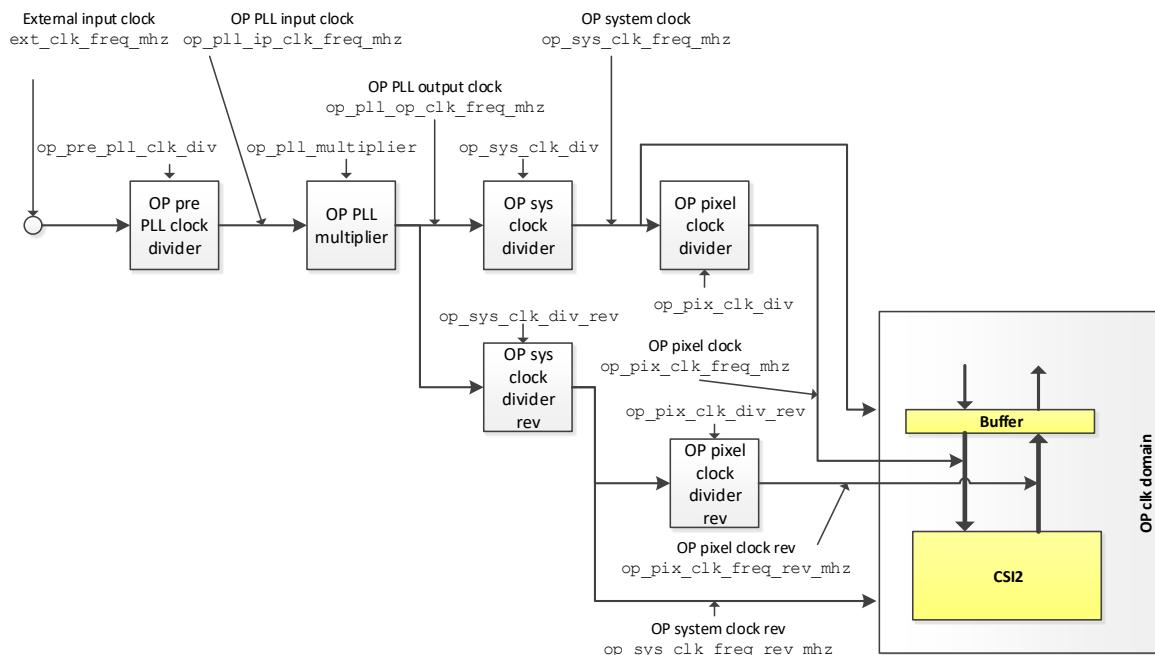
## D.2 Clock Tree of USL Image Sensor

5201  
 5202  
 5203 For ALP Mode operation, USL\_REV Mode requires additions to the standard clock tree. An image sensor supports the following definitions for the clocking if so indicated by Bit 0 of register **USL\_support\_capability**.

**Table 300 USL Support Capability Register**

Field Name	Type	RW	Comment
<b>USL_support_capability</b>	8-bit unsigned integer	RO	<p><b>Bit 0:</b>            1: CCS USL clock definition supported            0: Not supported</p> <p><b>Bit 1:</b>            1: CCS USL_REV mode clock tree supported            0: Not supported</p> <p><b>Bit 2:</b>            1: CCS USL_REV mode clock calculation supported            0: Not supported</p> <p><b>Other Bits</b>            Reserved for future use</p>

5204 It is assumed that a USL sensor typically uses two PLLs, one for VT-domain and one for OP-domain, but  
 5205 that is not a necessity. An image sensor supports **op\_sys\_clk\_div\_rev** and **op\_pix\_clk\_div\_rev** dividers  
 5206 registers for the reverse direction if so indicated by Bit 1 of register **USL\_support\_capability**. **Figure 106**  
 5207 shows details of the OP clock tree branch with USL support for an image sensor with two PLLs, one for VT  
 5208 domain (not shown) and one for OP domain (shown).



**Figure 106 Example of USL Clock Tree**

5210

**Table 301 USL\_REV Clock Divider Registers**

Register Name	Type	RW	Comment
<b>op_pix_clk_div_rev</b>	16-bit unsigned integer	RW	The valid <b>op_pix_clk_div_rev</b> values range from <b>min_op_pix_clk_div_rev</b> to <b>max_op_pix_clk_div_rev</b> (inclusive). <b>Default:</b> Image-sensor-specific
<b>op_sys_clk_div_rev</b>	16-bit unsigned integer	RW	Only even <b>op_sys_clk_div_rev</b> values are supported (mandatory), with the exception of <b>op_sys_clk_div_rev</b> = 1. The minimum and maximum supported values (inclusive) are described by <b>min_op_sys_clk_div_rev</b> to <b>max_op_sys_clk_div_rev</b> . <b>Examples:</b> 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 <b>Default:</b> Image-sensor-specific

5211 The Host shall program the USL\_REV Mode speed to be within the image sensor's limits. The image sensor  
 5212 limits are defined by registers in **Table 302**. Note that some registers have either 32-bit IEEE float or 32-bit  
 5213 unsigned iReal as a Type, depending upon capability register **clock\_capa\_type\_capability**.

5214

**Table 302 USL Clock Set-Up Capability Registers**

Register Name	Type	RW	Comment
<b>min_op_sys_clk_div_rev</b>	16-bit unsigned integer	RO	Minimum output system clock divider value for USL_REV Mode
<b>max_op_sys_clk_div_rev</b>	16-bit unsigned integer	RO	Maximum output system clock divider value for USL_REV Mode
<b>min_op_sys_clk_freq_rev_mhz</b>	32-bit IEEE float or 32-bit unsigned iReal	RO	Minimum output system clock frequency for USL_REV Mode <b>Units:</b> MHz
<b>max_op_sys_clk_freq_rev_mhz</b>	32-bit IEEE float or 32-bit unsigned iReal	RO	Maximum output system clock frequency for USL_REV Mode <b>Units:</b> MHz
<b>min_op_pix_clk_div_rev</b>	16-bit unsigned integer	RO	Minimum output pixel clock divider value for USL_REV Mode
<b>max_op_pix_clk_div_rev</b>	16-bit unsigned integer	RO	Maximum output pixel clock divider value for USL_REV Mode
<b>min_op_pix_clk_freq_rev_mhz</b>	32-bit IEEE float or 32-bit unsigned iReal	RO	Minimum output pixel clock frequency for USL_REV Mode <b>Units:</b> MHz
<b>max_op_pix_clk_freq_rev_mhz</b>	32-bit IEEE float or 32-bit unsigned iReal	RO	Maximum output pixel clock frequency for USL_REV Mode <b>Units:</b> MHz
<b>max_bitrate_rev_d_mode_mbps</b>	32-bit unsigned iReal	RO	Maximum bitrate for USL_REV Mode in D-PHY mode <b>Units:</b> Mbps
<b>max_symbolrate_rev_c_mode_msps</b>	32-bit unsigned iReal	RO	Maximum Symbol rate for USL_REV Mode in C-PHY mode <b>Units:</b> Msps

An image sensor supports CCS clock calculations for the reverse direction if so indicated by Bit 2 of register **USL\_support\_capability**. This clock calculation with D-PHY and with C-PHY is defined as follows:

- In reverse mode, there is no strict definition of the correct speed for **op\_pix\_clk\_freq\_rev\_mhz**. The default value of **op\_pix\_clk\_div\_rev** should be valid for all transmissions in USL\_REV Mode.
- With D-PHY, the Host should program register **op\_sys\_clk\_div\_rev** to match the link speed, assuming Single Data Rate. For example, for USL\_REV Mode running at 100 Mbit/s, **op\_sys\_clk\_freq\_rev\_mhz** is programmed to be 100 MHz.
- With C-PHY, the Host should program register **op\_sys\_clk\_div\_rev** to either match the link speed, or run the internal speed higher than the link, assuming Single Data Rate. For example, for USL\_REV Mode running at 100 Mbit/s (and therefore a Symbol rate of 100 Mbit/s / 16\*7 = 43.75 Msym/s), **op\_sys\_clk\_freq\_rev\_mhz** is programmed to be 43.75 MHz or higher.
- With C-PHY, if an image sensor automatically generates needed internal clocking from the C-PHY embedded clock, then the image sensor may omit implementing registers **op\_sys\_clk\_div\_rev** and **op\_pix\_clk\_div\_rev** and related limit registers defined in *Table 302*; however, register **max\_symbolrate\_rev\_c\_mode\_msps** shall be implemented.

With USL sensors, the clock tree and PLL settings impacting USL\_REV Mode speed shall be programmed by using **grouped\_parameter\_hold** functionality. The Host shall release **grouped\_parameter\_hold** after updating the clock tree and PLL settings, which shall trigger the image sensor to take new settings into use. This is done to ensure that only properly configured clock tree and PLL settings are used. The image sensor may require that further communication with it be delayed until configuration of its internal clock tree has finished. Such a delay should be less than 1 ms. The image sensor shall silently ignore any commands sent to it before it is ready to process them.

An image sensor supporting USL shall support the USL Mode switching registers listed in *Table 303 [MIPI12]*.

**Table 303 USL Switching Registers**

Register Name	Type	RW	Comment
<b>TX_USL_REV_ENTRY</b>	16-bit unsigned integer	RW	Control register, introduced in <i>[MIPI12]</i> .
<b>TX_USL_REV_Clock_Counter</b>	16-bit unsigned integer	RW	Counter register, introduced in <i>[MIPI12]</i> .
<b>TX_USL_REV_LP_Counter</b>	16-bit unsigned integer	RW	Counter register, introduced in <i>[MIPI12]</i> .
<b>TX_USL_REV_Frame_Counter</b>	16-bit unsigned integer	RW	Counter register, introduced in <i>[MIPI12]</i> .
<b>TX_USL_REV_Chronological_Timer</b>	16-bit unsigned integer	RW	Counter register, introduced in <i>[MIPI12]</i> .
<b>TX_USL_FWD_ENTRY</b>	16-bit unsigned integer	RW	Control register, introduced in <i>[MIPI12]</i> .

5241 To control mode changes between Software Standby and Streaming, the following shall be taken into account.  
 5242 To start streaming, the Host will first configure USL-counters (if needed) and set register **mode\_select** to 1  
 5243 before commanding link turnaround via register **TX\_USL\_FWD\_ENTRY**. If **mode\_select** is not set to 1, then  
 5244 the image sensor shall not stream image data in USL\_FWD mode, and shall instead return only NACK/ACK  
 5245 and the requested read data information.

5246 An image sensor shall also support the registers shown in *Table 304 [MIPI12]*.

5247

**Table 304 Additional USL Registers**

Register Name	Type	RW	Comment
<b>TX_USL_GPIO</b>	16-bit unsigned integer	RW	Control register, introduced in <a href="#">[MIPI12]</a> .
<b>TX_USL_Operation</b>	16-bit unsigned integer	RW	Control register, introduced in <a href="#">[MIPI12]</a> . <b>Bit 0</b> 1: Same functionality as setting software_reset to 1
<b>TX_USL_SNS_BTA_ACK_TIMEOUT</b>	16-bit unsigned integer	RO / RO Dynamic	Status register, introduced in <a href="#">[MIPI12]</a> .
<b>TX_USL_APP_BTA_ACK_TIMEOUT</b>	16-bit unsigned integer	RW	Control register, introduced in <a href="#">[MIPI12]</a> .

### D.3 USL Image Sensor Receiver LRTE Capability and Control Registers

A USL image sensor may optionally support CSI-2 LRTE packet transmissions in the forward and/or reverse direction.

Forward direction LRTE capability and control registers for high-speed D-PHY and C-PHY transmissions are described in *Section 7.7.1*. USL image sensor reverse direction LRTE capability registers are shown in *Table 305* for both high-speed ALP Mode and LP/LVLP Mode (i.e., LPDT) transmissions.

Note that *[MIPI12]* requires all USL image sensors with D-PHY ALP Mode CSI-2 receivers supporting D-PHY LRTE Option 2 to also support reception of EoTp short packets.

Table 305 USL Image Sensor CSI-2 Receiver LRTE Capability Registers

Register Name	Type	RW	Comment
<b>LRTE_usl_cphy_hs_rx_capability</b>	16-bit unsigned integer	RO	<p>LRTE capability for image sensor ALP Mode C-PHY RX</p> <p><b>Bits 6-0</b> (applicable if Bit 15 = 1) Minimum number of Spacer words supported following a USL packet</p> <p><b>Bits 13-7</b> (applicable if Bit 15 = 1) Maximum number of Spacer words supported following a USL packet</p> <p><b>Bit 14</b> (applicable if Bit 15 = 1) 1: Supports reception of Spacer words without PDQ signaling</p> <p><b>Bit 15</b> 1: Supports reception of high-speed USL packets using CSI-2 LRTE with ALP Mode C-PHY</p>
<b>LRTE_usl_cphy_lp_rx_capability</b>	16-bit unsigned integer	RO	<p>LRTE capability for image sensor C-PHY LPDT RX</p> <p><b>Bits 6-0</b> (applicable if Bit 15 = 1) Minimum fixed number of Spacer bytes supported between USL packets</p> <p><b>Bits 13-7</b> (applicable if Bit 15 = 1) Maximum fixed number of Spacer bytes supported between USL packets</p> <p><b>Bit 14:</b> Reserved</p> <p><b>Bit 15</b> 1: Supports reception of USL packets using CSI-2 LRTE for LPDT</p>

Register Name	Type	RW	Comment
<b>LRTE_usl_dphy_hs_rx_capability_1</b>	16-bit unsigned integer	RO	<p>LRTE capability for image sensor ALP Mode D-PHY RX with variable length Spacers</p> <p><b>Bits 6-0</b> (applicable if Bit 15 = 1) Minimum number of Spacer bytes supported between USL packets</p> <p><b>Bits 13-7</b> (applicable if Bit 15 = 1) Maximum number of Spacer bytes supported between USL packets</p> <p><b>Bit 14</b> (applicable if Bit 15 = 1) 0: Number of Spacer bytes is unconstrained 1: Number of Spacer bytes must be a multiple of four</p> <p><b>Bit 15</b> 1: Supports reception of USL packets using CSI-2 LRTE Option 2 for D-PHY ALP Mode with variable length Spacers</p>
<b>LRTE_usl_dphy_hs_rx_capability_2</b>	16-bit unsigned integer	RO	<p>LRTE capability for image sensor ALP Mode D-PHY RX with fixed length Spacers</p> <p><b>Bits 6-0</b> (applicable if Bit 15 = 1) Minimum fixed number of Spacer bytes supported between USL packets</p> <p><b>Bits 13-7</b> (applicable if Bit 15 = 1) Maximum fixed number of Spacer bytes supported between USL packets</p> <p><b>Bit 14:</b> Reserved</p> <p><b>Bit 15</b> 1: Supports reception of USL packets using CSI-2 LRTE Option 2 for D-PHY ALP Mode with fixed length Spacers</p>
<b>LRTE_usl_dphy_lp_rx_capability</b>	16-bit unsigned integer	RO	<p>LRTE capability for image sensor D-PHY LPDT RX</p> <p><b>Bits 6-0</b> (applicable if Bit 15 = 1) Minimum fixed number of Spacer bytes supported between USL packets</p> <p><b>Bits 13-7</b> (applicable if Bit 15 = 1) Maximum fixed number of Spacer bytes supported between USL packets</p> <p><b>Bit 14:</b> Reserved</p> <p><b>Bit 15</b> 1: Supports reception of USL packets using CSI-2 LRTE for LPDT</p>
<b>LRTE_usl_rx_max_burst_size</b>	16-bit unsigned integer	RO	The maximum total number of packet payload bytes the image sensor is capable of receiving in a single LRTE multi-packet high-speed or LPDT burst transmission

5256 The **SNS\_USL\_LPDT\_LRTE\_cphy** and/or **SNS\_USL\_LPDT\_LRTE\_dphy** control registers shown in **Table**  
 5257 **306** shall be supported by image sensors capable of transmitting USL packets using LPDT and LRTE. They  
 5258 are used to enable or disable LRTE for such packets and to set the number of Spacers between them (disabled  
 5259 by default).

5260 The **LRTE\_usl\_rx\_enable\_cphy** and/or **LRTE\_usl\_rx\_enable\_dphy** control registers shown in **Table 306**  
 5261 shall be supported by image sensors capable of receiving any type of USL packet transmissions using LRTE.  
 5262 They are used to enable or disable LRTE for USL packet receptions (disabled by default).

5263

**Table 306 USL LRTE Control Registers**

Register Name	Type	RW	Comment
<b>SNS_USL_LPDT_LRTE_cphy</b>	8-bit unsigned integer	RW	LRTE control for image sensor C-PHY LPDT TX <b>Bits 6-0</b> (applicable if Bit 7 = 1) The fixed number of Spacers inserted between USL packets transmitted using LPDT (0 to 127) <b>Bit 7</b> 1: Enable LRTE for USL packets transmitted using C-PHY LPDT <b>Default:</b> 0
<b>SNS_USL_LPDT_LRTE_dphy</b>	8-bit unsigned integer	RW	LRTE control for image sensor D-PHY LPDT TX <b>Bits 6-0</b> (applicable if Bit 7 = 1) The fixed number of Spacers inserted between USL packets transmitted using LPDT (0 to 127) <b>Bit 7</b> 1: Enable LRTE for USL packets transmitted using D-PHY LPDT <b>Default:</b> 0
<b>LRTE_usl_rx_enable_cphy</b>	8-bit unsigned integer	RW	LRTE control for image sensor C-PHY RX 0: Disable LRTE 1: Enable LRTE for ALP Mode 3: Enable LRTE for LPDT Other values: reserved <b>Default:</b> 0
<b>LRTE_usl_rx_enable_dphy</b>	8-bit unsigned integer	RW	LRTE control for image sensor D-PHY RX 0: Disable LRTE 1: Enable LRTE for ALP Mode with variable length Spacers 2: Enable LRTE for ALP Mode with fixed length Spacers 3: Enable LRTE for LPDT Other values: reserved <b>Default:</b> 0

This page intentionally left blank.

## Participants

The list below includes those persons who participated in the Working Group that developed this Specification and who consented to appear on this list.

Radha Krishna Atukula, NVIDIA	Takashi Miyamoto, Sony Group Corporation
Andy Baldman, Keysight Technologies Inc.	Alexander Mokhoria, Robert Bosch GmbH
Nadav Banet, Valens Semiconductor	Raj, Kumar Nagpal, Synopsys, Inc.
Craig Bezek, Teledyne LeCroy	Makoto Nariya, Sony Group Corporation
Thomas Blon, Silicon Line GmbH	Kavitha Naveen, Tektronix, Inc.
Alexander Brill, Intel Corporation	Yoshitomo Osawa, Sony Group Corporation
Goon Cha, Samsung Electronics, Co.	Jangwoo Park, LX Semicon Co., Ltd.
Steven Chang, MediaTek Inc.	Allan Paul, Cadence Design Systems, Inc.
Geraud Cheenne, STMicroelectronics	Ron Persky, Valens Semiconductor
Teong, Rong Chua, Sony Group Corporation	Radu Pitigoi-Aron, Qualcomm Incorporated
Vivet Corera, Arasan Chip Systems, Inc.	Kondalarao Polisetti, Advanced Micro Devices, Inc.
Stephen Creaney, Cadence Design Systems, Inc.	Alex Qiu, NVIDIA
Yoshihiko Deoka, Sony Group Corporation	Loren Reiss, Cadence Design Systems, Inc.
Sophie Fu, Advanced Micro Devices, Inc.	Matthew Ronning, Sony Group Corporation
James Goel, Qualcomm Incorporated	Shunmugam Samiapillai, Arasan Chip Systems, Inc.
Jason Gonzalez, Qualcomm Incorporated	Hugo Santos, Synopsys, Inc.
Philip Hawkes, Qualcomm Incorporated	Frank Seto, Samsung Electronics, Co.
Thomas Hogenmueller, Robert Bosch GmbH	Keren, Shmueli Sidi, Valens Semiconductor
Inho Jung, LX Semicon Co., Ltd.	Ariel Sobelman, Valens Semiconductor
Timo Kaikumaa, Intel Corporation	Greg Stewart, Analogix Semiconductor, Inc.
JaeHyuck Kang, Samsung Electronics, Co.	Dale Stoltzka, Samsung Electronics, Co.
Soichi Kayamori, Sony Group Corporation	Hiroo Takahashi, Sony Group Corporation
Samson Kim, Qualcomm Incorporated	Ashraf Takla, Mixel, Inc.
Tom Kopet, onsemi	Giuseppe Tofanicchio, STMicroelectronics
Marcin Kowalewski, Synopsys, Inc.	Krishna Veni, Arasan Chip Systems, Inc.
Wei, Cheng Ku, MediaTek Inc.	Frank Wang, OmniVision Technologies, Inc.
Daechul Kwon, Samsung Electronics, Co.	Guizhen Wang, HiSilicon Technologies Co. Ltd.
Ariel Lasry, Qualcomm Incorporated	Rick Wietfeldt, Qualcomm Incorporated
Ethan Lau, MediaTek Inc.	George Wiley, Qualcomm Incorporated
Wonseok Lee, Samsung Electronics, Co.	Stephen Wong, Intel Corporation
William Lo, Axonne Inc.	Charles Wu, OmniVision Technologies, Inc.
Ky MacPherson, Silicon Labs, Inc.	Kevin Yee, Samsung Electronics, Co.
Larry Madar, Google LLC	Michel Yeh, MediaTek Inc.
Cedric Marta, Synopsys, Inc.	Naoki Yokoshima, Sony Group Corporation
Nuno Martins, onsemi	Sang, Young Youn, Google LLC
Grzegorz Matczak, Synopsys, Inc.	Eunseung Yun, Samsung Electronics, Co.

**Past contributors to version 1.1:**

Sakari Ailus, Intel Corporation  
Rob Anhofer, MIPI Alliance (Team)  
Radha Krishna Atukula, NVIDIA  
Uwe Beutnagel-Buchner, Robert Bosch GmbH  
Gyeonghan Cha, Samsung Electronics, Co.  
Teong Rong Chua, Sony Corporation  
Zhang Chunrong, Advanced Micro Devices, Inc.  
Tomer Cohen, Samsung Electronics, Co.  
Al Czamara, Test Evolution  
Chris Grigg, MIPI Alliance (Team)  
Jason Hawken, Advanced Micro Devices, Inc.  
Guy Hutchison, Dryv.io  
Hyosun Joo, SK Hynix  
Kai Ruben Josefson, OmniVision Technologies, Inc.  
Tom Kopet, ON Semiconductor  
Saurabh Kumar, Synopsys, Inc.  
Mark Lewis, Cadence Design Systems, Inc.  
Larry Madar, Google, Inc.  
Kumaravel Manickam, L&T Technology Services  
Cedric Marta, Synopsys, Inc.  
Mikko Muukki, HiSilicon Technologies Co. Ltd.  
Josh Pan, MediaTek Inc.

Prachi Patel, Cadence Design Systems, Inc.  
Allan Paul, Cadence Design Systems, Inc.  
Radu Pitigoi-Aron, Qualcomm Incorporated  
Kondalarao Polisetti, Xilinx Inc.  
Quan Lin Qiu, NVIDIA  
Rahul R, L&T Technology Services  
Nitin Sarangdhar, Intel Corporation  
Yon Jun Shin, Samsung Electronics, Co.  
Scott Shuey, Luxshare-ICT, Inc.  
Richard Sproul, Cadence Design Systems, Inc.  
Vinoth Srinivasan, L&T Technology Services  
Hiroo Takahashi, Sony Corporation  
Haran Thanigasalam, Intel Corporation  
Giuseppe Tofanicchio, STMicroelectronics  
Giri Venkat, ON Semiconductor  
Ayshwarya Venkataramanan, Robert Bosch GmbH  
Rick Wietfeldt, Qualcomm Incorporated  
George Wiley, Qualcomm Incorporated  
David Woolf, University of New Hampshire  
InterOperability Lab (UNH-IOL)  
Charles Wu, OmniVision Technologies, Inc.

**Past contributors to version 1.0:**

Sakari Ailus, Intel Corporation  
Rob Anhofer, MIPI Alliance (Team)  
Radha Krishna Atukula, NVIDIA  
Chua Teong Rong, Sony Corporation  
Zhang Chunrong, Advanced Micro Devices, Inc.  
Tatsuyuki Fukushima, Teradyne Inc.  
Chris Grigg, MIPI Alliance (Team)  
Jason Hawken, Advanced Micro Devices, Inc.  
Yoshida Hirofumi, Sony Corporation  
Hsien-Chang Ho, MediaTek Inc.  
Yukichi Inoue, Teradyne Inc.  
Grant Jennings, Lattice Semiconductor Corp.  
Nadine Kolment, Introspect Test Technology Inc.  
Tom Kopet, ON Semiconductor  
Thomas Krause, Texas Instruments Incorporated  
Cedric Marta, Synopsys, Inc.  
Hideki Mitsubayashi, Sony Corporation

Mikko Muukki, HiSilicon Technologies Co. Ltd.  
Raj Kumar Nagpal, Synopsys, Inc.  
Manuel Ortiz, Intel Corporation  
Radu Pitigoi-Aron, Qualcomm Incorporated  
Alex Qiu, NVIDIA  
Matthew Ronning, Sony Corporation  
Yaron Schwartz, Cadence Design Systems, Inc.  
Sho Sengoku, Qualcomm Incorporated  
Yacov Simhony, Cadence Design Systems, Inc.  
Richard Sproul, Cadence Design Systems, Inc.  
Tatsuya Sugioka, Sony Corporation  
Hiroo Takahashi, Sony Corporation  
Haran Thanigasalam, Intel Corporation  
Rick Wietfeldt, Qualcomm Incorporated  
George Wiley, Qualcomm Incorporated  
Charles Wu, OmniVision Technologies, Inc.