



## **DRAFT** MIPI Alliance Specification for Display Serial Interface

**Draft Version 1.02.00 Revision 0.07 – 30 March 2010**

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

**1 NOTICE OF DISCLAIMER**

2 The material contained herein is not a license, either expressly or impliedly, to any IPR owned or controlled  
3 by any of the authors or developers of this material or MIPI®. The material contained herein is provided on  
4 an "AS IS" basis and to the maximum extent permitted by applicable law, this material is provided AS IS  
5 AND WITH ALL FAULTS, and the authors and developers of this material and MIPI hereby disclaim all  
6 other warranties and conditions, either express, implied or statutory, including, but not limited to, any (if  
7 any) implied warranties, duties or conditions of merchantability, of fitness for a particular purpose, of  
8 accuracy or completeness of responses, of results, of workmanlike effort, of lack of viruses, and of lack of  
9 negligence.

10 All materials contained herein are protected by copyright laws, and may not be reproduced, republished,  
11 distributed, transmitted, displayed, broadcast or otherwise exploited in any manner without the express  
12 prior written permission of MIPI Alliance. MIPI, MIPI Alliance and the dotted rainbow arch and all related  
13 trademarks, tradenames, and other intellectual property are the exclusive property of MIPI Alliance and  
14 cannot be used without its express prior written permission.

15 ALSO, THERE IS NO WARRANTY OF CONDITION OF TITLE, QUIET ENJOYMENT, QUIET  
16 POSSESSION, CORRESPONDENCE TO DESCRIPTION OR NON-INFRINGEMENT WITH REGARD  
17 TO THIS MATERIAL OR THE CONTENTS OF THIS DOCUMENT. IN NO EVENT WILL ANY  
18 AUTHOR OR DEVELOPER OF THIS MATERIAL OR THE CONTENTS OF THIS DOCUMENT OR  
19 MIPI BE LIABLE TO ANY OTHER PARTY FOR THE COST OF PROCURING SUBSTITUTE  
20 GOODS OR SERVICES, LOST PROFITS, LOSS OF USE, LOSS OF DATA, OR ANY INCIDENTAL,  
21 CONSEQUENTIAL, DIRECT, INDIRECT, OR SPECIAL DAMAGES WHETHER UNDER  
22 CONTRACT, TORT, WARRANTY, OR OTHERWISE, ARISING IN ANY WAY OUT OF THIS OR  
23 ANY OTHER AGREEMENT, SPECIFICATION OR DOCUMENT RELATING TO THIS MATERIAL,  
24 WHETHER OR NOT SUCH PARTY HAD ADVANCE NOTICE OF THE POSSIBILITY OF SUCH  
25 DAMAGES.

26 Without limiting the generality of this Disclaimer stated above, the user of the contents of this Document is  
27 further notified that MIPI: (a) does not evaluate, test or verify the accuracy, soundness or credibility of the  
28 contents of this Document; (b) does not monitor or enforce compliance with the contents of this Document;  
29 and (c) does not certify, test, or in any manner investigate products or services or any claims of compliance  
30 with the contents of this Document. The use or implementation of the contents of this Document may  
31 involve or require the use of intellectual property rights ("IPR") including (but not limited to) patents,  
32 patent applications, or copyrights owned by one or more parties, whether or not Members of MIPI. MIPI  
33 does not make any search or investigation for IPR, nor does MIPI require or request the disclosure of any  
34 IPR or claims of IPR as respects the contents of this Document or otherwise.

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

36 MIPI Alliance, Inc.  
37 c/o IEEE-ISTO  
38 445 Hoes Lane  
39 Piscataway, NJ 08854  
40 Attn: Board Secretary

41

## 42 Contents

43	Draft Version 1.02.00 Revision 0.07 – 30 March 2010.....	i
44	1 Overview .....	10
45	1.1 Scope .....	10
46	1.2 Purpose .....	10
47	2 Terminology (informative).....	11
48	2.1 Definitions .....	11
49	2.2 Abbreviations .....	12
50	2.3 Acronyms .....	12
51	3 References (informative).....	15
52	3.1 Display Bus Interface Standard for Parallel Signaling (DBI-2) .....	15
53	3.2 Display Pixel Interface Standard for Parallel Signaling (DPI-2).....	16
54	3.3 MIPI Alliance Specification for Display Command Set (DCS) .....	16
55	3.4 MIPI Alliance Standard for Camera Serial Interface 2 (CSI-2).....	16
56	3.5 MIPI Alliance Specification for D-PHY (D-PHY).....	16
57	4 DSI Introduction.....	17
58	4.1 DSI Layer Definitions .....	18
59	4.2 Command and Video Modes .....	19
60	4.2.1 Command Mode .....	19
61	4.2.2 Video Mode Operation .....	19
62	4.2.3 Virtual Channel Capability .....	20
63	5 DSI Physical Layer.....	21
64	5.1 Data Flow Control .....	21
65	5.2 Bidirectionality and Low Power Signaling Policy .....	21
66	5.3 Command Mode Interfaces .....	22
67	5.4 Video Mode Interfaces .....	22
68	5.5 Bidirectional Control Mechanism .....	22
69	5.6 Clock Management.....	23
70	5.6.1 Clock Requirements .....	23
71	5.6.2 Clock Power and Timing.....	24
72	5.7 System Power-Up and Initialization.....	24
73	6 Multi-Lane Distribution and Merging .....	26
74	6.1 Multi-Lane Interoperability and Lane-number Mismatch .....	27
75	6.1.1 Clock Considerations with Multi-Lane.....	28
76	6.1.2 Bidirectionality and Multi-Lane Capability.....	28
77	6.1.3 SoT and EoT in Multi-Lane Configurations.....	28

78	7	Low-Level Protocol Errors and Contention .....	31
79	7.1	Low-Level Protocol Errors .....	31
80	7.1.1	SoT Error .....	31
81	7.1.2	SoT Sync Error .....	32
82	7.1.3	EoT Sync Error .....	32
83	7.1.4	Escape Mode Entry Command Error .....	33
84	7.1.5	LP Transmission Sync Error .....	33
85	7.1.6	False Control Error .....	33
86	7.2	Contention Detection and Recovery .....	34
87	7.2.1	Contention Detection in LP Mode .....	34
88	7.2.2	Contention Recovery Using Timers .....	34
89	7.3	Additional Timers .....	37
90	7.3.1	Turnaround Acknowledge Timeout (TA_TO) .....	37
91	7.3.2	Peripheral Reset Timeout (PR_TO) .....	37
92	7.4	Acknowledge and Error Reporting Mechanism .....	38
93	8	DSI Protocol .....	39
94	8.1	Multiple Packets per Transmission .....	39
95	8.2	Packet Composition .....	40
96	8.3	Endian Policy .....	41
97	8.4	General Packet Structure .....	41
98	8.4.1	Long Packet Format .....	41
99	8.4.2	Short Packet Format .....	43
100	8.5	Common Packet Elements .....	43
101	8.5.1	Data Identifier Byte .....	43
102	8.5.2	Error Correction Code .....	44
103	8.6	Interleaved Data Streams .....	45
104	8.6.1	Interleaved Data Streams and Bidirectionality .....	45
105	8.7	Processor to Peripheral Direction (Processor-Sourced) Packet Data Types .....	46
106	8.8	Processor-to-Peripheral Transactions – Detailed Format Description .....	47
107	8.8.1	Sync Event (H Start, H End, V Start, V End), Data Type = XX 0001 (0xX1) .....	47
108	8.8.2	EoTp, Data Type = 00 1000 (0x08) .....	47
109	8.8.3	Color Mode Off Command, Data Type = 00 0010 (0x02) .....	48
110	8.8.4	Color Mode On Command, Data Type = 01 0010 (0x12) .....	48
111	8.8.5	Shutdown Peripheral Command, Data Type = 10 0010 (0x22) .....	49
112	8.8.6	Turn On Peripheral Command, Data Type = 11 0010 (0x32) .....	49
113	8.8.7	Generic Short WRITE Packet with 0, 1, or 2 parameters, Data Types = 00 0011 (0x03), 01	
114		0011 (0x13), 10 0011 (0x23), Respectively .....	49

115	8.8.8	Generic READ Request with 0, 1, or 2 Parameters, Data Types = 00 0100 (0x04), 01 0100 (0x14), 10 0100(0x24), Respectively .....	49
117	8.8.9	DCS Commands .....	50
118	8.8.10	Set Maximum Return Packet Size, Data Type = 11 0111 (0x37).....	51
119	8.8.11	Null Packet (Long), Data Type = 00 1001 (0x09).....	51
120	8.8.12	Blanking Packet (Long), Data Type = 01 1001 (0x19).....	51
121	8.8.13	Generic Long Write, Data Type = 10 1001 (0x29).....	51
122	8.8.14	Loosely Packed Pixel Stream, 20-bit YCbCr 4:2:2 Format, Data Type = 00 1100 (0x0C).....	51
123	8.8.15	Packed Pixel Stream, 24-bit YCbCr 4:2:2 Format, Data Type = 01 1100 (0x1C).....	53
124	8.8.16	Packed Pixel Stream, 16-bit YCbCr 4:2:2 Format, Data Type = 10 1100 (0x2C).....	54
125	8.8.17	Packed Pixel Stream, 30-bit Format, Long Packet, Data Type = 00 1101 (0x0D) .....	54
126	8.8.18	Packed Pixel Stream, 36-bit Format, Long Packet, Data Type = 01 1101 (0x1D) .....	55
127	8.8.19	Packed Pixel Stream, 12-bit YCbCr 4:2:0 Format, Data Type = 11 1101 (0x3D) .....	56
128	8.8.20	Packed Pixel Stream, 16-bit Format, Long Packet, Data Type 00 1110 (0x0E).....	57
129	8.8.21	Packed Pixel Stream, 18-bit Format, Long Packet, Data Type = 01 1110 (0x1E).....	58
130	8.8.22	Pixel Stream, 18-bit Format in Three Bytes, Long Packet, Data Type = 10 1110 (0x2E).....	60
131	8.8.23	Packed Pixel Stream, 24-bit Format, Long Packet, Data Type = 11 1110 (0x3E).....	61
132	8.8.24	DO NOT USE and Reserved Data Types .....	62
133	8.9	Peripheral-to-Processor (Reverse Direction) LP Transmissions .....	62
134	8.9.1	Packet Structure for Peripheral-to-Processor LP Transmissions .....	62
135	8.9.2	System Requirements for ECC and Checksum and Packet Format.....	63
136	8.9.3	Appropriate Responses to Commands and ACK Requests.....	63
137	8.9.4	Format of Acknowledge and Error Report and Read Response Data Types .....	65
138	8.9.5	Error Reporting Format .....	65
139	8.10	Peripheral-to-Processor Transactions – Detailed Format Description.....	67
140	8.10.1	Acknowledge and Error Report, Data Type 00 0010 (0x02) .....	68
141	8.10.2	Generic Short Read Response, 1 or 2 Bytes, Data Types = 01 0001 or 01 0010, Respectively	68
142		68	
143	8.10.3	Generic Long Read Response with Optional Checksum, Data Type = 01 1010 (0x1A).....	68
144	8.10.4	DCS Long Read Response with Optional Checksum, Data Type 01 1100 (0x1C) .....	69
145	8.10.5	DCS Short Read Response, 1 or 2 Bytes, Data Types = 10 0001 or 10 0010, Respectively .	69
146	8.10.6	Multiple Transmissions and Error Reporting .....	69
147	8.10.7	Clearing Error Bits.....	69
148	8.11	Video Mode Interface Timing .....	69
149	8.11.1	Transmission Packet Sequences .....	70
150	8.11.2	Non-Burst Mode with Sync Pulses .....	71
151	8.11.3	Non-Burst Mode with Sync Events .....	72
152	8.11.4	Burst Mode .....	73

153 8.11.5 Parameters .....74

154 8.12 TE Signaling in DSI .....75

155 9 Error-Correcting Code (ECC) and Checksum .....77

156 9.1 Packet Header Error Detection/Correction .....77

157 9.2 Hamming Code Theory .....77

158 9.3 Hamming-modified Code Applied to DSI Packet Headers .....78

159 9.4 ECC Generation on the Transmitter .....81

160 9.5 Applying ECC on the Receiver .....82

161 9.6 Checksum Generation for Long Packet Payloads.....82

162 10 Compliance, Interoperability, and Optional Capabilities .....84

163 10.1 Display Resolutions.....84

164 10.2 Pixel Formats.....85

165 10.2.1 Video Mode .....85

166 10.2.2 Command Mode .....85

167 10.3 Number of Lanes .....85

168 10.4 Maximum Lane Frequency.....85

169 10.5 Bidirectional Communication.....86

170 10.6 ECC and Checksum Capabilities.....86

171 10.7 Display Architecture.....86

172 10.8 Multiple Peripheral Support .....86

173 10.9 EoTp Support and Interoperability .....86

174 Annex A Contention Detection and Recovery Mechanisms (informative).....87

175 A.1 PHY Detected Contention .....87

176 A.1.1 Protocol Response to PHY Detected Faults.....87

177 Annex B Checksum Generation Example (informative) .....93

178 Annex C Interlaced Video Transmission Sourcing.....95

179

180

181	<b>Figures</b>	
182	Figure 1 DSI Transmitter and Receiver Interface.....	17
183	Figure 2 DSI Layers .....	18
184	Figure 3 Basic HS Transmission Structure.....	21
185	Figure 4 Peripheral Power-Up Sequencing Example .....	25
186	Figure 5 Lane Distributor Conceptual Overview .....	26
187	Figure 6 Lane Merger Conceptual Overview .....	27
188	Figure 7 Four-Lane Transmitter with Two-Lane Receiver Example .....	28
189	Figure 8 Two Lane HS Transmission Example.....	29
190	Figure 9 Three Lane HS Transmission Example.....	30
191	Figure 10 HS Transmission Examples with EoTp disabled .....	40
192	Figure 11 HS Transmission Examples with EoTp enabled .....	40
193	Figure 12 Endian Example (Long Packet).....	41
194	Figure 13 Long Packet Structure.....	42
195	Figure 14 Short Packet Structure.....	43
196	Figure 15 Data Identifier Byte.....	44
197	Figure 16 Interleaved Data Stream Example with EoTp disabled.....	45
198	Figure 17 Logical Channel Block Diagram (Receiver Case) .....	45
199	Figure 18 20-bit per Pixel – YCbCr 4:2:2 Format, Long Packet.....	52
200	Figure 19 24-bit per Pixel – YCbCr 4:2:2 Format, Long Packet.....	53
201	Figure 20 16-bit per Pixel – YCbCr 4:2:2 Format, Long Packet.....	54
202	Figure 21 30-bit per Pixel (Packed) – RGB Color Format, Long Packet .....	55
203	Figure 22 36-bit per Pixel (Packed) – RGB Color Format, Long Packet .....	56
204	Figure 23 12-bit per Pixel – YCbCr 4:2:0 Format (Odd Line), Long Packet .....	57
205	Figure 24 12-bit per Pixel – YCbCr 4:2:0 Format (Even Line), Long Packet.....	57
206	Figure 25 16-bit per Pixel – RGB Color Format, Long Packet .....	58
207	Figure 26 18-bit per Pixel (Packed) – RGB Color Format, Long Packet .....	59
208	Figure 27 18-bit per Pixel (Loosely Packed) – RGB Color Format, Long Packet .....	60
209	Figure 28 24-bit per Pixel – RGB Color Format, Long Packet .....	61
210	Figure 29 Video Mode Interface Timing Legend.....	71
211	Figure 30 Video Mode Interface Timing: Non-Burst Transmission with Sync Start and End.....	72
212	Figure 31 Video Mode Interface Timing: Non-burst Transmission with Sync Events .....	73
213	Figure 32 Video Mode Interface Timing: Burst Transmission.....	74
214	Figure 33 24-bit ECC generation on TX side .....	81
215	Figure 34 24-bit ECC on RX Side Including Error Correction .....	82
216	Figure 35 Checksum Transmission .....	83

217 Figure 36 16-bit CRC Generation Using a Shift Register .....83  
218 Figure 37 LP High  $\leftrightarrow$  LP Low Contention Case 1 .....89  
219 Figure 38 LP High  $\leftrightarrow$  LP Low Contention Case 2 .....91  
220 Figure 39 LP High  $\leftrightarrow$  LP Low Contention Case 3 .....92  
221 Figure 40 Video Mode Interface Timing: Non-burst Transmission with Sync Start and End (Interlaced  
222 Video).....95  
223 Figure 41 Video Mode Interface Timing: Non-burst Transmission with Sync Events (Interlaced Video) ...96  
224  
225



226

**Tables**

227	Table 1 Sequence of Events to Resolve SoT Error (HS RX Side) .....	32
228	Table 2 Sequence of Events to Resolve SoT Sync Error (HS RX Side).....	32
229	Table 3 Sequence of Events to Resolve EoT Sync Error (HS RX Side) .....	33
230	Table 4 Sequence of Events to Resolve Escape Mode Entry Command Error (RX Side) .....	33
231	Table 5 Sequence of Events to Resolve LP Transmission Sync Error (RX Side) .....	33
232	Table 6 Sequence of Events to Resolve False Control Error (RX Side).....	34
233	Table 7 Low-Level Protocol Error Detection and Reporting .....	34
234	Table 8 Required Timers and Timeout Summary .....	35
235	Table 9 Sequence of Events for HS RX Timeout (Peripheral initially HS RX) .....	35
236	Table 10 Sequence of Events for HS TX Timeout (Host Processor initially HS TX).....	36
237	Table 11 Sequence of Events for LP TX-Peripheral Timeout (Peripheral initially LP TX).....	36
238	Table 12 Sequence of Events for Host Processor Wait Timeout (Peripheral initially TX) .....	36
239	Table 13 Sequence of Events for BTA Acknowledge Timeout (Peripheral initially TX).....	37
240	Table 14 Sequence of Events for BTA Acknowledge Timeout (Host Processor initially TX) .....	37
241	Table 15 Sequence of Events for Peripheral Reset Timeout .....	37
242	Table 16 Data Types for Processor-sourced Packets.....	46
243	Table 17 EoT Support for Host and Peripheral .....	48
244	Table 18 Error Report Bit Definitions .....	66
245	Table 19 Data Types for Peripheral-sourced Packets.....	67
246	Table 20 Required Peripheral Timing Parameters.....	74
247	Table 21 ECC Syndrome Association Matrix .....	78
248	Table 22 ECC Parity Generation Rules .....	79
249	Table 23 Display Resolutions.....	84
250	Table 24 LP High $\leftrightarrow$ LP Low Contention Case 1 .....	87
251	Table 25 LP High $\leftrightarrow$ LP Low Contention Case 2 .....	90
252	Table 26 LP High $\leftrightarrow$ LP Low Contention Case 3 .....	92

253

# 254 **DRAFT MIPI Alliance Specification for** 255 **Display Serial Interface**

## 256 **1 Overview**

257 The Display Serial Interface Specification defines protocols between a host processor and peripheral  
258 devices that adhere to MIPI Alliance specifications for mobile device interfaces. The DSI specification  
259 builds on existing specifications by adopting pixel formats and command set defined in [MIPI02],  
260 [MIPI03], and [MIPI01].

### 261 **1.1 Scope**

262 Interface protocols as well as a description of signal timing relationships are within the scope of this  
263 document.

264 Electrical specifications and physical specifications are out of scope for this document. In addition, legacy  
265 interfaces such as DPI-2 and DBI-2 are also out of scope for this document. Furthermore, device usage of  
266 auxiliary buses such as I<sup>2</sup>C or SPI, while not precluded by this specification, are also not within its scope.

### 267 **1.2 Purpose**

268 The Display Serial Interface specification defines a high-speed serial interface between a peripheral, such  
269 as an active-matrix display module, and a host processor in a mobile device. By standardizing this  
270 interface, components may be developed that provide higher performance, lower power, less EMI and  
271 fewer pins than current devices, while maintaining compatibility across products from multiple vendors.  
272

## 273 **2 Terminology (informative)**

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

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

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

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

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

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

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

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

292 Numbers are decimal unless otherwise indicated. Hexadecimal numbers have a “0x” prefix. Binary  
293 numbers are prefixed by “0b”.

### 294 **2.1 Definitions**

295 **Forward Direction:** The signal direction is defined relative to the direction of the high-speed serial clock.  
296 Transmission from the side sending the clock to the side receiving the clock is the forward direction.

297 **Half duplex:** Bidirectional data transmission over a Lane allowing both transmission and reception but  
298 only in one direction at a time.

299 **HS Transmission:** Sending one or more packets in the forward direction in HS Mode. A HS Transmission  
300 is delimited before and after packet transmission by LP-11 states.

301 **Host Processor:** Hardware and software that provides the core functionality of a mobile device.

302 **Lane:** Consists of two complementary Lane Modules communicating via two-line, point-to-point Lane  
303 Interconnects. A Lane can be used for either Data or Clock signal transmission.

304 **Lane Interconnect:** Two-line, point-to-point interconnect used for both differential high-speed signaling  
305 and low-power, single-ended signaling.

306 **Lane Module:** Module at each side of the Lane for driving and/or receiving signals on the Lane.

307 **Link:** A connection between two devices containing one Clock Lane and at least one Data Lane. A Link  
308 consists of two PHYs and two Lane Interconnects.

309 **LP Transmission:** Sending one or more packets in either direction in LP Mode or Escape Mode. A LP  
310 Transmission is delimited before and after packet transmission by LP-11 states.

311 **Packet:** A group of four or more bytes organized in a specified way to transfer data across the interface. All  
312 packets have a minimum specified set of components. The byte is the fundamental unit of data from which  
313 packets are made.

314 **Payload:** Application data only – with all Link synchronization, header, ECC and checksum and other  
315 protocol-related information removed. This is the “core” of transmissions between host processor and  
316 peripheral.

317 **PHY:** The set of Lane Modules on one side of a Link.

318 **PHY Configuration:** A set of Lanes that represent a possible Link. A PHY configuration consists of a  
319 minimum of two Lanes: one Clock Lane and one or more Data Lanes.

320 **Reverse Direction:** Reverse direction is the opposite of the forward direction. See the description for  
321 Forward Direction.

322 **Transmission:** Refers to either HS or LP Transmission. See the HS Transmission and LP Transmission  
323 definitions for descriptions of the different transmission modes.

324 **Virtual Channel:** Multiple independent data streams for up to four peripherals are supported by this  
325 specification. The data stream for each peripheral is a *Virtual Channel*. These data streams may be  
326 interleaved and sent as sequential packets, with each packet dedicated to a particular peripheral or channel.  
327 Packet protocol includes information that directs each packet to its intended peripheral.

328 **Word Count:** Number of bytes within the payload.

## 329 **2.2 Abbreviations**

330 e.g. For example

## 331 **2.3 Acronyms**

332 AIP Application Independent Protocol

333 AM Active matrix (display technology)

334 ASP Application Specific Protocol

335 BLLP Blanking or Low Power interval

336 BPP Bits per Pixel

337 BTA Bus Turn-Around

338 CSI Camera Serial Interface

339 DBI Display Bus Interface

340	DI	Data Identifier
341	DMA	Direct Memory Access
342	DPI	Display Pixel Interface
343	DSI	Display Serial Interface
344	<u>DT</u>	<u>Data Type</u>
345	ECC	Error-Correcting Code
346	EMI	Electro Magnetic interference
347	<u>EoTp</u>	<u>End of Transmission Packet</u>
348	ESD	Electrostatic Discharge
349	<u>Fps</u>	<u>Frames per second</u>
350	HBP	Horizontal Back Porch
351	HFP	Horizontal Front Porch
352	<u>HS</u>	<u>High Speed</u>
353	HSA	Horizontal Sync Active
354	HSE	Horizontal Sync End
355	HSS	Horizontal Sync Start
356	ISTO	Industry Standards and Technology Organization
357	LP	Low Power
358	LPS	Low Power State (state of serial data line when not transferring high-speed serial data)
359	LSB	Least Significant Bit
360	Mbps	Megabits per second
361	MIPI	Mobile Industry Processor Interface
362	MSB	Most Significant Bit
363	PF	Packet Footer
364	PH	Packet Header
365	PHY	Physical Layer
366	PPI	PHY-Protocol Interface

367	QCIF	Quarter-size CIF (resolution 176x144 pixels or 144x176 pixels)
368	QVGA	Quarter-size Video Graphics Array (resolution 320x240 pixels or 240x320 pixels)
369	RGB	Color presentation (Red, Green, Blue)
370	SLVS	Scalable Low Voltage Signaling
371	<u>SoT</u>	<u>Start of Transmission</u>
372	SVGA	Super Video Graphics Array (resolution 800x600 pixels or 600x800 pixels)
373	ULPS	Ultra-low Power State
374	VGA	Video Graphics Array (resolution 640x480 pixels or 480x640 pixels)
375	VSA	Vertical Sync Active
376	VSE	Vertical Sync End
377	VSS	Vertical Sync Start
378	<u>WC</u>	<u>Word Count</u>
379	WVGA	Wide VGA (resolution 800x480 pixels or 480x800 pixels)
380		

### 381 **3 References (informative)**

- 382 [MIP101] *MIPI Alliance Specification for Display Command Set*, version 1.02.00, MIPI Alliance,  
383 In Press
- 384 [MIP102] *MIPI Alliance Standard for Display Bus Interface (DBI-2)*, version 2.00, MIPI Alliance,  
385 29 November 2005
- 386 [MIP103] *MIPI Alliance Standard for Display Pixel Interface (DPI-2)*, version 2.00, MIPI Alliance,  
387 15 September 2005
- 388 [MIP104] *MIPI Alliance Specification for D-PHY*, version 0.90.00, MIPI Alliance, 8 October 2007
- 389 [CEA01] CEA-861-E, *A DTV Profile for Uncompressed High Speed Digital Interfaces*,  
390 <[http://www.ce.org/Standards/browseByCommittee\\_2641.asp](http://www.ce.org/Standards/browseByCommittee_2641.asp)>, Consumer Electronics  
391 Association, March 2008
- 392 [ITU01] BT.601-6, *Studio encoding parameters of digital television for standard 4:3 and wide  
393 screen 16:9 aspect ratios*, <<http://www.itu.int/rec/R-REC-BT.601-6-200701-I/en>>,  
394 International Telecommunications Union, 23 February 2007
- 395 [ITU02] BT.709-5, *Parameter values for the HDTV standards for production and international  
396 programme exchange*, <<http://www.itu.int/rec/R-REC-BT.709-5-200204-I/en>>,  
397 International Telecommunications Union, 27 August 2009
- 398 [ITU03] BT.656-5, *Interface for digital component video signals in 525-line and 625-line  
399 television systems operating at the 4:2:2 level of Recommendation ITU-R BT.601*,  
400 <<http://www.itu.int/rec/R-REC-BT.656-5-200712-I/en>>, International  
401 Telecommunications Union, 1 January 2008
- 402 [SMPT01] EG 36-2000, *Transformations Between Television Component Color Signals*, Society for  
403 Motion Picture and Television Engineers, 23 March 2000

404 Much of DSI is based on existing MIPI Alliance specifications as well as several MIPI Alliance  
405 specifications in simultaneous development. In the Application Layer, DSI duplicates pixel formats used in  
406 [MIP103] when it is in *Video Mode* operation. For display modules with a display controller and frame  
407 buffer, DSI shares a common command set with [MIP102]. The command set is documented in [MIP101].

#### 408 **3.1 Display Bus Interface Standard for Parallel Signaling (DBI-2)**

409 DBI-2 is a MIPI Alliance standard for parallel interfaces to display modules having display controllers and  
410 frame buffers. For systems based on these standards, the host processor loads images to the on-panel frame  
411 buffer through the display processor. Once loaded, the display controller manages all display refresh  
412 functions on the display module without further intervention from the host processor. Image updates  
413 require the host processor to write new data into the frame buffer.

414 DBI-2 specifies a parallel interface where data can be sent to the peripheral over an 8-, 9- or 16-bit-wide  
415 data bus, with additional control signals. DBI-2 supports a 1-bit data bus interface mode as well.

416 The DSI specification supports a Command Mode of operation. Like the parallel DBI, a DSI-compliant  
417 interface sends commands and parameters to the display. However, all information in DSI is first serialized

418 before transmission to the display module. At the display, serial information is transformed back to parallel  
419 data and control signals for the on-panel display controller. Similarly, the display module can return status  
420 information and requested memory data to the host processor, using the same serial data path.

### 421 **3.2 Display Pixel Interface Standard for Parallel Signaling (DPI-2)**

422 DPI-2 is a MIPI Alliance standard for parallel interfaces to display modules without on-panel display  
423 controller or frame buffer. These display modules rely on a steady flow of pixel data from host processor to  
424 the display, to maintain an image without flicker or other visual artifacts. MIPI Alliance standards  
425 document several pixel formats for *Active Matrix* (AM) display modules.

426 Like DBI-2, DPI-2 is a MIPI Alliance standard for parallel interfaces. The data path may be 16-, 18-, or 24-  
427 bits wide, depending on pixel format(s) supported by the display module. This document refers to DPI  
428 mode of operation as Video Mode.

429 Some display modules that use Video Mode in normal operation also make use of a simplified form of  
430 Command Mode, when in low-power state. These display modules can shut down the streaming video  
431 interface and continue to refresh the screen from a small local frame buffer, at reduced resolution and pixel  
432 depth. The local frame buffer shall be loaded, prior to interface shutdown, with image content to be  
433 displayed when in low-power operation. These display modules can switch mode in response to power-  
434 control commands.

### 435 **3.3 MIPI Alliance Specification for Display Command Set (DCS)**

436 DCS is a MIPI Alliance specification for the command set used by DSI and DBI-2 standards. Commands  
437 are sent from the host processor to the display module. On the display module, a display controller receives  
438 and interprets commands, then takes appropriate action. Commands fall into four broad categories: read  
439 register, write register, read memory and write memory. A command may be accompanied by multiple  
440 parameters.

### 441 **3.4 MIPI Alliance Standard for Camera Serial Interface 2 (CSI-2)**

442 CSI-2 is a MIPI Alliance standard for serial interface between a camera module and host processor. It is  
443 based on the same physical layer technology and low-level protocols as DSI. Some significant differences  
444 between DSI and CSI-2 are:

- 445 • CSI-2 uses unidirectional high-speed Link, whereas DSI is half-duplex bidirectional Link
- 446 • CSI-2 makes use of a secondary channel, based on I<sup>2</sup>C, for control and status functions

447 CSI-2 data direction is from peripheral (Camera Module) to host processor, while DSI's primary data  
448 direction is from host processor to peripheral (Display Module).

### 449 **3.5 MIPI Alliance Specification for D-PHY (D-PHY)**

450 [MIPI04] provides the physical layer definition for DSI. The functionality specified by the D-PHY  
451 specification covers all electrical and timing aspects, as well as low-level protocols, signaling, and message  
452 transmissions in various operating modes.

453

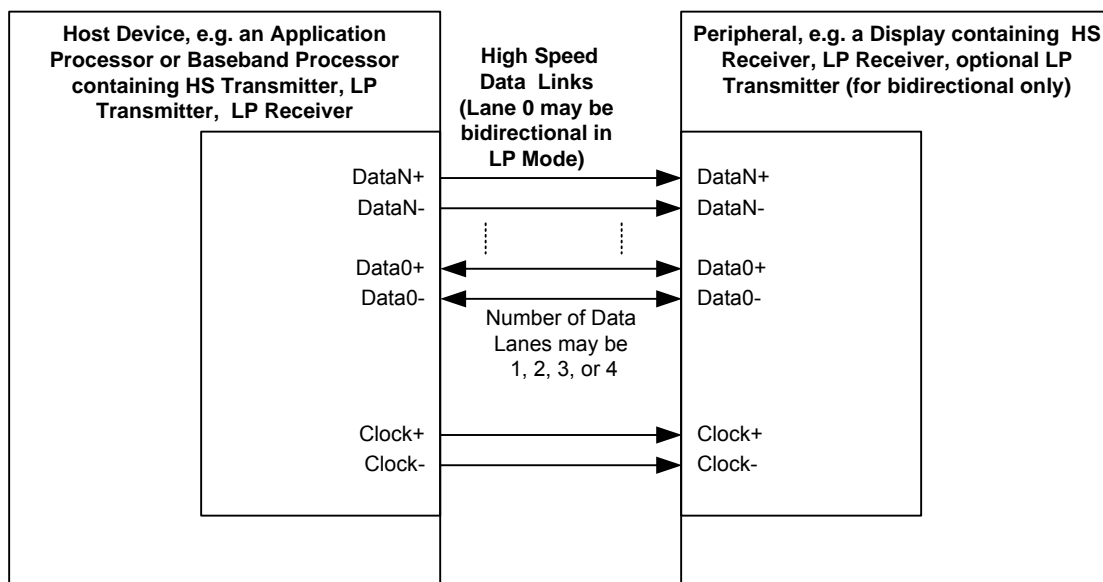


454 **4 DSI Introduction**

455 DSI specifies the interface between a host processor and a peripheral such as a display module. It builds on  
 456 existing MIPI Alliance specifications by adopting pixel formats and command set specified in DPI-2, DBI-  
 457 2 and DCS standards.

458 Figure 1 shows a simplified DSI interface. From a conceptual viewpoint, a DSI-compliant interface  
 459 performs the same functions as interfaces based on DBI-2 and DPI-2 standards or similar parallel display  
 460 interfaces. It sends pixels or commands to the peripheral, and can read back status or pixel information  
 461 from the peripheral. The main difference is that DSI serializes all pixel data, commands, and events that, in  
 462 traditional or legacy interfaces, are normally conveyed to and from the peripheral on a parallel data bus  
 463 with additional control signals.

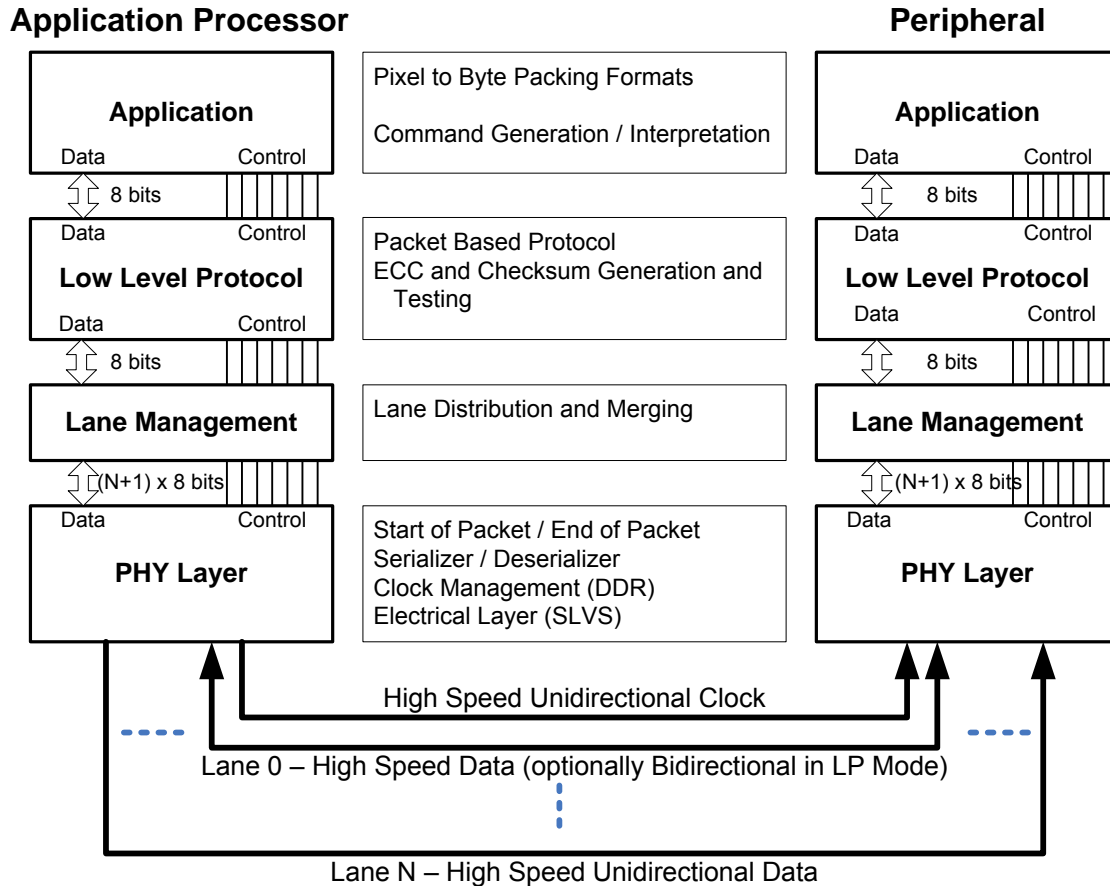
464 From a system or software point of view, the serialization and deserialization operations should be  
 465 transparent. The most visible, and unavoidable, consequence of transformation to serial data and back to  
 466 parallel is increased latency for transactions that require a response from the peripheral. For example,  
 467 reading a pixel from the frame buffer on a display module has a higher latency using DSI than DBI.  
 468 Another fundamental difference is the host processor's inability during a read transaction to throttle the  
 469 rate, or size, of returned data.



470

471

**Figure 1 DSI Transmitter and Receiver Interface**

472 **4.1 DSI Layer Definitions**

473

474

**Figure 2 DSI Layers**

475 A conceptual view of DSI organizes the interface into several functional layers. A description of the layers  
 476 follows and is also shown in Figure 2.

477 **PHY Layer:** The *PHY Layer* specifies transmission medium (electrical conductors), the input/output  
 478 circuitry and the clocking mechanism that captures “ones” and “zeroes” from the serial bit stream. This part  
 479 of the specification documents the characteristics of the transmission medium, electrical parameters for  
 480 signaling and the timing relationship between clock and Data Lanes.

481 The mechanism for signaling Start of Transmission (SoT) and End of Transmission (EoT) is specified, as  
 482 well as other “out of band” information that can be conveyed between transmitting and receiving PHYs.  
 483 Bit-level and byte-level synchronization mechanisms are included as part of the PHY. Note that the  
 484 electrical basis for DSI (SLVS) has two distinct modes of operation, each with its own set of electrical  
 485 parameters.

486 The PHY layer is described in [MIPI04].

487 **Lane Management Layer:** DSI is Lane-scalable for increased performance. The number of data signals  
 488 may be 1, 2, 3, or 4 depending on the bandwidth requirements of the application. The transmitter side of the  
 489 interface distributes the outgoing data stream to one or more Lanes (“distributor” function). On the

490 receiving end, the interface collects bytes from the Lanes and merges them together into a recombined data  
491 stream that restores the original stream sequence (“merger” function).

492 **Protocol Layer:** At the lowest level, DSI protocol specifies the sequence and value of bits and bytes  
493 traversing the interface. It specifies how bytes are organized into defined groups called packets. The  
494 protocol defines required headers for each packet, and how header information is generated and interpreted.  
495 The transmitting side of the interface appends header and error-checking information to data being  
496 transmitted. On the receiving side, the header is stripped off and interpreted by corresponding logic in the  
497 receiver. Error-checking information may be used to test the integrity of incoming data. DSI protocol also  
498 documents how packets may be tagged for interleaving multiple command or data streams to separate  
499 destinations using a single DSI.

500 **Application Layer:** This layer describes higher-level encoding and interpretation of data contained in the  
501 data stream. Depending on the display subsystem architecture, it may consist of pixels having a prescribed  
502 format, or of commands that are interpreted by the display controller inside a display module. The DSI  
503 specification describes the mapping of pixel values, commands and command parameters to bytes in the  
504 packet assembly. See [MIPI01].

## 505 4.2 Command and Video Modes

506 DSI-compliant peripherals support either of two basic modes of operation: Command Mode and Video  
507 Mode. Which mode is used depends on the architecture and capabilities of the peripheral. The mode  
508 definitions reflect the primary intended use of DSI for display interconnect, but are not intended to restrict  
509 DSI from operating in other applications.

510 Typically, a peripheral is capable of Command Mode operation or Video Mode operation. Some Video  
511 Mode display modules also include a simplified form of Command Mode operation in which the display  
512 module may refresh its screen from a reduced-size, or partial, frame buffer, and the interface (DSI) to the  
513 host processor may be shut down to reduce power consumption.

### 514 4.2.1 Command Mode

515 Command Mode refers to operation in which transactions primarily take the form of sending commands  
516 and data to a peripheral, such as a display module, that incorporates a display controller. The display  
517 controller may include local registers and a frame buffer. Systems using Command Mode write to, and read  
518 from, the registers and frame buffer memory. The host processor indirectly controls activity at the  
519 peripheral by sending commands, parameters and data to the display controller. The host processor can also  
520 read display module status information or the contents of the frame memory. Command Mode operation  
521 requires a bidirectional interface.

### 522 4.2.2 Video Mode Operation

523 Video Mode refers to operation in which transfers from the host processor to the peripheral take the form of  
524 a real-time pixel stream. In normal operation, the display module relies on the host processor to provide  
525 image data at sufficient bandwidth to avoid flicker or other visible artifacts in the displayed image. Video  
526 information should only be transmitted using High Speed Mode.

527 Some Video Mode architectures may include a simple timing controller and partial frame buffer, used to  
528 maintain a partial-screen or lower-resolution image in standby or Low Power Mode. This permits the  
529 interface to be shut down to reduce power consumption.

530 To reduce complexity and cost, systems that only operate in Video Mode may use a unidirectional data  
531 path.

### 532 **4.2.3 Virtual Channel Capability**

533 While this specification only addresses the connection of a host processor to a single peripheral, DSI  
534 incorporates a virtual channel capability for communication between a host processor and multiple,  
535 physical display modules. . A bridge device may create multiple, separate connections to display modules  
536 or other devices or a display module or device may support multiple virtual channels. Display modules are  
537 completely independent, may operate simultaneously, and may be of different display architecture types,  
538 limited only by the total bandwidth available over the shared DSI Link. The details of connecting multiple  
539 peripherals to a single Link are beyond the scope of this document.

540 Since interface bandwidth is shared between peripherals, there are constraints that limit the physical extent  
541 and performance of multiple-peripheral systems.

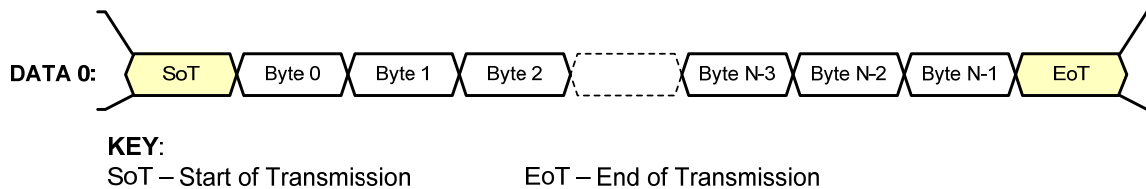
542 The DSI protocol permits up to four virtual channels, enabling traffic for multiple peripherals to share a  
543 common DSI Link. For example, in some high-resolution display designs, multiple physical drivers serve  
544 different areas of a common display panel. Each driver is integrated with its own display controller that  
545 connects to the host processor through DSI. Using virtual channels, the display controller directs data to the  
546 individual drivers, eliminating the need for multiple interfaces or complex multiplexing schemes. Virtual  
547 channels may also be employed by devices where one channel is a bidirectional Command Mode channel  
548 and the second channel is a Video Mode unidirectional channel. Virtual channels may be employed by a  
549 display module or a DSI bridge device receiving interlaced video from the host-processor, where one  
550 channel is corresponding to the first field and another channel is the second field of an interlaced video  
551 frame. The DSI specification makes no requirements on the specific value assigned to each virtual channel  
552 used to designate interlaced fields, For clarity, the first interlaced video field may be assigned as DI[7:6] =  
553 0b00 and the second interlaced video field may be assigned DI[7:6] = 0b01.  
554

## 555 5 DSI Physical Layer

556 This section provides a brief overview of the physical layer used in DSI. See [MIPI04] for more details.

557 Information is transferred between host processor and peripheral using one or more serial data signals and  
558 accompanying serial clock. The action of sending high-speed serial data across the bus is called a *HS*  
559 *transmission* or *burst*.

560 Between transmissions, the differential data signal or Lane goes to a low-power state (LPS). Interfaces  
561 should be in LPS when they are not actively transmitting or receiving high-speed data. Figure 3 shows the  
562 basic structure of a HS transmission.  $N$  is the total number of bytes sent in the transmission.



564 **Figure 3 Basic HS Transmission Structure**

565 D-PHY low-level protocol specifies a minimum data unit of one byte, and a transmission contains an  
566 integer number of bytes.

### 567 5.1 Data Flow Control

568 There is no handshake between the Protocol and PHY layers that permit the Protocol layer to throttle data  
569 transfer to, or from, the PHY layer once transmission is underway. Packets shall be sent and received in  
570 their entirety and without interruption. The Protocol layer and data buffering on both ends of the Link shall  
571 always have bandwidth equal to, or greater than, PHY layer circuitry. A practical consequence is that the  
572 system implementer should ensure that receivers have bandwidth capability that is equal to, or greater than,  
573 that of the transmitter.

### 574 5.2 Bidirectionality and Low Power Signaling Policy

575 The physical layer for a DSI implementation is composed of one to four Data Lanes and one Clock Lane. In  
576 a Command Mode system, Data Lane 0 shall be bidirectional; additional Data Lanes shall be unidirectional.  
577 In a Video Mode system, Data Lane 0 may be bidirectional or unidirectional; additional Data Lanes shall be  
578 unidirectional. See sections 5.3 and 5.4 for details.

579 For both interface types, the Clock Lane shall be driven by the host processor only, never by the peripheral.

580 Forward direction Low Power transmissions shall use Data Lane 0 only. Reverse direction transmissions on  
581 Data Lane 0 shall use Low Power Mode only. The peripheral shall be capable of receiving any transmission  
582 in Low Power or High Speed Mode. Note that transmission bandwidth is substantially reduced when  
583 transmitting in LP mode.

584 For bidirectional Lanes, data shall be transmitted in the peripheral-to-processor, or reverse, direction using  
585 Low-Power (LP) Mode only. See [MIPI04] for details on the different modes of transmission.

586 The interface between PHY and Protocol layers has several signals controlling bus direction. When a host  
 587 transmitter requires a response from a peripheral, e.g. returning READ data or status information, it asserts  
 588 TurnRequest to its PHY during the last packet of the transmission. This tells the PHY layer to assert the  
 589 Bus Turn-Around (BTA) command following the EoT sequence.

590 When a peripheral receives the Bus Turn-Around command, its PHY layer asserts TurnRequest as an input  
 591 to the Protocol layer. This tells the receiving Protocol layer that it shall prepare to send a response to the  
 592 host processor. Normally, the packet just received tells the Protocol layer what information to send once the  
 593 bus is available for transmitting to the host processor.

594 After transmitting its response, the peripheral similarly hands bus control back to the host processor using a  
 595 TurnRequest to its own PHY layer.

### 596 **5.3 Command Mode Interfaces**

597 The minimum physical layer requirement for a DSI host processor operating in Command Mode is:

- 598 • Data Lane Module: CIL-MFAA (HS-TX, LP-TX, LP-RX, and LP-CD)
- 599 • Clock Lane Module: CIL-MCNN (HS-TX, LP-TX)

600 The minimum physical layer requirement for a DSI peripheral operating in Command Mode is:

- 601 • Data Lane Module: CIL-SFAA (HS-RX, LP-RX, LP-TX, and LP-CD)
- 602 • Clock Lane Module: CIL-SCNN (HS-RX, LP-RX)

603 A Bidirectional Link shall support reverse-direction Escape Mode for Data Lane 0 to support LPDT for  
 604 read data as well as ACK and TE Trigger Messages issued by the peripheral. In the forward direction, Data  
 605 Lane 0 shall support LPDT as described in [MIPI04]. All Trigger messages shall be communicated across  
 606 Data Lane 0.

### 607 **5.4 Video Mode Interfaces**

608 The minimum physical layer requirement for a DSI transmitter operating in Video Mode is:

- 609 • Data Lane Module: CIL-MFAN (HS-TX, LP-TX)
- 610 • Clock Lane Module: CIL-MCNN (HS-TX, LP-TX)

611 The minimum physical layer requirement for a DSI receiver operating in Video Mode is:

- 612 • Data Lane Module: CIL-SFAN (HS-RX, LP-RX)
- 613 • Clock Lane Module: CIL-SCNN (HS-RX, LP-RX)

614 In the forward direction, Data Lane 0 shall support LPDT as described in [MIPI04]. All Trigger messages  
 615 shall be communicated across Data Lane 0.

### 616 **5.5 Bidirectional Control Mechanism**

617 Turning the bus around is controlled by a token-passing mechanism: the host processor sends a Bus Turn-  
 618 Around (BTA) request, which conveys to the peripheral its intention to release, or stop driving, the data  
 619 path after which the peripheral can transmit one or more packets back to the host processor. When it is  
 620 finished, the peripheral shall return control of the bus back to the host processor. Bus Turn-Around is  
 621 signaled using an Escape Mode mechanism provided by PHY-level protocol.

622 In bidirectional systems, there is a remote chance of erroneous behavior due to EMI that could result in bus  
623 contention. Mechanisms are provided in this specification for recovering from any bus contention event  
624 without forcing “hard reset” of the entire system.

## 625 5.6 Clock Management

626 DSI Clock is a signal from the host processor to the peripheral. In some systems, it may serve multiple  
627 functions:

628 **DSI Bit Clock:** Across the Link, DSI Clock is used as the source-synchronous bit clock for capturing serial  
629 data bits in the receiver PHY. This clock shall be active while data is being transferred.

630 **Byte Clock:** Divided down, DSI Clock is used to generate a byte clock at the conceptual interface between  
631 the Protocol and Application layers. During HS transmission, each byte of data is accompanied by a byte  
632 clock. Like the DSI Bit Clock, the byte clock shall be active while data is being transferred. At the Protocol  
633 layer to Application layer interface, all actions are synchronized to the byte clock.

634 **Application Clock(s):** Divided-down versions of DSI Bit Clock may be used for other clocked functions at  
635 the peripheral. These “application clocks” may need to run at times when no serial data is being transferred,  
636 or they may need to run constantly (continuous clock) to support active circuitry at the peripheral. Details  
637 of how such additional clocks are generated and used are beyond the scope of this document.

638 For continuous clock behavior, the Clock Lane remains in high-speed mode generating active clock signals  
639 between HS data packet transmissions. For non-continuous clock behavior, the Clock Lane enters the LP-  
640 11 state between HS data packet transmissions.

### 641 5.6.1 Clock Requirements

642 All DSI transmitters and receivers shall support continuous clock behavior on the Clock Lane, and  
643 optionally may support non-continuous clock behavior. A DSI host processor shall support continuous  
644 clock for systems that require it, as well as having the capability of shutting down the serial clock to reduce  
645 power.

646 Note that the host processor controls the desired mode of clock operation. Host protocol and applications  
647 control Clock Lane operating mode (High Speed or Low Power mode). System designers are responsible  
648 for understanding the clock requirements for peripherals attached to DSI and controlling clock behavior in  
649 accordance with those requirements.

650 Note that in Low Power signaling mode, LP clock is functionally embedded in the data signals. When LP  
651 data transmission ends, the clock effectively stops and subsequent LP clocks are not available to the  
652 peripheral. The peripheral shall not require additional bits, bytes, or packets from the host processor in  
653 order to complete processing or pipeline movement of received data in LP mode transmissions. There are a  
654 variety of ways to meet this requirement. For example, the peripheral may generate its own clock or it may  
655 require the host processor to keep the HS serial clock running.

656 The handshake process for BTA allows only limited mismatch of Escape Mode clock frequencies between  
657 a host processor and a peripheral. The Escape Mode frequency ratio between host processor and peripheral  
658 shall not exceed 3:2. The host processor is responsible for controlling its own clock frequency to match the  
659 peripheral. The host processor LP clock frequency shall be in the range of 67% to 150% of peripheral LP  
660 clock frequency. Therefore, the peripheral implementer shall specify a peripheral’s nominal LP clock  
661 frequency and the guaranteed accuracy.

## 662 5.6.2 Clock Power and Timing

663 Additional timing requirements in [MIPI04] specify the timing relationship between the power state of data  
664 signal(s) and the power state of the clock signal. It is the responsibility of the host processor to observe this  
665 timing relationship. If the DSI Clock runs continuously, these timing requirements do not apply.

## 666 5.7 System Power-Up and Initialization

667 System power-up is a multi-state process that depends not only on initialization of the master (host  
668 processor) and slave (peripheral) devices, but also possibly on internal delays within the slave device. This  
669 section specifies the parameters necessary for operation, and makes several recommendations to help  
670 ensure the system power-up process is robust.

671 After power-up, the host processor shall observe an initialization period,  $T_{INIT}$ , during which it shall drive a  
672 sustained TX-Stop state (LP-11) on all Lanes of the Link. See [MIPI04] for descriptions of  $T_{INIT}$  and the  
673 TX-Stop state.

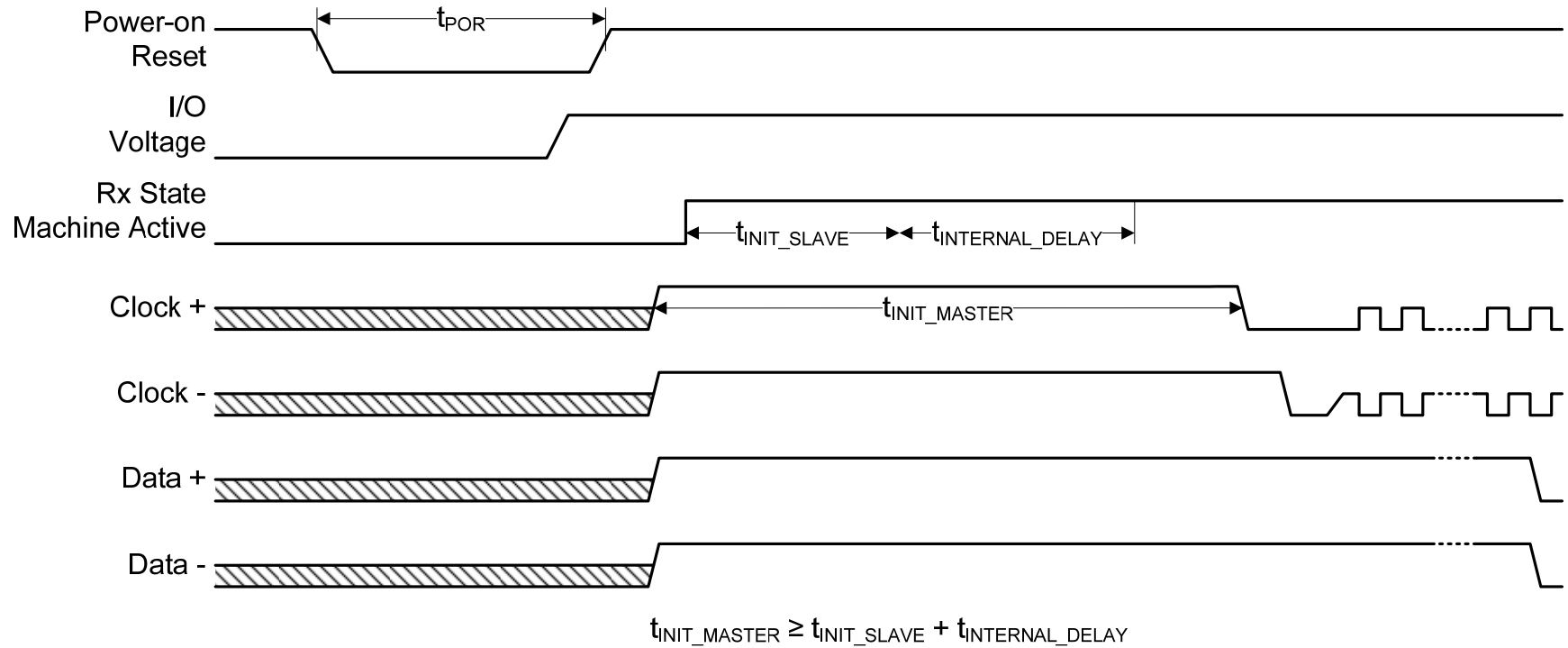
674 Peripherals shall power up in the RX-Stop state and monitor the Link to determine if the host processor has  
675 asserted a TX-Stop state for at least the  $T_{INIT}$  period. The peripheral shall ignore all Link states prior to  
676 detection of a  $T_{INIT}$  event. The peripheral shall be ready to accept bus transactions immediately following  
677 the end of the  $T_{INIT}$  period.

678 Detecting the  $T_{INIT}$  event requires some minimal timing capability on the peripheral. However, accuracy is  
679 not critical as long as a  $T_{INIT}$  event can be reliably detected; an R-C timer with  $\pm 30\%$  accuracy is acceptable  
680 in most cases.

681 If the peripheral requires a longer period after power-up than the  $T_{INIT}$  period driven by the host processor,  
682 this requirement shall be declared in peripheral product information or data sheets. The host processor shall  
683 observe the required additional time after peripheral power-up.


684 Figure 4 illustrates an example power-up sequence for a DSI display module. In the figure, a power-on  
685 reset (POR) mechanism is assumed for initialization. Internally within the display module, de-assertion of  
686 POR could happen after both I/O and core voltages are stable. The worst case  $t_{POR}$  parameter can be defined  
687 by the display module data sheet.  $t_{INIT\_SLAVE}$  represents the minimum initialization period ( $T_{INIT}$ ) defined in  
688 [MIPI04] for a host driving LP-11 to the display. This interval starts immediately after the  $t_{POR}$  period. The  
689 peripheral might need an additional  $t_{INTERNAL\_DELAY}$  time to reach a functional state after power-up. In this  
690 case,  $t_{INTERNAL\_DELAY}$  should also be defined in the display module data sheet. In this example, the host's  
691  $t_{INIT\_MASTER}$  parameter is programmed for driving LP-11 for a period longer than the sum of  $t_{INIT\_SLAVE}$  and  
692  $t_{INTERNAL\_DELAY}$ . The display module ignores all Lane activities during this time.





693

694

 Any drive state except LP-11, LP-10 or LP-01

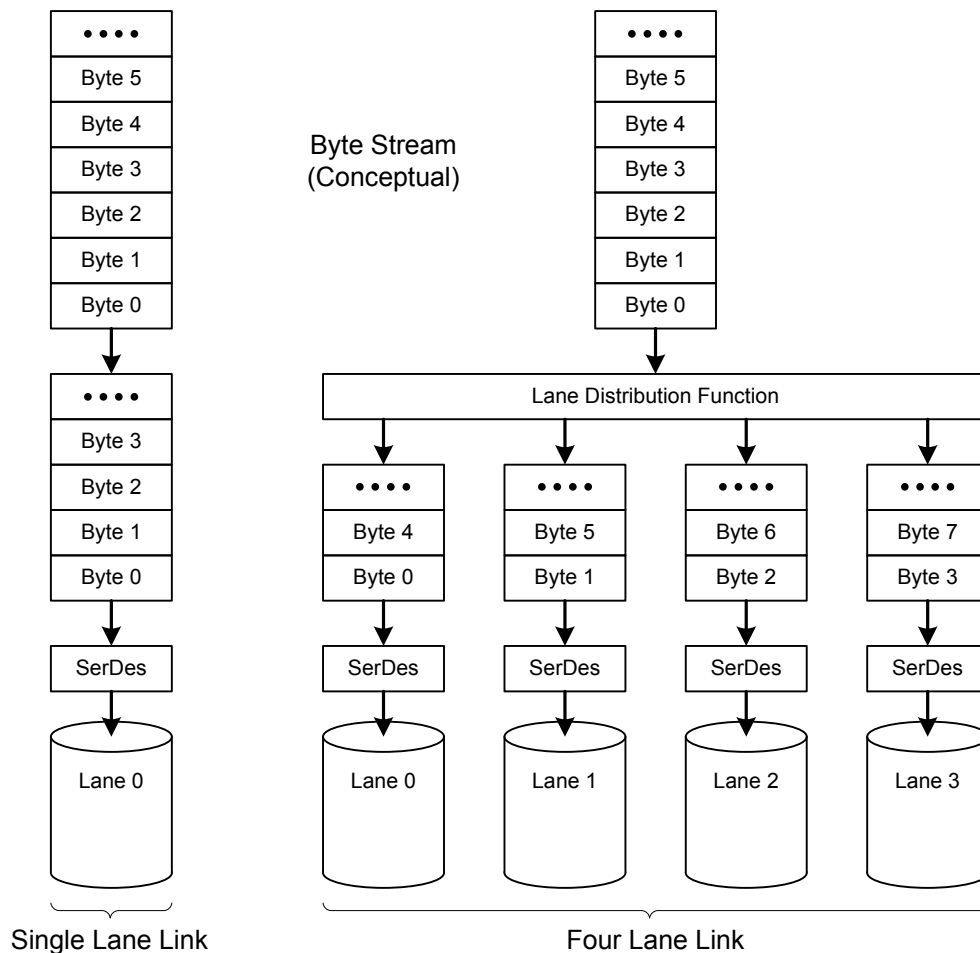
**Figure 4 Peripheral Power-Up Sequencing Example**

## 695 6 Multi-Lane Distribution and Merging

696 DSI is a Lane-scalable interface. Applications requiring more bandwidth than that provided by one Data  
 697 Lane may expand the data path to two, three, or four Lanes wide and obtain approximately linear increases  
 698 in peak bus bandwidth. This document explicitly defines the mapping between application data and the  
 699 serial bit stream to ensure compatibility between host processors and peripherals that make use of multiple  
 700 Lanes.

701 Multi-Lane implementations shall use a single common clock signal, shared by all Data Lanes.

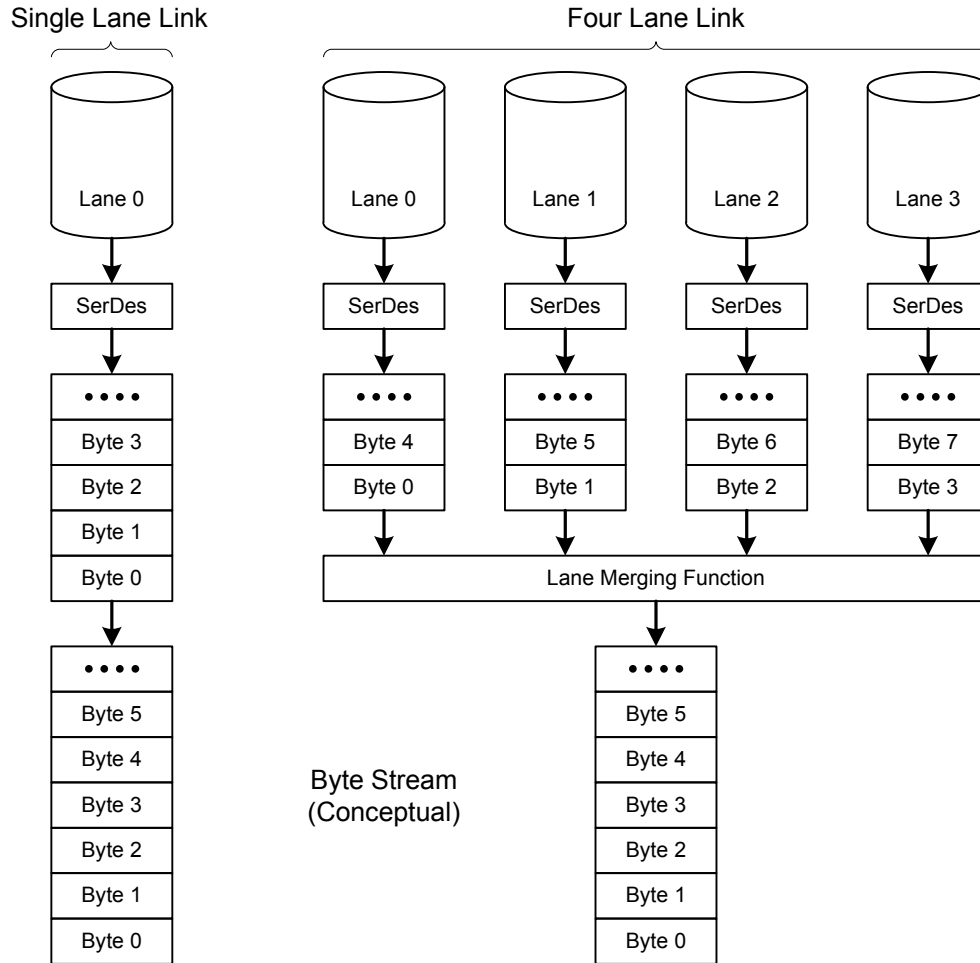
702 Conceptually, between the PHY and higher functional blocks is a layer that enables multi-Lane operation.  
 703 In the transmitter, shown in Figure 5, this layer distributes a sequence of packet bytes across N Lanes,  
 704 where each Lane is an independent block of logic and interface circuitry. In the receiver, shown in Figure 6,  
 705 the layer collects incoming bytes from N Lanes and consolidates the bytes into complete packets to pass  
 706 into the following packet decomposer.



707

708

**Figure 5 Lane Distributor Conceptual Overview**



709

710

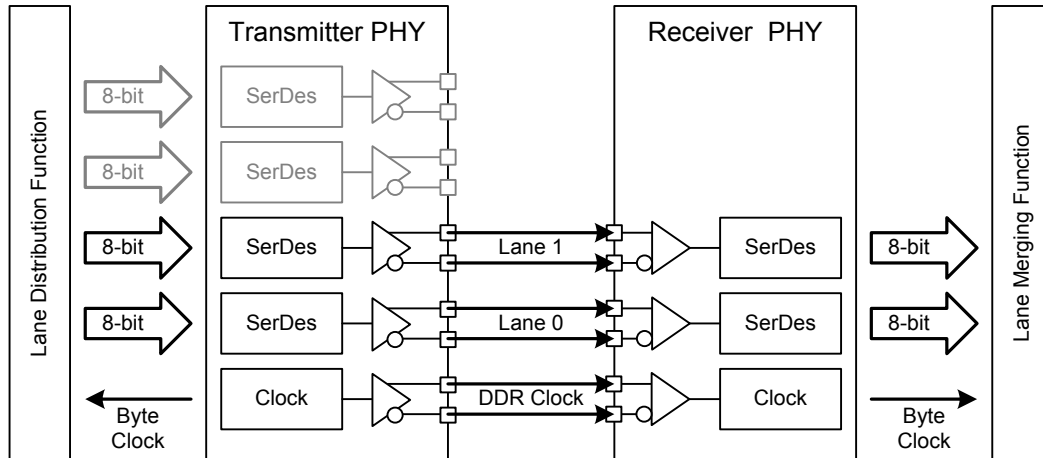
**Figure 6 Lane Merger Conceptual Overview**

711 The Lane Distributor takes a HS transmission of arbitrary byte length, buffers N bytes, where N is the  
 712 number of Lanes implemented in the interface, and sends groups of N bytes in parallel across the N Lanes.  
 713 Before sending data, all Lanes perform the SoT sequence in parallel to indicate to their corresponding  
 714 receiving units that the first byte of a packet is beginning. After SoT, the Lanes send groups of N bytes  
 715 from the first packet in parallel, following a round-robin process. For example, with a two Lane system,  
 716 byte 0 of the packet goes to Lane 0, byte 1 goes to Lane 1, byte 2 to Lane 0, byte 3 to Lane 1 and so on.

### 717 6.1 Multi-Lane Interoperability and Lane-number Mismatch

718 The number of Lanes used shall be a static parameter. It shall be fixed at the time of system design or initial  
 719 configuration and may not change dynamically. Typically, the peripheral's bandwidth requirement and its  
 720 corresponding Lane configuration establishes the number of Lanes used in a system.

721 The host processor shall be configured to support the same number of Lanes required by the peripheral.  
 722 Specifically, a host processor with N-Lane capability ( $N > 1$ ) shall be capable of operation using fewer  
 723 Lanes, to ensure interoperability with peripherals having M Lanes, where  $N > M$ .



724

725

**Figure 7 Four-Lane Transmitter with Two-Lane Receiver Example**

726

### 6.1.1 Clock Considerations with Multi-Lane

727

728

729

730

At EoT, the Protocol layer shall base its control of the common DSI Clock signal on the timing requirements for the last active Lane Module. If the Protocol layer puts the DSI Clock into LPS between HS transmissions to save power, it shall respect the timing requirement for DSI Clock relative to all serial data signals during the EoT sequence.

731

732

Prior to SoT, timing requirements for DSI Clock startup relative to all serial data signals shall similarly be respected.

733

### 6.1.2 Bidirectionality and Multi-Lane Capability

734

735

736

Peripherals typically do not have substantial bandwidth requirements for returning data to the host processor. To keep designs simple and improve interoperability, all DSI-compliant systems shall only use Lane 0 in LP Mode for returning data from a peripheral to the host processor.

737

### 6.1.3 SoT and EoT in Multi-Lane Configurations

738

739

740

741

Since a HS transmission is composed of an arbitrary number of bytes that may not be an integer multiple of the number of Lanes, some Lanes may run out of data before others. Therefore, the Lane Management layer, as it buffers up the final set of less-than-N bytes, de-asserts its “valid data” signal into all Lanes for which there is no further data.

742

743

Although all Lanes start simultaneously with parallel SoTs, each Lane operates independently and may complete the HS transmission before the other Lanes, sending an EoT one cycle (byte) earlier.

744

745

746

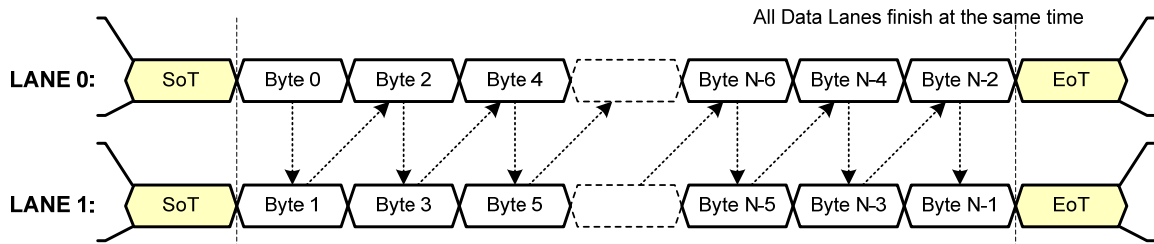
The N PHYs on the receiving end of the Link collect bytes in parallel and feed them into the Lane Management layer. The Lane Management layer reconstructs the original sequence of bytes in the transmission.

747

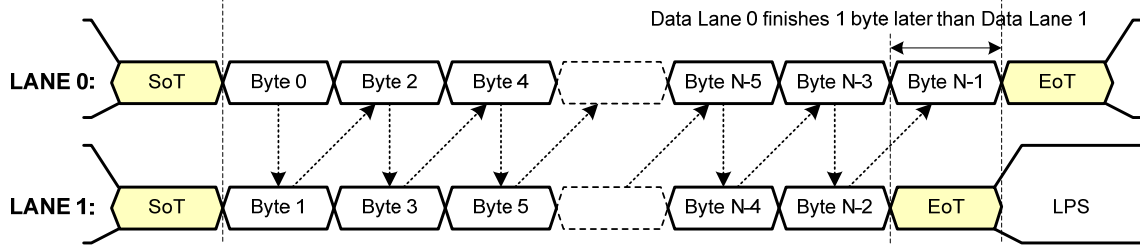
748

Figure 8 and Figure 9 illustrate a variety of ways a HS transmission can terminate for different number of Lanes and packet lengths.

Number of Bytes,  $N$ , transmitted is an integer multiple of the number of lanes:



Number of Bytes,  $N$ , transmitted is NOT an integer multiple of the number of lanes:



**KEY:**

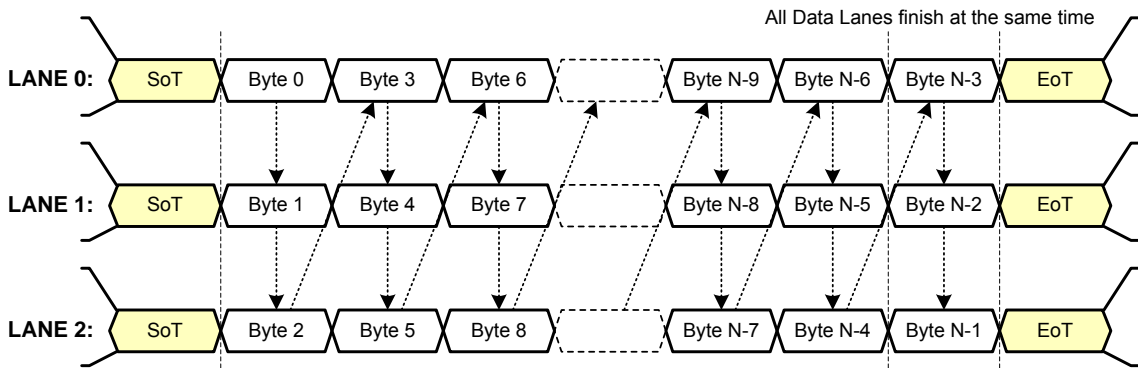
LPS – Low Power State      SoT – Start of Transmission      EoT – End of Transmission

749

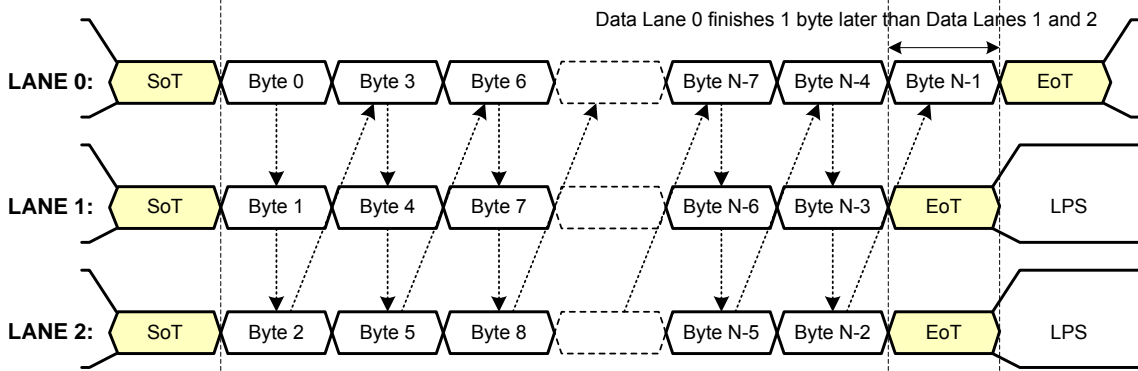
750

**Figure 8 Two Lane HS Transmission Example**

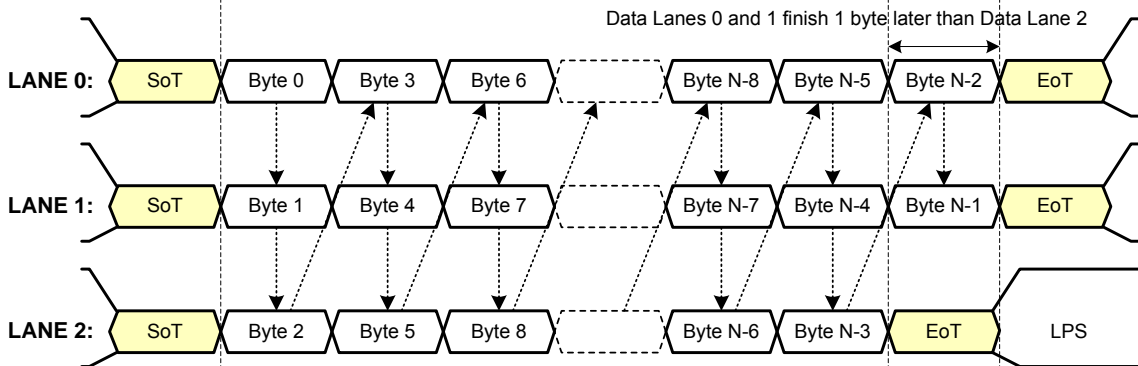
Number of Bytes, N, transmitted is an integer multiple of the number of lanes:



Number of Bytes, N, transmitted is NOT an integer multiple of the number of lanes (Example 1):



Number of Bytes, N, transmitted is NOT an integer multiple of the number of lanes (Example 2):



KEY:

LPS – Low Power State

SoT – Start of Transmission

EoT – End of Transmission

751

752

753

Figure 9 Three Lane HS Transmission Example

## 754 **7 Low-Level Protocol Errors and Contention**

755 For DSI systems there is a possibility that EMI, ESD or other transient-error mechanisms might cause one  
756 end of the Link to go to an erroneous state, or for the Link to transmit corrupted data.

757 In some cases, a transient error in a state machine, or in a clock or data signal, may result in detectable low-  
758 level protocol errors that indicate associated data is, or is likely to be, corrupt. Mechanisms for detecting  
759 and responding to such errors are detailed in the following sections.

760 In other cases, a bidirectional PHY that should be receiving data could begin transmitting while the  
761 authorized transmitter is simultaneously driving the same data line, causing contention and lost data.

762 This section documents the minimum required functionality for recovering from certain low-level protocol  
763 errors and contention. Low-level protocol errors are detected by logic in the PHY, while contention  
764 problems are resolved using contention detectors and timers. Actual contention in DSI-based systems will  
765 be very rare. In most cases, the appropriate use of timers enables recovery from a transient contention  
766 situation.

767 Note that contention-related features are of no benefit for unidirectional DSI Links. However, the “common  
768 mode fault” can still occur in unidirectional systems.

769 The following sections specify the minimum required functionality for detection of low-level protocol  
770 errors, for contention recovery, and associated timers for host processors and peripherals using DSI.

### 771 **7.1 Low-Level Protocol Errors**

772 Logic in the PHY can detect some classes of low-level protocol errors. These errors shall be communicated  
773 to the Protocol layer via the PHY-Protocol Interface. The following errors shall be identified and stored by  
774 the peripheral as status bits for later reporting to the host processor:

- 775 • SoT Error
- 776 • SoT Sync Error
- 777 • EoT Sync Error
- 778 • Escape Mode Entry Command Error
- 779 • LP Transmission Sync Error
- 780 • False Control Error

781 The mechanism for reporting and clearing these error bits is detailed in section 8.10.7. Note that  
782 unidirectional DSI peripherals are exempt from the reporting requirement since they cannot report such  
783 errors to the host processor.

#### 784 **7.1.1 SoT Error**

785 The leader sequence for Start of High-Speed Transmission (SoT) is fault tolerant for any single-bit error  
786 and some multi-bit errors. The received synchronization bits and following data packet might therefore still  
787 be uncorrupted if an error is detected, but confidence in the integrity of payload data is lower. This  
788 condition shall be communicated to the protocol with *SoT Error* flag.

789

**Table 1 Sequence of Events to Resolve SoT Error (HS RX Side)**

PHY	Protocol
Detect SoT Error	
Assert <i>SoT Error</i> flag to protocol	Receive and store <i>SoT Error</i> flag
	Send <i>SoT Error</i> in <i>Acknowledge and Error Report</i> packet, if requested; take no other action based on received HS transmission

790 *SoT Error* is detected by the peripheral PHY. If an acknowledge response is expected, the peripheral shall  
 791 send a response using Data Type 0x02 (*Acknowledge and Error Report*) and set the *SoT Error* bit in the  
 792 return packet to the host processor. The peripheral should take no other action based on the potentially  
 793 corrupted received HS transmission.

### 794 7.1.2 SoT Sync Error

795 If the SoT leader sequence is corrupted in a way that proper synchronization cannot be expected, *SoT Sync*  
 796 *Error* shall be flagged. Subsequent data in the HS transmission is probably corrupt and should not be used.

797

**Table 2 Sequence of Events to Resolve SoT Sync Error (HS RX Side)**

PHY	Protocol
Detect <i>SoT Sync Error</i>	
Assert <i>SoT Sync Error</i> to protocol	Receive and store <i>SoT Sync Error</i> flag
May choose not to pass corrupted data to Protocol layer	Send <i>SoT Sync Error</i> with <i>Acknowledge and Error Report</i> packet if requested; take no other action based on received transmission

798 *SoT Sync Error* is detected by the peripheral PHY. If an acknowledge response is expected, the peripheral  
 799 shall send a response using Data Type 0x02 (*Acknowledge and Error Report*) and set the *SoT Sync Error*  
 800 bit in the return packet to the host processor. Since data is probably corrupted, no command shall be  
 801 interpreted or acted upon in the peripheral. No WRITE activity shall be undertaken in the peripheral.

### 802 7.1.3 EoT Sync Error

803 DSI is a byte-oriented protocol. All uncorrupted HS transmissions contain an integer number of bytes. If,  
 804 during EoT sequence, the peripheral PHY detects that the last byte does not match a byte boundary, *EoT*  
 805 *Sync Error* shall be flagged. If an *Acknowledge* response is expected, the peripheral shall send an  
 806 *Acknowledge and Error Report* packet. The peripheral shall set the *EoT Sync Error* bit in the Error Report  
 807 bytes of the return packet to the host processor.

808 If possible, the peripheral should take no action, especially WRITE activity, in response to the intended  
 809 command. Since this error is not recognized until the end of the packet, some irreversible actions may take  
 810 place before the error is detected.



811

**Table 3 Sequence of Events to Resolve EoT Sync Error (HS RX Side)**

Receiving PHY	Receiving Protocol
Detect EoT Sync Error	
Notify Protocol of <i>EoT Sync Error</i>	Receive and store <i>EoT Sync Error</i> flag
	Ignore HS transmission if possible; assert <i>EoT Sync Error</i> if Acknowledge is requested

812

**7.1.4 Escape Mode Entry Command Error**

813 If the Link begins an Escape Mode sequence, but the Escape Mode Entry command is not recognized by  
 814 the receiving PHY Lane, the receiver shall flag *Escape Mode Entry Command* error. This scenario could be  
 815 a legitimate command, from the transmitter point of view, that's not recognized or understood by the  
 816 receiving protocol. In bidirectional systems, receivers in both ends of the Link shall detect and flag  
 817 unrecognized Escape Mode sequences. Only the peripheral reports this error.

818

**Table 4 Sequence of Events to Resolve Escape Mode Entry Command Error (RX Side)**

Receiving PHY	Receiving Protocol
Detect <i>Escape Mode Entry Command</i> Error	
Notify Protocol of <i>Escape Mode Entry Command</i> Error	Observe <i>Escape Mode Entry Command</i> Error flag
Go to <i>Escape Wait</i> until Stop state is observed	Ignore Escape Mode transmission (if any)
Observe <i>Stop</i> state	
Return to LP-RX Control mode	set Escape Mode Entry Command Error bit

819

**7.1.5 LP Transmission Sync Error**

820 This error flag is asserted if received data is not synchronized to a byte boundary at the end of Low-Power  
 821 Transmission. In bidirectional systems, receivers in both ends of the Link shall detect and flag LP  
 822 Transmission Sync errors. Only the peripheral reports this error.

823

**Table 5 Sequence of Events to Resolve LP Transmission Sync Error (RX Side)**

Receiving PHY	Receiving Protocol
Detect <i>LP Transmission Sync Error</i>	
Notify Protocol of <i>LP Transmission Sync Error</i>	Receive <i>LP Transmission Sync Error</i> flag
Return to <i>LP-RX Control</i> mode until Stop state is observed	Ignore Escape Mode transmission if possible, set appropriate error bit and wait

824

**7.1.6 False Control Error**

825 If a peripheral detects LP-10 (LP request) not followed by the remainder of a valid escape or turnaround  
 826 sequence or if it detects LP-01 (HS request) not followed by a bridge state (LP-00), a False Control Error  
 827 shall be captured in the error status register and reported back to the host after the next BTA. This error  
 828 should be flagged locally to the receiving protocol layer, e.g. when a host detects LP-10 not followed by the  
 829 remainder of a valid escape or turnaround sequence.

830

**Table 6 Sequence of Events to Resolve False Control Error (RX Side)**

Receiving PHY	Receiving Protocol
Detect <i>False Control Error</i>	
Notify Protocol of <i>False Control Error</i>	Observe <i>False Control Error</i> flag, set appropriate error bit and wait
Ignore Turnaround or Escape Mode request	
Remain in <i>LP-RECEIVE STATE Control</i> mode until <i>Stop</i> state is observed	

831

832

**Table 7 Low-Level Protocol Error Detection and Reporting**

Error Detected	HS Unidirectional, LP Unidirectional, no Escape Mode		HS Unidirectional, LP Bidirectional with Escape Mode	
	Host Processor	Peripheral	Host Processor	Peripheral
SoT Error	NA	Detect, no report	NA	Detect and report
SoT Sync Error	NA	Detect, no report	NA	Detect and report
EoT Sync Error	NA	Detect, no report	NA	Detect and report
Escape Mode Entry Command Error	No	No	Detect and flag	Detect and report
LP Transmission Sync Error	No	No	Detect and flag	Detect and report
False Control Error	No	No	Detect and flag	Detect and report

## 833 7.2 Contention Detection and Recovery

834 Contention is a potentially serious problem that, although very rare, could cause the system to hang and  
 835 force a hard reset or power off / on cycle to recover. DSI specifies two mechanisms to minimize this  
 836 problem and enable easier recovery: contention detectors in the PHY for LP Mode contention, and timers  
 837 for other forms of contention and common-mode faults.

### 838 7.2.1 Contention Detection in LP Mode

839 In bidirectional Links, contention detectors in the PHY shall detect two types of contention faults: LP High  
 840 Fault and LP Low Fault. Refer to [MIPI04] for definitions of LP High and LP Low faults. The peripheral  
 841 shall set *Contention Detected* in the Error Report bytes, when it detects either of the contention faults.

842 Annex A provides detailed descriptions and state diagrams for PHY-based detection and recovery  
 843 procedures for LP contention faults. The state diagrams show a sequence of events beginning with  
 844 detection, and ending with return to normal operation.

### 845 7.2.2 Contention Recovery Using Timers

846 The PHY cannot detect all forms of contention. Although they do not directly detect contention, the use of  
 847 appropriate timers ensures that any contention that does happen is of limited duration. The peripheral shall  
 848 set *Peripheral Timeout Error* in the Error Report bytes, when the peripheral detects either HS RX Timer or  
 849 LP TX Timer – Peripheral, defined in section 7.2.2.1, has expired,

850 The time-out mechanisms described in this section are useful for recovering from contention failures,  
851 without forcing the system to undergo a hard reset (power off-on cycle).

### 852 7.2.2.1 Summary of Required Contention Recovery Timers

853 Table 8 specifies the minimum required set of timers for contention recovery in a DSI system.

854 **Table 8 Required Timers and Timeout Summary**

Timer	Timeout	Abbreviation	Requirement
HS RX Timer	HS RX Timeout	HRX_TO	R in bidirectional peripheral
HS TX Timer	HS TX Timeout	HTX_TO	R in host
LP TX Timer – Peripheral	LP_TX-P Timeout	LTX-P_TO	R in bidirectional peripheral
LP RX Timer – Host Processor	LP_RX-H Timeout	LRX-H_TO	R in host

### 855 7.2.2.2 HS RX Timeout (HRX\_TO) in Peripheral

856 This timer is useful for recovering from some transient errors that may result in contention or common-  
857 mode fault. The HRX\_TO timer directly monitors the time a peripheral's HS receiver stays in High-Speed  
858 mode. It is programmed to be longer than the maximum duration of a High-Speed transmission expected by  
859 the peripheral receiver. HS RX timeout will signal an error during HS RX mode if EoT is not received  
860 before the timeout expires.

861 Combined with HTX\_TO, these timers ensure that a transient error will limit contention in HS mode to the  
862 timeout period, and the bus will return to a normal LP state. The Timeout value is protocol specific. HS RX  
863 Timeout shall be used for Bidirectional Links and for Unidirectional Links with Escape Mode. HS RX  
864 Timeout is recommended for all DSI peripherals and required for all bidirectional DSI peripherals.

865 **Table 9 Sequence of Events for HS RX Timeout (Peripheral initially HS RX)**

Host Processor Side	Peripheral Side
Drives bus HS-TX	HS RX Timeout Timer Expires
	Transition to LP-RX
End HS transmission normally, or HS-TX timeout	Peripheral waits for <i>Stop</i> state before responding to bus activity.
Transition to <i>Stop</i> state (LP-11)	Observe <i>Stop</i> state and flag error

866 During this mode, the HS clock is active and can be used for the HS RX Timer in the peripheral.

867 The LP High Fault and LP Low Fault are caused by both sides of the Link transmitting simultaneously.  
868 Note, the LP High Fault and LP Low Fault are only applicable for bidirectional Data Lanes.

869 The Common Mode fault occurs when the transmitter and receiver are not in the same communication  
870 mode, e.g. transmitter (host processor) is driving LP-01 or LP-10, while the receiver (peripheral) is in HS-  
871 RX mode with terminator connected. There is no contention, but the receiver will not capture transmitted  
872 data correctly. This fault may occur in both bidirectional and unidirectional lanes. After HS RX timeout,  
873 the peripheral returns to LP-RX mode and normal operation may resume. Note that in the case of a  
874 common-mode fault, there may be no DSI serial clock from the host processor. Therefore, another clock  
875 source for HRX\_TO timer may be required.

876 **7.2.2.3 HS TX Timeout (HTX\_TO) in Host Processor**

877 This timer is used to monitor a host processor's own length of HS transmission. It is programmed to be  
 878 longer than the expected maximum duration of a High-Speed transmission. The maximum HS transmission  
 879 length is protocol-specific. If the timer expires, the processor forces a clean termination of HS transmission  
 880 and enters EoT sequence, then drives LP-11 state. This timeout is required for all host processors.

881 **Table 10 Sequence of Events for HS TX Timeout (Host Processor initially HS TX)**

Host Processor Side	Peripheral Side
Host Processor in HS TX mode	Peripheral in HS RX mode
HS TX Timeout Timer expires, forces EoT	
Host Processor drives <i>Stop</i> state (LP-11)	Peripheral observes EoT and <i>Stop</i> state (LP-RX)

882 Note that the peripheral HS-RX timeout (see section 7.2.2.2) should be set to a value shorter than the host  
 883 processor's HS-TX timer so that the peripheral has returned to LP-RX state and is ready for further  
 884 commands following receipt of LP-11 from the host processor.

885 **7.2.2.4 LP TX-Peripheral Timeout (LTX-P\_TO)**

886 This timer is used to monitor the peripheral's own length of LP transmission (bus possession time) when in  
 887 LP TX mode. The maximum transmission length in LP TX is determined by protocol and data formats.  
 888 This timeout is useful for recovering from LP-contention. LP TX-Peripheral Timeout is required for  
 889 bidirectional peripherals.

890 **Table 11 Sequence of Events for LP TX-Peripheral Timeout (Peripheral initially LP TX)**

Host Processor Side	Peripheral Side
(possible contention)	Peripheral in LP TX mode
	LP TX-P Timeout Timer Expires
	Transition to LP-RX
Detect contention, or Host LP-RX Timeout	Peripheral waits for <i>Stop</i> state before responding to bus activity.
Drive LP-11 <i>Stop</i> state	Observe <i>Stop</i> state in LP-RX mode

891 Note that host processor LP-RX timeout (see section 7.2.2.5) should be set to a *longer* value than the  
 892 peripheral's LP-TX-P timer, so that the peripheral has returned to LP-RX state and is ready for further  
 893 commands following receipt of LP-11 from the host processor.

894 **7.2.2.5 LP-RX Host Processor Timeout (LRX-H\_TO)**

895 The LP-RX timeout period in the Host Processor shall be greater than the LP TX-Peripheral timeout. Since  
 896 both timers begin counting at approximately the same time, this ensures the peripheral has returned to LP-  
 897 RX mode and is waiting for bus activity (commands from Host Processor, etc.) when LP-RX timer expires  
 898 in the host. The timeout value is protocol specific. This timer is required for all Host Processors.

899 **Table 12 Sequence of Events for Host Processor Wait Timeout (Peripheral initially TX)**

Host Processor Side	Peripheral Side
Host Processor in LP RX mode	(peripheral LP-TX timeout)

Host Processor Side	Peripheral Side
Host Processor LP-RX Timer expires	Peripheral waiting in LP-RX mode
Host Processor drives <i>Stop</i> state (LP-11)	Peripheral observes <i>Stop</i> state in LP-RX mode

### 900 7.3 Additional Timers

901 Additional timers are used to detect bus turnaround problems and to ensure sufficient wait time after a  
902 RESET Trigger Message is sent to the peripheral.

#### 903 7.3.1 Turnaround Acknowledge Timeout (TA\_TO)

904 When either end of the Link issues BTA (Bus Turn-Around), its PHY shall monitor the sequence of Data  
905 Lane states during the ensuing turnaround process. In a normal BTA sequence, the turnaround completes  
906 within a bounded time, with the other end of the Link finally taking bus possession and driving LP-11 (*Stop*  
907 *state*) on the bus. If the sequence is observed not to complete (by the previously-transmitting PHY) within  
908 the specified time period, the timer TA\_TO expires. The side of the Link that issued the BTA then begins a  
909 recovery procedure, or re-sends BTA. The specified period shall be longer than the maximum possible  
910 turnaround delay for the unit to which the turnaround request was sent. TA\_TO is an optional timer.

911 **Table 13 Sequence of Events for BTA Acknowledge Timeout (Peripheral initially TX)**

Host Processor Side	Peripheral Side
Host in LP RX mode	Peripheral in LP TX mode
	Send Turnaround back to Host
(no change)	Turnaround Acknowledgement Timeout
	Transition to LP-RX

912 **Table 14 Sequence of Events for BTA Acknowledge Timeout (Host Processor initially TX)**

Host Processor Side	Peripheral Side
Host Processor in HS TX or LP TX mode	Peripheral in LP RX mode
Request Turnaround	
Turnaround Acknowledgement Timeout	(no change)
Return to <i>Stop</i> state (LP-11)	

#### 913 7.3.2 Peripheral Reset Timeout (PR\_TO)

914 When a peripheral is reset, it requires a period of time before it is ready for normal operation. This timer is  
915 programmed with a value longer than the specified time required to complete the reset sequence. After it  
916 expires, the host may resume normal operation with the peripheral. The timeout value is peripheral-  
917 specific. This is an optional timer.

918 **Table 15 Sequence of Events for Peripheral Reset Timeout**

Host Processor Side	Peripheral Side
Send <i>Reset Entry</i> command	Receive <i>Reset Entry</i> Command
Return to <i>Stop</i> state (LP-11)	Initiate reset sequence
	Complete reset sequence

Host Processor Side	Peripheral Side
Peripheral Reset Timeout	
Resume Normal Operation.	Wait for bus activity

#### 919 **7.4 Acknowledge and Error Reporting Mechanism**

920 In a bidirectional Link, the peripheral monitors transmissions from the host processor using detection  
 921 features and timers specified in this section. Error information related to the transmission shall be stored in  
 922 the peripheral. Errors from multiple transmissions shall be stored and accumulated until a BTA following a  
 923 transmission provides the opportunity for the peripheral to report errors to the host processor.

924 The host processor may request a command acknowledge and error information related to any transmission  
 925 by asserting Bus Turnaround with the transmission. The peripheral shall respond with ACK Trigger  
 926 Message if there are no errors and with *Acknowledge and Error Report* packet if any errors were detected  
 927 in previous transmissions. Appropriate flags shall be set to indicate what errors were detected on the  
 928 preceding transmissions. If the transmission was a Read request, the peripheral shall return READ data  
 929 without issuing additional ACK Trigger Message or an *Acknowledge and Error Report* packet if no errors  
 930 were detected. If there was an error in the Read request, the peripheral shall return the appropriate  
 931 *Acknowledge and Error Report* unless the error was a single-bit correctable error. In that case, the  
 932 peripheral shall return the requested READ data packet followed by *Acknowledge and Error Report* packet  
 933 with appropriate error bits set.

934 Once errors are reported, the Error Register shall have all bits set to zero.

935 See section 8.10.1 for more detail on *Acknowledge and Error Report* protocols.

936

## 937 **8 DSI Protocol**

938 On the transmitter side of a DSI Link, parallel data, signal events, and commands are converted in the  
939 Protocol layer to packets, following the packet organization documented in this section. The Protocol layer  
940 appends packet-protocol information and headers, and then sends complete bytes through the Lane  
941 Management layer to the PHY. Packets are serialized by the PHY and sent across the serial Link. The  
942 receiver side of a DSI Link performs the converse of the transmitter side, decomposing the packet into  
943 parallel data, signal events and commands.

944 If there are multiple Lanes, the Lane Management layer distributes bytes to separate PHYs, one PHY per  
945 Lane, as described in Section 6. Packet protocol and formats are independent of the number of Lanes used.

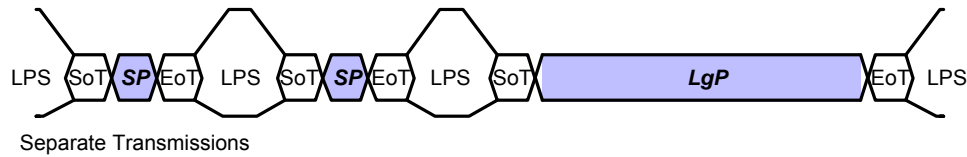
### 946 **8.1 Multiple Packets per Transmission**

947 In its simplest form, a transmission may contain one packet. If many packets are to be transmitted, the  
948 overhead of frequent switching between LPS and High-Speed Mode will severely limit bandwidth if  
949 packets are sent separately, e.g. one packet per transmission.

950 The DSI protocol permits multiple packets to be concatenated, which substantially boosts effective  
951 bandwidth. This is useful for events such as peripheral initialization, where many registers may be loaded  
952 with separate write commands at system startup.

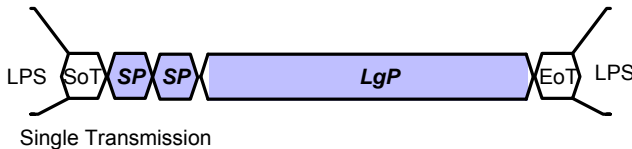
953 There are two modes of data transmission, HS and LP transmission modes, at the PHY layer. Before a HS  
954 transmission can be started, the transmitter PHY issues a SoT sequence to the receiver. After that, data or  
955 command packets can be transmitted in HS mode. Multiple packets may exist within a single HS  
956 transmission and the end of transmission is always signaled at the PHY layer using a dedicated EoT  
957 sequence. In order to enhance the overall robustness of the system, DSI defines a dedicated EoT packet  
958 (EoTp) at the protocol layer (section 8.8.2) for signaling the end of HS transmission. For backwards  
959 compatibility with earlier DSI systems, the capability of generating and interpreting this EoTp can be  
960 enabled or disabled. The method of enabling or disabling this capability is out of scope for this document.  
961 PHY-based EoT and SoT sequences are defined in [MIPI04].

962 The top diagram in Figure 10 illustrates a case where multiple packets are being sent separately with EoTp  
963 support disabled. In HS mode, time gaps between packets shall result in separate HS transmissions for each  
964 packet, with a SoT, LPS, and EoT issued by the PHY layer between packets. This constraint does not apply  
965 to LP transmissions. The bottom diagram in Figure 10 demonstrates a case where multiple packets are  
966 concatenated within a single HS transmission.



**KEY:**

- LPS – Low Power State
- SoT – Start of Transmission
- EoT – End of Transmission
- SP – Short Packet
- LgP – Long Packet



967

968

**Figure 10 HS Transmission Examples with EoTp disabled**

969

Figure 11 depicts HS transmission cases where EoTp generation is enabled. In the figure, EoT short packets are highlighted in red. The top diagram illustrates a case where a host is intending to send a short packet followed by a long packet using two separate transmissions. In this case, an additional EoT short packet is generated before each transmission ends. This mechanism provides a more robust environment, at the expense of increased overhead (four extra bytes per transmission) compared to cases where EoTp generation is disabled, i.e. the system only relies on the PHY layer EoT sequence for signaling the end of HS transmission. The overhead imposed by enabling EoTp can be minimized by sending multiple long and short packets within a single transmission as illustrated by the bottom diagram in Figure 11.

970

971

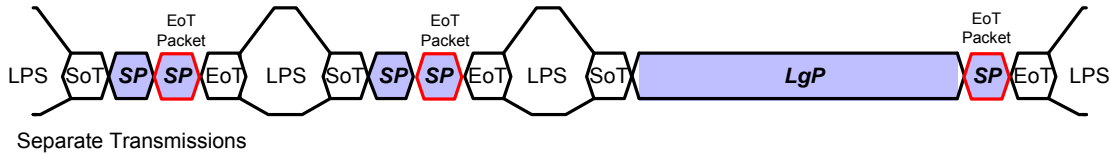
972

973

974

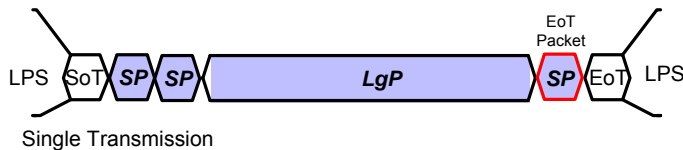
975

976



**KEY:**

- LPS – Low Power State
- SoT – Start of Transmission
- EoT – End of Transmission
- SP – Short Packet
- LgP – Long Packet



977

978

**Figure 11 HS Transmission Examples with EoTp enabled**

**8.2 Packet Composition**

980

The first byte of the packet, the Data Identifier (DI), includes information specifying the type of the packet. For example, in Video Mode systems in a display application the logical unit for a packet may be one horizontal display line. Command Mode systems send commands and an associated set of parameters, with the number of parameters depending on the command type.

981

982

983



984 Packet sizes fall into two categories:

- 985 • **Short packets** are four bytes in length including the ECC. Short packets are used for most  
986 Command Mode commands and associated parameters. Other Short packets convey events like H  
987 Sync and V Sync edges. Because they are Short packets they can convey accurate timing  
988 information to logic at the peripheral.
- 989 • **Long packets** specify the payload length using a two-byte Word Count field. Payloads may be  
990 from 0 to  $2^{16} - 1$  bytes long. Therefore, a Long packet may be up to 65,541 bytes in length. Long  
991 packets permit transmission of large blocks of pixel or other data.

992 A special case of Command Mode operation is video-rate (update) streaming, which takes the form of an  
993 arbitrarily long stream of pixel or other data transmitted to the peripheral. As all DSI transactions use  
994 packets, the video stream shall be broken into separate packets. This “packetization” may be done by  
995 hardware or software. The peripheral may then reassemble the packets into a continuous video stream for  
996 display.

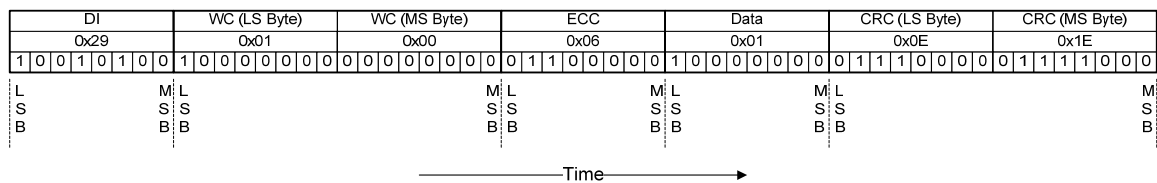
997 The *Set Maximum Return Packet Size* command allows the host processor to limit the size of response  
998 packets coming from a peripheral. See section 8.8.10 for a description of the command.

### 999 8.3 Endian Policy

1000 All packet data traverses the interface as bytes. Sequentially, a transmitter shall send data LSB first, MSB  
1001 last. For packets with multi-byte fields, the least significant byte shall be transmitted first unless otherwise  
1002 specified.

1003 Figure 12 shows a complete Long packet data transmission. Note, the figure shows the byte values in  
1004 standard positional notation, i.e. MSB on the left and LSB on the right, while the bits are shown in  
1005 chronological order with the LSB on the left, the MSB on the right and time increasing left to right.

1006 See section 8.4.1 for a description of the Long packet format.



1007  
1008

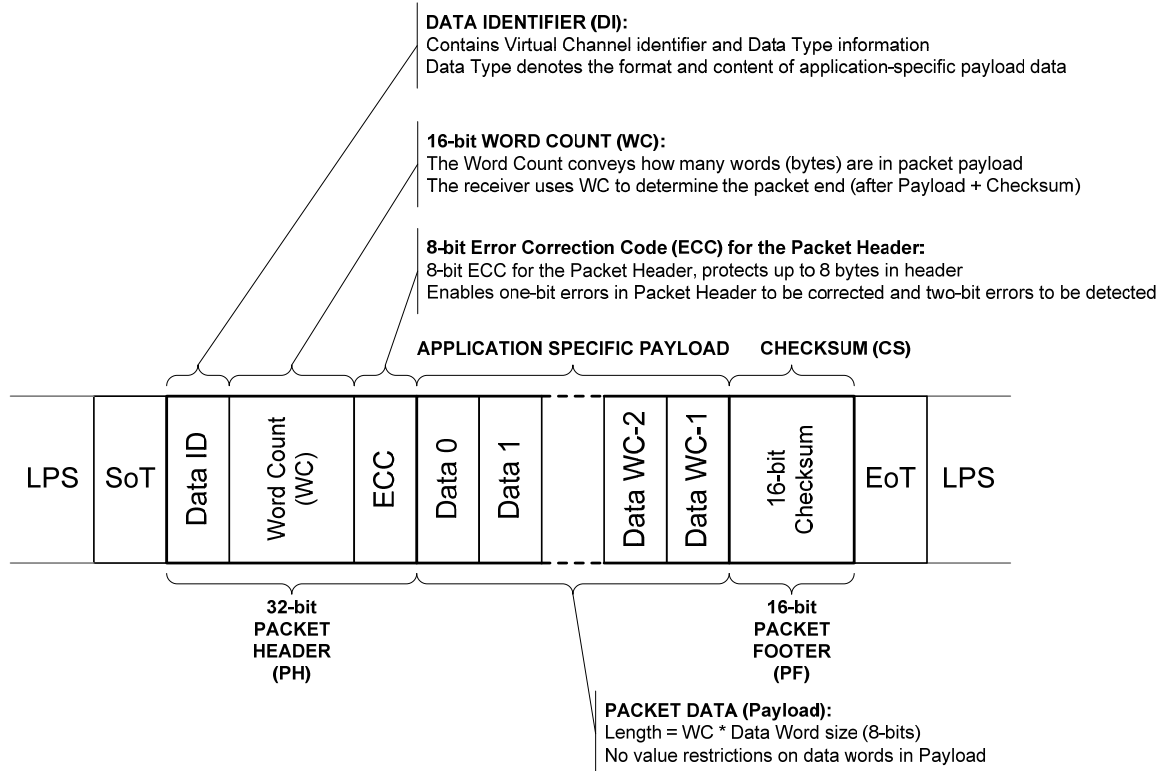
1009 **Figure 12 Endian Example (Long Packet)**

### 1010 8.4 General Packet Structure

1011 Two packet structures are defined for low-level protocol communication: Long packets and Short packets.  
1012 For both packet structures, the Data Identifier is always the first byte of the packet.

#### 1013 8.4.1 Long Packet Format

1014 Figure 13 shows the structure of the Long packet. A Long packet shall consist of three elements: a 32-bit  
1015 Packet Header (PH), an application-specific Data Payload with a variable number of bytes, and a 16-bit  
1016 Packet Footer (PF). The Packet Header is further composed of three elements: an 8-bit Data Identifier, a  
1017 16-bit Word Count, and 8-bit ECC. The Packet Footer has one element, a 16-bit checksum. Long packets  
1018 can be from 6 to 65,541 bytes in length.



1019  
1020

1021

**Figure 13 Long Packet Structure**

1022 The Data Identifier defines the Virtual Channel for the data and the Data Type for the application specific  
1023 payload data. See sections 8.8 through 8.10 for descriptions of Data Types.

1024 The Word Count defines the number of bytes in the Data Payload between the end of the Packet Header  
1025 and the start of the Packet Footer. Neither the Packet Header nor the Packet Footer shall be included in the  
1026 Word Count.

1027 The Error Correction Code (ECC) byte allows single-bit errors to be corrected and 2-bit errors to be  
1028 detected in the Packet Header. This includes both the Data Identifier and Word Count fields.

1029 After the end of the Packet Header, the receiver reads the next Word Count \* bytes of the Data Payload.  
1030 Within the Data Payload block, there are no limitations on the value of a data word, i.e. no embedded codes  
1031 are used.

1032 Once the receiver has read the Data Payload it reads the Checksum in the Packet Footer. The host processor  
1033 shall always calculate and transmit a Checksum in the Packet Footer. Peripherals are not required to  
1034 calculate a Checksum. Also note the special case of zero-byte Data Payload: if the payload has length 0,  
1035 then the Checksum calculation results in (0xFFFF). If the Checksum is not calculated, the Packet Footer  
1036 shall consist of two bytes of all zeros (0x0000). See section 9 for more information on calculating the  
1037 Checksum.

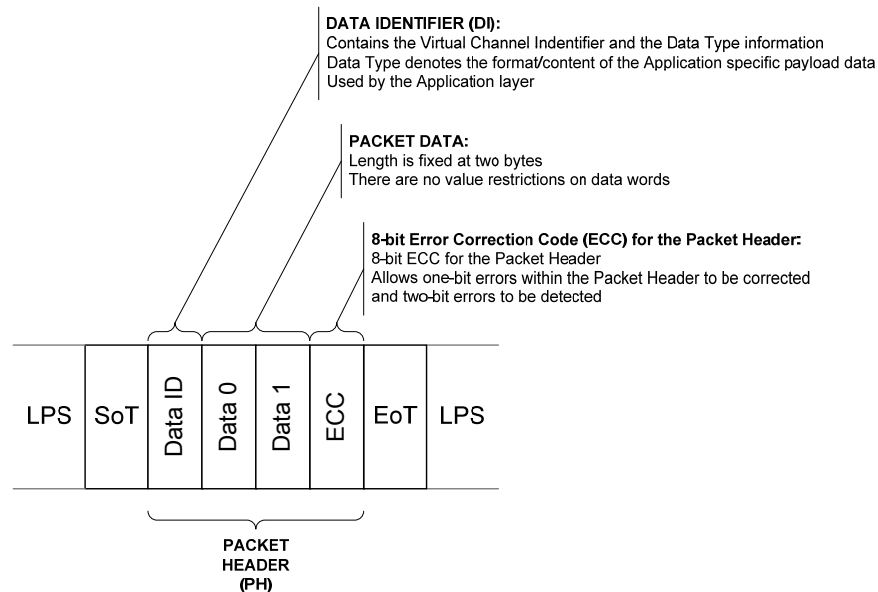
1038 In the generic case, the length of the Data Payload shall be a multiple of bytes. In addition, each data format  
1039 may impose additional restrictions on the length of the payload data, e.g. multiple of four bytes.

1040 Each byte shall be transmitted least significant bit first. Payload data may be transmitted in any byte order  
 1041 restricted only by data format requirements. Multi-byte elements such as Word Count and Checksum shall  
 1042 be transmitted least significant byte first.

#### 1043 **8.4.2 Short Packet Format**

1044 Figure 14 shows the structure of the Short packet. See sections 8.8 through 8.10 for descriptions of the Data  
 1045 Types. A Short packet shall contain an 8-bit Data ID followed by two command or data bytes and an 8-bit  
 1046 ECC; a Packet Footer shall not be present. Short packets shall be four bytes in length.

1047 The Error Correction Code (ECC) byte allows single-bit errors to be corrected and 2-bit errors to be  
 1048 detected in the Short packet.



1049

1050

**Figure 14 Short Packet Structure**

#### 1051 **8.5 Common Packet Elements**

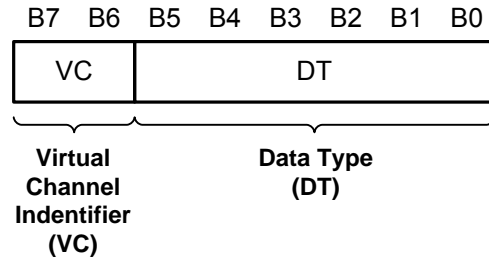
1052 Long and Short packets have several common elements that are described in this section.

##### 1053 **8.5.1 Data Identifier Byte**

1054 The first byte of any packet is the DI (Data Identifier) byte. Figure 15 shows the composition of the Data  
 1055 Identifier (DI) byte.

1056 DI[7:6]: These two bits identify the data as directed to one of four virtual channels.

1057 DI[5:0]: These six bits specify the Data Type.



1058

1059

**Figure 15 Data Identifier Byte**

1060

### 8.5.1.1 Virtual Channel Identifier – VC field, DI[7:6]

1061

A processor may service up to four peripherals with tagged commands or blocks of data, using the Virtual Channel ID field of the header for packets targeted at different peripherals.

1062

1063

The Virtual Channel ID enables one serial stream to service two or more virtual peripherals by multiplexing packets onto a common transmission channel. Note that packets sent in a single transmission each have their own Virtual Channel assignment and can be directed to different peripherals. Although the DSI protocol permits communication with multiple peripherals, this specification only addresses the connection of a host processor to a single peripheral. Implementation details for connection to more than one physical peripheral are beyond the scope of this document.

1064

1065

1066

1067

1068

1069

### 8.5.1.2 Data Type Field DT[5:0]

1070

The Data Type field specifies if the packet is a Long or Short packet type and the packet format. The Data Type field, along with the Word Count field for Long packets, informs the receiver of how many bytes to expect in the remainder of the packet. This is necessary because there are no special packet start / end sync codes to indicate the beginning and end of a packet. This permits packets to convey arbitrary data, but it also requires the packet header to explicitly specify the size of the packet.

1071

1072

1073

1074

1075

When the receiving logic has counted down to the end of a packet, it shall assume the next data is either the header of a new packet or the EoT (End of Transmission) sequence.

1076

1077

## 8.5.2 Error Correction Code

1078

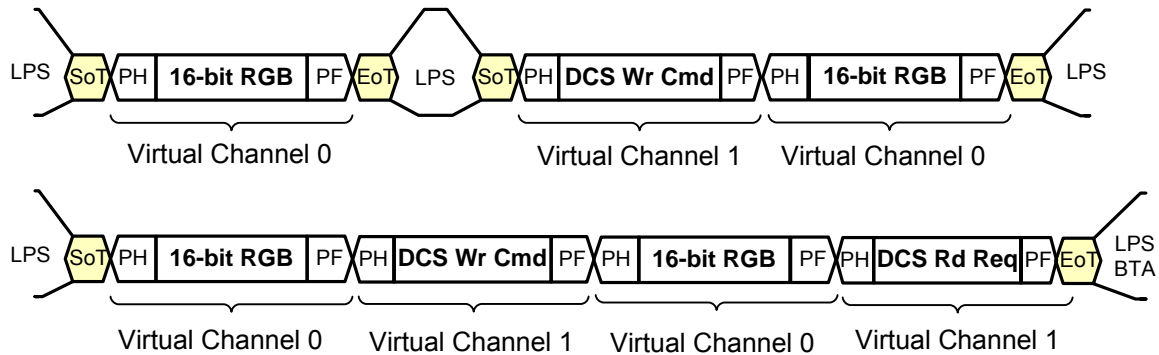
The Error Correction Code allows single-bit errors to be corrected and 2-bit errors to be detected in the Packet Header. The host processor shall always calculate and transmit an ECC byte. Peripherals shall support ECC in both forward- and reverse-direction communications. See section 9 for more information on coding and decoding the ECC and section 8.9.2 for ECC and Checksum requirements.

1079

1080

1081

1082 **8.6 Interleaved Data Streams**



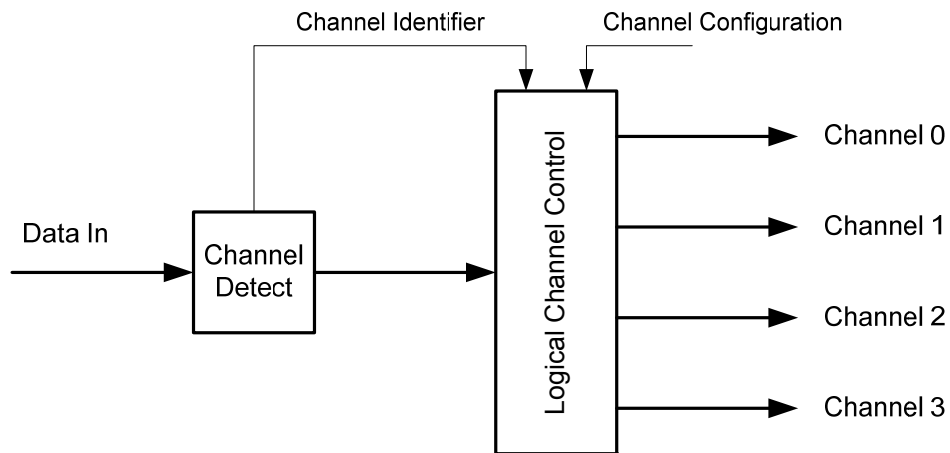
**KEY:**

- LPS – Low Power State
- SoT – Start of Transmission
- EoT – End of Transmission
- PH – Packet Header
- PF – Packet Footer
- BTA – Bus Turn-Around

1083  
1084

**Figure 16 Interleaved Data Stream Example with EoTp disabled**

1085 One application for multiple channels is a high-resolution display using two or more separate driver ICs on  
 1086 a single display module. Each driver IC addresses only a portion of the columns on the display device.  
 1087 Each driver IC captures and displays only the packet contents targeted for that driver and ignores the other  
 1088 packets. See Figure 17.



1089  
1090

**Figure 17 Logical Channel Block Diagram (Receiver Case)**

1091 **8.6.1 Interleaved Data Streams and Bidirectionality**

1092 When multiple peripherals have bidirectional capability there shall be a clear and unambiguous means for  
 1093 returning READ data, events and status back to the host processor from the intended peripheral. The  
 1094 combination of BTA and the Virtual Channel ID ensures no confusion over which peripheral is expected to  
 1095 respond to any request from the peripheral. Returning packets shall be tagged with the ID of the peripheral  
 1096 that sent the packet.

1097 A consequence of bidirectionality is any transmission from the host processor shall contain no more than  
 1098 one packet requiring a peripheral response. This applies regardless of the number of peripherals that may be  
 1099 connected via the Link to the host processor.

## 1100 8.7 Processor to Peripheral Direction (Processor-Sourced) Packet Data Types

1101 The set of transaction types sent from the host processor to a peripheral, such as a display module, are  
 1102 shown in Table 16.

1103 **Table 16 Data Types for Processor-sourced Packets**

Data Type, hex	Data Type, binary	Description	Packet Size
0x01	00 0001	Sync Event, V Sync Start	Short
0x11	01 0001	Sync Event, V Sync End	Short
0x21	10 0001	Sync Event, H Sync Start	Short
0x31	11 0001	Sync Event, H Sync End	Short
0x08	00 1000	End of Transmission packet (EoTp)	Short
0x02	00 0010	Color Mode (CM) Off Command	Short
0x12	01 0010	Color Mode (CM) On Command	Short
0x22	10 0010	Shut Down Peripheral Command	Short
0x32	11 0010	Turn On Peripheral Command	Short
0x03	00 0011	Generic Short WRITE, no parameters	Short
0x13	01 0011	Generic Short WRITE, 1 parameter	Short
0x23	10 0011	Generic Short WRITE, 2 parameters	Short
0x04	00 0100	Generic READ, no parameters	Short
0x14	01 0100	Generic READ, 1 parameter	Short
0x24	10 0100	Generic READ, 2 parameters	Short
0x05	00 0101	DCS Short WRITE, no parameters	Short
0x15	01 0101	DCS Short WRITE, 1 parameter	Short
0x06	00 0110	DCS READ, no parameters	Short
0x37	11 0111	Set Maximum Return Packet Size	Short
0x09	00 1001	Null Packet, no data	Long
0x19	01 1001	Blanking Packet, no data	Long
0x29	10 1001	Generic Long Write	Long
0x39	11 1001	DCS Long Write/write_LUT Command Packet	Long
0x0C	00 1100	Loosely Packed Pixel Stream, 20-bit YCbCr, 4:2:2 Format	Long
0x1C	01 1100	Packed Pixel Stream, 24-bit YCbCr, 4:2:2 Format	Long
0x2C	10 1100	Packed Pixel Stream, 16-bit YCbCr, 4:2:2 Format	Long
0x0D	00 1101	Packed Pixel Stream, 30-bit RGB, 10-10-10 Format	Long
0x1D	01 1101	Packed Pixel Stream, 36-bit RGB, 12-12-12 Format	Long

Data Type, hex	Data Type, binary	Description	Packet Size
0x3D	11 1101	Packed Pixel Stream, 12-bit YCbCr, 4:2:0 Format	Long
0x0E	00 1110	Packed Pixel Stream, 16-bit RGB, 5-6-5 Format	Long
0x1E	01 1110	Packed Pixel Stream, 18-bit RGB, 6-6-6 Format	Long
0x2E	10 1110	Loosely Packed Pixel Stream, 18-bit RGB, 6-6-6 Format	Long
0x3E	11 1110	Packed Pixel Stream, 24-bit RGB, 8-8-8 Format	Long
0xX0 and 0xFF, unspecified	XX 0000 XX 1111	DO NOT USE All unspecified codes are reserved	

## 1104 8.8 Processor-to-Peripheral Transactions – Detailed Format Description

### 1105 8.8.1 Sync Event (H Start, H End, V Start, V End), Data Type = XX 0001 (0xX1)

1106 Sync Events are Short packets and, therefore, can time-accurately represent events like the start and end of  
 1107 sync pulses. As “start” and “end” are separate and distinct events, the length of sync pulses, as well as  
 1108 position relative to active pixel data, e.g. front and back porch display timing, may be accurately conveyed  
 1109 to the peripheral. The Sync Events are defined as follows:

- 1110 • Data Type = 00 0001 (0x01) V Sync Start
- 1111 • Data Type = 01 0001 (0x11) V Sync End
- 1112 • Data Type = 10 0001 (0x21) H Sync Start
- 1113 • Data Type = 11 0001 (0x31) H Sync End

1114 In order to represent timing information as accurately as possible a V Sync Start event represents the start  
 1115 of the VSA and also implies an H Sync Start event for the first line of the VSA. Similarly, a V Sync End  
 1116 event implies an H Sync Start event for the last line of the VSA. If the host processor sources interlaced  
 1117 video, horizontal sync timing follows standard interlaced video conventions for the video format being used  
 1118 and are beyond the scope of this specification. See [CEA01] for timing details of interlaced video formats.  
 1119 The first field of interlaced video follows the same rules to imply H Sync Start. The peripheral (display),  
 1120 when receiving the interlaced second video field, shall not imply an H Sync Start at the V Sync Start and V  
 1121 Sync End timing. Refer to Annex C for a detailed progression order of event packets for progressive scan  
 1122 and interlaced scan video timing.

1123 Sync events should occur in pairs, Sync Start and Sync End, if accurate pulse-length information needs to  
 1124 be conveyed. Alternatively, if only a single point (event) in time is required, a single sync event (normally,  
 1125 Sync Start) may be transmitted to the peripheral. Sync events may be concatenated with blanking packets to  
 1126 convey inter-line timing accurately and avoid the overhead of switching between LPS and HS for every  
 1127 event. Note there is a power penalty for keeping the data line in HS mode, however.

1128 Display modules that do not need traditional sync/blanking/pixel timing should transmit pixel data in a  
 1129 high-speed burst then put the bus in Low Power Mode, for reduced power consumption. The recommended  
 1130 burst size is a scan line of pixels, which may be temporarily stored in a line buffer on the display module.

### 1131 8.8.2 EoTp, Data Type = 00 1000 (0x08)

1132 This short packet is used for indicating the end of a HS transmission to the data link layer. As a result,  
 1133 detection of the end of HS transmission may be decoupled from physical layer characteristics. [MIPI04]

1134 defines an EoT sequence composed of a series of all 1's or 0's depending on the last bit of the last packet  
 1135 within a HS transmission. Due to potential errors, the EoT sequence could be interpreted incorrectly as  
 1136 valid data types. Although EoT errors are not expected to happen frequently, the addition of this packet will  
 1137 enhance overall system reliability.

1138 Devices compliant to earlier revisions of the DSI specification do not support EoTp generation or detection.  
 1139 A Host or peripheral device compliant to this revision of DSI specification shall incorporate capability of  
 1140 supporting EoTp. The device shall also provide an implementation-specific means for enabling and  
 1141 disabling this capability to ensure interoperability with earlier DSI devices that do not support the EoTp.

1142 The main objective of the EoTp is to enhance overall robustness of the system during HS transmission  
 1143 mode. Therefore, DSI transmitters should not generate an EoTp when transmitting in LP mode. The Data  
 1144 Link layer of DSI receivers shall detect and interpret arriving EoTps regardless of transmission mode (HS  
 1145 or LP modes) in order to decouple itself from the physical layer. Table 17 describes how DSI mandates  
 1146 EoTp support for different transmission and reception modes.

1147 **Table 17 EoT Support for Host and Peripheral**

DSI Host (EoT capability enabled)				DSI Peripheral (EoT capability enabled)			
HS Mode		LP Mode		HS Mode		LP Mode	
Receive	Transmit	Receive	Transmit	Receive	Transmit	Receive	Transmit
Not Applicable	"Shall"	"Shall"	"Should not"	"Shall"	Not Applicable	"Shall"	"Should not"

1148 Unlike other DSI packets, an EoTp has a fixed format as follows:

- 1149 • Data Type = DI [5:0] = 0b001000
- 1150 • Virtual Channel = DI [7:6] = 0b00
- 1151 • Payload Data [15:0] = 0x0F0F
- 1152 • ECC [7:0] = 0x01

1153 The virtual channel identifier associated with an EoTp is fixed to 0, regardless of the number of different  
 1154 virtual channels present within the same transmission. For multi-Lane systems, the EoTp bytes are  
 1155 distributed across multiple Lanes.

### 1156 **8.8.3 Color Mode Off Command, Data Type = 00 0010 (0x02)**

1157 *Color Mode Off* is a Short packet command that returns a Video Mode display module from low-color  
 1158 mode to normal display operation.

### 1159 **8.8.4 Color Mode On Command, Data Type = 01 0010 (0x12)**

1160 *Color Mode On* is a Short packet command that switches a Video Mode display module to a low-color  
 1161 mode for power saving.



1162 **8.8.5 Shutdown Peripheral Command, Data Type = 10 0010 (0x22)**

1163 *Shutdown Peripheral* command is a Short packet command that turns off the display in a Video Mode  
1164 display module for power saving. Note the interface shall remain powered in order to receive the turn-on,  
1165 or wake-up, command.

1166 **8.8.6 Turn On Peripheral Command, Data Type = 11 0010 (0x32)**

1167 *Turn On Peripheral* command is Short packet command that turns on the display in a Video Mode display  
1168 module for normal display operation.

1169 **8.8.7 Generic Short WRITE Packet with 0, 1, or 2 parameters, Data Types = 00  
1170 0011 (0x03), 01 0011 (0x13), 10 0011 (0x23), Respectively**

1171 *Generic Short WRITE* command is a Short packet type for sending generic data to the peripheral. The  
1172 format and interpretation of the contents of this packet are outside the scope of this document. It is the  
1173 responsibility of the system designer to ensure that both the host processor and peripheral agree on the  
1174 format and interpretation of such data.

1175 The complete packet shall be four bytes in length including an ECC byte. The two Data Type MSBs, bits  
1176 [5:4], indicate the number of valid parameters (0, 1, or 2). For single-byte parameters, the parameter shall  
1177 be sent in the first data byte following the DI byte and the second data byte shall be set to 0x00.

1178 **8.8.8 Generic READ Request with 0, 1, or 2 Parameters, Data Types = 00 0100  
1179 (0x04), 01 0100 (0x14), 10 0100(0x24), Respectively**

1180 *Generic READ* request is a Short packet requesting data from the peripheral. The format and interpretation  
1181 of the parameters of this packet, and of returned data, are outside the scope of this document. It is the  
1182 responsibility of the system designer to ensure that both the host processor and peripheral agree on the  
1183 format and interpretation of such data.

1184 Returned data may be of Short or Long packet format. Note the *Set Max Return Packet Size* command  
1185 limits the size of returning packets so that the host processor can prevent buffer overflow conditions when  
1186 receiving data from the peripheral. If the returning block of data is larger than the maximum return packet  
1187 size specified, the read response will require more than one transmission. The host processor shall send  
1188 multiple Generic READ requests in separate transmissions if the requested data block is larger than the  
1189 maximum packet size.

1190 The complete packet shall be four bytes in length including an ECC byte. The two Data Type MSBs, bits  
1191 [5:4], indicate the number of valid parameters (0, 1, or 2). For single byte parameters, the parameter shall  
1192 be sent in the first data byte following the DI byte and the second data byte shall be set to 0x00.

1193 Since this is a read command, BTA shall be asserted by the host processor following this request.

1194 The peripheral shall respond to Generic READ Request in one of the following ways:

- 1195 • If an error was detected by the peripheral, it shall send *Acknowledge and Error Report*. If an ECC  
1196 error in the request was detected and corrected, the peripheral shall transmit the requested READ  
1197 data packet with the *Acknowledge and Error Report* packet appended, in the same transmission.
- 1198 • If no error was detected by the peripheral, it shall send the requested READ packet (Short or  
1199 Long) with appropriate ECC and Checksum, if Checksum is enabled.

1200 A Generic READ request shall be the only, or last, packet of a transmission. Following the transmission the  
 1201 host processor sends BTA. Having given control of the bus to the peripheral, the host processor will expect  
 1202 the peripheral to transmit the appropriate response packet and then return bus possession to the host  
 1203 processor.

## 1204 **8.8.9 DCS Commands**

1205 DCS is a standardized command set intended for Command Mode display modules. The interpretation of  
 1206 DCS commands is supplied in [MIPI01].

1207 For DCS short commands, the first byte following the Data Identifier Byte is the *DCS Command Byte*. If  
 1208 the DCS command does not require parameters, the second payload byte shall be 0x00.

1209 If a DCS Command requires more than one parameter, the command shall be sent as a Long Packet type.

### 1210 **8.8.9.1 DCS Short Write Command, 0 or 1 parameter, Data Types = 00 0101 (0x05), 01 1211 0101 (0x15), Respectively**

1212 *DCS Short Write* command is used to write a single data byte to a peripheral such as a display module. The  
 1213 packet is a Short packet composed of a Data ID byte, a DCS Write command, an optional parameter byte  
 1214 and an ECC byte. Data Type bit 4 shall be set to 1 if there is a valid parameter byte, and shall be set to 0 if  
 1215 there is no valid parameter byte. If a parameter is not required, the parameter byte shall be 0x00. If *DCS  
 1216 Short Write* command, followed by BTA, is sent to a bidirectional peripheral, the peripheral shall respond  
 1217 with ACK Trigger Message unless an error was detected in the host-to-peripheral transmission. If the  
 1218 peripheral detects an error in the transmission, the peripheral shall respond with *Acknowledge and Error  
 1219 Report*. If the peripheral is a Video Mode display on a unidirectional DSI, it shall ignore BTA. See Table  
 1220 19.

### 1221 **8.8.9.2 DCS Read Request, No Parameters, Data Type = 00 0110 (0x06)**

1222 DCS READ commands are used to request data from a display module. This packet is a Short packet  
 1223 composed of a Data ID byte, a DCS Read command, a byte set to 0x00 and an ECC byte. Since this is a  
 1224 read command, BTA shall be asserted by the host processor following completion of the transmission.  
 1225 Depending on the type of READ requested in the DCS Command Byte, the peripheral may respond with a  
 1226 DCS Short Read Response or DCS Long Read Response.

1227 The read response may be more than one packet in the case of DCS Long Read Response, if the returning  
 1228 block of data is larger than the maximum return packet size specified. In that case, the host processor shall  
 1229 send multiple DCS Read Request commands to transfer the complete data block. See section 8.8.10 for  
 1230 details on setting the read packet size.

1231 The peripheral shall respond to DCS READ Request in one of the following ways:

- 1232 • If an error was detected by the peripheral, it shall send *Acknowledge and Error Report*. If an ECC  
 1233 error in the request was detected and corrected, the peripheral shall send the requested READ data  
 1234 packet followed by the *Acknowledge and Error Report* packet in the same transmission.
- 1235 • If no error was detected by the peripheral, it shall send the requested READ packet (Short or  
 1236 Long) with appropriate ECC and Checksum, if either or both features are enabled.

1237 A DCS Read Request packet shall be the only, or last, packet of a transmission. Following the transmission,  
 1238 the host processor sends BTA. Having given control of the bus to the peripheral, the host processor will  
 1239 expect the peripheral to transmit the appropriate response packet and then return bus possession to the host  
 1240 processor.

1241 **8.8.9.3 DCS Long Write / write\_LUT Command, Data Type = 11 1001 (0x39)**

1242 *DCS Long Write/write\_LUT Command* is used to send larger blocks of data to a display module that  
1243 implements the Display Command Set.

1244 The packet consists of the DI byte, a two-byte WC, an ECC byte, followed by the *DCS Command Byte*, a  
1245 payload of length WC minus one bytes, and a two-byte checksum.

1246 **8.8.10 Set Maximum Return Packet Size, Data Type = 11 0111 (0x37)**

1247 *Set Maximum Return Packet Size* is a four-byte command packet (including ECC) that specifies the  
1248 maximum size of the payload in a Long packet transmitted from peripheral back to the host processor. The  
1249 order of bytes in *Set Maximum Return Packet Size* is: Data ID, two-byte value for maximum return packet  
1250 size, followed by the ECC byte. Note that the two-byte value is transmitted with LS byte first. This  
1251 command shall be ignored by peripherals with unidirectional DSI interfaces.

1252 During a power-on or Reset sequence, the Maximum Return Packet Size shall be set by the peripheral to a  
1253 default value of one. This parameter should be set by the host processor to the desired value in the  
1254 initialization routine before commencing normal operation.

1255 **8.8.11 Null Packet (Long), Data Type = 00 1001 (0x09)**

1256 *Null Packet* is a mechanism for keeping the serial Data Lane(s) in High-Speed mode while sending dummy  
1257 data. This is a Long packet. Like all packets, its content shall be an integer number of bytes.

1258 The Null Packet consists of the DI byte, a two-byte WC, ECC byte, and “null” payload of WC bytes,  
1259 ending with a two-byte Checksum. Actual data values sent are irrelevant because the peripheral does not  
1260 capture or store the data. However, ECC and Checksum shall be generated and transmitted to the  
1261 peripheral.

1262 **8.8.12 Blanking Packet (Long), Data Type = 01 1001 (0x19)**

1263 A Blanking packet is used to convey blanking timing information in a Long packet. Normally, the packet  
1264 represents a period between active scan lines of a Video Mode display, where traditional display timing is  
1265 provided from the host processor to the display module. The blanking period may have *Sync Event* packets  
1266 interspersed between blanking segments. Like all packets, the Blanking packet contents shall be an integer  
1267 number of bytes. Blanking packets may contain arbitrary data as payload.

1268 The Blanking packet consists of the DI byte, a two-byte WC, an ECC byte, a payload of length WC bytes,  
1269 and a two-byte checksum.

1270 **8.8.13 Generic Long Write, Data Type = 10 1001 (0x29)**

1271 *Generic Long Write Packet* is used to transmit arbitrary blocks of data from a host processor to a peripheral  
1272 in a Long packet. The packet consists of the DI byte, a two-byte WC, an ECC byte, a payload of length WC  
1273 bytes and a two-byte checksum.

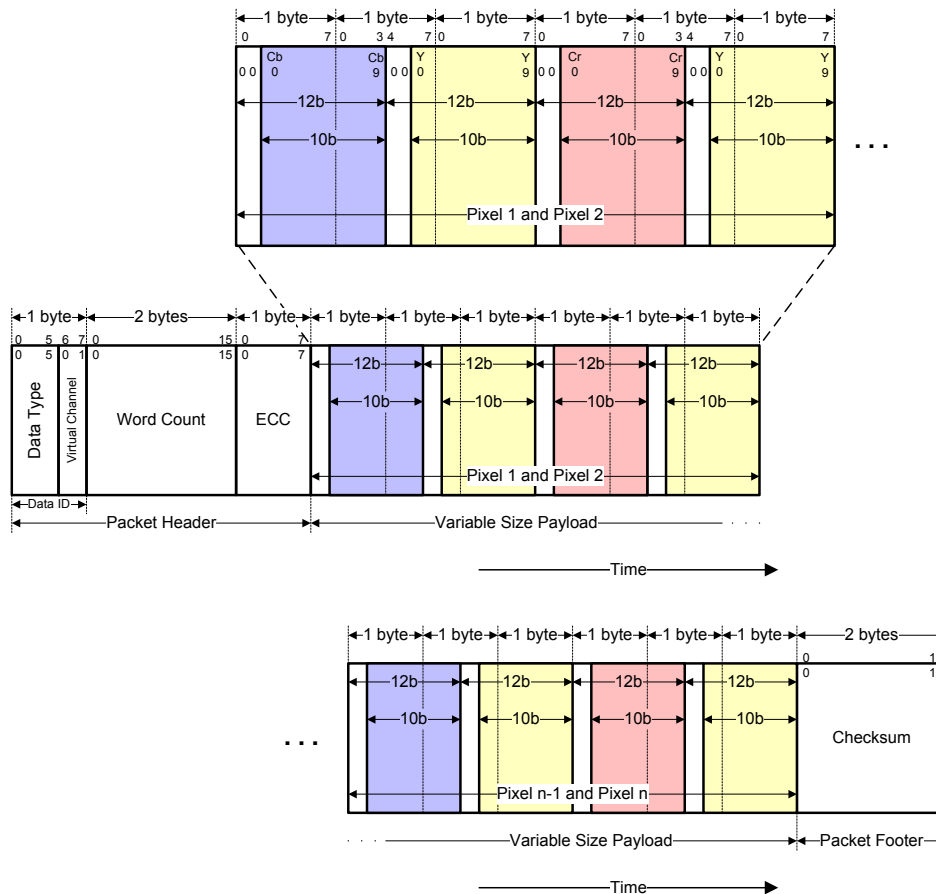
1274 **8.8.14 Loosely Packed Pixel Stream, 20-bit YCbCr 4:2:2 Format, Data Type = 00  
1275 1100 (0x0C)**

1276 *Loosely Packed Pixel Stream 20-bit YCbCr 4:2:2 Format* shown in Figure 18 is a Long packet used to  
1277 transmit image data formatted as 20-bits per pixel to a Video Mode display module. The packet consists of

1278 the DI byte, a two-byte, non-zero WC, an ECC byte, a payload of length WC bytes and a two-byte  
 1279 Checksum.

1280 When transmitting standard definition video, e.g. NTSC 480i30 or PAL 525i25, the pixel format is ITU-R  
 1281 Recommendation BT.601 (see [ITU01]). When transmitting high definition video, e.g. 1080i25, 1080i30 or  
 1282 720p60, the pixel format is ITU-R Recommendation BT.709 (see [ITU02]). Component ordering follows  
 1283 ITU-R Recommendation BT.656 (see [ITU03]).

1284 A pixel shall have ten bits for each of the Y-, Cb-, and Cr-components loosely packed into 12-bit fields as  
 1285 shown in Figure 18. The 10-bit component value shall be justified such that the most significant bits of the  
 1286 12-bit field, b[11:2] holds the 10-bit component value, d[9:0]. The least significant bits of the 12-bit field,  
 1287 b[1:0], shall be 00b. Within a component, the LSB is sent first, the MSB last.



1288

1289

**Figure 18 20-bit per Pixel – YCbCr 4:2:2 Format, Long Packet**

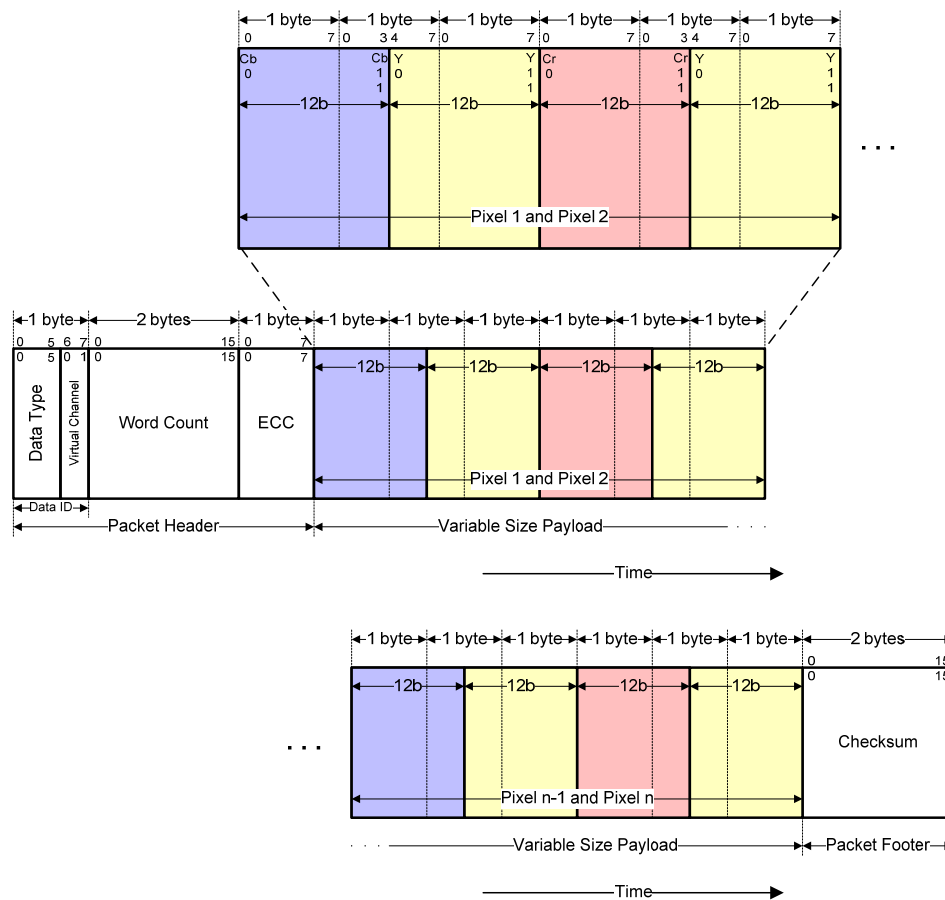
1290 With this format, pixel boundaries align with certain byte boundaries. The value in WC (size of payload in  
 1291 bytes) shall be any non-zero value divisible into an integer by six. Allowable values for WC = {6, 12, 18,...  
 1292 65 532}.

1293 **8.8.15 Packed Pixel Stream, 24-bit YCbCr 4:2:2 Format, Data Type = 01 1100**  
 1294 **(0x1C)**

1295 *Packed Pixel Stream 24-bit YCbCr 4:2:2 Format* shown in Figure 19 is a Long packet used to transmit  
 1296 image data formatted as 24-bits per pixel to a Video Mode display module. The packet consists of the DI  
 1297 byte, a two-byte, non-zero WC, an ECC byte, a payload of length WC bytes and a two-byte Checksum.

1298 When transmitting standard definition video, e.g. NTSC 480i30 or PAL 525i25, the pixel format is ITU-R  
 1299 Recommendation BT.601 (see [ITU01]). When transmitting high definition video, e.g. 1080i25, 1080i30 or  
 1300 720p60, the pixel format is ITU-R Recommendation BT.709 (see [ITU02]). Component ordering follows  
 1301 ITU-R Recommendation BT.656 (see [ITU03]).

1302 A pixel shall have twelve bits for each of the Y-, Cb-, and Cr-components as shown in Figure 19. Within a  
 1303 component, the LSB is sent first, the MSB last.



1304

1305

**Figure 19 24-bit per Pixel – YCbCr 4:2:2 Format, Long Packet**

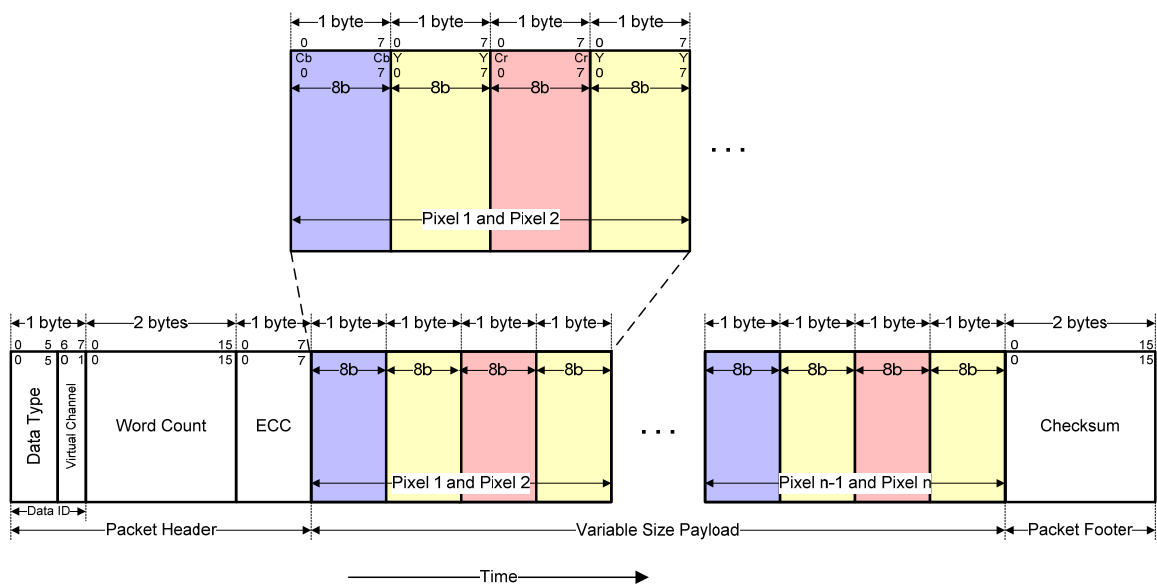
1306 With this format, pixel boundaries align with certain byte boundaries. The value in WC (size of payload in  
 1307 bytes) shall be any non-zero value divisible into an integer by six. Allowable values for WC = {6, 12, 18,...  
 1308 65 532}.

1309 **8.8.16 Packed Pixel Stream, 16-bit YCbCr 4:2:2 Format, Data Type = 10 1100**  
 1310 **(0x2C)**

1311 *Packed Pixel Stream 16-bit YCbCr 4:2:2 Format* shown in Figure 20 is a Long packet used to transmit  
 1312 image data formatted as 16-bits per pixel to a Video Mode display module. The packet consists of the DI  
 1313 byte, a two-byte, non-zero WC, an ECC byte, a payload of length WC bytes and a two-byte Checksum.

1314 When transmitting standard definition video, e.g. NTSC 480i30 or PAL 525i25, the pixel format is ITU-R  
 1315 Recommendation BT.601 (see [ITU01]). When transmitting high definition video, e.g. 1080i25, 1080i30 or  
 1316 720p60, the pixel format is ITU-R Recommendation BT.709 (see [ITU02]). Component ordering follows  
 1317 ITU-R Recommendation BT.656 (see [ITU03]).

1318 A pixel shall have eight bits for each of the Y-, Cb-, and Cr-components. Within a component, the LSB is  
 1319 sent first, the MSB last.



1320

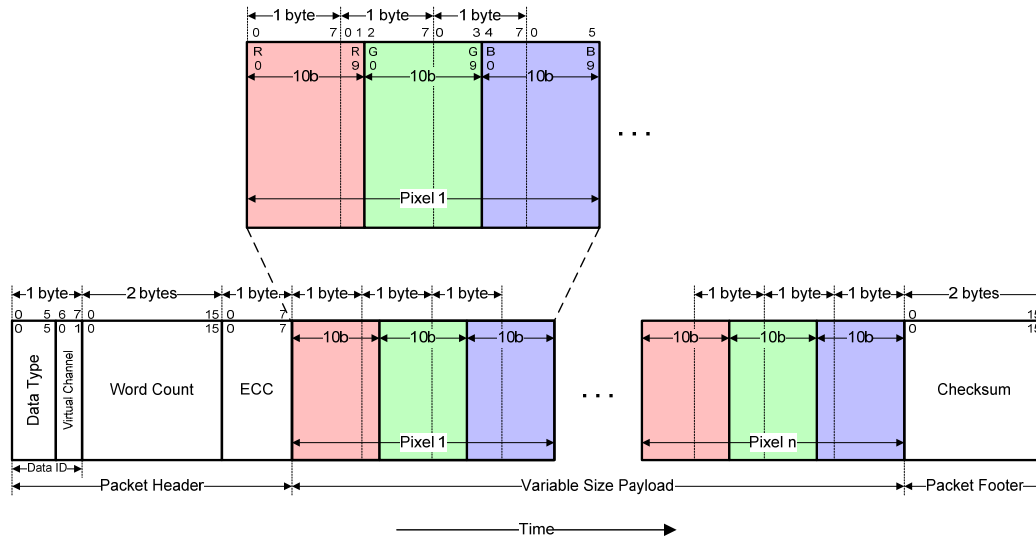
1321

**Figure 20 16-bit per Pixel – YCbCr 4:2:2 Format, Long Packet**

1322 With this format, pixel boundaries align with certain byte boundaries. The value in WC (size of payload in  
 1323 bytes) shall be any non-zero value divisible into an integer by four. Allowable values for WC = {4, 8, 12,...  
 1324 65 532}.

1325 **8.8.17 Packed Pixel Stream, 30-bit Format, Long Packet, Data Type = 00 1101**  
 1326 **(0x0D)**

1327 *Packed Pixel Stream 30-Bit Format* shown in Figure 21 is a Long packet used to transmit image data  
 1328 formatted as 30-bit pixels to a Video Mode display module. The packet consists of the DI byte, a two-byte,  
 1329 non-zero WC, an ECC byte, a payload of length WC bytes and a two-byte Checksum. The pixel format is  
 1330 red (10 bits), green (10 bits) and blue (10 bits), in that order. Within a color component, the LSB is sent  
 1331 first, the MSB last.



1332

1333

**Figure 21 30-bit per Pixel (Packed) – RGB Color Format, Long Packet**

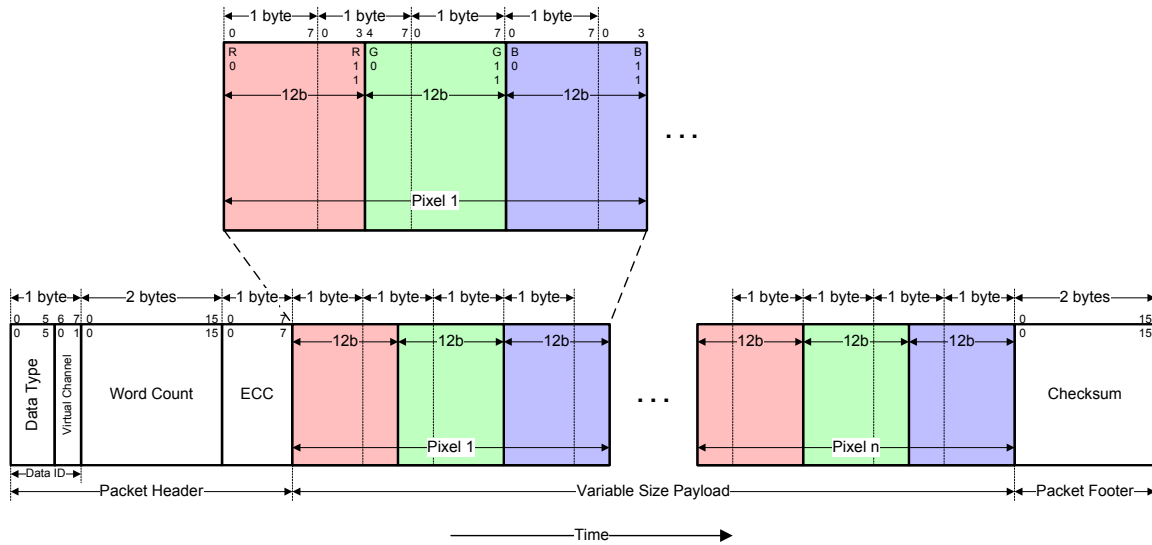
1334 This format uses sRGB color space. However, this Data Type may apply to other color spaces or data  
 1335 transfers using 30-bits per pixel when the color space, or related formatting information, is explicitly  
 1336 defined by a prior display command. For example, a future revision of [MIPI01] may extend the Data Type  
 1337 to include color spaces that differ from sRGB. The scope and nature of the formatting command is outside  
 1338 the scope of this document.

1339 With this format, pixel boundaries align with byte boundaries every four pixels (fifteen bytes). The total  
 1340 line width (displayed plus non-displayed pixels) should be a multiple of fifteen bytes. However, the value  
 1341 in WC (size of payload in bytes) shall not be restricted to non-zero values divisible by fifteen.

1342 Any trailing bits within a byte not entirely used by pixel data shall be zero. For example, a packet with only  
 1343 one pixel requires two trailing zero bits in the fourth data byte. If the pixel RGB value is 0b1111111111  
 1344 1111111111 1111111111, the fourth byte value equals 0x3F. The entire packet with VC = 0b00 would be  
 1345 0x0D 01 00 1E FF FF FF 3F B4 36.

#### 1346 **8.8.18 Packed Pixel Stream, 36-bit Format, Long Packet, Data Type = 01 1101** 1347 **(0x1D)**

1348 *Packed Pixel Stream 36-Bit Format* shown in Figure 22 is a Long packet used to transmit image data  
 1349 formatted as 36-bit pixels to a Video Mode display module. The packet consists of the DI byte, a two-byte,  
 1350 non-zero WC, an ECC byte, a payload of length WC bytes and a two-byte Checksum. The pixel format is  
 1351 red (12 bits), green (12 bits) and blue (12 bits), in that order. Within a color component, the LSB is sent  
 1352 first, the MSB last.



1353

1354

**Figure 22 36-bit per Pixel (Packed) – RGB Color Format, Long Packet**

1355 This format uses sRGB color space. However, this Data Type may apply to other color spaces or data  
 1356 transfers using 36-bits per pixel when the color space, or related formatting information, is explicitly  
 1357 defined by a prior display command. For example, a future revision of [MIPI01] may extend the Data Type  
 1358 to include color spaces that differ from sRGB. The scope and nature of the formatting command is outside  
 1359 the scope of this document.

1360 With this format, pixel boundaries align with byte boundaries every two pixels (nine bytes). The total line  
 1361 width (displayed plus non-displayed pixels) should be a multiple of nine bytes. However, the value in WC  
 1362 (size of payload in bytes) shall not be restricted to non-zero values divisible by nine.

1363 Any trailing bits within a byte not entirely used by pixel data shall be zero. For example, a packet with only  
 1364 one pixel requires four trailing zero bits in the fifth payload byte. If the pixel RGB value is  
 1365 0b111111111111 111111111111 111111111111, the fifth byte value equals 0x0F. The entire packet with  
 1366 VC = 0b00 would be 0x1D 01 00 0D FF FF FF FF 0F 4C 1C.

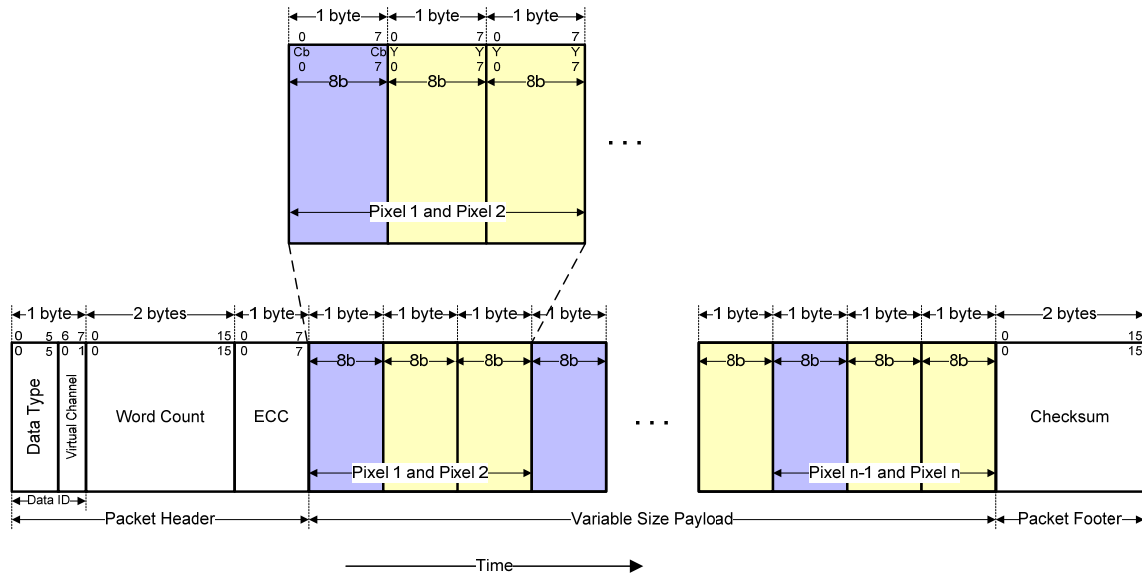
### 1367 **8.8.19 Packed Pixel Stream, 12-bit YCbCr 4:2:0 Format, Data Type = 11 1101** 1368 **(0x3D)**

1369 *Packed Pixel Stream 12-bit YCbCr 4:2:0 Format* shown in Figure 23 and Figure 24 is a Long packet used  
 1370 to transmit image data formatted as 12-bits per pixel to a Video Mode display module. The packet consists  
 1371 of the DI byte, a two-byte, non-zero WC, an ECC byte, a payload of length WC bytes and a two-byte  
 1372 Checksum.

1373 When transmitting standard definition video, e.g. NTSC 480i30 or PAL 525i25, the pixel format is ITU-R  
 1374 Recommendation BT.601 (see [ITU01]). When transmitting high definition video, e.g. 1080i25, 1080i30 or  
 1375 720p60, the pixel format is ITU-R Recommendation BT.709 (see [ITU02]).

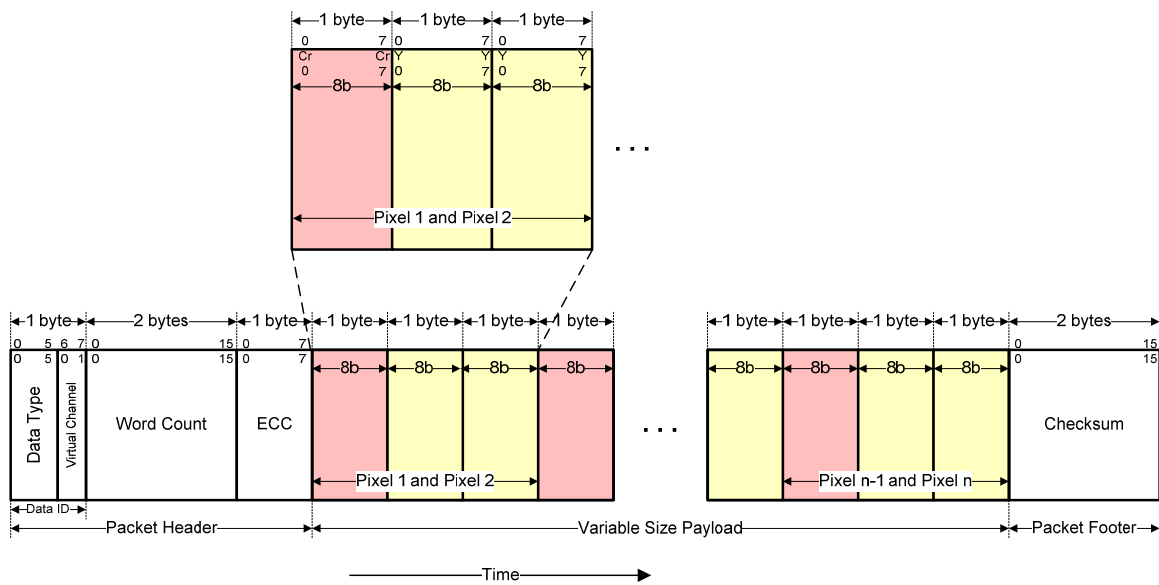
1376 A pixel shall have eight bits for each of the Y-, Cb-, and Cr-components. Within a component, the LSB is  
 1377 sent first, the MSB last. Cb- and Y-components are sent on odd lines as shown in Figure 23 while Cr- and  
 1378 Y-components are sent on even lines as shown in Figure 24.





1379  
1380

**Figure 23 12-bit per Pixel – YCbCr 4:2:0 Format (Odd Line), Long Packet**



1381  
1382  
1383

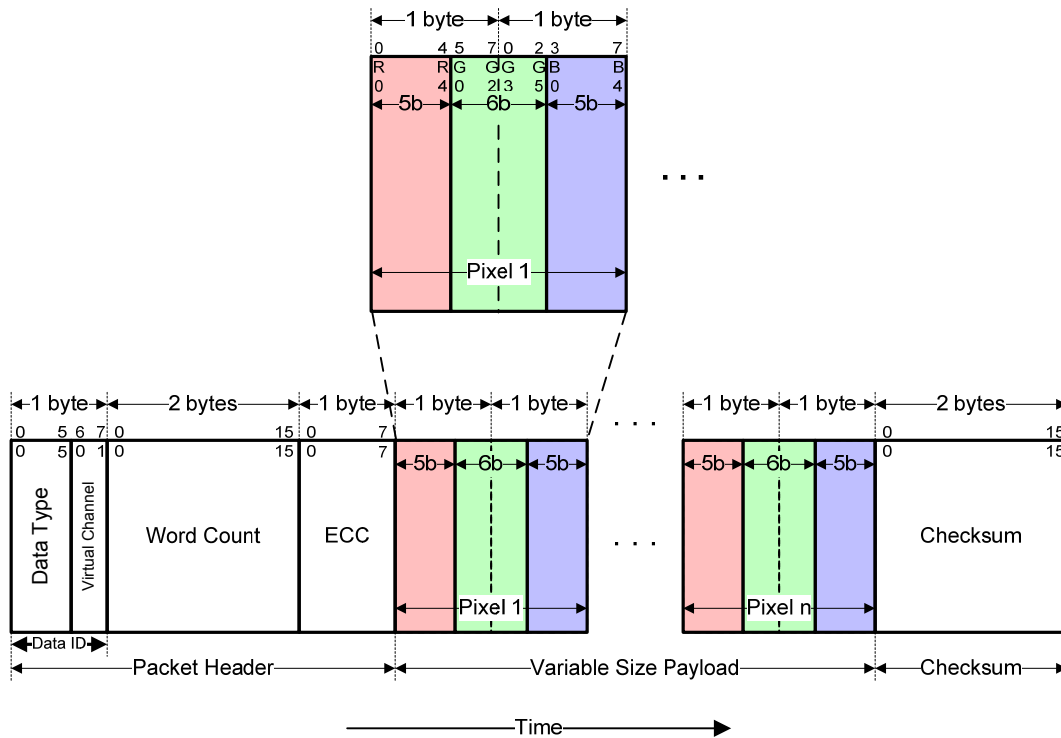
**Figure 24 12-bit per Pixel – YCbCr 4:2:0 Format (Even Line), Long Packet**

1384 The value in WC (size of payload in bytes) shall be any non-zero value divisible into an integer by three.  
1385 Allowable values for WC = {3, 6, 9,... 65 535}.

1386 **8.8.20 Packed Pixel Stream, 16-bit Format, Long Packet, Data Type 00 1110**  
1387 **(0x0E)**

1388 *Packed Pixel Stream 16-Bit Format* shown in Figure 25 is a Long packet used to transmit image data  
1389 formatted as 16-bit pixels to a Video Mode display module. The packet consists of the DI byte, a two-byte  
1390 WC, an ECC byte, a payload of length WC bytes and a two-byte checksum. Pixel format is five bits red, six

1391 bits green, five bits blue, in that order. Note that the “Green” component is split across two bytes. Within a  
 1392 color component, the LSB is sent first, the MSB last.



1393  
 1394

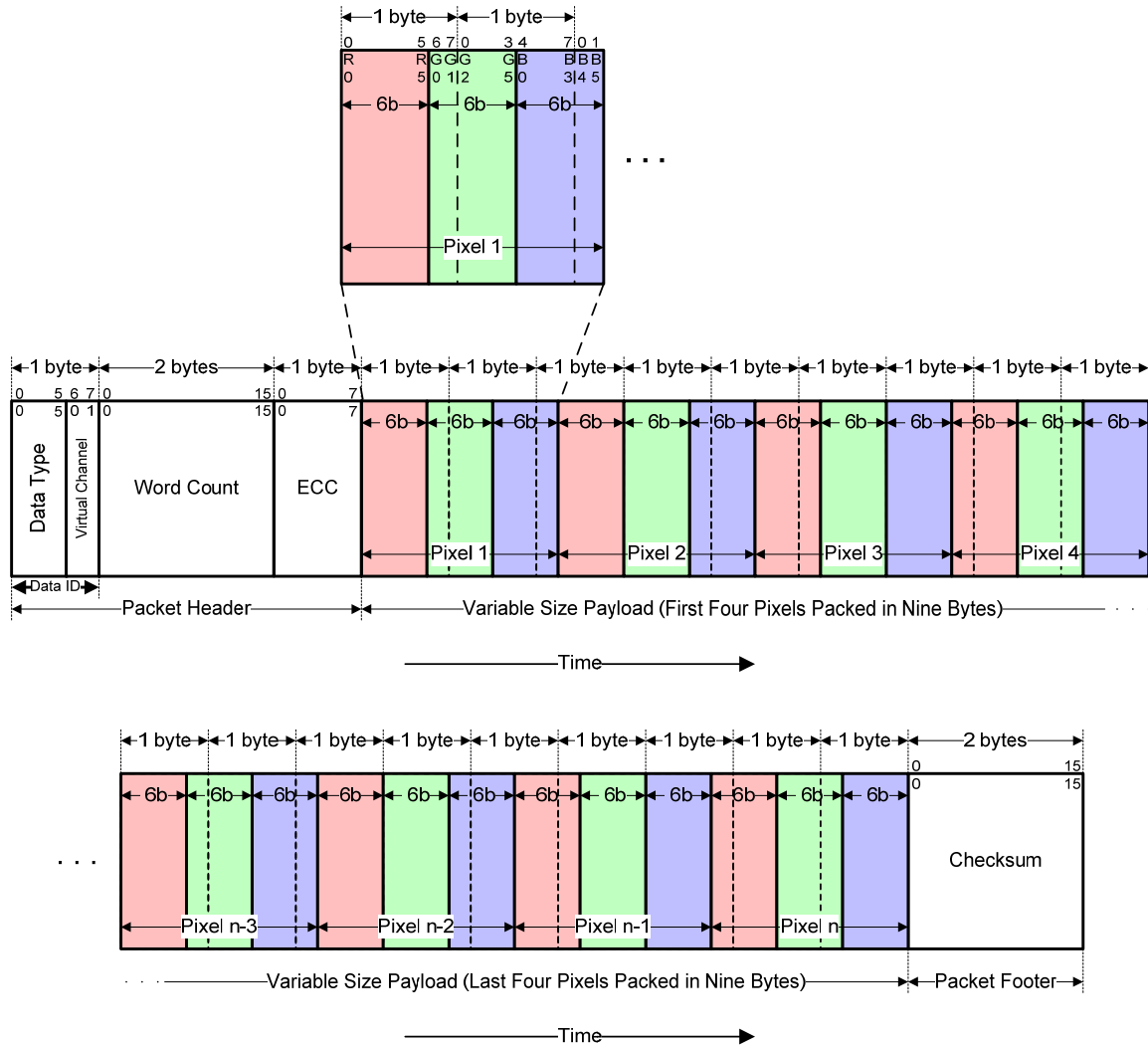
**Figure 25 16-bit per Pixel – RGB Color Format, Long Packet**

1395 With this format, pixel boundaries align with byte boundaries every two bytes. The total line width  
 1396 (displayed plus non-displayed pixels) should be a multiple of two bytes.

1397 Normally, the display module has no frame buffer of its own, so all image data shall be supplied by the host  
 1398 processor at a sufficiently high rate to avoid flicker or other visible artifacts.

1399 **8.8.21 Packed Pixel Stream, 18-bit Format, Long Packet, Data Type = 01 1110**  
 1400 **(0x1E)**

1401 *Packed Pixel Stream 18-Bit Format (Packed)* shown in Figure 26 is a Long packet. It is used to transmit  
 1402 RGB image data formatted as pixels to a Video Mode display module that displays 18-bit pixels. The packet  
 1403 consists of the DI byte, a two-byte WC, an ECC byte, a payload of length WC bytes and a two-byte  
 1404 Checksum. Pixel format is red (6 bits), green (6 bits) and blue (6 bits), in that order. Within a  
 1405 color component, the LSB is sent first, the MSB last.



1406  
1407

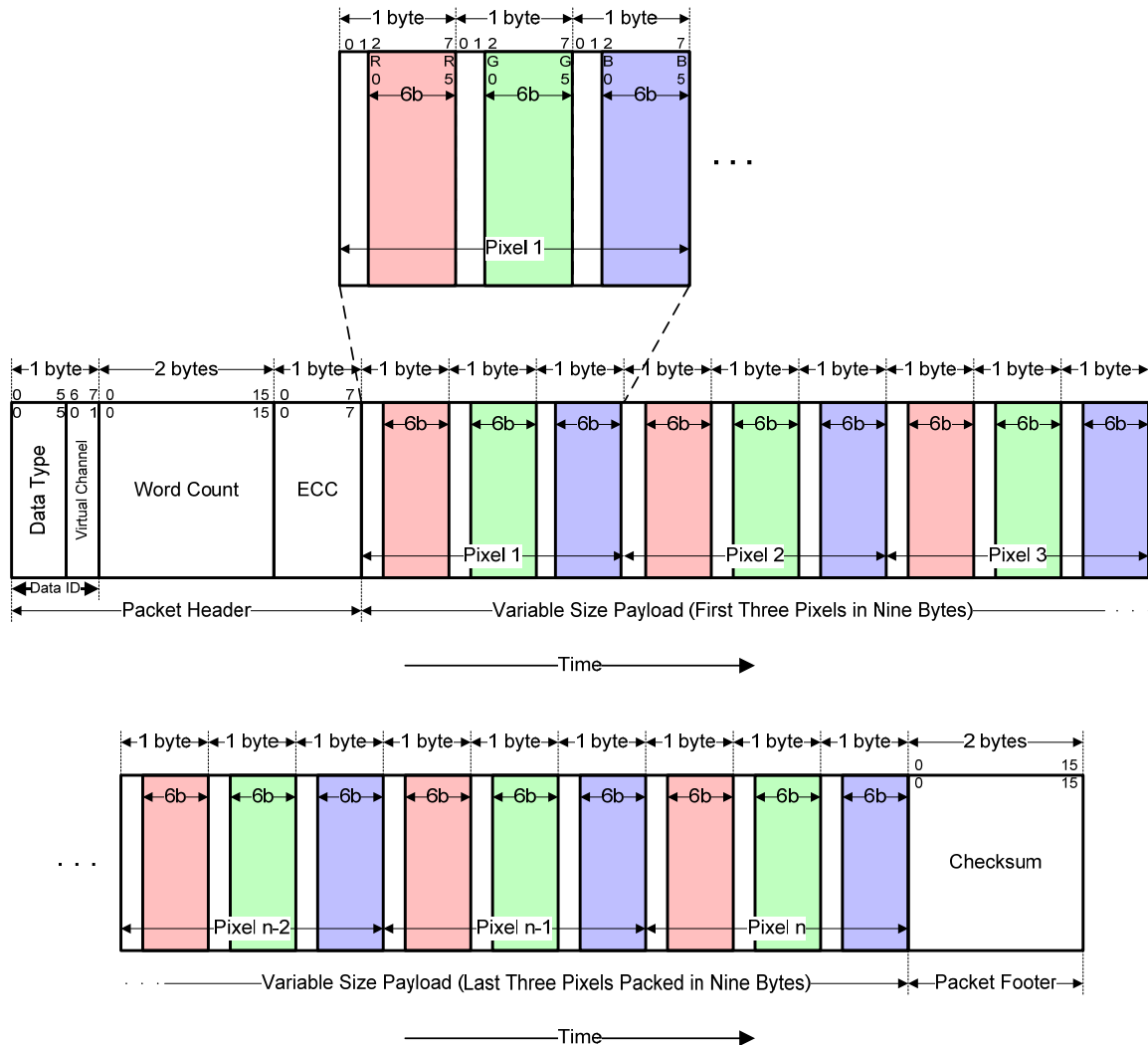
**Figure 26 18-bit per Pixel (Packed) – RGB Color Format, Long Packet**

1408 Note that pixel boundaries only align with byte boundaries every four pixels (nine bytes). Preferably,  
 1409 display modules employing this format have a horizontal extent (width in pixels) evenly divisible by four,  
 1410 so no partial bytes remain at the end of the display line data. If the active (displayed) horizontal width is not  
 1411 a multiple of four pixels, the transmitter shall send additional fill pixels at the end of the display line to  
 1412 make the transmitted width a multiple of four pixels. The receiving peripheral shall not display the fill  
 1413 pixels when refreshing the display device. For example, if a display device has an active display width of  
 1414 399 pixels, the transmitter should send 400 pixels in one or more packets. The receiver should display the  
 1415 first 399 pixels and discard the last pixel of the transmission.

1416 With this format, the total line width (displayed plus non-displayed pixels) should be a multiple of four  
 1417 pixels (nine bytes).

1418 **8.8.22 Pixel Stream, 18-bit Format in Three Bytes, Long Packet, Data Type = 10**  
 1419 **1110 (0x2E)**

1420 In the *18-bit Pixel Loosely Packed* format, each R, G, or B color component is six bits, but is shifted to the  
 1421 upper bits of the byte, such that the valid pixel bits occupy bits [7:2] of each byte as shown in Figure 27.  
 1422 Bits [1:0] of each payload byte representing active pixels are ignored. As a result, each pixel requires three  
 1423 bytes as it is transmitted across the Link. This requires more bandwidth than the “packed” format, but  
 1424 requires less shifting and multiplexing logic in the packing and unpacking functions on each end of the  
 1425 Link.



1426

1427

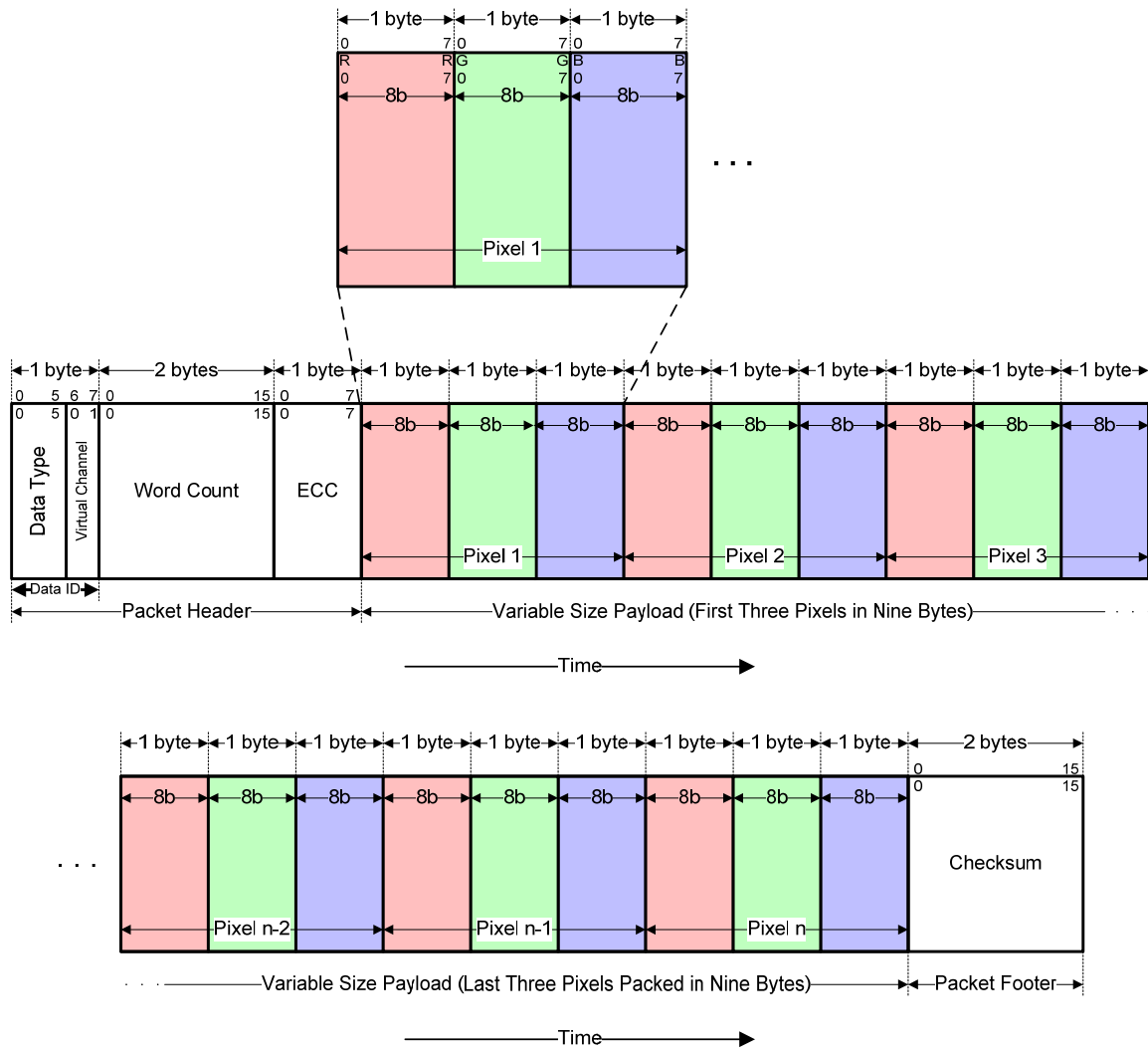
**Figure 27 18-bit per Pixel (Loosely Packed) – RGB Color Format, Long Packet**

1428 This format is used to transmit RGB image data formatted as pixels to a Video Mode display module that  
 1429 displays 18-bit pixels. The packet consists of the DI byte, a two-byte WC, an ECC byte, a payload of length  
 1430 WC bytes and a two-byte Checksum. The pixel format is red (6 bits), green (6 bits) and blue (6 bits) in that  
 1431 order. Within a color component, the LSB is sent first, the MSB last.

1432 With this format, pixel boundaries align with byte boundaries every three bytes. The total line width  
 1433 (displayed plus non-displayed pixels) should be a multiple of three bytes.

1434 **8.8.23 Packed Pixel Stream, 24-bit Format, Long Packet, Data Type = 11 1110**  
 1435 **(0x3E)**

1436 *Packed Pixel Stream 24-Bit Format* shown in Figure 28 is a Long packet. It is used to transmit image data  
 1437 formatted as 24-bit pixels to a Video Mode display module. The packet consists of the DI byte, a two-byte  
 1438 WC, an ECC byte, a payload of length WC bytes and a two-byte Checksum. The pixel format is red (8  
 1439 bits), green (8 bits) and blue (8 bits), in that order. Each color component occupies one byte in the pixel  
 1440 stream; no components are split across byte boundaries. Within a color component, the LSB is sent first,  
 1441 the MSB last.



1442

1443

**Figure 28 24-bit per Pixel – RGB Color Format, Long Packet**

1444 With this format, pixel boundaries align with byte boundaries every three bytes. The total line width  
 1445 (displayed plus non-displayed pixels) should be a multiple of three bytes.

#### 1446 **8.8.24 DO NOT USE and Reserved Data Types**

1447 Data Type codes with four LSBs = 0000 or 1111 shall not be used. All other non-specified Data Type  
1448 codes are reserved.

1449 Note that DT encoding is specified so that all data types have at least one 0-1 or 1-0 transition in the four  
1450 bits DT bits [3:0]. This ensures a transition within the first four bits of the serial data stream of every  
1451 packet. DSI protocol or the PHY can use this information to determine quickly, following the end of each  
1452 packet, if the next bits represent the start of a new packet (transition within four bits) or an EoT sequence  
1453 (no transition for at least four bits).

### 1454 **8.9 Peripheral-to-Processor (Reverse Direction) LP Transmissions**

1455 All Command Mode systems require bidirectional capability for returning READ data, acknowledge, or  
1456 error information to the host processor. Multi-Lane systems shall use Lane 0 for all peripheral-to-processor  
1457 transmissions; other Lanes shall be unidirectional.

1458 Reverse-direction signaling shall only use LP (Low Power) mode of transmission.

1459 Simple, low-cost systems using display modules which work exclusively in Video Mode may be  
1460 configured with unidirectional DSI for all Lanes. In such systems, no acknowledge or error reporting is  
1461 possible using DSI, and no requirements specified in this section apply to such systems. However, these  
1462 systems shall have ECC checking and correction capability, which enables them to correct single-bit errors  
1463 in headers and Short packets, even if they cannot report the error.

1464 Command Mode systems that use DCS shall have a bidirectional data path. Short packets and the header of  
1465 Long packets shall use ECC and may use Checksum to provide a higher level of data integrity. The  
1466 Checksum feature enables detection of errors in the payload of Long packets.

#### 1467 **8.9.1 Packet Structure for Peripheral-to-Processor LP Transmissions**

1468 Packet structure for peripheral-to-processor transactions is the same as for the processor-to-peripheral  
1469 direction.

1470 As in the processor-to-peripheral direction, two basic packet formats are specified: Short and Long. For  
1471 both types, an ECC byte shall be calculated to cover the Packet Header data. ECC calculation is the same in  
1472 the peripheral as in the host processor. For Long packets, error checking on the Data Payload, i.e. all bytes  
1473 after the Packet Header, is optional. If the Checksum is not calculated by the peripheral the Packet Footer  
1474 shall be 0x0000.

1475 BTA shall take place after every peripheral-to-processor transaction. This returns bus control to the host  
1476 processor following the completion of the LP transmission from the peripheral.

1477 Peripheral-to-processor transactions are of four basic types:

- 1478 • *Tearing Effect (TE)* is a Trigger message sent to convey display timing information to the host  
1479 processor. *Trigger* messages are single byte packets sent by a peripheral's PHY layer in response  
1480 to a signal from the DSI protocol layer. See [MIPI04] for a description of Trigger messages.
- 1481 • *Acknowledge* is a Trigger Message sent when the current transmission, as well as all preceding  
1482 transmissions since the last peripheral to host communication, i.e. either triggers or packets, is  
1483 received by the peripheral with no errors.

- 1484 • *Acknowledge and Error Report* is a Short packet sent if any errors were detected in preceding  
1485 transmissions from the host processor. Once reported, accumulated errors in the error register are  
1486 cleared.
- 1487 • *Response to Read Request* may be a Short or Long packet that returns data requested by the  
1488 preceding READ command from the processor.

## 1489 **8.9.2 System Requirements for ECC and Checksum and Packet Format**

1490 A peripheral shall implement ECC, and may optionally implement checksum.

1491 ECC support is the capability of generating ECC bytes locally from incoming packet headers and  
1492 comparing the results to the ECC fields of incoming packet headers in order to determine if an error has  
1493 occurred. DSI ECC provides detection and correction of single-bit errors and detection of multiple-bit  
1494 errors. See sections 9.4 and 9.5 for information on generating and applying ECC, respectively.

1495 For Command Mode peripherals, if a single-bit error has occurred the peripheral shall correct the error, set  
1496 the appropriate error bit (section 8.9.5) and report the error to the Host at the next available opportunity.  
1497 The packet can be used as if no error occurred. If a multiple-bit error is detected, the receiver shall drop the  
1498 packet and the rest of the transmission, set the relevant error bit and report the error back to the Host at the  
1499 next available opportunity. When the peripheral is reporting to the Host, it shall compute and send the  
1500 correct ECC based on the content of the header being transmitted.

1501 For Video Mode peripherals, if a single-bit error has occurred the peripheral shall correct the error and use  
1502 the packet as if no error occurred. If a multiple-bit error is detected, the receiver shall drop the packet and  
1503 the rest of the transmission. Since DSI Links may be unidirectional in Video Mode, error reporting  
1504 capabilities in these cases are application specific and out of scope of this document.

1505 Host processors shall implement both ECC and checksum capabilities. ECC and Checksum capabilities  
1506 shall be separately enabled or disabled so that a host processor can match a peripheral's capability when  
1507 checking return data from the peripheral. Note, in previous revisions of DSI peripheral support for ECC  
1508 was optional. See section 10.6. The mechanism for enabling and disabling Checksum capability is out of  
1509 scope for this document.

1510 An ECC byte can be applied to both Short and Long packets. Checksum bytes shall only be applied to  
1511 Long packets.

1512 Host processors and peripherals shall provide ECC support in both the Forward and Reverse  
1513 communication directions.

1514 Host processors, and peripherals that implement Checksum, shall provide Checksum capabilities in both  
1515 the Forward and Reverse communication directions.

1516 See section 8.4 for a description of the ECC and Checksum bytes.

## 1517 **8.9.3 Appropriate Responses to Commands and ACK Requests**

1518 In general, if the host processor completes a transmission to the peripheral with BTA asserted, the  
1519 peripheral shall respond with one or more appropriate packet(s), and then return bus ownership to the host  
1520 processor. If BTA is not asserted following a transmission from the host processor, the peripheral shall not  
1521 communicate an *Acknowledge* or error information back to the host processor.

1522 Interpretation of processor-to-peripheral transactions with BTA asserted, and the expected responses, are as  
1523 follows:

- 1524 • Following a non-Read command, the peripheral shall respond with *Acknowledge* if no errors were  
1525 detected and stored since the last peripheral to host communication, i.e. either triggers or packets.
- 1526 • Following a Read request, the peripheral shall send the requested READ data if no errors were  
1527 detected and stored since the last peripheral to host communication, i.e. either triggers or packets.
- 1528 • Following a Read request if only a single-bit ECC error was detected and corrected, the peripheral  
1529 shall send the requested READ data in a Long or Short packet, followed by a 4-byte *Acknowledge*  
1530 *and Error Report* packet in the same LP transmission. The Error Report shall have the *ECC Error*  
1531 *– Single Bit* flag set, as well as any error bits from any preceding transmissions stored since the  
1532 last peripheral to host communication.
- 1533 • Following a non-Read command if only a single-bit ECC error was detected and corrected, the  
1534 peripheral shall proceed to execute the command, and shall respond to BTA by sending a 4-byte  
1535 *Acknowledge and Error Report* packet. The Error Report shall have the *ECC Error – Single Bit*  
1536 flag set, as well as any error bits from any preceding transmissions stored since the last peripheral  
1537 to host communication.
- 1538 • Following a Read request, if multi-bit ECC errors were detected and not corrected, the peripheral  
1539 shall send a 4-byte *Acknowledge and Error Report* packet without sending Read data. The Error  
1540 Report shall have the *ECC Error – Multi-Bit* flag set, as well as any error bits from any preceding  
1541 transmissions stored since the last peripheral to host communication.
- 1542 • Following a non-Read command, if multi-bit ECC errors were detected and not corrected, the  
1543 peripheral shall not execute the command, and shall send a 4-byte *Acknowledge and Error Report*  
1544 packet. The Error Report shall have the *ECC Error – Multi-Bit* flag set, as well as any error bits  
1545 from any preceding transmissions stored since the last peripheral to host communication.
- 1546 • Following any command, if *SoT Error*, *SoT Sync Error* or *DSI VC ID Invalid* or DSI protocol  
1547 violation was detected, or the DSI command was not recognized, the peripheral shall send a 4-byte  
1548 *Acknowledge and Error Report* response, with the appropriate error flags set, as well as any error  
1549 bits from any preceding transmissions stored since the last peripheral to host communication, in  
1550 the two-byte error field. Only the *Acknowledge and Error Report* packet shall be transmitted; no  
1551 read or write accesses shall take place on the peripheral in response.
- 1552 • Following any command, if *EoT Sync Error* or *LP Transmit Sync Error* is detected, or a checksum  
1553 error is detected in the payload, the peripheral shall send a 4-byte *Acknowledge and Error Report*  
1554 packet with the appropriate error flags set, as well as any error bits from any preceding  
1555 transmissions stored since the last peripheral to host communication. For a read command, only  
1556 the *Acknowledge and Error Report* packet shall be transmitted; no read data shall be sent by the  
1557 peripheral in response.

1558 Refer to section 7 for how the peripheral acts when encountering Escape Mode Entry Command Error, Low  
1559 Level Transmit Sync Error and False Control Error. Section 7.2.2.2 elaborates on HS Receive Timeout  
1560 Error.

1561 Once reported to the host processor, all errors documented in this section are cleared from the Error  
1562 Register. Other error types may be detected, stored, and reported by a peripheral, but the mechanisms for  
1563 flagging, reporting, and clearing such errors are outside the scope of this document.



1564 **8.9.4 Format of Acknowledge and Error Report and Read Response Data**  
 1565 **Types**

1566 *Acknowledge and Error Report* confirms that the preceding command or data sent from the host processor  
 1567 to a peripheral was received, and indicates what types of error were detected on the transmission and any  
 1568 preceding transmissions. Note that if errors accumulate from multiple preceding transmissions, it may be  
 1569 difficult or impossible to identify which transmission contained the error. This message is a Short packet of  
 1570 four bytes, taking the form:

- 1571 • Byte 0: Data Identifier (Virtual Channel ID + Acknowledge Data Type)
- 1572 • Byte 1: Error Report bits 0-7
- 1573 • Byte 2: Error Report bits 8-15
- 1574 • ECC byte covering the header

1575 *Acknowledge* is sent using a Trigger message. See [MIPI04] for a description of Trigger messages:

- 1576 • Byte 0: 00100001 (shown here in first bit [left] to last bit [right] sequence)

1577 *Response to Read Request* returns data requested by the preceding READ command from the processor.  
 1578 These may be short or Long packets. The format for short READ packet responses is:

- 1579 • Byte 0: Data Identifier (Virtual Channel ID + Data Type)
- 1580 • Bytes 1, 2: READ data, may be one or two bytes. For single byte parameters, the parameter shall  
 1581 be returned in Byte 1 and Byte 2 shall be set to 0x00.
- 1582 • ECC byte covering the header

1583 The format for long READ packet responses is:

- 1584 • Byte 0: Data Identifier (Virtual Channel ID + Data Type)
- 1585 • Bytes 1-2: Word Count N (N = 0 to 65, 535)
- 1586 • ECC byte covering the header
- 1587 • N Bytes: READ data, may be from 1 to N bytes
- 1588 • Checksum, two bytes (16-bit checksum)
- 1589 • If Checksum is not calculated by the peripheral, send 0x0000

1590 **8.9.5 Error Reporting Format**

1591 An error report is a Short packet comprised of two bytes following the DI byte, with an ECC byte  
 1592 following the Error Report bytes. By convention, detection and reporting of each error type is signified by  
 1593 setting the corresponding bit to “1”. Table 18 shows the bit assignment for all error reporting.

1594

**Table 18 Error Report Bit Definitions**

Bit	Description
0	SoT Error
1	SoT Sync Error
2	EoT Sync Error
3	Escape Mode Entry Command Error
4	Low-Power Transmit Sync Error
5	Peripheral Timeout Error
6	False Control Error
7	Contention Detected
8	ECC Error, single-bit (detected and corrected)
9	ECC Error, multi-bit (detected, not corrected)
10	Checksum Error (Long packet only)
11	DSI Data Type Not Recognized
12	DSI VC ID Invalid
13	Invalid Transmission Length
14	Reserved
15	DSI Protocol Violation

1595 The first eight bits, bit 0 through bit 7, are related to the physical layer errors that are described in sections  
 1596 7.1 and 7.2. Bits 8 and 9 are related to single-bit and multi-bit ECC errors. The remaining bits indicate DSI  
 1597 protocol-specific errors.

1598 A single-bit ECC error implies that the receiver has already corrected the error and continued with the  
 1599 previous transmission. Therefore, the data does not need to be retransmitted. A Checksum error can be  
 1600 detected and reported back to Host using a Bidirectional Link by a peripheral that has implemented CRC  
 1601 checking capability. A Host may retransmit the data or not.

1602 A DSI Data Type Not Recognized error is caused by receiving a Data Type that is either not defined or is  
 1603 defined but not implemented by the peripheral, e.g. a Command Mode peripheral may not implement  
 1604 Video Mode-specific commands such as streaming 18-bit packed RGB pixels. After encountering an  
 1605 unrecognized Data Type or multiple-bit ECC error, the receiver effectively loses packet boundaries within  
 1606 a transmission and shall drop the transmission from the point where the error was detected.

1607 DSI VC ID Invalid error is reported whenever a peripheral encounters a packet header with an  
 1608 unrecognizable VC ID.

1609 An Invalid Transmission Length error is detected whenever a peripheral receives an incorrect number of  
 1610 bytes within a particular transmission. For example, if the WC field of the header does not match the actual  
 1611 number of payload bytes for a particular packet. Depending on the number, as well as the contents, of the  
 1612 bytes following the error, there is a good chance that other types of errors such as Checksum, ECC or  
 1613 unrecognized Data Type could be detected. Another example would be a case where peripheral receives a  
 1614 short packet, i.e. four bytes plus EoT within a transmission, with a long Data Type code in the header. In  
 1615 general, the Host is responsible for maintaining the integrity of the DSI protocol. If the ECC field was  
 1616 detected correctly, implying that host may have made a mistake by inserting a wrong Data Type into the

1617 short packet, the following EoTp could be interpreted as payload for the previous packet by a peripheral.  
 1618 Depending on the WC field, a Checksum error or an unrecognized Data Type error could be detected. In  
 1619 effect, the receiver detects an invalid transmission length, sets bit 13 and reports it back to the host after the  
 1620 first BTA opportunity.

1621 In the previous example, the peripheral can also detect that an EoTp was not received correctly, which  
 1622 implies a protocol violation. Bit 15 is used to indicate DSI protocol violations where a peripheral  
 1623 encounters a situation where an expected EoTp was not received at the end of a transmission or an expected  
 1624 BTA was not received after a read request. Although host devices should maintain DSI protocol integrity,  
 1625 DSI peripherals shall be able to detect both these cases of protocol violation.

1626 Other protocol violation scenarios exist, but since there are only a limited number of bits for reporting  
 1627 errors, an extension mechanism is required. Peripheral vendors shall specify an implementation-specific  
 1628 error status register where a Host can obtain additional information regarding what type of protocol  
 1629 violation occurred by issuing a read request, e.g. via a generic DSI read packet, after receiving an  
 1630 *Acknowledge and Error Report* packet with bit 15 set. The type of protocol violations, along with the  
 1631 address of the particular error status register and the generic read packet format used to address this register  
 1632 shall be documented in the relevant peripheral data sheet. The peripheral data sheet and documentation  
 1633 format is out of scope for this document.

## 1634 8.10 Peripheral-to-Processor Transactions – Detailed Format Description

1635 Table 19 presents the complete set of peripheral-to-processor Data Types.

1636 **Table 19 Data Types for Peripheral-sourced Packets**

Data Type, hex	Data Type, binary	Description	Packet Size
0x00 – 0x01	00 000X	Reserved	Short
0x02	00 0010	Acknowledge and Error Report	Short
0x03 – 0x07	00 0011 – 00 0111	Reserved	
0x08	00 1000	End of Transmission packet (EoTp)	Short
0x09 – 0x10	00 1001 – 01 0000	Reserved	
0x11	01 0001	Generic Short READ Response, 1 byte returned	Short
0x12	01 0010	Generic Short READ Response, 2 bytes returned	Short
0x13 – 0x19	01 0011 – 01 1001	Reserved	
0x1A	01 1010	Generic Long READ Response	Long
0x1B	01 1011	Reserved	
0x1C	01 1100	DCS Long READ Response	Long
0x1D – 0x20	01 1101 – 10 0000	Reserved	
0x21	10 0001	DCS Short READ Response, 1 byte returned	Short
0x22	10 0010	DCS Short READ Response, 2 bytes returned	Short
0x23 – 0x3F	10 0011 –	Reserved	

Data Type, hex	Data Type, binary	Description	Packet Size
	11 1111		

1637 **8.10.1 Acknowledge and Error Report, Data Type 00 0010 (0x02)**

1638 *Acknowledge and Error Report* is sent in response to any command, or read request, with BTA asserted  
 1639 when a reportable error is detected in the preceding, or earlier, transmission from the host processor. In the  
 1640 case of a correctable ECC error, this packet is sent following the requested READ data packet in the same  
 1641 LP transmission.

1642 When multiple peripherals share a single DSI, the *Acknowledge and Error Report* packet shall be tagged  
 1643 with the Virtual Channel ID 0b00.

1644 Although some errors, such as a correctable ECC error, can be associated with a packet targeted at a  
 1645 specific peripheral, an uncorrectable error cannot be associated with any particular peripheral. Additionally,  
 1646 many detectable error types are PHY-level transmission errors and cannot be associated with specific  
 1647 packets.

1648 **8.10.2 Generic Short Read Response, 1 or 2 Bytes, Data Types = 01 0001 or 01**  
 1649 **0010, Respectively**

1650 This is the short-packet response to *Generic READ Request*. Packet composition is the Data Identifier (DI)  
 1651 byte, two bytes of payload data and an ECC byte. The number of valid bytes is indicated by the Data Type  
 1652 LSBs, DT bits [1:0]. DT = 01 0001 indicates one byte and DT = 01 0010 indicates two bytes are returned.  
 1653 For a single-byte read response, valid data shall be returned in the first (LS) byte, and the second (MS) byte  
 1654 shall be sent as 0x00.

1655 This form of data transfer may be used for other features incorporated on the peripheral, such as a touch-  
 1656 screen integrated on the display module. Data formats for such applications are outside the scope of this  
 1657 document.

1658 If the command itself is possibly corrupt, due to an uncorrectable ECC error, SoT or SoT Sync error, the  
 1659 requested READ data packet shall not be sent and only the *Acknowledge and Error Report* packet shall be  
 1660 sent.

1661 **8.10.3 Generic Long Read Response with Optional Checksum, Data Type = 01**  
 1662 **1010 (0x1A)**

1663 This is the long-packet response to *Generic READ Request*. Packet composition is the Data Identifier (DI)  
 1664 byte followed by a two-byte Word Count, an ECC byte, N bytes of payload, and a two-byte Checksum. If  
 1665 the peripheral is Checksum capable, it shall return a calculated two-byte Checksum appended to the N-byte  
 1666 payload data. If the peripheral does not support Checksum it shall return 0x0000.

1667 If the command itself is possibly corrupt, due to an uncorrectable ECC error, SoT or SoT Sync error, the  
 1668 requested READ data packet shall not be sent and only the *Acknowledge and Error Report* packet shall be  
 1669 sent.

1670 **8.10.4 DCS Long Read Response with Optional Checksum, Data Type 01 1100**  
 1671 **(0x1C)**

1672 This is a Long packet response to *DCS Read Request*. Packet composition is the Data Identifier (DI) byte  
 1673 followed by a two-byte Word Count, an ECC byte, N bytes of payload, and a two-byte Checksum. If the  
 1674 peripheral is Checksum capable, it shall return a calculated two-byte Checksum appended to the N-byte  
 1675 payload data. If the peripheral does not support Checksum it shall return 0x0000.

1676 If the DCS command itself is possibly corrupt, due to uncorrectable ECC error, SoT or SoT Sync error, the  
 1677 requested READ data packet shall not be sent and only the *Acknowledge and Error Report* packet shall be  
 1678 sent.

1679 **8.10.5 DCS Short Read Response, 1 or 2 Bytes, Data Types = 10 0001 or 10**  
 1680 **0010, Respectively**

1681 This is the short-packet response to *DCS Read Request*. Packet composition is the Data Identifier (DI) byte,  
 1682 two bytes of payload data and an ECC byte. The number of valid bytes is indicated by the Data Type LSBs,  
 1683 DT bits [1:0]. DT = 01 0001 indicates one byte and DT = 01 0010 indicates two bytes are returned. For a  
 1684 single-byte read response, valid data shall be returned in the first (LS) byte, and the second (MS) byte shall  
 1685 be sent as 0x00.

1686 If the command itself is possibly corrupt, due to an uncorrectable ECC error, SoT or SoT Sync error, the  
 1687 requested READ data packet shall not be sent and only the *Acknowledge and Error Report* packet shall be  
 1688 sent.

1689 **8.10.6 Multiple Transmissions and Error Reporting**

1690 A peripheral shall report all errors documented in Table 18, when a command or request is followed by  
 1691 BTA giving bus possession to the peripheral. Peripheral shall accumulate errors from multiple transactions  
 1692 up until a time that host is issuing a BTA. After that, only one ACK Trigger Message or *Acknowledge and*  
 1693 *Error Report* packet shall be returned regardless of the number of packets or transmissions. Notice that host  
 1694 may not be able to associate each error to a particular packet or transmission causing that error.

1695 If receiving an *Acknowledge and Error Report* for each and every packet is desired, software can send  
 1696 individual packets within separate transmissions. In this case, a BTA follows each individual transmission.  
 1697 Furthermore, the peripheral may choose to store other information about errors that may be recovered by  
 1698 the host processor at a later time. The format and access mechanism of such additional error information is  
 1699 outside the scope of this document.

1700 **8.10.7 Clearing Error Bits**

1701 Errors shall be accumulated by the peripheral during single or multiple transmissions and only cleared after  
 1702 they have been reported back to the host processor. Errors are transmitted as part of an *Acknowledge and*  
 1703 *Error Report* response after the host issues a BTA.

1704 **8.11 Video Mode Interface Timing**

1705 Video Mode peripherals require pixel data delivered in real time. This section specifies the format and  
 1706 timing of DSI traffic for this type of display module.

### 1707 8.11.1 Transmission Packet Sequences

1708 DSI supports several formats, or packet sequences, for Video Mode data transmission. The peripheral's  
 1709 timing requirements dictate which format is appropriate. In the following sections, *Burst Mode* refers to  
 1710 time-compression of the RGB pixel (active video) portion of the transmission. In addition, these terms are  
 1711 used throughout the following sections:

- 1712 • Non-Burst Mode with Sync Pulses – enables the peripheral to accurately reconstruct original video  
 1713 timing, including sync pulse widths.
- 1714 • Non-Burst Mode with Sync Events – similar to above, but accurate reconstruction of sync pulse  
 1715 widths is not required, so a single *Sync Event* is substituted.
- 1716 • Burst mode – RGB pixel packets are time-compressed, leaving more time during a scan line for  
 1717 LP mode (saving power) or for multiplexing other transmissions onto the DSI link.

1718 Note that for accurate reconstruction of timing, packet overhead including Data ID, ECC, and Checksum  
 1719 bytes should be taken into consideration.

1720 The host processor shall support all of the packet sequences in this section. A Video Mode peripheral shall  
 1721 support at least one of the packet sequences in this section. The peripheral shall not require any additional  
 1722 constraints regarding packet sequence or packet timing. The peripheral supplier shall document all relevant  
 1723 timing parameters listed in Table 20.

1724 In the following figures the Blanking or Low-Power Interval (BLLP) is defined as a period during which  
 1725 video packets such as pixel-stream and sync event packets are not actively transmitted to the peripheral.

1726 To enable PHY synchronization the host processor should periodically end HS transmission and drive the  
 1727 Data Lanes to the LP state. This transition should take place at least once per frame; shown as LPM in the  
 1728 figures in this section. It is recommended to return to LP state once per scanline during the horizontal  
 1729 blanking time. Regardless of the frequency of BLLP periods, the host processor is responsible for meeting  
 1730 all documented peripheral timing requirements. Note, at lower frequencies BLLP periods will approach, or  
 1731 become, zero, and burst mode will be indistinguishable from non-burst mode.

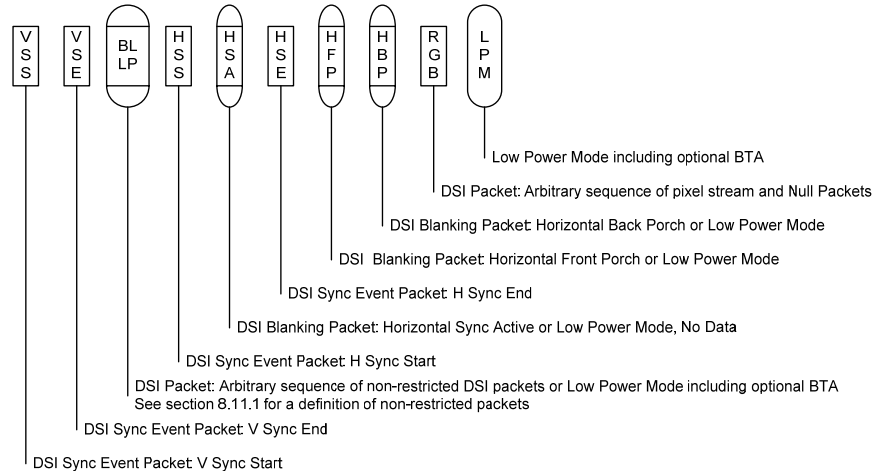
1732 During the BLLP the DSI Link may do any of the following:

- 1733 • Remain in Idle Mode with the host processor in LP-11 state and the peripheral in LP-RX
- 1734 • Transmit one or more non-video packets from the host processor to the peripheral using Escape  
 1735 Mode
- 1736 • Transmit one or more non-video packets from the host processor to the peripheral using HS Mode
- 1737 • If the previous processor-to-peripheral transmission ended with BTA, transmit one or more  
 1738 packets from the peripheral to the host processor using Escape Mode
- 1739 • Transmit one or more packets from the host processor to a different peripheral using a different  
 1740 Virtual Channel ID

1741 The sequence of packets within the BLLP or RGB portion of a HS transmission is arbitrary. The host  
 1742 processor may compose any sequence of packets, including iterations, within the limits of the packet format  
 1743 definitions. For all timing cases, the first line of a frame shall start with VSS; all other lines shall start with  
 1744 VSE or HSS. Note that the position of synchronization packets, such as VSS and HSS, in time is of utmost  
 1745 importance since this has a direct impact on the visual performance of the display panel.

1746 Normally, RGB pixel data is sent with one full scanline of pixels in a single packet. If necessary, a  
 1747 horizontal scanline of active pixels may be divided into two or more packets. However, individual pixels  
 1748 shall not be split across packets.

1749 Transmission packet components used in the figures in this section are defined in Figure 29 unless  
 1750 otherwise specified.



1751

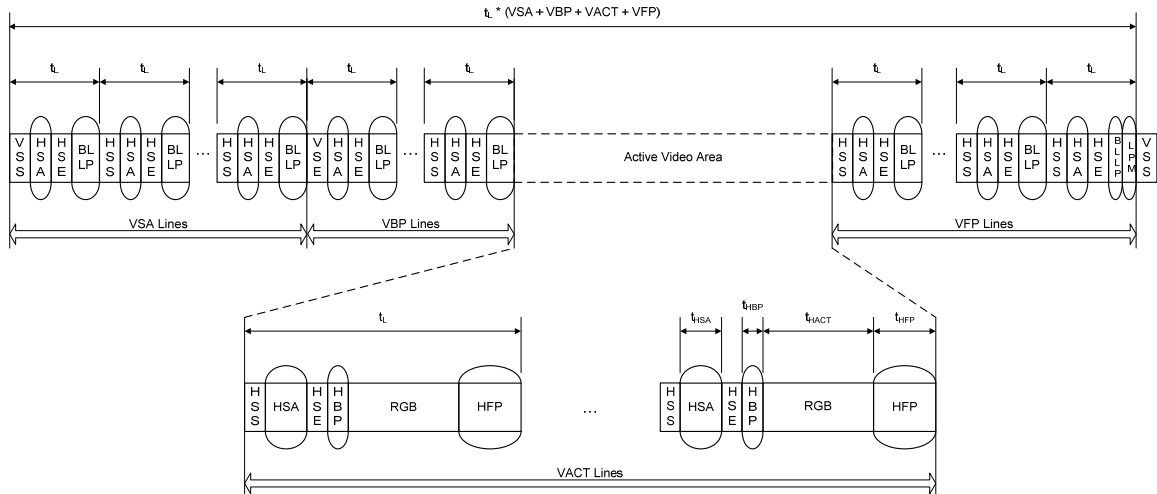
1752

**Figure 29 Video Mode Interface Timing Legend**

1753 If a peripheral timing specification for HBP or HFP minimum period is zero, the corresponding Blanking  
 1754 Packet may be omitted. If the HBP or HFP maximum period is zero, the corresponding blanking packet  
 1755 shall be omitted.

### 1756 8.11.2 Non-Burst Mode with Sync Pulses

1757 With this format, the goal is to accurately convey DPI-type timing over the DSI serial Link. This includes  
 1758 matching DPI pixel-transmission rates, and widths of timing events like sync pulses. Accordingly,  
 1759 synchronization periods are defined using packets transmitting both start and end of sync pulses. An  
 1760 example of this mode is shown in Figure 30.



1761  
1762

1763 **Figure 30 Video Mode Interface Timing: Non-Burst Transmission with Sync Start and End**

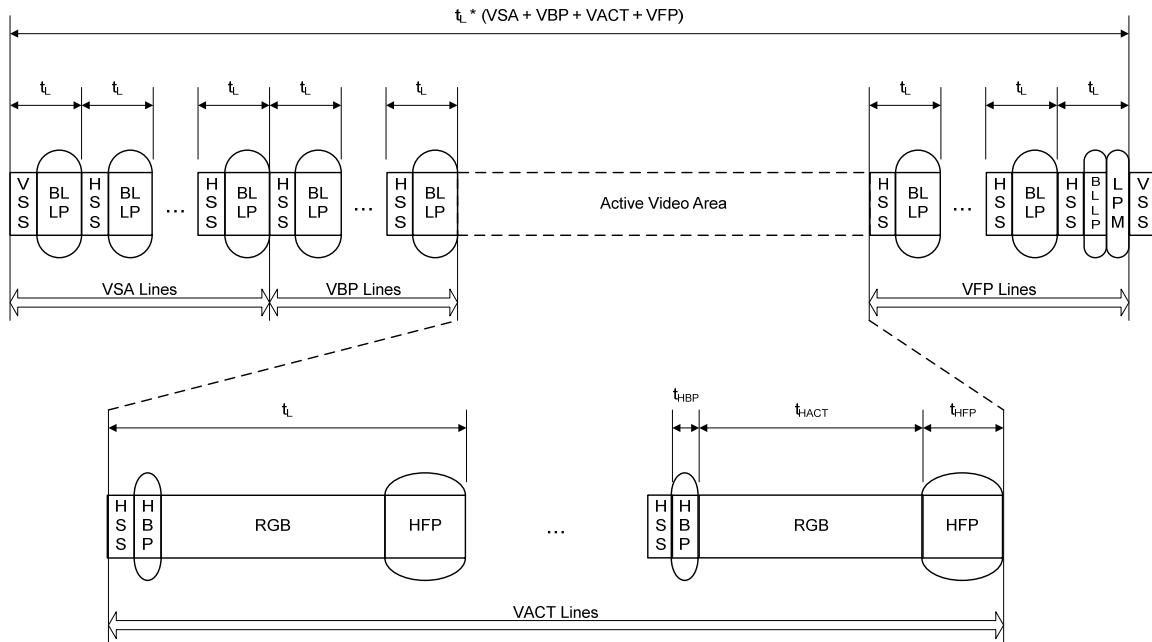
1764 Normally, periods shown as HSA (Horizontal Sync Active), HBP (Horizontal Back Porch) and HFP  
1765 (Horizontal Front Porch) are filled by Blanking Packets, with lengths (including packet overhead)  
1766 calculated to match the period specified by the peripheral's data sheet. Alternatively, if there is sufficient  
1767 time to transition from HS to LP mode and back again, a timed interval in LP mode may substitute for a  
1768 Blanking Packet, thus saving power. During HSA, HBP and HFP periods, the bus should stay in the LP-11  
1769 state.

1770 Refer to Annex C for the method of Video Mode interface timing for non-burst transmission with Sync  
1771 Start and Sync End sourcing interlaced video.

### 1772 8.11.3 Non-Burst Mode with Sync Events

1773 This mode is a simplification of the format described in section 8.11.2. Only the start of each  
1774 synchronization pulse is transmitted. The peripheral may regenerate sync pulses as needed from each Sync  
1775 Event packet received. Pixels are transmitted at the same rate as they would in a corresponding parallel  
1776 display interface such as DPI-2. An example of this mode is shown in Figure 31.





1777  
1778

1779 **Figure 31 Video Mode Interface Timing: Non-burst Transmission with Sync Events**

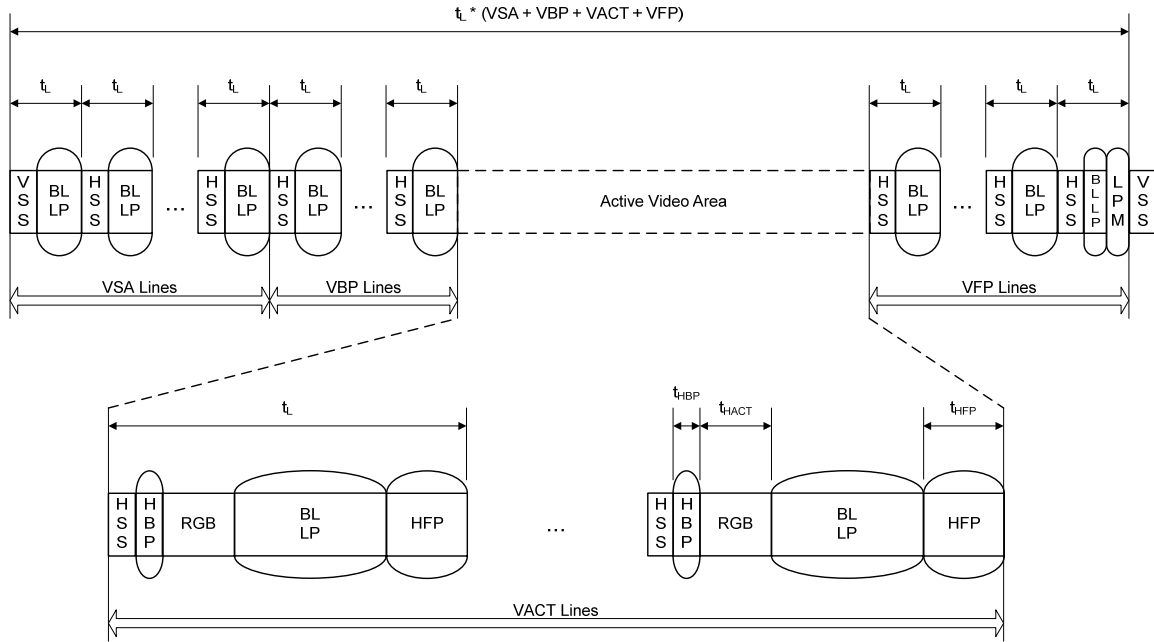
1780 As with the previous Non-Burst Mode, if there is sufficient time to transition from HS to LP mode and  
1781 back again, a timed interval in LP mode may substitute for a Blanking Packet, thus saving power.

1782 Refer to Annex C for the method of Video Mode interface timing for non-burst transmission with Sync  
1783 Events sourcing interlaced video.

#### 1784 **8.11.4 Burst Mode**

1785 In this mode, blocks of pixel data can be transferred in a shorter time using a time-compressed burst format.  
1786 This is a good strategy to reduce overall DSI power consumption, as well as enabling larger blocks of time  
1787 for other data transmissions over the Link in either direction.

1788 There may be a line buffer or similar memory on the peripheral to accommodate incoming data at high  
1789 speed. Following HS pixel data transmission, the bus may stay in HS Mode for sending blanking packets or  
1790 go to Low Power Mode, during which it may remain idle, i.e. the host processor remains in LP-11 state, or  
1791 LP transmission may take place in either direction. If the peripheral takes control of the bus for sending  
1792 data to the host processor, its transmission time shall be limited to ensure data underflow does not occur  
1793 from its internal buffer memory to the display device. An example of this mode is shown in Figure 32.



1794  
1795

1796

**Figure 32 Video Mode Interface Timing: Burst Transmission**

1797  
1798

Similar to the Non-Burst Mode scenario, if there is sufficient time to transition from HS to LP mode and back again, a timed interval in LP mode may substitute for a Blanking Packet, thus saving power.

1799

**8.11.5 Parameters**

1800  
1801  
1802

Table 20 documents the parameters used in the preceding figures. Peripheral supplier companies are responsible for specifying suitable values for all blank fields in the table. The host processor shall meet these requirements to ensure interoperability.

1803  
1804  
1805

For periods when Data Lanes are in LP Mode, the peripheral shall also specify whether the DSI Clock Lane may go to LP. The host processor is responsible for meeting minimum timing relationships between clock activity and HS transmission on the Data Lanes as documented in [MIPI04].

1806

**Table 20 Required Peripheral Timing Parameters**

Parameter	Description	Minimum	Maximum	Units	Comment
$b_{r_{PHY}}$	Bit rate total on all Lanes			Mbps	Depends on PHY implementation
$t_L$	Line time			$\mu s$	Define range to meet frame rate
$t_{HSA}$	Horizontal sync active			$\mu s$	
$t_{HBP}$	Horizontal back porch			$\mu s$	
$t_{HACT}$	Time for image data			$\mu s$	Defining min = 0 allows max PHY speed
HACT	Active pixels per line			pixels	

Parameter	Description	Minimum	Maximum	Units	Comment
t <sub>HFP</sub>	Horizontal front porch			μs	No upper limit as long as line time is met
VSA	Vertical sync active			lines	Number of lines in the vertical sync area
VBP	Vertical back porch			lines	
VACT	Active lines per frame			lines	
VFP	Vertical front porch			lines	

## 1807 8.12 TE Signaling in DSI

1808 A Command Mode display module has its own timing controller and local frame buffer for display refresh.  
 1809 In some cases the host processor needs to be notified of timing events on the display module, e.g. the start  
 1810 of vertical blanking or similar timing information. In a traditional parallel-bus interface like DBI-2, a  
 1811 dedicated signal wire labeled TE (Tearing Effect) is provided to convey such timing information to the host  
 1812 processor. In a DSI system, the same information, with reasonably low latency, shall be transmitted from  
 1813 the display module to the host processor when requested, using the bidirectional Data Lane.

1814 The PHY for DSI has no inherent interrupt capability from peripheral to host processor so the host  
 1815 processor shall either rely on polling, or it shall give bus ownership to the peripheral for extended periods,  
 1816 as it does not know when the peripheral will send the TE message.

1817 For polling to the display module, the host processor shall detect the current scan line information with a  
 1818 DCS command such as get\_scan\_line to avoid Tearing Effects. For TE-reporting from the display module,  
 1819 the TE-reporting function is enabled and disabled by three DCS commands to the display module's  
 1820 controller: set\_tear\_on, set\_tear\_scanline, and set\_tear\_off. See [MIPI01] for details.

1821 set\_tear\_on and set\_tear\_scanline are sent to the display module as DSI Data Type 0x15 (DCS Short Write,  
 1822 one parameter) and DSI Data Type 0x39 (DCS Long Write/write\_LUT), respectively. The host processor  
 1823 ends the transmission with Bus Turn-Around asserted, giving bus possession to the display module. Since  
 1824 the display module's DSI Protocol layer does not interpret DCS commands, but only passes them through  
 1825 to the display controller, it responds with a normal Acknowledge and returns bus possession to the host  
 1826 processor. In this state, the display module cannot report TE events to the host processor since it does not  
 1827 have bus possession.

1828 To enable TE-reporting, the host processor shall give bus possession to the display module without an  
 1829 accompanying DSI command transmission after TE reporting has been enabled. This is accomplished by  
 1830 the host processor's protocol logic asserting (internal) Bus Turn-Around signal to its D-PHY functional  
 1831 block. The PHY layer will then initiate a Bus Turn-Around sequence in LP mode, which gives bus  
 1832 possession to the display module.

1833 Since the timing of a TE event is, by definition, unknown to the host processor, the host processor shall  
 1834 give bus possession to the display module and then wait for up to one video frame period for the TE  
 1835 response. During this time, the host processor cannot send new commands, or requests to the display  
 1836 module, because it does not have bus possession.

1837 When the TE event takes place the display module shall send TE event information in LP mode using a  
 1838 specified trigger message available with D-PHY protocol via the following sequence:

- 1839 • The display module shall send the LP Escape Mode sequence

1840           • The display module shall then send the trigger message byte 01011101 (shown here in first bit to  
1841           last bit sequence)

1842           • The display module shall then return bus possession to the host processor

1843 This Trigger Message is reserved by DSI for TE signaling only and shall not be used for any other purpose  
1844 in a DSI-compliant interface.

1845 See [MIPI01] for detailed descriptions of the TE related commands, and command and parameter formats.  
1846

## 1847 9 Error-Correcting Code (ECC) and Checksum

### 1848 9.1 Packet Header Error Detection/Correction

1849 The host processor in a DSI-based system shall generate an error-correction code (ECC) and append it to  
 1850 the header of every packet sent to the peripheral. The ECC takes the form of a single byte following the  
 1851 header bytes. The ECC byte shall provide single-bit error correction and 2-bit error detection for the entire  
 1852 Packet Header. See Figure 13 and Figure 14 for descriptions of the Long and Short Packet Headers,  
 1853 respectively.

1854 ECC shall always be generated and appended in the Packet Header from the host processor. Peripherals  
 1855 with Bidirectional Links shall also generate and send ECC.

1856 Peripherals in unidirectional DSI systems, although they cannot report errors to the host, shall still take  
 1857 advantage of ECC for correcting single-bit errors in the Packet Header.

### 1858 9.2 Hamming Code Theory

1859 The number of parity or error check bits required is given by the Hamming rule, and is a function of the  
 1860 number of bits of information transmitted. The Hamming rule is expressed by the following inequality:

1861 
$$d + p + 1 <= 2^p$$
 where  $d$  is the number of data bits and  $p$  is the number of parity bits.

1862 The result of appending the computed parity bits to the data bits is called the Hamming code word. The size  
 1863 of the code word  $c$  is  $d+p$ , and a Hamming code word is described by the ordered set  $(c, d)$ .

1864 A Hamming code word is generated by multiplying the data bits by a generator matrix  $\mathbf{G}$ . This  
 1865 multiplication's result is called the code word vector  $(c1, c2, c3, \dots, cn)$ , consisting of the original data bits  
 1866 and the calculated parity bits. The generator matrix  $\mathbf{G}$  used in constructing Hamming codes consists of  $\mathbf{I}$ ,  
 1867 the identity matrix, and a parity generation matrix  $\mathbf{A}$ :

1868 
$$\mathbf{G} = [ \mathbf{I} \mid \mathbf{A} ]$$

1869 The Packet Header plus the ECC code can be obtained as:  $\text{PH} = \text{p} * \mathbf{G}$  where  $\text{p}$  represents the header and  $\mathbf{G}$  is  
 1870 the corresponding generator matrix.

1871 Validating the received code word  $r$  involves multiplying it by a parity check to form  $s$ , the syndrome or  
 1872 parity check vector:  $s = \mathbf{H} * \text{PH}$  where  $\text{PH}$  is the received Packet Header and  $\mathbf{H}$  is the parity check matrix:

1873 
$$\mathbf{H} = [ \mathbf{A}^T \mid \mathbf{I} ]$$

1874 If all elements of  $s$  are zero, the code word was received correctly. If  $s$  contains non-zero elements, then at  
 1875 least one error is present. If the header has a single-bit error, then the syndrome  $s$  matches one of the  
 1876 elements of  $\mathbf{H}$ , which will point to the bit in error. Furthermore, if the bit in error is a parity bit, then the  
 1877 syndrome will be one of the elements on  $\mathbf{I}$ , or else it will be the data bit identified by the position of the  
 1878 syndrome in  $\mathbf{A}^T$ .

### 1879 9.3 Hamming-modified Code Applied to DSI Packet Headers

1880 Hamming codes use parity to correct a single-bit error or detect a two-bit error, but are not capable of doing  
 1881 both simultaneously. DSI uses Hamming-modified codes where an extra parity bit is used to support both  
 1882 single error correction as well as two-bit error detection. For example a 7+1 bit Hamming-modified code  
 1883 (72, 64) allows for protection of up to 64 data bits. DSI systems shall use a 5+1 bit Hamming-modified  
 1884 code (30, 24), allowing for protection of up to twenty-four data bits. The addition of a parity bit allows a  
 1885 five bit Hamming code to correct a single-bit error and detect a two-bit error simultaneously.

1886 Since Packet Headers are fixed at four bytes (twenty-four data bits and eight ECC bits), P6 and P7 of the  
 1887 ECC byte are unused and shall be set to zero by the transmitter. The receiver shall ignore P6 and P7 and set  
 1888 both bits to zero before processing ECC. Table 21 shows a compact way to specify the encoding of parity  
 1889 and decoding of syndromes.

1890 **Table 21 ECC Syndrome Association Matrix**

	d2d1d0							
d5d4d3	0b000	0b001	0b010	0b011	0b100	0b101	0b110	0b111
0b000	0x07	0x0B	0x0D	0x0E	0x13	0x15	0x16	0x19
0b001	0x1A	0x1C	0x23	0x25	0x26	0x29	0x2A	0x2C
0b010	0x31	0x32	0x34	0x38	0x1F	0x2F	0x37	0x3B
0b011	0x43	0x45	0x46	0x49	0x4A	0x4C	0x51	0x52
0b100	0x54	0x58	0x61	0x62	0x64	0x68	0x70	0x83
0b101	0x85	0x86	0x89	0x8A	0x3D	0x3E	0x4F	0x57
0b110	0x8C	0x91	0x92	0x94	0x98	0xA1	0xA2	0xA4
0b111	0xA8	0xB0	0xC1	0xC2	0xC4	0xC8	0xD0	0xE0

1891 Each cell in the matrix represents a syndrome and each syndrome in the matrix is MSB left aligned:

1892 e.g. 0x07=0b0000\_0111=P7P6P5P4P3P2P1P0

1893 The top row defines the three LSB of data position bit, and the left column defines the three MSB of data  
 1894 position bit for a total of 64-bit positions.

1895 e.g. 38th bit position (D37) is encoded 0b100\_101 and has the syndrome 0x68.

1896 To correct a single bit error, the syndrome shall be one of the syndromes in the table, which will identify  
 1897 the bit position in error. The syndrome is calculated as:

1898  $S = P_{SEND} \wedge P_{RECEIVED}$  where  $P_{SEND}$  is the 6-bit ECC field in the header and  $P_{RECEIVED}$  is the  
 1899 calculated parity of the received header.

1900 Table 22 represents the same information as in Table 21, organized to provide better insight into how parity  
 1901 bits are formed from data bits.

1902

Table 22 ECC Parity Generation Rules

Data Bit	P7	P6	P5	P4	P3	P2	P1	P0	Hex
0	0	0	0	0	0	1	1	1	0x07
1	0	0	0	0	1	0	1	1	0x0B
2	0	0	0	0	1	1	0	1	0x0D
3	0	0	0	0	1	1	1	0	0x0E
4	0	0	0	1	0	0	1	1	0x13
5	0	0	0	1	0	1	0	1	0x15
6	0	0	0	1	0	1	1	0	0x16
7	0	0	0	1	1	0	0	1	0x19
8	0	0	0	1	1	0	1	0	0x1A
9	0	0	0	1	1	1	0	0	0x1C
10	0	0	1	0	0	0	1	1	0x23
11	0	0	1	0	0	1	0	1	0x25
12	0	0	1	0	0	1	1	0	0x26
13	0	0	1	0	1	0	0	1	0x29
14	0	0	1	0	1	0	1	0	0x2A
15	0	0	1	0	1	1	0	0	0x2C
16	0	0	1	1	0	0	0	1	0x31
17	0	0	1	1	0	0	1	0	0x32
18	0	0	1	1	0	1	0	0	0x34
19	0	0	1	1	1	0	0	0	0x38
20	0	0	0	1	1	1	1	1	0x1F
21	0	0	1	0	1	1	1	1	0x2F
22	0	0	1	1	0	1	1	1	0x37
23	0	0	1	1	1	0	1	1	0x3B
24	0	1	0	0	0	0	1	1	0x43
25	0	1	0	0	0	1	0	1	0x45
26	0	1	0	0	0	1	1	0	0x46
27	0	1	0	0	1	0	0	1	0x49
28	0	1	0	0	1	0	1	0	0x4A
29	0	1	0	0	1	1	0	0	0x4C
30	0	1	0	1	0	0	0	1	0x51
31	0	1	0	1	0	0	1	0	0x52
32	0	1	0	1	0	1	0	0	0x54
33	0	1	0	1	1	0	0	0	0x58

Data Bit	P7	P6	P5	P4	P3	P2	P1	P0	Hex
34	0	1	1	0	0	0	0	1	0x61
35	0	1	1	0	0	0	1	0	0x62
36	0	1	1	0	0	1	0	0	0x64
37	0	1	1	0	1	0	0	0	0x68
38	0	1	1	1	0	0	0	0	0x70
39	1	0	0	0	0	0	1	1	0x83
40	1	0	0	0	0	1	0	1	0x85
41	1	0	0	0	0	1	1	0	0x86
42	1	0	0	0	1	0	0	1	0x89
43	1	0	0	0	1	0	1	0	0x8A
44	0	0	1	1	1	1	0	1	0x3D
45	0	0	1	1	1	1	1	0	0x3E
46	0	1	0	0	1	1	1	1	0x4F
47	0	1	0	1	0	1	1	1	0x57
48	1	0	0	0	1	1	0	0	0x8C
49	1	0	0	1	0	0	0	1	0x91
50	1	0	0	1	0	0	1	0	0x92
51	1	0	0	1	0	1	0	0	0x94
52	1	0	0	1	1	0	0	0	0x98
53	1	0	1	0	0	0	0	1	0xA1
54	1	0	1	0	0	0	1	0	0xA2
55	1	0	1	0	0	1	0	0	0xA4
56	1	0	1	0	1	0	0	0	0xA8
57	1	0	1	1	0	0	0	0	0xB0
58	1	1	0	0	0	0	0	1	0xC1
59	1	1	0	0	0	0	1	0	0xC2
60	1	1	0	0	0	1	0	0	0xC4
61	1	1	0	0	1	0	0	0	0xC8
62	1	1	0	1	0	0	0	0	0xD0
63	1	1	1	0	0	0	0	0	0xE0

1903 To derive parity bit P3, the “ones” in the P3 column define if the corresponding bit position Di (as noted in  
 1904 the green column) is used in calculation of P3 parity bit or not. For example,

1905 
$$P3 = D1 \wedge D2 \wedge D3 \wedge D7 \wedge D8 \wedge D9 \wedge D13 \wedge D14 \wedge D15 \wedge D19 \wedge D20 \wedge D21 \wedge D23$$



1906 The first twenty-four data bits, D0 to D23, in Table 22 contain the complete DSI Packet Header. The  
 1907 remaining bits, D24 to D63, are informative (shown in yellow in the table) and not relevant to DSI.  
 1908 Therefore, the parity bit calculation can be optimized to:

1909  $P7=0$

1910  $P6=0$

1911  $P5=D10 \wedge D11 \wedge D12 \wedge D13 \wedge D14 \wedge D15 \wedge D16 \wedge D17 \wedge D18 \wedge D19 \wedge D21 \wedge D22 \wedge D23$

1912  $P4=D4 \wedge D5 \wedge D6 \wedge D7 \wedge D8 \wedge D9 \wedge D16 \wedge D17 \wedge D18 \wedge D19 \wedge D20 \wedge D22 \wedge D23$

1913  $P3=D1 \wedge D2 \wedge D3 \wedge D7 \wedge D8 \wedge D9 \wedge D13 \wedge D14 \wedge D15 \wedge D19 \wedge D20 \wedge D21 \wedge D23$

1914  $P2=D0 \wedge D2 \wedge D3 \wedge D5 \wedge D6 \wedge D9 \wedge D11 \wedge D12 \wedge D15 \wedge D18 \wedge D20 \wedge D21 \wedge D22$

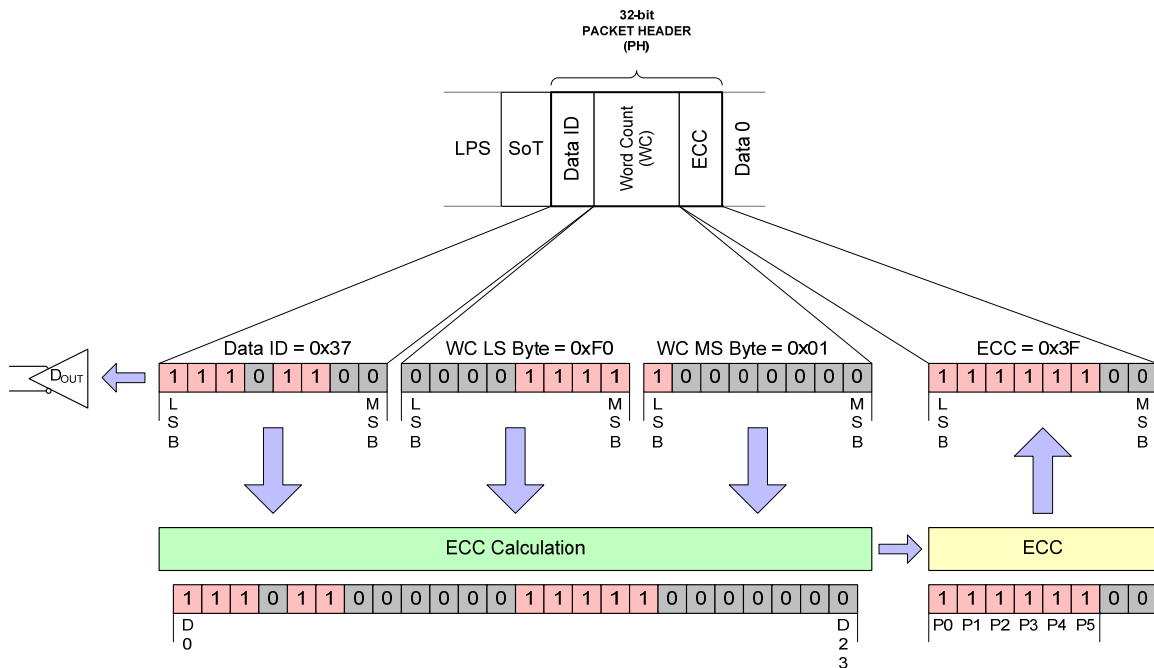
1915  $P1=D0 \wedge D1 \wedge D3 \wedge D4 \wedge D6 \wedge D8 \wedge D10 \wedge D12 \wedge D14 \wedge D17 \wedge D20 \wedge D21 \wedge D22 \wedge D23$

1916  $P0=D0 \wedge D1 \wedge D2 \wedge D4 \wedge D5 \wedge D7 \wedge D10 \wedge D11 \wedge D13 \wedge D16 \wedge D20 \wedge D21 \wedge D22 \wedge D23$

1917 Note, the parity bits relevant to the ECC calculation, P0 through P5, in the table are shown in red and the  
 1918 unused bits, P6 and P7, are shown in blue.

1919 **9.4 ECC Generation on the Transmitter**

1920 ECC is generated from the twenty-four data bits within the Packet Header as illustrated in Figure 33, which  
 1921 also serves as an ECC calculation example. Note that the DSI protocol uses a four byte Packet Header. See  
 1922 section 8.4.1 and section 8.4.2 for Packet Header descriptions for Long and Short packets, respectively.



1923  
 1924

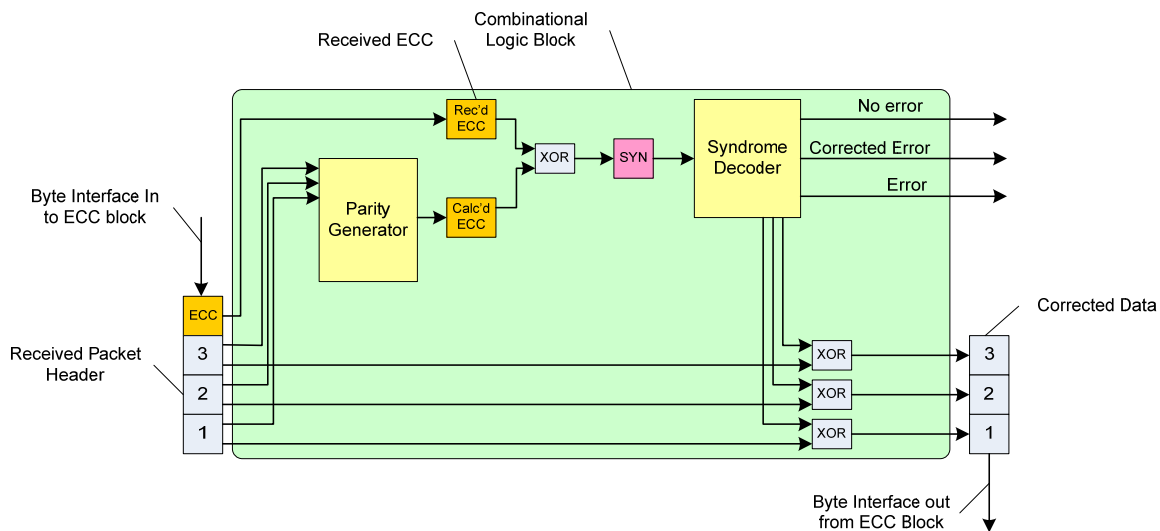
1925

**Figure 33 24-bit ECC generation on TX side**

## 1926 9.5 Applying ECC on the Receiver

1927 Applying ECC on the receiver involves generating a new ECC for the received packet, computing the  
 1928 syndrome using the new ECC and the received ECC, decoding the syndrome to find if a single-error has  
 1929 occurred and if so, correcting the error. If a multiple-bit error is identified, it is flagged and reported to the  
 1930 transmitter. Note, error reporting is only applicable to bidirectional DSI implementations.

1931 ECC generation on the receiver side shall apply the same padding rules as ECC generation for  
 1932 transmission.



1933

1934

**Figure 34 24-bit ECC on RX Side Including Error Correction**

1935 Decoding the syndrome has three aspects:

- 1936 • Testing for errors in the Packet Header. If syndrome = 0, no errors are present.
- 1937 • Test for a single-bit error in the Packet Header by comparing the generated syndrome with the  
 1938 matrix in Table 21. If the syndrome matches one of the entries in the table, then a single-bit error  
 1939 has occurred and the corresponding bit is in error. This position in the Packet Header shall be  
 1940 complemented to correct the error. Also, if the syndrome is one of the rows of the identity matrix  
 1941 **I**, then a parity bit is in error. If the syndrome cannot be identified then a multi-bit error has  
 1942 occurred. In this case the Packet Header is corrupted and cannot be restored. Therefore, the Multi-  
 1943 bit Error Flag shall be set.
- 1944 • Correcting the single-bit error if detected, as indicated above.

## 1945 9.6 Checksum Generation for Long Packet Payloads

1946 Long packets are comprised of a Packet Header protected by an ECC byte as specified in sections 9.3  
 1947 through 9.5, and a payload of 0 to  $2^{16} - 1$  bytes. To detect errors in transmission of Long packets, a  
 1948 checksum is calculated over the payload portion of the data packet. Note that, for the special case of a zero-  
 1949 length payload, the 2-byte checksum is set to 0xFFFF.

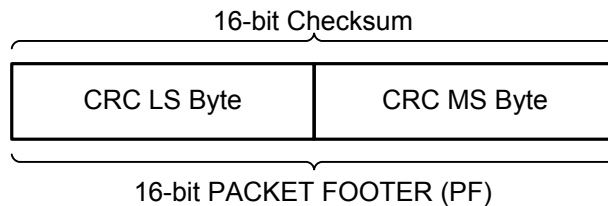
1950 The checksum can only indicate the presence of one or more errors in the payload. Unlike ECC, the  
 1951 checksum does not enable error correction. For this reason, checksum calculation is not useful for  
 1952 unidirectional DSI implementations since the peripheral has no means of reporting errors to the host  
 1953 processor.

1954 Checksum generation and transmission is mandatory for host processors sending Long packets to  
 1955 peripherals. It is optional for peripherals transmitting Long packets to the host processor. However, the  
 1956 format of Long packets is fixed; peripherals that do not support checksum generation shall transmit two  
 1957 bytes having value 0x0000 in place of the checksum bytes when sending Long packets to the host  
 1958 processor.

1959 The host processor shall disable checksum checking for received Long packets from peripherals that do not  
 1960 support checksum generation.

1961 The checksum shall be realized as a 16-bit CRC with a generator polynomial of  $x^{16}+x^{12}+x^5+x^0$ .

1962 The transmission of the checksum is illustrated in Figure 35. The LS byte is sent first, followed by the MS  
 1963 byte. Note that within the byte, the LS bit is sent first.

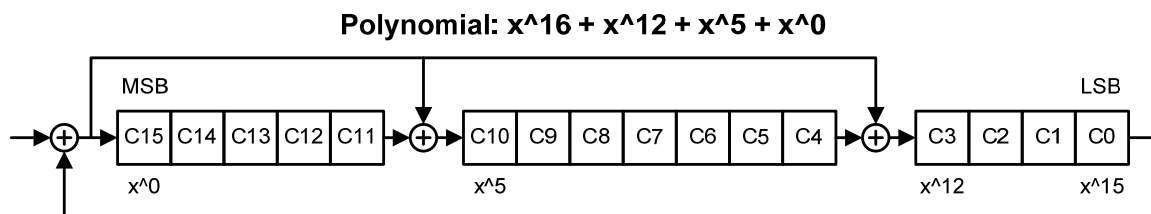


1964

1965

**Figure 35 Checksum Transmission**

1966 The CRC implementation is presented in Figure 36. The CRC shift register shall be initialized to 0xFFFF  
 1967 before packet data enters. Packet data not including the Packet Header then enters as a bitwise data stream  
 1968 from the left, LS bit first. Each bit is fed through the CRC shift register before it is passed to the output for  
 1969 transmission to the peripheral. After all bytes in the packet payload have passed through the CRC shift  
 1970 register, the shift register contains the checksum. C15 contains the checksum's MSB and C0 the LSB of the  
 1971 16-bit checksum. The checksum is then appended to the data stream and sent to the receiver. The receiver  
 1972 uses its own generated CRC to verify that no errors have occurred in transmission. See Annex B for an  
 1973 example of checksum generation.



1974

1975

1976

**Figure 36 16-bit CRC Generation Using a Shift Register**

1977 Section 8.10.1 documents the peripheral response to detection of an error in a Long packet payload.

1978

## 1979 **10 Compliance, Interoperability, and Optional Capabilities**

1980 This section documents requirements and classifications for MIPI-compliant host processors and  
 1981 peripherals. There are a number of categories of potential differences or attributes that shall be considered  
 1982 to ensure interoperability between a host processor and a peripheral, such as a display module:

1983 Manufacturers shall document a DSI device's capabilities and specifications for the parameters listed in this  
 1984 section.

- 1985 1. Display Resolutions
- 1986 2. Pixel Formats
- 1987 3. Number of Lanes
- 1988 4. Maximum Lane Frequency
- 1989 5. Bidirectional Communication and Escape Mode Support
- 1990 6. ECC and Checksum capabilities
- 1991 7. Display Architecture
- 1992 8. Multiple Peripheral Support

1993 EoTp support and interoperability between DSI v1.01-compliant and earlier revision devices are discussed  
 1994 in section 10.9.

1995 In general, the peripheral chooses one option from each category in the list above. For example, a display  
 1996 module may implement a resolution of 320x240 (QVGA), a pixel format of 16-bpp and use two Lanes to  
 1997 achieve its required bandwidth. Its data path has bidirectional capability, it does not implement  
 1998 checksum-testing capability, and it operates in Video Mode only.

### 1999 **10.1 Display Resolutions**

2000 Host processors shall implement one or more of the display resolutions in Table 23.

2001

**Table 23 Display Resolutions**

Resolution	Horizontal Extent	Vertical Extent
QQVGA	160	120
QCIF	176	144
QCIF+	176	208
QCIF+	176	220
QVGA	320	240
CIF	352	288
CIF+	352	416

Resolution	Horizontal Extent	Vertical Extent
CIF+	352	440
(1/2)VGA	320	480
(2/3)VGA	640	320
VGA	640	480
WVGA	800	480
SVGA	800	600
XVGA	1024	768

2002 **10.2 Pixel Formats**

2003 Pixel formats for Video Mode and Command Mode are defined in the following sections.

2004 **10.2.1 Video Mode**

2005 Peripherals shall implement at least one of the following pixel formats. Host processors shall implement all  
2006 of the following pixel formats; all other Video Mode formats in section 8.8 are optional:

- 2007 1. 16 bpp (5, 6, 5 RGB), each pixel using two bytes; see section 8.8.20
- 2008 2. 18 bpp (6, 6, 6 RGB) packed; see section 8.8.21
- 2009 3. 18 bpp (6, 6, 6 RGB) loosely packed into three bytes; see section 8.8.22
- 2010 4. 24 bpp (8, 8, 8 RGB), each pixel using three bytes; see section 8.8.23

2011 **10.2.2 Command Mode**

2012 Peripherals shall implement at least one of the pixel formats, and host processors should implement all of  
2013 the pixel formats, defined in [MIPI01].

2014 **10.3 Number of Lanes**

2015 In normal operation a peripheral uses the number of Lanes required for its bandwidth needs.

2016 The host processor shall implement a minimum of one Data Lane; additional Lane capability is optional. A  
2017 host processor with multi-Lane capability (N Lanes) shall be able to operate with any number of Lanes  
2018 from one to N, to match the fixed number of Lanes in peripherals using one to N Lanes. See section 6.1 for  
2019 more details.

2020 **10.4 Maximum Lane Frequency**

2021 The maximum Lane frequency shall be documented by the DSI device manufacturer. The Lane frequency  
2022 shall adhere to the specifications in [MIPI04].

## 2023 **10.5 Bidirectional Communication**

2024 Because Command Mode depends on the use of the READ command, a Command Mode display module  
2025 shall implement bidirectional communications. For display modules without on-panel buffers that work  
2026 only in Video Mode, bidirectional operation on DSI is optional.

2027 Since a host processor may implement both Command- and Video Modes of operations, it should support  
2028 bidirectional operation and Escape Mode transmission and reception.

## 2029 **10.6 ECC and Checksum Capabilities**

2030 A DSI host processor shall calculate and transmit an ECC byte for both Long and Short packets. The host  
2031 processor shall also calculate and transmit a two-byte Checksum for Long packets. A DSI peripheral shall  
2032 support ECC, but may support Checksum. If a peripheral does not calculate Checksum it shall still be  
2033 capable of receiving Checksum bytes from the host processor. If a peripheral supports bidirectional  
2034 communications and does not support Checksum it shall send bytes of all zeros in the appropriate fields.  
2035 For interoperability with earlier revision of DSI peripherals where ECC was considered an optional feature,  
2036 host shall be able to enable/disable ECC capability based on the particular peripheral ECC support  
2037 capability. The enabling/disabling mechanism is out of scope of DSI. In effect, if an earlier revision  
2038 peripheral was not supporting ECC, it shall still be capable of receiving ECC byte from the host and  
2039 sending an all zero ECC byte back to the host for responses over a bidirectional link. See section 9 for more  
2040 details on ECC and Checksum.

## 2041 **10.7 Display Architecture**

2042 A display module may implement Type 1, Type 2, Type 3 or Type 4 display architecture as described in  
2043 [MIPI02] and [MIPI03]. Type 1 architecture works in Command Mode only. Type 2 and Type 3  
2044 architectures use the DSI interface for both Command- and Video Modes of operation. Type 4 architectures  
2045 operate in Video Mode only, although there may be additional control signals. Therefore, a peripheral may  
2046 use Command Mode only, Video Mode only, or both Command- and Video Modes of operation.

2047 The host processor may support either or both Command- and Video Modes of operation. If the host  
2048 processor supports Command Mode, it shall also support the mandatory command set specified in  
2049 [MIPI01].

## 2050 **10.8 Multiple Peripheral Support**

2051 DSI supports multiple peripherals per DSI Link using the Virtual Channel field of the Data Identifier byte.  
2052 See sections 4.2.3 and 8.5.1 for more details.

2053 A host processor should support a minimum of two peripherals.

## 2054 **10.9 EoTp Support and Interoperability**

2055 EoTp generation or detection is mandatory for devices compliant with this version of the DSI specification.  
2056 Devices compliant to DSI specification v1.0 and earlier do not support EoTp. In order to ensure  
2057 interoperability with earlier devices, current devices shall provide a means to enable or disable EoTp  
2058 generation or detection. In effect, this capability can be disabled by the system designer whenever a device  
2059 on either side of the Link does not support EoTp.  
2060

## 2061 **Annex A Contention Detection and Recovery Mechanisms (informative)**

2062 The following describes optional capabilities at the PHY and Protocol layers that provide additional  
 2063 robustness for a DSI Link against possible data-signal contention as a consequence of transient errors in the  
 2064 system. These capabilities improve the system's chances of detecting any of several possible contention  
 2065 cases, and provide mechanisms for "graceful" recovery without resorting to a hard reset.

2066 These capabilities combine circuitry in the I/O cell, to directly detect contention, with logic and timers in  
 2067 the protocol to avert and recover from other forms of contention.

### 2068 **A.1 PHY Detected Contention**

2069 The PHY can detect two types of contention faults: LP High Fault and LP Low Fault.

2070 The LP High Fault and LP Low Fault are caused by both sides of the Link transmitting simultaneously.  
 2071 Note, the LP High Fault and LP Low Fault are only applicable for bidirectional Data Lanes. Refer to  
 2072 [MIPI04] for definition of LP High and LP Low faults.

#### 2073 **A.1.1 Protocol Response to PHY Detected Faults**

2074 The Protocol shall specify how both ends of the Link respond when contention is flagged. It shall ensure  
 2075 that both devices return to *Stop* state (LP-11), with one side going to *Stop TX* and the other to *Stop RX*.

2076 When both PHYs are in LP mode, one or both PHYs will detect contention between LP-0 and LP-1.

2077 The following tables describe the resolution sequences for different types of contention and detection.

2078 Table sequences:

- 2079 • Sequence of events to resolve LP High  $\leftrightarrow$  LP Low Contention
- 2080 • Case 1: Both sides initially detect the contention
- 2081 • Case 2: Only the Host Processor initially detects contention
- 2082 • Case 3: Only the Peripheral initially detects contention

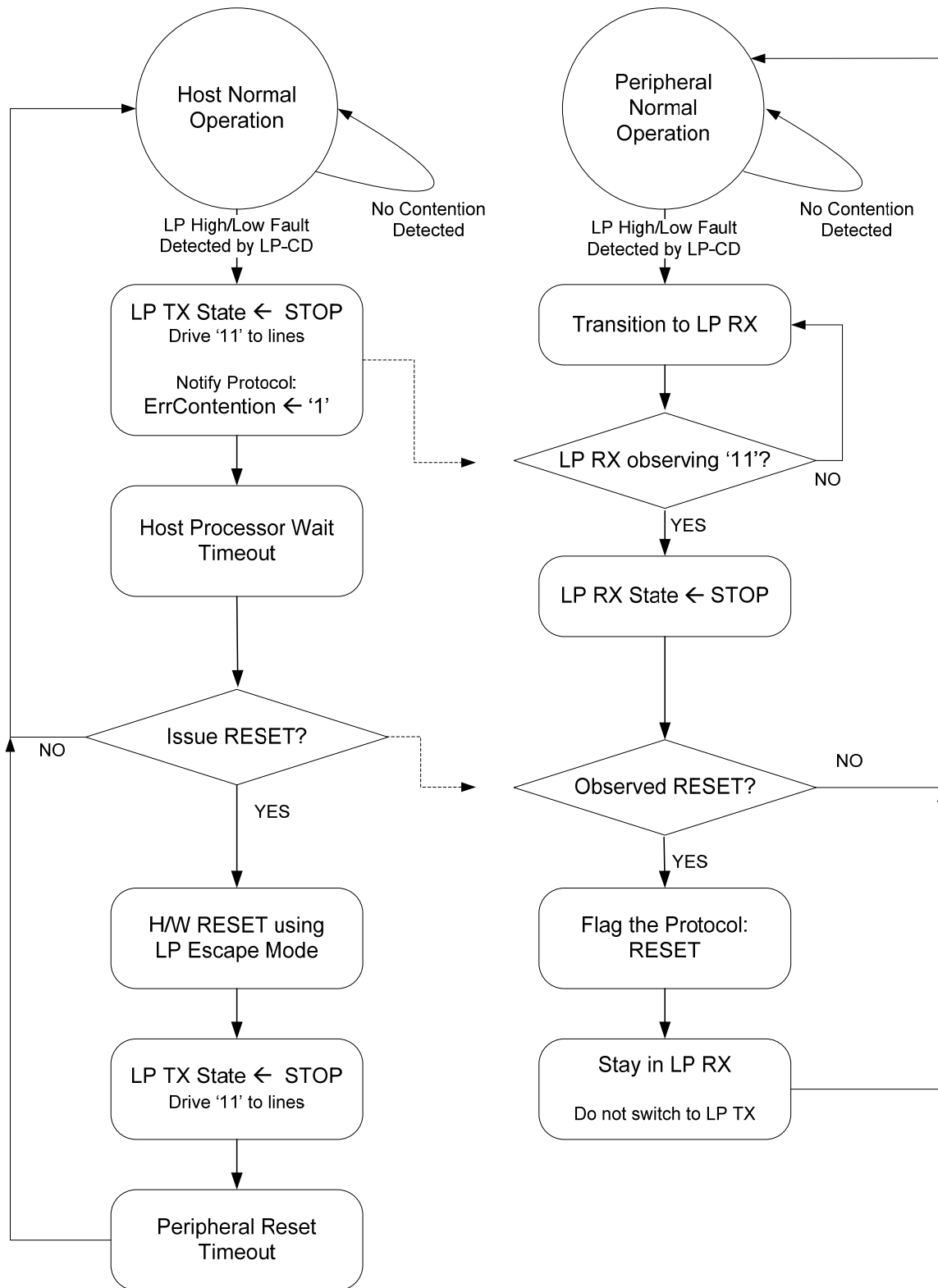
2083 **Table 24 LP High  $\leftrightarrow$  LP Low Contention Case 1**

Host Processor Side		Peripheral Side	
Protocol	PHY	PHY	Protocol
	Detect <i>LP High Fault</i> or <i>LP Low Fault</i>	Detect <i>LP High Fault</i> or <i>LP Low Fault</i>	
	Transition to <i>Stop</i> State (LP-11)	Transition to LP-RX	Set <i>Contention Detected</i> in Error Report (see Table 18)
Host Processor Wait Timeout		Peripheral waits until it observes <i>Stop</i> state before responding	
		Observe <i>Stop</i> state	
Request Reset Entry Command to PHY (optional)	Send Reset Entry Command	Observe Reset Entry Command	

Host Processor Side		Peripheral Side	
Protocol	PHY	PHY	Protocol
		Flag Protocol about Reset Command	Observe Reset Entry Command
			Reset Peripheral
	Return to Stop State (LP-11)	Remain in LP-RX	(reset may continue)
Peripheral Reset Timeout. Wait until Peripheral completes Reset before resuming normal operation.	Continue normal operation.		Reset completes

2084 Note: The protocol may want to request a Reset after contention is flagged a single time. Alternately, the  
2085 protocol may choose not to Reset but instead continue normal operation after detecting a single contention.  
2086 It could then initiate a Reset after multiple contentions are flagged, or never initiate a Reset.





2087  
2088

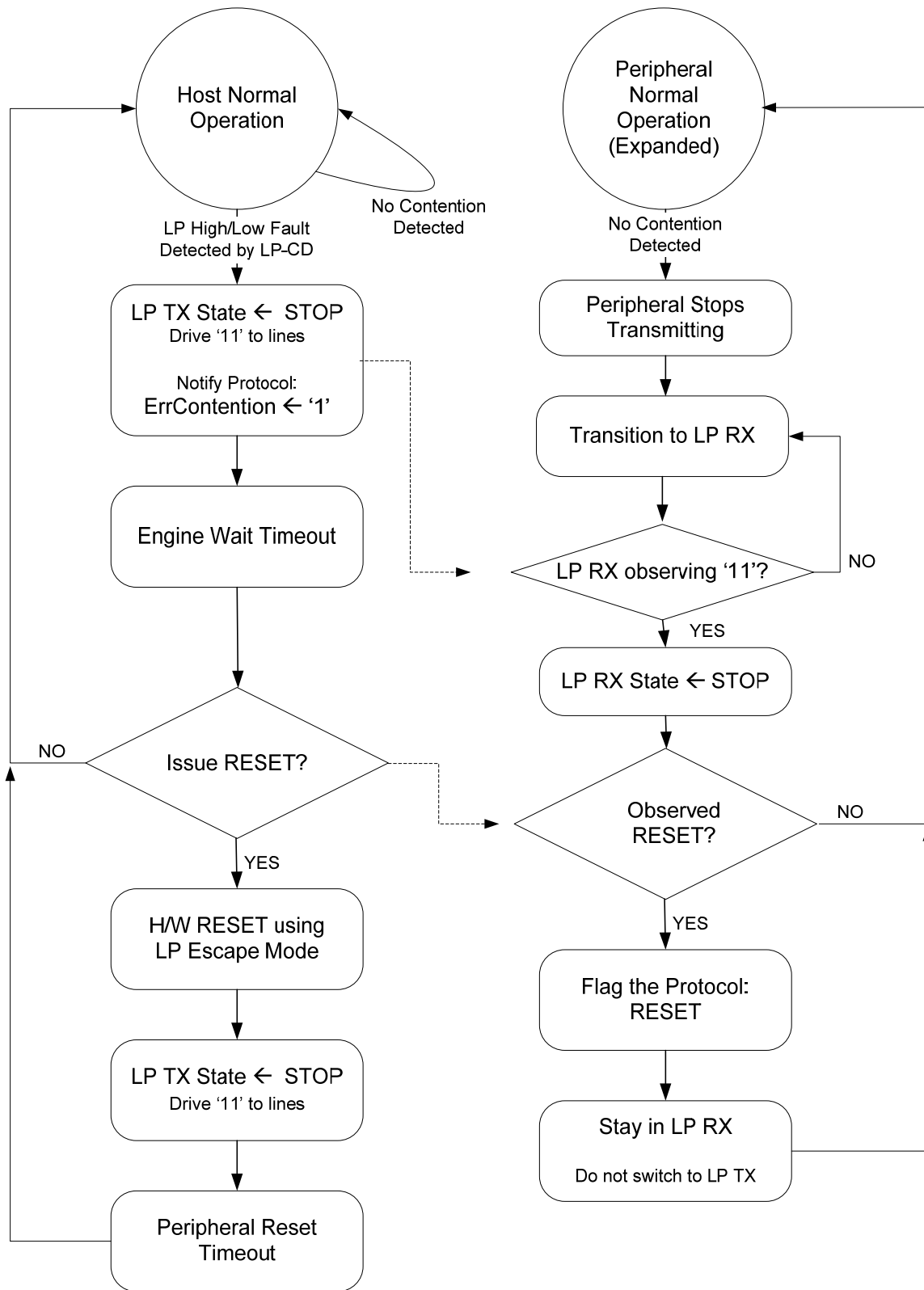
2089

**Figure 37 LP High ↔ LP Low Contention Case 1**

2090

Table 25 LP High ↔ LP Low Contention Case 2

Host Processor Side		Peripheral Side	
Protocol	PHY	PHY	Protocol
	Detect <i>LP High Fault</i> or <i>LP Low Fault</i>	No EL contention detected	
	Transition to <i>Stop State</i> (LP-11)	No EL contention detected	
Host Processor Wait Timeout			Peripheral Bus Possession Timeout
		Transition to LP-RX	
		Observe <i>Stop state</i>	
Request <i>Reset Entry</i> command to PHY	Send <i>Reset Entry</i> command	Observe <i>Reset Entry</i> command	
		Flag Protocol: <i>Reset</i> command received	Observe <i>Reset Command</i>
			Reset Peripheral
	Return to <i>Stop state</i> (LP-11)	Remain in LP-RX	(reset continues)
Peripheral Reset Timeout. Wait until peripheral completes Reset before resuming normal operation.	Continue normal operation.		Reset completes



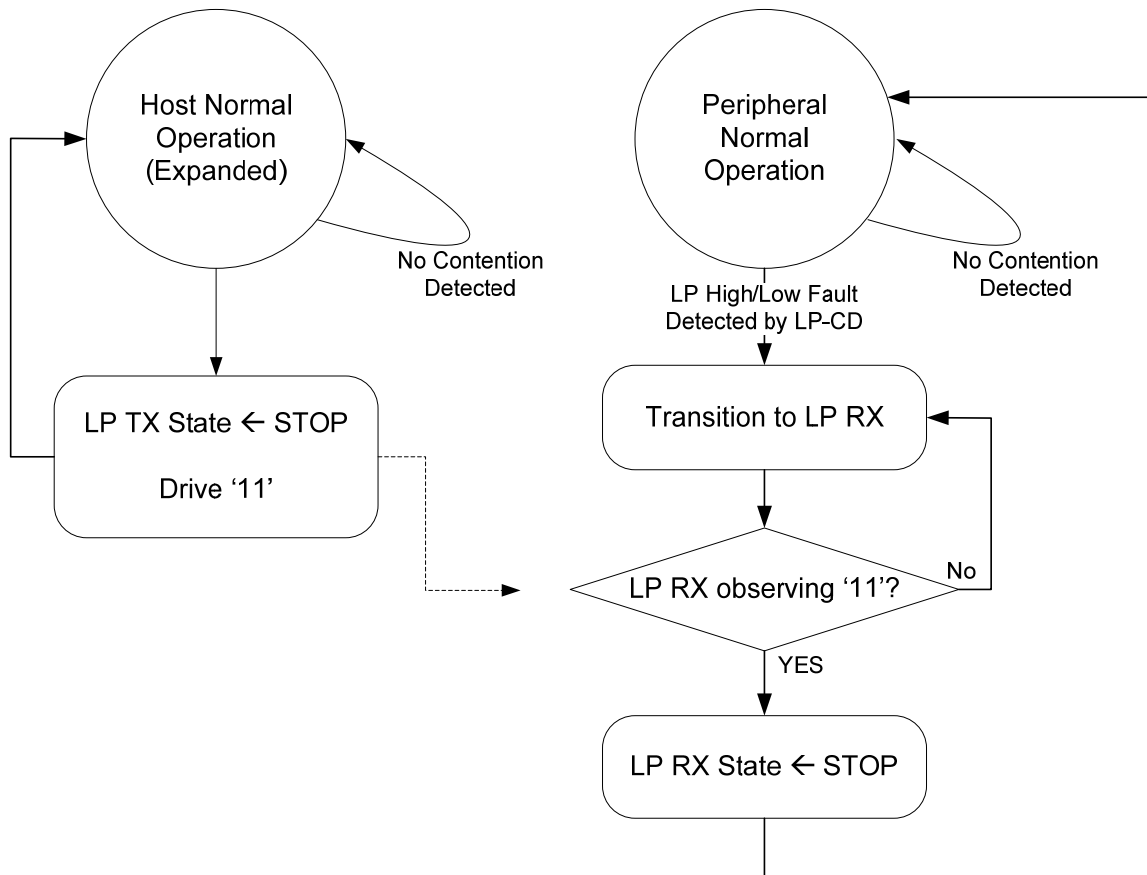
2091  
2092

Figure 38 LP High ↔ LP Low Contention Case 2

2093

**Table 26 LP High ↔ LP Low Contention Case 3**

Host Processor Side		Peripheral Side	
Protocol	PHY	PHY	Protocol
	No detection of EL contention	Detect LP High Fault or LP Low Fault	
		Transition to LP-RX	Set Contention Detected in Error Report (see Table 18)
		Peripheral waits until it observes Stop state before responding to bus activity.	
	Normal transition to Stop State (LP-11)	Observe Stop State	



2094

2095

2096

**Figure 39 LP High ↔ LP Low Contention Case 3**

2097 **Annex B Checksum Generation Example (informative)**

2098 The following C/C++ program provides a simple software routine to calculate CRC of a packet of variable  
 2099 length. The main routine calls subroutine CalculateCRC16 to calculate the CRC based on the data in  
 2100 one of the gpcTestData[ ] arrays and prints the CRC results.

2101

```

2102 /* ***** DECLARATIONS ***** */
2103 #include <stdio.h>
2104
2105 /* Start of Test Data */
2106 static unsigned char gpcTestData0[] = { 0x00 };
2107 static unsigned char gpcTestData1[] = { 0x01 };
2108 static unsigned char gpcTestData2[] = { 0xFF, 0x00, 0x00, 0x00, 0x1E,
2109     0xF0, 0x1E, 0xC7, 0x4F, 0x82, 0x78, 0xC5, 0x82, 0xE0, 0x8C, 0x70,
2110     0xD2, 0x3C, 0x78, 0xE9, 0xFF, 0x00, 0x00, 0x01 };
2111 static unsigned char gpcTestData3[] = { 0xFF, 0x00, 0x00, 0x02, 0xB9,
2112     0xDC, 0xF3, 0x72, 0xBB, 0xD4, 0xB8, 0x5A, 0xC8, 0x75, 0xC2, 0x7C,
2113     0x81, 0xF8, 0x05, 0xDF, 0xFF, 0x00, 0x00, 0x01 };
2114 #define NUMBER_OF_TEST_DATA0_BYTES 1
2115 #define NUMBER_OF_TEST_DATA1_BYTES 1
2116 #define NUMBER_OF_TEST_DATA2_BYTES 24
2117 #define NUMBER_OF_TEST_DATA3_BYTES 24
2118 /* End of Test Data */
2119
2120 unsigned short CalculateCRC16( unsigned char *pcDataStream, unsigned
2121 short sNumberOfDataBytes );
2122
2123 /* ***** MAIN ROUTINE ***** */
2124 void main( void )
2125 {
2126     unsigned short sCRC16Result;
2127     sCRC16Result = CalculateCRC16( gpcTestData2,
2128         NUMBER_OF_TEST_DATA2_BYTES );
2129     printf( "Checksum CS[15:0] = 0x%04X\n", sCRC16Result );
2130 }
2131 /* ***** END OF MAIN ***** START OF CRC CALCULATION ***** */
2132
2133 /* CRC16 Polynomial, logically inverted 0x1021 for x^16+x^15+x^5+x^0 */
2134 static unsigned short gsCRC16GenerationCode = 0x8408;
2135
2136 unsigned short CalculateCRC16( unsigned char *pcDataStream, unsigned
2137 short sNumberOfDataBytes )
2138 {
2139     /*
2140     sCRC16Result: the return of this function,
2141     sByteCounter: address pointer to count the number of the
2142     calculated data bytes
2143     cBitCounter: counter for bit shift (0 to 7)
2144     cCurrentData: byte size buffer to duplicate the calculated data
2145     byte for a bit shift operation
2146     */
2147     unsigned short sByteCounter;
  
```

```

2148     unsigned char cBitCounter;
2149     unsigned char cCurrentData;
2150     unsigned short sCRC16Result = 0xFFFF;
2151     if ( sNumberOfDataBytes > 0 )
2152     {
2153         for ( sByteCounter = 0; sByteCounter < sNumberOfDataBytes;
2154              sByteCounter++ )
2155         {
2156             cCurrentData = *( pcDataStream + sByteCounter );
2157             for ( cBitCounter = 0; cBitCounter < 8;
2158                  cBitCounter++ )
2159             {
2160                 if ( ( ( sCRC16Result & 0x0001 ) ^ ( ( 0x0001 *
2161                  cCurrentData) & 0x0001 ) ) > 0 )
2162                     sCRC16Result = ( ( sCRC16Result >> 1 )
2163                      & 0x7FFF ) ^ gsCRC16GenerationCode;
2164                 else
2165                     sCRC16Result = ( sCRC16Result >> 1 )
2166                      & 0x7FFF;
2167                 cCurrentData = ( cCurrentData >> 1 ) & 0x7F;
2168             }
2169         }
2170     }
2171     return sCRC16Result;
2172 }
2173 /* ***** END OF SUBROUTINE TO CALCULATE CRC ***** */

```

2174 Outputs from the various input streams are as follows:

2175 Data (gpcTestData0): 00

2176 Checksum CS[15:0] = 0x0F87

2177 Data (gpcTestData1): 01

2178 Checksum CS[15:0] = 0x1E0E

2179 Data (gpcTestData2): FF 00 00 00 1E F0 1E C7 4F 82 78 C5 82 E0 8C 70 D2 3C  
2180 78 E9 FF 00 00 01

2181 Checksum CS[15:0] = 0xE569

2182 Data (gpcTestData3): FF 00 00 02 B9 DC F3 72 BB D4 B8 5A C8 75 C2 7C 81 F8  
2183 05 DF FF 00 00 01

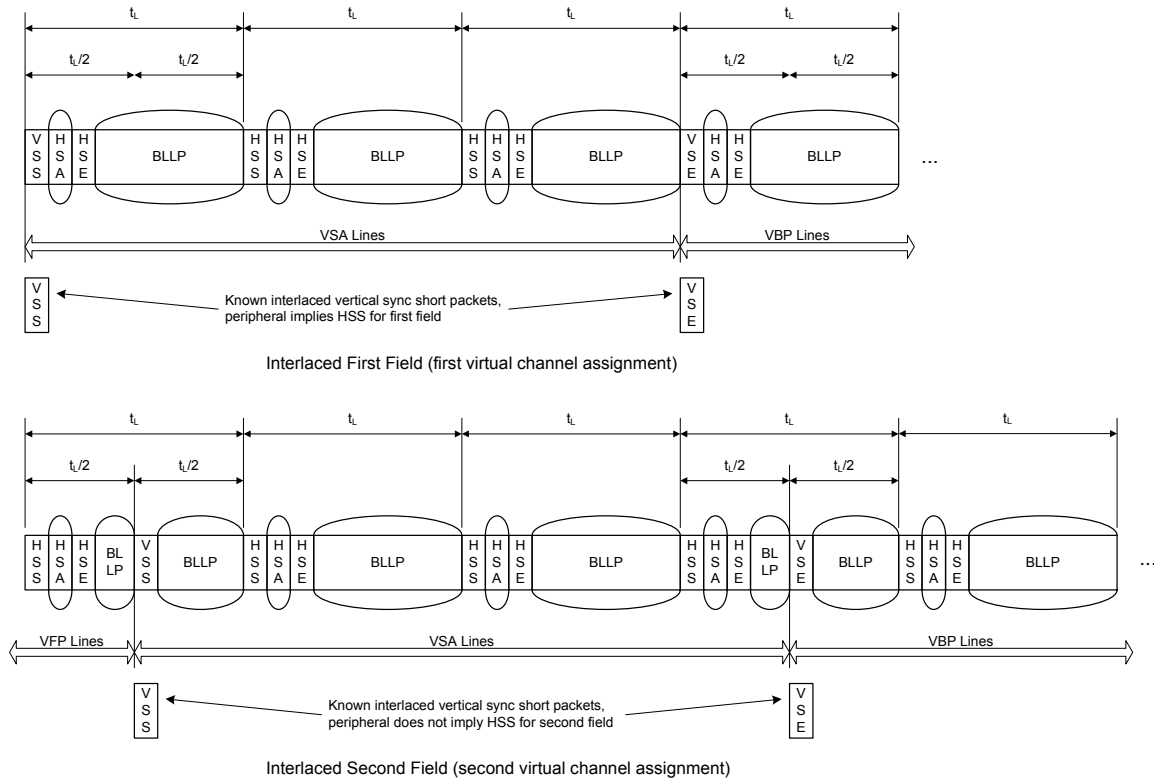
2184 Checksum CS[15:0] = 0x00F0

2185

2186 **Annex C Interlaced Video Transmission Sourcing**

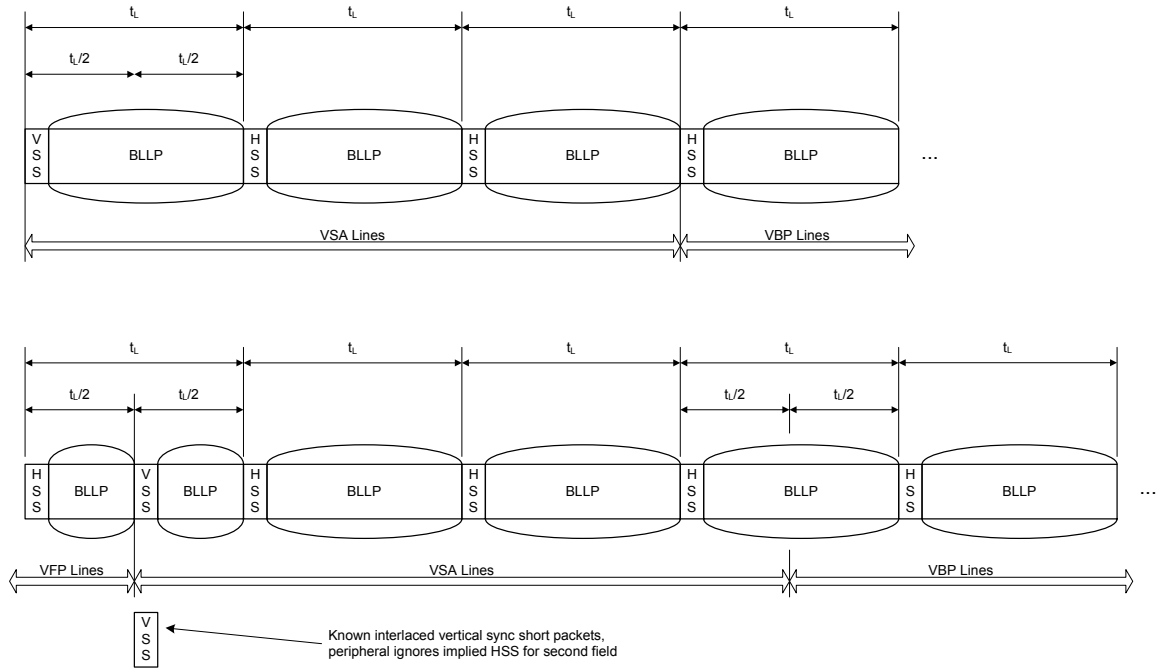
2187 In this annex, the diagrams are normative only in the sense that they are defining a method of transporting  
 2188 interlaced video with this specification. The use of interlaced video and its support by host, display module  
 2189 or other peripheral is optional.

2190 An example of the video mode interface timing for non-burst transmission with Sync Start and End  
 2191 sourcing interlaced video is shown in Figure 40. Note that in the first field, no timing differs from Figure  
 2192 30. In the second field, note HSS is not implied at the V Sync Start and V Sync End timing pulses.



2193  
 2194 **Figure 40 Video Mode Interface Timing: Non-burst Transmission with Sync Start and End**  
 2195 **(Interlaced Video)**

2196 An example of the video mode interface timing for non-burst transmission with Sync Events sourcing  
 2197 interlaced video is shown in Figure 41. Note that in the first field, no timing differs from the previous  
 2198 example. In the second field, note HSS is not implied at the V Sync Start timing event.



2199  
 2200  
 2201

**Figure 41 Video Mode Interface Timing: Non-burst Transmission with Sync Events (Interlaced Video)**