



MIPI Alliance Standard for Display Serial Interface V1.0

MIPI Board approved 5 April 2006

*** Caution to Implementers ***

This document is a MIPI Specification formally approved by the MIPI Alliance Board of Directors per the process defined in the MIPI Alliance Bylaws. However, the Display Working Group has identified certain technical issues in this approved version of the specification that are pending further review and which may require revisions or corrections to this document in the near future. Such revisions, if any, will be handled via the formal specification revision process as defined in the Bylaws.

A Release Notes document has been prepared by the Display Working Group and is available to all members. The intent of the Release Notes is to provide a list of known technical issues under further discussion with the working group. This may not be an exhaustive list; its purpose is to simply catalog known issues as of this release date. Implementers of this specification should be aware of these facts, and take them into consideration as they work with the specification.

Release Notes for the Display Serial Interface Specification can be found at the following direct, permanent link:

<https://www.mipi.org/members/file.asp?id=4844>



MIPI Alliance Standard for Display Serial Interface

Version 1.00a – 19 April 2006

MIPI Board Approved 5-Apr-2006

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

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

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

31 MIPI Alliance, Inc.
32 c/o IEEE-ISTO
33 445 Hoes Lane
34 Piscataway, NJ 08854
35 Attn: Board Secretary

36 Contents

37	Version 1.00 – 13 April 2006	i
38	1 Overview	8
39	1.1 Scope	8
40	1.2 Purpose	8
41	2 Terminology (Informational).....	9
42	2.1 Definitions	9
43	2.2 Abbreviations	10
44	2.3 Acronyms	10
45	3 References (Informational).....	13
46	3.1 DBI and DBI-2 (Display Bus Interface Standards for Parallel Signaling)	13
47	3.2 DPI and DPI-2 (Display Pixel Interface Standards for Parallel Signaling)	13
48	3.3 DCS (Display Command Set).....	14
49	3.4 CSI-2 (Camera Serial Interface 2)	14
50	3.5 D-PHY (MIPI Alliance Standard for Physical Layer).....	14
51	4 DSI Introduction.....	15
52	4.1 DSI Layer Definitions	16
53	4.2 Command and Video Modes	17
54	4.2.1 Command Mode	17
55	4.2.2 Video Mode Operation	17
56	4.2.3 Virtual Channel Capability	18
57	5 DSI Physical Layer	19
58	5.1 Data Flow Control	19
59	5.2 Bidirectionality and Low Power Signaling Policy.....	19
60	5.3 Command Mode Interfaces	20
61	5.4 Video Mode Interfaces	20
62	5.5 Bidirectional Control Mechanism.....	20

63	5.6	Clock Management.....	21
64	5.6.1	Clock Requirements	21
65	5.6.2	Clock Power and Timing.....	22
66	6	Multi-Lane Distribution and Merging	23
67	6.1	Multi-Lane Interoperability and Lane-number Mismatch	24
68	6.1.1	Clock Considerations with Multi-Lane.....	25
69	6.1.2	Bi-directionality and Multi-Lane Capability	25
70	6.1.3	SoT and EoT in Multi-Lane Configurations.....	25
71	7	Low-Level Protocol Errors and Contention.....	28
72	7.1	Low-Level Protocol Errors.....	28
73	7.1.1	SoT Error	28
74	7.1.2	SoT Sync Error.....	29
75	7.1.3	EoT Sync Error.....	29
76	7.1.4	Escape Mode Entry Command Error.....	30
77	7.1.5	LP Transmission Sync Error.....	30
78	7.1.6	False Control Error	31
79	7.2	Contention Detection and Recovery.....	31
80	7.2.1	Contention Detection in LP Mode.....	32
81	7.2.2	Contention Recovery Using Timers	32
82	7.3	Additional Timers.....	34
83	7.3.1	Turnaround Acknowledge Timeout (TA_TO).....	34
84	7.3.2	Peripheral Reset Timeout (PR_TO).....	35
85	7.4	Acknowledge and Error Reporting Mechanism	35
86	8	DSI Protocol.....	37
87	8.1	Multiple Packets per Transmission.....	37
88	8.2	Packet Composition.....	37
89	8.3	Endian Policy.....	38
90	8.4	General Packet Structure	38

91	8.4.1	Long Packet Format.....	38
92	8.4.2	Short Packet Format	40
93	8.5	Common Packet Elements.....	40
94	8.5.1	Data Identifier Byte	40
95	8.5.2	Error Correction Code	41
96	8.6	Interleaved Data Streams.....	41
97	8.6.1	Interleaved Data Streams and Bi-directionality.....	42
98	8.7	Processor to Peripheral Direction (Processor-Sourced) Packet Data Types.....	42
99	8.8	Processor-to-Peripheral Transactions – Detailed Format Description.....	43
100	8.8.1	Sync Event (H Start, H End, V Start, V End), Data Type = xx 0001 (x1h).....	43
101	8.8.2	Color Mode On Command, Data Type = 00 0010 (02h).....	44
102	8.8.3	Color Mode Off Command, Data Type = 01 0010 (12h)	44
103	8.8.4	Shutdown Peripheral Command, Data Type = 10 0010 (22h).....	44
104	8.8.5	Turn On Peripheral Command, Data Type = 11 0010 (32h)	44
105	8.8.6	Generic Short WRITE Packet, 0 to 7 Parameters, Data Type = xx x011 (x3h and xBh)	44
106	8.8.7	Generic READ Request, 0 to 7 Parameters, Data Type = xx x100 (x4h and xCh).....	44
107	8.8.8	DCS Commands	45
108	8.8.9	Set Maximum Return Packet Size, Data Type = 11 0111 (37h).....	46
109	8.8.10	Null Packet (Long), Data Type = 00 1001 (09h).....	46
110	8.8.11	Blanking Packet (Long), Data Type = 01 1001 (19h).....	46
111	8.8.12	Generic Non-Image Data (Long), Data Type = 10 1001 (29h).....	47
112	8.8.13	Packed Pixel Stream, 16-bit Format, Long packet, Data Type 00 1110 (0Eh).....	47
113	8.8.14	Packed Pixel Stream, 18-bit Format, Long packet, Data type = 01 1110 (1Eh).....	48
114	8.8.15	Pixel Stream, 18-bit Format in Three Bytes, Long packet, Data Type = 10 1110 (2Eh).....	49
115	8.8.16	Packed Pixel Stream, 24-bit Format, Long packet, Data Type = 11 1110 (3Eh).....	50
116	8.8.17	DO NOT USE and Reserved Data Types.....	50
117	8.9	Peripheral-to-Processor (Reverse Direction) LP Transmissions	51
118	8.9.1	Packet Structure for Peripheral-to-Processor LP Transmissions	51

119	8.9.2	System Requirements for ECC and Checksum and Packet Format.....	51
120	8.9.3	Appropriate Responses to Commands and ACK Requests.....	52
121	8.9.4	Format of Acknowledge with Error Report and Read Response Data Types.....	53
122	8.9.5	Error-Reporting Format.....	53
123	8.10	Peripheral-to-Processor Transactions – Detailed Format Description.....	54
124	8.10.1	Acknowledge with Error Report, Data Type 00 0010 (02h).....	55
125	8.10.2	Generic Short Read Response with Optional ECC, Data Type 01 0xxx (10h – 17h).....	55
126	8.10.3	Generic Long Read Response with Optional ECC and Checksum, Data Type = 01 1010	
127	(1Ah)	55	
128	8.10.4	DCS Long Read Response with Optional ECC and Checksum, Data Type 01 1100 (1Ch)..	56
129	8.10.5	DCS Short Read Response with Optional ECC, Data Type 10 0xxx (20h – 27h).....	56
130	8.10.6	Multiple-packet Transmission and Error Reporting	56
131	8.10.7	Clearing Error Bits.....	56
132	8.11	Video Mode Interface Timing	56
133	8.11.1	Traffic Sequences	57
134	8.11.2	Non-Burst Mode with Sync Pulses.....	58
135	8.11.3	Non-Burst Mode with Sync Events	58
136	8.11.4	Burst Mode	59
137	8.11.5	Parameters	60
138	8.12	TE Signaling in DSI	61
139	9	Error-Correcting Code (ECC) and Checksum	63
140	9.1	Hamming Code for Packet Header Error Detection/Correction	63
141	9.2	Hamming-modified Code for DSI.....	63
142	9.3	ECC Generation on the Transmitter and Byte-Padding.....	67
143	9.4	Applying ECC and Byte-Padding on the Receiver.....	67
144	9.5	Checksum Generation for Long Packet Payloads.....	68
145	10	Compliance, Interoperability, and Optional Capabilities.....	70
146	10.1	Display Resolutions.....	70

147	10.2	Pixel Formats.....	71
148	10.3	Number of Lanes	71
149	10.4	Maximum Lane Frequency.....	71
150	10.5	Bidirectional Communication.....	71
151	10.6	ECC and Checksum Capabilities.....	72
152	10.7	Display Architecture.....	72
153	10.8	Multiple Peripheral Support	72
154	Annex A (Informative) Contention Detection and Recovery Mechanisms		73
155	A.1	PHY Detected Contention	73
156	A.1.1	Protocol Response to PHY Detected Faults.....	73

157 **MIPI Alliance Standard for Display Serial Interface**

158 **1 Overview**

159 The Display Serial Interface (DSI) specification defines protocols between a host processor and peripheral
160 devices that adhere to MIPI Alliance specifications for mobile device interfaces. The DSI specification
161 builds on existing standards by adopting pixel formats and command set defined in MIPI Alliance
162 standards for DBI-2 [2], DPI-2 [3], and DCS [1].

163 **1.1 Scope**

164 Interface protocols as well as a description of signal timing relationships are within the scope of this
165 specification.

166 Electrical specifications and physical specifications are out of scope for this document. In addition, legacy
167 interfaces such as DPI-2 and DBI-2 are also out of scope for this specification. Furthermore, device usage
168 of auxiliary buses such as I²C or SPI, while not precluded by this specification, are also not within its
169 scope.

170 **1.2 Purpose**

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

176 2 Terminology (Informational)

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

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

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

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

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

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

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

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

194 2.1 Definitions

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

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

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

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

202 **Lane:** Consists of two complementary Lane Modules communicating via two-line, point-to-point Lane
203 Interconnects. A Lane is used for either Data or Clock signal transmission.

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

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

207 **Link:** A complete connection between two devices containing one Clock Lane and at least one Data Lane.

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

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

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

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

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

219 **Reverse Direction:** Reverse direction is the opposite of the forward direction. See the description for
 220 Forward Direction.

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

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

227 **Word Count:** Number of bytes.

228 **2.2 Abbreviations**

229 e.g. For example

230 **2.3 Acronyms**

231 AM Active matrix (display technology)

232 AIP Application Independent Protocol

233 ASP Application Specific Protocol

234 BLLP Blanking or Low Power interval

235 BPP Bits per Pixel

236 BTA Bus Turn-Around

237 CSI Camera Serial Interface

238 DBI Display Bus Interface

239 DI Data Identifier

240 DMA Direct Memory Access

241 DPI Display Pixel Interface

242	DSI	Display Serial Interface
243	DT	Data Type
244	ECC	Error-Correcting Code
245	EMI	Electro Magnetic interference
246	EoT	End of Transmission
247	ESD	Electrostatic Discharge
248	Fps	Frames per second
249	HS	High Speed
250	ISTO	Industry Standards and Technology Organization
251	LLP	Low-Level Protocol
252	LP	Low Power
253	LPI	Low Power Interval
254	LPS	Low Power State (state of serial data line when not transferring high-speed serial data)
255	LSB	Least Significant Bit
256	Mbps	Megabits per second
257	MIPI	Mobile Industry Processor Interface
258	MSB	Most Significant Bit
259	PE	Packet End
260	PF	Packet Footer
261	PH	Packet Header
262	PHY	Physical Layer
263	PI	Packet Identifier
264	PPI	PHY-Protocol Interface
265	PS	Packet Start
266	PT	Packet Type
267	PWB	Printed Wired Board
268	QCIF	Quarter-size CIF (resolution 176x144 pixels or 144x176 pixels)
269	QVGA	Quarter-size Video Graphics Array (resolution 320x240 pixels or 240x320 pixels)

270	RAM	Random Access Memory
271	RGB	Color presentation (Red, Green, Blue)
272	SLVS	Scalable Low Voltage Signaling
273	SoT	Start of Transmission
274	SVGA	Super Video Graphics Array (resolution 800x600 pixels or 600x800 pixels)
275	VGA	Video Graphics Array (resolution 640x480 pixels or 480x640 pixels)
276	VSA	Vertical Sync Active
277	WVGA	Wide VGA (resolution 800x480 pixels or 480x800 pixels)
278	WC	Word Count

279 3 References (Informational)

- 280 [1] MIPI Alliance Standard for Display Command Set, version 1.00, April 2006
- 281 [2] MIPI Alliance Standard for Display Bus Interface, version 2.00, November 2005
- 282 [3] MIPI Alliance Standard for Display Parallel Interface, version 2.00, September 2005
- 283 [4] MIPI Alliance Standard for D-PHY, version 0.65, November 2005
- 284 Design and Analysis of Fault Tolerant Digital System by Barry W. Johnson
- 285 Error Correcting Codes: Hamming Distance by Don Johnson paper
- 286 Intel 8206 error detection and correction unit datasheet
- 287 National DP8400-2 Expandable Error Checker/Corrector datasheet

288 Much of DSI is based on existing MIPI Alliance standards as well as several MIPI Alliance standards in
289 simultaneous development. In the Application Layer, DSI duplicates pixel formats used in *MIPI Alliance*
290 *Standard for Display Parallel Interface* [3] when it is in *Video Mode* operation. For display modules with a
291 display controller and frame buffer, DSI shares a common command set with *MIPI Alliance Standard for*
292 *Display Bus Interface* [2]. The command set is documented in *MIPI Alliance Standard for Display*
293 *Command Set* [1].

294 3.1 DBI and DBI-2 (Display Bus Interface Standards for Parallel Signaling)

295 DBI and DBI-2 are MIPI Alliance specifications for parallel interfaces to **display modules having display**
296 **controllers and frame buffers**. For systems based on these specifications, the host processor loads images to
297 the on-panel frame buffer through the display processor. Once loaded, the display controller manages all
298 display refresh functions on the display module without further intervention from the host processor. Image
299 updates require the host processor to write new data into the frame buffer.

300 DBI and DBI-2 specify a parallel interface; that is, data is sent to the peripheral over an 8-, 9- or 16-bit-
301 wide parallel data bus, with additional control signals.

302 The DSI specification supports a Command Mode of operation. Like the parallel DBI, a DSI-compliant
303 interface sends commands and parameters to the display. However, all information in DSI is first serialized
304 before transmission to the display module. At the display, serial information is transformed back to parallel
305 data and control signals for the on-panel display controller. Similarly, the display module can return status
306 information and requested memory data to the host processor, using the same serial data path.

307 3.2 DPI and DPI-2 (Display Pixel Interface Standards for Parallel Signaling)

308 DPI and DPI-2 are MIPI Alliance specifications for parallel interfaces to display modules without on-panel
309 display controller or frame buffer. These display modules rely on a steady flow of pixel data from host
310 processor to the display, to maintain an image without flicker or other visual artifacts. MIPI Alliance
311 specifications document several pixel formats for *Active Matrix* (AM) display modules.

312 Like DBI and DBI-2, DPI and DPI-2 are specifications for parallel interfaces. The data path may be 16-
313 18-, or 24-bits wide, depending on pixel format(s) supported by the display module. This specification
314 refers to DPI mode of operation as Video Mode.

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

321 **3.3 DCS (Display Command Set)**

322 DCS is a specification for the command set used by DSI and DBI-2 specifications. Commands are sent
323 from the host processor to the display module. On the display module, a display controller receives and
324 interprets commands, then takes appropriate action. Commands fall into four broad categories: **read**
325 **register**, **write register**, **read memory** and **write memory**. A command may be accompanied by multiple
326 parameters.

327 **3.4 CSI-2 (Camera Serial Interface 2)**

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

- 331 • CSI-2 uses unidirectional high-speed Link, whereas DSI is half-duplex bidirectional Link
- 332 • CSI-2 makes use of a secondary channel, based on I²C, for control and status functions

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

335 **3.5 D-PHY (MIPI Alliance Standard for Physical Layer)**

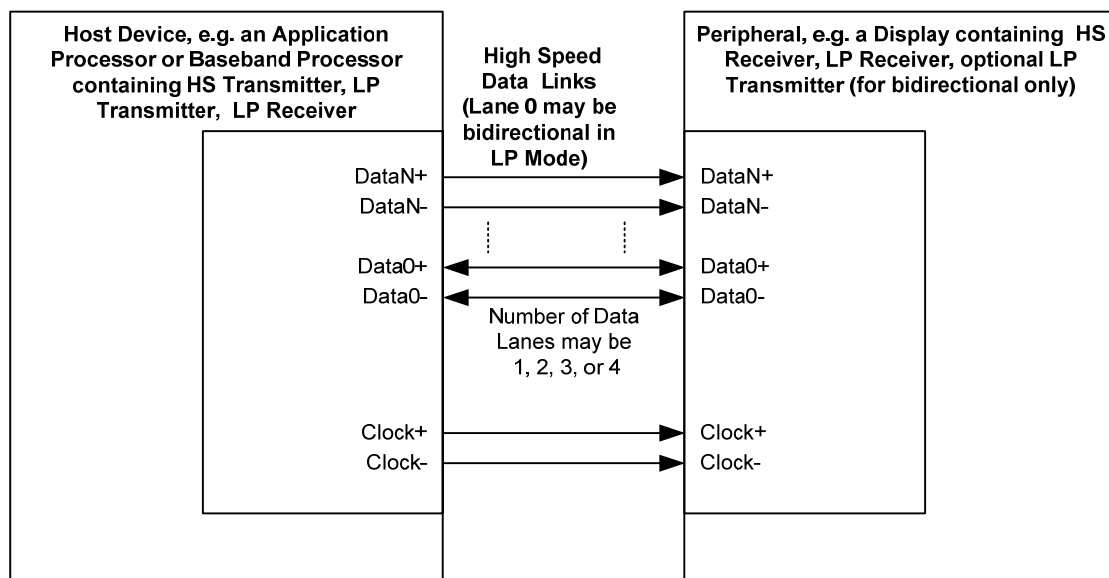
336 *MIPI Alliance Standard for D-PHY* [4] provides the physical layer definition for DSI. The functionality
337 specified by the D-PHY standard covers all electrical and timing aspects, as well as low-level protocols,
338 signaling, and message transmissions in various operating modes.

339 **4 DSI Introduction**

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

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

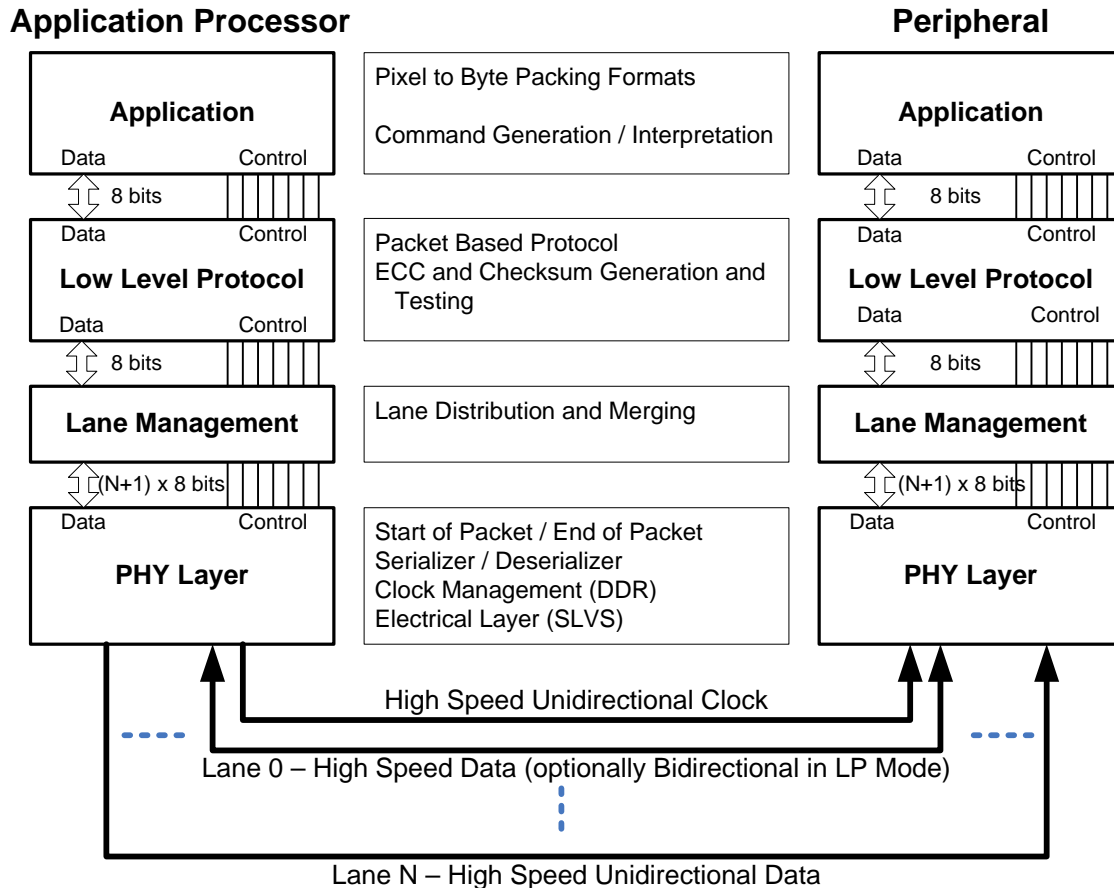
349 From a system or software point of view, the serialization and deserialization operations should be
 350 transparent. The most visible, and unavoidable, consequence of transformation to serial data and back to
 351 parallel is increased latency for transactions that require a response from the peripheral.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



355

356

Figure 1 DSI Transmitter and Receiver Interface

357 **4.1 DSI Layer Definitions**

358

359

Figure 2 DSI Layers

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

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

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

371 The PHY layer is described in *MIPI Alliance Standard for D-PHY* [4].

372 **Lane Management Layer:** DSI is Lane-scalable for increased performance. The number of data signals
 373 may be 1, 2, 3, or 4 depending on the bandwidth requirements of the application. The transmitter side of the
 374 interface distributes the outgoing data stream to one or more Lanes (“distributor” function). On the
 375 receiving end, the interface collects bytes from the Lanes and merges them together into a recombined data
 376 stream that restores the original stream sequence (“merger” function).

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

385 **Application Layer:** This layer describes higher-level encoding and interpretation of data contained in the
386 data stream. Depending on the display subsystem architecture, it may consist of pixels having a prescribed
387 format, or of commands that are interpreted by the display controller inside a display module. The DSI
388 specification describes the mapping of pixel values, commands and command parameters to bytes in the
389 packet assembly. See *MIPI Alliance Standard for Display Command Set* [1].

390 4.2 Command and Video Modes

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

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

399 4.2.1 Command Mode

400 Command Mode refers to operation in which transactions primarily take the form of **sending commands**
401 **and data to a peripheral**, such as a display module, that incorporates a display controller. The display
402 controller may include **local registers** and a **frame buffer**. Systems using Command Mode write to, and read
403 from, the registers and frame buffer memory. The host processor indirectly controls activity at the
404 peripheral by sending commands, parameters and data to the display controller. The host processor can also
405 read display module status information or the contents of the frame memory. Command Mode operation
406 requires a bidirectional interface.

407 4.2.2 Video Mode Operation

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

412 Some Video Mode architectures may include a simple timing controller and partial frame buffer, used to
413 maintain a partial-screen or lower-resolution image in standby or low-power mode. This permits the
414 interface to be shut down to reduce power consumption.

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

417 4.2.3 Virtual Channel Capability

418 While this specification only addresses the connection of a host processor to a single peripheral, DSI
419 incorporates a virtual channel capability for communication between a host processor and multiple,
420 physical display modules. Display modules are completely independent, may operate simultaneously, and
421 may be of different display architecture types, limited only by the total bandwidth available over the shared
422 DSI Link. The details of connecting multiple peripherals to a single Link are beyond the scope of this
423 document.

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

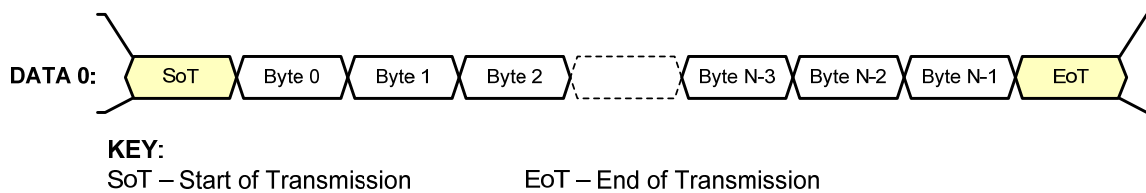
426 The DSI protocol permits up to **four virtual channels**, enabling traffic for multiple peripherals to share a
427 common DSI Link. In some high-resolution display designs, multiple physical drivers serve different areas
428 of a common display panel. Each driver is integrated with its own display controller that connects to the
429 host processor through DSI. Using virtual channels, the display controller directs data to the individual
430 drivers, eliminating the need for multiple interfaces or complex multiplexing schemes.

431 5 DSI Physical Layer

432 This section provides a brief overview of the physical layer used in DSI. See *MIPI Alliance Standard for*
433 *D-PHY* [4] for more details.

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

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



441 **Figure 3 Basic HS Transmission Structure**

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

444 5.1 Data Flow Control

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

451 5.2 Bidirectionality and Low Power Signaling Policy

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

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

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

461 For bidirectional Lanes, data shall be transmitted in the peripheral-to-processor, or reverse, direction using
462 Low-Power (LP) Mode only. See *MIPI Alliance Standard for D-PHY* [4] for details on the different modes
463 of transmission.

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

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

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

474 **5.3 Command Mode Interfaces**

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

- 476 • Data Lane Module: CIL-MUYY (HS-TX, LP-TX, LP-RX, and LP-CD)
- 477 • Clock Lane Module: CIL-MCNN (HS-TX, LP-TX)

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

- 479 • Data Lane Module: CIL-SUYY (HS-RX, LP-RX, LP-TX, and LP-CD)
- 480 • Clock Lane Module: CIL-SCNN (HS-RX, LP-RX)

481 Bidirectional Links shall support reverse-direction Escape Mode as well as forward direction Escape Mode.

482 **5.4 Video Mode Interfaces**

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

- 484 • Data Lane Module: CIL-MUNN (HS-TX, LP-TX)
- 485 • Clock Lane Module: CIL-MCNN (HS-TX, LP-TX)

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

- 487 • Data Lane Module: CIL-SUNN (HS-RX, LP-RX)
- 488 • Clock Lane Module: CIL-SCNN (HS-RX, LP-RX)

489 All DSI implementations should support forward escape ULPM on all Data Lanes.

490 **5.5 Bidirectional Control Mechanism**

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

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

499 **5.6 Clock Management**

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

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

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

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

512 For continuous clock behavior, the Clock Lane remains in high-speed mode generating active clock signals
513 between HS data packet transmissions. For non-continuous clock behavior, the Clock Lane enters the LP-
514 11 state between HS data packet transmissions.

515 **5.6.1 Clock Requirements**

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

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

524 Note that in Low Power signaling mode, LP clock is functionally embedded in the data signals. When LP
525 data transmission ends, the clock effectively stops and subsequent LP clocks are not available to the
526 peripheral. If the peripheral requires additional clocks to advance the state of its logic, to move data through
527 sequential buffers, or similar, it may be necessary to add ‘dummy’ data bytes to the LP transmission to
528 effect forward progress of state machines or to advance data through sequential logic.

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

535 **5.6.2 Clock Power and Timing**

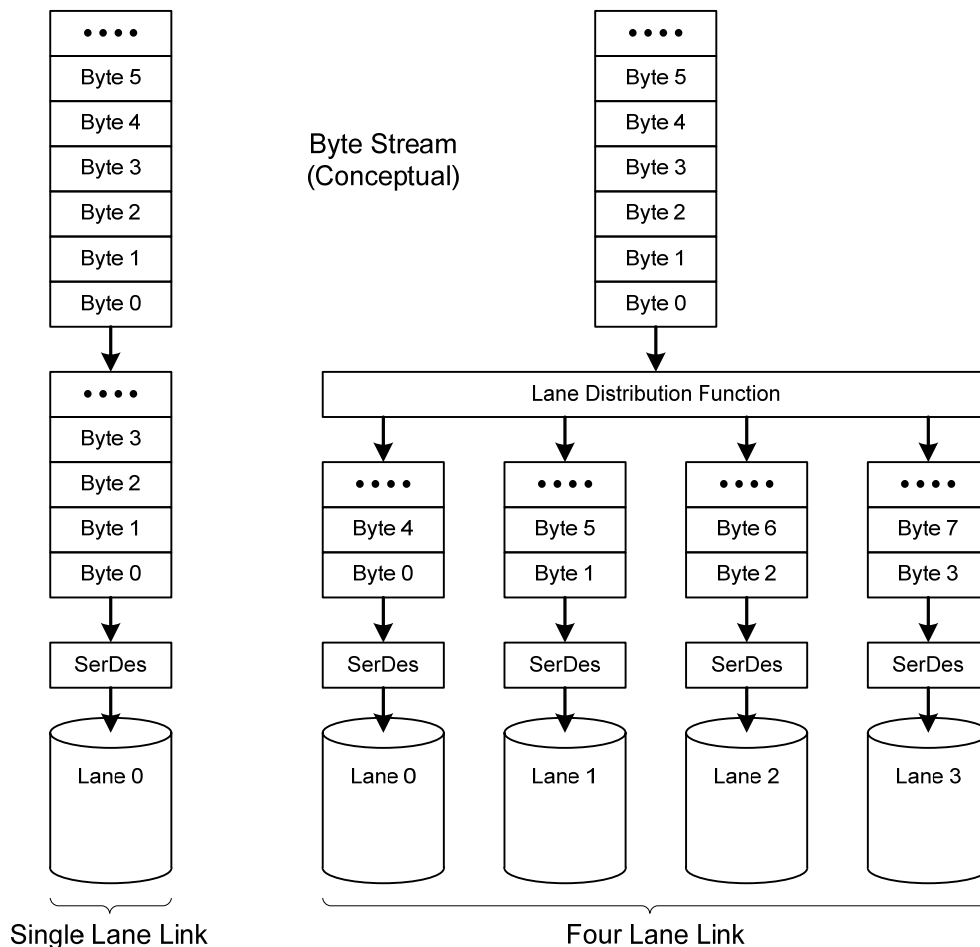
536 Additional timing requirements in *MIPI Alliance Standard for D-PHY* [4] specify the timing relationship
537 between the power state of data signal(s) and the power state of the clock signal. It is the responsibility of
538 the host processor to observe this timing relationship. If the DSI Clock runs continuously, these timing
539 requirements do not apply.

540 6 Multi-Lane Distribution and Merging

541 DSI is a Lane-scalable specification. Applications requiring more bandwidth than that provided by one
 542 Data Lane may expand the data path to two, three, or four Lanes wide and obtain approximately linear
 543 increases in peak bus bandwidth. This specification explicitly documents the mapping between application
 544 data and the serial bit stream to ensure compatibility between host processors and peripherals that make use
 545 of multiple Lanes.

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

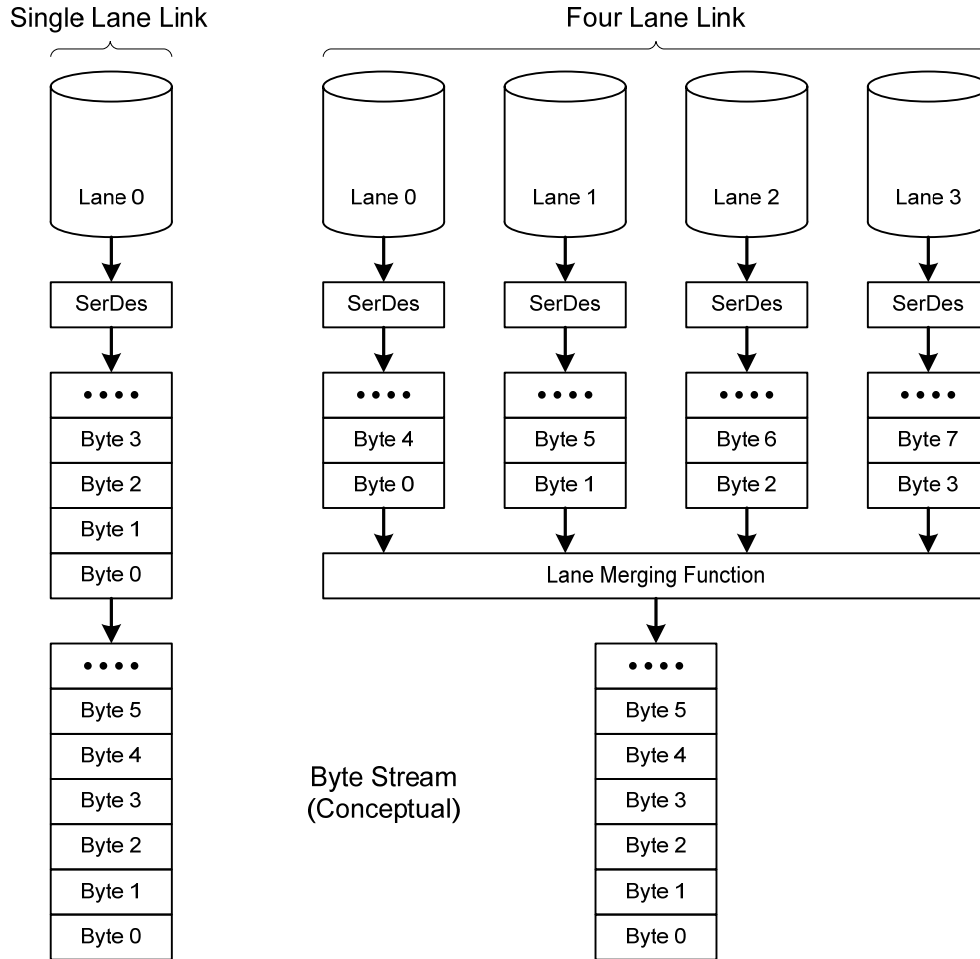
547 Conceptually, between the PHY and higher functional blocks is a layer that enables multi-Lane operation.
 548 In the transmitter, shown in Figure 4, this layer distributes a sequence of packet bytes across N Lanes,
 549 where each Lane is an independent block of logic and interface circuitry. In the receiver, shown in Figure 5,
 550 the layer collects incoming bytes from N Lanes and consolidates the bytes into complete packets to pass
 551 into the following packet decomposer.



552

553

Figure 4 Lane Distributor Conceptual Overview



554

555

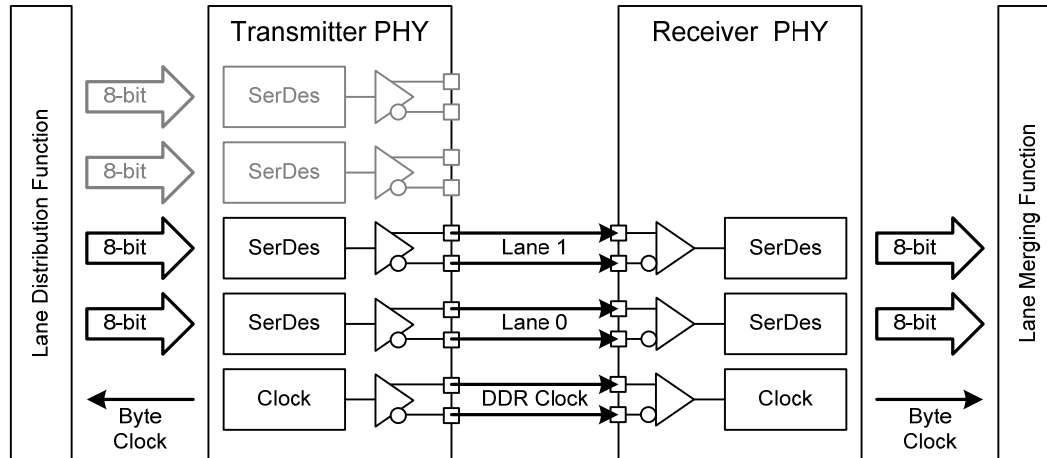
Figure 5 Lane Merger Conceptual Overview

556 The Lane Distributor takes a HS transmission of arbitrary byte length, buffers N bytes, where N is the
 557 number of Lanes implemented in the interface, and sends groups of N bytes in parallel across the N Lanes.
 558 Before sending data, all Lanes perform the SoT sequence in parallel to indicate to their corresponding
 559 receiving units that the first byte of a packet is beginning. After SoT, the Lanes send groups of N bytes
 560 from the first packet in parallel, following a round-robin process. For example, with a two Lane system,
 561 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.

562 6.1 Multi-Lane Interoperability and Lane-number Mismatch

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

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



569

570

Figure 6 Four-Lane Transmitter with Two-Lane Receiver Example

571 6.1.1 Clock Considerations with Multi-Lane

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

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

578 6.1.2 Bi-directionality and Multi-Lane Capability

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

582 6.1.3 SoT and EoT in Multi-Lane Configurations

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

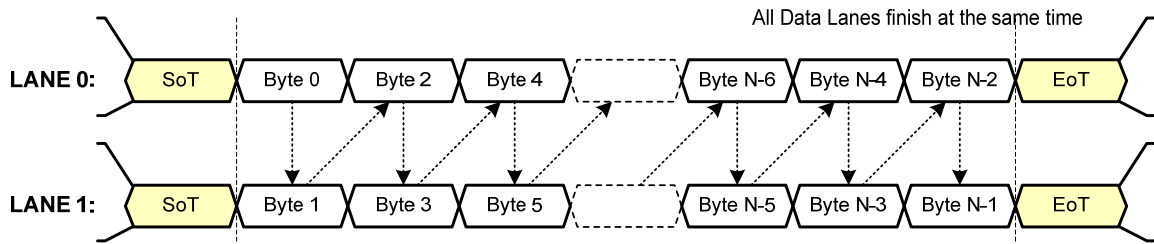
587 Although all Lanes start simultaneously with parallel SoTs, each Lane operates independently and may
 588 complete the HS transmission before the other Lanes, sending an EoT one cycle (byte) earlier.

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

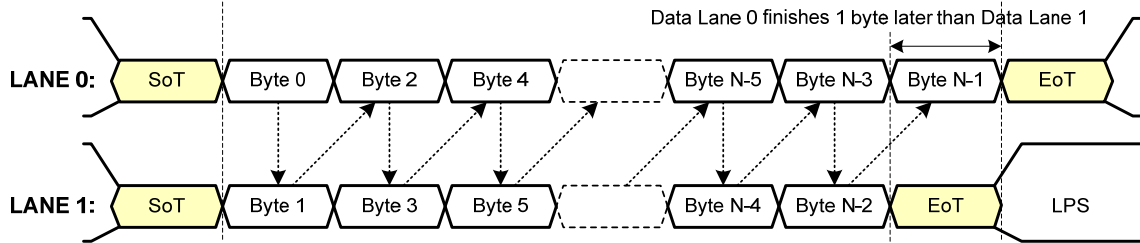
592 Figure 7 and Figure 8 illustrate a variety of ways a HS transmission can terminate for different number of
 593 Lanes and packet lengths.

594 Note the special case of a multi-Lane implementation, having N Lanes, which may occasionally send a
 595 short, HS transmission where the packet length is less than N. In this case, Lanes without data to transmit
 596 shall remain in LPS.

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:

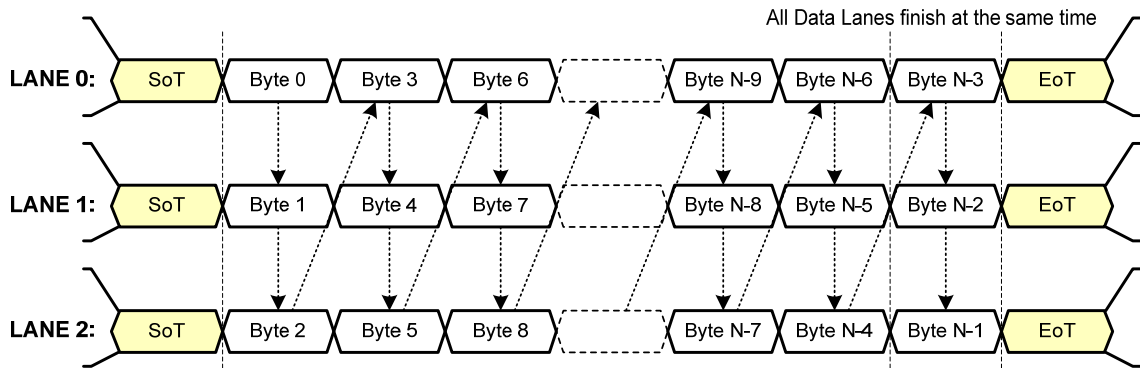
LPS – Low Power State SoT – Start of Transmission EoT – End of Transmission

597

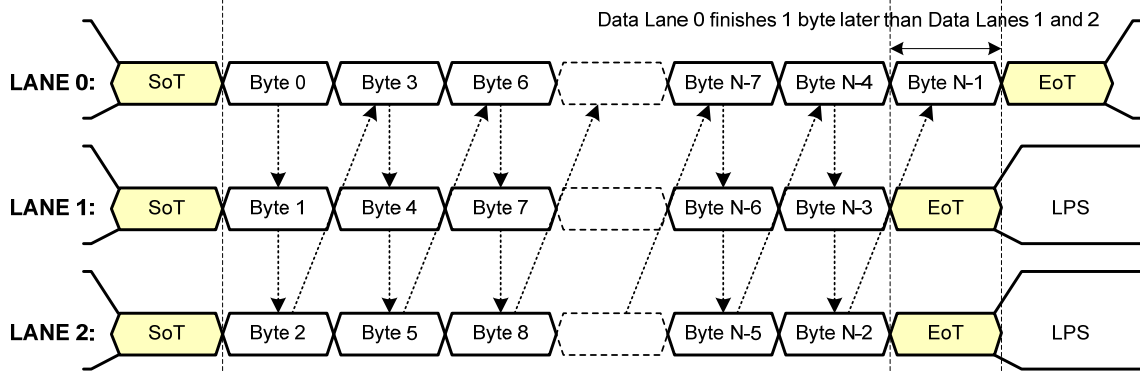
598

Figure 7 Two Lane HS Transmission Example

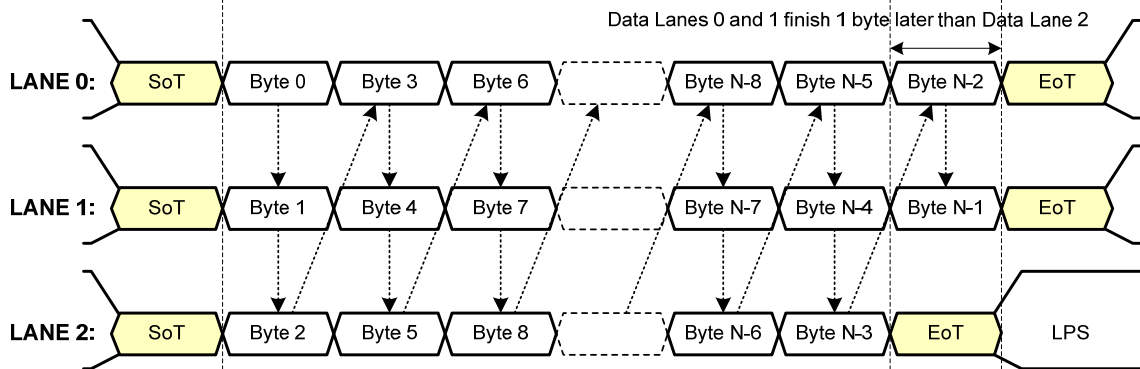
Number of Bytes, N, transmitted is an integer multiple of the number of lanes:



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

599
600

Figure 8 Three Lane HS Transmission Example

601 **7 Low-Level Protocol Errors and Contention**

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

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

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

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

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

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

618 **7.1 Low-Level Protocol Errors**

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

- 622 • SoT Error
- 623 • SoT Sync Error
- 624 • EoT Sync Error
- 625 • Escape Mode Entry Command Error
- 626 • LP Transmission Sync Error
- 627 • False Control Error

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

631 **7.1.1 SoT Error**

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

636

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 ACK packet, if requested; take no other action based on received HS transmission

637 *SoT Error* is detected by the peripheral PHY. If an acknowledge response is expected, the peripheral shall
 638 send a response using Data Type 02h (*Acknowledge with Error Report*) and set the *SoT Error* bit in the
 639 return packet to the host processor. The peripheral should take no other action based on the potentially
 640 corrupted received HS transmission.

641 7.1.2 SoT Sync Error

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

644 **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 ACK packet if requested; take no other action based on received transmission

645 *SoT Sync Error* is detected by the peripheral PHY. If an acknowledge response is expected, the peripheral
 646 shall send a response using Data Type 02h (*Acknowledge with Error Report*) and set the *SoT Sync Error* bit
 647 in the return packet to the host processor. Since data is probably corrupted, no command shall be
 648 interpreted or acted upon in the peripheral. No WRITE activity shall be undertaken in the peripheral.

649 7.1.3 EoT Sync Error

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

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

658

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

Receiving PHY	Receiving Protocol
Detect <i>EoT Sync Error</i>	
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 <i>Acknowledge</i> is requested

659

7.1.4 Escape Mode Entry Command Error

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

665

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 Error</i> flag
Go to <i>Escape Wait</i> until <i>Stop</i> 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

666

7.1.5 LP Transmission Sync Error

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

670

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 <i>Stop</i> state is observed	Ignore Escape Mode transmission if possible, set appropriate error bit and wait

671 **7.1.6 False Control Error**

672 If a received LP-01 or LP-10 State is followed by a *Stop* state instead of the expected Turnaround or Escape
673 Mode sequence, this error shall be flagged to the Protocol layer.

674 **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	

675

676 **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

677 **7.2 Contention Detection and Recovery**

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

682 7.2.1 Contention Detection in LP Mode

683 In bidirectional Links, contention detectors in the PHY shall detect two types of contention faults: LP High
684 Fault and LP Low Fault.

685 An LP High Fault occurs when a LP transmitter is driving high and the pin voltage is less than V_{IL} . An LP
686 Low Fault occurs when a LP transmitter is driving low and the pin voltage is greater than V_{ILF} .

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

690 7.2.2 Contention Recovery Using Timers

691 The PHY cannot detect all forms of contention. Although they do not directly detect contention, the use of
692 appropriate timers will ensure that any contention that does happen will be of limited duration.

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

695 7.2.2.1 Summary of Required Contention Recovery Timers

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

697 **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

698 7.2.2.2 HS RX Timeout (HRX_TO) in Peripheral

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

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

708

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

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

710 The LP High Fault and LP Low Fault are caused by both sides of the Link transmitting simultaneously.
711 Note, the LP High Fault and LP Low Fault are only applicable for bidirectional data lanes.

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

719 7.2.2.3 HS TX Timeout (HTX_TO) in Host Processor

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

724 **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)

725 7.2.2.4 LP TX-Peripheral Timeout (LTX-P_TO)

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

730 **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

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

734 7.2.2.5 LP-RX Host Processor Timeout (LRX-H_TO)

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

739 **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 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

740 7.3 Additional Timers

741 Additional timers are used to detect bus turnaround problems and to ensure sufficient wait time after *Reset*
742 is sent to the peripheral.

743 7.3.1 Turnaround Acknowledge Timeout (TA_TO)

744 When either end of the Link issues BTA (Bus Turn-Around), its PHY shall monitor the sequence of data-
745 lane states during the ensuing turnaround process. In a normal BTA sequence, the turnaround completes
746 within a bounded time, with the other end of the Link finally taking bus possession and driving LP-11 (*Stop*
747 state) on the bus. If the sequence is observed not to complete (by the previously-transmitting PHY) within
748 the specified time period, the timer TA_TO times out and begins a recovery procedure or re-sends BTA.
749 This specified period shall be longer than the maximum possible turnaround delay for the unit to which the
750 turnaround request was sent. This is an optional timer.

751 **Table 13 Sequence of Events for Turnaround Acknowledge Timeout (Peripheral initially**
 752 **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

753 **Table 14 Sequence of Events for Turnaround Acknowledge Timeout (Host Processor**
 754 **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)	

755 7.3.2 Peripheral Reset Timeout (PR_TO)

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

760 **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
Peripheral Reset Timeout	
Resume Normal Operation.	Wait for bus activity

761 7.4 Acknowledge and Error Reporting Mechanism

762 In a bidirectional Link, the peripheral monitors each transmission from the host processor, using detection
 763 features and timers specified in this section. Error information related to the transmission shall be stored in
 764 the peripheral.

765 The host processor may request a command acknowledge and error information related to any transmission
766 by asserting Bus Turnaround with the transmission. The peripheral shall respond with ACK alone if there
767 are no errors, and with ACK + Error Report if any errors were detected in the previous transmission.
768 Appropriate flags shall be set to indicate what errors were detected on the preceding transmission. If the
769 transmission was a Read request, the peripheral shall return READ data without ACK or Error report if no
770 errors were detected. If there was an error in the Read request, the peripheral will return the appropriate
771 ACK + Error Report.

772 See section 8.10 for more detail on ACK and Error Report protocols.

773 8 DSI Protocol

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

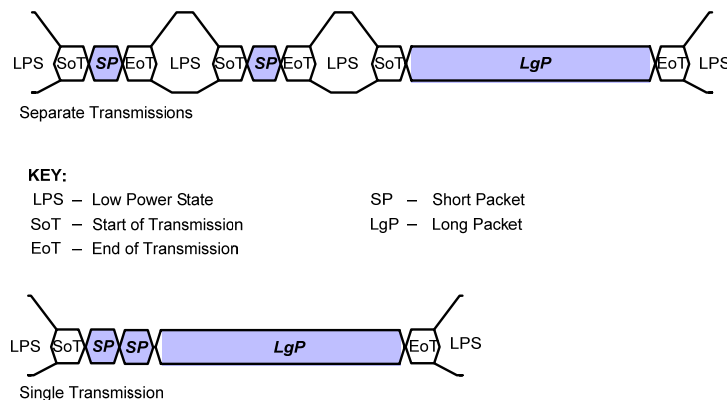
780 If there are multiple Lanes, the Lane Management layer distributes bytes to separate PHYs, one PHY per
 781 Lane, as described in Section 6. Packet protocol and formats are independent of the number of Lanes used.

782 8.1 Multiple Packets per Transmission

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

786 The DSI protocol permits multiple packets to be concatenated, which substantially boosts effective
 787 bandwidth. This is useful for events such as peripheral initialization, where many registers may be loaded
 788 with separate write commands at system startup. Figure 9 illustrates multiple packets being sent separately,
 789 and as concatenated packets in a single HS transmission.

790 In HS Mode, time gaps between packets shall result in separate HS transmissions for each packet, with a
 791 SoT, LPS, and EoT between packets. This constraint does not apply to LP transmissions.



792

793

Figure 9 Multiple Packet HS Transmission Example

794 8.2 Packet Composition

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

799 Packet sizes fall into two categories:

- 800 • **Short packets** specify the payload length using the Data Type field and are from two to nine bytes
801 in length. See Table 16 and Table 18 for payload lengths. Short packets are used for most
802 Command Mode commands and associated parameters. Other Short packets convey events like H
803 Sync and V Sync edges. Because they are Short packets they can convey accurate timing
804 information to logic at the peripheral.
- 805 • **Long packets** specify the payload length using a two-byte Word Count field. Payloads may be
806 from 0 to $2^{16} - 1$ bytes long. Therefore, a Long packet may be up to 65,541 bytes in length. Long
807 packets permit transmission of large blocks of pixel or other data.

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

813 The *Set Maximum Return Packet Size* command allows the host processor to limit the size of response
814 packets coming from a peripheral. See section 8.8.8.3 for a description of the command.

815 **8.3 Endian Policy**

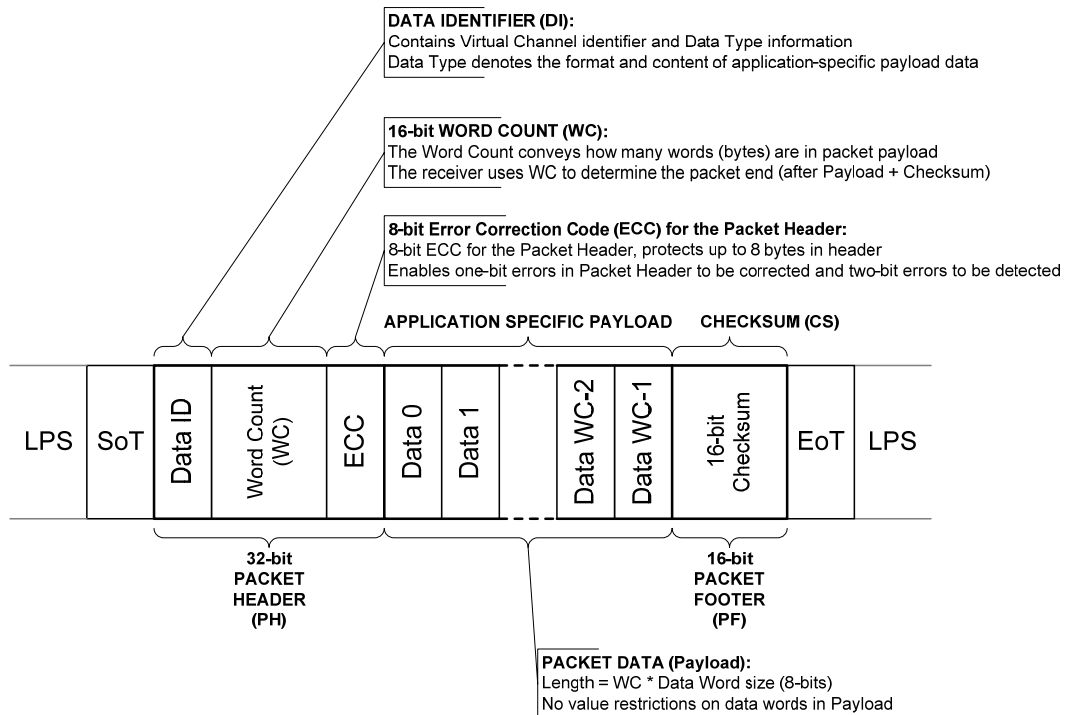
816 All packet data traverses the interface as bytes. Sequentially, a transmitter shall send data LSB first, MSB
817 last. For packets with multibyte fields, the least significant byte shall be transmitted first except as indicated
818 in the packet definition.

819 **8.4 General Packet Structure**

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

822 **8.4.1 Long Packet Format**

823 Figure 10 shows the structure of the Long packet. A Long packet shall consist of three elements: a 32-bit
824 Packet Header (PH), an application-specific Data Payload with a variable number of bytes, and a 16-bit
825 Packet Footer (PF). The Packet Header is further composed of three elements: an 8-bit Data Identifier, a
826 16-bit Word Count, and 8-bit ECC. The Packet Footer has one element, a 16-bit checksum. Long packets
827 can be from 6 to 65,541 bytes in length.



828

829

Figure 10 Long Packet Structure

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

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

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

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

840 Once the receiver has read the Data Payload it reads the Checksum in the Packet Footer. The host processor
841 shall always calculate and transmit a Checksum in the Packet Footer. Peripherals are not required to
842 calculate a Checksum. Also note the special case of zero-byte Data Payload: if the payload has length 0,
843 then the Checksum calculation results in (FFFFh). If the Checksum is not calculated, the Packet Footer
844 shall consist of two bytes of all zeros (0000h). See section 9 for more information on calculating the
845 Checksum.

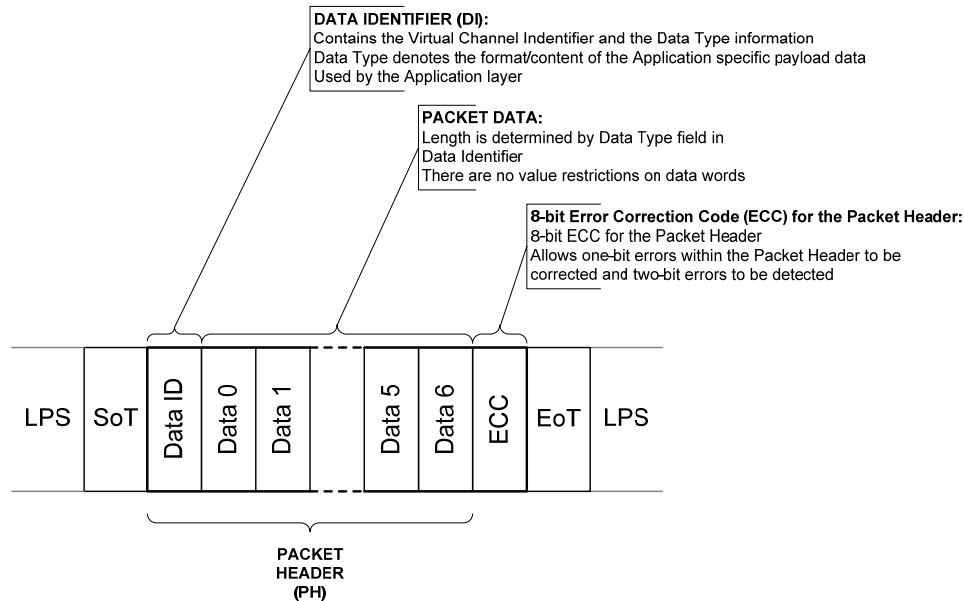
846 In the generic case, the length of the Data Payload shall be a multiple of bytes. In addition, each data format
847 may impose additional restrictions on the length of the payload data, e.g. multiple of four bytes.

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

851 **8.4.2 Short Packet Format**

852 Figure 11 shows the structure of the Short packet. See sections 8.8 through 8.10 for descriptions of the Data
 853 Types. A Short packet shall contain an 8-bit Data ID followed by zero to seven bytes and an 8-bit ECC; a
 854 Packet Footer shall not be present. Short packets can be from two to nine bytes in length.

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



857

858

Figure 11 Short packet Structure

859 **8.5 Common Packet Elements**

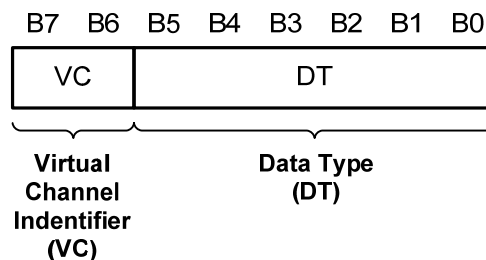
860 Long and Short packets have several common elements that are described in this section.

861 **8.5.1 Data Identifier Byte**

862 The first byte of any packet is the DI (Data Identifier) byte. Figure 12 shows the composition of the Data
 863 Identifier (DI) byte.

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

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



866

867

Figure 12 Data Identifier Byte

868 8.5.1.1 Virtual Channel Identifier – VC field, DI[7:6]

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

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

877 8.5.1.2 Data Type Field DT[5:0]

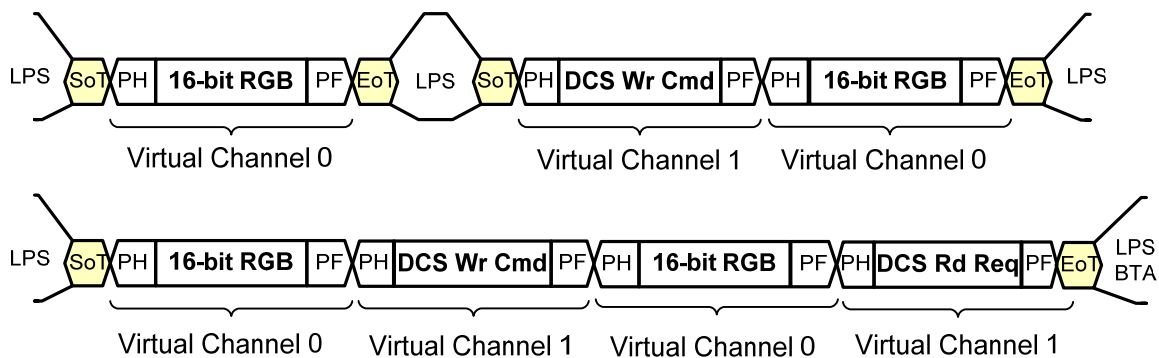
878 The Data Type field specifies the size, format and, in some cases, the interpretation of the packet contents.
879 For example, in the minimal case the DT field is the packet contents. By specifying the packet size, it
880 informs the receiver of how many bytes to expect in the remainder of the packet. This is necessary because
881 there are no special packet start / end sync codes to indicate the beginning and end of a packet. This permits
882 packets to convey arbitrary data, but it also requires the packet header to explicitly specify the size of the
883 packet.

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

886 8.5.2 Error Correction Code

887 The Error Correction Code allows single-bit errors to be corrected and 2-bit errors to be detected in the
888 Packet Header. The host processor shall always calculate and transmit an ECC byte. Peripherals are not
889 required to calculate an ECC byte. If the ECC is not used, a single byte of all zeros (00h) shall be
890 transmitted. See section 9 for more information on coding and decoding the ECC.

891 8.6 Interleaved Data Streams



KEY:

LPS – Low Power State

SoT – Start of Transmission

EoT – End of Transmission

PH – Packet Header

PF – Packet Footer

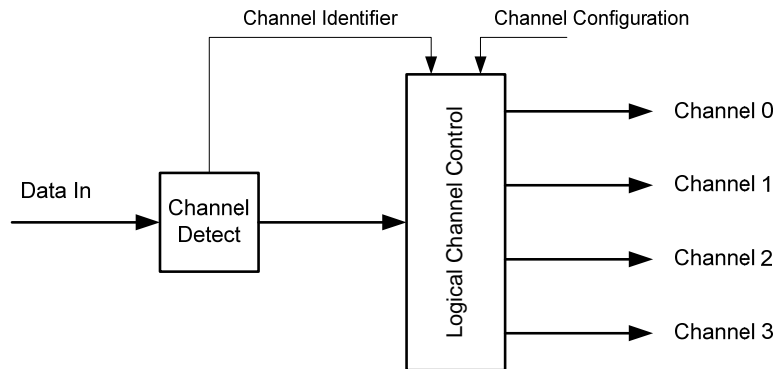
BTA – Bus Turn-Around

892
893

894

Figure 13 Interleaved Data Stream Example

895 One application for multiple channels is a high-resolution display using two or more separate driver ICs on
 896 a single display module. Each driver IC addresses only a portion of the columns on the display device.
 897 Each driver IC captures and displays only the packet contents targeted for that driver and ignores the other
 898 packets. See Figure 14.



899

900

Figure 14 Logical Channel Block Diagram (Receiver Case)

901 8.6.1 Interleaved Data Streams and Bi-directionality

902 When multiple peripherals have bidirectional capability there shall be a clear and unambiguous means for
 903 returning READ data, events and status back to the host processor from the intended peripheral. The
 904 combination of BTA and the Virtual Channel ID ensures no confusion over which peripheral is expected to
 905 respond to any request from the peripheral.

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

909 8.7 Processor to Peripheral Direction (Processor-Sourced) Packet Data Types

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

912

Table 16 Data Types for Processor-sourced Packets

Data Type, hex	Data Type, binary	Description	Packet Size
01h	00 0001	Sync Event, V Sync Start	Short
11h	01 0001	Sync Event, V Sync End	Short
21h	10 0001	Sync Event, H Sync Start	Short
31h	11 0001	Sync Event, H Sync End	Short
02h	00 0010	Color Mode (CM) Off Command	Short
12h	01 0010	Color Mode (CM) On Command	Short
22h	10 0010	Shut Down Peripheral Command	Short

Data Type, hex	Data Type, binary	Description	Packet Size
32h	11 0010	Turn On Peripheral Command	Short
x3h and xBh	xx x011	Generic WRITE, 0-7 parameters, bits 5:3 = parameter count	Short
x4h and xCh	xx x100	Generic READ, 0-7 parameters, bits 5:3 = parameter count	Short
x5h and xDh	xx x101	DCS WRITE, 0-6 parameters, bits 5:3 = parameter count + 1	Short
06h	00 0110	DCS READ, no parameters	Short
37h	11 0111	Set Maximum Return Packet Size	Short
09h	00 1001	Null Packet, no data	Long
19h	01 1001	Blanking Packet, no data	Long
29h	10 1001	Generic Non-image Packet	Long
39h	11 1001	DCS Long Write/write_LUT Command Packet	Long
0Eh	00 1110	Packed Pixel Stream, 16-bit RGB, 5-6-5 Format	Long
1Eh	01 1110	Packed Pixel Stream, 18-bit RGB, 6-6-6 Format	Long
2Eh	10 1110	Loosely Packed Pixel Stream, 18-bit RGB, 6-6-6 Format	Long
3Eh	11 1110	Packed Pixel Stream, 24-bit RGB, 8-8-8 Format	Long
x0h and xFh, unspecified	xx 0000 xx 1111	DO NOT USE All unspecified codes are reserved	

913 8.8 Processor-to-Peripheral Transactions – Detailed Format Description

914 8.8.1 Sync Event (H Start, H End, V Start, V End), Data Type = xx 0001 (x1h)

915 Sync Events are two-byte packets (one command byte, one ECC byte) and therefore can time-accurately
 916 represent events like the start and end of sync pulses. As “start” and “end” are separate and distinct events,
 917 the length of sync pulses, as well as position relative to active pixel data, e.g. front and back porch display
 918 timing, may be accurately conveyed to the peripheral. The Sync Events are defined as follows:

- 919 • Data Type = 00 0001 (01h) V Sync Start
- 920 • Data Type = 01 0001 (11h) V Sync End
- 921 • Data Type = 10 0001 (21h) H Sync Start
- 922 • Data Type = 11 0001 (31h) H Sync End

923 In order to represent timing information as accurately as possible a V Sync Start event represents the start
 924 of the VSA and also implies a H Sync Start event for the first line of the VSA. Similarly, a V Sync End
 925 event implies a H Sync Start event for the last line of the VSA.

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

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

934 **8.8.2 Color Mode On Command, Data Type = 00 0010 (02h)**

935 *Color Mode On* is a single-byte packet command (two bytes with ECC) that switches a Video Mode
 936 display module to a low-color mode for power saving.

937 **8.8.3 Color Mode Off Command, Data Type = 01 0010 (12h)**

938 *Color Mode Off* is a single-byte packet (two bytes with ECC) command that returns a Video Mode display
 939 module from low-color mode to normal display operation.

940 **8.8.4 Shutdown Peripheral Command, Data Type = 10 0010 (22h)**

941 *Shutdown Peripheral* command is a two-byte packet (one command byte, one ECC byte) that turns off the
 942 display in a Video Mode display module for power saving. Note the interface shall remain powered in
 943 order to receive the turn-on, or wake-up, command.

944 **8.8.5 Turn On Peripheral Command, Data Type = 11 0010 (32h)**

945 *Turn On Peripheral* command is a single-byte packet (two bytes with ECC) that turns on the display in a
 946 Video Mode display module for normal display operation.

947 **8.8.6 Generic Short WRITE Packet, 0 to 7 Parameters, Data Type = xx x011 (x3h and xBh)**

948 *Generic Short WRITE* command is a Short packet type for sending generic data to the peripheral. The
 949 format and interpretation of the contents of this packet are outside the scope of this specification. It is the
 950 responsibility of the system designer to ensure that both the host processor and peripheral agree on the
 951 format and interpretation of such data.

952 The complete packet may be up to nine bytes in length including an ECC byte. The number of bytes
 953 beyond the header (DI) byte is explicitly specified by a 3-bit field, DT[5:3]

954 **8.8.7 Generic READ Request, 0 to 7 Parameters, Data Type = xx x100 (x4h and xCh)**

955 *Generic READ* request is a Short packet requesting data from the peripheral. The format and interpretation
 956 of the parameters of this packet, and of returned data, are outside the scope of this specification. It is the
 957 responsibility of the system designer to ensure that both the host processor and peripheral agree on the
 958 format and interpretation of such data.

959 Returned data may be of Short or Long packet format. Note the *Set Max Return Packet Size* command
 960 limits the size of returning packets so that the host processor can prevent buffer overflow conditions when
 961 receiving data from the peripheral. If the returning block of data is larger than the maximum return packet
 962 size specified, the read response will require more than one transmission. The host processor shall send

963 multiple Generic READ requests in separate transmissions if the requested data block is larger than the
964 maximum packet size.

965 The complete command packet may be up to nine bytes in length including the ECC byte. The number of
966 bytes beyond the header (DI) byte is explicitly specified by a 3-bit field, DT[5:3]. Since this is a read
967 command, BTA shall be asserted by the host processor following this request.

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

- 969 • If an error was detected by the peripheral, it shall send *Acknowledge with Error Report*. If an ECC
970 error in the request was detected and corrected, the peripheral shall transmit the requested READ
971 data packet with the error report packet appended, in the same transmission.
- 972 • If no error was detected by the peripheral, it shall send the requested READ packet (Short or
973 Long) with appropriate ECC and Checksum, if either or both features are enabled.

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

978 **8.8.8 DCS Commands**

979 DCS is a standardized command set intended for Command Mode display modules. The interpretation of
980 DCS commands is supplied in *MIPI Alliance Standard for Display Command Set* [1].

981 For DCS short commands, the first byte following the Data Identifier Byte is the *DCS Command Byte*.
982 Following the command byte may be from zero to six *DCS Command Parameters*, with each parameter
983 one byte in length.

984 Bits [5:3] of the Data Type (DT) field specify the number of parameters, N, plus the *DCS Command Byte*.
985 This specifies the packet length to the receiver and is used to determine when the last byte of the DCS
986 command packet has been transmitted. Using N+1 permits DCS packets to be parsed by receiving logic the
987 same as generic packets, the extra byte being the DCS command itself. For example, if a DCS Short Write
988 command was accompanied by three parameters, DT[5:3] should be set to 4h (100b) and DT[5:0] would
989 therefore be 25h (10 0101b).

990 **8.8.8.1 DCS Short Write Command, 0 to 6 parameters, Data Type = xx x101 (x5h and xDh)**

991 *DCS Short Write* command is used to write data to a peripheral such as a display module. DT[5:3] indicate
992 the number of parameters. One ECC byte shall follow the command and any parameters bytes. If *DCS*
993 *Short Write* command, followed by BTA, is sent to a bidirectional peripheral, the peripheral shall respond
994 with *Acknowledge* unless an error was detected in the host-to-peripheral transmission. If the peripheral
995 detects an error in the transmission, the peripheral shall respond with *Acknowledge with Error Report*. If
996 the peripheral is a Video Mode display on a unidirectional DSI, it shall ignore BTA. See Table 18.

997 **8.8.8.2 DCS Read Request, No Parameters, Data Type = 00 0110 (06h)**

998 DCS READ commands are used to request data from a display module. The first byte following the Data
999 Identifier byte is the DCS Command Byte, in this case specifying a read command. Following the
1000 Command is an ECC byte. Depending on the type of READ requested in the DCS Command Byte, the
1001 peripheral may respond with a DCS Short Read Response or DCS Long Read Response.

1002 The read response may be more than one packet in the case of DCS Long Read Response, if the returning
1003 block of data is larger than the maximum return packet size specified. In that case, the host processor shall

1004 send multiple DCS Read Request commands to transfer the complete data block. See section 8.8.8.3 for
1005 details on setting the read packet size.

1006 The peripheral shall respond to DCS READ Request in one of the following ways:

- 1007 • If an error was detected by the peripheral, it shall send *Acknowledge with Error Report*. If an ECC
1008 error in the request was detected and corrected, the peripheral shall send the requested READ data
1009 packet, with appropriate ECC if the feature is enabled, following the error report packet, in the
1010 same transmission.
- 1011 • If no error was detected by the peripheral, it shall send the requested READ packet (Short or
1012 Long) with appropriate ECC and Checksum, if either or both features are enabled.

1013 A DCS Read Request packet shall be the only, or last, packet of a transmission. Following the transmission,
1014 the host processor sends BTA. Having given control of the bus to the peripheral, the host processor will
1015 expect the peripheral to transmit the appropriate response packet and then return bus possession to the host
1016 processor.

1017 **8.8.8.3 DCS Long Write / write_LUT Command, Data Type = 11 1001 (39h)**

1018 *DCS Long Write/write_LUT Command* is used to send larger blocks of data to a display module that
1019 implements the Display Command Set.

1020 The packet consists of the DI byte, a two-byte WC, an ECC byte, followed by the *DCS Command Byte*, a
1021 payload of length WC minus one bytes, and a two-byte checksum.

1022 **8.8.9 Set Maximum Return Packet Size, Data Type = 11 0111 (37h)**

1023 *Set Maximum Return Packet Size* is a four-byte command packet (including ECC) that specifies the
1024 maximum size of the payload in a Long packet transmitted from peripheral back to the host processor. The
1025 order of bytes in *Set Maximum Return Packet Size* is: Data ID, two-byte value for maximum return packet
1026 size, followed by the ECC byte. Note that the two-byte value is transmitted with LS byte first. This
1027 command shall be ignored by peripherals with unidirectional DSI interfaces.

1028 During a power-on or Reset sequence, the Maximum Return Packet Size shall be set by the peripheral to a
1029 default value of one. This parameter should be set by the host processor to the desired value in the
1030 initialization routine before commencing normal operation.

1031 **8.8.10 Null Packet (Long), Data Type = 00 1001 (09h)**

1032 *Null Packet* is a mechanism for keeping the serial Data Lane(s) in High-Speed mode while sending dummy
1033 data. This is a Long packet. Like all packets, its content shall be an integer number of bytes.

1034 The Null Packet consists of the DI byte, a two-byte WC, ECC byte, and “null” payload of WC bytes,
1035 ending with a two-byte Checksum. Actual data values sent are irrelevant because the peripheral does not
1036 capture or store the data. However, ECC and Checksum shall be generated and transmitted to the
1037 peripheral.

1038 **8.8.11 Blanking Packet (Long), Data Type = 01 1001 (19h)**

1039 A Blanking packet is used to convey blanking timing information in a Long packet. Normally, the packet
1040 represents a period between active scan lines of a Video Mode display, where traditional display timing is
1041 provided from the host processor to the display module. The blanking period may have *Sync Event* packets

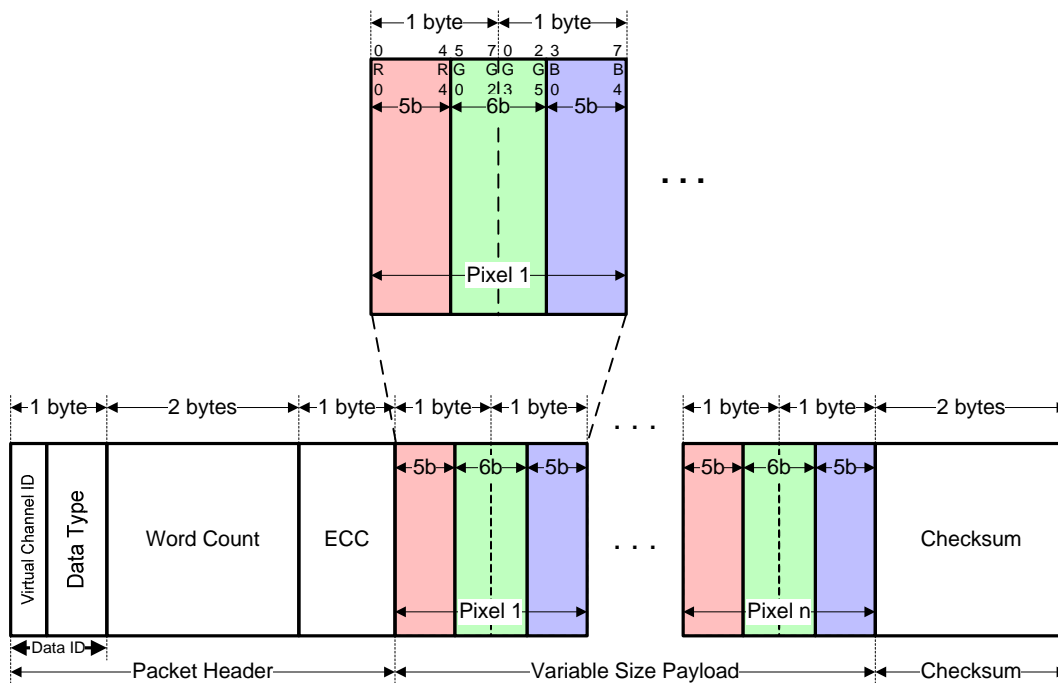
1042 interspersed between blanking segments. Like all packets, the Blanking packet contents shall be an integer
1043 number of bytes.

1044 The Blanking packet consists of the DI byte, a two-byte WC, an ECC byte, a payload of length WC bytes,
1045 and a two-byte checksum.

1046 8.8.12 Generic Non-Image Data (Long), Data Type = 10 1001 (29h)

1047 *Generic Non-Image Data Packet* is used to transmit arbitrary blocks of data from a host processor to a
1048 peripheral in a Long packet. The packet consists of the DI byte, a two-byte WC, an ECC byte, a payload of
1049 length WC bytes and a two-byte checksum.

1050 8.8.13 Packed Pixel Stream, 16-bit Format, Long packet, Data Type 00 1110 (0Eh)



1051

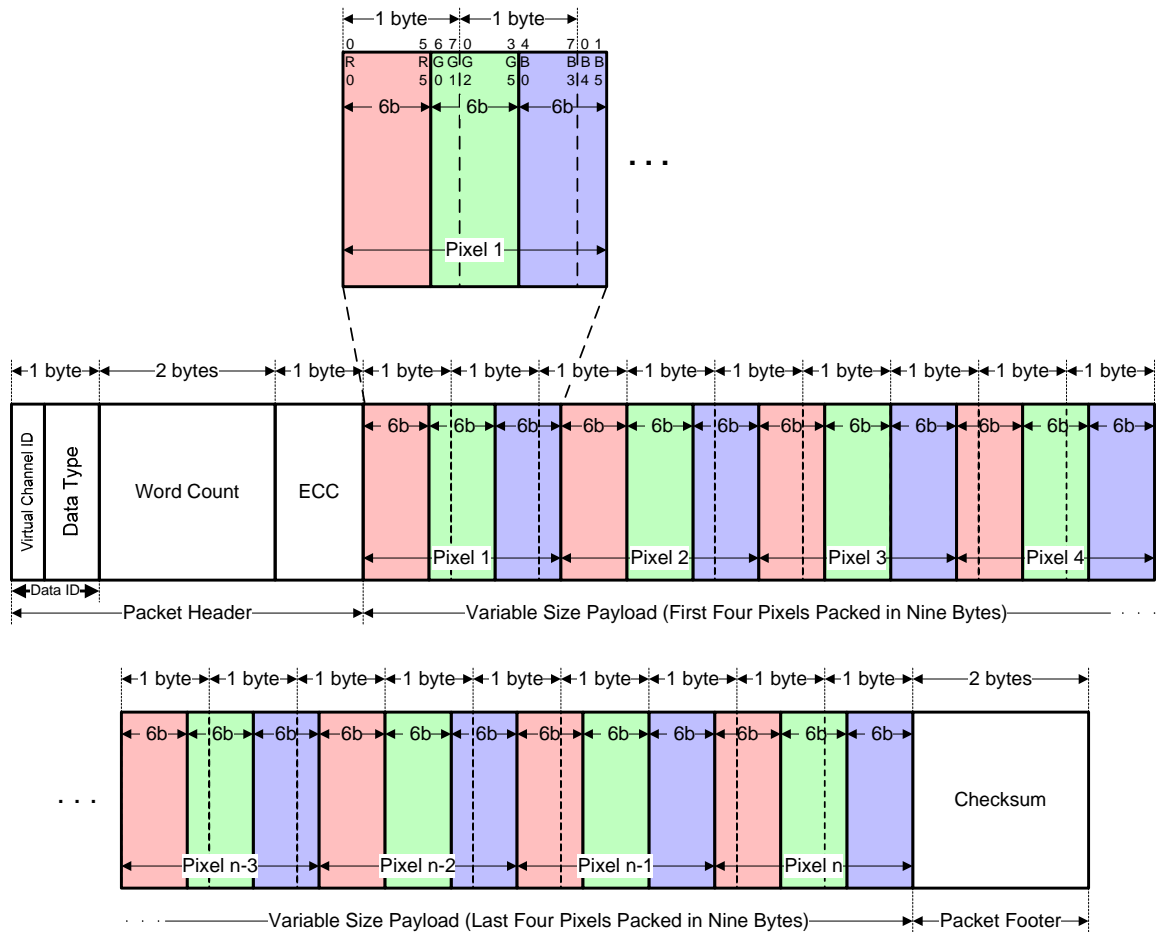
1052

Figure 15: 16-bit per Pixel – RGB Color Format, Long packet

1053 *Packed Pixel Stream 16-Bit Format* is a Long packet used to transmit image data formatted as 16-bit pixels
1054 to a Video Mode display module. The packet consists of the DI byte, a two-byte WC, an ECC byte, a
1055 payload of length WC bytes and a two-byte checksum. Pixel format is five bits red, six bits green, five bits
1056 blue, in that order. Note that the “Green” component is split across two bytes. Within a color component,
1057 the LSB is sent first, the MSB last.

1058 With this format, it is strongly recommended that TOTAL line width be a multiple of one pixel (two bytes)
1059 and that timing in the host display controller use that time unit for its activity, including assertion of
1060 Transmit Request to its PHY layer. This ensures that every scan line has the same synchronous relationship
1061 between the Byte clock and Pixel clock.

1062 Normally, the display has no frame buffer of its own, so all image data shall be supplied by the host
1063 processor at a sufficiently high rate to avoid flicker or other visible artifacts.

1064 **8.8.14 Packed Pixel Stream, 18-bit Format, Long packet, Data type = 01 1110 (1Eh)**

1065

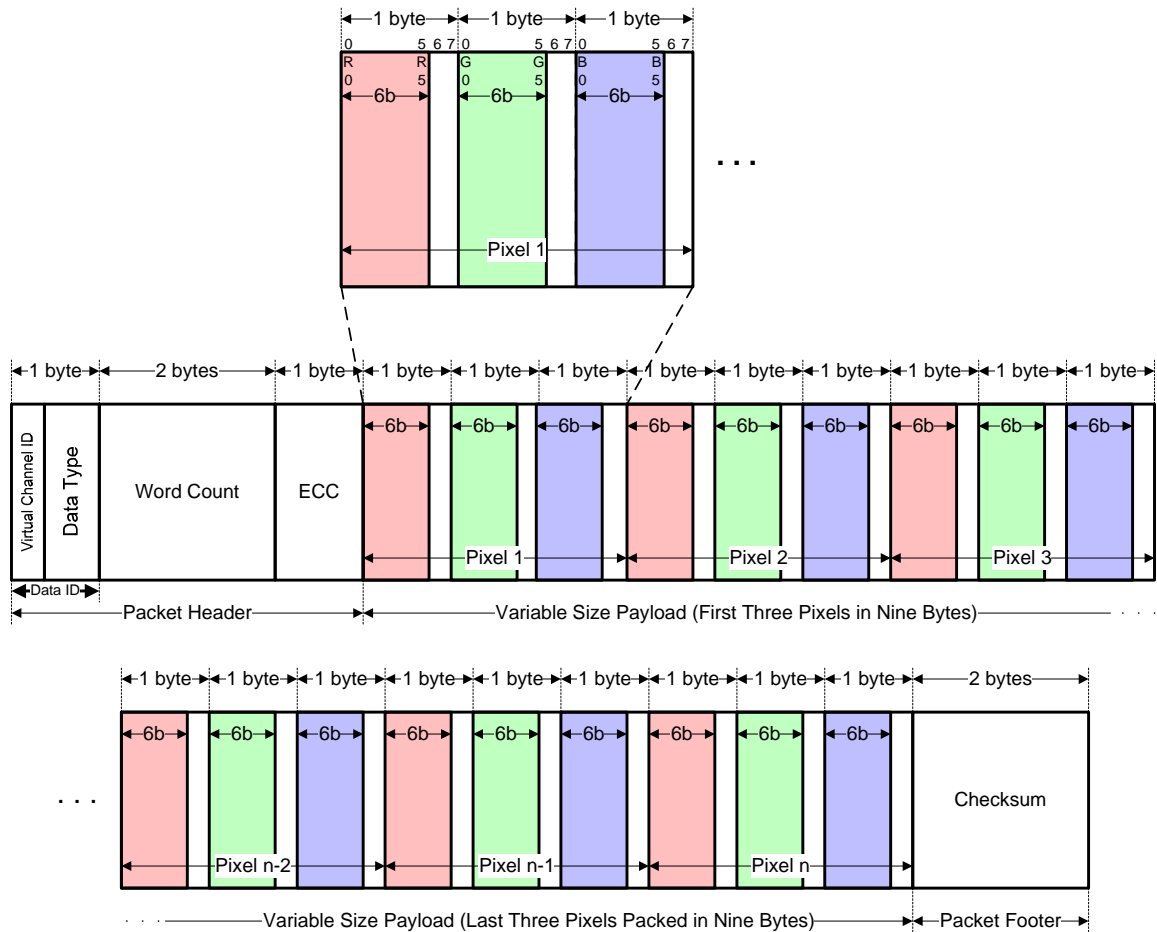
1066

Figure 16: 18-bit per Pixel (Packed) – RGB Color Format, Long packet

1067 *Packed Pixel Stream 18-Bit Format (Packed)* is a Long packet. It is used to transmit RGB image data
 1068 formatted as pixels to a Video Mode display module that displays 18-bit pixels. The packet consists of the
 1069 DI byte, a two-byte WC, an ECC byte, a payload of length WC bytes, and a two-byte Checksum. Pixel
 1070 format is red (6 bits), green (6 bits), and blue (6 bits), in that order. Within a color component, the LSB is
 1071 sent first, the MSB last.

1072 Note that pixel boundaries only line up with byte boundaries every four pixels (nine bytes). Preferably,
 1073 display modules employing this format have a horizontal extent (width in pixels) evenly divisible by four,
 1074 so no partial bytes remain at the end of the display line data. It is possible to send pixel data that represent
 1075 a line width that is not a multiple of four pixels, but display logic on the receiver end shall dispose of the
 1076 extra bits of the partial byte at the end of active display and ensure a “clean start” for the next line.

1077 With this format, it is strongly recommended that the total line width be a multiple of four pixels (nine
 1078 bytes) and that timing in the host processor use that time unit (four pixel duration) for its activity, including
 1079 assertion of Transmit Request to its PHY layer. This ensures that every scan line has the same synchronous
 1080 relationship between Byte clock and Pixel clock.

1081 **8.8.15 Pixel Stream, 18-bit Format in Three Bytes, Long packet, Data Type = 10 1110 (2Eh)**

1082

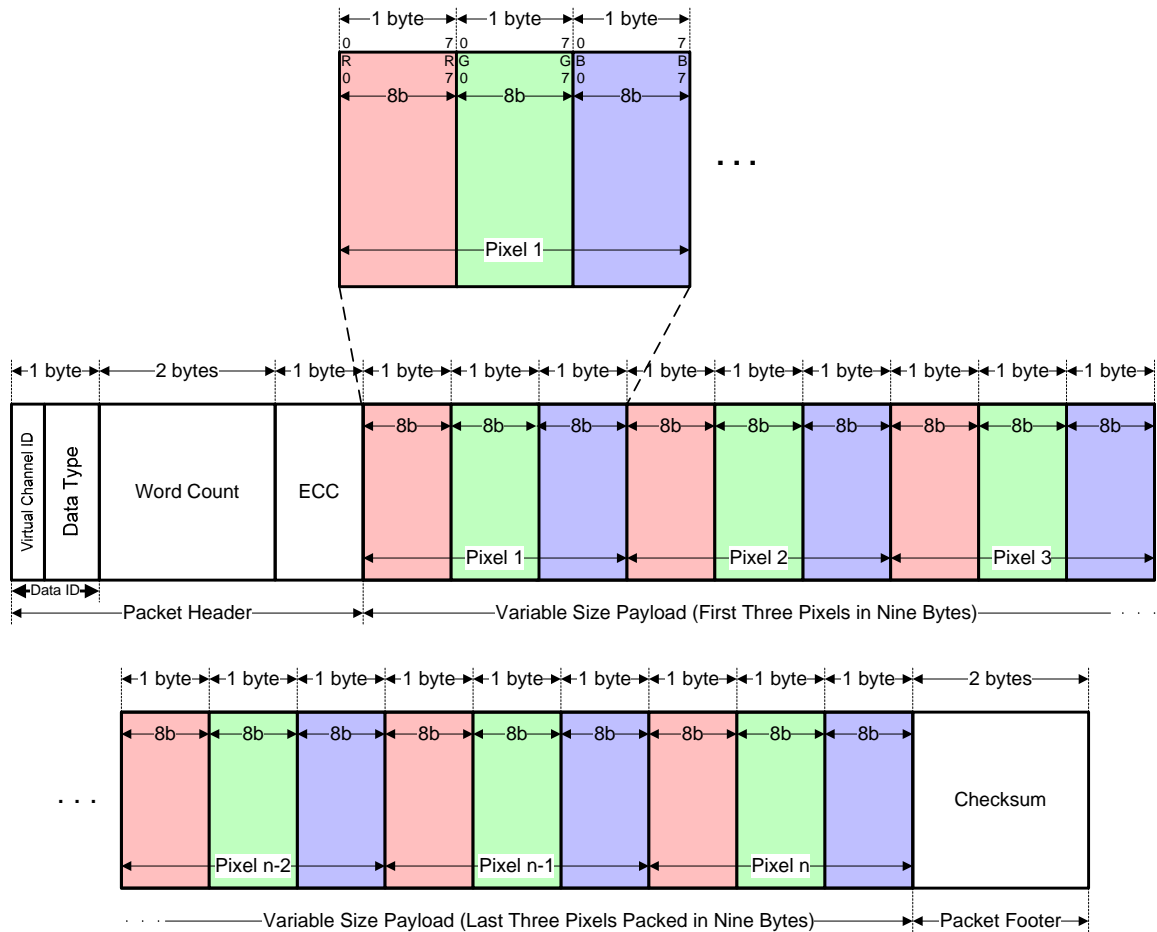
1083

Figure 17: 18-bit per Pixel (Loosely Packed) – RGB Color Format, Long packet

1084 In the *18-bit Pixel Loosely Packed* format, each R, G, or B color component is six bits but is shifted to the
 1085 upper bits of the byte, such that the valid pixel bits occupy bits [7:2] of each byte. Bits [1:0] of each
 1086 payload byte representing active pixels are ignored. As a result, each pixel requires three bytes as it is
 1087 transmitted across the Link. This requires more bandwidth than the “packed” format, but requires less
 1088 shifting and multiplexing logic in the packing and unpacking functions on each end of the Link.

1089 This format is used to transmit RGB image data formatted as pixels to a Video Mode display module that
 1090 displays 18-bit pixels. The packet consists of the DI byte, a two-byte WC, an ECC byte, a payload of length
 1091 WC bytes and a two-byte Checksum. The pixel format is red (6 bits), green (6 bits) and blue (6 bits) in that
 1092 order. Within a color component, the LSB is sent first, the MSB last.

1093 With this format, pixel boundaries line up with byte boundaries every three bytes. It is strongly
 1094 recommended that the total line width be a multiple of three bytes and that timing in the host processor use
 1095 that time unit (three bytes) for its activity, including assertion of Transmit Request to its PHY layer. This
 1096 ensures that every scan line has the same synchronous relationship between the Byte clock and Pixel clock.

1097 **8.8.16 Packed Pixel Stream, 24-bit Format, Long packet, Data Type = 11 1110 (3Eh)**

1098

1099

Figure 18: 24-bit per Pixel – RGB Color Format, Long packet

1100 *Packed Pixel Stream 24-Bit Format* is a Long packet. It is used to transmit image data formatted as 24-bit
 1101 pixels to a Video Mode display module. The packet consists of the DI byte, a two-byte WC, an ECC byte, a
 1102 payload of length WC bytes and a two-byte Checksum. The pixel format is red (8 bits), green (8 bits) and
 1103 blue (8 bits), in that order. Each color component occupies one byte in the pixel stream; no components are
 1104 split across byte boundaries. Within a color component, the LSB is sent first, the MSB last.

1105 With this format, pixel boundaries line up with byte boundaries every three bytes. It is strongly
 1106 recommended that the total line width be a multiple of three bytes and that timing in the host processor use
 1107 that time unit (three bytes) for its activity, including assertion of Transmit Request to its PHY layer. This
 1108 ensures that every scan line has the same synchronous relationship between the Byte clock and Pixel clock.

1109 **8.8.17 DO NOT USE and Reserved Data Types**

1110 Data Type codes with four LSBs = 0000 or 1111 shall not be used. All other non-specified Data Type
 1111 codes are reserved.

1112 Note that DT encoding is specified so that all data types have at least one 0-1 or 1-0 transition in the four
 1113 bits DT bits [3:0]. This ensures a transition within the first four bits of the serial data stream of every
 1114 packet. DSI protocol or the PHY can use this information to determine quickly, following the end of each

1115 packet, if the next bits represent the start of a new packet (transition within four bits) or an EoT sequence
1116 (no transition for at least four bits).

1117 **8.9 Peripheral-to-Processor (Reverse Direction) LP Transmissions**

1118 All Command Mode systems require bidirectional capability for returning READ data, acknowledge, or
1119 error information to the host processor. Multi-Lane systems shall use Lane 0 for all peripheral-to-processor
1120 transmissions; other Lanes shall be unidirectional.

1121 Reverse-direction signaling shall only use LP (Low Power) mode of transmission.

1122 Simple, low-cost systems using display modules which work exclusively in Video Mode may be
1123 configured with unidirectional DSI for all Lanes. In such systems, no acknowledge or error reporting is
1124 possible using DSI, and no requirements specified in this section apply to such systems. However, these
1125 systems may have ECC checking and correction capability, which enables them to correct single-bit errors
1126 in headers and Short packets, even if they cannot report the error. If a peripheral has ECC capability then
1127 the ECC capability shall be implemented as documented in this specification.

1128 Command Mode systems that use DCS shall have a bidirectional data path. Short packets and the header of
1129 Long packets may use ECC and Checksum to provide a higher level of data integrity. The Checksum
1130 feature enables detection of errors in the payload of Long packets.

1131 **8.9.1 Packet Structure for Peripheral-to-Processor LP Transmissions**

1132 Packet structure for peripheral-to-processor transactions is the same as for the processor-to-peripheral
1133 direction.

1134 As in the processor-to-peripheral direction, two basic packet formats are specified: Short and Long. For
1135 both types, an ECC byte may be calculated to cover the Packet Header data. If ECC is not used then the
1136 ECC byte shall be 00h. ECC calculation is the same in the peripheral as in the host processor. For Long
1137 packets, error checking on the Data Payload, i.e. all bytes after the Packet Header, is also optional. If the
1138 Checksum is not calculated by the peripheral the Packet Footer shall be 0000h.

1139 BTA shall take place after every peripheral-to-processor transaction. This returns bus control to the host
1140 processor following the completion of the LP transmission from the peripheral.

1141 Peripheral-to-processor transactions are of three basic types:

1142 *Event Notification* is a *Trigger* message, sent by the peripheral's PHY layer in response to an event
1143 generated or detected by the protocol or application controller in the peripheral. See *MIPI Alliance*
1144 *Standard for D-PHY* [4] for a description of this message.

1145 *Acknowledge and Acknowledge with Error Report* confirms that the prior command or data from processor
1146 to peripheral was received, and indicates if any of several possible error types were detected on the
1147 transmission. These are Short packets.

1148 *Response to Read Request* returns data requested by the preceding READ command from the processor.
1149 These may be short or Long packets.

1150 **8.9.2 System Requirements for ECC and Checksum and Packet Format**

1151 A peripheral may optionally implement ECC, checksum or both.

1152 Host processors shall implement both ECC and checksum capabilities. Both capabilities shall be separately
 1153 enabled so that a host processor can match a peripheral's capabilities. The mechanism for enabling and
 1154 disabling the ECC and checksum capabilities is out of scope for this specification.

1155 An ECC byte can be applied to both Short and Long packets. Checksum bytes shall only be applied to
 1156 Long packets.

1157 Host processors, and peripherals that implement ECC, shall provide ECC capabilities in both the Forward
 1158 and Reverse communication directions.

1159 Host processors, and peripherals that implement Checksum, shall provide Checksum capabilities in both
 1160 the Forward and Reverse communication directions.

1161 See section 8.4 for a description of the ECC and Checksum bytes.

1162 **8.9.3 Appropriate Responses to Commands and ACK Requests**

1163 In general, if the host processor completes a transmission to the peripheral with BTA asserted, the
 1164 peripheral shall respond with one or more appropriate packet(s), and then return bus ownership to the host
 1165 processor. If BTA is not asserted following a transmission from the host processor, the peripheral shall not
 1166 communicate an Acknowledge or other error information back to the host processor.

1167 Interpretation of processor-to-peripheral transactions with BTA asserted, and the expected responses, are as
 1168 follows:

- 1169 • Following a non-Read command in which no error was detected, the peripheral shall respond with
 1170 *Acknowledge*.
- 1171 • Following a Read request in which no error was detected, the peripheral shall send the requested
 1172 READ data.
- 1173 • Following a Read request in which a single-bit ECC error was detected and corrected, the
 1174 peripheral shall send the requested READ data in a Long or Short packet, followed by a 4-byte
 1175 *Acknowledge with Error Report* packet in the same LP transmission. The Error Report shall have
 1176 the ECC Error – Single Bit flag set.
- 1177 • Following a non-Read command in which a single-bit ECC error was detected and corrected, the
 1178 peripheral shall proceed to execute the command, and shall respond to BTA by sending a 4-byte
 1179 *Acknowledge with Error Report* packet. The Error Report shall have the ECC Error – Single Bit
 1180 flag set.
- 1181 • Following a Read request in which multi-bit ECC errors were detected and not corrected, the
 1182 peripheral shall send a 4-byte *Acknowledge with Error Report* packet without sending Read data.
 1183 The Error Report shall have the ECC Error – Multi-Bit flag set.
- 1184 • Following a non-Read command in which multi-bit ECC errors were detected and not corrected,
 1185 the peripheral shall not execute the command, and shall send a 4-byte *Acknowledge with Error*
 1186 *Report* packet. The Error Report shall have the ECC Error – Multi-Bit flag set.
- 1187 • Following any command in which *SoT Error*, *SoT Sync Error* or *DSI VC ID Invalid* was detected,
 1188 or the DSI command was not recognized, the peripheral shall send a 4-byte *Acknowledge with*
 1189 *Error Report* response, with the appropriate error flags set in the two-byte error field. Only the
 1190 ACK/Error Report packet shall be transmitted; no read or write accesses shall take place on the
 1191 peripheral in response.
- 1192 • Following any command in which *EoT Sync Error* or *LP Transmit Sync Error* is detected, or a
 1193 checksum error is detected in the payload, the peripheral shall send a 4-byte *Acknowledge with*
 1194 *Error Report* packet with the appropriate error flags set.

1195 **8.9.4 Format of Acknowledge with Error Report and Read Response Data Types**

1196 *Acknowledge with Error Report* confirms that the preceding command or data from processor to peripheral
 1197 was received, and indicates what types of error were detected on the transmission. This response is a Short
 1198 packet of four bytes, taking the form:

- 1199 • Byte 0: Data Identifier (Virtual Channel ID + Acknowledge Data Type)
- 1200 • Byte 1: Error Report bits 0-7
- 1201 • Byte 2: Error Report bits 8-15
- 1202 • ECC byte covering bytes 0-2
- 1203 ○ If ECC is not calculated by the peripheral, send 00h

1204 *Acknowledge* is a short packet of two bytes, taking the form:

- 1205 • Byte 0: Data Identifier + Acknowledge Data Type
- 1206 • Byte 1: ECC Byte covering Byte 0
- 1207 ○ If ECC is not calculated by the peripheral, send 00h

1208 *Response to Read Request* returns data requested by the preceding READ command from the processor.
 1209 These may be short or Long packets. The format for short READ packet responses is:

- 1210 • Byte 0: Data Identifier (Virtual Channel ID + Data Type)
- 1211 • Bytes 1-7: READ data, may be from one to seven bytes, length indicated by Data Type [2:0]
- 1212 • ECC byte covering bytes 0-7
- 1213 ○ If ECC is not calculated by the peripheral, send 00h

1214 The format for long READ packet responses is:

- 1215 • Byte 0: Data Identifier (Virtual Channel ID + Data Type)
- 1216 • Bytes 1-2: Word Count N (N = 0 to 65, 535)
- 1217 • ECC byte covering bytes 0-2
- 1218 ○ If ECC is not calculated by the peripheral, send 00h
- 1219 • N Bytes: READ data, may be from 1 to N bytes
- 1220 • Checksum, two bytes (16-bit checksum)
- 1221 ○ If Checksum is not calculated by the peripheral, send 0000h

1222 **8.9.5 Error-Reporting Format**

1223 An error report is comprised of two bytes following the DI byte, with an ECC byte following the error
 1224 report bytes. By convention, detection and reporting of each error type is signified by the corresponding bit
 1225 set to “1”. Table 17 shows the bit assignment for all error reporting.

1226

Table 17 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	HS Receive Timeout Error
6	reserved
7	reserved
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	reserved
14	reserved
15	reserved

1227 8.10 Peripheral-to-Processor Transactions – Detailed Format Description

1228 Table 18 presents the complete set of peripheral-to-processor Data Types.

1229 **Table 18 Data Types for Peripheral-sourced Packets**

Data Type, hex	Data Type, binary	Description	Packet Size
00h – 01h	00 000x	Reserved	Short
02h	00 0010	Acknowledge with Error Report	Short
03h – 0Fh	00 0011 – 00 1111	Reserved	
10h – 17h	01 0xxx	Generic Short READ Response, xxx = number of bytes returned	Short

Data Type, hex	Data Type, binary	Description	Packet Size
18h	01 1000	Reserved	
19h	01 1001	Acknowledge	Short
1Ah	01 1010	Generic Long READ Response	Long
1Bh	01 1011	Reserved	
1Ch	01 1100	DCS Long READ Response	Long
1Dh – 1Fh	01 1101 – 01 1111	Reserved	
20h – 27h	10 0xxx	DCS Short READ Response, 0-7 parameters, bits 2:0 = parameter count	Short
28h	10 1000	Reserved	
29h – 3Fh	10 1001 – 11 1111	Reserved	

1230 **8.10.1 Acknowledge with Error Report, Data Type 00 0010 (02h)**

1231 *Acknowledge with Error Report* is sent in response to any command, or read request, with BTA asserted
 1232 when a reportable error is detected in the preceding transmission from the host processor. In the case of a
 1233 correctible ECC error, this packet is sent following the requested READ data packet in the same LP
 1234 transmission.

1235 **8.10.2 Generic Short Read Response with Optional ECC, Data Type 01 0xxx (10h – 17h)**

1236 This is the short-packet response to *Generic Read Request*. Packet composition is the Data Identifier (DI)
 1237 byte, up to seven bytes of payload data followed by optional ECC byte. DT bits [2:0] indicate the number
 1238 of payload data bytes in the packet. If the peripheral is ECC-capable, it shall check the incoming request for
 1239 errors, and return the requested READ data with ECC byte appended to the packet covering up to eight
 1240 bytes (DI + payload data).

1241 This form of data transfer may be used for other features incorporated on the peripheral, such as a touch-
 1242 screen integrated on the display module. Data formats for such applications are outside the scope of this
 1243 specification.

1244 **8.10.3 Generic Long Read Response with Optional ECC and Checksum, Data Type = 01** 1245 **1010 (1Ah)**

1246 This is the long-packet response to *Generic Read Request*. Packet composition is the Data Identifier (DI)
 1247 byte followed by a two-byte Word Count, an ECC byte, N bytes of payload, and a two-byte Checksum. If
 1248 the peripheral is ECC-capable, it shall check the incoming command for errors and return the requested
 1249 READ data with ECC byte appended to the Packet Header (DI + Word Count). If the peripheral does not
 1250 support ECC it shall return 00h. If the peripheral is Checksum capable, it shall return a calculated two-byte

1251 Checksum appended to the N-byte payload data. If the peripheral does not support Checksum it shall return
1252 0000h.

1253 If the command itself is possibly corrupt, due to an uncorrectable ECC error, SoT or SoT Sync error, the
1254 requested READ data packet shall not be sent after the *Acknowledge with Error Report* packet.

1255 **8.10.4 DCS Long Read Response with Optional ECC and Checksum, Data Type 01 1100**
1256 **(1Ch)**

1257 This is a Long packet response to *DCS Read Request*. Packet composition is the Data Identifier (DI) byte
1258 followed by a two-byte Word Count, an ECC byte, N bytes of payload, and a two-byte Checksum. If the
1259 peripheral is ECC-capable, it shall check the incoming command for errors and return the requested READ
1260 data with ECC byte appended to the header (DI + Word Count). If the peripheral does not support ECC it
1261 shall return 00h. If the peripheral is Checksum capable, it shall return a calculated two-byte Checksum
1262 appended to the N-byte payload data. If the peripheral does not support Checksum it shall return 0000h.

1263 If the DCS command itself is possibly corrupt, due to uncorrectable ECC error, SoT or SoT Sync error, the
1264 requested READ data packet shall not be sent after the *Acknowledge with Error Report* packet.

1265 **8.10.5 DCS Short Read Response with Optional ECC, Data Type 10 0xxx (20h – 27h)**

1266 This is the short-packet response to *DCS Read Request*. Packet composition is the Data Identifier (DI) byte
1267 followed by up to seven bytes of payload data followed by an ECC byte. Data Type (DT) bits [2:0] indicate
1268 the number of payload bytes in the packet. If the peripheral is ECC-capable, it shall check the incoming
1269 request for errors, and return the requested READ data with ECC byte appended to the packet covering up
1270 to eight bytes (DI + payload data).

1271 **8.10.6 Multiple-packet Transmission and Error Reporting**

1272 A peripheral shall flag and report all errors that are detected in a transmission, if bus possession is given to
1273 the peripheral at the end of the transmission. Only one ACK + Error Report shall be returned per
1274 transmission, regardless of the number of packets in the transmission. If a transmission contained multiple
1275 packets it may not be possible to associate a particular error with the packet that generated it.

1276 If collecting error reports from each and every packet is a high priority, software can send command and
1277 data packets individually, one per transmission. In addition, a peripheral may choose to store accumulated
1278 results in memory on the peripheral, and the host processor may recover the record with a block read from
1279 memory at a later time.

1280 **8.10.7 Clearing Error Bits**

1281 Once reported, DSI error flags shall be cleared by the peripheral. If bus possession is not given to the
1282 peripheral before the next processor-to-peripheral transmission, any error information from the first
1283 transmission shall be cleared from the DSI error register before reporting the error information for the next
1284 processor-to-peripheral transmission. Note that this does not preclude retaining the error information
1285 internally on the peripheral. However it is not stored and transmitted as part of a subsequent ACK + Error
1286 Report response.

1287 **8.11 Video Mode Interface Timing**

1288 Video Mode peripherals require pixel data delivered in real time. This section specifies the format and
1289 timing of DSI traffic for this type of display module.

1290 8.11.1 Traffic Sequences

1291 The host processor shall support all of the traffic sequences in this section. A Video Mode peripheral shall
 1292 support at least one of the traffic sequences in this section. The peripheral shall not require any additional
 1293 constraints regarding traffic sequence or packet timing. The peripheral supplier shall document all relevant
 1294 timing parameters listed in Table 19.

1295 In the following figures BLLP is defined as a period during which video packets such as pixel-stream and
 1296 sync event packets are not actively transmitted to the peripheral.

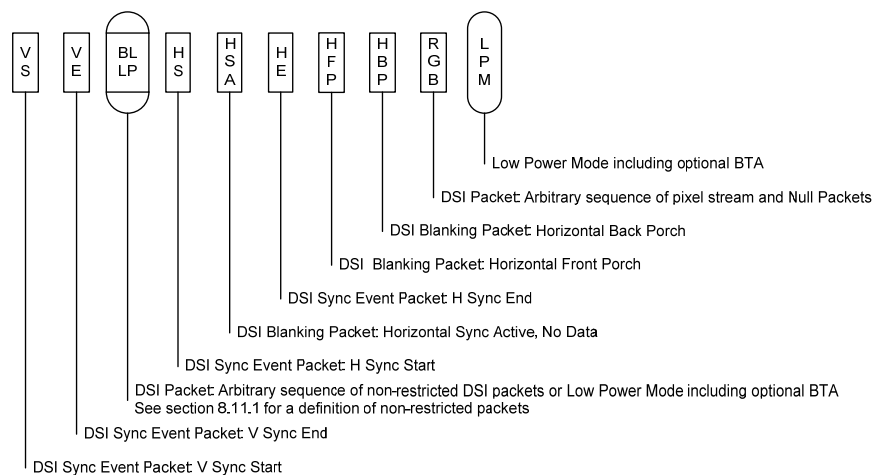
1297 To enable PHY synchronization the host processor should periodically end HS transmission and drive the
 1298 Data Lanes to the LP state. This transition should take place at least once per frame; shown as LPM in the
 1299 figures in this section. It is recommended to return to LP state once per scanline during the horizontal
 1300 blanking time. Regardless of the frequency of BLLP periods, the host processor is responsible for meeting
 1301 all documented peripheral timing requirements. Note, at lower frequencies BLLP periods will approach, or
 1302 become, zero, and burst mode will be indistinguishable from non-burst mode.

1303 During the BLLP the DSI Link may do any of the following:

- 1304 • Remain in Idle Mode with the host processor in LP-11 state and the peripheral in LP-RX
- 1305 • Transmit one or more non-video packets from the host processor to the peripheral using Escape
 1306 Mode
- 1307 • Transmit one or more non-video packets from the host processor to the peripheral using HS Mode
- 1308 • If the previous processor-to-peripheral transmission ended with BTA, transmit one or more
 1309 packets from the peripheral to the host processor using Escape Mode
- 1310 • Transmit one or more packets in HS Mode from the host processor to a different peripheral using a
 1311 different Virtual Channel ID

1312 In HS transmissions containing multiple packets, such as BLLP and RGB, the sequence of packets is
 1313 arbitrary. The host processor may compose any sequence of packets, including iterations, within the limits
 1314 of the packet format definitions. For all timing cases, the first line of a frame shall start with VS; all other
 1315 lines shall start with HS. This is also true in the special case when VSA+VBP=0. Note that the position of
 1316 synchronization packets, such as VS and HS, in time is of utmost importance since this has a direct impact
 1317 on the visual performance of the display panel.

1318 Traffic units used in the figures in this section are defined in Figure 19 unless otherwise specified.



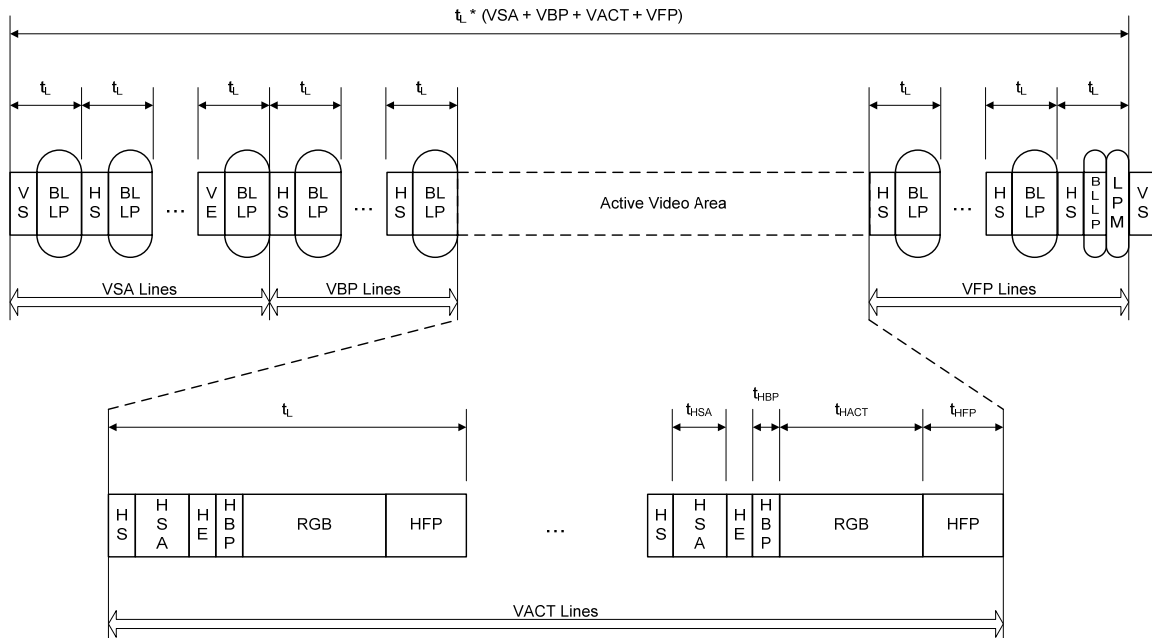
1319

1320

Figure 19 DSI Video Mode Interface Timing Legend

1321 **8.11.2 Non-Burst Mode with Sync Pulses**

1322 With this format, the goal is to accurately convey DPI-type timing over the DSI serial Link. This includes
 1323 matching DPI pixel-transmission rates, and widths of timing events like sync pulses. Accordingly,
 1324 synchronization periods are defined using packets transmitting both start and end of sync pulses. An
 1325 example of this mode is shown in Figure 20.

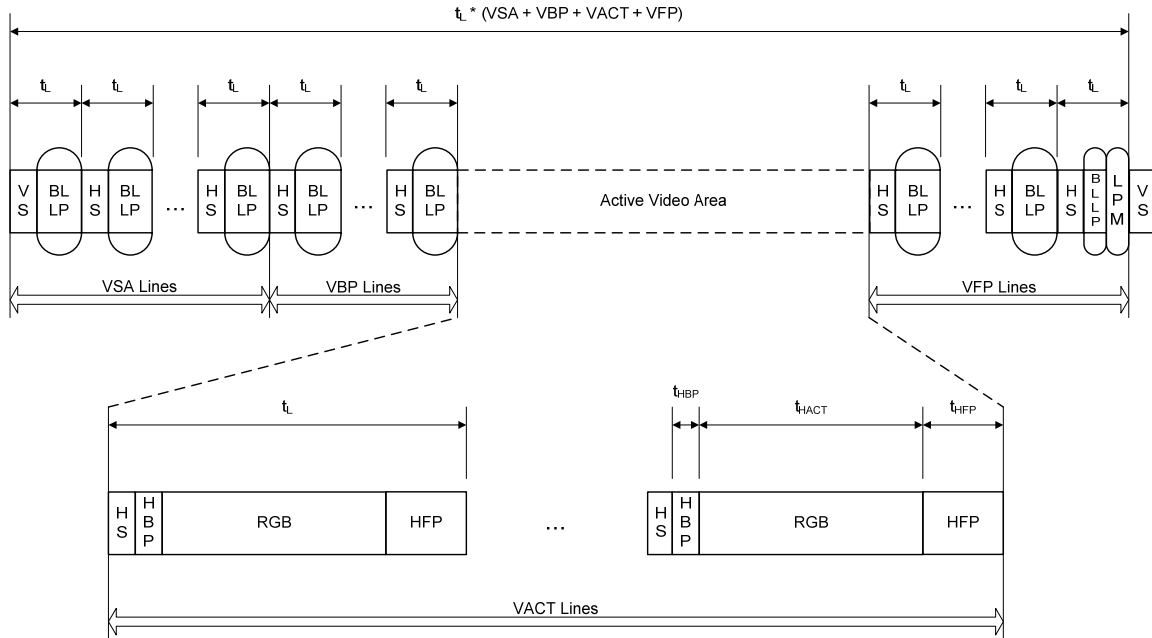


1326
1327

1328 **Figure 20 DSI Video Mode Interface Timing: Non-burst Communication with Start and End**

1329 **8.11.3 Non-Burst Mode with Sync Events**

1330 This mode is a simplification of the format described in section 8.11.2. Only the start of each
 1331 synchronization pulse is transmitted. The peripheral may regenerate sync pulses as needed from each Sync
 1332 Event packet received. Pixels are transmitted at the same rate as they would in a corresponding parallel
 1333 display interface such as DPI-2. An example of this mode is shown in Figure 21.



1334
1335

1336

Figure 21 DSI Video Mode Interface Timing: Non-burst Communication

1337

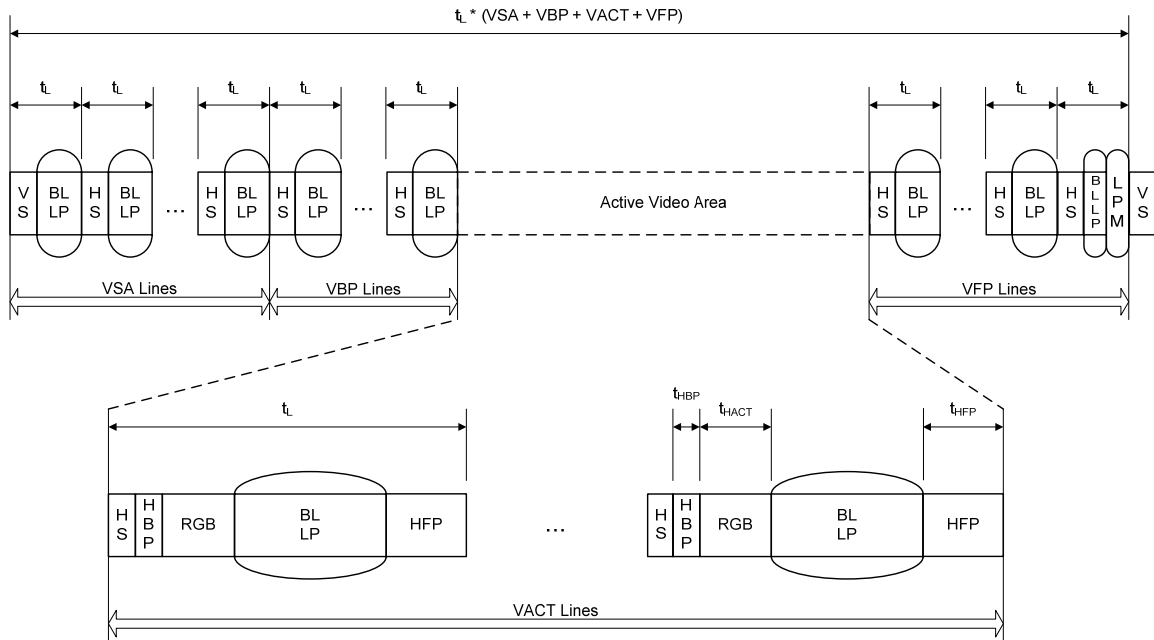
8.11.4 Burst Mode

1338
1339
1340

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

1341
1342
1343
1344
1345
1346

There may be a line buffer or similar memory on the peripheral to accommodate incoming data at high speed. Following HS pixel data transmission, the bus goes to Low Power Mode, during which it may remain idle, i.e. the host processor remains in LP-11 state, or LP transmission may take place in either direction. If the peripheral takes control of the bus for sending data to the host processor, its transmission time shall be limited to ensure data underflow does not occur from its internal buffer memory to the display device. An example of this mode is shown in Figure 22.



1347
1348

1349

Figure 22 DSI Video Mode Interface Timing: Burst Communication

1350

8.11.5 Parameters

1351
1352
1353

Table 19 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.

1354
1355
1356

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 *MIPI Alliance Standard for D-PHY* [4].

1357

Table 19 Required Peripheral Timing Parameters

Parameter	Description	Minimum	Maximum	Units	Comment
br_{PHY}	Bit rate total on all Lanes			Mbps	Depends on PHY implementation
t_L	Line time			μs	Define range to meet frame rate
t_{HBP}	Horizontal back porch			μs	
t_{HACT}	Time for image data			μs	Defining min = 0 allows max PHY speed
HACT	Active pixels per line			pixels	
t_{HFP}	Horizontal front porch			μs	No upper limit as long as line time is met

Parameter	Description	Minimum	Maximum	Units	Comment
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	

1358 8.12 TE Signaling in DSI

1359 A Command Mode display module has its own timing controller and local frame buffer for display refresh.
 1360 In some cases the host processor needs to be notified of timing events on the display module, e.g. the start
 1361 of vertical blanking or similar timing information. In a traditional parallel-bus interface like DBI-2, a
 1362 dedicated signal wire labeled TE (Tearing Effect) is provided to convey such timing information to the host
 1363 processor. In a DSI system, the same information, with reasonably low latency, shall be transmitted from
 1364 the display module to the host processor when requested, using the bidirectional Data Lane.

1365 The PHY for DSI has no inherent interrupt capability from peripheral to host processor so the host
 1366 processor shall either rely on polling, or it shall give bus ownership to the peripheral for extended periods,
 1367 as it does not know when the peripheral will send the TE message.

1368 The TE-reporting function is enabled and disabled by three DCS commands to the display module's
 1369 controller: `set_tear_on`, `set_tear_at_line_on`, and `set_tear_off`. See *MIPI Alliance Standard for Display*
 1370 *Command Set* [1] for details.

1371 `set_tear_on` and `set_tear_at_line_on` are sent to the display module as DSI Data Type 19h (DCS Short
 1372 Write, one parameter and two parameters, respectively) along with the `set_tear_on` or `set_tear_at_line_on`
 1373 command byte. The host processor ends the transmission with Bus Turn-Around asserted, giving bus
 1374 possession to the display module. Since the display module's DSI Protocol layer does not interpret DCS
 1375 commands, but only passes them through to the display controller, it responds with a normal *Acknowledge*
 1376 and returns bus possession to the host processor. In this state, the display module cannot report TE events
 1377 to the host processor since it does not have bus possession.

1378 To enable TE-reporting, the host processor shall give bus possession to the display module without an
 1379 accompanying DSI command transmission after TE reporting has been enabled. This is accomplished by
 1380 the host processor's protocol logic asserting (internal) Bus Turn-Around signal to its D-PHY functional
 1381 block. The PHY layer will then initiate a Bus Turn-Around sequence in LP mode, which gives bus
 1382 possession to the display module.

1383 Since the timing of a TE event is, by definition, unknown to the host processor, the host processor shall
 1384 give bus possession to the display module and then wait for up to one video frame period for the TE
 1385 response. During this time, the host processor cannot send new commands, or requests to the display
 1386 module, because it does not have bus possession.

1387 When the TE event takes place the display module shall send TE event information in LP mode using a
 1388 specified trigger message available with D-PHY protocol via the following sequence:

- 1389 • The display module shall send the LP Escape Mode sequence
- 1390 • The display module shall then send the trigger message byte 01011101 (shown here in first bit to
 1391 last bit sequence)

- 1392 • The display module shall then return bus possession to the host processor
- 1393 This Escape Mode sequence is reserved by DSI for TE signaling only and shall not be used for any other
1394 purpose in a DSI-compliant interface.
- 1395 See *MIPI Alliance Standard for Display Command Set* [1] for detailed descriptions of the TE related
1396 commands, and command and parameter formats.

1397 9 Error-Correcting Code (ECC) and Checksum

1398 9.1 Hamming Code for Packet Header Error Detection/Correction

1399 The host processor in a DSI-based system shall generate an error-correction code (ECC) and append it to
 1400 the header of every packet sent to the peripheral. The ECC takes the form of a single byte following the
 1401 header bytes. It shall provide single-bit error correction and 2-bit error detection for the DI (Data Identifier)
 1402 byte and up to seven additional bytes of the Packet Header, including all header parameters and two-byte
 1403 Word Count (WC) for Long packets.

1404 ECC shall always be generated and appended in the Packet Header from the host processor. Generating and
 1405 sending ECC from peripherals to the host is optional. However, the packet format is fixed; a peripheral that
 1406 does not support ECC shall send a byte having value 00h in place of the ECC byte.

1407 Peripherals in unidirectional DSI systems, although they cannot report errors to the host, may still take
 1408 advantage of ECC for correcting single-bit errors in the Packet Header.

1409 The number of parity or error check bits required is given by the Hamming rule, and is a function of the
 1410 number of bits of information transmitted. The Hamming rule is expressed by the following inequality:

1411
$$d + p + 1 \leq 2^p$$
 where d is the number of data bits and p is the number of parity bits.

1412 The result of appending the computed parity bits to the data bits is called the Hamming code word. The size
 1413 of the code word c is $d+p$, and a Hamming code word is described by the ordered set (c, d) . For DSI, eight
 1414 bytes (64-bits) of data are protected by 8-bits of computed parity, so the set is written (72, 64).

1415 A Hamming code word is generated by multiplying the data bits by a generator matrix \mathbf{G} . This
 1416 multiplication's result is called the code word vector $(c1, c2, c3, \dots, cn)$, consisting of the original data bits
 1417 and the calculated parity bits. The generator matrix \mathbf{G} used in constructing Hamming codes consists of \mathbf{I} ,
 1418 the identity matrix, and a parity generation matrix \mathbf{A} :

$$1419 \quad \mathbf{G} = [\mathbf{I} \mid \mathbf{A}]$$

1420 The Packet Header plus the ECC code can be obtained as: $\text{PH} = \text{p} * \mathbf{G}$ where p represents the header and \mathbf{G} is
 1421 the corresponding generator matrix.

1422 Validating the received code word r involves multiplying it by a parity check to form s , the syndrome or
 1423 parity check vector: $\text{s} = \mathbf{H} * \text{PH}$ where PH is the received Packet Header and \mathbf{H} is the parity check matrix:

$$1424 \quad \mathbf{H} = [\mathbf{A}^T \mid \mathbf{I}]$$

1425 If all elements of s are zero, the code word was received correctly. If s contains non-zero elements, then at
 1426 least one error is present. If the header has a single-bit error, then the syndrome s matches one of the
 1427 elements of \mathbf{H} , which will point to the bit in error. Furthermore, if the bit in error is a parity bit, then the
 1428 syndrome will be one of the elements on \mathbf{I} , or else it will be the data bit identified by the position of the
 1429 syndrome in \mathbf{A}^T .

1430 9.2 Hamming-modified Code for DSI

1431 For DSI, the error correcting code used is a 7+1 bits Hamming-modified code (72, 64). This class of
 1432 Hamming code can correct a single-bit error or detect a two-bit error, but is not capable of doing both
 1433 simultaneously, so one extra parity bit is added. The code used, is built to allow same syndromes to correct

1434 first 24-bits in a 64-bit sequence and those syndromes to be 6-bits wide. To specify in a compact way the
 1435 encoding of parity and decoding of syndromes, the following matrix is used:

1436

Table 20 ECC Syndrome Association Matrix

	0b1000	0b0000	0b0001	0b0010	0b0011	0b0100	0b0101	0b0110	0b0111
0b1000	0x07	0x0B	0x0D	0x0E	0x13	0x15	0x16	0x19	
0b0001	0x1A	0x1C	0x23	0x25	0x26	0x29	0x2A	0x2C	
0b0010	0x31	0x32	0x34	0x38	0x1F	0x2F	0x37	0x3B	
0b0011	0x43	0x45	0x46	0x49	0x4A	0x4C	0x51	0x52	
0b0100	0x54	0x58	0x61	0x62	0x64	0x68	0x70	0x83	
0b0101	0x85	0x86	0x89	0x8A	0x3D	0x3E	0x4F	0x57	
0b0110	0x8C	0x91	0x92	0x94	0x98	0xA1	0xA2	0xA4	
0b0111	0xA8	0xB0	0xC1	0xC2	0xC4	0xC8	0xD0	0xE0	

1437 Each cell in the matrix represents a syndrome and each syndrome in the matrix is MSB left aligned:

1438 e.g. 0x07=0b0000_0111=P7P6P5P4P3P2P1P0

1439 The top row defines the three LSB of data position bit, and the left column defines the three MSB of data
 1440 position bit for a total of 64-bit positions.

1441 e.g. 37th bit position is encoded 0b100_101 and has the syndrome 0x68.

1442 To correct a single bit error, the syndrome shall be one of the syndromes in the table, which will identify
 1443 the bit position in error. The syndrome is calculated as:

1444 $S = P_{SEND} \hat{P}_{RECEIVED}$ where P_{SEND} is the 8-bit ECC field in the header and $P_{RECEIVED}$ is the
 1445 calculated parity of the received header.

1446 Table 21 represents the same information as in Table 20, organized to provide better insight into how parity
 1447 bits are formed from data bits.

1448

Table 21 ECC Parity Generation Rules

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

Bit	P7	P6	P5	P4	P3	P2	P1	P0	Hex
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

Bit	P7	P6	P5	P4	P3	P2	P1	P0	Hex
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
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

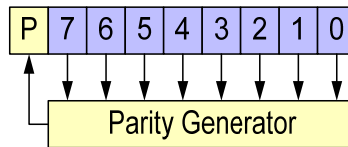
Bit	P7	P6	P5	P4	P3	P2	P1	P0	Hex
58	1	1	0	0	0	0	0	1	0xC1
59	1	1	0	0	0	0	1	0	0xC2
60	1	1	0	0	0	1	0	0	0xC4
61	1	1	0	0	1	0	0	0	0xC8
62	1	1	0	1	0	0	0	0	0xD0
63	1	1	1	0	0	0	0	0	0xE0

1449 To derive parity byte P7, the “ones” in the P7 column define if the corresponding bit position D_i (as noted
1450 in the green column) is used in calculation of P7 parity bit or not. For example,

1451 $P7 = D39 \wedge D40 \wedge D41 \wedge D42 \wedge D43 \wedge D48 \wedge D49 \wedge D50 \wedge D51 \wedge D52 \wedge D53 \wedge D54 \wedge D55 \wedge D56 \wedge D57 \wedge D58 \wedge D59$
1452 $\wedge D60 \wedge D61 \wedge D62 \wedge D63$

1453 9.3 ECC Generation on the Transmitter and Byte-Padding

1454 ECC can be generated using a parallel approach as depicted in Figure 23 for a 64-bit header:



1455

1456

Figure 23 64-bit ECC generation on TX side

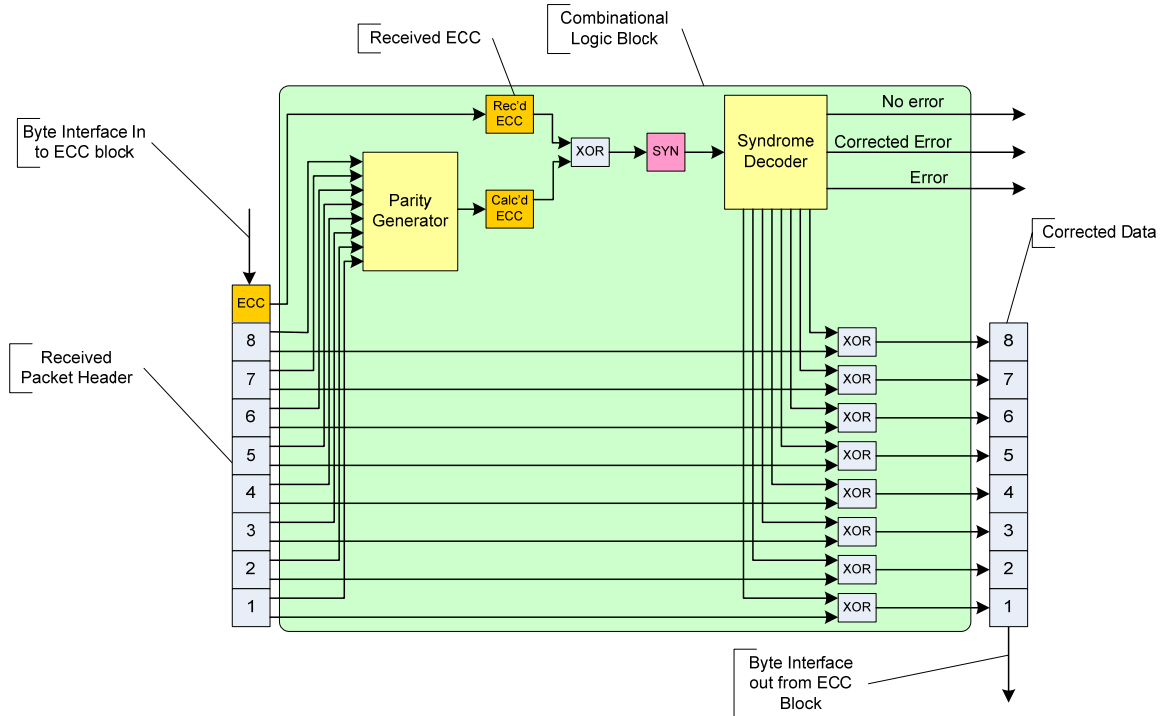
1457 Note that the DSI protocol permits headers, not including the ECC byte itself, to vary in length from one to
1458 eight bytes. Since ECC generation for DSI requires a fixed word length of 64-bits, any header shorter than
1459 eight bytes shall be padded with additional bytes to form a full eight-byte value for ECC generation and
1460 checking. All “pad” bytes shall be appended to the MSB side of the Packet Header – that is, to the left of
1461 the Data Identifier byte. All padding bytes shall take the value 00h for the purpose of generating the ECC
1462 byte.

1463 Peripherals that do not support ECC generation or checking shall transmit a byte having value 00h in place
1464 of the ECC byte, when sending packets to the host processor. The host processor shall disable ECC
1465 checking for received headers from peripherals that do not support ECC generation.

1466 9.4 Applying ECC and Byte-Padding on the Receiver

1467 Applying ECC on RX side involves generating a new ECC for the received packet, computing the
1468 syndrome using the new ECC and the received ECC, decoding the syndrome to find if a single-error has
1469 occurred and if so, correct it. If a multiple-bit error is identified, it is flagged and reported (not applicable to
1470 unidirectional DSI, however).

1471 For headers of less than eight bytes, ECC generation on the receiver side shall apply the same byte-padding
1472 rules as ECC generation for transmission: all pad bytes shall be appended to the left of the Data Identifier
1473 byte, and all pad bytes shall take the value 00h.



1474
1475

1476

Figure 24 64-bit ECC on RX Side Including Error Correction

1477 Decoding the syndrome has three aspects:

- 1478
- Testing for errors in the Packet Header. If syndrome = 0, no errors are present.
 - 1479 • Test for a single-bit error in the Packet Header by comparing the generated syndrome with the
 - 1480 matrix in Table 20. If the syndrome matches one of the entries in the table, then a single-bit error
 - 1481 has occurred and the corresponding bit is in error. This position in the Packet Header shall be
 - 1482 complemented to correct the error. Also, if the syndrome is one of the rows of the identity matrix
 - 1483 **I**, then a parity bit is in error. If the syndrome cannot be identified then a multi-bit error has
 - 1484 occurred. In this case the Packet Header is corrupted and cannot be restored. Therefore, the Multi-
 - 1485 bit Error Flag shall be set.
 - 1486 • Correcting the single-bit error if detected, as indicated above.

1487 9.5 Checksum Generation for Long Packet Payloads

1488 Long packets are comprised of a header – protected by ECC as specified above – and a payload of 0 to
1489 $2^{16} - 1$ bytes. To detect errors in transmission of Long packets, a checksum is calculated over the payload
1490 portion of the data packet. (Note that, for the special case of zero-length payload, the 2-byte checksum is
1491 set to FFFFh).

1492 The checksum can only indicate the presence of one or more errors in the payload. Unlike ECC, the
1493 checksum does not enable error correction. For this reason, checksum calculation is not useful for
1494 unidirectional DSI implementations since the peripheral has no means of reporting errors to the host
1495 processor.

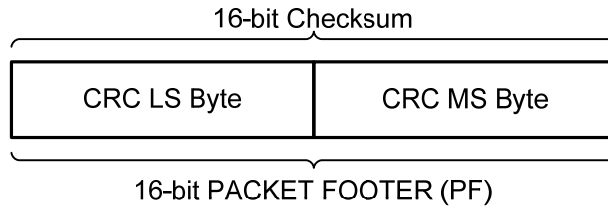
1496 Checksum generation and transmission is mandatory for host processors sending Long packets to
1497 peripherals. It is optional for peripherals transmitting Long packets to the host processor. However, the

1498 format of Long packets is fixed; peripherals that do not support checksum generation shall transmit two
 1499 bytes having value 0000h in place of the checksum bytes when sending Long packets to the host processor.

1500 The host processor shall disable checksum checking for received Long packets from peripherals that do not
 1501 support checksum generation.

1502 The checksum is realized as 16-bit CRC. The generator polynomial is $x^{16}+x^{12}+x^5+x^0$.

1503 The transmission of the checksum is illustrated in Figure 25. The LS byte is sent first, followed by the MS
 1504 byte. Note that within the byte, the LS bit is sent first.

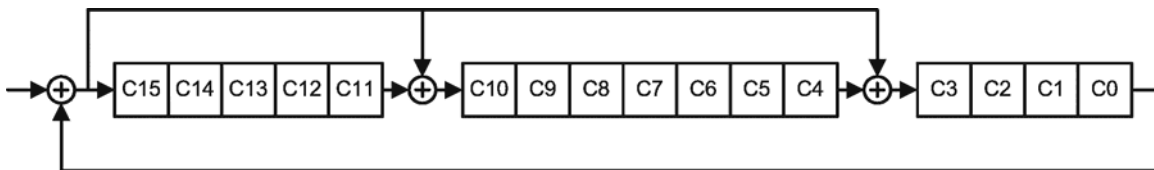


1505

1506

Figure 25 Checksum Transmission

1507 An example of CRC implementation is presented in Figure 26. The CRC shift register shall be initialized to
 1508 FFFFh before packet data enters. Packet payload data not including the header then enters as a bitwise data
 1509 stream from the left. Each bit is fed through the CRC shift register before it is passed to output for
 1510 transmission to the peripheral. After all pixels in the packet payload have passed through the CRC shift
 1511 register, the shift register contains the checksum. The checksum is then appended to the data stream and
 1512 sent over DSI to the receiver. The receiver uses its own generated CRC to verify that no errors have
 1513 occurred in transmission.



Polynomial: $x^{16} + x^{12} + x^5 + x^0$

Note: C15 represents x^0 , C0 represents x^{15}

1514

1515

Figure 26 Example implementation of CCITT 16-bit CRC generation using shift register

1516

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

1517 **10 Compliance, Interoperability, and Optional Capabilities**

1518 This section documents requirements and classifications for MIPI-compliant host processors and
 1519 peripherals. There are a number of categories of potential differences or attributes that shall be considered
 1520 to ensure interoperability between a host processor and a peripheral, such as a display module:

1521 Manufacturers shall document a DSI device's capabilities and specifications for the parameters listed in
 1522 this section.

- 1523 1. Display Resolutions
- 1524 2. Pixel Formats
- 1525 3. Number of Lanes
- 1526 4. Maximum Lane Frequency
- 1527 5. Bidirectional Communication and Escape Mode Support
- 1528 6. ECC and Checksum capabilities
- 1529 7. Display Architecture
- 1530 8. Multiple Peripheral Support

1531 In general, the peripheral chooses one option from each category in the list above. For example, a display
 1532 module may implement a resolution of 320x240 (QVGA), a pixel format of 16-bpp and use two Lanes to
 1533 achieve its required bandwidth. Its data path has bidirectional capability, it does not implement ECC or
 1534 checksum-testing capability, and it operates in Video Mode only.

1535 **10.1 Display Resolutions**

1536 Host processors shall implement one or more of the display resolutions in Table 22.

1537 **Table 22 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

1538 **10.2 Pixel Formats**

1539 Peripherals shall implement one of the following pixel formats. Host processors shall implement all of the
1540 following pixel formats.

- 1541 1. 16 bpp (5, 6, 5 RGB), each pixel using two bytes; see section 8.8.13
- 1542 2. 18 bpp (6, 6, 6 RGB) packed; see section 8.8.14
- 1543 3. 18 bpp (6, 6, 6 RGB) loosely packed into three bytes; see section 8.8.15
- 1544 4. 24 bpp (8, 8, 8 RGB), each pixel using three bytes; see section 8.8.16

1545 **10.3 Number of Lanes**

1546 In normal operation a peripheral uses the number of Lanes required for its bandwidth needs.

1547 The host processor shall implement a minimum of one Data Lane; additional Lane capability is optional. A
1548 host processor with multi-Lane capability (N Lanes) shall be able to operate with any number of Lanes
1549 from one to N, to match the fixed number of Lanes in peripherals using one to N Lanes. See section 6.1 for
1550 more details.

1551 **10.4 Maximum Lane Frequency**

1552 The maximum Lane frequency shall be documented by the DSI device manufacturer. The Lane frequency
1553 shall adhere to the specifications in *MIPI Alliance Standard for D-PHY* [4].

1554 **10.5 Bidirectional Communication**

1555 Because Command Mode depends on the use of the READ command, a Command Mode display module
1556 shall implement bidirectional communications. For display modules without on-panel buffers that work
1557 only in Video Mode, bidirectional operation on DSI is optional.

1558 Since a host processor may implement both Command- and Video Modes of operations, it should support
1559 bidirectional operation and Escape Mode transmission and reception.

1560 10.6 ECC and Checksum Capabilities

1561 A DSI host processor shall calculate and transmit an ECC byte for both Long and Short packets. The host
1562 processor shall also calculate and transmit a two-byte Checksum for Long packets. A DSI peripheral may
1563 support ECC, Checksum, or both. If a peripheral does not calculate ECC or Checksum it shall still be
1564 capable of receiving ECC and Checksum bytes from the host processor. If a peripheral supports
1565 bidirectional communications and does not support ECC or Checksum it shall send bytes of all zeros in the
1566 appropriate fields. See section 9 for more details on ECC and Checksum.

1567 10.7 Display Architecture

1568 A display module may implement Type 1, Type 2, Type 3 or Type 4 display architecture as described in
1569 *MIPI Alliance Standard for Display Bus Interface* [2] and *MIPI Alliance Standard for Display Pixel*
1570 *Interface* [3]. Type 1 architecture works in Command Mode only. Type 2 and Type 3 architectures use the
1571 DSI interface for both Command- and Video Modes of operation. Type 4 architectures operate in Video
1572 Mode only, although there may be additional control signals. Therefore, a peripheral may use Command
1573 Mode only, Video Mode only, or both Command- and Video Modes of operation.

1574 The host processor may support either or both Command- and Video Modes of operation. If the host
1575 processor supports Command Mode, it shall also support the mandatory command set specified in *MIPI*
1576 *Alliance Standard for Display Command Set* [1].

1577 10.8 Multiple Peripheral Support

1578 DSI supports multiple peripherals per DSI Link using the Virtual Channel field of the Data Identifier byte.
1579 See sections 4.2.3 and 8.5.1 for more details.

1580 A host processor should support a minimum of two peripherals.

1581 **Annex A (Informative)**
 1582 **Contention Detection and Recovery**
 1583 **Mechanisms**

1584 The following describes optional capabilities at the PHY and Protocol layers that provide additional
 1585 robustness for a DSI Link against possible data-signal contention as a consequence of transient errors in the
 1586 system. These capabilities improve the system's chances of detecting any of several possible contention
 1587 cases, and provide mechanisms for "graceful" recovery without resorting to a hard reset.

1588 These capabilities combine circuitry in the I/O cell, to directly detect contention, with logic and timers in
 1589 the protocol to avert and recover from other forms of contention.

1590 **A.1 PHY Detected Contention**

1591 The PHY can detect two types of contention faults: LP High Fault and LP Low Fault.

1592 An LP High Fault occurs when a LP transmitter is driving high and the pin voltage is less than V_{IL} .

1593 An LP Low Fault occurs when a LP transmitter is driving low and the pin voltage is greater than V_{ILF} .

1594 The LP High Fault and LP Low Fault are caused by both sides of the Link transmitting simultaneously.
 1595 Note, the LP High Fault and LP Low Fault are only applicable for bidirectional Data Lanes.

1596 **A.1.1 Protocol Response to PHY Detected Faults**

1597 The Protocol shall specify how both ends of the Link respond when contention is flagged. It shall ensure
 1598 that both devices return to *Stop* state (LP-11), with one side going to *Stop TX* and the other to *Stop RX*.

1599 When both PHYs are in LP mode, one or both PHYs will detect contention between LP-0 and LP-1.

1600 The following tables describe the resolution sequences for different types of contention and detection.

1601 Table sequences:

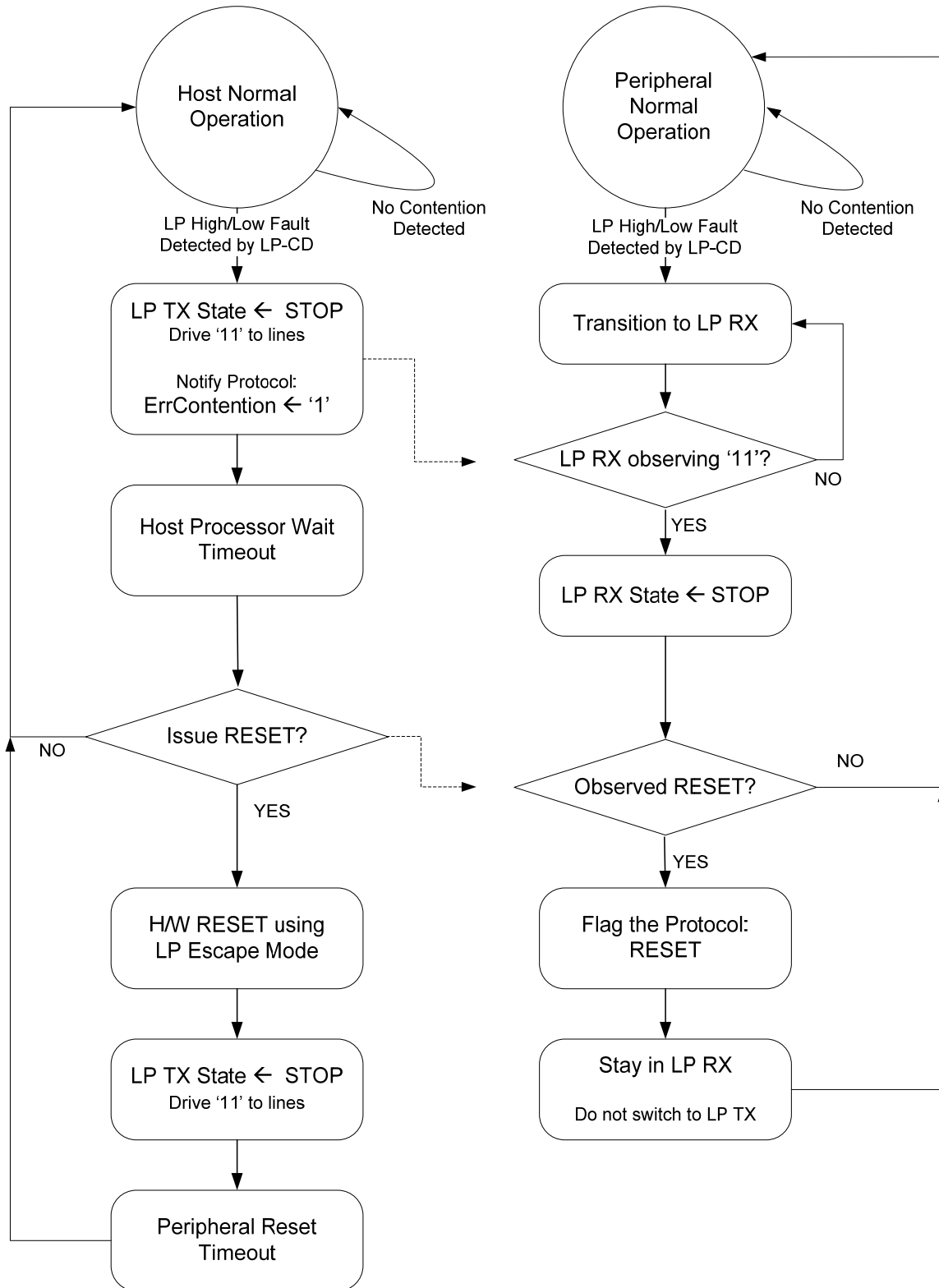
- 1602 • Sequence of events to resolve LP High \leftrightarrow LP Low Contention
- 1603 • Case 1: Both sides initially detect the contention
- 1604 • Case 2: Only the Host Processor initially detects contention
- 1605 • Case 3: Only the Peripheral initially detects contention

1606 **Table 23 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>	

Host Processor Side		Peripheral Side	
Protocol	PHY	PHY	Protocol
	Transition to <i>Stop</i> State (LP-11)	Transition to LP-RX	
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	
		Flag Protocol about Reset Command	Observe Reset Entry Command
			Reset Peripheral
	Return to <i>Stop</i> 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

1607 Note: The protocol may want to request a Reset after contention is flagged a single time. Alternately, the
1608 protocol may choose not to Reset but instead continue normal operation after detecting a single contention.
1609 It could then initiate a Reset after multiple contentions are flagged, or never initiate a Reset.



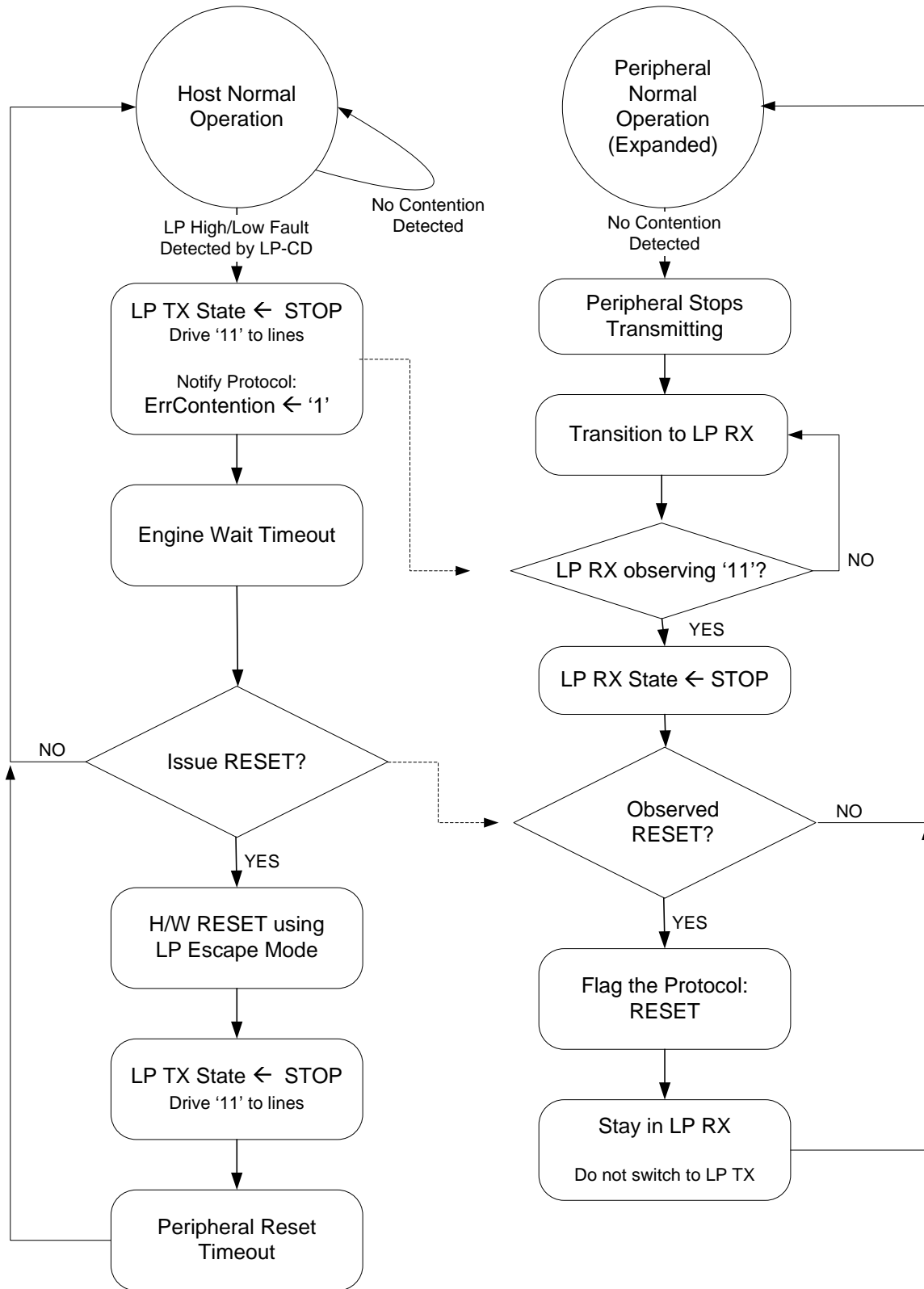
1610
1611

Figure 27 LP High ↔ LP Low Contention Case 1

1612

Table 24 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</i> state	
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</i> Command
			Reset Peripheral
	Return to <i>Stop</i> state (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



1613

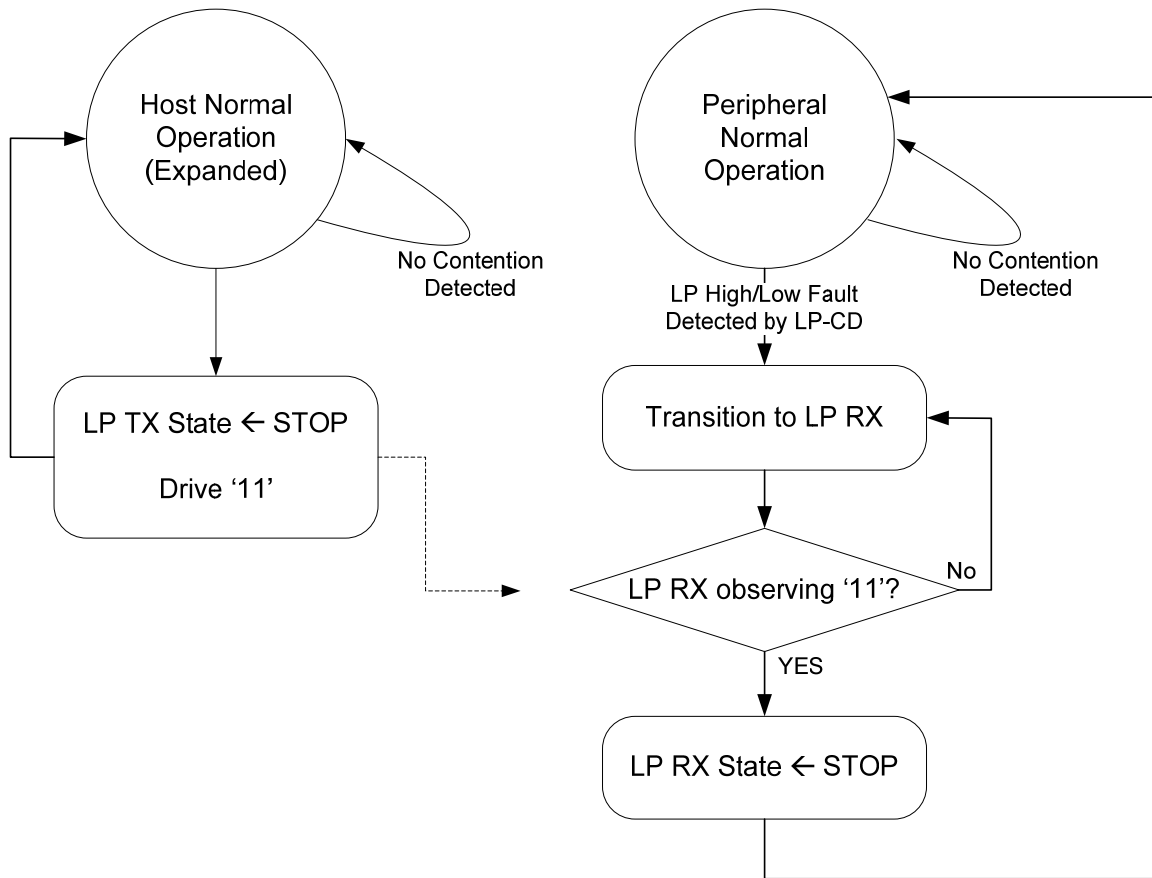
1614

Figure 28 LP High \leftrightarrow LP Low Contention Case 2

1615

Table 25 LP High ↔ LP Low Contention Case 3

Host Processor Side		Peripheral Side	
Protocol	PHY	PHY	Protocol
	No detection of EL contention	Detect <i>LP High Fault</i> or <i>LP Low Fault</i>	
		Transition to LP-RX	
		Peripheral waits until it observes <i>Stop</i> state before responding to bus activity.	
	Normal transition to <i>Stop State (LP-11)</i>	Observe <i>Stop State</i>	



1616
1617

1618

Figure 29 LP High ↔ LP Low Contention Case 3