MIPI Alliance Specification for Display Serial Interface

Version 1.01.00 – 21 February 2008

MIPI Board Approved 18-Jun-2008

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

1 **NOTICE OF DISCLAIMER**

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

10 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

Contents

165

166 # **Figures**

199

Tables

227 # MIPI Alliance Specification for Display Serial
228 # Interface

229 ## 1   Overview

230 The Display Serial Interface Specification defines protocols between a host processor and peripheral
231 devices that adhere to MIPI Alliance specifications for mobile device interfaces. The DSI specification
232 builds on existing specifications by adopting pixel formats and command set defined in MIPI Alliance
233 specifications for DBI-2[2], DPI-2[3], and DCS[1].

234 ### 1.1   Scope

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

237 Electrical specifications and physical specifications are out of scope for this document. In addition, legacy
238 interfaces such as DPI-2 and DBI-2 are also out of scope for this document. Furthermore, device usage of
239 auxiliary buses such as I^2C or SPI, while not precluded by this specification, are also not within its scope.

240 ### 1.2   Purpose

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

2   Terminology (informative)

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

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

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

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

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

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

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

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

Numbers are decimal unless otherwise indicated. Hexadecimal numbers have a "0x" prefix or an "h" suffix. Binary numbers are prefixed by "0b".

2.1   Definitions

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

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

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

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

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

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

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

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

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

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

286 **Payload:** Application data only – with all Link synchronization, header, ECC and checksum and other
287 protocol-related information removed. This is the "core" of transmissions between host processor and
288 peripheral.

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

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

292 **Reverse Direction:** Reverse direction is the opposite of the forward direction. See the description for
293 Forward Direction.

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

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

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

301 ## 2.2   Abbreviations

302 e.g.   For example

303 ## 2.3   Acronyms

304 AIP   Application Independent Protocol

305 AM   Active matrix (display technology)

306 ASP   Application Specific Protocol

307 BLLP   Blanking or Low Power interval

308 BPP   Bits per Pixel

309 BTA   Bus Turn-Around

310 CSI   Camera Serial Interface

311 DBI   Display Bus Interface

312	DI	Data Identifier
313	DMA	Direct Memory Access
314	DPI	Display Pixel Interface
315	DSI	Display Serial Interface
316	DT	Data Type
317	ECC	Error-Correcting Code
318	EMI	Electro Magnetic interference
319	EoTp	End of Transmission Packet
320	ESD	Electrostatic Discharge
321	Fps	Frames per second
322	HBP	Horizontal Back Porch
323	HFP	Horizontal Front Porch
324	HS	High Speed
325	HSA	Horizontal Sync Active
326	HSE	Horizontal Sync End
327	HSS	Horizontal Sync Start
328	ISTO	Industry Standards and Technology Organization
329	LP	Low Power
330	LPS	Low Power State (state of serial data line when not transferring high-speed serial data)
331	LSB	Least Significant Bit
332	Mbps	Megabits per second
333	MIPI	Mobile Industry Processor Interface
334	MSB	Most Significant Bit
335	PF	Packet Footer
336	PH	Packet Header
337	PHY	Physical Layer
338	PPI	PHY-Protocol Interface
339	QCIF	Quarter-size CIF (resolution 176x144 pixels or 144x176 pixels)

340	QVGA	Quarter-size Video Graphics Array (resolution 320x240 pixels or 240x320 pixels)
341	RGB	Color presentation (Red, Green, Blue)
342	SLVS	Scalable Low Voltage Signaling
343	SoT	Start of Transmission
344	SVGA	Super Video Graphics Array (resolution 800x600 pixels or 600x800 pixels)
345	ULPS	Ultra-low Power State
346	VGA	Video Graphics Array (resolution 640x480 pixels or 480x640 pixels)
347	VSA	Vertical Sync Active
348	VSE	Vertical Sync End
349	VSS	Vertical Sync Start
350	WC	Word Count
351	WVGA	Wide VGA (resolution 800x480 pixels or 480x800 pixels)

352 ## 3   References (informative)

353 [1]   *MIPI Alliance Specification for Display Command Set (DCS)*, version 1.02.00, MIPI
354   Alliance, In Press

355 [2]   *MIPI Alliance Standard for Display Bus Interface (DBI-2)*, version 2.00, MIPI Alliance,
356   29 November 2005

357 [3]   *MIPI Alliance Standard for Display Pixel Interface (DPI-2)*, version 2.00, MIPI Alliance,
358   15 September 2005

359 [4]   *MIPI Alliance Specification for D-PHY*, version 0.90.00, MIPI Alliance, 8 October 2007

360   Johnson, Barry W.; *The Design and Analysis of Fault Tolerant Digital Systems*, ISBN
361   978-0-200107570-0, Addison-Wesley, January 1989

362   Johnson, Don; Connexions, "Error Correcting Codes: Hamming Distance",
363   <http://cnx.org/content/m10283/latest/>, 3 June 2007

364   *Intel 8206 Error Detection and Correction Unit*, datasheet

365   *National DP8400-2 Expandable Error Checker/Corrector*, datasheet

366 Much of DSI is based on existing MIPI Alliance specifications as well as several MIPI Alliance
367 specifications in simultaneous development. In the Application Layer, DSI duplicates pixel formats used in
368 *MIPI Alliance Standard for Display Parallel Interface*[3] when it is in *Video Mode* operation. For display
369 modules with a display controller and frame buffer, DSI shares a common command set with *MIPI Alliance*
370 *Standard for Display Bus Interface (DBI-2)*[2]. The command set is documented in *MIPI Alliance*
371 *Specification for Display Command Set (DCS)*[1].

372 ### 3.1   Display Bus Interface Standards for Parallel Signaling (DBI and DBI-2)

373 DBI and DBI-2 are MIPI Alliance standards for parallel interfaces to display modules having display
374 controllers and frame buffers. For systems based on these standards, the host processor loads images to the
375 on-panel frame buffer through the display processor. Once loaded, the display controller manages all
376 display refresh functions on the display module without further intervention from the host processor. Image
377 updates require the host processor to write new data into the frame buffer.

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

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

385 ### 3.2   Display Pixel Interface Standards for Parallel Signaling (DPI and DPI-2)

386 DPI and DPI-2 are MIPI Alliance standards for parallel interfaces to display modules without on-panel
387 display controller or frame buffer. These display modules rely on a steady flow of pixel data from host

388   processor to the display, to maintain an image without flicker or other visual artifacts. MIPI Alliance
389   standards document several pixel formats for *Active Matrix* (AM) display modules.

390   Like DBI and DBI-2, DPI and DPI-2 are MIPI Alliance standards for parallel interfaces. The data path may
391   be 16-, 18-, or 24-bits wide, depending on pixel format(s) supported by the display module. This document
392   refers to DPI mode of operation as Video Mode.

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

3.3   MIPI Alliance Standard for Display Command Set (DCS)

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

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

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

409   • CSI-2 uses unidirectional high-speed Link, whereas DSI is half-duplex bidirectional Link

410   • CSI-2 makes use of a secondary channel, based on I^2C, for control and status functions

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

3.5   MIPI Alliance Specification for D-PHY (D-PHY)

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

417 **4 DSI Introduction**

418 DSI specifies the interface between a host processor and a peripheral such as a display module. It builds on
419 existing MIPI Alliance specifications by adopting pixel formats and command set specified in DPI-2, DBI-
420 2 and DCS standards.

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

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

433
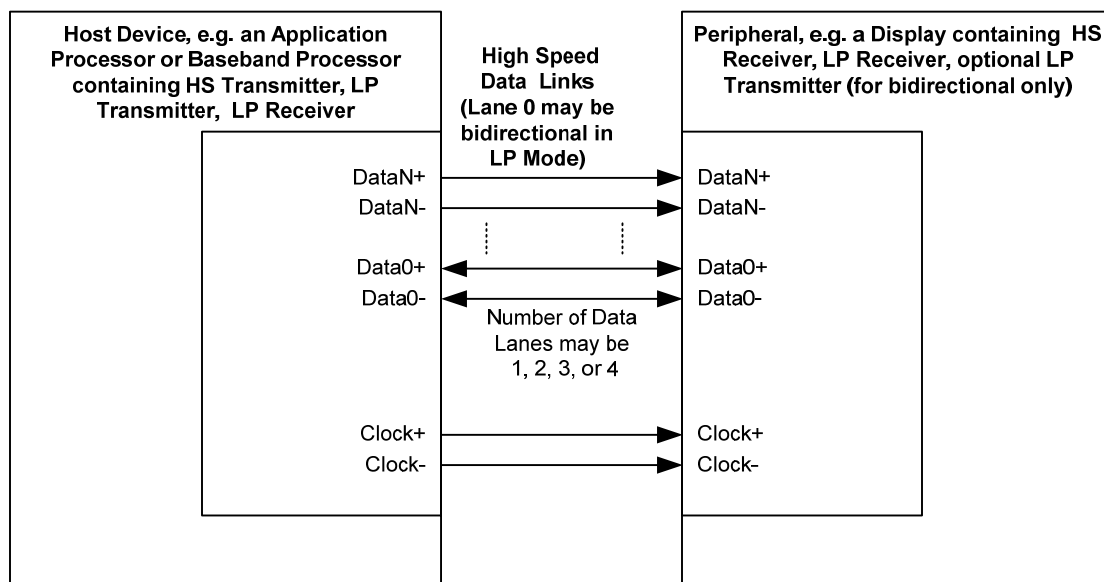
434 **Figure 1 DSI Transmitter and Receiver Interface**

435 **4.1 DSI Layer Definitions**
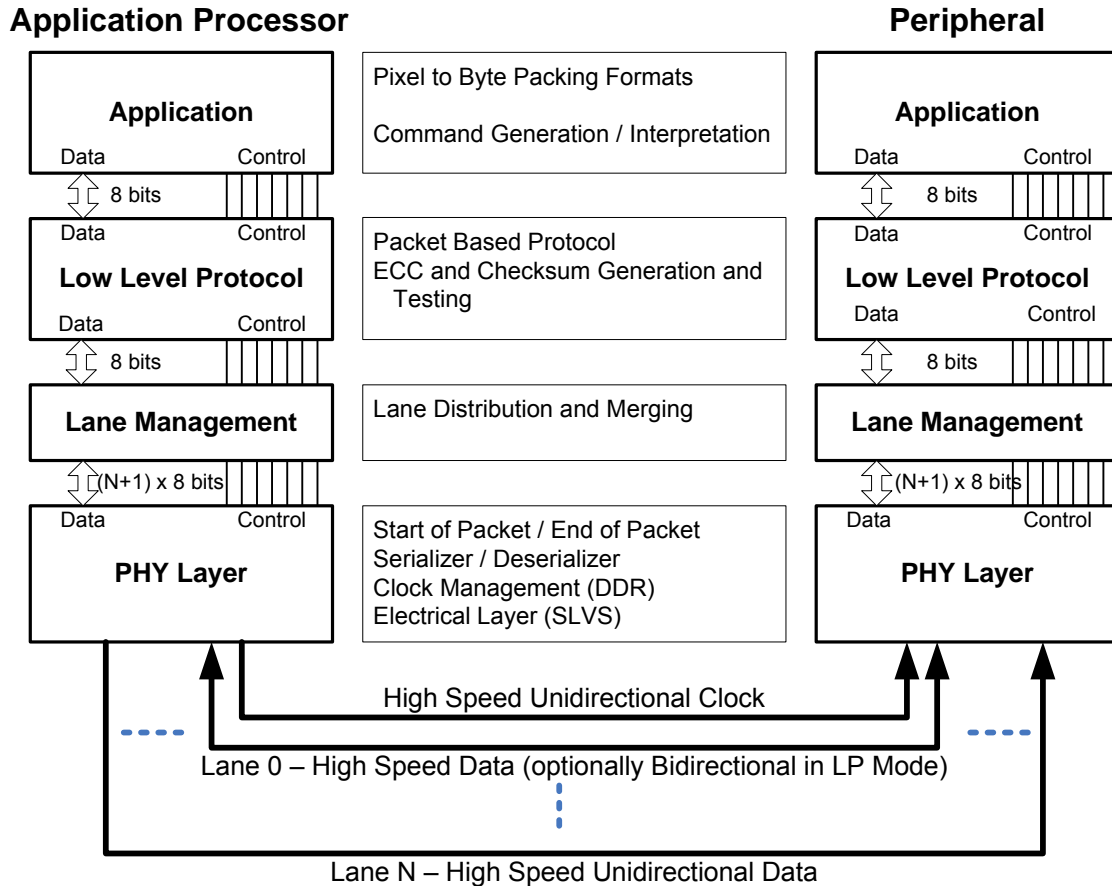
436
437 **Figure 2 DSI Layers**

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

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

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

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

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

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

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

468   4.2   Command and Video Modes

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

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

477   4.2.1   Command Mode

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

485   4.2.2   Video Mode Operation

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

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

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

4.2.3 Virtual Channel Capability

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

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

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

509 # 5   DSI Physical Layer

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

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

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

DATA 0:   SoT | Byte 0 | Byte 1 | Byte 2 | ... | Byte N-3 | Byte N-2 | Byte N-1 | EoT

518   **KEY:**
      SoT – Start of Transmission   EoT – End of Transmission
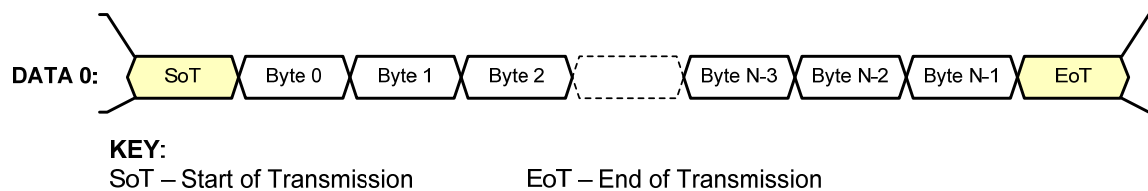
519   **Figure 3 Basic HS Transmission Structure**

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

522 ## 5.1   Data Flow Control

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

529 ## 5.2   Bidirectionality and Low Power Signaling Policy

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

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

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

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

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

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

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

5.3   Command Mode Interfaces

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

554   • Data Lane Module: CIL-MFAA (HS-TX, LP-TX, LP-RX, and LP-CD)

555   • Clock Lane Module: CIL-MCNN (HS-TX, LP-TX)

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

557   • Data Lane Module: CIL-SFAA (HS-RX, LP-RX, LP-TX, and LP-CD)

558   • Clock Lane Module: CIL-SCNN (HS-RX, LP-RX)

559   A Bidirectional Link shall support reverse-direction Escape Mode for Data Lane 0 to support LPDT for
560   read data as well as ACK and TE Trigger Messages issued by the peripheral. In the forward direction, Data
561   Lane 0 shall support LPDT as described in *MIPI Alliance Specification for D-PHY*[4]. All Trigger
562   messages shall be communicated across Data Lane 0.

5.4   Video Mode Interfaces

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

565   • Data Lane Module: CIL-MFAN (HS-TX, LP-TX)

566   • Clock Lane Module: CIL-MCNN (HS-TX, LP-TX)

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

568   • Data Lane Module: CIL-SFAN (HS-RX, LP-RX)

569   • Clock Lane Module: CIL-SCNN (HS-RX, LP-RX)

570   In the forward direction, Data Lane 0 shall support LPDT as described in *MIPI Alliance Specification for
571   D-PHY*[4]. All Trigger messages shall be communicated across Data Lane 0.

5.5   Bidirectional Control Mechanism

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

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

581   5.6   Clock Management

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

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

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

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

594   For continuous clock behavior, the Clock Lane remains in high-speed mode generating active clock signals
595   between HS data packet transmissions. For non-continuous clock behavior, the Clock Lane enters the LP-
596   11 state between HS data packet transmissions.

597   5.6.1   Clock Requirements

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

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

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

612   The handshake process for BTA allows only limited mismatch of Escape Mode clock frequencies between
613   a host processor and a peripheral. The Escape Mode frequency ratio between host processor and peripheral
614   shall not exceed 3:2. The host processor is responsible for controlling its own clock frequency to match the
615   peripheral. The host processor LP clock frequency shall be in the range of 67% to 150% of peripheral LP
616   clock frequency. Therefore, the peripheral implementer shall specify a peripheral's nominal LP clock
617   frequency and the guaranteed accuracy.

618 ## 5.6.2   Clock Power and Timing

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

623 ## 5.7   <mark>System Power-Up and Initialization</mark>

624 After power-on, the host processor shall observe an initialization period, T_{INIT}, during which it shall drive a
625 sustained TX-Stop state (LP-11) on all Lanes of the Link. See *MIPI Alliance Specification for D-PHY*[4]
626 for descriptions of T_{INIT} and the TX-Stop state.

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

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

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

637 Figure 4 illustrates an example power-up sequence for a DSI display module. In the figure, a power-on
638 reset (POR) mechanism is assumed for initialization. Internally within the display module, de-assertion of
639 POR could happen after both I/O and core voltages are stable. The worst case t_{POR} parameter can be defined
640 by the display module data sheet. t_{INIT_SLAVE} represents the minimum initialization period (T_{INIT}) defined in
641 *MIPI Alliance Specification for D-PHY*[4] for a host driving LP-11 to the display. This interval starts
642 immediately after the t_{POR} period. The peripheral might need an additional $t_{INTERNAL\_DELAY}$ time to reach a
643 functional state after power-up. In this case, $t_{INTERNAL\_DELAY}$ should also be defined in the display module
644 data sheet. In this example, the host's t_{INIT_MASTER} parameter is programmed for driving LP-11 for a period
645 longer than the sum of t_{POR}, t_{INIT_SLAVE} and $t_{INTERNAL\_DELAY}$. The display module may ignore all Lane
646 activities during this time.

647

External V$_{DD}$

Power-on
Reset

t_{POR}

I/O
Voltage

Core
Voltage

Rx State
Machine Active

t_{INIT_SLAVE} $t_{INTERNAL\_DELAY}$

Clock +

t_{INIT_MASTER}

Clock -
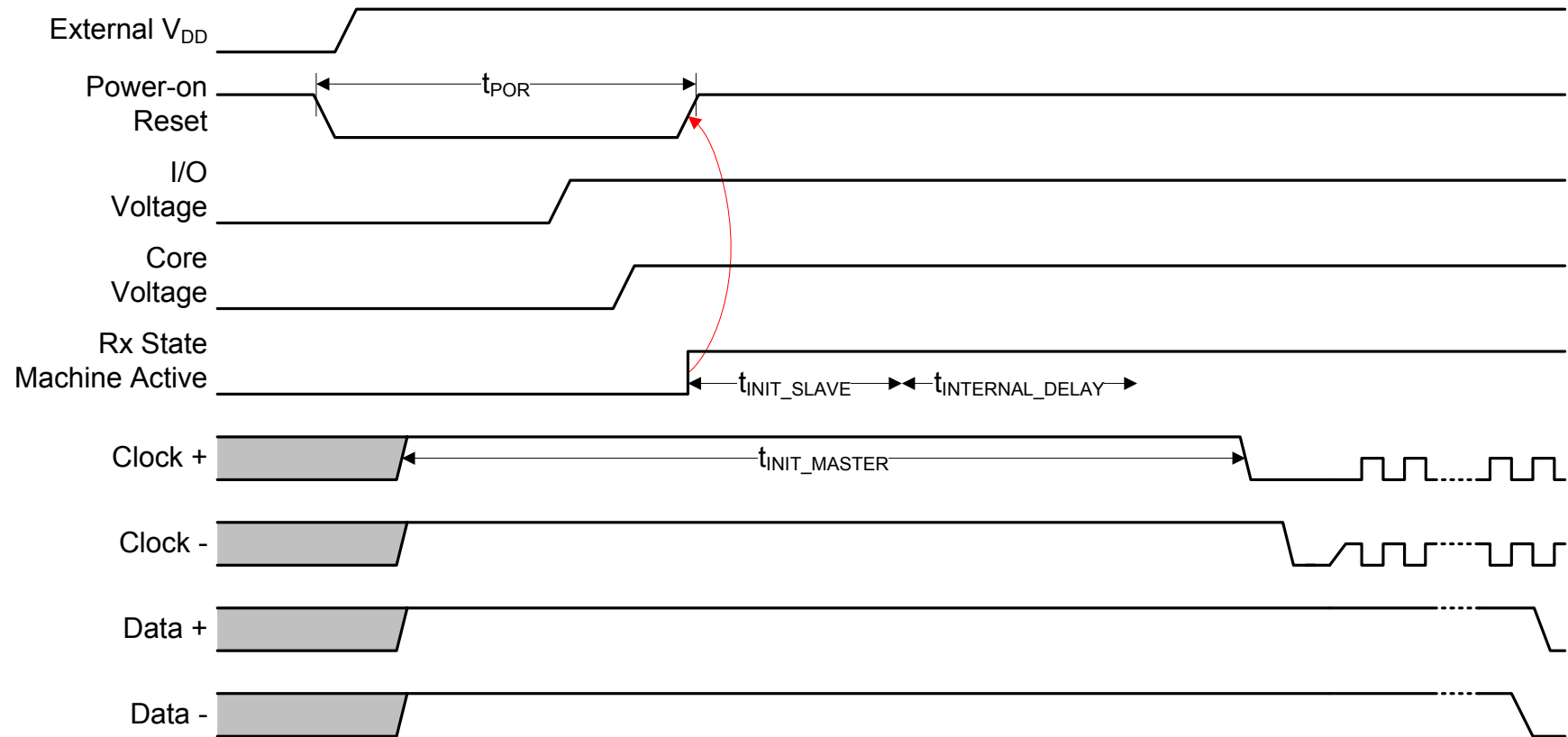
Data +

Data -

$t_{INIT\_MASTER} \geq t_{POR} + t_{INIT\_SLAVE} + t_{INTERNAL\_DELAY}$

648
649

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

651 **6   Multi-Lane Distribution and Merging**

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

657 <mark>Multi-Lane implementations shall use a single common clock signal, shared by all Data Lanes.</mark>

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

663
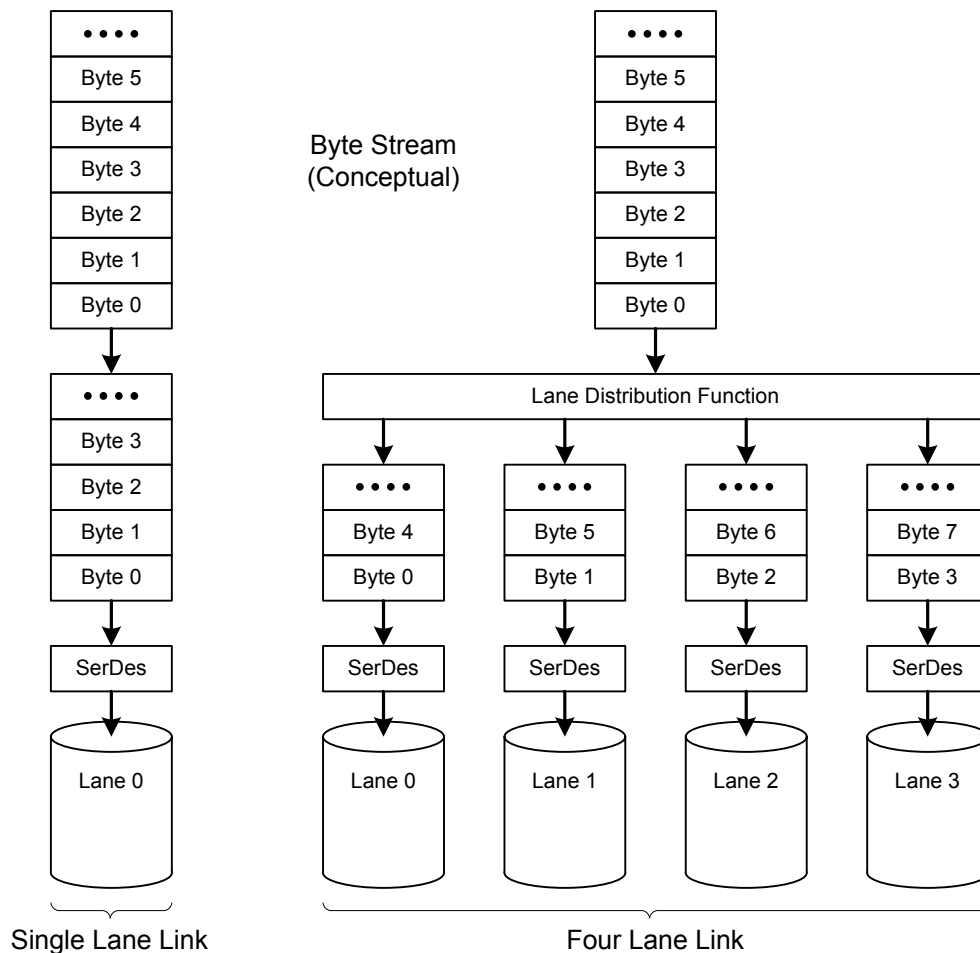
664 **Figure 5 Lane Distributor Conceptual Overview**

Single Lane Link   Four Lane Link

Lane 0   Lane 0   Lane 1   Lane 2   Lane 3

SerDes   SerDes   SerDes   SerDes   SerDes

• • • •   • • • •   • • • •   • • • •   • • • •

Byte 3   Byte 4   Byte 5   Byte 6   Byte 7
Byte 2   Byte 0   Byte 1   Byte 2   Byte 3
Byte 1
Byte 0

Lane Merging Function

• • • •   • • • •

Byte 5   Byte 5
Byte 4   Byte Stream   Byte 4
Byte 3   (Conceptual)   Byte 3
Byte 2   Byte 2
Byte 1   Byte 1
Byte 0   Byte 0

665
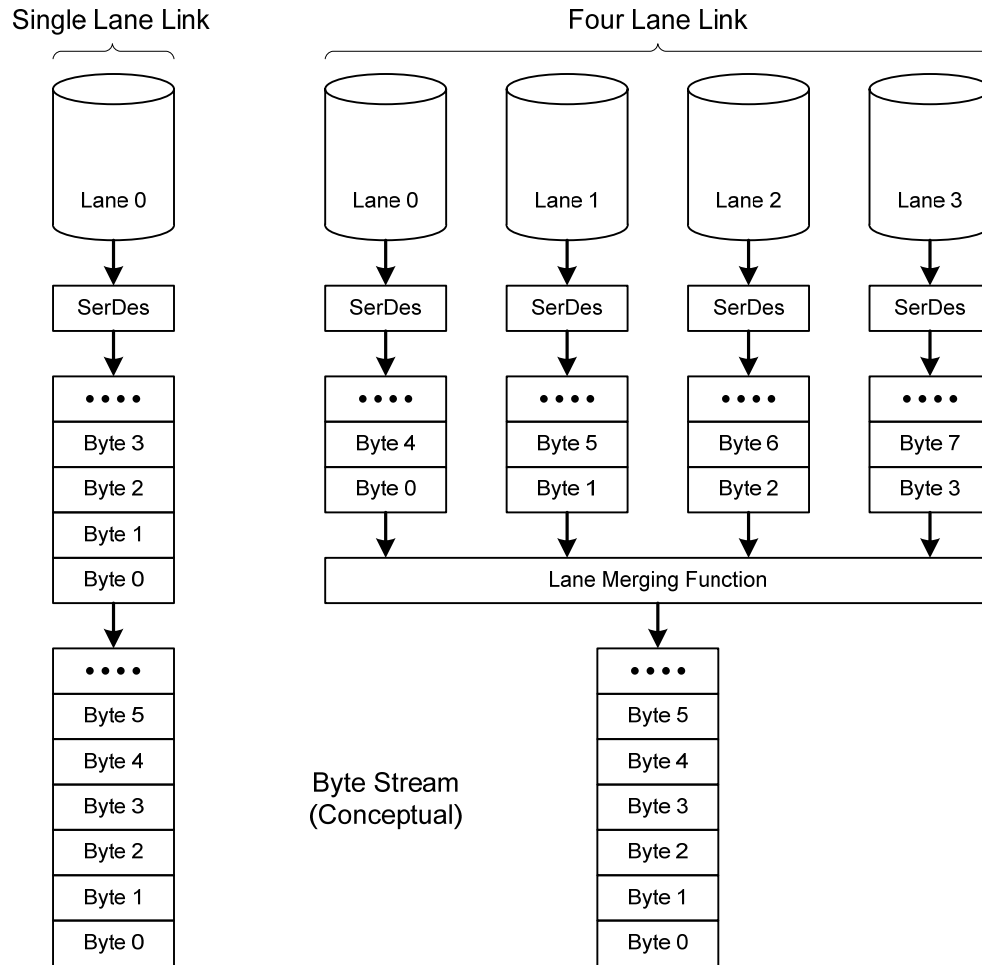
Figure 6 Lane Merger Conceptual Overview

667 The Lane Distributor takes a HS transmission of arbitrary byte length, buffers N bytes, where N is the
668 number of Lanes implemented in the interface, and sends groups of N bytes in parallel across the N Lanes.
669 Before sending data, all Lanes perform the SoT sequence in parallel to indicate to their corresponding
670 receiving units that the first byte of a packet is beginning. After SoT, the Lanes send groups of N bytes
671 from the first packet in parallel, following a round-robin process. For example, with a two Lane system,
672 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.

6.1   Multi-Lane Interoperability and Lane-number Mismatch

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

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

680
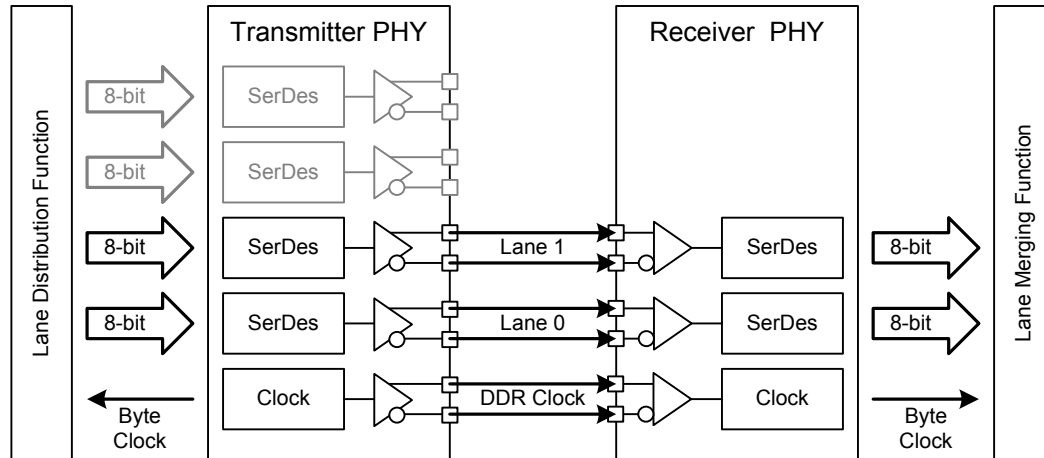
681 **Figure 7 Four-Lane Transmitter with Two-Lane Receiver Example**

682 ### 6.1.1 Clock Considerations with Multi-Lane

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

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

689 ### 6.1.2 Bi-directionality and Multi-Lane Capability

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

693 ### 6.1.3 SoT and EoT in Multi-Lane Configurations

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

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

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

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

Number of Bytes, N, transmitted is an integer multiple of the number of lanes:
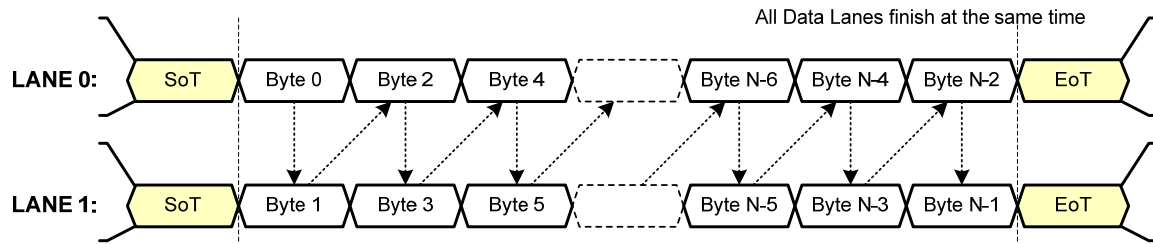
All Data Lanes finish at the same time

LANE 0: | SoT | Byte 0 | Byte 2 | Byte 4 | | Byte N-6 | Byte N-4 | Byte N-2 | EoT

LANE 1: | SoT | Byte 1 | Byte 3 | Byte 5 | | Byte N-5 | Byte N-3 | Byte N-1 | EoT

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

Data Lane 0 finishes 1 byte later than Data Lane 1

LANE 0: | SoT | Byte 0 | Byte 2 | Byte 4 | | Byte N-5 | Byte N-3 | Byte N-1 | EoT

LANE 1: | SoT | Byte 1 | Byte 3 | Byte 5 | | Byte N-4 | Byte N-2 | EoT | LPS

KEY:
LPS – Low Power State　　　SoT – Start of Transmission　　　EoT – End of Transmission
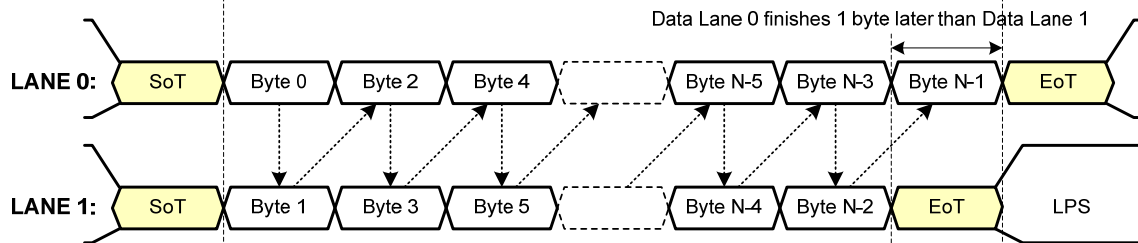
705
706　　　　　　　　　**Figure 8 Two Lane HS Transmission Example**

Number of Bytes, N, transmitted is an integer multiple of the number of lanes:
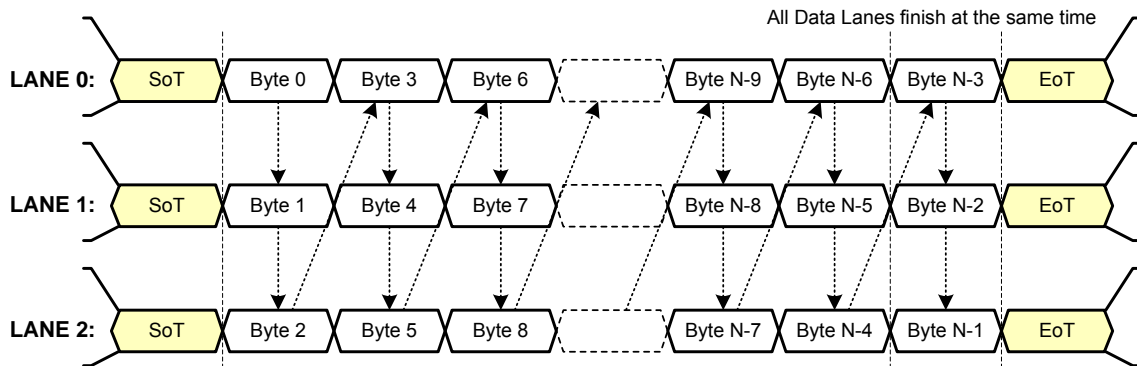
All Data Lanes finish at the same time

LANE 0:	SoT	Byte 0	Byte 3	Byte 6		Byte N-9	Byte N-6	Byte N-3	EoT
LANE 1:	SoT	Byte 1	Byte 4	Byte 7		Byte N-8	Byte N-5	Byte N-2	EoT
LANE 2:	SoT	Byte 2	Byte 5	Byte 8		Byte N-7	Byte N-4	Byte N-1	EoT

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

Data Lane 0 finishes 1 byte later than Data Lanes 1 and 2

LANE 0:	SoT	Byte 0	Byte 3	Byte 6		Byte N-7	Byte N-4	Byte N-1	EoT
LANE 1:	SoT	Byte 1	Byte 4	Byte 7		Byte N-6	Byte N-3	EoT	LPS
LANE 2:	SoT	Byte 2	Byte 5	Byte 8		Byte N-5	Byte N-2	EoT	LPS

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

Data Lanes 0 and 1 finish 1 byte later than Data Lane 2

LANE 0:	SoT	Byte 0	Byte 3	Byte 6		Byte N-8	Byte N-5	Byte N-2	EoT
LANE 1:	SoT	Byte 1	Byte 4	Byte 7		Byte N-7	Byte N-4	Byte N-1	EoT
LANE 2:	SoT	Byte 2	Byte 5	Byte 8		Byte N-6	Byte N-3	EoT	LPS

KEY:
LPS – Low Power State   SoT – Start of Transmission   EoT – End of Transmission
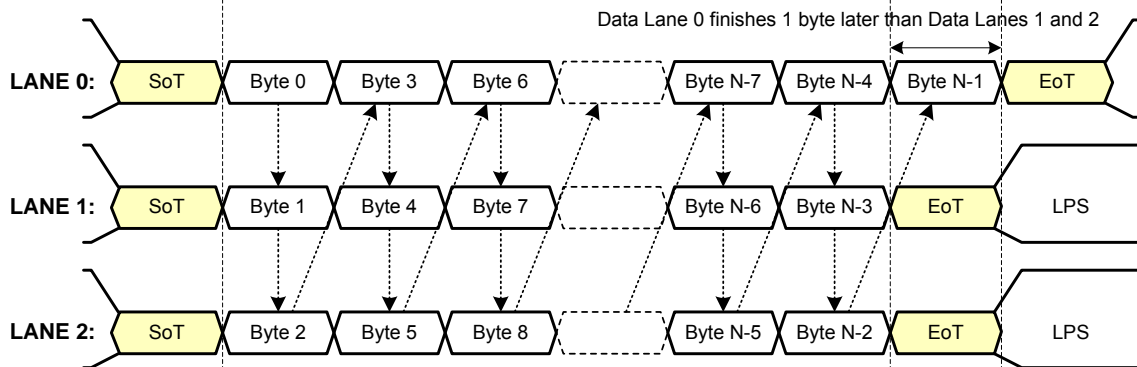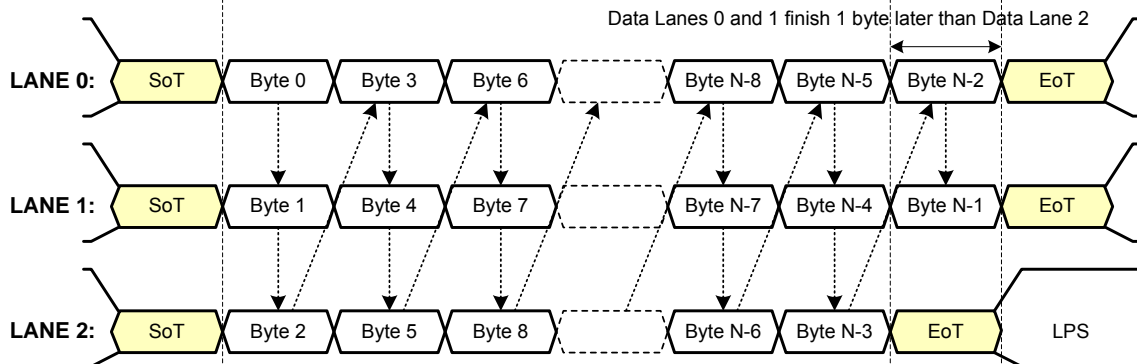
707
708   **Figure 9 Three Lane HS Transmission Example**

709 ## 7   Low-Level Protocol Errors and Contention

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

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

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

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

722   Note that contention-related features are of no benefit for unidirectional DSI Links. However, the "common
723   mode fault" can still occur in unidirectional systems.

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

726   ## 7.1   Low-Level Protocol Errors

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

730   - SoT Error

731   - SoT Sync Error

732   - EoT Sync Error

733   - Escape Mode Entry Command Error

734   - LP Transmission Sync Error

735   - False Control Error

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

739   ### 7.1.1   SoT Error

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

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

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

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

7.1.2   SoT Sync Error

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

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

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

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

7.1.3   EoT Sync Error

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

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

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

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

767   ### 7.1.4   Escape Mode Entry Command Error

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

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

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

774   ### 7.1.5   LP Transmission Sync Error

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

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

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

779   ### 7.1.6   False Control Error

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

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

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

786

787   **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

788   ## 7.2   Contention Detection and Recovery

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

793   ### 7.2.1   Contention Detection in LP Mode

794   In bidirectional Links, contention detectors in the PHY shall detect two types of contention faults: LP High
795   Fault and LP Low Fault. Refer to *MIPI Alliance Specification for D-PHY*[4] for definitions of LP High and
796   LP Low faults.

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

800   ### 7.2.2   Contention Recovery Using Timers

801   The PHY cannot detect all forms of contention. Although they do not directly detect contention, the use of
802   appropriate timers ensures that any contention that does happen is of limited duration.

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

805 **7.2.2.1   Summary of Required Contention Recovery Timers**

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

807   **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

808 **7.2.2.2   HS RX Timeout (HRX_TO) in Peripheral**

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

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

818   **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 *Stop* state before responding to bus activity.
Transition to *Stop* state (LP-11)	Observe *Stop* state and flag error

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

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

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

829 **7.2.2.3   HS TX Timeout (HTX_TO) in Host Processor**

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

834   **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 *Stop* state (LP-11)	Peripheral observes EoT and *Stop* state (LP-RX)

835 **7.2.2.4   LP TX-Peripheral Timeout (LTX-P_TO)**

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

840   **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 *Stop* state before responding to bus activity.
Drive LP-11 *Stop* state	Observe *Stop* state in LP-RX mode

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

844 **7.2.2.5   LP-RX Host Processor Timeout (LRX-H_TO)**

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

849   **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 *Stop* state (LP-11)	Peripheral observes *Stop* state in LP-RX mode

850   ## 7.3   Additional Timers

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

853   ### 7.3.1   Turnaround Acknowledge Timeout (TA_TO)

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

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

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

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

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

863   ### 7.3.2   Peripheral Reset Timeout (PR_TO)

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

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

Host Processor Side	Peripheral Side
Send *Reset Entry* command	Receive *Reset Entry* Command
Return to *Stop* state (LP-11)	Initiate reset sequence
	Complete reset sequence
Peripheral Reset Timeout	
Resume Normal Operation.	Wait for bus activity

869 ## 7.4   Acknowledge and Error Reporting Mechanism

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

874 The host processor may request a command acknowledge and error information related to any transmission
875 by asserting Bus Turnaround with the transmission. The peripheral shall respond with ACK Trigger
876 Message if there are no errors and with *Acknowledge and Error Report* packet if any errors were detected
877 in previous transmissions. Appropriate flags shall be set to indicate what errors were detected on the
878 preceding transmissions. If the transmission was a Read request, the peripheral shall return READ data
879 without issuing additional ACK Trigger Message or an *Acknowledge and Error Report* packet if no errors
880 were detected. If there was an error in the Read request, the peripheral shall return the appropriate
881 *Acknowledge and Error Report* unless the error was a single-bit correctable error. In that case, the
882 peripheral shall return the requested READ data packet followed by *Acknowledge and Error Report* packet
883 with appropriate error bits set.

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

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

886   ## 8   DSI Protocol

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

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

895   ### 8.1   Multiple Packets per Transmission

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

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

902   There are two modes of data transmission, HS and LP transmission modes, at the PHY layer. Before a HS
903   transmission can be started, the transmitter PHY issues a SoT sequence to the receiver. After that, data or
904   command packets can be transmitted in HS mode. Multiple packets may exist within a single HS
905   transmission and the end of transmission is always signaled at the PHY layer using a dedicated EoT
906   sequence. In order to enhance the overall robustness of the system, DSI defines a dedicated EoT packet
907   (EoTp) at the protocol layer (section 8.8.2) for signaling the end of HS transmission. For backwards
908   compatibility with earlier DSI systems, the capability of generating and interpreting this EoTp can be
909   enabled or disabled. The method of enabling or disabling this capability is out of scope for this document.
910   PHY-based EoT and SoT sequences are defined in *MIPI Alliance Specification for D-PHY*[4].

911   The top diagram in Figure 10 illustrates a case where multiple packets are being sent separately with EoTp
912   support disabled. In HS mode, time gaps between packets shall result in separate HS transmissions for each
913   packet, with a SoT, LPS, and EoT issued by the PHY layer between packets. This constraint does not apply
914   to LP transmissions. The bottom diagram in Figure 10 demonstrates a case where multiple packets are
915   concatenated within a single HS transmission.
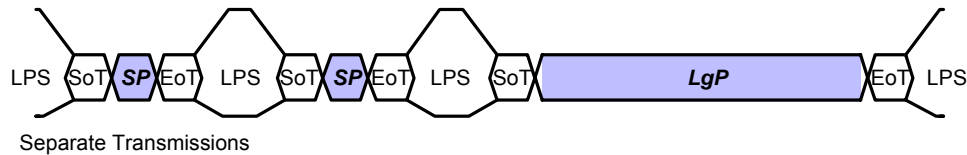
LPS ⟨SoT⟩⟨**SP**⟩⟨EoT⟩ LPS ⟨SoT⟩⟨**SP**⟩⟨EoT⟩ LPS ⟨SoT⟩⟨   **LgP**   ⟩⟨EoT⟩ LPS

Separate Transmissions

KEY:

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

LPS ⟨SoT⟩⟨**SP**⟩⟨**SP**⟩⟨   **LgP**   ⟩⟨EoT⟩ LPS

Single Transmission

916
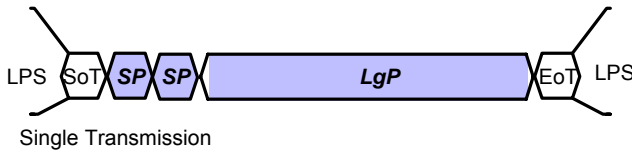
917   **Figure 10 HS Transmission Examples with EoTp disabled**

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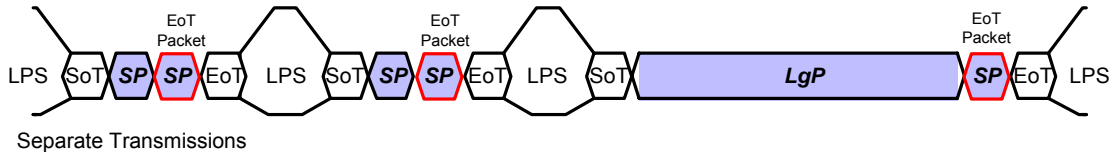
LPS ⟨SoT⟩⟨**SP**⟩⟨**SP**⟩⟨EoT⟩ LPS ⟨SoT⟩⟨**SP**⟩⟨**SP**⟩⟨EoT⟩ LPS ⟨SoT⟩⟨   **LgP**   ⟩⟨**SP**⟩⟨EoT⟩ LPS

Separate Transmissions

KEY:

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

LPS ⟨SoT⟩⟨**SP**⟩⟨**SP**⟩⟨   **LgP**   ⟩⟨**SP**⟩⟨EoT⟩ LPS

Single Transmission

926
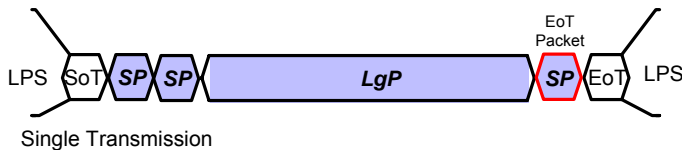
927   **Figure 11 HS Transmission Examples with EoTp enabled**

928   ## 8.2   Packet Composition

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

933   Packet sizes fall into two categories:

934   • **Short packets** are four bytes in length including the ECC. Short packets are used for most
935   Command Mode commands and associated parameters. Other Short packets convey events like H
936   Sync and V Sync edges. Because they are Short packets they can convey accurate timing
937   information to logic at the peripheral.

938   • **Long packets** specify the payload length using a two-byte Word Count field. Payloads may be
939   from 0 to 2^{16} - 1 bytes long. Therefore, a Long packet may be up to 65,541 bytes in length. Long
940   packets permit transmission of large blocks of pixel or other data.

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

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

8.3   Endian Policy

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

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

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

DI	WC (LS Byte)	WC (MS Byte)	ECC	Data	CRC (LS Byte)	CRC (MS Byte)
0x29	0x01	0x00	0x06	0x01	0x0E	0x1E
1 0 0 1 0 1 0 0	1 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 1 1 0 0 0 0 0	1 0 0 0 0 0 0 0	0 1 1 1 0 0 0 0	0 1 1 1 1 0 0 0

←——————Time——————→

956
957
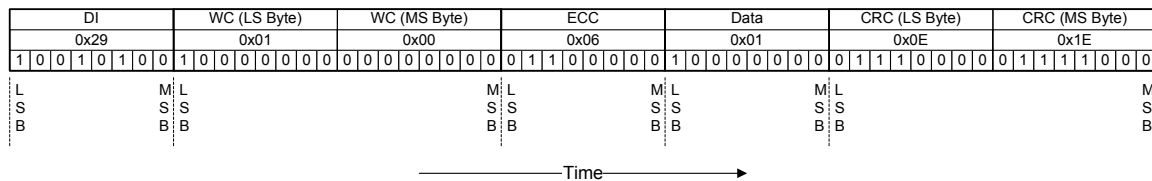
958   **Figure 12 Endian Example (Long Packet)**

8.4   General Packet Structure

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

8.4.1   Long Packet Format

963   Figure 13 shows the structure of the Long packet. A Long packet shall consist of three elements: a 32-bit
964   Packet Header (PH), an application-specific Data Payload with a variable number of bytes, and a 16-bit
965   Packet Footer (PF). The Packet Header is further composed of three elements: an 8-bit Data Identifier, a
966   16-bit Word Count, and 8-bit ECC. The Packet Footer has one element, a 16-bit checksum. Long packets
967   can be from 6 to 65,541 bytes in length.

DATA IDENTIFIER (DI):
Contains Virtual Channel identifier and Data Type information
Data Type denotes the format and content of application-specific payload data

16-bit WORD COUNT (WC):
The Word Count conveys how many words (bytes) are in packet payload
The receiver uses WC to determine the packet end (after Payload + Checksum)

8-bit Error Correction Code (ECC) for the Packet Header:
8-bit ECC for the Packet Header, protects up to 8 bytes in header
Enables one-bit errors in Packet Header to be corrected and two-bit errors to be detected

APPLICATION SPECIFIC PAYLOAD   CHECKSUM (CS)

| LPS | SoT | Data ID | Word Count (WC) | ECC | Data 0 | Data 1 | Data WC-2 | Data WC-1 | 16-bit Checksum | EoT | LPS |

32-bit PACKET HEADER (PH)

16-bit PACKET FOOTER (PF)

PACKET DATA (Payload):
Length = WC * Data Word size (8-bits)
No value restrictions on data words in Payload

968
969

970   **Figure 13 Long Packet Structure**

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

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

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

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

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

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

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

992   ## 8.4.2 Short Packet Format

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

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

DATA IDENTIFIER (DI):
Contains the Virtual Channel Indentifier and the Data Type information
Data Type denotes the format/content of the Application specific payload data
Used by the Application layer

PACKET DATA:
Length is fixed at two bytes
There are no value restrictions on data words

8-bit Error Correction Code (ECC) for the Packet Header:
8-bit ECC for the Packet Header
Allows one-bit errors within the Packet Header to be corrected
and two-bit errors to be detected

| LPS | SoT | Data ID | Data 0 | Data 1 | ECC | EoT | LPS |

PACKET HEADER (PH)

998

999   **Figure 14 Short Packet Structure**

1000   ## 8.5 Common Packet Elements

1001   Long and Short packets have several common elements that are described in this section.

1002   ### 8.5.1 Data Identifier Byte

1003   The first byte of any packet is the DI (Data Identifier) byte. Figure 15 shows the composition of the Data
1004   Identifier (DI) byte.

1005   DI[7:6]: These two bits identify the data as directed to one of four virtual channels.

1006   DI[5:0]: These six bits specify the Data Type.

B7	B6	B5	B4	B3	B2	B1	B0
VC		DT					

Virtual Channel Indentifier (VC)   **Data Type (DT)**

1007
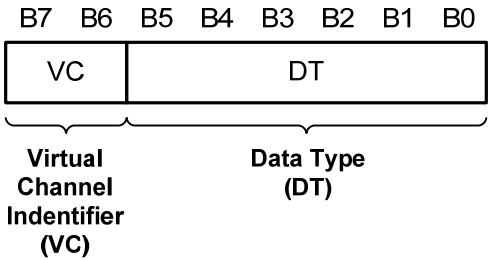
1008   **Figure 15 Data Identifier Byte**

1009   **8.5.1.1   Virtual Channel Identifier – VC field, DI[7:6]**

1010   A processor may service up to four peripherals with tagged commands or blocks of data, using the Virtual
1011   Channel ID field of the header for packets targeted at different peripherals.

1012   The Virtual Channel ID enables one serial stream to service two or more virtual peripherals by
1013   multiplexing packets onto a common transmission channel. Note that packets sent in a single transmission
1014   each have their own Virtual Channel assignment and can be directed to different peripherals. Although the
1015   DSI protocol permits communication with multiple peripherals, this specification only addresses the
1016   connection of a host processor to a single peripheral. Implementation details for connection to more than
1017   one physical peripheral are beyond the scope of this document.

1018   **8.5.1.2   Data Type Field DT[5:0]**

1019   The Data Type field specifies if the packet is a Long or Short packet type and the packet format. The Data
1020   Type field, along with the Word Count field for Long packets, informs the receiver of how many bytes to
1021   expect in the remainder of the packet. This is necessary because there are no special packet start / end sync
1022   codes to indicate the beginning and end of a packet. This permits packets to convey arbitrary data, but it
1023   also requires the packet header to explicitly specify the size of the packet.

1024   When the receiving logic has counted down to the end of a packet, it shall assume the next data is either the
1025   header of a new packet or the EoT (End of Transmission) sequence.

1026   **8.5.2   Error Correction Code**

1027   The Error Correction Code allows single-bit errors to be corrected and 2-bit errors to be detected in the
1028   Packet Header. The host processor shall always calculate and transmit an ECC byte. Peripherals shall
1029   support ECC in both forward- and reverse-direction communications. See section 9 for more information
1030   on coding and decoding the ECC and section 8.9.2 for ECC and Checksum requirements.

1031   **8.6   Interleaved Data Streams**

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

1032
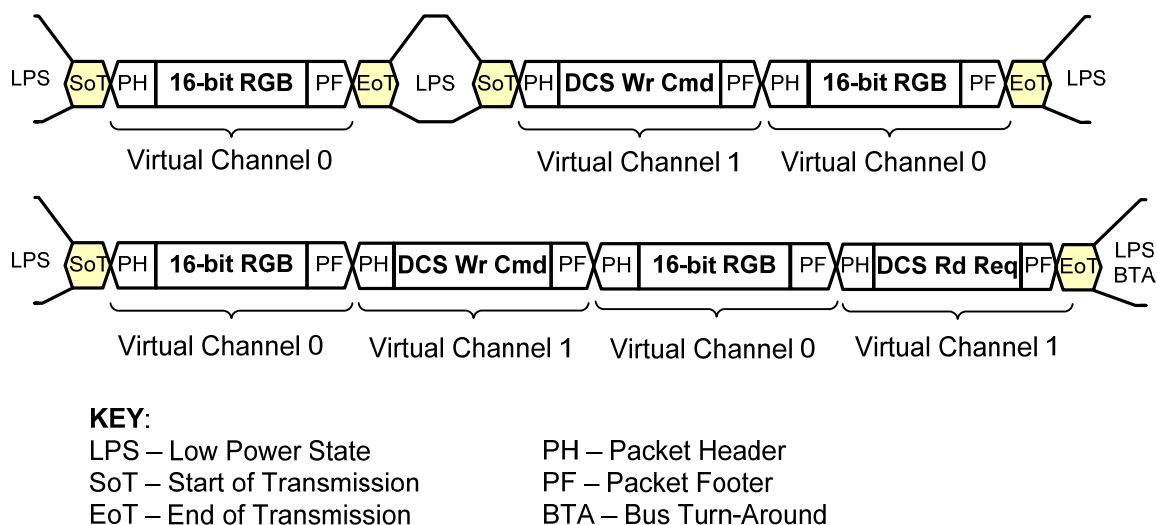
1033   **Figure 16 Interleaved Data Stream Example with EoTp disabled**

1034   One application for multiple channels is a high-resolution display using two or more separate driver ICs on
1035   a single display module. Each driver IC addresses only a portion of the columns on the display device.

1036   Each driver IC captures and displays only the packet contents targeted for that driver and ignores the other
1037   packets. See Figure 17.
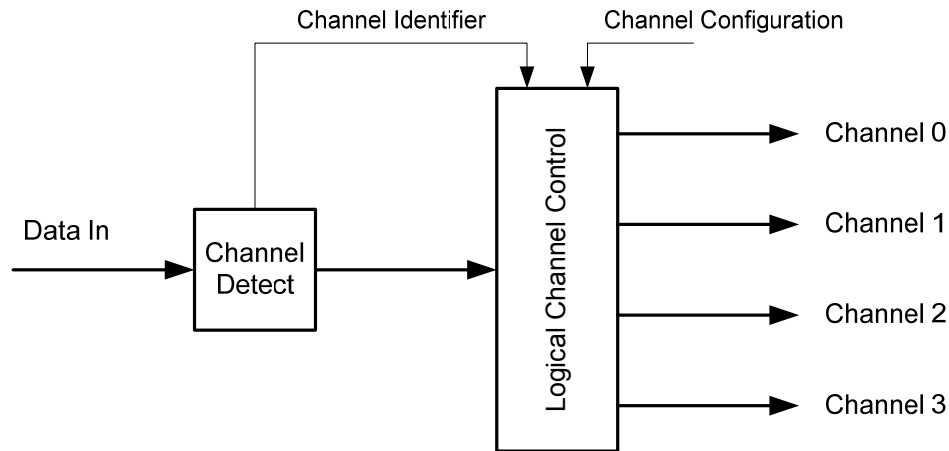
1038
1039   **Figure 17 Logical Channel Block Diagram (Receiver Case)**

1040   **8.6.1   Interleaved Data Streams and Bi-directionality**

1041   When multiple peripherals have bidirectional capability there shall be a clear and unambiguous means for
1042   returning READ data, events and status back to the host processor from the intended peripheral. The
1043   combination of BTA and the Virtual Channel ID ensures no confusion over which peripheral is expected to
1044   respond to any request from the peripheral. Returning packets shall be tagged with the ID of the peripheral
1045   that sent the packet.

1046   A consequence of bidirectionality is any transmission from the host processor shall contain no more than
1047   one packet requiring a peripheral response. This applies regardless of the number of peripherals that may be
1048   connected via the Link to the host processor.

1049   **8.7   Processor to Peripheral Direction (Processor-Sourced) Packet Data Types**

1050   The set of transaction types sent from the host processor to a peripheral, such as a display module, are
1051   shown in Table 16.

1052   **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
08h	00 1000	End of Transmission packet (EoTp)	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
32h	11 0010	Turn On Peripheral Command	Short

Data Type, hex	Data Type, binary	Description	Packet Size
03h	00 0011	Generic Short WRITE, no parameters	Short
13h	01 0011	Generic Short WRITE, 1 parameter	Short
23h	10 0011	Generic Short WRITE, 2 parameters	Short
04h	00 0100	Generic READ, no parameters	Short
14h	01 0100	Generic READ, 1 parameter	Short
24h	10 0100	Generic READ, 2 parameters	Short
05h	00 0101	DCS Short WRITE, no parameters	Short
15h	01 0101	DCS Short WRITE, 1 parameter	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 Long Write	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	

1053 **8.8   Processor-to-Peripheral Transactions – Detailed Format Description**

1054 **8.8.1   Sync Event (H Start, H End, V Start, V End), Data Type = xx 0001 (x1h)**

1055 Sync Events are Short packets and, therefore, can time-accurately represent events like the start and end of
1056 sync pulses. As "start" and "end" are separate and distinct events, the length of sync pulses, as well as
1057 position relative to active pixel data, e.g. front and back porch display timing, may be accurately conveyed
1058 to the peripheral. The Sync Events are defined as follows:

1059   • Data Type = 00 0001 (01h)   V Sync Start

1060   • Data Type = 01 0001 (11h)   V Sync End

1061   • Data Type = 10 0001 (21h)   H Sync Start

1062   • Data Type = 11 0001 (31h)   H Sync End

1063 In order to represent timing information as accurately as possible a V Sync Start event represents the start
1064 of the VSA and also implies an H Sync Start event for the first line of the VSA. Similarly, a V Sync End
1065 event implies an H Sync Start event for the last line of the VSA.

1066   Sync events should occur in pairs, Sync Start and Sync End, if accurate pulse-length information needs to
1067   be conveyed. Alternatively, if only a single point (event) in time is required, a single sync event (normally,
1068   Sync Start) may be transmitted to the peripheral. Sync events may be concatenated with blanking packets to
1069   convey inter-line timing accurately and avoid the overhead of switching between LPS and HS for every
1070   event. Note there is a power penalty for keeping the data line in HS mode, however.

1071   Display modules that do not need traditional sync/blanking/pixel timing should transmit pixel data in a
1072   high-speed burst then put the bus in Low Power Mode, for reduced power consumption. The recommended
1073   burst size is a scan line of pixels, which may be temporarily stored in a line buffer on the display module.

8.8.2   EoTp, Data Type = 00 1000 (08h)

1075   This short packet is used for indicating the end of a HS transmission to the data link layer. As a result,
1076   detection of the end of HS transmission may be decoupled from physical layer characteristics. *MIPI*
1077   *Alliance Specification for D-PHY*[4] defines an EoT sequence composed of a series of all 1's or 0's
1078   depending on the last bit of the last packet within a HS transmission. Due to potential errors, the EoT
1079   sequence could be interpreted incorrectly as valid data types. Although EoT errors are not expected to
1080   happen frequently, the addition of this packet will enhance overall system reliability.

1081   Devices compliant to earlier revisions of the DSI specification do not support EoTp generation or detection.
1082   A Host or peripheral device compliant to this revision of DSI specification shall incorporate capability of
1083   supporting EoTp. The device shall also provide an implementation-specific means for enabling and
1084   disabling this capability to ensure interoperability with earlier DSI devices that do not support the EoTp.

1085   The main objective of the EoTp is to enhance overall robustness of the system during HS transmission
1086   mode. Therefore, DSI transmitters should not generate an EoTp when transmitting in LP mode. The Data
1087   Link layer of DSI receivers shall detect and interpret arriving EoTps regardless of transmission mode (HS
1088   or LP modes) in order to decouple itself from the physical layer. Table 17 describes how DSI mandates
1089   EoTp support for different transmission and reception modes.

1090   **Table 17 EoT Support for Host and Peripheral**

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

1091   Unlike other DSI packets, an EoTp has a fixed format as follows:

1092   • Data Type = DI [5:0] = 0b001000

1093   • Virtual Channel = DI [7:6] = 0b00

1094   • Payload Data [15:0] = 0x0F0F

1095   • ECC [7:0] = 0x01

1096   The virtual channel identifier associated with an EoTp is fixed to 0, regardless of the number of different
1097   virtual channels present within the same transmission. For multi-Lane systems, the EoTp bytes are
1098   distributed across multiple Lanes.

1099 ### 8.8.3   Color Mode Off Command, Data Type = 00 0010 (02h)

1100 *Color Mode Off* is a Short packet command that returns a Video Mode display module from low-color
1101 mode to normal display operation.

1102 ### 8.8.4   Color Mode On Command, Data Type = 01 0010 (12h)

1103 *Color Mode On* is a Short packet command that switches a Video Mode display module to a low-color
1104 mode for power saving.

1105 ### 8.8.5   Shutdown Peripheral Command, Data Type = 10 0010 (22h)

1106 *Shutdown Peripheral* command is a Short packet command that turns off the display in a Video Mode
1107 display module for power saving. Note the interface shall remain powered in order to receive the turn-on,
1108 or wake-up, command.

1109 ### 8.8.6   Turn On Peripheral Command, Data Type = 11 0010 (32h)

1110 *Turn On Peripheral* command is Short packet command that turns on the display in a Video Mode display
1111 module for normal display operation.

1112 ### 8.8.7   Generic Short WRITE Packet with 0, 1, or 2 parameters, Data Types = 00
1113 ### 0011 (03h), 01 0011 (13h), 10 0011 (23h), Respectively

1114 *Generic Short WRITE* command is a Short packet type for sending generic data to the peripheral. The
1115 format and interpretation of the contents of this packet are outside the scope of this document. It is the
1116 responsibility of the system designer to ensure that both the host processor and peripheral agree on the
1117 format and interpretation of such data.

1118 The complete packet shall be four bytes in length including an ECC byte. The two Data Type MSBs, bits
1119 [5:4], indicate the number of valid parameters (0, 1, or 2). For single-byte parameters, the parameter shall
1120 be sent in the first data byte following the DI byte and the second data byte shall be set to 00h.

1121 ### 8.8.8   Generic READ Request with 0, 1, or 2 Parameters, Data Types = 00 0100
1122 ### (04h), 01 0100 (14h), 10 0100(24h), Respectively

1123 *Generic READ* request is a Short packet requesting data from the peripheral. The format and interpretation
1124 of the parameters of this packet, and of returned data, are outside the scope of this document. It is the
1125 responsibility of the system designer to ensure that both the host processor and peripheral agree on the
1126 format and interpretation of such data.

1127 Returned data may be of Short or Long packet format. Note the *Set Max Return Packet Size* command
1128 limits the size of returning packets so that the host processor can prevent buffer overflow conditions when
1129 receiving data from the peripheral. If the returning block of data is larger than the maximum return packet
1130 size specified, the read response will require more than one transmission. The host processor shall send
1131 multiple Generic READ requests in separate transmissions if the requested data block is larger than the
1132 maximum packet size.

1133 The complete packet shall be four bytes in length including an ECC byte. The two Data Type MSBs, bits
1134 [5:4], indicate the number of valid parameters (0, 1, or 2). For single byte parameters, the parameter shall
1135 be sent in the first data byte following the DI byte and the second data byte shall be set to 00h.

1136 Since this is a read command, BTA shall be asserted by the host processor following this request.

1137   The peripheral shall respond to Generic READ Request in one of the following ways:

1138   • If an error was detected by the peripheral, it shall send *Acknowledge and Error Report*. If an ECC
1139   error in the request was detected and corrected, the peripheral shall transmit the requested READ
1140   data packet with the *Acknowledge and Error Report* packet appended, in the same transmission.

1141   • If no error was detected by the peripheral, it shall send the requested READ packet (Short or
1142   Long) with appropriate ECC and Checksum, if Checksum is enabled.

1143   A Generic READ request shall be the only, or last, packet of a transmission. Following the transmission the
1144   host processor sends BTA. Having given control of the bus to the peripheral, the host processor will expect
1145   the peripheral to transmit the appropriate response packet and then return bus possession to the host
1146   processor.

1147 8.8.9   DCS Commands

1148   DCS is a standardized command set intended for Command Mode display modules. The interpretation of
1149   DCS commands is supplied in *MIPI Alliance Standard for Display Command Set (DCS)[1]*.

1150   For DCS short commands, the first byte following the Data Identifier Byte is the *DCS Command Byte*. If
1151   the DCS command does not require parameters, the second payload byte shall be 00h.

1152   If a DCS Command requires more than one parameter, the command shall be sent as a Long Packet type.

1153 8.8.9.1   DCS Short Write Command, 0 or 1 parameter, Data Types = 00 0101 (05h), 01 0101
1154   (15h), Respectively

1155   *DCS Short Write* command is used to write a single data byte to a peripheral such as a display module. The
1156   packet is a Short packet composed of a Data ID byte, a DCS Write command, an optional parameter byte
1157   and an ECC byte. Data Type bit 4 shall be set to 1 if there is a valid parameter byte, and shall be set to 0 if
1158   there is no valid parameter byte. If a parameter is not required, the parameter byte shall be 00h. If *DCS
1159   Short Write* command, followed by BTA, is sent to a bidirectional peripheral, the peripheral shall respond
1160   with ACK Trigger Message unless an error was detected in the host-to-peripheral transmission. If the
1161   peripheral detects an error in the transmission, the peripheral shall respond with *Acknowledge and Error
1162   Report*. If the peripheral is a Video Mode display on a unidirectional DSI, it shall ignore BTA. See Table
1163   19.

1164 8.8.9.2   DCS Read Request, No Parameters, Data Type = 00 0110 (06h)

1165   DCS READ commands are used to request data from a display module. This packet is a Short packet
1166   composed of a Data ID byte, a DCS Read command, a byte set to 00h and an ECC byte. Since this is a read
1167   command, BTA shall be asserted by the host processor following completion of the transmission.
1168   Depending on the type of READ requested in the DCS Command Byte, the peripheral may respond with a
1169   DCS Short Read Response or DCS Long Read Response.

1170   The read response may be more than one packet in the case of DCS Long Read Response, if the returning
1171   block of data is larger than the maximum return packet size specified. In that case, the host processor shall
1172   send multiple DCS Read Request commands to transfer the complete data block. See section 8.8.10 for
1173   details on setting the read packet size.

1174   The peripheral shall respond to DCS READ Request in one of the following ways:

1175   • If an error was detected by the peripheral, it shall send *Acknowledge and Error Report*. If an ECC
1176   error in the request was detected and corrected, the peripheral shall send the requested READ data
1177   packet followed by the *Acknowledge and Error Report* packet in the same transmission.

1178   • If no error was detected by the peripheral, it shall send the requested READ packet (Short or
1179   Long) with appropriate ECC and Checksum, if either or both features are enabled.

1180 A DCS Read Request packet shall be the only, or last, packet of a transmission. Following the transmission,
1181 the host processor sends BTA. Having given control of the bus to the peripheral, the host processor will
1182 expect the peripheral to transmit the appropriate response packet and then return bus possession to the host
1183 processor.

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

1184

1185 *DCS Long Write/write_LUT Command* is used to send larger blocks of data to a display module that
1186 implements the Display Command Set.

1187 The packet consists of the DI byte, a two-byte WC, an ECC byte, followed by the *DCS Command Byte*, a
1188 payload of length WC minus one bytes, and a two-byte checksum.

8.8.10 Set Maximum Return Packet Size, Data Type = 11 0111 (37h)

1189

1190 *Set Maximum Return Packet Size* is a four-byte command packet (including ECC) that specifies the
1191 maximum size of the payload in a Long packet transmitted from peripheral back to the host processor. The
1192 order of bytes in *Set Maximum Return Packet Size* is: Data ID, two-byte value for maximum return packet
1193 size, followed by the ECC byte. Note that the two-byte value is transmitted with LS byte first. This
1194 command shall be ignored by peripherals with unidirectional DSI interfaces.

1195 During a power-on or Reset sequence, the Maximum Return Packet Size shall be set by the peripheral to a
1196 default value of one. This parameter should be set by the host processor to the desired value in the
1197 initialization routine before commencing normal operation.

8.8.11 Null Packet (Long), Data Type = 00 1001 (09h)

1198

1199 *Null Packet* is a mechanism for keeping the serial Data Lane(s) in High-Speed mode while sending dummy
1200 data. This is a Long packet. Like all packets, its content shall be an integer number of bytes.

1201 The Null Packet consists of the DI byte, a two-byte WC, ECC byte, and "null" payload of WC bytes,
1202 ending with a two-byte Checksum. Actual data values sent are irrelevant because the peripheral does not
1203 capture or store the data. However, ECC and Checksum shall be generated and transmitted to the
1204 peripheral.

8.8.12 Blanking Packet (Long), Data Type = 01 1001 (19h)

1205

1206 A Blanking packet is used to convey blanking timing information in a Long packet. Normally, the packet
1207 represents a period between active scan lines of a Video Mode display, where traditional display timing is
1208 provided from the host processor to the display module. The blanking period may have *Sync Event* packets
1209 interspersed between blanking segments. Like all packets, the Blanking packet contents shall be an integer
1210 number of bytes. Blanking packets may contain arbitrary data as payload.

1211 The Blanking packet consists of the DI byte, a two-byte WC, an ECC byte, a payload of length WC bytes,
1212 and a two-byte checksum.

8.8.13 Generic Long Write, Data Type = 10 1001 (29h)

1213

1214 *Generic Long Write Packet* is used to transmit arbitrary blocks of data from a host processor to a peripheral
1215 in a Long packet. The packet consists of the DI byte, a two-byte WC, an ECC byte, a payload of length WC
1216 bytes and a two-byte checksum.

1217 **8.8.14 Packed Pixel Stream, 16-bit Format, Long packet, Data Type 00 1110 (0Eh)**

1218
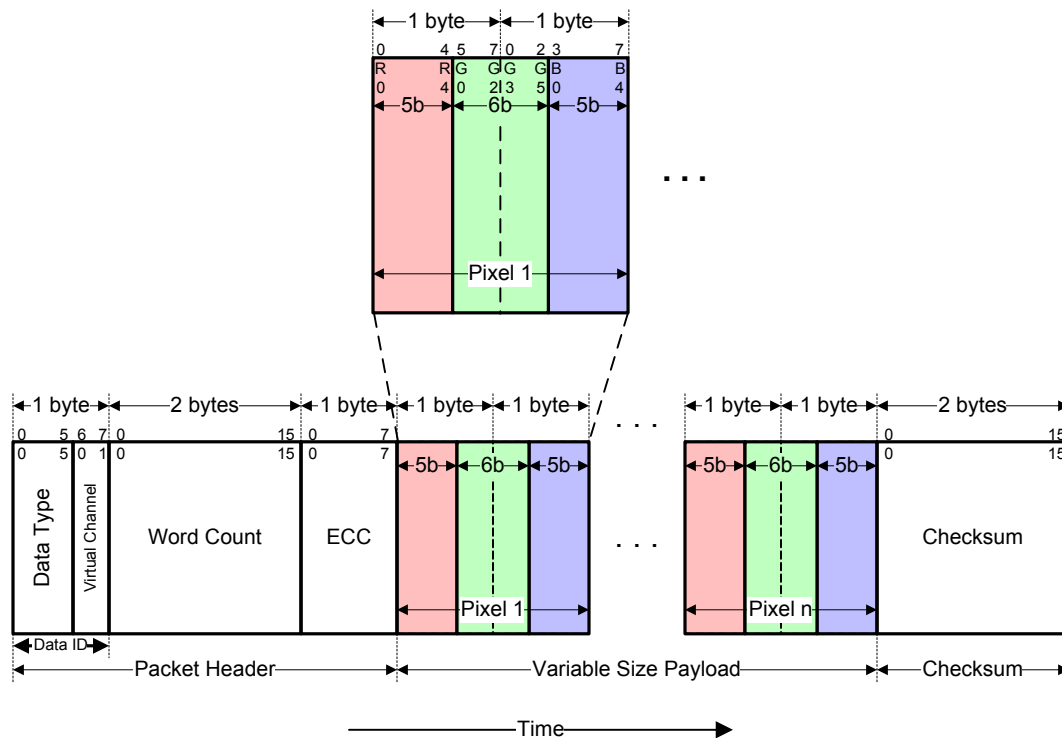
1219 **Figure 18 16-bit per Pixel – RGB Color Format, Long packet**

1220 *Packed Pixel Stream 16-Bit Format* is a Long packet used to transmit image data formatted as 16-bit pixels
1221 to a Video Mode display module. The packet consists of the DI byte, a two-byte WC, an ECC byte, a
1222 payload of length WC bytes and a two-byte checksum. Pixel format is five bits red, six bits green, five bits
1223 blue, in that order. Note that the "Green" component is split across two bytes. Within a color component,
1224 the LSB is sent first, the MSB last.

1225 With this format, pixel boundaries align with byte boundaries every two bytes. The total line width
1226 (displayed plus non-displayed pixels) should be a multiple of two bytes.

1227 Normally, the display module has no frame buffer of its own, so all image data shall be supplied by the host
1228 processor at a sufficiently high rate to avoid flicker or other visible artifacts.

1229   **8.8.15 Packed Pixel Stream, 18-bit Format, Long packet, Data type = 01 1110 (1Eh)**

1230
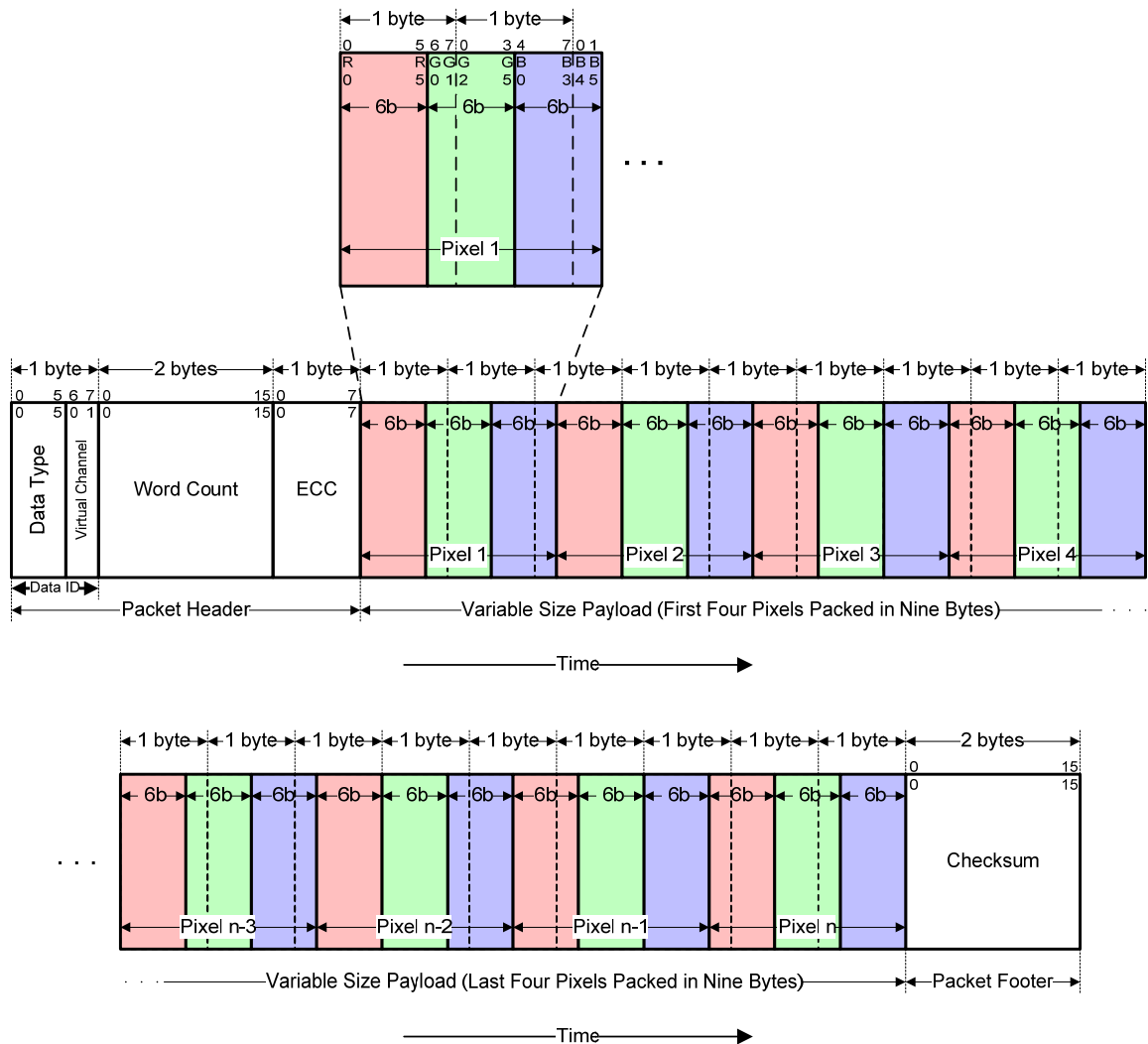
1231   **Figure 19 18-bit per Pixel (Packed) – RGB Color Format, Long packet**

1232   *Packed Pixel Stream 18-Bit Format (Packed)* is a Long packet. It is used to transmit RGB image data
1233   formatted as pixels to a Video Mode display module that displays 18-bit pixels The packet consists of the
1234   DI byte, a two-byte WC, an ECC byte, a payload of length WC bytes and a two-byte Checksum. Pixel
1235   format is red (6 bits), green (6 bits) and blue (6 bits), in that order. Within a color component, the LSB is
1236   sent first, the MSB last.

1237   Note that pixel boundaries only align with byte boundaries every four pixels (nine bytes). Preferably,
1238   display modules employing this format have a horizontal extent (width in pixels) evenly divisible by four,
1239   so no partial bytes remain at the end of the display line data. If the active (displayed) horizontal width is not
1240   a multiple of four pixels, the transmitter shall send additional fill pixels at the end of the display line to
1241   make the transmitted width a multiple of four pixels. The receiving peripheral shall not display the fill
1242   pixels when refreshing the display device. For example, if a display device has an active display width of
1243   399 pixels, the transmitter should send 400 pixels in one or more packets. The receiver should display the
1244   first 399 pixels and discard the last pixel of the transmission.

1245   With this format, the total line width (displayed plus non-displayed pixels) should be a multiple of four
1246   pixels (nine bytes).

1247 **8.8.16 Pixel Stream, 18-bit Format in Three Bytes, Long packet, Data Type = 10**
1248 **1110 (2Eh)**

1249

1250 **Figure 20 18-bit per Pixel (Loosely Packed) – RGB Color Format, Long packet**

1251 In the *18-bit Pixel Loosely Packed* format, each R, G, or B color component is six bits but is shifted to the
1252 upper bits of the byte, such that the valid pixel bits occupy bits [7:2] of each byte. Bits [1:0] of each
1253 payload byte representing active pixels are ignored. As a result, each pixel requires three bytes as it is
1254 transmitted across the Link. This requires more bandwidth than the "packed" format, but requires less
1255 shifting and multiplexing logic in the packing and unpacking functions on each end of the Link.

1256 This format is used to transmit RGB image data formatted as pixels to a Video Mode display module that
1257 displays 18-bit pixels. The packet consists of the DI byte, a two-byte WC, an ECC byte, a payload of length
1258 WC bytes and a two-byte Checksum. The pixel format is red (6 bits), green (6 bits) and blue (6 bits) in that
1259 order. Within a color component, the LSB is sent first, the MSB last.

1260 With this format, pixel boundaries align with byte boundaries every three bytes. The total line width
1261 (displayed plus non-displayed pixels) should be a multiple of three bytes.

1262   **8.8.17 Packed Pixel Stream, 24-bit Format, Long packet, Data Type = 11 1110 (3Eh)**
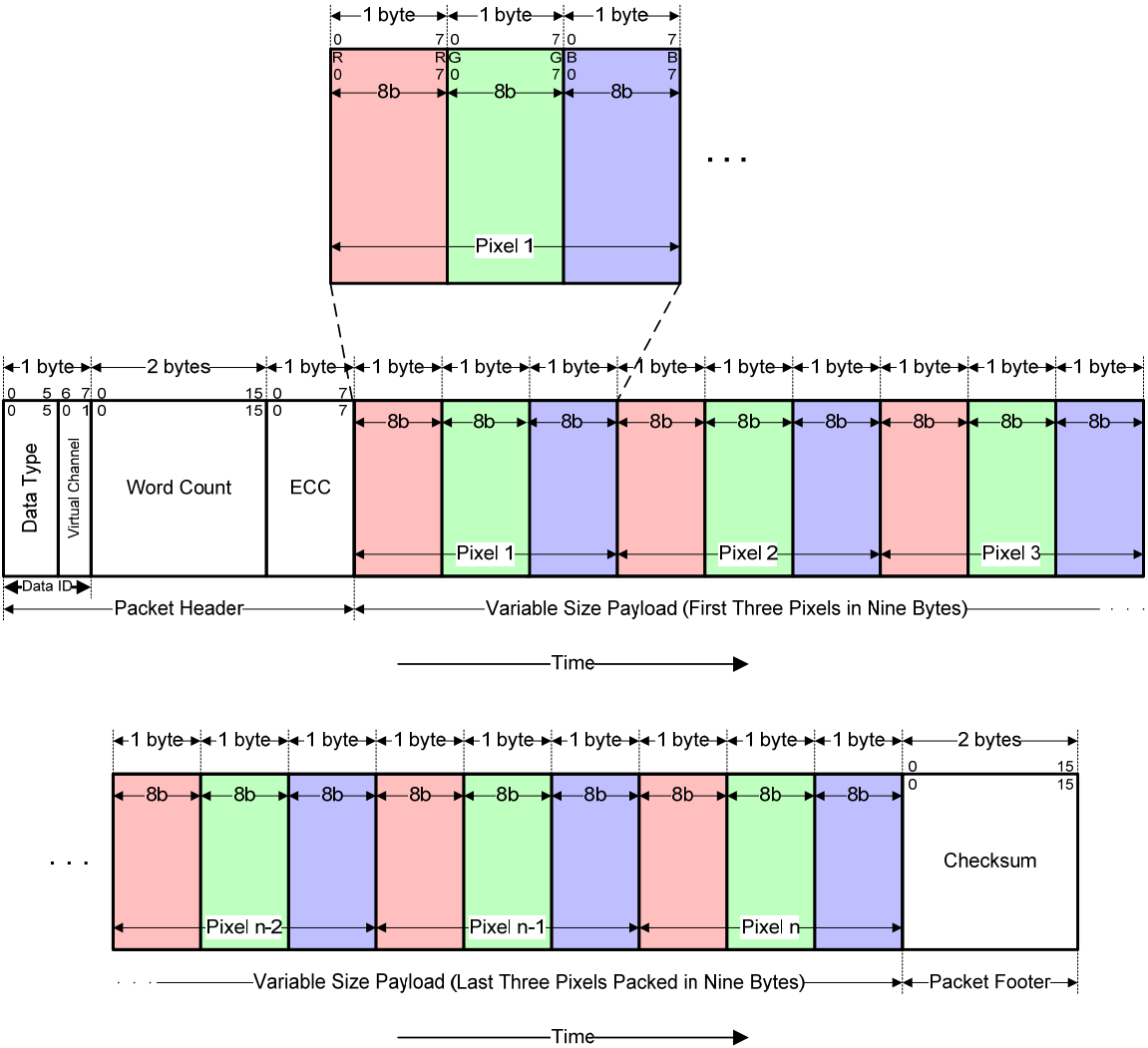
1263

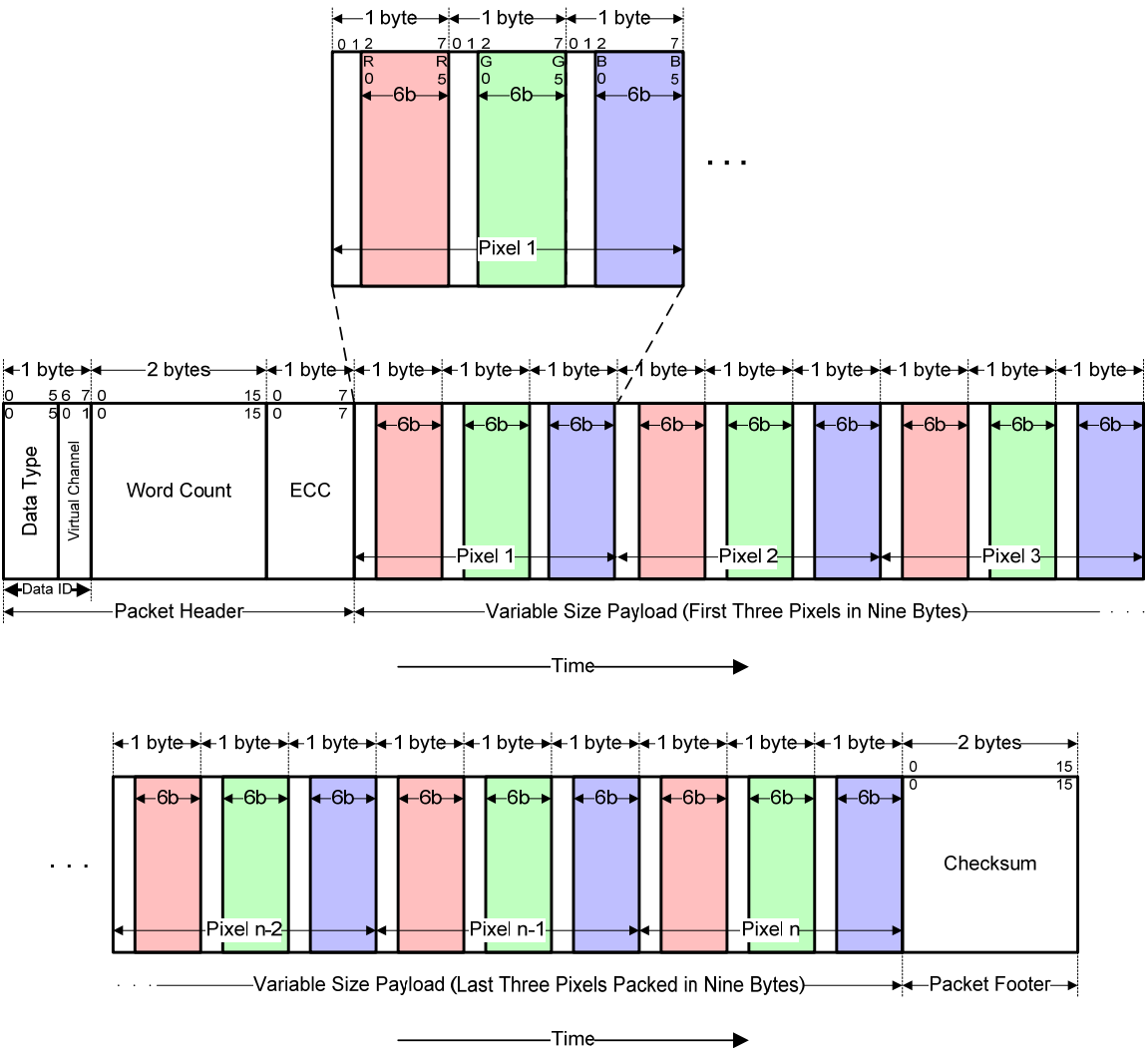1264   **Figure 21 24-bit per Pixel – RGB Color Format, Long packet**

1265   *Packed Pixel Stream 24-Bit Format* is a Long packet. It is used to transmit image data formatted as 24-bit
1266   pixels to a Video Mode display module. The packet consists of the DI byte, a two-byte WC, an ECC byte, a
1267   payload of length WC bytes and a two-byte Checksum. The pixel format is red (8 bits), green (8 bits) and
1268   blue (8 bits), in that order. Each color component occupies one byte in the pixel stream; no components are
1269   split across byte boundaries. Within a color component, the LSB is sent first, the MSB last.

1270   With this format, pixel boundaries align with byte boundaries every three bytes. The total line width
1271   (displayed plus non-displayed pixels) should be a multiple of three bytes.

1272   **8.8.18 DO NOT USE and Reserved Data Types**

1273   Data Type codes with four LSBs = 0000 or 1111 shall not be used. All other non-specified Data Type
1274   codes are reserved.

1275   Note that DT encoding is specified so that all data types have at least one 0-1 or 1-0 transition in the four
1276   bits DT bits [3:0]. This ensures a transition within the first four bits of the serial data stream of every

1277   packet. DSI protocol or the PHY can use this information to determine quickly, following the end of each
1278   packet, if the next bits represent the start of a new packet (transition within four bits) or an EoT sequence
1279   (no transition for at least four bits).

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

1281   All Command Mode systems require bidirectional capability for returning READ data, acknowledge, or
1282   error information to the host processor. Multi-Lane systems shall use Lane 0 for all peripheral-to-processor
1283   transmissions; other Lanes shall be unidirectional.

1284   Reverse-direction signaling shall only use LP (Low Power) mode of transmission.

1285   Simple, low-cost systems using display modules which work exclusively in Video Mode may be
1286   configured with unidirectional DSI for all Lanes. In such systems, no acknowledge or error reporting is
1287   possible using DSI, and no requirements specified in this section apply to such systems. However, these
1288   systems shall have ECC checking and correction capability, which enables them to correct single-bit errors
1289   in headers and Short packets, even if they cannot report the error.

1290   Command Mode systems that use DCS shall have a bidirectional data path. Short packets and the header of
1291   Long packets shall use ECC and may use Checksum to provide a higher level of data integrity. The
1292   Checksum feature enables detection of errors in the payload of Long packets.

8.9.1   Packet Structure for Peripheral-to-Processor LP Transmissions

1294   Packet structure for peripheral-to-processor transactions is the same as for the processor-to-peripheral
1295   direction.

1296   As in the processor-to-peripheral direction, two basic packet formats are specified: Short and Long. For
1297   both types, an ECC byte shall be calculated to cover the Packet Header data. ECC calculation is the same in
1298   the peripheral as in the host processor. For Long packets, error checking on the Data Payload, i.e. all bytes
1299   after the Packet Header, is optional. If the Checksum is not calculated by the peripheral the Packet Footer
1300   shall be 0000h.

1301   BTA shall take place after every peripheral-to-processor transaction. This returns bus control to the host
1302   processor following the completion of the LP transmission from the peripheral.

1303   Peripheral-to-processor transactions are of four basic types:

1304   • *Tearing Effect (TE)* is a Trigger message sent to convey display timing information to the host
1305   processor. *Trigger* messages are single byte packets sent by a peripheral's PHY layer in response
1306   to a signal from the DSI protocol layer. See *MIPI Alliance Specification for D-PHY*[4] for a
1307   description of Trigger messages.

1308   • *Acknowledge* is a Trigger Message sent when the current transmission, as well as all preceding
1309   transmissions since the last peripheral to host communication, i.e. either triggers or packets, is
1310   received by the peripheral with no errors.

1311   • *Acknowledge and Error Report* is a Short packet sent if any errors were detected in preceding
1312   transmissions from the host processor. Once reported, accumulated errors in the error register are
1313   cleared.

1314   • *Response to Read Request* may be a Short or Long packet that returns data requested by the
1315   preceding READ command from the processor.

1316   **8.9.2   System Requirements for ECC and Checksum and Packet Format**

1317   A peripheral shall implement ECC, and may optionally implement checksum.

1318   ECC support is the capability of generating ECC bytes locally from incoming packet headers and
1319   comparing the results to the ECC fields of incoming packet headers in order to determine if an error has
1320   occurred. DSI ECC provides detection and correction of single-bit errors and detection of multiple-bit
1321   errors. See sections 9.4 and 9.5 for information on generating and applying ECC, respectively.

1322   For Command Mode peripherals, if a single-bit error has occurred the peripheral shall correct the error, set
1323   the appropriate error bit (section 8.9.5) and report the error to the Host at the next available opportunity.
1324   The packet can be used as if no error occurred. If a multiple-bit error is detected, the receiver shall drop the
1325   packet and the rest of the transmission, set the relevant error bit and report the error back to the Host at the
1326   next available opportunity. When the peripheral is reporting to the Host, it shall compute and send the
1327   correct ECC based on the content of the header being transmitted.

1328   For Video Mode peripherals, if a single-bit error has occurred the peripheral shall correct the error and use
1329   the packet as if no error occurred. If a multiple-bit error is detected, the receiver shall drop the packet and
1330   the rest of the transmission. Since DSI Links may be unidirectional in Video Mode, error reporting
1331   capabilities in these cases are application specific and out of scope of this document.

1332   Host processors shall implement both ECC and checksum capabilities. ECC and Checksum capabilities
1333   shall be separately enabled or disabled so that a host processor can match a peripheral's capability when
1334   checking return data from the peripheral. Note, in previous revisions of DSI peripheral support for ECC
1335   was optional. See section 10.6. The mechanism for enabling and disabling Checksum capability is out of
1336   scope for this document.

1337   An ECC byte can be applied to both Short and Long packets. Checksum bytes shall only be applied to
1338   Long packets.

1339   Host processors and peripherals shall provide ECC support in both the Forward and Reverse
1340   communication directions.

1341   Host processors, and peripherals that implement Checksum, shall provide Checksum capabilities in both
1342   the Forward and Reverse communication directions.

1343   See section 8.4 for a description of the ECC and Checksum bytes.

1344   **8.9.3   Appropriate Responses to Commands and ACK Requests**

1345   In general, if the host processor completes a transmission to the peripheral with BTA asserted, the
1346   peripheral shall respond with one or more appropriate packet(s), and then return bus ownership to the host
1347   processor. If BTA is not asserted following a transmission from the host processor, the peripheral shall not
1348   communicate an *Acknowledge* or error information back to the host processor.

1349   Interpretation of processor-to-peripheral transactions with BTA asserted, and the expected responses, are as
1350   follows:

1351   •   Following a non-Read command, the peripheral shall respond with *Acknowledge* if no errors were
1352   detected and stored since the last peripheral to host communication, i.e. either triggers or packets.

1353   •   Following a Read request, the peripheral shall send the requested READ data if no errors were
1354   detected and stored since the last peripheral to host communication, i.e. either triggers or packets.

1355   •   Following a Read request if only a single-bit ECC error was detected and corrected, the peripheral
1356   shall send the requested READ data in a Long or Short packet, followed by a 4-byte *Acknowledge*

1357   *and Error Report* packet in the same LP transmission. The Error Report shall have the *ECC Error*
1358   *– Single Bit* flag set, as well as any error bits from any preceding transmissions stored since the
1359   last peripheral to host communication.

1360   •   Following a non-Read command if only a single-bit ECC error was detected and corrected, the
1361   peripheral shall proceed to execute the command, and shall respond to BTA by sending a 4-byte
1362   *Acknowledge and Error Report* packet. The Error Report shall have the *ECC Error – Single Bit*
1363   flag set, as well as any error bits from any preceding transmissions stored since the last peripheral
1364   to host communication.

1365   •   Following a Read request, if multi-bit ECC errors were detected and not corrected, the peripheral
1366   shall send a 4-byte *Acknowledge and Error Report* packet without sending Read data. The Error
1367   Report shall have the *ECC Error – Multi-Bit* flag set, as well as any error bits from any preceding
1368   transmissions stored since the last peripheral to host communication.

1369   •   Following a non-Read command, if multi-bit ECC errors were detected and not corrected, the
1370   peripheral shall not execute the command, and shall send a 4-byte *Acknowledge and Error Report*
1371   packet. The Error Report shall have the *ECC Error – Multi-Bit* flag set, as well as any error bits
1372   from any preceding transmissions stored since the last peripheral to host communication.

1373   •   Following any command, if *SoT Error*, *SoT Sync Error* or *DSI VC ID Invalid* or DSI protocol
1374   violation was detected, or the DSI command was not recognized, the peripheral shall send a 4-byte
1375   *Acknowledge and Error Report* response, with the appropriate error flags set, as well as any error
1376   bits from any preceding transmissions stored since the last peripheral to host communication, in
1377   the two-byte error field. Only the *Acknowledge and Error Report* packet shall be transmitted; no
1378   read or write accesses shall take place on the peripheral in response.

1379   •   Following any command, if *EoT Sync Error* or *LP Transmit Sync Error* is detected, or a checksum
1380   error is detected in the payload, the peripheral shall send a 4-byte *Acknowledge and Error Report*
1381   packet with the appropriate error flags set, as well as any error bits from any preceding
1382   transmissions stored since the last peripheral to host communication. For a read command, only
1383   the *Acknowledge and Error Report* packet shall be transmitted; no read data shall be sent by the
1384   peripheral in response.

1385   Refer to section 7 for how the peripheral acts when encountering Escape Mode Entry Command Error, Low
1386   Level Transmit Sync Error and False Control Error. Section 7.2.2.2 elaborates on HS Receive Timeout
1387   Error.

1388   Once reported to the host processor, all errors documented in this section are cleared from the Error
1389   Register. Other error types may be detected, stored, and reported by a peripheral, but the mechanisms for
1390   flagging, reporting, and clearing such errors are outside the scope of this document.

8.9.4   Format of Acknowledge and Error Report and Read Response Data Types

1392   *Acknowledge and Error Report* confirms that the preceding command or data sent from the host processor
1393   to a peripheral was received, and indicates what types of error were detected on the transmission and any
1394   preceding transmissions. Note that if errors accumulate from multiple preceding transmissions, it may be
1395   difficult or impossible to identify which transmission contained the error. This message is a Short packet of
1396   four bytes, taking the form:

1397   •   Byte 0: Data Identifier (Virtual Channel ID + Acknowledge Data Type)

1398   •   Byte 1: Error Report bits 0-7

1399   •   Byte 2: Error Report bits 8-15

1400   •   ECC byte covering the header

1401 *Acknowledge* is sent using a Trigger message. See *MIPI Alliance Document for D-PHY*[4] for a description
1402 of Trigger messages:

1403 • Byte 0: 00100001 (shown here in first bit [left] to last bit [right] sequence)

1404 *Response to Read Request* returns data requested by the preceding READ command from the processor.
1405 These may be short or Long packets. The format for short READ packet responses is:

1406 • Byte 0: Data Identifier (Virtual Channel ID + Data Type)

1407 • Bytes 1, 2: READ data, may be one or two bytes. For single byte parameters, the parameter shall
1408 be returned in Byte 1 and Byte 2 shall be set to 00h.

1409 • ECC byte covering the header

1410 The format for long READ packet responses is:

1411 • Byte 0: Data Identifier (Virtual Channel ID + Data Type)

1412 • Bytes 1-2: Word Count N (N = 0 to 65, 535)

1413 • ECC byte covering the header

1414 • N Bytes: READ data, may be from 1 to N bytes

1415 • Checksum, two bytes (16-bit checksum)

1416 • If Checksum is not calculated by the peripheral, send 0000h

1417 ## 8.9.5 Error Reporting Format

1418 An error report is a Short packet comprised of two bytes following the DI byte, with an ECC byte
1419 following the Error Report bytes. By convention, detection and reporting of each error type is signified by
1420 setting the corresponding bit to "1". Table 18 shows the bit assignment for all error reporting.

1421 **Table 18 Error Report Bit Definitions**

Bit	Description
0	SoT Error
1	SoT Sync Error
2	EoT Sync Error
3	Escape Mode Entry Command Error
4	Low-Power Transmit Sync Error
5	HS Receive Timeout Error
6	False Control Error
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	Invalid Transmission Length

Bit	Description
14	Reserved
15	DSI Protocol Violation

1422 The first seven bits, bit 0 through bit 6, are related to the physical layer errors that are described in sections
1423 7.1 and 7.2. Bits 8 and 9 are related to single-bit and multi-bit ECC errors. The remaining bits indicate DSI
1424 protocol-specific errors.

1425 A single-bit ECC error implies that the receiver has already corrected the error and continued with the
1426 previous transmission. Therefore, the data does not need to be retransmitted. A Checksum error can be
1427 detected and reported back to Host using a Bidirectional Link by a peripheral that has implemented CRC
1428 checking capability. A Host may retransmit the data or not.

1429 A DSI Data Type Not Recognized error is caused by receiving a Data Type that is either not defined or is
1430 defined but not implemented by the peripheral, e.g. a command mode peripheral may not implement video
1431 mode-specific commands such as streaming 18-bit packed RGB pixels. After encountering an unrecognized
1432 Data Type or multiple-bit ECC error, the receiver effectively loses packet boundaries within a transmission
1433 and shall drop the transmission from the point where the error was detected.

1434 DSI VC ID Invalid error is reported whenever a peripheral encounters a packet header with an
1435 unrecognizable VC ID.

1436 An Invalid Transmission Length error is detected whenever a peripheral receives an incorrect number of
1437 bytes within a particular transmission. For example, if the WC field of the header does not match the actual
1438 number of payload bytes for a particular packet. Depending on the number, as well as the contents, of the
1439 bytes following the error, there is a good chance that other types of errors such as Checksum, ECC or
1440 unrecognized Data Type could be detected. Another example would be a case where peripheral receives a
1441 short packet, i.e. four bytes plus EoT within a transmission, with a long Data Type code in the header. In
1442 general, the Host is responsible for maintaining the integrity of the DSI protocol. If the ECC field was
1443 detected correctly, implying that host may have made a mistake by inserting a wrong Data Type into the
1444 short packet, the following EoTp could be interpreted as payload for the previous packet by a peripheral.
1445 Depending on the WC field, a Checksum error or an unrecognized Data Type error could be detected. In
1446 effect, the receiver detects an invalid transmission length, sets bit 13 and reports it back to the host after the
1447 first BTA opportunity.

1448 In the previous example, the peripheral can also detect that an EoTp was not received correctly, which
1449 implies a protocol violation. Bit 15 is used to indicate DSI protocol violations where a peripheral
1450 encounters a situation where an expected EoTp was not received at the end of a transmission or an expected
1451 BTA was not received after a read request. Although host devices should maintain DSI protocol integrity,
1452 DSI peripherals shall be able to detect both these cases of protocol violation.

1453 Other protocol violation scenarios exist, but since there are only a limited number of bits for reporting
1454 errors, an extension mechanism is required. Peripheral vendors shall specify an implementation-specific
1455 error status register where a Host can obtain additional information regarding what type of protocol
1456 violation occurred by issuing a read request, e.g. via a generic DSI read packet, after receiving an
1457 *Acknowledge and Error Report* packet with bit 15 set. The type of protocol violations, along with the
1458 address of the particular error status register and the generic read packet format used to address this register
1459 shall be documented in the relevant peripheral data sheet. The peripheral data sheet and documentation
1460 format is out of scope for this document.

8.10 Peripheral-to-Processor Transactions – Detailed Format Description

1462 Table 19 presents the complete set of peripheral-to-processor Data Types.

1463   **Table 19 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 and Error Report	Short
03h – 07h	00 0011 – 00 0111	Reserved	
08h	00 1000	End of Transmission packet (EoTp)	Short
09h – 10h	00 1001 – 01 0000	Reserved	
11h	01 0001	Generic Short READ Response, 1 byte returned	Short
12h	01 0010	Generic Short READ Response, 2 bytes returned	Short
13h – 19h	01 0011 – 01 1001	Reserved	
1Ah	01 1010	Generic Long READ Response	Long
1Bh	01 1011	Reserved	
1Ch	01 1100	DCS Long READ Response	Long
1Dh – 20h	01 1101 – 10 0000	Reserved	
21h	10 0001	DCS Short READ Response, 1 byte returned	Short
22h	10 0010	DCS Short READ Response, 2 bytes returned	Short
23h – 3Fh	10 0011 – 11 1111	Reserved	

1464   8.10.1 Acknowledge and Error Report, Data Type 00 0010 (02h)

1465   *Acknowledge and Error Report* is sent in response to any command, or read request, with BTA asserted
1466   when a reportable error is detected in the preceding, or earlier, transmission from the host processor. In the
1467   case of a correctible ECC error, this packet is sent following the requested READ data packet in the same
1468   LP transmission.

1469   When multiple peripherals share a single DSI, the *Acknowledge and Error Report* packet shall be tagged
1470   with the Virtual Channel ID 0b00.

1471   Although some errors, such as a correctable ECC error, can be associated with a packet targeted at a
1472   specific peripheral, an uncorrectable error cannot be associated with any particular peripheral. Additionally,
1473   many detectable error types are PHY-level transmission errors and cannot be associated with specific
1474   packets.

1475   8.10.2 Generic Short Read Response, 1 or 2 Bytes, Data Types = 01 0001 or 01
1476   0010, Respectively

1477   This is the short-packet response to *Generic READ Request*. Packet composition is the Data Identifier (DI)
1478   byte, two bytes of payload data and an ECC byte. The number of valid bytes is indicated by the Data Type
1479   LSBs, DT bits [1:0]. DT = 01 0001 indicates one byte and DT = 01 0010 indicates two bytes are returned.

1480   For a single-byte read response, valid data shall be returned in the first (LS) byte, and the second (MS) byte
1481   shall be sent as 00h.

1482   This form of data transfer may be used for other features incorporated on the peripheral, such as a touch-
1483   screen integrated on the display module. Data formats for such applications are outside the scope of this
1484   document.

1485   If the command itself is possibly corrupt, due to an uncorrectable ECC error, SoT or SoT Sync error, the
1486   requested READ data packet shall not be sent and only the *Acknowledge and Error Report* packet shall be
1487   sent.

8.10.3 Generic Long Read Response with Optional Checksum, Data Type = 01 1010 (1Ah)

1490   This is the long-packet response to *Generic READ Request*. Packet composition is the Data Identifier (DI)
1491   byte followed by a two-byte Word Count, an ECC byte, N bytes of payload, and a two-byte Checksum. If
1492   the peripheral is Checksum capable, it shall return a calculated two-byte Checksum appended to the N-byte
1493   payload data. If the peripheral does not support Checksum it shall return 0000h.

1494   If the command itself is possibly corrupt, due to an uncorrectable ECC error, SoT or SoT Sync error, the
1495   requested READ data packet shall not be sent and only the *Acknowledge and Error Report* packet shall be
1496   sent.

8.10.4 DCS Long Read Response with Optional Checksum, Data Type 01 1100 (1Ch)

1499   This is a Long packet response to *DCS Read Request*. Packet composition is the Data Identifier (DI) byte
1500   followed by a two-byte Word Count, an ECC byte, N bytes of payload, and a two-byte Checksum. If the
1501   peripheral is Checksum capable, it shall return a calculated two-byte Checksum appended to the N-byte
1502   payload data. If the peripheral does not support Checksum it shall return 0000h.

1503   If the DCS command itself is possibly corrupt, due to uncorrectable ECC error, SoT or SoT Sync error, the
1504   requested READ data packet shall not be sent and only the *Acknowledge and Error Report* packet shall be
1505   sent.

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

1508   This is the short-packet response to *DCS Read Request*. Packet composition is the Data Identifier (DI) byte,
1509   two bytes of payload data and an ECC byte. The number of valid bytes is indicated by the Data Type LSBs,
1510   DT bits [1:0]. DT = 01 0001 indicates one byte and DT = 01 0010 indicates two bytes are returned. For a
1511   single-byte read response, valid data shall be returned in the first (LS) byte, and the second (MS) byte shall
1512   be sent as 00h.

1513   If the command itself is possibly corrupt, due to an uncorrectable ECC error, SoT or SoT Sync error, the
1514   requested READ data packet shall not be sent and only the *Acknowledge and Error Report* packet shall be
1515   sent.

8.10.6 Multiple Transmissions and Error Reporting

1517   A peripheral shall report all errors documented in Table 18, when a command or request is followed by
1518   BTA giving bus possession to the peripheral. Peripheral shall accumulate errors from multiple transactions
1519   up until a time that host is issuing a BTA. After that, only one ACK Trigger Message or *Acknowledge and*
1520   *Error Report* packet shall be returned regardless of the number of packets or transmissions. Notice that host
1521   may not be able to associate each error to a particular packet or transmission causing that error.

1522   If receiving an *Acknowledge and Error Report* for each and every packet is desired, software can send
1523   individual packets within separate transmissions. In this case, a BTA follows each individual transmission.
1524   Furthermore, the peripheral may choose to store other information about errors that may be recovered by
1525   the host processor at a later time. The format and access mechanism of such additional error information is
1526   outside the scope of this document.

8.10.7 Clearing Error Bits

1528   Errors shall be accumulated by the peripheral during single or multiple transmissions and only cleared after
1529   they have been reported back to the host processor. Errors are transmitted as part of an *Acknowledge and*
1530   *Error Report* response after the host issues a BTA.

8.11 Video Mode Interface Timing

1532   Video Mode peripherals require pixel data delivered in real time. This section specifies the format and
1533   timing of DSI traffic for this type of display module.

8.11.1 Transmission Packet Sequences

1535   DSI supports several formats, or packet sequences, for Video Mode data transmission. The peripheral's
1536   timing requirements dictate which format is appropriate. In the following sections, *Burst Mode* refers to
1537   time-compression of the RGB pixel (active video) portion of the transmission. In addition, these terms are
1538   used throughout the following sections:

1539   • Non-Burst Mode with Sync Pulses – enables the peripheral to accurately reconstruct original video
1540   timing, including sync pulse widths.

1541   • Non-Burst Mode with Sync Events – similar to above, but accurate reconstruction of sync pulse
1542   widths is not required, so a single *Sync Event* is substituted.

1543   • Burst mode – RGB pixel packets are time-compressed, leaving more time during a scan line for
1544   LP mode (saving power) or for multiplexing other transmissions onto the DSI link.

1545   Note that for accurate reconstruction of timing, packet overhead including Data ID, ECC, and Checksum
1546   bytes should be taken into consideration.

1547   The host processor shall support all of the packet sequences in this section. A Video Mode peripheral shall
1548   support at least one of the packet sequences in this section. The peripheral shall not require any additional
1549   constraints regarding packet sequence or packet timing. The peripheral supplier shall document all relevant
1550   timing parameters listed in Table 20.

1551   In the following figures the Blanking or Low-Power Interval (BLLP) is defined as a period during which
1552   video packets such as pixel-stream and sync event packets are not actively transmitted to the peripheral.

1553   To enable PHY synchronization the host processor should periodically end HS transmission and drive the
1554   Data Lanes to the LP state. This transition should take place at least once per frame; shown as LPM in the
1555   figures in this section. It is recommended to return to LP state once per scanline during the horizontal
1556   blanking time. Regardless of the frequency of BLLP periods, the host processor is responsible for meeting
1557   all documented peripheral timing requirements. Note, at lower frequencies BLLP periods will approach, or
1558   become, zero, and burst mode will be indistinguishable from non-burst mode.

1559   During the BLLP the DSI Link may do any of the following:

1560   • Remain in Idle Mode with the host processor in LP-11 state and the peripheral in LP-RX

1561   • Transmit one or more non-video packets from the host processor to the peripheral using Escape
1562   Mode

1563 • Transmit one or more non-video packets from the host processor to the peripheral using HS Mode

1564 • If the previous processor-to-peripheral transmission ended with BTA, transmit one or more
1565 packets from the peripheral to the host processor using Escape Mode

1566 • Transmit one or more packets from the host processor to a different peripheral using a different
1567 Virtual Channel ID

1568 The sequence of packets within the BLLP or RGB portion of a HS transmission is arbitrary. The host
1569 processor may compose any sequence of packets, including iterations, within the limits of the packet format
1570 definitions. For all timing cases, the first line of a frame shall start with VSS; all other lines shall start with
1571 VSE or HSS. Note that the position of synchronization packets, such as VSS and HSS, in time is of utmost
1572 importance since this has a direct impact on the visual performance of the display panel.

1573 Normally, RGB pixel data is sent with one full scanline of pixels in a single packet. If necessary, a
1574 horizontal scanline of active pixels may be divided into two or more packets. However, individual pixels
1575 shall not be split across packets.

1576 Transmission packet components used in the figures in this section are defined in Figure 22 unless
1577 otherwise specified.
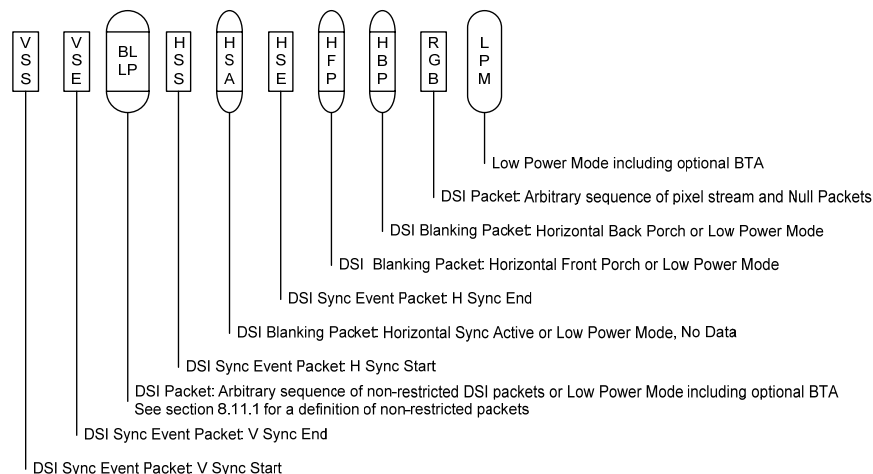
1578
1579 **Figure 22 Video Mode Interface Timing Legend**

1580 If a peripheral timing specification for HBP or HFP minimum period is zero, the corresponding Blanking
1581 Packet may be omitted. If the HBP or HFP maximum period is zero, the corresponding blanking packet
1582 shall be omitted.

1583 **8.11.2 Non-Burst Mode with Sync Pulses**

1584 With this format, the goal is to accurately convey DPI-type timing over the DSI serial Link. This includes
1585 matching DPI pixel-transmission rates, and widths of timing events like sync pulses. Accordingly,
1586 synchronization periods are defined using packets transmitting both start and end of sync pulses. An
1587 example of this mode is shown in Figure 23.

1588
1589
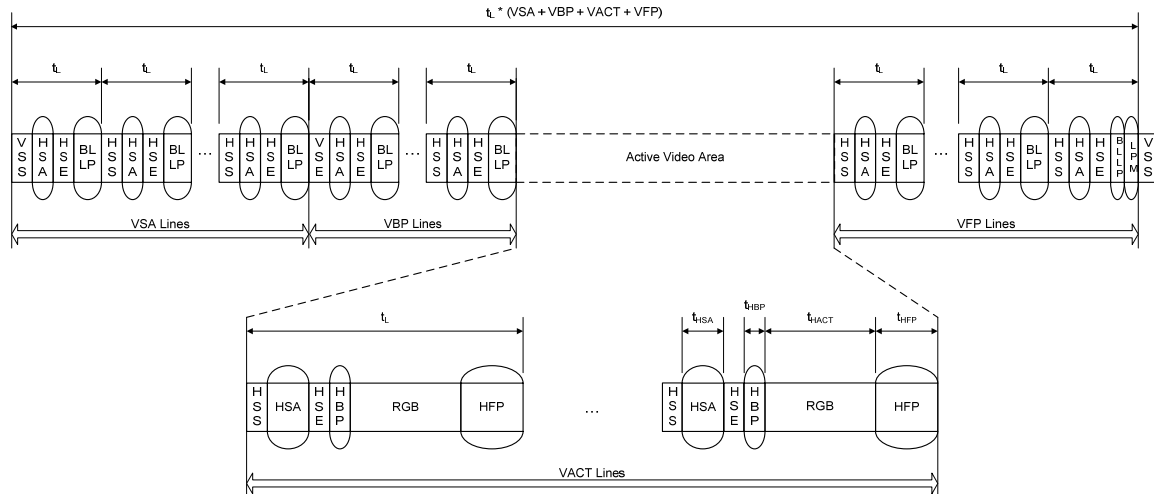
Figure 23 Video Mode Interface Timing: Non-Burst Transmission with Sync Start and End

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

8.11.3 Non-Burst Mode with Sync Events

This mode is a simplification of the format described in section 8.11.2. Only the start of each synchronization pulse is transmitted. The peripheral may regenerate sync pulses as needed from each Sync Event packet received. Pixels are transmitted at the same rate as they would in a corresponding parallel display interface such as DPI-2. An example of this mode is shown in Figure 24.

1602
1603
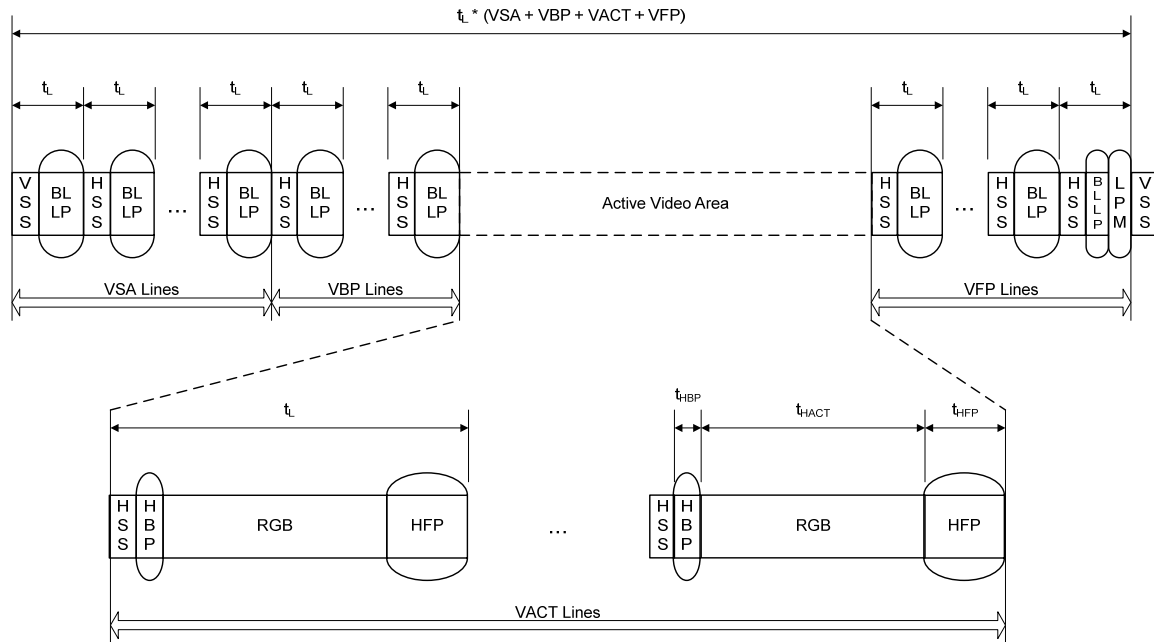
Figure 24 Video Mode Interface Timing: Non-burst Transmission

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

8.11.4 Burst Mode

In this mode, blocks of pixel data can be transferred in a shorter time using a time-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.

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 may stay in HS Mode for sending blanking packets or go 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 25.

1617
1618
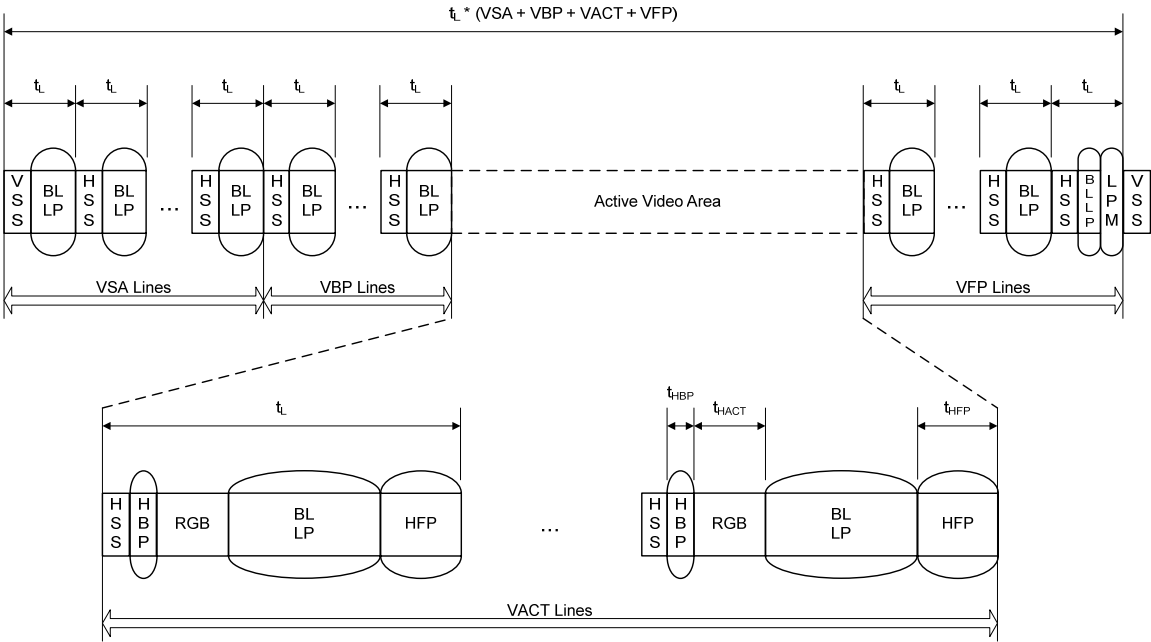
Figure 25 Video Mode Interface Timing: Burst Transmission

1620 Similar to the Non-Burst Mode scenario, if there is sufficient time to transition from HS to LP mode and
1621 back again, a timed interval in LP mode may substitute for a Blanking Packet, thus saving power.

8.11.5 Parameters

1623 Table 20 documents the parameters used in the preceding figures. Peripheral supplier companies are
1624 responsible for specifying suitable values for all blank fields in the table. The host processor shall meet
1625 these requirements to ensure interoperability.

1626 For periods when Data Lanes are in LP Mode, the peripheral shall also specify whether the DSI Clock Lane
1627 may go to LP. The host processor is responsible for meeting minimum timing relationships between clock
1628 activity and HS transmission on the Data Lanes as documented in *MIPI Alliance Specification for D-
1629 PHY*[4].

1630   **Table 20 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_{HSA}	Horizontal sync active			µs	
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	

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

8.12 TE Signaling in DSI

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

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

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

set_tear_on and set_tear_scanline are sent to the display module as DSI Data Type 15h (DCS Short Write, one parameter) and DSI Data Type 39h (DCS Long Write/writeLUT), respectively. The host processor ends the transmission with Bus Turn-Around asserted, giving bus possession to the display module. Since the display module's DSI Protocol layer does not interpret DCS commands, but only passes them through to the display controller, it responds with a normal Acknowledge and returns bus possession to the host processor. In this state, the display module cannot report TE events to the host processor since it does not have bus possession.

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

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

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

- The display module shall send the LP Escape Mode sequence

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

1665   •   The display module shall then return bus possession to the host processor

1666 This Trigger Message is reserved by DSI for TE signaling only and shall not be used for any other purpose
1667 in a DSI-compliant interface.

1668 See *MIPI Alliance Standard for Display Command Set (DCS)*[1] for detailed descriptions of the TE related
1669 commands, and command and parameter formats.

9   Error-Correcting Code (ECC) and Checksum

1670

9.1   Packet Header Error Detection/Correction

1671

1672   The host processor in a DSI-based system shall generate an error-correction code (ECC) and append it to
1673   the header of every packet sent to the peripheral. The ECC takes the form of a single byte following the
1674   header bytes. The ECC byte shall provide single-bit error correction and 2-bit error detection for the entire
1675   Packet Header. See Figure 13 and Figure 14 for descriptions of the Long and Short Packet Headers,
1676   respectively.

1677   ECC shall always be generated and appended in the Packet Header from the host processor. Peripherals
1678   with Bidirectional Links shall also generate and send ECC.

1679   Peripherals in unidirectional DSI systems, although they cannot report errors to the host, shall still take
1680   advantage of ECC for correcting single-bit errors in the Packet Header.

9.2   Hamming Code Theory

1681

1682   The number of parity or error check bits required is given by the Hamming rule, and is a function of the
1683   number of bits of information transmitted. The Hamming rule is expressed by the following inequality:

1684 $d + p + 1 <= 2^p$ where d is the number of data bits and p is the number of parity bits.

1685   The result of appending the computed parity bits to the data bits is called the Hamming code word. The size
1686   of the code word c is $d+p$, and a Hamming code word is described by the ordered set (c, d).

1687   A Hamming code word is generated by multiplying the data bits by a generator matrix G. This
1688   multiplication's result is called the code word vector $(c1, c2, c3,...cn)$, consisting of the original data bits
1689   and the calculated parity bits. The generator matrix G used in constructing Hamming codes consists of I,
1690   the identity matrix, and a parity generation matrix A:

1691 $G = [\, I \mid A\,]$

1692   The Packet Header plus the ECC code can be obtained as: PH=p*G where p represents the header and G is
1693   the corresponding generator matrix.

1694   Validating the received code word r involves multiplying it by a parity check to form s, the syndrome or
1695   parity check vector: s = H*PH where PH is the received Packet Header and H is the parity check matrix:

1696 $H = [A^T \mid I]$

1697   If all elements of s are zero, the code word was received correctly. If s contains non-zero elements, then at
1698   least one error is present. If the header has a single-bit error, then the syndrome s matches one of the
1699   elements of H, which will point to the bit in error. Furthermore, if the bit in error is a parity bit, then the
1700   syndrome will be one of the elements on I, or else it will be the data bit identified by the position of the
1701   syndrome in A^T.

9.3   Hamming-modified Code Applied to DSI Packet Headers

1702

1703   Hamming codes use parity to correct a single-bit error or detect a two-bit error, but are not capable of doing
1704   both simultaneously. DSI uses Hamming-modified codes where an extra parity bit is used to support both

1705   single error correction as well as two-bit error detection. For example a 7+1 bit Hamming-modified code
1706   (72, 64) allows for protection of up to 64 data bits. DSI systems shall use a 5+1 bit Hamming-modified
1707   code (30, 24), allowing for protection of up to twenty-four data bits. The addition of a parity bit allows a
1708   five bit Hamming code to correct a single-bit error and detect a two-bit error simultaneously.

1709   Since Packet Headers are fixed at four bytes (twenty-four data bits and eight ECC bits), P6 and P7 of the
1710   ECC byte are unused and shall be set to zero by the transmitter. The receiver shall ignore P6 and P7 and set
1711   both bits to zero before processing ECC. Table 21 shows a compact way to specify the encoding of parity
1712   and decoding of syndromes.

1713   **Table 21 ECC Syndrome Association Matrix**

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

1714   Each cell in the matrix represents a syndrome and each syndrome in the matrix is MSB left aligned:

1715   e.g. 0x07=0b0000_0111=P7P6P5P4P3P2P1P0

1716   The top row defines the three LSB of data position bit, and the left column defines the three MSB of data
1717   position bit for a total of 64-bit positions.

1718   e.g. 38th bit position (D37) is encoded 0b100_101 and has the syndrome 0x68.

1719   To correct a single bit error, the syndrome shall be one of the syndromes in the table, which will identify
1720   the bit position in error. The syndrome is calculated as:

1721 $S=P_{SEND}{}^{\wedge}P_{RECEIVED}$ where P_{SEND} is the 6-bit ECC field in the header and $P_{RECEIVED}$ is the
1722   calculated parity of the received header.

1723   Table 22 represents the same information as in Table 21, organized to provide better insight into how parity
1724   bits are formed from data bits.

Table 22 ECC Parity Generation Rules

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

Data Bit	P7	P6	P5	P4	P3	P2	P1	P0	Hex
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
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

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

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

1729   The first twenty-four data bits, D0 to D23, in Table 22 contain the complete DSI Packet Header. The
1730   remaining bits, D24 to D63, are informative (shown in yellow in the table) and not relevant to DSI.
1731   Therefore, the parity bit calculation can be optimized to:

1732   P7=0

1733   P6=0

1734   P5=D10^D11^D12^D13^D14^D15^D16^D17^D18^D19^D21^D22^D23

1735   P4=D4^D5^D6^D7^D8^D9^D16^D17^D18^D19^D20^D22^D23

1736   P3=D1^D2^D3^D7^D8^D9^D13^D14^D15^D19^D20^D21^D23

1737   P2=D0^D2^D3^D5^D6^D9^D11^D12^D15^D18^D20^D21^D22

1738   P1=D0^D1^D3^D4^D6^D8^D10^D12^D14^D17^D20^D21^D22^D23

1739   P0=D0^D1^D2^D4^D5^D7^D10^D11^D13^D16^D20^D21^D22^D23

1740   Note, the parity bits relevant to the ECC calculation, P0 through P5, in the table are shown in red and the
1741   unused bits, P6 and P7, are shown in blue.

9.4   ECC Generation on the Transmitter

1742

1743   ECC is generated from the twenty-four data bits within the Packet Header as illustrated in Figure 26, which
1744   also serves as an ECC calculation example. Note that the DSI protocol uses a four byte Packet Header. See
1745   section 8.4.1 and section 8.4.2 for Packet Header descriptions for Long and Short packets, respectively.
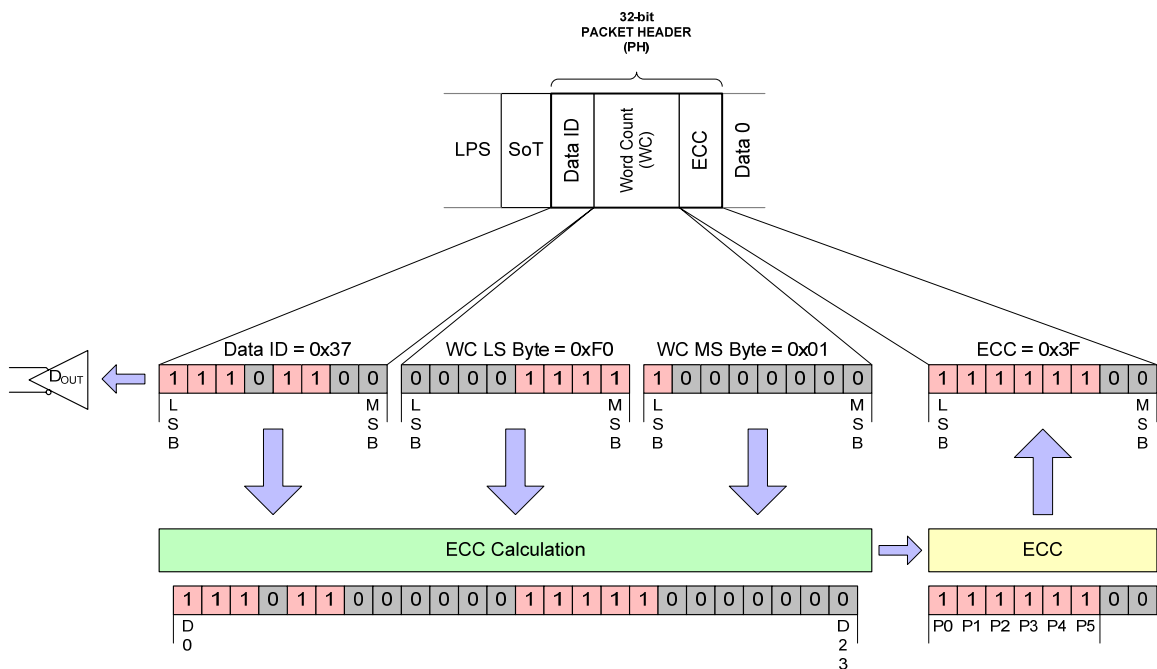
1746
1747

1748   **Figure 26 24-bit ECC generation on TX side**

1749 ## 9.5   Applying ECC on the Receiver

1750   Applying ECC on the receiver involves generating a new ECC for the received packet, computing the
1751   syndrome using the new ECC and the received ECC, decoding the syndrome to find if a single-error has
1752   occurred and if so, correcting the error. If a multiple-bit error is identified, it is flagged and reported to the
1753   transmitter. Note, error reporting is only applicable to bidirectional DSI implementations.

1754   ECC generation on the receiver side shall apply the same padding rules as ECC generation for
1755   transmission.
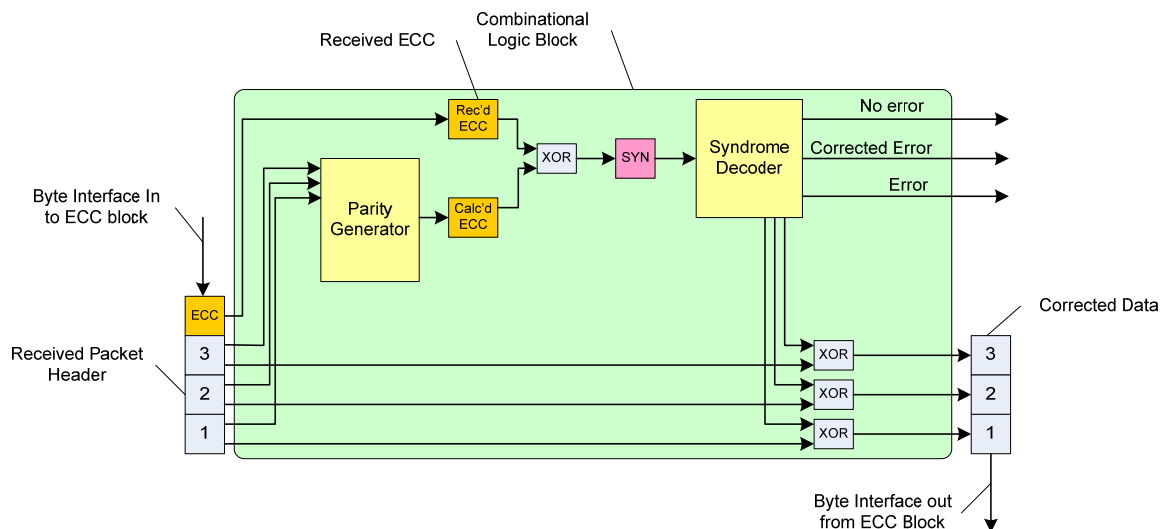
1756
1757   **Figure 27 24-bit ECC on RX Side Including Error Correction**

1758   Decoding the syndrome has three aspects:

1759   •   Testing for errors in the Packet Header. If syndrome = 0, no errors are present.

1760   •   Test for a single-bit error in the Packet Header by comparing the generated syndrome with the
1761   matrix in Table 21. If the syndrome matches one of the entries in the table, then a single-bit error
1762   has occurred and the corresponding bit is in error. This position in the Packet Header shall be
1763   complemented to correct the error. Also, if the syndrome is one of the rows of the identity matrix
1764   **I**, then a parity bit is in error. If the syndrome cannot be identified then a multi-bit error has
1765   occurred. In this case the Packet Header is corrupted and cannot be restored. Therefore, the Multi-
1766   bit Error Flag shall be set.

1767   •   Correcting the single-bit error if detected, as indicated above.

1768   ## 9.6   Checksum Generation for Long Packet Payloads

1769   Long packets are comprised of a Packet Header protected by an ECC byte as specified in sections 9.3
1770   through 9.5, and a payload of 0 to 2^{16} - 1 bytes. To detect errors in transmission of Long packets, a
1771   checksum is calculated over the payload portion of the data packet. Note that, for the special case of a zero-
1772   length payload, the 2-byte checksum is set to FFFFh.

1773   The checksum can only indicate the presence of one or more errors in the payload. Unlike ECC, the
1774   checksum does not enable error correction. For this reason, checksum calculation is not useful for
1775   unidirectional DSI implementations since the peripheral has no means of reporting errors to the host
1776   processor.

1777 Checksum generation and transmission is mandatory for host processors sending Long packets to
1778 peripherals. It is optional for peripherals transmitting Long packets to the host processor. However, the
1779 format of Long packets is fixed; peripherals that do not support checksum generation shall transmit two
1780 bytes having value 0000h in place of the checksum bytes when sending Long packets to the host processor.

1781 The host processor shall disable checksum checking for received Long packets from peripherals that do not
1782 support checksum generation.

1783 The checksum shall be realized as a 16-bit CRC with a generator polynomial of $x^{16}+x^{12}+x^5+x^0$.

1784 The transmission of the checksum is illustrated in Figure 28. The LS byte is sent first, followed by the MS
1785 byte. Note that within the byte, the LS bit is sent first.

16-bit Checksum

CRC LS Byte	CRC MS Byte

16-bit PACKET FOOTER (PF)
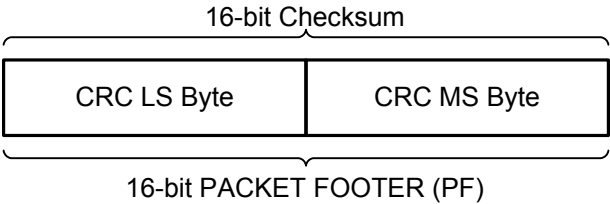
1786
1787   **Figure 28 Checksum Transmission**

1788 The CRC implementation is presented in Figure 29. The CRC shift register shall be initialized to FFFFh
1789 before packet data enters. Packet data not including the Packet Header then enters as a bitwise data stream
1790 from the left, LS bit first. Each bit is fed through the CRC shift register before it is passed to the output for
1791 transmission to the peripheral. After all bytes in the packet payload have passed through the CRC shift
1792 register, the shift register contains the checksum. C15 contains the checksum's MSB and C0 the LSB of the
1793 16-bit checksum. The checksum is then appended to the data stream and sent to the receiver. The receiver
1794 uses its own generated CRC to verify that no errors have occurred in transmission. See Annex B for an
1795 example of checksum generation.

Polynomial: $x^{16} + x^{12} + x^5 + x^0$
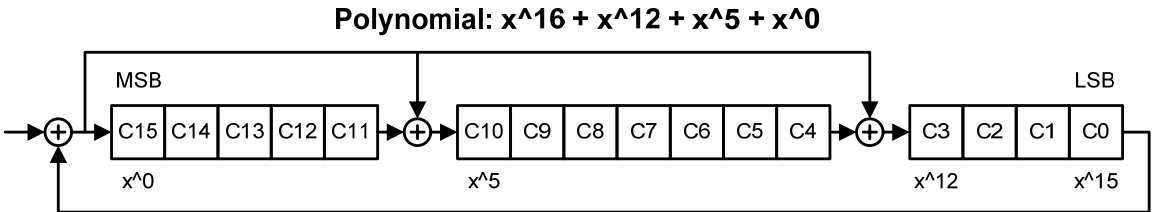
1796
1797
1798   **Figure 29 16-bit CRC Generation Using a Shift Register**

1799 Section 8.10.1 documents the peripheral response to detection of an error in a Long packet payload.

10 Compliance, Interoperability, and Optional Capabilities

1800

1801 This section documents requirements and classifications for MIPI-compliant host processors and
1802 peripherals. There are a number of categories of potential differences or attributes that shall be considered
1803 to ensure interoperability between a host processor and a peripheral, such as a display module:

1804 Manufacturers shall document a DSI device's capabilities and specifications for the parameters listed in
1805 this section.

1806   1. Display Resolutions

1807   2. Pixel Formats

1808   3. Number of Lanes

1809   4. Maximum Lane Frequency

1810   5. Bidirectional Communication and Escape Mode Support

1811   6. ECC and Checksum capabilities

1812   7. Display Architecture

1813   8. Multiple Peripheral Support

1814 EoTp support and interoperability between DSI v1.01-compliant and earlier revision devices are discussed
1815 in section 10.9.

1816 In general, the peripheral chooses one option from each category in the list above. For example, a display
1817 module may implement a resolution of 320x240 (QVGA), a pixel format of 16-bpp and use two Lanes to
1818 achieve its required bandwidth. Its data path has bidirectional capability, it does not implement
1819 checksum-testing capability, and it operates in Video Mode only.

10.1 Display Resolutions

1820

1821 Host processors shall implement one or more of the display resolutions in Table 23.

1822 **Table 23 Display Resolutions**

Resolution	Horizontal Extent	Vertical Extent
QQVGA	160	120
QCIF	176	144
QCIF+	176	208
QCIF+	176	220
QVGA	320	240
CIF	352	288
CIF+	352	416
CIF+	352	440

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

10.2 Pixel Formats

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

10.2.1 Video Mode

Peripherals shall implement at least one of the following pixel formats. Host processors shall implement all of the following pixel formats.

1. 16 bpp (5, 6, 5 RGB), each pixel using two bytes; see section 8.8.14

2. 18 bpp (6, 6, 6 RGB) packed; see section 8.8.15

3. 18 bpp (6, 6, 6 RGB) loosely packed into three bytes; see section 8.8.16

4. 24 bpp (8, 8, 8 RGB), each pixel using three bytes; see section 8.8.17

10.2.2 Command Mode

Peripherals shall implement at least one of the pixel formats, and host processors should implement all of the pixel formats, defined in *MIPI Alliance Standard for Display Command Set (DCS)*[1].

10.3 Number of Lanes

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

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

10.4 Maximum Lane Frequency

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

1844 ## 10.5 Bidirectional Communication

1845 Because Command Mode depends on the use of the READ command, a Command Mode display module
1846 shall implement bidirectional communications. For display modules without on-panel buffers that work
1847 only in Video Mode, bidirectional operation on DSI is optional.

1848 Since a host processor may implement both Command- and Video Modes of operations, it should support
1849 bidirectional operation and Escape Mode transmission and reception.

1850 ## 10.6 ECC and Checksum Capabilities

1851 A DSI host processor shall calculate and transmit an ECC byte for both Long and Short packets. The host
1852 processor shall also calculate and transmit a two-byte Checksum for Long packets. A DSI peripheral shall
1853 support ECC, but may support Checksum. If a peripheral does not calculate Checksum it shall still be
1854 capable of receiving Checksum bytes from the host processor. If a peripheral supports bidirectional
1855 communications and does not support Checksum it shall send bytes of all zeros in the appropriate fields.
1856 For interoperability with earlier revision of DSI peripherals where ECC was considered an optional feature,
1857 host shall be able to enable/disable ECC capability based on the particular peripheral ECC support
1858 capability. The enabling/disabling mechanism is out of scope of DSI. In effect, if an earlier revision
1859 peripheral was not supporting ECC, it shall still be capable of receiving ECC byte from the host and
1860 sending an all zero ECC byte back to the host for responses over a bidirectional link. See section 9 for more
1861 details on ECC and Checksum.

1862 ## 10.7 Display Architecture

1863 A display module may implement Type 1, Type 2, Type 3 or Type 4 display architecture as described in
1864 *MIPI Alliance Standard for Display Bus Interface (DBI-2)*[2] and *MIPI Alliance Standard for Display*
1865 *Pixel Interface (DPI-2)*[3]. Type 1 architecture works in Command Mode only. Type 2 and Type 3
1866 architectures use the DSI interface for both Command- and Video Modes of operation. Type 4 architectures
1867 operate in Video Mode only, although there may be additional control signals. Therefore, a peripheral may
1868 use Command Mode only, Video Mode only, or both Command- and Video Modes of operation.

1869 The host processor may support either or both Command- and Video Modes of operation. If the host
1870 processor supports Command Mode, it shall also support the mandatory command set specified in *MIPI*
1871 *Alliance Standard for Display Command Set (DCS)*[1].

1872 ## 10.8 Multiple Peripheral Support

1873 DSI supports multiple peripherals per DSI Link using the Virtual Channel field of the Data Identifier byte.
1874 See sections 4.2.3 and 8.5.1 for more details.

1875 A host processor should support a minimum of two peripherals.

1876 ## 10.9 EoTp Support and Interoperability

1877 EoTp generation or detection is mandatory for devices compliant with this version of the DSI specification.
1878 Devices compliant to DSI specification v1.0 and earlier do not support EoTp. In order to ensure
1879 interoperability with earlier devices, current devices shall provide a means to enable or disable EoTp
1880 generation or detection. In effect, this capability can be disabled by the system designer whenever a device
1881 on either side of the Link does not support EoTp.

1882 Annex A Contention Detection and Recovery
1883 Mechanisms (informative)

1884 The following describes optional capabilities at the PHY and Protocol layers that provide additional
1885 robustness for a DSI Link against possible data-signal contention as a consequence of transient errors in the
1886 system. These capabilities improve the system's chances of detecting any of several possible contention
1887 cases, and provide mechanisms for "graceful" recovery without resorting to a hard reset.

1888 These capabilities combine circuitry in the I/O cell, to directly detect contention, with logic and timers in
1889 the protocol to avert and recover from other forms of contention.

1890 A.1   PHY Detected Contention

1891 The PHY can detect two types of contention faults: LP High Fault and LP Low Fault.

1892 The LP High Fault and LP Low Fault are caused by both sides of the Link transmitting simultaneously.
1893 Note, the LP High Fault and LP Low Fault are only applicable for bidirectional Data Lanes. Refer to *MIPI*
1894 *Alliance Specification for D-PHY*[4] for definition of LP High and LP Low faults.

1895 A.1.1   Protocol Response to PHY Detected Faults

1896 The Protocol shall specify how both ends of the Link respond when contention is flagged. It shall ensure
1897 that both devices return to *Stop* state (LP-11), with one side going to *Stop TX* and the other to *Stop RX*.

1898 When both PHYs are in LP mode, one or both PHYs will detect contention between LP-0 and LP-1.

1899 The following tables describe the resolution sequences for different types of contention and detection.

1900 Table sequences:

1901   • Sequence of events to resolve LP High ←→ LP Low Contention

1902   • Case 1: Both sides initially detect the contention

1903   • Case 2: Only the Host Processor initially detects contention

1904   • Case 3: Only the Peripheral initially detects contention

1905   **Table 24 LP High ←→ LP Low Contention Case 1**

Host Processor Side		Peripheral Side	
Protocol	**PHY**	**PHY**	**Protocol**
	Detect *LP High Fault* or *LP Low Fault*	Detect *LP High Fault* or *LP Low Fault*	
	Transition to *Stop* State (LP-11)	Transition to LP-RX	
Host Processor Wait Timeout		Peripheral waits until it observes *Stop* state before responding	
		Observe *Stop* state	

Host Processor Side		Peripheral Side	
Protocol	**PHY**	**PHY**	**Protocol**
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 Stop State (LP-11)	Remain in LP-RX	(reset may continue)
Peripheral Reset Timeout. Wait until Peripheral completes Reset before resuming normal operation.	Continue normal operation.		Reset completes

1906   Note: The protocol may want to request a Reset after contention is flagged a single time. Alternately, the
1907   protocol may choose not to Reset but instead continue normal operation after detecting a single contention.
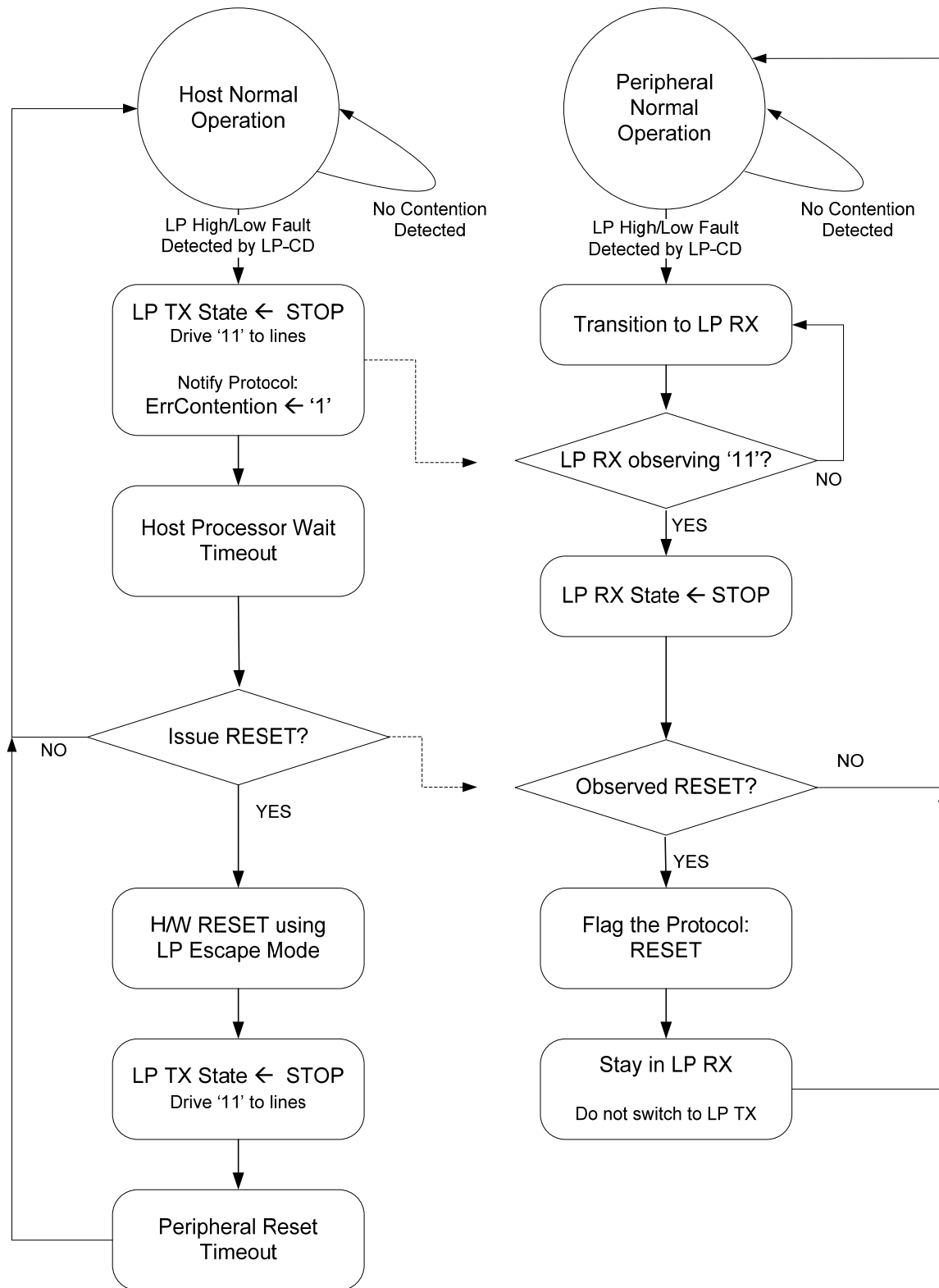1908   It could then initiate a Reset after multiple contentions are flagged, or never initiate a Reset.

Figure 30 LP High ←→ LP Low Contention Case 1

1909
1910

1911

1912 **Table 25 LP High ←→ LP Low Contention Case 2**

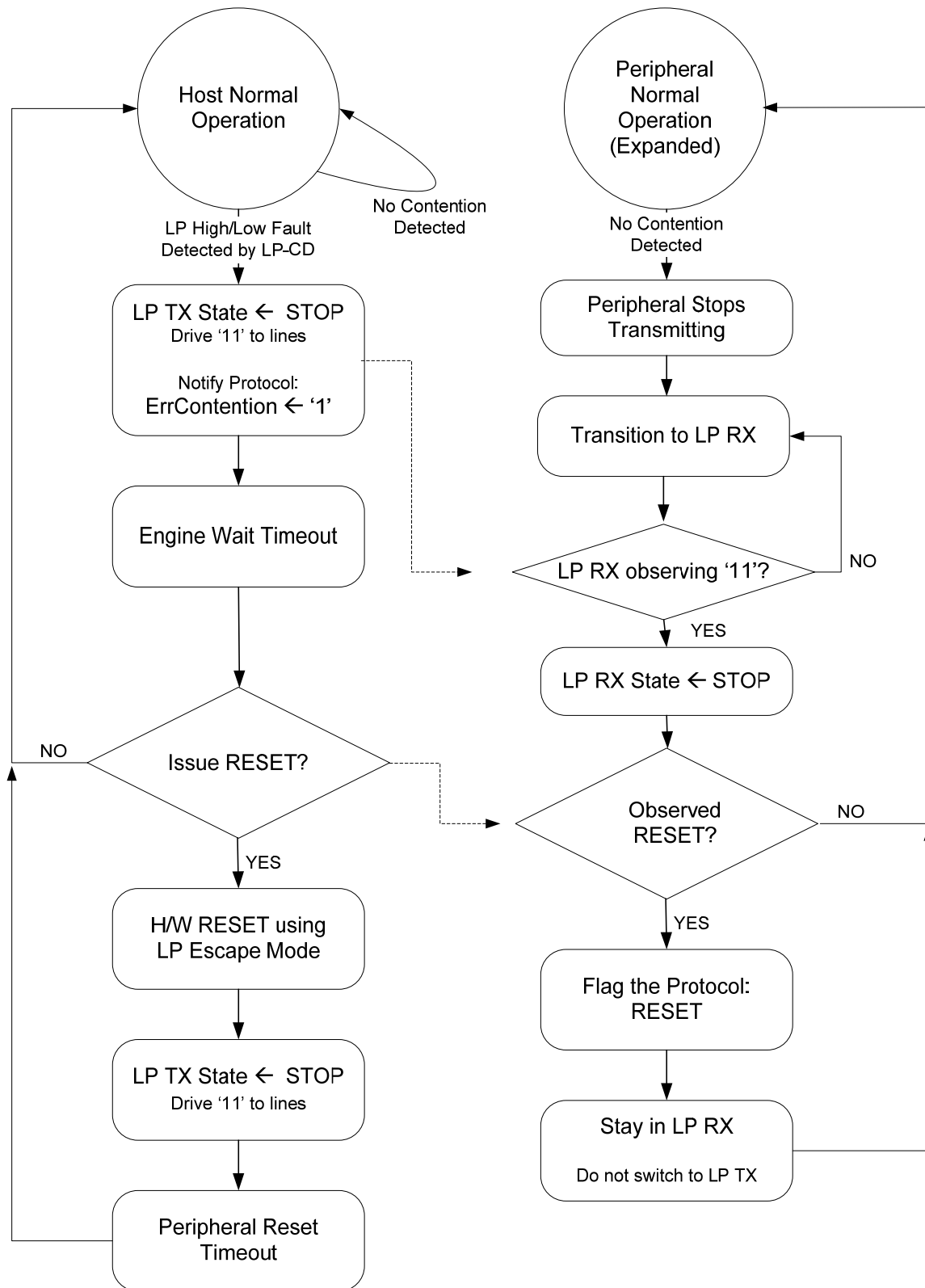| Host Processor Side		Peripheral Side	
Protocol	**PHY**	**PHY**	**Protocol**
	Detect *LP High Fault* or *LP Low Fault*	No EL contention detected	
	Transition to *Stop* State (LP-11)	No EL contention detected	
Host Processor Wait Timeout			Peripheral Bus Possession Timeout
		Transition to LP-RX	
		Observe *Stop* state	
Request *Reset Entry* command to PHY	Send *Reset Entry* command	Observe *Reset Entry* command	
		Flag Protocol: *Reset* command received	Observe *Reset* Command
			Reset Peripheral
	Return to *Stop* 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

Figure 31 LP High ←→ LP Low Contention Case 2

1913
1914
1915

1916   **Table 26 LP High ←→ LP Low Contention Case 3**

Host Processor Side		Peripheral Side	
Protocol	**PHY**	**PHY**	**Protocol**
	No detection of EL contention	Detect *LP High Fault* or *LP Low Fault*	
		Transition to LP-RX	
		Peripheral waits until it observes *Stop* state before responding to bus activity.	
	Normal transition to *Stop* State (LP-11)	Observe *Stop* State	
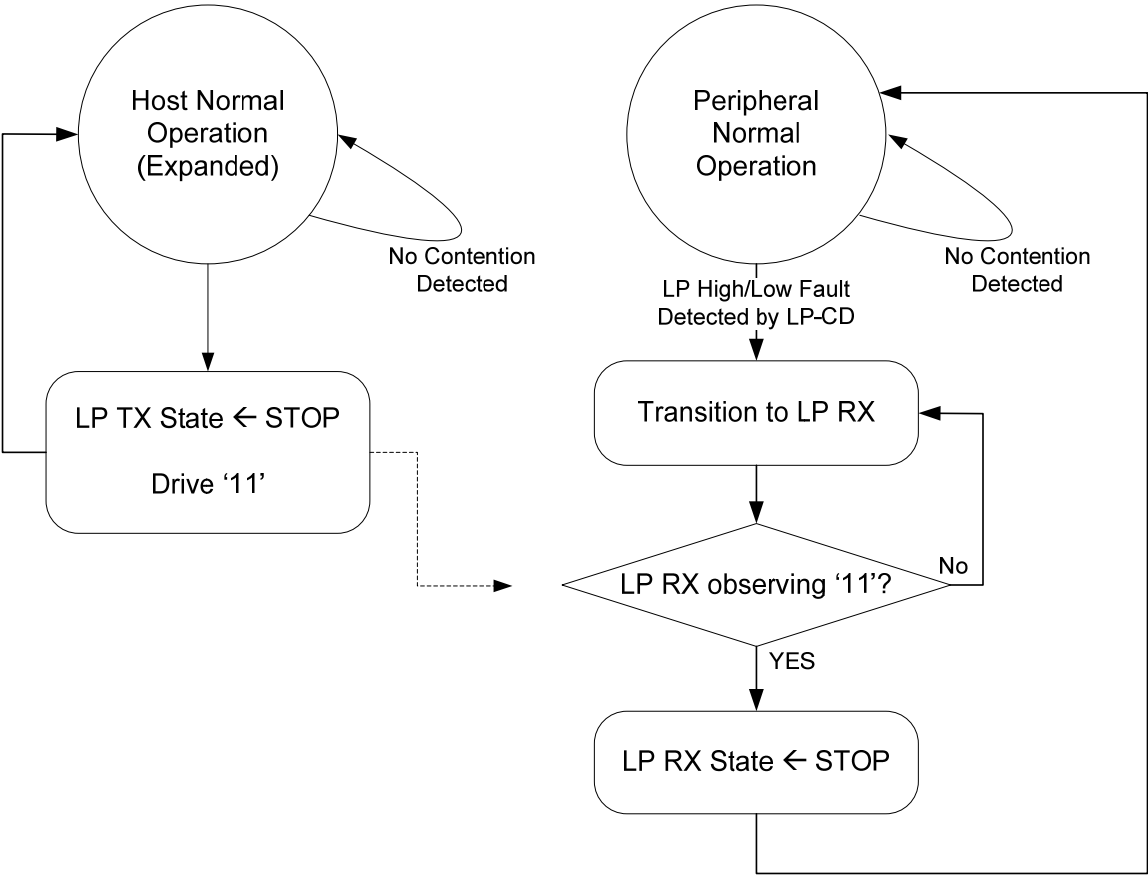
1917
1918   **Figure 32 LP High ←→ LP Low Contention Case 3**

1919 Annex B Checksum Generation Example
1920   (informative)

1921 The following C/C++ program provides a simple software routine to calculate CRC of a packet of variable
1922 length. The `main` routine calls subroutine `CalculateCRC16` to calculate the CRC based on the data in
1923 one of the `gpcTestData[]` arrays and prints the CRC results.

```
1924
1925 /* *********************** DECLARATIONS *********************** */
1926 #include <stdio.h>
1927
1928 /* Start of Test Data */
1929 static unsigned char gpcTestData0[] = { 0x00 };
1930 static unsigned char gpcTestData1[] = { 0x01 };
1931 static unsigned char gpcTestData2[] = { 0xFF, 0x00, 0x00, 0x00, 0x1E,
1932       0xF0, 0x1E, 0xC7, 0x4F, 0x82, 0x78, 0xC5, 0x82, 0xE0, 0x8C, 0x70,
1933       0xD2, 0x3C, 0x78, 0xE9, 0xFF, 0x00, 0x00, 0x01 };
1934 static unsigned char gpcTestData3[] = { 0xFF, 0x00, 0x00, 0x02, 0xB9,
1935       0xDC, 0xF3, 0x72, 0xBB, 0xD4, 0xB8, 0x5A, 0xC8, 0x75, 0xC2, 0x7C,
1936       0x81, 0xF8, 0x05, 0xDF, 0xFF, 0x00, 0x00, 0x01 };
1937 #define NUMBER_OF_TEST_DATA0_BYTES 1
1938 #define NUMBER_OF_TEST_DATA1_BYTES 1
1939 #define NUMBER_OF_TEST_DATA2_BYTES 24
1940 #define NUMBER_OF_TEST_DATA3_BYTES 24
1941 /* End of Test Data */
1942
1943 unsigned short CalculateCRC16( unsigned char *pcDataStream, unsigned
1944 short sNumberOfDataBytes );
1945
1946 /* *********************** MAIN ROUTINE *********************** */
1947 void main( void )
1948 {
1949       unsigned short sCRC16Result;
1950       sCRC16Result = CalculateCRC16( gpcTestData2,
1951                                     NUMBER_OF_TEST_DATA2_BYTES );
1952       printf( "Checksum CS[15:0] = 0x%04X\n", sCRC16Result );
1953 }
1954 /* ********** END OF MAIN ****** START OF CRC CALCULATION ********** */
1955
1956 /* CRC16 Polynomial, logically inverted 0x1021 for x^16+x^15+x^5+x^0 */
1957 static unsigned short gsCRC16GenerationCode = 0x8408;
1958
1959 unsigned short CalculateCRC16( unsigned char *pcDataStream, unsigned
1960                               short sNumberOfDataBytes )
1961 {
1962       /*
1963       sCRC16Result: the return of this function,
1964       sByteCounter: address pointer to count the number of the
1965                     calculated data bytes
1966       cBitCounter: counter for bit shift (0 to 7)
1967       cCurrentData: byte size buffer to duplicate the calculated data
1968                     byte for a bit shift operation
1969       */
1970       unsigned short sByteCounter;
1971       unsigned char cBitCounter;
```

```
1972          unsigned char cCurrentData;
1973          unsigned short sCRC16Result = 0xFFFF;
1974          if ( sNumberOfDataBytes > 0 )
1975          {
1976                 for ( sByteCounter = 0; sByteCounter < sNumberOfDataBytes;
1977                     sByteCounter++ )
1978                 {
1979                     cCurrentData = *( pcDataStream + sByteCounter );
1980                     for ( cBitCounter = 0; cBitCounter < 8;
1981                         cBitCounter++ )
1982                     {
1983                         if ( ( ( sCRC16Result & 0x0001 ) ^ ( ( 0x0001 *
1984                             cCurrentData) & 0x0001 ) ) > 0 )
1985                             sCRC16Result = ( ( sCRC16Result >> 1 )
1986                                 & 0x7FFF ) ^ gsCRC16GenerationCode;
1987                         else
1988                             sCRC16Result = ( sCRC16Result >> 1 )
1989                                             & 0x7FFF;
1990                         cCurrentData = (cCurrentData >> 1 ) & 0x7F;
1991                     }
1992                 }
1993          }
1994          return sCRC16Result;
1995   }
1996   /* *************** END OF SUBROUTINE TO CALCULATE CRC ************* */
```

1997   Outputs from the various input streams are as follows:

1998   Data (gpcTestData0): 00
1999   Checksum CS[15:0] = 0x0F87

2000   Data (gpcTestData1): 01
2001   Checksum CS[15:0] = 0x1E0E

2002   Data (gpcTestData2): FF 00 00 00 1E F0 1E C7 4F 82 78 C5 82 E0 8C 70 D2 3C
2003   78 E9 FF 00 00 01
2004   Checksum CS[15:0] = 0xE569

2005   Data (gpcTestData3): FF 00 00 02 B9 DC F3 72 BB D4 B8 5A C8 75 C2 7C 81 F8
2006   05 DF FF 00 00 01
2007   Checksum CS[15:0] = 0x00F0