MIPI Alliance Standard for Display Serial Interface V1.0

MIPI Board approved 5 April 2006

* Caution to Implementers *

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

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

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

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

MIPI Alliance Standard for Display Serial Interface

Version 1.00a – 19 April 2006

MIPI Board Approved 5-Apr-2006

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

NOTICE OF DISCLAIMER

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

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

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

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

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

Contents

Version 1.00 – 13 April 2006 i

1 Overview 8

1.1 Scope 8

1.2 Purpose 8

2 Terminology (Informational) 9

2.1 Definitions 9

2.2 Abbreviations 10

2.3 Acronyms 10

3 References (Informational) 13

3.1 DBI and DBI-2 (Display Bus Interface Standards for Parallel Signaling) 13

3.2 DPI and DPI-2 (Display Pixel Interface Standards for Parallel Signaling) 13

3.3 DCS (Display Command Set) 14

3.4 CSI-2 (Camera Serial Interface 2) 14

3.5 D-PHY (MIPI Alliance Standard for Physical Layer) 14

4 DSI Introduction 15

4.1 DSI Layer Definitions 16

4.2 Command and Video Modes 17

4.2.1 Command Mode 17

4.2.2 Video Mode Operation 17

4.2.3 Virtual Channel Capability 18

5 DSI Physical Layer 19

5.1 Data Flow Control 19

5.2 Bidirectionality and Low Power Signaling Policy 19

5.3 Command Mode Interfaces 20

5.4 Video Mode Interfaces 20

5.5 Bidirectional Control Mechanism 20

5.6 Clock Management....21
5.6.1 Clock Requirements....21
5.6.2 Clock Power and Timing....22
6 Multi-Lane Distribution and Merging....23
6.1 Multi-Lane Interoperability and Lane-number Mismatch....24
6.1.1 Clock Considerations with Multi-Lane....25
6.1.2 Bi-directionality and Multi-Lane Capability....25
6.1.3 SoT and EoT in Multi-Lane Configurations....25
7 Low-Level Protocol Errors and Contention....28
7.1 Low-Level Protocol Errors....28
7.1.1 SoT Error....28
7.1.2 SoT Sync Error....29
7.1.3 EoT Sync Error....29
7.1.4 Escape Mode Entry Command Error....30
7.1.5 LP Transmission Sync Error....30
7.1.6 False Control Error....31
7.2 Contention Detection and Recovery....31
7.2.1 Contention Detection in LP Mode....32
7.2.2 Contention Recovery Using Timers....32
7.3 Additional Timers....34
7.3.1 Turnaround Acknowledge Timeout (TA_TO)....34
7.3.2 Peripheral Reset Timeout (PR_TO)....35
7.4 Acknowledge and Error Reporting Mechanism....35
8 DSI Protocol....37
8.1 Multiple Packets per Transmission....37
8.2 Packet Composition....37
8.3 Endian Policy....38
8.4 General Packet Structure....38

8.4.1 Long Packet Format 38

8.4.2 Short Packet Format 40

8.5 Common Packet Elements 40

8.5.1 Data Identifier Byte 40

8.5.2 Error Correction Code 41

8.6 Interleaved Data Streams 41

8.6.1 Interleaved Data Streams and Bi-directionality 42

8.7 Processor to Peripheral Direction (Processor-Sourced) Packet Data Types 42

8.8 Processor-to-Peripheral Transactions – Detailed Format Description 43

8.8.1 Sync Event (H Start, H End, V Start, V End), Data Type = xx 0001 (x1h) 43

8.8.2 Color Mode On Command, Data Type = 00 0010 (02h) 44

8.8.3 Color Mode Off Command, Data Type = 01 0010 (12h) 44

8.8.4 Shutdown Peripheral Command, Data Type = 10 0010 (22h) 44

8.8.5 Turn On Peripheral Command, Data Type = 11 0010 (32h) 44

8.8.6 Generic Short WRITE Packet, 0 to 7 Parameters, Data Type = xx x011 (x3h and xBh) 44

8.8.7 Generic READ Request, 0 to 7 Parameters, Data Type = xx x100 (x4h and xCh) 44

8.8.8 DCS Commands 45

8.8.9 Set Maximum Return Packet Size, Data Type = 11 0111 (37h) 46

8.8.10 Null Packet (Long), Data Type = 00 1001 (09h) 46

8.8.11 Blanking Packet (Long), Data Type = 01 1001 (19h) 46

8.8.12 Generic Non-Image Data (Long), Data Type = 10 1001 (29h) 47

8.8.13 Packed Pixel Stream, 16-bit Format, Long packet, Data Type 00 1110 (0Eh) 47

8.8.14 Packed Pixel Stream, 18-bit Format, Long packet, Data type = 01 1110 (1Eh) 48

8.8.15 Pixel Stream, 18-bit Format in Three Bytes, Long packet, Data Type = 10 1110 (2Eh) 49

8.8.16 Packed Pixel Stream, 24-bit Format, Long packet, Data Type = 11 1110 (3Eh) 50

8.8.17 DO NOT USE and Reserved Data Types 50

8.9 Peripheral-to-Processor (Reverse Direction) LP Transmissions 51

8.9.1 Packet Structure for Peripheral-to-Processor LP Transmissions 51

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

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

MIPI Alliance Standard for Display Serial Interface

1 Overview

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

1.1 Scope

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

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

1.2 Purpose

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

2 Terminology (Informational)

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.

2.1 Definitions

发送方会发送时钟控制传输

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 is 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.

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

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

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

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

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

最小一根数据线，一根时钟线

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

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

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

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

Word Count: Number of bytes.

2.2 Abbreviations

e.g. For example

2.3 Acronyms

AM Active matrix (display technology)

AIP Application Independent Protocol

ASP Application Specific Protocol

BLLP Blanking or Low Power interval

BPP Bits per Pixel

BTA Bus Turn-Around

CSI Camera Serial Interface

DBI Display Bus Interface

DI Data Identifier

DMA Direct Memory Access

DPI Display Pixel Interface

DSI	Display Serial Interface
DT	Data Type
ECC	Error-Correcting Code
EMI	Electro Magnetic interference
EoT	End of Transmission
ESD	Electrostatic Discharge
Fps	Frames per second
HS	High Speed
ISTO	Industry Standards and Technology Organization
LLP	Low-Level Protocol
LP	Low Power
LPI	Low Power Interval
LPS	Low Power State (state of serial data line when not transferring high-speed serial data)
LSB	Least Significant Bit
Mbps	Megabits per second
MIPI	Mobile Industry Processor Interface
MSB	Most Significant Bit
PE	Packet End
PF	Packet Footer
PH	Packet Header
PHY	Physical Layer
PI	Packet Identifier
PPI	PHY-Protocol Interface
PS	Packet Start
PT	Packet Type
PWB	Printed Wired Board
QCIF	Quarter-size CIF (resolution 176x144 pixels or 144x176 pixels)
QVGA	Quarter-size Video Graphics Array (resolution 320x240 pixels or 240x320 pixels)

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

3 References (Informational)

[1] MIPI Alliance Standard for Display Command Set, version 1.00, April 2006

[2] MIPI Alliance Standard for Display Bus Interface, version 2.00, November 2005

[3] MIPI Alliance Standard for Display Parallel Interface, version 2.00, September 2005

[4] MIPI Alliance Standard for D-PHY, version 0.65, November 2005

Design and Analysis of Fault Tolerant Digital System by Barry W. Johnson

Error Correcting Codes: Hamming Distance by Don Johnson paper

Intel 8206 error detection and correction unit datasheet

National DP8400-2 Expandable Error Checker/Corrector datasheet

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

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

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

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

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

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

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

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

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

3.3 DCS (Display Command Set)

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

3.4 CSI-2 (Camera Serial Interface 2)

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

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

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

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

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

4 DSI Introduction

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

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

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

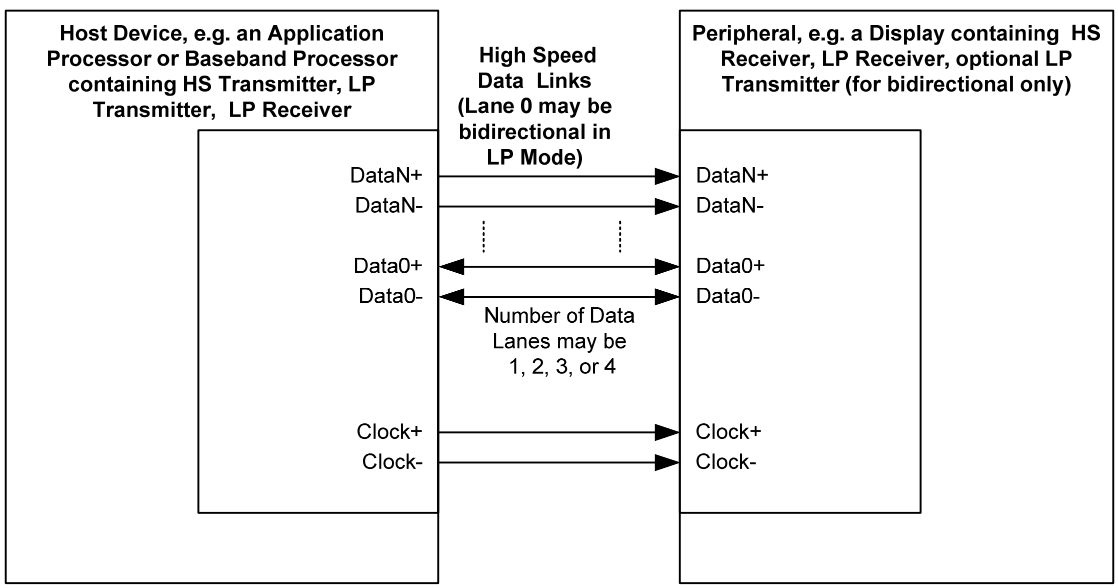


Figure 1 DSI Transmitter and Receiver Interface

4.1 DSI Layer Definitions

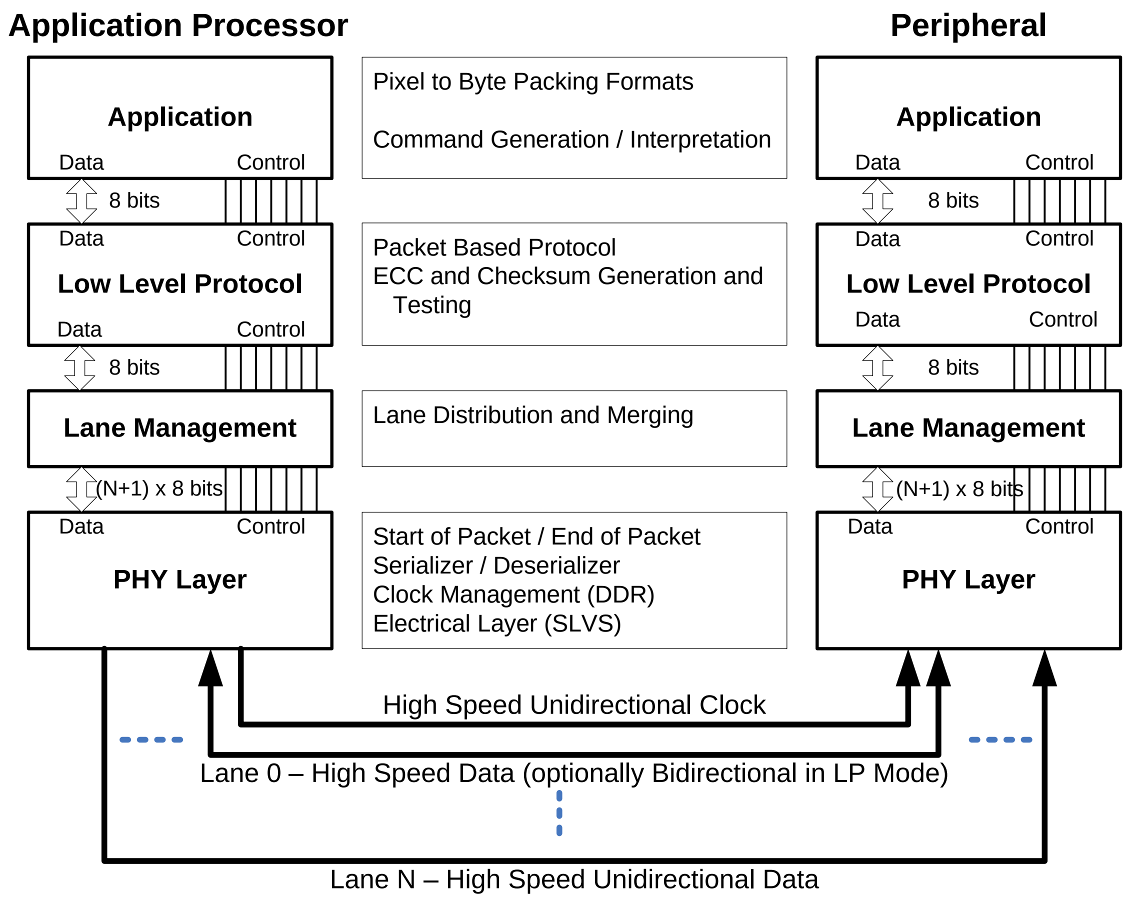


Figure 2 DSI Layers

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

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

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

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

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

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

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

4.2 Command and Video Modes

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

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

4.2.1 Command Mode

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

4.2.2 Video Mode Operation

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

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

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

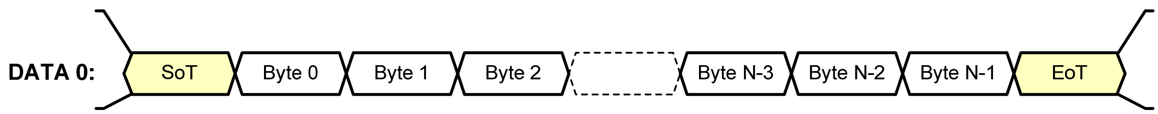
The DSI protocol permits up to four virtual channels, enabling traffic for multiple peripherals to share a common DSI Link. In some high-resolution display designs, multiple physical drivers serve different areas of a common display panel. Each driver is integrated with its own display controller that 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.

5 DSI Physical Layer

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

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

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



KEY:
SoT – Start of Transmission EoT – End of Transmission

Figure 3 Basic HS Transmission Structure

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

5.1 Data Flow Control

传输完才停止

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

命令模式中：data0 是双向的，其他单向，显示模式中：data0 可双可单，其他单向

5.2 Bidirectionality and Low Power Signaling Policy

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

For both interface types, the Clock Lane shall be driven by the host processor only, never by the peripheral. clk 只能由主机发送

低速模式只用 data0 传输

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

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

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

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

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

5.3 Command Mode Interfaces

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

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

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

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

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

5.4 Video Mode Interfaces

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

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

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

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

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

5.5 Bidirectional Control Mechanism

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

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

5.6 Clock Management

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

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

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

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

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

5.6.1 Clock Requirements

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

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

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

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

5.6.2 Clock Power and Timing

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

6 Multi-Lane Distribution and Merging

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

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

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

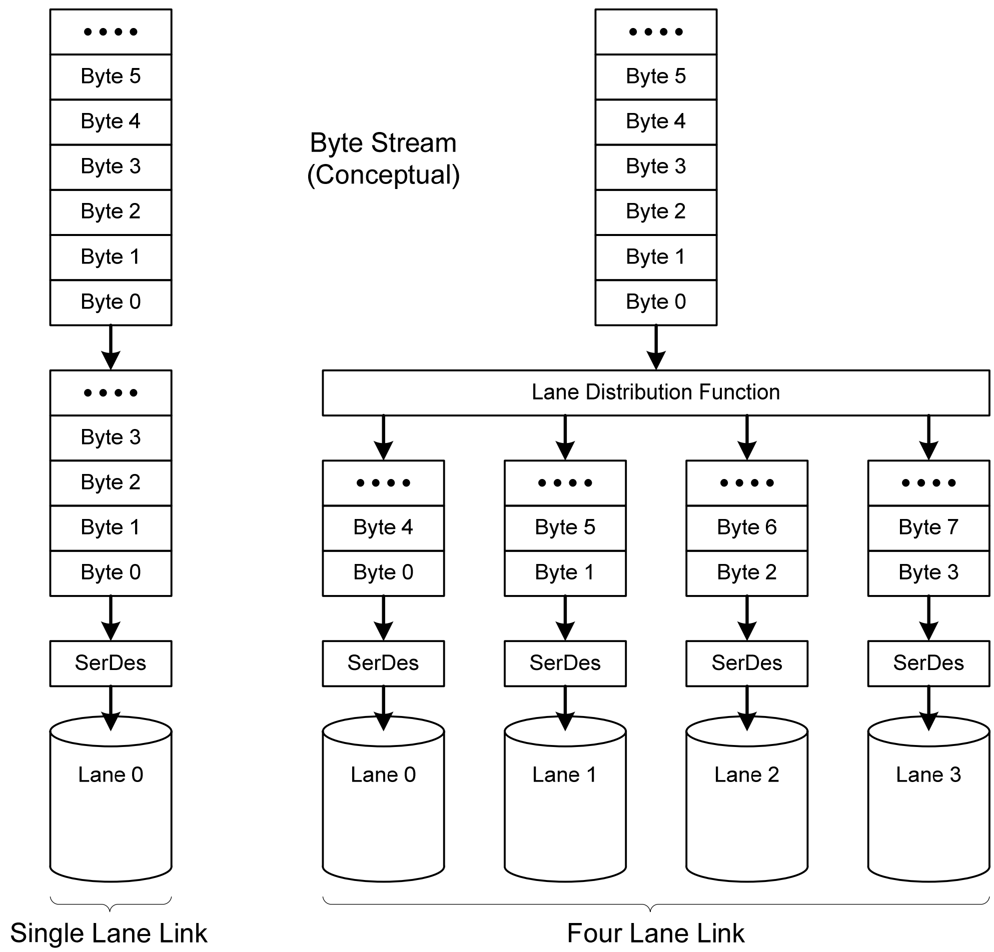


Figure 4 Lane Distributor Conceptual Overview

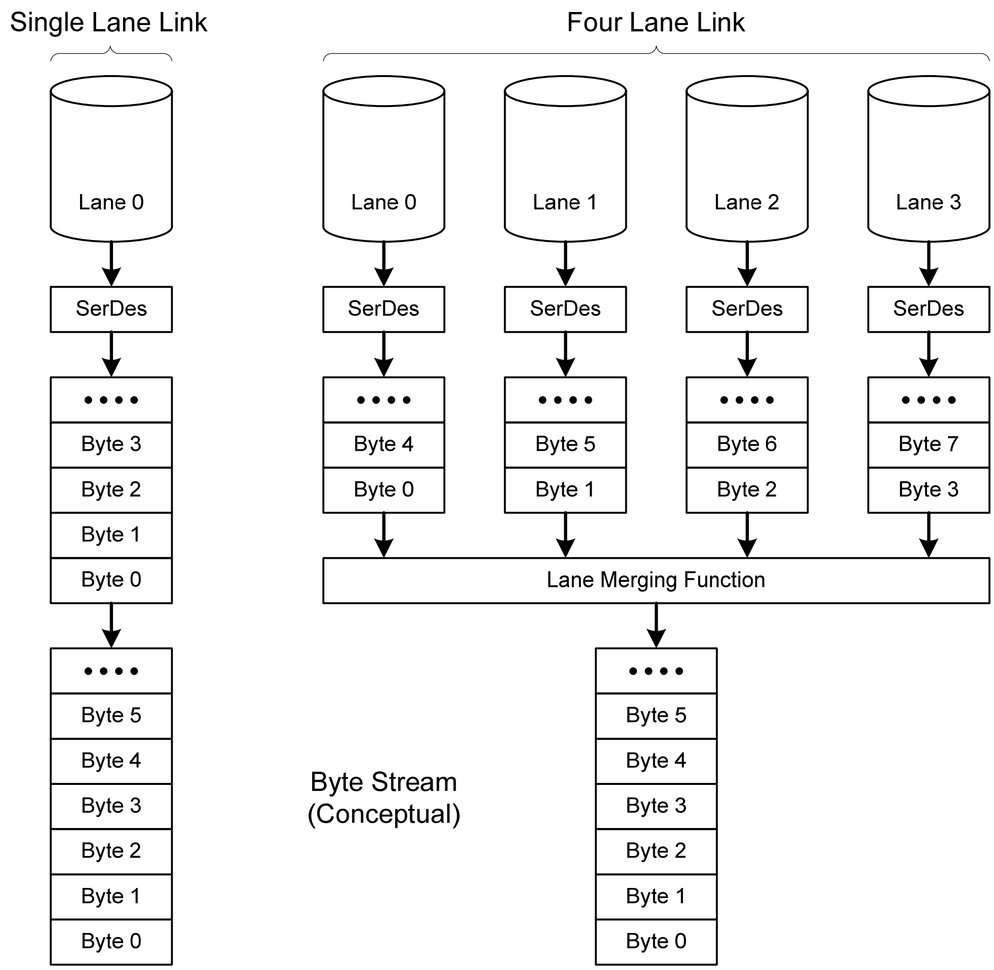


Figure 5 Lane Merger Conceptual Overview

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

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

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

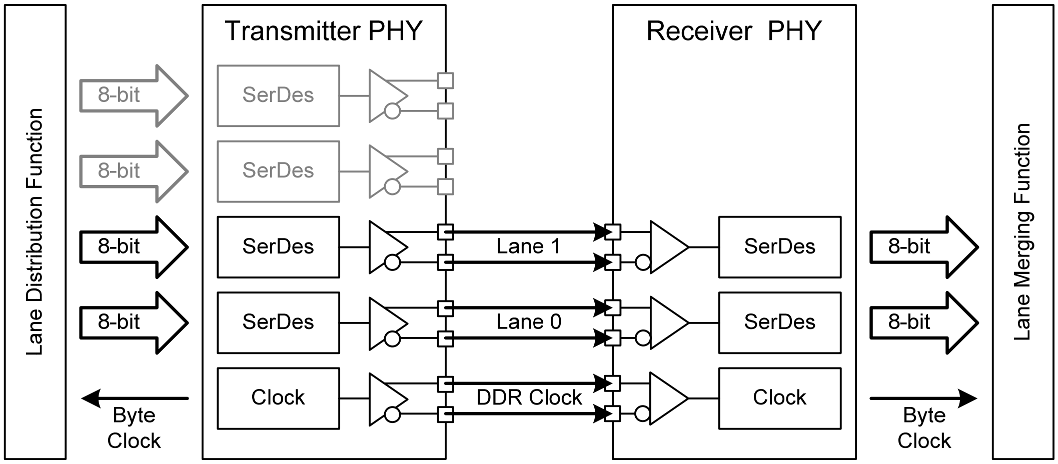


Figure 6 Four-Lane Transmitter with Two-Lane Receiver Example

6.1.1 Clock Considerations with Multi-Lane

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

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

6.1.2 Bi-directionality and Multi-Lane Capability

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

6.1.3 SoT and EoT in Multi-Lane Configurations

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

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

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

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

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

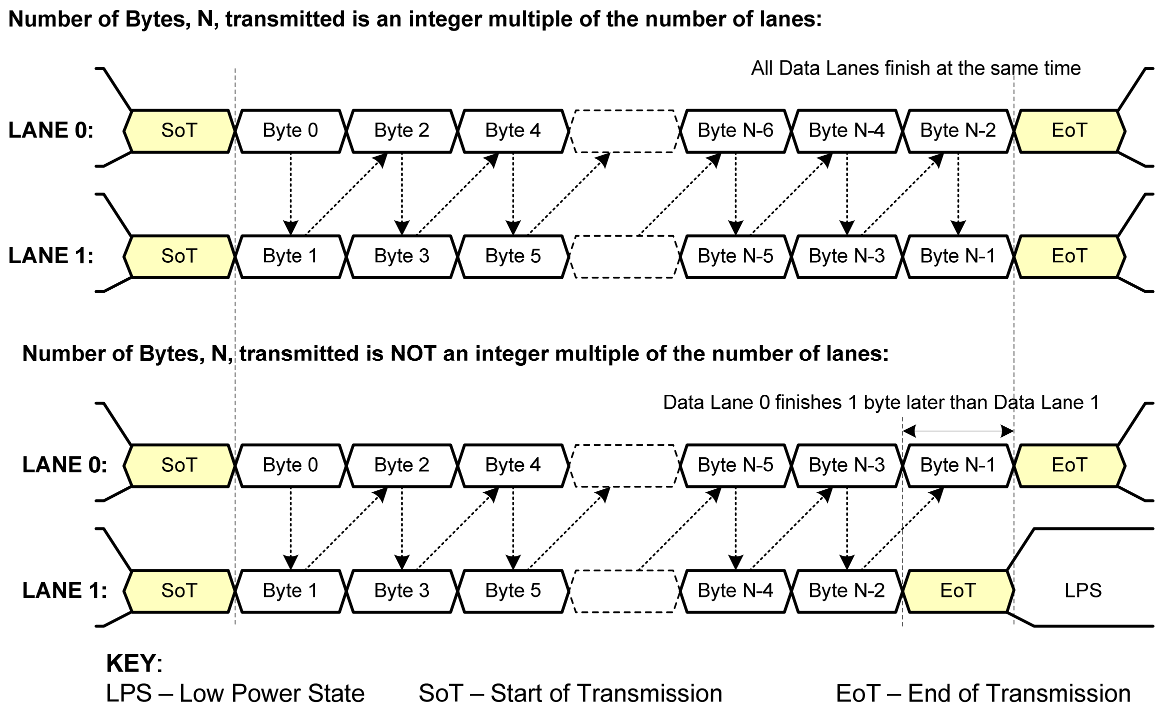
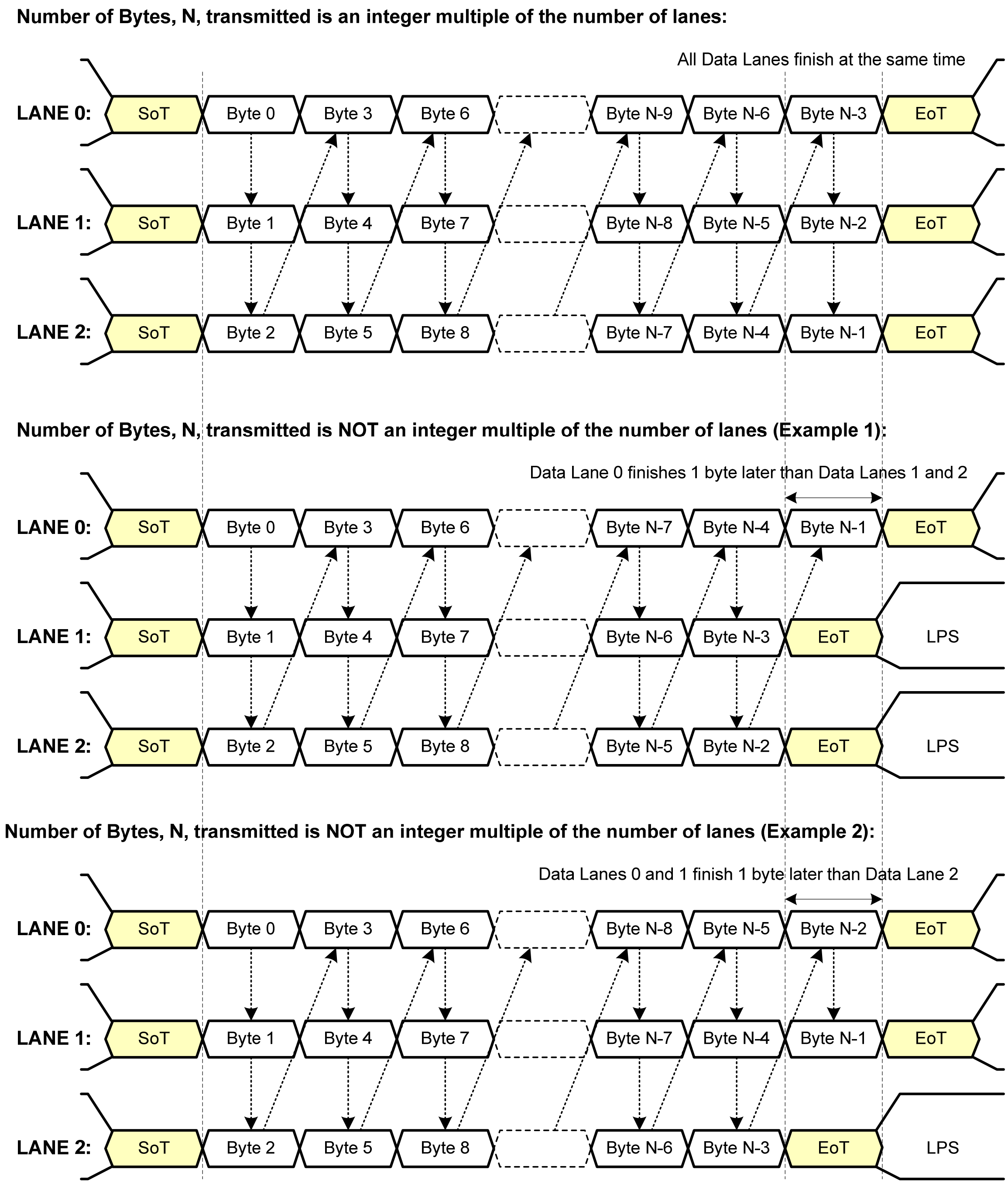


Figure 7 Two Lane HS Transmission Example



599 **KEY**:
LPS – Low Power State SoT – Start of Transmission EoT – End of Transmission

600 **Figure 8 Three Lane HS Transmission Example**

7 Low-Level Protocol Errors and Contention

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

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

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

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

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

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

7.1 Low-Level Protocol Errors

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

- SoT Error
- SoT Sync Error
- EoT Sync Error
- Escape Mode Entry Command Error
- LP Transmission Sync Error
- False Control Error

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

7.1.1 SoT Error

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

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

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

7.1.2 SoT Sync Error

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

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

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

7.1.3 EoT Sync Error

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

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

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

7.1.4 Escape Mode Entry Command Error

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

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

7.1.5 LP Transmission Sync Error

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

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

7.1.6 False Control Error

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

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	

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

7.2 Contention Detection and Recovery

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

7.2.1 Contention Detection in LP Mode

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

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

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

7.2.2 Contention Recovery Using Timers

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

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

7.2.2.1 Summary of Required Contention Recovery Timers

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

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

7.2.2.2 HS RX Timeout (HRX_TO) in Peripheral

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

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

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

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

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

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

7.2.2.3 HS TX Timeout (HTX_TO) in Host Processor

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

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)

7.2.2.4 LP TX-Peripheral Timeout (LTX-P_TO)

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

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

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

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

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

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

7.3 Additional Timers

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

7.3.1 Turnaround Acknowledge Timeout (TA_TO)

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

Table 13 Sequence of Events for Turnaround 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

Table 14 Sequence of Events for Turnaround 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)	

7.3.2 Peripheral Reset Timeout (PR_TO)

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

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

7.4 Acknowledge and Error Reporting Mechanism

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

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

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

8 DSI Protocol

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

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

8.1 Multiple Packets per Transmission

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

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

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

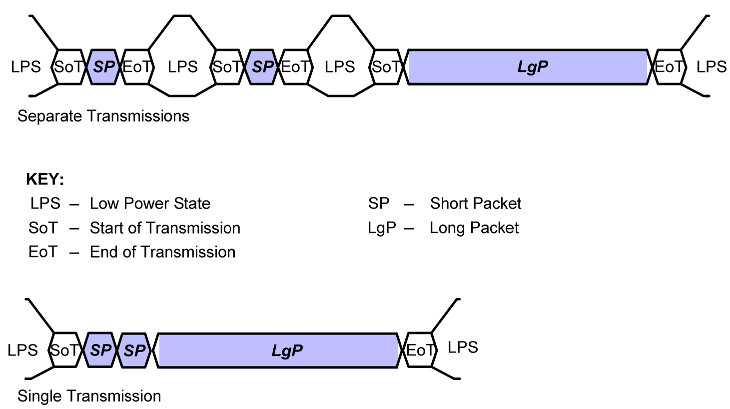


Figure 9 Multiple Packet HS Transmission Example

8.2 Packet Composition

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

Packet sizes fall into two categories:

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

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

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

8.3 Endian Policy

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

8.4 General Packet Structure

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

8.4.1 Long Packet Format

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

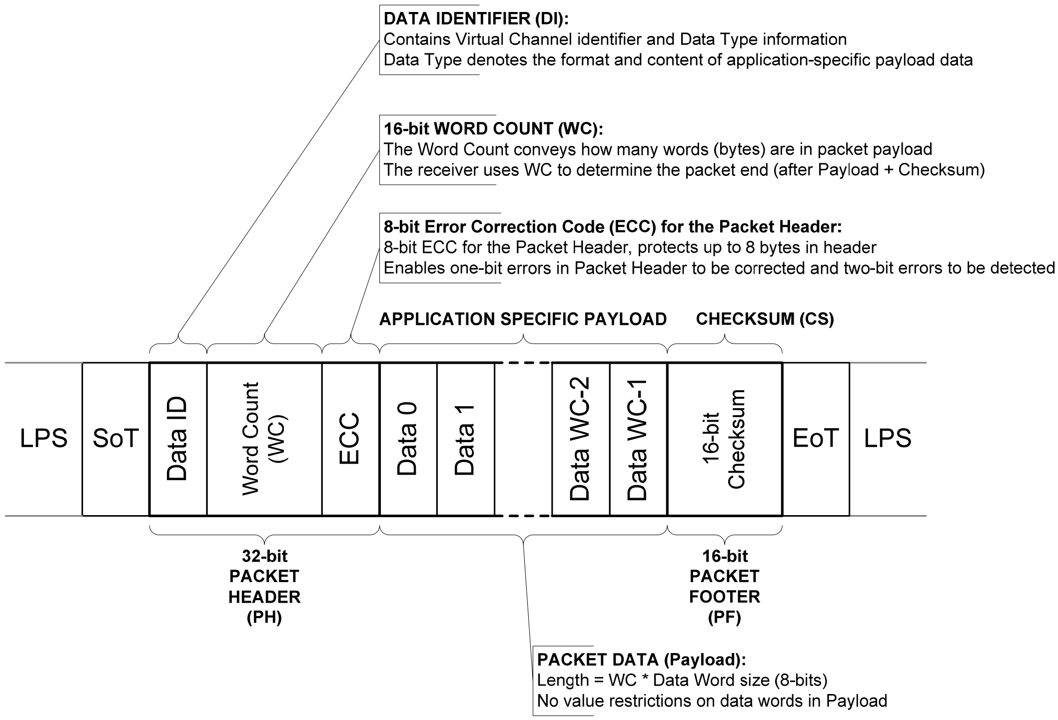


Figure 10 Long Packet Structure

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

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

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

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

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

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

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

8.4.2 Short Packet Format

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

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

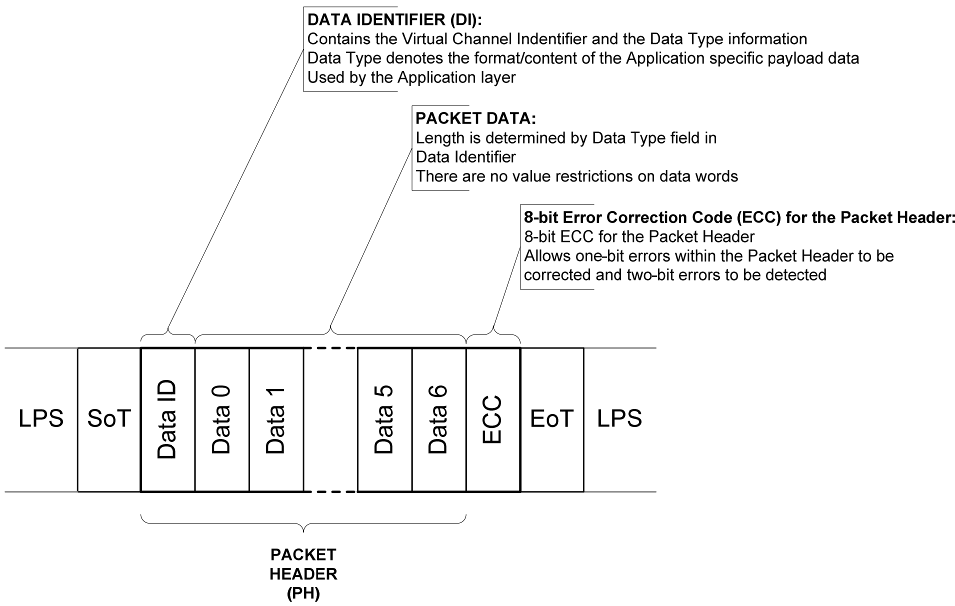


Figure 11 Short packet Structure

8.5 Common Packet Elements

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

8.5.1 Data Identifier Byte

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

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

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

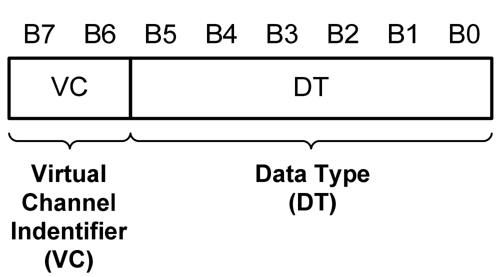


Figure 12 Data Identifier Byte

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

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

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

8.5.1.2 Data Type Field DT[5:0]

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

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

8.5.2 Error Correction Code

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

8.6 Interleaved Data Streams

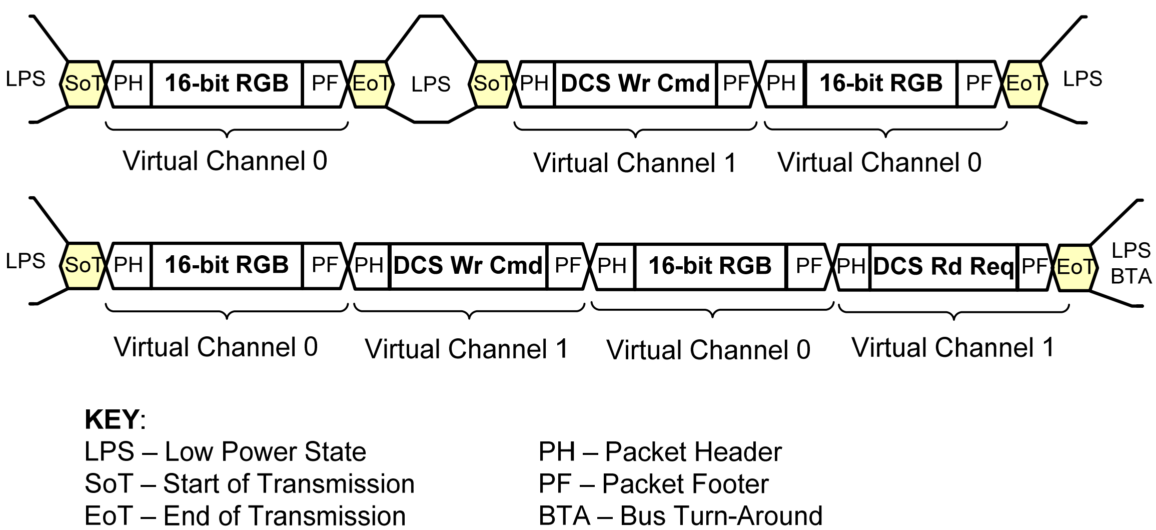


Figure 13 Interleaved Data Stream Example

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

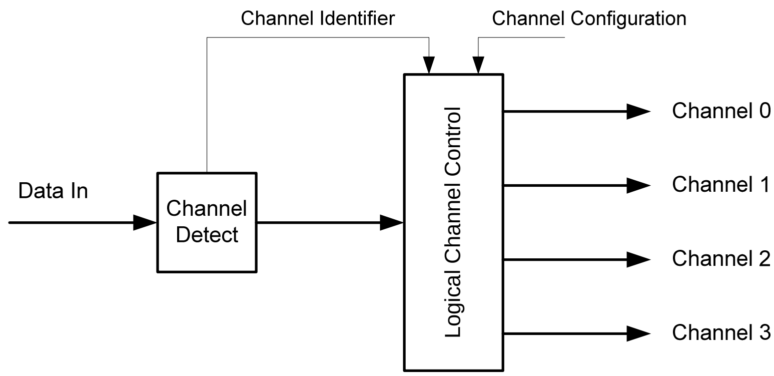


Figure 14 Logical Channel Block Diagram (Receiver Case)

8.6.1 Interleaved Data Streams and Bi-directionality

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

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

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

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

Table 16 Data Types for Processor-sourced Packets

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

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

8.8 Processor-to-Peripheral Transactions – Detailed Format Description

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

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

- Data Type = 00 0001 (01h) V Sync Start
- Data Type = 01 0001 (11h) V Sync End
- Data Type = 10 0001 (21h) H Sync Start
- Data Type = 11 0001 (31h) H Sync End

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

8.8.8 DCS Commands

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

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

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

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

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

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

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

The read response may be more than one packet in the case of DCS Long Read Response, if the returning block of data is larger than the maximum return packet size specified. In that case, the host processor shall send multiple DCS Read Request commands to transfer the complete data block. See section 8.8.8.3 for details on setting the read packet size.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

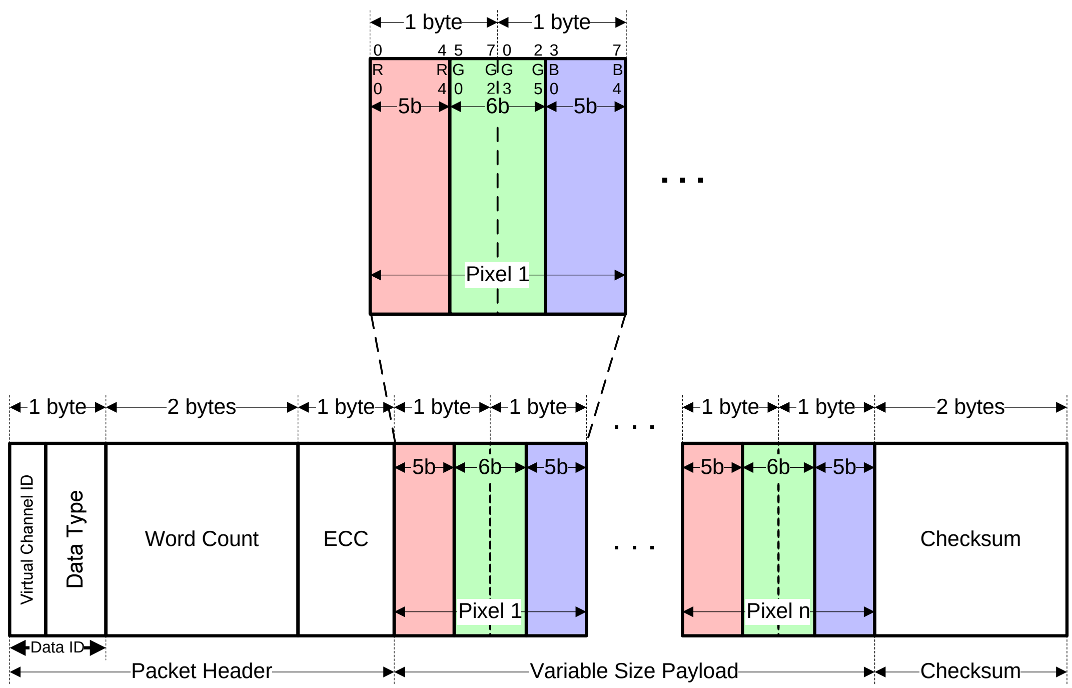


Figure 15: 16-bit per Pixel – RGB Color Format, Long packet

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

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

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

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

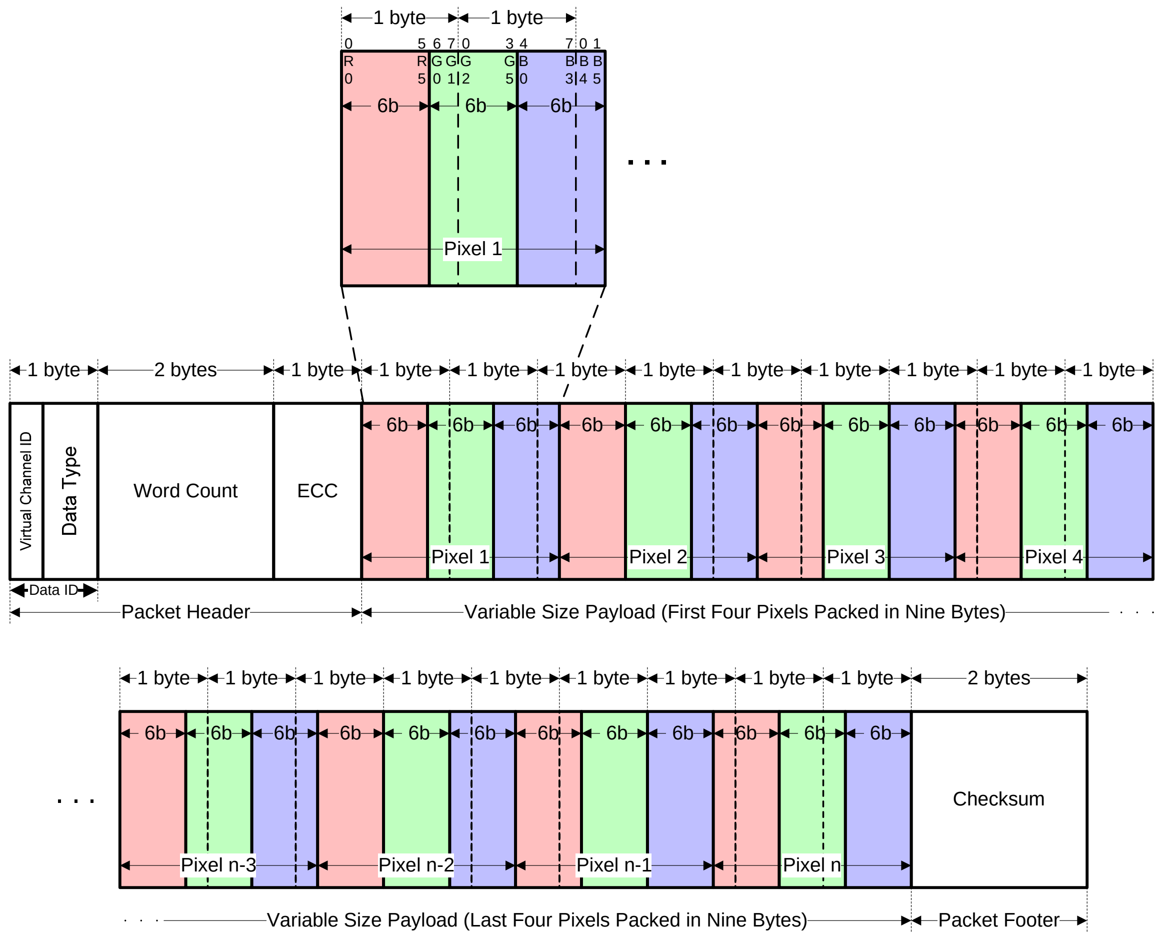


Figure 16: 18-bit per Pixel (Packed) – RGB Color Format, Long packet

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

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

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

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

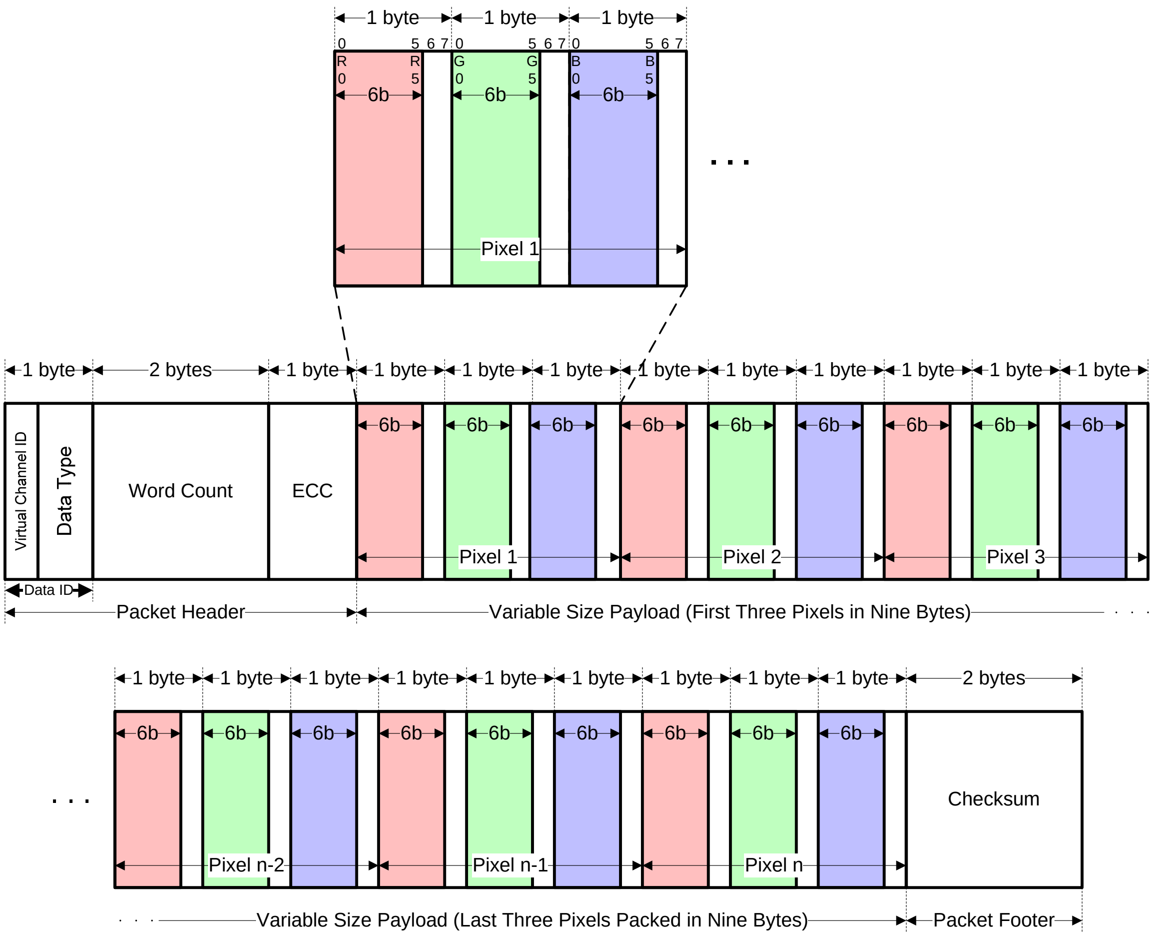


Figure 17: 18-bit per Pixel (Loosely Packed) – RGB Color Format, Long packet

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

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

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

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

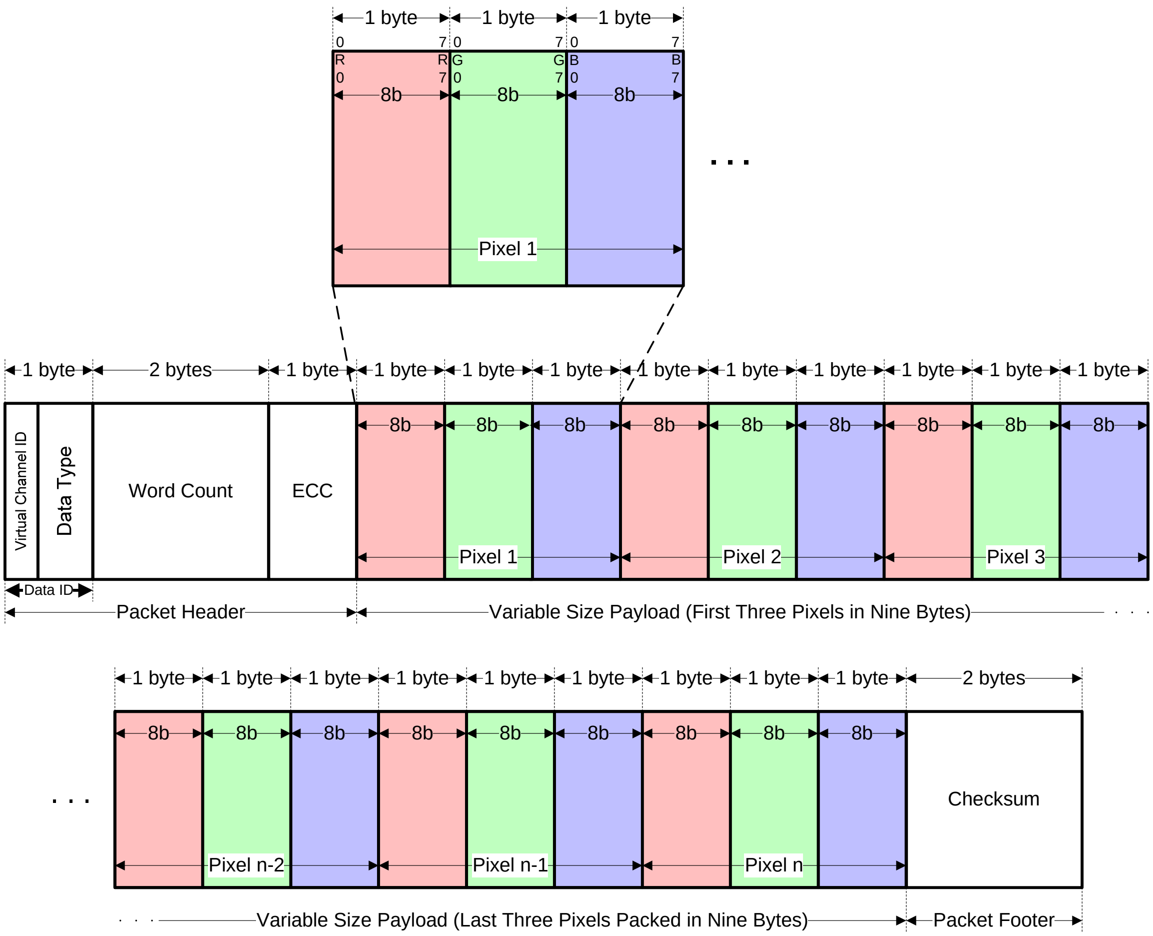


Figure 18: 24-bit per Pixel – RGB Color Format, Long packet

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

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

8.8.17 DO NOT USE and Reserved Data Types

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

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

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

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

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

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

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

8.9.1 Packet Structure for Peripheral-to-Processor LP Transmissions

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

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

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

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

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

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

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

8.9.2 System Requirements for ECC and Checksum and Packet Format

A peripheral may optionally implement ECC, checksum or both.

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

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

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

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

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

8.9.3 Appropriate Responses to Commands and ACK Requests

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

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

- Following a non-Read command in which no error was detected, the peripheral shall respond with *Acknowledge*.
- Following a Read request in which no error was detected, the peripheral shall send the requested READ data.
- Following a Read request in which a single-bit ECC error was detected and corrected, the peripheral shall send the requested READ data in a Long or Short packet, followed by a 4-byte *Acknowledge with Error Report* packet in the same LP transmission. The Error Report shall have the ECC Error – Single Bit flag set.
- Following a non-Read command in which a single-bit ECC error was detected and corrected, the peripheral shall proceed to execute the command, and shall respond to BTA by sending a 4-byte *Acknowledge with Error Report* packet. The Error Report shall have the ECC Error – Single Bit flag set.
- Following a Read request in which multi-bit ECC errors were detected and not corrected, the peripheral shall send a 4-byte *Acknowledge with Error Report* packet without sending Read data. The Error Report shall have the ECC Error – Multi-Bit flag set.
- Following a non-Read command in which multi-bit ECC errors were detected and not corrected, the peripheral shall not execute the command, and shall send a 4-byte *Acknowledge with Error Report* packet. The Error Report shall have the ECC Error – Multi-Bit flag set.
- Following any command in which *SoT Error*, *SoT Sync Error* or *DSI VC ID Invalid* was detected, or the DSI command was not recognized, the peripheral shall send a 4-byte *Acknowledge with Error Report* response, with the appropriate error flags set in the two-byte error field. Only the ACK/Error Report packet shall be transmitted; no read or write accesses shall take place on the peripheral in response.
- Following any command in which *EoT Sync Error* or *LP Transmit Sync Error* is detected, or a checksum error is detected in the payload, the peripheral shall send a 4-byte *Acknowledge with Error Report* packet with the appropriate error flags set.

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

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

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

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

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

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

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

The format for long READ packet responses is:

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

8.9.5 Error-Reporting Format

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

1226 **Table 17 Error Report Bit Definitions**

Bit	Description
0	SoT Error
1	SoT Sync Error
2	EoT Sync Error
3	Escape Mode Entry Command Error
4	Low-Power Transmit Sync Error
5	HS Receive Timeout Error
6	reserved
7	reserved
8	ECC Error, single-bit (detected and corrected)
9	ECC Error, multi-bit (detected, not corrected)
10	Checksum Error (long packet only)
11	DSI Data Type Not Recognized
12	DSI VC ID Invalid
13	reserved
14	reserved
15	reserved

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

8.10.6 Multiple-packet Transmission and Error Reporting

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

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

8.10.7 Clearing Error Bits

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

8.11 Video Mode Interface Timing

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

8.11.1 Traffic Sequences

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

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

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

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

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

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

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

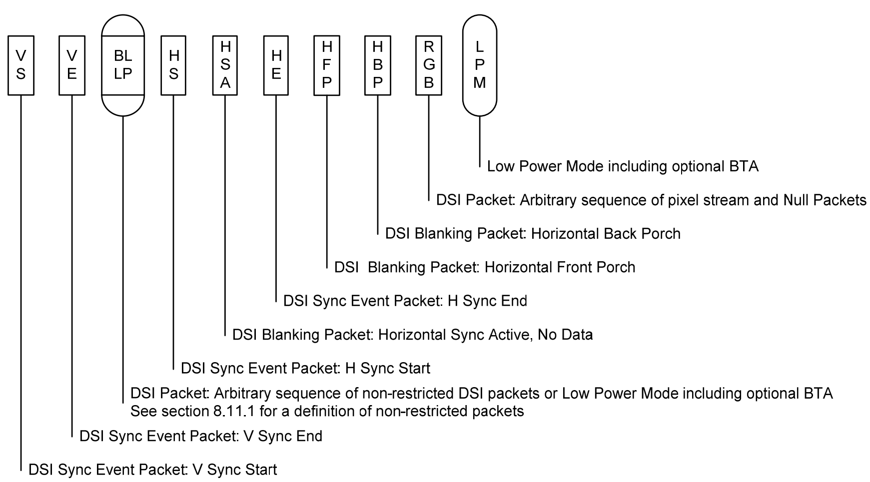


Figure 19 DSI Video Mode Interface Timing Legend

8.11.2 Non-Burst Mode with Sync Pulses

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

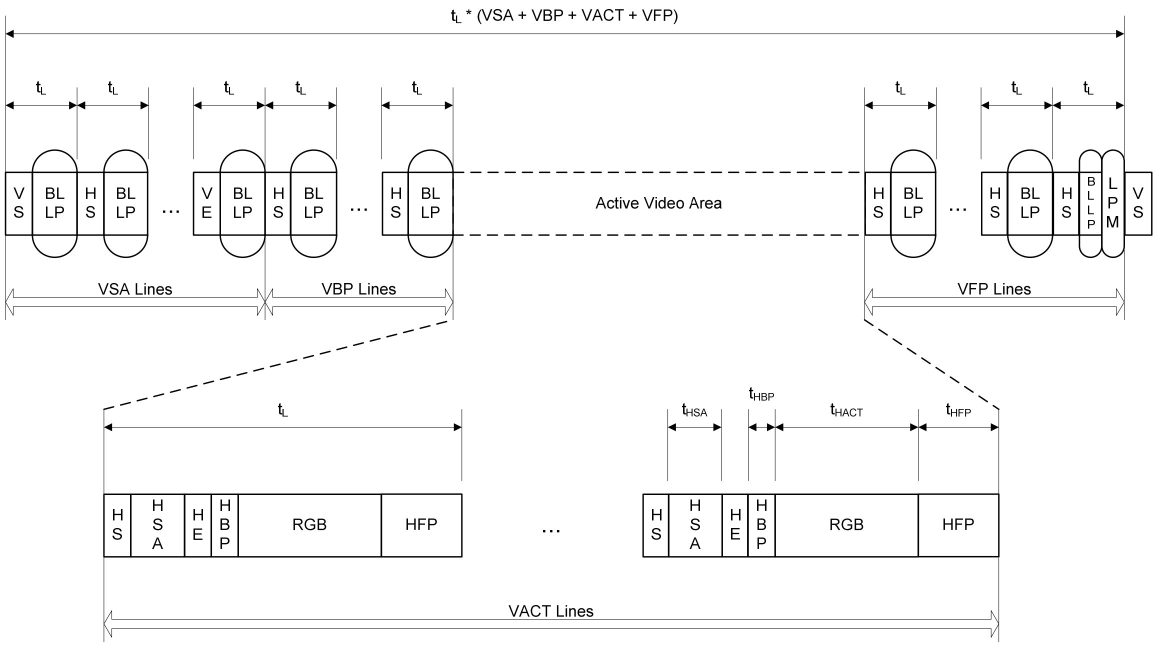


Figure 20 DSI Video Mode Interface Timing: Non-burst Communication with Start and End

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 21.

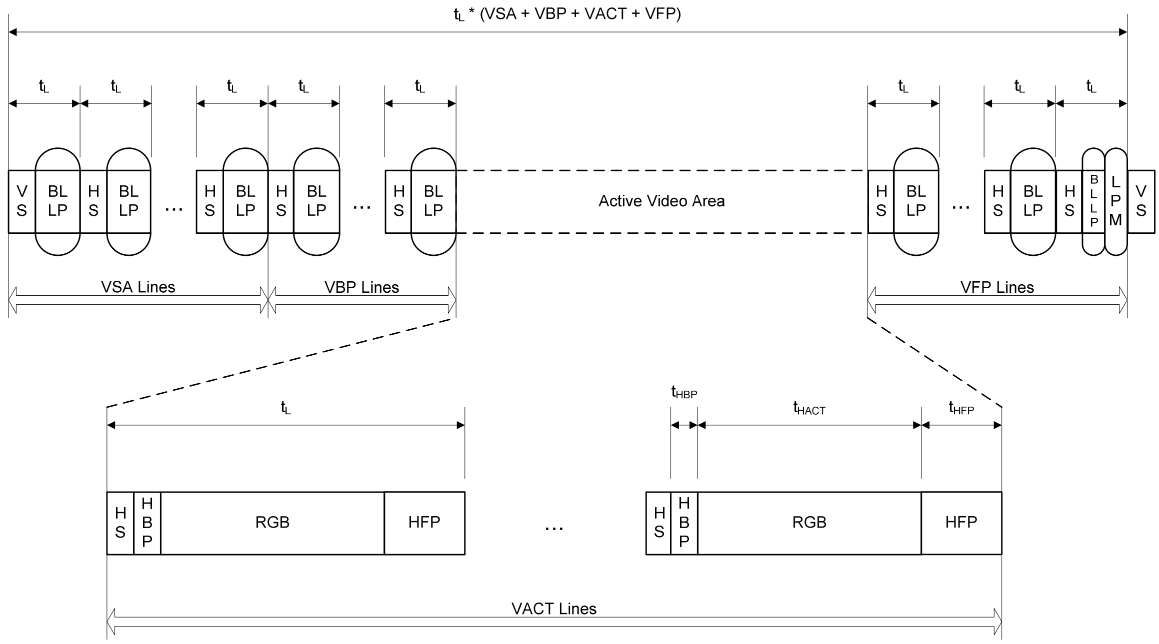


Figure 21 DSI Video Mode Interface Timing: Non-burst Communication

8.11.4 Burst Mode

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

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

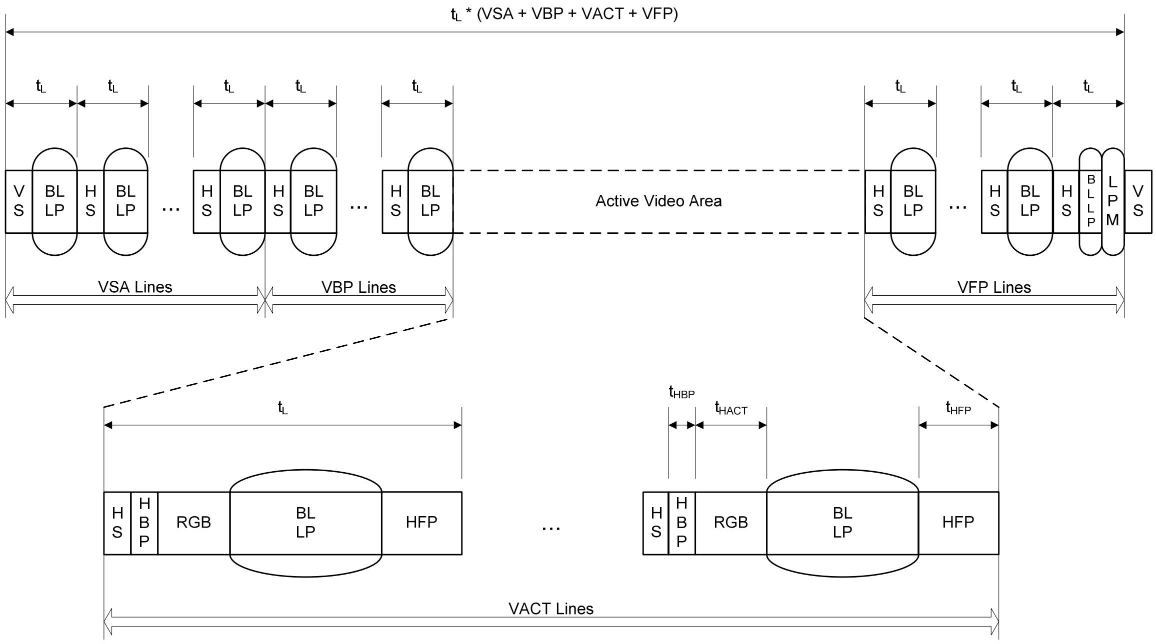


Figure 22 DSI Video Mode Interface Timing: Burst Communication

8.11.5 Parameters

Table 19 documents the parameters used in the preceding figures. Peripheral supplier companies are responsible for specifying suitable values for all blank fields in the table. The host processor shall meet these requirements to ensure interoperability.

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

Table 19 Required Peripheral Timing Parameters

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

Parameter	Description	Minimum	Maximum	Units	Comment
VSA	Vertical sync active			lines	Number of lines in the vertical sync area
VBP	Vertical back porch			lines	
VACT	Active lines per frame			lines	
VFP	Vertical front porch			lines	

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_at_line_on, and set_tear_off. See *MIPI Alliance Standard for Display Command Set* [1] for details.

set_tear_on and set_tear_at_line_on are sent to the display module as DSI Data Type 19h (DCS Short Write, one parameter and two parameters, respectively) along with the set_tear_on or set_tear_at_line_on command byte. 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)

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

This Escape Mode sequence is reserved by DSI for TE signaling only and shall not be used for any other purpose in a DSI-compliant interface.

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

9 Error-Correcting Code (ECC) and Checksum

9.1 Hamming Code for Packet Header Error Detection/Correction

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

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

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

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

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

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

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

$$\mathbf{G} = [\ \mathbf{I}\ |\ \mathbf{A}\ ]$$

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

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

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

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

9.2 Hamming-modified Code for DSI

For DSI, the error correcting code used is a 7+1 bits Hamming-modified code (72, 64). This class of Hamming code can correct a single-bit error or detect a two-bit error, but is not capable of doing both simultaneously, so one extra parity bit is added. The code used, is built to allow same syndromes to correct first 24-bits in a 64-bit sequence and those syndromes to be 6-bits wide. To specify in a compact way the encoding of parity and decoding of syndromes, the following matrix is used:

Table 20 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

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

e.g. 0x07=0b0000_0111=P7P6P5P4P3P2P1P0

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

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

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

$S=P_{SEND}$^$P_{RECEIVED}$ where P_{SEND} is the 8-bit ECC field in the header and $P_{RECEIVED}$ is the calculated parity of the received header.

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

Table 21 ECC Parity Generation Rules

Bit	P7	P6	P5	P4	P3	P2	P1	P0	Hex
0	0	0	0	0	0	1	1	1	0x07
1	0	0	0	0	1	0	1	1	0x0B
2	0	0	0	0	1	1	0	1	0x0D
3	0	0	0	0	1	1	1	0	0x0E

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

Bit	P7	P6	P5	P4	P3	P2	P1	P0	Hex
31	0	1	0	1	0	0	1	0	0x52
32	0	1	0	1	0	1	0	0	0x54
33	0	1	0	1	1	0	0	0	0x58
34	0	1	1	0	0	0	0	1	0x61
35	0	1	1	0	0	0	1	0	0x62
36	0	1	1	0	0	1	0	0	0x64
37	0	1	1	0	1	0	0	0	0x68
38	0	1	1	1	0	0	0	0	0x70
39	1	0	0	0	0	0	1	1	0x83
40	1	0	0	0	0	1	0	1	0x85
41	1	0	0	0	0	1	1	0	0x86
42	1	0	0	0	1	0	0	1	0x89
43	1	0	0	0	1	0	1	0	0x8A
44	0	0	1	1	1	1	0	1	0x3D
45	0	0	1	1	1	1	1	0	0x3E
46	0	1	0	0	1	1	1	1	0x4F
47	0	1	0	1	0	1	1	1	0x57
48	1	0	0	0	1	1	0	0	0x8C
49	1	0	0	1	0	0	0	1	0x91
50	1	0	0	1	0	0	1	0	0x92
51	1	0	0	1	0	1	0	0	0x94
52	1	0	0	1	1	0	0	0	0x98
53	1	0	1	0	0	0	0	1	0xA1
54	1	0	1	0	0	0	1	0	0xA2
55	1	0	1	0	0	1	0	0	0xA4
56	1	0	1	0	1	0	0	0	0xA8
57	1	0	1	1	0	0	0	0	0xB0

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

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

P7 = D39^D40^D41^D42^D43^D48^D49^D50^D51^D52^D53^D54^D55^D56^D57^D58^D59 ^D60^D61^D62^D63

9.3 ECC Generation on the Transmitter and Byte-Padding

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

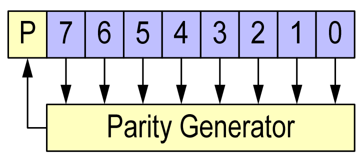


Figure 23 64-bit ECC generation on TX side

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

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

9.4 Applying ECC and Byte-Padding on the Receiver

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

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

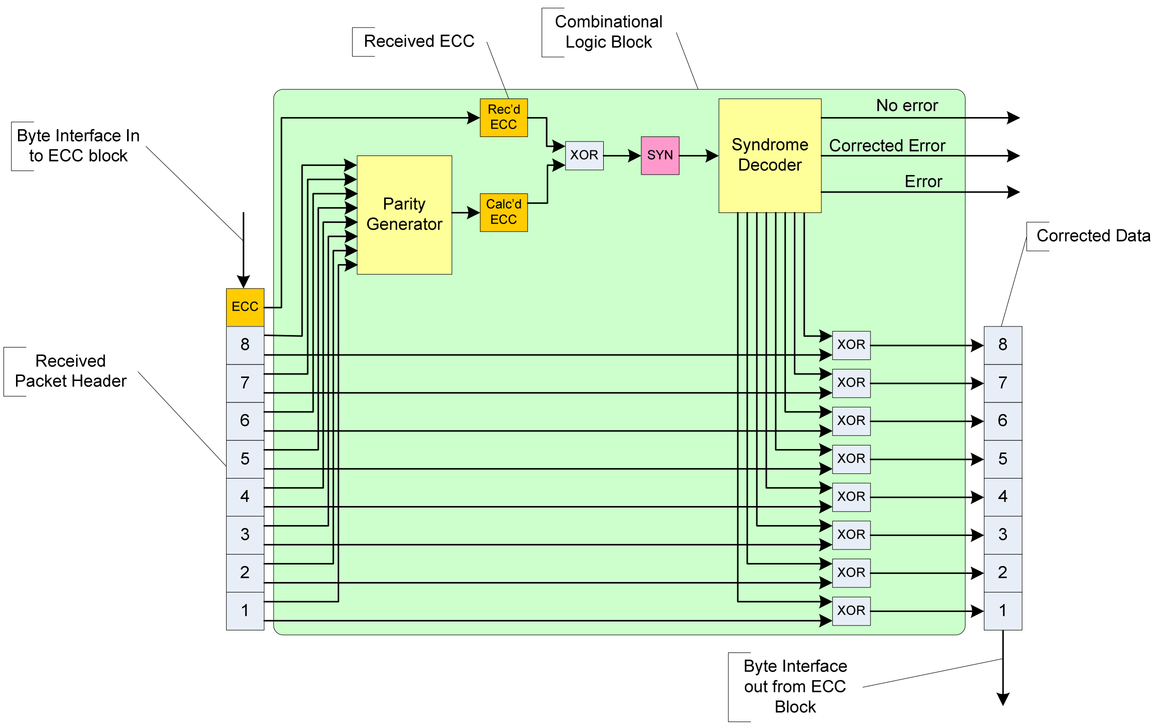


Figure 24 64-bit ECC on RX Side Including Error Correction

Decoding the syndrome has three aspects:

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

9.5 Checksum Generation for Long Packet Payloads

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

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

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

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

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

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

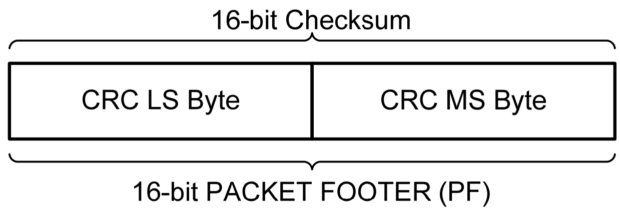


Figure 25 Checksum Transmission

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

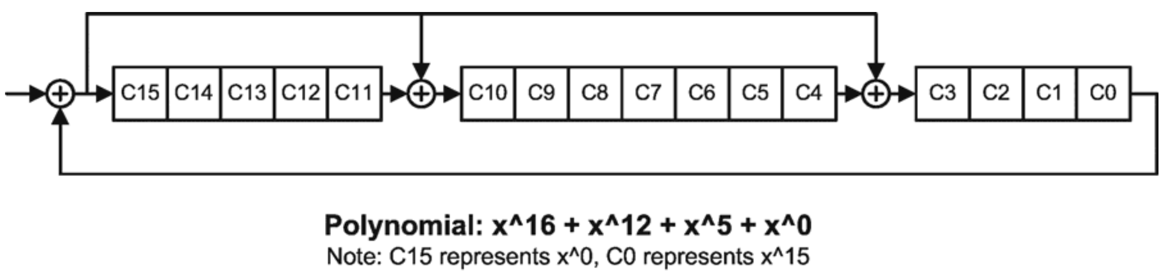


Figure 26 Example implementation of CCITT 16-bit CRC generation using shift register

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

10 Compliance, Interoperability, and Optional Capabilities

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

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

1. Display Resolutions
2. Pixel Formats
3. Number of Lanes
4. Maximum Lane Frequency
5. Bidirectional Communication and Escape Mode Support
6. ECC and Checksum capabilities
7. Display Architecture
8. Multiple Peripheral Support

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

10.1 Display Resolutions

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

Table 22 Display Resolutions

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

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

10.2 Pixel Formats

Peripherals shall implement 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 8.8.13
2. 18 bpp (6, 6, 6 RGB) packed; see section 8.8.14
3. 18 bpp (6, 6, 6 RGB) loosely packed into three bytes; see section 8.8.15
4. 24 bpp (8, 8, 8 RGB), each pixel using three bytes; see section 8.8.16

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 Standard for D-PHY* [4].

10.5 Bidirectional Communication

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

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

10.6 ECC and Checksum Capabilities

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

10.7 Display Architecture

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

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

10.8 Multiple Peripheral Support

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

A host processor should support a minimum of two peripherals.

Annex A (Informative) Contention Detection and Recovery Mechanisms

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

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

A.1 PHY Detected Contention

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

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

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

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

A.1.1 Protocol Response to PHY Detected Faults

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

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

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

Table sequences:

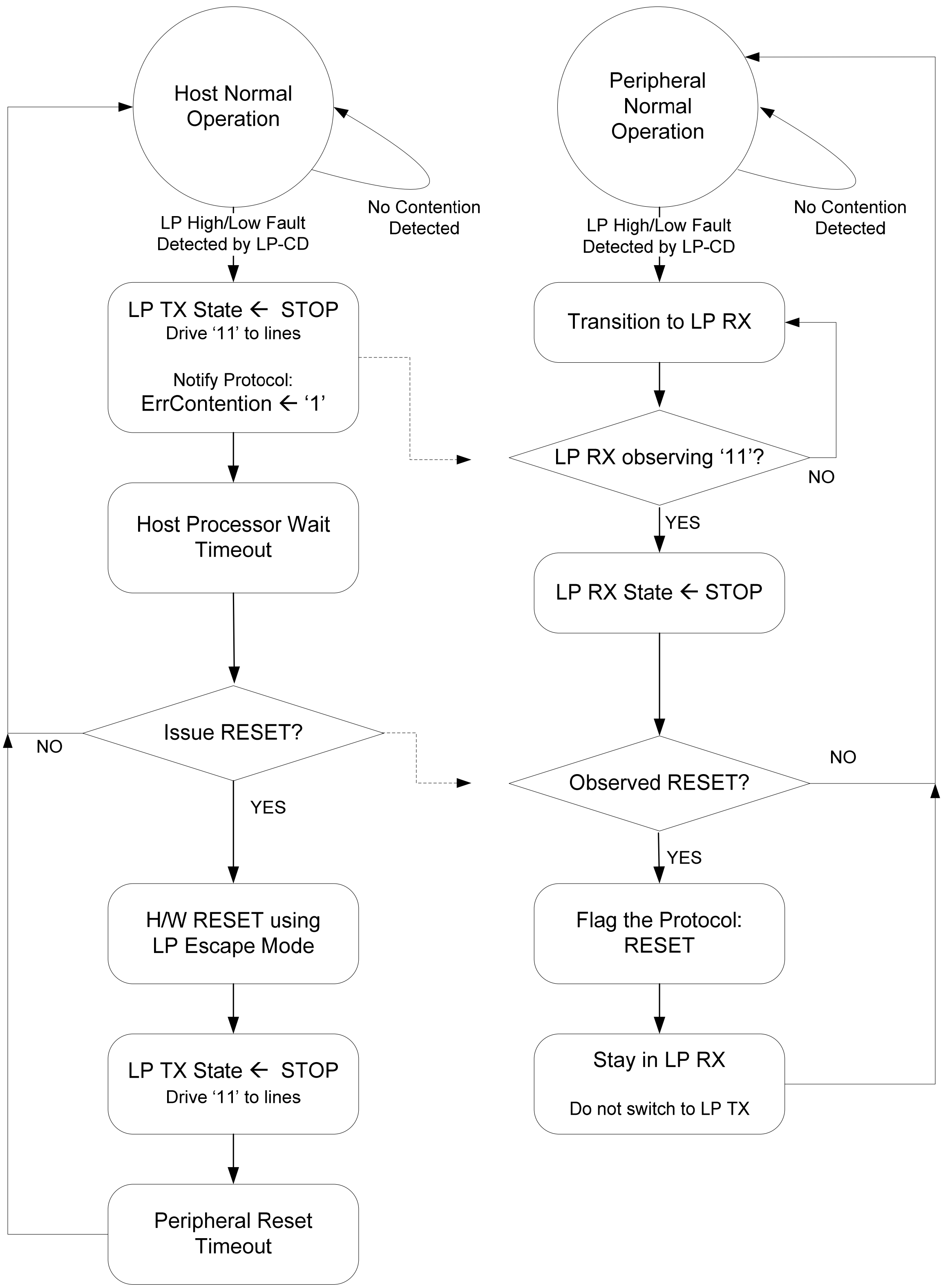
- Sequence of events to resolve LP High ←→ LP Low Contention
    - Case 1: Both sides initially detect the contention
    - Case 2: Only the Host Processor initially detects contention
    - Case 3: Only the Peripheral initially detects contention

Table 23 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*	

Host Processor Side		Peripheral Side	
Protocol	**PHY**	**PHY**	**Protocol**
	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	
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

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

1610 

1611 **Figure 27 LP High ←→ LP Low Contention Case 1**

Table 24 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

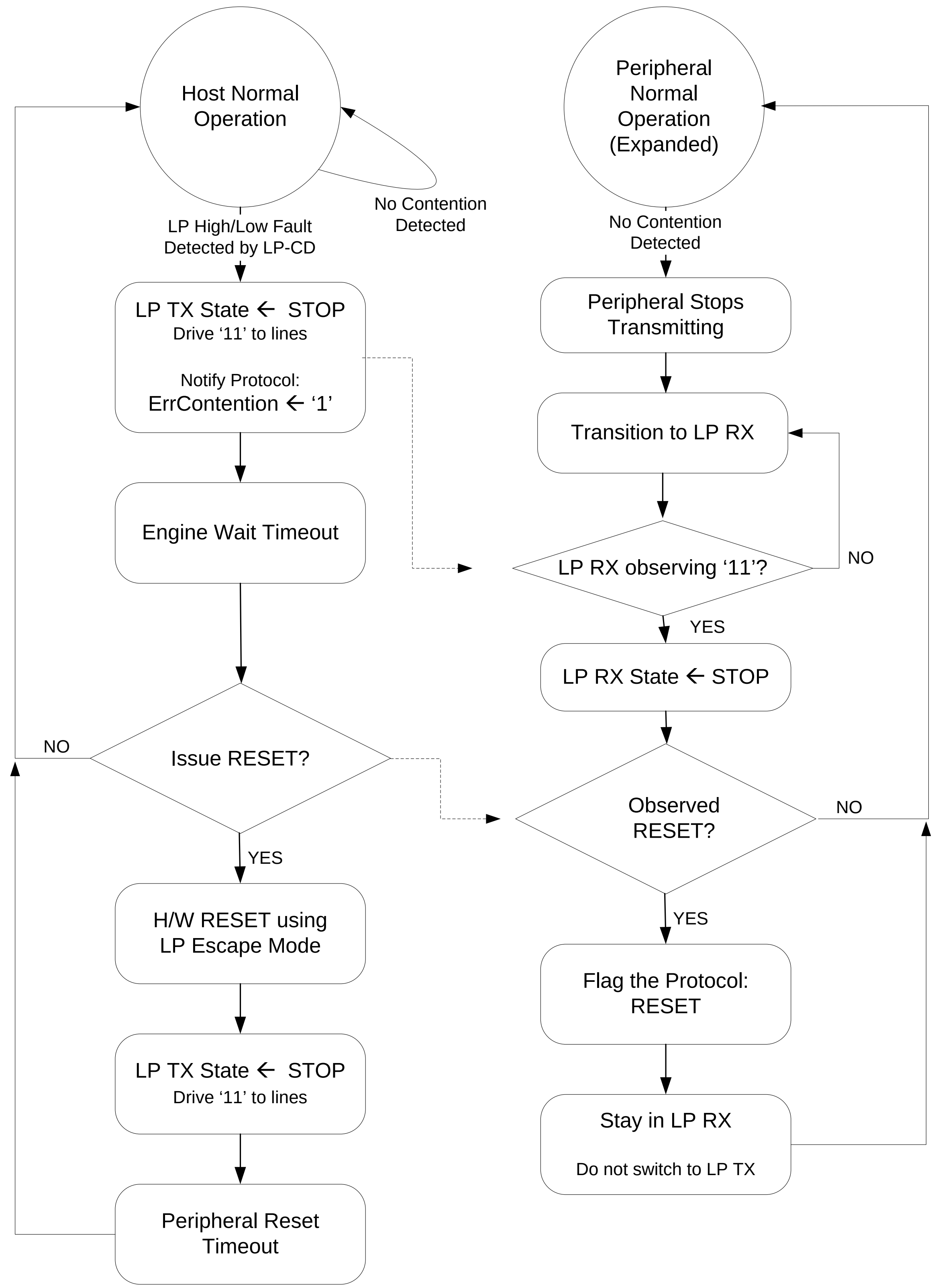


Figure 28 LP High ←→ LP Low Contention Case 2

Table 25 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	

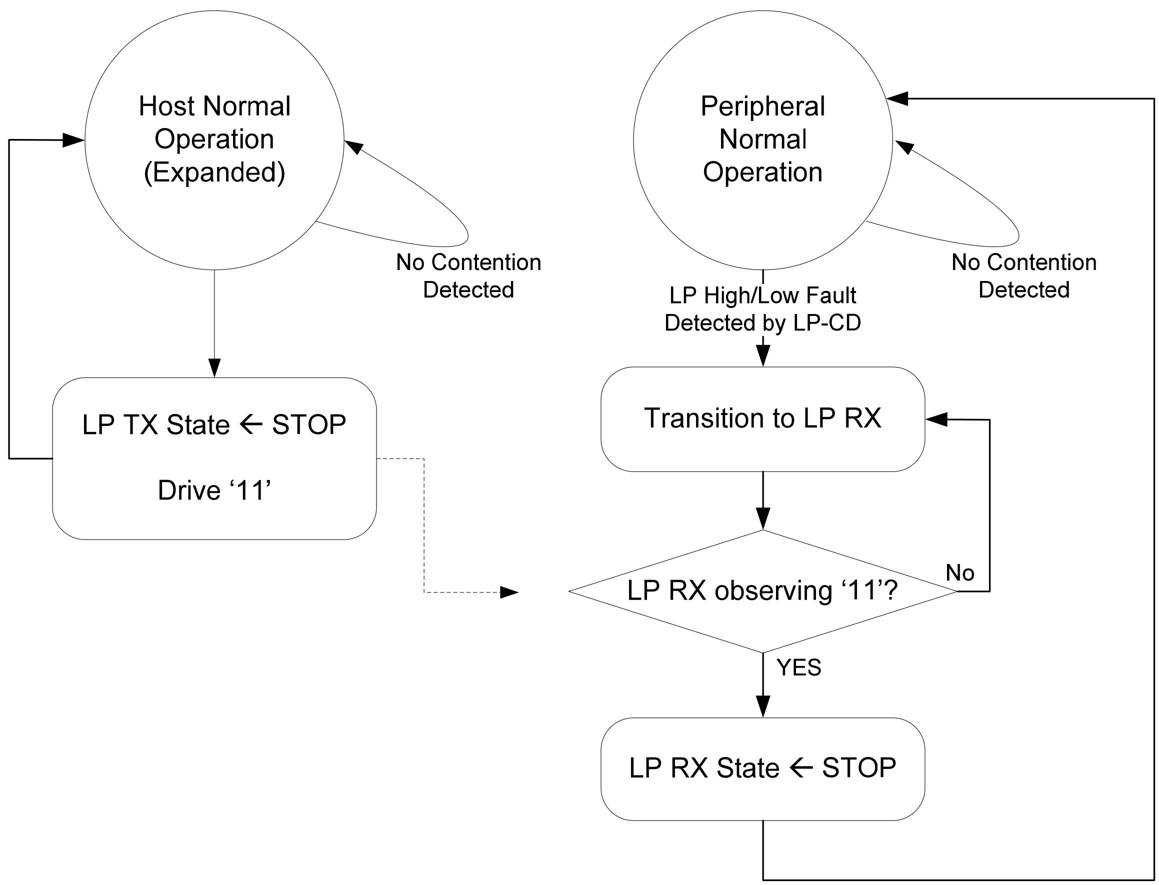


Figure 29 LP High ←→ LP Low Contention Case 3