Specification for Display Serial Interface (DSI^SM^)

Version 1.3

23 March 2015

MIPI Board Adopted 10 March 2015

Corrections Approved 23 March 2015

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

Contents

Figures

Tables

Release History

Date	Release	Description
2006-04-19	v1.00a	Initial MIPI Alliance Board-approved release.
2008-02-21	v1.01.00	Major update including editorial updates and terminology corrections. Added Section 5.7 “System Power-up and Initialization”, packets for EoT, Short Write, DCS Write, Generic Reads of different lengths, Generic Long Write, Color Mode Command, ECC requirement.
2010-06-28	v1.02.00	Minor update adding methods to carry 10-bit, 12-bit per component color content and clarifications for virtual channel usage by interlaced video, power-up initialization, updated reportable errors and included several editorial changes.
2012-04-06	v1.1	Board-approved Release. Added support for Stereoscopic Display Formats.
2014-06-08	v1.2	Board-approved Release. Added support for display stream compression and support for multiple DSI Link receiver synchronization.
2015-03-23	v1.3	Board-approved Release. Added support for Sub Links (Split Link), Deskew, and Checksum for Test Mode. Post-Adoption typo Corrections.

This page intentionally left blank.

1 Overview

The Display Serial Interface 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 specifications by adopting pixel formats and command set defined in [MIPI02], [MIPI03], and [MIPI01].

1.1 Scope

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

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 document. 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 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 (informative)

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

> The word *shall* is used to indicate mandatory requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted (*shall* equals *is required to*).
>
> The use of the word *must* is deprecated and shall not be used when stating mandatory requirements; must is used only to describe unavoidable situations.
>
> The use of the word *will* is deprecated and shall not be used when stating mandatory requirements; *will* is only used in statements of fact.
>
> The word *should* is used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain course of action is deprecated but not prohibited (*should* equals *is recommended that*).
>
> The word *may* is used to indicate a course of action permissible within the limits of the standard (*may* equals *is permitted*).
>
> The word *can* is used for statements of possibility and capability, whether material, physical, or causal (*can* equals *is able to*).

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

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

2.1 Definitions

Bitstream: The sequence of data bytes resulting from the coding of image data. The bit stream does not contain a header or syntax markers.

Codestream: A sequence of data bytes composed of a bitstream and any header and syntax markers necessary for decoding. The codestream boundary usually coincides with a frame boundary, but does not need to do so.

Command Queue: A queue that contains one or more frame synchronized commands in each display driver in common within a display module.

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.

Frame-based: The data transfer mode that sends an entire left or right view followed by the corresponding right or left view**,** respectively.

Frame Synchronized Command: A command, data type transaction or register write that is synchronized across multiple DSI link instantiations that connect to different display drivers contained in one display module.

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.

Landscape/Portrait Orientation: The orientation the display is viewed by a user.

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

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

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

Line-based: The data transfer mode that sends an entire left or right line followed by the corresponding right or left line**,** respectively.

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

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 four 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.

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.

Picture Parameter Set: A number of bytes defined by the coding system to program coding-dependent parameters. The encoder and decoder shall have the same picture parameter set in order to create and interpret the codestream.

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

Slice: A set of compressed bits that represents a specified set of samples. The set of samples forms a rectangle in the horizontal and vertical dimensions. This set of bits is independently decodable. Decoding of any one slice does not depend on the availability of another slice or on the decoded result of another slice.

Stereoscopic Image: A pair of offset images of a scene (views) that renders content to both the left eye and right eye to produce the perception of depth.

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 within the payload.

2.2 Abbreviations

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

2.3 Acronyms

AIP	Application Independent Protocol
AM	Active matrix (display technology)
ASP	Application Specific Protocol
BLLP	Blanking or Low Power interval
BPP	Bits Per Pixel
BTA	Bus Turn-Around
CBR	Constant Bit Rate
CSI	Camera Serial Interface
DBI	Display Bus Interface
DI	Data Identifier
DMA	Direct Memory Access
DPI	Display Pixel Interface
DSC	Display Stream Compression; also used in context with the Display Stream Compression (DSC) Standard, *[VESA01]*
DSI	Display Serial Interface
DT	Data Type
ECC	Error-Correcting Code
EMI	Electro Magnetic interference
EoTp	End of Transmission Packet

ESD	Electrostatic Discharge
FSC	Frame Synchronized Command or Commands
Fps	Frames per second
HBP	Horizontal Back Porch
HFP	Horizontal Front Porch
HS	High Speed
HSA	Horizontal Sync Active
HSE	Horizontal Sync End
HSS	Horizontal Sync Start
ISTO	Industry Standards and Technology Organization
LP	Low Power
LPS	Low Power State (state of serial data line when not transferring high-speed serial data)
LSB	Least Significant Bit
Mbps	Megabits per second
MSB	Most Significant Bit
PF	Packet Footer
PH	Packet Header
PHY	Physical Layer
PPI	PHY-Protocol Interface
PPS	Picture Parameter Set
QCIF	Quarter-size CIF (resolution 176x144 pixels or 144x176 pixels)
QVGA	Quarter-size Video Graphics Array (resolution 320x240 pixels or 240x320 pixels)
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)
ULPS	Ultra-low Power State
VBR	Variable Bit Rate

VLC	Variable Length Coding
VGA	Video Graphics Array (resolution 640x480 pixels or 480x640 pixels)
VSA	Vertical Sync Active
VSE	Vertical Sync End
VSS	Vertical Sync Start
WC	Word Count
WVGA	Wide VGA (resolution 800x480 pixels or 480x800 pixels)

3 References

[MIPI01] *MIPI Alliance Specification for Display Command Set*, version 1.3, MIPI Alliance, Inc., In Press. (Informative)

[MIPI02] *MIPI Alliance Standard for Display Bus Interface (DBI-2)*, version 2.00, MIPI Alliance, Inc., 29 November 2005. (Informative)

[MIPI03] *MIPI Alliance Standard for Display Pixel Interface (DPI-2)*, version 2.00, MIPI Alliance, Inc., 15 September 2005. (Informative)

[MIPI04] *MIPI Alliance Specification for D-PHY*, version 1.2, MIPI Alliance, Inc.,10 September 2014.

[MIPI05] *MIPI Alliance Specification for Stereoscopic Display Formats (SDF)*, version 1.0, MIPI Alliance, Inc., 14 March 2012. (Informative)

[CEA01] CEA-861-E, *A DTV Profile for Uncompressed High Speed Digital Interfaces*, <http://www.ce.org/Standards/browseByCommittee_2641.asp>, Consumer Electronics Association, March 2008. (Informative)

[ITU01] BT.601-6, *Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios*, <http://www.itu.int/rec/R-REC-BT.601-6-200701-I/en>, International Telecommunications Union, 23 February 2007. (Informative)

[ITU02] BT.709-5, *Parameter values for the HDTV standards for production and international programme exchange*, <http://www.itu.int/rec/R-REC-BT.709-5-200204-I/en>, International Telecommunications Union, 27 August 2009. (Informative)

[ITU03] BT.656-5, *Interface for digital component video signals in 525-line and 625-line television systems operating at the 4:2:2 level of Recommendation ITU-R BT.601*, <http://www.itu.int/rec/R-REC-BT.656-5-200712-I/en>, International Telecommunications Union, 1 January 2008. (Informative)

[SMPT01] EG 36-2000, *Transformations Between Television Component Color Signals*, Society for Motion Picture and Television Engineers, 23 March 2000. (Informative)

[VESA01] *Display Stream Compression Standard*, v 1.1. Published by VESA www.vesa.org, In Press, 2014.

Much of DSI is based on existing MIPI Alliance Specifications as well as several MIPI Alliance specifications in simultaneous development. In the Application Layer, DSI duplicates pixel formats used in [MIPI03] when it is in *Video Mode* operation. For display modules with a display controller and frame buffer, DSI shares a common command set with [MIPI02]. The command set is documented in [MIPI01].

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

DBI-2 is a MIPI Alliance standard for parallel interfaces to display modules having display controllers and frame buffers. For systems based on these standards, 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-2 specifies a parallel interface where data can be sent to the peripheral over an 8-, 9- or 16-bit-wide data bus, with additional control signals. DBI-2 supports a 1-bit data bus interface mode as well.

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 Display Pixel Interface Standard for Parallel Signaling (DPI-2)

DPI-2 is a MIPI Alliance standard 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 standards document several pixel formats for *Active Matrix* (AM) display modules.

Like DBI-2, DPI-2 is a MIPI Alliance standard 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 document 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 MIPI Alliance Specification for Display Command Set (DCS)

DCS is a MIPI Alliance specification for the command set used by DSI and DBI-2 standards. 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 MIPI Alliance Standard for Camera Serial Interface 2 (CSI-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 between DSI and CSI-2 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 MIPI Alliance Specification for D-PHY (D-PHY)

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

3.6 MIPI Alliance Specification for Stereoscopic Display Formats (SDF)

[MIPI05] defines methods to transmit stereoscopic image data between a mobile device host processor and a display peripheral, usually over a high-speed serial link. The nature of mobile devices and how these devices are used lead to various parameters and design options available for a stereoscopic display.

DSI requires the image formats described in [MIPI05] when transmitting stereoscopic image data.

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 specifications 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 has 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.

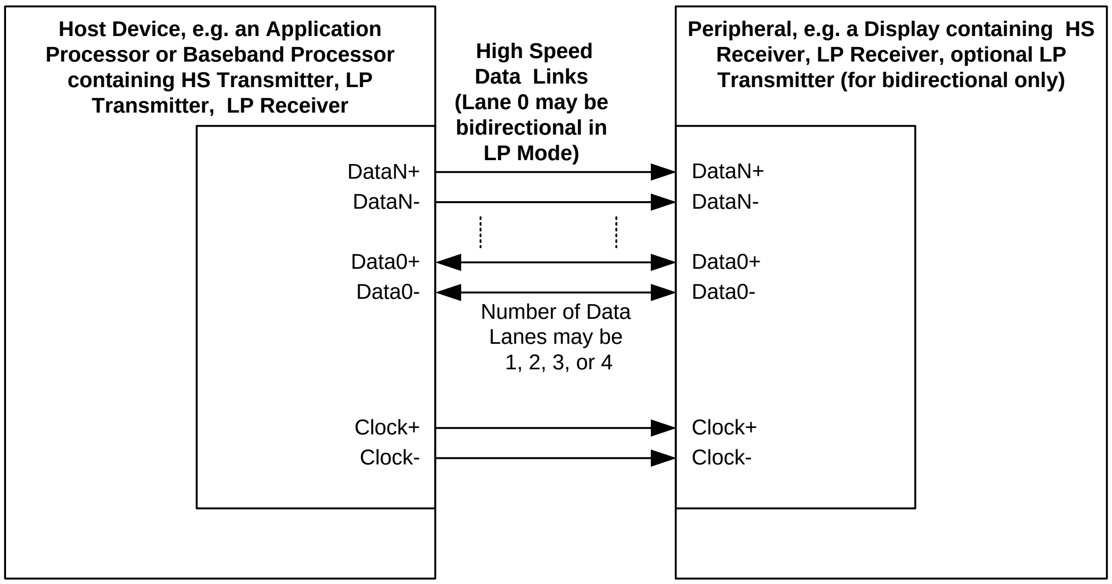

Host Device, e.g. an Application Processor or Baseband Processor containing HS Transmitter, LP Transmitter, LP Receiver
High Speed Data Links (Lane 0 may be bidirectional in LP Mode)
Peripheral, e.g. a Display containing HS Receiver, LP Receiver, optional LP Transmitter (for bidirectional only)
DataN+
DataN-
Data0+
Data0-
Number of Data Lanes may be 1, 2, 3, or 4
Clock+
Clock-
DataN+
DataN-
Data0+
Data0-
Clock+
Clock-

Figure 1 DSI Transmitter and Receiver Interface

4.1 DSI Layer Definitions

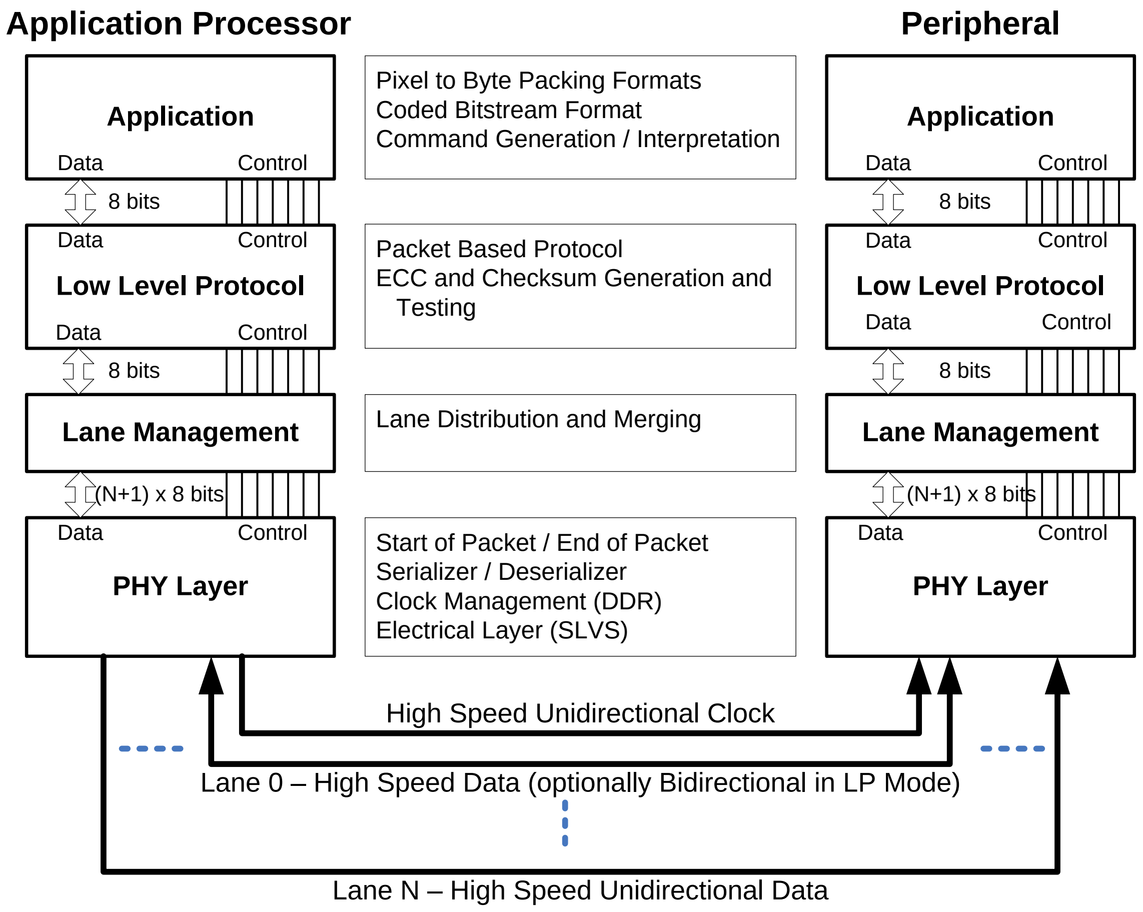

Application Processor
Peripheral
Application
Data
Control
8 bits
Low Level Protocol
Lane Management
(N+1) x 8 bits
PHY Layer
Pixel to Byte Packing Formats
Coded Bitstream Format
Command Generation / Interpretation
Packet Based Protocol
ECC and Checksum Generation and Testing
Lane Distribution and Merging
Start of Packet / End of Packet
Serializer / Deserializer
Clock Management (DDR)
Electrical Layer (SLVS)
High Speed Unidirectional Clock
Lane 0 – High Speed Data (optionally Bidirectional in LP Mode)
Lane N – High Speed Unidirectional Data

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 [MIPI04].

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 or coded bitstreams 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, bitstreams, commands and command parameters to bytes in the packet assembly. See [MIPI01].

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 display modules also include a simplified form of Command Mode operation in which the display module may refresh its screen from a reduced-size, or partial compressed or partial uncompressed 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 compressed or an uncompressed 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. . A bridge device may create multiple, separate connections to display modules or other devices or a display module or device may support multiple virtual channels. 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. For example, 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. Virtual channels may also be employed by devices where one channel is a bidirectional Command Mode channel and the second channel is a Video Mode unidirectional channel. Virtual channels may be employed by a display module or a DSI bridge device receiving interlaced video from the host-processor, where one channel is corresponding to the first field and another channel is the second field of an interlaced video frame. The DSI specification makes no requirements on the specific value assigned to each virtual channel used to designate interlaced fields, For clarity, the first interlaced video field may be assigned as DI[7:6] = 0b00 and the second interlaced video field may be assigned DI[7:6] = 0b01.

5 DSI Physical Layer

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

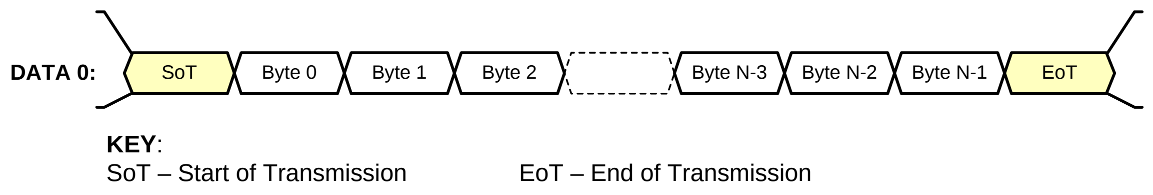


Figure 3 Basic HS Transmission Structure

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

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.

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 Section 5.3 and Section 5.4 for details.

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

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 [MIPI04] 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 tells 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-MFAA (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-SFAA (HS-RX, LP-RX, LP-TX, and LP-CD)
- Clock Lane Module: CIL-SCNN (HS-RX, LP-RX)

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

5.4 Video Mode Interfaces

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

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

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

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 document.

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.

The physical layer [MIPI04] contains the ability to de-skew clock-to-lane timing. Physical layer de-skewing shall be performed in a manner that does not interfere with any display data, as follows:

1. Initial de-skew shall be completed prior to the transmission of DSI packets.
2. Periodic de-skew shall support continuous clock mode operation.
3. Periodic de-skew shall be contained within any one video mode line, and shall meet these additional requirements to support video mode timing defined in Section 8.1:
    - Periodic de-skew shall occur in a video back porch line or a video mode front porch line
    - Periodic de-skew shall not overlap any timing characters in the line, HSA, HSS, HSE, VSS, and VSE.
4. The DSI Transmitter shall not initiate a de-skew operation during the time between when the DSI Transmitter has asserted a TurnRequest and when the host has regained bus control.

In addition, a de-skew operation should not overlap an out-of-band TE event from the DSI receiver, and a de-skew operation should not be initiated until triggered by a TE event, either in-band or out-of-band.

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. The peripheral shall not require additional bits, bytes, or packets from the host processor in order to complete processing or pipeline movement of received data in LP mode transmissions. There are a variety of ways to meet this requirement. For example, the peripheral may generate its own clock or it may require the host processor to keep the HS serial clock running.

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 [MIPI04] 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.

5.7 System Power-Up and Initialization

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

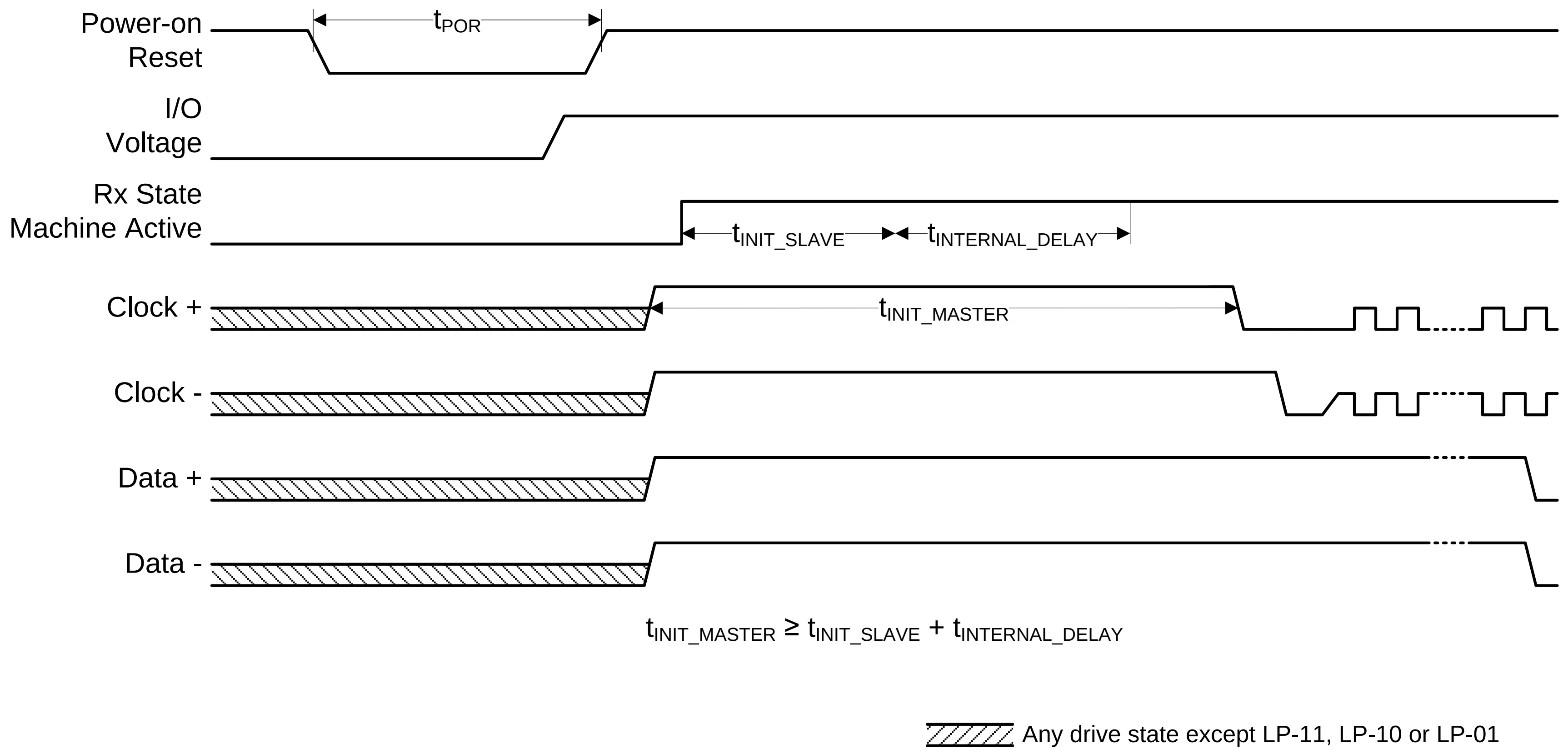
After power-up, the host processor shall observe an initialization period, T_{INIT}, during which it shall drive a sustained TX-Stop state (LP-11) on all Lanes of the Link. See [MIPI04] for descriptions of T_{INIT} and the TX-Stop state.

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

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

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

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



754

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

6 Multi-Lane Distribution and Merging

DSI is a Lane-scalable interface. 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 document explicitly defines 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 5, 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 6, the layer collects incoming bytes from N Lanes and consolidates the bytes into complete packets to pass into the following packet decomposer.

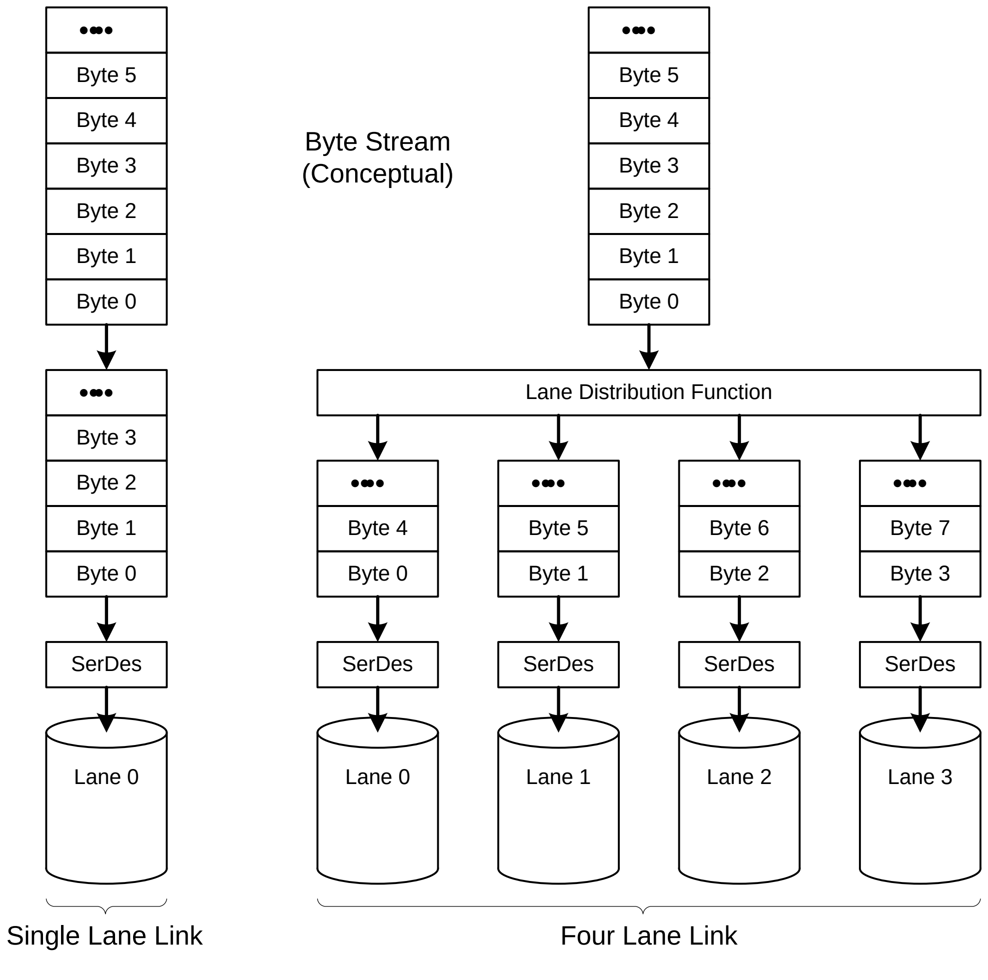


Figure 5 Lane Distributor Conceptual Overview

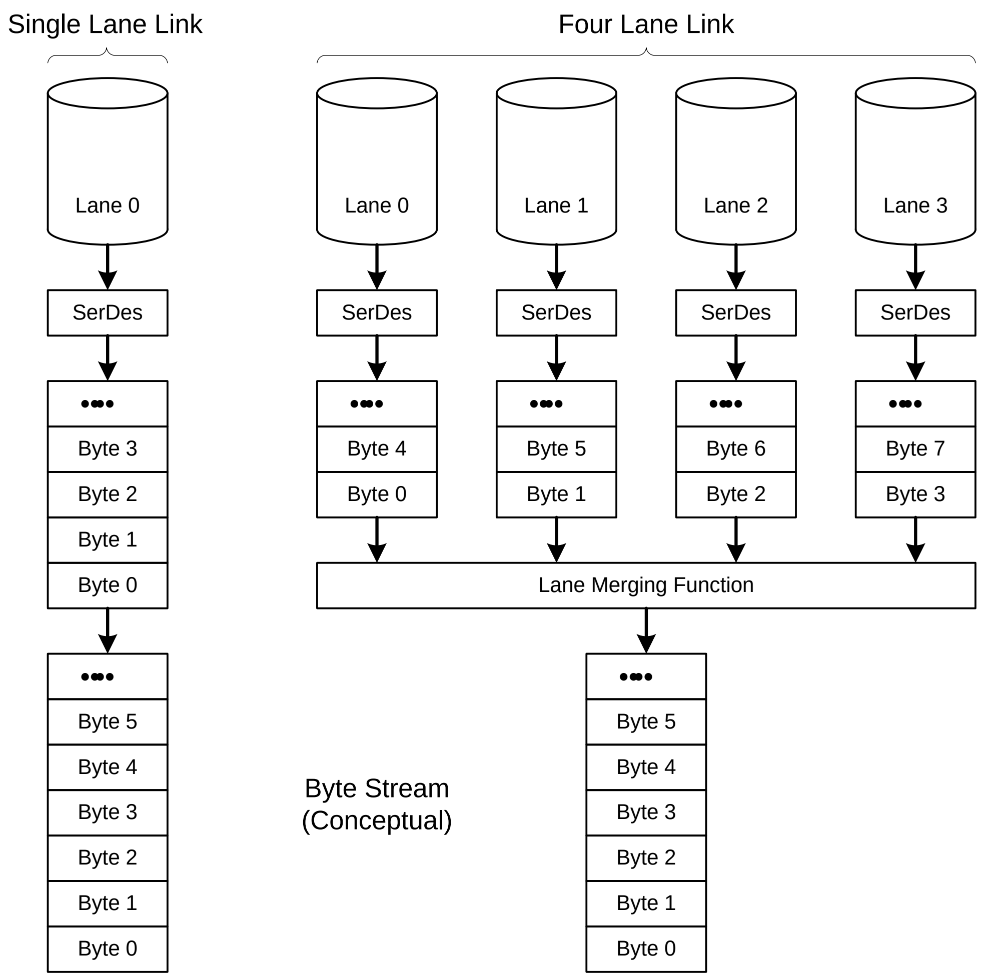


Figure 6 Lane Merger Conceptual Overview

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.

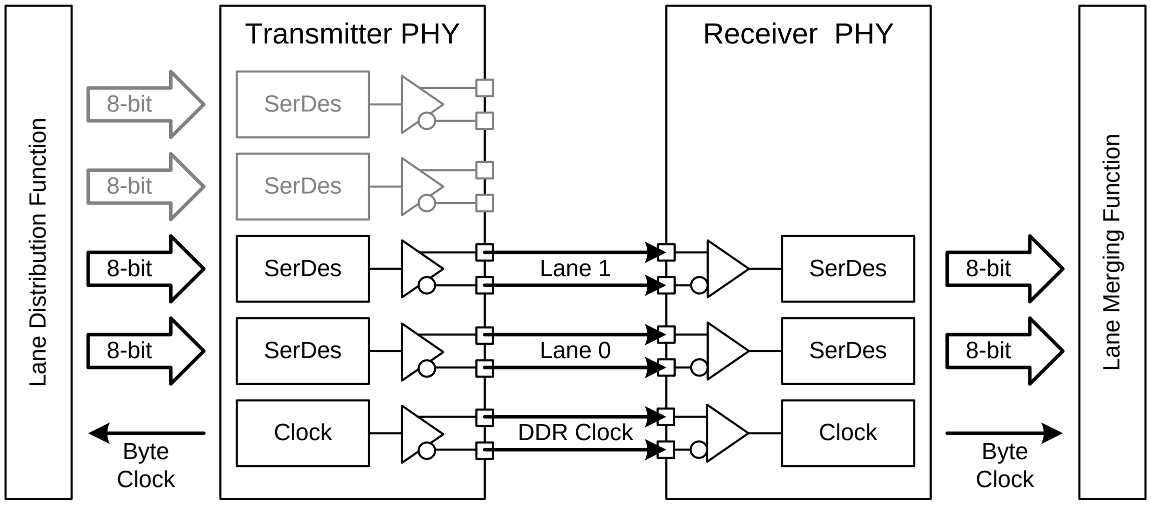


Figure 7 Four-Lane Transmitter with Two-Lane Receiver Example

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 Bidirectionality 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.

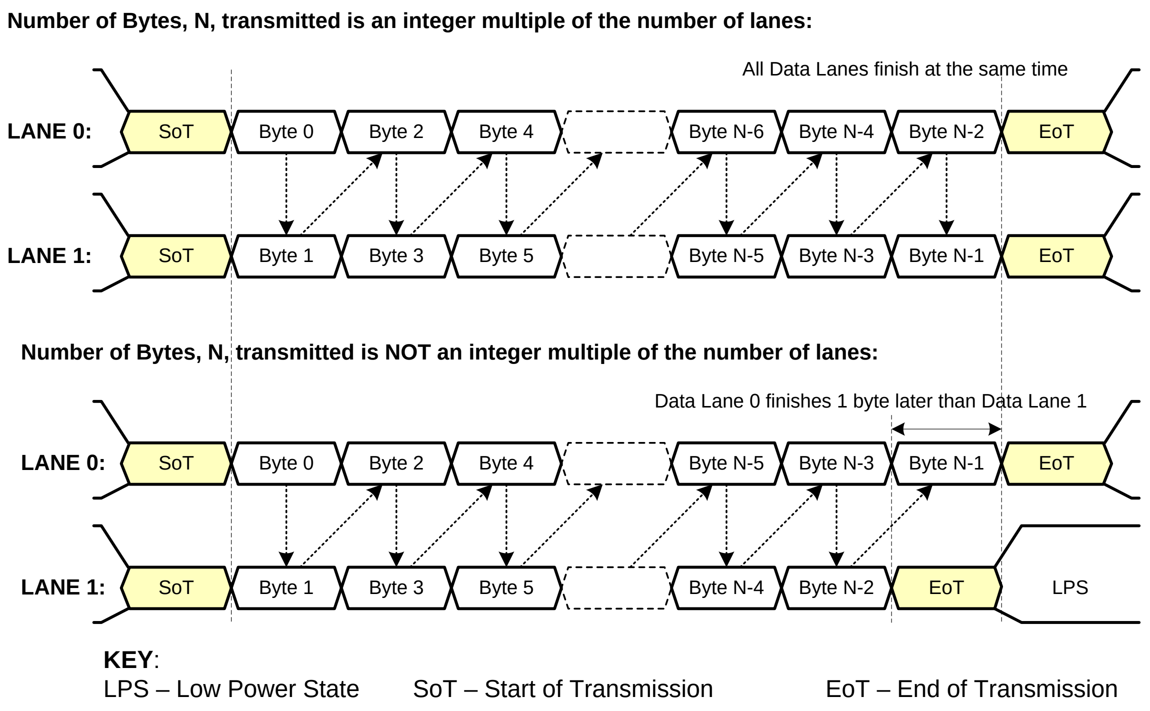
6.1.3 SoT and EoT in Multi-Lane Configurations

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.

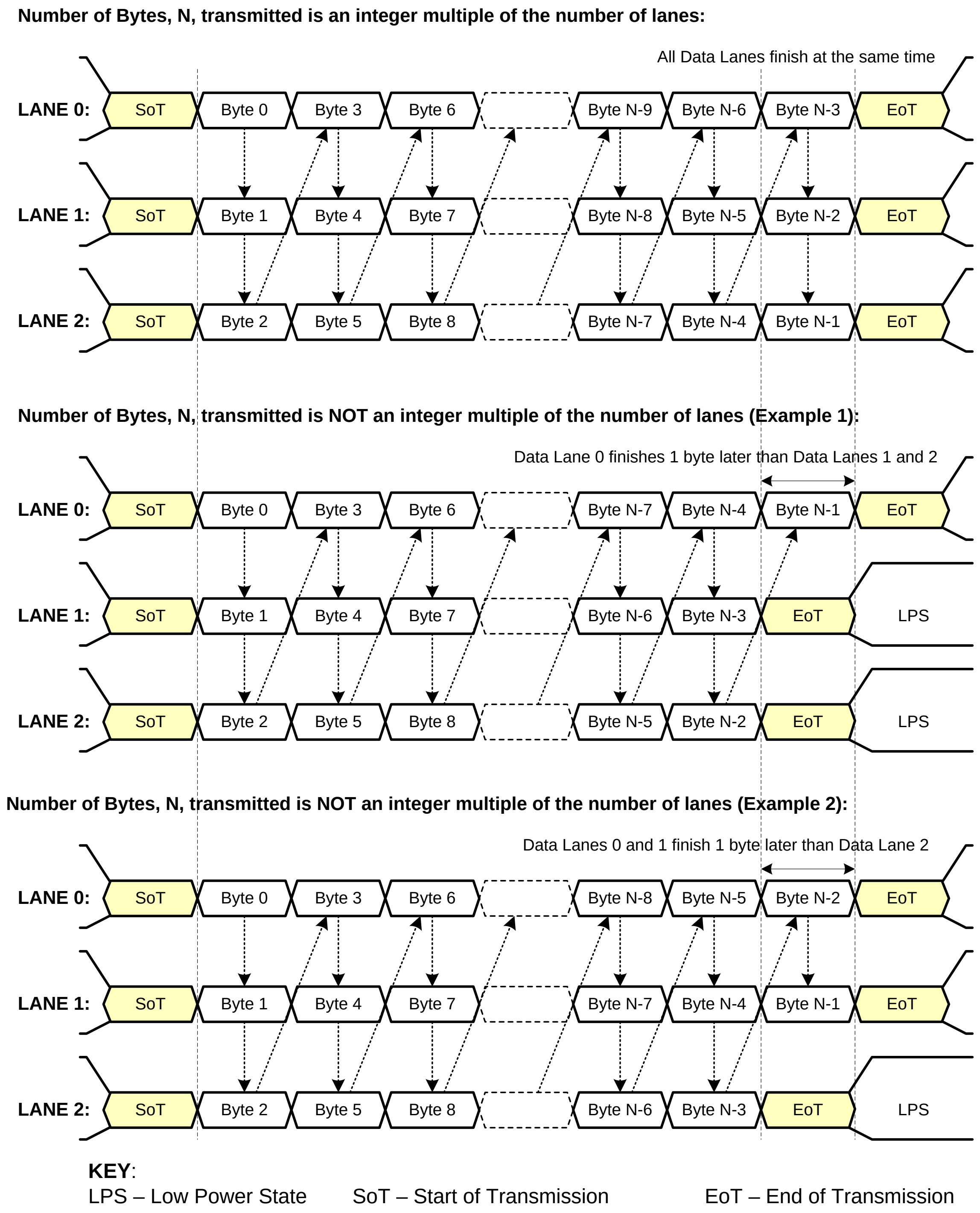
Figure 8 and Figure 9 illustrate a variety of ways a HS transmission can terminate for different number of Lanes and packet lengths.


Number of Bytes, N, transmitted is an integer multiple of the number of lanes:
All Data Lanes finish at the same time
LANE 0:
SoT
Byte 0
Byte 2
Byte 4
Byte N-6
Byte N-4
Byte N-2
EoT
LANE 1:
SoT
Byte 1
Byte 3
Byte 5
Byte N-5
Byte N-3
Byte N-1
EoT
Number of Bytes, N, transmitted is NOT an integer multiple of the number of lanes:
Data Lane 0 finishes 1 byte later than Data Lane 1
LANE 0:
SoT
Byte 0
Byte 2
Byte 4
Byte N-5
Byte N-3
Byte N-1
EoT
LANE 1:
SoT
Byte 1
Byte 3
Byte 5
Byte N-4
Byte N-2
EoT
LPS
KEY:
LPS – Low Power State
SoT – Start of Transmission
EoT – End of Transmission

810

811 **Figure 8 Two Lane HS Transmission Example**

812


Number of Bytes, N, transmitted is an integer multiple of the number of lanes:
All Data Lanes finish at the same time
LANE 0: SoT Byte 0 Byte 3 Byte 6 Byte N-9 Byte N-6 Byte N-3 EoT
LANE 1: SoT Byte 1 Byte 4 Byte 7 Byte N-8 Byte N-5 Byte N-2 EoT
LANE 2: SoT Byte 2 Byte 5 Byte 8 Byte N-7 Byte N-4 Byte N-1 EoT
Number of Bytes, N, transmitted is NOT an integer multiple of the number of lanes (Example 1):
Data Lane 0 finishes 1 byte later than Data Lanes 1 and 2
LANE 0: SoT Byte 0 Byte 3 Byte 6 Byte N-7 Byte N-4 Byte N-1 EoT
LANE 1: SoT Byte 1 Byte 4 Byte 7 Byte N-6 Byte N-3 EoT LPS
LANE 2: SoT Byte 2 Byte 5 Byte 8 Byte N-5 Byte N-2 EoT LPS
Number of Bytes, N, transmitted is NOT an integer multiple of the number of lanes (Example 2):
Data Lanes 0 and 1 finish 1 byte later than Data Lane 2
LANE 0: SoT Byte 0 Byte 3 Byte 6 Byte N-8 Byte N-5 Byte N-2 EoT
LANE 1: SoT Byte 1 Byte 4 Byte 7 Byte N-7 Byte N-4 Byte N-1 EoT
LANE 2: SoT Byte 2 Byte 5 Byte 8 Byte N-6 Byte N-3 EoT LPS
KEY:
LPS – Low Power State
SoT – Start of Transmission
EoT – End of Transmission

813

814 **Figure 9 Three Lane HS Transmission Example**

6.2 Multi-DSI Receiver Configuration with DSI Sub-Links

6.2.1 Architecture for a Multi-DSI Receiver Configuration

This specification optionally supports a DSI Transmitter that connects to two, three or four DSI Receivers by splitting the DSI Link (split link) with DSI Sub-Links. This configuration option allows one DSI Transmitter to split into two, three or four Sub-Links that share a common logical clock. Using a common logical clock usually minimizes skew between separate Sub-Links and one DSI Transmitter and increases the number of DSI Link connections supported by one DSI Transmitter. Splitting DSI Links increases the number of DSI Receivers connected to one DSI Transmitter in an application processor. As panel resolutions increase, multiple DSI Links are needed in order to route display streams to separate parts of the panel without adding separate DSI Links. Unlike Section 4, Figure 1 where there is a one-to-one association between a DSI Transmitter and a DSI Receiver, the multi-receiver configuration is typically used to connect one DSI Transmitter to one display panel and the panel module contains multiple DSI Receivers that route display data to separate portions of the panel for fan out reasons. Implementers can support such a panel either with multiple, separate DSI Links in the Transmitter or more flexibly with one DSI Transmitter block that can create DSI Sub-links.

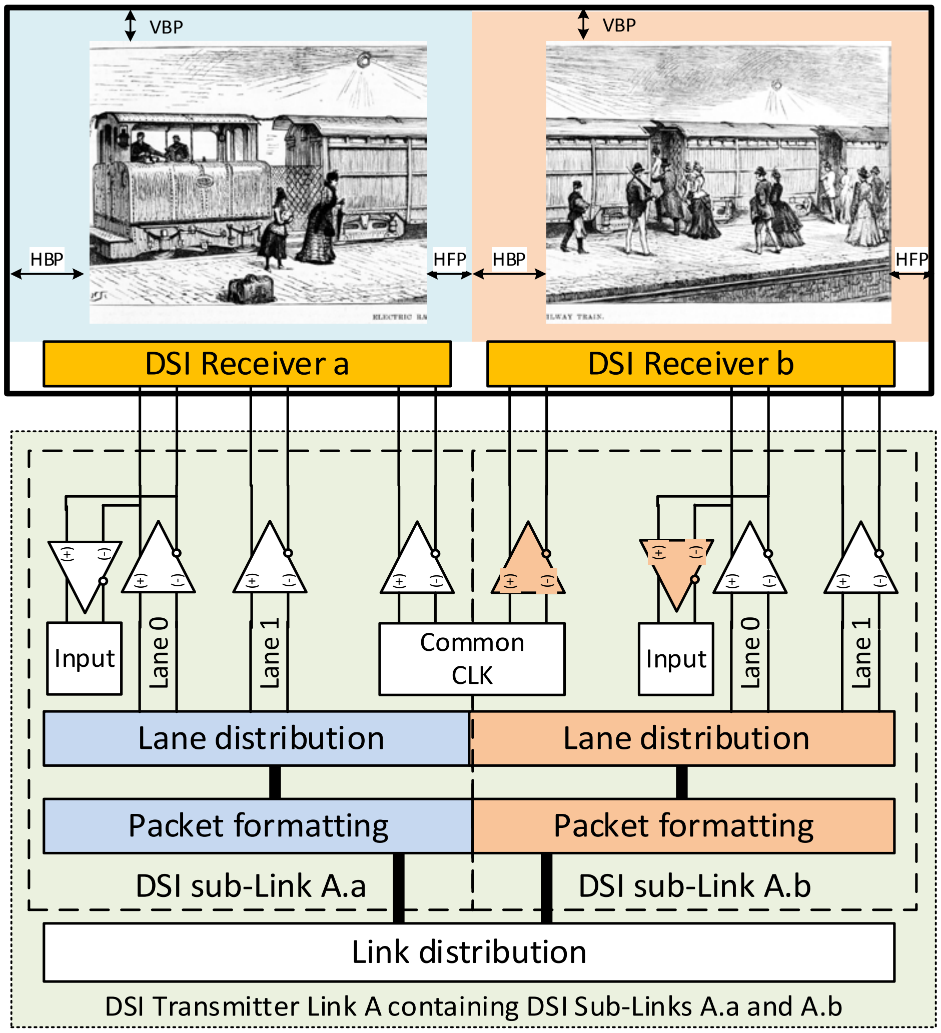


Figure 10 Image Rendered by a Panel Transported by Two DSI Sub-Links

The multi-receiver configurations are illustrated in Figures 10, 11, 12 and 13.

DSI Transmitter configurations that allow a DSI Link to split into Sub-Links shall require the same supported functions as required by this specification for a DSI Link. If a DSI Link can be split, the Sub-Links shall carry symmetrical, evenly divided payloads. That is, each Sub-Link carries equal data to the whole panel.

The options for a DSI Transmitter to split based on the DSI Lane count are:

1. A DSI Link with two Lanes may split into only two DSI Sub-Links; each sub-Link has one DSI Lane and shares the common logical DSI Clock. Figure 11 illustrates this multiple DSI Receiver configuration with two DSI Receivers. Note that the Lane numbers shown in Figure 11 are referenced to the DSI Receiver. If the DSI Transmitter and DSI Receiver in a Sub-Link include LP Mode lane reversal, Lane 0 of that Sub-Link shall support lane reversal.

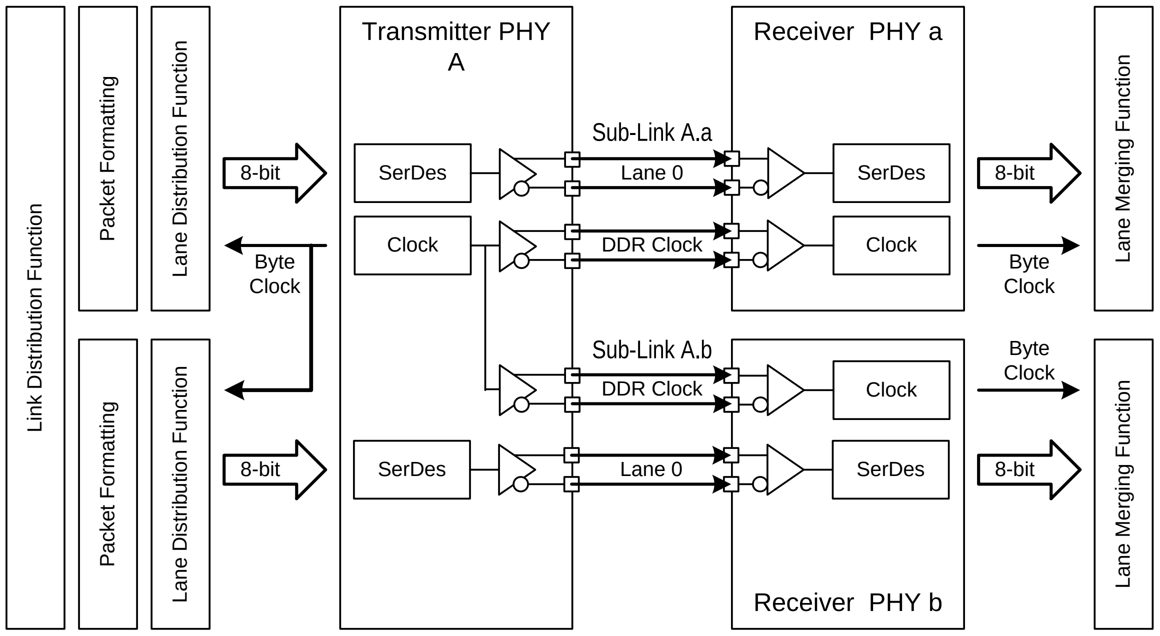


Figure 11 Example of Two DSI Receivers Connected by One-DSI Lane Sub-Links

2. A DSI Link with three Lanes or four Lanes with one Lane unconnected may split into only three DSI Sub-Links; each Sub-Link has one DSI Lane and shares the common logical DSI Clock. Figure 12 illustrates this multiple DSI Receiver configuration with three DSI Receivers. Note that the Lane numbers shown in Figure 12 are referenced to the DSI Receiver. If the DSI Transmitter and DSI Receiver in a Sub-Link include LP Mode lane reversal, Lane 0 of that Sub-Link shall support lane reversal.

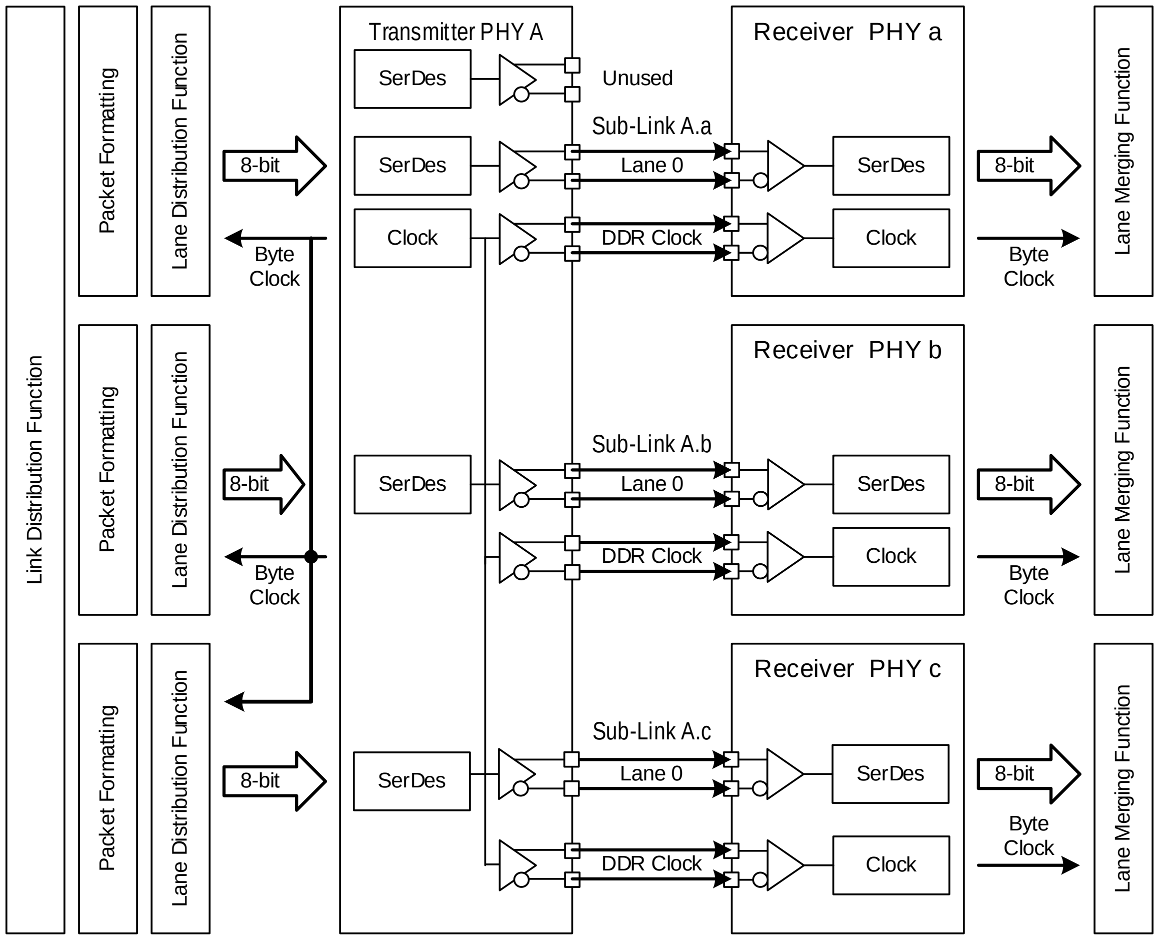


Figure 12 Example of Three DSI Receivers Connected by One-DSI Lane Sub-Links

3. A DSI Link with four Lanes may split in either of two ways:

    1. Into two DSI Sub-Links, each with two DSI Lanes; each Sub-Link shares the common logical DSI Clock. Figure 12 illustrates this multiple DSI Receiver configuration with three DSI Receivers. Note that the Lane numbers shown in Figure 12 are referenced to the DSI Receiver. If the DSI Transmitter and DSI Receiver in a Sub-Link include LP Mode lane reversal, Lane 0 of that Sub-Link shall support lane reversal.

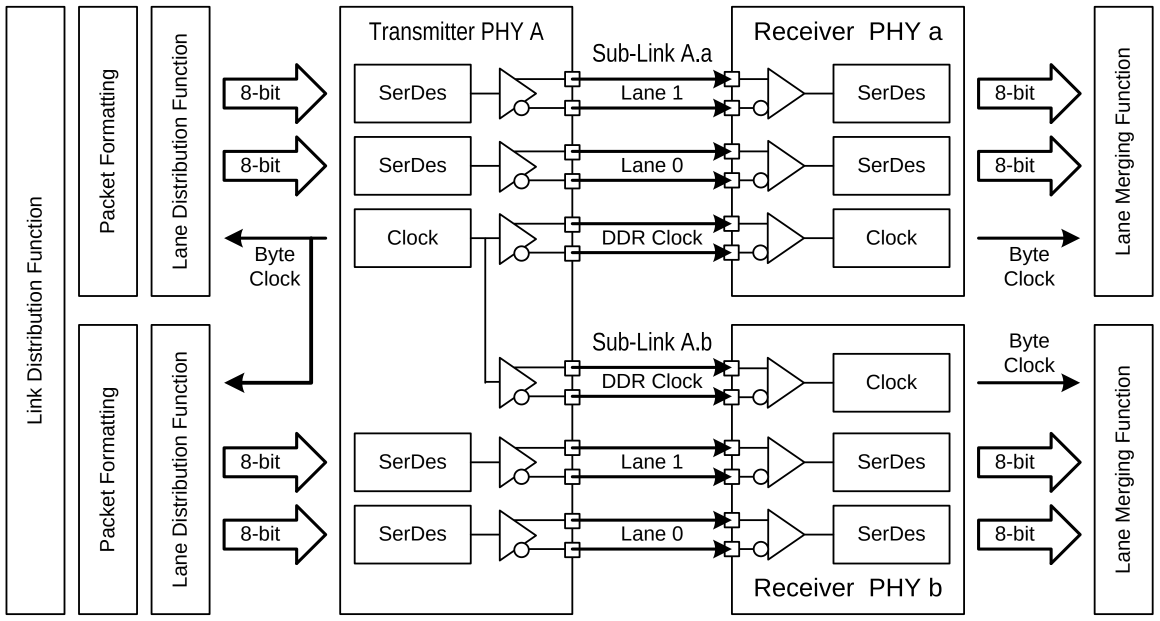


Figure 13 Example of Two DSI Receivers Connected by Two-DSI Lane Sub-Links

2. Into four DSI Sub-Links, each with one DSI Lane; each Sub-Link shares the common logical DSI Clock. Figure 14 illustrates this multiple DSI Receiver configuration with three DSI Receivers. Note that the Lane numbers shown in Figure 14 are referenced to the DSI Receiver. If the DSI Transmitter and DSI Receiver in a Sub-Link include LP Mode lane reversal, Lane 0 of that Sub-Link shall support lane reversal.

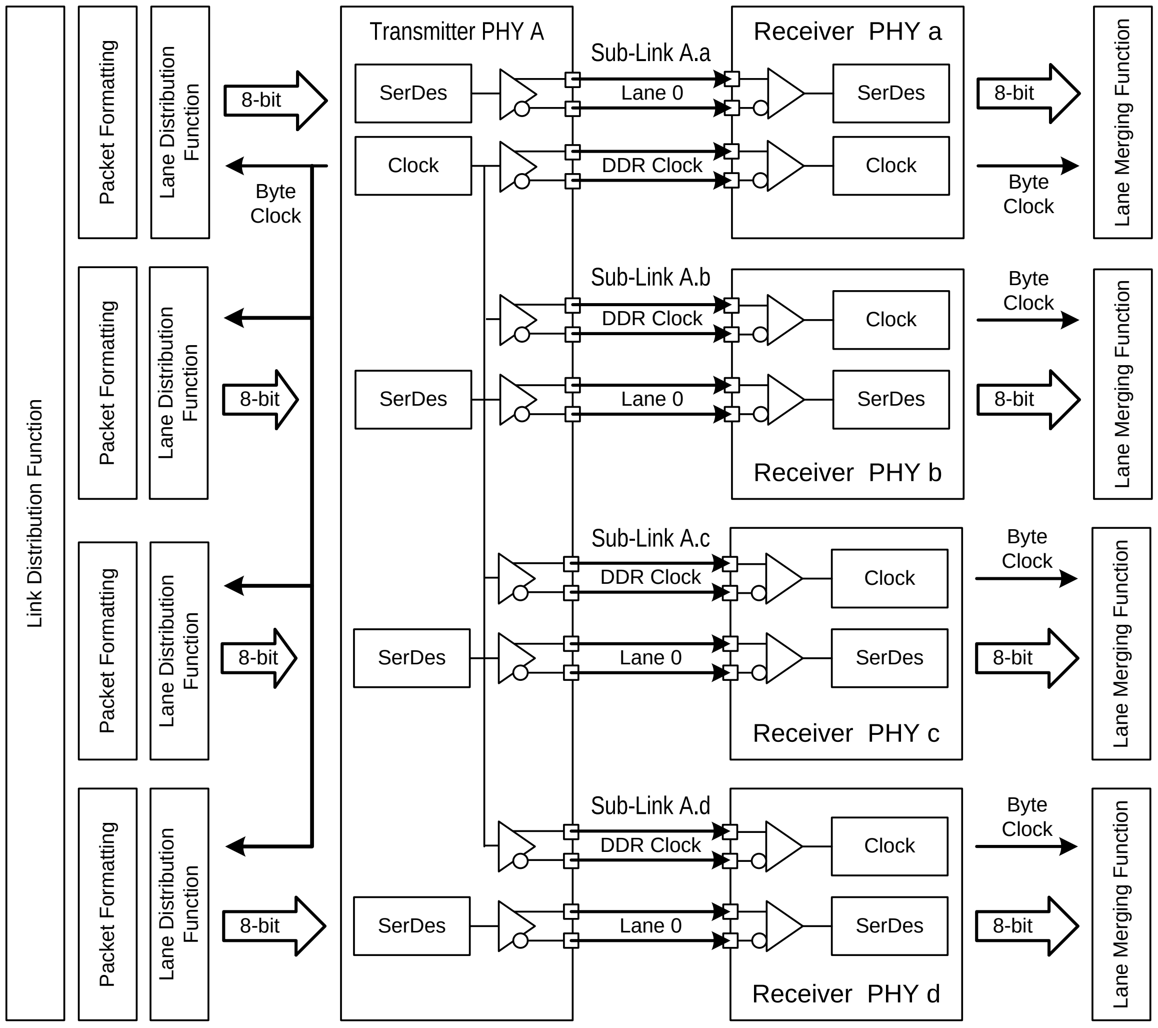


Figure 14 Example of Four DSI Receivers Connected by Sub-Links of One DSI Lane Each

A byte stream is presented to the Lane Distribution Function as illustrated in Figure 15. The Lane Distribution Function is unchanged from DSI Multi-Lane configurations shown in Figure 8 and Figure 9 (Section 6.1.3) and earlier in Section 6. The transmitter implementation is responsible for creating parallel byte streams, for example, with separate PPI interfaces for each byte stream or a Link to Sub-Link distribution function built into the DSI Link block.

6.2.2 Lane Mapping for a Multi-DSI Receiver Configuration

The byte-to-Lane mapping in Section 6 applies to Sub-Links. A one-Lane Sub-Link transports the byte stream in byte order, and a two-Lane Sub-Link transports the byte stream alternating odd and even bytes to Lanes 0 and 1, respectively, in the same fashion as two-Lane DSI Link.

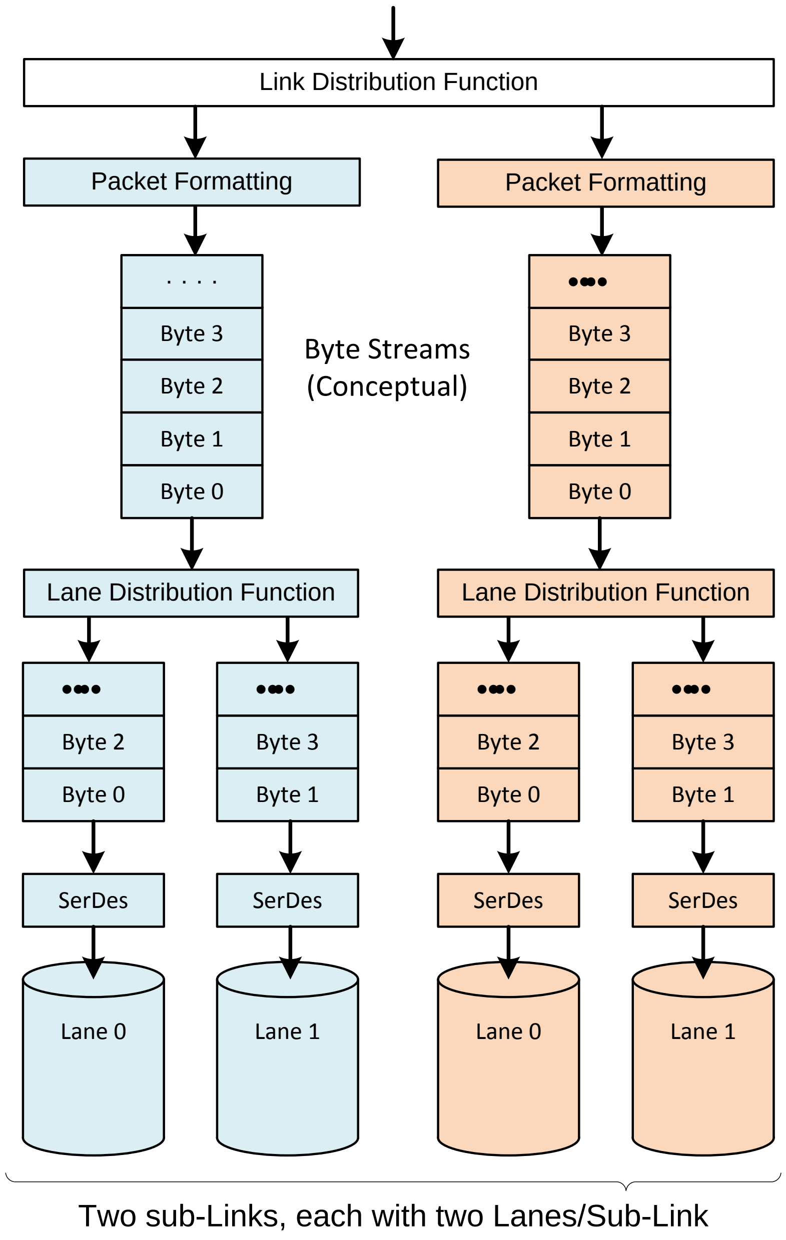

Link Distribution Function
Packet Formatting
Packet Formatting
. . . .
Byte 3
Byte 2
Byte 1
Byte 0
Byte Streams (Conceptual)
Lane Distribution Function
Lane Distribution Function
Byte 2
Byte 0
Byte 3
Byte 1
SerDes
Lane 0
Lane 1
Two sub-Links, each with two Lanes/Sub-Link

Figure 15 Conceptual Streams with a Two Sub-Link Distributor

Mapping bit streams over Sub-Links follows the same guidelines as if the DSI Links are separate. The application processor sends uncompressed pixel data or compressed bit streams to the appropriate DSI Transmitter or the DSI Transmitter configured with DSI Sub-Links. Messaging and control signaling within the application processor upstream of the DSI Transmitter is outside the scope of this Specification.

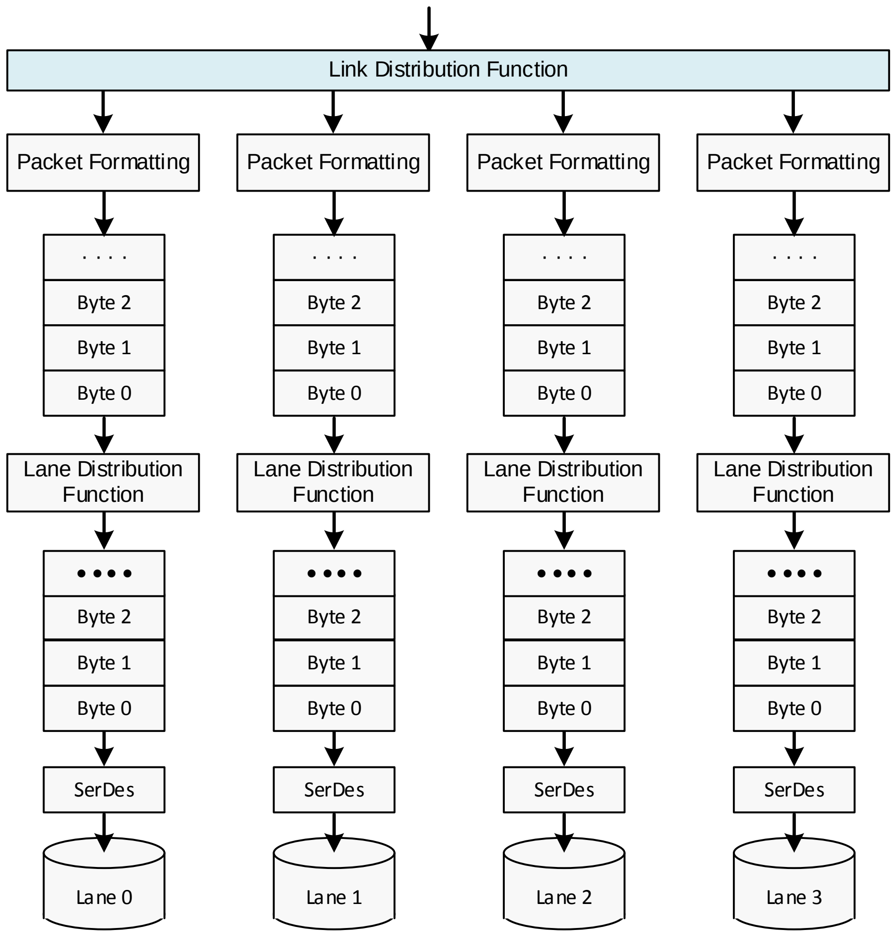


Figure 16 Conceptual Streams with a Four Sub-Link Distributor

6.2.3 Video Mode Lane Timing for a DSI Sub-Link

The following diagram illustrates the timing for a multi-DSI Receiver configuration. In Figure 17, two DSI Receivers will receive data. There will be a skew between lanes dependent on position of data in the DSI Transmitter buffer and from physical layer channel effects.

In no case should the skew exceed the HFP blanking period. However, this Specification makes no requirement on the Lane-to-Lane skew and the Sub-Link to Sub-Link skew between DSI Sub-Links. Implementers should refer to the panel specification for a specific limit to ensure interoperability.

The host shall transport video data on Sub-links simultaneously. Figure 17 shows an example of the host timing where each raster-scanned line fills a line time on each Sub-Link equally.

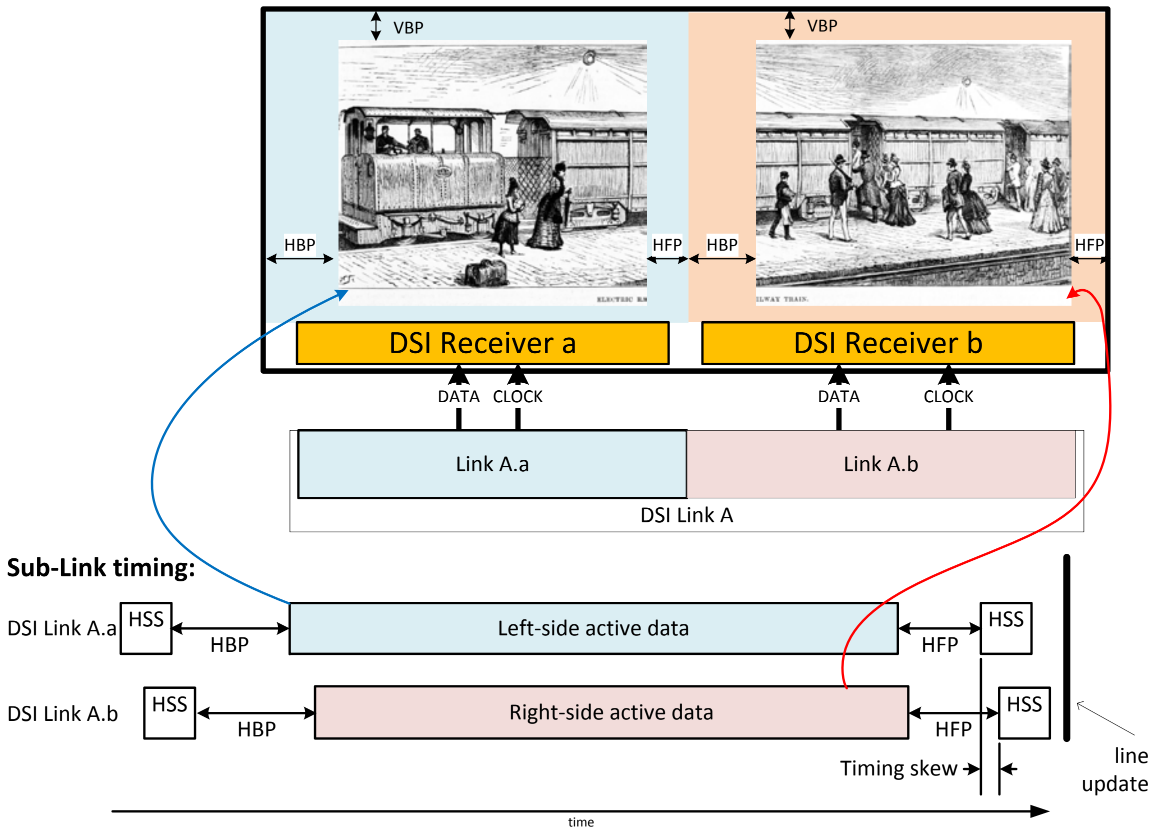


Figure 17 Example of Defined Lane Skew for a Two Sub-Link Configuration

Each Sub-Link shall duplicate the horizontal and vertical synchronization packets (HSS, HSE, VSS, VSE) across all links to preserve video timing to all portions of the panel.

6.2.4 Command Mode Use with DSI Sub-Links in a Multi-DSI Receiver Configuration (informative)

Each Sub-Link transports data based on memory writes using appropriate DCS commands. A panel spans the paired coordinates (0, 0) to (HxN – 1, V – 1), where H is the horizontal pixel size addressed by each Sub-Link, V is the vertical pixel dimension of the display panel, and N is the number of Sub-Links. Data is written to each Sub-Link with appropriate addressing. The manner of addressing each Sub-Link is outside the scope of this specification.

The frame buffer addressed by any Sub-Link is independent of adjoining Sub-Links. Figure 18 shows two memory map diagrams, the top one depicting the host frame and the bottom one depicting a two-Sub-Link panel module. The coordinates of the update region for each Sub-Link are highlighted in the Panel diagram.

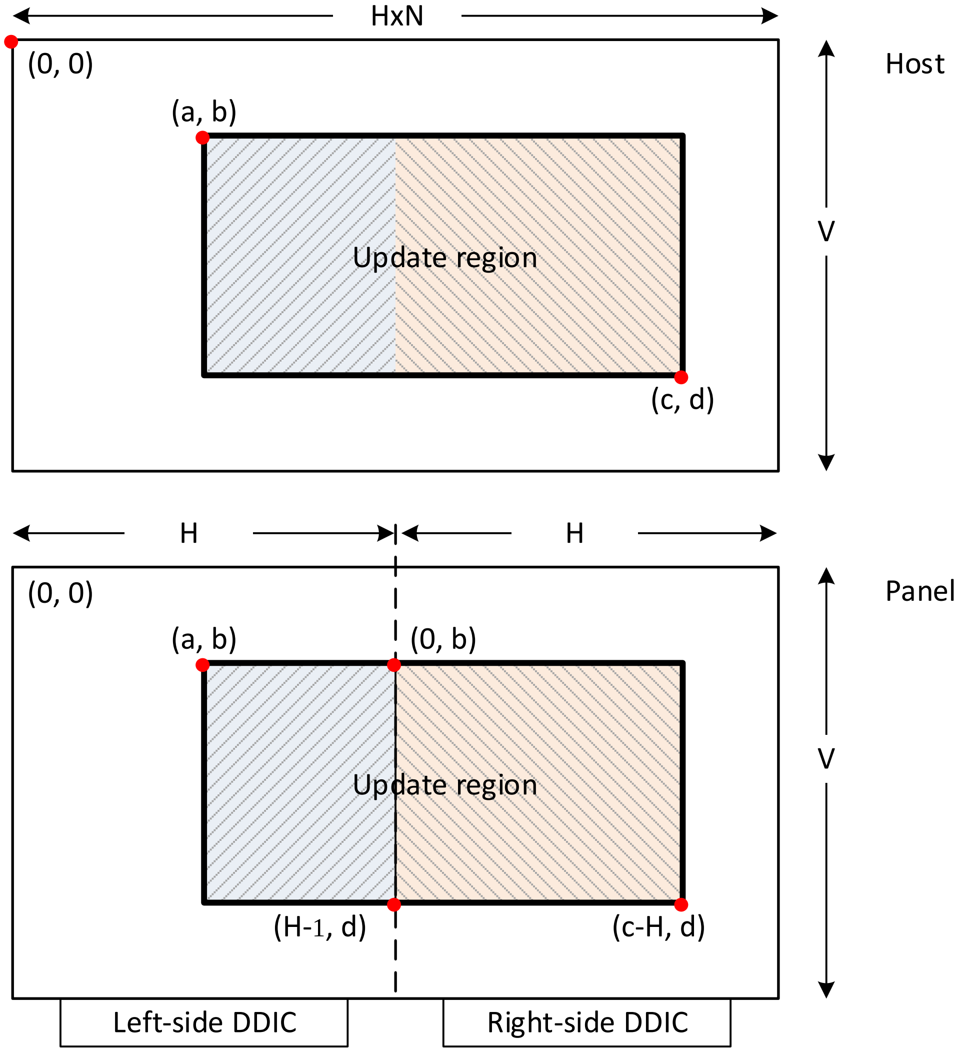


Figure 18 Example Coordinates for Memory Updates Over Two DSI Sub-Links

In the Figure 18 example, a host (GPU) maps a frame for the entire DSI Link into coordinates divided between two Sub-Links. The example suggests updating a region with corners at (a, b) and (c, d) in the host frame memory. The upper left corner (a, b) in the host maps to (a, b) in the Sub-Link for the left-side of the panel, but the lower right horizontal coordinate is limited to H – 1. The right-side Sub-Link begins its address at (0, b) and can address the remainder of the update region to the lower right dimensions (c – H, d).

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 enables 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 is 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

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

SoT Error is detected by the peripheral PHY. If an acknowledge response is expected, the peripheral shall send a response using Data Type 0x02 (*Acknowledge and 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 Acknowledge and Error Report 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 0x02 (*Acknowledge and 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 an *Acknowledge and Error Report* packet. 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 peripheral detects LP-10 (LP request) not followed by the remainder of a valid escape or turnaround sequence or if it detects LP-01 (HS request) not followed by a bridge state (LP-00), a False Control Error shall be captured in the error status register and reported back to the host after the next BTA. This error should be flagged locally to the receiving protocol layer, e.g. when a host detects LP-10 not followed by the remainder of a valid escape or turnaround sequence.

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. Refer to [MIPI04] for definitions of LP High and LP Low faults. The peripheral shall set *Contention Detected* in the Error Report bytes, when it detects either of the contention faults.

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 ensures that any contention that does happen is of limited duration. The peripheral shall set *Peripheral Timeout Error* in the Error Report bytes, when the peripheral detects either HS RX Timer or LP TX Timer – Peripheral, defined in Section 7.2.2.1, has expired,

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

Timer	Timeout	Abbreviation	Requirement
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)

Note that the peripheral HS-RX timeout (see Section 7.2.2.2) should be set to a value shorter than the host processor’s HS-TX 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.

7.2.2.4 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 Section 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 a RESET Trigger Message 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 expires. The side of the Link that issued the BTA then begins a recovery procedure, or re-sends BTA. The specified period shall be longer than the maximum possible turnaround delay for the unit to which the turnaround request was sent. TA_TO is an optional timer.

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

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

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

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

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

7.3.3 Peripheral Response Timeout (PRESP_TO)

Due to design architecture limitations, a peripheral might not be able to respond to certain received packets or requests immediately, and might require some time before being able to respond properly. The duration of this delay is beyond the scope of this document.

One example of this situation is when a multi-bit ECC error occurs on a READ request. If the READ request is followed immediately by a BTA, the peripheral might transmit BTA Accept, followed by a READ Response, when the peripheral should have transmitted Acknowledge and Error Report to notify the host processor of the error condition.

To allow a host processor to account for this delay, the manufacturer of a peripheral shall define a Peripheral Response Timeout (PRESP_TO) to indicate the time necessary after an event until the peripheral can be expected to correctly process received packets, or provide a proper response. A different value for PRESP_TO may be defined for each of the following cases:

- Bus Turn Around
- LPDT READ Request
- LPDT WRITE Request
- HS READ Request
- HS WRITE Request

PRESP_TO begins when the host processor returns to the LP-11 state after the occurrence of any of the previous events. The value of PRESP_TO is specific to the peripheral and beyond the scope of this document.

Each case shall be documented by the peripheral manufacturer in the peripheral data sheet or product documentation. The host processor shall wait for the PRESP_TO of the peripheral to expire after any of the previously indicated events before transmitting any other packets or messages, including BTA.

7.4 Acknowledge and Error Reporting Mechanism

In a bidirectional Link, the peripheral monitors transmissions 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. Errors from multiple transmissions shall be stored and accumulated until a BTA following a transmission provides the opportunity for the peripheral to report errors to the host processor.

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 Trigger Message if there are no errors and with *Acknowledge and Error Report* packet if any errors were detected in previous transmissions. Appropriate flags shall be set to indicate what errors were detected on the preceding transmissions. If the transmission was a Read request, the peripheral shall return READ data without issuing additional ACK Trigger Message or an *Acknowledge and Error Report* packet if no errors were detected. If there was an error in the Read request, the peripheral shall return the appropriate *Acknowledge and Error Report* unless the error was a single-bit correctable error. In that case, the peripheral shall return the requested READ data packet followed by *Acknowledge and Error Report* packet with appropriate error bits set.

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

See Section 8.10.1 for more detail on *Acknowledge 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.

There are two modes of data transmission, HS and LP transmission modes, at the PHY layer. Before a HS transmission can be started, the transmitter PHY issues a SoT sequence to the receiver. After that, data or command packets can be transmitted in HS mode. Multiple packets may exist within a single HS transmission and the end of transmission is always signaled at the PHY layer using a dedicated EoT sequence. In order to enhance the overall robustness of the system, DSI defines a dedicated EoT packet (EoTp) at the protocol layer (Section 1) for signaling the end of HS transmission. For backwards compatibility with earlier DSI systems, the capability of generating and interpreting this EoTp can be enabled or disabled. The method of enabling or disabling this capability is out of scope for this document. PHY-based EoT and SoT sequences are defined in [MIPI04].

The top diagram in Figure 19 illustrates a case where multiple packets are being sent separately with EoTp support disabled. In HS mode, time gaps between packets shall result in separate HS transmissions for each packet, with a SoT, LPS, and EoT issued by the PHY layer between packets. This constraint does not apply to LP transmissions. The bottom diagram in Figure 19 demonstrates a case where multiple packets are concatenated within a single HS transmission.

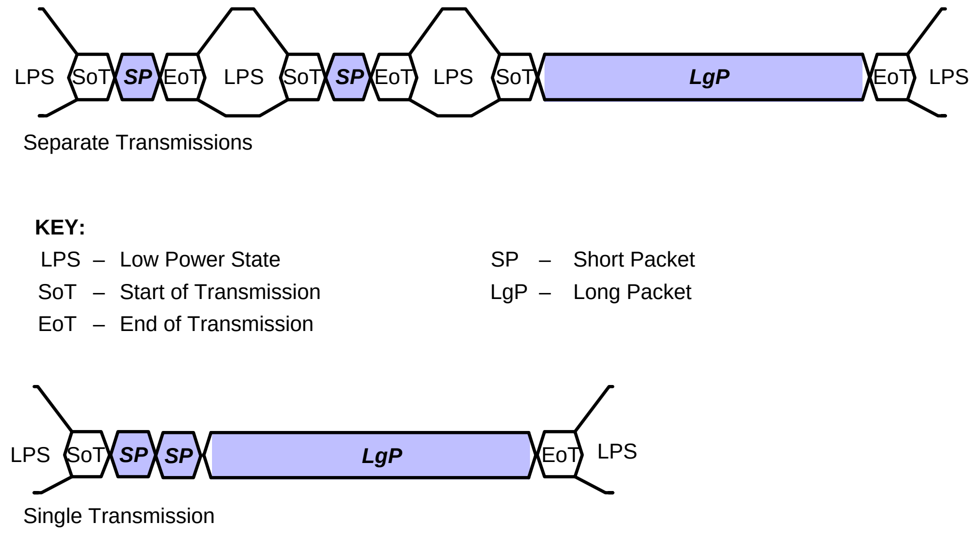


Figure 19 HS Transmission Examples with EoTp Disabled

Figure 20 depicts HS transmission cases where EoTp generation is enabled. In the figure, EoT short packets are highlighted in red. The top diagram illustrates a case where a host is intending to send a short packet followed by a long packet using two separate transmissions. In this case, an additional EoT short packet is generated before each transmission ends. This mechanism provides a more robust environment, at the expense of increased overhead (four extra bytes per transmission) compared to cases where EoTp generation is disabled, i.e. the system only relies on the PHY layer EoT sequence for signaling the end of HS transmission. The overhead imposed by enabling EoTp can be minimized by sending multiple long and short packets within a single transmission as illustrated by the bottom diagram in Figure 20.

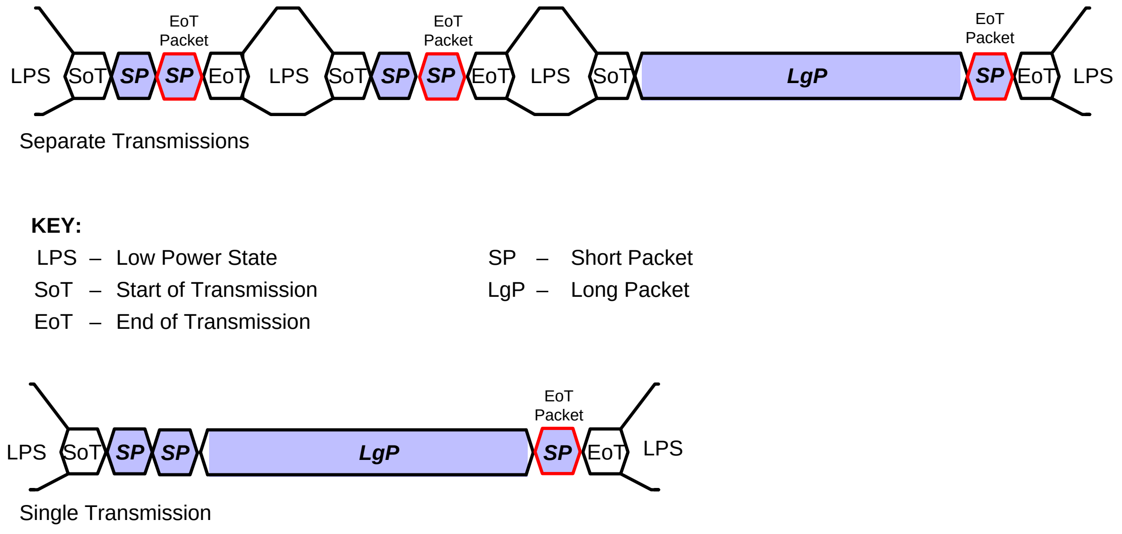


Figure 20 HS Transmission Examples with EoTp Enabled

8.2 Packet Composition

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

Packet sizes fall into two categories:

- ***Short packets*** are four bytes in length including the ECC. 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 8.8.10 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 multi-byte fields, the least significant byte shall be transmitted first unless otherwise specified.

Figure 21 shows a complete Long packet data transmission. Note, the figure shows the byte values in standard positional notation, while the bits are shown in chronological order with the LSB on the left, the MSB on the right and time increasing left to right.

See Section 8.4.1 for a description of the Long packet format.

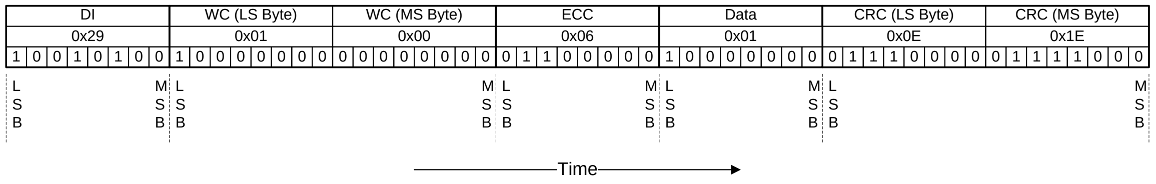


Figure 21 Endian Example (Long Packet)

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 22 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.

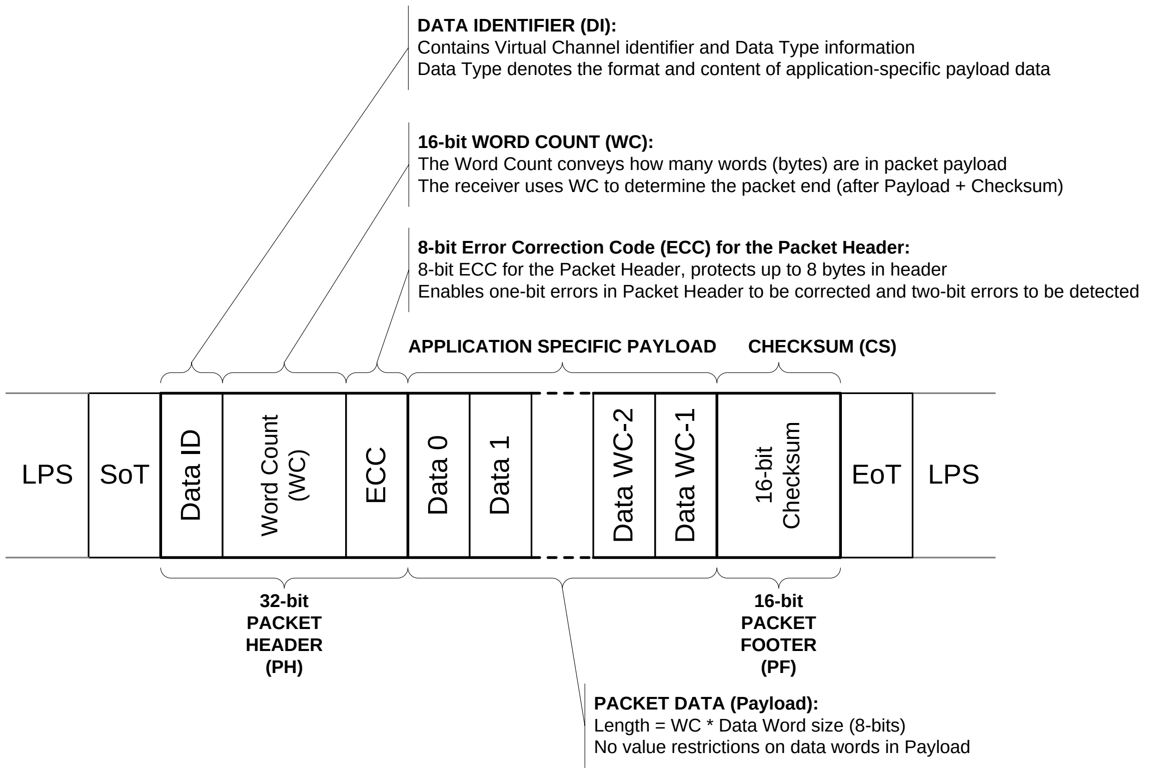

DATA IDENTIFIER (DI):
Contains Virtual Channel identifier and Data Type information
Data Type denotes the format and content of application-specific payload data
16-bit WORD COUNT (WC):
The Word Count conveys how many words (bytes) are in packet payload
The receiver uses WC to determine the packet end (after Payload + Checksum)
8-bit Error Correction Code (ECC) for the Packet Header:
8-bit ECC for the Packet Header, protects up to 8 bytes in header
Enables one-bit errors in Packet Header to be corrected and two-bit errors to be detected
APPLICATION SPECIFIC PAYLOAD
CHECKSUM (CS)
LPS
SoT
Data ID
Word Count (WC)
ECC
Data 0
Data 1
Data WC-2
Data WC-1
16-bit Checksum
EoT
LPS
32-bit PACKET HEADER (PH)
16-bit PACKET FOOTER (PF)
PACKET DATA (Payload):
Length = WC * Data Word size (8-bits)
No value restrictions on data words in Payload

Figure 22 Long Packet Structure

The Data Identifier defines the Virtual Channel for the data and the Data Type for the application specific payload data. See Section 8.7.2 through Section 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 (0xFFFF). If the Checksum is not calculated, the Packet Footer shall consist of two bytes of all zeros (0x0000). 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 23 shows the structure of the Short packet. See Section 8.7.2 through Section 8.10 for descriptions of the Data Types. A Short packet shall contain an 8-bit Data ID followed by two command or data bytes and an 8-bit ECC; a Packet Footer shall not be present. Short packets shall be four 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.

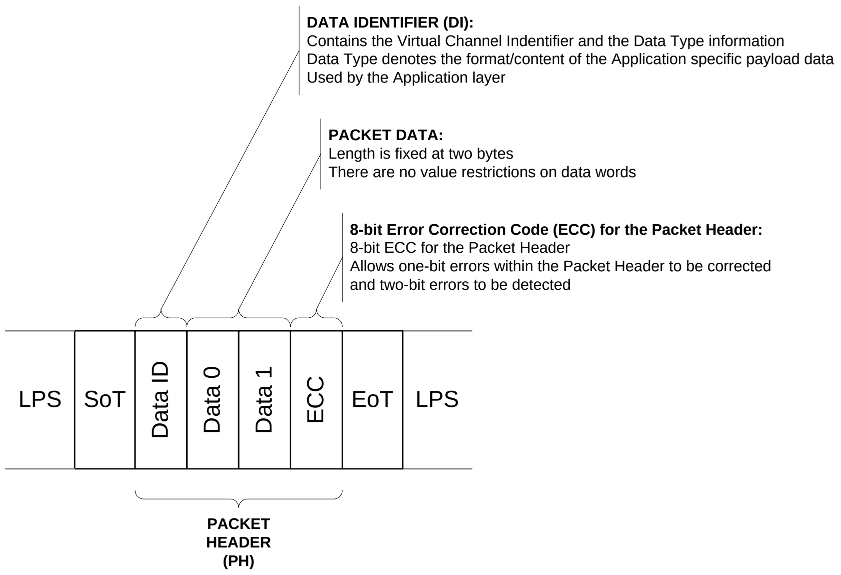

DATA IDENTIFIER (DI):
Contains the Virtual Channel Indentifier and the Data Type information
Data Type denotes the format/content of the Application specific payload data
Used by the Application layer
PACKET DATA:
Length is fixed at two bytes
There are no value restrictions on data words
8-bit Error Correction Code (ECC) for the Packet Header:
8-bit ECC for the Packet Header
Allows one-bit errors within the Packet Header to be corrected
and two-bit errors to be detected
LPS
SoT
Data ID
Data 0
Data 1
ECC
EoT
LPS
PACKET
HEADER
(PH)

Figure 23 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 package is the DI (Data Identifier) byte. Figure 24 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.

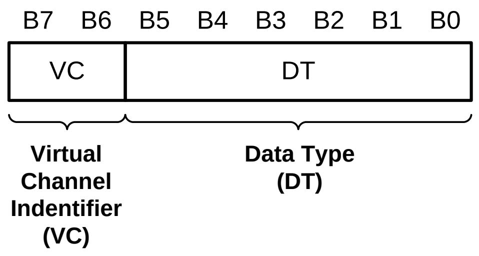


Figure 24 Data Identifier Byte

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]

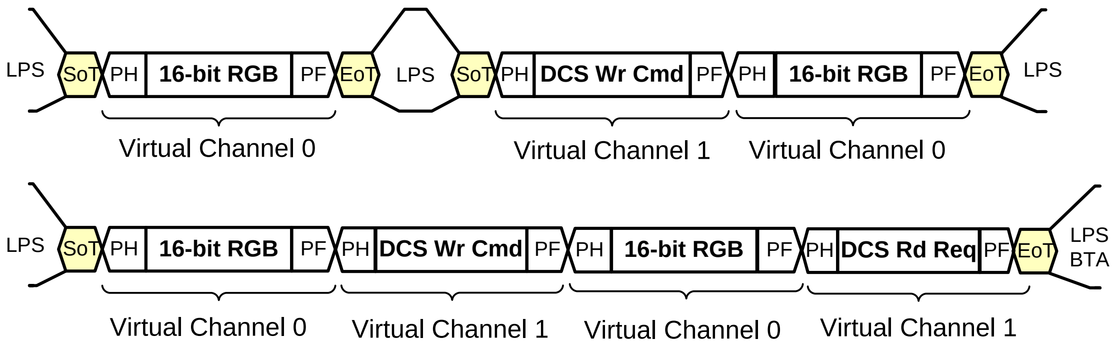
The Data Type field specifies if the packet is a Long or Short packet type and the packet format. The Data Type field, along with the Word Count field for Long packets, informs the receiver of how many bytes to expect in the remainder of the packet. This is necessary because there are no special packet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

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 shall support ECC in both forward- and reverse-direction communications. See Section 9 for more information on coding and decoding the ECC and Section 8.9.2 for ECC and Checksum requirements.

8.6 Interleaved Data Streams



KEY:

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

Figure 25 Interleaved Data Stream Example with EoTp Disabled

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 26.

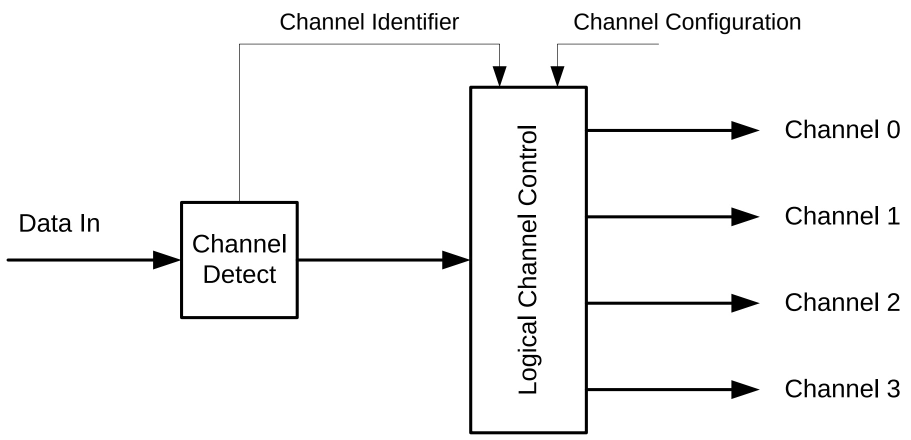


Figure 26 Logical Channel Block Diagram (Receiver Case)

8.6.1 Interleaved Data Streams and Bidirectionality

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. Returning packets shall be tagged with the ID of the peripheral that sent the packet.

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

8.7.1 Processor-sourced Data Type Summary

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
0x01	00 0001	Sync Event, V Sync Start	Short
0x11	01 0001	Sync Event, V Sync End	Short
0x21	10 0001	Sync Event, H Sync Start	Short
0x31	11 0001	Sync Event, H Sync End	Short
0x07	00 0111	Compression Mode Command	Short
0x08	00 1000	End of Transmission packet (EoTp)	Short
0x02	00 0010	Color Mode (CM) Off Command	Short
0x12	01 0010	Color Mode (CM) On Command	Short
0x22	10 0010	Shut Down Peripheral Command	Short
0x32	11 0010	Turn On Peripheral Command	Short

Data Type, hex	Data Type, binary	Description	Packet Size
0x03	00 0011	Generic Short WRITE, no parameters	Short
0x13	01 0011	Generic Short WRITE, 1 parameter	Short
0x23	10 0011	Generic Short WRITE, 2 parameters	Short
0x04	00 0100	Generic READ, no parameters	Short
0x14	01 0100	Generic READ, 1 parameter	Short
0x24	10 0100	Generic READ, 2 parameters	Short
0x05	00 0101	DCS Short WRITE, no parameters	Short
0x15	01 0101	DCS Short WRITE, 1 parameter	Short
0x06	00 0110	DCS READ, no parameters	Short
0x16	01 0110	Execute Queue	Short
0x37	11 0111	Set Maximum Return Packet Size	Short
0x09	00 1001	Null Packet, no data	Long
0x19	01 1001	Blanking Packet, no data	Long
0x29	10 1001	Generic Long Write	Long
0x39	11 1001	DCS Long Write/write_LUT Command Packet	Long
0x0A	01 1010	Picture Parameter Set	Long
0x0B	00 1011	Compressed Pixel Stream	Long
0x0C	00 1100	Loosely Packed Pixel Stream, 20-bit YCbCr, 4:2:2 Format	Long
0x1C	01 1100	Packed Pixel Stream, 24-bit YCbCr, 4:2:2 Format	Long
0x2C	10 1100	Packed Pixel Stream, 16-bit YCbCr, 4:2:2 Format	Long
0x0D	00 1101	Packed Pixel Stream, 30-bit RGB, 10-10-10 Format	Long
0x1D	01 1101	Packed Pixel Stream, 36-bit RGB, 12-12-12 Format	Long
0x3D	11 1101	Packed Pixel Stream, 12-bit YCbCr, 4:2:0 Format	Long
0x0E	00 1110	Packed Pixel Stream, 16-bit RGB, 5-6-5 Format	Long
0x1E	01 1110	Packed Pixel Stream, 18-bit RGB, 6-6-6 Format	Long
0x2E	10 1110	Loosely Packed Pixel Stream, 18-bit RGB, 6-6-6 Format	Long
0x3E	11 1110	Packed Pixel Stream, 24-bit RGB, 8-8-8 Format	Long
0xX0 and 0xXF, unspecified	XX 0000 XX 1111	DO NOT USE All unspecified codes are reserved	

8.7.2 Frame Synchronized Transactions

The display module may optionally support and designate in the manufacturer data sheet if a data type transaction can be frame synchronized. This capability synchronizes the time when data type transactions may take effect in multiple display drivers contained in one display module. If a display module contains multiple display drivers, some set of commands, data types or register writes can be designated as a FSC.

All FSC transactions are placed in a command queue in first-in, first-out order until a controlling display driver receives an Execute Queue Command. The subset of transactions which are designated as FSC is implementation-specific and is outside the scope of this specification.

The command queue execution begins by sending an Execute Queue transaction to a designated controlling display driver via a designated DSI link. At the first internal vertical blanking interval after the controlling display driver receives an Execute Queue, all FSCs in the command queues of all the display drivers are executed. Only FSCs will be placed into a command queue; therefore any transaction not designated as an FSC by the display module is not queued.

After sending an Execute Queue, a DSI transmitter shall not send an FSC until after the last pixel for the frame following the first VSS after receiving the Execute Queue command. The sequence requirement allows a display receiver to flush the present queue before receiving one or more new FSC. This restriction applies in both video and command modes. For clarity, refer to the state diagram in Figure 27.

The depth of the command queue is implementation-specific and outside the scope of this specification.

The method used by the controlling display driver to synchronize execution across multiple display drivers is internal to the display module and outside the scope of this specification.

In order to synchronize commands and maintain accurate timing, the clocking of multiple DSI links should rely on one common timing source in the host processor.

Figure 27 is an example showing when FSC are queued, restricted from being sent and executed when using video mode. The same sequence is in force in command mode where queued commands are executed at the internal vertical blanking interval of the display driver.

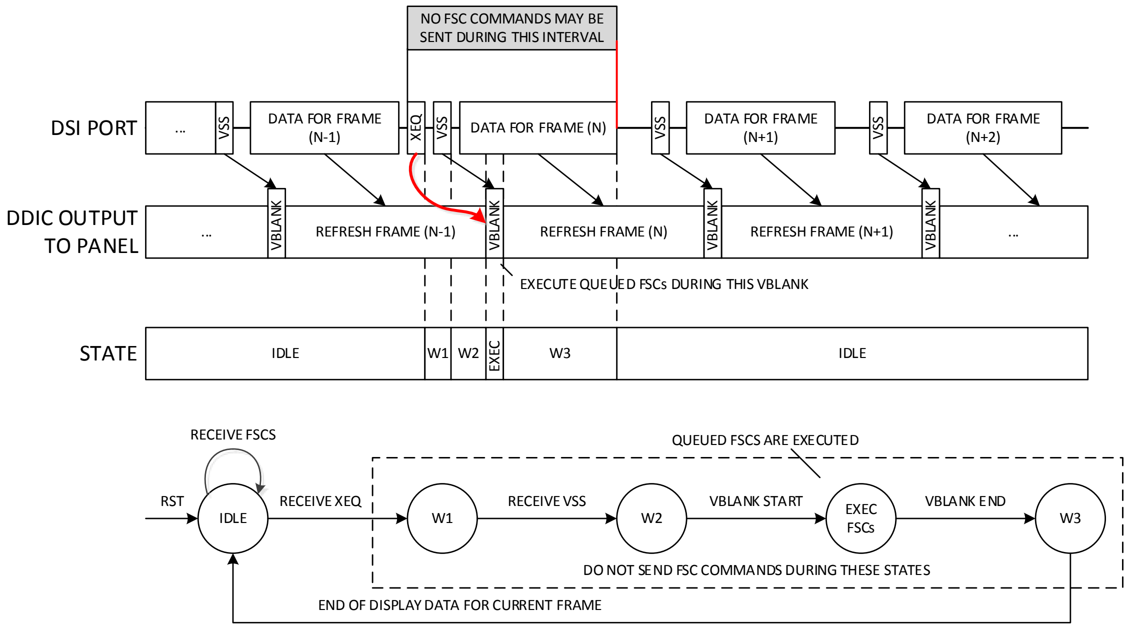


Figure 27 Frame Synchronized Transaction Timing and State Diagram

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 (0xX1)

Sync Events are Short packets 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 (0x01)	V Sync Start
• Data Type = 01 0001 (0x11)	V Sync End
• Data Type = 10 0001 (0x21)	H Sync Start
• Data Type = 11 0001 (0x31)	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 an H Sync Start event for the first line of the VSA. Similarly, a V Sync End event implies an H Sync Start event for the last line of the VSA. If the host processor sources interlaced video, horizontal sync timing follows standard interlaced video conventions for the video format being used and are beyond the scope of this document. See [CEA01] for timing details of interlaced video formats. The first field of interlaced video follows the same rules to imply H Sync Start. The peripheral (display), when receiving the interlaced second video field, shall not imply an H Sync Start at the V Sync Start and V Sync End timing. Refer to Annex C for a detailed progression order of event packets for progressive scan and interlaced scan video timing.

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.

8.8.1.1 Sync Event Payloads

Limited use of a Sync Event payload in a Short packet might send information from a host processor to a display peripheral. This technique is useful for one-byte payloads, in particular when an effect might apply at the beginning, or end, of the frame, and when using a DCS Short WRITE might not be desirable, or supported.

The Data 0 payload of a V Sync Start Event shall indicate if special data payload is present. If Data 0 = 0x00, the peripheral may ignore the contents of the remaining payload bytes. If any bit of Data 0 is not zero, the peripheral shall interpret the contents of Data 1 payload based of the context defined in Table 17.

Table 17 Context Definitions for Vertical Sync Start Event Data 0 Payload

Bit	Definition
7	Reserved for future use as the payload extension indicator when more than one payload byte might be present.
6	Reserved
5	Reserved
4	Reserved
3	3D Control payload is present
2	Reserved
1	Reserved
0	Reserved

8.8.1.2 Stereoscopic Display Control in Video Mode (3D Control)

DSI supports viewing a stereoscopic image with a display peripheral operating in video mode. The method of data delivery to the display peripheral shall be specified using the Short-packet Data 1 payload in the VSS at the beginning of a frame. A host processor shall send 3D Control information at every change of the 3D Control information, or more frequently, as specified in the display peripheral data sheet. The bits of the Data 1 payload are summarized in Table 18.

Table 18 3D Control Payload in Vertical Sync Start Event Data 1 Payload

Bit	Description[1]
7	Reserved, set to ‘0’.
6	Reserved, set to ‘0’.
5	3DL/R – Left / Right Order
4	3DVSYNC – Second VSYNC Enabled between Left and Right Images
3	3DFMT[1:0] (B3:2]) – 3D Image Format
2	
1	3DMODE[1:0] (B[1:0]) – 3D Mode On / Off, Display Orientation
0	

1. *See Section 5.1 of [MIPI05] for detailed descriptions.*

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

This short packet is used for indicating the end of a HS transmission to the data link layer. As a result, detection of the end of HS transmission may be decoupled from physical layer characteristics. [MIPI04] defines an EoT sequence composed of a series of all 1’s or 0’s depending on the last bit of the last packet within a HS transmission. Due to potential errors, the EoT sequence could be interpreted incorrectly as valid data types. Although EoT errors are not expected to happen frequently, the addition of this packet will enhance overall system reliability.

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

The main objective of the EoTp is to enhance overall robustness of the system during HS transmission mode. Therefore, DSI transmitters should not generate an EoTp when transmitting in LP mode. The Data Link layer of DSI receivers shall detect and interpret arriving EoTps regardless of transmission mode (HS or LP modes) in order to decouple itself from the physical layer. Table 19 describes how DSI mandates EoTp support for different transmission and reception modes.

Table 19 EoT Support for Host and Peripheral

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

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

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

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

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

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

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

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

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

Shutdown Peripheral command is a Short packet command 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.

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

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

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

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

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

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

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 and Error Report*. If an ECC error in the request was detected and corrected, the peripheral shall transmit the requested READ data packet with the *Acknowledge and 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 Checksum is 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.

8.8.9 DCS Commands

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

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

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

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

DCS Short Write command is used to write a single data byte to a peripheral such as a display module. The packet is a Short packet composed of a Data ID byte, a DCS Write command, an optional parameter byte and an ECC byte. Data Type bit 4 shall be set to 1 if there is a valid parameter byte, and shall be set to 0 if there is no valid parameter byte. If a parameter is not required, the parameter byte shall be 0x00. If *DCS Short Write* command, followed by BTA, is sent to a bidirectional peripheral, the peripheral shall respond with ACK Trigger Message 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 and Error Report*. If the peripheral is a Video Mode display on a unidirectional DSI, it shall ignore BTA. See Table 22.

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

DCS READ commands are used to request data from a display module. This packet is a Short packet composed of a Data ID byte, a DCS Read command, a byte set to 0x00 and an ECC byte. Since this is a read command, BTA shall be asserted by the host processor following completion of the transmission. 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 8.8.10 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 and Error Report*. If an ECC error in the request was detected and corrected, the peripheral shall send the requested READ data packet followed by the *Acknowledge and 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.

8.8.9.3 DCS Long Write / write_LUT Command, Data Type = 11 1001 (0x39)

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.

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

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.

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

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.

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

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. Blanking packets may contain arbitrary data as payload.

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.

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

Generic Long Write 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.

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

Loosely Packed Pixel Stream 20-bit YCbCr 4:2:2 Format shown in Figure 28 is a Long packet used to transmit image data formatted as 20-bits per pixel to a Video Mode display module. The packet consists of the DI byte, a two-byte, non-zero WC, an ECC byte, a payload of length WC bytes and a two-byte Checksum.

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

A pixel shall have ten bits for each of the Y-, Cb-, and Cr-components loosely packed into 12-bit fields as shown in Figure 28. The 10-bit component value shall be justified such that the most significant bits of the 12-bit field, b[11:2] holds the 10-bit component value, d[9:0]. The least significant bits of the 12-bit field, b[1:0], shall be 00b. Within a component, the LSB is sent first, the MSB last.

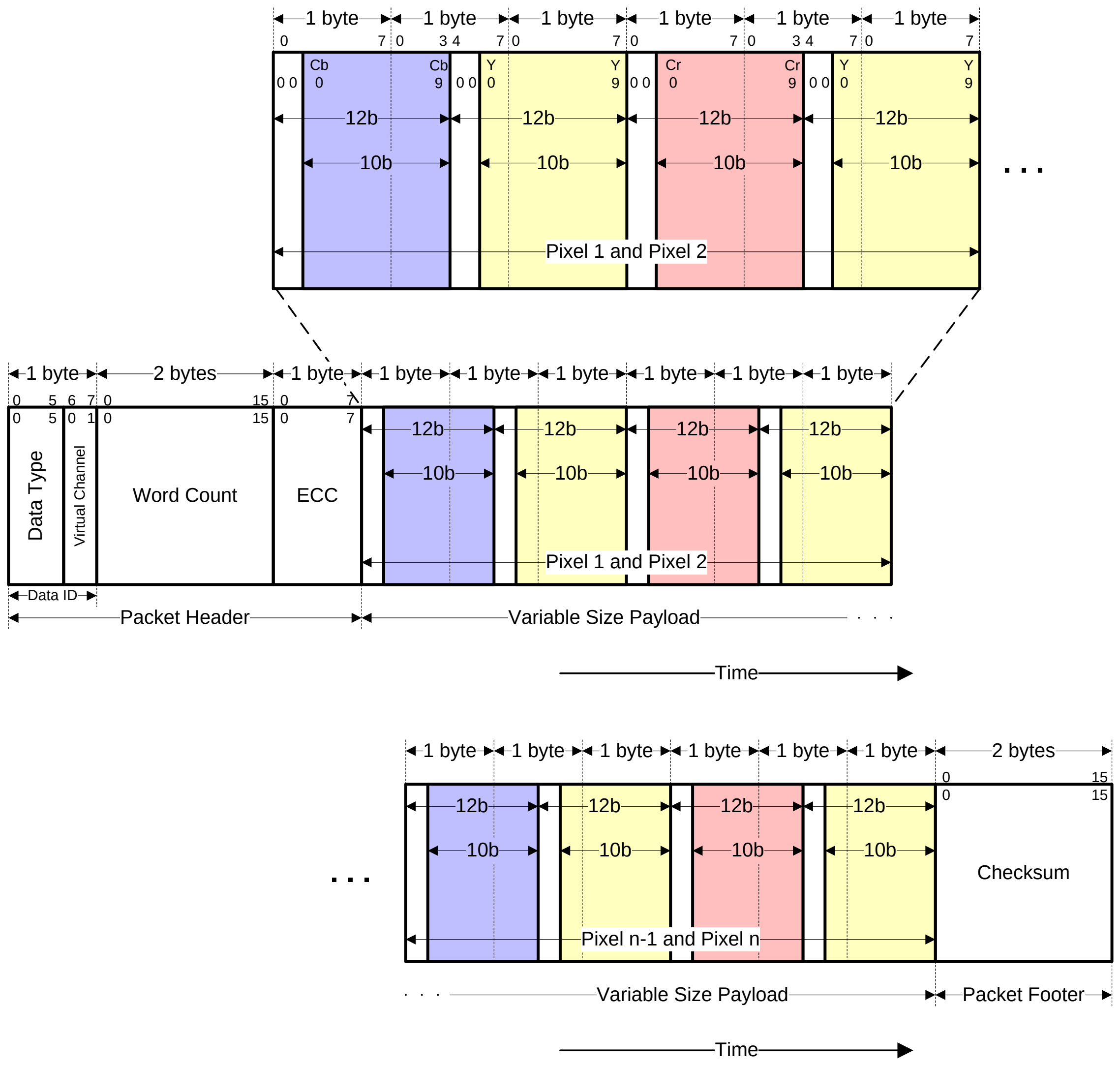


Figure 28 20-bit per Pixel – YCbCr 4:2:2 Format, Long Packet

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

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

***Packed Pixel Stream 24-bit YCbCr 4:2:2 Format* shown in**

Figure 29 is a Long packet used to transmit image data formatted as 24-bits per pixel to a Video Mode display module. The packet consists of the DI byte, a two-byte, non-zero WC, an ECC byte, a payload of length WC bytes and a two-byte Checksum.

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

A pixel shall have twelve bits for each of the Y-, Cb-, and Cr-components as shown in Figure 29. Within a component, the LSB is sent first, the MSB last.

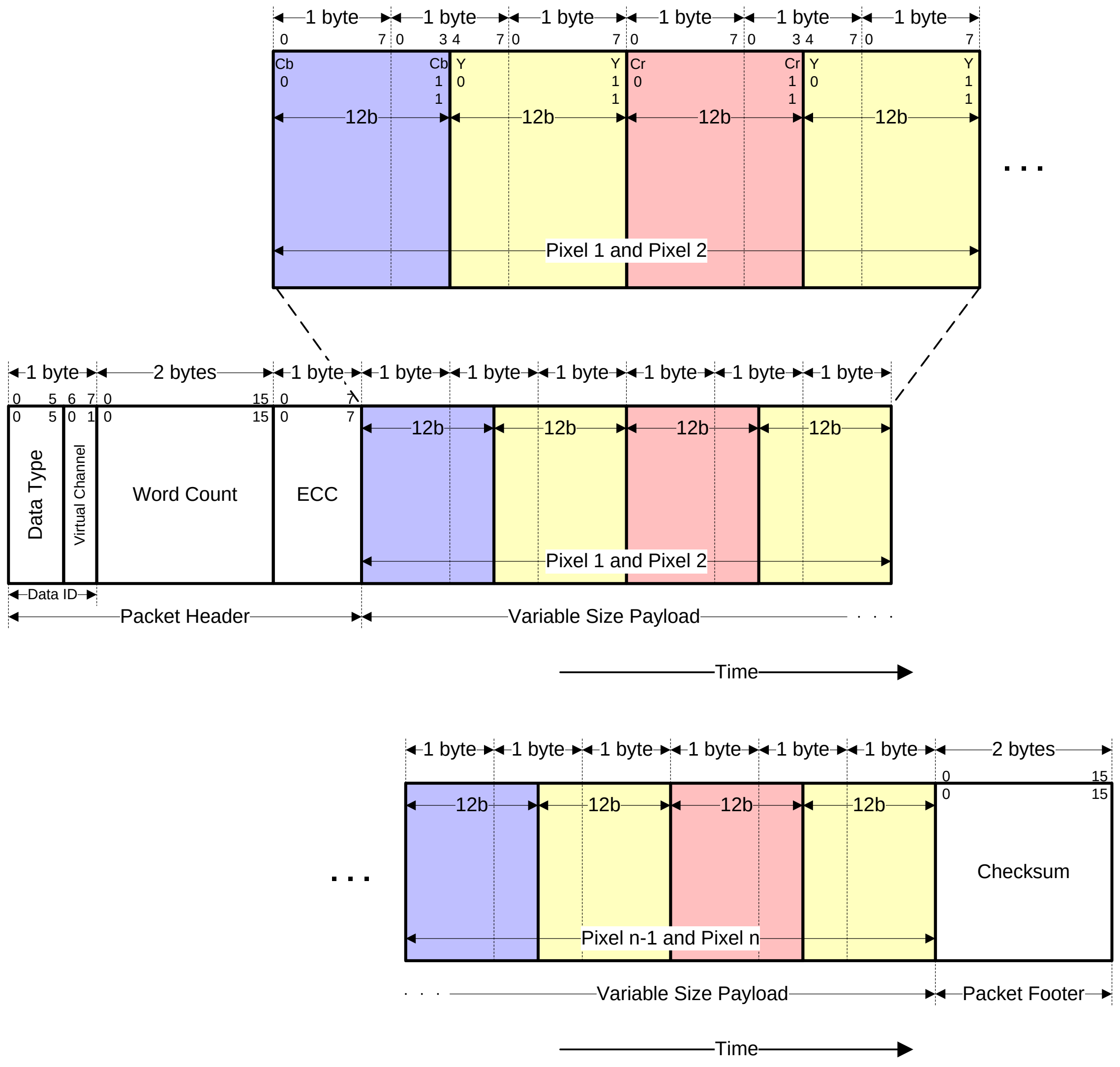


Figure 29 24-bit per Pixel – YCbCr 4:2:2 Format, Long Packet

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

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

Packed Pixel Stream 16-bit YCbCr 4:2:2 Format shown in Figure 30 is a Long packet used to transmit image data formatted as 16-bits per pixel to a Video Mode display module. The packet consists of the DI byte, a two-byte, non-zero WC, an ECC byte, a payload of length WC bytes and a two-byte Checksum.

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

720p60, the pixel format is ITU-R Recommendation BT.709 (see [ITU02]). Component ordering follows ITU-R Recommendation BT.656 (see [ITU03]).

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

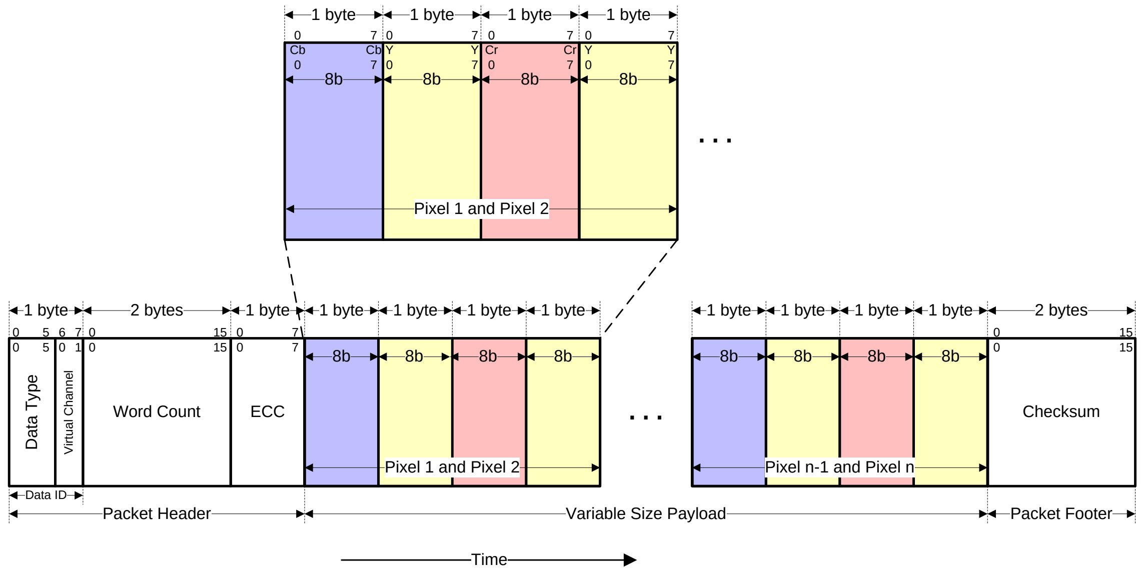


Figure 30 16-bit per Pixel – YCbCr 4:2:2 Format, Long Packet

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

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

Packed Pixel Stream 30-Bit Format shown in Figure 31 is a Long packet used to transmit image data formatted as 30-bit pixels to a Video Mode display module. The packet consists of the DI byte, a two-byte, non-zero WC, an ECC byte, a payload of length WC bytes and a two-byte Checksum. The pixel format is red (10 bits), green (10 bits) and blue (10 bits), in that order. Within a color component, the LSB is sent first, the MSB last.

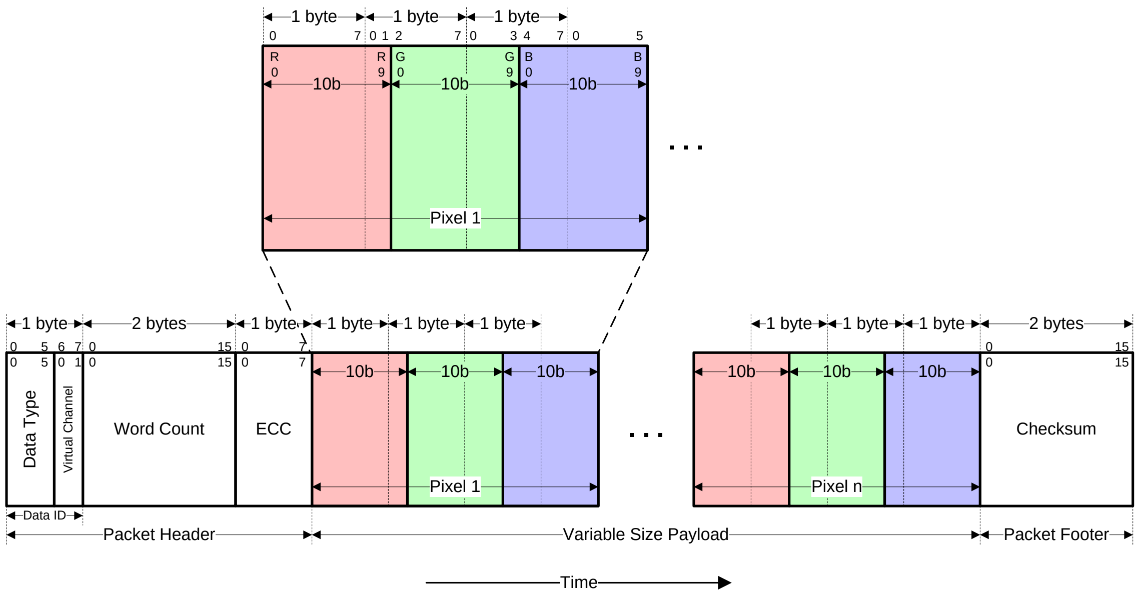


Figure 31 30-bit per Pixel (Packed) – RGB Color Format, Long Packet

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

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

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

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

Packed Pixel Stream 36-Bit Format shown in Figure 32 is a Long packet used to transmit image data formatted as 36-bit pixels to a Video Mode display module. The packet consists of the DI byte, a two-byte, non-zero WC, an ECC byte, a payload of length WC bytes and a two-byte Checksum. The pixel format is red (12 bits), green (12 bits) and blue (12 bits), in that order. Within a color component, the LSB is sent first, the MSB last.

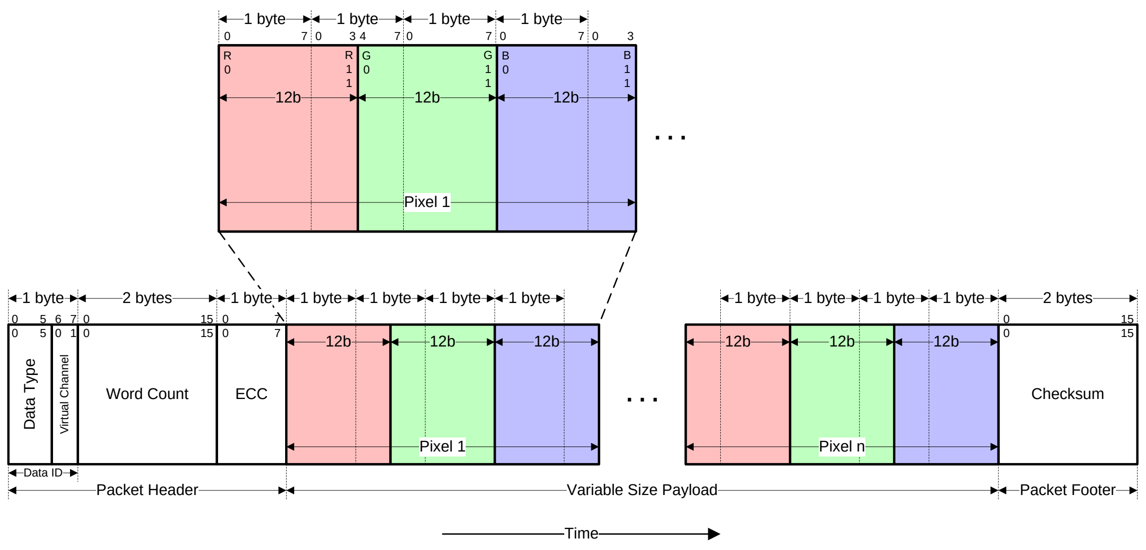


Figure 32 36-bit per Pixel (Packed) – RGB Color Format, Long Packet

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

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

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

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

Packed Pixel Stream 12-bit YCbCr 4:2:0 Format shown in Figure 33 and Figure 34 is a Long packet used to transmit image data formatted as 12-bits per pixel to a Video Mode display module. The packet consists of the DI byte, a two-byte, non-zero WC, an ECC byte, a payload of length WC bytes and a two-byte Checksum.

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

A pixel shall have eight bits for each of the Y-, Cb-, and Cr-components. Within a component, the LSB is sent first, the MSB last. Cb- and Y-components are sent on odd lines as shown in Figure 33 while Cr- and Y-components are sent on even lines as shown in Figure 34.

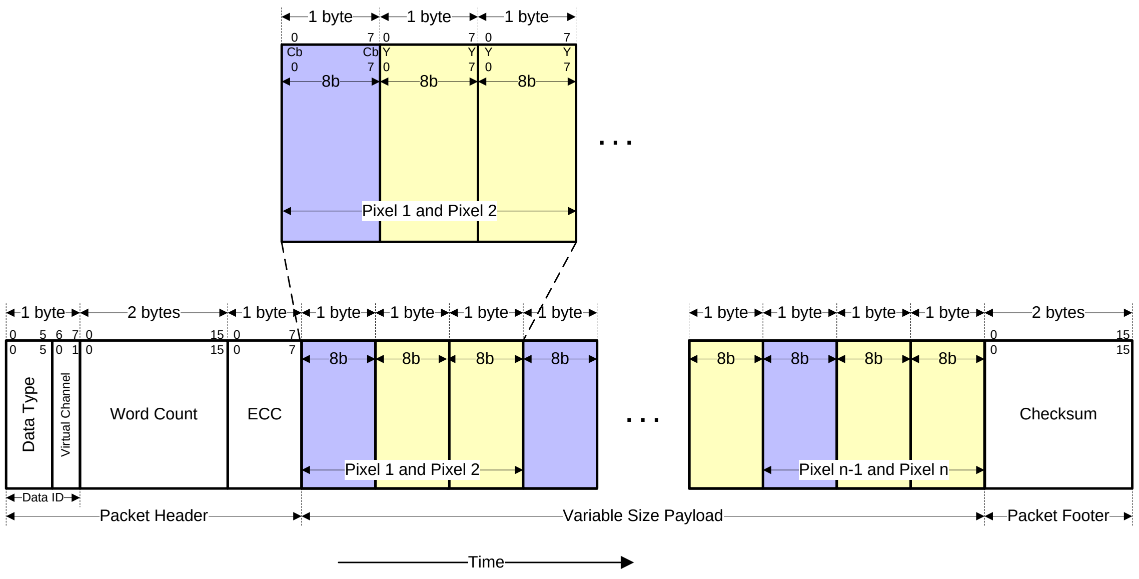


Figure 33 12-bit per Pixel – YCbCr 4:2:0 Format (Odd Line), Long Packet

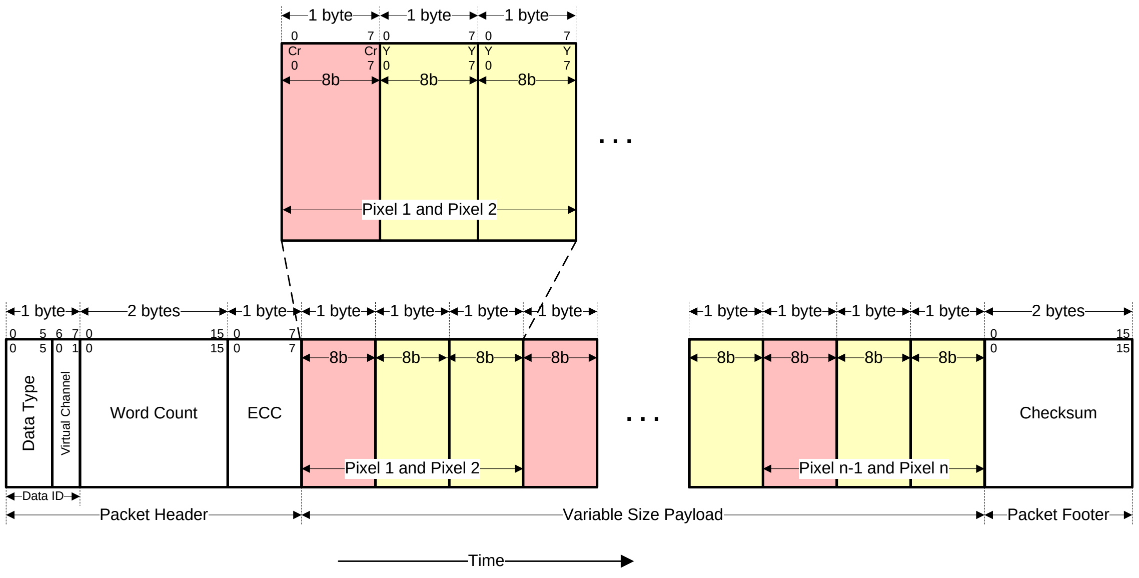


Figure 34 12-bit per Pixel – YCbCr 4:2:0 Format (Even Line), Long Packet

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

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

Packed Pixel Stream 16-Bit Format shown in Figure 35 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.

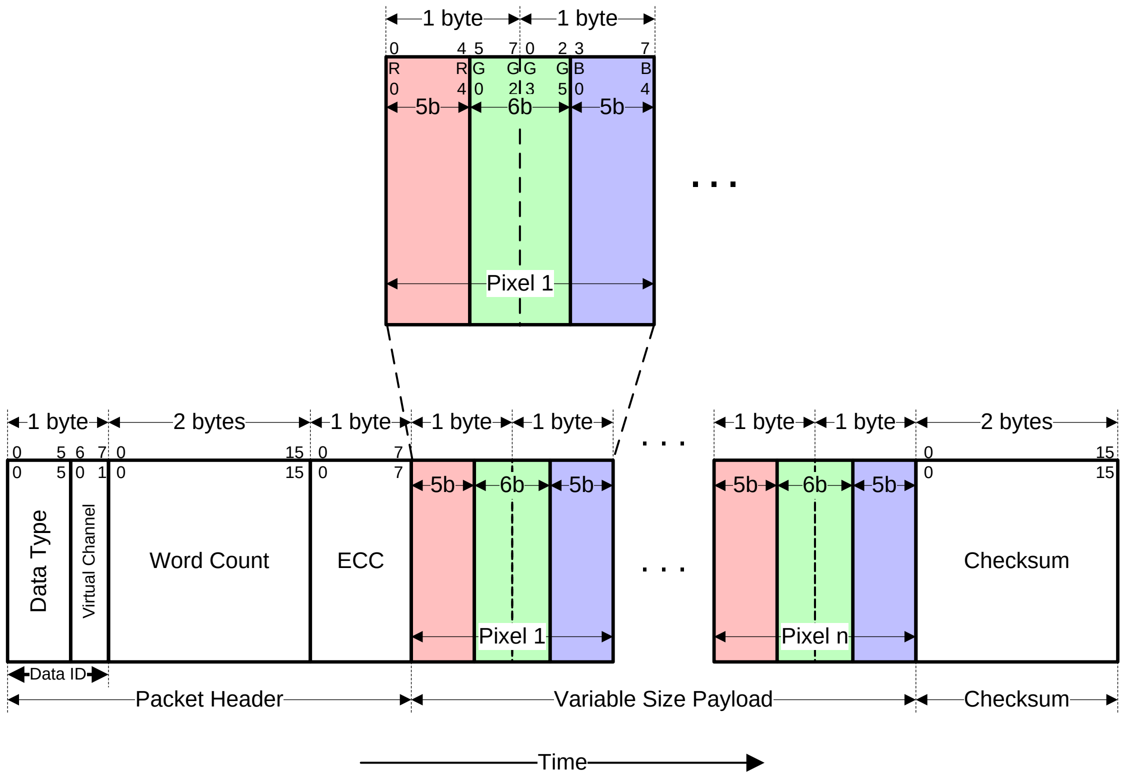


Figure 35 16-bit per Pixel – RGB Color Format, Long Packet

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

Normally, the display module 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.

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

Packed Pixel Stream 18-Bit Format (Packed) shown in Figure 36 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.

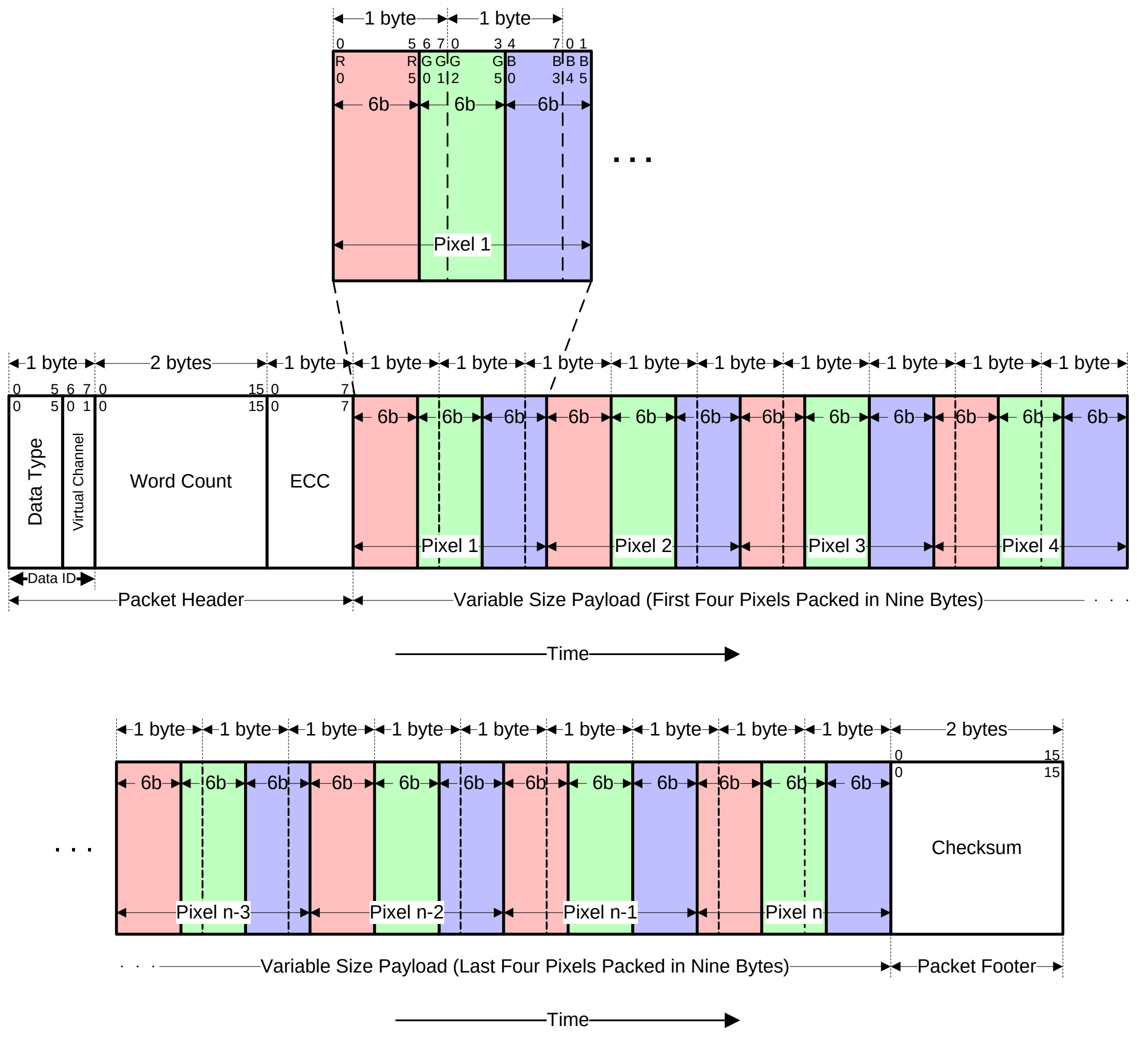


Figure 36 18-bit per Pixel (Packed) – RGB Color Format, Long Packet

Note that pixel boundaries only align 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. If the active (displayed) horizontal width is not a multiple of four pixels, the transmitter shall send additional fill pixels at the end of the display line to make the transmitted width a multiple of four pixels. The receiving peripheral shall not display the fill pixels when refreshing the display device. For example, if a display device has an active display width of 399 pixels, the transmitter should send 400 pixels in one or more packets. The receiver should display the first 399 pixels and discard the last pixel of the transmission.

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

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

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 as shown in

Figure 37. 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.

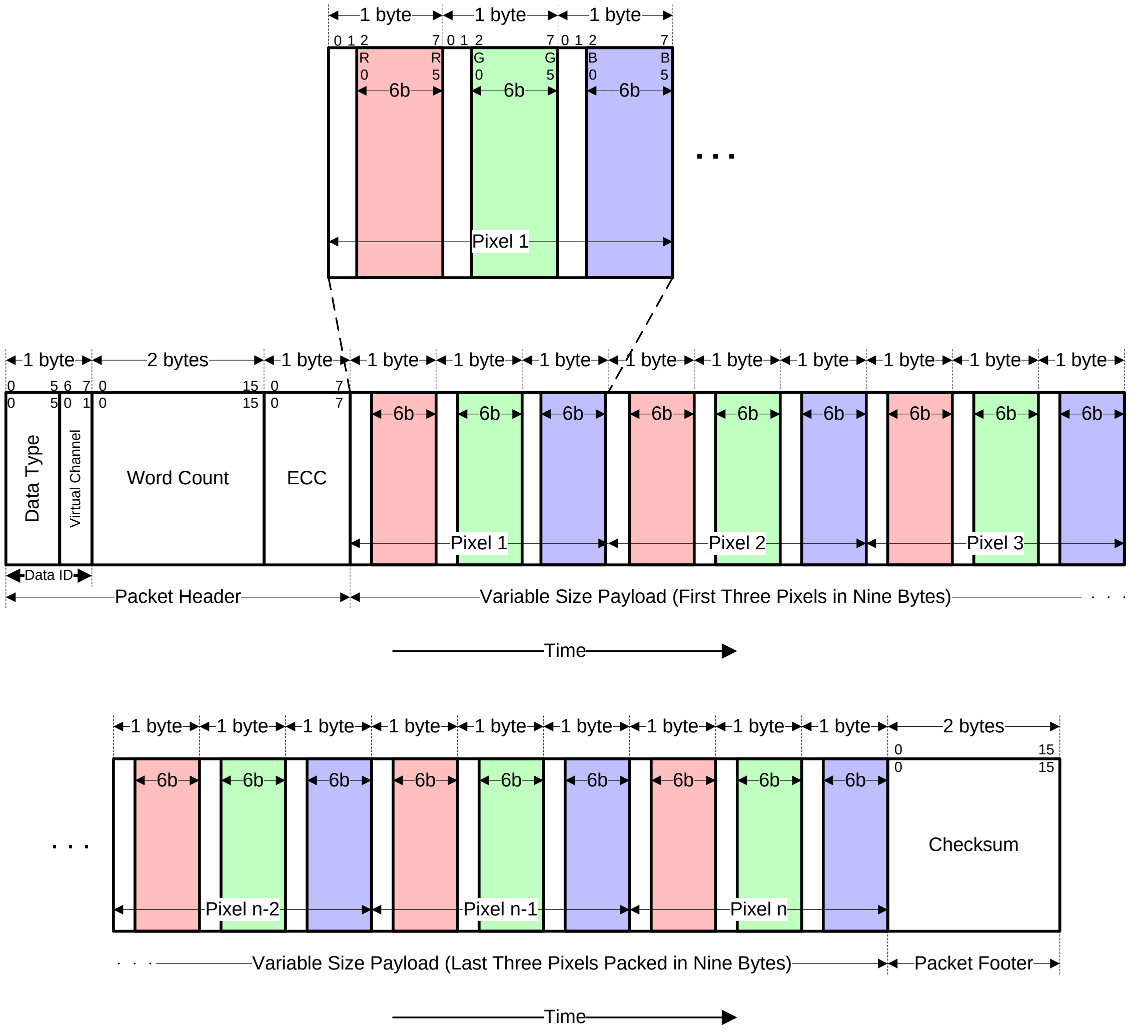


Figure 37 18-bit per Pixel (Loosely Packed) – RGB Color Format, Long Packet

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 align with byte boundaries every three bytes. The total line width (displayed plus non-displayed pixels) should be a multiple of three bytes.

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

Packed Pixel Stream 24-Bit Format shown in

Figure 38 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.

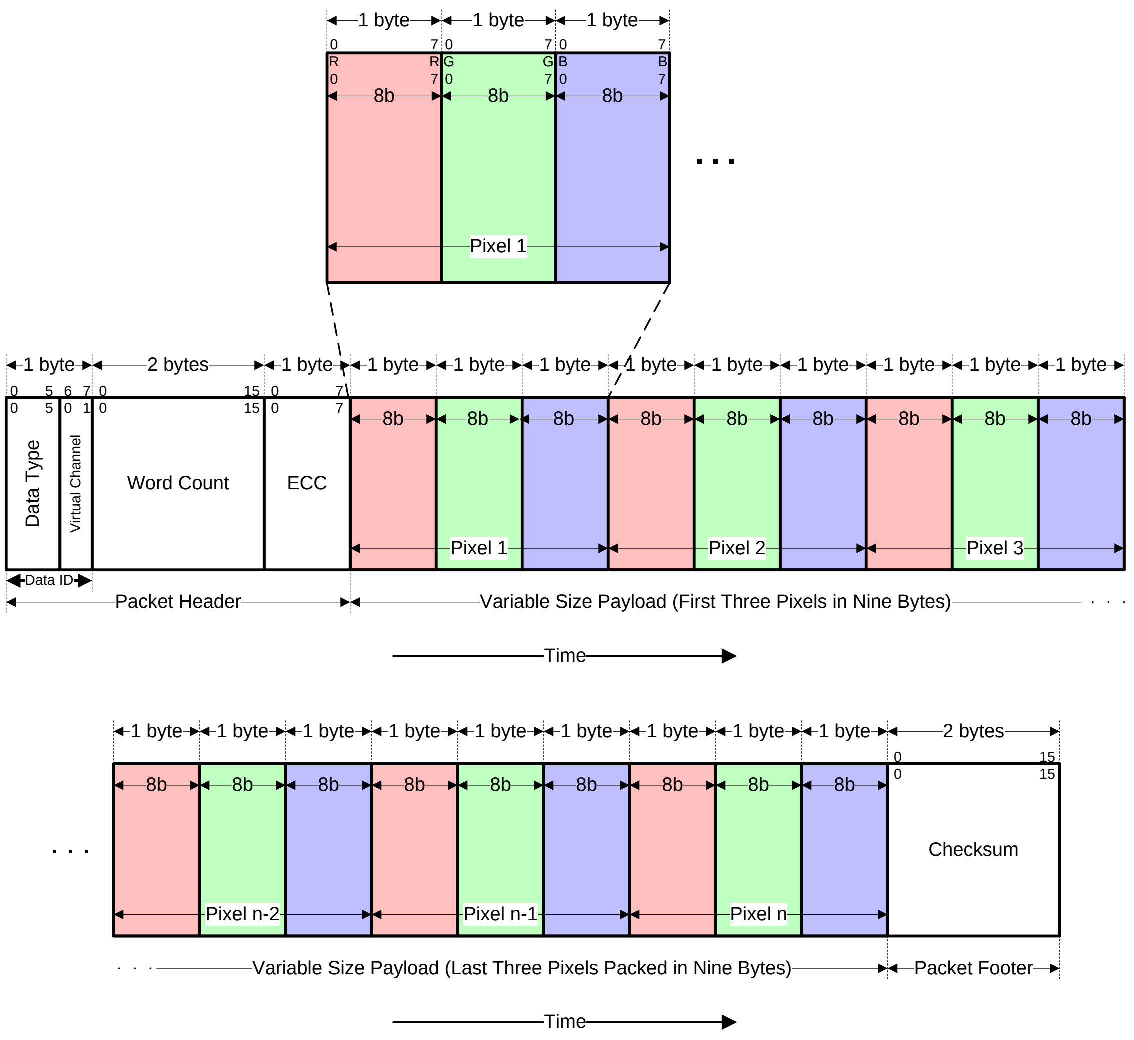


Figure 38 24-bit per Pixel – RGB Color Format, Long Packet

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

8.8.24 Compressed Pixel Stream, Long Packet, Data Type = 00 1011 (0x0B)

The Compressed Pixel Stream is a long packet that carries compressed data to a Video Mode display module. This packet shall consist of a DI byte, a two-byte non-zero WC, an ECC byte, a payload containing WC bytes and a two-byte checksum, shown in Figure 39.

This is an optional packet, but if the pixel data is compressed, this packet shall carry the compressed image data when the Compression Mode packet (Section 8.8.25) has signaled that compression is enabled. The Compressed Pixel Stream's structure shall follow requirements defined in Section 8.13 and the compressed image data shall be an integral number of bytes.

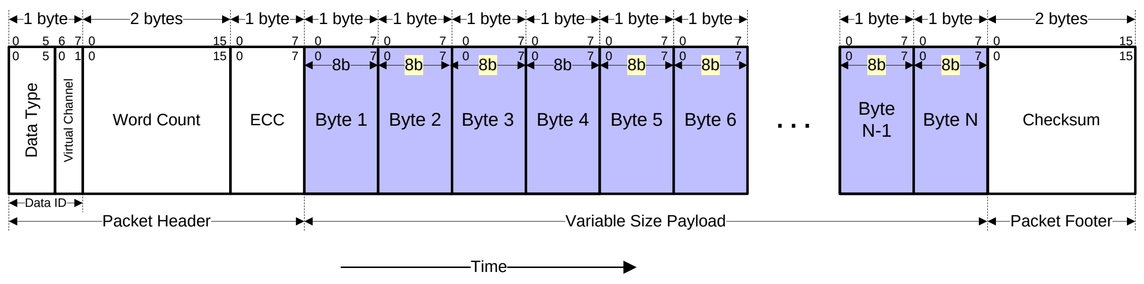


Figure 39 Compressed Pixel Stream Format, Long Packet

A compressed image is composed of data slices (a full slice usually ranges over several lines). The packet carries data from one or more compressed slice segments (called slice-widths) or chunks within one line time period. An annex in this specification can limit the number of slice-widths of data carried in one line time. See Annex D for deploying the coding system in [VESA01].

The following two figures illustrate transporting a single slice-width of data or transporting multiple slice-widths of data in this packet. The slices have the same horizontal size that equates to the same packet size in a constant bit rate coding system. This behavior is defined by the coding system.

For example, if the image is compressed to have one slice-width of data horizontally, Figure 40 (a) shows the portion of slice 0 transported in line time period j is fully contained in one packet. In this case, Figure 40 (a) is the only means to transport this portion of the slice-width of data. If the image is compressed to have more slices horizontally, Figure 40 (b) shows the Compressed Pixel Stream packet contains multiple slice-widths of data.

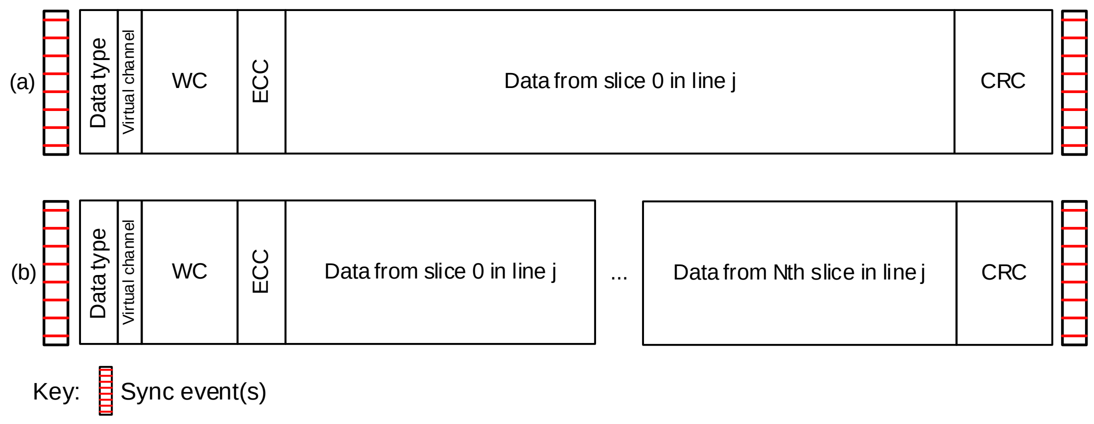

(a)
Data type
Virtual channel
WC
ECC
Data from slice 0 in line j
CRC
(b)
Data from slice 0 in line j
...
Data from Nth slice in line j
Key:
Sync event(s)

Figure 40 One Line Containing One Packet with Data from One or More Compressed Slices

If the image is compressed to have more than one slice horizontally, an integral number of slice-widths of data may be carried by sequential Compressed Pixel Stream packets in one line time j. Figure 41 shows sequential packets that can contain slice-widths of data. This is one method that can segregate slice-widths when the receiver has multiple instantiations of decoders and this packet structure is used to identify slice-width boundaries.

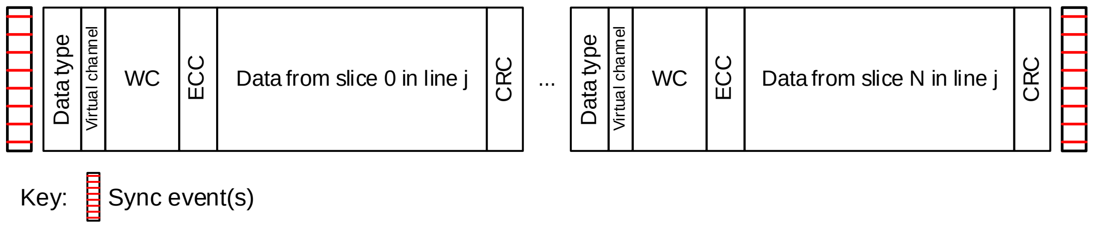


Figure 41 One Line Containing More than One Compressed Pixel Stream Packet

The Compressed Pixel Stream does not limit usage to any specific MIPI-compliant bitstream or standard. This packet shall only carry compressed image data with any pixel arrangement or component depth supported by the decoder.

8.8.25 Compression Mode Command, Data Type = 00 0111 (0x07)

Some display stream compression parameters may be configured using the Compression Mode Command, which is a short packet consisting of a DI byte, a two-byte payload and an ECC byte.

This packet signals whether compression is enabled or disabled and the coding system used to create a bitstream or codestream that is carried by the Compressed Pixel Stream data type transaction. See Section 8.8.24.

This data type writes the compression mode parameters listed in Table 20 to the peripheral in advance of the codestream with timing dependency or event dependency defined by requirements in Section 8.13.4.

Table 20 Compression Mode Parameters[1]

SP Byte	Bit Location	Bit Description And Assigned Values
Data 0	7:6	Reserved, bits equal 0
Data 0	5:4	PPS selector 00 = PPS Table 1 (or no tables stored, default reset value) 01 = PPS Table 2 10 = PPS Table 3 11 = PPS Table 4
Data 0	3	Reserved
Data 0	2:1	Algorithm identifier 00 = VESA DSC Standard 1.1 11 = vendor-specific algorithm 01, 10 = reserved, not used
Data 0	0	0 = compression disabled (default) 1 = compression enabled
Data 1	7:0	Reserved, bits equal 0

1. *Applies to video mode or command mode. In command mode, all bits are readable and writable, except reserved bits that use a defined value or are disallowed.*

The Command mode contains a two-bit PPS Selector that may be used to enable a pre-stored PPS Table for controlling the compression decoder parameters. The processor sends this data type to change the decoder parameters that shall take effect following the next vertical sync or internal vertical sync (if using command mode). The PPS Selector is an optional field; if no table is stored in the receiver, the selector shall be 00b.

8.8.26 Picture Parameter Set (0x0A)

The Picture Parameter Set (PPS) data type is a long packet used to transmit a pre-defined set of parameters that control a compression coding system. The packet shall consist of a DI byte, a two-byte, non-zero WC, an ECC byte, a payload containing WC bytes, and a two-byte checksum.

The size, content and timing context of this long packet is defined by the coding system designated in the Compression Mode data type, Section 8.8.25, and transported in the Compressed Image Format data type, Section 8.8.24. Using this long packet, for example, may replace adding header markers to a bitstream.

When transporting the DSC bitstream, refer to guidelines and normative requirements in Annex D.

If the peripheral receiver supports multiple, stored PPS tables, the received new PPS data is stored in the table designated by the Compression Mode PPS Selector if the table is writable.

8.8.27 Execute Queue (0x16)

Execute Queue is a short packet sent to one display driver that controls and synchronizes other display drivers in a display module to execute frame synchronized commands (FSC) stored in an execution queue. Each display driver within a display module may have stored one or more commands in an execution queue.

The display driver receiving the Execute Queue shall:

1. Synchronize display drivers to execute queued FSC coincident with the next vertical sync pulse or event, if using video mode.

2. Synchronize display drivers to execute queued FSC coincident with the next tearing effect internally processed by the display driver when in command mode.

The display driver receiving the Execute Queue contains a mechanism to synchronize each display driver to initiate processing of the queued commands. The means to synchronize separate display drivers is not within scope of this specification.

The values of short packet data bytes 0 and 1 are unspecified and ignored by the peripheral.

Execute Queue sent to DDICs other than the master DDIC are allowed but are ignored.

8.8.28 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 shall 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.

Command Mode systems that use DCS shall have a bidirectional data path. Short packets and the header of Long packets shall use ECC and may use 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 shall be calculated to cover the Packet Header data. 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 optional. If the Checksum is not calculated by the peripheral the Packet Footer shall be 0x0000.

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 four basic types:

- *Tearing Effect (TE)* is a Trigger message sent to convey display timing information to the host processor. *Trigger* messages are single byte packets sent by a peripheral's PHY layer in response to a signal from the DSI protocol layer. See [MIPI04] for a description of Trigger messages.
- *Acknowledge* is a Trigger Message sent when the current transmission, as well as all preceding transmissions since the last peripheral to host communication, i.e. either triggers or packets, is received by the peripheral with no errors.
- *Acknowledge and Error Report* is a Short packet sent if any errors were detected in preceding transmissions from the host processor. Once reported, accumulated errors in the error register are cleared.
- *Response to Read Request* may be a Short or Long packet that returns data requested by the preceding READ command from the processor.

8.9.2 System Requirements for ECC and Checksum and Packet Format

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

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

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

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

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

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 shall provide ECC support 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 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, the peripheral shall respond with *Acknowledge* if no errors were detected and stored since the last peripheral to host communication, i.e. either triggers or packets.
- Following a Read request, the peripheral shall send the requested READ data if no errors were detected and stored since the last peripheral to host communication, i.e. either triggers or packets.
- Following a Read request if only 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 and Error Report* packet in the same LP transmission. The Error Report shall have the *ECC Error – Single Bit* flag set, as well as any error bits from any preceding transmissions stored since the last peripheral to host communication.
- Following a non-Read command if only 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 and Error Report* packet. The Error Report shall have the *ECC Error – Single Bit* flag set, as well as any error bits from any preceding transmissions stored since the last peripheral to host communication.
- Following a Read request, if multi-bit ECC errors were detected and not corrected, the peripheral shall send a 4-byte *Acknowledge and Error Report* packet without sending Read data. The Error Report shall have the *ECC Error – Multi-Bit* flag set, as well as any error bits from any preceding transmissions stored since the last peripheral to host communication.

- Following a non-Read command, if multi-bit ECC errors were detected and not corrected, the peripheral shall not execute the command, and shall send a 4-byte *Acknowledge and Error Report* packet. The Error Report shall have the *ECC Error – Multi-Bit* flag set, as well as any error bits from any preceding transmissions stored since the last peripheral to host communication.
- Following any command, if *SoT Error*, *SoT Sync Error* or *DSI VC ID Invalid* or DSI protocol violation was detected, or the DSI command was not recognized, the peripheral shall send a 4-byte *Acknowledge and Error Report* response, with the appropriate error flags set, as well as any error bits from any preceding transmissions stored since the last peripheral to host communication, in the two-byte error field. Only the *Acknowledge and Error Report* packet shall be transmitted; no read or write accesses shall take place on the peripheral in response.
- Following any command, if *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 and Error Report* packet with the appropriate error flags set, as well as any error bits from any preceding transmissions stored since the last peripheral to host communication. For a read command, only the *Acknowledge and Error Report* packet shall be transmitted; no read data shall be sent by the peripheral in response.

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

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

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

Acknowledge and Error Report confirms that the preceding command or data sent from the host processor to a peripheral was received, and indicates what types of error were detected on the transmission and any preceding transmissions. Note that if errors accumulate from multiple preceding transmissions, it may be difficult or impossible to identify which transmission contained the error. This message 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 the header

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

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

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, 2: READ data, may be one or two bytes. For single byte parameters, the parameter shall be returned in Byte 1 and Byte 2 shall be set to 0x00.
- ECC byte covering the header

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 the header
- 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 0x0000

8.9.5 Error Reporting Format

An error report is a Short packet 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 setting the corresponding bit to “1”. Table 21 shows the bit assignment for all error reporting.

Table 21 Error Report Bit Definitions

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

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

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

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

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

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

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

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

8.10 Peripheral-to-Processor Transactions – Detailed Format Description

Table 22 presents the complete set of peripheral-to-processor Data Types.

Table 22 Data Types for Peripheral-Sourced Packets

Data Type, hex	Data Type, binary	Description	Packet Size
0x00 – 0x01	00 000X	Reserved	Short
0x02	00 0010	Acknowledge and Error Report	Short
0x03 – 0x07	00 0011 – 00 0111	Reserved	
0x08	00 1000	End of Transmission packet (EoTp)	Short
0x09 – 0x10	00 1001 – 01 0000	Reserved	
0x11	01 0001	Generic Short READ Response, 1 byte returned	Short
0x12	01 0010	Generic Short READ Response, 2 bytes returned	Short
0x13 – 0x19	01 0011 – 01 1001	Reserved	

Data Type, hex	Data Type, binary	Description	Packet Size
0x1A	01 1010	Generic Long READ Response	Long
0x1B	01 1011	Reserved	
0x1C	01 1100	DCS Long READ Response	Long
0x1D – 0x20	01 1101 – 10 0000	Reserved	
0x21	10 0001	DCS Short READ Response, 1 byte returned	Short
0x22	10 0010	DCS Short READ Response, 2 bytes returned	Short
0x23 – 0x3F	10 0011 – 11 1111	Reserved	

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

Acknowledge and Error Report is sent in response to any command, or read request, with BTA asserted when a reportable error is detected in the preceding, or earlier, 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.

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

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

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

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

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 document.

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 and only the *Acknowledge and Error Report* packet shall be sent.

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

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 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 0x0000.

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 and only the *Acknowledge and Error Report* packet shall be sent.

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

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 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 0x0000.

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 and only the *Acknowledge and Error Report* packet shall be sent.

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

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

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 and only the *Acknowledge and Error Report* packet shall be sent.

8.10.6 Multiple Transmissions and Error Reporting

A peripheral shall report all errors documented in Table 21, when a command or request is followed by BTA giving bus possession to the peripheral. Peripheral shall accumulate errors from multiple transactions up until a time that host is issuing a BTA. After that, only one ACK Trigger Message or *Acknowledge and Error Report* packet shall be returned regardless of the number of packets or transmissions. Notice that host may not be able to associate each error to a particular packet or transmission causing that error.

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

8.10.7 Clearing Error Bits

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

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 Transmission Packet Sequences

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

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

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

The host processor shall support all of the packet sequences in this section. A Video Mode peripheral shall support at least one of the packet sequences in this section. The peripheral shall not require any additional constraints regarding packet sequence or packet timing. The peripheral supplier shall document all relevant timing parameters listed in Table 23.

In the following figures the Blanking or Low-Power Interval (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. The host processor should 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.

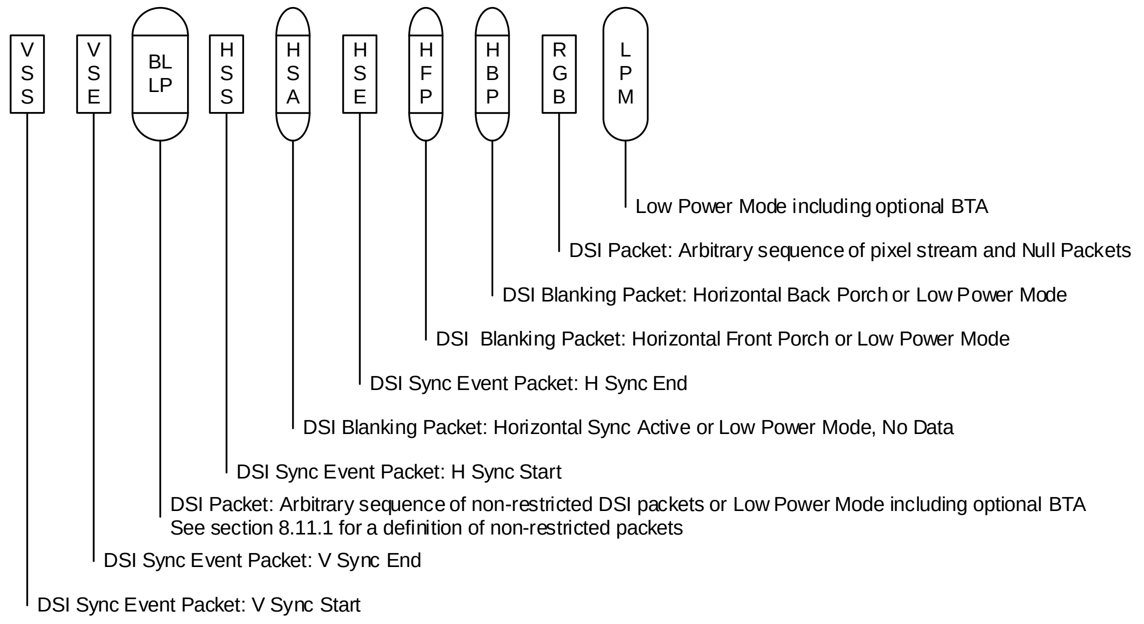
During the BLLP the DSI Link may do any of the following:

- 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 from the host processor to a different peripheral using a different Virtual Channel ID

The sequence of packets within the BLLP or RGB portion of a HS transmission 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 VSS; all other lines shall start with VSE or HSS. Note that the position of synchronization packets, such as VSS and HSS, in time is of utmost importance since this has a direct impact on the visual performance of the display panel.

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

Transmission packet components used in the figures in this section are defined in Figure 42 unless otherwise specified.



VSS
VSE
BLLP
HSS
HSA
HSE
HFP
HBP
RGB
LPM
Low Power Mode including optional BTA
DSI Packet: Arbitrary sequence of pixel stream and Null Packets
DSI Blanking Packet: Horizontal Back Porch or Low Power Mode
DSI Blanking Packet: Horizontal Front Porch or Low Power Mode
DSI Sync Event Packet: H Sync End
DSI Blanking Packet: Horizontal Sync Active or Low Power Mode, No Data
DSI Sync Event Packet: H Sync Start
DSI Packet: Arbitrary sequence of non-restricted DSI packets or Low Power Mode including optional BTA
See section 8.11.1 for a definition of non-restricted packets
DSI Sync Event Packet: V Sync End
DSI Sync Event Packet: V Sync Start

Figure 42 Video Mode Interface Timing Legend

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

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 43.

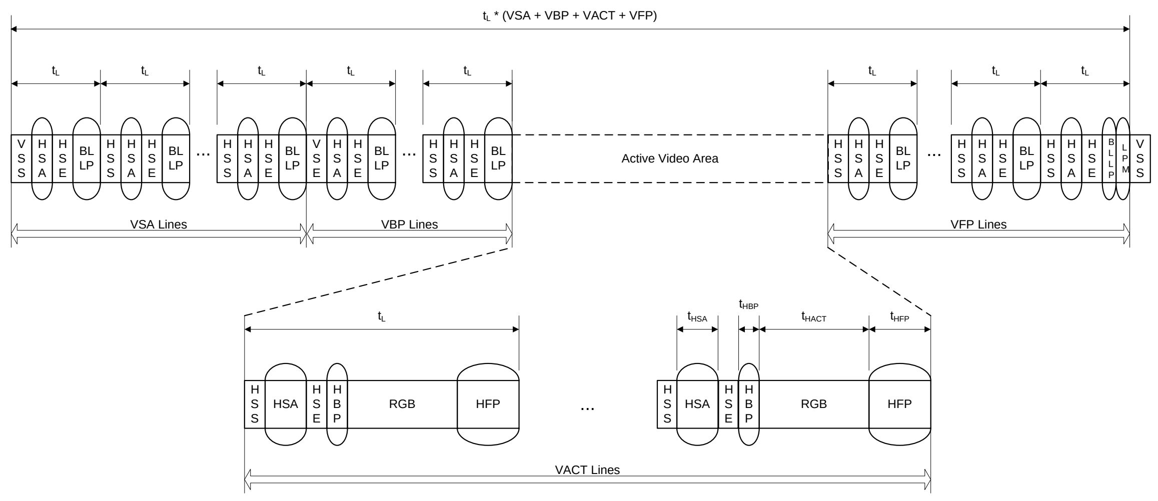


Figure 43 Video Mode Interface Timing: Non-Burst Transmission with Sync Start and End

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

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

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 44.

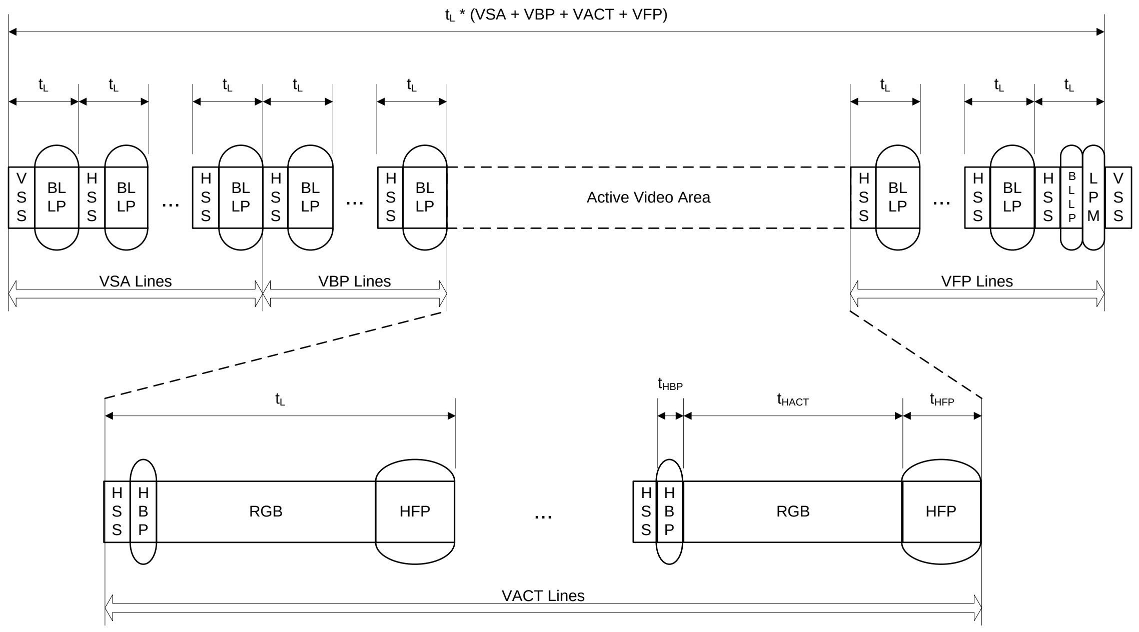


Figure 44 Video Mode Interface Timing: Non-Burst Transmission with Sync Events

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

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

8.11.4 Burst Mode

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

There may be a line buffer or similar memory on the peripheral to accommodate incoming data at high speed. Following HS pixel data transmission, the bus may stay in HS Mode for sending blanking packets or go to Low Power Mode, during which it may remain idle, i.e. the host processor remains in LP-11 state, or

LP transmission may take place in either direction. If the peripheral takes control of the bus for sending data to the host processor, its transmission time shall be limited to ensure data underflow does not occur from its internal buffer memory to the display device. An example of this mode is shown in Figure 45.

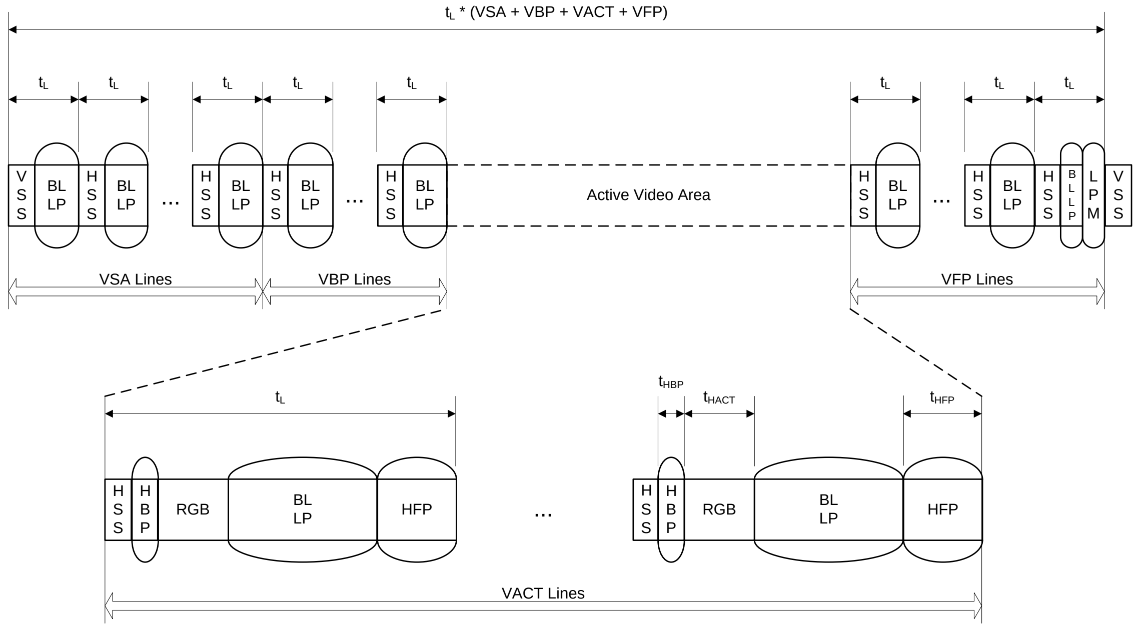


Figure 45 Video Mode Interface Timing: Burst Transmission

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

8.11.5 Parameters

Table 23 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 [MIPI04].

Table 23 Required Peripheral Timing Parameters

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

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

8.12 TE Signaling in DSI

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

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

For polling to the display module, the host processor shall detect the current scan line information with a DCS command such as get_scan_line to avoid Tearing Effects. For TE-reporting from the display module, the TE-reporting function is enabled and disabled by three DCS commands to the display module's controller: set_tear_on, set_tear_scanline, and set_tear_off. See [MIPI01] for details.

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

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

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

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

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

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

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

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

8.13 DSI with Display Stream Compression

8.13.1 Compression Transport Requirements

The following Sections 8.13.1 through 8.13.5 shall apply when DSI carries bitstreams. Compressed pixel data shall be decoded in the Peripheral. The Peripheral may store the data from a bitstream in a local frame buffer before decoding.

The pixel data coding shall transmit a fixed number of bits over a slice that contains compressed pixel data. The following relationship shall be true over a slice used by the coding system. The size of the slice, in horizontal and vertical dimensions determines how many pixels are in each slice. This specification shall use a bits per pixel compression factor.

The following relationship shall be true for each slice in a DSI frame.

$$\text{bits / slice} = \text{bits / pixel} \times \text{pixels / slice}$$

The bits per slice shall be an integral number of bytes guaranteed by choosing a bits per slice value supported by a compliant codestream such that the above equation is met. Therefore, the bits per frame is an integral number of bytes since one frame is the largest possible slice.

8.13.2 Transport Buffer Model (Informative)

The decoder and encoder may require buffers to ensure real-time operation. The decoder manufacturer defines supported upper and lower boundary sizes for a slice. The coding system encodes and decodes, in real time, all content with no underflow or overflow. Refer to the coding system for guidelines regarding buffering required by the coding system as a result of ensuring real time operation without underflow or overflow. Additional buffering between the transport layer and the coding system is out of scope of the DSI specification.

If the coding system creates a constant bit rate (CBR) codestream, the transport layer can predict the rate of pixels in order to fill the codestream data format long packets. The Compressed Pixel Stream (Section 8.8.24) long packet contains as many bytes of the codestream as determined in the application requirements. For example, video mode requires timely insertion of HSS packets.

If the coding system creates a variable bit rate (VBR) codestream, the transport layer may need a mechanism to either fill a Compressed Pixel Stream long packet or include a transport buffer that ensures the long packet contains the exact number of bytes defined in the long packet header. Whatever method is used to fill long packets with a VBR codestream is outside the scope of this specification.

8.13.3 Compression with Video Modes

The compressed stream may use any of the video mode interface timings specified in Section 8.11: non-burst mode with sync pulses; non-burst mode with sync events; or burst mode. Display stream compression transmits compressed pixel data that shall be decoded in the peripheral.

The video mode interface timing diagrams in Sections 8.11.2, 8.11.3 and 8.11.4 apply to display stream compression. "RGB" shown in Figure 41 to Figure 44, respectively, shall represent compressed pixel data.

The burst mode that also carries codestreams acts identically to video burst mode with uncompressed data. There is an HBP and HFP based on the manufacturers data sheet.

8.13.4 Compression-Related Parameters

The coding system shall define a picture parameter set comprising all parameters that are required to create and interpret a codestream. The device manufacturer may define a mechanism that allows the PPS to be updated on a frame-by-frame basis. The coding system standard defines the PPS and any synchronization required between the PPS changes and those changes affecting the codestream.

Device manufacturers shall specify in a product data sheet values for parameters in Table 24.

Table 24 Required Peripheral Parameters for Compression

Parameter	Description	Minimum	Maximum	Units	Comments
	Slice horizontal size range			Frame width, either in number of pixels or percentage	Data sheet best practice: be clear about what combinations are supported
	Slice vertical size range			Lines	
	Slice area			Pixels	
	Line buffer			Kilobytes	Codec only
	Rate buffer			Kilobytes	Codec only
	Transport buffer			Kilobytes	
	Compression value range			bpp	
	Compression resolution range			bpp	

8.13.5 Display Stream Compression with Command Mode

This section shall apply when DSI carries codestreams and operates in command mode. In addition, sections Compression Mode (Section 8.8.25) and Compression-related parameters (Section 8.13.4) shall apply also with compression scheme in command mode.

Display devices operating in command mode with frame memory can optionally have a decoder implemented for display stream compression purposes. In such case, the Compression Mode scheme is enabled through the Compression Mode data type (Section 8.8.25). When Compression Mode status is enabled, the display device shall treat all incoming pixel data, which is written to the frame memory, as a compressed codestream.

Display Command Set [MIPI01] describes display stream compression transport in command mode for Architecture Type 1 display devices.

9 Error-Correcting Code (ECC) and Checksum

9.1 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. The ECC byte shall provide single-bit error correction and 2-bit error detection for the entire Packet Header. See Figure 22 and Figure 23 for descriptions of the Long and Short Packet Headers, respectively.

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

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

9.2 Hamming Code Theory

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).

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 $\mathbf{A}^T$.

9.3 Hamming-Modified Code Applied to DSI Packet Headers

Hamming codes use parity to correct a single-bit error or detect a two-bit error, but are not capable of doing both simultaneously. DSI uses Hamming-modified codes where an extra parity bit is used to support both single error correction as well as two-bit error detection. For example a 7+1 bit Hamming-modified code

(72, 64) allows for protection of up to 64 data bits. DSI systems shall use a 5+1 bit Hamming-modified code (30, 24), allowing for protection of up to twenty-four data bits. The addition of a parity bit allows a five bit Hamming code to correct a single-bit error and detect a two-bit error simultaneously.

Since Packet Headers are fixed at four bytes (twenty-four data bits and eight ECC bits), P6 and P7 of the ECC byte are unused and shall be set to zero by the transmitter. The receiver shall ignore P6 and P7 and set both bits to zero before processing ECC. Table 25 shows a compact way to specify the encoding of parity and decoding of syndromes.

Table 25 ECC Syndrome Association Matrix

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

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. 38th bit position (D37) 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:

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

Table 26 represents the same information as in Table 25, organized to provide better insight into how parity bits are formed from data bits.

Table 26 ECC Parity Generation Rules

Data Bit	P7	P6	P5	P4	P3	P2	P1	P0	Hex
0	0	0	0	0	0	1	1	1	0x07
1	0	0	0	0	1	0	1	1	0x0B
2	0	0	0	0	1	1	0	1	0x0D
3	0	0	0	0	1	1	1	0	0x0E
4	0	0	0	1	0	0	1	1	0x13
5	0	0	0	1	0	1	0	1	0x15

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

Data Bit	P7	P6	P5	P4	P3	P2	P1	P0	Hex
43	1	0	0	0	1	0	1	0	0x8A
44	0	0	1	1	1	1	0	1	0x3D
45	0	0	1	1	1	1	1	0	0x3E
46	0	1	0	0	1	1	1	1	0x4F
47	0	1	0	1	0	1	1	1	0x57
48	1	0	0	0	1	1	0	0	0x8C
49	1	0	0	1	0	0	0	1	0x91
50	1	0	0	1	0	0	1	0	0x92
51	1	0	0	1	0	1	0	0	0x94
52	1	0	0	1	1	0	0	0	0x98
53	1	0	1	0	0	0	0	1	0xA1
54	1	0	1	0	0	0	1	0	0xA2
55	1	0	1	0	0	1	0	0	0xA4
56	1	0	1	0	1	0	0	0	0xA8
57	1	0	1	1	0	0	0	0	0xB0
58	1	1	0	0	0	0	0	1	0xC1
59	1	1	0	0	0	0	1	0	0xC2
60	1	1	0	0	0	1	0	0	0xC4
61	1	1	0	0	1	0	0	0	0xC8
62	1	1	0	1	0	0	0	0	0xD0
63	1	1	1	0	0	0	0	0	0xE0

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

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

The first twenty-four data bits, D0 to D23, in Table 26 contain the complete DSI Packet Header. The remaining bits, D24 to D63, are informative (shown in yellow in the table) and not relevant to DSI. Therefore, the parity bit calculation can be optimized to:

P7=0

P6=0

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

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

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

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

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

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

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

9.4 ECC Generation on the Transmitter

ECC is generated from the twenty-four data bits within the Packet Header as illustrated in Figure 46, which also serves as an ECC calculation example. Note that the DSI protocol uses a four byte Packet Header. See Section 8.4.1 and Section 8.4.2 for Packet Header descriptions for Long and Short packets, respectively.

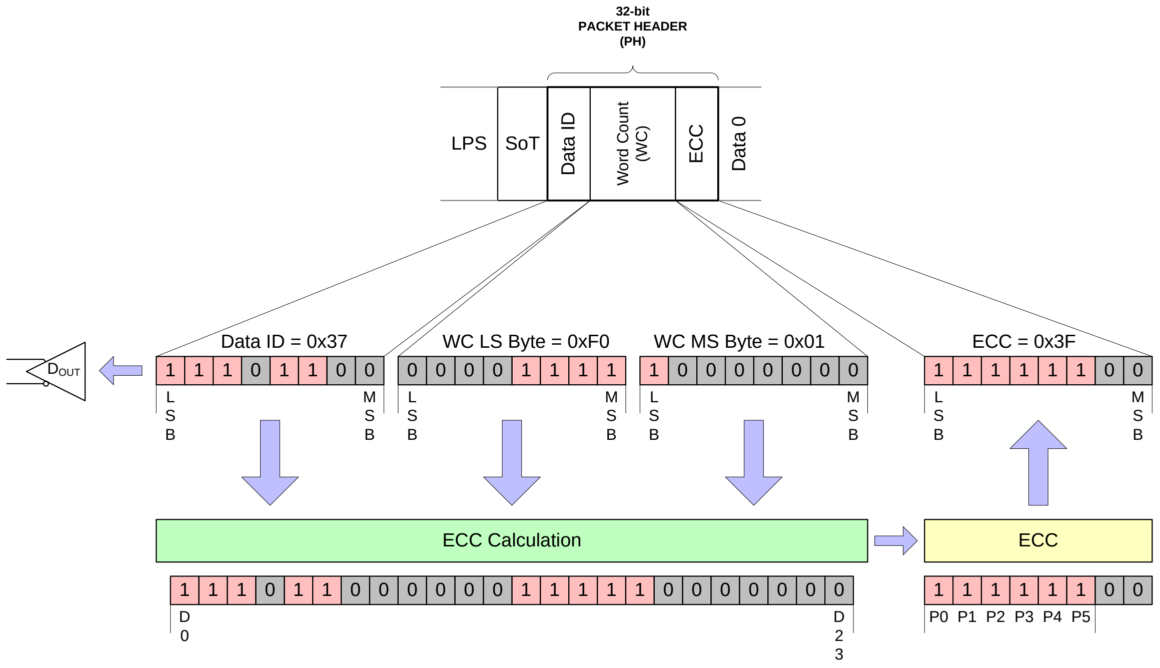


Figure 46 24-bit ECC Generation on TX side

9.5 Applying ECC on the Receiver

Applying ECC on the receiver 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, correcting the error. If a multiple-bit error is identified, it is flagged and reported to the transmitter. Note, error reporting is only applicable to bidirectional DSI implementations.

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

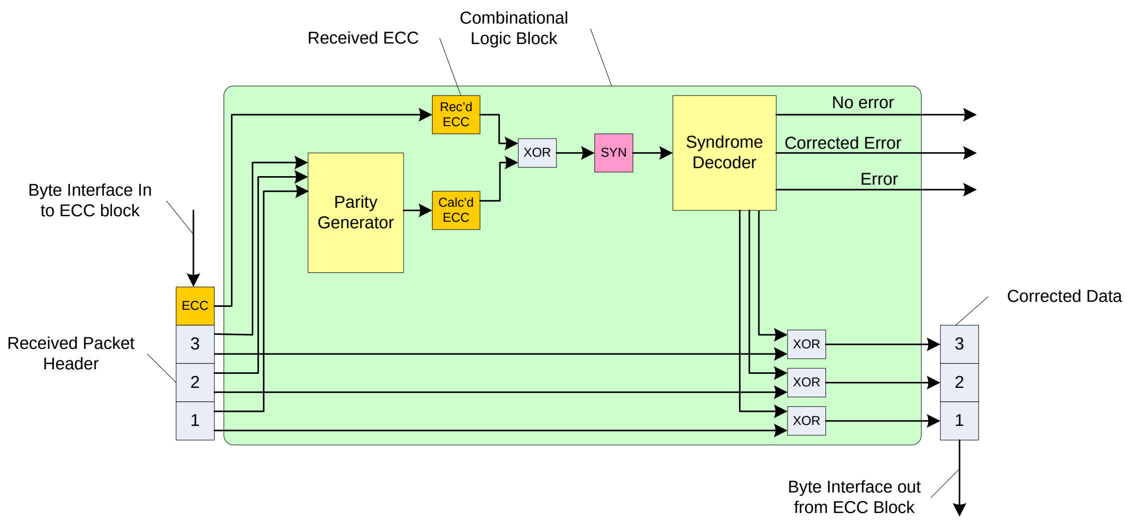


Figure 47 24-bit ECC on RX Side Including Error Correction

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 25. 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.

9.6 Checksum Generation for Long Packet Payloads

Long packets are comprised of a Packet Header protected by an ECC byte as specified in Section 9.3 through Section 9.5, 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 a zero-length payload, the 2-byte checksum is set to 0xFFFF.

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 0x0000 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 shall be realized as a 16-bit CRC with a generator polynomial of x^16+x^12+x^5+x^0.

The transmission of the checksum is illustrated in Figure 48. The LS byte is sent first, followed by the MS byte. Note that within the byte, the LS bit is sent first.

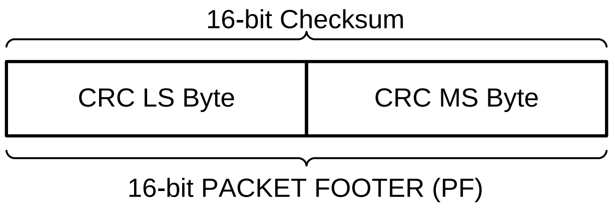


Figure 48 Checksum Transmission

The CRC implementation is presented in Figure 49. The CRC shift register shall be initialized to 0xFFFF before packet data enters. Packet data not including the Packet Header then enters as a bitwise data stream from the left, LS bit first. Each bit is fed through the CRC shift register before it is passed to the output for transmission to the peripheral. After all bytes in the packet payload have passed through the CRC shift register, the shift register contains the checksum. C15 contains the checksum's MSB and C0 the LSB of the 16-bit checksum. The checksum is then appended to the data stream and sent to the receiver. The receiver uses its own generated CRC to verify that no errors have occurred in transmission. See Annex B for an example of checksum generation.

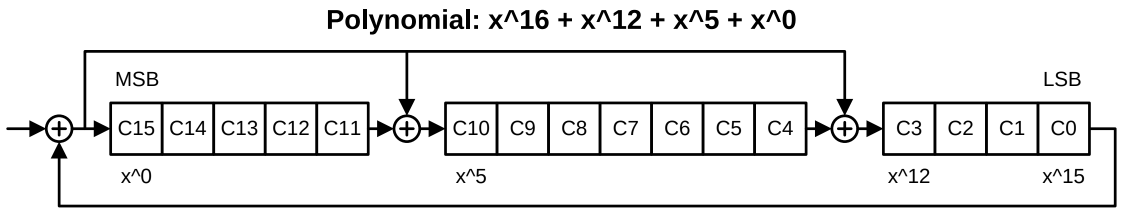


Figure 49 16-bit CRC Generation Using a Shift Register

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

9.7 Checksum Generation for Reconstructed Image Test Mode

A checksum can be calculated over a complete frame or image of pixels reconstructed from a bitstream. This checksum is accessible through the DCS command, get_image_checksum_ct or get_image_checksum_rgb, described in [MIPI01]. The pixel values of a reconstructed image can be predicted, therefore this section describes requirements to create a checksum and verify the compliant operation of a decoder or decoders in a display module.

A display driver or display module shall contain the described checksum when using a decoder compliant with this specification and the DSI receiver supports bus-turn-around features for reading a DCS register (see [MIPI01]).

The checksum shall be calculated as a 16-bit CRC with the same generator polynomial as used for a Long Packet Footer as described in Section 9.6 and the checksum byte order in the checksum register is the same order as in Section 9.6. The CRC implementation is presented in Figure 50. The CRC shift register shall be initialized to 0xFFFF.

This checksum is available as a test mode function, therefore two options are allowed for a DSI receiver implementation to report a checksum value shown in Figure 50. For example, the DSC coding system supported by this specification, [VESA01], uses a color transform in the coding process. The DSI receiver shall contain a checksum that calculates based on:

A. Color transformed space, if available from the decoder (most significant 8 bits in each component) and accessed by get_image_checksum_ct(), or

B. The pixel space used by the display, usually three component red-green-blue, which is a second checksum, get_image_checcksum_rgb(), that is optional and only present if an inverse color transform is present in a decoder.

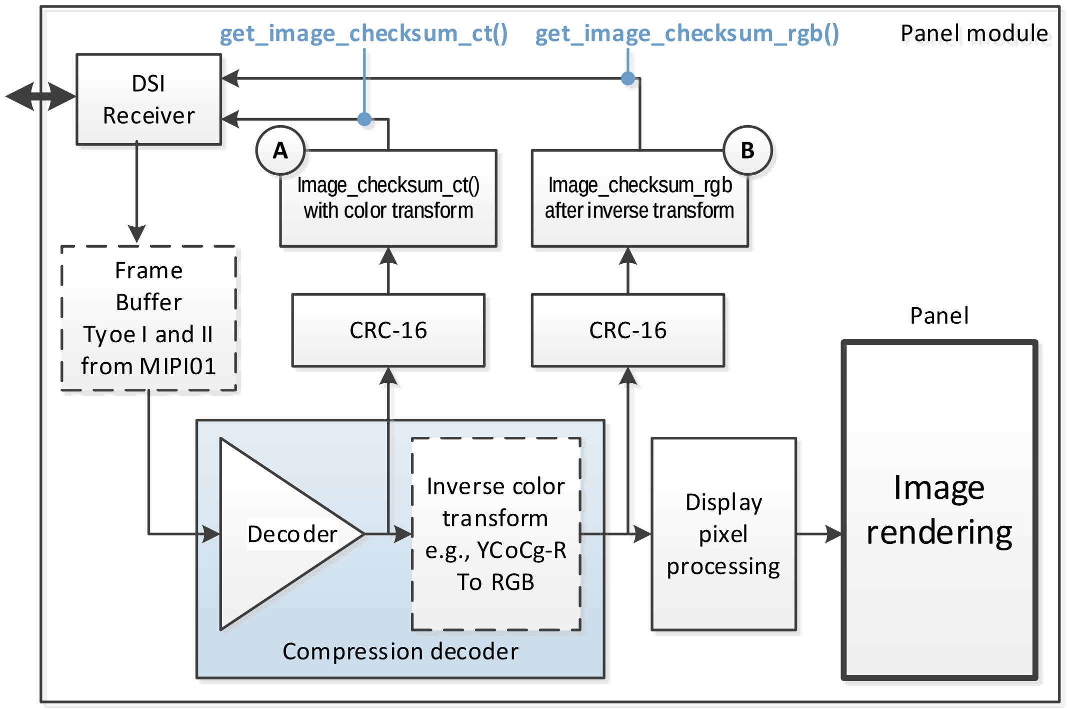

get_image_checksum_ct()
get_image_checksum_rgb()
Panel module
DSI Receiver
A
Image_checksum_ct() with color transform
B
Image_checksum_rgb after inverse transform
Frame Buffer Tyoe I and II from MIPI01
CRC-16
CRC-16
Panel
Decoder
Inverse color transform e.g., YCoCg-R To RGB
Display pixel processing
Image rendering
Compression decoder

Figure 50 CRC-16 Calculates Image-Based Checksum Options (A: Inverse Color Transformed Space, B: In Panel Pixels)

The DSI receiver manufacturer's data sheet shall report the color space and transformed component type (A) or pixel components (B) used for a CRC-16 input. In all cases, the checksum shall use the CRC-16 algorithm compliant with this specification (Section 9.6) and reported in the DSC command compliant to [MIPI01].

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

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

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 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 27.

Table 27 Display Resolutions

Resolution	Horizontal Extent	Vertical Extent
QQVGA	160	120
QCIF	176	144
QCIF+	176	208
QCIF+	176	220
QVGA	320	240
CIF	352	288
CIF+	352	416
CIF+	352	440
(1/2)VGA	320	480

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

10.2 Pixel Formats

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

10.2.1 Video Mode

Peripherals shall implement at least one of the following pixel formats. Host processors shall implement all of the following uncompressed pixel formats, or the compressed data format; all other Video Mode formats in Section 8.7.2 are optional:

1. 16 bpp (5, 6, 5 RGB), each pixel using two bytes; see Section 8.8.20
2. 18 bpp (6, 6, 6 RGB) packed; see Section 8.8.21
3. 18 bpp (6, 6, 6 RGB) loosely packed into three bytes; see Section 8.8.22
4. 24 bpp (8, 8, 8 RGB), each pixel using three bytes; see Section 8.8.23
5. Compressed data (optional for host)

10.2.2 Command Mode

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

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 [MIPI04].

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 shall support ECC, but may support Checksum. If a peripheral does not calculate Checksum it shall still be capable of receiving Checksum bytes from the host processor. If a peripheral supports bidirectional communications and does not support Checksum it shall send bytes of all zeros in the appropriate fields. For interoperability with earlier revision of DSI peripherals where ECC was considered an optional feature, host shall be able to enable/disable ECC capability based on the particular peripheral ECC support capability. The enabling/disabling mechanism is out of scope of DSI. In effect, if an earlier revision peripheral was not supporting ECC, it shall still be capable of receiving ECC byte from the host and sending an all zero ECC byte back to the host for responses over a bidirectional link. 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 [MIPI02] and [MIPI03]. 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 [MIPI01].

10.8 Multiple Peripheral Support

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

A host processor should support a minimum of two peripherals.

10.9 EoTp Support and Interoperability

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

Annex A Contention Detection and Recovery Mechanisms (informative)

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.

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. Refer to [MIPI04] for definition of LP High and LP Low faults.

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:

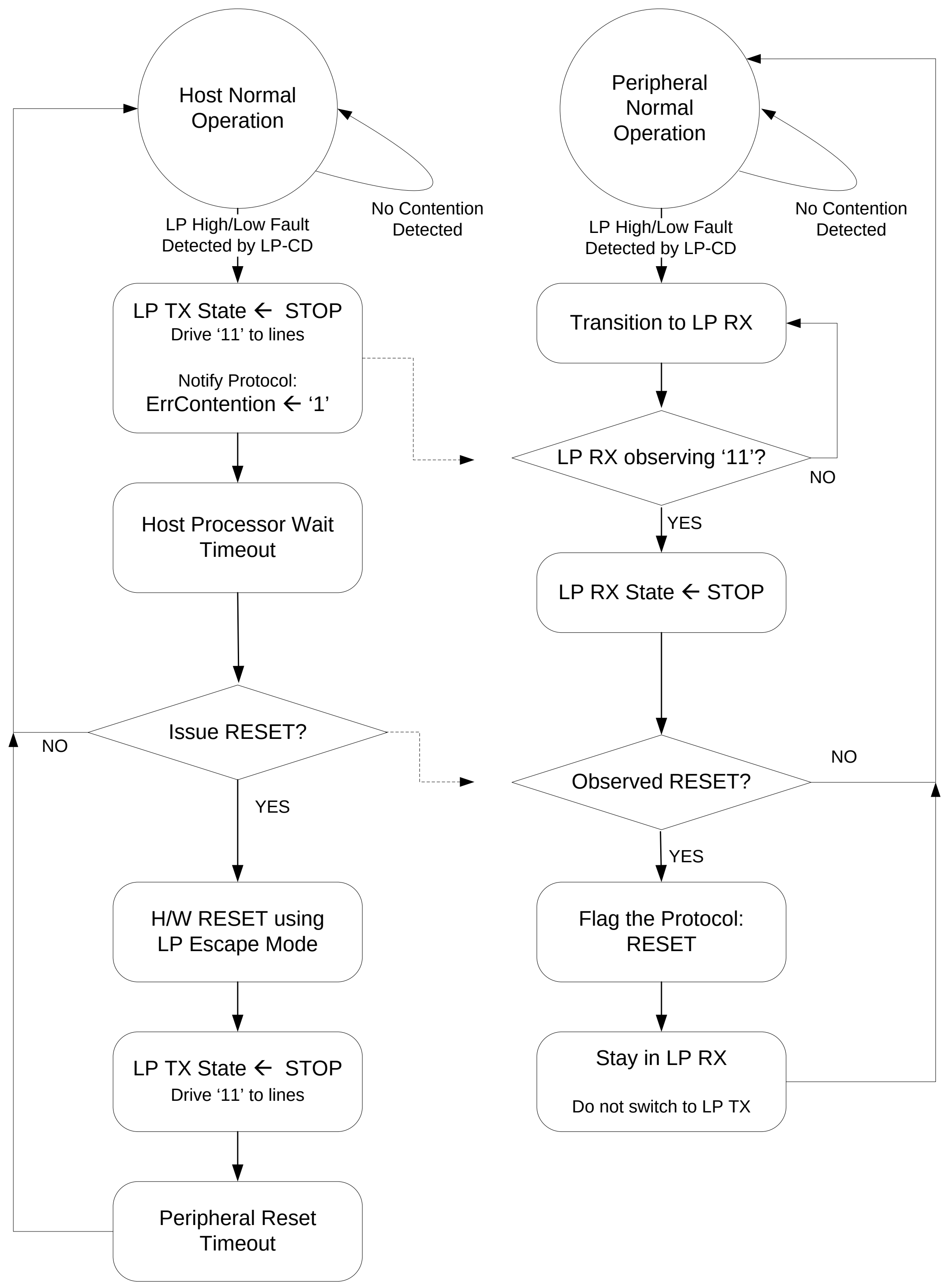
- Sequence of events to resolve LP High ←→ LP Low Contention
    - Case 1: Both sides initially detect the contention
    - Case 2: Only the Host Processor initially detects contention
    - Case 3: Only the Peripheral initially detects contention

Table 28 LP High ←→ LP Low Contention Case 1

Host Processor Side		**Peripheral Side**	
Protocol	**PHY**	**PHY**	**Protocol**
	Detect *LP High Fault* or *LP Low Fault*	Detect *LP High Fault* or *LP Low Fault*	
	Transition to *Stop* State (LP-11)	Transition to LP-RX	Set *Contention Detected* in Error Report (see Table 21)
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	

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

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



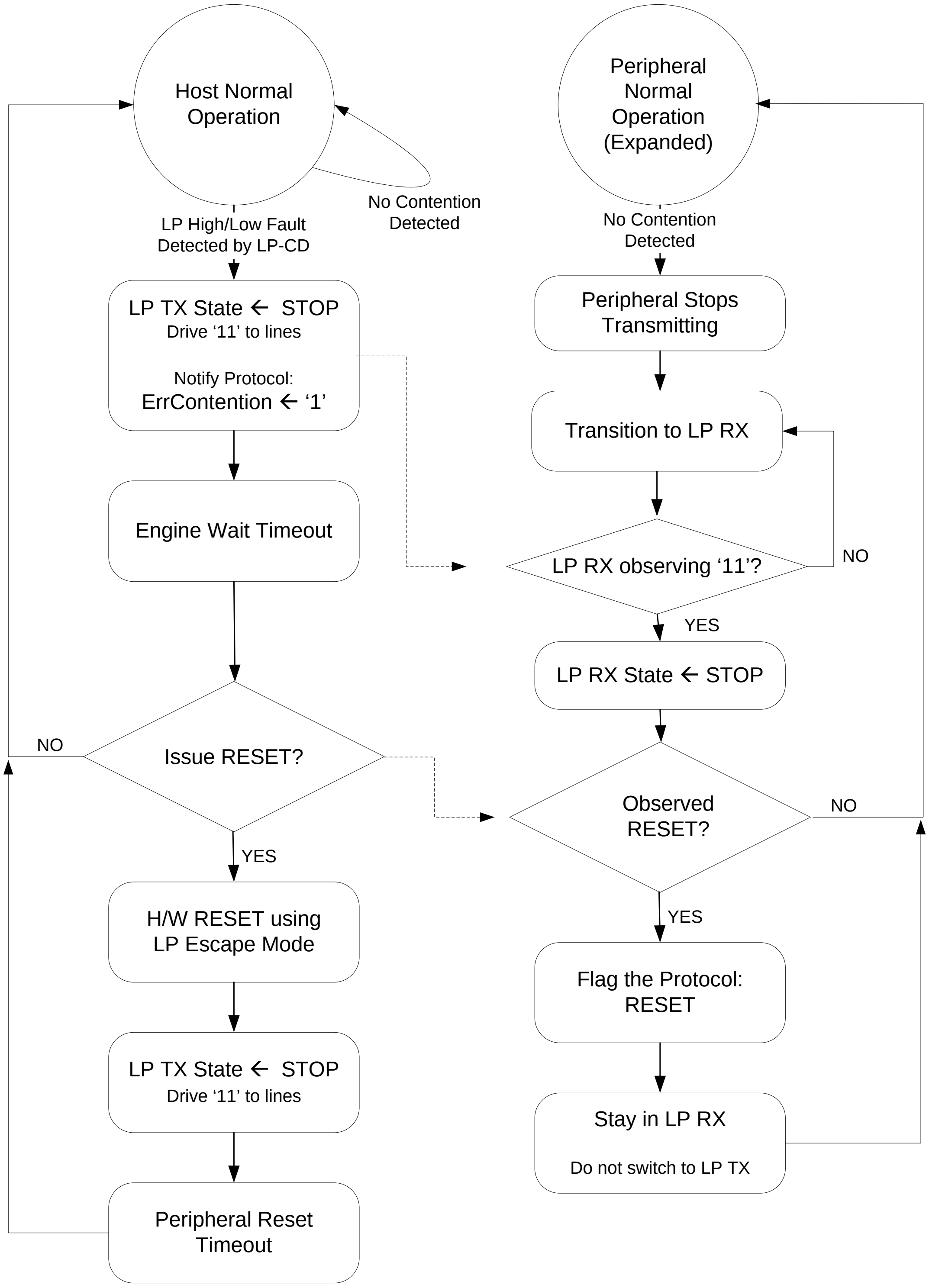
2471

2472 **Figure 51 LP High ←→ LP Low Contention Case 1**

2473 **Table 29 LP High ←→ LP Low Contention Case 2**

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

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



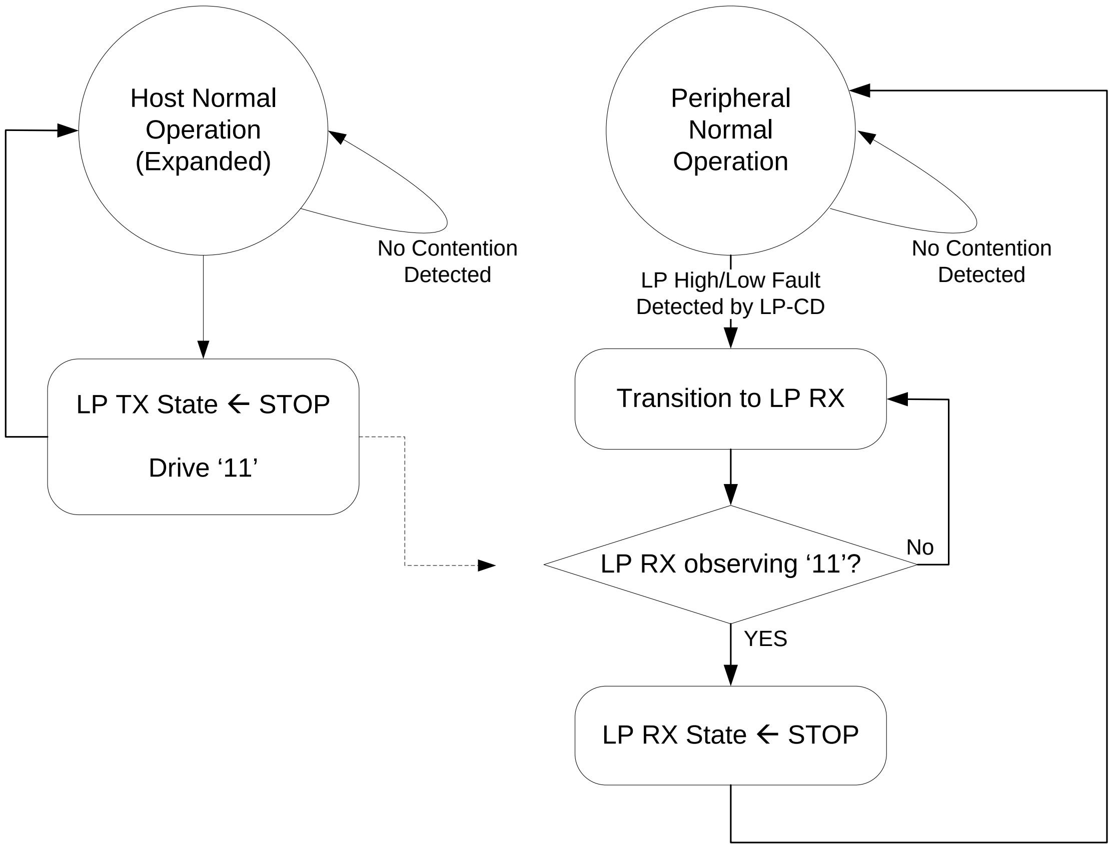
2474

2475 **Figure 52 LP High ←→ LP Low Contention Case 2**

2476 **Table 30 LP High ←→ LP Low Contention Case 3**

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

Host Processor Side		**Peripheral Side**	
Protocol	**PHY**	**PHY**	**Protocol**
	No detection of EL contention	Detect *LP High Fault* or *LP Low Fault*	
		Transition to LP-RX	Set *Contention Detected* in Error Report (see Table 21)
		Peripheral waits until it observes *Stop* state before responding to bus activity.	
	Normal transition to *Stop* State (LP-11)	Observe *Stop* State	



2477
2478

2479 **Figure 53 LP High ←→ LP Low Contention Case 3**

Annex B Checksum Generation Example (informative)

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

```
/* ************************ DECLARATIONS *********************** */
#include <stdio.h>

/* Start of Test Data */
static unsigned char gpcTestData0[] = { 0x00 };
static unsigned char gpcTestData1[] = { 0x01 };
static unsigned char gpcTestData2[] = { 0xFF, 0x00, 0x00, 0x00, 0x1E,
   0xF0, 0x1E, 0xC7, 0x4F, 0x82, 0x78, 0xC5, 0x82, 0xE0, 0x8C, 0x70,
   0xD2, 0x3C, 0x78, 0xE9, 0xFF, 0x00, 0x00, 0x01 };
static unsigned char gpcTestData3[] = { 0xFF, 0x00, 0x00, 0x02, 0xB9,
   0xDC, 0xF3, 0x72, 0xBB, 0xD4, 0xB8, 0x5A, 0xC8, 0x75, 0xC2, 0x7C,
   0x81, 0xF8, 0x05, 0xDF, 0xFF, 0x00, 0x00, 0x01 };
#define NUMBER_OF_TEST_DATA0_BYTES 1
#define NUMBER_OF_TEST_DATA1_BYTES 1
#define NUMBER_OF_TEST_DATA2_BYTES 24
#define NUMBER_OF_TEST_DATA3_BYTES 24
/* End of Test Data */

unsigned short CalculateCRC16( unsigned char *pcDataStream, unsigned
short sNumberOfDataBytes );

/* ************************* MAIN ROUTINE ************************ */
void main( void )
{
   unsigned short sCRC16Result;
   sCRC16Result = CalculateCRC16( gpcTestData2,
                NUMBER_OF_TEST_DATA2_BYTES );
   printf( "Checksum CS[15:0] = 0x%04X\n", sCRC16Result );
}
/* ********** END OF MAIN ****** START OF CRC CALCULATION ********** */

/* CRC16 Polynomial, logically inverted 0x1021 for x^16+x^15+x^5+x^0 */
static unsigned short gsCRC16GenerationCode = 0x8408;

unsigned short CalculateCRC16( unsigned char *pcDataStream, unsigned
                short sNumberOfDataBytes )
{
   /*
   sCRC16Result: the return of this function,
   sByteCounter: address pointer to count the number of the
          calculated data bytes
   cBitCounter: counter for bit shift (0 to 7)
   cCurrentData: byte size buffer to duplicate the calculated data
          byte for a bit shift operation
   */
   unsigned short sByteCounter;
   unsigned char cBitCounter;
   unsigned char cCurrentData;
   unsigned short sCRC16Result = 0xFFFF;
   if ( sNumberOfDataBytes > 0 )
   {
      for ( sByteCounter = 0; sByteCounter < sNumberOfDataBytes;
         sByteCounter++ )
      {
         cCurrentData = *( pcDataStream + sByteCounter );
         for ( cBitCounter = 0; cBitCounter < 8;
            cBitCounter++ )
         {
            if ( ( ( sCRC16Result & 0x0001 ) ^ ( ( 0x0001 *
                cCurrentData) & 0x0001 ) ) > 0 )
               sCRC16Result = ( ( sCRC16Result >> 1 )
                  & 0x7FFF ) ^ gsCRC16GenerationCode;
            else
               sCRC16Result = ( sCRC16Result >> 1 )
                        & 0x7FFF;
            cCurrentData = (cCurrentData >> 1 ) & 0x7F;
         }
      }
   }
   return sCRC16Result;
}
/* *************** END OF SUBROUTINE TO CALCULATE CRC ************** */
```

Outputs from the various input streams are as follows:

Data (`gpcTestData0`): `00`

`Checksum CS[15:0] = 0x0F87`

Data (`gpcTestData1`): `01`

`Checksum CS[15:0] = 0x1E0E`

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

`Checksum CS[15:0] = 0xE569`

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

`Checksum CS[15:0] = 0x00F0`

Annex C Interlaced Video Transmission Sourcing

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

An example of the video mode interface timing for non-burst transmission with Sync Start and End sourcing interlaced video is shown in Figure 54. Note that in the first field, no timing differs from Figure 43. In the second field, note HSS is not implied at the V Sync Start and V Sync End timing pulses.

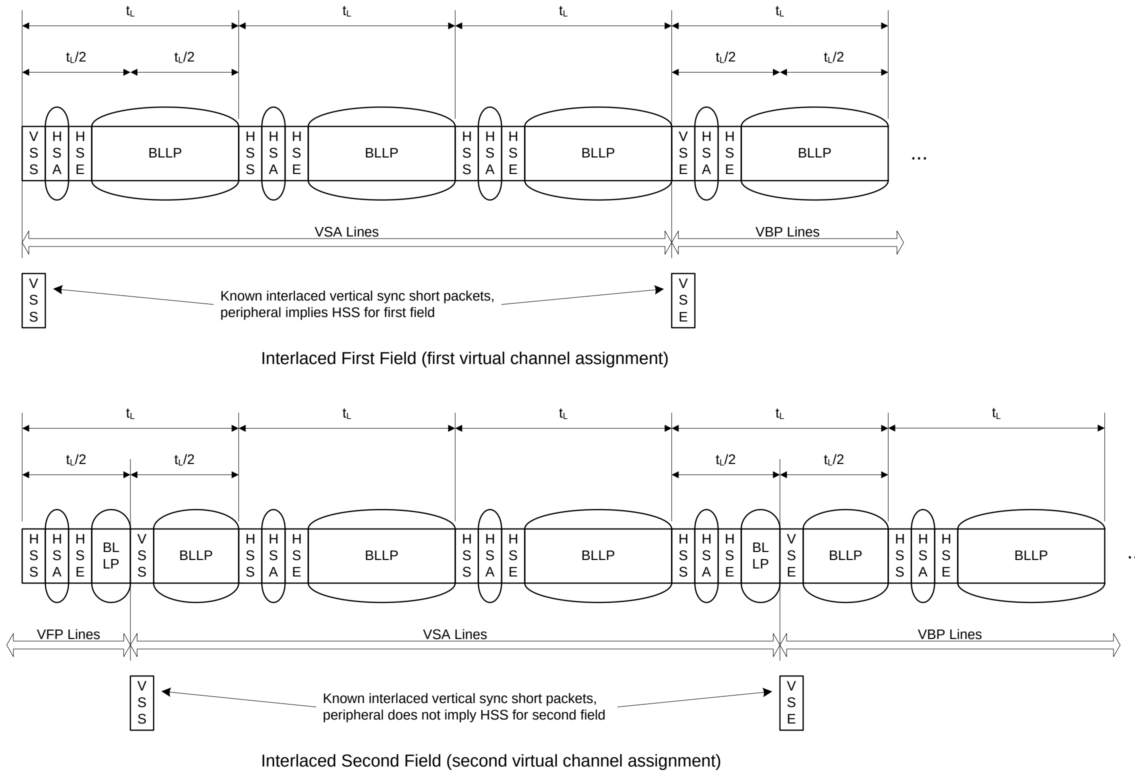

t_L
t_L/2
VSS
HSA
HSE
BLLP
HSS
VSE
...
VSA Lines
VBP Lines
Known interlaced vertical sync short packets, peripheral implies HSS for first field
Interlaced First Field (first virtual channel assignment)
BL LP
VFP Lines
Known interlaced vertical sync short packets, peripheral does not imply HSS for second field
Interlaced Second Field (second virtual channel assignment)

Figure 54 Video Mode Interface Timing: Non-Burst Transmission with Sync Start and End (Interlaced Video)

An example of the video mode interface timing for non-burst transmission with Sync Events sourcing interlaced video is shown in Figure 55. Note that in the first field, no timing differs from the previous example. In the second field, note HSS is not implied at the V Sync Start timing event.

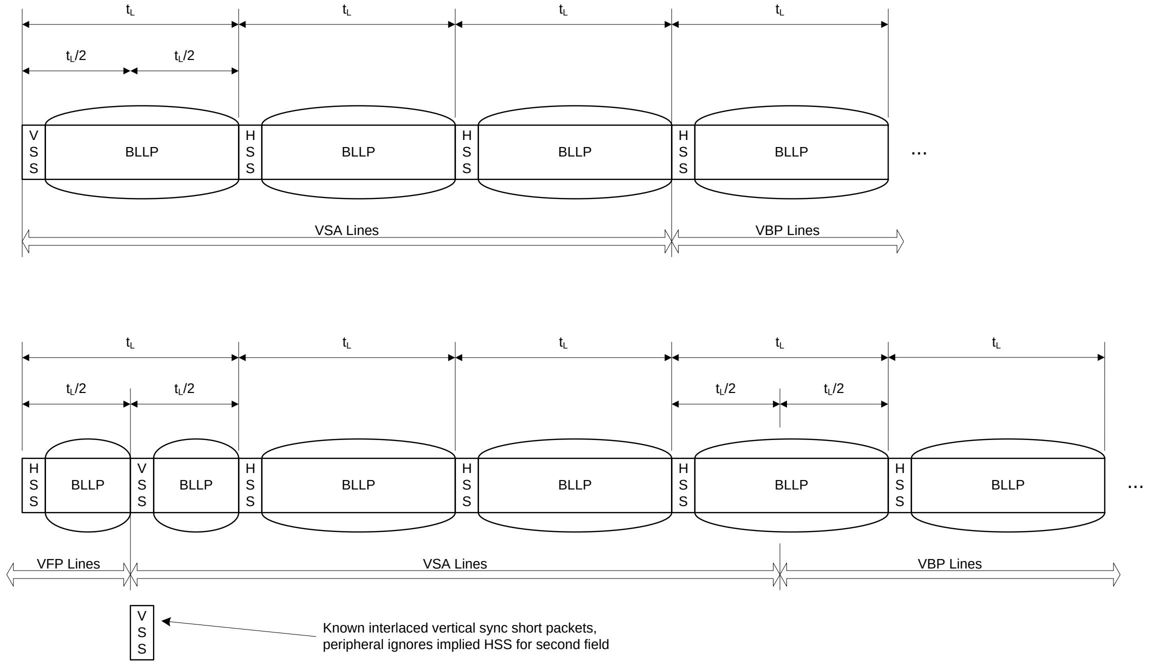

t_L
t_L/2
VSS
HSS
BLLP
VSA Lines
VBP Lines
VFP Lines
Known interlaced vertical sync short packets, peripheral ignores implied HSS for second field

Figure 55 Video Mode Interface Timing: Non-Burst Transmission with Sync Events (Interlaced Video)

Annex D Profile for DSI Transport for VESA Display Stream Compression

(This annex is normative when using DSC from reference VESA01)

D.1 Profile Normative Requirements

D.1.1 Encoder Instantiations

The DSC-compliant encoder shall be associated only with a DSI transmitter processor, and shall be used to encode a CBR bitstream.

D.1.2 Decoder Instantiations

The DSC-compliant decoder shall be associated only with a DSI receiver peripheral, and shall be used to decode a CBR bitstream.

D.1.3 Horizontal Slice Size

One peripheral (DSI link) shall support no more than four horizontal slices. The DSI peripheral specifies the number of supported horizontal slices, e.g., 1, 2, 3 or 4. Also, see Table 31 for additional restrictions of the slice_width value.

For clarity, the coding system [VESA01] requires one only slice dimension for a picture.

D.1.4 Vertical Slice Size

The vertical frame dimension shall be evenly divisible by the number of lines per slice as the DSC encoder always transmits data in complete slices, see Table 31.

D.1.5 DSC parameter minimum requirements

The following parameters represent a minimum required set of behaviors when implementing VESA DSC in a DSI link.

- Profile 8 refers to a DSC parameter table with 8 bpp and 8 bpc.
- Profile 12 refers to a DSC parameter table with 12 bpp and 8 bpc.
- Generic profile has no specific VESA DSC reference table but uses the VESA DSC Annex D and VESA DSC Annex E equations to build a DSI profile.

Table 31. DSC Required Parameters

Name	Profile 8	Profile 12	Generic Profile
Bits_per_pixel	= 8	= 12	≥6
dsc_version_major	= 1	= 1	= 1
dsc_version_minor	= 1	= 1	= 1
pps_identifier	Allowed	Allowed	Allowed
bits_per_component	= 8	= 8	= 8 or 10

Name	Profile 8	Profile 12	Generic Profile
linebuf_depth	= bpc + 1	= bpc + 1	= bpc + 1
block_pred_enable	= 0 or 1	= 0 or 1	= 0 or 1
convert_rgb	= 1	= 1	= 1
enable_422	= 0	= 0	= 0
vbr_enable	= 0	= 0	= 0
pic_height	Height of entire picture	Height of entire picture	Height of entire picture
pic_width	Width of entire picture	Width of entire picture	Width of entire picture
slice_height	≥8 and (pic_height) modulo (slice_height) = 0.	≥8 and (pic_height) modulo (slice_height) = 0.	≥4 and (pic_height) modulo (slice_height) = 0.
slice_width	(pic_width modulo slice_width) = 0	(pic_width modulo slice_width) = 0	(pic_width modulo slice_width) = 0
Number of pixels / slice	≥ 15,000	≥ 15,000	≥ 15,000
chunk_size (formula is informative)	chunk_size = ceil(bits_per_pixel * slice_width / 8)	chunk_size = ceil(bits_per_pixel * slice_width / 8)	chunk_size = ceil(bits_per_pixel * slice_width / 8)
initial_xmit_delay	Fixed (Refer to DSC Annex E [VESA01])	Fixed (Refer to DSC Annex E [VESA01])	Unrestricted
initial_dec_delay	Fixed (Refer to DSC Annex E [VESA01])	Fixed (Refer to DSC Annex E [VESA01])	Unrestricted
initial_scale_value (formula is informative)	Fixed (Refer to DSC Annex E [VESA01]) rc_model_size / (rc_model_size - initial_offset)	Fixed (Refer to DSC Annex E [VESA01]) rc_model_size / (rc_model_size - initial_offset)	Fixed (Refer to DSC Annex E [VESA01]) rc_model_size / (rc_model_size - initial_offset)
scale_increment_interval	Calculate by the formula from DSC Annex E [VESA01]	Calculate by the formula from DSC Annex E [VESA01]	Calculate by the formula from DSC Annex E [VESA01]
scale_decrement_interval	Calculate by the formula from DSC Annex E [VESA01]	Calculate by the formula from DSC Annex E [VESA01]	Calculate by the formula from DSC Annex E [VESA01]
first_line_bpg_offset	Fixed (Refer to DSC Annex E [VESA01])	Fixed (Refer to DSC Annex E [VESA01])	Unrestricted
nfl_bpg_offset	Calculate by the formula from DSC Annex E [VESA01]	Calculate by the formula from DSC Annex E [VESA01]	Calculate by the formula from DSC Annex E [VESA01]
slice_bpg_offset	Calculate by the formula from DSC Annex E [VESA01]	Calculate by the formula from DSC Annex E [VESA01]	Calculate by the formula from DSC Annex E [VESA01]
initial_offset	Fixed (Refer to DSC Annex E [VESA01])	Fixed (Refer to DSC Annex E [VESA01])	Unrestricted
final_offset	Calculate by the formula from DSC Annex E [VESA01]	Calculate by the formula from DSC Annex E [VESA01]	Calculate by the formula from DSC Annex E [VESA01]

Name	Profile 8	Profile 12	Generic Profile
flatness_min_qp	Fixed (Refer to DSC Annex E [VESA01])	Fixed (Refer to DSC Annex E [VESA01])	Unrestricted
flatness_max_qp	Fixed (Refer to DSC Annex E [VESA01])	Fixed (Refer to DSC Annex E [VESA01])	Unrestricted
rc_parameter_set	Fixed (Refer to DSC Annex E [VESA01])	Fixed (Refer to DSC Annex E [VESA01])	Unrestricted

D.1.6 PPS Requirements

A picture parameter set is said to be transmitted in one of the following ways in byte-aligned data fields:

1. Using one PPS transaction (Section 8.8.26) where the PPS is defined by the DSC Standard [VESA01].
2. Implicitly when the DSI peripheral is reset, a PPS preset in the peripheral takes effect immediately.
3. If the DSI peripheral contains one or more PPS tables, the Compression Mode short packet (Section 8.8.25) may signal a PPS data change by identifying a unique, stored PPS set. A stored PPS may be pre-programmed or updated and stored by a previous PPS transaction.

A PPS change takes effect in a video mode peripheral that is processing a data stream after the next vertical sync, not within the same frame, as shown in Figure 56. The PPS change takes effect in a command mode peripheral after the next signaled tearing effect, that is, either after a TE signal pulse or a TE Trigger Message.

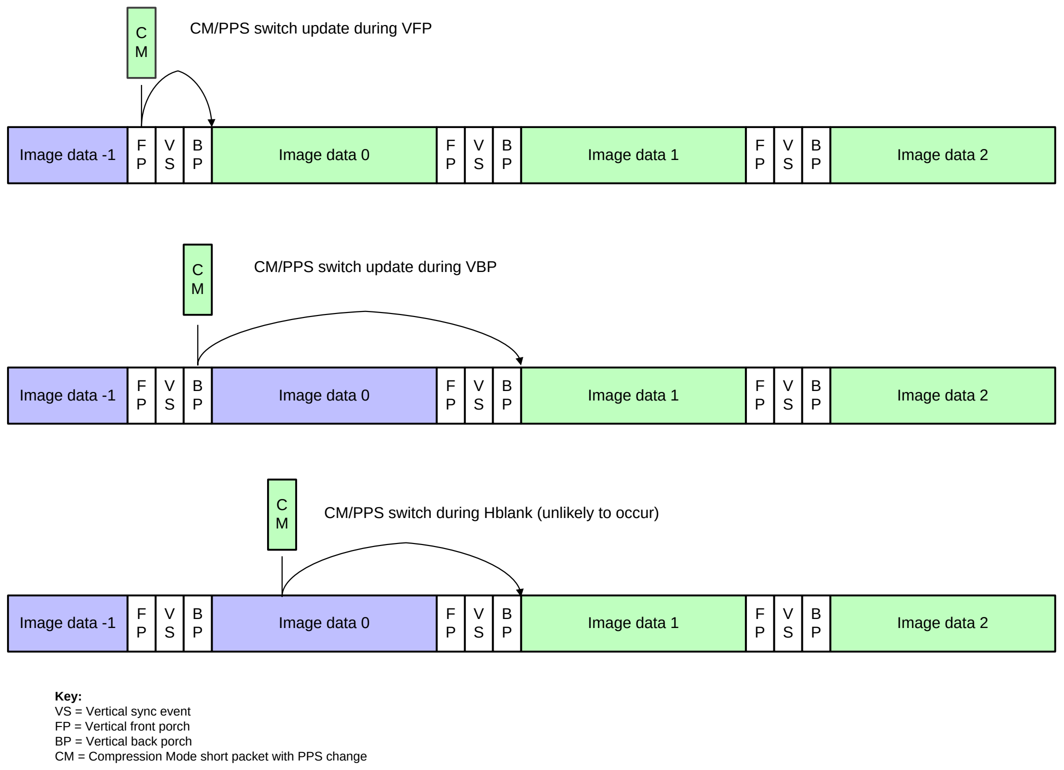

CM
CM/PPS switch update during VFP
Image data -1
FP
VS
BP
Image data 0
Image data 1
Image data 2
CM/PPS switch update during VBP
CM/PPS switch during Hblank (unlikely to occur)
Key:
VS = Vertical sync event
FP = Vertical front porch
BP = Vertical back porch
CM = Compression Mode short packet with PPS change

Figure 56 PPS Update by Setting the Compression Mode Short Packet

D.2 Implementation Guidelines (informative)

D.2.1 Vertical Slice Size

The vertical frame dimension should be evenly divisible by the number of lines per slice as the DSC encoder always transmits data in complete slices.

D.2.2 DSC Guidelines

Implementers should refer to guidelines in the VESA DSC Standard Annexes D and E [VESA01] to optimize quality performance.