MIPI - Camera Serial Interface 2 (CSI-2)

Table of Contents

Overview

Scope

The Camera Serial Interface 2 specification defines an interface between a peripheral device (camera) and a host processor (baseband, application engine). The purpose of this document is to specify a standard interface between a camera and a host processor for mobile applications.

Purpose

Demand for increasingly higher image resolutions is pushing the bandwidth capacity of existing host processor-to-camera sensor interfaces. Common parallel interfaces are difficult to expand, require many interconnects and consume relatively large amounts of power

Terminology

Definitions

Lane: A differential conductor pair, used for data transmission. For CSI-2 a data Lane is unidirectional.

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

Payload: Application data only – with all sync, header, ECC and checksum and other protocol-related information removed. This is the “core” of transmissions between application processor and peripheral.

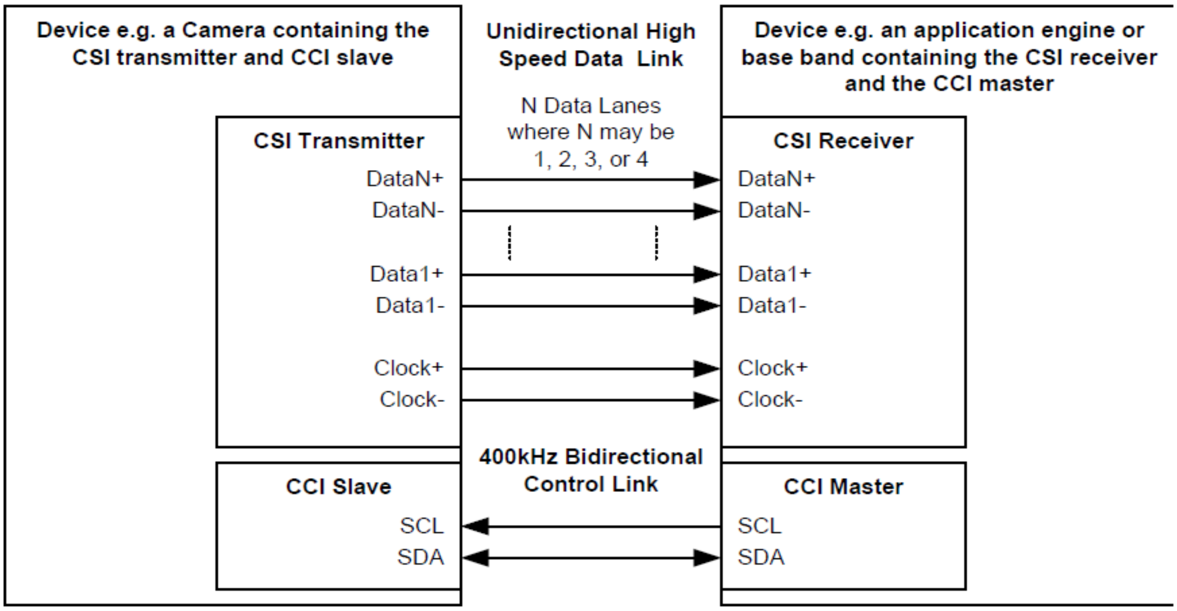
Sleep Mode: Sleep mode (SLM) is a leakage level only power consumption mode.

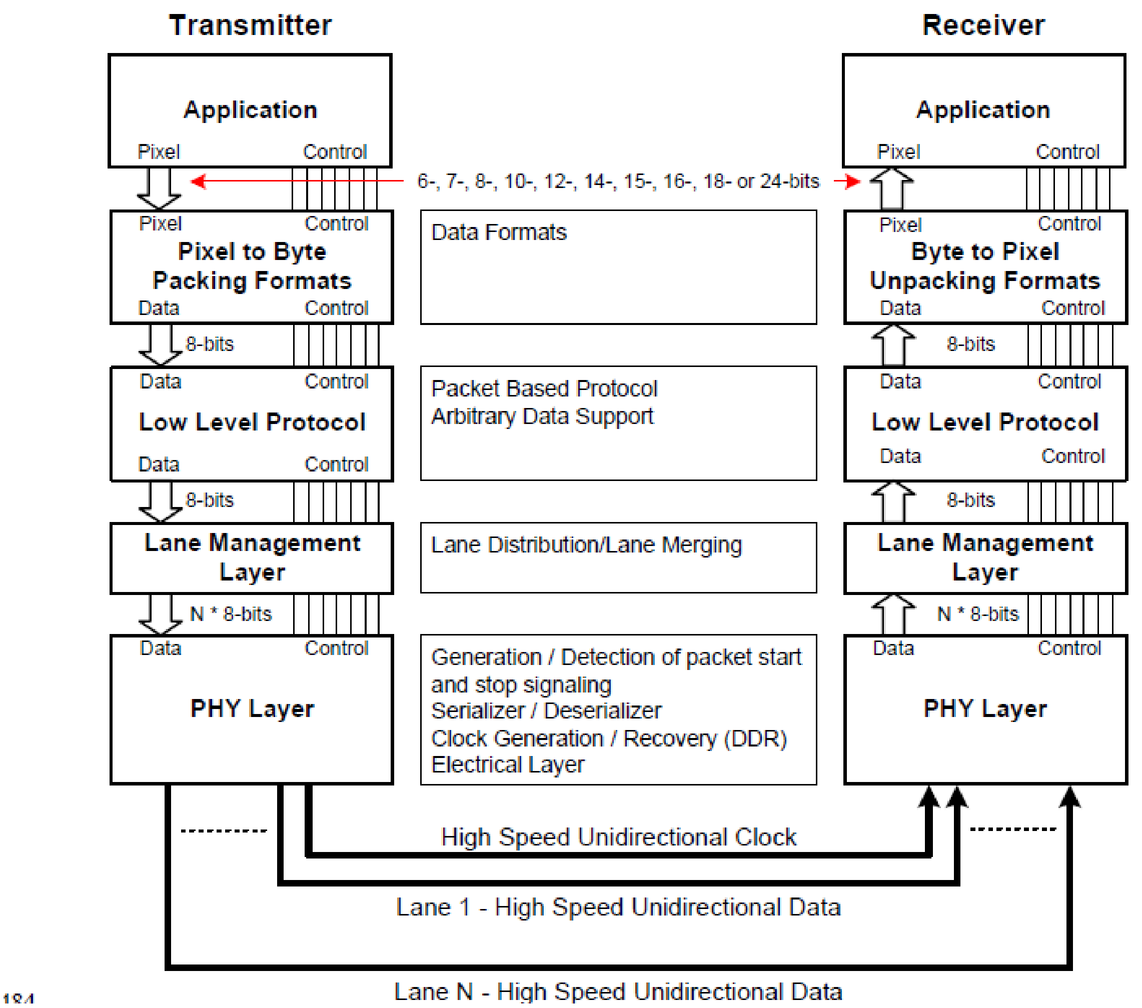
Transmission: The time during which high-speed serial data is actively traversing the bus. A transmission comprised of one or more packets. A transmission is bounded by SoT (Start of Transmission) and EoT (End of Transmission) at beginning and end, respectively.

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 links each packet to its intended peripheral.

Overview of CSI-2

The CSI-2 specification defines standard data transmission and control interfaces between transmitter and receiver. Data transmission interface (referred as CSI-2) is unidirectional differential serial interface with data and clock signals; the physical layer of this interface is the *MIPI Alliance Specification for D-PHY* [MIPI01].


Device e.g. a Camera containing the CSI transmitter and CCI slave
Unidirectional High Speed Data Link
Device e.g. an application engine or base band containing the CSI receiver and the CCI master
N Data Lanes where N may be 1, 2, 3, or 4
CSI Transmitter
DataN+
DataN-
Data1+
Data1-
Clock+
Clock-
CSI Receiver
DataN+
DataN-
Data1+
Data1-
Clock+
Clock-
400kHz Bidirectional Control Link
CCI Slave
SCL
SDA
CCI Master
SCL
SDA


Transmitter
Receiver
Application
Pixel
Control
6-, 7-, 8-, 10-, 12-, 14-, 15-, 16-, 18- or 24-bits
Pixel to Byte Packing Formats
Data Formats
Byte to Pixel Unpacking Formats
Data
8-bits
Low Level Protocol
Packet Based Protocol
Arbitrary Data Support
Lane Management Layer
Lane Distribution/Lane Merging
N * 8-bits
PHY Layer
Generation / Detection of packet start and stop signaling
Serializer / Deserializer
Clock Generation / Recovery (DDR)
Electrical Layer
High Speed Unidirectional Clock
Lane 1 - High Speed Unidirectional Data
Lane N - High Speed Unidirectional Data
184

CSI-2 Layer Definitions:

PHY Layer. The PHY Layer specifies the 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 signalling and the timing relationship between clock and data Lanes.

Protocol Layer. The Protocol layer is composed of several layers, each with distinct responsibilities. The CSI-2 protocol enables multiple data streams using a single interface on the host processor. The Protocol layer specifies how multiple data streams may be tagged and interleaved so each data stream can be properly reconstructed.

Pixel/Byte Packing/Unpacking Layer. The CSI-2 supports image applications with varying pixel formats from six to twenty-four bits per pixels. In the transmitter this layer packs pixels from the Application layer into bytes before sending the data to the Low Level Protocol layer. In the receiver this layer unpacks bytes from the Low Level Protocol layer into pixels before sending the data to the

Application layer. Eight bits per pixel data is transferred unchanged by this layer.

Low Level Protocol. The Low Level Protocol (LLP) includes the means of establishing bit level and byte-level synchronization for serial data transferred between SoT (Start of Transmission) and EoT (End of Transmission) events and for passing data to the next layer. The minimum data granularity of the LLP is one byte. The LLP also includes assignment of bit-value interpretation within the byte, i.e. the “Endian” assignment.

Lane Management. CSI-2 is Lane-scalable for increased performance. The number of data Lanes may be one, two, three or four depending on the bandwidth requirements of the application. The transmitting side of the interface distributes (“distributor” function) the outgoing data stream to one or more Lanes. On the receiving side, the interface collects bytes from the Lanes and merges (“merger” function) them together into a recombined data stream that restores the original stream sequence. Data within the Protocol layer is organized as packets. The transmitting side of the interface appends header and optional error-checking information on to data to be transmitted at the Low Level Protocol layer. On the receiving side, the header is stripped off at the Low Level Protocol layer and interpreted by corresponding logic in the receiver. Error-checking information may be used to test the integrity of incoming data.

Application Layer. This layer describes higher-level encoding and interpretation of data contained in the data stream. The CSI-2 specification describes the mapping of pixel values to bytes.

Physical Layer

The CSI-2 uses the [MIPI01] physical layer.
The physical layer for a CSI-2 implementation is composed of between one and four unidirectional data Lanes and one clock Lane. All CSI-2 transmitters and receivers shall support continuous clock behaviour on the Clock Lane, and optionally may support non-continuous clock behaviour.
For continuous clock behaviour the Clock Lane remains in high-speed mode generating active clock signals between the transmissions of data packets.

For non-continuous clock behaviour the Clock Lane enters the LP-11 state between the transmissions of data packets.

The minimum physical layer requirement for a CSI-2 transmitter is

- Data Lane Module: Unidirectional master, HS-TX, LP-TX and a CIL-MFEN function

- Clock Lane Module: Unidirectional master, HS-TX, LP-TX and a CIL-MCNN function

The minimum physical layer requirement for a CSI-2 receiver is

- Data Lane Module: Unidirectional slave, HS-RX, LP-RX, and a CIL-SFEN function
- Clock Lane Module: Unidirectional slave, HS-RX, LP-RX, and a CIL-SCNN function

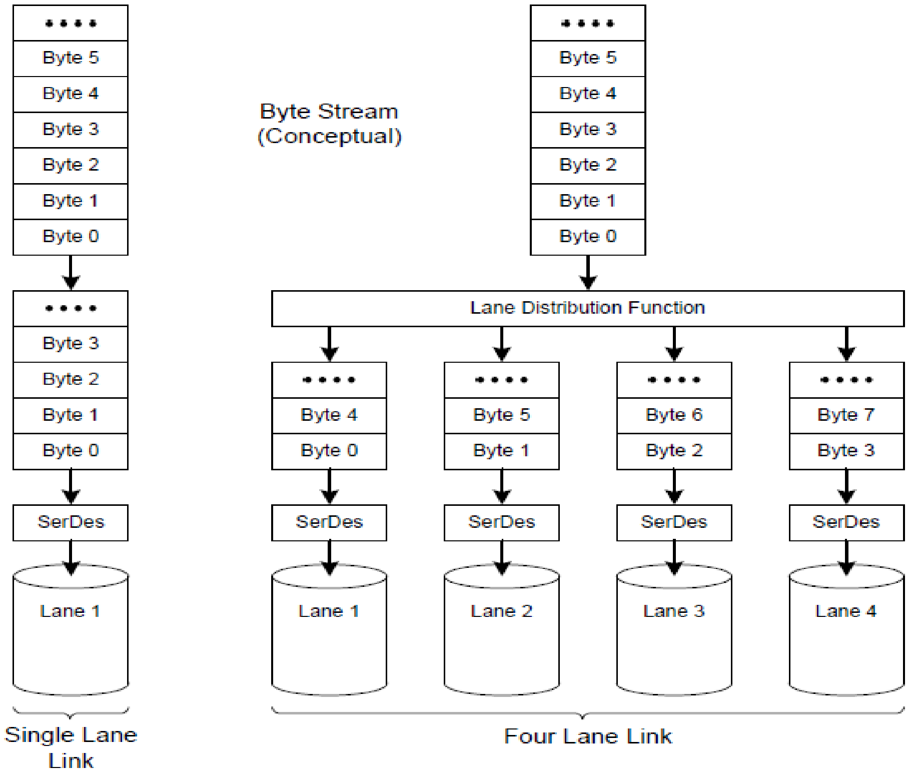
All CSI-2 implementations shall support forward escape ULPS on all Data Lanes.

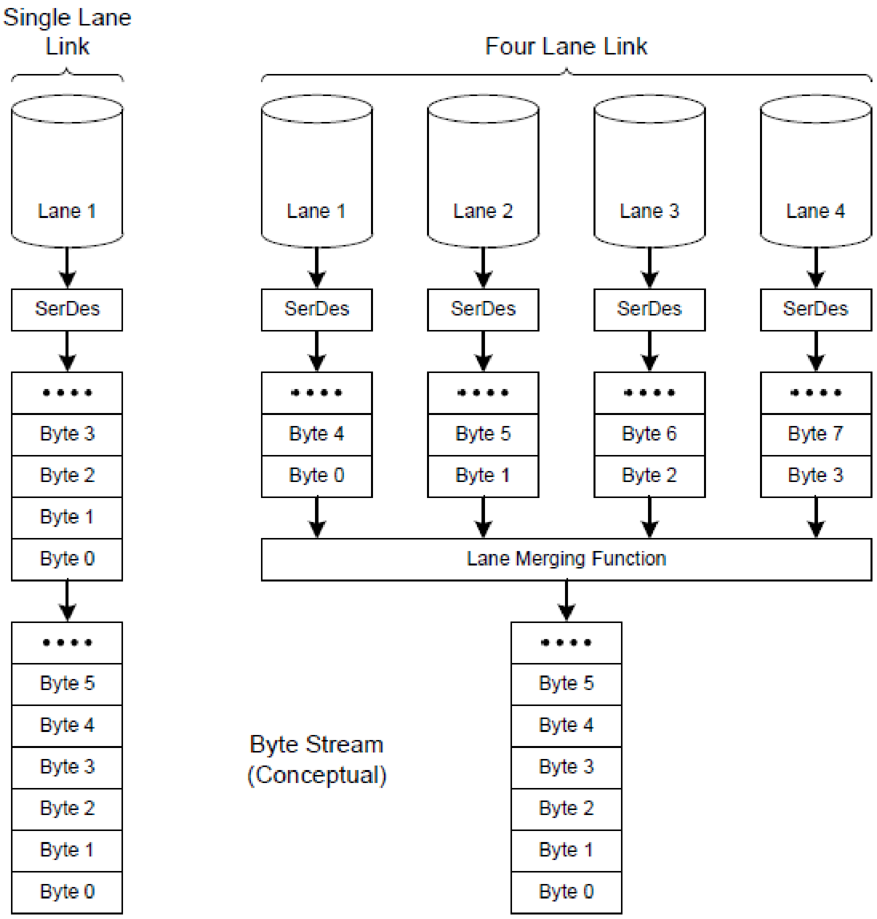
Multi-Lane Distribution and Merging

CSI-2 is a Lane-scalable specification. Applications requiring more bandwidth than that provided by one data Lane, or those trying to avoid high clock rates, can expand the data path to two, three, or four Lanes wide and obtain approximately linear increases in peak bus bandwidth. The mapping between data at higher layers and the serial bit stream is explicitly defined to ensure compatibility between host processors and peripherals that make use of multiple data Lanes.

Conceptually, between the PHY and higher functional layers is a layer that handles multi-Lane configurations. In the transmitter, the layer distributes a sequence of packet bytes across N Lanes, where each Lane is an independent unit of physical-layer logic (serializers, etc.) and transmission circuitry. In the receiver, it collects incoming bytes from N Lanes and consolidates (merges) them into complete packets to pass into the packet decomposer.

Conceptual Overview of the Lane Distributor Function

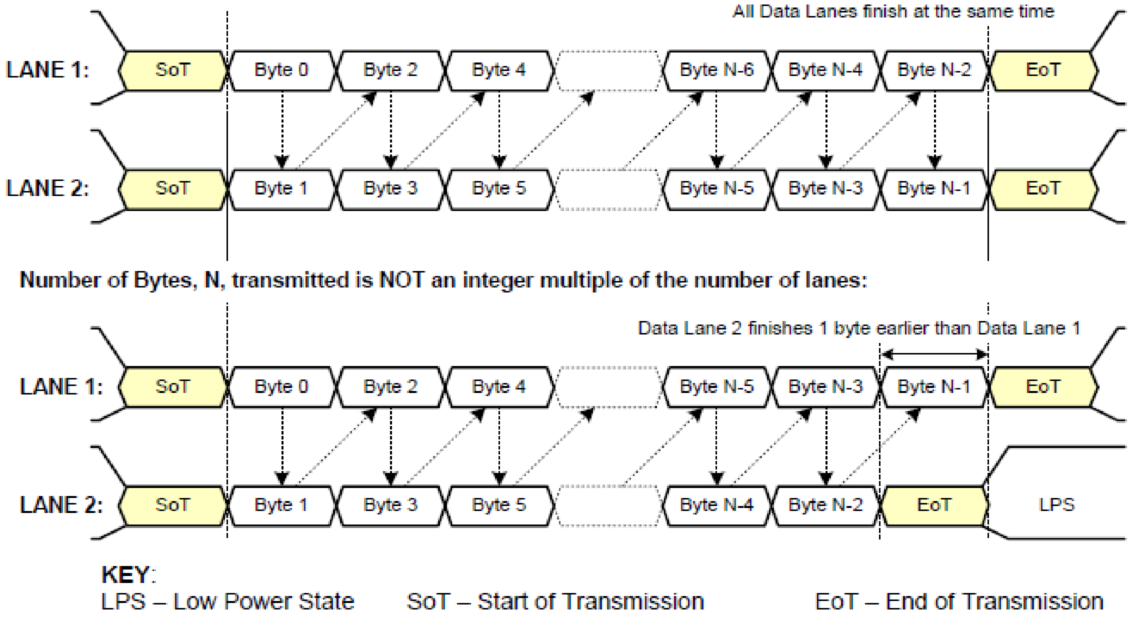




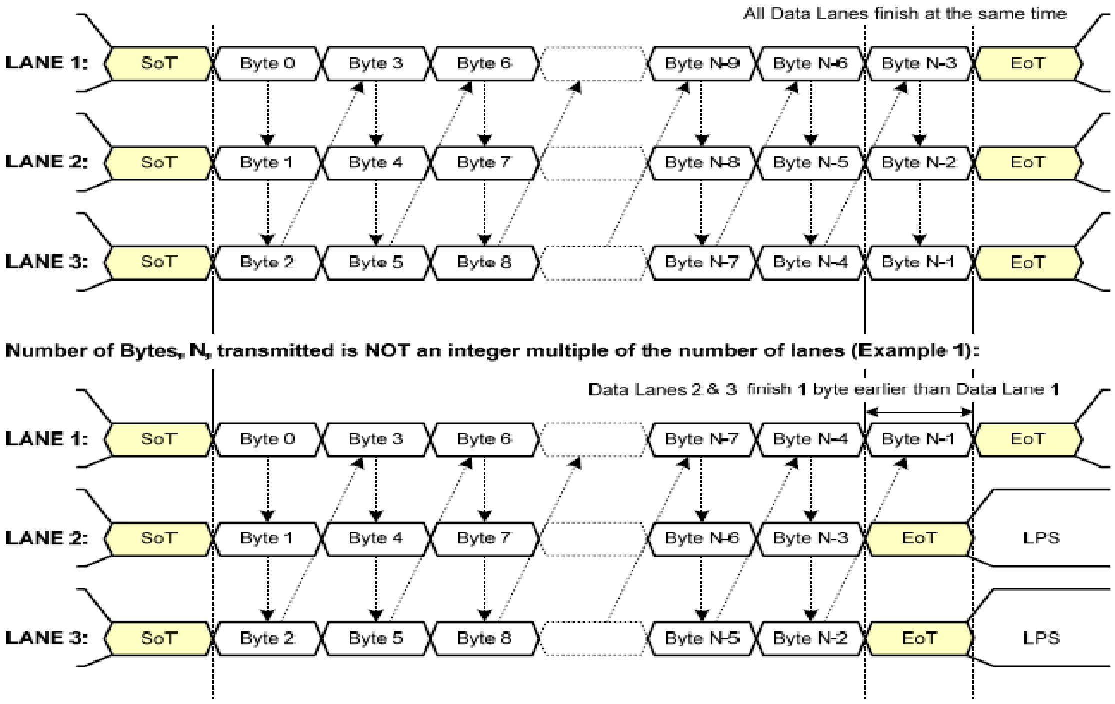
The Lane distributor takes a transmission of arbitrary byte length, buffers up N bytes (where N = number of Lanes), and then sends groups of N bytes in parallel across N Lanes. Before sending data, all Lanes perform the SoT sequence in parallel to indicate to their corresponding receiving units that the first byte of a packet is beginning. After SoT, the Lanes send groups of successive bytes from the first packet in parallel, following a round-robin process.

Examples:

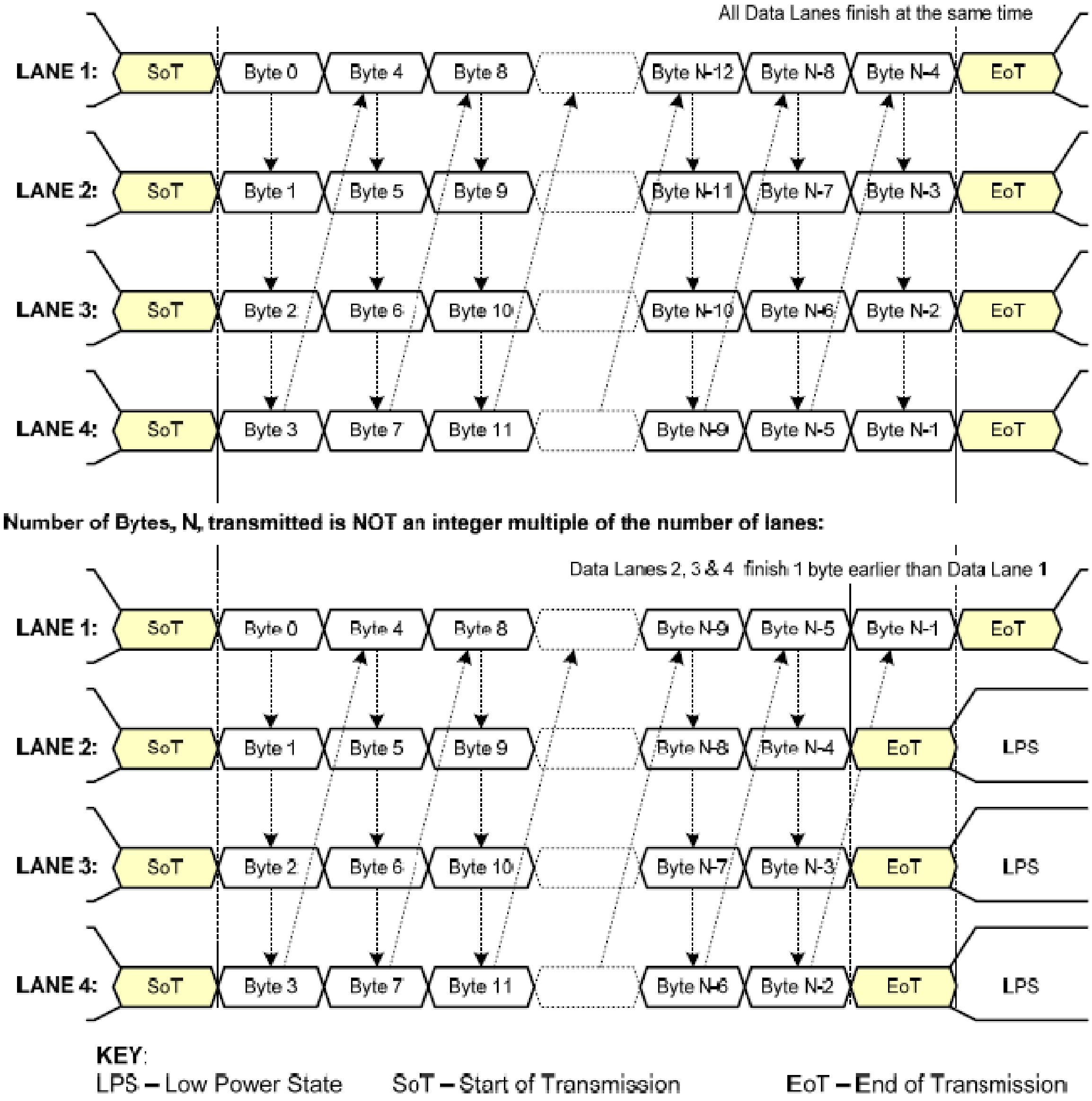
- 2-Lane system (Figure): byte 0 of the packet goes to Lane 1, byte 1 goes to Lane 2, byte 2 to Lane 1, byte 3 goes to Lane 2, byte 4 goes to Lane 1 and so on.



- 3-Lane system (Figure): byte 0 of the packet goes to Lane 1, byte 1 goes to Lane 2, byte 2 to Lane 3, byte 3 goes to Lane 1, byte 4 goes to Lane 2 and so on.



- 4-Lane system (Figure): byte 0 of the packet goes to Lane 1, byte 1 goes to Lane 2, byte 2 to Lane 3, byte 3 goes to Lane 4, byte 4 goes to Lane 1 and so on



Multi-Lane Interoperability

The Lane distribution and merging layers shall be reconfigurable via the Camera Control Interface when more than one data Lane is used.

An "N" data Lane receiver shall be connected with an "M" data Lane transmitter, by CCI configuration of the Lane distribution and merging layers within the CSI-2 transmitter and receiver when more than one data Lane is used. Thus, a receiver with four data Lanes shall work with transmitters with one, two, three or four data Lanes. Likewise, a transmitter with four data Lanes shall work with receivers with four or fewer data Lanes. Transmitter Lanes 1 to M shall be connected to the receiver Lanes 1 to M.

- If M<=N then there is no loss of performance – the receiver has sufficient data Lanes to match the transmitter

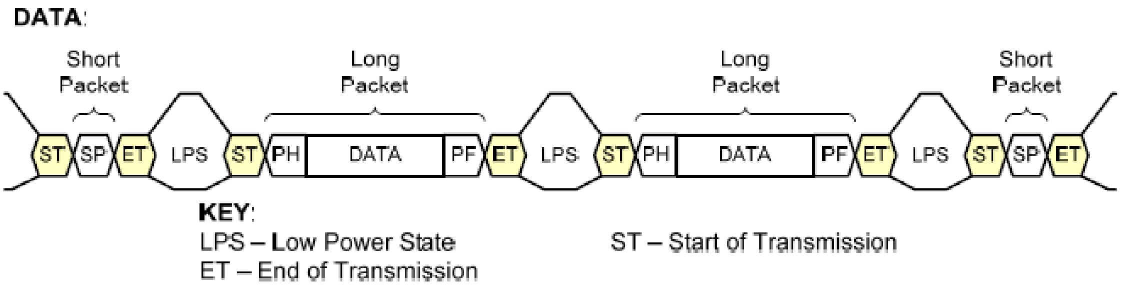
- If M> N then there may be a loss of performance (e.g. frame rate) as the receiver has fewer data Lanes than the transmitter.

Low Level Protocol

The Low Level Protocol (LLP) is a byte orientated, packet based protocol that supports the transport of arbitrary data using **Short and Long pa**cket formats. For simplicity, all examples in this section are single Lane configurations.

Low Level Protocol Features:

- Transport of arbitrary data (Payload independent)
- 8-bit word size
- Support for up to four interleaved virtual channels on the same link
- Special packets for frame start, frame end, line start and line end information
- Descriptor for the type, pixel depth and format of the Application Specific Payload data
- 16-bit Checksum Code for error detection

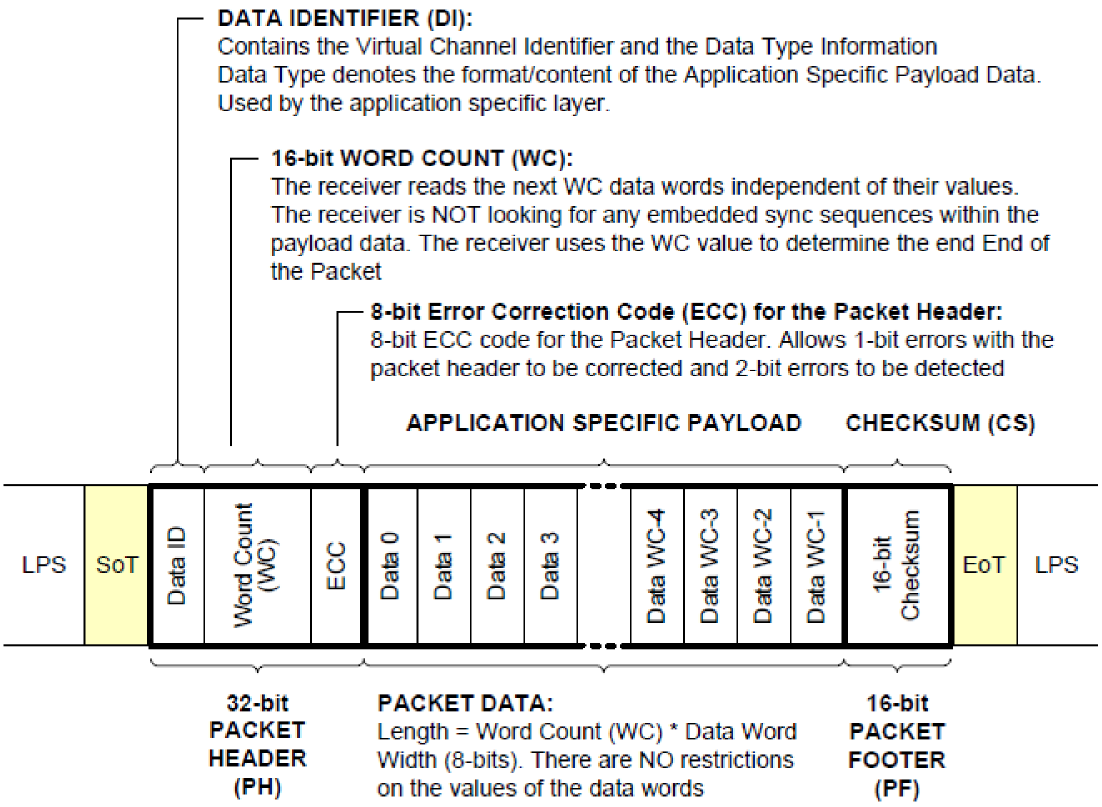


Low Level Protocol Packet Format

Two packet structures are defined for low-level protocol communication: Long packets and Short packets. For each packet structure exit from the low power state followed by the Start of Transmission (SoT) sequence indicates the start of the packet. The End of Transmission (EoT) sequence followed by the low power state indicates the end of the packet.

Low Level Protocol Long Packet Format

Figure shows the structure of the Low Level Protocol Long Packet. A Long Packet shall be identified by **Data Types 0x10 to 0x37**. See Table 3 for a description of the Data Types. A Long Packet shall consist of three elements: a 32-bit Packet Header (PH), an application specific Data Payload with a variable number of 8-bit data words 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 Field and an 8-bit ECC. The Packet footer has one element, a 16-bit checksum.


DATA IDENTIFIER (DI):
Contains the Virtual Channel Identifier and the Data Type Information
Data Type denotes the format/content of the Application Specific Payload Data.
Used by the application specific layer.
16-bit WORD COUNT (WC):
The receiver reads the next WC data words independent of their values.
The receiver is NOT looking for any embedded sync sequences within the payload data. The receiver uses the WC value to determine the end End of the Packet
8-bit Error Correction Code (ECC) for the Packet Header:
8-bit ECC code for the Packet Header. Allows 1-bit errors with the packet header to be corrected and 2-bit errors to be detected
APPLICATION SPECIFIC PAYLOAD
CHECKSUM (CS)
LPS
SoT
Data ID
Word Count (WC)
ECC
Data 0
Data 1
Data 2
Data 3
Data WC-4
Data WC-3
Data WC-2
Data WC-1
16-bit Checksum
EoT
LPS
32-bit PACKET HEADER (PH)
PACKET DATA:
Length = Word Count (WC) * Data Word Width (8-bits). There are NO restrictions on the values of the data words
16-bit PACKET FOOTER (PF)

The Data Identifier defines the Virtual Channel for the data and the Data Type for the application specific payload data.

The Word Count defines the number of 8-bit data words 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 value and the word count value.

After the end of the Packet Header the receiver reads the next Word Count * 8-bit data words of the Data Payload. While reading the Data Payload the receiver shall not look for any embedded sync codes. Therefore, there are no limitations on the value of a data word.

Once the receiver has read the Data Payload it reads the checksum in the Packet Footer. In the generic case, the length of the Data Payload shall be a multiple of 8-bit data words. 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, Checksum and the Short packet 16-bit Data Field shall be transmitted least significant byte first.

After the EoT sequence the receiver begins looking for the next SoT sequence.

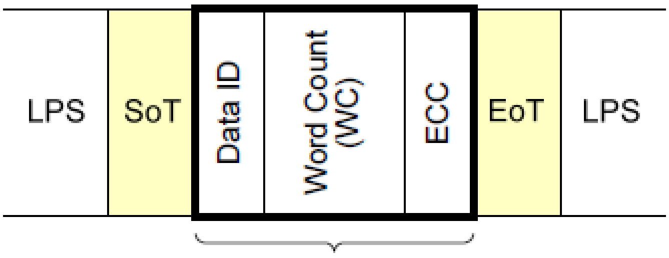
Low Level Protocol Short Packet Format

Figure shows the structure of the Low Level Protocol Short Packet. A Short Packet shall be identified by **Data Types 0x00 to 0x0F**.

A Short Packet shall contain only a Packet Header; a Packet Footer shall not be present. The Word Count Field in the Packet Header shall be replaced by a Short Packet Data Field.
For Frame Synchronization Data Types the Short Packet Data Field shall be the frame number. For Line Synchronization Data Types the Short Packet Data Field shall be the line number. See Table for a description of the Frame and Line synchronization Data Types.

For Generic Short Packet Data Types the content of the Short Packet Data Field shall be user defined.

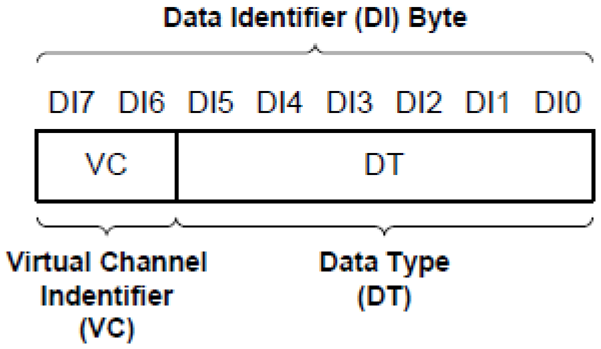
The Error Correction Code (ECC) byte allows single-bit errors to be corrected and 2-bit errors to be detected in the Short Packet.



32-bit SHORT PACKET (SH)
Data Type (DT) = 0x00 – 0x0F

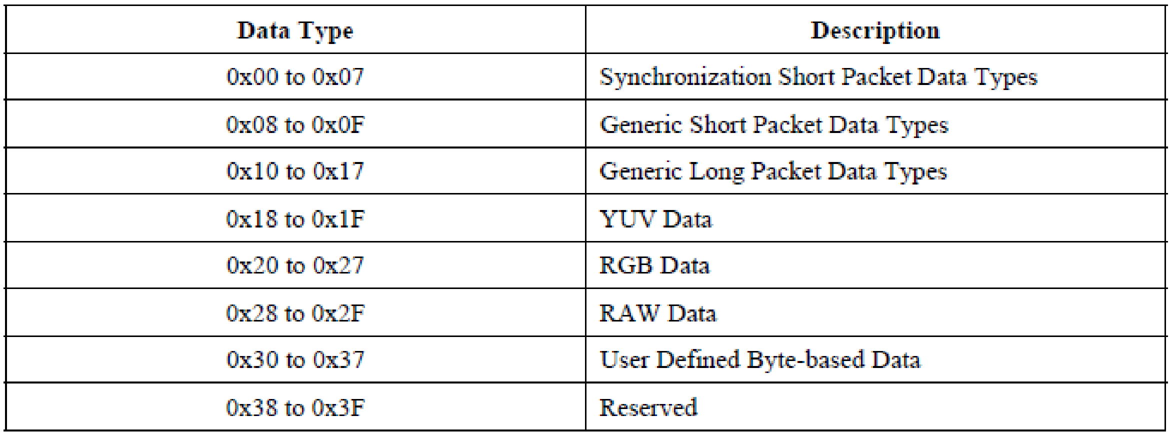
Data Identifier (DI)

The Data Identifier byte contains the Virtual Channel Identifier (VC) value and the Data Type (DT) value as illustrated in Figure. The Virtual Channel Identifier is contained in the two MS bits of the Data Identifier Byte. The Data Type value is contained in the six LS bits of the Data Identifier Byte.


Data Identifier (DI) Byte
DI7 DI6 DI5 DI4 DI3 DI2 DI1 DI0
VC
DT
Virtual Channel Indentifier (VC)
Data Type (DT)

Data Type (DT)

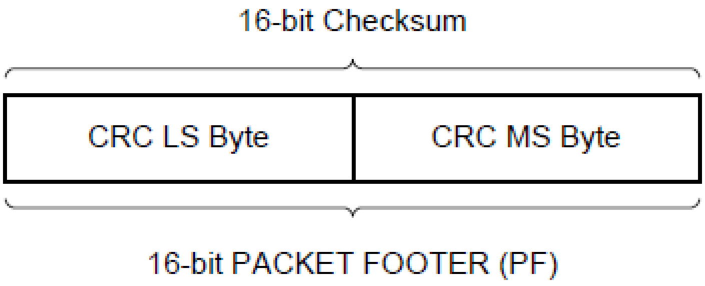
The Data Type value specifies the format and content of the payload data. A maximum of sixty-four data types are supported.
There are eight different data type classes as shown in Table. Within each class there are up to eight different data type definitions. The first two classes denote short packet data types. The remaining six classes denote long packet data types.



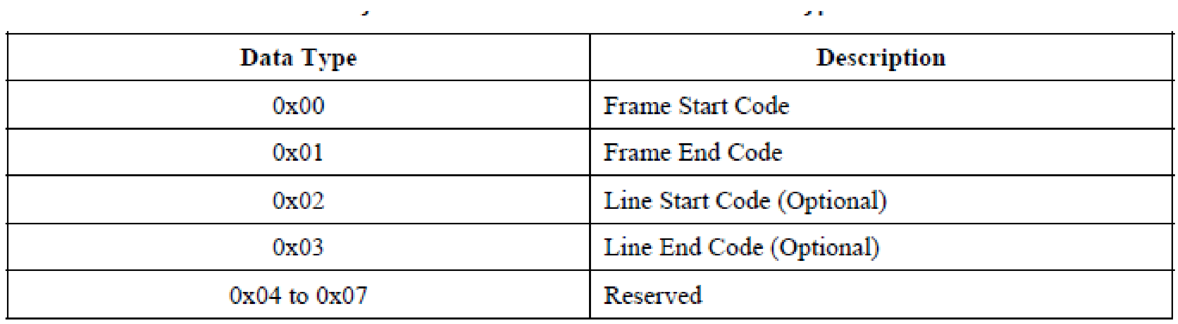
Data Type	Description
0x00 to 0x07	Synchronization Short Packet Data Types
0x08 to 0x0F	Generic Short Packet Data Types
0x10 to 0x17	Generic Long Packet Data Types
0x18 to 0x1F	YUV Data
0x20 to 0x27	RGB Data
0x28 to 0x2F	RAW Data
0x30 to 0x37	User Defined Byte-based Data
0x38 to 0x3F	Reserved

Checksum Generation

To detect possible errors in transmission, a checksum is calculated over each data packet. The checksum is realized as 16-bit CRC. The generator polynomial is x16+x12+x5+x0.



Synchronization Short Packet Data Type Codes



Data Type	Description
0x00	Frame Start Code
0x01	Frame End Code
0x02	Line Start Code (Optional)
0x03	Line End Code (Optional)
0x04 to 0x07	Reserved

Frame Synchronization Packets

Each image frame shall begin with a Frame Start (FS) Packet containing the Frame Start Code. The FS Packet shall be followed by one or more long packets containing image data and zero or more short packets containing synchronization codes. Each image frame shall end with a Frame End (FE) Packet containing the Frame End Code.

Line Synchronization Packets

Line synchronization packets are optional.
For Line Start (LS) and Line End (LE) synchronization packets the Short Packet Data Field shall contain a 16-bit line number. This line number shall be the same for the LS and LE packets corresponding to a given line. Line numbers are logical line numbers and are not necessarily equal to the physical line numbers

The 16-bit line number, when used, shall be non-zero to distinguish it from the case where line number is inoperative and remains set to zero.

The behaviour of the 16-bit line number shall be as one of the following:

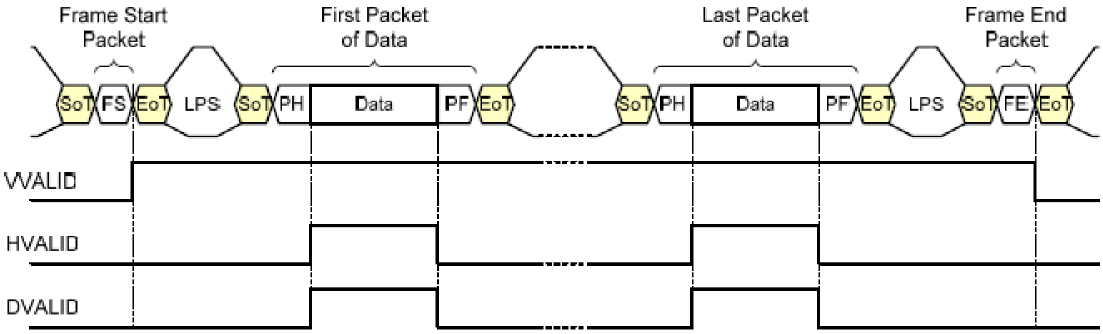
- Line number is always zero – line number is inoperative.

- Line number increments by one for every LS packet within the same Virtual Channel and the same Data Type. The line number is periodically reset to one for the first LS packet after a FS packet. The intended usage is for progressive scan (non- interlaced) video data streams. The line number must be a non-zero value.

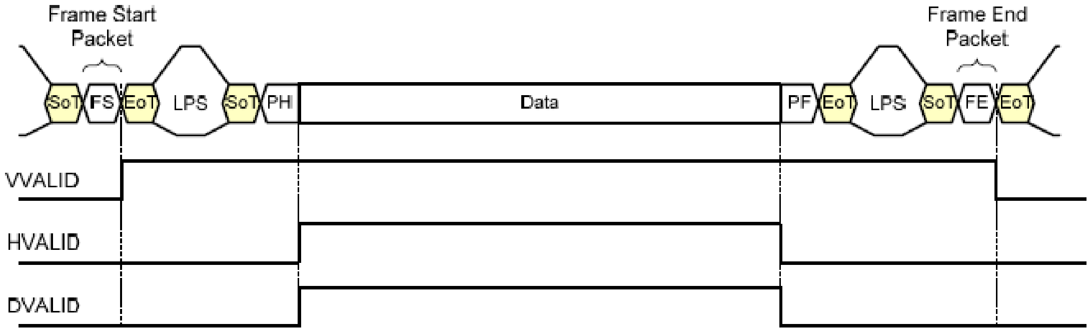
- Line number increments by the same arbitrary step value greater than one for every LS packet within the same Virtual Channel and the same Data Type. The line number is periodically reset to a non-zero arbitrary start value for the first LS packet after a FS packet. The arbitrary start value may be different between successive frames. The intended usage is for interlaced video data streams.

Packet Spacing Examples

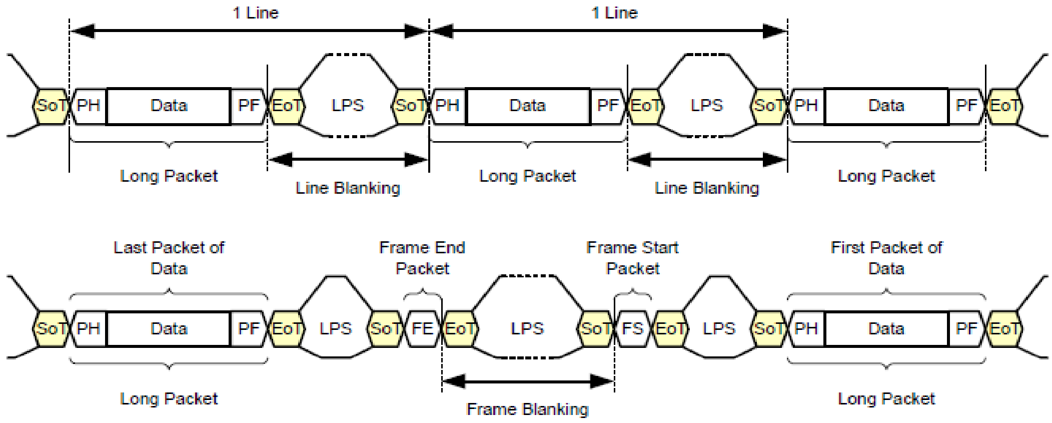
Note that the VVALID, HVALID and DVALID signals in the figures in this section are only concepts to help illustrate the behaviour of the frame start/end and line start/end packets. The VVALID, HVALID and DVALID signals do not form part of the specification.



KEY:
SoT – Start of Transmission
PH – Packet Header
FS – Frame Start
LS – Line Start
EoT – End of Transmission LPS – Low Power State
PF – Packet Footer
FE – Frame End
LE – Line End



KEY:
SoT – Start of Transmission
PH – Packet Header
FS – Frame Start
LS – Line Start
EoT – End of Transmission LPS – Low Power State
PF – Packet Footer
FE – Frame End
LE – Line End



KEY:
SoT – Start of Transmission
PH – Packet Header
FS – Frame Start
LS – Line Start
EoT – End of Transmission LPS – Low Power State
PF – Packet Footer
FE – Frame End
LE – Line End

Colour Spaces

The colour space definitions in this section are simply references to other standards. The references are included only for informative purposes and not for compliance. The colour space used is not limited to the References given

RGB Colour Space Definition

In this specification, the abbreviation RGB means the nonlinear sR'G'B' colour space in 8-bit representation based on the definition of RGB in IEC 61966.

The 8-bit representation results as RGB888. The conversion to the more commonly used RGB565 format is achieved by scaling the 8-bit values to five bits (blue and red) and six bits (green). The scaling can be done either by simply dropping the LSBs or rounding.

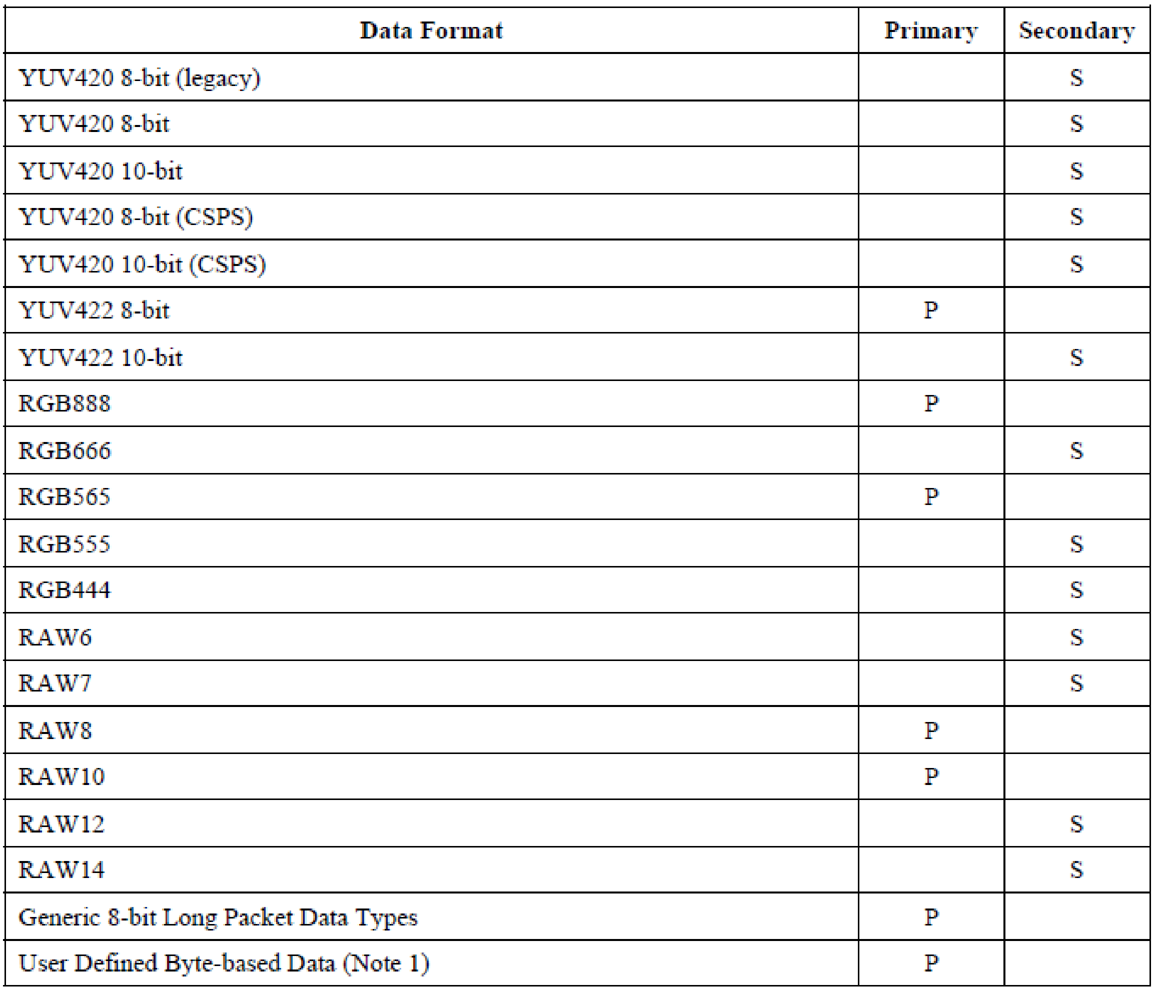
YUV Colour Space Definition

In this specification, the abbreviation YUV refers to the 8-bit gamma corrected Y'CBCR colour space defined in ITU-R BT601.4.

Data Formats

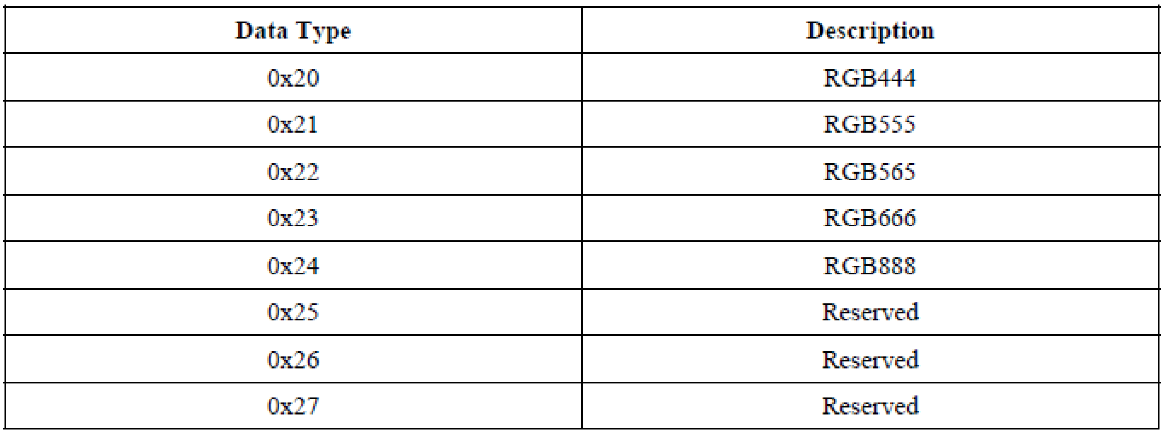
The intent of this section is to provide a definitive reference for data formats typically used in CSI-2 applications. Table 8 summarizes the formats, followed by individual definitions for each format. Generic data types not shown in the table are described in section 11.1. For simplicity, all examples are single Lane configurations.

The formats most widely used in CSI-2 applications are distinguished by a “primary” designation in Table 8. Transmitter implementations of CSI-2 should support at least one of these primary formats. Receiver implementations of CSI-2 should support all of the primary formats. The packet payload data format shall agree with the Data Type value in the Packet Header. See Section 9.4for a description of the Data Type values.



Data Format	Primary	Secondary
YUV420 8-bit (legacy)		S
YUV420 8-bit		S
YUV420 10-bit		S
YUV420 8-bit (CSPS)		S
YUV420 10-bit (CSPS)		S
YUV422 8-bit	P	
YUV422 10-bit		S
RGB888	P	
RGB666		S
RGB565	P	
RGB555		S
RGB444		S
RAW6		S
RAW7		S
RAW8	P	
RAW10	P	
RAW12		S
RAW14		S
Generic 8-bit Long Packet Data Types	P	
User Defined Byte-based Data (Note 1)	P	

RGB Image Data



Data Type	Description
0x20	RGB444
0x21	RGB555
0x22	RGB565
0x23	RGB666
0x24	RGB888
0x25	Reserved
0x26	Reserved
0x27	Reserved

RGB888

RGB888 data transmission is performed by transmitting a BGR byte sequence

Table 17 RGB888 Packet Data Size Constraints

Pixels	Bytes	Bits
1	3	24

Bit order in transmission follows the general CSI-2 rule, LSB first. The pixel to byte mapping is illustrated in Figure 78.

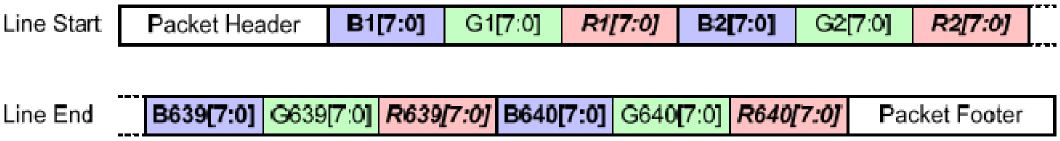


Figure 77 RGB888 Transmission

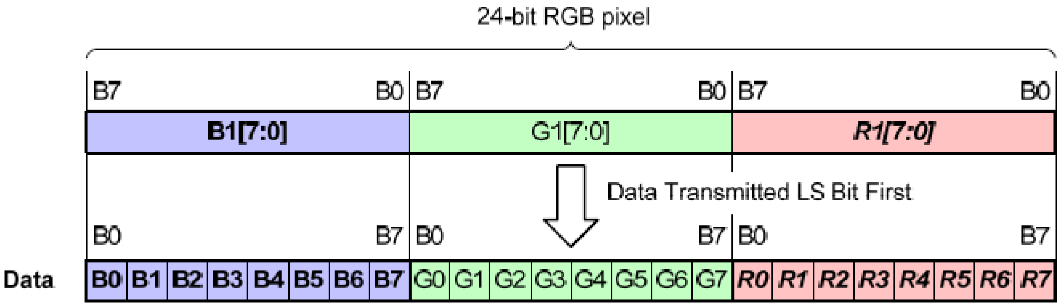


Figure 78 RGB888 Transmission in CSI-2 Bus Bitwise Illustration

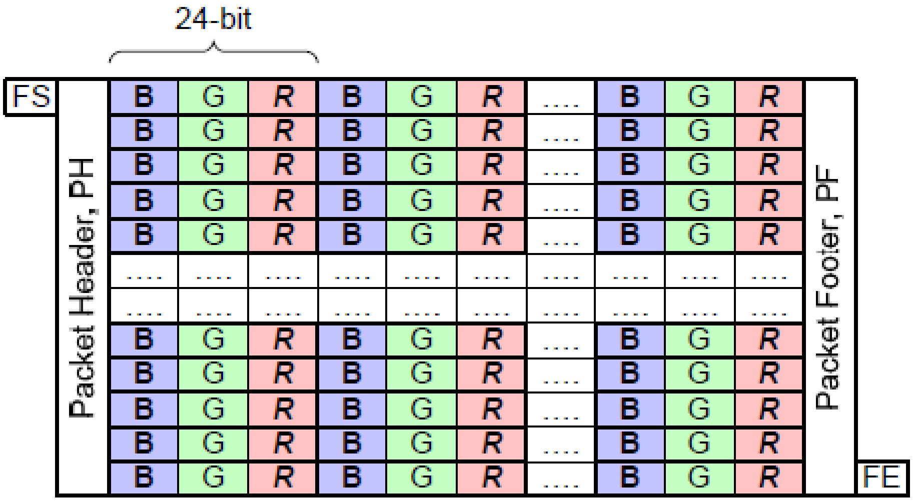


Figure 79 RGB888 Frame Format

RGB565

RGB565 data transmission is performed by transmitting B0…B4, G0…G5, R0…R4 in a 16-bit sequence. This sequence is illustrated in Figure 83. The frame format for RGB565 is presented in the Figure 85

Table 19 RGB565 Packet Data Size Constraints

Pixels	Bytes	Bits
1	2	16

Bit order in transmission follows the general CSI-2 rule, LSB first. In RGB565 case the length of one data word is 16-bits, not eight bits. The word wise flip is done for 16-bit BGR words i.e. instead of flipping each byte (8-bits), each two bytes (16-bits) are flipped. This is illustrated in Figure 84.

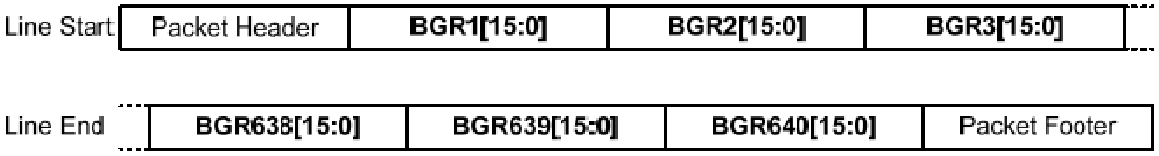


Figure 83 RGB565 Transmission with 16-bit BGR Words

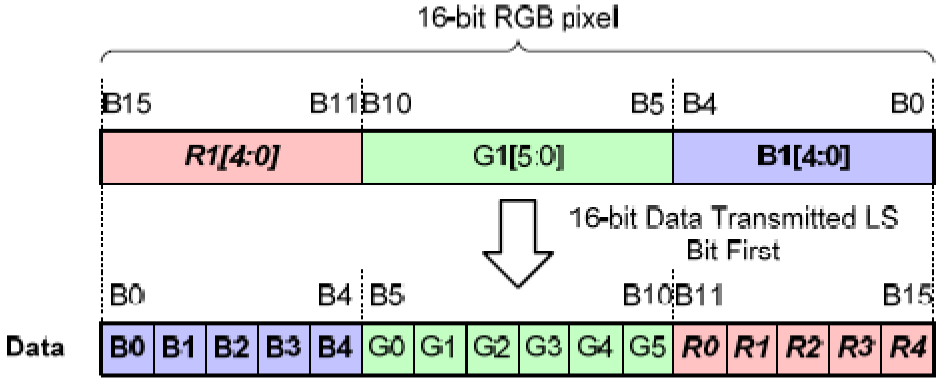
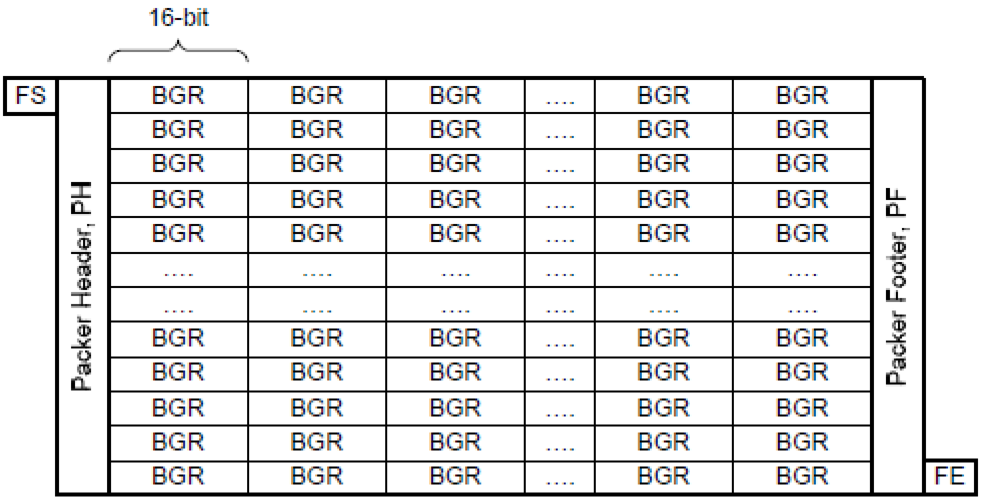


Figure 84 RGB565 Transmission on CSI-2 Bus Bitwise Illustration



BGR	BGR	BGR	....	BGR	BGR
BGR	BGR	BGR	....	BGR	BGR
BGR	BGR	BGR	....	BGR	BGR
BGR	BGR	BGR	....	BGR	BGR
BGR	BGR	BGR	....	BGR	BGR
....	....	....	....	....	....
....	....	....	....	....	....
BGR	BGR	BGR	....	BGR	BGR
BGR	BGR	BGR	....	BGR	BGR
BGR	BGR	BGR	....	BGR	BGR
BGR	BGR	BGR	....	BGR	BGR
BGR	BGR	BGR	....	BGR	BGR

Figure 85 RGB565 Frame Format