ICS 33.160.01
CCS M 70

Group Standard

T/SUCA 001.2—2024

General Purpose Multimedia Interface Specification 1.0 Part 2: The Protocols

Issued on December 28, 2024 Implemented on January 31, 2025

Issued by Shenzhen 8K UHD Video Industry Cooperation Alliance

Contents

Preface

This document was drafted in accordance with GB/T 1.1-2020 *Directives for Standardization - Part 1: Rules for the Structure and Drafting of Standardizing Documents*.

This document is Part 2 of T/SUCA 001 *General Purpose Multimedia Interface Specification*. T/SUCA 001 includes the following parts proposed to be published:

—Part 1: Architecture, specifying a general purpose multimedia interface (GPMI) architecture that supports information transmission between consumer electronic devices.

—Part 2: Protocols, specifying the protocols of the electrical layer, logical layer, transport layer, and adaptation layer of the GPMI.

—Part 3: Connectors and Cables, specifying the technical requirements for connectors and cables using GPMI Type-B.

—Part 4: Power Supply, describing the architecture of the power supply for GPMIs, and specifying the electrical characteristics and timing requirements, physical layer, protocol layer, application layer, and power input and output requirements.

—Part 5: Alternate Mode over Type-C, specifying the method of using GPMI signals via USB Type-C ports.

This Standard was proposed by and is under the centralized management of Shenzhen 8K UHD Video Industry Cooperation Alliance.

This standard is drafted by: Huawei Technologies Co., Ltd., HiSilicon Technologies Co., Ltd., China Electronics Standardization Institute, Shenzhen Skyworth-RGB Electronics Co., Ltd., Shenzhen Skyworth Digital Technology Co., Ltd., China Mobile (Hangzhou) Information Technology Co., Ltd., National Engineering Laboratory for Digital TV (Beijing), Guilin University of Electronic Science and Technology, Shenzhen CESI Information Technology Co., Ltd., Sinolink Technologies (Beijing) Co., Ltd., Hisense Visual Technology Co.,Ltd., Konka Group Co., Ltd., Hangzhou Hikvision Digital Technology Co., Ltd., Shenzhen National Engineering Laboratory of Digital Television Co., Ltd., Guangdong National UHD Video Innovation Center, Shenzhen Bajiuling Optoelectronics Technology Co., Ltd., Shenzhen Wanliyan Technology Co., Ltd., Ceyear Technologies Co., Ltd., Jiangsu Anlan-WK Electronics Co., Ltd., Malanshan Audio&Video Laboratory, Guangdong Wire & Cable Association, and Shenzhen 8K UHD Video Industry Cooperation Alliance.

Main drafters of this standard: Wei Jiayi, Fan Kefeng, Li Zhengbing, Sun Qifeng, He Jianhong, Dong Guiguan, Zhao Xiaoying, Zhou Weiguang, Wu Dongxing, Zhang Ran, Xiao Kai, Shi Rujie, Liu Yan, Chen Yanqin, Yang Bing, Fu Qiang, Ren Zhongyue, Shi Xuan, Zhao Gan, Xu Yaoling, Yu Yang, Li Xinguo, Mao Ke, Xu Chuanpei, Zhang Manhua, Liang Jiyun, Feng Nanfei, Su Yi, Chen Yijun, Liang Yutong, Li Siyuan, Fu Yuhong, Su Jinshui, Zhang Linyu, Ni Xin, Long Shiqiang, Zheng Lifang, Lin Zhigeng, Peng Hui, Yang Xin, Xu Hui, Qiao Houcai, Qi Jian, Li Yun, Long Hua, Chen Yunzhou, Zhou Jun, Gong Shuqiang, Yuan Guoping, Qiu Xiaoyong, Jin Hua, Zhang Junping, Li Zaikuan, Meng Xiance, Gu Anwen, and Zhai Yongqi.

General Purpose Multimedia Interface Specification Part 2: The Protocols

1 Scope

This document is applicable to the design and development of the general purpose multimedia interface (GPMI). It may be used as a reference for devices using general purpose multimedia interfaces.

This document specifies the protocols of the electrical layer, logical layer, transport layer, and adaptation layer of the GPMI, and describes the packaging method of audio and video adapters and management adapters.

This document is applicable to the design, development, test, and application of the GPMI, and can also be used as a reference for information transmission between other electronic devices.

2 Normative References

The following documents constitute, through normative references in the text, indispensable provisions of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

T/SUCA 001.1-2024 General Purpose Multimedia Interface Specification Part 1: Architecture

T/SUCA 001.3-2024 General Purpose Multimedia Interface Specification Part 3: Connectors and Cables

T/SUCA 001.4-2024 General Purpose Multimedia Interface Specification Part 4: Power Supply

T/SUCA 001.5-2024 General Purpose Multimedia Interface Specification Part 5: Alternate Mode over Type-C

IEC 60958-1 Digital Audio Interface—Part 1: General

IEC 60958-3 Digital Audio Interface—Part 3: Consumer Applications

IEC 60958-4 Digital Audio Interface—Part 4: Professional Applications

IEC 61937 Digital Audio—Interface for Nonlinear PCM Encoded Audio Bitstreams Applying IEC 60958

CTA-861-H A DTV Profile for Uncompressed High Speed Digital Interfaces

CVT VESA Coordinated Video Timings Standard

DMT Display Monitor Timing Version 1.0, Rev. 13

3 Terms, Definitions, and Abbreviations

3.1 Terms and Definitions

The terms and definitions defined in T/SUCA 001.1 and the following apply to this document.

3.1.1 transmitter side/receiver side

The two ends of a link (usually containing multiple lanes) are called the transmitter side and the receiver side respectively, and the data flows from the transmitter side to the receiver side.

3.1.2 credit

The specific byte length in cache, which is used to describe the size of the cache space.

3.1.3 device comprehensive capability description

A system framework and data structure for declaring the capabilities supported by a device, such as device parameters, performance attributes, and audio/video capabilities.

3.1.4 retimer

A circuit that can generate an output signal using a clock independent of the input signal.

3.2 Abbreviations

For the purposes of this document, the following abbreviations apply.

ADCP: advanced digital content protection

ASP: audio sample packet

AVP: active video packet

Ack/Nack: Ack/Nack message

BER: bit error rate

CDR: clock and data recovery

CF: control frame

CFSLM: control frame send location message

CLFM: clock lock feedback message

CRC: cyclic redundancy check

CTLE: continuous-time linear equalizer

DCCD: device comprehensive capability description

DDJ: data dependent jitter

DE: display enable

DFE: decision-feedback equalizer

DIP: descriptive information packet

DW: double word

EDP: encryption description packet

EMI: electromagnetic interference

ERR_RM: error report message

EQFM: equilibrium feedback message

FEC: forward error correction

FFE: feed-forward equalizer

GPMI: general purpose multimedia interface

HBP: horizontal blanking packet

HSync: horizontal synchronization

JTF: jitter transfer function

KDP: key distribution packet

LDAM: lane direction adjust message

LLB: logical layer block

LLCF: logical layer control frame

LLCF_DS: logical layer control frame, data start

LLCF_EI: logical layer control frame, electrical idle

LLCF_EIE: logical layer control frame, electrical idle exit

LLCF_TS0: logical layer control frame, training sequence 0

LLCF_TS1: logical layer control frame, training sequence1

LLCF_TS2: logical layer control frame, training sequence2

LLDP: logical layer data packet

LLFM: lane lock feedback message

LLMMP: logical layer main link management packet

LLMMT: logical layer management message type

LLPH: logical layer packet header

LLPL: logical layer packet length

LLPP: logical layer packet payload

LLPT: logical layer packet type

LLSMP: logical layer sideband link management packet

LLSP: logical layer sideband link packet

LNSM: lane state machine

LPCM: linear pulse-code modulation

LWAM: link width adjust message

NRZ: non-return-to-zero

PLL: phase lock loop

PLLC: perceptual lossless compression

PRBS: pseudo-random binary sequence

RBuff: receiver buffer

RJ: random jitter

RL: return loss

RS: Reed-Solomon code

RX: receiver

SBRX: sideband link receiver

SBTX: sideband link transmitter

SJ: sinusoidal jitter

SSC: spread spectrum clocking

ShuttleID: shuttle identification

TBuff: transmitter buffer

TDR: time domain reflectometry

TJ: total jitter

TLDP: transport layer data packet

TLMP: transport layer management packet

TLMDP: transport layer management data packet

TLP: transport layer packet

TP: test point

TSM: training start message

TX: transmitter

UDJ: uncorrelated and deterministic jitter

UDJ_{ADJ}: uncorrelated and deterministic jitter adjusted value

UJ: uncorrelated jitter

VBP: vertical blanking packet

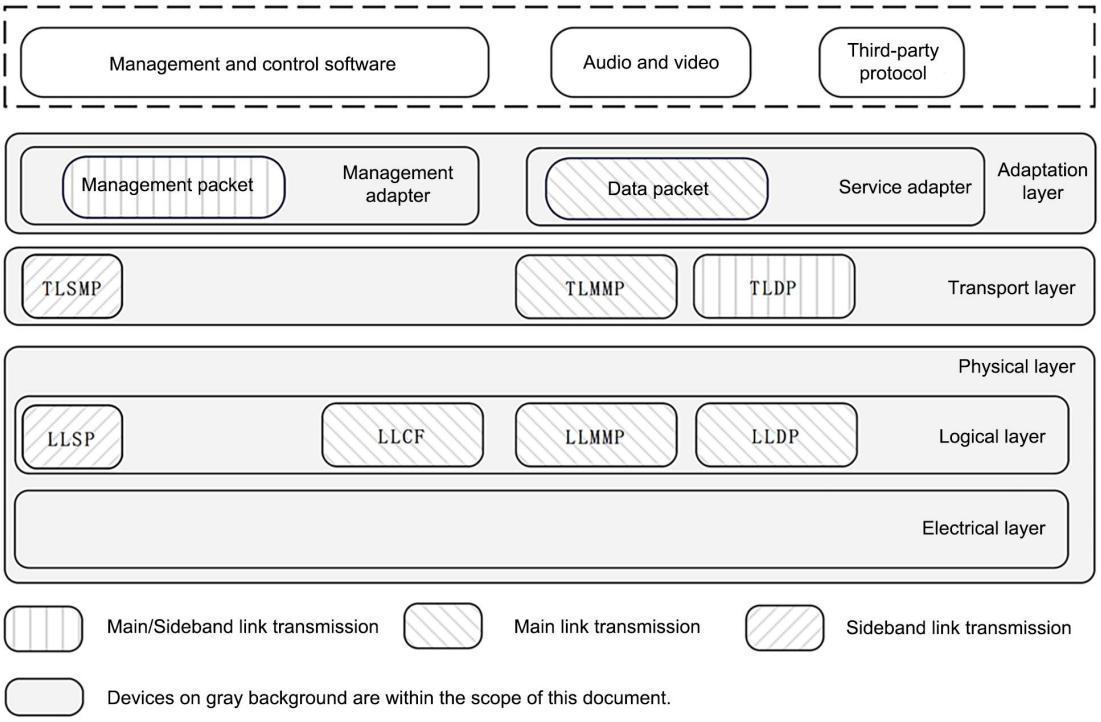
VSync: vertical synchronization

WRR: weighted round robin

4 Protocol Architecture

The GPMI protocol architecture is shown in Figure 1. The protocol stack is divided into a physical layer, a transport layer, and an adaptation layer.

Figure 1 Schematic diagram of the protocol stack



The physical layer is divided into a logical layer and an electrical layer. The electrical layer implements signal modulation, sending, receiving, equilibrium, spread spectrum, and clock recovery, while the logical layer handles FEC encoding/decoding, scrambling/descrambling, and link training.

The transport layer processes and forwards service streams, such as video and audio, as well as management and control information, and performs bandwidth management for all service flows.

The adaptation layer connects end devices and external components, and through corresponding adapters, implements functions such as audio and video transmitting service adaptation, audio and video receiving service adaptation, third-party protocol service adaptation, and management and control adaptation. The adaptation layer includes a management adapter and a service adapter.

5 Electrical Layer

5.1 Overview

To ensure the quality of electrical layer signals, this section describes the functions, technical requirements, and test methods of the electrical layer, including the lane structure, main link electrical layer technical requirements, and sideband link electrical layer technical requirements.

The four-lane mode in this section corresponds to the simplified-specification mode in Table 133, and the eight-lane mode corresponds to the full-specification mode in Table 133.

5.2 Lane Structure

As shown in Figures 2 and 3, the GPMI lane structure includes a main link lane and a sideband link lane. The main link includes four differential lanes, each of which contains a transmitter (TX)

and a receiver (RX) at both sides. The sideband link lane includes two single-ended lanes, each of which includes a sideband link transmitter (SLTX) and a sideband link receiver (SLRX). When the sideband link electrical layer is working, two lanes are opened at the same time in opposite directions to achieve full-duplex communication.

Figure 2 Schematic diagram of four-lane mode, forward insertion

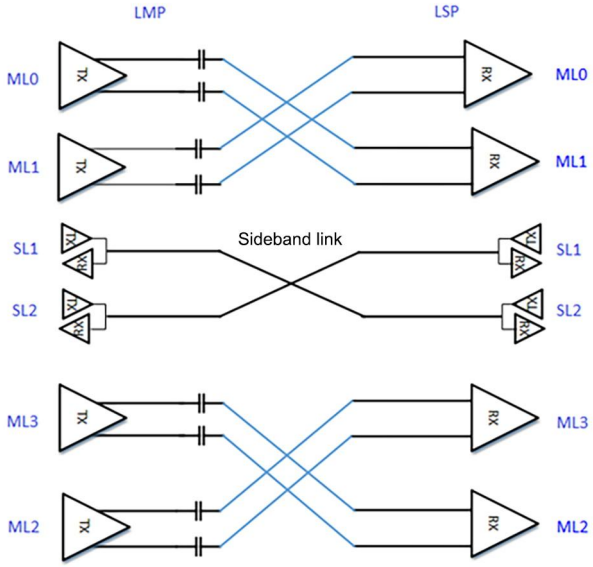
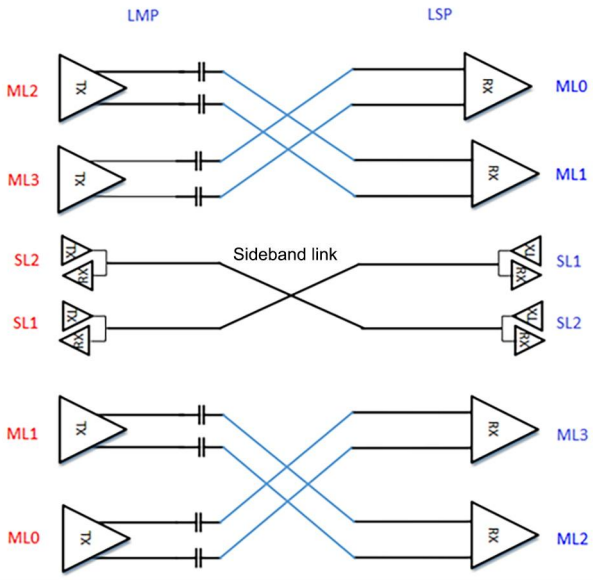


Figure 3 Schematic diagram of four-lane mode, reverse insertion



The reference design of the eight-lane electrical layer is shown in Appendix A.1.

5.3 Main Link

5.3.1 Basic Requirements

The main link differential lane signal adopts NRZ coding and AC coupling. The transmission rate of each differential lane can be configured to 2 Gbps, 4 Gbps, 6 Gbps, 8 Gbps, 10 Gbps, 12 Gbps, 16 Gbps, 20 Gbps, and 24 Gbps. The rates in the same orientation shall be the same. The minimum requirement for the Pre-FEC Bit Error Rate (BER) at the electrical layer is 1×10^{-12}. The Spread Spectrum Clocking (SSC) shall be supported to reduce Electromagnetic Interference (EMI).

The system configuration prior to the transmitter and receiver test is shown in Appendix B.

5.3.2 Transmitter

5.3.2.1 General Electrical Parameters

The general parameters of the TX include:

(a) Differential signal peak-to-peak value

As shown in Figure 4, the differential signal ($V_{DIFF}(\mathrm{t})$) is defined as the voltage difference between the positive signal ($V_{D+}(\mathrm{t})$) and the negative signal ($V_{D-}(\mathrm{t})$) of the transmission line, that is:

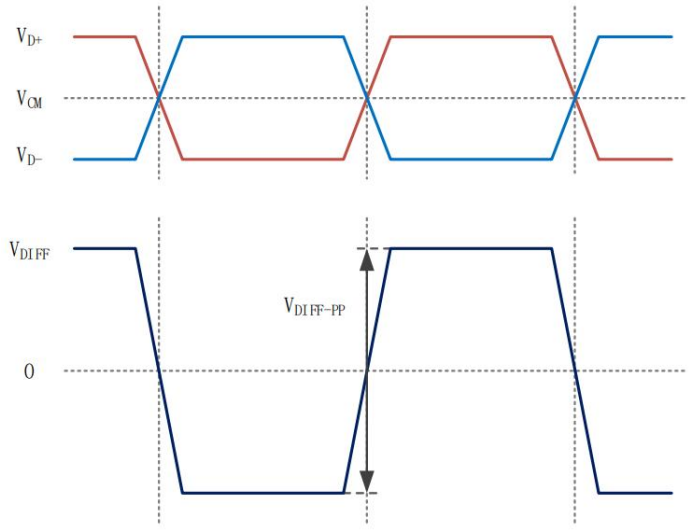
$$V_{DIFF}(\mathrm{t}) = V_{D+}(\mathrm{t}) - V_{D-}(\mathrm{t}) \quad \text{.....(1)}$$

The differential signal peak-to-peak value ($V_{TX-DIFF-PP}$) is defined as:

$$V_{\mathrm{DIFF-PP}} = \max(V_{DIFF}) - \min(V_{DIFF}) \quad \text{.....(2)}$$

The range of the differential signal peak-to-peak value ($V_{TX-DIFF-PP}$) shall comply with the provisions in Table 1.

Figure 4 Schematic diagram of the differential signal



(b) Terminating resistor

To reduce signal reflection and return loss during transmission, a terminating resistor should be connected at the end to match the impedance.

The impedance value range of the terminating resistor ($R_{TX-DIFF-DC}$) shall comply with the provisions of Table 1.

(c) AC coupling capacitor

The TX and RX use AC coupling for data transmission.

The value range of AC coupling capacitor (C_{AC}) shall comply with the provisions in Table 1.

(d) Short-circuit current limitation

To protect the circuit, the main link should specify the maximum short-circuit current $I_{TX-SHORT}$ of the TX and RX, and the value range should comply with the provisions of Table 1.

(e) TX AC common mode level

As shown in Table 1, the common mode level of the differential signal ($V_{CM(t)}$) is defined as the average value of the positive and negative signals of the transmission line, that is:

$$V_{CM(t)} = (V_{D+}(t) + V_{D-}(t))/2 \quad \text{.....(3)}$$

TX AC common mode level peak-to-peak value is defined as:

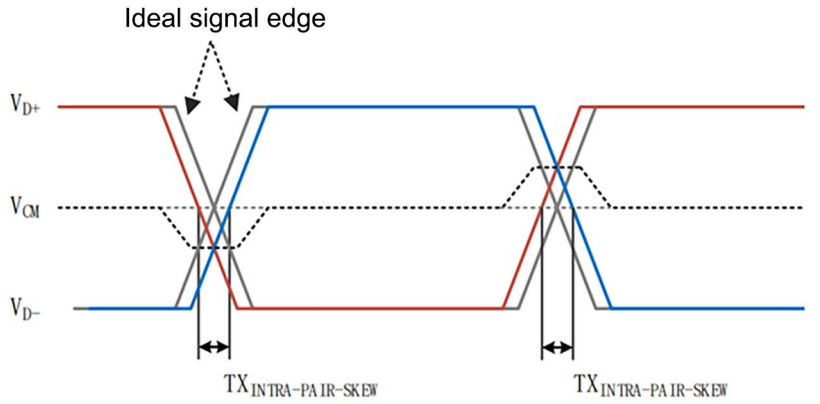
$$V_{TX-CM-PP} = \max(V_{CM}) - \min(V_{CM}) \quad \text{.....(4)}$$

The range of the common mode level peak-to-peak value ($V_{TX-CM-PP}$) shall comply with the provisions of Table 1.

(f) Differential intra-pair skew

The differential intra-pair skew is defined as the difference in the transmission delay of the signal between the positive and negative poles of the differential line. As shown in Table 1, the differential intra-pair skew is measured by the time difference between the positive and negative pole signals passing through (V_{CM}). The main link TX shall specify the differential intra-pair skew ($TX_{INTRA-PAIR-SKEW}$). The value range shall comply with the provisions of Table 1.

Figure 5 Differential intra-pair skew



(g) Differential inter-pair skew

The differential inter-pair skew is the difference in transmission delay between different TX output differential pairs. The main link TX shall specify the differential inter-pair skew ($TX_{INTER-PAIR-SKEW}$). The value range shall comply with the provisions of Table 1.

The technical requirements for the general electrical parameters of the TX are shown in Table 1.

Table 1 General electrical parameters of the TX

Parameter Name	Symbol	Minimum Value	Typical Value	Maximum Value	Unit	Remarks
Differential signal peak-to-peak value in normal amplitude mode	$V_{TX-DIFF-PP}$	400	1000	1400	mV	@TP2
Terminating resistor	$R_{TX-DIFF-DC}$	70	90	110	Ω	@TP2
AC coupling capacitor	C_{AC}	75		280	nF	@TP2
TX short-circuit current limitation	$I_{TX-SHORT}$			82.5	mA	3.3 V/40 Ω
TX AC common mode level peak-to-peak value	$V_{TX-CM-PP}$			100	mV	This parameter should be tested.
Differential intra-pair skew	$TX_{INTRA-PAIR-SKEW1}$			0.4	UI	@TP3_EQ, 6 Gbps+
Differential intra-pair skew	$TX_{INTRA-PAIR-SKEW2}$			75	ps	@TP2, 2/4 Gbps
Differential inter-pair skew TP3	$TX_{INTER-PAIR-SKEW1}$			450	UI	@TP3, 6 Gbps+; Skew allocation (informative): ● The TP1 skew of the transmitter side does not exceed 80 UI. ● Each retimer skew does not exceed 80 UI. ● The skew introduced by the transmission medium (PCB trace + cable) does not

Parameter Name	Symbol	Minimum Value	Typical Value	Maximum Value	Unit	Remarks
						exceed 50 UI.
Differential inter-pair skew TP2	$TX_{INTER-PAIR-SKEW2}$			270	UI	@TP2, 2/4 Gbps; Skew allocation (informative): ● The TP1 skew of the transmitter side does not exceed 80 UI. ● Each retimer skew does not exceed 80 UI. ● (3) The skew introduced by the transmission medium (PCB trace) does not exceed 30 UI.

5.3.2.2 Rate-related Electrical Parameters

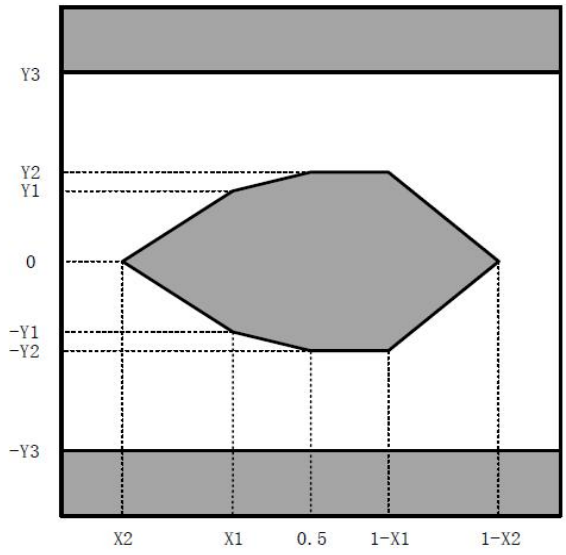
The TX rate-related parameters are as follows, where the eye diagram horizontal skew is measured based on no less than 1×10^6 UI, and the jitter parameters are extrapolated to 1×10^{-12} BER based on the measured jitter:

(a) Eye diagram parameters

The eye diagram template of the TX signal at 2 Gbps and 4 Gbps rates is shown in Figure 6. The inner template of the eye diagram is an octagon symmetrical along the horizontal axis. This standard quantitatively describes the eye diagram template through the following parameters:

- Unit interval (UI): the time width for transmitting the minimum information unit (bit), which is also the unit period of eye diagram measurement.
- Eye diagram horizontal skew (X1, X2): the horizontal skew of two measuring points of the template in an eye diagram, in UI.
- Inner eye height (Y1, Y2): the minimum allowable eye height at the horizontal skew X1 and 0.5 UI of the eye template, respectively.
- External eye height (Y3): the maximum allowable eye height by the eye diagram template.

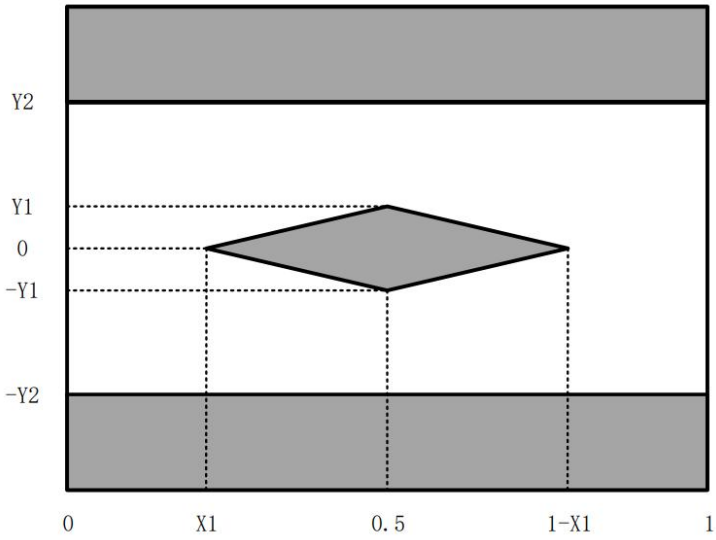
Figure 6 Eye diagram template at 2 Gbps and 4 Gbps



The eye diagram templates of TX signals at 6 Gbps, 8 Gbps, 10 Gbps, 12 Gbps, 16 Gbps, 20 Gbps, and 24 Gbps are shown in Table 7. This standard quantitatively describes the eye diagram template through the following parameters:

- Unit interval (UI).
- Eye diagram horizontal skew (X1): the allowable horizontal skew of eye diagram diamond template, in UI.
- Inner eye height (Y1), external eye height (Y2): the minimum and maximum values of the eye height allowed by the eye diagram template, respectively. The measured horizontal position is at the center of the eye diagram at 0.5 UI.

Figure 7 TX eye diagram templates of 6 Gbps, 8 Gbps, 10 Gbps, 12 Gbps, 16 Gbps, 20 Gbps, and 24 Gbps



(b) Jitter parameters

TJ is divided into DDJ and UJ according to whether the jitter is data-dependent or not, and satisfies:

$$TJ = \mathrm{DDJ} + \mathrm{UJ} \quad \text{.....(5)}$$

UJ can be further divided into random jitter (RJ) and uncorrelated and deterministic jitter (UDJ):

$$UJ = \mathrm{RJ} + \mathrm{UDJ} \quad \text{.....(6)}$$

Note: RJ is 14.7 times the RMS value of RJ.

At 2 Gbps and 4 Gbps rates, the eye diagram and jitter parameters at TP2 in this document shall comply with the provisions of Table 2.

Table 2 TP2 eye diagram and jitter indicators at 2 Gbps and 4 Gbps rates

Parameter Name	Symbol	Minimum Value	Maximum Value	Unit	Remarks
UI	UI	499.85@2 Gbps	500.15@2 Gbps	ps	±300 ppm, SSC effect not considered
		249.925@4 Gbps	250.075@4 Gbps		
Inner eye height	Y1	140		mV	
Inner eye height	Y2	180		mV	
External eye height	Y3		700	mV	
Eye diagram horizontal skew	X1		0.34	UI	
Eye diagram horizontal skew	X2		0.18	UI	
TJ	TJ		0.5	UI	
DDJ	DDJ		0.2	UI	TX FFE enablement allowed
UJ	UJ		0.3	UI	
RJ	RJ		0.15	UI	

At 6 Gbps and 8 Gbps rates, the eye diagram and jitter parameters at TP3_EQ (CTLE only) in this standard shall comply with the provisions of Table 3.

Table 3 TP3_EQ eye diagram and jitter indicators at 6 Gbps and 8 Gbps rates

Parameter Name	Symbol	Minimum Value	Maximum Value	Unit	Remarks

Parameter Name	Symbol	Minimum Value	Maximum Value	Unit	Remarks
UI	UI	166.6166@6 Gbps	166.7167@6 Gbps	ps	±300 ppm, SSC effect not considered; This parameter should be tested at TP2.
		124.9625@8 Gbps	125.0375@8 Gbps		
Inner eye height	Y1	70		mV	
External eye height	Y2		650	mV	
Eye diagram horizontal skew	X1		0.24	UI	
TJ	TJ		0.6	UI	
DDJ	DDJ		0.3	UI	TX FFE enablement allowed
UJ	UJ		0.3	UI	
RJ	RJ		0.14	UI	

The eye diagram and jitter parameters of TP3_EQ (CTLE+DFE, optional) in this standard shall comply with the provisions of Tables 4 to 7 at 10 Gbps, 12 Gbps, 16 Gbps, 20 Gbps, and 24 Gbps rates.

Table 4 TP3_EQ eye diagram and jitter indicators at 10 Gbps and 12 Gbps rates

Parameter Name	Symbol	Minimum Value	Maximum Value	Unit	Remarks
UI	UI	99.97@10 Gbps	100.03@10 Gbps	ps	±300 ppm, SSC effect not considered; This parameter should be tested at TP2.
		83.3083@12 Gbps	83.3583@12 Gbps		
Inner eye height	Y1	55		mV	
External eye height	Y2		650	mV	
Eye diagram horizontal skew	X1		0.24	UI	
TJ	TJ		0.6	UI	SSC included
DDJ	DDJ		0.3	UI	TX FFE enablement allowed

Parameter Name	Symbol	Minimum Value	Maximum Value	Unit	Remarks
UJ	UJ		0.3	UI	
RJ	RJ		0.14	UI	

Table 5 TP3_EQ eye diagram and jitter indicators at 16 Gbps rate

Parameter Name	Symbol	Minimum Value	Maximum Value	Unit	Remarks
UI	UI	62.48125	62.51875	ps	±300 ppm, SSC effect not considered; This parameter should be tested at TP2.
Inner eye height	Y1	50		mV	
External eye height	Y2		650	mV	
Eye diagram horizontal skew	X1		0.24	UI	
TJ	TJ		0.6	UI	SSC included
DDJ	DDJ		0.3	UI	TX FFE enablement allowed
UJ	UJ		0.3	UI	
RJ	RJ		0.14	UI	

Table 6 TP3_EQ eye diagram and jitter indicators at 20 Gbps rate

Parameter Name	Symbol	Minimum Value	Maximum Value	Unit	Remarks
UI	UI	49.985	50.015	ps	±300 ppm, SSC effect not considered; This parameter should be tested at TP2.
Inner eye height	Y1	50		mV	
External eye height	Y2		650	mV	
Eye diagram horizontal skew	X1		0.24	UI	
TJ	TJ		0.62	UI	SSC included

Parameter Name	Symbol	Minimum Value	Maximum Value	Unit	Remarks
DDJ	DDJ		0.32	UI	TX FFE enablement allowed
UJ	UJ		0.3	UI	
RJ	RJ		0.14	UI	

Table 7 TP3_EQ eye diagram and jitter indicators at 24 Gbps rate

Parameter Name	Symbol	Minimum Value	Maximum Value	Unit	Remarks
UI	UI	41.6542	41.6792	ps	±300 ppm, SSC effect not considered; This parameter should be tested at TP2.
Inner eye height	Y1	45		mV	
External eye height	Y2		650	mV	
Eye diagram horizontal skew	X1		0.24	UI	
TJ	TJ		0.62	UI	SSC included
DDJ	DDJ		0.32	UI	TX FFE enablement allowed
UJ	UJ		0.3	UI	
RJ	RJ		0.14	UI	

5.3.2.3 Spread Spectrum Clocking

The TX uses the SSC technology to reduce EMI of the signal. The parameters of the SSC are shown in Figure 8 specifies the parameter technical requirements, such as clock offset range, down spread range, and spread frequency.

Figure 8 SSC parameters

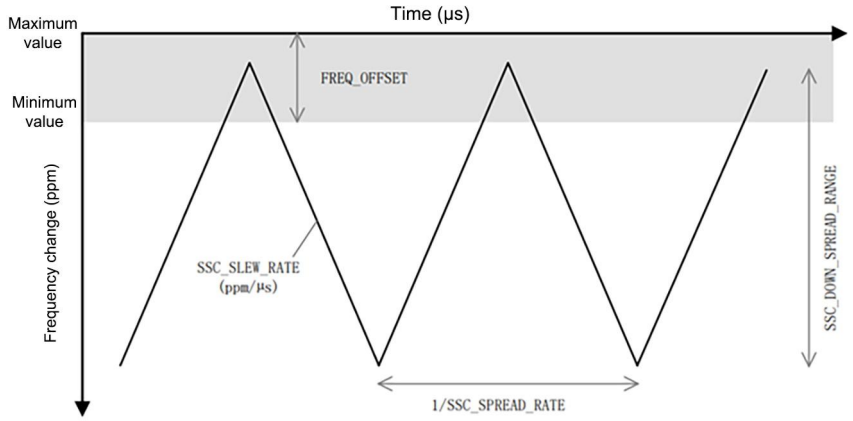


Table 8 SSC parameters

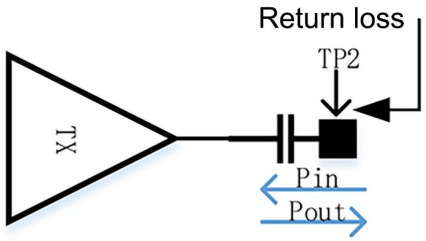
Parameter Name	Symbol	Minimum Value	Maximum Value	Unit	Remarks
Clock offset range	FREQ_OFFSET	–300	300	ppm	Note 1
Down spread range	SSC_DOWN_SPREAD_RANGE	0.4	0.5	%	
Spread frequency	SSC_SPREAD_RATE	30	33	KHz	
Spread spectrum deviation	SSC_PHASE_DEVIATION	2.5	22	ns pp	Note 2
Frequency deviation rate	SSC_SLEW_RATE		1250	ppm/us	

Note 1: SSC_DOWN_SPREAD_RANGE represents the percentage of the maximum drop in signal frequency after modulation relative to the frequency before modulation. The main link signal adopts a down-spread modulation, and the actual operating frequency of the modulated system is always below the nominal frequency. All spread spectra are located at TP2.

Note 2: The transmitter side uses the PRBS31 code pattern, performs tests at TP2, and uses a second-order low-pass filter with a gain of –3 dB at 5 MHz for the phase. The phase accumulation time is 0.5 us, and the PRBS31 polynomial is $Y(x) = x^{31} + x^{28} + 1$.

5.3.2.4 Return Loss

Figure 9 Return loss of the TX



As shown in Figure 9, the TX return loss (RL) is defined as the ratio of the TP2 point reflection power *P*out to the input power *P*in, and the formula is as follows:

$$\text{Return Loss(dB)} = 10 \cdot log_{10}\frac{Pout}{Pin} \qquad (7)$$

In the formula:

*P*out: reflected power, in milliwatts (mW);

*P*in: input power, in milliwatts (mW).

Input power *P*in can be differential input power or common mode input power, corresponding to differential RL and common mode RL, respectively

When the differential RL of the TX is measured, its reference differential impedance (including lane and terminating resistor) shall match the impedance and be measured with the connector. Errors caused by the test fixture shall be compensated and shall not affect the test results.

The differential RL (RL_{DIFF}) shall meet the following formula requirements:

$$RL_{DIFF} = \begin{cases} f_{\text{Nyquis}} \le 4GHz: & \\ -6.5 & f \le 2.5\text{GHz} \\ -3 & 2.5GHz < f \le 5GHz \\ f_{\text{Nyquis}} > 4GHz: & \\ -8 & f \le 2.5\text{GHz} \\ -1.5 + 9.5 \cdot log_{10}\left(\frac{f}{14.4}\right) & 2.5GHz < f \le 12\text{GHz} \\ -1.5 & f > 12\text{GHz} \end{cases} \qquad (8)$$

In the formula:

f_{Nyquist}: maximum Nyquist frequency supported by the chip, in GHz.

Note: For 16 Gbps NRZ signals, the Nyquist frequency is 8 GHz.

The common mode RL (RL_{CM}) shall meet the following formula requirements:

$$RL_{CM} = \begin{cases} -6 & f \le 2.5GHz \\ -3 & 2.5Ghz < f \le f_{\text{Nyquist}} \end{cases} \qquad (9)$$

5.3.2.5 Full-Link Insertion Loss

The Type-B full-link insertion loss (IL) shall comply with the provisions of Table 9, and the Type-C full-link IL shall comply with the provisions of Table 10. The device IL in the table includes chip packaging, PCB, connectors, and others.

Note: The full-link IL is end-to-end, including chip packaging, PCB, connectors, cables, and others.

Table 9 Type-B full-link IL

Rate	Full-Link Insertion Loss	Description
6 Gbps, 8 Gbps	–28 dB	For HS2 cables, the device IL should be less than –4.25 dB and that of cables should not exceed –19.5 dB.
10 Gbps, 12 Gbps, 16 Gbps	–28 dB	For HS3 cables, the device IL should be less than –7.5 dB and that of cables should not exceed –13 dB.
20 Gbps, 24 Gbps	–26 dB	For HS4 cables, the device IL should be less than –6.5 dB and that of cables should not exceed –13 dB.

Table 10 Type-C full-link IL

Rate	Full-Link Insertion Loss	Device IL
4 Gbps	–17 dB	Less than –5.7 dB (recommended)
6 Gbps	–22 dB	Less than –7 dB (recommended)
8 Gbps	–20.5 dB	Less than –7.65 dB (recommended)
10 Gbps	–25 dB	Less than –9.5 dB (recommended)
12 Gbps	–26 dB	Less than –9.5 dB (recommended)
16 Gbps	–26 dB	Less than –9.5 dB (recommended)
20 Gbps	–26 dB	Less than –9.25 dB (recommended)
24 Gbps	–28 dB	Less than –9.5 dB (recommended)

5.3.3 RX

5.3.3.1 Voltage Stress Test Environment

The RX performance is tested by the voltage stress test method. There are two types of devices to test the RX: the minimum attenuation device, as shown in Figure 10, and the maximum attenuation device, as shown in Figure 11.

Figure 10 Minimum attenuation device

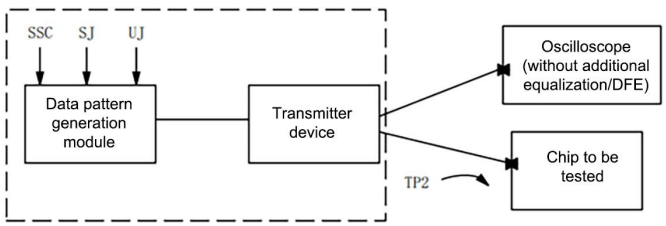
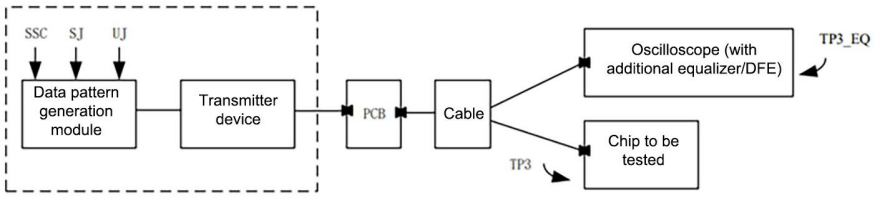


Figure 11 Maximum attenuation device



In the minimum attenuation device, the transmitting device is directly connected to the chip under test, eliminating path attenuation. In this scenario, the oscilloscope does not have any additional equilibrium capabilities and can only display eye diagram indicators.

In the maximum attenuation device, the signal transmission path includes a printed circuit board (PCB), connectors, and cables. Under this device setting, the oscilloscope sets a reference equilibrium that matches the transmission rate and displays the equalized signal eye diagram indicators.

5.3.3.2 General Electrical Parameters

The general electrical parameters in Table 11 shall be specified when the RX is tested.

Table 11 General electrical parameters of the RX

Parameter Name	Symbol	Minimum Value	Maximum Value	Unit	Remarks
Sweep sinusoidal jitter tolerance[Note 1]	S_{Jtol}	0.12		UI	Voltage stress test
Terminating resistor impedance[Note 2]	RRX-DIFF-DC	70	110	Ω	
Differential inter-pair skew tolerance	$RX_{INTER-PAIR-SKEW1}$		450	UI	@TP3
Note 1: Sweep sinusoidal jitter tolerance (SJ_{tol}): During the RX performance test, periodic sinusoidal jitter of different frequencies shall be superimposed on the signal at the transmitting end; Note 2: Terminating resistor: To reduce signal reflection and return loss during transmission, the RX shall connect to a terminating resistor at the end to match the impedance.					

5.3.3.3 Rate-related Parameters

At 2 Gbps and 4 Gbps rates, the electrical parameters of the input signal during RX testing shall comply with the provisions of Table 12.

Table 12 TP2 index parameters in the RX voltage stress test at 2 Gbps and 4 Gbps

Parameter Name	Symbol	Minimum Value	Maximum Value	Unit	Remarks

Parameter Name	Symbol	Minimum Value	Maximum Value	Unit	Remarks
UI	UI	499.85	500.15	ps	300 ppm, SSC effect not considered
		249.925	250.075		
TJ	TJ		0.5	UI	
DDJ	DDJ		0.2	UI	
UJ	UJ		0.3	UI	
Inner eye height	Y1	120	140	mV	
External eye height	Y2		650	mV	
Differential intra-pair skew	$T_{INTRA-PAIR-SKEW}$		75	ps	

At 6 Gbps and 8 Gbps rates, the electrical parameters of the input signal during RX testing shall comply with the provisions of Table 13.

Table 13 TP3_EQ (CTLE) index parameters in the RX voltage stress test at 6 Gbps and 8 Gbps

Parameter Name	Symbol	Minimum Value	Maximum Value	Unit	Remarks
UI	UI	166.6167	166.7167	ps	300 ppm, SSC effect not considered
		124.9625	125.0375		
TJ	TJ		0.6	UI	
DDJ	DDJ		0.3	UI	
UJ	UJ		0.3	UI	
Inner eye height	Y1	65	75	mV	
External eye height	Y2		650	mV	
Differential intra-pair skew	$T_{INTRA-PAIR-SKEW}$		0.42	UI	

At 10 Gbps and 12 Gbps rates, the electrical parameters of the input signal during RX testing shall comply with the provisions of Table 14.

Table 14 TP3_EQ [CTLE + DFE (optional)] index parameters in the RX voltage stress test at 10 Gbps and 12 Gbps

Parameter Name	Symbol	Minimum Value	Maximum Value	Unit	Remarks
UI	UI	99.97	100.03	ps	300 ppm, SSC

Parameter Name	Symbol	Minimum Value	Maximum Value	Unit	Remarks
		83.308	83.358		effect not considered
TJ	TJ		0.6	UI	
DDJ	DDJ		0.3	UI	
UJ	UJ		0.3	UI	
Inner eye height	Y1	50	60	mV	
External eye height	Y2		650	mV	
Differential intra-pair skew	$T_{INTRA-PAIR-SKEW}$		0.42	UI	

At a 16 Gbps rate, the electrical parameters of the input signal during RX testing shall comply with the provisions of Table 15.

Table 15 TP3_EQ [CTLE + DFE (optional)] index parameters in the RX voltage stress test at a 16 Gbps

Parameter Name	Symbol	Minimum Value	Maximum Value	Unit	Remarks
UI	UI	62.48125	62.51875	ps	300 ppm, SSC effect not considered
TJ	TJ		0.6	UI	
DDJ	DDJ		0.3	UI	
UJ	UJ		0.3	UI	
Inner eye height	Y1	45	55	mV	
External eye height	Y2		650	mV	
Differential intra-pair skew	$T_{INTRA-PAIR-SKEW}$		0.42	UI	

At 20 Gbps and 24 Gbps rates, the electrical parameters of the input signal during RX testing shall comply with the provisions of Table 16.

Table 16 TP3_EQ [CTLE + DFE (optional)] index parameters in the RX voltage stress test at 20 Gbps and 24 Gbps

Parameter Name	Symbol	Minimum Value	Maximum Value	Unit	Remarks
UI	UI	49.985	50.015	ps	300 ppm, SSC

Parameter Name	Symbol	Minimum Value	Maximum Value	Unit	Remarks
		41.6542	41.6792		effect not considered
TJ	TJ		0.62	UI	
DDJ	DDJ		0.32	UI	
UJ	UJ		0.3	UI	
Inner eye height	Y1	40	50	mV	
External eye height	Y2		650	mV	
Differential intra-pair skew	$T_{INTRA-PAIR-SKEW}$		0.42	UI	

5.3.3.4 Jitter Tolerance

This section describes the jitter tolerance (JTOL) index of the RX. The JTOL reflects the RX's response to input jitter and is related to the RJ, UDJ, and DDJ. The calculation formula of the JTOL is as follows:

$$\mathrm{JTOL(s)} = DDJ + UJ \quad \text{.....(10)}$$

where,

$$\mathrm{UJ} = \mathrm{RJ} + \mathrm{UDJ} \quad \text{.....(11)}$$

When the JTOL of the RX is tested, the UDJ is divided into two parts: response to sweep sinusoidal jitter ($\mathrm{SJ_{SWEEP}(s)}$) and correction value of the UDJ ($\mathrm{UDJ_{ADJ}}$):

$$\begin{cases} \mathrm{SJ_{SWEEP}(s)} = \frac{\mathrm{SJ_{tol}}}{|\mathrm{JTF(s)}|} \\ \mathrm{UDJ_{ADJ}(s)} = \mathrm{UDJ} - \mathrm{SJ_{SWEEP}(s)} \end{cases} \quad (12)$$

The function of $\mathrm{UDJ_{ADJ}(s)}$ is to add a certain amount of SJ or RJ at different test frequencies to correct the total UDJ value to meet the test requirements.

As shown in Table 17, the RX shall specify the technical requirements for JTOL at different rates, including DDJ, RJ, and UDJ.

Table 17 jitter tolerance-related parameters

Test Environment	Rate	TJ	DDJ	UJ	
				RJ	UDJ
Minimum attenuation device	2 Gbps, 4 Gbps	< 0.35	-	0.15	0.15
	6 Gbps, 8 Gbps	< 0.35	-	0.14	0.16
	10 Gbps, 12 Gbps	< 0.35	-	0.14	0.16
	16 Gbps	< 0.35	-	0.14	0.16

Test Environment	Rate	TJ	DDJ	UJ	
				RJ	UDJ
	20 Gbps, 24 Gbps	< 0.35	-	0.14	0.16
Maximum attenuation device	2 Gbps, 4 Gbps	0.5	0.2	0.15	0.15
	6 Gbps, 8 Gbps	0.6	0.3	0.14	0.16
	10 Gbps, 12 Gbps	0.6	0.3	0.14	0.16
	16 Gbps	0.6	0.3	0.14	0.16
	20 Gbps, 24 Gbps	0.62	0.32	0.14	0.16
Note: When a minimum attenuation device is used for testing, the transmitting device performs FFE adjustment and adjusts the DDJ value to the minimum to meet the requirement that TJ is less than the specified value.					

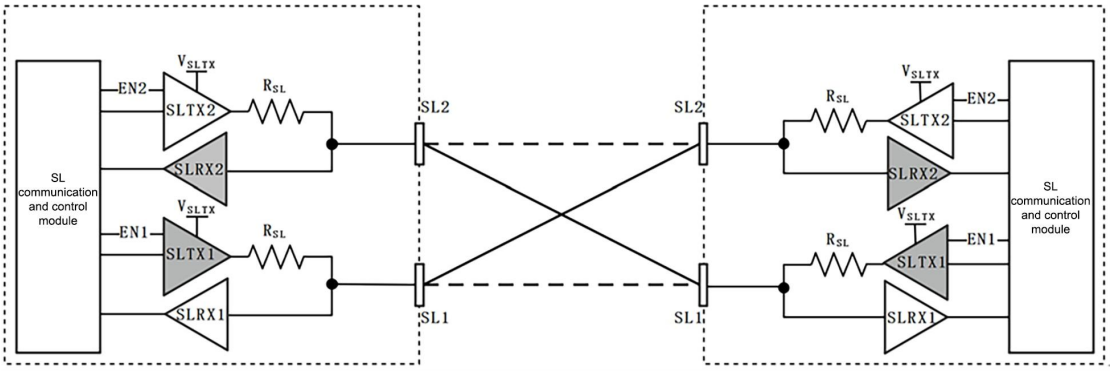
5.4 Sideband Link

The sideband link supports a rate of 12.5 Mbps per single-ended lane, and the signal is DC coupled.

When the low-rate single-ended TX is working (enabled state), the single-ended output impedance complies with the provisions of Table 18; when the TX is not enabled, the single-ended output impedance is in high-resistance state (greater than 1 MΩ). The input internal resistance of the single-ended RX is greater than 1 MΩ and correctly identifies the single-ended level signal sent by the peer TX.

See Figure 12 for the lane circuit in sideband link.

Figure 12 Sideband link circuit



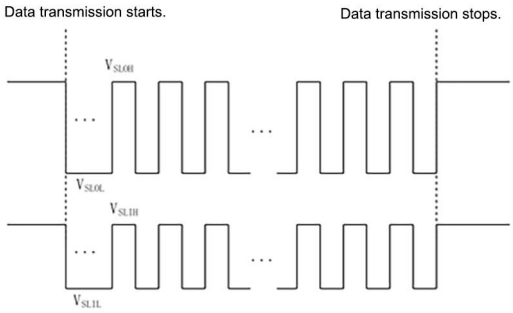
The solid line in the above figure indicates the forward insertion orientation, and the dotted line is the connection line indicates the reverse insertion orientation. The SL communication and control module enables EN1 or EN2 according to the insertion orientation. In forward insertion, SLTX1 is enabled through EN1 (the SLTX1 and SLRX2 in gray in the figure being working). In reverse

insertion, SLTX2 is enabled through EN2 (the SLTX2 and SLRX1 in white in the figure being working).

When the connection remains stable, the TX can send single-ended signals, and the internal resistance meets the requirements. The high and low levels are V_{SLOH} and V_{SLOL}, respectively. After the data transmission is completed, the level returns to V_{SLOH}.

The lane working waveform of the sideband link is shown in Figure 13.

Figure 13 Sideband link lane working waveform



The electrical parameters of the sideband link transmitter (SLTX) and sideband link receiver (SLRX) comply with the technical requirements specified in Table 18.

Table 18 Electrical parameters of the sideband link

Parameter Name	Symbol	Minimum Value	Maximum Value	Unit	Remarks
TX output high-level amplitude	V_{SLOH}	2.4	3.47	V	
TX output low-level amplitude	V_{SLOL}	–0.05	0.35	V	
RX input high-level amplitude	V_{SLIH}	1.95	3.77	V	
RX input low-level amplitude	V_{SLIL}	–0.35	0.65	V	
Single-ended output impedance by TX in enabled state	R_{SLTXON}	25	90	Ω	
TX single-ended pull-up resistor	R_{PULL_UP}	7.0	10.5	KΩ	
RX pull-down resistor	R_{PULL_DN}	0.7	1.05	MΩ	
RX input capacitor	C_{RXIN}		8	pF	
Residual voltage when RX is disconnected from TX	V_{DISCRX}		0.4	V	Tested when an external 1 MΩ pull-down resistor is connected
High current inputting RX	I_{RXIH}		25	μA	
Low current inputting RX	I_{RXIL}		0.4	μA	

Parameter Name	Symbol	Minimum Value	Maximum Value	Unit	Remarks
Rising and falling edge time	T_{RF}	2	15	ns	20%–80%
Single bit width	T_{UI}	78.4	81.6	ns	2%

6 Logical Layer

6.1 Overview

The logical layer is mainly responsible for maintaining the link lane state, encoding the transport layer packets and transmitting the packets to the electrical layer, and decoding the electrical layer data and transmitting the data to the transport layer. The logical layer is divided into the main link logical layer and the sideband link logical layer.

The data structure includes a main link data structure and a sideband link data structure, wherein the main link data structure includes a logic block, a logical layer packet, and a control frame; the sideband link data structure includes a sideband link logical layer packet.

The logical layer can support four-lane mode or eight-lane mode. See Table 133. In this section, the four-lane mode is the simplified-specification mode, and the eight-lane mode is the full-specification mode.

The upper layer in this section refers to the software that manages adapters or implements related functions.

6.2 Data Structure

6.2.1 Basic Requirements

The main link logical layer uses two code types to implement data processing and link management functions. The first code type is the logic block (LLB), which completes the transmission of transport layer packets and logical layer management packets. The second code type is the logical layer control frame (LLCF), which realizes link management functions such as link training and state update. The LLB completes the encapsulation of logical layer data packet (LLDP) and logical layer main link management packet (LLMMP). LLB and LLCF are transmitted through the main link. LLCF uses independent lanes for transmission to ensure the integrity of the LLCF code pattern on a single lane.

The main link logical layer also includes a logical layer sideband link management packet (LLSMP), which is transmitted through the sideband link.

The sideband link logical layer uses the sideband link logical layer packet (LLSP) to realize the encapsulation of the logical layer sideband link management packet (LLSMP) and the transport layer packet. The packets are transmitted on the sideband link.

For illegal packets and non-state-expected packets, the processing principles are as follows:

- The transmitter side is prohibited from sending illegal packets;
- The transmitter side is prohibited from sending non-state-expected packets in any state;
- The receiver side can ignore or respond to illegal packets and non-state-expected packets.

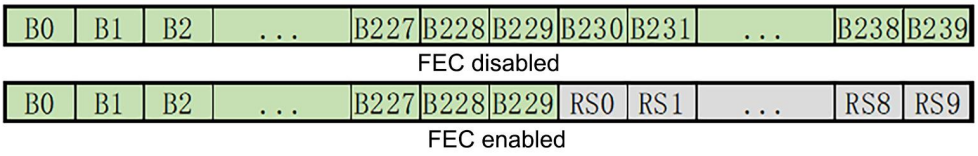
6.2.2 Logic Block

The logical layer uses LLB as the basic unit of data transmission.

(a) Four-lane mode

In four-lane mode, one LLB consists of 240 bytes, as shown in Figure 14. When FEC is disabled, all 240 bytes in the LLB constitute the LLB body; when FEC is enabled, the first 230 bytes in the LLB constitute the LLB body, and the last 10 bytes constitute the LLB tail. The LLB body is filled with LLDP and LLMMP, and the LLB tail corresponds to the FEC-encoded check bytes.

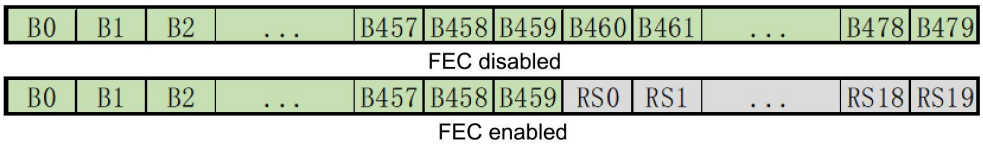
Figure 14 LLB structure in four-lane mode



(b) Eight-lane mode

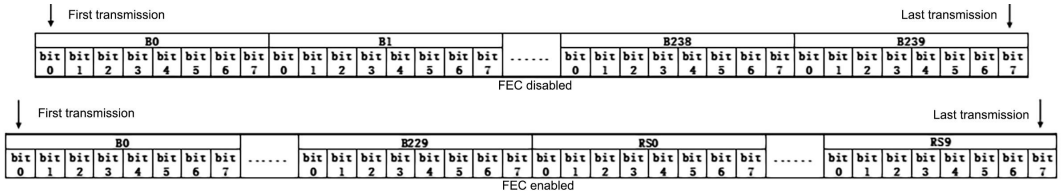
In eight-lane mode, one LLB consists of 480 bytes, as shown in Figure 15. When FEC is disabled, all 480 bytes in the LLB constitute the LLB body; when FEC is enabled, the first 460 bytes in the LLB constitute the LLB body, and the last 20 bytes constitute the LLB tail. The LLB body is filled with LLDP and LLMMP, and the LLB tail corresponds to the FEC-encoded check bytes.

Figure 15 LLB structure in eight-lane mode



At the electrical layer, data transmission is sent and received byte by byte according to the LLB arrangement. Single bytes are sent and received starting from the low bit by default. The order of sending and receiving LLB in four-lane mode is shown in Figure 16.

Figure 16 LLB data transmission sequence



6.2.3 Logical Layer Packet

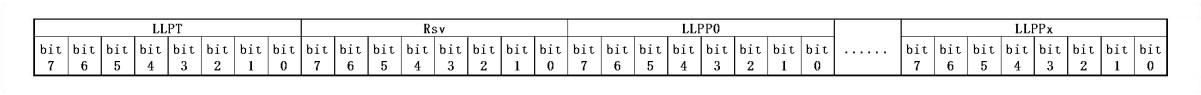
6.2.3.1 Overview

The transport layer packet and the logical layer management packet are encapsulated into the structure of a logical layer packet, which includes a logical layer data packet and a logical layer management packet.

6.2.3.2 Logical Layer Data Packet

Logical layer data packet (LLDP) implements the encapsulation of transport layer packets. LLDP can only be transmitted via ML. As shown in Figure 17, LLDP includes logical layer packet header (LLPH) and logical layer packet payload (LLPP).

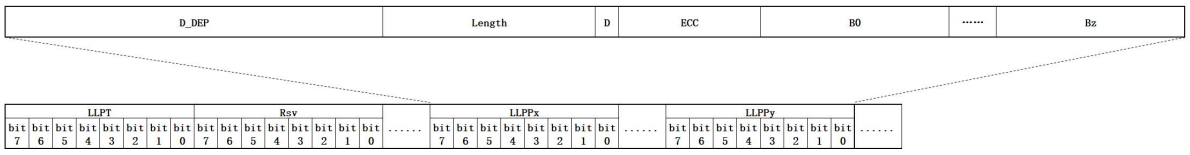
Figure 17 LLDP structure



LLPH is fixed to 2 bytes, of which the LLPT field indicates the current data packet type, and the LLPT of LLDP is fixed to 0x0F; the Rsv field is a reserved field, fixed to 0x0. LLPP fills the transport layer packets. Refer to the encapsulation block for the length.

The mapping relationship between the TLP and LLDP is shown in Figure 18. The byte order during mapping is shown in C.2 (LLPPx = 0).

Figure 18 Mapping relationship between the TLP and LLDP



The mapping scenarios are as follows:

(a) LLPPx is equal to LLPP0, and LLDP fills TLP data starting from the first byte.

(b) LLPPx is not equal to LLPP0, LLDP starts filling TLP data from the middle, and the data filled before LLPPx is the TLP data before the current TLP.

(c) LLPPy is not the last byte of the current LLDP, and Bz is the last byte of TLP. That is, the TLP data is filled in the current LLDP, and the data filled after LLPPy is the TLP data after the current TLP.

(d) LLPPy is the last byte of the current LLDP, and Bz is the last byte of TLP. That is, the TLP data has just been filled in the current LLDP.

(e) LLPPy is the last byte of the current LLDP, and Bz is not the last byte of TLP. That is, the current LLDP has not completed all data filling of the current TLP, and the remaining data of the current TLP starts to be filled from LLPP0 of the next LLDP.

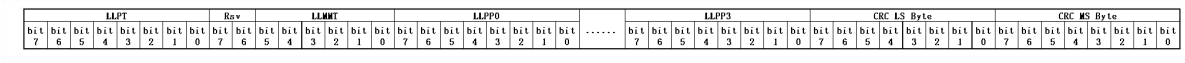
6.2.3.3 Logical Layer Management Packet

6.2.3.3.1 Overview

The logical layer management packet implements the packet processing of the logical layer management messages, including the logical layer main link management packet (LLMMP) and the logical layer sideband link management packet (LLSMP). The LLMMP is transmitted through ML, and the LLSMP is transmitted through SL. When ML is in HS state, it is advisable to use ML to transmit logical layer management packets.

As shown in Figure 19, the LLMMP is fixed to 8 bytes and consists of LLPH, LLPP, and CRC check information.

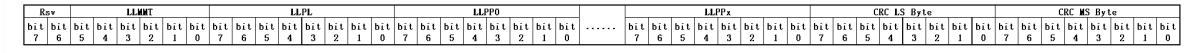
Figure 19 LLMMP structure



The length of LLPH is fixed to 2 bytes, where the LLPT field indicates the current data packet type, and the LLPT of the LLMMP is fixed to 0xF0; the Rsv field is reserved and filled with 0x0; the Logical Layer Management Message Type (LLMMT) field indicates the message type carried by the current LLMMP. The LLPP length is fixed to 4 bytes, and the content is determined by the current logical layer management message to be transmitted. The message length is less than 4 bytes, it is fixedly filled with 0x0. Fixed to 2 bytes, the CRC uses CRC16 and fills in the check information of all parameters in the Rsv field, LLMMT field, and LLPP field of the current data packet.

As shown in Figure 20, a complete LLSMP structure is described. The entire data message length is not fixed and consists of LLPH, LLPP, and CRC.

Figure 20 LLSMP structure



The LLPH length is fixed to 2 bytes, of which the Rsv field and LLMMT field are consistent with the LLMMP; the logical layer packet length (LLPL) field indicates the current LLSMP message payload length. The length and content of the LLPP are determined by the logical layer management message currently to be transmitted. Fixed to 2 bytes, the CRC uses CRC16 and fills in the check information of all parameters in other fields of the current data message except the CRC field.

Logical layer management messages are shown in Table 19, which is a summary of logical layer management messages supported by the logical layer. In order to reduce link anomalies caused by message loss and improve the robustness of the system, when TSM, LLFM, CLFM_Ack, EQFM_Ack, Ack/Nack, LPRM, LWAM, and LDAM messages in the table are sent through the main link, they need to be sent directly three times and placed in three consecutive LLBs. The receiver side only needs to receive and parse one of the messages, and ignore the rest. If an abnormality occurs in the main link during the three transmissions through the main link, the transmitter side stops transmitting and does not need to switch to the sideband link for transmission. If sent via a sideband link, it only needs to be sent once. For messages that require a response (such as LPRM/LWAM/LDAM), the receiver side only needs to respond to one of the repeated messages, and ignore the rest. If two logical layer management messages with exactly the same parameters are sent through the main link, the interval between the two messages must be more than tLLMMP_same_interval.

Table 19 Summary of logical layer management messages

Message Type	Abbreviation	Description	Transmission Mode	Replied or Not
0x02	CLFM_Ack	Clock lock feedback response message, responding to the clock lock feedback message.	ML/SL	No
0x03	EQFM_Ack	Equilibrium feedback response message, responding to the equilibrium feedback message.	ML/SL	No
0x11	TSM	Link training start message, starting a link training.	ML/SL	No
0x12	CLFM	Clock lock feedback message, feeding back the clock lock result & Swing update request.	ML/SL	Yes
0x13	EQFM	Equilibrium feedback message, feeding back the balance result & FFE update request.	ML/SL	Yes
0x14	LLFM	Lane lock feedback message, feeding back the lane lock result.	ML/SL	No
0x21	LPRM	Low power consumption request message.	ML/SL	Yes
0x22	LWAM	Link width adjustment message.	ML/SL	Yes
0x23	LDAM	Lane direction adjustment message.	ML/SL	Yes
0x24	CFSLM	CF position sending message. The transmitter side informs the receiver side of the subsequent CF sending position.	ML	No
0x2A	Ack/Nack	Response message, responding to the received management information.	ML/SL	No
0x2B	ERR_RM	Error report message, reporting the abnormal issues detected.	ML/SL	Yes

6.2.3.3.2 Training Start Message

The TSM is a collection of link training start flags, including the training start flags of all transmission lanes of the current port.

The TSM is used when a link training is started. Application scenarios include the following: In the initial link establishment scenario, after the port initialization is completed, a TSM is sent to start link training; in the abnormal retraining scenario, a TSM is sent to start link retraining; when the LP3 low power consumption is exited, a TSM is sent to start link recovery. After receiving the TSM, the receiver side starts the training of the designated lane in the TSM.

The message parameters of each lane are independent. If the lane is disabled or does not exist, the corresponding message parameter is fixed to 0.

The TSM does not require a response. However, the peer needs to send a clock lock feedback message within the tTSMResponse period.

The TSM supports transmission through ML and SL. The LLMMP arrangement corresponding to ML transmission is shown in Table 20, and the LLSMP arrangement corresponding to SL transmission is shown in Table 21.

Table 20 Arrangement format of the TSM in LLMMP

Byte	Bit	Field Name	Description
0	7:0	LLPT	Message type, configured as 0xF0.
1	5:0	LLMMT	Logical layer management message type, configured as 0x11.
	7:6	Reserved	Reserved.
2	7:0	TS0–TS7	Training start. Bit 0–7 represent the training state in lane 0–7: 1b: training initiated; 0b: training not initiated
3–5	7:0	Reserved	Reserved.
6–7	7:0	CRC16	Check code, checking bytes 1–5.

Table 21 Arrangement format of the TSM in LLSMP

Byte	Bit	Field Name	Description
0	5:0	LLMMT	Logical layer management message type, configured as 0x11.
	7:6	Reserved	Reserved.
1	7:0	LLPL	Message payload length parameter, equal to 1.
2	7:0	TS0–TS7	Training start. Bit 0–7 represent the training state in lane 0–7: 1b: training initiated; 0b: training not initiated
3–4	7:0	CRC16	Check code, checking bytes 0–2.

6.2.3.3.3 Clock Lock Feedback Message

The clock lock feedback message (CLFM) is a collection of clock lock result flags and Swing update request messages, including the clock lock results and Swing update requests of all receiving lanes of the current port. CLFM_Ack is a response message of CLFM.

CLFM is used in the clock lock phase to feed back on the current clock lock result at the receiving end and initiate a request to update Swing parameters for lanes that have failed clock lock.

Different CLFM feedback clock lock messages can be used independently for each lane, or the same CLFM feedback clock lock message can be combined.

CLFM requires a response. The transmitter side performs pattern switching for the lane locked by the clock, completes Swing update for the lane whose Swing parameters need to be adjusted, and responds with Ack for the lane that completes the above operations. Otherwise, it responds with Nack. The usage rules of CLFM_Ack are similar to those of CLFM. Each lane responds independently. Different CLFM_Ack or a combined CLFM_Ack can be used for response. Examples of usage rules are as follows:

(a) CLFM0 feeds back multiple lane clock lock information, and can use CLFM_Ack0 and CLFM_Ack1 to respond to part or all of the lane clock lock information fed back by CLFM0, respectively.

(b) CLFM0 feeds back partial lane clock lock information, and CLFM1 feeds back partial lane clock lock information. CLFM_Ack0 can be used to respond to part or all of the lane clock lock information fed back by CLFM0 and CLFM1 at the same time.

The message parameters of each lane are independent. If the lane is disabled or does not exist, the corresponding message parameter is fixed to 0.

CLFM supports transmission via ML and SL, and the clock lock feedback message of each lane is represented by two bytes.

The LLMMP arrangement format corresponding to CLFM in ML transmission is shown in Table 22. Through ML transmission, a single CLFM supports up to two lanes of clock lock feedback messages.

The LLSMP arrangement format corresponding to CLFM in SL transmission is shown in Table 23. Through SL transmission, a single CLFM supports up to eight lanes of clock lock feedback messages.

Table 22 Arrangement format of CLFM in LLMMP

Byte	Bit	Field Name	Description
0	7:0	LLPT	Message type, configured as 0xF0.
1	5:0	LLMMT	Logical layer management message type, configured as 0x12.
	7:6	Reserved	Reserved.
2	2:0	Lane_ID	Lane number.
	7:3	Reserved	Reserved.
3	3:0	Swing_Req	Lane request Swing gear indicated by Lane_ID of byte 2: 0h: gear 0; 1h: gear 1; 2h: gear 2; 3h: gear 3; 4h: gear 4; 5h: gear 5; 6h: gear 6;

Byte	Bit	Field Name	Description
			7h: gear 7; 8h to Eh: reserved; Fh: Swing gear request invalid.
	4	CL	Lane clock lock result indicated by Lane_ID of byte 2 (effective only when the request for Swing gear is invalid): 1b: Clock lock succeeded; 0b: Clock lock failed.
	7:5	Reserved	Reserved.
4	2:0	Lane_ID	Lane number.
	7:3	Reserved	Reserved.
5	3:0	Swing_Req	Lane request Swing gear indicated by Lane_ID of byte 4: 0h: gear 0; 1h: gear 1; 2h: gear 2; 3h: gear 3; 4h: gear 4; 5h: gear 5; 6h: gear 6; 7h: gear 7; 8h to Eh: reserved; Fh: Swing gear request invalid.
	4	CL	Lane clock lock result indicated by Lane_ID of byte 4 (valid only when the request for Swing gear is invalid): 1b: Clock lock succeeded; 0b: Clock lock failed.
	7:5	Reserved	Reserved.
6–7	7:0	CRC16	Check code, checking bytes 1–5.

Table 23 Arrangement format of CLFM in LLSMP

Byte	Bit	Field Name	Description
0	5:0	LLMMT	Logical layer management message type, configured as 0x12.
	7:6	Reserved	Reserved.
1	7:0	LLPL	Message payload length parameter, equal to

Byte	Bit	Field Name	Description
			N-1.
2	2:0	Lane_ID	Lane number.
	7:3	Reserved	Reserved.
3	3:0	Swing_Req	Lane request Swing gear indicated by Lane_ID of byte 2: 0h: gear 0; 1h: gear 1; 2h: gear 2; 3h: gear 3; 4h: gear 4; 5h: gear 5; 6h: gear 6; 7h: gear 7; 8h to Eh: reserved; Fh: Swing gear request invalid.
	4	CL	Lane clock lock result indicated by Lane_ID of byte 2 (effective only when the request for Swing gear is invalid): 1b: Clock lock succeeded; 0b: Clock lock failed.
	7:5	Reserved	Reserved.
4–N	7:0	Clock lock feedback message of other lanes	The clock lock feedback information of each lane is represented by two bytes. When CLFM feeds back a single lane message, N is equal to 3, and byte 3 is the last 1 payload of LLSMP; When CLFM feeds back 2 lane messages, N is equal to 5. Refer to bytes 2–3 for bytes 4–5 parameters; When CLFM feeds back 3 lane messages, N is equal to 7. Refer to bytes 2–3 for bytes 4–7 parameters; When CLFM feeds back 4 lane messages, N is equal to 9. Refer to bytes 2–3 for bytes 4–9 parameters; When CLFM feeds back 5 lane messages, N is equal to 11. Refer to bytes 2–3 for bytes 4–11 parameters; When CLFM feeds back 6 lane messages, N is equal to 13. Refer to bytes 2–3 for bytes 4–13 parameters; When CLFM feeds back 7 lane messages, N is equal to 15. Refer to bytes 2–3 for bytes

Byte	Bit	Field Name	Description
			4–15 parameters; When CLFM feeds back 8 lane messages, N is equal to 17. Refer to bytes 2–3 for bytes 4–17 parameters.
N + 1–N + 2	7:0	CRC16	Check code, checking bytes 0–N.

CLFM_Ack supports transmission through ML and SL. The LLMMP arrangement format corresponding to ML transmission is shown in Table 24, and the LLSMP arrangement format corresponding to SL transmission is shown in Table 25. Through ML transmission and SL transmission, a single CLFM_Ack supports the feedback of clock lock message responses for up to four lanes.

Table 24 Arrangement format of CLFM_Ack in LLMMP.

Byte	Bit	Field Name	Description
0	7:0	LLPT	Message type, configured as 0xF0.
1	5:0	LLMMT	Logical layer management message type, configured as 0x02.
	7:6	Reserved	Reserved.
2	7:0	CLFM0_Ack–CLFM7_Ack	Bit 0–7 represents the clock lock feedback response of lane 0–7: 0b: Nack; 1b: Ack.
3–5	7:0	Reserved	Reserved.
6–7	7:0	CRC16	Check code, checking bytes 1–5.

Table 25 Arrangement format of CLFM_Ack in LLSMP

Byte	Bit	Field Name	Description
0	5:0	LLMMT	Logical layer management message type, configured as 0x02.
	7:6	Reserved	Reserved.
1	7:0	LLPL	Message payload length parameter, equal to 1.
2	7:0	CLFM0_Ack–CLFM7_Ack	Bit 0–7 represents the clock lock feedback response of lane 0–7: 0b: Nack; 1b: Ack.
3–4	7:0	CRC16	Check code, checking bytes 0–2.

6.2.3.3.4 Equilibrium Feedback Message

The equilibrium feedback message (EQFM) is a collection of equilibrium result flags and FFE parameter update request messages, including the equilibrium results and FFE parameter update requests of all receiving lanes of the current port. EQFM_Ack is the response message of EQFM.

EQFM is used in the equilibrium phase. It is used by the receiving end to feedback the current equilibrium result and initiate a request to update FFE parameters for the lane that failed the equilibrium. Different EQFM feedback equilibrium messages can be used independently for each lane, or the same EQFM feedback equilibrium message can be combined.

EQFM requires a response. The transmitter side performs pattern switching for the lane that has been successfully equalized, completes FFE update for the lane whose FFE parameters need to be adjusted, and responds with Ack for the lane that completes the above operations. Otherwise, it responds with Nack. For the usage rules of EQFM_Ack, please refer to CLFM_Ack. Each lane can respond independently or in combination.

The message parameters of each lane are independent. If the lane is disabled or does not exist, the corresponding message parameter is fixed to 0.

EQFM supports transmission via ML and SL, and the equilibrium feedback message of each lane is represented by two bytes.

The LLMMP arrangement format corresponding to EQFM in ML transmission is shown in Table 26. Through ML transmission, a single EQFM supports up to two lanes of equilibrium messages.

The LLSMP arrangement format corresponding to EQFM in SL transmission is shown in Table 27. Through SL transmission, a single EQFM supports up to eight lanes of equilibrium messages.

The actual gear of FFE consists of three parameters, namely PRE1, POST1 and POST2. The original values of the three parameters range from 0–15, as shown in Table 133.

Example 1: Define the parameter Supported_FFE_Level[11:0] in port capability negotiation, which will constrain the value range of the three parameters of FFE's actual gear.

PRE1_mask = Supported_FFE_Level[3:0];

POST1_mask = Supported_FFE_Level[7:4];

POST2_mask = Supported_FFE_Level[11:8].

Example 2: Record the actual gear position of each FFE parameter as G, which is recorded as PRE1_G, POST1_G, and POST2_G, respectively. Take PRE1_G as an example to illustrate the corresponding relationship between the value range of PRE1_G and part of PRE1_mask:

PRE1_mask = 0000b: The actual gear of PRE1_G can be 0;

PRE1_mask = 0010b: The actual gear of PRE1_G can be 0 or 2;

PRE1_mask = 1100b: The actual gear of PRE1_G can be 0, 4, 8, or 12;

PRE1_mask = 1110b: The actual gear of PRE1_G can be 0, 2, 4, 6, 8, 10, 12, or 14;

PRE1_mask = 1111b: The actual gear of PRE1_G can be 0–15.

The values of PRE1_mask are not all listed above, but as long as all bits that are 1 are continuous, they are considered to be compliant values. The corresponding relationship is the same as above with PRE1_G.

Example 3: Record the bit corresponding to the lowest bit 1 in each FFE parameter mask as M, which are recorded as PRE1_M, POST1_M, and POST2_M, respectively. The PRE1_mask is taken as an example to illustrate its relationship with PRE1_M:

PRE1_mask=0000b: PRE1_M=0; PRE1_mask=1000b: PRE1_M=3;

PRE1_mask=1100b: PRE1_M=2; PRE1_mask=1110b: PRE1_M=1;

The number of bits set to 1 in each FFE parameter mask is N, denoted as PRE1_N, POST1_N, and POST2_N, respectively. The calculation formula is as follows:

PRE1_N = PRE1_mask[3] + PRE1_mask[2] + PRE1_mask[1] + PRE1_mask[0];

POST1_N = POST1_mask[3] + POST1_mask[2] + POST1_mask[1] + POST1_mask[0];

POST2_N = POST2_mask[3] + POST2_mask[2] + POST2_mask[1] + POST2_mask[0];

The final definition of the mapping relationship is:

FFE_Req = (PRE1_G >> PRE1_M) << (POST1_N + POST2_N)

+ (POST1_G >> POST1_M) << POST2_N

+ (POST2_G >> POST2_M)

Table 26 Arrangement format of EQFM in LLMMP

Byte	Bit	Field Name	Description
0	7:0	LLPT	Message type, configured as 0xF0.
1	5:0	LLMMT	Logical layer management message type, configured as 0x13.
	7:6	Reserved	Reserved.
2	2:0	Lane_ID	Lane number.
	3	Reserved	Reserved.
	6:4	Swing_Req	Lane request Swing gear indicated by Lane_ID of byte 2: 0h: gear 0; 1h: gear 1; 2h: gear 2; 3h: gear 3; 4h: gear 4; 5h: gear 5; 6h: gear 6; 7h: gear 7; (Effective only when Swing_Req_Enable is valid)
	7	Swing_Req_Enable	Lane Swing gear adjustment enablement indicated by Lane_ID of byte 2: 1b: Swing gear adjustment is valid; 0b: Swing gear adjustment is invalid.
3	6:0	FFE_Req	The Lane_ID of byte 2 indicates the lane request FFE gear, which supports up to 64 gears. The number of gears corresponds to

Byte	Bit	Field Name	Description
			the FFE_Req parameter one by one (mapping described in this section); 7Fh: The requested FFE gear is invalid.
	7	EQD	Lane equilibrium result indicated by Lane_ID of byte 2 (effective only when the requested FFE gear and requested Swing gear are invalid): 1b: Equilibrium succeeded; 0b: Equilibrium failed.
4	2:0	Lane_ID	Lane number.
	3	Reserved	Reserved.
	6:4	Swing_Req	Lane request Swing gear indicated by Lane_ID of byte 4: 0h: gear 0; 1h: gear 1; 2h: gear 2; 3h: gear 3; 4h: gear 4; 5h: gear 5; 6h: gear 6; 7h: gear 7; (Effective only when Swing_Req_Enable is valid)
4	7	Swing_Req_Enable	Lane Swing gear adjustment enablement indicated by Lane_ID of byte 4: 1b: Swing gear adjustment is valid; 0b: Swing gear adjustment is invalid.
5	6:0	FFE_Req	The lane indicated by Lane_ID of byte 4 requests FFE gear, which supports up to 64 gears. The number of gears corresponds to the FFE_Req parameter one by one; 7Fh: The requested FFE gear is invalid.
	7	EQD	Lane equilibrium result indicated by Lane_ID of byte 4 (effective only when the requested FFE gear and requested Swing gear are invalid): 1b: Equilibrium succeeded; 0b: Equilibrium failed.
6–7	7:0	CRC16	Check code, checking bytes 1–5.

Table 27 Arrangement format of EQFM in LLSMP

Byte	Bit	Field Name	Description
0	5:0	LLMMT	Logical layer management message type, configured as 0x13.
	7:6	Reserved	Reserved.
1	7:0	LLPL	Message payload length parameter, equal to N-1.
2	2:0	Lane_ID	Lane number.
	3	Reserved	Reserved.
	6:4	Swing_Req	Lane request Swing gear indicated by Lane_ID of byte 2: 0h: gear 0; 1h: gear 1; 2h: gear 2; 3h: gear 3; 4h: gear 4; 5h: gear 5; 6h: gear 6; 7h: gear 7; (Effective only when Swing_Req_Enable is valid)
	7	Swing_Req_Enable	Lane Swing gear adjustment enablement indicated by Lane_ID of byte 2: 1b: Swing gear adjustment is valid, 0b: Swing gear adjustment is invalid.
3	6:0	FFE_Req	The lane indicated by Lane_ID of byte 2 requests FFE gear, which supports up to 64 gears. The number of gears corresponds to the FFE_Req parameter one by one; 7Fh: The requested FFE gear is invalid.
3	7	EQD	Lane equilibrium result indicated by byte 2 Lane_ID (valid only when the requested FFE gear and requested Swing gear are invalid): 1b: Equilibrium succeeded; 0b: Equilibrium failed.
4–N	7:0	Equilibrium feedback message of other lanes. For details, see the description.	The equilibrium feedback information of each lane is represented by two bytes: When EQFM feeds back a single lane message, N is equal to 3, and byte 3 is the last 1 payload of LLSMP; When EQFM feeds back 2 lane messages,

Byte	Bit	Field Name	Description
			N is equal to 5. Refer to bytes 2–3 for bytes 4–5 parameters; When EQFM feeds back 3 lane messages, N is equal to 7. Refer to bytes 2–3 for bytes 4–7 parameters; When EQFM feeds back 4 lane messages, N is equal to 9. Refer to bytes 2–3 for bytes 4–9 parameters; When EQFM feeds back 5 lane messages, N is equal to 11. Refer to bytes 2–3 for bytes 4–11 parameters; When EQFM feeds back 6 lane messages, N is equal to 13. Refer to bytes 2–3 for bytes 4–13 parameters; When EQFM feeds back 7 lane messages, N is equal to 15. Refer to bytes 2–3 for bytes 4–15 parameters; When EQFM feeds back 8 lane messages, N is equal to 17. Refer to bytes 2–3 for bytes 4–17 parameters.
N + 1–N + 2	7:0	CRC16	Check code, checking bytes 0–N.

EQFM_Ack supports transmission via ML and SL. The LLMMP arrangement format corresponding to EQFM_Ack in ML transmission is shown in Table 28. The LLSMP arrangement format corresponding to EQFM_Ack in SL transmission is shown in Table 29. Through ML transmission and SL transmission, a single EQFM_Ack supports the feedback of equilibrium feedback lock message responses for up to eight lanes.

Table 28 Arrangement format of EQFM_Ack in LLMMP

Byte	Bit	Field Name	Description
0	7:0	LLPT	Message type, configured as 0xF0.
1	5:0	LLMMT	Logical layer management message type, configured as 0x03.
	7:6	Reserved	Reserved.
2	7:0	EQFM0_Ack–EQFM7_Ack	Bit 0–7 represent the equilibrium feedback response of lane 0–7: 0b: Nack; 1b: Ack.
3–5	7:0	Reserved	Reserved.
6–7	7:0	CRC16	Check code, checking bytes 1–5.

Table 29 Arrangement format of EQFM_Ack in LLSMP

Byte	Bit	Field Name	Description
0	5:0	LLMMT	Logical layer management message type, configured as 0x03.
	7:6	Reserved	Reserved.
1	7:0	LLPL	Message payload length parameter, equal to 1.
2	7:0	EQFM0_Ack–EQFM7_Ack	Bit 0–7 represent the equilibrium feedback response of lane 0–7: 0b: Nack; 1b: Ack.
3–4	7:0	CRC16	Check code, checking bytes 0–2.

6.2.3.3.5 Lane Lock Feedback Message

The lane lock feedback message (LLFM) is a collection of lane lock result flags, including the lane lock results of all receiving lanes of the current port.

The LLFM is used to feed back the lane lock results of all receiving lanes at the RX end during the lane lock phase. All lanes should feed back lane lock results simultaneously and use the same LLFM. The lane that knows its lock result first should wait for the other lanes to know their lock result (lane lock success or failure) before feeding back the LLFM.

The LLFM does not require a response, and the TX switches the pattern for the lanes that have been successfully locked.

The message parameters of each lane are independent. If the lane is disabled or does not exist, the corresponding message parameter is fixed to 0.

The LLFM supports transmission via ML and SL. The LLMMP arrangement format corresponding to LLFM in ML transmission is shown in Table 30. The LLSMP arrangement format corresponding to LLFM in SL transmission is shown in Table 31.

Table 30 Arrangement format of LLFM in LLMMP

Byte	Bit	Field Name	Description
0	7:0	LLPT	Message type, configured as 0xF0.
1	5:0	LLMMT	Logical layer management message type, configured as 0x14.
	7:6	Reserved	Reserved.
2	7:0	LL0–LL7	Lane lock. Bit 0–7 represent the locking state of lane 0–7: 1b: locked; 0b: not locked.
3–5	7:0	Reserved	Reserved.

Byte	Bit	Field Name	Description
6–7	7:0	CRC16	Check code, checking bytes 1–5.

Table 31 Arrangement format of LLFM in LLSMP

Byte	Bit	Field Name	Description
0	5:0	LLMMT	Logical layer management message type, configured as 0x14.
	7:6	Reserved	Reserved.
1	7:0	LLPL	Message payload length parameter, equal to 1.
2	7:0	LL0–LL7	Lane lock. Bit 0–7 represent the locking state of lane 0–7: 1b: locked; 0b: not locked.
3–4	7:0	CRC16	Check code, checking bytes 0–2.

6.2.3.3.6 Lower Power Consumption Request Message

The lower power request message (LPRM) is a set of low power consumption request parameters, including entering/exiting the low power consumption request and the low power consumption request gear.

The LPRM is used when the link is in a service transmission state or an LP3 low power consumption state. In the service transmission state, when there is no service data or management message to be transmitted, a low power consumption request can be initiated. The low power consumption request must include a low power consumption gear; in the LP3 low power consumption state, when service data transmission needs to be restarted, an exit low power consumption request can be initiated. The low power consumption gear corresponding to the exit low power consumption request must be LP3. The current port negotiates with the other end through LPRM to enter or exit low power consumption control of lane links.

The LPRM needs to respond except for LP0f. The RX end agrees to the low power consumption request and needs to respond with Ack to LPRM; otherwise, it needs to respond with Nack. LP0f does not require a response, forcing the peer end to accept the LP0f low power consumption request.

The LPRM supports transmission via ML and SL. It is recommended to use ML transmission when entering LPRM, and only transmission via SL is supported when exiting LPRM. The LLMMP arrangement format corresponding to ML transmission is shown in Table 32. The LLSMP arrangement format corresponding to SL transmission is shown in Table 33.

Table 32 Arrangement format of LPRM in LLMMP

Byte	Bit	Field Name	Description
0	7:0	LLPT	Message type, configured as 0xF0.

Byte	Bit	Field Name	Description
1	5:0	LLMMT	Logical layer management message type, configured as 0x21.
	7:6	Reserved	Reserved.
2	7:0	LP_GEAR	Low power consumption request gear: 00h: low power consumption entering LP0; 01h: low power consumption entering LP1; 02h: low power consumption entering LP2; 03h: low power consumption entering LP3; 10h: low power consumption entering LP0f; other values: reserved.
3	0	LP_MODE	Low power consumption request mode: 0b: entering the mode; 1b: exiting the mode;
	1	LP_WAY	Low power consumption request direction: 0b: unidirectional; 1b: bidirectional (used for low power consumption of ports).
	7:2	Reserved	Reserved.
4–5	7:0	Reserved	Reserved.
6–7	7:0	CRC	Check code, checking bytes 1–5.

Table 33 Arrangement format of LPRM in LLSMP

Byte	Bit	Field Name	Description
0	5:0	LLMMT	Logical layer management message type, configured as 0x21.
	7:6	Reserved	Reserved.
1	7:0	LLPL	Message payload length parameter, equal to 2.
2	7:0	LP_GEAR	Low power consumption request gear: 00h: low power consumption entering LP0; 01h: low power consumption entering LP1; 02h: low power consumption entering LP2; 03h: low power consumption entering LP3; 10h: low power consumption entering LP0f; other values: reserved.
3	0	LP_MODE	Low power consumption request mode: 0b: entering the mode; 1b: exiting the mode;

Byte	Bit	Field Name	Description
	1	LP_WAY	Low power consumption request direction: 0b: unidirectional; 1b: bidirectional (used for low power consumption of ports).
	7:2	Reserved	Reserved.
4–5	7:0	CRC	Check code, checking bytes 0–3.

6.2.3.3.7 Link Width Adjust Message

The link width adjust message (LWAM) is a set of link width adjust request parameters, including the adjusted link width parameters and the low power consumption gear corresponding to the lane that needs to be closed after the link width adjustment. The LWAM is used to adjust the state of inter-device lanes.

The LWAM is initiated when the link bandwidth request needs to be adjusted during service transmission. The port transmission link width is adjusted by negotiation with the peer end.

The LWAM requires a response. The RX end agrees to the link width adjustment request and responds with Ack to LWAM; otherwise, it needs to respond with Nack.

The LWAM supports transmission via ML and SL. The LLMMP arrangement format corresponding to ML transmission is shown in Table 34. The LLSMP arrangement format corresponding to SL transmission is shown in Table 35.

Table 34 Arrangement format of LWAM in LLMMP

Byte	Bit	Field Name	Description
0	7:0	LLPT	Message type, configured as 0xF0.
1	5:0	LLMMT	Logical layer management message type, configured as 0x22.
	7:6	Reserved	Reserved.
2	7:0	LW0_Req–LW7_Req	The working conditions of each lane after adjustment. Bit 0–7 represent the working conditions of lane 0–7: 0: lane disabled 1: lane enabled
3	3:0	LP_MODE	In the link width reduction scenario, the lane to be disabled enters the low power consumption level: 00h: the lane to be disabled entering LP0; 01h: the lane to be disabled entering LP1; 02h: the lane to be disabled entering LP2; 03h: the lane to be disabled entering disable; other values: reserved.
	7:4	Reserved	Reserved.

Byte	Bit	Field Name	Description
4–5	7:0	Reserved	Reserved.
6–7	7:0	CRC16	Check code, checking bytes 1–5.

Table 35 Arrangement format of LWAM in LLSMP

Byte	Bit	Field Name	Description
0	5:0	LLMMT	Logical layer management message type, configured as 0x22.
	7:6	Reserved	Reserved.
1	7:0	LLPL	Message payload length parameter, equal to 2.
2	7:0	LW0_Req–LW7_Req	The working conditions of each lane after adjustment. Bit 0–7 represent the working conditions of lane 0–7: 0: lane disabled 1: lane enabled
3	3:0	LP_MODE	In the link width reduction scenario, the lane to be disabled enters the low power consumption level: 00h: the lane to be disabled entering LP0; 01h: the lane to be disabled entering LP1; 02h: the lane to be disabled entering LP2; 03h: the lane to be disabled entering disable; other values: reserved.
	7:4	Reserved	Reserved.
4–5	7:0	CRC16	Check code, checking bytes 0–2.

6.2.3.3.8 Lane Direction Adjust Message

The lane direction adjust message (LDAM) is a collection of lane direction adjust request parameters, including the specific lane number that requires adjustment.

The LDAM is used when the link is in the service transmission state. The LDAM is initiated when the link bandwidth needs adjustment but cannot be adjusted by link width adjustment. The specified lane direction between ports is adjusted by negotiation with the peer end.

The LDAM requires a response. The RX end agrees to the lane direction adjustment request and responds with Ack to LDAM; otherwise, it needs to respond with Nack.

The message parameters of each lane are independent. If the lane is disabled or does not exist, the corresponding message parameter is fixed to 0.

The LDAM supports transmission via ML and SL. The LLMMP arrangement format corresponding to ML transmission is shown in Table 36. The LLSMP arrangement format corresponding to SL transmission is shown in Table 37.

Table 36 Arrangement format of LDAM in LLMMP

Byte	Bit	Field Name	Description
0	7:0	LLPT	Message type, configured as 0xF0.
1	5:0	LLMMT	Logical layer management message type, configured as 0x23.
	7:6	Reserved	Reserved.
2	7:0	LD0_Req–LD7_Req	Bit 0–7 represent the direction adjust request parameters of lane 0–lane 7: 1b: lane direction adjust request; 0b: lane direction not adjusted.
3–5	7:0	Reserved	Reserved.
6–7	7:0	CRC16	Check code, checking bytes 1–5.

Table 37 Arrangement format of LDAM in LLSMP.

Byte	Bit	Field Name	Description
0	5:0	LLMMT	Logical layer management message type, configured as 0x23.
	7:6	Reserved	Reserved.
1	7:0	LLPL	Message payload length parameter, equal to 1.
2	0	LD0_Req–LD7_Req	Bit 0–7 represent the direction adjust request parameters of lane 0–lane 7: 1b: lane direction adjust request; 0b: lane direction not adjusted.
3–4	7:0	CRC16	Check code, checking bytes 0–2.

6.2.3.3.9 Control Frame Send Location Message

The control frame send location message (CFSLM) is an information parameter indicating the subsequent Control Frame (CF) transmission position.

The CFSLM is used when the link is in a service transmission state. Before the link state changes, a CF needs to be sent. The TX needs to send a CFSLM to achieve communication between the TX sending CF and the RX detecting CF. The location of the CFSLM sent by the TX, the parameter information in the CFSLM, and the subsequent CF sending location must be completely matched. That is, after sending the LLB containing CFSLM and then the CF send location (CF_SL) to specify an LLB, a CF must be sent.

The CFSLM does not require a response.

The CFSLM can only be transmitted via ML. The corresponding LLMMP arrangement format is shown in Table 38.

Table 38 Arrangement format of CFSLM in LLMMP

Byte	Bit	Field Name	Description
0	7:0	LLPT	Message type, configured as 0xF0.
1	5:0	LLMMT	Logical layer management message type, configured as 0x24.
	7:6	Reserved	Reserved.
2	7:0	CF_SL	CF send location: xxh: After the current LLB is sent, send xxh LLBs and then send the CF. (The minimum value of xx is determined by port capacity.)
3–5	7:0	Reserved	Reserved.
6–7	7:0	CRC16	Check code, checking bytes 1–5.

6.2.3.3.10 Ack/Nack Message

The Ack/Nack message (Ack/Nack) is a set of responses to the management message.

The Ack/Nack is used when there is a management message requiring an answer. It is used to respond to the received information and feed back the response or decision of the current port to a piece of information.

The Ack/Nack does not require a response.

The Ack/Nack supports transmission via ML and SL. The LLMMP arrangement format corresponding to ML transmission is shown in Table 39. The LLSMP arrangement format corresponding to SL transmission is shown in Table 40.

Table 39 Arrangement format of Ack/Nack in LLMMP

Byte	Bit	Field Name	Description
0	7:0	LLPT	Message type, configured as 0xF0.
1	5:0	LLMMT	Logical layer management message type, configured as 0x2A.
	7:6	Reserved	Reserved.
2	7:0	Ack/Nack	Response message: 0Fh: Ack; F0h: Nack.
3	5:0	LLMMT_Ack/Nack	Response type, indicating which management message type to respond to:

			Response to LPRM: 0x21 Response to LWAM: 0x22 Response to LDAM: 0x23 Response to ERR_RM: 0x2B
	7:6	Reserved	Reserved.
4–5	7:0	Reserved	Reserved.
6–7	7:0	CRC16	Check code, checking bytes 1–5.

Table 40 Arrangement format of Ack/Nack in LLSMP

Byte	Bit	Field Name	Description
0	5:0	LLMMT	Logical layer management message type, configured as 0x2A.
	7:6	Reserved	Reserved.
1	7:0	LLPL	Message payload length parameter, equal to 2.
2	7:0	Ack/Nack	Response message: 0Fh: Ack; F0h: Nack.
3	5:0	LLMMT_Ack/Nack	Response type, indicating which management message type to respond to: Response to LPRM: 0x21 Response to LWAM: 0x22 Response to LDAM: 0x23 Response to ERR_RM: 0x2B
	7:6	Reserved	Reserved.
4–5	7:0	CRC16	Check code, checking bytes 0–3.

6.2.3.3.11 Error Report Message

The error report message (ERR_RM) is a collection of exception messages that require recovery or retraining. Refer to 6.3.6 for abnormal messages.

The ERR_RM is used when a link is abnormal and needs to be recovered or retrained. It is used to report the errors identified by the current port to the peer port. The current port constructs and sends the ERR_RM according to the type of error occurring in the logical layer, and then enters the Recovery or INIT state; after the peer end receives the ERR_RM and completes the analysis, it enters the Recovery state or INIT state. After the port enters the INIT state, retraining is started.

The ERR_RM needs to respond. The RX receives the ERR_RM and enters the Recovery or INIT state before responding with Ack.

The ERR_RM supports transmission via ML and SL. The LLMMP arrangement format corresponding to ML transmission is shown in Table 41. The LLSMP arrangement format corresponding to SL transmission is shown in Table 42.

Table 41 Arrangement format of ERR_RM in LLMMP

Byte	Bit	Field Name	Description
0	7:0	LLPT	Message type, configured as 0xF0.
1	5:0	LLMMT	Logical layer management message type, configured as 0x2B.
	7:6	Reserved	Reserved.
2	0	CFD	An error indicating CF degradation was reported.
	1	RDD	An error indicating RS decoding degradation was reported.
	2	LLDPD	An error indicating LLDP parsing degradation was reported.
	3	LLMMPD	An error indicating LLMMP parsing degradation was reported.
	4	PTE	A message header error was reported.
	5	DLCRE	A link error was reported.
	6	TXTTE	An error indicating a TX link training timeout was reported.
	7	Reserved	Reserved.
3	0	TSM_F	An error indicating a training start message failure was reported.
	1	CLFM_F	An error indicating a clock lock message failure equilibrium feedback message failure was reported.
	2	EQFM_F	An equilibrium feedback message failure was reported.
	3	LDSE	A lane De-Skew error was reported.
	4	TTE	A training timeout error was reported.
	5	LPRM_F	A low power consumption request message failure was reported.
	6	LWAM_F	A link width adjust request message failure was reported.
	7	LDAM_F	A link direction adjust request message failure was reported.
4	0	CFE	A CF error was reported.

Byte	Bit	Field Name	Description
	1	CFDF	A CF detection failure was reported.
	2	RDE	An RS decoding error was reported.
	3	LLDPE	An LLDP parsing error was reported.
	4	LLMMPE	An LLMMP parsing error was reported.
	5	Reserved	Reserved.
	6	ECCE	An ECC error was reported.
	7	Reserved	Reserved.
5	7:0	Reserved	Reserved.
6–7	7:0	CRC16	Check code, checking bytes 1–5.

Table 42 Arrangement format of ERR_RM in LLSMP

Byte	Bit	Field Name	Description
0	5:0	LLMMT	Logical layer management message type, configured as 0x2B.
	7:6	Reserved	Reserved.
1	7:0	LLPL	Message payload length parameter, equal to 3.
2	0	CFD	An error indicating CF degradation was reported.
	1	RDD	An error indicating RS decoding degradation was reported.
	2	LLDPD	An error indicating LLDP parsing degradation was reported.
	3	LLMMPD	An error indicating LLMMP parsing degradation was reported.
	4	PTE	A message header error was reported.
	5	DLCRE	A link error was reported.
	6	TXTTE	An error indicating a TX link training timeout was reported.
	7	Reserved	Reserved.
3	0	TSM_F	An error indicating a training start message failure was reported.
	1	CLFM_F	An error indicating a clock lock message failure equilibrium feedback message failure was reported.

Byte	Bit	Field Name	Description
	2	EQFM_F	An equilibrium feedback message failure was reported.
	3	LDSE	A lane De-Skew error was reported.
	4	TTE	A training timeout error was reported.
	5	LPRM_F	A low power consumption request message failure was reported.
	6	LWAM_F	A link width adjust request message failure was reported.
	7	LDAM_F	A link direction adjust request message failure was reported.
4	0	CFE	A CF error was reported.
	1	CFDF	A CF detection failure was reported.
	2	RDE	An RS decoding error was reported.
	3	LLDPE	An LLDP parsing error was reported.
	4	LLMMPE	An LLMMP parsing error was reported.
	5	Reserved	Reserved.
	6	ECCE	An ECC error was reported.
	7	Reserved	Reserved.
5–6	7:0	CRC16	Check code, checking bytes 0–4.

6.2.3.4 Response Mechanism

The response mechanism of each management command is shown in Tables 43 to 48.

Table 43 Description of CLFM response mechanism in each scenario

Lane State Corresponding to Lane_ID	Swing_Req	CL	Ack/Nack	Lane next status
In clock lock process	0h to 7h	0b/1b	Ack	Waiting for the clock lock
	Fh (invalid Swing gear request)	1b	Ack	Waiting for lane equilibrium
		0b	Ack	Returning to INIT state
	8h–Eh (invalid value)	0b/1b	Ignore	Waiting for the clock lock

Lane State Corresponding to Lane_ID	Swing_Req	CL	Ack/Nack	Lane next status
In equilibrium process	Fh	1b	Ack	Waiting for lane equilibrium
	0h to Fh	0b/1b	Ignore	Waiting for lane equilibrium
Not in clock lock or equilibrium process	0h to Fh	0b/1b	Ignore	Ignore

Table 44 Description of EQFM response mechanism in each scenario

Lane State Corresponding to Lane_ID	FFE_Req	Swing_Req_Enable	EQD	Ack/Nack	Lane next status
In equilibrium process	0h–the maximum gear supported by the actual PHY	1b	0/1	Ack	Waiting for lane equilibrium
		0b	0/1	Ack	Waiting for lane equilibrium
	7Fh	1b	0/1	Ack	Waiting for lane equilibrium
		0b	1	Ack	Waiting for lane lock
		0b	0	Ack	Returning to INIT
	Exceeding the maximum gear supported by the actual PHY	0/1	0/1	Ignore	Waiting for lane lock
In lane lock process	7Fh	0b	1	Ack	Waiting for lane lock
	Not 7Fh	0/1	0/1	Ignore	Waiting for lane lock
Not in equilibrium or lane lock process	0h to 7Fh	0/1	0/1	Ignore	Waiting for lane lock

Table 45 Description of LPRM response mechanism in each scenario

RXLKSM Link State	LP_GEAR	LP_MODE	LP_WAY	ACK /NACK	Link next status
The link is in one of the following states: ● Disable; ● INIT; ● Training; ● Recovery; ● LP0f; ● LP0/1/2.	00h to FFh	0/1	0/1	Ignore	Not triggering jump
HS	00h to 03h	0	0/1	ACK	Entering the low power consumption process
		1	0/1	Ignore	Not triggering jump
	10h	0	0	ACK	Entering the low power consumption process
			1	Ignore	Not triggering jump
		1	0/1	Ignore	Not triggering jump
	04h to 0Fh, 11h to FFh (invalid value)	0/1	0/1	Ignore	Not triggering jump
LP3	03h	0	NA	Ignore	Not triggering jump
	03h	1	1	ACK	Entering the wake-up process
			0	Ignore	Not triggering jump
	Non-03h	0/1	0/1	Ignore	Not triggering jump

Table 46 Description of LWAM response mechanism in each scenario

Link State	LW_Req	LP_MODE	Ack/Nack	Link next status
Non HS	0h to FFh	0h to Fh	Ignore	Not triggering jump
HS	All lanes to be enabled as instructed by the LWAM are enabled	0h to Fh	Nack	Not triggering jump
	All lanes to be disabled as instructed by the LWAM are disabled	0h to Fh	Nack	Not triggering jump
	The LWAM instructs lanes to be enabled and disabled at the same time	0h to Fh	Nack	Not triggering jump
	The LWAM only instructs disabled lanes to be enabled, and the lanes to be enabled are in the same low power consumption state	0h to Fh	Ack	Entering the lane upsizing process
	The LWAM only instructs disabled lanes to be enabled, and the lanes to be enabled are in the different low power consumption state (disable/LPx)	0h to Fh	Nack	Not triggering jump
	The LWAM only instructs enabled lanes to be disabled, and the low power consumption state of continuously disabled lanes is consistent	0h to 3h	Ack	Entering the lane downsizing process
		4h to Fh	Ignore	Not triggering jump
	The LWAM only instructs enabled lanes to be disabled, and the low power consumption state of continuously disabled lanes is inconsistent	0h to Fh	Nack	Not triggering jump

Table 47 Description of LDAM response mechanism in each scenario

Link State	LDx_Req	Lane Mode	Lane State	Ack/Nack	Link next status
Non HS	0h to FFh	NA	NA	NA	Ignored; not triggering jump
HS	0h	NA	NA	Nack	Not triggering jump
	1h to FFh	The direction-switch	NA	Nack	Not triggering

Link State	LDx_Req	Lane Mode	Lane State	Ack/Nack	Link next status
		ed lane includes the RX lane at the TX end			jump
		The direction-switched lane does not include the RX lane at the TX end	The direction-switched lane has the following states: disable; LP0f/LPx; recovery; INIT.	Nack	Not triggering jump
			The direction-switched lane includes some HS lanes	Ack	Entering the direction switching process
			The direction-switched lane includes all HS lanes	Ack	Entering the direction switching process

Table 48 Description of ERR_RM response mechanism in each scenario

Link State	ERR_RM	Ack/Nack	Link next status
Disable	NA	Ignore	Ignore
INIT	Instructing to jump to INIT	Ack (ERR_RM retransmission scenario)	Not triggering jump
	Instructing to jump to Recovery	Ignore	Ignore
Training	Instructing to jump to INIT	Ack	Entering the retraining process
	Instructing to jump to Recovery	-	Ignore
HS	Instructing to jump to INIT	Ack	Entering the retraining process
	Instructing to jump to Recovery	Ack	Entering the recovery process
Recovery	Instructing to jump to INIT	Ack	Entering the retraining process
	Instructing to jump to Recovery	Ignore	Ignore
The link is in one of the	Instructing to jump to	Ack	Entering the

Link State	ERR_RM	Ack/Nack	Link next status
following states: LP0f/LP0/LP1/LP2/LP3	INIT		retraining process
	Instructing to jump to Recovery	Ignore	Ignore

6.2.4 Control Frame

6.2.4.1 Overview

A control frame (CF) is transmitted on a single lane to send information such as link management or training sequences. When CFs need to be transmitted, all contents of each CF must be completely transmitted on a single lane; continuity must be ensured between CFs and between CFs and data frames, that is, no content can be inserted between two frames. The insertion position of a CF is instructed by the logical layer management message CFSLM. The CF structure is shown in Figure 21.

Figure 21 CF data structure

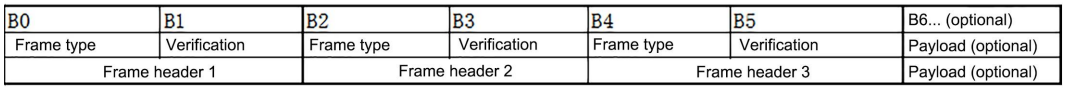


Table 49 CF field segment

Field Name	Description
Frame type	Indicates the frame type, as described in Table 50. Length: 1 byte.
Verification	Indicates the check bit for frame type. Length: 1 byte. When CRC8 is used, the check bit can correct 1-bit errors and detect 2-bit errors.
Payload	Indicates the information carried by a specific type of CF. Refer to the description of each CF. Length: variable.

The contents of frame header 1, frame header 2, and frame header 3 in the CF structure are the same, forming a repetition code.

Unless otherwise specified, the CF (frame header + payload) is not scrambled or precoded.

The frame header (frame type + check) of the CF is optional. The CFs supported in the logical layer are shown in Table 50.

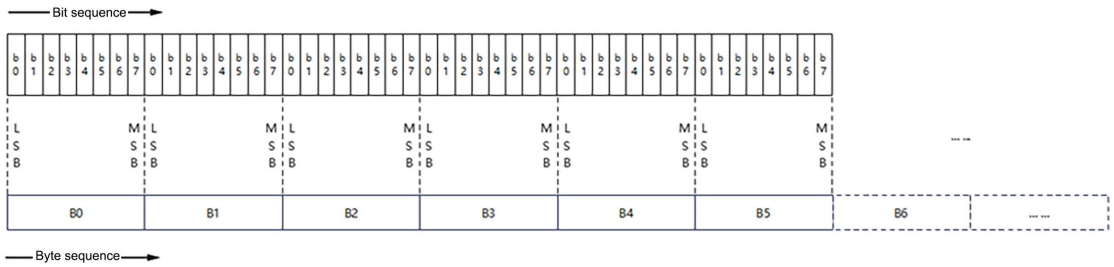
Table 50 CF types

Control Frame	Frame Type	Verification	Payload Length	Description
LLCF_TS0	--	--	Variable	A clock lock sequence with a 01 payload.
LLCF_TS1	--	--	Variable	A training sequence with a PRBS11 payload.

Control Frame	Frame Type	Verification	Payload Length	Description
LLCF_TS2	0x6C	0x56	8 bytes	A sequence with a PRBS11 payload, which is used for sending and receiving synchronization confirmation.
LLCF_EI	0x65	0x69	0	Marks the end of frame data.
LLCF_DS	0x4B	0xA3	0	Marks the start of a new LLB transmission.
LLCF_EIE	--	--	Variable	Used to wake from electrical idle.
LLCF_PAD	0xD2	0x65	Variable	Used for data filling and scrambling code seed reset when adjusting the number of main link lanes.

The CF transmission bit sequence is shown in Figure 22.

Figure 22 CF data transmission bit sequence



The CF transmission bit sequence is consistent with the line bit sequence. For constant values, such as 0xABCD, 0xCD is the byte received first (B0) and 0xAB is the byte received later (B1).

6.2.4.2 Logical Layer Control Frame

6.2.4.2.1 LLCF_TS0

LLCF_TS0 is used for clock lock in the training phase, with a payload of 0xAA of any length.

6.2.4.2.2 LLCF_TS1

LLCF_TS1 is used for lane parameter tuning calculations during the training phase.

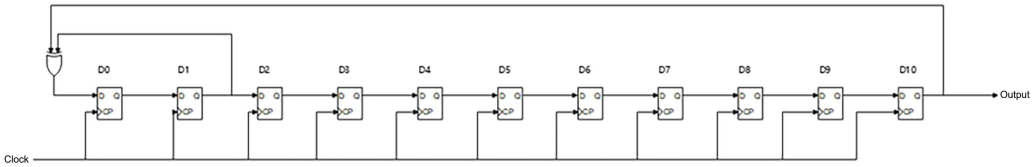
The payload is PRBS11 with an arbitrary length.

Polynomial: $G(x) = x^{11} + x^2 + 1$.

Seed: 00110011001b.

The corresponding LFSR is shown in Figure 23.

Figure 23 LFSR of LLCF_TS1 PRBS11



The PRBS sequence is continuously generated, and the generated bit sequence is consistent with that sent on the line.

The first groups of 64-bit binary bit sequence examples are shown in Table 51.

Table 51 LLCF_TS1 payload example

No.	Payload	
0	00110011001011010010111000101110	01010001101000101001000001000010
1	10100000010001010101111101110110	01110010010110000101110000001001
2	11111110100110011011110000111001	11111001011001101000111100010100
3	11010111010010001101111010110110	01000111000010100111111011110011
4	00011000011111000000110010101011	01000100011101011111001000110010
5	11111000010011000000101101010100	01110111010110001101110000011100
6	10101001111011101001101100010110	11010111000111011000001101101010
7	11011011101101101100011100011111	00101001100001000011111101010011
8	00100010110100000100100000001011	11111110110011001110000111100101
9	01100101110110100011011011111000	11100110101100001110110101001110
10	00100001101011111100010011001010	00011110111111001110011000011010
11	01010110111101110001100011010110	10110111000100100111110100001100
…	… …	

6.2.4.2.3 LLCF_TS2

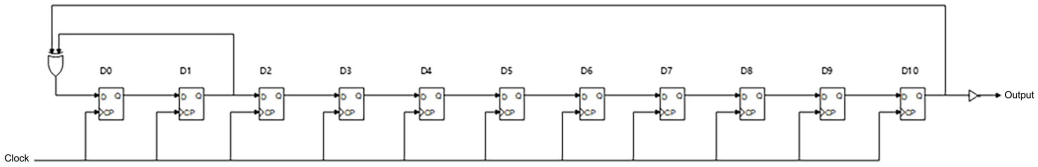
LLCF_TS2 is used to send and receive confirmation of synchronization state. The payload length is fixed at 8 bytes.

The payload is the bit flip (bit inversion) of the PRBS11 sequence.

Polynomial: $G(x) = x^{11} + x^2 + 1$.

Seed: 00110011001b.

Figure 24 Payload generation of LLCF_TS2



The PRBS sequence is continuously generated, and the generated bit sequence is consistent with that sent on the line.

The first groups of 64-bit binary bit sequence examples are shown in Table 52.

Table 52 LLCF_TS2 payload example

No.	Payload
0	1100110011010010110100011101000 1 1010111001011101011011111011101
1	0101111110111010101000001000 1001 1000110110100111101000111111011 0
2	0000000101100110010000111100011 0 0000011010011001011100001110101 1
3	0010100010110111001000010100 1001 1011100011110101100000010000110 0
4	1110011110000011111100110101010 0 1011011100010100000110111001101
5	0000011110110011111101001010101 1 1000100010100111001000111110001 1
6	0101011000010001011001001110100 1 0010100011100010011111001001010 1
7	0010010001001001001110001110000 0 1101011001111011110000001010110 0
8	1101110100101111101101111111010 0 0000000100110011000111100001101 0
9	1001101000100101110010010000011 1 0001100101001111000100101011000 1
10	1101111001010000001110110011010 1 1110000100000011000110011110010 1
11	1010100100001000111001110010100 1 0100100011101101100000101111001 1
…	… …

6.2.4.2.4 LLCF_EI

LLCF_EI is used to mark the end of transmission in this lane, after which data are discarded. If the lane stops receiving data, it must be re-enabled to restart data reception. This CF has no payload.

For the specific LLCF_EI sending time and lane restart control, refer to 6.3.3 State Management.

6.2.4.2.5 LLCF_DS

LLCF_DS marks the starting position of a new LLB, and the LLB data is transmitted immediately after LLCF_DS. This CF has no payload.

The TX should transmit LLCF_DS simultaneously on all valid lanes during the first transmission or lane number switching (including low power consumption reasons).

Within the allowable skew (Table 1 TP1 skew of the TX end), all valid lane signals should be transmitted simultaneously.

6.2.4.2.6 LLCF_EIE

LLCF_EIE is used to wake up the TX through a high-rate link line.

At a single lane rate of HS1 (2 Gbps or 4 Gbps), the payload length is 16 bytes and the payload is fixed at 0x0000_FFFF_0000_FFFF_0000_FFFF_0000_FFFF (0x0000_FFFF repeated 4 times in a row).

At a single lane rate of HS2 (6 Gbps or 8 Gbps), the payload length is 32 bytes and the payload is fixed at 0x0000_0000_FFFF_FFFF repeated 4 times in a row.

At a single lane rate of HS3 (10 Gbps, 12 Gbps, or 16 Gbps), the payload length is 64 bytes and the payload is fixed at 0x0000_0000_0000_0000_FFFF_FFFF_ FFFF_FFFF repeated 4 times in a row.

At a single lane rate of HS4 (20 Gbps or 24 Gbps), the payload length is 96 bytes and the payload is fixed at 0x0000_0000_0000_0000_0000_0000_ FFFF_FFFF_ FFFF_FFFF_FFFF_FFFF repeated 4 times in a row.

6.2.4.2.7 LLCF_PAD

This CF is used for filling. This CF is only allowed to be sent after a link is established and before the LLB is transmitted; it is not allowed to be sent after the LLB is transmitted. The RX directly discards this CF after receiving it.

The payload length of this CF is variable, and the payload structure is as follows.

Figure 25 LLCF_PAD payload structure

B6 (optional)	B7... (optional)	Bn-2	Bn-1	Bn
0x0F (optional)	0x0F... (optional)	0xF0	PCRC	
Filling (optional)		End indication	Payload verification	

The payload starts with a padding byte 0x0F, which is sent continuously until the end. An end indication byte 0xF0 is sent at the end of the padding. Finally, a 2-byte CRC16 is sent, calculated over both the padding and the end indicator. Besides the payload check, the length of the remaining payload is less than 32 bytes.

The end indication byte can be sent directly without sending the padding byte. At this time, the CRC only calculates the CRC16 of the end indication byte.

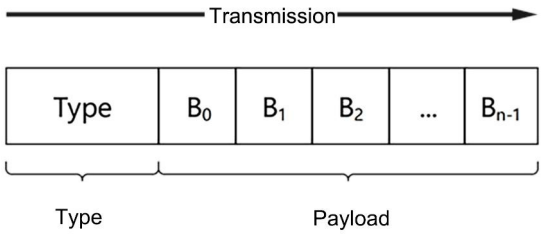
The frame structure of the above conditions is shown in Figure 26.

Figure 26 LLCF_PAD payload structure (no padding bytes)

B6	B7	B8
0xF0	PCRC	
End indication	Payload verification	

6.2.5 Sideband Link Logical Layer Packet

Figure 27 Sideband link logical layer packet



In the figure, B_0 to B_{n-1} are n-byte-long payloads. B_0 indicates the first byte data input to the sideband link.

Type is a one-byte indicator of the payload type.

Table 53 Sideband link logical layer packet

Type	Package Type Description	Payload Description
0x0F	Transport layer data packet	Transport layer data.
0x33	High-rate link logical layer control packet	High-rate link logical layer control packet.

The byte transmission order of the packet data is consistent with the byte order on the line.

6.3 Main Link

6.3.1 Overview

The port main link determines the electrical parameters of the connection between ports through link training, and completes the link establishment based on the negotiated parameters to enter the service transmission state.

The main link logical layer maintains the link state through state management. When the main link enters the service transmission state, the main link logical layer performs encoding and decoding functions to complete the conversion between electrical layer data and transport layer data. When the port performs link recovery, a link can be quickly established based on the saved parameters.

6.3.2 Link Training

6.3.2.1 Overview

Link training includes independent training processes for each lane and multi-lane alignment (deskew).

The lane-independent training process includes lane clock lock, lane equilibrium, and lane lock. The success of the lane-independent training process indicates that the lane has successfully completed the lane clock lock, lane equilibrium, and lane lock, that is, the LLFM indicates that the

lane lock is successful. The failure of the lane-independence training process indicates that lane clock lock, lane equilibrium, or lane lock has not been successfully completed.

6.3.2.2 Clock Lock

6.3.2.2.1 Process Introduction

After the link training phase begins, the lane to be trained needs to be initialized at a specified rate, and then the lane clock is recovered and locked.

When lane clock recovery and lock are performed, the TX of the lane to be trained continuously sends LLCF_TS0. In the initial phase, the TX needs to select the maximum Swing gear supported by the TX lane to send LLCF_TS0. The TX then sends a logical layer management message packet TSM to the RX and indicates in the TSM all lane numbers that initiate training. After the RX receives the TSM, the corresponding lane can start clock recovery and lock.

To prevent RX clock lock failure, the TX can provide up to eight Swing adjustable gears.

Note: Vendors can evaluate and define "whether the clock lock is successful" by themselves to ensure that the CDR at the RX is locked.

At the beginning of lane clock recovery and lock, the TX should select the maximum Swing gear to send LLCF_TS0 to ensure that the RX can normally complete the clock lock at the initial phase of lane clock recovery and lock.

If the TX of a lane uses the maximum Swing gear and the RX of the lane cannot complete the clock lock, the rate training of the lane fails and the following process starts execution:

(a) The RX needs to inform the TX through the logical layer management packet CLFM and indicate that the lane training has failed.

(b) After receiving the CLFM, the TX resets the lane (setting the state machine of the corresponding lane to INIT state), and then feeds back the Ack corresponding to the CLFM to the RX.

(c) After receiving the Ack, the RX resets the lane (setting the state machine of the corresponding lane to INIT state).

If the clock of the RX of a lane is locked successfully and there is no need to adjust the Swing gear of the TX, follow the following process:

(a) The RX needs to inform the TX through the logical layer management packet CLFM, and indicate that the lane clock is successfully locked and the Swing gear of the peer end will no longer be adjusted.

(b) After receiving the CLFM, the TX switches the pattern of the clock-locked lane. Then, the TX continues to send LLCF_TS1, and feeds back the Ack corresponding to the CLFM to the RX. The clock-locked lane of the TX enters the lane equilibrium phase.

(c) After the RX detects Ack, the lane locked by the RX clock enters the lane equilibrium phase.

If the clock of a lane at the RX is successfully locked and the Swing gear of the TX needs to be adjusted to reduce the link power consumption, the following process should be executed:

(a) The RX requests the TX to reduce the corresponding lane Swing by sending a logical layer management packet CLFM, and indicates the gear to be adjusted.

(b) After receiving the CLFM, the TX completes the Swing adjustment for the lane whose Swing parameters need to be adjusted, and then feeds back the Ack corresponding to the CLFM to the RX.

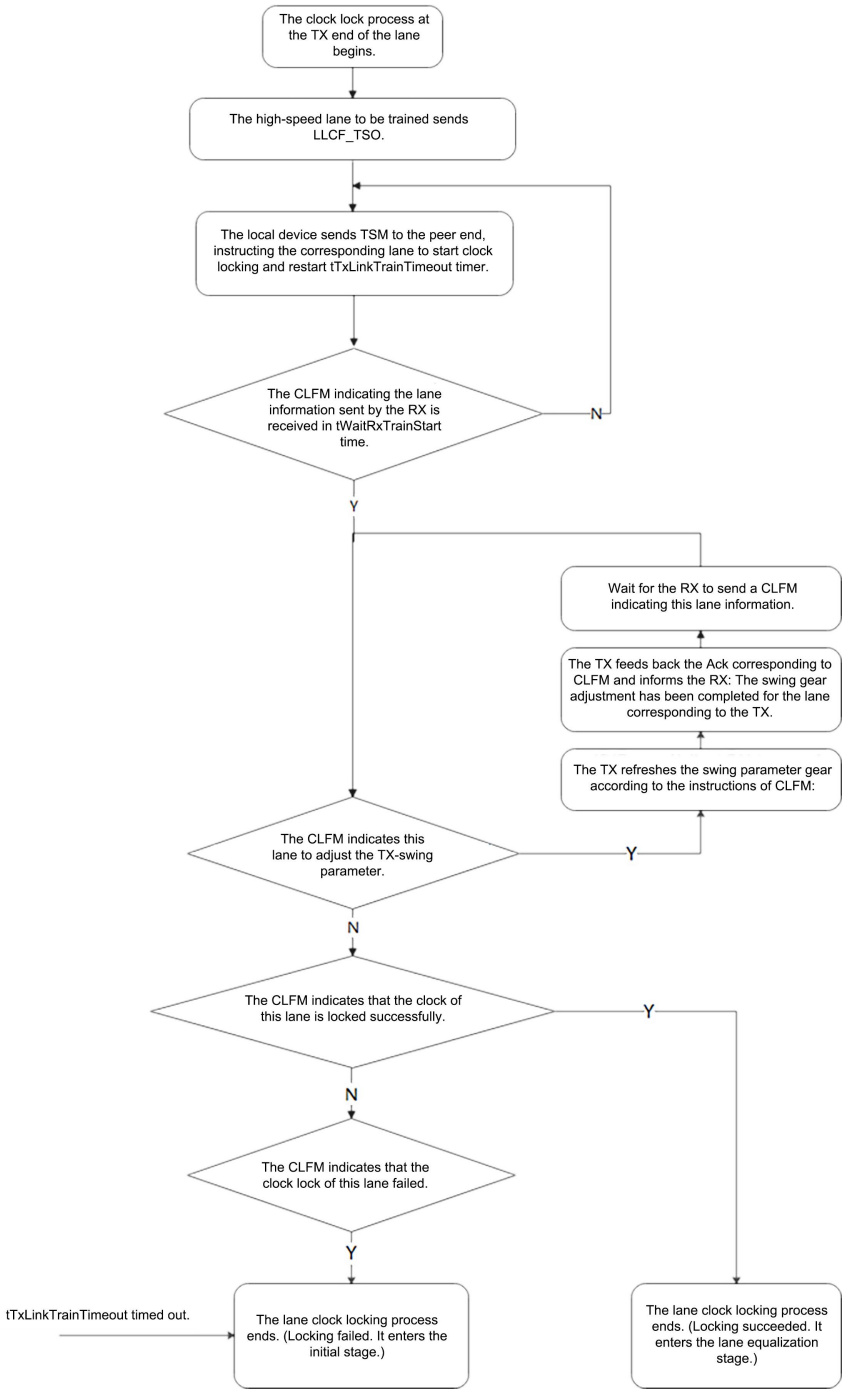
(c) After the RX detects Ack, it re-evaluates the clock lock and recovery of the lane that has adjusted Swing.

(1) If the RX fails to re-evaluate the clock lock, it is necessary to continue to request an increase in the TX's Swing gear by sending a logical layer management packet CLFM. Repeat this process until an appropriate TX's Swing gear is found to meet both the clock lock and power consumption requirements.

(2) If the RX re-evaluates that the clock lock is successful, it can continue to send a logical layer management packet CLFM to request lowering the Swing gear of the TX to reduce power consumption. Repeat this process, or send a logical layer management packet CLFM to indicate that the lane clock lock is successful, and that the Swing gear of the peer end does not need adjustment. The RX enters the lane equilibrium phase.

The clock lock process of the single lane of the TX is as follows.

Figure 28 Clock lock process at the TX of a lane

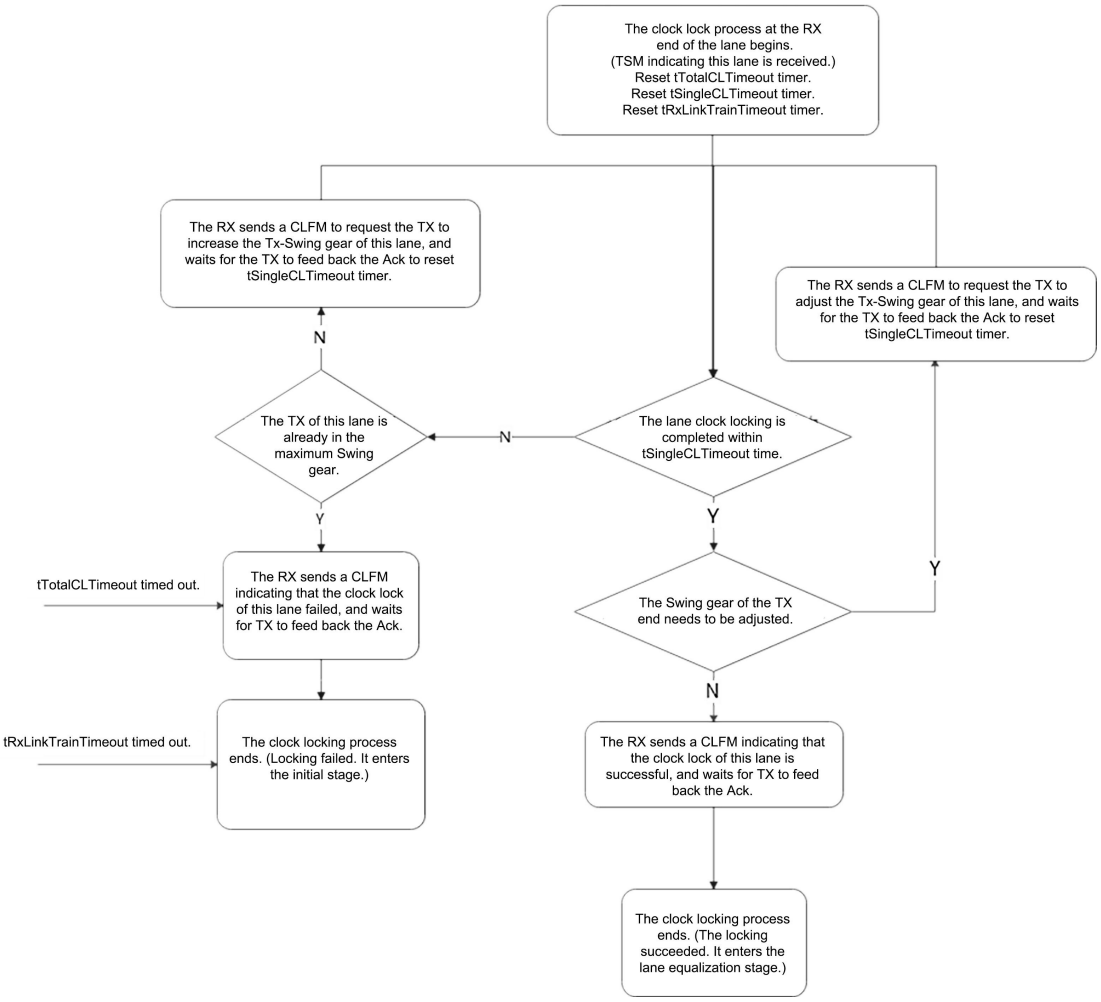


In the above figure, it is necessary to count the number of tWaitRxTrainStart timeouts and report them to the upper layer. After each timeout, the number of timeouts is added by one. When the sending link is re-established, the timeout counter is cleared.

After the tTxLinkTrainTimeout times out, it needs to be reported to the upper layer.

The clock lock process of the single-lane RX is as follows.

Figure 29 Clock lock process at the RX of a lane

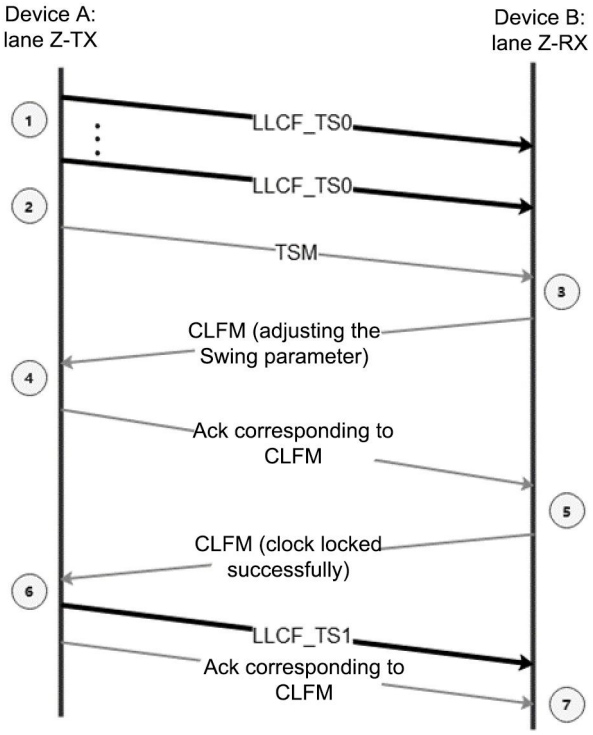


In the above figure, after the tSingleClTimeout times out, follow the process above. If tTotalClTimeout times out, it needs to be reported to the upper layer.

6.3.2.2.2 Exchange Process

The following is a schematic diagram of the clock lock process for any lane. The lane number is Z. Device A is the TX device of lane Z, and device B is the RX device of lane Z. Devices A and B are directly connected to each other. Figure 30 shows the exchange information of their connected ports.

Figure 30 Clock lock exchange



The process is detailed as follows:

(1) After the TX of lane Z completes initialization, it starts to perform lane clock recovery, and device A continuously sends LLCF_TS0 back-to-back through lane Z.

(2) Device A sends a TSM to device B, indicating that the TX of lane Z has started sending the lane clock lock code pattern LLCF_TS0. It shall be ensured that the TSM is sent no earlier than LLCF_TS0 through lane Z.

Note: The TSM here supports transmission through the main link and sideband link. If the current high-rate transmission link of device A is available (in a non-disconnected state, that is, the corresponding TXLKSM is in HS state), device A can currently send the TSM to device B through the high-rate transmission link. Otherwise, it should send the TSM to device B through the sideband link.

(1) Device B receives the TSM, and the RX end of lane Z can start lane clock lock.

The RX of lane Z starts to perform lane clock lock. If the clock cannot be locked, it requests the TX of lane Z to switch new Swing parameters through the CLFM.

(2) After receiving the CLFM of the corresponding lane Z to switch Swing parameters, device A executes the following process:

Step 1: Adjust the Swing parameters of the TX of lane Z based on the message indication of the CLFM.

Step 2: Device A sends an Ack corresponding to the CLFM to notify device B that the TX of lane Z has adjusted the Swing parameters.

Note: In order to prevent the loss of the Ack response message corresponding to the CLFM, device A needs to reply with the corresponding Ack response message every time it receives a

CLFM sent by device B. To ensure the effective transmission of the message, after receiving a repeated CLFM sent by device B, device A should reply with the corresponding Ack response message through the sideband lane. The same response mechanism works for the following EQFM, LPRM, LWAM, LDAM, and ERR_RM messages.

(1) After receiving the Ack that the corresponding lane Z has adjusted the Swing parameters, device B starts to re-evaluate the lane clock lock. If the expectations are not met, device B requests new parameters through the CLFM (the same as process 4). If the expectations are met, device B informs device A through the CLFM that the lane Z clock lock is successfully completed, and starts to execute the lane equilibrium process after receiving the corresponding Ack.

Note: To avoid CLFM loss, device B must retransmit the CLFM via the sideband lane if no Ack is received from device A within the time specified by tCLFMAck. CLFM can be retransmitted a maximum of LLMP_MaxReSendTime (See 6.3.5 for the specific description of training parameter requirements) times. If no Ack sent by device A is received after sending CLFM for LLMP_MaxReSendTime times, an ERR_RM is sent to device A and an abnormal LLMP handshake is reported. The same LLMP retransmission and handshake exception reporting mechanism works for the following EQFM, LPRM, LWAM, LDAM, and ERR_RM messages.

(2) Upon receiving the CLFM indicating a successful clock lock on Lane Z, device A completes the clock lock procedure for that lane. Lane Z then switches its code pattern and begins continuously transmitting LLCF_TS1 ordered sets. Subsequently, device A sends an Ack response for the CLFM to device B's RX and initiates the lane equilibrium process.

(3) After receiving the Ack corresponding to the CLFM indicating that the lane Z clock lock is successful, device B initiates the lane equilibrium process.

6.3.2.3 Lane Equilibrium

6.3.2.3.1 Process Introduction

After the lane clock lock is completed, both the TX and RX of the lane enter the lane equilibrium phase.

During lane equilibrium, the TX of the lane to be trained continuously sends LLCF_TS1. The RX receives and detects LLCF_TS1, and performs equilibrium training on the receiving end of the lane to evaluate the equilibrium training results.

To prevent the RX's equilibrium training results from not meeting expectations, the TX can adjust up to 64 levels of FFE parameters and 8 levels of Swing parameters.

Note: The "equilibrium training results meeting expectations" described above can be evaluated by vendors through error detection, eye diagram measurement, and other methods to ensure that the BER after equilibrium training is less than the protocol specification standard (1×10^{-12}).

If the RX of the lane detects a continuous 128-byte LLCF_TS1 and the equilibrium training result meets the expectation, it can be considered that the equilibrium training of the lane is successful.

If the equilibrium training of a lane at the RX fails, the corresponding lane at the TX can be requested to adjust the FFE and Swing parameters through the logical layer management message EQFM, as well as the gear position. After receiving the EQFM, the TX adjusts the FFE and Swing parameter for the target lane, and then feeds back the Ack corresponding to the EQFM to the RX. After the RX detects the Ack, it re-performs lane equilibrium training on the lane that has adjusted FFE and Swing parameters and detects LLCF_TS1.

If the TX Swing gear of a lane has been adjusted to the maximum and all FFE gear equilibrium training does not meet expectations, the RX needs to inform the TX through the logical layer

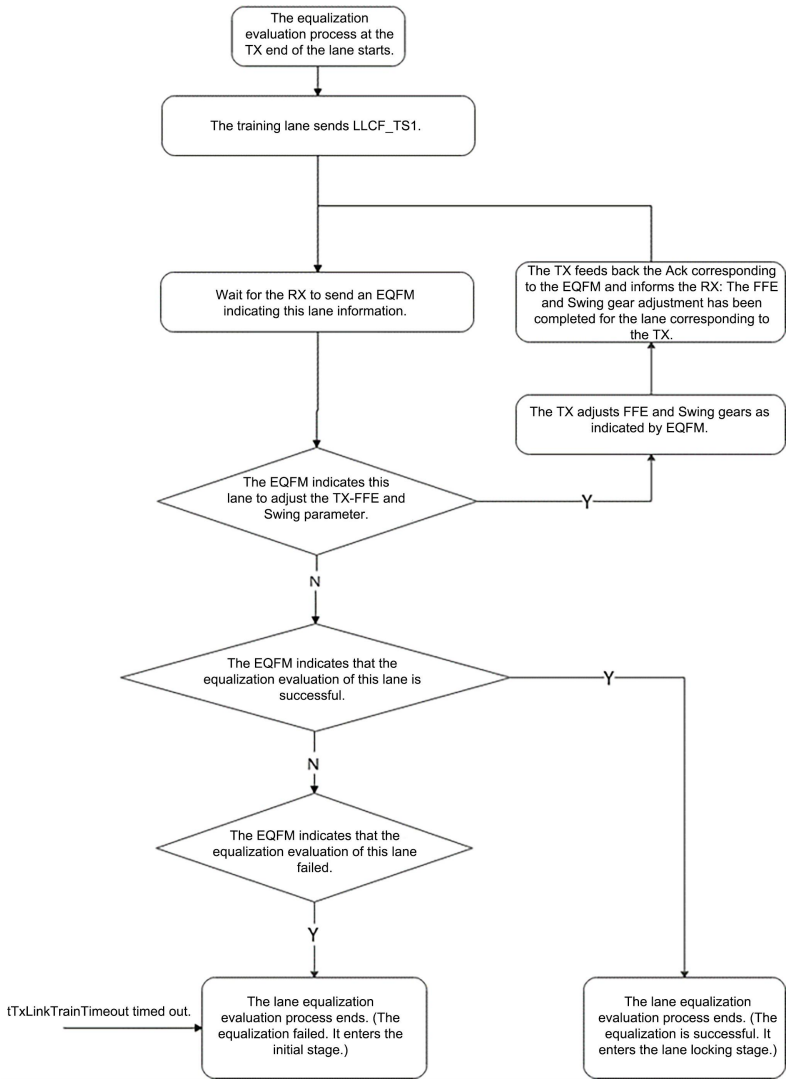
management message EQFM and indicate that the lane training has failed. After receiving the EQFM, the TX resets the lane (setting the state machine of the corresponding lane to INIT state), and then feeds back the Ack corresponding to the EQFM to the RX. After receiving the Ack, the RX resets the lane (setting the state machine of the corresponding lane to INIT state).

If the equilibrium training of a lane at the RX is successful, the RX informs the TX through the logical layer management message EQFM and indicates that the lane training is successful. After receiving EQFM, the TX switches the pattern of the lane indicating successful lane equilibrium in EQFM. After switching, it continuously sends LLCF_TS2, and then feeds back the Ack corresponding to the EQFM to the RX. The TX of this lane enters the lane lock phase. After the RX detects the Ack, the RX of this lane enters the lane lock phase.

If all lane equilibrium training of this link fails, the current rate training fails.

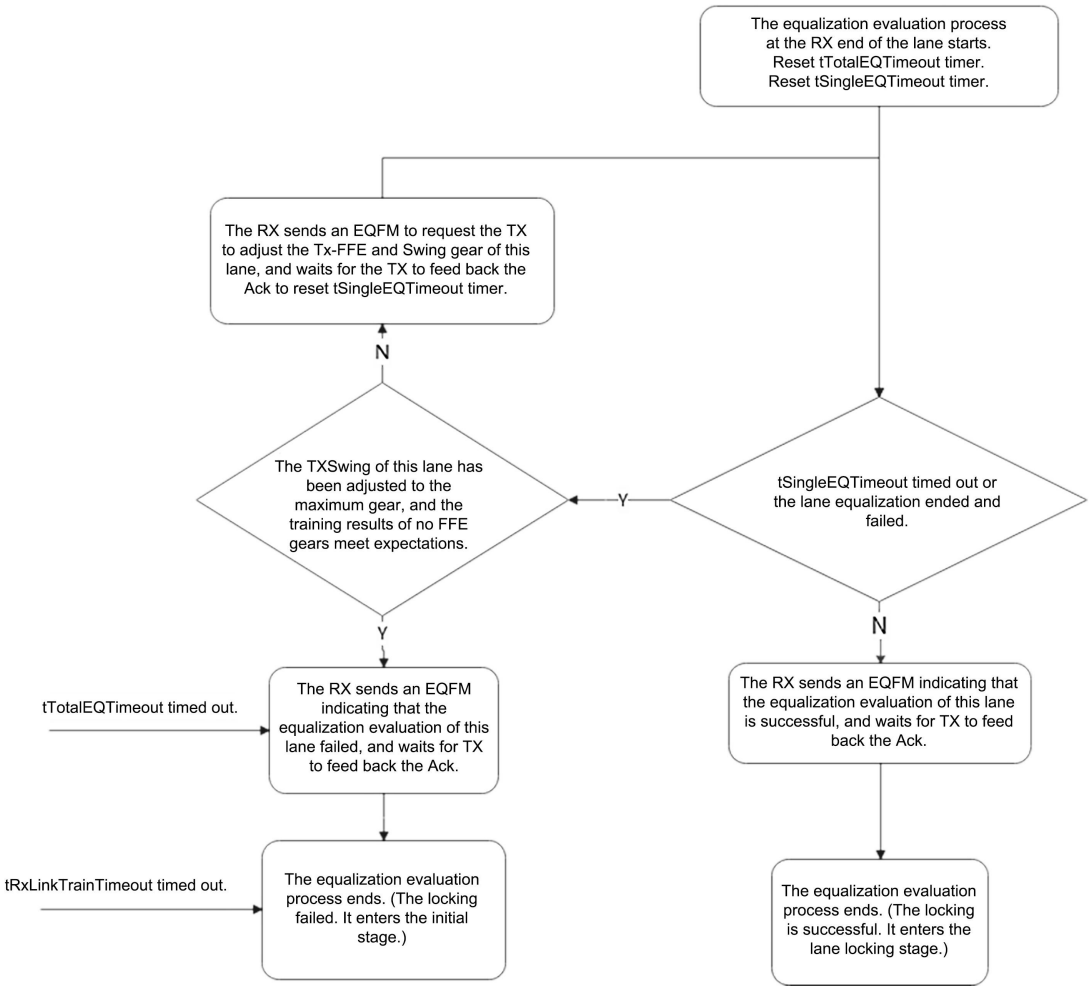
The process of single-lane TX equilibrium is detailed in Figure 31.

Figure 31 Lane TX equilibrium process



The process of single-lane RX equilibrium is detailed in Figure 32.

Figure 32 Lane RX equilibrium process



In the above figure, after tSingleEQTimeout times out, follow the process above. If tTotalEQTimeout times out, it needs to be reported to the upper layer.

To enable faster lane recovery, it is also necessary to support a mechanism for the RX of the lane to switch the TX FFE parameters based on the EQFM during the lane recovery phase (LNSM.recovery).

Note: For the RX of the lane, there are the following implementation suggestions when executing equilibrium:

(a) The RX can poll multiple sets of FFE parameters and Swing at the TX to achieve the best equilibrium results.

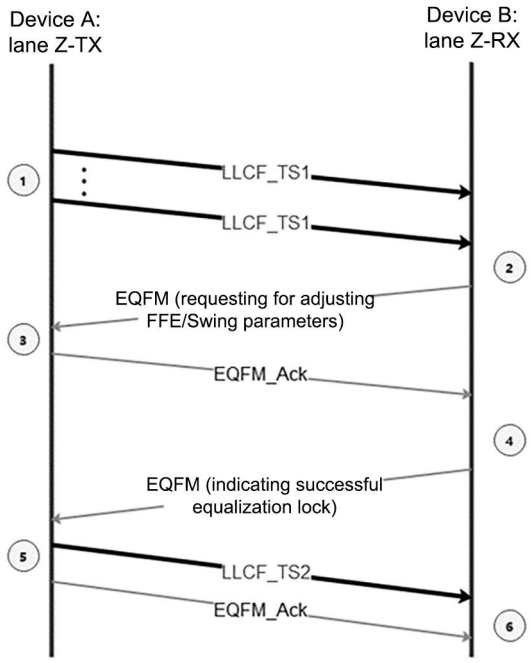
(b) The RX should save the adaptive parameters after successful equilibrium of each rate and store them in the non-volatile storage of the device to facilitate rapid link establishment next time.

6.3.2.3.2 Exchange Process

The following is a schematic diagram of the equilibrium process for any lane. The lane number is Z. Device A is the TX device of lane Z, and device B is the RX device of lane Z. Devices A and B are

directly connected to each other. Figure 33 shows the exchange information of their connected ports.

Figure 33 Lane equilibrium exchange process



The process is detailed as follows:

(1) After device A receives the CLFM indicating that lane Z has completed clock lock and feeds back the corresponding Ack, the TX of lane Z starts the lane equilibrium process, and the TX of lane Z continuously sends the link training sequence LLCF_TS1 during the equilibrium phase.

(2) After device B receives the Ack corresponding to the CLFM indicating that lane Z has completed clock lock, the RX of lane Z starts to perform equilibrium training and evaluates whether the training meets the expectations. If not, it requests lane Z through the EQFM, and the TX switches new FFE parameters.

(3) After receiving the EQFM corresponding to the switched FFE parameters of lane Z, device A executes the following process:

Step 1: Adjust the FFE/Swing parameters of the TX of lane Z based on the message indication of the EQFM.

Step 2: Device A sends an Ack corresponding to the EQFM to notify device B that the TX of lane Z has adjusted the FFE/Swing parameters.

(4) After receiving the Ack that the corresponding lane Z has adjusted the FFE parameters, device B restarts to equilibrium and evaluates whether the equilibrium training meets the expectations. If not, it requests new parameters through the EQFM (same as process 2). If it meets the expectations, it informs device A through the EQFM that the equilibrium of lane Z is successfully completed.

(5) After device A receives the EQFM indicating that lane Z equilibrium is successful, lane Z switches the transmitted code pattern, and continues to send LLCF_TS2. Then device A feeds back the Ack corresponding to the EQFM of device B to the RX, and starts executing the lane locking process.

(6) After receiving the Ack corresponding to the EQFM indicating that the lane Z equilibrium is successful, device B initiates the lane lock process.

6.3.2.4 Lane Lock

A lane will enter the lane lock mode in the following scenarios:

(a) In the initial link establishment (normal link establishment) process, after the lane equilibrium training is completed, it enters the lane lock phase.

(b) In the initial link establishment (quick link establishment) process, after the lane initialization is completed, it enters the lane lock phase.

In this process, after the lane to be trained is initialized, the lane TX continues to send LLCF_TS2. The TX then sends a logical layer management message TSM to the RX and indicates in the TSM all lane numbers that initiate lane lock. After the RX receives the TSM, the corresponding lane can start lane lock.

Some lanes do not support quick link establishment. If any lane fails to establish a quick link, the entire link establishment is considered a failure, and the result will be reported to software.

(c) When the link is restored, it enters the lane lock phase.

Lane lock is mainly used for data boundary lock of lanes.

During lane locking, the TX of the lane to be trained continuously sends LLCF_TS2. The RX samples the incoming bit stream and correlates it with the LLCF_TS2 code pattern (successfully matching both the frame header and payload) to identify the start and end of LLCF_TS2. The RX locking state of a lane can be divided into 3 states: unlocked state, aligned state, and locked state.

(a) Unlocked state: When the lane lock starts, it is in this state by default. In this state, the RX of the lane continuously receives and detects the bit stream and matches it with LLCF_TS2. Once the matching is completed, the data boundary alignment is completed, the current boundary alignment position is recorded, and the RX enters the aligned state.

(b) Aligned state: In this state, the RX of the lane is aligned according to the original boundary position, continuously receives and detects the bit stream, and matches it with LLCF_TS2. If four consecutive LLCF_TS2 are matched (both the header and payload are matched), the RX enters the locked state. If continuous LLCF_TS2 cannot be identified, the RX jumps back to the unlocked state.

(c) Locked state: The main link completes the lane lock, and the boundary lock position is not allowed to be adjusted.

When the RX of a lane enters the locked state, it is deemed that the lane has been successfully locked.

After each lane enters the lane lock process, the tLaneLockTimeout timeout should be enabled. If the lane RX can complete the lane lock within tLaneLockTimeout time, it is considered a successful lane lock. If the lane cannot be locked within tLaneLockTimeout, it is considered a lane lock failure and an exception is reported to the upper layer.

The successful or unsuccessful lock of a lane is considered the completion of a single lane locking process.

During the initial link establishment (normal link establishment) process, after all RX lanes targeted for locking on the link's receiving end have completed their individual lane locking processes, the RX must notify the TX uniformly via a logical layer management message LLFM. This message shall indicate which lanes have been successfully locked and which have failed. The RX with successfully locked lanes proceeds to the multi-lane alignment (deskew) phase, while those that

failed to lock must be reset (with their corresponding lane state machines returned to the INIT state).

After the TX receives the LLFM, if the RX of all corresponding lanes are successfully locked, the transmitting lane TX enters the multi-lane deskew phase, and the TX of the lane that fails to lock needs to be reset (setting the corresponding lane state machine to the INIT state).

Note: In the process of initial link establishment (normal link establishment), the RX lane to be locked refers to the lane that needs to be established at the beginning of link establishment and successfully completes the lane clock locking and lane equilibrium processes (the lane that fails in lane clock locking and lane equilibrium is not the lane to be locked).

During the initial link establishment (quick link establishment) process and link recovery, if the lane locking fails, the LLFM indicating the lane lock timeout is sent, and the ERRRM is sent to indicate TEE. After receiving the Ack from the peer end, the INIT state is entered. For details, see Section 6.3.6.

6.3.2.5 Multi-lane Alignment

6.3.2.5.1 Process Introduction

In multi-lane alignment, after the link TX completes lane locking, it immediately sends LLCF_PAD and LLCF_DS simultaneously on the link where LLFM indicates that the clock lock is successful. Then, high-rate service data LLBs can be sent on these lanes.

The length of the LLCF_PAD (immediately preceding LLCF_DS) transmitted on different lanes may have a deviation, and the length deviation value is the LaneX_tx_data_dly configured by the upper layer.

In multi-lane alignment, after the RX of the lane to be aligned completes lane locking, LLCF_DS is continuously detected within tWaitDsTimeout.

In the multi-lane alignment stage, there are two types of judgment scenarios for whether the RX has completed lane locking:

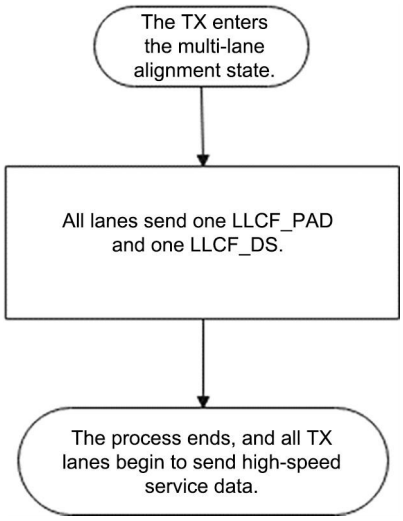
(a) The RX has fed back the LLFM.

(b) The RX to be recovered at the receiving end has detected 16 LLCF_TS2 in the RECOVERY.fast_lock phase.

If LLCF_DS is detected, the RX performs multi-lane alignment based on LLCF_DS and then starts to receive high-rate service data.

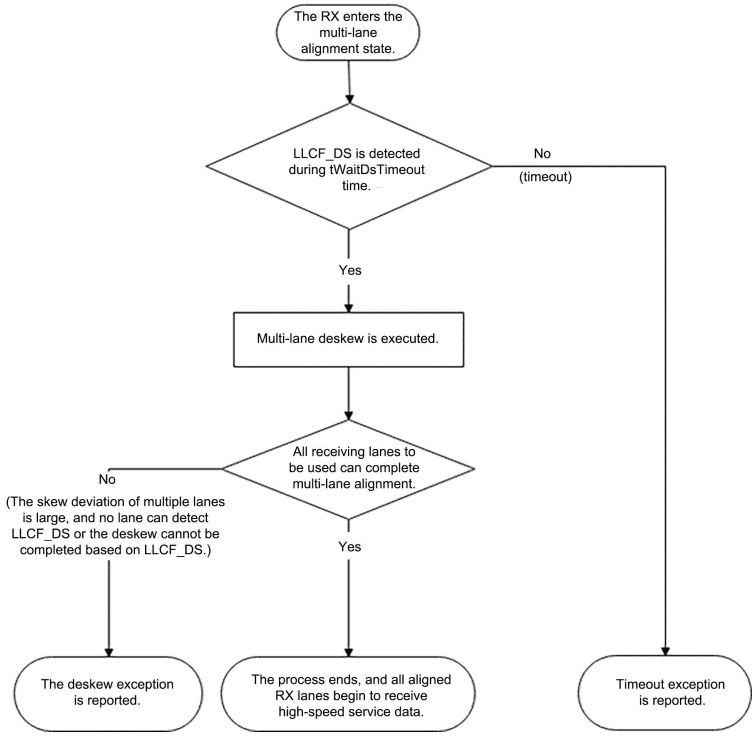
After all TX lanes are locked, they enter the TX lane alignment state. The process of TX performing lane alignment is shown in Figure 34.

Figure 34 Multi-lane alignment process at the TX



After all RX lanes are locked, they enter the RX lane alignment state. The process of RX performing lane alignment is shown in Figure 35.

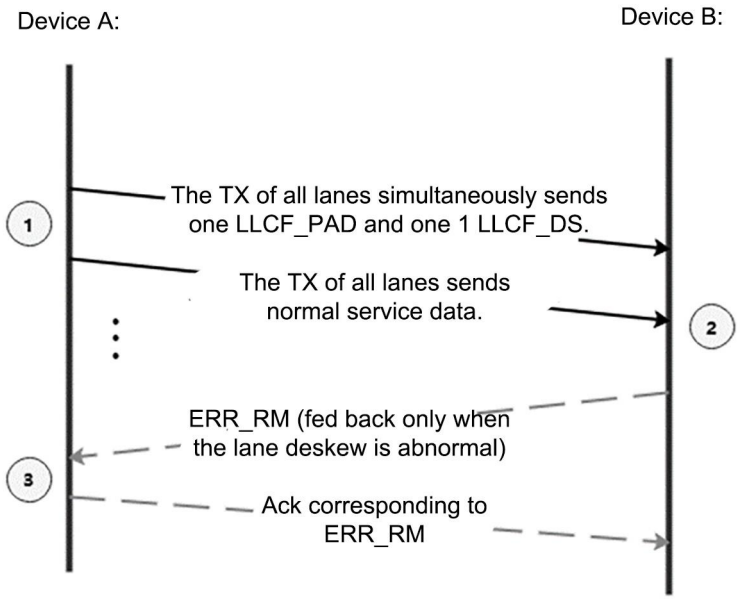
Figure 35 Multi-lane alignment process at the RX



6.3.2.5.2 Exchange Process

Device A and device B are directly connected to each other. Figure 36 shows the exchange information of their connected ports.

Figure 36 Multi-lane alignment exchange in normal mode



The process is detailed as follows:

(1) After the TX of all lanes to be aligned completes lane locking (receives the LLFM), the TX of all lanes simultaneously sends one LLCF_PAD and one LLCF_DS. Then, normal service data is sent.

Note 1: During multi-lane alignment in the link training (link state machine in Training state) and link recovery (link state machine in Recovery state) stages, the length of LLCF_PAD (immediately preceding LLCF_DS) sent by different lanes may deviate. The length deviation value is the LaneX_tx_data_dly configured by the upper layer.

Note 2: During lane direction switching and link width switching (link state machine in HS state), the length of LLCF_PAD (immediately preceding LLCF_DS) sent by different lanes is not allowed to deviate, and the LLCF_PAD sent by all lanes is cf_pad_length.

(2) If device B detects LLCF_DS, the RX performs multi-lane alignment based on LLCF_DS. If an exception in multi-lane alignment is detected (the skew deviation of LLCF_DS between multiple lanes is too large, or some lanes cannot receive LLCF_DS), it will be fed back to device A through ERR_RM that the deskew at the RX of the current link is abnormal, and it will fall back to the initial state of the link and wait for retraining. If no exception in multi-lane alignment is detected, high-rate service data will be received and parsed normally.

(3) If device A receives an exception in the link RX deskew fed back by device B through ERR_RM, it needs to report the exception, fall back to the initial state of the link, and start retraining.

6.3.3 State Management

6.3.3.1 Overview

The ports in this document only support unidirectional links, so the main link of each port contains a transmitting link (the link at the TX of this port) or a receiving link (the link at the RX of this port).

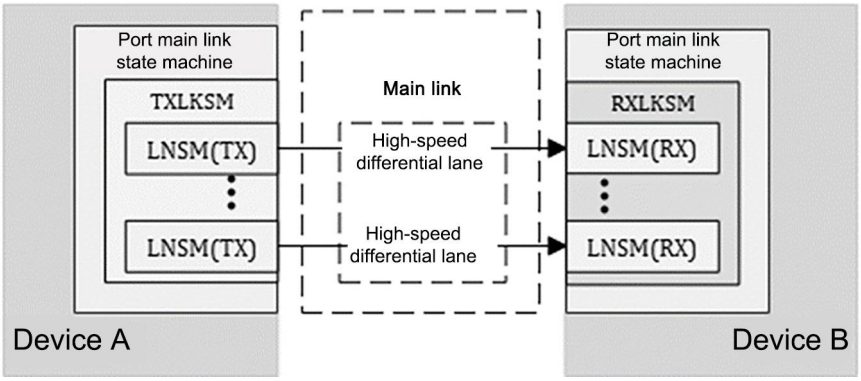
As shown in Figure 37, each port maintains a port main link state machine.

The transmitter link state machine (TXLKSM) manages the state of the port transmitting link. Each port has at most one TXLKSM.

The receiver link state machine (RXLKSM) manages the state of the port receiving link. Each port has at most one RXLKSM.

The lane state machine (LNSM) separately manages the state of one lane of the current port. Based on the lane application mode, LNSM is divided into TX mode and RX mode, corresponding to LNSM (TX) and LNSM (RX) in the figure. LNSM (TX) manages the state of a TX lane on the current port. LNSM (RX manages the state of a RX lane on the current port.

Figure 37 Main link state machine



In the transmission link, the state and behavior of different TX lanes are relatively independent and controlled by their own lane state machines. The rate state of all TX lanes shall be consistent.

In the receiving link, the state and behavior of different RX lanes are relatively independent and controlled by their own lane state machines. The rate state of all RX lanes should be consistent.

The total number of lanes per port is not greater than 8, so the total number of LNSM per port is not greater than 8.

Each port should report the port main link state based on the state of TXLKSM or RXLKSM.

6.3.3.2 Anti-aging Mechanism of Lanes (Informative)

When high-rate lanes are used for a long time, they will face the problem of aging. In order to extend the service life of high-rate lanes, this document provides a reference for implementing the anti-aging mechanism of lanes.

When a decision is made to establish a link lane, the upper layer of the LMP can designate a lane with a specific serial number to establish a link based on the lane usage duration or count to ensure consistent utilization frequency across all lanes. Then, the LMP is notified in the port configuration stage to achieve lane anti-aging.

There are two methods to select a lane:

(a) The random mechanism is used. Each time a link is established, a lane number is randomly selected for link establishment.

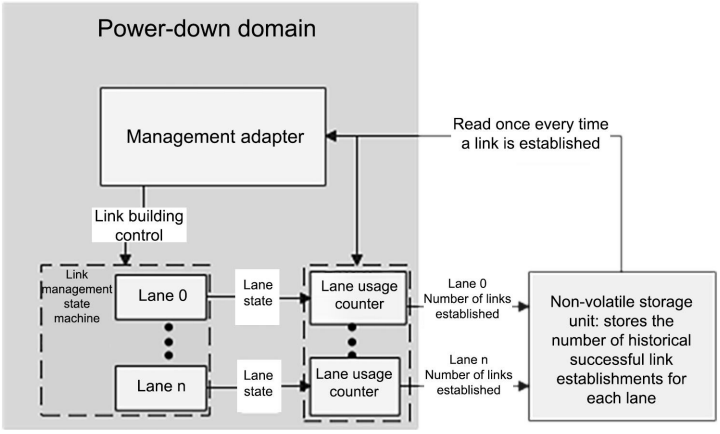
(b) The system provides a non-volatile memory to record the historical usage count of each lane. For details, see the following:

(1) Before a link is established, the management adapter reads the historical cumulative usage counts of all lanes on this end.

(2) Upon link establishment, the port counts lane usage based on state. If a lane is linked for high-rate data transmission, its historical usage count is incremented by one and updated in the system's non-volatile memory.

(3) When selecting lanes, the port should prioritize those with lower historical usage counts as indicated by the lane usage counter while still meeting the required service bandwidth.

Figure 38 Anti-aging mechanism of lanes

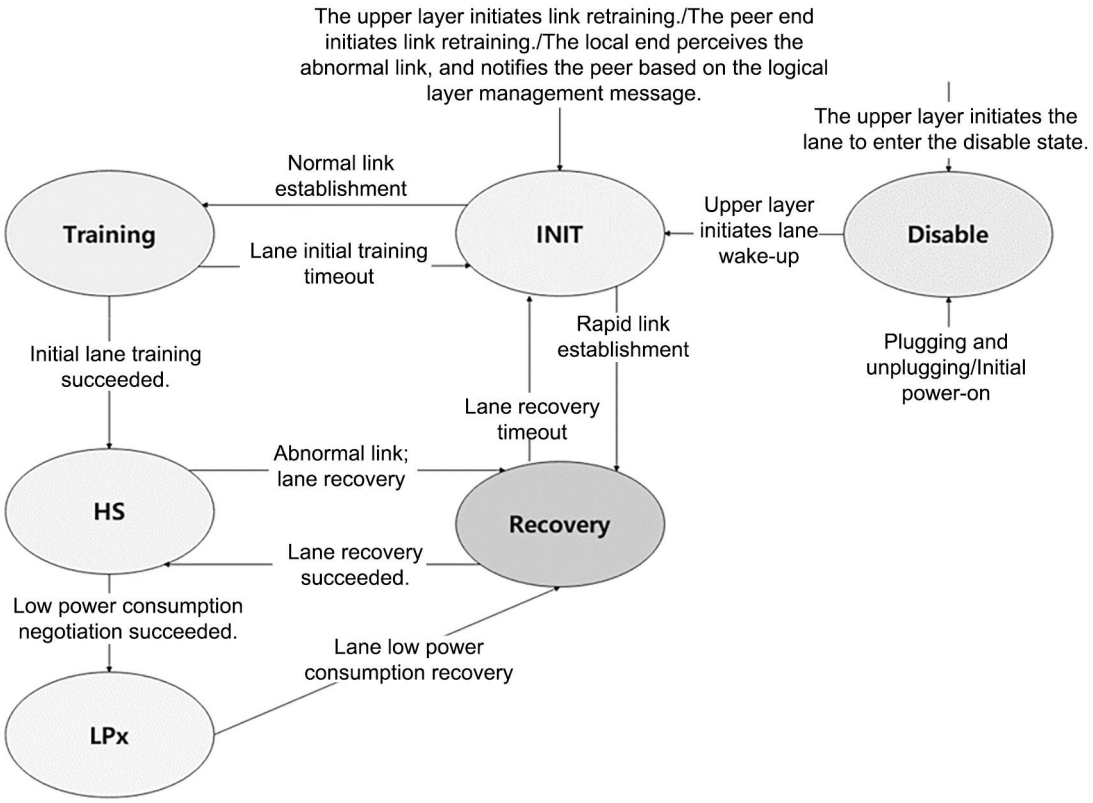


6.3.3.3 Lane State

6.3.3.3.1 State Introduction

The LNSM manages the state of each port lane independently, with each lane operating in either TX or RX mode. The lane state jump relationship is shown in Figure 39.

Figure 39 LNSM state jump relationship



The description of each lane state is shown in Table 54.

Table 54 Description of LNSM state

LNSM State	State Introduction
Disable	Lane invalid state. The upper layer disables the lane, or it is in its initial power-on or disconnected state. The lane is disabled in this state.
INIT	Lane initialization state. In this state, the lane has not yet started initial training. Wait for the entire lane to be initialized.
Training	Lane training state. In this state, the lane is performing lane training, including lane clock lock, lane equilibrium, lane lock, and multi-lane deskew.
HS	Lane service transmission state. In this state, the lane can perform high-rate service data (LLB) transmission.
Recovery	Lane recovery state. In this state, this lane is performing quick lane recovery, including lane equilibrium, lane locking, and multi-lane alignment (deskew).

LNSM State	State Introduction
LPx	Lane low power consumption state. In this state, the lane is in a low power consumption state with different power consumption levels.

In this section, a disabled lane indicates that the lane is in Disable state. The lane to be enabled refers to the TX or RX lane that has been negotiated during the port configuration phase for high-rate data transmission and remains in a Disable state.

6.3.3.3.2 Lane Invalid State

6.3.3.3.2.1 State Behavior

This state is the default state and the lane is disabled. All data paths and control-related logic and states are in the reset state. After the port (See 9.4.5 for details.) completes the capability negotiation, the state machine jumps out of this state and performs lane initialization.

6.3.3.3.2.2 Entry Conditions

If any of the following scenarios occurs, the lane enters the LNSM.Disable state:

- Initial power-on state.
- Port disconnected (unplugged).
- In any state of LNSM, the upper layer configures the disabling indicator lane_disable=1 for this lane.

  Note: The lane disabling indicator lane_disable is configured by the management adapter or software. This document introduces the lane_disable indicator for simplicity.
- The link width adjust message sent or received by this end carries the information that this lane becomes disable_lane after the lane width switching process.
- In LNSM (TX) mode, the lane direction adjust message sent by this end carries the information that this lane is switched from TX to RX lane. After completing the lane direction switching process, LNSM (TX) enters the Disable state.
- In the LNSM (RX) mode, the lane direction adjust message received by this end carries the information that this lane is switched from RX to TX lane. After completing the lane direction switching process, LNSM (RX) enters the Disable state.

6.3.3.3.2.3 Exit Conditions

If any of the following scenarios occurs, the lane will exit the LNSM.Disable state and jump to the LNSM.INIT state:

- The upper layer configures the lane disabling indicator lane_disable=0 for this lane, and initiates lane wake-up.
- The link width adjust message sent or received by this end carries the information that this lane wakes up from the disable_lane state.
- In the LNSM (RX) mode, the lane direction adjust message received by this end carries the information that this lane is switched from RX to TX lane. After completing the lane direction

switching process, LNSM switches to TX mode, and then LNSM (TX) exits the Disable state.

- In LNSM (TX) mode, the lane direction adjust message sent in this section carries a message that the lane is switched from TX to RX. After completing the lane direction switching process, LNSM switches to RX mode, and then LNSM (RX) exits the Disable state.

6.3.3.3.3 Lane Initialization State

6.3.3.3.3.1 State Behavior

This state is the initial state. The lane can be initialized, but it has not yet started to transmit data.

6.3.3.3.3.2 Entry Conditions

If any of the following scenarios occurs, the lane enters LNSM.INIT:

- In any state of LNSM, the upper layer/peer end specifies this link for retraining.
- The conditions for LNSM.Disable to jump to LNSM.INIT are met.

6.3.3.3.3.3 Exit Conditions

Table 55 Description of LNSM.INIT exit conditions

Next State	Jump Conditions
Recovery	The lane initialization is completed (high-rate training data can be sent), and the link training mode of the link corresponding to the lane is quick training (the fast training indicator fast_training of this port is 1).
Training	The lane initialization is completed (high-rate training data can be sent), and the link training mode of the link corresponding to the lane is normal training (the fast training indicator fast_training of this port is 0).
Note 1: At this time, the TX device of the lane needs to send a TSM to the RX device to notify that the initialization of this lane has been completed and start link training. Note 2: The quick training indicator fast_training=1 is used to indicate that on this link, both the local and peer RX have obtained the negotiated rate based on capability negotiation (including port rate, Cableinfo, and other exchanges), and a set of reliable receiving lane equilibrium parameters have been obtained based on the negotiated rate. If both ports have agreed on the current rate without performing clock locking and lane equilibrium, the fast_training value is specified after port negotiation. If the quick training fails, execute the link training process in 6.3.2.	

6.3.3.3.4 Lane Training State

6.3.3.3.4.1 State Behavior

The TX and RX of the lane enter the Training state after training initialization. In the Training state, the TX lane is enabled and can continuously send CFs required for training; the RX lane is enabled to support high-rate lane training.

6.3.3.3.4.2 TX Mode Training Sub-state

Figure 40 LNSM (TX) training sub-state transition relationship

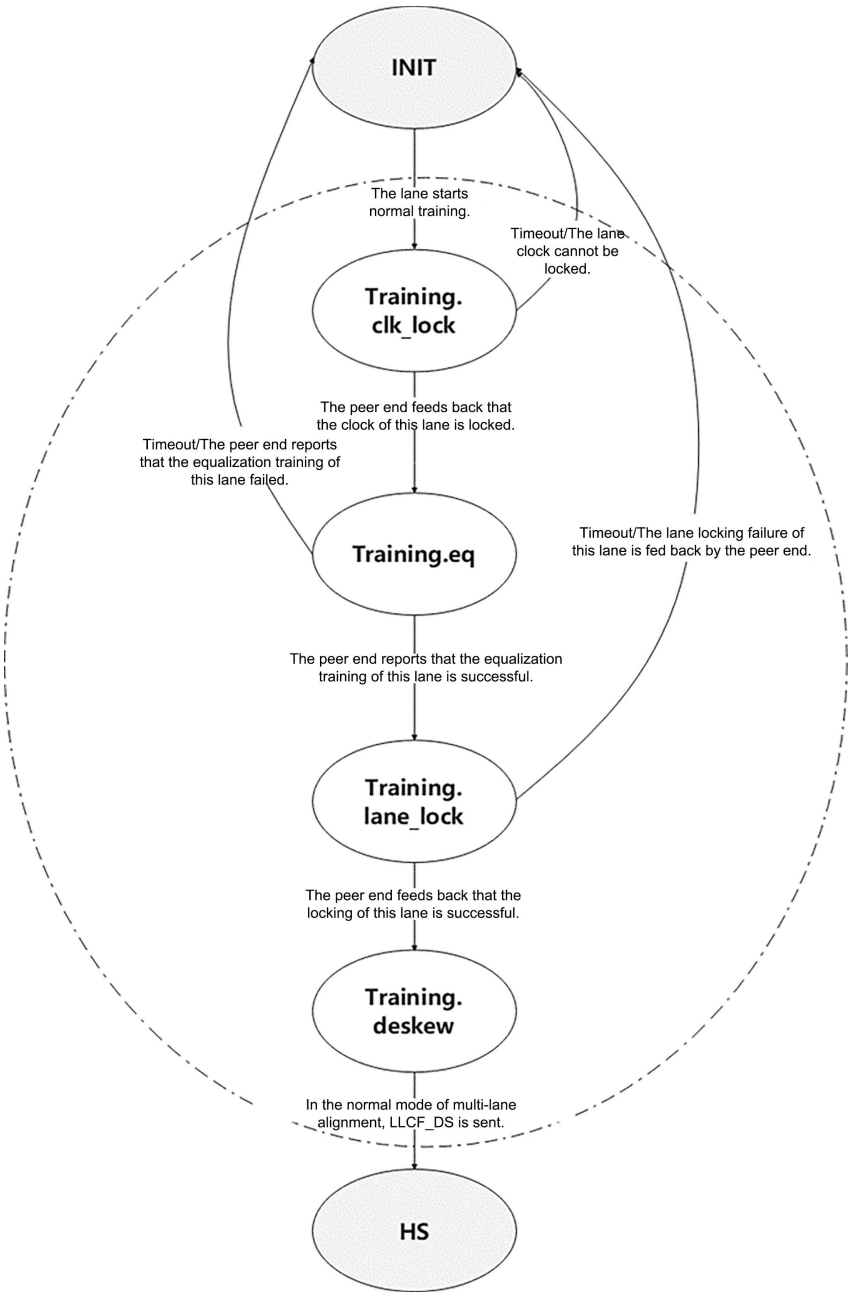


Table 56 Conditions for exiting LNSM (TX) training

Training Sub-state	State Description	Exit Conditions
clk_lock	This state is used to wait for the lane to complete the clock lock; in this state, the lane TX	● After the local end receives the CLFM indication that the RX of this lane has completed the clock lock and completes the following process, the LNSM (TX) of this lane jumps to the Training.eq state.

Training Sub-state	State Description	Exit Conditions
	continuously sends LLCF_TS0 back-to-back.	● This lane performs pattern switching on the clock-locked lane and continuously transmits LLCF_TS1 after switching. ● The local end feeds back the Ack corresponding to the CLFM to the peer end. ● After the local end receives a CLFM indicating that the clock lock of the lane RX has failed, and feeds back the Ack corresponding to the CLFM to the peer end, the LNSM (TX) of this lane jumps to the INIT state. ● The local tTxLinkTrainTimeout times out and jumps to the INIT state.
eq	This state is used to wait for the lane to complete equilibrium; in this state, the lane TX continuously sends LLCF_TS1 back-to-back.	● After the local end receives the EQFM indicating that the RX lane equilibrium of this lane is successful and completes the following process, the LNSM (TX) of this lane jumps to the Training.lane_lock state. ● This lane performs pattern switching on the lane that has successfully equalized and continuously transmits LLCF_TS2 after switching. ● The local end feeds back the Ack corresponding to the EQFM to the peer end. ● After the local end receives an EQFM indicating that the equilibrium of the lane RX has failed, and feeds back the Ack corresponding to the EQFM to the peer end, the LNSM (TX) of this lane jumps to the INIT state. ● The local tTxLinkTrainTimeout times out and jumps to the INIT state.
lane_lock	This state is used to wait for the lane to complete lock; in this state, the lane TX continuously sends LLCF_TS2 back-to-back.	● The local end receives an LLFM indicating that the RX lane of this lane is successfully locked, and the LNSM (TX) of this lane jumps to the Training.deskew state. ● The local end receives an LLFM indicating that the RX lane of this lane has failed to lock, and the LNSM (TX) of this lane jumps to the INIT state. ● The local tTxLinkTrainTimeout times out and jumps to the INIT state.
deskew	This state is used to complete multi-lane alignment; in this state, the TX of the lane sends LLCF_DS.	In normal alignment mode, after receiving the LLFM, the local end initiates a multi-lane deskew. This lane TX sends one LLCF_PAD and one LLCF_DS, and the LNSM (TX) of this lane jumps to the HS state.

6.3.3.3.4.3 RX Mode Sub-state Description and Exit Conditions

Figure 41 LNSM (RX) training sub-state transition relationship

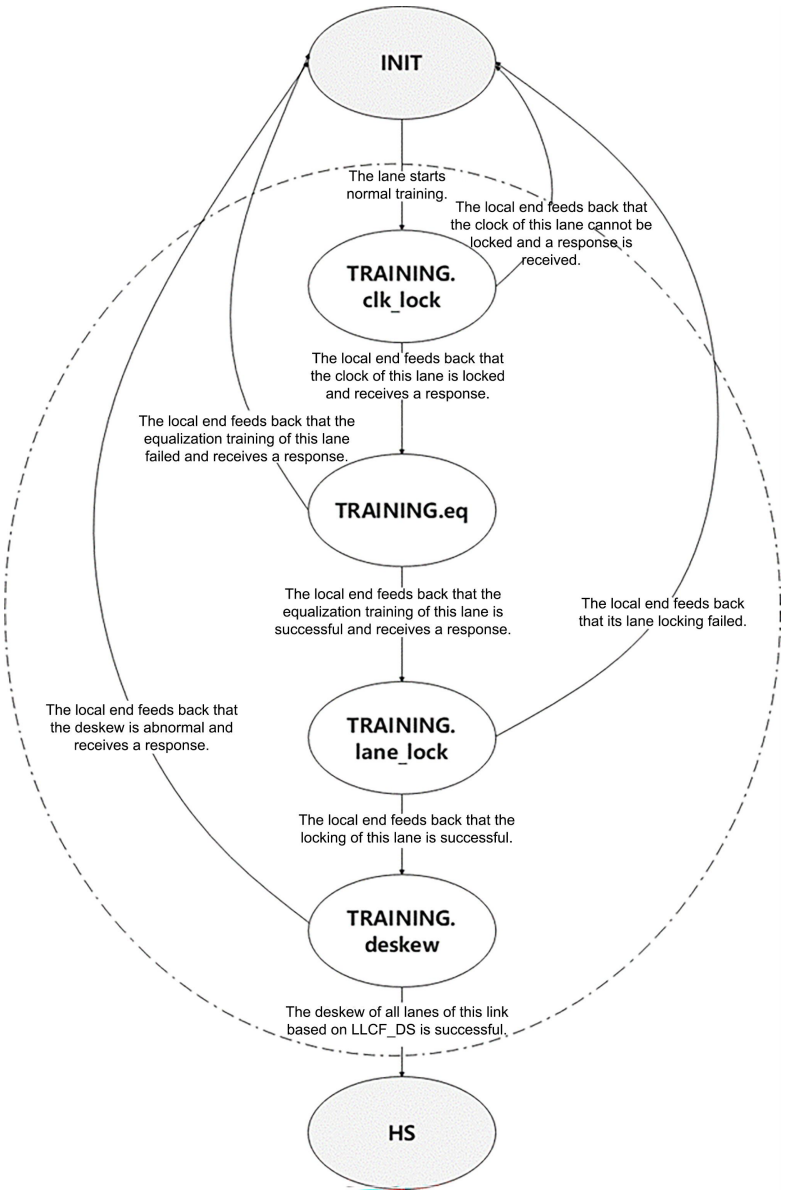


Table 57 Description of conditions for exiting LNSM (RX).training

Training Sub-state	State Description	Exit Conditions
clk_lock	Used to wait for the lane to complete the lane clock lock.	• After the RX clock of this lane is successfully locked, the local end sends a CLFM indicating that the RX lane clock of this lane is successfully locked. After receiving the Ack feedback from the peer end, the LNSM (RX) of this lane jumps to the LNSM.Training.eq state. • After the RX clock lock of this lane fails, the local end sends a CLFM indicating that the RX lane clock lock of this lane has failed, and receives an Ack feedback from the

Training Sub-state	State Description	Exit Conditions
		peer end. The LNSM (RX) of this lane jumps to the LNSM.INIT state.
eq	Used to wait for the lane to complete lane equilibrium.	• After the RX equilibrium training of this lane is successful, the local end sends an EQFM indicating that the RX equilibrium training of this lane is successful. After receiving the Ack fed back by the peer end, the LNSM (RX) of this lane jumps to the LNSM.Training.lane_lock state. • After the RX equilibrium training of this lane fails, the local end sends an EQFM indicating that the RX equilibrium training of this lane has failed. After receiving the Ack fed back by the peer end, the LNSM (RX) of this lane jumps to the LNSM.INIT state.
lane_lock	Used to wait for the lane to complete lane lock.	• After the RX lane of this lane is successfully locked, the local end sends an LLFM indicating that the RX lane of this lane is successfully locked, and the LNSM (RX) of this lane jumps to the LNSM.Training.deskew state. • After the RX lane lock of this lane fails, the local end sends an LLFM indicating that the RX lane multi-locking of this lane has failed, and the LNSM (RX) of this lane jumps to the LNSM.INIT state.
deskew	Used to complete multi-lane alignment.	• If the RX of this link succeeds based on LLCF_DS deskew, the LNSM (RX) of this lane jumps to the LNSM.HS state. • If the RX of this link fails based on LLCF_DS deskew, the local end feeds back the lane deskew error based on the ERR_RM message and receives a response to the corresponding ERR_RM message. Then, the LNSM (RX) of this lane jumps to the LNSM.INIT state. • If the local end tWaitDsTimeout times out, the local end feeds back a deskew timeout error based on the ERR_RM message and receives a response to the corresponding ERR_RM message. Then, the LNSM (RX) of this lane jumps to the LNSM.INIT state.

6.3.3.3.5 Lane Service Transmission State

6.3.3.3.5.1 State Behavior

After the TX and RX of the lane complete training, the lane enters the HS state.

In the HS state, the TX lane is enabled and can continuously send high-rate service data (LLBs).

In the HS state, the RX lane is enabled and can continuously receive and parse high-rate service data.

6.3.3.3.5.2 Conditions for TX Mode Exiting the HS State

After the local end receives the ERR_RM message that the link needs to be restored (see the exception handling mechanism for details), the TX lane of the local end stops sending LLBs and continuously sends LLCF_TS2 instead. After feeding back the Ack corresponding to ERR_RM, if the fast training indication of this link fast_recovery = 1, LNSM (TX) jumps to the Recovery.fast_lock state.

Note: The fast recovery indication fast_recovery = 1 is used to indicate whether the RX supports fast recovery on this link. The value 0 indicates that fast recovery is not supported; the value 1 indicates that fast recovery is supported. This capability is specified in the port capability negotiation.

After the local end receives the ERR_RM message that the link needs to be restored (see the exception handling mechanism for details), the TX lane of the local end stops sending LLBs and continuously sends LLCF_TS2 instead. After feeding back the Ack corresponding to ERR_RM, if the fast training indication of this link fast_recovery = 0, LNSM (TX) jumps to the Recovery.re_lock state.

After the local end receives the ERR_RM message that requires link retraining and feeds back the corresponding Ack, LNSM (TX) jumps to the INIT state.

After TXLKSM initiates the control of LNSM (TX) entering LPX in this lane (see the jump instructions for TXLKSM.HS entering low power consumption for details), this lane sends LLCF_EI, and LNSM (TX) jumps to LPx. The specific LPx sub-state that LNSM (TX) enters is described as follows:

- If the target low power consumption state for this lane is LP0f, jump to LPx.LP0f.
- If the target low power consumption state for this lane is LP0, jump to LPx.LP0.
- If the target low power consumption state for this lane is LP1, jump to LPx.LP1.
- If the target low power consumption state for this lane is LP2, jump to LPx.LP2.
- If the target low power consumption state for this lane is LP3, jump to LPx.LP3.

After the local end transmits the link width adjust message LWAM and receives the corresponding Ack, if this link width adjustment handshake requires a low power consumption entry request for this lane, and after this lane has finished sending LLCF_EI, the LNSM (TX) transitions to the state specified by LP_MODE in the LWAM message. After the local end transmits the lane direction adjust message LDAM and receives the corresponding Ack, if this lane direction adjust handshake requires the TX of this lane to switch to RX, and after this lane has finished sending LLCF_EI, the LNSM (TX) transitions to the Disabled state. Subsequently, the LNSM (TX) corresponding to this lane is deprecated, and its state is managed by LNSM (RX).

6.3.3.3.5.3 Conditions for RX Mode Exiting the HS State

If the local end detects an exception in the receiving link and determines that link recovery is required, it will send an ERR_RM message (see exception handling mechanism for details) indicating the need for link recovery. After receiving the corresponding Ack from the peer end, if the fast_recovery indicator is set to 1, the LNSM (RX) will transition to the Recovery.Fast_lock state.

If the local end detects an exception in the receiving link and determines that link recovery is required, it will send an ERR_RM message (see exception handling mechanism for details) indicating the need for link recovery. After receiving the corresponding Ack from the peer end, if the fast_recovery indicator is set to 0, the LNSM (RX) will transition to the Recovery.re_lock state.

If the local end detects an exception in the receiving link and determines that link retraining is required, it will send an ERR_RM message (see exception handling mechanism for details) indicating the need for link retraining. After receiving the corresponding Ack from the peer end, the LNSM (RX) transitions to the INIT state.

RXLKSM initiates the control of LNSM (RX) entering LPx in this lane (see the following RXLKSM.HS transition description), and after the lane receives LLCF_EI, LNSM (RX) jumps to LPx. The specific LPx sub-state that LNSM (RX) enters is described as follows:

- If the target low power consumption state for this lane is LP0f, jump to LPx.LP0f.
- If the target low power consumption state for this lane is LP0, jump to LPx.LP0.
- If the target low power consumption state for this lane is LP1, jump to LPx.LP1.
- If the target low power consumption state for this lane is LP2, jump to LPx.LP2.
- If the target low power consumption state for this lane is LP3, jump to LPx.LP3.

After the local end receives the link width adjust message LWAM and feeds back the corresponding Ack, if this link width adjustment handshake requires a low power consumption entry request for this lane, and after this lane has detected LLCF_EI, the LNSM (RX) transitions to the state specified by LP_MODE in the LWAM message.

After the local end receives the lane direction adjust message LDAM and feeds back a corresponding Ack, if this lane direction adjustment handshake requires the RX of this lane to switch to TX, and upon detection of LLCF_EI by this lane, the LNSM (RX) transitions to the Disabled state. Subsequently, the LNSM (RX) corresponding to this lane is deprecated, and its state management is transferred to LNSM (TX).

6.3.3.3.6 Lane Recovery State

6.3.3.3.6.1 State Behavior

When the link is recovered after an exception and the lane wakes up from low power consumption, the lane enters the Recovery state.

When the link enters this state, all TX data path functions, including FEC coding, lane distribution, scrambling, precoding, multiplexing and other functions are reset, and all data lanes send LLCF_TS2 at the same time for link recovery (removing the multi-lane skew that previously used different LLCF_PAD lengths).

When the link enters this state, all RX data path functions, including de-precoding, descrambling, lane combination, FEC decoding and error correction, multiplexing decoding, deskew, and other functions, are reset.

The TX lane in the Recovery state is enabled, and continuously sends CFs for lane equilibrium, lane lock, and multi-lane deskew.

The RX lane in the Recovery state is enabled, and continuously receives and detects CFs for lane equilibrium, lane lock, and multi-lane deskew.

6.3.3.3.6.2 TX Mode Sub-state Description and Exit Conditions

Figure 42 LNSM (TX).Recovery sub-state transition relationship

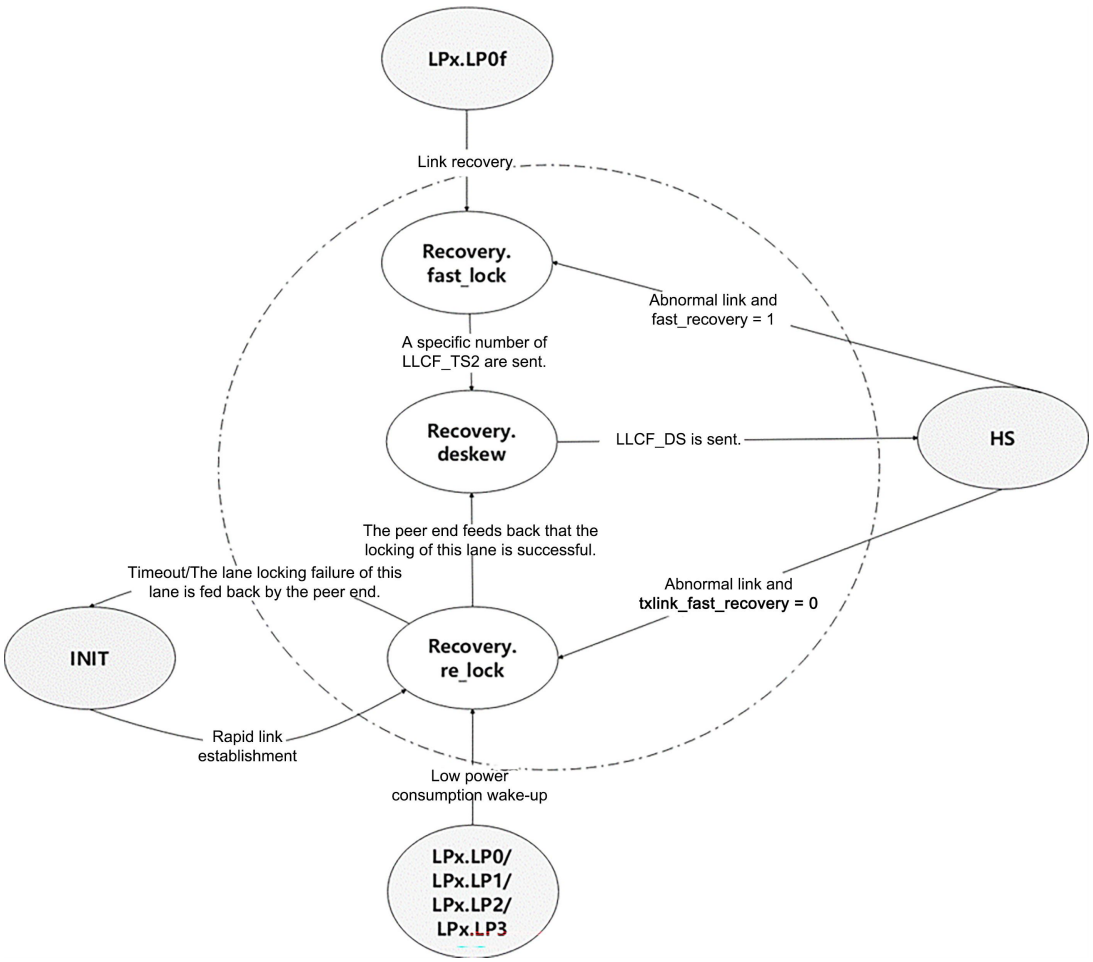


Table 58 LNSM (TX).Recovery exit conditions

Recovery Sub-state	State Description	Exit Conditions
fast_lock	This state is used to wait for the lane to complete lock; in this state, the lane TX continuously sends a specific number of LLCF_TS2 back-to-back.	After this lane sends a specific number of LLCF_TS2, the LNSM (TX) of this lane jumps to the Recovery.deskew state. [a]
re_lock	This state is used to wait for the lane to complete lock; in this state, the lane TX continuously sends LLCF_TS2 back-to-back.	● After the local end receives the LLFM indicating that all RX lanes of this link have been successfully locked, the LNSM (TX) of this lane jumps to the Recovery.deskew state. ● After the local end receives an ERR_RM indicating that not all RX lanes of this link have been successfully locked, and feeds

Recovery Sub-state	State Description	Exit Conditions
		back the Ack corresponding to the ERR_RM to the peer end, the LNSM (TX) of this lane jumps to the INIT state. ● The local tLinkRecoveryTimeout times out and jumps to the INIT state.
deskew	This state is used to complete multi-lane alignment; in this state, the TX of the lane sends LLCF_DS.	After the local end receives the LLFM indicating that all RX lanes of this link are successfully locked, the TX lane sends one LLCF_PAD and one LLCF_DS, the LNSM (TX) of this lane jumps to the HS state.
[a] The number of LLCF_TS2 is negotiated by the management adapter capability.		

6.3.3.3.6.3 RX Mode Sub-state Description and Exit Conditions

Figure 43 LNSM (RX).Recovery sub-state transition relationship

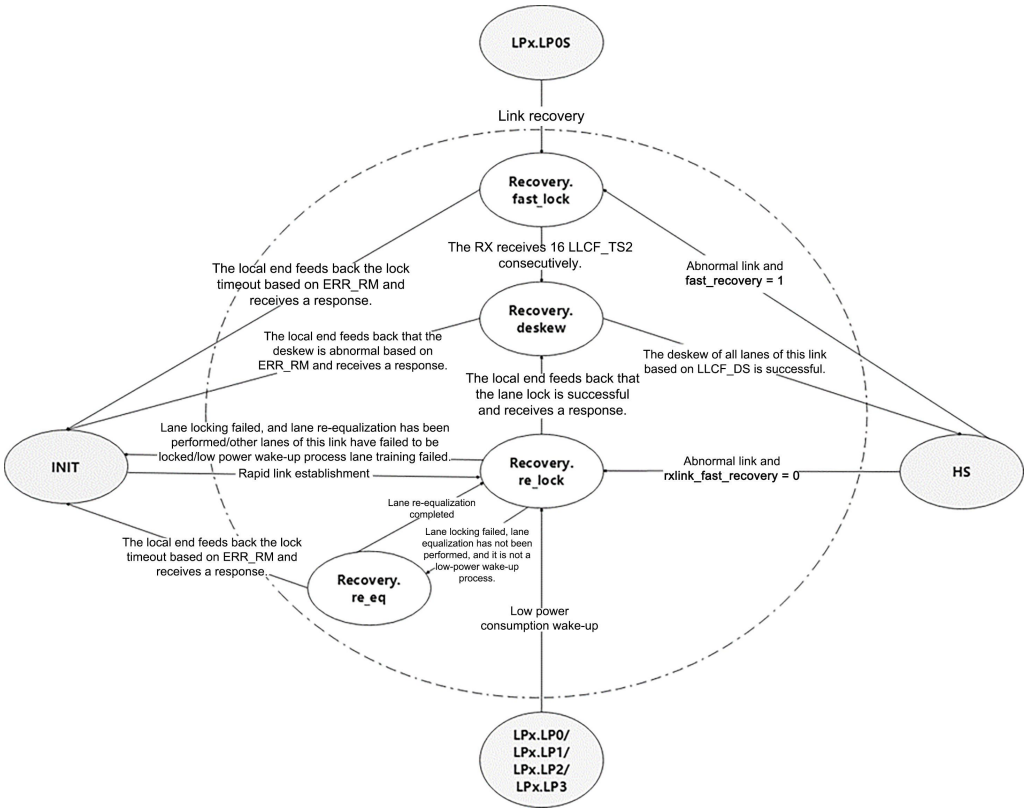


Table 59 LNSM (RX).Recovery exit conditions

Recovery Sub-state	State Description	Exit Conditions

Recovery Sub-state	State Description	Exit Conditions
fast_lock	Used to wait for the lane to complete lane lock.	● After the RX of this lane receives and detects a specified number (not less than 16) of LLCF_TS2 in succession, the LNSM (RX) of this lane jumps to the LNSM.Recovery.deskew state. ● The local tFastLockTimeout timed out. After the local end sends an ERR_RM (TTE) indicating that the clock lock of the RX lane of this link has timed out and receives the Ack from the peer end, the LNSM (RX) of this lane jumps to the LNSM.INIT state.
re_lock	Used to wait for the lane to complete lane lock.	● If the following conditions are met at the same time, the LNSM (RX) of this lane jumps to the LNSM.Recovery.re_eq state. ● This end supports re-equilibrium during the link recovery phase (depending on the design specifications of each vendor for the electrical layer of the RX, this document does not impose any mandatory restrictions). ● The current LNSM is not awakened to Recovery from the LPx state. ● Note: If Recovery is performed during the LPx wake-up phase, it will cause excessive delay when the link wakes up from low power consumption. ● The RX end of this lane cannot be successfully locked within the tRecReLock time. ● This lane has not been re-equalized during this link recovery. ● If the RX end of this lane cannot be successfully locked within the tRecReLock time and does not meet the conditions for LNSM (RX) to jump to the Recovery.re_eq state, then after this end sends an ERR_RM (TTE) indicating that the RX lane lock of this link has timed out and receives the Ack fed back by the other end, the LNSM (RX) of this lane jumps to the LNSM.INIT state. ● After the RX lane of this lane is successfully locked, and all the RX lanes to be recovered at this end are successfully locked, this end sends an LLFM indicating that the RX lane of this lane is successfully locked, and the LNSM (RX) of this lane jumps to the LNSM.Recovery.deskew state. ● If the RX end of this lane is successfully locked in the lane, but any other RX lane recovery times out (at any phase of Recovery.relock or Recovery.re_eq), then after this end sends an ERR_RM indicating that the RX lane lock of this link has timed out and receives the Ack fed back by the peer end, the LNSM (RX) of this lane jumps to the LNSM.INIT state.

Recovery Sub-state	State Description	Exit Conditions
re_eq	Used to wait for the lane to complete re-equilibrium.	● The RX end of this lane completes lane re-equilibrium within the tRecReEqTimeout time, and the LNSM (RX) of this lane jumps to the LNSM.Recovery.relock state. ● The local tRecReEqTimeout timed out. After the local end sends an ERR_RM (TTE) indicating that the lock of the RX lane of this link has timed out and receives the Ack from the peer end, the LNSM (RX) of this lane jumps to the LNSM.INIT state.
deskew	Used to complete multi-lane alignment.	● If the RX of this link succeeds based on LLCF_DS deskew, the LNSM (RX) of this lane jumps to the LNSM.HS state. ● If the RX of this link fails based on LLCF_DS deskew, the local end feeds back the lane deskew error based on the ERR_RM message and receives a response to the corresponding ERR_RM message. Then, the LNSM (RX) of this lane jumps to the INIT state. ● If the local end tWaitDsTimeout times out, the local end feeds back a deskew timeout error based on the ERR_RM message and receives a response to the corresponding ERR_RM message. Then, the LNSM (RX) of this lane jumps to the INIT state.

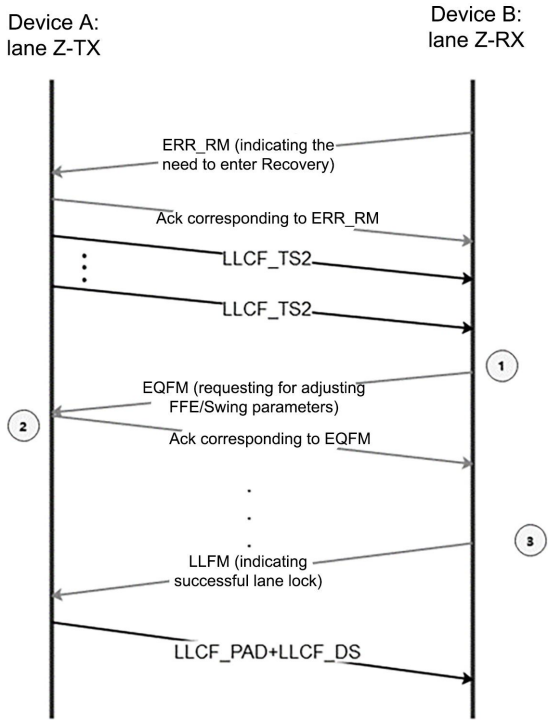
During lane re-equilibrium, the RX can readjust the lane parameters in the following two ways:

(a) The RX re-executes the receiver equilibrium adaptive algorithm to adjust the local RX lane, and no longer requests the peer TX to adjust FFE and Swing parameters through EQFM (the adaptive algorithm is related to the vendors' implementation).

(b) The RX uses a similar method to lane equilibrium in the training phase to evaluate whether the equilibrium training results meet expectations. If not, it requests the peer end TX to adjust FFE and Swing parameters based on EQFM, and then continues to evaluate whether the equilibrium training results meet expectations. If the evaluation results of equilibrium training meet expectations, it is considered that lane re-equilibrium has been completed.

- At this stage, the EQFM is only used for the RX to request adjustment of TX's FFE and Swing parameters without feedback on equilibrium success or failure. Therefore, the EQD field in the EQFM fed back by the RX is invalid, and the Swing_Req and FFE_Req fields are valid, that is, the scenario where the Swing_Req_Enable field is 0b and the FFE_Req field is 7Fh is not allowed.
- At this stage, if the TX receives the EQFM sent by the RX requesting adjustment of TX's FFE and Swing parameters, it also needs to feed back the corresponding EQFM_Ack. Figure 44 shows the port exchange using this method to adjust lane parameters.

Figure 44 Lane parameter adjustment for lane re-equilibrium



The process is detailed as follows:

(1) After device B sends ERR_RM to instruct device A to start link recovery and receives the response corresponding to ERRRM, it performs lane locking based on LLCF_TS2 sent by device A. After the lane locking fails, lane re-equilibrium begins.

Device B evaluates whether the re-equilibrium training meets expectations during the process. If not, it requests lane Z through EQFM and the TX switches to new FFE/Swing parameters.

(2) After receiving the EQFM of the corresponding lane Z switching FFE/Swing parameters, device A executes it in the following order:

Step 1: Adjust the FFE/Swing parameters of the TX of lane Z based on the message indication of the EQFM.

Step 2: Device A sends an Ack corresponding to the EQFM to notify device B that the TX of lane Z has adjusted the FFE/Swing parameters.

(3) After receiving the Ack that the corresponding lane Z has adjusted the FFE parameters, device B restarts to equilibrium and evaluates whether the equilibrium training meets the expectations. If not, it requests new parameters through the EQFM (same as process 1). If it meets the expectations, it continues to execute lane locking and informs device A through LLFM that the clock lock of lane Z is completed.

6.3.3.3.7 Lane Low Power Consumption State

6.3.3.3.7.1 Power Domain

Table 60 Power domain

Power Domain	Function
VCC power	Main service function.
VSL power	Management logic of VCC power; Sideband link related functions; Physical layer calibration, equilibrium, and other link training parameter storage register; Other special registers related to power management and fast wake-up of LP3.

6.3.3.3.7.2 Lane Power Consumption State

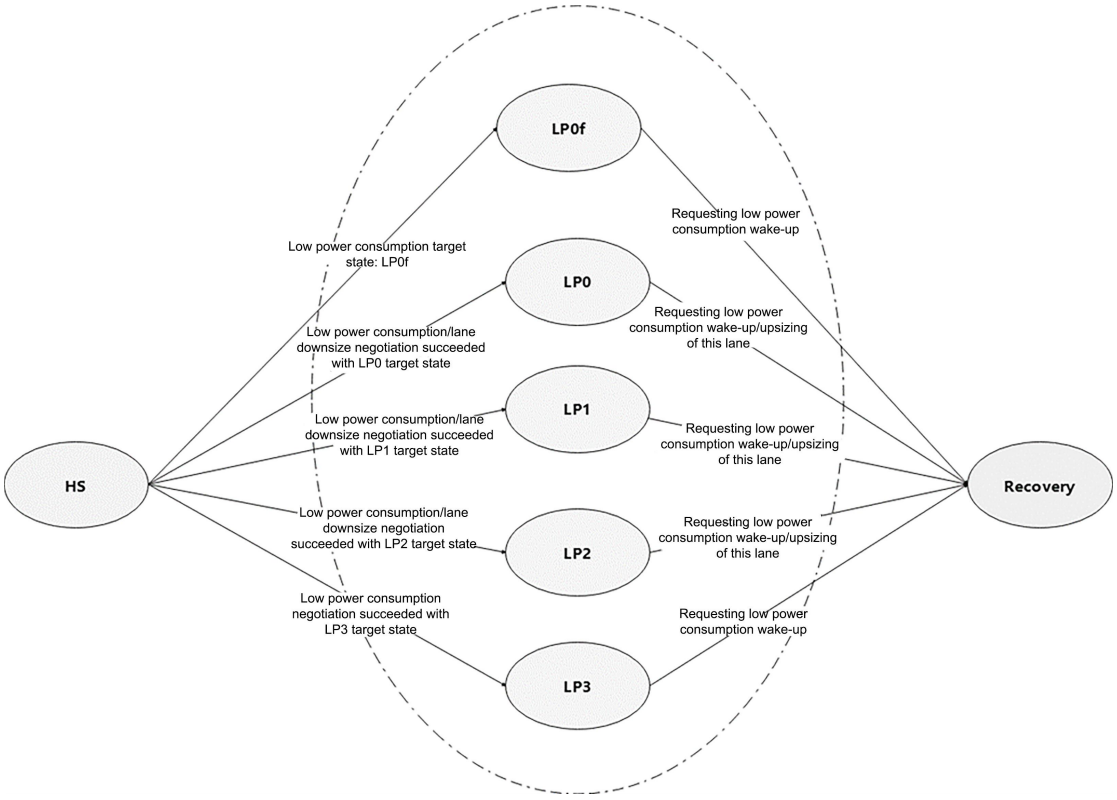
Table 61 Power domain

Power Consumption Level	Power0	Power1	Power2	Power3	Power4
Corresponding lane state machine state	HS	LPx.LP0f, LPx.LP0	LPx.LP1	LPx.LP2	LPx.LP3, Disable
High-rate service transmission	ON	OFF	OFF	OFF	OFF
State Introduction	High-rate service transmission state.	Low power consumption state; it can be quickly awakened.	Low power consumption state; PLL can be turned off, Slow wake-up.	Low power consumption state; PLL can be turned off, Slow wake-up. Note: Compared with power2, this power consumption state's more functions can be turned off at the analog TX and RX to save power.	Low power consumption state; All functions except the sideband link can be turned off, and the VCC power supply can be turned off.

Power Consumption Level	Power0	Power1	Power2	Power3	Power4
RX to wake-up ability based on LLCF_EIE	NA	ON	ON	ON	OFF
VCC power state	ON	ON	ON	ON	ON/OFF

6.3.3.3.7.3 LNSM.LPx Sub-state and Exit Conditions

Figure 45 LNSM LPx sub-state transition relationship



TX mode sub-state description and exit conditions are shown in Table 62.

Table 62 Description of LNSM (TX).LPx exit conditions

LPx Sub-state	Exit Conditions
LP0f	● If the TXLKSM initiates the control of LNSM (TX) of this lane to exit LPx, LNSM (TX) of this lane jumps to Recovery.fast_lock state after N_EIE LLCF_EIE is sent by this lane.
LP0–LP2	● If the TXLKSM initiates the control of LNSM (TX) of this lane to exit LPx, LNSM

LPx Sub-state	Exit Conditions
	(TX) of this lane jumps to Recovery.re_lock state after N_EIE LLCF_EIE is sent by this lane. ● The local end has sent the link width adjust request LWAM and received the Ack feedback from the peer end. If a wake-up request for this lane is included in the LWAM, the LNSM (TX) in this lane transitions to the Recovery.re_lock state after N_LLCF_EIE is sent by this lane.
LP3	● If the TXLKSM initiates the control of LNSM (TX) of this lane to exit LPx, LNSM (TX) of this lane jumps to Recovery.re_lock state after TSM is sent by this lane.

The RX mode sub-state description and exit conditions are shown in Table 63.

Table 63 Conditions for LNSM (RX) to exit LPx

LPx Sub-state	Exit Conditions
LP0f	● After RXLKSM initiates the control to make LNSM (RX) in this lane exit LPX, LNSM (RX) in this lane transitions to the Recovery.fast_lock state.
LP0–LP2	● After RXLKSM initiates the control to make LNSM (RX) in this lane exit LPX, LNSM (RX) in this lane transitions to the Recovery.re_lock state. (For wake-up in link low power scenarios) ● The local end receives the LWAM transmitted by the peer end and feeds back an Ack message. If a wake-up request for this lane is included in the LWAM, the LNSM (RX) in this lane transitions to the Recovery.re_lock state. (For wake-up in link width adjustment scenarios)
LP3	● After RXLKSM initiates the control to make LNSM (RX) in this lane exit LPX, LNSM (RX) in this lane transitions to the Recovery.re_lock state. (For wake-up in link width adjustment scenarios and wake-up in link low power scenarios)

6.3.3.4 TX Link State

6.3.3.4.1 State Introduction

A TXLKSM manages the TX link state of the entire port. The state transition relationship and its introduction are shown in Figure 46.

Figure 46 TXLKSM state transition relationship

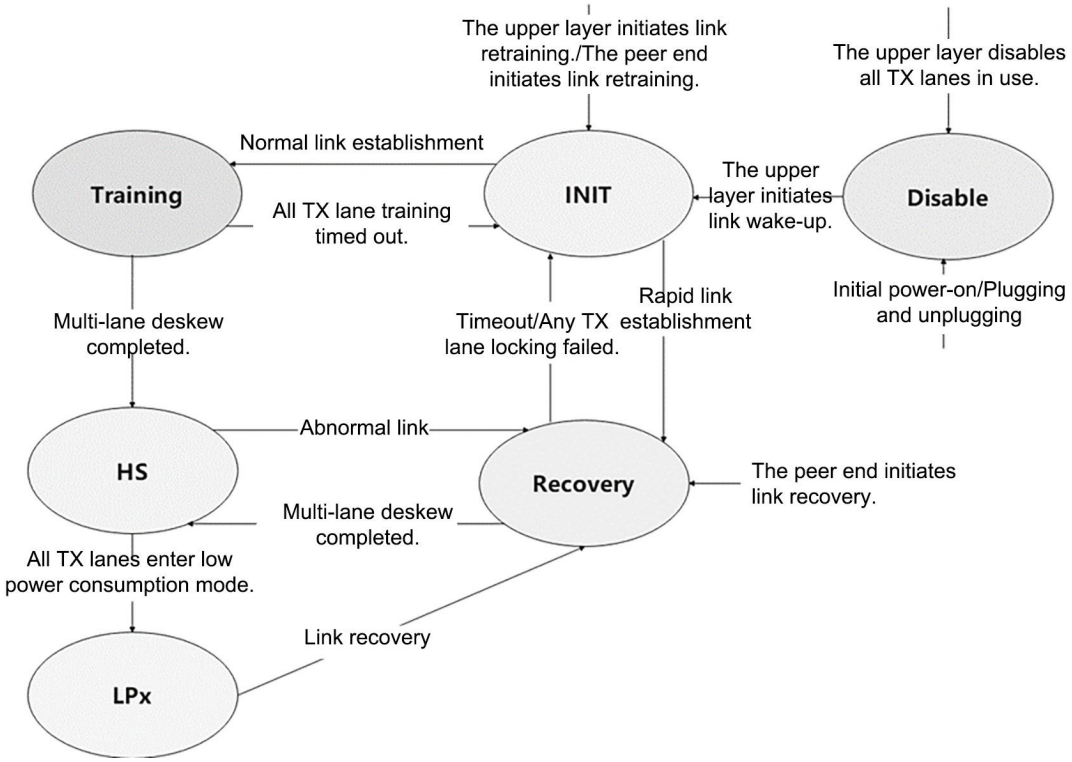


Table 64 TXLKSM states

TXLKSM State	State Introduction
Disable	Link invalid state. The upper layer disables all TX lanes, or it is in initial power-on or disconnected state. In this state, the TX link has not started initial training.
INIT	Link initialization state. This state corresponds to the port configuration state of the port state machine. In this state, after the port capability negotiation is completed, all TX lanes at the local end perform electrical layer initialization based on the link configuration results. Link exceptions may lead to this state.
Training	Link training state. In this state, TX lanes to be enabled at the local end perform initial lane training.
HS	Link service transmission state. In this state, successfully trained TX lanes can perform high-speed service data transmission together.
Recovery	Link recovery state. In this state, all TX lanes to be enabled at the local end perform lane

TXLKSM State	State Introduction
	recovery.
LPx	Link low power state. No high-speed service is transmitting, and the TX lanes to be enabled at the local end are all in the low power state.

The TXLKSM and the LNSM (TX) are in an interdependent master-slave relationship. The TXLKSM completes multi-lane TX collaborative link operations, and LNSM (TX) completes independent single lane TX operations.

When the TXLKSM is in Training, HS, Recovery, and LPx states, the TXLKSM layer initiates link-level operations (entering/exiting low power, training, recovery, or multi-lane deskew) according to the state of the entire link, and controls LNSM (TX) to perform different operations.

6.3.3.4.2 Link Invalid State

6.3.3.4.2.1 State Behavior

This state is the default state, and the main TX link of this port has not been enabled. All data paths and control-related logic and states of the main TX link are in the reset state.

After the port (see the management adapter description for details) completes the port capability negotiation, the TXLKSM exits this state and performs port configuration and high-speed link initialization.

In this state, LNSMs of all TX lanes at the local end are in the LNSM.Disable state.

6.3.3.4.2.2 Entry Conditions

If any of the following scenarios occurs, this link enters the TXLKSM.Disable state:

- Initial power-on state.
- Port disconnected.
- The upper layer configures the lane_disable indicator of all TX lanes at the local end to 1.

6.3.3.4.2.3 Exit Conditions

If the upper layer configures the lane_disable indicator of any TX lane at the local end to 0, the TXLKSM transitions from the TXLKSM.Disable state to the TXLKSM.INIT state.

6.3.3.4.3 Link Initialization State

6.3.3.4.3.1 State Behavior

This state is the initial state, and the main TX link of this port has not started to transmit data. All data paths and control-related logic and states of the main TX link are in the initial state.

After the port completes the port capability negotiation, the TX link at the local end performs link initialization based on the negotiated rate. After all TX lanes to be enabled are initialized, the TXLKSM exits this state and starts the link establishment process.

In this state, LNSMs of TX lanes at the local end can be in the following states:

- LNSM.INIT for the LNSMs of the TX lanes which are not fully initialized or have training failures
- LNSM.Disable for the LNSMs of the TX lanes which are not enabled for initial negotiation or are disabled by the software

6.3.3.4.3.2 Entry Conditions

In any of the following scenarios, the TXLKSM of this link transitions to the TXLKSM.INIT state:

- The conditions for transition from TXLKSM.Disable to TXLKSM.INIT are met.
- The local end receives an ERR_RM from the peer end, indicating that this link needs to be retrained, and returns an Ack message.
- The upper layer initiates retraining of this link.

6.3.3.4.3.3 Exit Conditions

Table 65 Conditions for exiting the TXLKSM.INIT state

Next State Following TXLKSM	Jump Conditions
Recovery.relock	All TX lanes to be enabled are initialized (capable of supporting the transmission of control frames), and the fast training indicator of the port is fast_training = 1.
Training.lock	All TX lanes to be enabled are initialized (capable of supporting the transmission of control frames), and the fast training indicator of the port is fast_training = 0.

6.3.3.4.4 Link Training State

6.3.3.4.4.1 State Behavior

The Training state of the TXLKSM is used to complete the training of the TX link.

In this state, each TX lane independently performs lane clock recovery and lock, lane equilibrium, and lane lock. After all lanes are locked (LLFM from the peer end is received), the TXLKSM uniformly controls all TX lanes to perform multi-lane deskew.

In this state, LNSMs of TX lanes at the local end can be in the following states:

- LNSM.INIT for the LNSMs of the TX lanes which are not fully initialized or have training failures
- LNSM.Disable for the LNSMs of the TX lanes which are not enabled for initial negotiation or are disabled by the software
- LNSM.Training for the LNSMs of the TX lanes which are being retrained

6.3.3.4.4.2 Sub-state Description and Exit Conditions

Figure 47 TXLKSM.Training sub-state transition relationship

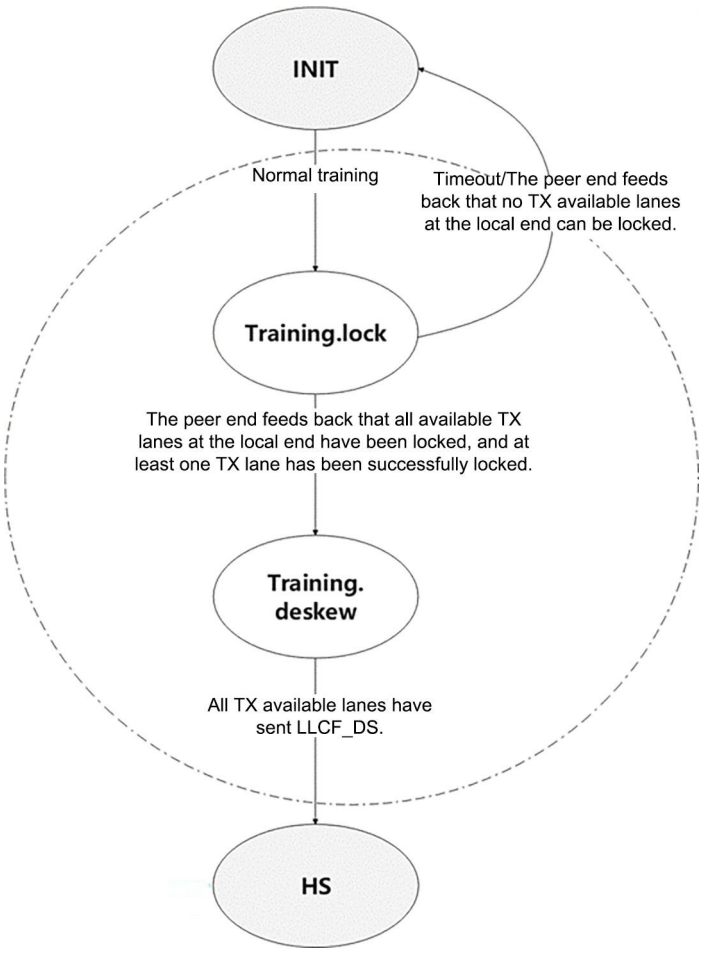


Table 66 Conditions for exiting the TXLKSM.Training state

TXLKSM Sub-state	State and Behavior	Exit Conditions
lock	For waiting for all TX lanes to complete lane training (lane clock lock, lane equilibrium, and lane lock process)	● If the local end receives the LLFM fed back by the peer end, and the LLFM indicates that at least one TX lane of this link has been successfully locked, the TXLKSM of this lane transitions to the Training.deskew state. ● If the local end receives the ERR_RM fed back by the peer end, and the ERR_RM indicates that all TX training of this link fails (failure of any process among lane clock lock, lane equilibrium, and lane lock), the TXLKSM of this lane transitions to the TXLKSM.INIT state. ● Otherwise, if tTxLinkTrainTimeout times out, the TXLKSM transitions to the TXLKSM.INIT state after timeout.

TXLKSM Sub-state	State and Behavior	Exit Conditions
deskew	For allowing all TX lanes on the link to complete multi-lane deskew	● TX lanes successfully locked (in the LNSM.Training.deskew state) at the local end jointly transmit one LLCF_PAD and one LLCF_DS, the TXLKSM transitions to the HS state.
Note: LLCF_PAD is only used to adjust the TX lane skew, and the length of LLCF_PAD transmitted by different lanes may be different.		

6.3.3.4.5 Link Service Transmission State

6.3.3.4.5.1 State Behavior

In the HS state, the TXLKSM can transmit high-speed service data (logical blocks). If one or more TX lanes of the TX link successfully establish links, the TXLKSM can establish a link to transition to the HS state.

After the TXLKSM establishes a link to transition to the HS state, the port link establishment state is reported to the upper layer according to the state of the RX link. The upper layer decides to transmit the logic blocks or not.

In this state, the data path of the TX link needs to perform operations such as FEC coding, lane distribution, scrambling, precoding, and multiplexing according to the rules established during port negotiation.

In this state, LNSMs of TX lanes at the local end can be in the following states:

- LNSM.INIT for the LNSMs of the TX lanes which have training failures
- LNSM.Disable for the LNSMs of the TX lanes which are not enabled for initial negotiation or are disabled by the software
- LNSM.LPx for the LNSMs of the low power TX lanes
- LNSM.Training for the LNSMs of the TX lanes being retrained
- LNSM.Recovery for the LNSMs of the TX lanes being recovered
- LNSM.HS for the LNSMs of other TX lanes that can transmit high-speed services

Note 1: When the link is in the TXLKSM.HS state, the LNSM of at least one TX lane is in the LNSM.HS state.

Note 2: After a TX lane with lane training failures reports an exception, if the upper layer decides not to enable this lane later, it needs to set the state of this lane to lane_disable.

6.3.3.4.5.2 Exit Conditions

After the local end receives an ERR-PRT message which requires the link to perform the link recovery operation (see the exception handling mechanism for details) and feeds back an Ack message, the TXLKSM transitions to the Recovery.lock state.

- Accordingly, if the fast training indicator of this link is fast_recovery = 0, the TXLKSM controls the LNSM (TX) of the TX lane being used at the local end to enter the Recovery.re_lock state for lane recovery.

- Accordingly, if the fast training indicator of this link is fast_recovery = 1, the TXLKSM controls the LNSM (TX) of the TX lane being used at the local end to enter the Recovery.Fast_lock state for lane recovery.

After the local end receives an ERR_RM that requires link retraining (see 6.3.6 for details) and feeds back an Ack message, the TXLKSM transitions to the INIT state.

- Accordingly, the TXLKSM controls the LNSM (TX) of the TX lane being used at the local end to enter the INIT state for lane retraining.

After the local end transmits an LPRM request for this link to enter the LP0f state, and all TX lanes of this link transmit the LLCF_EI, the TXLKSM transitions to the LPx.LP0f state.

- Accordingly, the TXLKSM controls the LNSM (TX) of the TX lane being used at the local end to enter the LPx.LP0f state.

After the local end transmits an LPRM request for this link (or this port) to enter the LPx state and receives an Ack message responding to the LPRM, and all TX lanes of this link transmit the LLCF_EI, the TXLKSM transitions to the LPx state.

- If the target low power state for this link is LP0, the TXLKSM transitions to LPx.LP0.
- If the target low power state for this link is LP1, the TXLKSM transitions to LPx.LP1.
- If the target low power state for this link is LP2, the TXLKSM transitions to LPx.LP2.
- If the target low power state for this link is LP3, the TXLKSM transitions to LPx.LP3.
- Accordingly, the TXLKSM controls the LNSM (TX) of the TX lane being used at the local end to enter the LPx state.

After the local end receives an LPRM request for this port to enter the LPx state and feeds back an Ack message responding to the LPRM, and all TX lanes of this link transmit the LLCF_EI, the TXLKSM transitions to the LPx state.

- If the target low power state for this port is LP0, the TXLKSM transitions to LPx.LP0.
- If the target low power state for this port is LP1, the TXLKSM transitions to LPx.LP1.
- If the target low power state for this port is LP2, the TXLKSM transitions to LPx.LP2.
- If the target low power state for this port is LP3, the TXLKSM transitions to LPx.LP3.
- Accordingly, the TXLKSM controls the LNSM (TX) of the TX lane being used at the local end to enter the LPx state.

Note: The local end needs to record that the TXLKSM enters the port low power state or single TX link low power state. If the TXLKSM enters the low power mode for a single TX link, only the TX link needs to be woken up during low power wake-up. If the TXLKSM enters the low power mode for links of a port, the port (including transmitting and RX links) needs to be woken up during low power wake-up.

6.3.3.4.6 Link Recovery State

6.3.3.4.6.1 State Behavior

The Recovery state of the TXLKSM is used for recovery of the TX link, including independent recovery of each lane (lane equilibrium and lane lock) and multi-lane deskew.

In this state, each TX lane independently performs lane equilibrium and lane lock. After all lanes are locked (LLFM from the peer end is received), the TXLKSM uniformly controls all TX lanes to perform multi-lane deskew.

When the link enters this state, all features of the TX data path, including FEC coding, lane distribution, scrambling code, precoding, and multiplexing, are reset, and all TX lanes simultaneously transmit LLCF_TS2 for link recovery.

In this state, LNSMs of TX lanes at the local end can be in the following states:

- LNSM.Disable for the LNSMs of the TX lanes which are not enabled for initial negotiation or are disabled by the software
- LNSM.LPx for the LNSMs of the low power TX lanes
- LNSM.Recovery for the LNSMs of the TX lanes which are being retrained
- LNSM.INIT for the LNSMs of the TX lanes which have training failures

6.3.3.4.6.2 Entry Conditions

If any of the following scenarios occurs, this link enters the TXLKSM.Recovery state:

- The conditions for INIT/HS/LPx to transition to the TXLKSM.Recovery state are met.
- The local end receives an ERR_RM from the peer end, indicating that the link needs to be recovered, and returns an Ack message.

6.3.3.4.6.3 Sub-state Description and Exit Conditions

Figure 48 TXLKSM.Recovery sub-state transition relationship

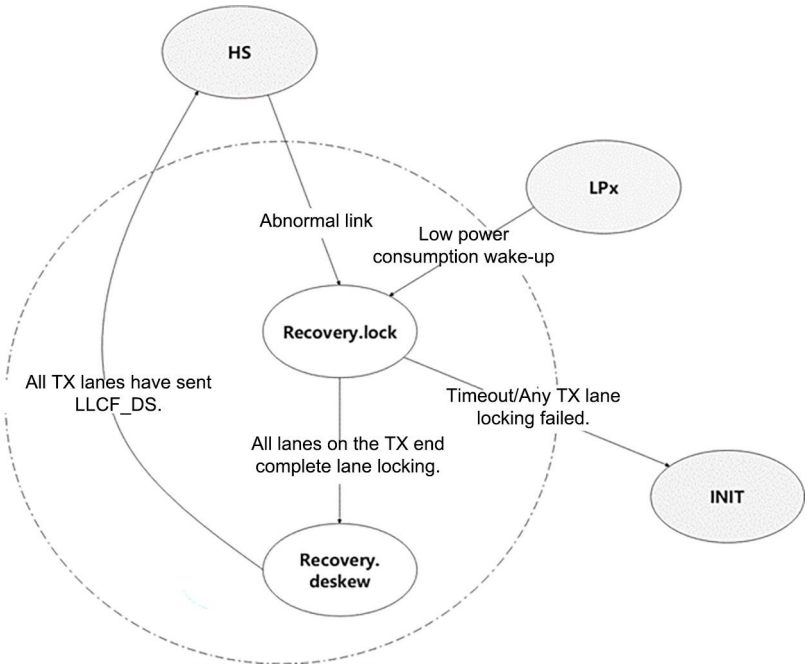


Table 67 Conditions for exiting the TXLKSM.Recovery state

Recovery Sub-state	State Description	Exit Conditions
lock	Used to wait for all TX lanes of the TX link to be locked	● If the local end receives the LLFM fed back by the peer end, and the LLFM indicates that all TX lanes in use of this link have been successfully locked, the TXLKSM transitions to the Recovery.deskew state. ● When this state is a result of link exceptions, if the fast recovery indicator is fast_recovery = 0, the local end receives the LLFM fed back by the peer end, and the LLFM indicates that all TX lanes to be enabled of this link have been successfully locked, the TXLKSM transitions to the Recovery.deskew state. ● When this state is a result of link exceptions, if the fast recovery indicator fast_recovery = 0, the local end receives the LLFM fed back by the peer end, and the LLFM indicates that a TX lane in use of this link is not locked, the TXLKSM transitions to the INIT state. ● When this state is a result of link exceptions, if the fast recovery indicator fast_recovery = 1, and all TX lanes at the local end have transmitted the specified number of LLCF_TS2, the TXLKSM transitions to the Recovery.deskew state. ● When fast training enters this state, if the local end receives the LLFM fed back by the peer end, and the LLFM indicates that all TX lanes in use of this link have been successfully locked, the TXLKSM transitions to the Recovery.deskew state. ● Otherwise, the tLinkRecoveryTimeout times out and the TXLKSM transitions to the INIT state.
deskew	Used to complete multi-lane alignment.	After all TX lanes of this link jointly transmit one LLCF_PAD and one LLCF_DS, the TXLKSM transitions to the HS state.

6.3.3.4.7 Link Low Power State

6.3.3.4.7.1 State Behavior

When the TXLKSM is in the LPx state, its high-speed link is in the low power state. In this state, the TX link cannot transmit high-speed data. According to the degree of power saving, the LPx state is divided into multiple gears: LP0f, LP0, LP1, LP2, and LP3.

In this state, LNSMs of TX lanes at the local end can be in the following states:

- LNSM.Disable for the LNSMs of the TX lanes which are not enabled for initial negotiation or are disabled by the software

- LNSM.INIT for the LNSMs of the TX lanes which have training failures
- LNSM.LPx for the LNSMs of the TX lanes in the low power state

Note: For initiation of LPRMs and LWAMs, all TX lanes that need to enter the low power state must enter the same low power sub-state (LP0f, LP0, LP1, LP2, or LP3).

6.3.3.4.7.2 Conditions for Exiting the LPx State

Figure 49 TXLKSM LPx sub-state transition relationship

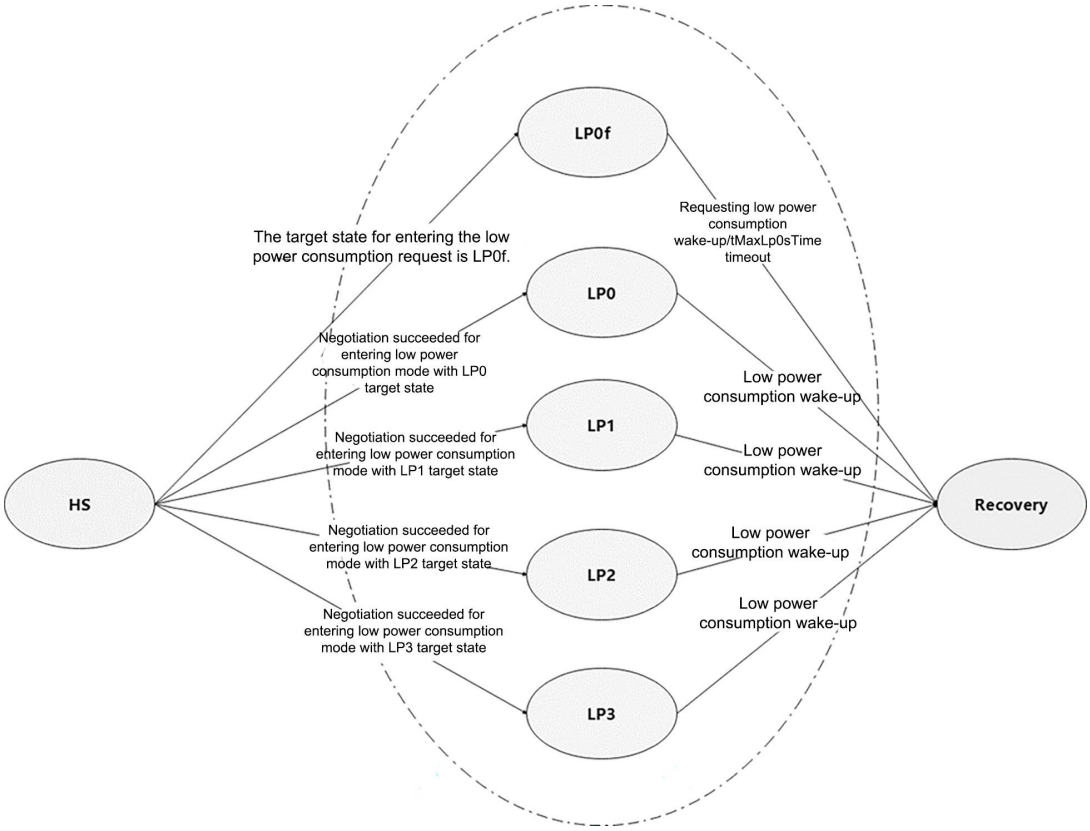


Table 67 Conditions for exiting the TXLKSM.LPx state

LPx Sub-state	Exit Conditions
LP0f	After the TX link at the local end receives a link wake-up request initiated by the upper layer, or the time when the TXLKSM at the local end is in the LPx.LP0f state exceeds the tMaxLp0fTime, the link wake-up will be initiated. The TXLKSM transitions to the Recovery.lock state. Accordingly, the TXLKSM controls the LNSM (TX) at the local end to enter the Recovery.fast_lock state.
LP0–LP2	After receiving the link (port) wake-up request initiated by the upper layer, the TX link at the local end initiates the link wake-up. The TXLKSM transitions to the Recovery.lock state. Accordingly, the TXLKSM controls the LNSM (TX) at the local end to enter the Recovery.re_lock state. If the local end is in the port low power state, and the RX link at the local end senses that any RX lane is not electrically idle, the TX link and RX link at the local

LPx Sub-state	Exit Conditions
	end will be woken up. The TXLKSM at the local end transitions to the Recovery.lock state. Accordingly, the TXLKSM controls the LNSM (TX) at the local end to enter the Recovery.re_lock state.
LP3	After the upper layer initiates the link (port) low power wake-up, and the local end transmits the LPRM for the link (port) and receives the corresponding response information, the link will be woken up. The TXLKSM transitions to the Recovery.lock state. Accordingly, the TXLKSM controls the LNSM (TX) at the local end to enter the Recovery.re_lock state. After the local end receives an LPRM for port low power wake-up transmitted by the peer end and feeds back a response, the link wake-up will be initiated. The TXLKSM transitions to the Recovery.lock state. Accordingly, the TXLKSM controls the LNSM (TX) at the local end to enter the Recovery.re_lock state.

6.3.3.5 RX Link State

6.3.3.5.1 State Introduction

An RXLKSM is used to manage the RX link state of the entire port. The state transition relationship and its introduction are as follows:

Figure 50 RXLKSM state transition relationship

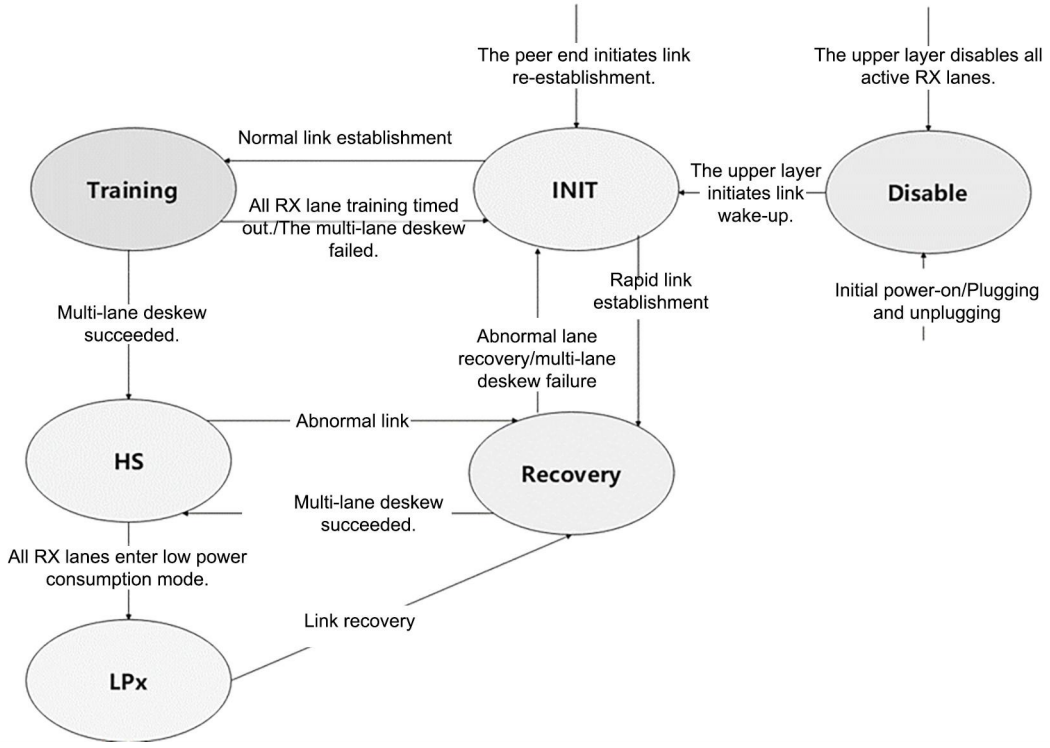


Table 68 RXLKSM states

RXLKSM State	State Introduction
Disable	Link invalid state. The upper layer disables all RX lanes, or it is in initial power-on or disconnected state. In this state, the RX link has not started initial training.
INIT	Link initialization state. This state corresponds to the port configuration state of the port state machine. In this state, after the port capability negotiation is completed, all RX lanes at the local end perform electrical layer initialization based on the link configuration results.
Training	Link training state. In this state, the RX lanes to be enabled at the local end perform lane training.
HS	Link service transmission state. In this state, the RX lanes that have been successfully trained can perform high-speed service data transmission together.
Recovery	Link recovery state. In this state, all RX lanes to be enabled at the local end perform lane recovery.
LPx	Link low power state. No high-speed service is transmitting, and the RX lanes to be enabled at the local end are all in the low power state.

The RXLKSM and the LNSM (RX) are in an interdependent master-slave relationship. The RXLKSM completes multi-lane RX collaborative link operations, and LNSM (RX) completes independent single lane RX operations.

6.3.3.5.2 Link Invalid State

6.3.3.5.2.1 State Behavior

This state is the default state, and the main RX link of this port has not been enabled. All data paths and control-related logic and states of the main RX link are in the reset state.

After the port completes the port capability negotiation, the RXLKSM exits this state and performs port configuration and high-speed link initialization.

In this state, LNSMs of all RX lanes at the local end are in the LNSM.Disable state.

6.3.3.5.2.2 Entry Conditions

If any of the following scenarios occurs, this link enters the RXLKSM.Disable state:

- Initial power-on state.
- Port disconnected.

- The upper layer configures the lane_disable indicator of all RX lanes at the local end to 1.

6.3.3.5.2.3 Exit Conditions

If the upper layer configures the lane_disable indicator of any RX lane at the local end to 0, the RXLKSM transitions from the RXLKSM.Disable state to the RXLKSM.INIT state.

6.3.3.5.3 Link Initialization State

6.3.3.5.3.1 State Behavior

This state is the initial state, and the main RX link of this port has not started to receive high-speed data. All data paths and control-related logic and states of the main RX link are in the initial state.

After the port state machine completes the port capability negotiation, the RX link at the local end performs link initialization based on the negotiated rate. After all RX lanes to be enabled are initialized, the RXLKSM exits this state and starts the high-speed link establishment process.

In this state, LNSMs of RX lanes at the local end can be in the following states:

- LNSM.INIT for the LNSMs of the RX lanes which are not fully initialized or have training failures
- LNSM.Disable for the LNSMs of the RX lanes which are not enabled for initial negotiation or are disabled by the software

6.3.3.5.3.2 Entry Conditions

In any of the following scenarios, the RXLKSM of this link transitions to the RXLKSM.INIT state:

- After the management adapter receives the capability negotiation message from the peer end, it issues the configurations.
- The conditions for transition from RXLKSM.Disable to RXLKSM.INIT are met.
- The local end transmits an ERR_RM indicating that this link needs training, and receives an Ack message.

6.3.3.5.3.3 Exit Conditions

Table 69 Conditions for exiting the RXLKSM.INIT state

Next State	Jump Conditions
RXLKSM.Recovery	All RX lanes to be enabled are initialized (capable of supporting the receiving of control frames), and the fast training indicator of the port is fast_training = 1.
RXLKSM.Training	All RX lanes to be enabled are initialized (capable of supporting the receiving of control frames), and the fast training indicator of the port is fast_training = 0.

6.3.3.5.4 Link Training State

6.3.3.5.4.1 State Behavior

The Training state of RXLKSM is used to complete the training of the RX link.

In this state, each RX lane independently performs lane clock recovery and lock, lane equilibrium, and lane lock. After all lanes are locked, the RXLKSM uniformly controls all RX lanes to perform multi-lane deskew.

In this state, LNSMs of RX lanes at the local end can be in the following states:

- LNSM.INIT for the LNSMs of the RX lanes which are not fully initialized or have training failures
- LNSM.Disable for the LNSMs of the RX lanes which are not enabled for initial negotiation or are disabled by the software
- LNSM.Training for the LNSMs of the RX lanes which are being retrained

6.3.3.5.4.2 Sub-state Description and Exit Conditions

Figure 51 RXLKSM. Training sub-state transition relationship

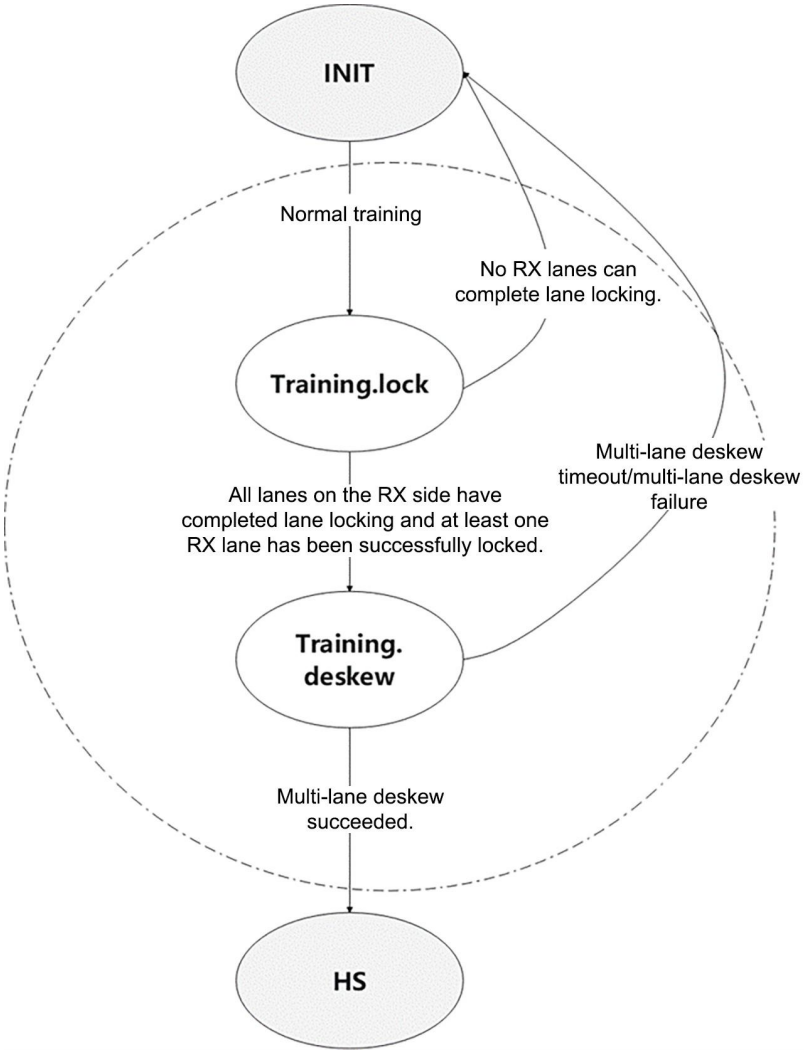


Table 70 Conditions for exiting the RXLKSM.Training state

Training Sub-state	State and Behavior	Exit Conditions
lock	For waiting for all RX lanes to complete independent lane training	● If all RX lanes at the local end have completed lane training (even if the lane clock lock, lane equilibrium, or lane lock process fails), and at least one RX lane has succeeded in all lane training processes (including lane clock lock, lane equilibrium, and lane lock), the RXLKSM transitions to the Training.deskew state after the local end feeds back an LLFM to the peer end. ● If all RX lanes at the local end have completed lane training (even if the lane clock lock, lane equilibrium, or lane lock process fails), but no RX lane has succeeded in all lane training processes (including lane clock lock, lane equilibrium, and lane lock), this end feeds back a CLFM, EQFM, and LLFM to report the training failure, and receives responses to these messages (a response is required for a CLFM or EQFM, but is not required for an LLFM). Then, the RXLKSM transitions to the INIT state.
deskew	For allowing all RX lanes on the link to complete multi-lane deskew	● If the RX end of this link succeeds based on LLCF_DS deskew, the RXLKSM transitions to the RXLKSM.HS state. ● If the RX end of this link fails based on LLCF_DS deskew, the local end feeds back a multi-lane deskew error based on the ERR_RM and receives a response to the ERR_RM. Then, the RXLKSM transitions to the RXLNSM.INIT state. ● If the tWaitDsTimeout at the local end times out, the local end feeds back a deskew timeout error based on the ERR_RM, and receives a response to the ERR_RM. Then, the RXLKSM transitions to the RXLKSM.INIT state.

6.3.3.5.5 Link Service Transmission State

6.3.3.5.5.1 State Behavior

In the HS state, the RXLKSM supports the transmitting and receiving of high-speed service data (logical blocks). If one or more RX lanes successfully establish links, the RXLKSM can establish a link to transition to the HS state.

After the RXLKSM establishes a link to transition to the HS state, the port link establishment state is reported to the upper layer according to the state of the RX link. The upper layer decides to transmit high-speed service data or not.

In this state, the data path of the RX link needs to perform operations such as FEC decoding, lane merging, descrambling, deprecoding, and demultiplexing according to the rules established during port negotiation.

In this state, LNSMs of RX lanes at the local end can be in the following states:

- LNSM.INIT for the LNSMs of the RX lanes which have training failures
- LNSM.Disable for the LNSMs of the RX lanes which are not enabled for initial negotiation or are disabled by the software
- LNSM.LPx for the LNSMs of the low power RX lanes
- LNSM.Training for the LNSMs of the RX lanes being retrained
- LNSM.Recovery for the LNSMs of the RX lanes being recovered
- LNSM.HS for the LNSMs of other TX lanes that can transmit high-speed services

Note 1: When the link is in the RXLKSM.HS state, the LNSM of at least one RX lane is in the LNSM.HS state.

Note 2: After an RX lane with lane training failures reports an exception, if the upper layer decides not to enable this lane later, it needs to set the state of this lane to lane_disable.

6.3.3.5.5.2 Conditions for Exiting the HS State

If the RX link at the local end detects an exception that requires link recovery, it transmits an ERR_RM to the peer end. After the peer end feeds back an Ack message responding to the ERR_RM, the RXLKSM transitions to the Recovery.lock state:

- If the fast training indicator of this link is fast_recovery = 0, the RXLKSM controls the LNSM (RX) of the RX lane being used at the local end to enter the Recovery.re_lock state for lane recovery.
- If the fast training indicator of this link is fast_recovery = 1, the RXLKSM controls the LNSM (RX) of the RX lane being used at the local end to enter the Recovery.Fast_lock state for lane recovery.

If the RX link at the local end detects an exception that requires link retraining, it transmits an ERR_RM to the peer end. After the peer end feeds back an Ack message responding to the ERR_RM, the RXLKSM transitions to the INIT state:

- The RXLKSM controls the LNSM (RX) of the RX lane being used at the local end to enter the INIT state for lane retraining.

After the local end receives an LPRM request for this link to enter the LP0f state, and all RX lanes of this link receive the LLCF_EI (at least one LLCF_EI is detected; this applies to the following scenarios involving LLCF_EI detection), the RXLKSM transitions to the LPx.LP0f state:

- The RXLKSM controls the LNSM (RX) of the RX lane being used at the local end to enter the LPx.LP0f state.

After the local end receives an LPRM request for this link (or this port) to enter the LPx state and feeds back an Ack message responding to the LPRM, and all RX lanes of this link receive the LLCF_EI, the RXLKSM transitions to the LPx state:

- If the target low power state for this link is LP0, the TXLKSM transitions to LPx.LP0.
- If the target low power state for this link is LP1, the TXLKSM transitions to LPx.LP1.
- If the target low power state for this link is LP2, the TXLKSM transitions to LPx.LP2.
- If the target low power state for this link is LP3, the TXLKSM transitions to LPx.LP3.
- The RXLKSM controls the LNSM (RX) of the RX lane being used at the local end to enter the LPx state.

After the local end transmits an LPRM request to enter the LPx state and receives an Ack message fed back by the peer end to respond to the LPRM, and all RX lanes of this link receive the LLCF_EI, the RXLKSM transitions to the LPx state:

- If the target low power state for this link is LP0, the TXLKSM transitions to LPx.LP0.
- If the target low power state for this link is LP1, the TXLKSM transitions to LPx.LP1.
- If the target low power state for this link is LP2, the TXLKSM transitions to LPx.LP2.
- If the target low power state for this link is LP3, the TXLKSM transitions to LPx.LP3.
- The RXLKSM controls the LNSM (RX) of the RX lane being used at the local end to enter the LPx state.

Note: The local end needs to record that the RXLKSM enters the port low power state or single RX link low power state. If the RXLKSM enters the low power mode for a single RX link, only the RX link needs to be woken up during low power wake-up. If the TXLKSM enters the low power mode for links of a port, the port (including transmitting and RX links) needs to be woken up during low power wake-up.

6.3.3.5.6 Link Recovery State

6.3.3.5.6.1 State Behavior

The recovery state of the RXLKSM is used for recovery of the RX link, including independent recovery of each lane (lane equilibrium and lane lock) and multi-lane deskew.

In this state, each RX lane independently performs lane equilibrium and lane lock. After all lanes are locked (LLFM is transmitted to the peer end), the RXLKSM uniformly controls all RX lanes to perform multi-lane deskew.

When the link enters this state, all RX data path features, including FEC decoding, lane merging, descrambling, deprecoding, demultiplexing, and deskew, are reset.

In this state, LNSMs of RX lanes at the local end can be in the following states:

- LNSM.Disable for the LNSMs of the RX lanes which are not enabled for initial negotiation or are disabled by the software

- LNSM.LPx for the LNSMs of the low power RX lanes
- LNSM.Recovery for the LNSMs of the RX lanes which are being retrained
- LNSM.INIT for the LNSMs of the RX lanes which have training failures

6.3.3.5.6.2 Sub-state Description and Exit Conditions

Figure 52 RXLKSM.Recovery sub-state transition relationship

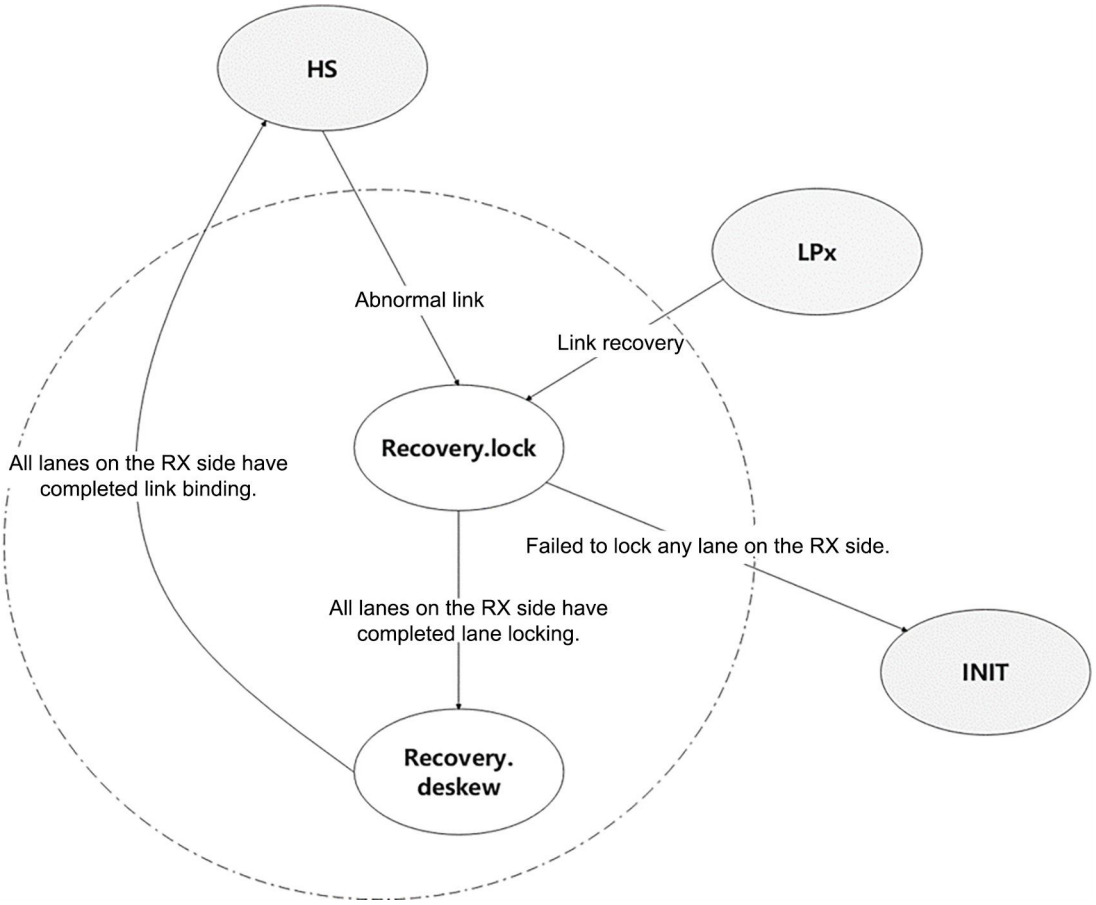


Table 71 Conditions for exiting the RXLKSM.Recovery state

Recovery Sub-state	State and Behavior	Exit Conditions
lock	For waiting for all RX lanes to complete lane lock	● If the fast recovery indicator fast_recovery = 0, after all RX lanes at the local end are locked, and the local end uniformly transmits LLFMs, the RXLKSM transitions to the Recovery.deskew state. ● If the fast recovery indicator fast_recovery = 1, after all RX lanes at the local end are locked (each lane receives a specified number of LLCF_TS2), the RXLKSM transitions to the Recovery.deskew state. ● If any RX lane at the local fails to be locked,

Recovery Sub-state	State and Behavior	Exit Conditions
		the local end transmits an ERR_RM to indicate the lane lock failure. After the local end receives an Ack message, the RXLKSM transitions to the INIT state.
deskew	For allowing all RX lanes on the link to complete multi-lane deskew	● If all RX ends of this link succeed based on LLCF_DS deskew, the RXLKSM transitions to the RXLKSM.HS state. ● If the RX end of this link fails based on LLCF_DS deskew, the local end feeds back a multi-lane deskew error based on the ERR_RM and receives a response to the ERR_RM. Then, the RXLKSM transitions to the RXLNSM.INIT state. ● If the tWaitDsTimeout at the local end times out, the local end feeds back a deskew timeout error based on the ERR_RM, and receives a response to the ERR_RM. Then, the RXLKSM transitions to the RXLNSM.INIT state.

6.3.3.5.7 Link Low Power State

6.3.3.5.7.1 State Behavior

When the RXLKSM is in the LPx state, its high-speed link is in the low power state. The RX link does not need to receive high-speed data. According to the degree of power saving, the LPx state is divided into multiple gears: LP0f, LP0, LP1, LP2, and LP3.

In this state, LNSMs of RX lanes at the local end can be in the following states:

- LNSM.Disable for the LNSMs of the RX lanes which are not enabled for initial negotiation or are disabled by the software
- LNSM.LPx for the LNSMs of the RX lanes in the low power state
- LNSM.INIT for the LNSMs of the RX lanes which have training failures

6.3.3.5.7.2 Sub-state Description and Exit Conditions

Figure 53 RXLKSM LPx sub-state transition relationship

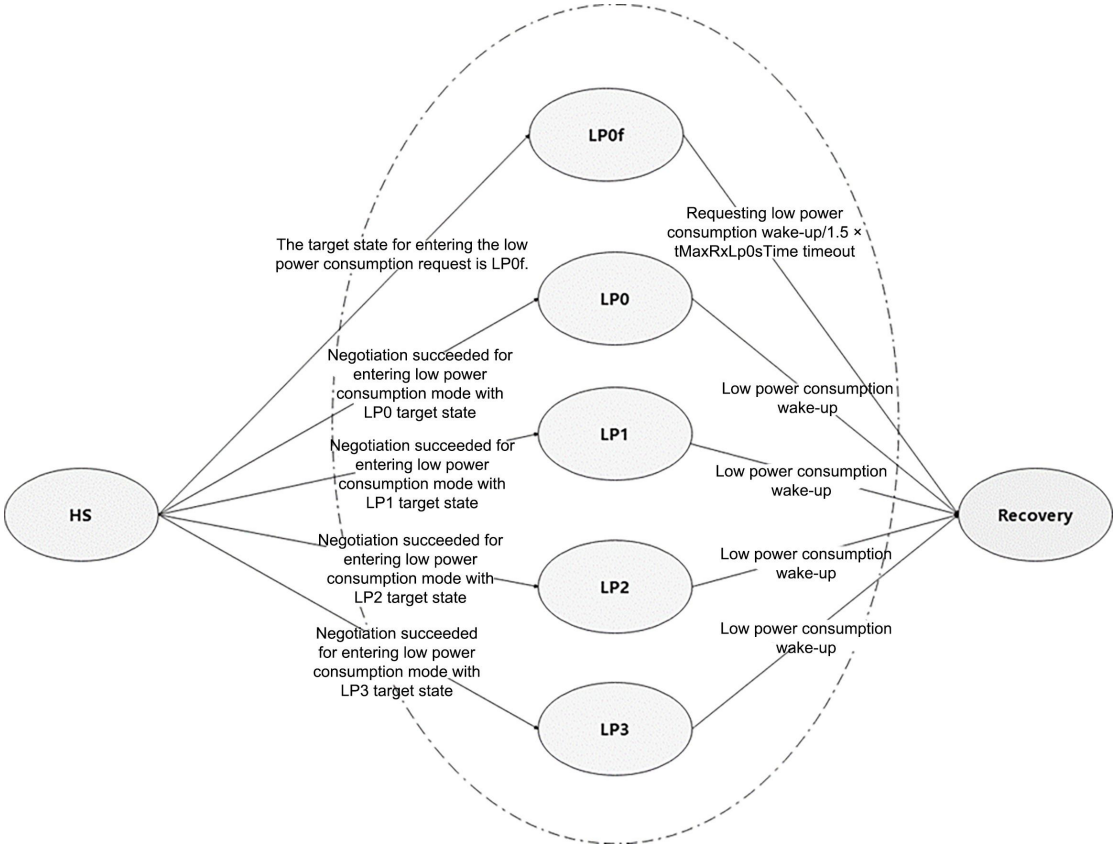


Table 72 Conditions for exiting the RXLKSM.LPx state

LPx Sub-state	Exit Conditions
LP0f	When the RX link senses that any RX lane is not electrically idle, or the RXLKSM at the local end is in the LPx.LP0f state for more than 1.5*tMaxRxLp0fTime, the link wake-up will be initiated. The RXLKSM transitions to the Recovery.lock state. The RXLKSM controls the LNSM (RX) at the local end to enter the Recovery.fast_lock state.
LP0–LP2	● If the RX link senses that any RX lane is not electrically idle, it will wake up the RX link at the local end. The RXLKSM transitions to the Recovery.lock state. The RXLKSM controls the LNSM (RX) at the local end to enter the Recovery.re_lock state. ● Otherwise, if the local end is in the port low power state, after receiving the port wake-up request from the upper layer, the local end will initiate the link wake-up. The TXLKSM transitions to the Recovery.lock state. The RXLKSM controls the LNSM (RX) at the local end to enter the Recovery.re_lock state.
LP3	● After the local end receives an LPRM for link/port low power wake-up transmitted by the peer end and feeds back a response, the link wake-up will

LPx Sub-state	Exit Conditions
	be initiated. The RXLKSM transitions to the Recovery.lock state. The RXLKSM controls the LNSM (RX) at the local end to enter the Recovery.re_lock state. ● Otherwise, if the local end is in the port low power state, after the local end receives a port wake-up request from the upper layer, it will transmit a port wake-up LPRM. After receiving a response, it will initiate the link wake-up. The RXLKSM transitions to the Recovery.lock state. The RXLKSM controls the LNSM (RX) at the local end to enter the Recovery.re_lock state.

6.3.3.6 Main Link States of Ports

The link layer needs to provide real-time feedback on the overall state of links for the upper layer to query.

For a port with only TX links, the overall link state is the same as that of the TXLKSM.

For a port with only RX links, the overall link state is the same as that of the RXLKSM.

After each restart of link training (triggered by an ERR_RM or by a link retraining command from the upper layer), links will be retrained. After the retraining is completed, the link training results are reported to the software as a port_state_change event. The training results of each lane are indicated in the port_state_change event.

When the overall link state is trained to the HS state, a port_state_change (training success) event is reported.

Otherwise, a port_state_change (training fail) event is reported to the software.

6.3.3.7 Low Power State and Process

6.3.3.7.1 Process of Entering the Link Low Power State

6.3.3.7.1.1 Overview

Requests for a link to enter the low power state include two mechanisms: unidirectional link entering low power and port entering low power. For the port entering low power mechanism, see Appendix D.

Unidirectional link entering low power means that all lanes in the TX or RX direction (never TX and RX directions) of a port enter the low power state. This mainly applies to scenarios where service (such as audio and video) data is unidirectionally transmitted. The request for a unidirectional link entering the low power state is initiated in the TX direction of the link.

A unidirectional link in the low power state can be at LP0f, LP0, LP1, or LP2 gears.

Any link is prohibited from repeatedly initiating the link low power process when performing initial link training, link recovery, or dynamic lane state switching, or in other low power processes.

Note: For a unidirectional link, a low-power request for the port to enter LP3 can be initiated to save more power.

6.3.3.7.1.2 Unidirectional Link Entering LP0f

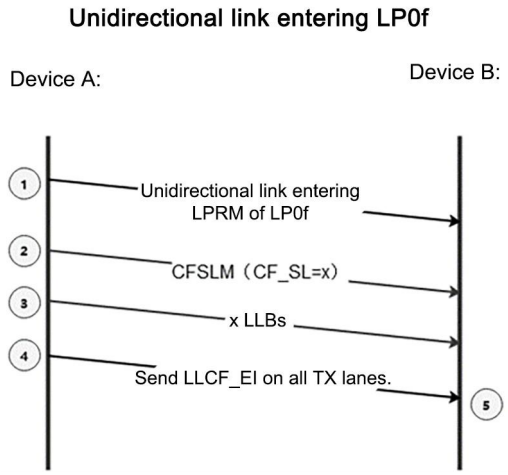
For application scenarios where service data flows are frequently cut off and frequently restored, the mechanism of unidirectional link entering LP0f can be used during the period when the data flow is cut off to save power.

Typical scenario: When the local device is an audio/video source device, and the logical layers of both local and peer devices support the LP0f gear, entering the LP0f state is allowed.

Note: LP0f is recommended for scenarios that require frequent transitions to/from the low power state (such as blanking blocks in videos). LP0f is highly correlated with the data source state.

The process of a unidirectional link entering LP0f is shown in Figure 54.

Figure 54 Process of a link entering LP0f



(1) Device A transmits a LPRM requesting unidirectional link entering LP0f to device B.

(2) Device A transmits one CFSLM to device B and specifies CF_SL as x in the CFSLM.

(3) Device A transmits x LLBs to device B.

(4) All TX lanes of device A transmit four LLCF_EI control frames, and then all TX lanes of device A enter the low power state.

(5) After device B receives the LPRM from device A and receives the LLCF_EI control frames in all RX lanes, all RX lanes of device B enter the low power state.

Note: Transmitting one CFSLM and x LLBs in steps 2 and 3 in the above figure is a necessary operation for switching the data flow from LLB to CF (control frame). This will not be specifically or repeatedly explained for subsequent switching of data flows from LLB to CF.

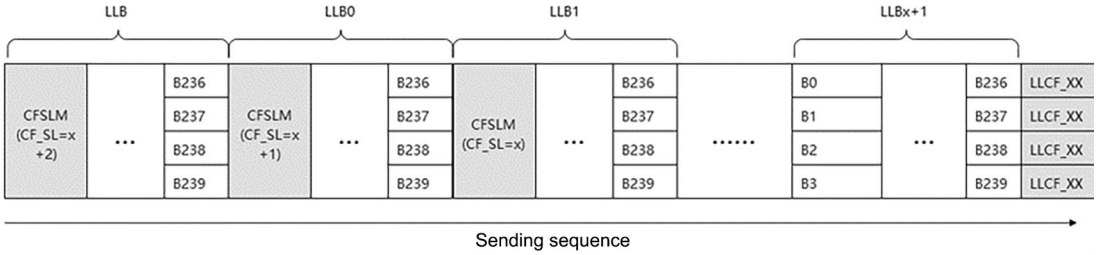
The process of switching a data flow from LLB to CF is as follows:

If the link state changes when the transmitting end of the main link is transmitting an LLB, and a control frame (CF) needs to be transmitted, the transmitting end shall transmit a CFSLM in advance to achieve communication between the CF transmitting position at the transmitting end and the CF detecting position at the receiving end.

As shown in Figure 55, assuming that the minimum acceptable CF_SL value agreed by the two ends is x0, in order to transmit LLCF_XX (including LLCF_TS0/1/2, LLCF_EI, LLCF_DS, LLCF_DST, LLCF_EIE, and LLCF_PAD) after x + 2 (x ≥ x0) LLBs are transmitted, the transmitting

end shall transmit CFSLMs in three consecutive LLBs, specify CF_SL as x + 2, x + 1, and x respectively, and then transmit LLCF_XX at the position of the x-th LLB after the last LLB is transmitted. In this document, the above transmission rules also apply to other descriptions indicating the position where the CFSLMs are transmitted.

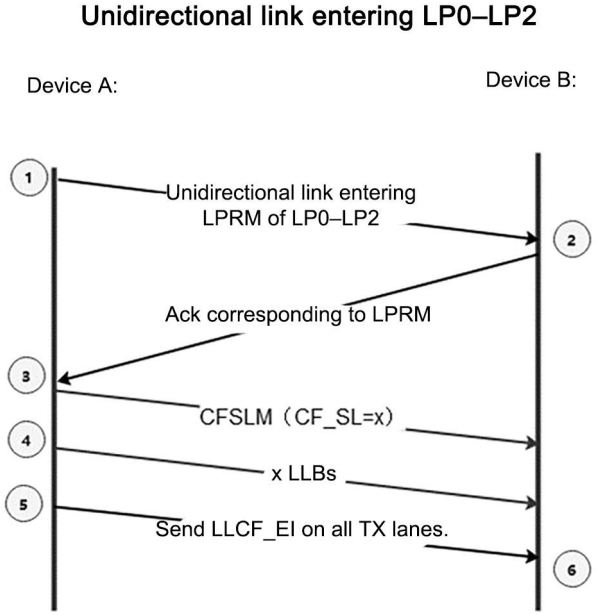
Figure 55 Sequence of inserting CFs in a data flow



6.3.3.7.1.3 Unidirectional Link Entering LP0–LP2

The process of a unidirectional link entering LP0–LP2 is shown in Figure 56.

Figure 56 Process of a link entering LP0–LP2



(1) Device A transmits a LPRM requesting unidirectional link entering LP0–LP2 to device B.

(2) Device B feeds back an Ack message responding to the LPRM to device A.

If device B feeds back a Nack response to the LPRM to device A, the process ends and neither port enters the low power state. For subsequent Nack processes of other low power requests, the same rules apply and will not be repeated.

(3) Device A transmits one CFSLM to device B and specifies CF_SL as x in the CFSLM.

(4) Device A transmits x LLBs to device B.

(5) After all TX lanes of device A transmit service data, they transmit four LLCF_EI control frames, and then all TX lanes of device A enter the low power state.

(6) After device B receives the LPRM from device A and receives the LLCF_EI control frames in all RX lanes, all RX lanes of device B enter the low power state.

6.3.3.7.1.4 Conditions for Refusing to Enter the Low Power State

When the peer device initiates a low power request, the local end shall decide whether it can enter the low power state according to the states of the audio and video adapter, management adapter, and other devices. If the rejection conditions are met, the local end shall feed back a Nack response to the peer end. Otherwise, it shall feed back an Ack response. The rejection conditions for various scenarios are shown in Table 73.

Table 73 Conditions for refusing to enter the low power state

Scenario	Conditions
ejecting a unidirectional link to enter LP0	Any of the following conditions is true: • The audio and video adapter refuses to enter LP0. • The management adapter refuses to enter LP0. • The current link does not support LP0.
Rejecting a unidirectional link to enter LP1	Any of the following conditions is true: • The audio and video adapter refuses to enter LP1. • The management adapter refuses to enter LP1. • The current link does not support LP1.
Rejecting a unidirectional link to enter LP2	Any of the following conditions is true: • The audio and video adapter refuses to enter LP2. • The management adapter refuses to enter LP2. • The current link does not support LP2.
Rejecting a unidirectional link to enter LP3	Any of the following conditions is true: • The audio and video adapter refuses to enter LP3. • The management adapter refuses to enter LP3. • The current link does not support LP3.

Note: The audio and video adapter and management adapter must implement low power entry and exit rejection conditions, based on the low power entry delay (tLPxEntry) and exit delays (tLP0Exit, tLP1Exit, and tLP2Exit) specified for each low power gear.

The maximum low power entry delay tLPxEntry specified in the Protocols is detailed in the training parameter requirements.

6.3.3.7.1.5 Exception Handling Mechanism for Entering the Low Power Process

6.3.3.7.1.5.1 Message Loss

Table 74 Troubleshooting message loss in the process of entering the low power state

Exception	Handling Mechanism
Loss of LPRM for a unidirectional link to enter LP0f	The low power responder (the port receiving the LPRM) detects a link error exception and initiates link retraining or link recovery through ERR_RM.
Loss/Exception of LPRM for a unidirectional link to enter LP0, LP1, LP2, or LP3	This type of request message requires the low power responder to feed back a response message. If the response message is lost/abnormal, the LPRM needs to be retransmitted. (See the description in 6.3.6)
Loss/Exception of the message responding to LPRM for a unidirectional link to enter LP0, LP1, LP2, or LP3	The requester retransmits the LPRM, and the responder feeds back an Ack or a Nack message after receiving the LPRM each time. After the requester fails to receive the corresponding response message after multiple retransmissions, link retraining is initiated through ERR_RM.
LLCF_EI loss/exception	The low power responder (the port receiving the LPRM) detects a link error exception and initiates link retraining or link recovery through ERR_RM. Note: If the receiving end detects any frame header in the LLCF_EI format at the position where an LLCF_EI is received, the LLCF_EI has been recognized.

6.3.3.7.2 Process of Exiting the Link Low Power State

6.3.3.7.2.1 Overview

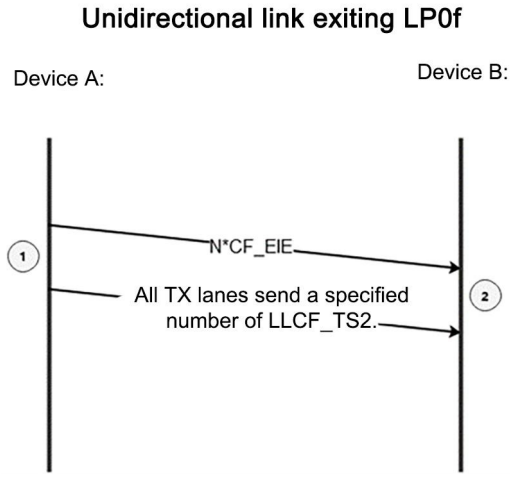
Requests for a link to exit the low power state include two mechanisms: unidirectional link exiting low power and port exiting low power. For the port exiting low power mechanism, see Appendix D.

A unidirectional link exiting low power means that all lanes in the TX or RX direction of the port exit the low power state. The request for a unidirectional link exiting the low power state is initiated in the TX direction of the link.

If the current link is in the link low power state, the low power exit initiating device can only initiate a link low power exit request. If it initiates other low power exit requests, these requests will be ignored. The processing is shown in Table 44.

6.3.3.7.2.2 Unidirectional Link Exiting LP0f

Figure 57 Schematic diagram of a unidirectional link exiting LP0f

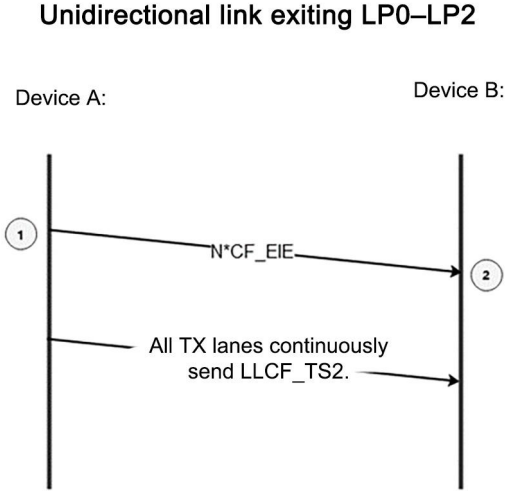


(1) The TX link of device A exits the low power state (the corresponding TXLKSM enters the Recovery state), and transmits N_EIE LLCF_EIE and a specified number (determined by the capability negotiation information) of LLCF_TS2 to device B.

(2) Device B senses that the receiving end is in a non-electrical idle state, and the RX link of device B exits the low power state (the corresponding RXLKSM enters the Recovery state).

6.3.3.7.2.3 Unidirectional Link Exiting LP0–LP2

Figure 58 Schematic diagram of a unidirectional link exiting LP0–LP2



(1) The TX link of device A exits the low power state (the corresponding TXLKSM enters the Recovery state) and transmits N_EIE LLCF_EIE to device B. Then, it continues to transmit LLCF_TS2 for link recovery.

(2) Device B senses that the receiving end is in a non-electrical idle state, and the RX link of device B exits the low power state (the corresponding RXLKSM enters the Recovery state).

6.3.3.7.2.4 Exception Handling Mechanism for Exiting the Low Power Process

Table 75 Troubleshooting message loss in the process of exiting the low power state

Exception	Handling Mechanism
The awakened end does not detect the LP0f wake-up request initiated by the peer end.	The awakened end will automatically wake up after LP0f times out. Then, it detects a lane lock timeout or a link error exception, and initiates link retraining or link recovery through ERR_RM.
The awakened end does not detect the LP0, LP1, or LP2 wake-up request initiated by the peer end.	The lane lock handshake times out during the recovery stage of the low power wake-up initiating end, and then link retraining is initiated through ERR_RM.

6.3.3.8 Dynamic Lane State Switching

6.3.3.8.1 Basic Requirements

Different from low power state entry/exit of the entire link, dynamic lane state switching is mainly used in applications where high-speed data services are constantly flowing. Before and after the dynamic lane state switching, at least one lane shall be kept in a high-speed service transmission state.

Dynamic lane state switching includes the following operations:

- Dynamic lane increase: When the current port has at least one TX lane for service transmission, dynamically increase one to seven (up to eight) TX lanes.
- Dynamic lane reduction: When the current port has at least two TX lanes for service transmission, dynamically reduce one to seven TX lanes, to a maximum extent of just one TX lane left.
- Dynamic lane direction switching: Dynamically switch one to eight TX lanes to RX lanes.
- If dynamic adjustment of the number of lanes is not supported, initial link establishment shall be based on the maximum number of lanes supported, and the port link establishment is successful only when the link establishment of all lanes is successful.

Any link is prohibited from repeatedly initiating the dynamic lane state switching or link low power process when performing initial link training, link recovery, link low power, or dynamic lane state switching. When a link is performing the initial link training, link recovery, link low power, or dynamic lane state switching process, if the upper layer issues a new link low power or dynamic lane state switching operation, the logic will ignore the new operation, continue to perform the original operation, and report that the link is busy. The link low power entry, lane reduction/increase, and lane direction switching processes can only be initiated by the transmitting end of a link in HS state.

The initiating end is prohibited from requesting a link in a non-HS state to enter the low power state or dynamically switch the lane state. Any such illegal request will be ignored by the receiving end.

The operations of dynamic link width switching and dynamic lane direction switching are initiated by the TX end of a port, that is, a port can only actively manage the width and direction of the local TX link. When a link is performing the initial link training, link recovery, link low power, or dynamic lane state switching process, if the upper layer issues a new link low power or dynamic lane state

switching operation, the logic will ignore the new operation, continue to perform the original operation, and report that the link is busy. The link low power entry, lane reduction/increase, and lane direction switching processes can only be initiated by the transmitting end of a link in HS state.

The initiating end is prohibited from requesting a link in a non-HS state to enter the low power state or dynamically switch the lane state. Any such illegal request will be ignored by the receiving end.

Note: To simplify the lane state switching process, complex lane state switching shall be simplified by the upper layer to atomic link operations consisting of multiple times of lane width switching/lane direction switching. Additionally, to accelerate the execution of the lane state switching process, the main link should be used for handshake information interaction as far as possible.

6.3.3.8.2 Dynamic Lane Reduction

After link training is completed, the LNSM (TX) of each valid lane is in HS state. When the port input bandwidth becomes smaller, some lanes can be controlled by the upper layer to enter LP0–LP2 or disable state.

If the process of dynamic lane reduction is performed multiple times for a link, lanes shall enter the low power state at the same gear in each process, that is, all reduced lanes are only allowed to be in disable or a specific low power state (at a gear within LP0–LP2).

In the process of dynamically reducing the number of lanes multiple times, if the gear (LP0–LP2) indicated by the LP_MODE field in a subsequent LWAM is inconsistent with the gear (also LP0–LP2) indicated by the LP_MODE field in the previous LWAM, the receiving end will return a Nack message for the subsequent LWAM to reject the illegal request.

The following is a constraint description based on an example where the initial state of device A is that four TX lanes at the transmitting end are in a service transmitting state, and later the dynamic lane reduction process is continuously executed twice.

(1) Device A transitions to the 2TX mode after the lane reduction process, and the low power state for the two lanes to be disabled as indicated in the LWAM is recorded as LP-M (a gear within LP0–LP2).

(2) After the above lane reduction process is completed, device A intends to transition to the 1TX mode through the lane reduction process. In this case, the LP_MODE field in the LWAM only allows the indication of 1 lane to be disabled to enter the LP-M state (consistent with the low power gear indicated by the previous LWAM) or disable state.

In addition, after the dynamic lane reduction process is executed for the current link multiple times, if some of the reduced lanes are disabled, while the remaining reduced lanes are at the same specific low power gear, each subsequent lane increase process can only be initiated for the lanes in the same state. That is, the lane increase process can be separately executed for lanes in disable state or for those in the same specific low power state. It is prohibited to simultaneously initiate a lane increase request for lanes at different gears. Otherwise, the receiving end will return a Nack message for the LWAM to reject the illegal request. When only some lanes need to be increased for the current link, it is recommended that the lanes to be increased should be those that were previously in a specific low power state, to shorten the duration of the lane increase process.

Take device A in initial state as an example. The initial state is that device A has four TX lanes, with lane0-TX in service transmitting state (HS), lane1-TX in disable state, and lane2-TX and lane3-TX in LP1 state. In this state, when an LWAM is initiated for the first time, requesting for executing the lane increase process, the process can only be executed in one of the following methods:

(1) If device A initiates an LWAM to increase one lane, lane1, lane2, or lane3 can be selected to execute the lane increase process.

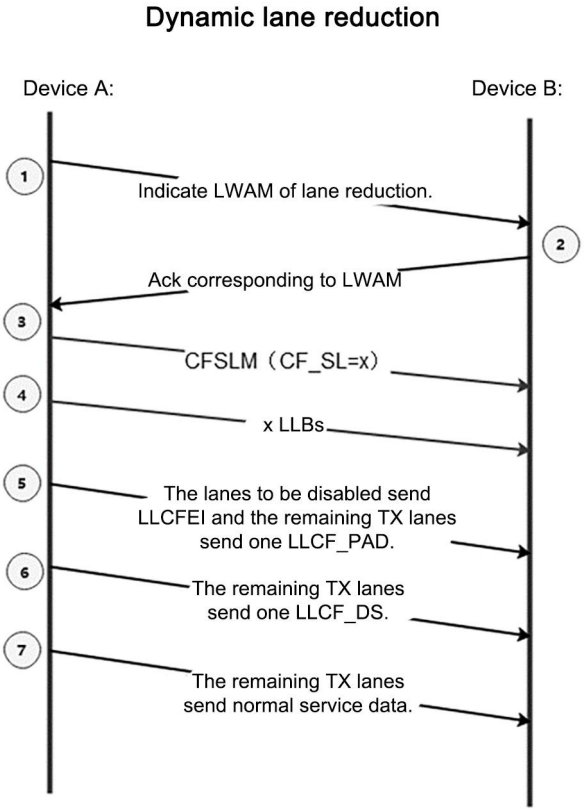
(2) If device A initiates an LWAM to increase two lanes, only lane2 and lane3 can be selected to execute the lane increase process.

If a link enters the low power state through the lane reduction process, it cannot return to the HS state through the low power exit process.

Take device A in initial state as an example. The initial state is device A has four TX lanes, with lane0/lane1-TX entering the LP1 state through the low power process, and lane2/lane3-TX entering the LP1 state through the lane reduction process. In this state, when the low power exit process is initiated, only lane2/lane3 returns to the HS state.

The dynamic lane reduction process is shown in Figure 59.

Figure 59 Dynamic lane reduction



(1) Device port A transmits an LWAM to device B, informing device B of the serial numbers of the lanes to be dynamically disabled and the low power state that these lanes will enter.

(2) After the RX end of device port B receives the LWAM, it feeds back an Ack or a Nack message to respond to the LWAM.

If the corresponding port of device B supports the dynamic link width switching, device B feeds back an Ack message. Otherwise, it feeds back a Nack message.

If device B feeds back a Nack message, the process ends. Device A and device B transmit and receive services based on the original number of lanes.

(3) After the port of device A receives the Ack message responding to the LWAM, device A transmits one CFSLM to device B and specifies CF_SL as x in the CFSLM.

(4) Device A transmits x LLBs to device B.

(5) After the lanes to be disabled of device A transmit four LLCF_EI control frames, they enter the low power state indicated in the LWAM.

After the lanes to be disabled of device B receive at least one LLCF_EI, they enter the low power state indicated in the previous LWAM. All remaining TX lanes in the high-speed state simultaneously transmit one LLCF_PAD (the PAD length transmitted by all TX lanes is cf_pad_length), to facilitate the state synchronization of local and peer data during multi-lane switching.

Note: The RX lane of device B needs to reset the scrambling seed after receiving the LLCF_PAD. The remaining high-speed TX lanes of device A also need to reset the scrambling seed after transmitting the LLCF_PAD.

(6) The port of device A simultaneously transmits one LLCF_DS through all remaining TX lanes in the high-speed state.

Device B re-identifies the starting position of the data frame according to the LLCF_DS and re-completes the deskew among lanes.

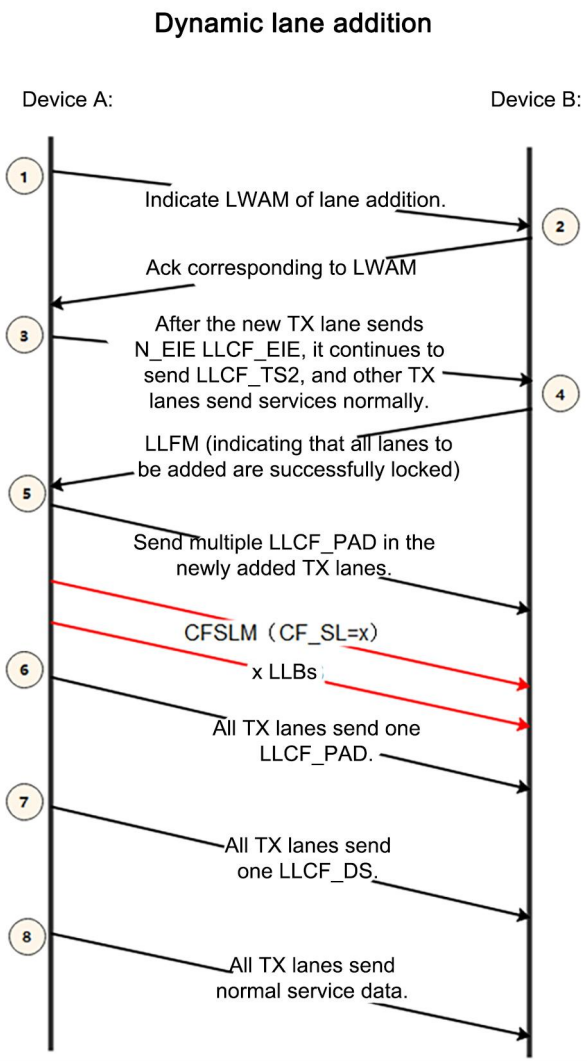
(7) The port of device A transmits normal services at the new link width.

6.3.3.8.3 Dynamic Lane Increase

After the bandwidth becomes larger, lanes in low power or disable state can be quickly switched back to HS state under the control of the upper layer to quickly restore the transport layer link bandwidth.

The dynamic lane increase process is shown in Figure 60.

Figure 60 Dynamic lane increase



(1) Device port A transmits an LWAM to device B, informing device B of the serial numbers of the lanes to be dynamically enabled.

(2) After the RX end of device port B receives the LWAM, it feeds back an Ack or a Nack message to respond to the LWAM.

If the corresponding port of device B supports the dynamic link width switching, device B feeds back an Ack message. Otherwise, it feeds back a Nack message.

If device B feeds back a Nack message, the process ends. Device A and device B transmit and receive services based on the original number of lanes.

(3) After the port of device A receives the Ack message responding to the LWAM, if the lanes to be awakened are in LP0–LP2 state, the port transmits N_EIE LLCF_EIE first and then continuously transmits LLCF_TS2. If the lanes to be awakened are in disable state, the port transmits TSM, as well as LLCF_TS0, LLCF_TS1, and LLCF_TS2 for lane training (if fast wake-up is supported, only LLCF_TS2 needs to be transmitted). During this process, other TX lanes transmit services normally.

After device B feeds back an Ack message responding to the LWAM, if the lanes to be awakened are in LP0-LP2 state, all lanes shall detect the non-electrical idle indication. (Lanes in LP0–LP2 state at the receiving end shall maintain the electrical idle detection feature. When the transmitting end transmits LLCF_EIE, the receiving end shall be able to detect the non-electrical idle indication.) Then, the low power wake-up process can be executed.
If the lanes to be awakened are in disable state, all lanes shall detect the TSM indicating lane start. Then, the low power wake-up process can be executed.

Note: The example is a scenario where the new TX lane of port A is awakened from the LPx state. If it is awakened from the disable state, this new lane needs to transmit LLCF_TS0, LLCF_TS1, and LLCF_S2 for lane training (for a fast wake-up, only LLCF_TS2), and device A also needs to transmit a TSM to instruct the lane to start initial training. The TSM requires all new lanes to start link training. If the TSM does not cover all lanes that require training, it will cause lock timeout of untrained lanes, which will cause the dynamic lane increase process to fail. Device A and device B transmit and receive services based on the original number of lanes.

(4) After device B senses that the new RX lanes detect a non-electrical idle state or receive TSMs from the peer end, these lanes start training. After the port of device B detects that all new RX lanes have completed training, it transmits an LLFM to indicate that all new lanes are successfully locked.

Note 1: The example is a scenario where the new TX lanes of port A are awakened from the LPx state. If these new TX lanes of port A are awakened from the disable state, it is necessary to wait for the new lanes to complete the initial training process (feedback messages in this process may include CLFM, EQFM, and LLFM).

Note 2: If the training of any new lanes is unsuccessful, the process ends after device B feeds back the CLFM, EQFM, or LLFM indicating that the training of these lanes is unsuccessful. Device A and device B transmit and receive services based on the original number of lanes.

(5) After device A receives the LLFM indicating that all new lanes are successfully locked, device A transmits several LLCF_PAD control frames in the new TX lanes to align the CF transmitting positions of the new lanes with those of the original lanes in the high-speed state. At the same time, device A transmits one CFSLM to device B and specifies CF_SL as x in the CFSLM. Then, device A transmits x LLBs to device B.

(6) Subsequently, device A simultaneously transmits one LLCF_PAD through all remaining TX lanes in the high-speed state (the PAD length transmitted by all TX lanes is cf_pad_length), to facilitate the state synchronization of local and peer data during multi-lane switching.

Note: The RX lane of device B needs to reset the scrambling seed after receiving the LLCF_PAD.

(7) Device A simultaneously transmits one LLCF_DS through all TX lanes in the high speed state.

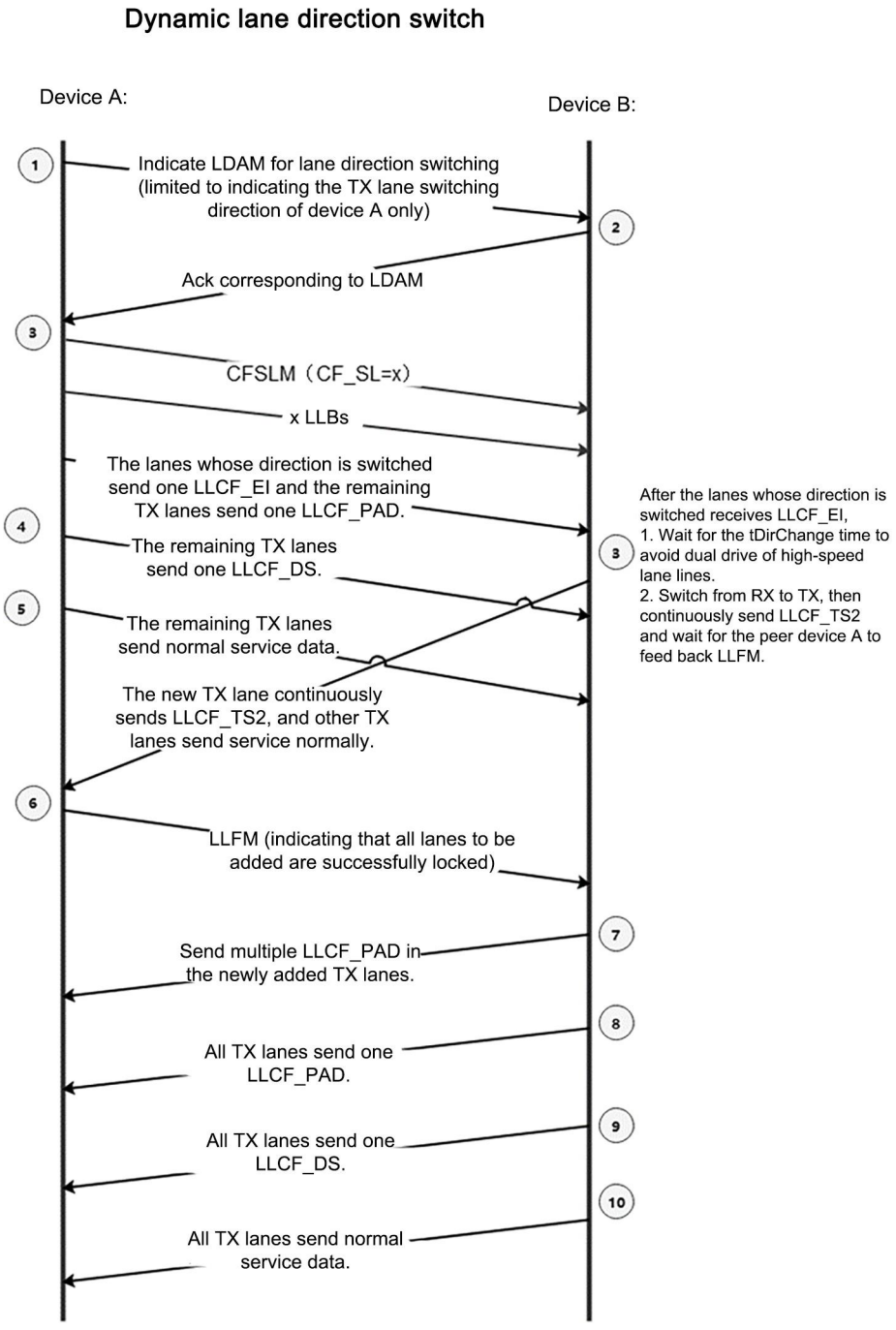
(8) The port of device A transmits normal services at the new link width.

6.3.3.8.4 Dynamic Lane Direction Switching (Informative)

After the link training is completed, if the TX link bandwidth at the local end is excessive, but the RX link bandwidth is insufficient, and the TX lane supports dynamic lane direction switching, the upper layer can control the direction switching of some TX lanes to compensate for the RX link bandwidth.

The process of dynamic lane direction switching is shown in Figure 61.

Figure 61 Lane direction switching



(1) Device port A transmits an LWAM to device B, informing device B of the serial numbers of the lanes to be dynamically switched.

Note: The lanes to be dynamically switched specified in the LDAM must be the TX lanes of device A.

(2) After the RX end of device port B receives the LDAM, it feeds back an Ack or a Nack message to respond to the LDAM.

If the corresponding port of device B supports the dynamic lane direction switching, device B feeds back an Ack message. Otherwise, it feeds back a Nack message.

If device B feeds back a Nack message, the process ends. Device A and device B transmit and receive services based on the original lane state.

(3) After the port of device A receives the Ack message responding to the LDAM, device A transmits one CFSLM to device B and specifies CF_SL as x in the CFSLM. Then, device A transmits x LLBs to device B. After the LLBs are completely transmitted and the lanes to be disabled of device A transmit four LLCF_EI control frames, LNSM (TX) enters the disable state and disables the TX lanes. All remaining TX lanes in the high-speed state simultaneously transmit one LLCF_PAD (the PAD length transmitted by all TX lanes is cf_pad_length), to facilitate the state synchronization of local and peer data during multi-lane switching.

Note: Before the port of device A receives the Ack message responding to the LWAM, device A still transmits normal service flows through the port.

After the lane to be switched at the port of device B receives the LLCF_EI, the following process is executed:

- Wait for the tDirChange time to avoid bidirectional driving of high-speed lane lines and ensure the switching time of the two ends.
- The RX lane to be switched is disabled, switched to the TX mode, and enabled in turn. Then, the LLCF_TS2 is continuously transmitted (The awakened lane in this example supports rapid link establishment. If this lane does not support fast link establishment, the LLCF_TS0 shall be transmitted. Then, after the lane clock lock handshake is completed, the LLCF_TS1 and LLCF_TS2 are transmitted.) Device B also needs to transmit a TSM to require corresponding lanes to start initial training. Then, wait for device A at the peer end to feed back LLFM (or CLFM/EQFM).

(4) The port of device A simultaneously transmits one LLCF_DS through all remaining TX lanes in the high-speed state.

(5) The TX at the port of device A transmits normal services at the new link width.

(6) After the port of device A detects that all lanes switched from TX to RX have completed training, it transmits an LLFM (The awakened lane in this example supports rapid link establishment. If this lane does not support rapid link establishment, the CLFM and EQFM need to be transmitted for clock lock handshake.) to indicate that all new lanes are successfully locked.

If the LLFM indicates that any new lane is not locked, the process ends. Device A and device B transmit and receive services based on the number of lanes transmitting services currently (that is, the number of TX lanes of device A has been reduced). If the lane direction adjustment covers all lanes, the number of TX lanes of device A will be reduced to 0.

(7) After device B receives the LLFM indicating that all new lanes are successfully locked, device B transmits several LLCF_PAD control frames in the new TX lanes to align the CF transmitting positions of the new lanes with those of the original lanes in the high-speed state.

(8) Device B simultaneously transmits one LLCF_PAD through all remaining TX lanes in the high-speed state (the PAD length is cf_pad_length), to facilitate the state synchronization of local and peer data during multi-lane switching.

Note: The RX lane of device A shall reset the scrambling seed after receiving the LLCF_PAD.

(9) Device B simultaneously transmits one LLCF_DS through all TX lanes in the high speed state.

(10) The port of device B transmits normal services at the new link width.

6.3.3.8.5 Exception Handling Mechanism for Dynamic Lane State Switching Process

Table 76 Troubleshooting message loss in the dynamic lane state switching process

Exception	Handling Mechanism
Loss/Exception of LWAM or LDAM requests for dynamic lane state switching	This type of request message requires the responder to feed back a response message. If the response message is lost/abnormal, the LWAM or LDAM needs to be retransmitted. (The exception handling mechanism is the same as that described in 6.3.6.)
Loss/Exception of the message responding to the LWAM or LDAM for dynamic lane state switching	The requester retransmits the LWAM or LDAM, and the responder feeds back an Ack or a Nack message after receiving the LWAM or LDAM each time. After the requester fails to receive the corresponding response message after multiple retransmissions, link retraining is initiated through ERR_RM.
LLCF_EI loss/exception	The responder of the dynamic lane switching command (the port receiving the LDAM) detects a link error exception and initiates link retraining or link recovery through ERR_RM. Note: If the receiving end detects any frame header in the LLCF_EI format at the position where an LLCF_EI is received, the LLCF_EI has been recognized.
LLCF_DS detection failure	The receiving end handles the deskew exception according to the Recovery state in the RXLKSM. Link retraining is initiated through ERR_RM.
LLCF_TS2 detection failure at the receiving end of the awakened lane	The receiving end handles the lane lock exception according to the Recovery state in the LNSM. Link retraining is initiated through ERR_RM.
Lane lock timeout at the generating end of an awakened lane	The receiving end handles the lane lock exception according to the Recovery state in the TXLKSM. Link retraining is initiated through ERR_RM.

6.3.3.8.6 Conflict Handling Mechanism for Lane State Switching Process

On one link, only one process can be executed at a time. Process conflict refers to a scenario where different ports simultaneously initiate process requests for one link. When a process conflict occurs on a link, a device process shall be rejected. The specific handling rules are as follows:

- A link exception fed back through an ERR_RM has the highest priority. The priority of link retraining is higher than that of link recovery.
- A unidirectional link entering LP0f has the second-highest priority.
- The priority of a process of the master device (LMP) is higher than that of the slave device (LSP). If the LMP does not receive a response after initiating a process, but receives a process request packet from the peer end, the LMP needs to transmit a Nack message to reject the request from the peer end.

- The peer end needs to actively identify whether the rejection is a process conflict and can independently decide whether to re-initiate the process after waiting for a period of time.

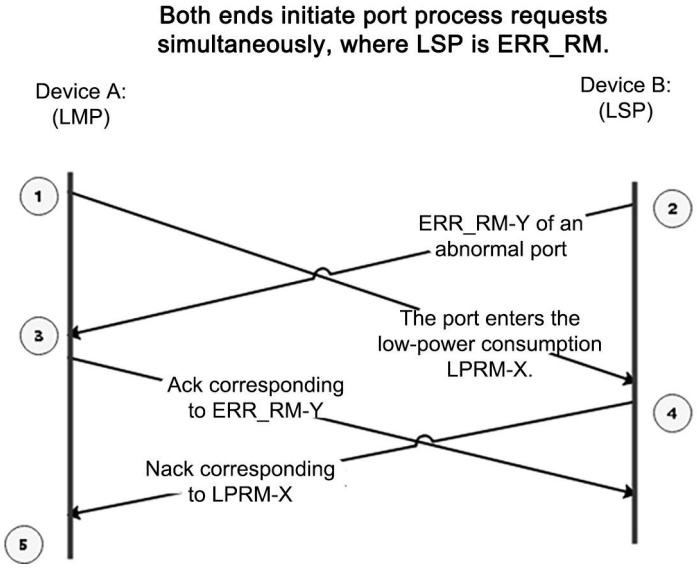
In case of process conflicts, process priorities shall meet the requirements in Table 77.

Table 77 Conflict handling priorities

LSP	**LMP**					
	Lane Increase/Reduction	**Lane Direction Switching**	**LP0f Request**	**LP0–3 Request**	**Err-Init**	**Err-Recovery**
Lane increase/reduction	LMP	LMP	LMP	LMP	LMP	LMP
Lane direction switching	LMP	LMP	LMP	LMP	LMP	LMP
LP0f request	LSP	LSP	NA	LSP	LMP	LMP
LP0–3 request	LMP	LMP	LMP	LMP	LMP	LMP
Err-Init	LSP	LSP	LSP	LSP	NA	LSP
Err-Recovery	LSP	LSP	LSP	LSP	LMP	NA
Note: NA means no conflict among processes, and the processes can be handled simultaneously, regardless of their priorities.						

Example: A specific process is shown below (take a scenario where the LMP transmits an LPRM and the LSP transmits an ERR_RM as an example):

Figure 62 Schematic diagram of handling process requests simultaneously transmitted by both ends



(a) Device A (LMP) transmits a port low power request LPRM-X to device B (LSP).

(b) Device B transmits a link exception request ERR_RM-Y to device A before it receives the LPRM-X transmitted by device A.

(c) After device A receives the ERR_RM sent by device B, if it finds that both ends initiate process requests at the same time and the process of device B has a higher priority, device A needs to feed back an Ack message responding to the ERR_RM to device B and terminate the LPRM-X process of device A. After device A transmits the Ack message responding to the ERR_RM, it enters the ERR_RM process handling state.

Note: After the LMP finds that the current processes conflict with each other, it has no need to wait for the LPRM-X handshake process to end. If it does not receive the message responding to the LPRM-X, it will not retransmit the LPRM-X.

(d) After device B receives the LPRM-X from device A, if it finds that both ends initiate process requests at the same time and the process of device B has a higher priority, it needs to feed back a Nack message responding to the LPRM to device A.

(e) Note: In process conflict scenarios, high-priority devices must reject requests from low-priority devices.

(f) Device A receives the Nack message responding to the LPRM-X and ignores the Nack message.

(g) After device B receives the Ack message responding to the ERR_RM, it enters the ERR_RM process handling state.

6.3.4 Coding/Decoding

6.3.4.1 Overview

This section describes the main link architecture of the logical layer, which consists of two parts: transmitter link (TX link) and receiver link (RX link). As shown in Figure 63, at the transmitting end, the logical layer performs multiplexing, FEC coding, lane distribution, scrambling, and precoding for the data transmitted to it from the transport layer, and then transmits these data to the electrical layer. At the receiving end, the logical layer performs deprecoding, descrambling, lane combination, FEC decoding, error correction, and demultiplexing for the data transmitted to it from the electrical layer, and then transmits these data to the transport layer. The details are as follows:

Multiplexing and demultiplexing: They realize LLDP and LLMMP encapsulation and parsing, and the conversion between data packets and LLBs.

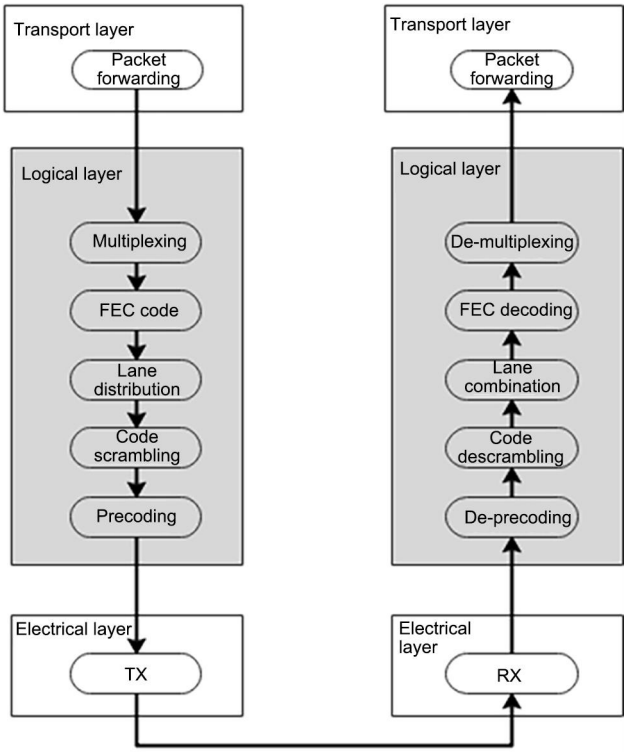
FEC coding and decoding: They realize the conversion between LLBs and FEC coding/decoding work spaces, and realize coding and decoding of data in the FEC coding/decoding work spaces.

Lane distribution and combination: They realize data distribution and combination on each lane according to the enabled lane configurations.

Scrambling and descrambling: Each lane independently realizes data scrambling and descrambling.

Precoding and deprecoding: Each lane independently realizes data precoding and deprecoding.

Figure 63 Main link structure of the logical layer



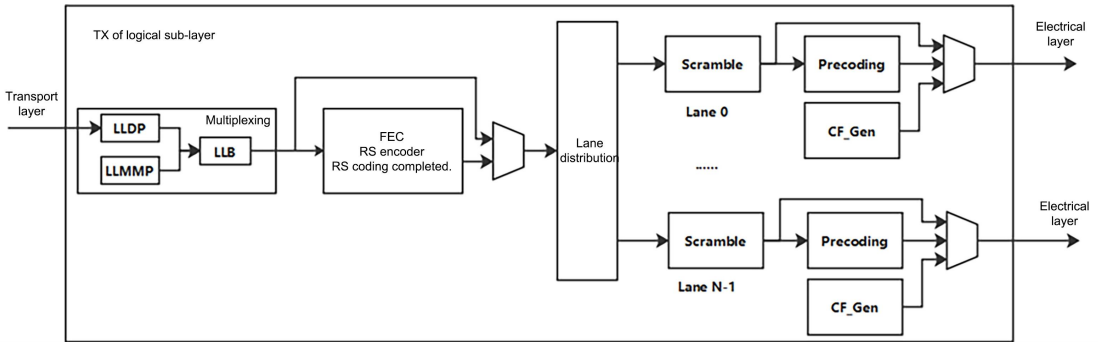
6.3.4.2 Logical Architecture

6.3.4.2.1 TX Link

The TX link receives, encodes, and forwards transport layer data, including multiplexing, FEC coding, and lane distribution. It has N TX lanes, and each lane has an independent scrambling module and a precoding module. The TX link supports eight TX lanes at most and one TX lane at least. The overall architecture is shown in Figure 64. The number of lanes shown in the figure is equal to the number of current link lanes.

At the transmitting end, the TX link completes the encapsulation and blocking of transport layer packets and logical layer management packets first. Then, it completes FEC coding, and evenly distributes LLBs to each lane. Finally, it completes data scrambling and precoding on each lane and transmits the scrambled and precoded data to the electrical layer. On the logical layer, FEC coding and precoding at the transmitting end shall support configurable enabling. For example: In scenarios with low bit error rates, FEC coding can be disabled. In scenarios where bit errors are mainly random errors, precoding can be disabled.

Figure 64 TX link architecture

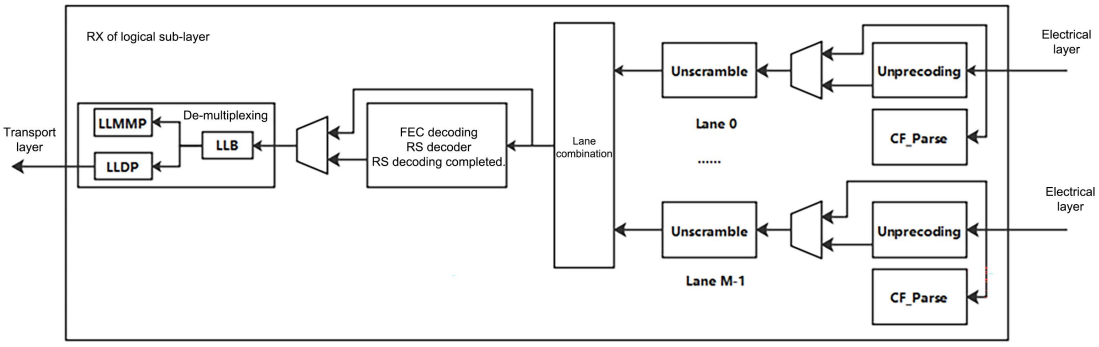


6.3.4.2.2 RX Link

The receiving end of the logical layer realizes the collection, decoding, and forwarding of electrical layer data, including demultiplexing, FEC decoding, and lane combination. It has M RX lanes, and each lane has an independent descrambling module and a deprecoding module. The RX link supports eight lanes at most and one lane at least. The number of RX link lanes must be consistent with the number of TX link lanes at the peer port. The overall architecture is shown in Figure 65. The number of lanes shown in the figure is equal to the number of current link lanes.

At the receiving end, each lane independently completes the acquisition, deprecoding, and descrambling of electrical layer data first. Then, the data from all lanes is combined to form LLBs, and FEC decoding and error correction are performed. Finally, LLBs are parsed to obtain logical layer management packets and transport layer packets. On the logical layer, FEC decoding and deprecoding at the receiving end shall support configurable enabling, and the configurations must be consistent with the FEC coding and precoding enabling configurations at the transmitting end.

Figure 65 RX link architecture



6.3.4.2.3 Port Link Architecture

Both bidirectional and unidirectional port link architectures are supported. A bidirectional link architecture contains a TX link and an RX link. A unidirectional link architecture only contains a TX link or an RX link. Based on different port types and lane modes, the logical sublayer has different link architectures. Based on the port type, LMP, LSP, and DRP are included. See T/SUCA 001.1 7.1.

(a) Four-lane mode

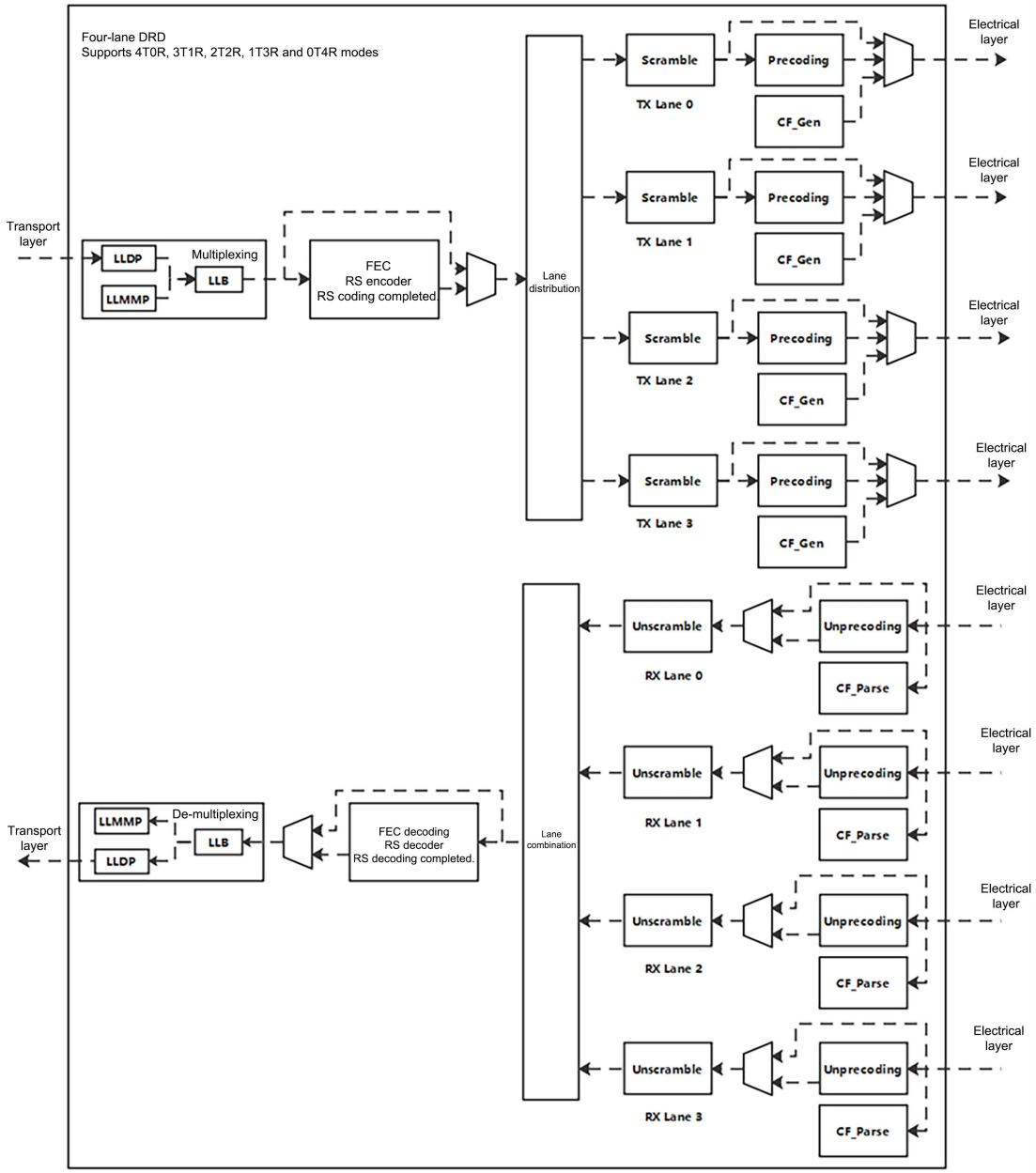
A four-lane LMP has a total of four lanes, and the 4T0R lane mode is supported. The 4T0R mode means a unidirectional link, and supports a four-lane TX link but no RX link.

A four-lane LSP has a total of four lanes, and the 0T4R lane mode is supported. The 0T4R mode means a unidirectional link, and supports a four-lane RX link but no TX link.

A four-lane DRP has a total of four lanes, and the 4T0R and 0T4R modes are supported. The 4T0R and 0T4R modes mean unidirectional links.

As shown in Figure 66, the link architecture of a four-lane DRP includes a TX link and an RX link. The TX link contains four TX lanes and the RX_Link contains four RX lanes. In the figure, the enabling of paths in dotted lines is controlled by the lane mode. The 4T0R and 0T4R modes are supported.

Figure 66 Link architecture of a four-lane DRP port



(b) Eight-lane mode

An eight-lane LMP has a total of eight lanes, and the 8T0R lane mode is supported. The 8T0R mode means a unidirectional link, and supports an eight-lane TX link but no RX link.

An eight-lane LSP has a total of eight lanes, and the 0T8R lane mode is supported. The 0T8R mode means a unidirectional link, and supports an eight-lane RX link but no TX link.

An eight-lane DRP has a total of eight lanes, and the 8T0R and 0T8R modes are supported. The 8T0R and 0T8R modes mean unidirectional links.

6.3.4.3 Multiplexing and Demultiplexing

6.3.4.3.1 Overview

Multiplexing multiplexes and encapsulates transport layer packets to form LLBs, and inserts logical layer management packets into the LLBs according to link management requirements. Demultiplexing parses LLBs into LLDPs and LLMMPs according to the LLPH in LLBs, and then parses data packets and management packets.

The multiplexing and demultiplexing layer interacts with the transport layer packets, and processes logical layer data and management messages.

6.3.4.3.2 Multiplexing

6.3.4.3.2.1 Encapsulation

Encapsulation means encapsulating logical layer management packets and transport layer packets. For encapsulation methods, see 6.2.3 Logical Layer Packets. Based on the features of different management information, different logical layer management packets have different encapsulation priorities. Logical layer management packets are inserted into LLBs in order of encapsulation priority from high to low.

Specific rules for logical layer management packets are as follows:

- Exception reporting and CFSLMs have high priorities.
- Other logical layer management packets have low priorities, including link training start message, clock lock feedback message, equilibrium feedback message, lane lock feedback message, low power request message, link width adjustment message, lane direction adjustment message, and response message.

6.3.4.3.2.2 Blocking

The multiplexing layer fills the encapsulated data packets into LLBs byte by byte during encapsulation. The size of filled LLBs is related to the port type.

LLB filling shall meet the following requirements:

- A maximum of two LLMMPs can be filled into one LLB.
- One LLDP can be filled into one LLB.
- LLMMPs are preferentially filled into LLBs. After LLMMP filling is completed, LLDPs can be filled.

(a) Four-lane mode

In four-lane mode, when FEC is enabled, the first 230 bytes of LLBs are used to fill data packets, and the last 10 bytes of LLBs are fixedly filled with 10 bytes of 0x00 data, which will be replaced by

check codewords after the subsequent FEC coding is completed. When FEC is not enabled, 240-byte LLBs are all used to fill data packets.

For a four-lane port with FEC enabled, the structure of the LLB filled with one LLMMP is shown in Figure 67.

For a four-lane port with FEC enabled, the structure of the LLB not filled with any LLMMP is shown in Figure 68.

For a four-lane port with FEC disabled, the LLB structure is similar to the above two structures, just without an LLB tail. The RS parameter needs to be replaced by the packet payload data of LLDP, and the LLDP packet length is increased by 10.

Figure 67 LLB structure (one LLMMP inserted) of a four-lane link

Structure		RS coding area				
		0	1	2	3	4
Logical block body	LLMMP0	PH0	PH1	PP0	PP1	PP2
		PP3	CRC0	CRC1	PH0	PH1
	LLDP	PP0	PP1	PP2	PP3	PP4
		PP5	PP6	PP7	PP8	PP9
		• • • • • •				
		PP205	PP206	PP207	PP208	PP209
		PP210	PP211	PP212	PP213	PP214
		PP215	PP216	PP217	PP218	PP219
Logical block tail	RS	RS0[0]	RS1[0]	RS2[0]	RS3[0]	RS4[0]
		RS0[1]	RS1[1]	RS2[1]	RS3[1]	RS4[1]

Figure 68 LLB structure (no LLMMP inserted) of a four-lane link

Structure		RS coding area				
		0	1	2	3	4
Logical block body	LLDP	PH0	PH1	PP0	PP1	PP2
		PP3	PP4	PP5	PP6	PP7
		PP8	PP9	PP10	PP11	PP12
		PP13	PP14	PP15	PP16	PP17
		• • • • • •				
		PP213	PP214	PP215	PP216	PP217
		PP218	PP219	PP220	PP221	PP222
		PP223	PP224	PP225	PP226	PP227
Logical block tail	RS	RS0[0]	RS1[0]	RS2[0]	RS3[0]	RS4[0]
		RS0[1]	RS1[1]	RS2[1]	RS3[1]	RS4[1]

(b) **Eight-lane mode**

In eight-lane mode, the LLB construction method is similar to that in four-lane mode. When FEC is enabled, the first 460 bytes of LLBs are used to fill data packets, and the last 20 bytes of LLBs are fixedly filled with 20 bytes of 0x00 data, which will be replaced by check codewords after the subsequent FEC coding is completed. When FEC is not enabled, 480-byte LLBs are all used to fill data packets.

In eight-lane mode, with FEC enabled, the structure of the LLB filled with two LLMMPs is shown in Figure 69.

In eight-lane mode, with FEC enabled, the structure of the LLB not filled with any LLMMP is shown in Figure 70.

In eight-lane mode, with FEC disabled, the LLB structure is similar to the above two structures, just without an LLB tail. The RS parameter needs to be replaced by the packet payload data of LLDP, and the LLDP packet length is increased by 20.

Figure 69 LLB structure (two LLMMPs inserted) of an eight-lane link

Structure		RS coding area									
		0	1	2	3	4	5	6	7	8	9
Logical block body	LLMMP0	PH0	PH1	PP0	PP1	PP2	PP3	CRC0	CRC1	PH0	PH1
	LLMMP1	PP0	PP1	PP2	PP3	CRC0	CRC1	PH0	PH1	PP0	PP1
	LLDP	PP2	PP3	PP4	PP5	PP6	PP7	PP8	PP9	PP10	PP11
		PP12	PP13	PP14	PP15	PP16	PP17	PP18	PP19	PP20	PP21
		• • • • • •									
		PP412	PP413	PP414	PP415	PP416	PP417	PP418	PP419	PP420	PP421
		PP422	PP423	PP424	PP425	PP426	PP427	PP428	PP429	PP430	PP431
		PP432	PP433	PP434	PP435	PP436	PP437	PP438	PP439	PP440	PP441
Logical block tail	RS	RS0[0]	RS1[0]	RS2[0]	RS3[0]	RS4[0]	RS5[0]	RS6[0]	RS7[0]	RS8[0]	RS9[0]
		RS0[1]	RS1[1]	RS2[1]	RS3[1]	RS4[1]	RS5[1]	RS6[1]	RS7[1]	RS8[1]	RS9[1]

Figure 70 LLB structure (no LLMMP inserted) of an eight-lane link

Structure		RS coding area									
		0	1	2	3	4	5	6	7	8	9
Logical block body	LLDP	PH0	PH1	PP0	PP1	PP2	PP3	PP4	PP5	PP6	PP7
		PP8	PP9	PP10	PP11	PP12	PP13	PP14	PP15	PP16	PP17
		PP18	PP19	PP20	PP21	PP22	PP23	PP24	PP25	PP26	PP27
		PP28	PP29	PP30	PP31	PP32	PP33	PP34	PP35	PP36	PP37
		• • • • • •									
		PP428	PP429	PP430	PP431	PP432	PP433	PP434	PP435	PP436	PP437
		PP438	PP439	PP440	PP441	PP442	PP443	PP444	PP445	PP446	PP447
		PP448	PP449	PP450	PP451	PP452	PP453	PP454	PP455	PP456	PP457
Logical block tail	RS	RS0[0]	RS1[0]	RS2[0]	RS3[0]	RS4[0]	RS5[0]	RS6[0]	RS7[0]	RS8[0]	RS9[0]
		RS0[1]	RS1[1]	RS2[1]	RS3[1]	RS4[1]	RS5[1]	RS6[1]	RS7[1]	RS8[1]	RS9[1]

6.3.4.3.3 De-multiplexing

6.3.4.3.3.1 Deblocking

Deblocking splits data in LLBs into LLMMPs and LLDPs.

To ensure the timeliness of logical layer management features, it is necessary to identify whether LLMMPs are included in LLBs during FEC decoding. If yes, the identified LLMMPs in LLBs shall be transmitted to the decapsulation layer for parsing. The specific rules are as follows:

(a) Check the first byte of an LLB. If this byte is an LLDP, the LLB does not have an LLMMP. If this byte is an LLMMP, the LLB contains LLMMP0. If the CRC of this LLMMP0 is correct, it will be transmitted to the decapsulation layer for parsing.

(b) When LLMMP0 exists, check the ninth byte of the LLB. If this byte is an LLDP, the LLB does not have LLMMP1. If this byte is an LLMMP, the LLB contains LLMMP1. If the CRC of this LLMMP1 is correct, it will be transmitted to the decapsulation layer for parsing.

After FEC decoding, deblocking of the LLB shall be performed again. The deblocking rules are the same as those before FEC decoding. The specific rules are as follows:

(a) If LLMMP0 is identified before FEC decoding and the CRC is correct, the first eight bytes of data of the LLB are directly dropped during deblocking after FEC decoding.

(b) If LLMMP0 is identified before FEC decoding and the CRC is incorrect, the LLB will be deblocked from byte 0 after FEC decoding. If the CRC is correct, LLMMP0 will be transmitted to the decapsulation layer for parsing.

(c) If LLMMP0 and LLMMP1 are identified before FEC decoding and both CRCs are correct, the first 16 bytes of data of the LLB are directly dropped during deblocking after FEC decoding.

(d) If LLMMP0 and LLMMP1 are identified before FEC decoding, and the CRC of LLMMP0 is correct, but the LLMMP1 parsing fails, the first eight bytes of data of the LLB are directly dropped during deblocking after FEC decoding, and deblocking starts from the ninth byte of the LLB.

(e) If LLMMP0 and LLMMP1 are identified before FEC decoding, and the CRC of LLMMP0 is incorrect, LLMMP1 will not be parsed, and after FEC decoding, deblocking will start from byte 0 of the LLB.

6.3.4.3.3.2 Decapsulation

Decapsulation realizes the processing of all data packets in LLBs, and performs different processing according to the type of data packet.

For LLDPs, after the packet header check is completed through decapsulation, the packet header data is directly dropped, and the packet payload data is transmitted to the transport layer.

For LLMMPs, after the data packet check is completed through decapsulation, message parameters are obtained to realize the logical layer management feature.

6.3.4.4 FEC Coding and Decoding

6.3.4.4.1 Overview

This section describes FEC coding specifications, coding methods for the transmitting end, decoding (check and error correction) methods for the receiving end, and performance.

For the main link, Reed-Solomon (48, 46) codes based on GF (256), that is, RS (48, 46) codes, are used. As shown in Figure 71, each RS work space (RS_ws) contains 46 bytes of data and two bytes of placeholders (for check information filling).

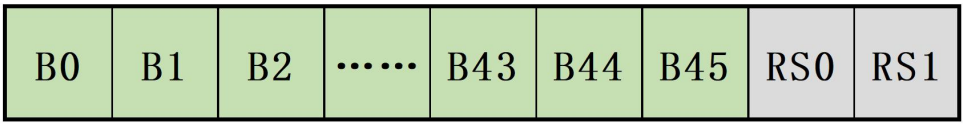


Figure 71 Schematic diagram of RS work spaces

For different link architectures, the number of RS work spaces contained in the FEC coding/decoding layer is different. In a four-lane port link architecture, the FEC coding/decoding layer fixedly contains five groups of RS work spaces to realize the coding and decoding of 230 bytes of LLBs.

6.3.4.4.2 RS Coding/Decoding

The FEC coding/decoding layer uses RS (48, 46) for coding and decoding. The primitive polynomial is as follows:

$$p(x) = x^8 + x^4 + x^3 + x^2 + 1 \quad (13)$$

The generator polynomial is as follows:

$$g(x) = x^2 + 3x + 2 \quad (14)$$

In the coding area, RS codes 46 bytes of data in the RS block to generate two bytes of check information. In the decoding area, RS verifies and corrects the 48 bytes of data and check information in the RS block to correct errors of any one byte of data in the RS block.

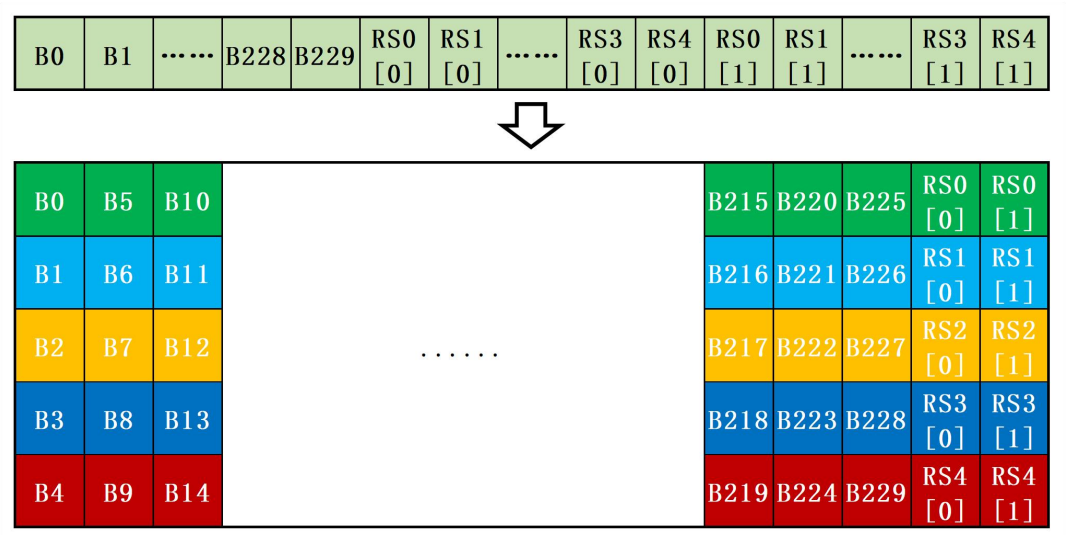
6.3.4.4.3 RS Mapping

Before RS coding/decoding, the transmitted LLBs need to be mapped into RS blocks, and then coded and decoded in the RS work spaces.

(a) Four-lane mode

For a four-lane link, one LLB has a total of 240 bytes which are evenly arranged in five RS blocks in byte order. Each RS block performs RS coding and decoding independently in its RS work space. The mapping relationship from LLBs to RS blocks is shown in Figure 72.

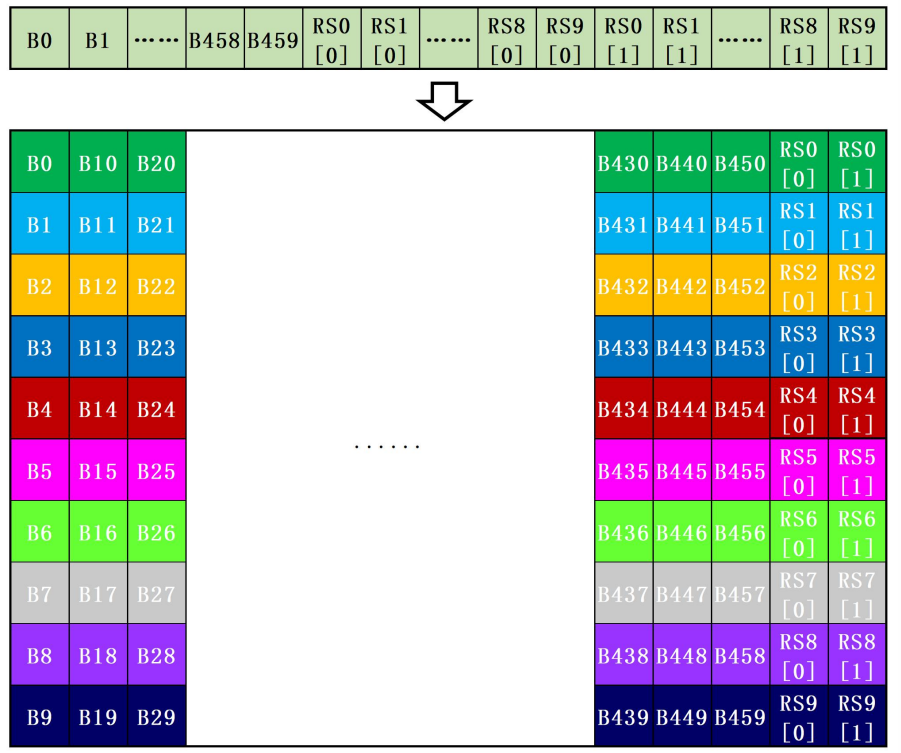
Figure 72 RS mapping of a four-lane port link



(b) Eight-lane mode

For an eight-lane link, one LLB has a total of 480 bytes which are evenly arranged in 10 RS blocks in byte order. Each RS block performs RS coding and decoding independently in its RS work space. The mapping relationship from LLBs to RS blocks is shown in Figure 73.

Figure 73 RS mapping of an eight-lane port link



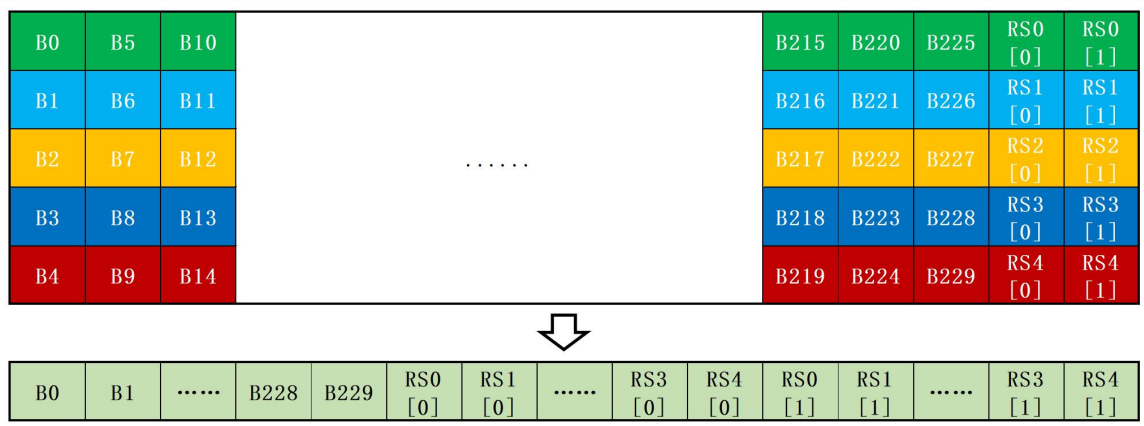
6.3.4.4.4 RS Inverse Mapping

After RS coding and decoding are completed, RS blocks need to be inversely mapped into LLBs.

(a) Four-lane mode

A four-lane link contains five RS blocks, and the data in these RS blocks are reversely mapped byte by byte to the transmitted LLBs according to the order of RS work spaces. In the coding area, the first 230 bytes in LLBs after reverse mapping are the same as the LLBs before RS mapping, and the last 10 bytes in LLBs are replaced by five groups of RS check data. In the decoding area, the first 230 bytes in LLBs after inverse mapping are error-corrected LLBs, and the last 10 bytes in LLBs are five groups of RS check data which will not be used after RS error correction. The mapping relationship from RS blocks to LLBs is shown in Figure 74.

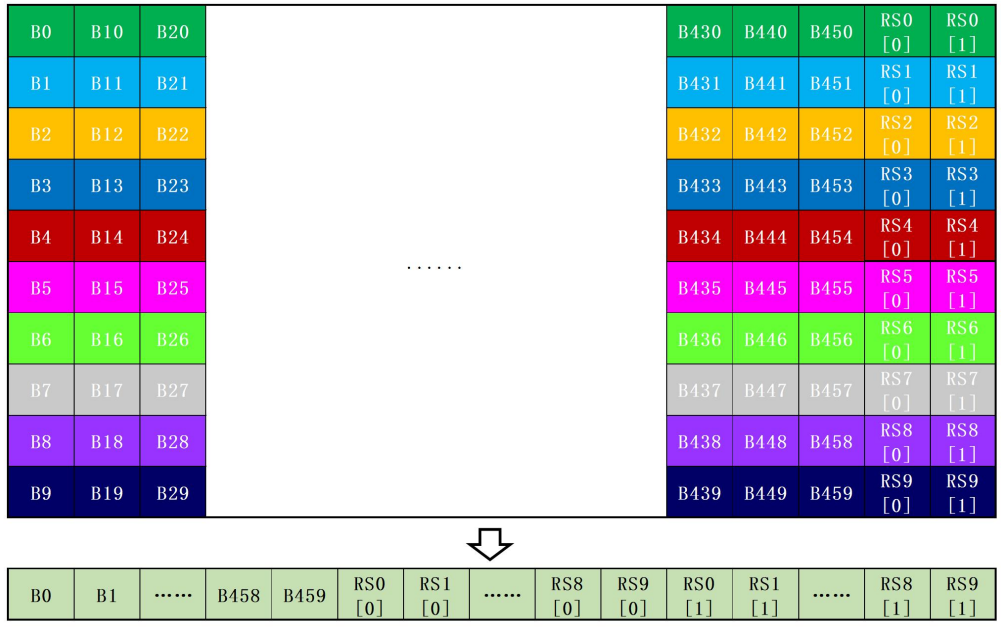
Figure 74 RS inverse mapping of a four-lane port link



(b) Eight-lane mode

An eight-lane link contains 10 RS blocks, and the data in these RS blocks are reversely mapped byte by byte to the transmitted LLBs according to the order of RS work spaces. In the coding area, the first 460 bytes in LLBs after reverse mapping are the same as the LLBs before RS mapping, and the last 20 bytes in LLBs are replaced by 10 groups of RS check data. In the decoding area, the first 460 bytes in LLBs after inverse mapping are error-corrected LLBs, and the last 20 bytes in LLBs are 10 groups of RS check data which will not be used after RS error correction. The mapping relationship from RS blocks to LLBs is shown in Figure 75.

Figure 75 RS inverse mapping of an eight-lane port link



6.3.4.5 Lane Distribution and Combination

6.3.4.5.1 Overview

When the logical layer supports the four-lane mode, it supports the enabling of four lanes at most.

When the logical layer supports the eight-lane mode, it supports the enabling of eight lanes at most.

Lane distribution and combination realize the mapping between LLBs and lane data based on the enabled lanes. At the transmitting end, the lane distribution feature is implemented to evenly distribute LLB data byte by byte to enabled lanes for transmission. At the receiving end, the lane combination feature is implemented to combine lane data byte by byte into LLBs according to the logical order of lanes.

All data between LLCF_DS and the next LLCF is a logical layer data frame (LLDF) structure, containing a finite number of complete LLBs. For lane allocation and combination, an LLDF is treated as a whole, and all data in an LLDF is continuously distributed and combined on each lane. Data distribution and combination shall ensure that the data transmission volume on each lane is the same. Otherwise, random data shall be filled to ensure that the LLCF code patterns transmitted on each lane after the LLDF are aligned. Figure 76 shows a transmission scenario with no random data filled for alignment. LLCF_XX is a control frame structure. Figure 77 shows a transmission scenario where random data is filled for alignment.

All LLDFs are independently interleaved, and the interwoven arrangement corresponds to the number of lanes for the current LLDF transmission.

Figure 76 Transmission scenario with no random data filled for alignment

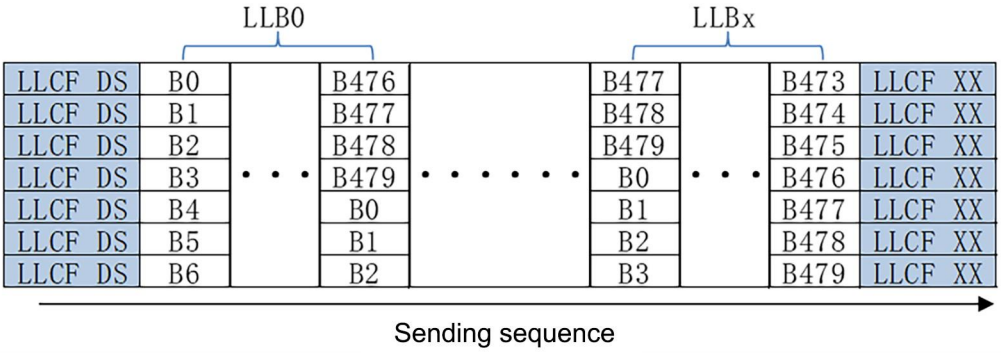
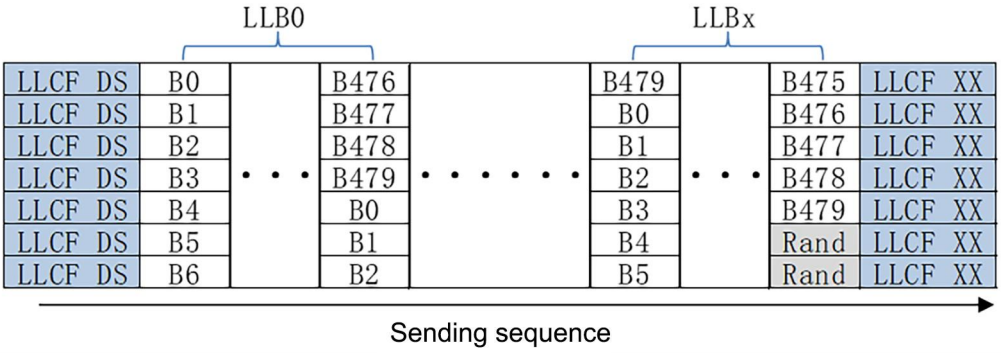
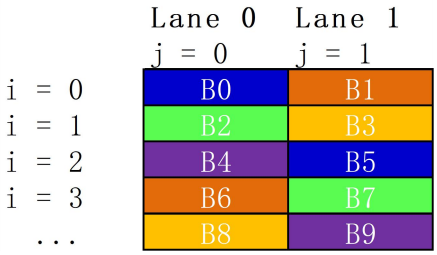


Figure 77 Transmission scenario with random data filled for alignment



The interwoven arrangement of byte data on lanes can be represented by a table, as shown in Figure 78.

Figure 78 Schematic diagram of interwoven arrangement



The above figure shows the data transmission sequence of the first LLB. Lane 0 transmits data B0 and lane 1 transmits data B1. During the time when the second byte data is transmitted, lane 0 transmits data B2 and lane 1 transmits data B3, and so on. Take i and j as coordinate rows and columns, and v as the subscript number of byte data at the corresponding position. As shown in the figure above, the data at the position where i is 2 and j is 0 is B4, and its subscript number is 4.

Generally, for a position at row i and column j in the coordinate, the data number v is calculated as follows:

g = gcd(N,M)

n = N/g

m = M/g

k = i*N + j

v = Int(k / g) * g + (((Int(Int(k / N) / m) Mod g) * (g - 1) + (k Mod g)) Mod g)

Where,

- N is the total number of current lanes, which is 1 to 4 for a four-lane scenario and 1 to 8 for an eight-lane scenario.
- M is the current number of interleaved coding areas, which is 5 for a four-lane port scenario and 10 for an eight-lane port scenario.
- i is the line number, starting from 0, with a step of 1.
- j is the column number, starting from 0, with a step of 1.
- v is the data number of a position at row i and column j.
- g, n, m, and k are the intermediate calculation results of auxiliary calculation.
- gcd represents the greatest common divisor, Int represents rounding down, and x Mod y represents x modulo y.

6.3.4.5.2 Interleaving in Four-Lane Mode

The four-lane mode supports the enabling of four lanes at most.

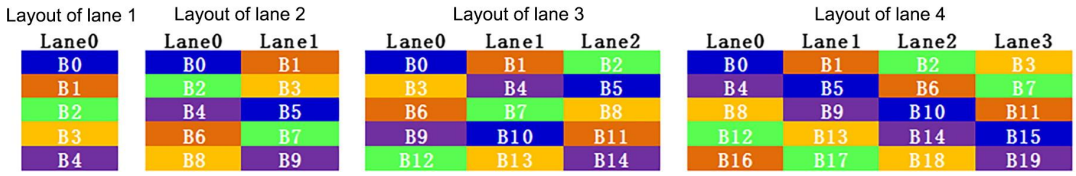
The data interwoven arrangements corresponding to enabling one to four lanes are shown in Figure 79. The arrangement is cyclic after every five data on each lane. The figure shows the minimum cyclic interleaved arrangements. The specific implementation is as follows:

(a) All LLB data is distributed byte-by-byte to each enabled lane in byte order.

(b) Data is distributed in lane order from low to high.

(c) All LLDF data is distributed according to the above rules, and the next LLDF is restored to the initial distribution state.

Figure 79 Interwoven arrangements for a four-lane port



6.3.4.5.3 Interleaving in Eight-Lane Mode

The eight-lane mode supports the enabling of eight lanes at most.

The data interwoven arrangements corresponding to enabling one to eight lanes are shown in Figure 80. The arrangement is cyclic after every 10 data on each lane. The figure shows the minimum cyclic interleaved arrangements. The specific implementation is as follows:

(a) All LLB data is distributed byte-by-byte to each enabled lane in byte order.

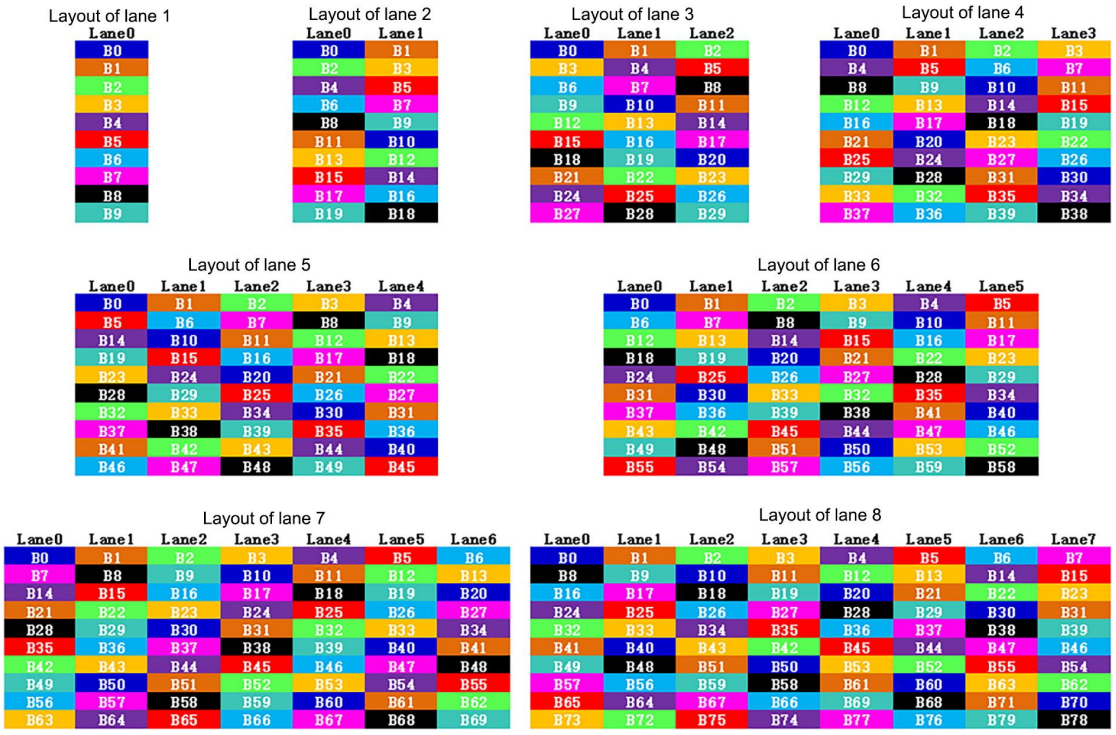
(b) Data is distributed in lane order from low to high.

(c) Based on the first two rules, when the number of enabled lanes is an even number, in all adjacent lanes, after every five bytes of data, a five-byte swap operation is performed. Specifically, every 10 bytes is treated as a group, where the first five bytes remain unprocessed, and the last five bytes are exchanged between odd and even lanes. This process cycles through, until the entire LLDF data processing is completed. The initial LLDF distribution state does not involve any byte swaps. Bytes in lane 0 swap with those in lane 1, bytes in lane 2 swap with those in lane 3, bytes in lane 4 swap with those in lane 5, and bytes in lane 6 swap with those in lane 7.

(d) Based on the first two rules, when the number of enabled lanes is five, in all lanes, data is transmitted cyclically in ascending order according to the lane sequence, and in each lane, every 10 bytes forms a cycle. Specifically, after every two bytes of data is transmitted, the subsequent data to be transmitted in lane 0 is moved to lane 1 for transmission, the subsequent data to be transmitted in lane 1 is moved to lane 2 for transmission, the subsequent data to be transmitted in lane 2 is moved to lane 3 for transmission, the subsequent data to be transmitted in lane 3 is moved to lane 4 for transmission, and the subsequent data to be transmitted in lane 4 is moved to lane 0 for transmission.

(e) All LLDF data is distributed according to the above rules, and the next LLDF is restored to the initial distribution state.

Figure 80 Interwoven arrangements for an eight-lane port



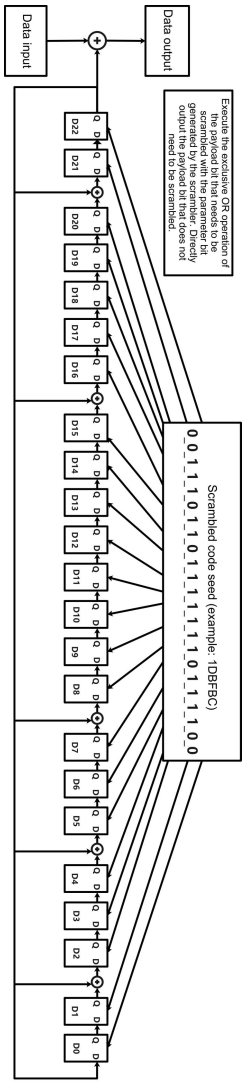
6.3.4.6 Scrambling and Descrambling

The main link shall support independent scrambling for each lane. Data is scrambled at the transmitting end and descrambled at the receiving end.

The scrambler achieves scrambling by using the random sequence generated by the linear feedback shift register (LFSR) and the data sequence for an XOR operation. In a multi-lane link, each lane has an independent LFSR for scrambling, and these LFSRs shall be synchronized. The receiving end, opposite to the transmitting end, performs the reverse operation. In a multi-lane link, the LFSR in each lane performs descrambling independently.

Figure 81 is a schematic diagram of the scrambling process.

Figure 81 Schematic diagram of scrambling



Scrambling shall follow the following rules:

(a) A scrambler is composed of LFSRs, and its generator polynomial is as follows:

$$g(x) = x^{23} + x^{21} + x^{16} + x^{8} + x^{5} + x^{2} + 1 \qquad (15)$$

(b) The initial values of LFSRs in the scrambler shall be reset according to LLCF_PAD (in little endian order). Each LFSR is reset and shifted independently. The length of each scrambling seed value is 23 bits.

- Seed 0: 0x1DBFBC, corresponding to the default value of logical lane 0.
- Seed 1: 0x0607BB, corresponding to the default value of logical lane 1.
- Seed 2: 0x1EC760, corresponding to the default value of logical lane 2.
- Seed 3：0x18C0DB, corresponding to the default value of logical lane 3.
- Seed 4：0x010F12, corresponding to the default value of logical lane 4.

- Seed 5：0x19CFC9, corresponding to the default value of logical lane 5.
- Seed 6: 0x0277CE, corresponding to the default value of logical lane 6.
- Seed 7: 0x1BB807, corresponding to the default value of logical lane 7.

(c) Output sequences of LFSRs shall be XORed with data in little-endian order, that is, the earliest output LFSR sequence bit is XORed with the earliest data bit transmitted on the lane.

(d) All bytes in LLBs are scrambled. After each bit is scrambled, the LFSR shifts by one bit.

(e) The control frame code pattern is not scrambled, and the LFSR does not shift.

(f) In high-speed lane service transmitting state, the scrambler is fixed in the ON state.

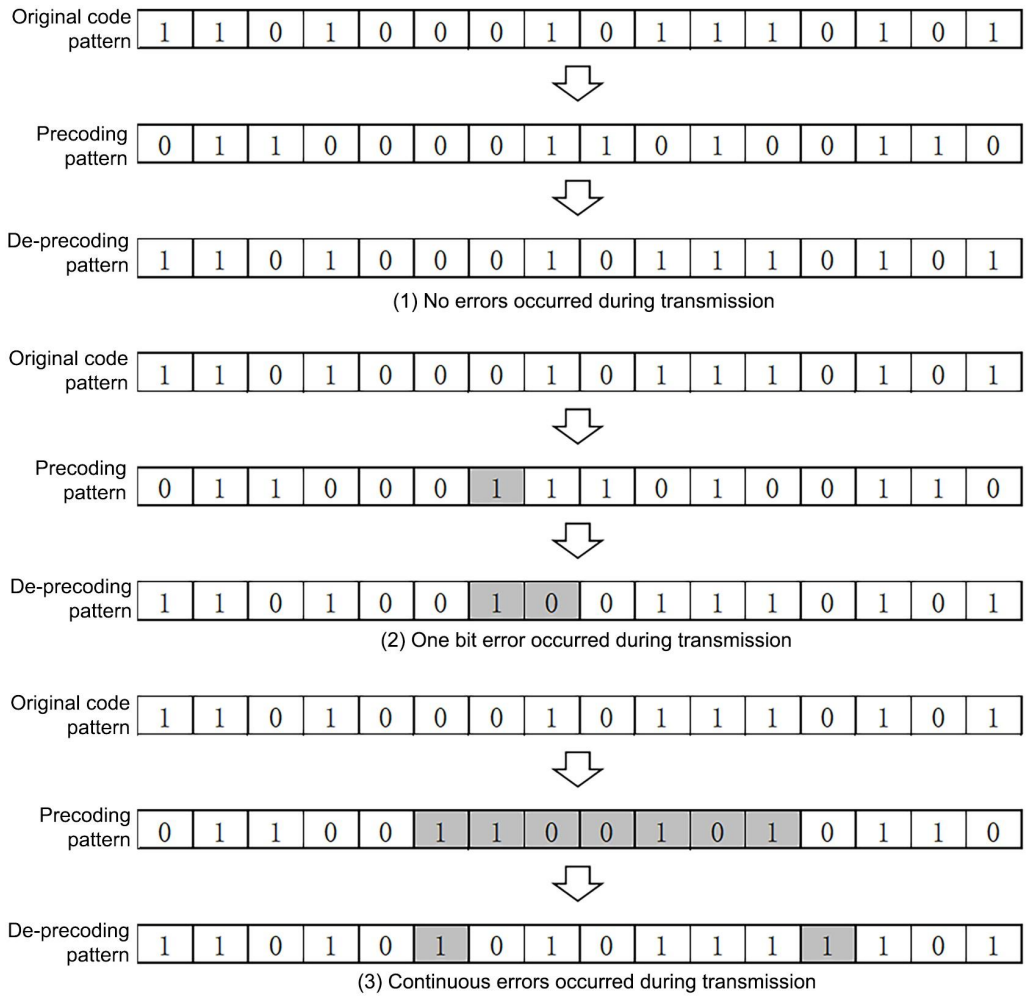
(g) When the LLCF_PAD code pattern is recognized, the scrambler is reset.

6.3.4.7 Precoding and Deprecoding

The main link shall support independent precoding for each lane. Data is precoded at the transmitting end and deprecoded at the receiving end. The logical layer control frame does not involve precoding or deprecoding. After an LLCF_DS is transmitted, the precoding module shall be reset before the first LLB. After an LLCF_DS is received, the predecoding module shall be reset before the first LLB.

The coding method is that every bit of data transmitted (before precoding) in each data frame shall be XORed with the preceding bit of data transmitted (after precoding), and the first bit transmitted is XORed with 1. The decoding method is that each one bit of data received (before deprecoding) shall be XORed with the preceding bit of data received (before deprecoding), and the first bit received is XORed with 1. For the specific implementation, see the example shown in Figure 82, where (1) is a scenario without transmission exceptions, (2) is a scenario with a one-bit data exception during transmission, and (3) is a scenario with consecutive transmission exceptions.

Figure 82 Schematic diagram of precoding and deprecoding



6.3.5 Training Parameter Requirements

Link training related time and parameters are shown in Table 78.

Table 78 Training time and parameter constraints

Parameter	Description	Min.	Max.	Unit
tWaitRxTrainStart	The maximum time for the transmitting end to wait for the receiving end to feed back a CLFM/LLFM after transmitting a TSM	-	10	ms
tSingleCLTimeout	The maximum time allowed for a single clock lock at the receiving end	-	2	ms
tTotalCLTimeout	The maximum time allowed for a clock lock at the receiving end	-	6	ms
tSingleEqTimeout	The maximum time allowed for each equilibrium at the receiving end	-	2	ms

Parameter	Description	Min.	Max.	Unit
tTotalEqTimeout	The maximum time allowed for each equilibrium at the receiving end	-	10	ms
tLaneLockTimeout	Time allowed for completion of lane lock	-	500	us
tWaitDsTimeout	Timeout period for the receiving end to wait for an LLCF_DS or LLCF_DST from the transmitting end after feeding back an LLFM	-	10	us
tTxLinkTrainTimeout	The maximum time allowed for each TX link training	-	25	ms
tRxLinkTrainTimeout	The maximum time allowed for each RX link training	-	25	ms
tMaxLp0fTime	The maximum time allowed for the LP0f state	-	100	ms
tLinkRecoveryTimeout	The maximum time allowed for each normal lane recovery	-	12	ms
tFastLockTimeout	The maximum time allowed for each fast lane recovery	-	50	us
tRecReLock	The maximum single lane lock time allowed for each normal lane recovery	-	100	us
tRecReEqTimeout	The maximum allowable equilibrium time during lane recovery	-	10	ms
cf_pad_length	Length of LLCF_PAD inserted during dynamic lane state switching	32	128	Byte
N_EIE	Number of occurrences of LLCF_EIE	16	32	Nr.
tLPxEntry	Low power entry delay (delay time from receiving an LRPM_ACK fed back by the peer end to entering the low power state)	-	50	us
tLP0Exit	LP0 wake-up time (delay time from initiating a low power wake-up request to exiting the low power state)	-	100	us
tLP1Exit	LP1 wake-up time (delay time from initiating a low power wake-up request to exiting the low power state)	-	300	us
tLP2Exit	LP2 wake-up time (delay time from initiating a low power wake-up request to exiting the low power state)	-	1	ms
tLP3Exit	LP3 wake-up time (delay time from initiating a low power wake-up request to exiting the low power state)	-	50	ms
LLMP_MaxReSendTime	The maximum number of LMP retransmitting times	-	3	Nr.
tTSMResponse	The maximum time allowed for the receiving	-	2.5	ms

Parameter	Description	Min.	Max.	Unit
	end to feed back a clock lock result after the transmitting end transmits a TSM			
tCLFMAck	The maximum time allowed for the transmitting end to respond to a CLFM from the receiving end	-	1	ms
tEQFMAck	The maximum time allowed for the transmitting end to respond to an EQFM from the receiving end	-	1	ms
tLLFMAck	The maximum time allowed for the transmitting end to respond to an LLFM from the receiving end	-	1	ms
tLPRMAck	The maximum time allowed for the receiving end to respond to an LPRM from the transmitting end	-	1	ms
tLWAMAck	The maximum time allowed for the receiving end to respond to an LWAM from the transmitting end	-	1	ms
tLDAMAck	The maximum time allowed for the receiving end to respond to an LDAM from the transmitting end	-	1	ms
tERR_RMAck	The maximum time allowed for the transmitting end to respond to an ERR_RM from the receiving end	-	1	ms
tDirChange	Waiting time to avoid bidirectional driving during lane direction switching	1	5	ms
tLLMMP_same_interval	Interval between transmitting two logical layer management messages with identical parameters through the main link	1	-	us

6.3.6 Exceptions and Handling Mechanisms

Definitions, detection mechanisms, response mechanisms, and reporting mechanisms related to logical layer exceptions are shown in Table 79. When an exception in the table occurs at the logical layer, it shall be handled according to the corresponding mechanism shown in Table 79.

For the same type of exception, the error detection threshold must be greater than the degradation detection threshold, for example, CF_EM > CF_DM. The count of degradation exceptions will be cleared during each recovery or when the detection threshold is reached, while the count of error exceptions will be cleared only when the error detection threshold is reached. The purpose of setting the error count is to allow error reporting and INIT state returning when link errors cannot be reduced after recovery.

Table 79 Logical layer exceptions and handling mechanisms

Type	Definition	Detection Mechanism	Response Mechanism	Reporting

			Receiving End	Transmitting End	
TSM Ack Timeout (TSM_AT)	TSM response timeout	The transmitting end does not receive a clock lock result from the peer end within tWaitRxTrainStart after transmitting a message.	-	A TSM is retransmitted, indicating transmission through SL.	Reporting the exception
TSM Fail (TSM_F)	Training start message failed	The TSM has been retransmitted N times.	After an exception report is detected, the whole link returns to the INIT state.	The whole link returns to the INIT state.	Reporting the exception, and setting the value at the TSM_F position in ERR_RM to 1
CLFM Ack Timeout (CLFM_AT)	CLFM response timeout	The receiving end does not receive a response from the peer end within tCLFMAck after feeding back a message.	A CLFM is fed back again, indicating transmission through SL.	-	Reporting the exception
CLFM Fail (CLFM_F)	Clock lock feedback message failed	The CLFM has been retransmitted N times.	The whole link returns to the INIT state.	After an exception report is detected, the whole link returns to the INIT state.	Reporting the exception, and setting the value at the CLFM_F position in ERR_RM to 1
Clock Lock Timeout (CLT)	Clock lock timeout	Each Lane performs detection independently. The RX receives a training start message, but fails to complete clock lock within tTotalCLTimeout.	The lane not completing clock lock feeds back a clock lock failure, and returns to the INIT state after receiving a response from the peer end.	The lane returns to the INIT state after receiving the clock lock failure from the peer end.	Reporting the exception
EQFM Ack Timeout (EQFM_AT)	EQFM response timeout	The receiving end does not receive a response from the peer end within tEQFMAck after feeding back a message.	An EQFM is fed back again, indicating transmission through SL.	-	Reporting the exception
EQFM Fail	Equilibrium feedback	The EQFM has been	The whole link returns to the	After an exception	Reporting the

Type	Definition	Detection Mechanism	Response Mechanism		Reporting Mechanism
			Receiving End	Transmitting End	
(EQFM_F)	message failed	retransmitted N times.	INIT state.	report is detected, the whole link returns to the INIT state.	exception, and setting the value at the EQFM_F position in ERR_RM to 1
Equilibrium Timeout (EQT)	Equilibrium timeout	Each lane performs detection independently. The RX completes clock lock and reports it, but fails to complete equilibrium within tTotalEQTimeout after receiving an Ack message from the TX.	The lane not completing equilibrium feeds back an equilibrium failure, and returns to the INIT state.	The lane returns to the INIT state after receiving the equilibrium failure from the peer end.	Reporting the exception
Lane Lock Timeout (LLT)	Lane lock timeout	The RX completes equilibrium and reports it, but fails to complete lane lock within tLaneLockTimeout after receiving an Ack message from the TX.	The lane not completing lane lock feeds back a lane lock failure, and returns to the INIT state.	The lane returns to the INIT state after receiving the lane lock failure from the peer end.	Reporting the exception
Link De-Skew Error (LDSE)	Lane deskew error	Each Lane performs detection independently, and all lanes at the receiving end cannot detect an LLCF_DS, or there is a lane deskew failure.	The whole link returns to the INIT state.	After an exception report is detected, the whole link returns to the INIT state.	Reporting the exception, and setting the value at the LBE position in ERR_RM to 1
Training Timeout Error (TTE)	Training timeout error	1. During initial link establishment, the RX fails to complete training within tRxLinkTrainTimeout after receiving a TSM. 2. During fast link recovery, RX performs link recovery after receiving an Ack message responding to an ERR_RM, but the operation times out (LNSM.Recovery.fast_lock timeout). 3. During normal link recovery, RX performs link recovery after receiving an	All lanes being trained currently return to the INIT state.	After an exception report is detected, all lanes being trained currently return to the INIT state.	Reporting the exception, and setting the value at the TTE position in ERR_RM to 1

Type	Definition	Detection Mechanism	Response Mechanism		Reporting Mechanism
			Receiving End	Transmitting End	
		Ack message responding to an ERR_RM, but the operation times out (LNSM.Recovery.re_lock or LNSM.Recovery.re_eq timeout). 4. The awakened end automatically wakes up after the LP0f times out, and a link exception is detected.			
LPRM Ack Timeout (LPRM_AT)	LPRM response timeout	The transmitting end does not receive a response from the peer end within tLPRMAck after transmitting a message.	-	An LPRM is retransmitted, indicating transmission through SL.	Reporting the exception
LPRM Fail (LPRM_F)	Low power request failed	The LPRM has been retransmitted N times.	-	-	Reporting the exception
LWAM Ack Timeout (LWAM_AT)	LWAM response timeout	The transmitting end does not receive a response from the peer end within tLWAMAck after transmitting a message.	-	An LWAM is retransmitted, indicating transmission through SL.	Reporting the exception
LWAM Fail (LWAM_F)	Link width adjust message failed	The LWAM has been retransmitted N times.	-	-	Reporting the exception
LDAM Ack Timeout (LDAM_AT)	LDAM response timeout	The transmitting end does not receive a response from the peer end within tLDAMAck after transmitting a message.	-	An LDAM is retransmitted, indicating transmission through SL.	Reporting the exception
LDAM Fail (LDAM _F)	LDAM request failed	The LDAM has been retransmitted N times.	-	-	Reporting the exception
ERR_RM Ack Timeout (ERR_RM_AT)	Error report message response timeout	The receiving end does not receive a response from the peer end within tERR_RMAck after feeding back a message.	An ERR_RM is retransmitted, indicating transmission through SL.	-	Reporting the exception
ERR_RM Fail (ERR_RM_F)	Error report message failed	The ERR_RM has been retransmitted N times.	-	-	Reporting the exception

Type	Definition	Detection Mechanism	Response Mechanism		Reporting Mechanism
			Receiving End	Transmitting End	
Control Frame Degrade (CFD)	Control frame degrade	A CF is detected at the specified position of a CFSLM. The CF code pattern matches, but an error exists.	If the CF exception threshold is set to 0, this exception is ignored.	-	Reporting the exception
			If the CF exception threshold is set to 1, the entire link enters the Recovery state.	After an exception report is detected, the entire link enters the Recovery state.	Reporting the exception, and setting the value at the CFD position in ERR_RM to 1
			If the CF exception threshold is set to CF_DN and detection threshold to CF_DM, when code pattern mismatches occur N times during M times of CF detection, the whole link enters the Recovery state.		
Control Frame Error (CFE)	Control frame error	A CF is detected at the specified position of a CFSLM. The CF code pattern matches, but an error exists.	If the CF exception threshold is set to 0, this exception is ignored.	-	Reporting the exception
			If the CF exception threshold is set to 1, the whole link returns to the INIT state.	After an exception report is detected, the whole link returns to the INIT state and the retraining starts.	Reporting the exception, and setting the value at the CFE position in ERR_RM to 1
			If the CF exception threshold is set to CF_EN and detection threshold to CF_EM, when code pattern		

Type	Definition	Detection Mechanism	Response Mechanism		Reporting Mechanism
			Receiving End	Transmitting End	
			mismatches occur N times during M times of CF detection, the whole link returns to the INIT state.		
Control Frame Detection failed (CFDF) RS Decoder Degrade (RDD)	Control frame detection failed RS decoder degrade	A CF is detected at the specified position of a CFSLM. The CF code pattern does not match. RS decoder exceptions are recognizable exceptions.	After an exception report is detected, the entire link enters the Recovery state.	After an exception report is detected, the entire link enters the Recovery state.	Reporting the exception, and setting the value at the CFE position in ERR_RM to 1
			If the RDD threshold is set to 0, this exception is ignored.	-	Reporting a software exception
			If the RDD threshold is set to 1, the entire link enters the Recovery state.	After an exception report is detected, the entire link enters the Recovery state.	Reporting the exception, and setting the value at the RDD position in ERR_RM to 1
			If the RDD threshold is set to RS_DN and detection threshold to RS_DM, when recognizable RS exceptions occur N times during M times of RS decoding, the whole link enters the Recovery state.		
RS Decoder Error (RDE)	RS decoder error	RS decoder exceptions are recognizable exceptions.	If the RDE threshold is set to 0, this exception is ignored.	-	Reporting the exception
			If the RDE	After an	Reporting

Type	Definition	Detection Mechanism	Response Mechanism		Reporting Mechanism
			Receiving End	Transmitting End	
			threshold is set to 1, the whole link returns to the INIT state.	exception report is detected, the whole link returns to the INIT state.	the exception, and setting the value at the RDE position in ERR_RM to 1
			If the RDE threshold is set to RS_EN and detection threshold to RS_DM, when recognizable RS exceptions occur N times during M times of RS decoding, the whole link returns to the INIT state.		
LLDP Degrade (LLDPD)	LLDP parsing degradation	LLDP header parsing error (single-bit error of PT)	If the LLDPD threshold is set to 0, this exception is ignored.	-	Reporting the exception
			If the LLDPD threshold is set to 1, the entire link enters the Recovery state.	After an exception report is detected, the entire link enters the Recovery state.	Reporting the exception, and setting the value at the LLDPD position in ERR_RM to 1
			If the LLDPD threshold is set to LLDP_DN and detection threshold to LLDP_DM, when LLDP parsing errors occur N times during M times of LLDP parsing, the entire link enters the Recovery state.		
LLDP Error (LLDPE)	LLDP parsing error	LLDP header parsing error (single-bit error of PT)	If the LLDPE threshold is set to 0, this exception is	-	Reporting the exception

Type	Definition	Detection Mechanism	Response Mechanism		Reporting Mechanism
			Receiving End	Transmitting End	
			ignored.		
			If the LLDPE threshold is set to 1, the whole link returns to the INIT state.	After an exception report is detected, the whole link returns to the INIT state.	Reporting the exception, and setting the value at the LLDPE position in ERR_RM to 1
			If the LLDPE threshold is set to LLDP_EN and detection threshold to LLDP_EM, when LLDP parsing errors occur N times during M times of LLDP parsing, the entire link returns to the INIT state.		
LLMMP Degrade (LLMMPD)	LLMMP parsing degradation	LLMMP header parsing error (single-bit error of PT), or LLMMP CRC failure	If the LLMMPD threshold is set to 0, this exception is ignored.	-	Reporting the exception
			If the LLMMPD threshold is set to 1, the entire link enters the Recovery state.	After an exception report is detected, the entire link enters the Recovery state.	Reporting the exception, and setting the value at the LLMMPD position in ERR_RM to 1
			If the LLMMPD threshold is set to LLMMP_DN and detection threshold to LLMMP_DM, when LLMMP parsing errors occur N times during M times of LLMMP parsing, the entire link enters the Recovery state.		

Type	Definition	Detection Mechanism	Response Mechanism		Reporting Mechanism
			Receiving End	Transmitting End	
LLMMP Error (LLMMPE)	LLMMP parsing error	LLMMP header parsing error (single-bit error of PT), or LLMMP CRC failure	If the LLMMPE threshold is set to 0, this exception is ignored.	-	Reporting the exception
			If the LLMMPE threshold is set to 1, the whole link returns to the INIT state.	After an exception report is detected, the whole link returns to the INIT state.	Reporting the exception, and setting the value at the LLMMPE position in ERR_RM to 1
			If the LLMMPE threshold is set to LLMMP_EN and detection threshold to LLMMP_EM, when LLMMP parsing errors occur N times during M times of LLMMP parsing, the entire link returns to the INIT state.		
Packet Type Error (PTE)	Packet header parsing error	A packet is not an LLDP or LLMMP through packet header detection (a multiple-bit error occurs).	The entire link enters the Recovery state.	After an exception report is detected, the entire link enters the Recovery state.	Reporting the exception, and setting the value at the PTE position in ERR_RM to 1
ECC Error (ECCE)	ECC error	An ECC error is detected at the TL layer.	The entire link enters the Recovery state.	After an exception report is detected, the entire link enters the Recovery state.	Reporting the exception, and setting the value at the ECCE position in ERR_RM to 1
Dynamic Link Change Recovery Error	During lane state switching, a link	During lane increase/reduction, lane direction switching, and link recovery processes, a	All lanes not in disable state return to the	After an exception report is detected, all	Reporting the exception, and setting

Type	Definition	Detection Mechanism	Response Mechanism		Reporting Mechanism
			Receiving End	Transmitting End	
(DLCRE)	exception occurs in a normal TX lane.	normal TX lane link in HS state is abnormal and enters the INIT state.	INIT state.	lanes not in disable state return to the INIT state.	the value at the DLCRE position in ERR_RM to 1
TXTEE	TX lane training timeout error	1. During initial link establishment, the TX fails to complete training within tTxLinkTrainTimeout after transmitting a TSM. 2. During fast link recovery, TX performs link recovery after receiving an ERRRM, but the operation times out (LNSM.Recovery.fast_lock timeout). 3. During normal link recovery, TX performs link recovery after receiving an ERRRM, but the operation times out (LNSM.Recovery.re_lock or LNSM.Recovery.re_eq timeout).	All lanes being trained currently return to the INIT state.	After an exception report is detected, all lanes being trained currently return to the INIT state.	Reporting the exception, and setting the value at the TXTTE position in ERR_RM to 1

Note 1: The operations involved in software exception reporting are related to the implementation methods of manufacturers and are not defined in this document.

Note 2: The RDD threshold, RDE threshold, LLDPD threshold, LLDPE threshold, LLMMPD threshold, and LLMMPE threshold are determined by the drivers of manufacturers and are not defined in this document.

6.3.7 Logical Layer Register (Informative)

The implementation suggestions for registers are shown in Table 80. Users can implement the following registers for software-hardware interaction.

Table 80 Logical layer registers

Bit Width/b	Field Name	Description	Type	Default Value
8*4	Lane0–7_tx_data_dly	Number of bytes of delayed data on lanes 0–7 at the transmitting end Note: It applies only when the transmitting end of a local device supports skew adjustment.	RW	0

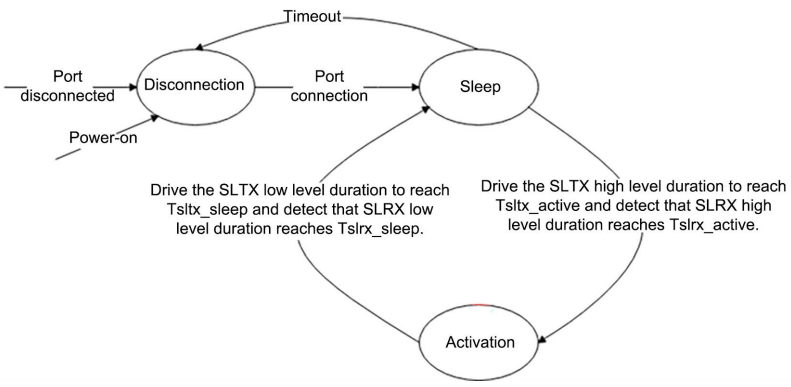
Bit Width/b	Field Name	Description	Type	Default Value
1*4	lane0–7_disable	Lane disabling/enabling (Considering the complexity/risks of software-hardware interaction, hardware will not be refreshed.)	RW	1
1	fast_training	The fast training indicator fast_training = 1 is used to indicate that on this link, both the local and peer RX have obtained the negotiated rate based on capability negotiation (including port rate, Cableinfo, and other exchanges), and a set of reliable receiving lane equilibrium parameters have been obtained based on the negotiated rate, or both ports have agreed that clock lock and lane equilibrium process are not required at the current rate. If the quick training fails, execute the link training process in 6.3.2.	RW	0
1	fast_recovery	Whether the receiving end supports fast recovery: 0b: not support 1b: support	RW	0
Note: The register field names are reference names. This document uses the field names in this table to facilitate scheme description.				

6.4 Sideband Link

6.4.1.1 Overview

A sideband link has three states: disconnected, sleep, and active. The sideband link states are shown in Figure 83. For the behavior and transition details of each state, see the description of each state.

Figure 83 Sideband link states



6.4.1.2 Disconnected State

When the sideband link is disconnected, it is inoperable.

6.4.1.3 Sleep State

6.4.1.3.1 State Description

When the sideband link is in sleep state, it cannot transmit or receive data. You can exit the sleep state by performing operations related to the sleep exit process.

6.4.1.3.2 Sleep Exit Process

When the sideband link in sleep state detects a message that needs to be transmitted or other implementation-related conditions prompting it to exit the sleep state, the following operations are performed:

(a) The transmitting end starts and continues to drive the SLTX to a high level.

(b) Starting from the moment when the transmitting end drives the SLTX to a high level:

(1) During Ttimeout_active, when the receiving end detects that SLRX is at a high level for Tslrx_active at least and the transmitting end drives the SLTX to a high level for Tsltx_active at least, the sideband link transitions to the active state.

(2) Otherwise, when Ttimeout_active is up, it transitions to the disconnected state. It is recommended that when Ttimeout_active is up, drive the SLTX to a low level for Tretry_active, and execute step 1 again. After this process is repeated twice, if timeout still occurs, the sideband link transitions to the disconnected state.

For the sideband link in sleep state, when the receiving end detects that SLRX is at a high level for Tslrx_active, the following operations are performed: The transmitting end starts to drive the SLTX to a high level. After Tsltx_active, the sideband link transitions to the active state.

Figure 84 shows the timing sequence of the sideband link state (link_state) from the sleep state (SLEEP) to the active state (ACTIVE). To distinguish the two ends of the link, in the figure, the serial number 0 is used to represent the initiating end and 1 the responding end. The SLRX at the responding end corresponds to the SLTX (SLTX0) at the initiating end, and the SLTX at the responding end corresponds to the SLRX (SLRX0) at the initiating end. Both ends of the link enter the active state from the sleep state according to the following process.

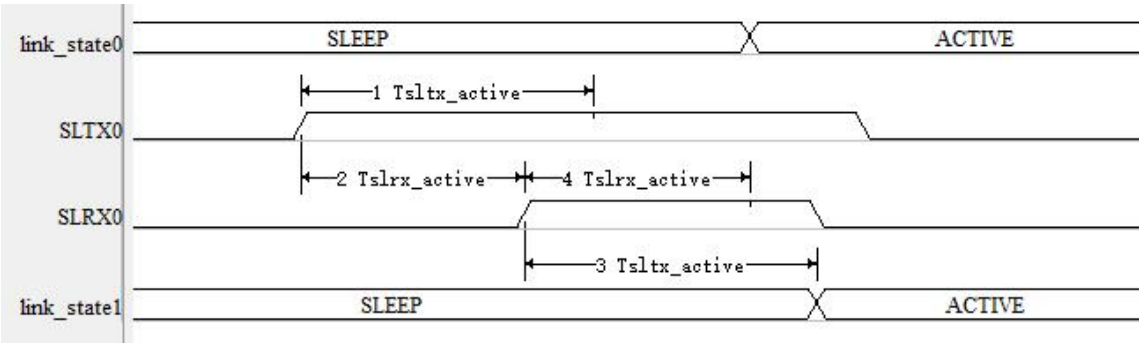
(1) The initiating end starts and continues to drive the SLTX to a high level.

(2) The responding end detects that SLRX is at a high level for Tslrx_active.

(3) The responding end starts and continuously drives the SLTX to a high level, and enters the active state after Tsltx_active.

(4) The initiating end detects that SLRX is at a high level for Tslrx_active. At this time, the SLTX has been driven at a high level for Tsltx_active, and the link enters the active state.

Figure 84 Timing sequence of the sideband link state from the sleep state to the active state



6.4.1.4 Active State

6.4.1.4.1 State Description

In active state, unless otherwise specified, the sideband link can transmit and receive data normally.

When the sideband link transitions to active state from sleep state, to compensate for the time difference between link state transitions at the two ends, the sideband link needs to wait for Twait before starting to transmit data.

The sideband link in active state can enter the sleep state by performing operations related to the sleep entry process.

6.4.1.4.2 Sleep Entry Process

When the sideband link in active state detects that no service has been transmitted for Tidle and intends to enter the sleep state, the following operations are performed:

(a) The transmission of all subsequent messages is suspended, and normal data receiving is allowed.

(b) The transmitting end starts and continues to drive the SLTX to a low level.

(1) During Ttimeout_sleep,

- If the receiving end detects that SLRX has been at a low level for Tslrx_sleep at least, and the transmitting end has driven the SLTX to a low level for Tsltx_sleep at least, the sleep entry process ends, and the sideband link enters the sleep state.
- If the receiving end detects a SOB, the sleep entry process ends, and the active state continues. The transmitting end drives the SLTX to a high level for Tsltx_active. After Twait, data can be transmitted normally. The sleep entry process can be re-executed according to service requirements.

(2) Otherwise, when Ttimeout_sleep is up, the sleep entry process ends, and the active state continues. The transmitting end drives the SLTX to a high level for Tsltx_active. After Twait, data can be transmitted normally. The sleep entry process can be re-executed according to service requirements.

After the sideband link initiates the sleep entry process, the process can only end according to the above operations (step 2), and cannot be canceled midway. Otherwise, the states at the transmitting and receiving ends cannot align.

For the sideband link in active state, when the receiving end detects that SLRX has been at a low level for Tslrx_sleep, it performs the following operations:

(a) If it agrees to enter the sleep state, it drives the SLTX to a low level for Tsltx_sleep, and then enters the sleep state.

(b) If it refuses to enter the sleep state, it stays in the active state, and transmits and receives data normally.

Once the sideband link agrees to enter the sleep state and drives the SLTX to a low level, the sleep entry process can only end after Tsltx_sleep, and cannot be canceled midway. Otherwise, the states at the transmitting and receiving ends cannot align.

Figure 85 shows the timing sequence of the sideband link from the active state (ACTIVE) to the sleep state (SLEEP). To distinguish the two ends of the link, in the figure, the serial number 0 is used to represent the initiating end and 1 the responding end. The SLRX at the responding end corresponds to the SLTX (SLTX0) at the initiating end, and the SLTX at the responding end corresponds to the SLRX (SLRX0) at the initiating end. Both ends of the link enter the sleep state from the active state according to the following process.

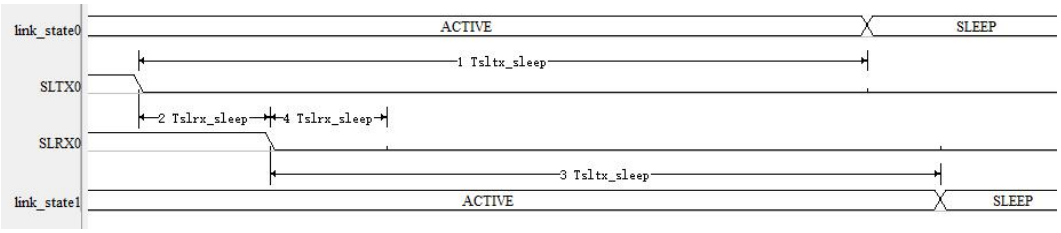
(1) The initiating end starts and continues to drive the SLTX to a low level.

(2) The responding end detects that SLRX is at a low level for Tslrx_sleep.

(3) The responding end starts and continuously drives the SLTX to a low level, and enters the sleep state after Tsltx_sleep.

(4) The initiating end detects that SLRX is at a low level for Tslrx_sleep, and continuously drives the SLTX to a low level for Tsltx_sleep. Then, the sideband link enters the sleep state.

Figure 85 Timing sequence of the sideband link state from the active state to the sleep state

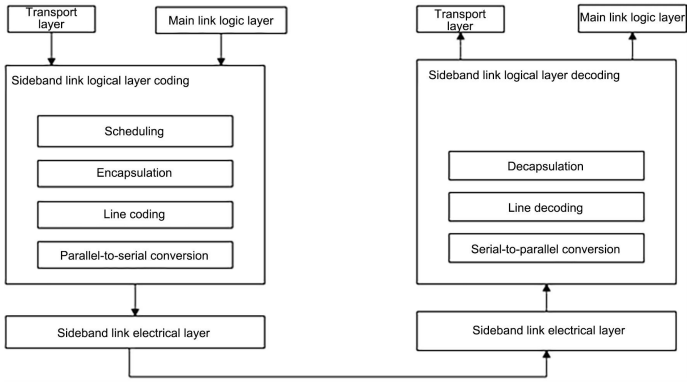


6.4.2 Coding/Decoding

6.4.2.1 Process

As shown in Figure 86, the sideband link codes transport layer packets and main link logical layer packets and transmits them to the sideband link electrical layer. The process includes scheduling, encapsulation, line coding, and parallel-to-serial conversion. The sideband link decodes the sideband link electrical layer data and transmits these data to the transport layer or main link logical layer. The process includes serial-to-parallel conversion, line decoding, and decapsulation.

Figure 86 Example of sideband link coding/decoding process



6.4.2.2 Function

6.4.2.2.1 Scheduling

The sideband link shall support the scheduling of packets from the transport layer and the main link logical layer. The main link logical layer has a high priority for packet transmission.

6.4.2.2.2 Encapsulation

The sideband link logical layer does not parse packets from the transport layer or the main link logical layer. Each packet input by the transport layer or the main link logical layer corresponds to a data transmission on the sideband link.

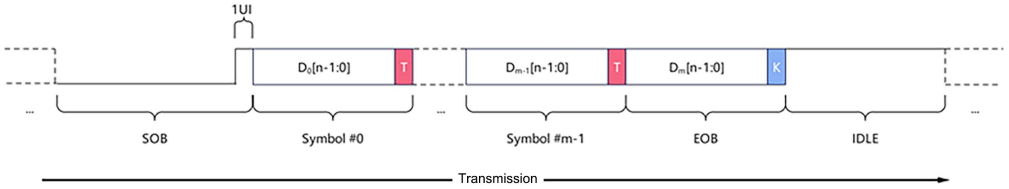
As shown in Figure 87, the sideband link logical layer identifies different packet types according to the packet headers added, to facilitate parsing at the receiving end.

6.4.2.2.3 Line coding

(a) Toggle Coding

A Toggle coding solution with n being 8 is adopted for the line coding of the sideband link logical layer. After Toggle coding, the byte data encapsulated by the sideband link logical layer is transmitted to the electrical layer line.

Figure 87 Schematic diagram of Toggle coding



The n-bit line coding is shown in the figure above.

One T bit or K bit is added to every n bits of data to form a symbol of n + 1 bits. A K bit is added to the last symbol, and T bits are added to other symbols.

Toggle bit (T bit): inversion of the preceding bit

Keep bit (K bit): same as the preceding bit

D_0–D_m correspond to sideband link transport layer packet data.

(b) Start of Burst

A start of burst (SOB) consists of n + 1 bits of 0 and one bit of 1, and is used to indicate the start of a valid burst transmission.

(c) Data Symbol

A data symbol consists of n bits of valid data and one T bit. The T bit is the inversion of the last bit of valid data, and can effectively ensure that there is a jump within each symbol.

(d) End of Burst

An end of burst (EOB) consists of n bits of valid data and one K bit, and is used to indicate the end of a valid burst transmission.

(e) Idle

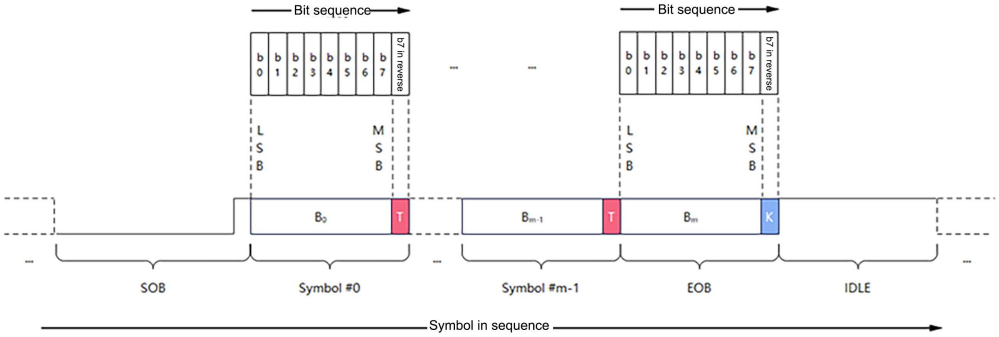
When a transmission on the sideband link is completed and there is no other data to be transmitted, the line enters the idle state.

The line remains at a high level in the idle state. The idle state length is greater than or equal to 0 UI. If it equals 0 UI, there is no idle state between two burst transmissions.

6.4.2.2.4 Parallel-to-serial Conversion

Data symbols undergoing line coding are distributed to the lanes at the transmitting end of the sideband link according to the method shown in Figure 88.

Figure 88 Schematic diagram of serialization (8-bit Toggle coding)



6.4.3 Time Parameters

Table 81 Time parameters of the sideband link

Symbol	Description	Minimum Value	Maximum Value	Unit
T_{slrx_active}	A continuous high level is detected on the SLRX during this period, indicating that the peer device requests or agrees to enter the activation state.	20	22	us

Symbol	Description	Minimum Value	Maximum Value	Unit
T_{sltx_active}	A continuous high level is driven on the SLTX during this period, indicating a request or consent to enter the active state.	25	27	us
T_{slrx_sleep}	A continuous low level is detected on the SLRX during this period, indicating that the peer device requests or agrees to enter the sleep state.	10	1000	us
T_{sltx_sleep}	A continuous low level is driven on the SLTX during this period, indicating a request or consent to enter the sleep state.	20	22	ms
Tidle	A minimum period of no data transmission, allowing the initiation of the sleep entry process	2	-	s
Twait	A wait period after which data can be transmitted when the sleep state transitions to the active state	10	-	us
Ttimeout_sleep	A timeout period for waiting for a response after a request for entering the sleep state is initiated	25	-	ms
Ttimeout_active	A timeout period for waiting for a response after a request for exiting the sleep state is initiated	100	-	ms
Tretry_active	A wait period after which a sleep state exit handshake can be initiated after the previous one times out	1		us

7 Transport Layer

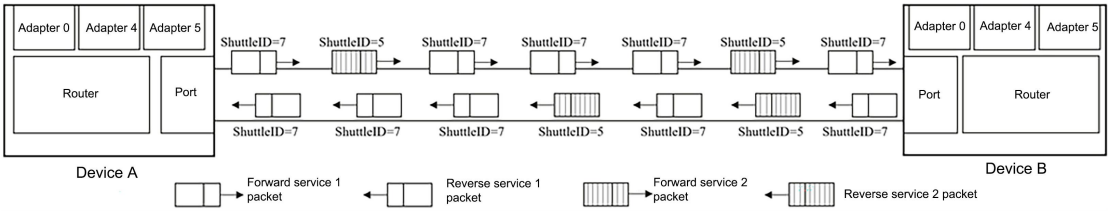
7.1 Overview

In a protocol stack, the transport layer is located between adapters and the logical layer. It receives and forwards packets from all adapters, and manages each packet flow during the forwarding process, including flow control, and distribution and combination of main and sideband links.

7.2 Routing and Forwarding Model of the Transport Layer

As shown in Figure 89, a shuttle is formed by transmitting packets of the same service flow on a link. The transport layer supports transmission of multiple service flows. Packets of different service flows are distinguished by ShuttleID. ShuttleID 7 identifies packets of service flow 1, and ShuttleID 5 identifies packets of service flow 2. ShuttleIDs are assigned by the management adapter.

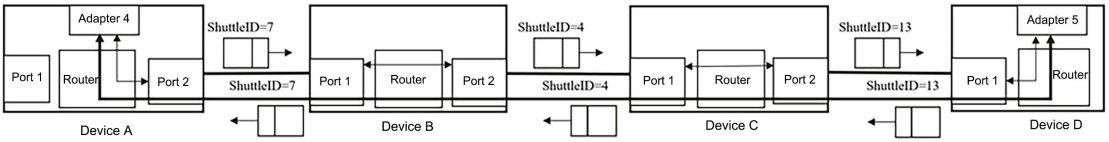
Figure 89 Schematic diagram of shuttles



The packets of a service flow may pass through multiple shuttles before they are transmitted to the target device. Multiple shuttles between two adapters can be cascaded to form a channel. As shown in Figure 90, the blue line represents a channel from adapter 4 of device A to adapter 5 of device D, and includes three shuttles:

- Shuttle from device A to device B, with the packet flow identified by ShuttleID = 7
- Shuttle from device B to device C, with the packet flow identified by ShuttleID = 5
- Shuttle from device C and device D, with the packet flow identified by ShuttleID = 13

Figure 90 Schematic diagram of a channel

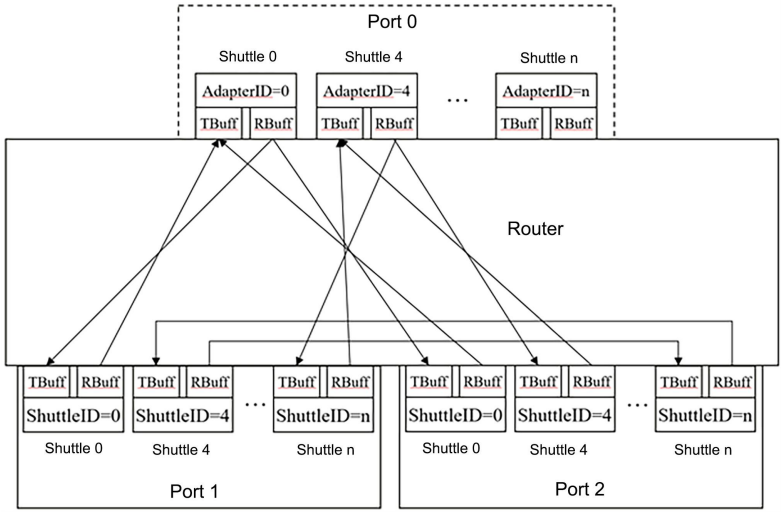


ShuttleID numbers and their assignment shall follow the following rules:

- The ShuttleID number in the packet received from the management adapter is fixed to 0, and ShuttleID numbers 1–3 are reserved.
- Different input ShuttleIDs of the same input port correspond to different service flows, and different output ShuttleIDs of the same output port correspond to different output flows.
- Output ShuttleIDs of different output ports are independent of each other, and the ShuttleID numbers can be the same.
- Input ShuttleIDs of input ports can be configured to multiple different output ports, indicating multicast replication of corresponding service flows.
- For a service flow transmitted in two directions on the same physical link, the same ShuttleID shall be used. If it is transmitted on different physical links, different ShuttleIDs may be used.
- The output ShuttleID assigned by the management adapter cannot exceed the maximum ShuttleID supported by the output port, and the ShuttleID output by the output port cannot exceed the maximum ShuttleID supported by the downstream input port.

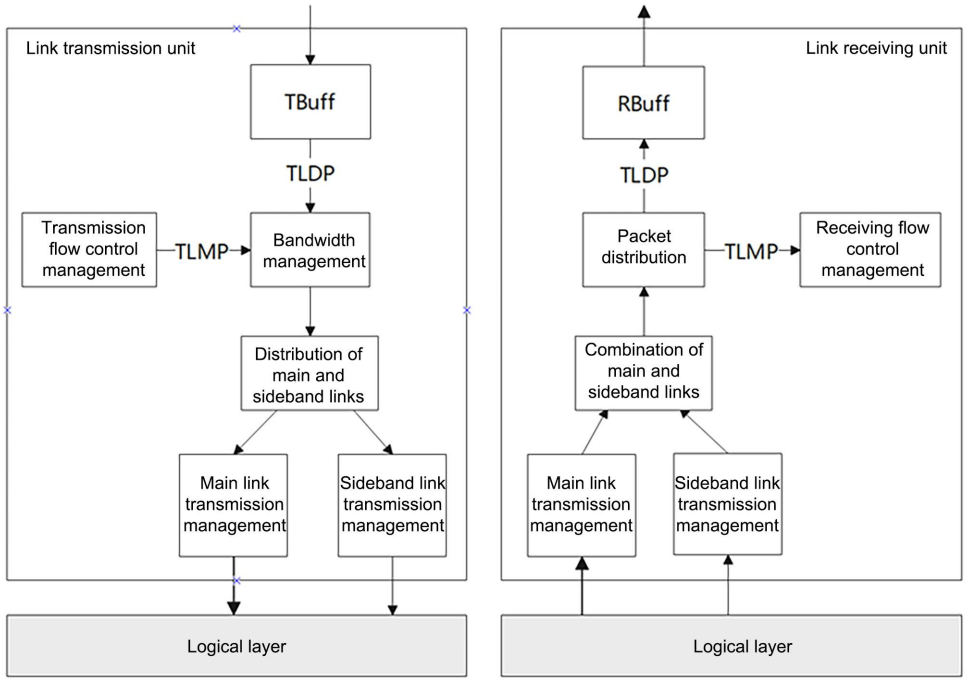
The packet forwarding model of the transport layer is shown in Figure 91. The transport layer takes out packets from the receiver buffer (RBuff) and forwards them to another port based on routing information, or replicates and forwards them to the transmitter buffers (TBuff) of multiple ports.

Figure 91 Packet forwarding model of the transport layer



All adapters above the transport layer in the protocol stack can be collectively regarded as a virtual port (Port 0). In this case, the AdapterID of each adapter is equivalent to a ShuttleID. Port 0 can forward packets to any port including itself. As shown in Figure 92, each physical port (non-Port 0) contains one link transmitting unit or one link receiving unit, and can only forward packets to ports other than itself.

Figure 92 Schematic diagram of a link transmitting unit and a link receiving unit



Note: The physical port implements the features of the link transmitting unit or link receiving unit.

Both TLDP and TLMP are transport layer packets. A TLDP means a transport layer data packet generated by the management adapter or audio and video adapter. A TLMP means a transport layer management packet generated and terminated at the transport layer.

In the link transmitting unit, Tbuff and transmitting flow control management module transmit TLDPs and TLMPs respectively. These packets, subject to the flow control of the bandwidth management module, are distributed to the main link or sideband link, and finally transmitted to the logical layer after passing through the main link management or sideband link management.

In the link receiving unit, after the packets transmitted by the logical layer are operated by the main link receiving management or sideband link receiving management, the main and sideband links are combined. Then, the packets are parsed into TLDPs and TLMPs through the message distribution operation, and transmitted to the RBuff and the receiving flow control management module respectively.

The packet input and output sequences shall comply with the following rules:

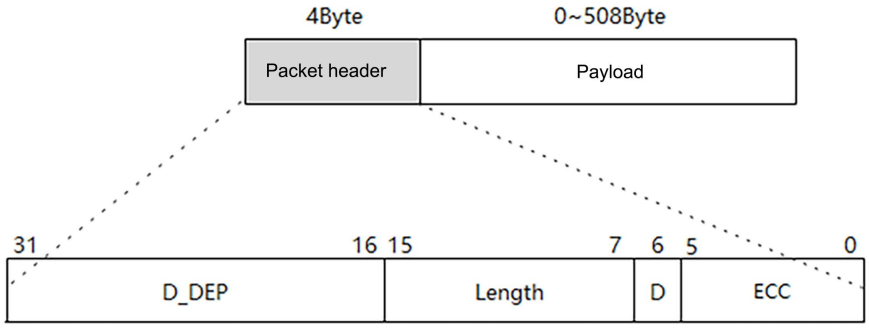
- In the same shuttle, when the transport layer transmits TLDPs through the main link, the packet input sequence is consistent with the packet output sequence.
- In the same shuttle, when the transport layer transmits TLDPs through the sideband link, the packet input sequence is consistent with the packet output sequence.
- For TLDPs in different shuttles, the transport layer does not guarantee the relationship between the packet input sequence and the packet output sequence.

7.3 Transport Layer Packet

7.3.1 Structure of Transport Layer Packets

The format of transport layer packets is shown in Figure 93, including the packet header and payload. A transport layer packet starts with a 4-byte packet header and supports a payload of 0–508 bytes. The payload requires alignment according to a 4-byte length. If the data is less than 4 bytes, it needs to be filled and aligned. It is recommended that data 0 should be filled. The receiving end shall have the ability to identify and ignore the filled data.

Figure 93 Structure of transport layer packets



The definition and description of the transport layer packet header are shown in Table 82.

Table 82 Definition and description of the transport layer packet header

Bit	Field Name	Description
31:16	D_DEP	It is related to "D". See TLDPs and TLMPs for details. When a transport layer packet is a TLMP, this domain includes the TLMP Type information. See Section 7.3.3 for details.
15:7	Length	It indicates the length of the packet payload. The packet receiver uses this value to parse the data in the payload. The maximum length is 508 bytes.
6	D	It is the flag bit of the transport layer data packet and transport layer management packet. 1b: TLDP 0b: TLMP
5:0	ECC	It checks the packet header bit [31:6].

The ECC check shall meet the requirements of Appendix C.6.

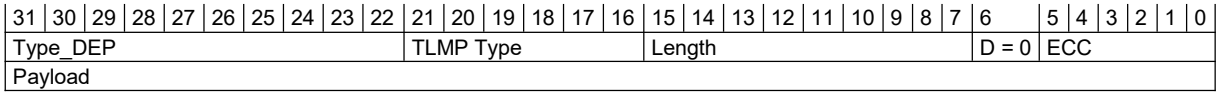
7.3.2 Transport Layer Data Packet

The requirements and field descriptions of TLDPs are defined by each adapter. See Section 8 Audio and Video Adapter and Chapter 9 Management Adapter for details.

7.3.3 Transport Layer Management Packet

A TLMP is used for management at the two ends of a link, and is generated and terminated at the transport layer. Its structure is shown in Figure 94.

Figure 94 TLMP structure



When a transport layer packet is a TLMP, "D_DEP" includes information such as TLMP type, and its definition is shown in Table 83.

Table 83 Definitions of TLMP fields

Bit	Field Name	Description
31:22	Type_DEP	The definition of TLMP Type varies. The requirements and field definitions of various packets are shown in sections 7.5, 7.8, and 7.9.
21:16	TLMP Type	TLMP types: 000000b: transport layer credit allocation packet (TLCAP) 000001b: transport layer credit allocation packet acknowledgment (TLCAP_ACK)

Bit	Field Name	Description
		000010b: transport layer credit consumption packet (TLCCP) 000011b: transport layer credit recovery packet (TLCRP) 000100b: transport layer flow control exception notification packet (TLFCENP) 001001b: transport layer main link idle packet (TLMIP) 001100b: transport layer sideband link management packet (TLSMP) Other values are reserved.
15:7	Length	It indicates the message payload length in bytes.
6	D	0b: TLMP
5:0	ECC	It checks the packet header bit [31:6].

Among these, flow control management packets such as the TLCAP, TLCAP_ACK, TLCCP, TLCRP, and TLFCENP are generated and terminated by the transport layer flow control management units at both ends of the link. The TLMIP is generated and terminated by the transport layer main link management units at both ends of the link. The transport layer sideband link acknowledgement packet (TLSAP) is generated and terminated by the transport layer sideband link management modules at both ends of the link. The minimum header spacing for non-idle transport layer packets transmitted over the main link is 64 bytes.

7.4 Routing and Forwarding

When the transport layer forwards a packet, the input port shall provide corresponding routing information to indicate the direction of the subsequent output port of the packet. Routing information is distributed and managed by the management adapter. It shall include the input ShuttleID, forwarding port, forwarding ShuttleID, valid flag, and other information of the packet in unicast/multicast situations. The routing information table (referred to as "routing table") for each input port is shown in Table 84.

Table 84 Routing information table

Input ShuttleID Number	Forwarding Entry Name	Forwarding Entry		
Minimum number of ShuttleID	Unicast 0	FwVld	FwPort	FwShuttleID
	Unicast 1	FwVld	FwPort	FwShuttleID
	…	…	…	…
	Unicast 15	FwVld	FwPort	FwShuttleID
…	…	…	…	…
Maximum number of ShuttleID	Unicast 0	FwVld	FwPort	FwShuttleID
	Unicast 1	FwVld	FwPort	FwShuttleID
	…	…	…	…
	Unicast 15	FwVld	FwPort	FwShuttleID

Note: FwVld indicates the valid flag bit, FwPort indicates the forwarding output port, and FwShuttleID indicates the ShuttleID number in the output port.

Each service flow in the input port corresponds to at least one forwarding entry and supports up to 16 valid forwarding entries. Multiple entries with FwPort as Port 0 are allowed, but each non-Port 0 entry can appear only once. When multiple valid forwarding entries exist, it indicates multicast replication of the input service flow. Each replicated service flow is treated as an independent unicast service flow.

Port 0 can forward packets to any port, including itself. Other ports (Port 1 to Port 15) can only forward packets to ports other than themselves.

The transport layer forwards management adapter packets received from each port to the management adapter and notifies the management adapter of the port number from which the packet was received. When sending management adapter packets, the transport layer transmits them to the corresponding port based on the port number provided by the management adapter.

When forwarding service flow packets, the transport layer replaces the ShuttleID in the packet header with the "FwShuttleID" from the forwarding entry. After refreshing the packet header ECC, the transport layer sends the packet to the "FwPort" specified in the forwarding entry. Before forwarding, the following possible abnormal scenarios should be noted:

- The management adapter provides a 1-bit Vld flag for each input service flow of each input port. When "Vld" is invalid, the transport layer discards the corresponding service flow.
- When the ShuttleID of the input service flow exceeds the maximum ShuttleID number supported by the input port, the transport layer discards the service flow packet and reports it to the management adapter.
- When the "FwVld" of the forwarding entry is invalid, the transport layer ignores the entry and does not forward the corresponding service flow packet.
- When the "FwPort" of the forwarding entry exceeds the maximum port number supported by the device, the transport layer ignores the entry and does not forward the corresponding service flow packet.
- When the "FwShuttleID" of the forwarding entry exceeds the maximum ShuttleID number supported by "FwPort", the transport layer ignores the entry, does not forward the corresponding service flow packet, and informs the management adapter.

The transport layer forwarding delay is defined as the time difference between the router receiving the last bit of a TLDP from the cable and sending the first bit of the same TLDP to the cable. The difference between the maximum and minimum transport layer forwarding delays is referred to as the transport layer forwarding delay jitter (tTLDPFwJitter). It is stipulated that the jitter should not exceed the values specified in Section 7.11.

Note: The transport layer forwarding jitter does not include delay introduced by scheduling other packets or transmitting logical layer control frames, and does not include waiting delay caused by insufficient credit.

7.5 Flow Control Management

Flow control ensures that the RBuff cache space of the link receiving unit does not overflow. Different types of service flow packets may use different mechanisms. When transmitting video service flows, the flow control disabling mechanism is used, that is, TLDPs are directly transmitted upon receipt without performing flow control on the video service flow.

For more details on flow control management, see Appendix E.

7.6 Bandwidth Management

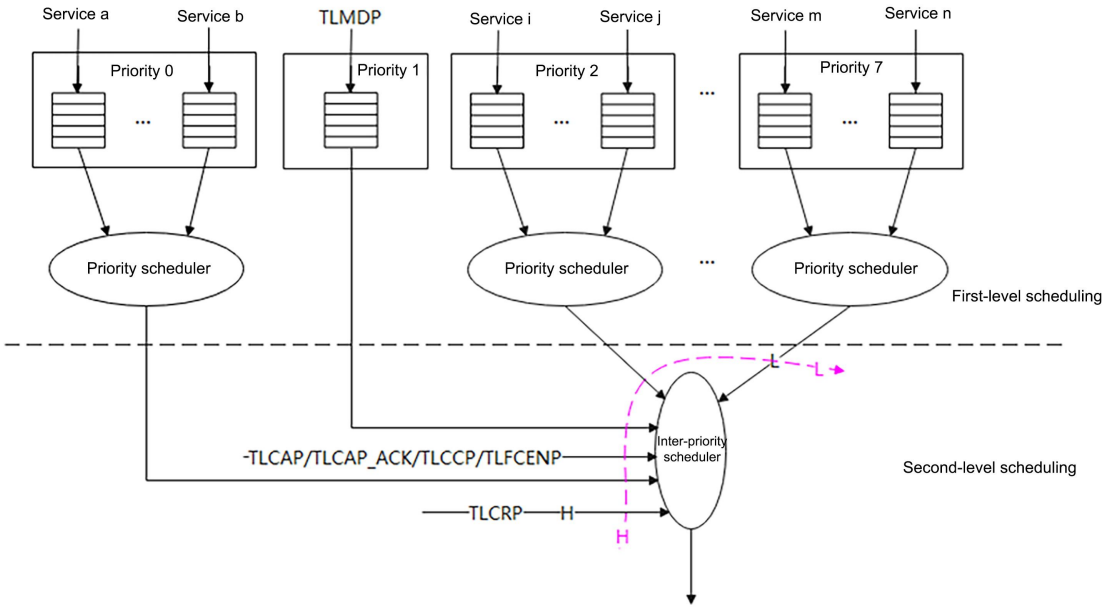
The link transmission unit includes a bandwidth management module that performs priority and bandwidth management for all transmitted packets. Different types of service flows have distinct priorities, and their weights reflect the bandwidth requested by each flow. The management adapter provides egress flow scheduling information for each service flow of each output port, as shown in Table 85.

Table 85 Requirements for the two-level scheduling mechanism

Name	Bit Width	Description
Priority	3	Priority of the service flow. Priorities decrease sequentially from 0 to 7, that is, 0 corresponds to the highest priority, and 7 corresponds to the lowest priority.
Weight	8	Weight for Weighted Round Robin (WRR). Values 1–255 are valid; 0 indicates no participation in scheduling.

The bandwidth manager incorporates a two-level scheduling mechanism: the intra-priority scheduler serves as the first level, and the inter-priority scheduler serves as the second level, as illustrated in Figure 95.

Figure 95 Bandwidth manager



The intra-priority scheduler schedules all service flows within the same priority level according to the following rules:

- The priority of audio-video service flow TLDPs is fixed at 0, and the priority of TLMDPs is fixed at 1. The priorities of other service flows cannot be configured to 1.

- Except for priority 1, all service flows within the same priority level are scheduled using WRR.
- WRR is performed on a per-packet basis. The number of packets scheduled for each service flow is proportional to its configured weight.
- After the management adapter configures new WRR weights, the scheduler immediately applies these new values for scheduling.

Note: TLMDPs are treated as management adapter packets. The transport layer receives TLMDPs from the management adapter and forwards them to the corresponding output ports based on routing information.

The inter-priority scheduler receives service flow packets from each intra-priority scheduler and performs SP scheduling with the TLCAP, TLCAP_ACK, TLCCP, TLCRP, and TLFCENP generated by the transport layer. This means higher-priority packets are scheduled first, and lower-priority packets are scheduled later. The order of priority from highest to lowest is:

- TLCRP.
- Service flow packets with priority of 0.
- TLCAP, TLCAP_ACK, TLCCP, and TLFCENP.
- TLMDP.
- Service flow packets with priorities 2 to 7. Priority 2 packets are scheduled first, and priority 7 packets are scheduled last.

7.7 Distribution and Combination of Main and Sideband Links

The General Purpose Multimedia Interface should have the capability to select either the main link or the sideband link for packet transmission based on service characteristics and requirements:

- Main link packet transmission has low latency but high power consumption. For example, the packets used for network management in the management adapter can be transmitted through the main link to reduce latency.
- Sideband link packet transmission has high latency but low power consumption. If the service is latency-tolerant and the bandwidth of the sideband link meets the transmission requirements, packets can be transmitted via the sideband link to lower power consumption.

The management adapter provides main/sideband link selection instructions for each output service flow (except ShuttleID 0) of each output port, as shown in Table 86.

Table 86 Main and sideband link selection instructions

Name	Bit Width	Description
MS	2	10b: Fixed main link. The link transmission unit sends packets to the main link transmission management unit. If the main link is in a low-power state (LPx), it must be awakened to the active state. 01b: Fixed sideband link. The link transmission unit sends packets to the sideband link transmission management unit. 00b/11b: Selection based on link status. When the logical layer main link is in the active state, packets are sent to the main link transmission management unit. Otherwise, packets are sent to the sideband link transmission management unit.

The link transmission unit sends packets to either the main or sideband link transmission management module based on the instructions. The link reception unit receives packets from both the main and sideband link reception management modules, performing main and sideband link combination.

7.8 Main Link Transmission and Reception Management

When the main link is in the active state, the link transmission unit needs to provide a continuous and stable data flow to the logical layer. If the transport layer has transmitted all available data, it must insert TLMIPs. These packets contain only the packet header, and their structure is shown in Figure 96.

Figure 96 TLMIP structure

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Rsvd										TLMP Type = 9						Length = 0									D = 0	ECC					

After receiving the TLMIP, the link reception unit performs ECC verification. If the ECC verification passes, the TLMIP should be identified and discarded. All other packets are transmitted to the combination of main and sideband links. If an ECCE is detected, the link reception unit corrects any single-bit ECCE and updates the "TLMIP ECCE correction counter". If an uncorrectable error is detected, the link reception unit discards the packet and updates the "TLMIP uncorrectable ECCE counter", and informs the management adapter and logic layer of the error event.

The link reception unit also performs ECC verification after receiving the other packets (non-TLMIPs). It corrects any single-bit ECCE and updates the "main link packet ECCE correction counter". If an uncorrectable error is detected, it discards the packet and updates the "main link packet uncorrectable ECCE counter", and informs the management adapter and logic layer of the error event.

7.9 Sideband Link Transmission and Reception Management

When transmitting packets over the sideband link, the link transmission unit must receive an acknowledgment packet from the link reception unit or wait for a period tNoTLSAP before sending the next packet. The specific value of tNoTLSAP is shown in Section 7.11. The TLSAP contains only a packet header. The "SMP Type" field is set to 0 to identify it as a sideband link acknowledgment packet. All other values are reserved. Its specific structure is shown in Figure 97.

Figure 97 TLSAP structure

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SMP Type = 0									TLMP Type = 12						Length = 0									D = 0	ECC					

If the link transmission unit does not receive a TLSAP within the tNoTLSAP period, it reports a timeout warning to the management adapter. The TLSAP has a higher priority than other packets transmitted on the sideband link and is transmitted first.

After receiving the TLSAP, the link reception unit performs ECC verification, corrects any single-bit ECCE and updates the "TLSAP ECCE correction counter". If an uncorrectable error is detected, it discards the packet and updates the "TLSAP uncorrectable ECCE counter".

7.10 Service Flow Establishment and Teardown

When establishing a service flow, the management adapter provides not only routing information but also the following ingress/egress flow information for each service flow, as summarized in Table 87.

Table 87 Ingress/Egress flow information summary

Name	Bit Width	Description
Vld	1	Valid service flow flag 0b indicates invalid 1b indicates valid
FC_STSH	2	Flow control mechanism 10b stands for an exclusive traffic control mechanism 01b stands for a shared flow control mechanism 00b/11b indicates the flow control disabling mechanism
Credit_Allocated_Shuttle	13	Number of credits allocated to the Shuttle in the RBuff. Valid when FC_STSH is an exclusive flow control mechanism.
MS	2	Main and sideband link selection identifier 10b: Fixed main link 01b: Fixed sideband link 00b/11b: Selection based on link status
Priority	3	Priority of the service flow. Priorities decrease sequentially from 0 to 7, that is, 0 corresponds to the highest priority, and 7 corresponds to the lowest priority.
Weight	8	Weight for Weighted Round Robin (WRR). Values 1–255 are valid; 0 indicates no participation in scheduling.

When establishing a forwarding path, the FwVld field of each forwarding entry in the routing information must be set to 1. When tearing down a forwarding path, the FwVld field of each forwarding entry in the routing information must be set to 0.

After the forwarding path is torn down, the link reception unit and link transmission unit must adhere to the following rules:

- Link reception unit: Discard all corresponding packets. If the discarded packets use a shared flow control mechanism, the number of their credits should be accounted for in Rx_Credits_Consumed_Shared and Rx_Credits_Recycled_Shared. For packets already in the cache, the whole packets should be discarded upon dequeue. If these discarded packets use a shared flow control mechanism, the number of their credits should be accounted for in Rx_Credits_Recycled_Shared.
- Link transmission unit: For packets using either an exclusive or shared flow control mechanism, the whole packets are discarded after scheduling. The number of credits of discarded shared flow control mechanism packets is not accounted for in Tx_Credits_Consumed_Shared.

Note: After tearing down a service flow, the data in the TBuff and RBuff must be cleared before the service flow can be re-established. The ingress/egress flow information must remain unchanged after the service flow is torn down until it is re-established.

7.11 Transport Layer Time Parameter

Table 88 Summary of transport layer time parameters

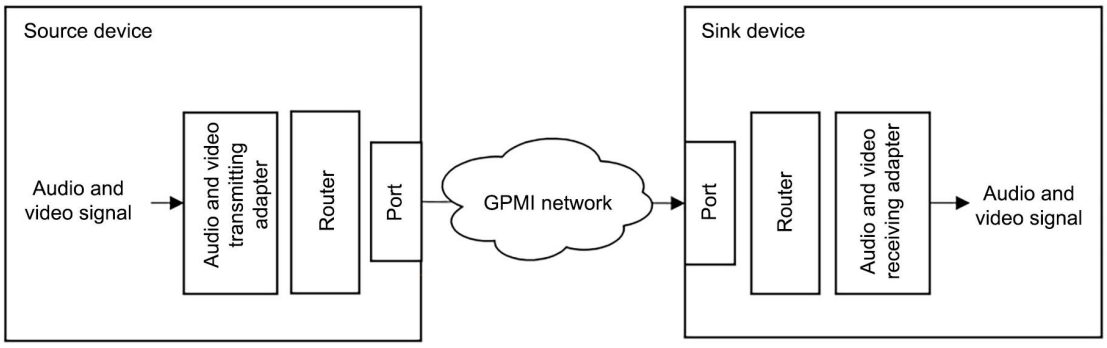
Parameter	Description	Minimum Value	Maximum Value	Unit
tNoTLSAP	Maximum time interval between transmitting the first bit of a packet to the sideband link and completely receiving a TLSAP from the sideband link.	-	10	ms
tTxTLCCPItv	Interval for sending TLCCPs	5	20	us
tTLCAP_ACK	Time interval between transmitting the first bit of a credit assignment entry and receiving the last bit of the TLCAP_ACK.	-	20	ms
tTxErrFC	Time for the transmitter to detect a flow control exception.	-	20	ms
tTLDPFwJitter	Maximum and minimum forwarding delay jitter of transport layer data packets in routers.	-	100	ns

8 Audio and Video Adapter

8.1 Overview

The audio and video adapter is a General Purpose Multimedia Interface audio and video signal processing unit. As shown in Figure 98, audio and video signals are converted into packets by the audio and video transmitting adapter. These packets are transmitted over the network to the corresponding audio and video receiving adapter, which then restores the packets to audio and video signals.

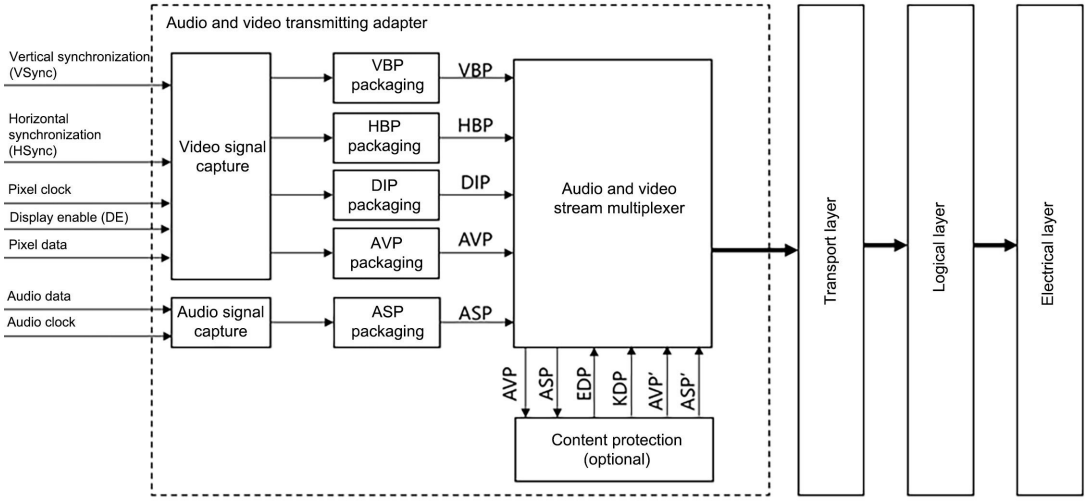
Figure 98 General Purpose Multimedia Interface video signal transmission and restoration



8.2 Logic Block Diagram of Audio and Video Adapter

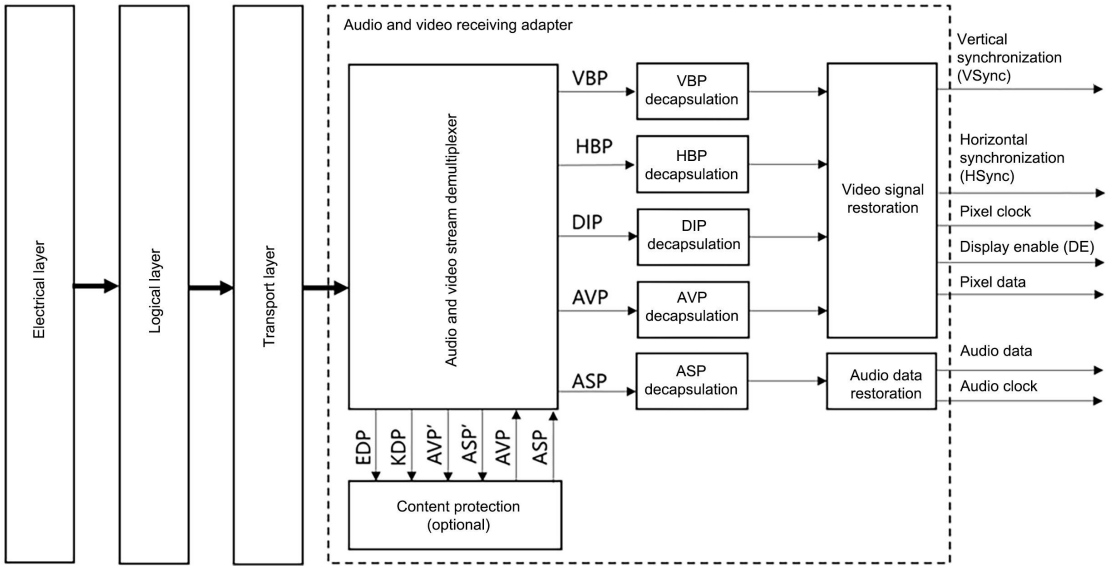
The logic block diagram of the audio and video transmitting adapter is shown in Figure 99. The audio and video transmitting adapter receives Vertical Synchronization (VSync), Horizontal Synchronization (HSync), pixel clock, Display Enable (DE), pixel data, audio data, and audio clock. It encapsulates these into corresponding packets, such as the Vertical Blanking Packet (VBP), Horizontal Blanking Packet (HBP), Active Video Packet (AVP), Audio Sample Packet (ASP), and Descriptive Information Packet (DIP). When content protection is enabled, the AVP and ASP are sent to an encryption module. This module encrypts the active video and audio samples to generate AVP' and ASP', and also produces an Encryption Description Packet (EDP) and a Key Distribution Packet (KDP). Finally, all packets are transmitted to the audio and video receiving adapter of the sink device through the transport layer and physical layer.

Figure 99 Logic block diagram of audio and video transmitting adapter



The logic block diagram of the audio and video receiving adapter is shown in Figure 100. The audio and video receiving adapter receives the audio and video stream and decomposes it into packets via a demultiplexer. When content protection is enabled, the decryption module outputs the AVP and ASP and ultimately restores VSync, HSync, pixel clock, DE, pixel data, audio data, and audio clock.

Figure 100 Logic block diagram of audio and video receiving adapter

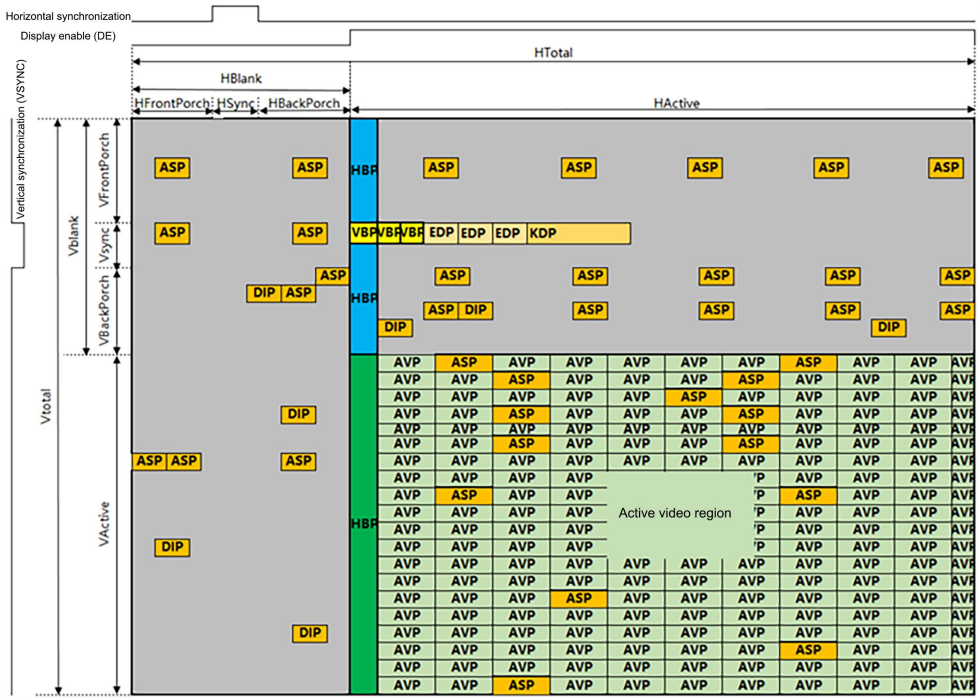


8.3 Audio and Video Signal

8.3.1 Video Frame and Video Timing

A video consists of multiple video frames, which are transmitted frame-by-frame through the General Purpose Multimedia Interface audio and video transmitter adapter. The video signal includes video data and video timing, where video timing indicates the structure of the video frame to be transmitted. An example of the structure of a single video frame is shown in Figure 101.

Figure 101 Video frame structure



Each video frame consists of multiple video lines, which are usually divided into the vertical blanking area, horizontal blanking area, and active video area. Video timing includes VSync, HSync, and DE, which represent the vertical synchronization signal, horizontal synchronization signal, and display enable signal, respectively. These signals are used to indicate the structure of the video frame.

Table 89 Abbreviations and descriptions in video frame structure

Abbreviation	Description
VSync	Vertical synchronization signal, the start mark of a video frame
HSync	Horizontal synchronization signal, the start mark of a video line
DE	Display enable signal, the start mark of an active video, video data transmission
VTotal	Number of video lines in one frame of video
VBlank	Number of video lines in the vertical blanking area
VActive	Number of video lines in the active video area
HTotal	Number of pixels in a video line
HBlank	Number of pixels in the horizontal blanking area of a video line
HActive	Number of pixels in the active video area of a video line

The transmission of video frames shall comply with the following provisions:

- The audio and video transmitting adapter shall transmit an HBP for each video line, and it shall be sent immediately at the end of the horizontal blanking.
- The audio and video transmitting adapter shall transmit three consecutive VBPs per frame. No ASP or DIP may be inserted between these VBPs. If VSync is positive, the adapter shall replace the HBP with a VBP at the video line where the rising edge of VSync occurs, followed immediately by two additional VBPs. If VSync is negative, the adapter shall replace the HBP with a VBP at the video line where the falling edge of VSync occurs, followed immediately by two additional VBPs.
- If content protection is enabled, the audio and video transmitting adapter shall transmit three EDPs after sending the VBP. In multicast scenarios, *K* KDPs (where *K* is the number of multicast devices) shall be transmitted immediately after the EDPs, and all must be transmitted before the next HBP.
- When DE is 1, the audio and video transmitting adapter obtains pixel data at the rising edge of the pixel clock (PixelClk). The pixel data is sequentially arranged to generate AVPs, which are then transmitted in order. Except for the AVP at the end of the video line, which may require padding data insertion, no padding data is inserted within the pixel arrangement of a single line. ASPs can be inserted between two AVPs, but the priority of the ASP is lower than that of the AVP.
- Audio data shall be encapsulated into ASPs for transmission. There are no restrictions on the timing of ASP transmission. Data packets carrying audio can be sent in the vertical blanking area, horizontal blanking area, or active video area.

8.3.2 Pixel Data Arrangement Rules

This section describes the arrangement scheme of pixel data in the active video area, including the arrangement method of pixel data in RGB, YcbCr444, YcbCr422, and YcbCr420 video formats (for specific examples, see Section 8.4.5):

—For continuous video pixels, the audio and video adapter should first arrange the first pixel data, and then arrange the second pixel data in the order of reception.

—For each line of video data, the pixel component arrangement is:

- For RGB format video, the audio and video adapter first arranges the R component, then the G component, and finally the B component.
- For YCbCr444 format video, the audio and video adapter first arranges the Cr component, then the Y component, and finally the Cb component.
- For YCbCr422 format video, the audio and video adapter first arranges the Y component, followed by the Cb or Cr components.
- For YCbCr420 format video, the audio and video adapter first arranges 2 Y components, and then arranges the Cb component for even-numbered lines and the Cr component for odd-numbered lines.

Pixels within each line shall be arranged continuously except at the end of the video line. When the pixel data at the end of a video line cannot fill up one symbol (1 symbol = 4 bytes), the audio and video adapter of the source device shall pad "0" bits for symbol alignment. This means that after the pixel data, "0" bits are padded until the last symbol containing pixel data is filled. The padding rules are as follows:

- "0" bits can only be padded in the AVP at the end of a video line.
- The length of padded "0" bits shall be less than the length of one symbol. There shall be no case where a symbol at the end of a video line contains only padding data.

The audio and video receiving adapter shall identify and discard the padded data using the HActivePixels in the Video Frame Control (VFC) information (see Section 8.4.2). Below is a calculation example for data padding, assuming a 10 bpc RGB video with a resolution of 1366 × 768.

Total data volume of one line of active video:

$$1366 \times 30 = 40980 \text{ bits}$$

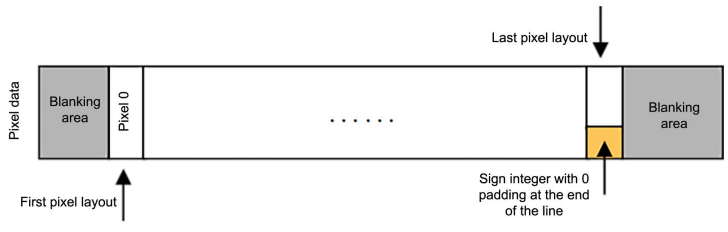
Total number of symbols after data arrangement:

$$40980 \text{ bits}/32 \text{ bits/symbol} \approx 1280.625 \text{ symbols}$$

After rounding up, the total number of symbols is 1281. The padding length in the last symbol is:

$$1281 \times 32 \text{ bits} - 40980 \text{ bits} = 12 \text{ bits}$$

Figure 102 Data padding



8.4 Audio and Video Adapter Packet

8.4.1 Audio and Video Adapter Packet Structure

Audio and video adapter packets consist of packet headers and payload. Their structure is shown in Figure 103, and the field description of the packet header is provided in Table 90.

Figure 103 Structure of audio and video adapter packet

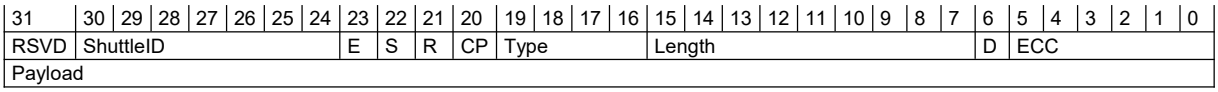


Table 90 Field description of audio and video adapter packet header

Field Name	Length (Bits)	Description
ECC	6	ECC, as defined in Section 7.3.1.
D	1	D is 1 bit, used to distinguish transport layer management packets.
Length	9	Indicates the length of the packet payload. The receiver uses this value to parse the data in the payload. Maximum length is 508 bytes.
Type	4	Type of audio and video adapter packets: 0000b: Undefined 0001b: VBP 0010b: HBP 0011b: DIP 0100b: ASP 0101b: AVP 0110b–1101b: Reserved 1110b: EDP 1111b: KDP
Flags	4	Flag field: CP: Content protection flag. 0b indicates content protection disabled; 1b indicates enabled. R: Reserved. S: Packet start flag. Marks the first packet of a line of pixels in an AVP, or the first packet of a DIP. E: Packet end flag. Marks the last packet of a line of pixels in an AVP, or the last packet of a DIP.
ShuttleID	7	Lane identifier for the corresponding service flow
RSVD	1	Reserved

8.4.2 VBP

After receiving the VSync signal, the audio and video transmitting adapter encapsulates the VBP to indicate the start of the vertical blanking area of a video frame. The structure of the VBP is shown in Figure 104.

Figure 104 VBP structure

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	ShuttleID							E = 1	S = 1	R = 0	CP = 0	Type = 1				Length = 28									D = 1	ECC					
Reserved			BitsPerComp					ALLM	DFR	QVT	Colorimetry					Reserved				PixelFormat				M	C	VFC_VER					
VP	HP	PixelClockFreq																													
HBlankPixels																HActivePixels															
HSyncPixels																HFrontPorch															
VBlankLines																VActiveLines															
VSyncLines																VFrontPorch															
CRC32																															

The transmission position of the VBP is shown in Figure 101. After the audio and video receiving adapter receives the VBP, it first performs the CRC. If the CRC fails, it proceeds to receive the next VBP until a correct VBP is received. In other words, the receiving adapter prioritizes the first accurately received VBP.

The VBP header description is shown in Table 91.

Table 91 VBP header description

Field Name	Length (Bits)	Description
ECC	6	ECC
D	1	Fixed at 1
Length	9	Fixed at 28
Type	4	Fixed at 0001b
Flags	4	CP: fixed at 0 R: fixed at 0 S: fixed at 1 E: fixed at 1
ShuttleID	7	Lane identifier for the corresponding service flow, ranging from 0 to 127
RSVD	1	Reserved

The payload of the VBP is defined as the VFC information, which describes the relevant parameters of the current video frame. The VFC field definitions are provided in Table 92.

Table 92 VFC field definition

Field Name	Length (Bits)	Description

Field Name	Length (Bits)	Description
VFC_VER	6	Version of the VFC information format. Default: 000001b; other values are reserved.
C CompressVideoMode	1	Perceptual Lossless Compression (PLLC) enable bit. 0b indicates PLLC disabled, and 1b indicates PLLC enabled.
M	1	Mute video flag When the video needs to output a blank screen, this bit is set to 1b. Upon receiving a video frame with MuteVideoFlag = 1, the sink device should handle it by displaying a black screen or retaining the previous frame.
PixelFormat	4	Pixel encoding format used for the current video frame: 0000b: RGB 0001b: YCbCr 4:4:4 0010b: YCbCr 4:2:2 0011b: YCbCr 4:2:0 0100b: Y-Only 0101b: ARGB 0110b: Raw Data Other values are reserved.
Reserved	4	Reserved
Colorimetry	5	Color space used for the video frame: RGB/ARGB: 00000b: sRGB 00001b: AdobeRGB 00010b: ITU-R BT.2020 R'G'B' 00011b: DCI-P3 R'G'B'(D65) 00100b: DCI-P3 R'G'B'(theater) Other values are reserved. YCbCr 4:4:4/YCbCr 4:2:2/YCbCr 4:2:0: 00000b: ITU-R BT.601 00001b: ITU-R BT.709 00010b: AdobeYCC601 00011b: ITU-R BT.2020 Y'CC'BCC'RC 00100b: ITU-R BT.2020 Y'CB'CR' 00101b: xvYCC601 00110b: xvYCC709 00111b: sYCC601

Field Name	Length (Bits)	Description
		Other values are reserved. Y-Only: 00000b: DICOM PART#14 Grayscale standard display Other values are reserved. RAW: 00000b: Custom configuration Other values are reserved.
QVT	1	Whether to enable fast video transmission mode 0b: disabled; 1b: enabled
DFR	1	Whether to enable dynamic frame rate refresh mode 0b: disabled; 1b: enabled
ALLM	1	Whether to enable the automatic low-latency mode 0b: disabled; 1b: enabled
BitsPerComp	5	Bit width of the video frame components: 01000b: 8bit 01010b: 10bit 01100b: 12bit 10000b: 16bit Other values are reserved.
Reserved	3	Reserved
PixelClockFreq	30	Video stream pixel clock frequency, in units of 1 kHz. Frequency deviation should be less than 500 ppm.
HP	1	HSync polarity 0b: high level; 1b: low level
VP	1	VSync polarity 0b: high level; 1b: low level
HActivePixels	16	Number of active horizontal pixels
HBlankPixels	16	Number of pixels in the horizontal blanking area
HFrontPorch	16	Number of front porch pixels in the horizontal blanking area
HsyncPixels	16	Number of Hsync pixels
VActiveLines	16	Number of active lines per frame
VBlankLines	16	Number of lines in the vertical blanking area
VFrontPorch	16	Number of front porch lines in the vertical blanking area
VSyncLines	16	VSync lines
CRC32	32	CRC check

8.4.3 HBP

HBPs consist of packet headers and payload. The structure of the HBP is shown in Figure 105.

Figure 105 HBP structure

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	ShuttleID							E = 1	S = 1	R = 0	CP = 0	Type = 2				Length = 8									D = 1	ECC					
VBlankFlag	Reserved																														
CRC32																															

HBP header fields are defined in Table 93. HBP payload mainly carries VBlank_Flag information, which is used to distinguish between VActive and VBlank areas. Its field definitions are shown in Table 94.

Table 93 HBP header field definition

Field Name	Length (Bits)	Description
ECC	6	ECC
D	1	Fixed at 1
Length	9	Fixed at 8
Type	4	Fixed at 0010b
Flags	4	CP: fixed at 0 R: fixed at 0 S: fixed at 1 E: fixed at 1
ShuttleID	7	Lane identifier for the corresponding service flow, ranging from 0 to 127
RSVD	1	Reserved

Table 94 HBP payload field definition

Field Name	Length (Bits)	Description
VBlankFlag	1	Distinguish between VActive and VBlank areas 0b: VActive area; 1b: VBlank area
Reserved	31	Reserved
CRC32	32	CRC check

8.4.4 DIP

8.4.4.1 DIP Structure and Description

DIPs are used to transmit configuration data related to video and audio. There are various types of DIPs, such as Audio Control, PLLC, Video Stream Info, Video Metadata, Vendor Extended, Source Product Description InfoFrame, Audio InfoFrame, and MPEG Source InfoFrame. DIPs are transmitted in the blanking area and have higher priority than ASPs. If other high-priority packets (such as VBPs or HBPs) are being transmitted, DIPs should be transmitted after these high-priority packets. The structure of the DIP is shown in Figure 106.

Figure 106 DIP structure

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	ShuttleID							E = 1	S = 1	R = 0	CP = 0	Type = 3				Length = 36									D = 1	ECC					
HB0								HB1								HB2								HB3							
DB0								DB1								DB2								DB3							
DB4								DB5								DB6								DB7							
DB8								DB9								DB10								DB11							
DB12								DB13								DB14								DB15							
DB16								DB17								DB18								DB19							
DB20								DB21								DB22								DB23							
DB24								DB25								DB26								DB27							
DB28								DB29								DB30								DB31							
CRC32																															

DIP header fields are defined in Table 95.

Table 95 DIP header field definitions

Field Name	Length (Bits)	Description
ECC	6	ECC
D	1	Fixed at 1
Length	9	Fixed at 40
Type	4	Fixed at 0011b
Flags	4	CP: fixed at 0 R: fixed at 0 S: The S flag of the first DIP is set to 1b when the length of a specific type of data exceeds 32 bytes and multiple DIPs are required to carry it, and is set to 0b in other cases. E: The E flag of the last DIP is set to 1b when the length of a specific type of data exceeds 32 bytes and multiple DIPs are required to carry it, and is set to 0b in other cases. Note: If the data length requires only one DIP, both S and E flags are set to 1b.
ShuttleID	7	Lane identifier for the corresponding service flow, ranging from 0 to 127
RSVD	1	Reserved

For the DIP payload, the field HB0 indicates the type of DIP. The DIP types and their corresponding HB0 values are defined in Table 96. The other fields and descriptions of the payload are determined by the DIP type, as detailed in Sections 8.4.4.2 to 8.4.4.7.

Table 96 DIP types

DIP Type	Value	Packet Transmission Frequency/Position
Reserved Reserved	00h	/
Audio Control	01h	When the audio format remains unchanged, it is transmitted once per frame in the vertical blanking area; when the audio format changes, it is transmitted immediately in the blanking area.
PLLC	02h	Transmitted once per video frame, after the VBP and before the last blanking line.
Video Stream Info	03h	Transmitted once per video frame in the vertical blanking area.
Video Metadata	04h	When HDR is enabled, transmitted once per video frame in the vertical blanking area, after the first line HBP and before the VBackPorch.
Vendor Extended	05h	Transmitted as needed in the blanking area, after the first line HBP in the vertical blanking area.
Reserved Reserved	06h to 7Fh	/
CTA-861-H Audio INFOFRAME	80h+Audio INFOFRAME Type(04h)	Consistent with that of Audio Control DIP.
CTA-861-H Non-audio INFOFRAME	80h+Non-Audio INFOFRAME Type	See the CTA-861-H standard.

A DIP can carry up to 32 bytes of descriptive information. When the length of data to be transmitted exceeds 32 bytes, multiple DIPs are required. In this case, the S flag in the header of the first DIP (DIP0) must be set to 1b to indicate the start, and the E flag in the header of the last DIP (DIP2) must be set to 1b to indicate the end. The S and E flags of intermediate packets should be set to 0b. If the data length can be carried by a single DIP, both S and E flags should be set to 1b. If the valid data in the last packet is less than 32 bytes, the length of the padding data is defined and identified by HB1–HB3 fields of the specific DIP type.

8.4.4.2 Audio Control DIP

The field definitions for the payload of the Audio Control DIP are provided in Table 97.

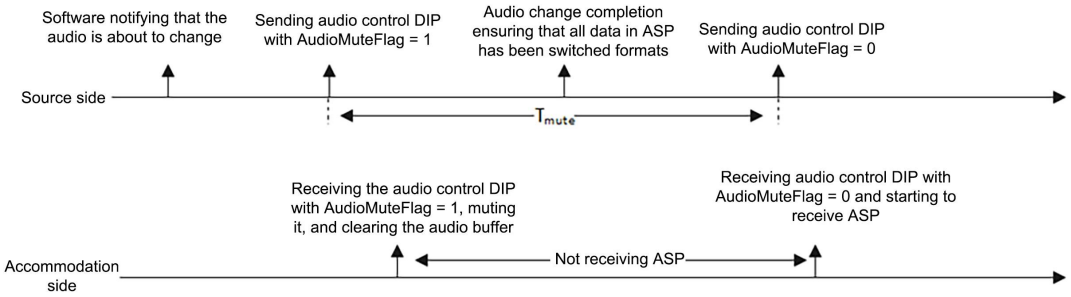
Table 97 Audio Control DIP payload field definitions

Byte	Name	Bit	Description
HB0	Descriptive Information Type	7:0	Type of audio control descriptive information, fixed at 01h
HB1	Reserved	7:0	Reserved
HB2	Version	7:0	Version number
HB3	Length	7:0	Fixed at 07h (DB0–DB6)
DB0	AudioMuteFlag	0	Audio mute flag bit When the audio is muted, it shall be set to 1b; otherwise, it shall be set to 0b
	Reserved	7:1	Reserved
DB1	AudioSampleFreq/Frame Rate	23:0	Audio sampling frequency or audio frame rate, in units of 1 Hz. The indicated audio sampling frequency or frame rate shall not deviate from the actual audio sampling frequency by more than 0.5%.
DB2			
DB3			
DB4	Audio Encoding Type	3:0	0000b: LPCM 0001b: IEC-61937 code Other values reserved
	High Bitrate Audio Flag	4	0b: non–high bitrate audio; 1b: high bitrate audio
	Reserved	7:5	Reserved
DB5	Number of Audio Channels	5:0	00000b: 2 channels 00001b: 4 channels 00010b: 6 channels … 01111b: 32 channels Other values are reserved.
	Audio Channel Parity Flag	6	0b: Actual number of channels is even 1b: Actual number of channels is odd
	Reserved	7	Reserved
DB6	Audio Sample Bit Width	1:0	00b:16bit 01b:20bit 10b:24bit 11b: reserved
	Reserved	7:2	Reserved
DB7–DB31	Reserved	/	Reserved

Byte	Name	Bit	Description
CRC32	CRC check	31:0	CRC32 value of HB0–DB31

The source device can set the AudioMuteFlag in the Audio Control DIP to 1b to mute audio. When switching audio formats (such as sampling frequency, number of lanes, and bit width), the source device shall set AudioMuteFlag to 1 and send the corresponding Audio Control DIP to prevent the sink device from outputting audio with audible artifacts such as noise or popping sounds. Figure 107 shows the recommended process for audio format switching.

Figure 107 Recommended process for audio format switching



When switching audio formats, the source device shall send an Audio Control DIP with AudioMuteFlag = 1. After the audio format switching is completed and T_{mute} is greater than 1 μs, the source device shall send another Audio Control DIP with AudioMuteFlag = 0. Upon receiving an Audio Control DIP with AudioMuteFlag = 1, the sink device shall mute the audio and clear the audio buffer. It shall not accept any ASP before receiving an Audio Control DIP with AudioMuteFlag = 0. After receiving the Audio Control DIP with AudioMuteFlag = 0, the sink device can receive ASPs.

DB1–DB3 in the Audio Control DIP payload identify the audio sampling frequency (AudioSampleFreq). The sampling frequency indicated by AudioSampleFreq shall not deviate from the actual audio sampling frequency by more than 0.5%. It should be noted that when transmitting high-bitrate IEC-61937 audio, the value in the AudioSampleFreq/Frame Rate field represents the audio sampling frequency/frame rate divided by 4.

DB5 in the Audio Control DIP payload indicates the number of audio channels. If the actual number of audio channels is even, the value of DB5 equals the true number of channels, and the Audio Channel Parity Flag is set to 0b. If the actual number of channels is odd, the value of DB5 equals the true number of channels plus 1, and the Audio Channel Parity Flag is set to 1b.

For all reserved fields, the source device must fill them with "0".

8.4.4.3 PLLC Parameter DIP

PLLC parameter DIP is used to transmit the compression parameter information of PLLC video. When sending PLLC video, a PLLC parameter DIP must be transmitted in the vertical blanking area of each video frame. The payload field definitions for the PLLC parameter DIP are provided in Table 98.

Table 98 PLLC parameter DIP payload field definitions

Byte	Name	Bit	Description
HB0	Descriptive Information Type	7:0	Fixed at 0x02.
HB1	INDEX	3:0	Sequence number of the PLLC parameter DIP, used to identify the current packet sequence and facilitate packet loss detection by the receiver.
	Reserved	7:4	Reserved
HB2	Version	7:0	Fixed at 0x01.
HB3	Length	7:0	Actual valid data length in this packet. When the E flag is 1b, this field indicates the number of valid bytes carried in the current DIP.
DB0	Payload 0	7:0	PH0, filled according to compression parameters.
DB1	Payload 1	7:0	PH1, filled according to compression parameters.
DB2	Payload 2	7:0	PH2, filled according to compression parameters.
…	…	7:0	…
DB29	Payload 29	7:0	PH29, filled according to compression parameters.
DB30	Payload 30	7:0	PH30, filled according to compression parameters.
DB31	Payload 31	7:0	PH31, filled according to compression parameters.
CRC32	CRC check	31:0	CRC32 value of HB0–DB31

When the length of the compression parameter is less than or equal to 32 bytes, it can be carried by 1 DIP. In this case, both the S flag and the E flag in the DIP header shall be set to 1, and the HB3 field shall indicate the actual compression parameter length.

When the length of the compression parameter is greater than 32 bytes, it needs to be encapsulated into multiple PLLC parameter DIPs. In this case, each packet carries 32 bytes of data (HB3 is fixed to 32). If the last packet has less than 32 bytes of data, it shall be padded with "0" bits, and the HB3 field shall indicate the actual number of valid bytes in that packet. In addition, the S flag in the first DIP shall be set to 1, and the E flag of the last DIP shall be set to 1. For detailed information on compression parameters, see *Perceptual Lossless Compression*.

8.4.4.4 Video Stream Info DIP

The Video Stream Info DIP is used to transmit functional information about the current video stream. It must be sent for every video frame. The payload field definitions for the Video Stream Info DIP are provided in Table 99.

Table 99 Video Stream Info DIP payload field definitions

Byte	Name	Bit	Description
HB0	Descriptive Information Type	7:0	Fixed at 03h
HB1	Reserved	7:0	Fixed at 0
HB2	Version	7:0	Fixed at 01h
HB3	Length	7:0	Fixed at 02h
DB0	CN	3:0	Video content type: 0000b: Unknown 0001b: Graphics 0010b: Photo 0011b: Cinema 0100b: Game Other values reserved
	QR	4	Quantitative range 0000b: Limited Range; 0001b: Full Range
	Rsvd	7:5	Reserved, filled with 0
DB1	AAR	3:0	Active aspect ratio: 0000b: Unknown 0001b: 4:3 0010b: 14:9 0011b: 16:9 Other values reserved
	PAR	7:4	Pixel aspect ratio: 0000b: Unknown 0001b: 4:3 0010b: 14:9 0011b: 16:9 Other values reserved
DB2–DB31	Reserved		Reserved
CRC32	Verification	31:0	CRC32 value of HB0–DB31

8.4.4.5 Video Metadata DIP

Video Metadata DIP is used to transmit video metadata information, and the field definitions of its payload are provided in Table 100.

Table 100 Video Metadata DIP payload field definitions

Byte	Name	Bit	Description
HB0	Descriptive Information Type	7:0	Fixed at 04h
HB1	Index	3:0	Sequence number of the Video Metadata DIP, used to identify the current packet sequence and facilitate packet loss detection by the receiver.
	Reserved	7:4	Reserved
HB2	Organization	7:4	Metadata standard organization: 0000b: SUCA 0001b: UWA 0010b: CTA 0011b: VESA Other values reserved
	Metadata Type	3:0	Metadata type, used to distinguish different metadata types of the same issuing organization
HB3	Length	7:0	Actual valid data length in this packet
DB0	Payload 0	7:0	Payload 0, filled according to actual metadata information
DB1	Payload 1	7:0	Payload 1, filled according to actual metadata information
DB2	Payload 2	7:0	Payload 2, filled according to actual metadata information
…	…	7:0	…
DB29	Payload 29	7:0	Payload 29, filled according to actual metadata information
DB30	Payload 30	7:0	Payload 30, filled according to actual metadata information
DB31	Payload 31	7:0	Payload 31, filled according to actual metadata information
CRC32	CRC check	31:0	CRC32 value of HB0–DB31

When the length of video metadata information is less than 32 bytes, it can be carried by 1 Video Metadata DIP. In this case, both the S flag and the E flag in the Video Metadata DIP header shall be set to 1, and the HB3 field shall indicate the actual metadata length. When the length of video metadata is greater than 32 bytes, it needs to be encapsulated into multiple Video Metadata DIPs. In this case, each packet carries 32 bytes of data (HB3 is fixed to 32). If the last packet has less than 32 bytes of data, it shall be padded with "0" bits, and the HB3 field shall indicate the actual number of valid bytes in that packet.

8.4.4.6 Vendor Extended DIP

Vendor Extended DIP is used to transmit vendor-defined data, and the field definitions of its payload are provided in Table 101.

Table 101 Vendor Extended DIP payload field definitions

Byte	Name	Bit	Description
HB0	DIP Type	7:0	Vendor Extended DIP, fixed at 05h
HB1	Index	3:0	Sequence number of the Vendor Extended DIP, used to identify the current packet sequence and facilitate packet loss detection by the receiver.
	Reserved	7:4	Reserved
HB2	Version	7:0	Version number
HB3	Length	7:0	Actual valid data length in this packet
DB0	Payload 0	7:0	Vendor identification code
DB1	Payload 1	7:0	Vendor identification code
DB2	Payload 2	7:0	Defined by the vendor
DB3	Payload 3	7:0	Defined by the vendor
…	…	…	…
DB29	Payload 29	7:0	Defined by the vendor
DB30	Payload 30	7:0	Defined by the vendor
DB31	Payload 31	7:0	Defined by the vendor
CRC32	CRC check	31:0	CRC32 value of HB0–DB31

The vendor identification code is 2 bytes long. For details, see Section 9.2.2.1.1. When the length of the vendor extended data is less than 32 bytes, it can be carried by 1 Vendor Extended DIP. In this case, both the S flag and the E flag in the Vendor Extended DIP header shall be set to 1, and the HB3 field shall indicate the actual length of vendor-defined data. When the length of vendor extended data is greater than 32 bytes, it needs to be encapsulated into multiple Vendor Extended DIPs. In this case, each packet carries 32 bytes of data (HB3 is fixed to 32). If the last packet has less than 32 bytes of data, it shall be padded with "0" bits, and the HB3 field shall indicate the actual number of valid bytes in that packet.

8.4.4.7 Mapping Relationship Between InfoFrame DIP and CTA-861-H

The content of information frames in the General Purpose Multimedia Interface shall comply with the definitions specified in CTA-861-H and be carried via InfoFrame DIPs. InfoFrame DIPs include three categories: Source Product Description InfoFrame DIP, Audio InfoFrame DIP, and MPEG Source InfoFrame DIP. The field definitions of the InfoFrame DIP payload and their mapping relationship with information frames in CTA-861-H are provided in Table 102.

Table 102 InfoFrame DIP payload field definitions and mapping to information frames in CTA-861-H

Byte	Name	Bit	Description
HB0	DIP Type	7:0	Source Product Description InfoFrame DIP: fixed at 83h; Audio InfoFrame DIP: fixed at 0x84h; MPEG Source InfoFrame DIP: fixed at 0x85h
HB1	Reserved	7:0	Reserved
HB2	Version	7:0	Version number, corresponding to the version of information frames in CTA-861-H
HB3	Length	7:0	Information frame length, corresponding to the length of information frames in CTA-861-H
DB0	Payload 0	7:0	See CTA-861-H Data Byte1
DB1	Payload 1	7:0	See CTA-861-H Data Byte2
…	…	…	…
DB27	Payload 29	7:0	See CTA-861-H
DB28–DB31	Reserved	7:0	Reserved
CRC32	CRC check	31:0	CRC32 value of HB0–DB31

For details about Source Product Description InfoFrame DIP, Audio InfoFrame DIP, and MPEG Source InfoFrame DIP, see Sections 6.5, 6.6, and 6.7 of CTA-861-H, respectively.

The transmission mechanism for the Audio InfoFrame DIP is consistent with that of the Audio Control DIP; that is, when the audio format remains unchanged, it is transmitted once per frame in the vertical blanking area; when the audio format changes, it is transmitted immediately in the blanking area. If any information in the Audio InfoFrame DIP conflicts with that in the Audio Control DIP, the information in the Audio Control DIP takes precedence.

The mapping relationship between speaker positions and audio transmission lanes is defined in the Audio InfoFrame DIP. For details, see Sections 6.6.2, 6.6.3, and 6.6.4 of CTA-861-H.

8.4.5 AVP

8.4.5.1 AVP Structure and Description

The audio and video transmitting adapter encapsulates the video data into AVP in the order of reception. Each AVP is encapsulated with 32 bits per line, and its structure is shown in Figure 108.

Figure 108 AVP structure

31	30 | 29 | 28 | 27 | 26 | 25 | 24	23	22	21	20	19 | 18 | 17 | 16	15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7	6	5 | 4 | 3 | 2 | 1 | 0
RSVD	ShuttleID	E	S	R	CP	Type = 5	Length	D	ECC
Active video pixel data									

AVPs consist of packet headers and payload. The payload is used to transmit active video pixel data or compressed video pixel data, with a maximum length of 508 bytes. AVP header fields are defined in Table 103.

Table 103 AVP header description

Field Name	Length (Bits)	Description
ECC	6	ECC
D	1	Fixed at 1
Length	9	Length of the packet payload, 4-byte alignment. Maximum length is 508 bytes.
Type	4	Fixed at 0101b
Flags	4	CP: 0 indicates content protection disabled; 1 indicates content protection enabled. R: fixed at 0 S: the first AVP of a video line E: the last AVP of a video line
ShuttleID	7	Lane identifier for the corresponding service flow, ranging from 0 to 127.
RSVD	1	Reserved

8.4.5.2 RGB/YCbCr444 8 bpc Video Pixel Data Arrangement

Figure 109 8 bpc RGB 4:4:4 video pixel data arrangement

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4byte	RSVD	ShuttleID							E	S	R	CP	Type = 5				Length									D = 1	ECC					
8byte	R0[7:0]								G0[7:0]								B0[7:0]								R1[7:0]							
12byte	G1[7:0]								B1[7:0]								R2[7:0]								G2[7:0]							
16byte	B2[7:0]								R3[7:0]								G3[7:0]								B3[7:0]							
…																																
512byte	…																															

For 8 bpc YCbCr 4:4:4 video, replace R with Cr, G with Y, and B with Cb. The arrangement method is consistent with the above description.

8.4.5.3 RGB/YCbCr444 10 bpc Video Pixel Data Arrangement

Figure 110 10 bpc RGB 4:4:4 video pixel data arrangement

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4byte	RSVD	ShuttleID							E	S	R	CP	Type = 5				Length									D = 1	ECC					
8byte	R0[9:0]										G0[9:0]										B0[9:0]										R1[9:8]	
12byte	R1[7:0]								G1[9:0]										B1[9:0]										R2[9:6]			
16byte	R2[5:0]						G2[9:0]										B2[9:0]										R3[9:4]					
20byte	R3[3:0]				G3[9:0]								B3[9:0]												R4[9:2]							
24byte	R4[1:0]	G4[9:0]									B4[9:0]										R5[9:0]											
28byte	G5[9:0]								B5[9:0]								R6[9:0]														G6[9:8]	
32byte	G6[7:0]								B6[9:0]								R7[9:0]												G7[9:6]			
36byte	G7[5:0]						B7[9:0]										R8[9:0]										G8[9:4]					
…																																
512byte	…																															

For 10 bpc YCbCr 4:4:4 video, replace R with Cr, G with Y, and B with Cb. The arrangement method is consistent with the above description.

8.4.5.4 RGB/YCbCr444 12bpc Video Pixel Data Arrangement

Figure 111 12 bpc RGB 4:4:4 video pixel data arrangement

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4byte	RSVD	ShuttleID							E	S	R	CP	Type = 5				Length									D = 1	ECC					
8byte	R0[11:0]												G0[11:0]												B0[11:4]							
12byte	B0[3:0]				R1[11:0]												G1[11:0]												B1[11:8]			
16byte	B1[7:0]								R2[11:0]												G2[11:0]											
20byte	B2[11:0]												R3[11:0]												G3[11:4]							
24byte	G3[3:0]				B3[11:0]												R4[11:0]												G4[11:8]			
28byte	G4[7:0]								B4[11:0]												R5[11:0]											
…																																
512byte	…																															

For 12 bpc YCbCr 4:4:4 video, replace R with Cr, G with Y, and B with Cb. The arrangement method is consistent with the above description.

8.4.5.5 RGB/YCbCr444 16 bpc Video Pixel Data Arrangement

Figure 112 16 bpc RGB 4:4:4 video pixel data arrangement

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4byte	RSVD	ShuttleID							E	S	R	CP	Type = 5				Length									D = 1	ECC					
8byte	R0[15:0]																G0[15:0]															
12byte	B0[15:0]																R1[15:0]															
16byte	G1[15:0]																B1[15:0]															
…																																
512byte	…																															

For 16 bpc YCbCr 4:4:4 video, replace R with Cr, G with Y, and B with Cb. The arrangement method is consistent with the above description.

8.4.5.6 YCbCr422 8 bpc Video Pixel Data Arrangement

Figure 113 8 bpc YcbCr 4:2:2 video pixel data arrangement

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4byte	RSVD	ShuttleID							E	S	R	CP	Type = 5				Length										D = 1	ECC				
8byte	Y0[7:0]								Cb0[7:0]								Y1[7:0]								Cr0[7:0]							
12byte	Y2[7:0]								Cb2[7:0]								Y3[7:0]								Cr2[7:0]							
16byte	Y4[7:0]								Cb4[7:0]								Y5[7:0]								Cr4[7:0]							
…																																
512byte	…																															

8.4.5.7 YCbCr422 10 bpc Video Pixel Data Arrangement

Figure 114 10 bpc YcbCr 4:2:2 video pixel data arrangement

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4byte	RSVD	ShuttleID							E	S	R	CP	Type = 5				Length										D = 1	ECC				
8byte	Y0[9:0]										Cb0[9:0]										Y1[9:0]								Cr0[9:8]			
12byte	Cr0[7:0]								Y2[9:0]										Cb2[9:0]										Y3[9:6]			
16byte	Y3[5:0]						Cr2[9:0]										Y4[9:0]										Cb4[9:4]					
20byte	Cb4[3:0]			Y5[9:0]									Cr4[9:0]								Y6[9:2]											
24byte	Y6[1:0]		Cb6[9:0]								Y7[9:0]								Cr6[9:0]													
28byte	Y8[9:0]								Cb8[9:0]												Y9[9:0]								Cr8[9:8]		
32byte	Cr8[7:0]								Y10[9:0]										Cb10[9:0]										Y11[9:6]			
36byte	Y11[5:0]						Cr10[9:0]										Y12[9:0]										Cb12[9:4]					
…																																
512byte	…																															

8.4.5.8 YCbCr422 12 bpc Video Pixel Data Arrangement

Figure 115 12 bpc YcbCr 4:2:2 video pixel data arrangement

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4byte	RSVD	ShuttleID							E	S	R	CP	Type = 5				Length										D = 1	ECC				
8byte	Y0[11:0]												Cb[11:0]												Y1[11:4]							
12byte	Y1[3:0]				Cr0[11:0]												Y2[11:0]												Cb2[11:8]			
16byte	Cb2[7:0]								Y3[11:0]												Cr2[11:0]											
20byte	Y4[11:0]												Cb4[11:0]												Y5[11:4]							
24byte	Y5[3:0]				Cr4[11:0]												Y6[11:0]												Cb6[11:8]			
28byte	Cb6[7:0]								Y7[11:0]												Cr6[11:0]											
…																																
512byte	…																															

8.4.5.9 YCbCr422 16 bpc Video Pixel Data Arrangement

Figure 116 16 bpc YcbCr 4:2:2 video pixel data arrangement

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4byte	RSVD	ShuttleID							E	S	R	CP	Type = 5				Length										D = 1	ECC				
8byte	Y0[15:0]																Cb0[15:0]															
12byte	Y1[15:0]																Cr0[15:0]															
16byte	Y2[15:0]																Cb2[15:0]															
⋮																																
512byte	…																															

8.4.5.10 YCbCr420 8 bpc Video Pixel Data Arrangement

Figure 117 8 bpc YCbCr 4:2:0 video pixel data even line arrangement

	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
4byte	RSVD ShuttleID	E S R CP Type = 5	Length	D = 1 ECC
8byte	Y1[7:0]	Y0[7:0]	Cb0[7:0]	Y3[7:0]
12byte	Y2[7:0]	Cb2[7:0]	Y5[7:0]	Y4[7:0]
16byte	Cb4[7:0]	Y7[7:0]	Y6[7:0]	Cb6[7:0]
⋮				
512byte	…			

Figure 118 8 bpc YCbCr 4:2:0 video pixel data odd line arrangement

	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
4byte	RSVD ShuttleID	E S R CP Type = 5	Length	D = 1 ECC
8byte	Y1[7:0]	Y0[7:0]	Cr0[7:0]	Y3[7:0]
12byte	Y2[7:0]	Cr2[7:0]	Y5[7:0]	Y4[7:0]
16byte	Cr4[7:0]	Y7[7:0]	Y6[7:0]	Cr6[7:0]
⋮				
512byte	…			

8.4.5.11 YCbCr420 10 bpc Video Pixel Data Arrangement

Figure 119 10 bpc YCbCr 4:2:0 video pixel data even line arrangement

	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
4byte	RSVD ShuttleID	E S R CP Type = 5	Length	D = 1 ECC
8byte	Y1[9:0]	Y0[9:0]	Cb0[9:0]	Y3[9:8]
12byte	Y3[7:0]	Y2[9:0]	Cb2[9:0]	Y5[9:6]
16byte	Y5[5:0]	Y4[9:0]	Cb4[9:0]	Y7[9:4]
20byte	Y7[3:0]	Y6[9:0]	Cb6[9:0]	Y9[9:2]
24byte	Y9[1:0]	Y8[9:0]	Cb8[9:0]	Y11[9:0]
28byte	Y10[9:0]	Cb10[9:0]	Y13[9:0]	Y12[9:8]
32byte	Y12[7:0]	Cb12[9:0]	Y15[9:0]	Y14[9:6]
36byte	Y14[5:0]	Cb14[9:0]	Y17[9:0]	Y16[9:4]
⋮				
512byte	…			

Figure 120 10 bpc YCbCr 4:2:0 video pixel data odd line arrangement

	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
4byte	RSVD ShuttleID	E S R CP Type = 5	Length	D = 1 ECC
8byte	Y1[9:0]	Y0[9:0]	Cr0[9:0]	Y3[9:8]
12byte	Y3[7:0]	Y2[9:0]	Cr2[9:0]	Y5[9:6]
16byte	Y5[5:0]	Y4[9:0]	Cr4[9:0]	Y7[9:4]
20byte	Y7[3:0]	Y6[9:0]	Cr6[9:0]	Y9[9:2]
24byte	Y9[1:0]	Y8[9:0]	Cr8[9:0]	Y11[9:0]
28byte	Y10[9:0]	Cr10[9:0]	Y13[9:0]	Y12[9:8]
32byte	Y12[7:0]	Cr12[9:0]	Y15[9:0]	Y14[9:6]
36byte	Y14[5:0]	Cr14[9:0]	Y17[9:0]	Y16[9:4]
⋮				
512byte	…			

8.4.5.12 YCbCr420 12 bpc Video Pixel Data Arrangement

Figure 121 12 bpc YCbCr 4:2:0 video pixel data even line arrangement

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4byte	RSVD			ShuttleID					E	S	R	CP	Type = 5				Length									D = 1	ECC					
8byte	Y1[11:0]												Y0[11:0]												Cb0[11:4]							
12byte	Cb0[3:0]				Y3[11:0]												Y2[11:0]														Cb2[11:8]		
16byte	Cb2[7:0]								Y5[11:0]												Y4[11:0]											
20byte	Cb4[11:0]												Y7[11:0]												Y6[11:4]							
24byte	Y6[3:0]				Cb6[11:0]												Y9[11:0]														Y8[11:8]		
28byte	Y8[7:0]								Cb8[11:0]												Y11[11:0]											
⋮																																
512byte	…																															

Figure 122 12 bpc YCbCr 4:2:0 video pixel data odd line arrangement

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4byte	RSVD			ShuttleID					E	S	R	CP	Type = 5				Length									D = 1	ECC					
8byte	Y1[11:0]												Y0[11:0]												Cr0[11:4]							
12byte	Cr0[3:0]				Y3[11:0]												Y2[11:0]														Cr2[11:8]		
16byte	Cr2[7:0]								Y5[11:0]												Y4[11:0]											
20byte	Cr4[11:0]												Y7[11:0]												Y6[11:4]							
24byte	Y6[3:0]				Cr6[11:0]												Y9[11:0]														Y8[11:8]		
28byte	Y8[7:0]								Cr8[11:0]												Y11[11:0]											
⋮																																
512byte	…																															

8.4.5.13 YCbCr420 16 bpc Video Pixel Data Arrangement

Figure 123 16 bpc YCbCr 4:2:0 video pixel data even line arrangement

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4byte	RSVD			ShuttleID					E	S	R	CP	Type = 5				Length									D = 1	ECC					
8byte	Y1[15:0]																Y0[15:0]															
12byte	Cb0[15:0]																Y3[15:0]															
16byte	Y2[15:0]																Cb2[15:0]															
⋮																																
512byte	…																															

Figure 124 16 bpc YCbCr 4:2:0 video pixel data odd line arrangement

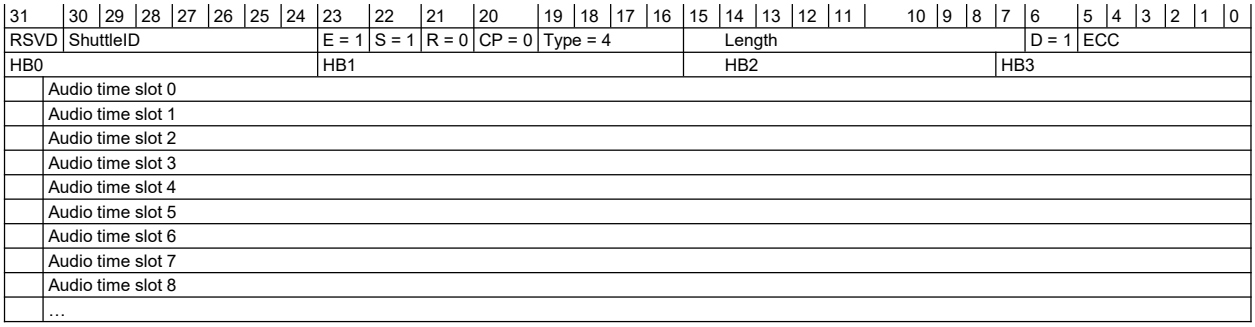
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4byte	RSVD			ShuttleID					E	S	R	CP	Type = 5				Length									D = 1	ECC					
8byte	Y1[15:0]																Y0[15:0]															
12byte	Cr0[15:0]																Y3[15:0]															
16byte	Y2[15:0]																Cr2[15:0]															
⋮																																
512byte	…																															

8.4.6 ASP

8.4.6.1 ASP Structure and Description

The audio and video transmitting adapter encapsulates the audio data that follows the IEC 60958 or IEC 61937 protocol into ASPs for audio data transmission. The ASP structure is shown in Figure 125. The payload length is variable. An audio sampling point cannot be transmitted across ASPs, and an ASP can contain multiple audio sampling points.

Figure 125 ASP structure



ASP header field definitions are provided in Table 104. HB0–HB3 are reserved bits and fixed to 0.

Table 104 ASP header field definitions

Field Name	Length (Bits)	Description
ECC	6	ECC
D	1	Fixed at 1
Length	9	Packet payload length, 4-byte alignment
Type	4	Fixed at 0100b
Flags	4	CP: 0b indicates content protection disabled; 1b indicates content protection enabled. R: fixed at 0 S: fixed at 1 E: fixed at 1
ShuttleID	7	Lane identifier for the corresponding service flow, ranging from 0 to 127.
RSVD	1	Reserved

8.4.6.2 Audio Time Slot

The General Purpose Multimedia Interface encapsulates audio data into 4-byte audio time slots. The field definitions of each audio time slot are provided in Table 105.

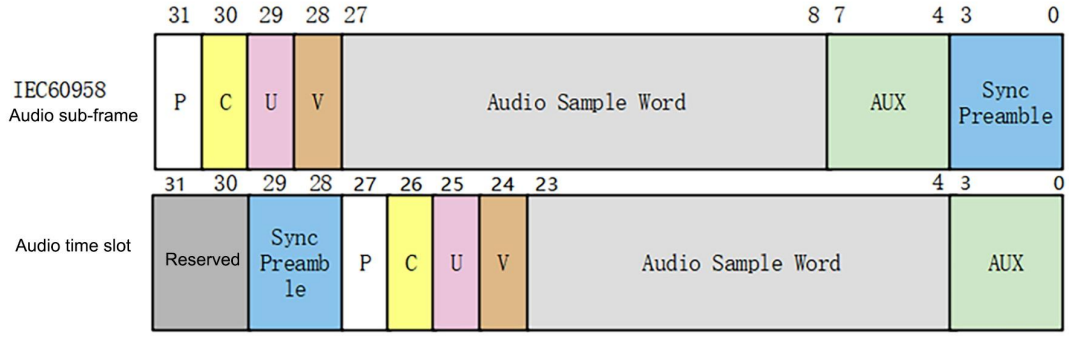
Table 105 Audio time slot field definitions

Field Name	Bit	Description
AUX	3:0	Corresponds to the AUX field of the IEC 60958 audio sub-frame.
Audio sampling data	23:4	Corresponds to the Audio Sample Word field of the IEC 60958 audio sub-frame.

Field Name	Bit	Description
PCUV	27:24	Corresponds to the P, C, U, and V fields of the IEC 60958 audio sub-frame.
Preamble	29:28	Indicates the audio sub-frame header being transmitted in the current audio time slot: 00b: B or Z, start of IEC60958 Block 01b: M or X, Sub-Frame 1 10b: W or Y, Sub-Frame 2 11b: Reserved
Reserved	31:30	Reserved

The mapping relationship between the audio sub-frame and the audio time slot of IEC 60958 is shown in Figure 126. The structure of the audio sub-frame shall comply with the requirements of Figure 1 in IEC 60958-1.

Figure 126 Field mapping between audio sub-frames and audio time slots



8.4.6.3 IEC 60958 LPCM Audio

When transmitting LPCM-format audio data via ASPs, the configuration of the Audio Control DIP shall adhere to the following rules:

- The Audio Encoding Type field shall be set to 0000b.
- If the actual number of audio channels is even, the value of the Number of Audio Channels field equals the true number of channels, and the Audio Channel Parity Flag is set to 0b. If the actual number of channels is odd, the value of the Number of Audio Channels field equals the true number of channels plus 1, and the Audio Channel Parity Flag is set to 1b.

The sampling and transmission rules for multi-channel audio data are as follows:

- Audio data that completes sampling first is transmitted first.
- Transmitted in ascending order (from lowest to highest channel number).
- After the data of all channels in the current audio sampling point is transmitted, the data of the next audio sampling point will be transmitted.

Below is an example of LPCM audio data arrangement. Sx denotes the x-th audio sample point. CH1-2, CH3-4, CH5-6, and CH7-8 represent channel 1 or 2, channel 3 or 4, channel 5 or 6, channel 7 or 8, respectively. SF1 and SF2 correspond to Sub-Frame 1 (with preamble B/Z or M/X) and Sub-Frame 2 (with preamble W/Y) of IEC60958, respectively.

The data arrangement for 2 audio channels transmitting 7 audio sampling points is illustrated in Figure 127.

Figure 127 Data arrangement example for 2 channels transmitting 7 audio sampling points

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				ShuttleID				E = 1	S = 1	R = 0	CP = 0	Type = 4				Length = 60									D = 1	ECC					
HB0								HB1								HB2								HB3							
S0.CH1-2.SF1																															
S0.CH1-2.SF2																															
S1.CH1-2.SF1																															
S1.CH1-2.SF2																															
S2.CH1-2.SF1																															
S2.CH1-2.SF2																															
S3.CH1-2.SF1																															
S3.CH1-2.SF2																															
S4.CH1-2.SF1																															
S4.CH1-2.SF2																															
S5.CH1-2.SF1																															
S5.CH1-2.SF2																															
S6.CH1-2.SF1																															
S6.CH1-2.SF2																															

The data arrangement for 4 audio channels transmitting 3 audio sampling points is illustrated in Figure 128.

Figure 128 Data arrangement example for 4 channels transmitting 3 audio sampling points

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD				ShuttleID				E = 1	S = 1	R = 0	CP = 0	Type = 4				Length = 52									D = 1	ECC					
HB0								HB1								HB2								HB3							
S0.CH1-2.SF1																															
S0.CH1-2.SF2																															
S0.CH3-4.SF1																															
S0.CH3-4.SF2																															
S1.CH1-2.SF1																															
S1.CH1-2.SF2																															
S1.CH3-4.SF1																															
S1.CH3-4.SF2																															
S2.CH1-2.SF1																															
S2.CH1-2.SF2																															
S2.CH3-4.SF1																															
S2.CH3-4.SF2																															

The data arrangement for 6 audio channels transmitting 2 audio sampling points is illustrated in Figure 129.

Figure 129 Data arrangement example for 6 channels transmitting 2 audio sampling points

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	ShuttleID							E = 1	S = 1	R = 0	CP = 0	Type = 4				Length = 52									D = 1	ECC					
HB0								HB1								HB2								HB3							
S0.CH1-2.SF1																															
S0.CH1-2.SF2																															
S0.CH3-4.SF1																															
S0.CH3-4.SF2																															
S0.CH5-6.SF1																															
S0.CH5-6.SF2																															
S1.CH1-2.SF1																															
S1.CH1-2.SF2																															
S1.CH3-4.SF1																															
S1.CH3-4.SF2																															
S1.CH5-6.SF1																															
S1.CH5-6.SF2																															

The data arrangement for 8 audio channels transmitting 1 audio sampling point is illustrated in Figure 130.

Figure 130 Data arrangement example for 8 channels transmitting 1 audio sampling point

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	ShuttleID							E = 1	S = 1	R = 0	CP = 0	Type = 4				Length = 36									D = 1	ECC					
HB0								HB1								HB2								HB3							
S0.CH1-2.SF1																															
S0.CH1-2.SF2																															
S0.CH3-4.SF1																															
S0.CH3-4.SF2																															
S0.CH5-6.SF1																															
S0.CH5-6.SF2																															
S0.CH7-8.SF1																															
S0.CH7-8.SF2																															

The data arrangement for 10 audio channels transmitting 1 audio sampling point is illustrated in Figure 131.

Figure 131 Data arrangement example for 10 channels transmitting 1 audio sampling point

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	ShuttleID							E = 1	S = 1	R = 0	CP = 0	Type = 4				Length = 44									D = 1	ECC					
HB0								HB1								HB2								HB3							
S0.CH1-2.SF1																															
S0.CH1-2.SF2																															
S0.CH3-4.SF1																															
S0.CH3-4.SF2																															
S0.CH5-6.SF1																															
S0.CH5-6.SF2																															
S0.CH7-8.SF1																															
S0.CH7-8.SF2																															
S0.CH9-10.SF1																															
S0.CH9-10.SF2																															

The data arrangement for 12 audio channels transmitting 1 audio sampling point is illustrated in Figure 132.

Figure 132 Data arrangement example for 12 channels transmitting 1 audio sampling point

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	ShuttleID							E = 1	S = 1	R = 0	CP = 0	Type = 4				Length = 52									D = 1	ECC					
HB0								HB1								HB2								HB3							
S0.CH1-2.SF1																															
S0.CH1-2.SF2																															
S0.CH3-4.SF1																															
S0.CH3-4.SF2																															
S0.CH5-6.SF1																															
S0.CH5-6.SF2																															
S0.CH7-8.SF1																															
S0.CH7-8.SF2																															
S0.CH9-10.SF1																															
S0.CH9-10.SF2																															
S0.CH11-12.SF1																															
S0.CH11-12.SF2																															

The data arrangement for 14 audio channels transmitting 1 audio sampling point is illustrated in Figure 133.

Figure 133 Data arrangement example for 14 channels transmitting 1 audio sampling point

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	ShuttleID							E = 1	S = 1	R = 0	CP = 0	Type = 4				Length = 60									D = 1	ECC					
HB0								HB1								HB2								HB3							
S0.CH1-2.SF1																															
S0.CH1-2.SF2																															
S0.CH3-4.SF1																															
S0.CH3-4.SF2																															
S0.CH5-6.SF1																															
S0.CH5-6.SF2																															
S0.CH7-8.SF1																															
S0.CH7-8.SF2																															
S0.CH9-10.SF1																															
S0.CH9-10.SF2																															
S0.CH11-12.SF1																															
S0.CH11-12.SF2																															
S0.CH13-14.SF1																															
S0.CH13-14.SF2																															

The data arrangement for 16 audio channels transmitting 1 audio sampling point is illustrated in Figure 134.

Figure 134 Data arrangement example for 16 channels transmitting 1 audio sampling point

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	ShuttleID							E = 1	S = 1	R = 0	CP = 0	Type = 4				Length = 68									D = 1	ECC					
HB0								HB1								HB2								HB3							
S0.CH1-2.SF1																															
S0.CH1-2.SF2																															
S0.CH3-4.SF1																															
S0.CH3-4.SF2																															
S0.CH5-6.SF1																															
S0.CH5-6.SF2																															
S0.CH7-8.SF1																															
S0.CH7-8.SF2																															
S0.CH9-10.SF1																															
S0.CH9-10.SF2																															
S0.CH11-12.SF1																															
S0.CH11-12.SF2																															
S0.CH13-14.SF1																															
S0.CH13-14.SF2																															
S0.CH15-16.SF1																															
S0.CH15-16.SF2																															

The data arrangement for 18 audio channels transmitting 1 audio sampling point is illustrated in Figure 135.

Figure 135 Data arrangement example for 18 channels transmitting 1 audio sampling point

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	ShuttleID							E = 1	S = 1	R = 0	CP = 0	Type = 4				Length = 76									D = 1	ECC					
HB0								HB1								HB2								HB3							
S0.CH1-2.SF1																															
S0.CH1-2.SF2																															
S0.CH3-4.SF1																															
S0.CH3-4.SF2																															
S0.CH5-6.SF1																															
S0.CH5-6.SF2																															
S0.CH7-8.SF1																															
S0.CH7-8.SF2																															
S0.CH9-10.SF1																															
S0.CH9-10.SF2																															
S0.CH11-12.SF1																															
S0.CH11-12.SF2																															
S0.CH13-14.SF1																															
S0.CH13-14.SF2																															
S0.CH15-16.SF1																															
S0.CH15-16.SF2																															
S0.CH17-18.SF1																															
S0.CH17-18.SF2																															

The data arrangement for 20 audio channels transmitting 1 audio sampling point is illustrated in Figure 136.

Figure 136 Data arrangement example for 20 channels transmitting 1 audio sampling point

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	ShuttleID							E = 1	S = 1	R = 0	CP = 0	Type = 4				Length = 84									D = 1	ECC					
HB0								HB1								HB2								HB3							
S0.CH1-2.SF1																															
S0.CH1-2.SF2																															
S0.CH3-4.SF1																															
S0.CH3-4.SF2																															
S0.CH5-6.SF1																															
S0.CH5-6.SF2																															
S0.CH7-8.SF1																															
S0.CH7-8.SF2																															
S0.CH9-10.SF1																															
S0.CH9-10.SF2																															
S0.CH11-12.SF1																															
S0.CH11-12.SF2																															
S0.CH13-14.SF1																															
S0.CH13-14.SF2																															
S0.CH15-16.SF1																															
S0.CH15-16.SF2																															
S0.CH17-18.SF1																															
S0.CH17-18.SF2																															
S0.CH19-20.SF1																															
S0.CH19-20.SF2																															

The data arrangement for 22 audio channels transmitting 1 audio sampling point is illustrated in Figure 137.

Figure 137 Data arrangement example for 22 channels transmitting 1 audio sampling point

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								E = 1	S = 1	R = 0	CP = 0	Type = 4				Length = 92									D = 1	ECC					
HB0								HB1								HB2								HB3							
S0.CH1-2.SF1																															
S0.CH1-2.SF2																															
S0.CH3-4.SF1																															
S0.CH3-4.SF2																															
S0.CH5-6.SF1																															
S0.CH5-6.SF2																															
S0.CH7-8.SF1																															
S0.CH7-8.SF2																															
S0.CH9-10.SF1																															
S0.CH9-10.SF2																															
S0.CH11-12.SF1																															
S0.CH11-12.SF2																															
S0.CH13-14.SF1																															
S0.CH13-14.SF2																															
S0.CH15-16.SF1																															
S0.CH15-16.SF2																															
S0.CH17-18.SF1																															
S0.CH17-18.SF2																															
S0.CH19-20.SF1																															
S0.CH19-20.SF2																															
S0.CH21-22.SF1																															
S0.CH21-22.SF2																															

The data arrangement for 24 audio channels transmitting 1 audio sampling point is illustrated in Figure 138.

Figure 138 Data arrangement example for 24 channels transmitting 1 audio sampling point

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								E = 1	S = 1	R = 0	CP = 0	Type = 4				Length = 100									D = 1	ECC					
HB0								HB1								HB2								HB3							
S0.CH1-2.SF1																															
S0.CH1-2.SF2																															
S0.CH3-4.SF1																															
S0.CH3-4.SF2																															
S0.CH5-6.SF1																															
S0.CH5-6.SF2																															
S0.CH7-8.SF1																															
S0.CH7-8.SF2																															
S0.CH9-10.SF1																															
S0.CH9-10.SF2																															
S0.CH11-12.SF1																															
S0.CH11-12.SF2																															
S0.CH13-14.SF1																															
S0.CH13-14.SF2																															
S0.CH15-16.SF1																															
S0.CH15-16.SF2																															
S0.CH17-18.SF1																															
S0.CH17-18.SF2																															
S0.CH19-20.SF1																															
S0.CH19-20.SF2																															
S0.CH21-22.SF1																															
S0.CH21-22.SF2																															
S0.CH23-24.SF1																															
S0.CH23-24.SF2																															

The data arrangement for 26 audio channels transmitting 1 audio sampling point is illustrated in Figure 139.

Figure 139 Data arrangement example for 26 channels transmitting 1 audio sampling point

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								E = 1	S = 1	R = 0	CP = 0	Type = 4				Length = 108									D = 1	ECC					
HB0								HB1								HB2								HB3							
S0.CH1-2.SF1																															
S0.CH1-2.SF2																															
S0.CH3-4.SF1																															
S0.CH3-4.SF2																															
S0.CH5-6.SF1																															
S0.CH5-6.SF2																															
S0.CH7-8.SF1																															
S0.CH7-8.SF2																															
S0.CH9-10.SF1																															
S0.CH9-10.SF2																															
S0.CH11-12.SF1																															
S0.CH11-12.SF2																															
S0.CH13-14.SF1																															
S0.CH13-14.SF2																															
S0.CH15-16.SF1																															
S0.CH15-16.SF2																															
S0.CH17-18.SF1																															
S0.CH17-18.SF2																															
S0.CH19-20.SF1																															
S0.CH19-20.SF2																															
S0.CH21-22.SF1																															
S0.CH21-22.SF2																															
S0.CH23-24.SF1																															
S0.CH23-24.SF2																															
S0.CH25-26.SF1																															
S0.CH25-26.SF2																															

The data arrangement for 28 audio channels transmitting 1 audio sampling point is illustrated in Figure 140.

Figure 140 Data arrangement example for 28 channels transmitting 1 audio sampling point

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								E = 1	S = 1	R = 0	CP = 0	Type = 4				Length = 116									D = 1	ECC					
HB0								HB1								HB2								HB3							
S0.CH1-2.SF1																															
S0.CH1-2.SF2																															
S0.CH3-4.SF1																															
S0.CH3-4.SF2																															
S0.CH5-6.SF1																															
S0.CH5-6.SF2																															
S0.CH7-8.SF1																															
S0.CH7-8.SF2																															
S0.CH9-10.SF1																															
S0.CH9-10.SF2																															
S0.CH11-12.SF1																															
S0.CH11-12.SF2																															
S0.CH13-14.SF1																															
S0.CH13-14.SF2																															
S0.CH15-16.SF1																															
S0.CH15-16.SF2																															
S0.CH17-18.SF1																															
S0.CH17-18.SF2																															
S0.CH19-20.SF1																															
S0.CH19-20.SF2																															
S0.CH21-22.SF1																															
S0.CH21-22.SF2																															
S0.CH23-24.SF1																															
S0.CH23-24.SF2																															
S0.CH25-26.SF1																															
S0.CH25-26.SF2																															
S0.CH27-28.SF1																															
S0.CH27-28.SF2																															

The data arrangement for 30 audio channels transmitting 1 audio sampling point is illustrated in Figure 141.

Figure 141 Data arrangement example for 30 channels transmitting 1 audio sampling point

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	ShuttleID							E = 1	S = 1	R = 0	CP = 0	Type = 4				Length = 124									D = 1	ECC					
HB0								HB1								HB2								HB3							
S0.CH1-2.SF1																															
S0.CH1-2.SF2																															
S0.CH3-4.SF1																															
S0.CH3-4.SF2																															
S0.CH5-6.SF1																															
S0.CH5-6.SF2																															
S0.CH7-8.SF1																															
S0.CH7-8.SF2																															
S0.CH9-10.SF1																															
S0.CH9-10.SF2																															
S0.CH11-12.SF1																															
S0.CH11-12.SF2																															
S0.CH13-14.SF1																															
S0.CH13-14.SF2																															
S0.CH15-16.SF1																															
S0.CH15-16.SF2																															
S0.CH17-18.SF1																															
S0.CH17-18.SF2																															
S0.CH19-20.SF1																															
S0.CH19-20.SF2																															
S0.CH21-22.SF1																															
S0.CH21-22.SF2																															
S0.CH23-24.SF1																															
S0.CH23-24.SF2																															
S0.CH25-26.SF1																															
S0.CH25-26.SF2																															
S0.CH27-28.SF1																															
S0.CH27-28.SF2																															
S0.CH29-30.SF1																															
S0.CH29-30.SF2																															

The data arrangement for 32 audio channels transmitting 1 audio sampling point is illustrated in Figure 142.

Figure 142 Data arrangement example for 32 channels transmitting 1 audio sampling point

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	ShuttleID							E = 1	S = 1	R = 0	CP = 0	Type = 4				Length = 132									D = 1	ECC					
HB0								HB1								HB2								HB3							
S0.CH1-2.SF1																															
S0.CH1-2.SF2																															
S0.CH3-4.SF1																															
S0.CH3-4.SF2																															
S0.CH5-6.SF1																															
S0.CH5-6.SF2																															
S0.CH7-8.SF1																															
S0.CH7-8.SF2																															
S0.CH9-10.SF1																															
S0.CH9-10.SF2																															
S0.CH11-12.SF1																															
S0.CH11-12.SF2																															
S0.CH13-14.SF1																															
S0.CH13-14.SF2																															
S0.CH15-16.SF1																															
S0.CH15-16.SF2																															
S0.CH17-18.SF1																															
S0.CH17-18.SF2																															
S0.CH19-20.SF1																															
S0.CH19-20.SF2																															
S0.CH21-22.SF1																															
S0.CH21-22.SF2																															
S0.CH23-24.SF1																															
S0.CH23-24.SF2																															
S0.CH25-26.SF1																															
S0.CH25-26.SF2																															
S0.CH27-28.SF1																															
S0.CH27-28.SF2																															
S0.CH29-30.SF1																															
S0.CH29-30.SF2																															
S0.CH31-32.SF1																															
S0.CH31-32.SF2																															

8.4.6.4 IEC 61937 Compressed Audio

When transmitting IEC 61937 compressed audio via ASPs, the configuration of the Audio Control DIP shall adhere to the following rules:

- The Audio Encoding Type field shall be set to 0001b.
- When transmitting non–high bitrate audio (IEC61937 encoding bitrate less than or equal to 6.144 Mbps), the value of Number of Audio Channels shall be set to 00000b (2 channels). The sink device can parse the actual number of channels of compressed audio and other information from the IEC61937 data flow.
- When transmitting non–high bitrate audio (IEC61937 encoding bitrate less than or equal to 6.144 Mbps), one ASP contains 7 compressed audio data frames.
- When transmitting high-bitrate audio (IEC61937 encoding bitrate greater than 6.144 Mbps), the value of Number of Audio Channels shall be set to 0x3 (8 channels). The sink device can parse the actual number of channels of compressed audio and other information from the IEC61937 data flow.
- When transmitting high-bitrate audio (IEC61937 encoding bitrate greater than 6.144 Mbps), one ASP contains 4 compressed audio data frames.

The following is an example of IEC 61937 compressed audio arrangement, where Frame represents one audio frame and SF1 and SF2 represent two sub-frames of the current audio frame. The arrangement of non–high bit rate audio is shown in Figure 143.

Figure 143 Example of non–high bitrate audio arrangement

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	ShuttleID							E = 1	S = 1	R = 0	CP = 0	Type = 4				Length = 60									D = 1	ECC					
HB0								HB1								HB2								HB3							
Frame0.SF1																															
Frame0.SF2																															
Frame1.SF1																															
Frame1.SF2																															
Frame2.SF1																															
Frame2.SF2																															
Frame3.SF1																															
Frame3.SF2																															
Frame4.SF1																															
Frame4.SF2																															
Frame5.SF1																															
Frame5.SF2																															
Frame6.SF1																															
Frame6.SF2																															

The arrangement of high-bitrate audio is shown in Figure 144.

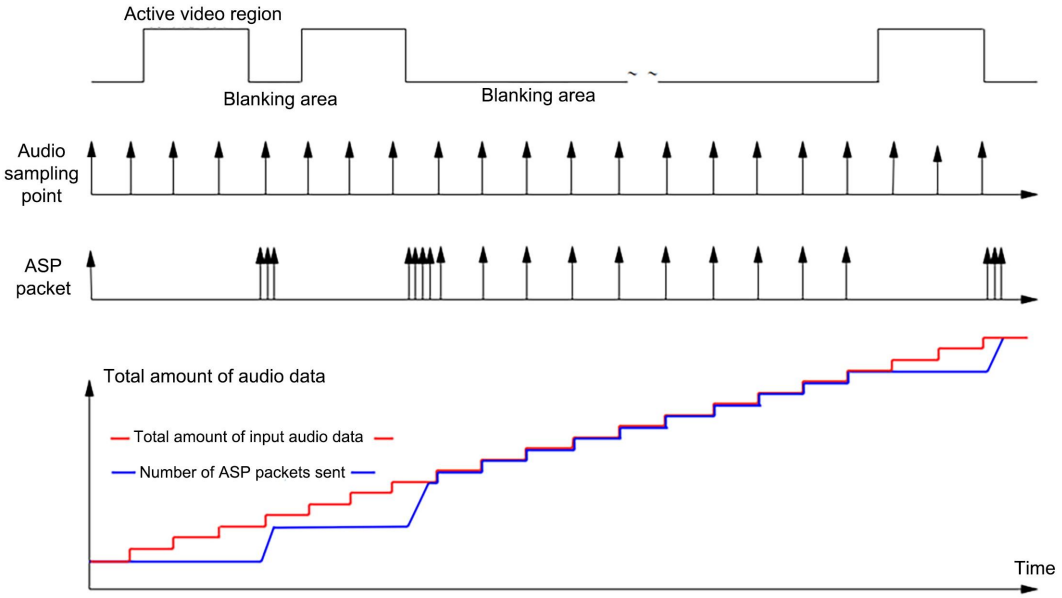
Figure 144 Example of high-bitrate audio arrangement

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	ShuttleID							E = 1	S = 1	R = 0	CP = 0	Type = 4				Length = 36									D = 1	ECC					
HB0								HB1								HB2								HB3							
Frame0.SF1																															
Frame0.SF2																															
Frame1.SF1																															
Frame1.SF2																															
Frame2.SF1																															
Frame2.SF2																															
Frame3.SF1																															
Frame3.SF2																															

8.4.6.5 Audio Data Buffering

The audio accumulation of the source device cannot exceed 2 lines. Figure 145 gives an example of AVP and ASP transmission timing. The audio data is evenly and continuously input to the audio and video transmitting adapter of the source device. In this example, each video line contains 4 to 5 ASPs awaiting transmission. To prevent loss of audio data awaiting transmission, the source device shall allocate a sufficiently large buffer area. Similarly, the sink device shall also provide a sufficiently large buffer area to prevent loss of received audio data.

Figure 145 Example of AVP and ASP transmission timing (when link bandwidth is close to the bandwidth required for video)



8.4.7 EDP

Whether in unicast or multicast scenarios, when content protection is enabled, the audio and video adapter transmits EDPs to explain how the transmitted content is encrypted and decrypted. The structure of EDP is shown in Figure 146, and EDP header fields are defined in Table 106. For requirements, definitions, and descriptions of payload fields, see *Advanced Digital Content Protection System Technical Specification*.

Figure 146 EDP structure

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD								ShuttleID								E	S	R = 0		CP = 0		Type = 14				Length				D = 1	ECC		
EDP0																EDP1										EDP2				EDP3			
…																…										…				…			
CRC32																															

EDP header fields are defined in Table 106.

Table 106 EDP header field definitions

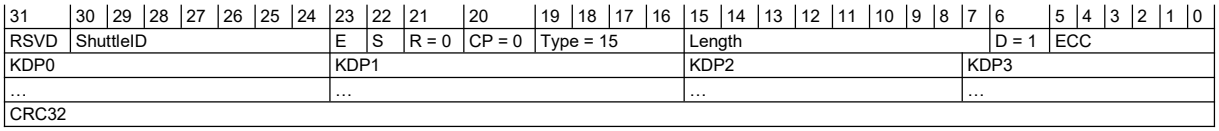
Field Name	Length (Bits)	Description
ECC	6	ECC
D	1	Fixed at 1
Length	9	Fixed at 28
Type	4	Fixed at 1110b
Flags	4	CP: fixed at 0

Field Name	Length (Bits)	Description
		R: fixed at 0 S: fixed at 1 E: fixed at 1
ShuttleID	7	Lane identifier for the corresponding service flow, ranging from 0 to 127.
RSVD	1	Reserved

8.4.8 KDP

In the multicast scenario, the audio and video transmitting adapter transmits the content key to the audio and video receiving adapter of each recipient through KDPs. The structure of KDP is shown in Figure 147, and KDP header fields are defined in Table 107. For requirements, definitions, and descriptions of payload fields, see *Advanced Digital Content Protection System Technical Specification*.

Figure 147 KDP structure



KDP header fields are defined in Table 107.

Table 107 KDP header field definitions

Field Name	Length (Bits)	Description
ECC	6	ECC
D	1	Fixed at 1
Length	9	Fixed at 48
Type	4	Fixed at 1111b
Flags	4	CP: fixed at 0 R: fixed at 0 S: fixed at 1 E: fixed at 1
ShuttleID	7	Lane identifier for the corresponding service flow, ranging from 0 to 127.
RSIVD	1	Reserved

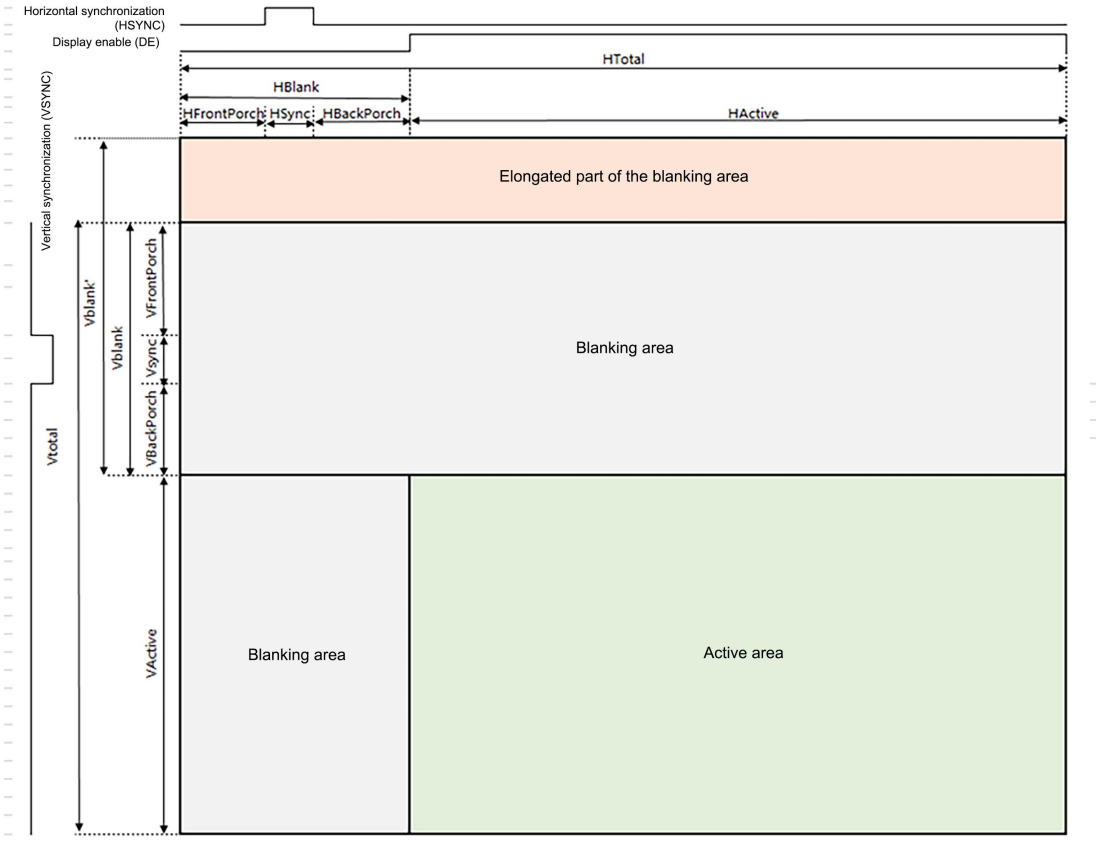
8.5 Advanced Audio and Video Features

8.5.1 Quick Video Transmission (QVT)

When the main link bandwidth is sufficient, the General Purpose Multimedia Interface introduces a QVT mode to reduce transmission latency. In QVT mode, the transmission time for active video data of each video frame is reduced, allowing the sink device to receive the content of each video frame more quickly. Since the video frame rate remains unchanged, the sink device will have more time to process each video frame. After completing QVT, the source device can choose to turn off certain modules (such as compression modules) to reduce power consumption.

In the QVT scenario, it is recommended that no packets be transmitted during the extended part of the blanking area so that the main link can enter low-power mode. The extended part of the blanking area is defined as shown in Figure 148. In the original format, the number of lines in the vertical blanking area is Vblank. After enabling QVT mode, the number of lines in the vertical blanking area becomes Vblank'. Vblank' – Vblank represents the extended part of the blanking area.

Figure 148 Example of the extended part of the blanking area



Before starting the QVT mode, the source device shall determine whether the audio and video receiving adapter supports QVT through the DCCD of the sink device, and obtain the maximum video bandwidth ($BMAX_{sink\_video}$) and the maximum pixel clock (($f_{sink\_pixel\_max}$)) supported by the sink device. At the same time, the source device obtains the maximum available bandwidth ($BMAX_{channel\_avail}$) supported by the channel by querying the bandwidth.

If the sink device supports QVT, calculate the maximum available video bandwidth:$BMAX_{video}$

$$BMAX_{video} = \min(BMAX_{sink_{video}}, BMAX_{channel_{avail}} \times \alpha)$$

Where α is the link bandwidth utilization rate, and $BMAX_{channel\_avail} \times \alpha$ is the maximum valid bandwidth supported by the channel.

Use $BMAX_{video}$ to calculate the pixel clock corresponding to the maximum available video bandwidth $f_{channel\_pixel\_max}$:

$$f_{channel\_pixel\_ax} = \frac{BMAX_{video}}{bpp}$$

Where bpp is the equivalent number of bits per pixel. For example, for YCbCr444 at 10-bit, bpp = 30; for YCbCr422 at 10-bit, bpp = 20; for YCbCr420 at 10-bit, bpp = 15.

Calculate the maximum available pixel clock in QVT mode f_{pixel_max}:

$$f_{pixel\_max} = \min(f_{channel\_pixel\_max}, f_{sink\_pixel\_max})$$

Use the maximum available pixel clock in QVT mode f_{pixel_max} to calculate the maximum vertical total lines supported by the sink device in the original format $Vtotal\_max$:

$$Vtotal_{max} = Floor(\frac{f_{pixel\_max}}{Htotal * Refresh\_Rate})$$

Where $Floor$ is the round-down function, $Htotal$ is the horizontal parameter of the original format, and $Refresh\_Rate$ is the refresh rate of the original format.

The source device selects the vertical total lines for QVT mode $Vtotal\_qvt$ from the vertical total lines in the original format $Vtotal$ to the maximum vertical total lines supported in the original format $Vtotal\_max$.

Where,

$$Vtotal < Vtotal\_qvt \leq Vtotal\_max$$

Then, the pixel clock in QVT mode f_{pixel_qvt} can be calculated:

$$f_{pixel\_qvt} = Htotal \times Vtotal_{qvt} \times Refresh\_Rate$$

The number of front porch lines in the vertical blanking area and the number of lines in the vertical blanking area in QVT mode are:

$$Vfront_{qvt} = Vtotal_{qvt} - Vtotal + Vfront$$

$$Vblank_{qvt} = Vtotal_{qvt} - Vactive$$

Where $Vfront$ is the number of front porch lines in the vertical blanking area of the original format, and $Vactive$ is the number of active lines per frame.

The magnification of QVT is:

$$N = \frac{f_{pixel\_qvt}}{f_{pixel}}$$

Where f_{pixel} is the pixel clock of the original format.

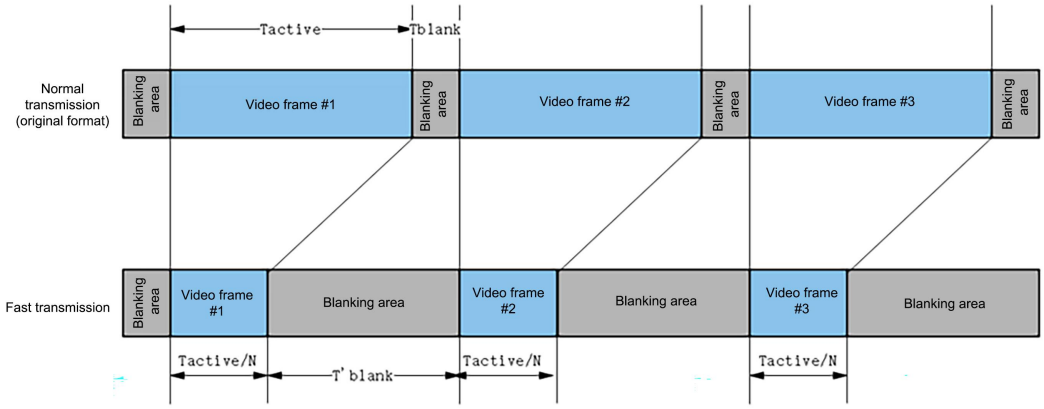
Therefore, in QVT mode, the bandwidth occupied by each video frame should be N times that of normal video transmission mode. As shown in Figure 149, THactive and THblank represent the active video transmission time per line and the horizontal blanking area transmission time in

normal video transmission mode, respectively. After enabling QVT mode, THactive and THblank are shortened to THactive' and THblank', satisfying:

$$THactive' = \frac{THactive}{N} \qquad THblank' = \frac{THblank}{N}$$

The active video transmission time per line is shortened, but the frame rate remains unchanged. Note: After QVT mode is enabled (QVT flag set to 1), the timing parameters in the VBP VFC information shall be those after QVT mode is enabled, namely the pixel clock f_{pixel_qvt}, number of front porch lines in the vertical blanking area $Vfront\_qvt$, and number of lines in the vertical blanking area $Vblank\_qvt$. Other parameters remain unchanged.

Figure 149 Example of video timing in different transmission modes



8.5.2 Auto Low Latency Mode (ALLM)

Different scenarios have different requirements for the display latency of the sink device. For instance, in gaming scenarios, the display latency of the sink device shall be as low as possible. To meet the needs of different scenarios, the General Purpose Multimedia Interface supports ALLM, which allows the source device to automatically enable or disable low-latency mode on the sink device.

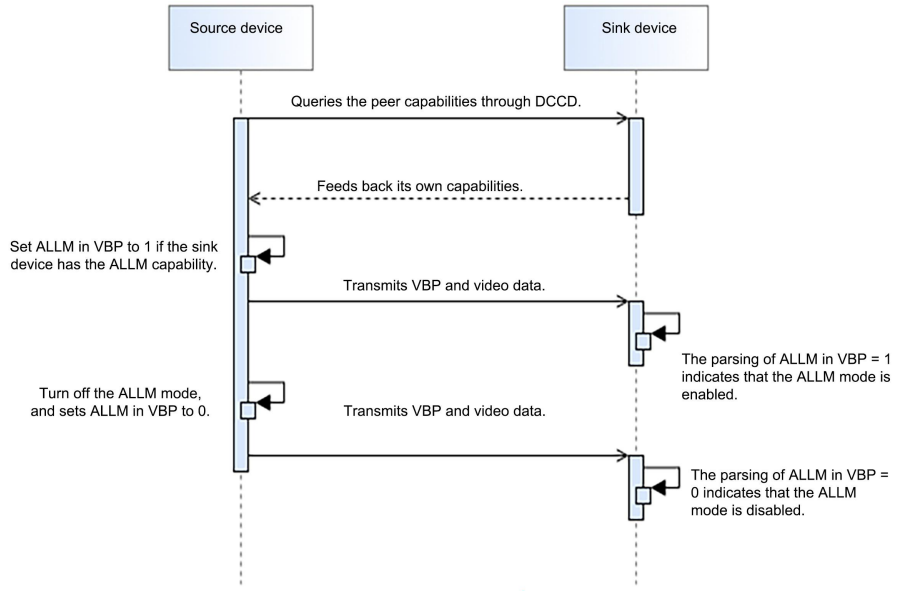
Before enabling ALLM, the source device shall query the capabilities of the sink device to determine whether it supports ALLM. After confirming that the sink device supports ALLM, the source device shall use the ALLM flag in the VBP to instruct the sink device to enter or exit low-latency mode. If the ALLM flag is set to 1, the sink device enters low-latency mode. If the ALLM flag is set to 0, the sink device exits low-latency mode.

Table 108 Example of enabling/disabling ALLM

Video Content Type	ALLM = 0	ALLM = 1
0x0: Unknown	Disable	Enable
0x1: Graphics	Disable	Enable
0x2: Photo	Disable	Enable
0x3: Cinema	Disable	Enable
0x4: Game	Disable	Enable

When entering or exiting low-latency mode, it is recommended that the sink device avoid visual artifacts (such as black or blurred screen) and audio anomalies (such as noise or popping sounds). Audio and video should remain synchronized. The process of enabling/disabling ALLM is shown in Figure 150. The source device shall first confirm the capabilities of the peer device, and the sink device shall determine whether to enable or disable low-latency mode based on the ALLM flag in the VBP.

Figure 150 Process of enabling/disabling ALLM

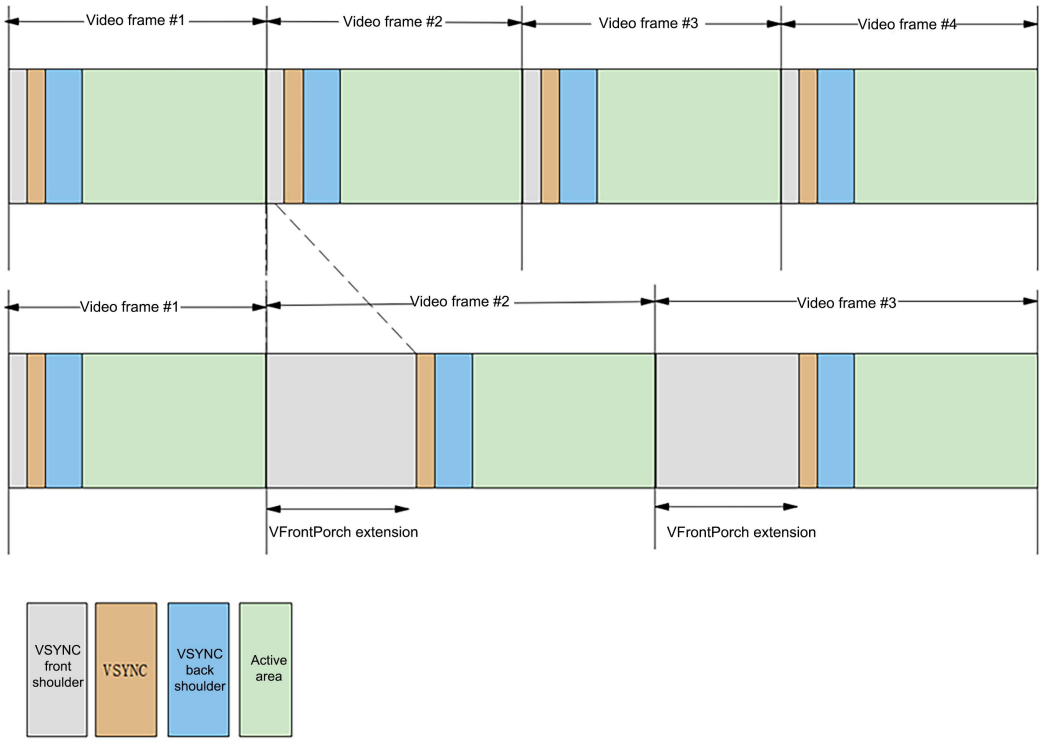


8.5.3 Dynamic Frame-Rate Refresh (DFR)

During video transmission, the frame rate may be unstable. For example, in gaming scenarios, the rendering time of different frames by the game engine may vary, leading to an unstable video frame rate. To ensure the sink device receives such data evenly and stably, the General Purpose Multimedia Interface introduces the DFR mode. This allows the sink device to dynamically adjust its display refresh rate in coordination with the source device, reducing visual artifacts such as screen tearing and meeting the low-latency requirements of gaming scenarios.

The source device dynamically adjusts frame rates by changing the time interval between consecutive video frames: The greater the time interval between adjacent video frames, the lower the video frame rate; conversely, the shorter the interval, the higher the frame rate. The source device adjusts the time interval by inserting or deleting additional video lines at the VFrontPorch, and the total insertion time is N times (N is a positive integer) the duration of one video line. As shown in Figure 151 , the VBackPorch and Vsync of video frame #2 remain unchanged, and the VFrontPorch is extended to delay the sending of video frame #2 to achieve the purpose of reducing the frame rate. Therefore, the dynamic frame rate refresh mode only adjusts the VFrontPorch time, and other video timing remains unchanged. When dynamic frame rate refresh is enabled, the minimum refresh rate that can be supported is 1 Hz.

Figure 151 Dynamic frame rate refresh mechanism

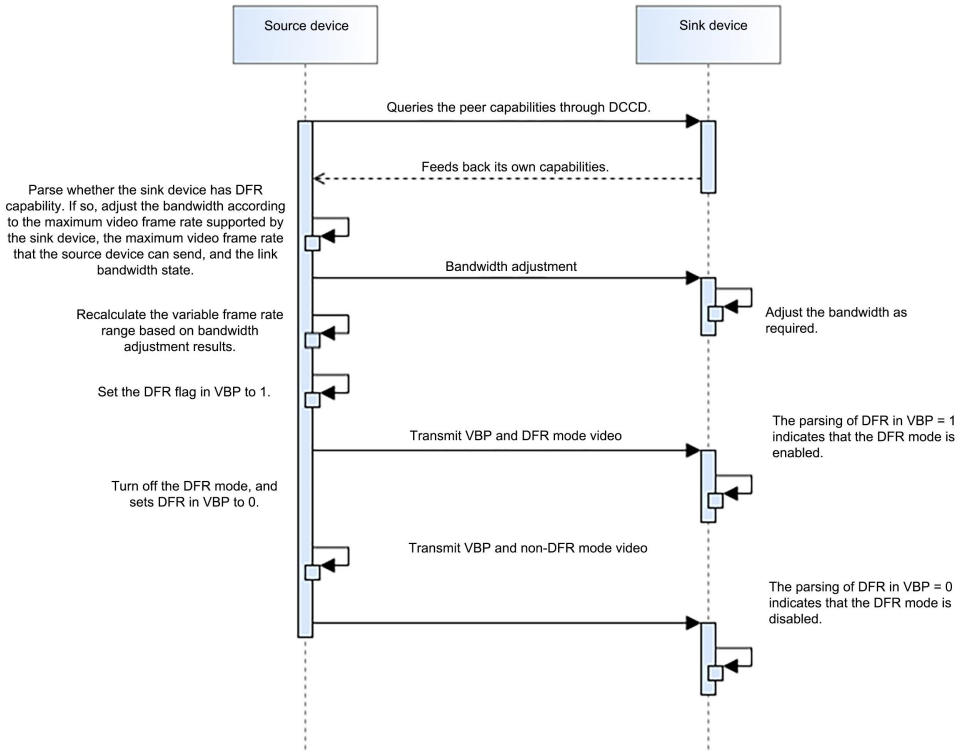


Before the dynamic frame rate refresh mode is enabled, the source device shall obtain the description of comprehensive capabilities of the sink device through capability query, read the DCCD, determine whether the sink device supports the dynamic frame rate refresh mode, and determine the minimum frame rate and maximum frame rate supported by the sink device through the DCCD, thereby establishing the range of N. The source device can dynamically adjust the video frame rate only when the sink device supports the dynamic frame rate refresh mode. The source device notifies the sink device through VBP packets, and shall ensure that the adjusted frame rate is between the minimum frame rate and the maximum frame rate supported by the sink device.

When the sink device has DFR capability, the source device needs to adjust the bandwidth according to the maximum video frame rate supported by the sink device, the maximum video frame rate that the source device itself can send, and the maximum video frame rate allowed by the current link bandwidth. The bandwidth is adjusted according to the minimum value of the above three. The range of video frame rates is adjusted according to the bandwidth adjustment result. If the requested bandwidth is exactly the minimum value of the above three when DFR is enabled, no bandwidth adjustment is required.

When DFR is enabled, the VBP parameters shall not change, that is to say, the format information of the maximum frame rate shall be placed in the VBP during DFR, the link bandwidth shall be allocated according to the maximum frame rate, and the sink device shall ignore the VFrontPorch parameters. The process of enabling and disabling dynamic frame rate refresh mode is shown in Figure 152.

Figure 152 Process of enabling and disabling DFR mode



8.5.4 HDR Video

8.5.4.1 HDR Static Metadata Mapping in CTA-861-H

The static metadata information of Dynamic Range and Mastering InfoFrame in CTA-861-H is carried by the video metadata DIP, and their mapping relationship is shown in Table 109.

Table 109 Video metadata DIP encapsulation Dynamic Range and Mastering InfoFrame static metadata example

Byte	Name	Bit	Description
HB0	Descriptive Information Type	7:0	Fixed to 04h
HB1	INDEX	3:0	Sequence number of the Video Metadata DIP, used to identify the current packet sequence and facilitate packet loss detection by the receiver.
	Reserved	7:4	Reserved.
HB2	Organization	7:4	0010b: CTA
	Metadata Type	3:0	0000b: static metadata
HB3	Length	7:0	Actual valid data length in this packet. When the E flag is 1, this field indicates the number of valid bytes carried in the current DIP.

Byte	Name	Bit	Description
DB0	Payload 0	7:0	InfoFrame Version number
DB1	Payload 1	7:0	Length of Info Frame
DB2	Payload 2	7:0	Data Byte 1
DB3	Payload 3	7:0	Data Byte 2
DB4	Payload 4	7:0	Data Byte 3(Static_Metadata)
DB27	Payload 27	7:0	Data Byte 26(Static_Metadata)
DB28	Payload 28	7:0	Reserved
…	…	7:0	…
DB31	Reserved	7:0	Reserved
CRC32	CRC check	7:0	CRC32 value of HB0–DB31

8.5.4.2 HDR Dynamic Metadata Mapping in CTA-861-H

The dynamic metadata information of Extended InfoFrame in CTA-861-H is carried by the video metadata DIP, and their mapping relationship is shown in Table 110.

Table 110 Video metadata DIP encapsulation Extended InfoFrame dynamic metadata example

Byte	Name	Bit	Description
HB0	Descriptive Information Type	7:0	Fixed at 04h
HB1	INDEX	3:0	Sequence number of the Video Metadata DIP, used to identify the current packet sequence and facilitate packet loss detection by the receiver.
	Reserved	7:4	Reserved
HB2	Organization	7:4	0010b: CTA
	Metadata Type	3:0	0001b: dynamic metadata.
HB3	Length	7:0	Actual valid data length in this packet. When the E flag is 1, this field indicates the number of valid bytes carried in the current DIP.
DB0	Payload 0	7:0	Extended InfoFrame Type LSB
DB1	Payload 1	7:0	Extended InfoFrame Type MSB
DB2	Payload 2	7:0	Length of following Extended InfoFrame Data LSB
DB3	Payload 3	7:0	Length of following Extended InfoFrame Data

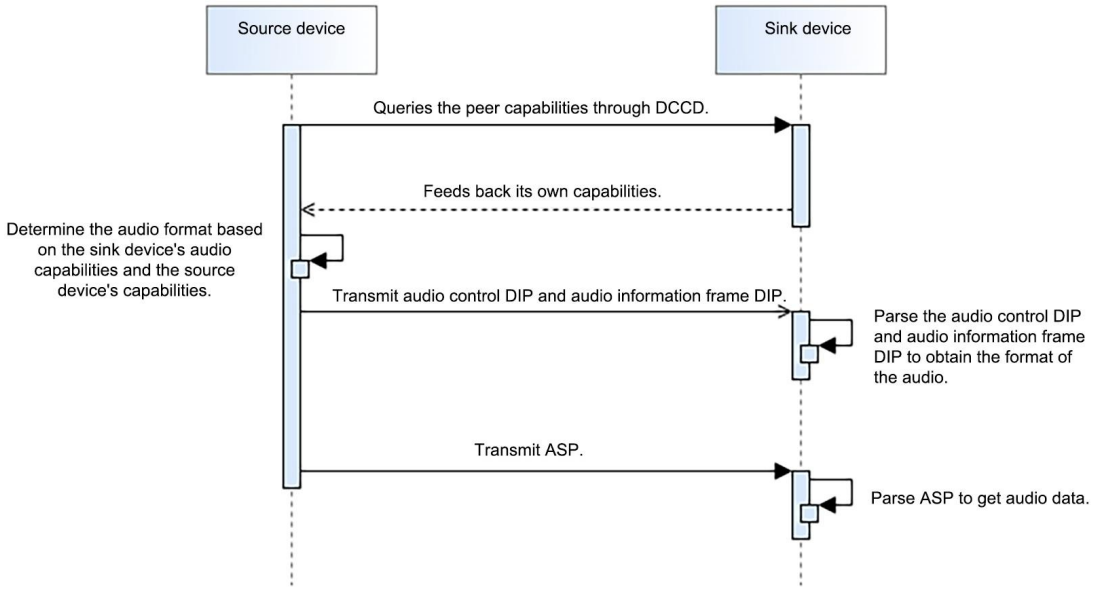
Byte	Name	Bit	Description
			MSB
DB4	Payload 4	7:0	Data Byte 1
DB5	Payload 5	7:0	Data Byte 2
…	…	7:0	…
DB30	Payload 30	7:0	Data Byte 27
DB31	Payload 31	7:0	…
CRC32	CRC check	31:0	CRC32 value of HB0–DB31

Dynamic metadata may exceed 32 bytes, in which case multiple video metadata DIP packets are required to carry it.

8.5.5 Multi-channel Audio Transmission

The GPMI supports multi-channel audio transmission. Before audio transmission, the source device shall obtain the sink device capabilities through DCCD. If the sink device supports audio in the IEC-61937 format and the source device itself also has the capability of the IEC-61937 format audio, the source device can transmit multi-channel audio in the IEC-61937 format; if the sink device does not support audio in the IEC-61937 format, the source device obtains the speaker information of the sink device through DCCD. According to the capabilities of the sink device, the source device sends multi-channel LPCM audio that can be supported by the sink device, indicates the corresponding number of channels in the audio control DIP packets, and specifies the corresponding speaker arrangement in the audio information frame DIP. The transmission process of multi-channel audio is shown in Figure 153.

Figure 153 Transmission process of multi-channel audio



8.6 PLLC Video Transmission

8.6.1 PLLC Video Arrangement

The GPMI supports video transmission in the PLLC compression format. For details about PLLC, see *Perceptual LossLess Compression*. PLLC compressed video is carried by AVP and is consistent with the arrangement of RGB 8bpc. The arrangement example is shown in Figure 154.

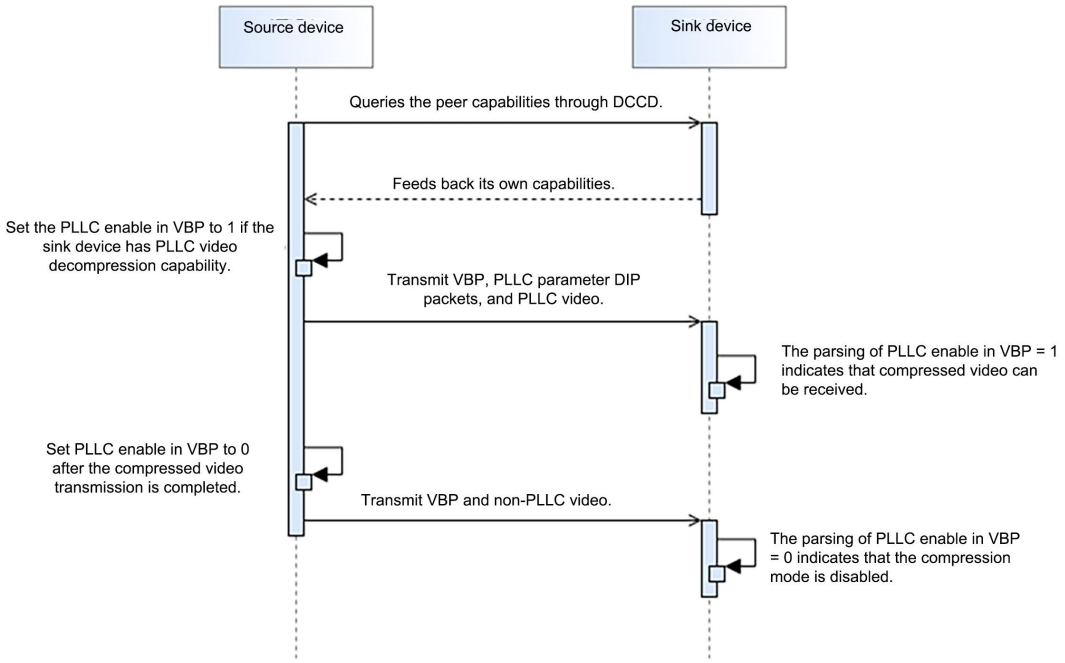
Figure 154 Example of PLLC video arrangement

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4byte	RSVD	ShuttleID							E	S	R	CP	Type=5				Length										D=1	ECC				
8byte	Byte0[7:0]								Byte1[7:0]								Byte2[7:0]								Byte3[7:0]							
12byte	Byte4[7:0]								Byte5[7:0]								Byte6[7:0]								Byte7[7:0]							
16byte	Byte8[7:0]								Byte9[7:0]								Byte10[7:0]								Byte11[7:0]							
⋮																																
512byte	…																															

8.6.2 Process of Enabling and Disabling PLLC

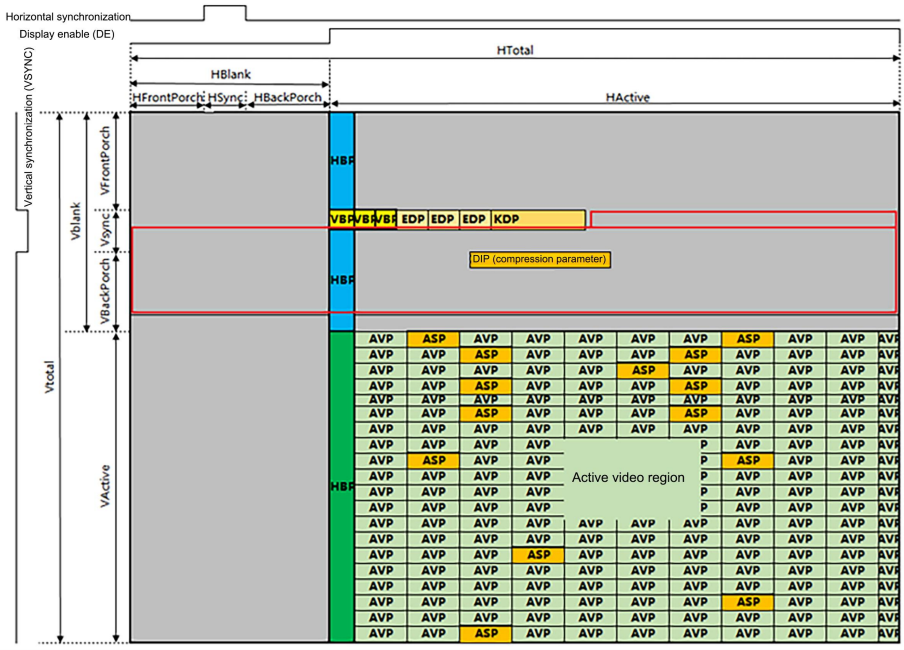
The process of enabling and disabling PLLC is shown in Figure 155. The source device shall first confirm the capabilities of the link partner device, and the sink device determines whether to enable/disable PLLC according to the PLLC enable bit in a VBP packet.

Figure 155 Process of enabling and disabling PLLC



When PLLC is enabled, the PLLC enable flag bit in VBP is set to 1. The PLLC parameter DIP shall be sent after the video frame VBP and before the last blanking line to ensure that the sink device receives and parses the PLLC parameter DIP in time. The sending window of the PLLC parameter DIP is shown in the red box of Figure 156.

Figure 156 PLLC Parameter DIP Sending Window

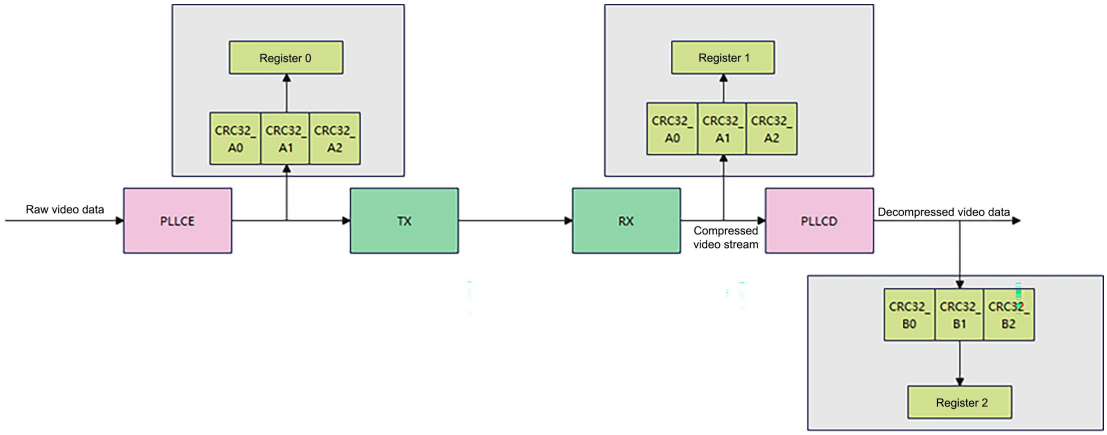


8.6.3 CRC Calculation of PLLC Data

PLLC uses CRC32 to calculate and verify compressed data to ensure the correctness of reconstructed images. The GPMI device with PLLC compression function shall have a frame-level CRC check module. The CRC calculation and verification process for compressed data is shown in Figure 157.

In the source device, a frame-level CRC32_A check shall be performed on the PLLC video stream and the CRC check bits shall be stored in local CRC register 0.

Figure 157 Example of CRC calculation and verification process for compressed data



CRC32_A consists of three CRC32 check modules. The data calculation and verification formats are shown in Table 111, where n represents the number of bytes in a frame of PLLC video.

Table 111 Example of data calculation and verification formats in CRC32_A

CRC32_A0				**CRC32_A1**				**CRC32_A2**			
[31:24]	**[23:16]**	**[15:8]**	**[7:0]**	**[31:24]**	**[23:16]**	**[15:8]**	**[7:0]**	**[31:24]**	**[23:16]**	**[15:8]**	**[7:0]**
Byte9	Byte6	Byte3	Byte0	Byte10	Byte7	Byte4	Byte1	Byte11	Byte8	Byte5	Byte2
Byte21	Byte18	Byte15	Byte12	Byte22	Byte19	Byte16	Byte13	Byte23	Byte20	Byte17	Byte14
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
Byte n-3	Byte n-6	Byte n-9	Byte n-12	Byte n-2	Byte n-5	Byte n-8	Byte n-11	Byte n-1	Byte n-4	Byte n-7	Byte n-10

The last byte of a PLLC video frame may be placed in CRC32_A0, CRC32_A1, and CRC32_A2. When it is less than 4 bytes, 0 needs to be padded to the high bit to achieve 4-byte alignment. The rules are as follows:

- When the last byte of a PLLC video frame is placed in CRC32_A0 Byte n-12, the high bit of CRC32_A0 is padded with 0, and CRC32_A1 and CRC32_A2 are not padded;
- When the last byte of a PLLC video frame is placed in CRC32_A1 Byte n-11, the high bits of CRC32_A0 and CRC32_A1 are padded with 0, and CRC32_A2 is not padded;
- When the last byte of a PLLC video frame is placed in CRC32_A1 Byte n-10, the high bits of CRC32_A0, CRC32_A1, and CRC32_A2 are all padded with 0.

Similarly, in other cases, the high bits of CRC32_A0, CRC32_A1, and CRC32_A2 are all padded with 0.

CRC32_B also consists of three CRC32 check modules, which are mainly used in CTS testing to verify whether the image reconstruction is correct during transmission. The data verification formats of CRC32_B are shown in Table 112, where m represents the number of pixels in the decompressed data.

Table 112 Example of data verification formats in CRC32_B

CRC32_B0		**CRC32_B1**		**CRC32_B2**	
[31:16]	**[15:0]**	**[31:16]**	**[15:0]**	**[31:16]**	**[15:0]**
R_Cr1'	R_Cr0'	G_Y1'	G_Y0'	B_Cb1'	B_Cb0'
R_Cr3'	R_Cr2'	G_Y3'	G_Y2'	B_Cb3'	B_Cb2'
. . .	. . .	. . .	. . .	. . .	. . .
R_Cr m-1'	R_Cr m-2'	G_Y m-1'	G_Y m-2'	B_Cb m-1'	B_Cb m-2'

R_Cr' represents the R component or Cr component in a pixel, G_Y' represents the G component or Y component in a pixel, and B_Cb' represents the B component or Cb component in a pixel.

If the video component bit width is less than 16 bpp, each component in CRC32_B needs to be padded with 0. When 0 is added for padding, the valid data is placed in the high bits, and the low bits are padded with 0. When the original video is encoded with YCbCr422 or YCbCr420 pixels, the missing pixel components caused by sampling shall be padded with 0.

8.7 Content Protection

8.7.1 Content Protection Process

For details about the ADCP authentication process, see *Technical Specification of Advanced Digital Content Protection System.*

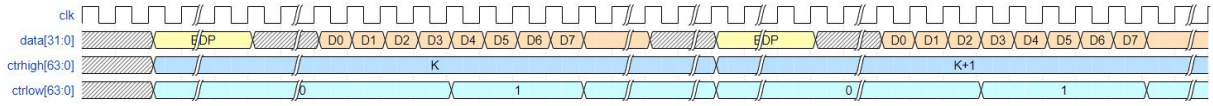
8.7.2 ADCP Data Encryption

The GPMI uses ADCP to encrypt AVP and ASP. When AVP and ASP are encrypted, the following regulations must be followed:

- When the ADCP encryption function is enabled, the CP field in the headers of AVP and ASP shall be set to 1b.
- ADCP does not encrypt the headers of AVP and ASP. It only encrypts their payloads.
- In the last AVP of a video line, "0" zeros may need to be inserted for 4-byte alignment. The data padded with "0" is treated as valid video data for ADCP encryption and decryption.

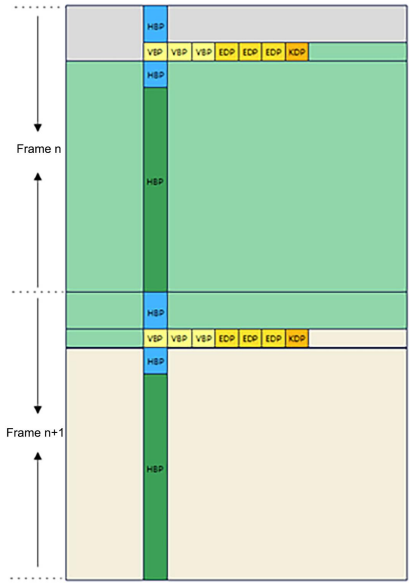
Figure 158 shows an example of the encryption timing of AVP and ASP payloads. In Figure 158, CtrHigh is the upper 64 bits of the counter Ctr used for encryption. Its uniqueness shall be ensured, and the initial CtrHigh can be generated based on a random number. When a new EDP is generated, a new CtrHigh is generated by adding the previous CtrHigh by 1. When the maximum value is reached, add 1 and wrap around to 0. After that, the counter continues to accumulate. The lower 64 bits of the counter CtrLow are set to 0 when CtrHigh in each frame of EDP is updated. After each 16-byte audio and video stream, the counter increases by 1. When the maximum value is reached, add 1 and wrap around to 0. After that, the counter continues to accumulate. In Figure 158, D0, D1... are 32-bit audio and video data that need to be encrypted, and K is the initial CtrHigh generated based on a random number.

Figure 158 Example of an encryption timing of AVP and ASP payloads.



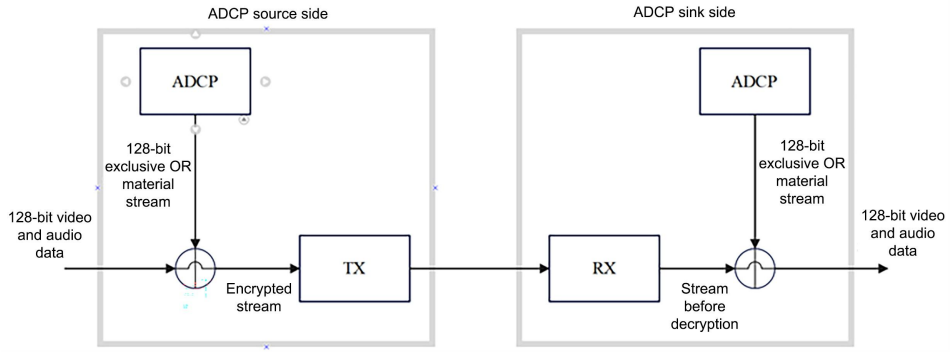
ADCP encryption and decryption for each frame start with EDP. The green area in Figure 159 is an example of the operational range of the nth frame EDP.

Figure 159 Example of EDP operational range



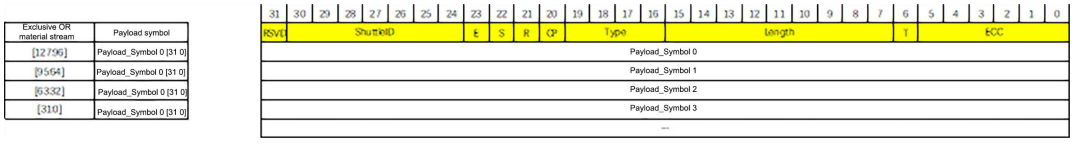
The data that requires encryption between two consecutive EDP frames is not necessarily an integer multiple of 16 bytes. Therefore, the last 16-byte exclusive OR material stream generated by ADCP may have a remainder, which needs to be discarded. When the next EDP starts, the newly generated exclusive OR material stream is used to encrypt the new data. The encryption and decryption process is shown in Figure 160.

Figure 160 Block diagram of an encryption and decryption process



The corresponding relationship between exclusive OR material streams and symbols of audio and video data is shown in Figure 161.

Figure 161 Corresponding relationship between exclusive OR material streams and symbols of audio and video data



9 Management Adapter

9.1 Basic Requirements

The management adapter has the capability to discover, manage, and configure the GPMI network. Among its main functional items, the device management module is used for the establishment and maintenance of the device topological structure; the port management module is used for the maintenance of the port state machine; the bandwidth management module is used for the state management of channels and for the negotiation and maintenance of information such as bandwidth; the device control module is used for device state maintenance.

The AdaptorID of a management adapter is fixed to 0, and the ShuttleID of a management adapter packet is fixed to 0. When the main link does not operate, management adapter packets are transmitted through the sideband link, and some of them are fixedly transmitted through the sideband link.

After receiving a management adapter packet with ShuttleID 0 from the port, the transport layer directly forwards the packet to the management adapter for processing and does not forward it to other ports. After receiving a packet from the management adapter, the transport layer forwards the packet to the port specified by the management adapter.

The management adapter is responsible for parsing the management adapter packet and forwarding it to the corresponding port according to its addressing type, or transmitting it to the corresponding module for processing according to its type.

The management adapter encapsulates messages of different types into corresponding management adapter packets and sends them to the transport layer. All Rsvd and Padding fields in the management adapter packets shall be 0.

9.2 Management Adapter Packet

9.2.1.1 Packet Format

The structure of a management adapter packet is shown in Figure 162, and includes the packet header, general fields, addressing information, message data, and CRC32 check fields.

Figure 162 Structure of a management adapter packet

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
Packet header			
General field			
Addressing information (optional)			
Message data (optional)			
CRC32			

The packet header contains the information that needs to be processed by the transport layer and the management adapter, and the header length is fixed to 4 bytes. The general field only contains the information required by the management adapter, and the field length is fixed to 4 bytes. The addressing information is optional, which is an address table or a forwarding list. For details, refer to Section 9.2.2. For details about message data, refer to Section 9.2.4. CRC32 is used to protect the general field, addressing information, and message data in the current management adapter packet.

The general field, addressing information, message data, and CRC32 are all payloads, and the total payload length does not exceed 508 bytes. When the length of a management message

exceeds 508 bytes, the message needs to be split into multiple management adapter packets for transmission. For the splitting rules, refer to Section 9.2.1.5.

9.2.1.2 Packet Header

The structure of a management adapter packet header is shown in Figure 163, and the description of each field is shown in Table 113.

Figure 163 Structure of a management adapter packet header

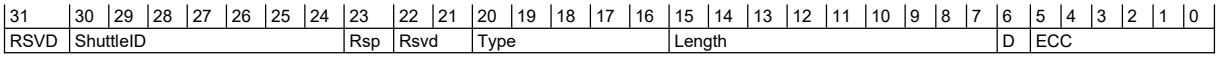


Table 113 Description of fields in a management adapter packet header

Field Name	Length (Bits)	Description
ECC	6	Error correction code
D	1	Fixed to 1
Length	9	Payload length
Type	5	The types of management messages transmitted by management adapter packets are detailed in Table 115.
Rsvd	2	Reserved.
Rsp	1	Response message flag bit. 0b indicates a request message, and 1b indicates a response message.
ShuttleID	7	Fixed to 0
RSVD	1	Reserved.

9.2.1.3 General Field

The general field structure of a management adapter packet is shown in Figure 164, and the description of each field is shown in Table 114.

Figure 164 General field structure of a management adapter packet

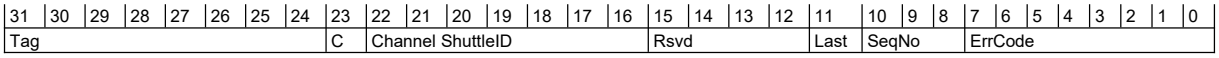


Table 114 Description of general fields in a management adapter packet

Field Name	Length (Bits)	Description
ErrCode	8	For ErrCodes, refer to Section 9.2.3.3.
SeqNo	3	The sequence number of the current management adapter packet ranges from 0 to 7. For details, refer to Section 9.2.1.5.

Field Name	Length (Bits)	Description
Last	1	For the flag bit of the last packet, refer to Section 9.2.1.5. 0b indicates the first or intermediate packet, and 1b indicates the last packet.
Rsvd	4	Reserved.
Channel ShuttleID	7	The field is used only by bandwidth application, bandwidth adjustment, bandwidth release, and channel removal messages. The channel removal message also uses the field as addressing information.
C (Channel)	1	Bandwidth query message usage. 0b: query the minimum value of the available bandwidth for the entire lane uplink and downlink; 1b: query the available bandwidth for uplink and downlink of egress on each level
Tag	8	Message identifier, ranging from 0 to 255 Each management adapter must maintain a message identifier internally, and the identifier is initialized to 0 after power-on. Every time the management adapter sends a management message from any port, the adapter must fill in this field with the internally maintained message identifier, and then increment the internally maintained message identifier cyclically. This field shall remain the same for all management adapter packets of the same management message.

9.2.1.4 Addressing Information

The addressing information may be an address table or a forwarding list, and management adapter packets of different types select one of them for use. However, there are two cases where the addressing information is not carried:

- The management adapter packets of some types use ShuttleID for addressing, and their addressing information uses the Channel ShuttleID in the general fields, including channel removal messages.
- The management adapter packets of some types are only transmitted between neighbors (between directly connected devices) and do not need to carry the addressing information, including topology query messages, topology change messages, adapter change messages, and port management messages.

9.2.1.5 Splitting Rules

Due to the limited length of a management adapter packet, a management message can be split into multiple management adapter packets for transmission. The sender follows the following rules.

– If the message is not split: Last is 1 and SeqNo is 0.

– If the message is split:

- If the current packet is the first or intermediate packet: Last is 0, SeqNo increments from 0 to 7 in order, and Length is 508;
- If the current packet is the last one: Last is 1, and SeqNo is the previous SeqNo plus 1;
- All packets contain a packet header, general fields, addressing information and CRC32, and all fields maintain the same value except Last, SeqNo, and CRC32. Only the message data portion is split;
- All packets of the same management message are sent using the same link (the main link or the sideband link).

The receiver's management adapter shall be able to determine the splitting situation based on Tag, Last, and SeqNo. When splitting occurs, the adapter can combine all packets into a complete management message. For management adapter packets with the same Tag, if their SeqNo becomes 0, the source device shall be considered to have resent the message, rather than an exception of "SeqNo discontinuity".

9.2.2 Addressing

There are three addressing methods for management adapter packets, namely address table, forwarding list, and channel. In addition, some management adapter packets are only transmitted between neighbors and do not require addressing. Such packets do not use any addressing method. If a management adapter packet requires addressing, it uses a fixed addressing method according to its type. For addressing methods used by management adapter packets of various types, see Table 115.

Table 115 Addressing methods for management adapter packets of various types

Type Value	Packet Type	Addressing Method
0	Topology query	No addressing required
1	Topology change	No addressing required
2	Adapter change	No addressing required
3	Read DCCD	Address table
4	Bandwidth query	Forwarding list
5	Bandwidth application	Forwarding list
6	Bandwidth adjustment	Forwarding list
7	Bandwidth release	Forwarding list
8	Channel removal	Channel
9	Reserved	/
10	Reserved	/
11	Reserved	/
12	Reserved	/
…	Reserved	/

Type Value	Packet Type	Addressing Method
27	Device control	Address table
28	Content protection	Address table
29	HID PassThrough	Address table
30	Reserved	/
31	Port management	No addressing required

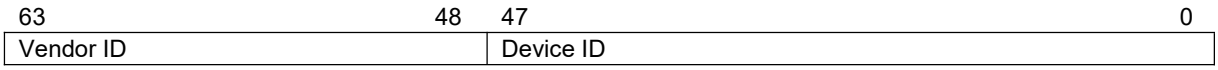
9.2.2.1 Address Table Addressing

9.2.2.1.1 Device Address

Each GPMI device has a unique 8-byte identifier that can uniquely identify a GPMI device, so the identifier is called a device address.

The device address consists of a Vendor ID and a Device ID. Vendor ID consists of 16 bits and is obtained through registration with the organization. Device ID is assigned by the device vendor when manufacturing GPMI devices and must be unique.

Figure 165 Device address format



9.2.2.1.2 Device Address Table

The device address table is used to record the corresponding relationship between a device address and a port, and forward a packet to the corresponding port according to the port number corresponding to the target device address. The generation rules of a device address table refer to Section 9.3.2.5. The device address table is shown in Table 116.

Table 116 Example of a device address **table**

Device Address	Port
0x0102030405060708	1
0x0102030405060709	2
…	…

9.2.2.1.3 Addressing Information

For a packet using an address table for addressing, the structure of the addressing information is shown in Figure 166, and the description of each field is shown in Table 117.

Figure 166 Structure of an addressing information format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Destination Address Low																															
Destination Address High																															
Source Address Low																															
Source Address High																															

Table 117 Description of fields in addressing information

Field Name	Length (Bits)	Description
Destination Address Low	32	Bit [31:0] of the target address
Destination Address High	32	Bit [63:32] of the target address
Source Address Low	32	Bit [31:0] of the source address
Source Address High	32	Bit [63:32] of the source address

The target address is the device address of the receiving device expected by the management adapter packet. The source address is the address of the device that sends the management adapter packet. When a device receives the management adapter packet:

—If the target address of the packet is the device's own address, the device is the target device of the packet.

—Otherwise, query the address table according to the target address of the packet and process the packet according to the query result:

- If there is a target address of the packet in the address table, and the corresponding port is different from the port receiving the packet, the packet is forwarded from the corresponding port.
- If there is a target address of the packet in the address table, but the corresponding port is the same as the port receiving the packet, the target address of the packet is considered to be in the device connected to this port, so the target device has received the packet and the device shall discard the packet.
- If the target address of the packet cannot be found in the address table, the device does not know whether a port is connected to the target device. The device shall flood this packet from all ports except the receiving port.

9.2.2.2 Forwarding List Addressing

9.2.2.2.1 Forwarding List

The forwarding list is a one-dimensional list containing multiple elements, and each element specifies the physical port number from which the packet is sent on each device. For example, the first element is the port number used by the device directly connected to this device to forward this packet, and so on. Each element of the forwarding list occupies 4 bits, ranging from 0 to 15, and 0 represents unused. The initial value is determined by the starting device. The length of the forwarding list is determined by the starting device and ranges from 0 to 126. When the forwarding list length is N, the distance to a device to be accessed is N + 1. Therefore, a GPMI device can access devices at a maximum distance of 127 from itself through the forwarding list.

In the topological structure shown in Figure 167, when a packet is sent from device A to device B, each level of the link has an updated forwarding list. The forwarding list carried by the packet when it is sent from device A is shown in Table 118.

Figure 167 Example of forwarding list topology

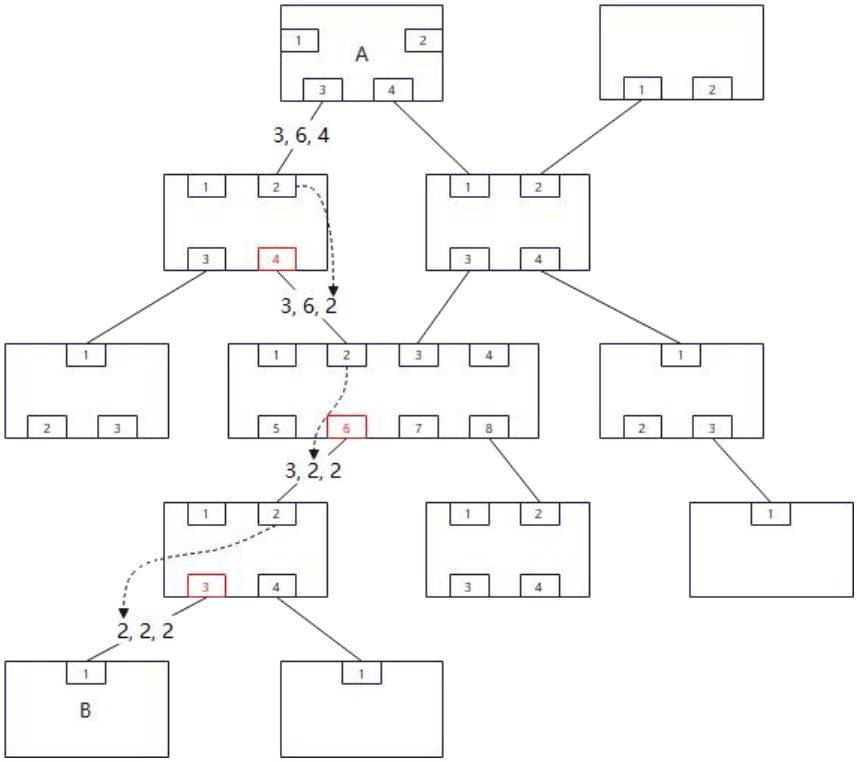


Table 118 Description of fields in addressing information

Level 3 PortID	Level 2 PortID	Level 1 PortID
3	6	4

9.2.2.2.2 Addressing Information

For a packet using a forwarding list for addressing, the structure of the addressing information is shown in Figure 168, and the description of each field is shown in Table 119.

Figure 168 Addressing information format of forwarding list addressing

31 30 29 28 27 26 25 24	23 22 21 20	19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
…	Level 2 PortID	Level 1 PortID	Rsvd	Forwarding List Length	Rsvd	Forwarding List Level
…						
padding					Level x PortID	…

Table 119 Description of fields in addressing information

Field Name	Length (Bits)	Description
Forwarding List Level	7	Current level, ranging from 1 to 127

Field Name	Length (Bits)	Description
Rsvd	1	Reserved
Forwarding List Length	7	Forwarding list length, ranging from 0 to 126
Rsvd	1	Reserved
Level 1 PortID	4	Level 1 port number, ranging from 1 to 15
Level 2 PortID	4	Level 2 port number, ranging from 1 to 15
…	…	…
Level x PortID	4	Level x port number, ranging from 1 to 15
padding	/	Filled with 0 to ensure 4-byte alignment, and the filling length ranging from 0 bit to 28 bits

FL Level indicates the level of the current device with the source device as the origin. When a device on the path receives a packet, if FL Level is 0 or FL Length+1, this device is the target device of this packet. Otherwise, this device is the intermediate device of this packet. When each intermediate device on the path forwards this packet, it shall use the FL Level element of the forwarding list as the sending port number of the packet. The target device does not need to forward the packet.

FL Length is the length of the forwarding list.

When each device on the path receives a packet, no matter whether it is the target device of the packet or not, it shall determine whether to process the message data according to the requirements of the message type.

After processing the message data, if the device is the intermediate device, it forwards the packet. Before forwarding, the addressing information needs to be updated according to the following process:

- Record the FL Level element of the forwarding list as the sending port number of a packet;
- Rewrite the FL Level element of the forwarding list as the receiving port number of the packet;
- If Rsp in the packet header is 0, rewrite the FL Level field as FL Level plus 1. Otherwise, rewrite the FL Level field as FL Level minus 1;
- The updated packet is sent from the previously recorded sending port.

If the device is the target device and Rsp in the packet header is 0, rewrite the FL Level field as FL Level minus 1, and the addressing information of the packet can be used as the addressing information of the response packet with Rsp set to 1. Then, the response packet is sent through the port receiving the request packet. After that, this device becomes the source device of the response packet.

9.2.2.3 Channel Addressing

The forwarding rules are basically the same as the routing and forwarding rules of the transport layer, except that:

- For a management adapter packet using a channel for addressing, the addressing information uses the Channel ShuttleID in the general field instead of the ShuttleID in the

packet header.

- Before forwarding, each intermediate device at every level shall replace the value of Channel ShuttleID with the ShuttleID used by this channel on the next-level link.

9.2.3 Transmission Rules

9.2.3.1 Overview

Management messages of each type are divided into request messages and response messages. Each time a request message is received, a corresponding response message shall be generated for reply. The Tag of the response message shall be consistent with that of the request message, except for special descriptions (such as content protection). For management messages transmitted by the sender that are split into multiple management adapter packets, the receiver shall reply with the response message after receiving the complete message, that is, the packet with Last set to 1.

The management adapter packets support transmission on the main link or the sideband link, where the port capability negotiation message is only transmitted through the sideband link. All management adapter packets of the same management message must be transmitted through the same link. The management adapter needs to notify the transport layer with each packet: the fixed main link, fixed sideband link, and adaptive selection. If a message is split into multiple packets for transmission, adaptive selection cannot be used.

The management adapter can send a new request message before receiving the response message to the previous request message. A management adapter can only send one request message without a response to the same target device, and can send at most two request messages without a response to different target devices on the same port. The management adapter shall ensure that if any device on the path needs to rely on the processing result of the previous request message when processing the next request message, that device must receive the corresponding response message for the previous request message before sending the next request message. For example, bandwidth-related operations are performed on the same flow.

When the management adapter receives the management adapter packets, if there is a CRC error, SeqNo discontinuity, or an addressing information parsing error, the management adapter directly discards the entire message and replies with a response message corresponding to the ErrCode. For specific ErrCodes, refer to 9.2.3.3. In addition, for management adapter packets with the same Tag, if their SeqNo becomes 0, the source device shall be considered to have resent the message, rather than an exception of "SeqNo discontinuity".

If the source device of the management message does not receive a response message within the timeout period, the retransmission can be performed, and the Tag of the retransmitted message remains unchanged. The timeout period is determined based on the distance between the source device and the target device, and each level of link is calculated as a minimum of 50 ms. The specific timeout period used by the management message of each type can be greater than 50 ms/link. If there are special provisions, they shall be clearly stated in the corresponding sections.

If the source device of the management message receives a response message with an ErrCode of non-zero, the source device can retransmit the message immediately without waiting for the timeout period, and the Tag of the retransmitted message remains unchanged.

After a timeout or receiving a response message with an ErrCode of non-zero, whether to perform the retransmission and the number of retransmissions shall be clearly stated in the corresponding sections.

9.2.3.2 General Processing

Message processing shall be based on the principle of first-come, first-served, that is, messages received earlier are processed and forwarded, and then those received later.

9.2.3.2.1 Message Sending

The device generates bandwidth management and device control messages according to service needs, and the management adapter is responsible for sending the corresponding messages.

9.2.3.2.2 Message Receiving

(a) Initiating Device

When a message is received, it is generally a response message. First, determine whether the ErrCode is non-0x0. If it is not zero, process the message according to the section. If it is 0x0, process the message according to the specific requirements of the message sections. Refer to Section 9.2.4 and Section 9.5.8 for more information.

(b) Intermediate Device

When a request message is received, the device's status and resource conditions need to be checked to confirm whether the message can be received and forwarded. If not, refer to Section 9.2.3.3.2 to generate the corresponding ErrCode and directly return a response message. Secondly, the packet header, general fields, and message content of the message are checked for validity. If an exception is found, refer to the description in Section 9.2.3.3.2 to generate a corresponding error and return a response message directly. When the intermediate device processes and forwards the message normally, refer to Section 9.5 and Section 9.6 for more information.

When receiving a response message, if the intermediate device needs to process the response message, such as the bandwidth release response message, the packet header, general fields, and message content of the message need to be checked for validity. Otherwise, forward the message directly without the validity check. When the message is checked for validity and an exception is found, refer to the description in Section 9.2.3.3.2 to generate a corresponding error, and continue forwarding after refreshing ErrCode.

(c) Target Device

When a request message is received, the device's status and resource conditions need to be checked to confirm whether the message can be received and forwarded. If not, refer to Section 9.2.3.3.2 to generate the corresponding ErrCode and directly return a response message. Secondly, the packet header, general fields, and message content are checked for validity. If an exception is found, refer to the description in Section 9.2.3.3.2 to generate a corresponding error and return a response message directly. When the target device processes and forwards the message normally, refer to Section 9.5 and 9.6 for more information.

9.2.3.2.3 Response Message

After receiving a request message, a corresponding response message is generated according to the request message. Response messages are subject to the following general processing rules:

(1) The input port number of the request message is used as the output port number of the response message.

(2) For addressing messages, the source address field of the response message is filled with the target address in the request message, and the target address field of the response message is filled with the source address in the request message.

(3) For messages involving CHShuttleID, the CHshuttleID field in the response message is filled with the ShuttleID input in the request message.

9.2.3.3 Error Handling

9.2.3.3.1 ErrCode Introduction

Common ErrCodes in message processing are shown in Table 120.

Table 120 Description of ErrCodes

Value	Name
0x0	Success
0x1	Busy
0x2	Device State Error
0x3	Function Unsupported
0x4	Path Invalid
0x5	Reject Requesture
0x6	Resources Insufficient
0x7–0x20	Reserved
0x21	CRC32 Error
0x22	Packets Length Mismatch
0x23	Unknown Type
0x24	Unknown ShuttleID
0x25	Packets Inconsecutive Sequence Numbers
0x26	Unknown CHShuttleID
0x27	Forwarding List Error
0x28	Unknown Source Address
0x30–0x50	Reserved
0x51	Payload Length Error
0x52	Unknown Source AdapterID
0x53	Unknown Destination AdapterID
0x54	AdapterID Mismatch with ShuttleID
0x55	Priority Mismatch with Adapter Type
0x56	Bandwidth Error
0x57	Unknown Device Control Command
0x58–0xFF	Reserved

9.2.3.3.2 ErrCode Generation

Errors usually occur at any device on the topology path or virtual channel, except for the device initiating messages. Common scenarios for ErrCode generation are as follows.

The device receives a request message:

(1) When the device is processing some services and cannot process the current request message, set ErrCode to 0x1 (Busy). For example, (a) When receiving a request message to read DCCD, the device is not ready for the DCCD data; (b) The device is in the Standby state and receives a device wake-up request message.

(2) When the device is in the Standby or other low power states and cannot complete the action required by the request message, ErrCode is set to 0x2 (Device State Error). For example, when the device is in the standby state, a bandwidth application request message is received.

(3) When the device receives a non-essential device control message and the current device does not support this function, ErrCode is set to 0x3 (Function Unsupport).

(4) When the device receives bandwidth negotiation and device control messages, it needs to make a comprehensive decision based on its own device status and service status. If the device decides to reject the request, ErrCode is set to 0x5 (Reject Requesture).

(5) When the management adapter checks the validity of packets and messages, if errors such as parameter mismatch and unknown parameters are detected in the packet header, general fields, or packet content, ErrCode needs to be set to the corresponding values in [0x21: 0x28] or [0x51: 0x57].

The device receives a response message:

(1) If ErrCode is not zero, the device forwards the message directly without any action.

(2) If ErrCode is zero, except for the message that requires processing of a response message, the device only needs to forward the response message directly, and no corresponding errors are generated.

Note: Some messages also need to be processed when a response message is received, such as a bandwidth application response message and a bandwidth release response message. The corresponding ErrCode scenarios and rules generated by the response message are consistent with the description of the device receiving the request message.

9.2.3.3.3 ErrCode Processing

ErrCodes are judged and processed in the device initiating the corresponding request message. Intermediate devices usually do not judge and process ErrCodes. If there are exceptions, they will be explained separately in the specific message process. The ErrCode values and the general recommended processing flows are described as follows:

(1) ErrCode=0x1 (Busy): It is recommended to wait at least 10 ms before starting the retransmission of the corresponding request message. The maximum number of retransmissions is five. If the response messages to five consecutive requests all have the ErrCode of 0x1 (Busy), a device in the current topology path has an exception that cannot be restored. It is recommended to select a new topology path to send the corresponding request message. If there are no more topology paths for selection, report the exception to the application for decision-making by the application or user. For example, the UI prompts the user to check the device state on the path, or to connect and disconnect, or restart the device.

(2) ErrCode=0x2 (Device State Error): It is recommended to select a new topology path to send the corresponding request message. If there are no more topology paths for selection, report

the exception to the application for decision-making by the application or user. For example, the UI prompts the user to check the device state on the path, or to connect and disconnect, or restart the device.

(3) ErrCode=0x3 (Function Unsupport): It is recommended to report the unsupported function directly to the service for decision-making.

(4) ErrCode=0x4 (Path Invalid): It is recommended to check whether the topology has changed, refresh the path information after updating the topology, and then resend the corresponding request message.

(5) ErrCode=0x5 (Reject Requesture): It is recommended to skip the request without retrying.

When ErrCode is [0x21–0x28] or [0x51–0x57], the management adapter checks the validity of the incorrect packet header, general fields, and message data. After the correct data is updated, the request message is retransmitted, and the maximum number of retransmissions is five.

9.2.3.4 Timeout Handling

After the initiating device sends the request message, the timeout period for waiting for the response message is described in Section 9.2.3. When multiple levels are passed, the timeout period needs to be multiplied by the number of levels as the total timeout period of the response message.

If no response message is received within the timeout period, the previous request message needs to be sent repeatedly. The maximum number of retransmissions is five.

9.2.3.5 Timeout Period Parameters

Table 121 Timeout period parameters of a management adapter

Parameter Name	Value	Description
tShuttleTimeout	50 ms	Lane message timeout period (message timeout period between adjacent devices)
tChannelTimeout	tShuttleTimeout * Level	Channel message timeout period (end-to-end timeout period during logical channel communication). Level is the maximum number of networking levels supported by the device.

9.2.4 Packet Type

9.2.4.1 Topology Query

9.2.4.1.1 Request Message

The packet format of the topology query request message is shown in Figure 169, and the description of each field is shown in Table 122.

Figure 169 Packet format of a topology query request message

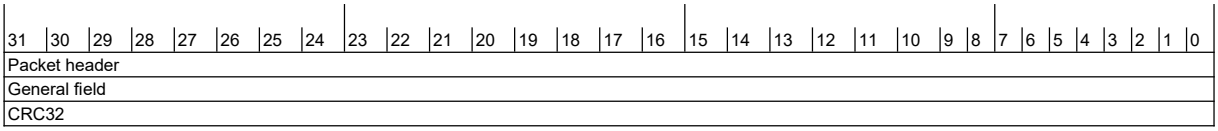


Table 122 Description of fields in a topology query request message packet

Field Name	Length (Bits)	Description
Packet header	32	Refer to Section 9.2.1.1, where Type set to 0 indicates topology query information, and Rsp set to 0 indicates a request message.
General field	32	Refer to Section 9.2.1.1.
CRC32	32	Check code

9.2.4.1.2 Response Message

The packet format of the topology query response message is shown in Figure 170, and the description of each field is shown in Table 123.

Figure 170 Packet format of a topology query response message

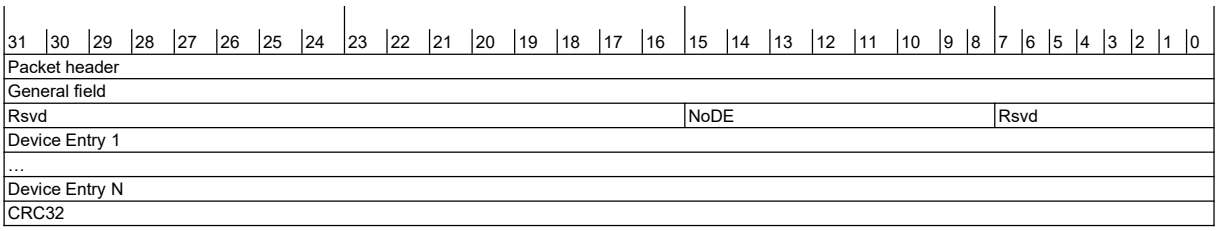


Table 123 Description of fields in a topology query response message packet

Field Name	Length (Bits)	Description
Packet header	32	Refer to Section 9.2.1.1, where Type set to 0 indicates topology query information, and Rsp set to 1 indicates a response message.
General field	32	Refer to Section 9.2.1.1.
Rsvd	8	Reserved.
NoDE	8	The number of device entries in this message, ranging from 1 to 128 The number of device entries in the message data must be consistent with this field.

Field Name	Length (Bits)	Description
		The device entry area of a topology change request message can carry up to 3,932 bytes. Some topology change request messages are generated based on topology query request response messages, so the device entry area of a topology query response message needs to be constrained to a maximum length of 3,932 bytes. If the limit is exceeded, no new device entries shall be added, and all device entries carried by this message must be complete.
Rsvd	16	Reserved.
Device Entry 1–N	…	Arrange NoDE device entries in sequence. It is recommended to take this device as the root node and use the depth-first or breadth-first method to traverse all device nodes in the network topology managed by this device. However, the management adapter receiving this message must support handling cases where the entries are unordered.
CRC32	32	Check code

9.2.4.1.2.1 Device Entry

The device entry format is shown in Figure 171, and the description of each field is shown in Table 124.

Figure 171 Device entry format

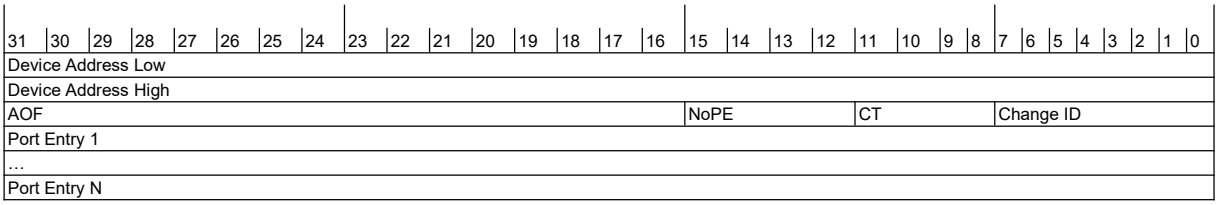


Table 124 Description of fields in a device entry

Field Name	Length (Bits)	Description
Device Address Low	32	This device entry describes bit [31:0] of the device address.
Device Address High	32	This device entry describes bit [63:32] of the device address.
Change ID	8	The source device of this message currently maintains the topology change identifier of the device described by this device entry.
Change Type (CT)	4	Device entry change type, ranging from 0 to 3:

Field Name	Length (Bits)	Description
		0h: delete the device (this device has been disconnected); 1h: change the device information (the status of one or some types of adapters in this device has changed); 2h: change the port entry (the status of one or some ports of this device has changed); 3h: change the entire device entry (this device is inserted or requires a complete overwrite of the original state). Topology query response message, this field is fixed to 3. Topology change request message, this field is filled with 0 to 3 according to the situation.
Number of Port Entry (NoPE)	4	The number of port entries in this message, ranging from 0 to 15 The number of port entries in the message data must be consistent with this field. If this field is greater than 0, the port entries in the message data must carry valid data.
Adapter Online Flag (AOF)	16	The field determines whether there are any online adapters of a certain type in this device. If there is more than one adapter of this type online, the corresponding bit shall be set to 1; otherwise, it is set to 0. An online adapter indicates that the actual function has been connected to the back of this adapter. For example, the audio and video transmission adapter has been connected to the audio and video source output port, or the audio and video sink adapter has been connected to the input port of the display or to the display panel inside the device. Bit [16]: audio and video transmission adapter. Bit [17]: audio and video receiving adapter. Bit [31:18]: reserved. If Change Type is 0 or 2, it is recommended to fill in this field with 0. If Change Type is 1 or 3, this field shall reflect the current adapter status of this device.
Port Entry 1–NoPE	…	Arrange NoPE port entries in sequence. It is recommended that the source device arrange the port entries to be sent in ascending order of port numbers. However, the management adapter receiving this message must support handling cases where the entries are unordered. Either of the two port entries at both ends of a link is reflected. If the two entries are to be reflected, the information carried by the two port entries must not conflict.

Field Name	Length (Bits)	Description
		When a topology query response message is constructed, whether to carry the port entries of ports in the disconnected state is not mandatory in this document. If NoPE is 0, no port entries are included.

9.2.4.1.2.2 Port Entry

The port entry format is shown in Figure 172, and the description of each field is shown in Table 125.

Figure 172 Port entry format

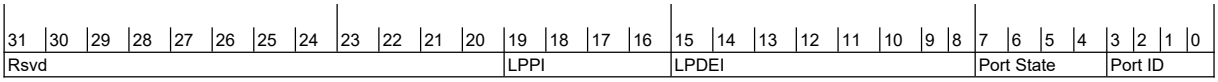


Table 125 Description of fields in a port entry

Field Name	Length (Bits)	Description
Port ID	4	The port identification number of the port described by this port entry, ranging from 1 to 15
Port State	4	Port connection state. 0h indicates the disconnected state, and 1h indicates the connected state. If the port state machine is in the ready or standby state and the link partner device has been merged into the topology, the port is in the connected state. In other cases, the port is in the disconnected state.
LPDEI (Link Partner Device Entry Index)	8	The link partner device is the device described by the LPDEI device entry in this message, ranging from 0 to NoDE. If the Port State field of this port entry indicates that the port is disconnected, the LPDEI field is invalid, and LPDEI is filled with 0.
Link Partner Port ID (LPPI)	4	Port ID of the link partner device, ranging from 0 to 15 If the Port State field indicates that the port is disconnected, the LPPI field is invalid, and LPPI is filled with 0.
Rsvd	12	Reserved.

9.2.4.2 Topology Change

9.2.4.2.1 Request Message

The packet format of the topology change request message is shown in Figure 173, and the description of each field is shown in Table 126.

Figure 173 Packet format of a topology change request message

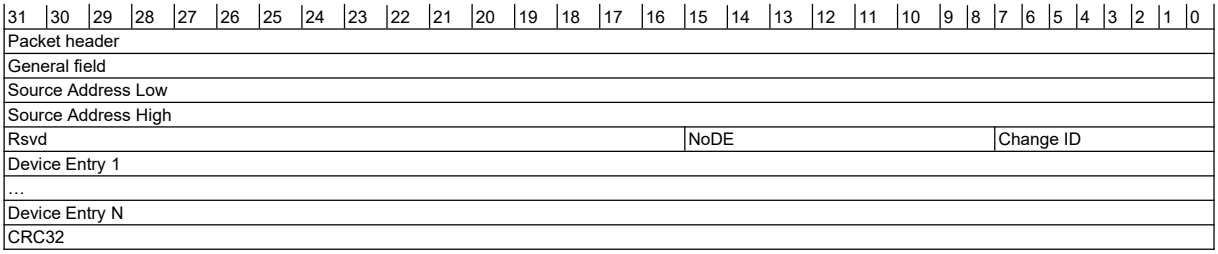


Table 126 Description of fields in a topology change request message packet

Field Name	Length (Bits)	Description
Packet header	32	Refer to Section 9.2.1.1, where Type set to 1 indicates a topology change message, and Rsp set to 0 indicates a request message.
General field	32	Refer to Section 9.2.1.1.
Source Address Low	32	Bit [31:0] of the source address that generates this message.
Source Address High	32	Bit [63:32] of the source address that generates this message.
Change ID	8	Topology change identifier, ranging from 0 to 255. Each management adapter must maintain a change identifier for the topology change request message internally, and the identifier is initialized to 0 after power-on. When the management adapter generates a topology change request message on any port, it must fill in this field with the internally maintained change identifier, and then increment the internally maintained change identifier cyclically.
Number of Device Entry (NoDE)	8	The number of device entries in this message, ranging from 1 to 128. The number of device entries in the message data must be consistent with this field. The device entry area of a topology change request message can carry up to 3,932 bytes. Some topology change request messages are generated based on topology query request response messages, so the device entry area of a topology query response message needs to be constrained to a maximum length of 3,932 bytes. If the

Field Name	Length (Bits)	Description
		limit is exceeded, no new device entries shall be added, and all device entries carried by this message must be complete.
Rsvd	8	Reserved.
Device Entry 1–N	…	Arrange NoDE device entries in sequence. See Table 123 for description.
CRC32	32	Check code

9.2.4.2.2 Response Message

The packet format of the topology change response message is shown in Figure 174, and the description of each field is shown in Table 127.

Figure 174 Packet format of a topology change response message

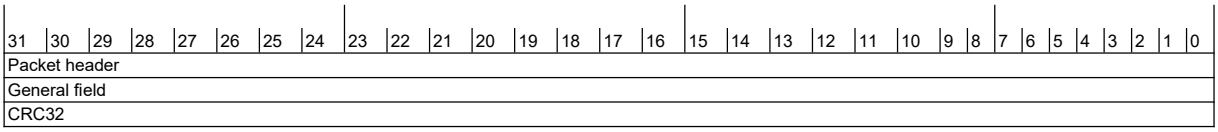


Table 127 Description of fields in a topology change response message packet

Field Name	Length (Bits)	Description
Packet header	32	Refer to Section 9.2.1.1, where Type set to 1 indicates a topology change message, and Rsp set to 1 indicates a response message.
General field	32	Refer to Section 9.2.1.1.
CRC32	32	Check code

9.2.4.3 Adapter Change

9.2.4.3.1 Request Message

The packet format of the adapter change request message is shown in Figure 175, and the description of each field is shown in Table 128.

Figure 175 Packet format of an adapter change request message

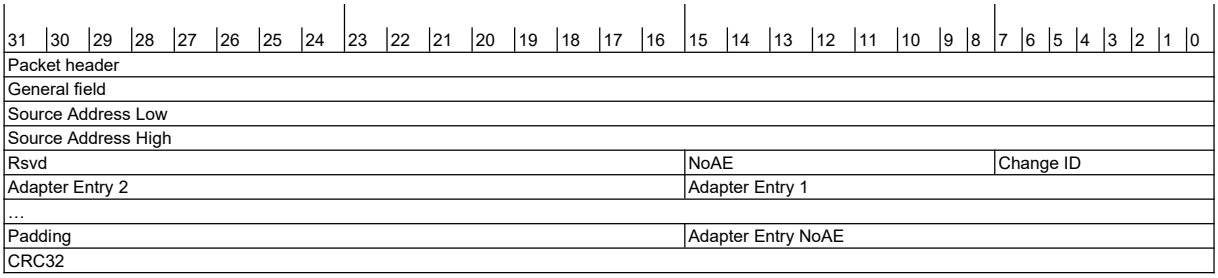


Table 128 Description of fields in an adapter change request message packet

Field Name	Length (Bits)	Description
Packet header	32	Refer to Section 9.2.1.1, where Type set to 2 indicates an adapter change message, and Rsp set to 0 indicates a request message.
General field	32	Refer to Section 9.2.1.1.
Source Address Low	32	Bit [31:0] of the source address that generates this message.
Source Address High	32	Bit [63:32] of the source address that generates this message.
Change ID	8	Topology change identifier, ranging from 0 to 255. Each management adapter must maintain a change identifier for the topology change request message internally, and the identifier is initialized to 0 after power-on. When the management adapter generates a topology change request message on any port, it must fill in this field with the internally maintained change identifier, and then increment the internally maintained change identifier cyclically.
Number of Adapter Entry (NoAE)	8	The number of adapter entries in this message, ranging from 1 to 127. The number of adapter entries in the message data must be consistent with this field.
Rsvd	16	Reserved.
Adapter Entry 1–NoAE	…	Arrange NoAE adapter entries in sequence. It is recommended that the adapters where changes occur be arranged in ascending order of their numbers.
Padding	16	If NoAE is an odd number, fill it with 0 to ensure 4-byte alignment.
CRC32	32	Check code

9.2.4.3.1.1 Adapter Entry

The adapter entry format is shown in Figure 176, and the description of each field is shown in Table 129.

Figure 176 Adapter entry format

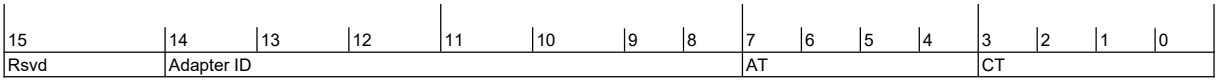


Table 129 Description of fields in an adapter entry

Field Name	Length (Bits)	Description
Change Type (CT)	4	Adapter entry change type: 0h: adapter offline; 1h: adapter online.
Adapter Type (AT)	4	Adapter type: 0h: audio and video source adapter; 1h: audio and video receiving adapter; Others: reserved.
Adapter ID	7	Adapter number, ranging from 1 to 127: 0000000b is the management adapter number, and the management adapter does not generate an adapter change request message.
Rsvd	1	Reserved.

9.2.4.3.2 Response Message

The packet format of the adapter change response message is shown in Figure 177, and the description of each field is shown in Table 130.

Figure 177 Packet format of an adapter change response message

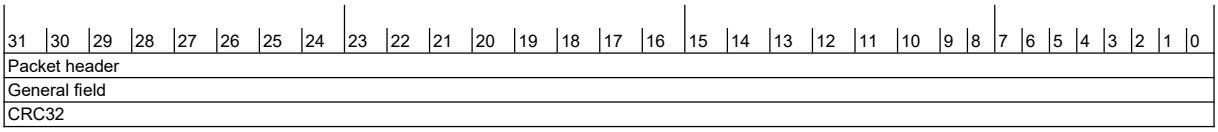


Table 130 Description of fields in an adapter change response message packet

Field Name	Length (Bits)	Description
Packet header	32	Refer to Section 9.2.1.1, where Type set to 2 indicates an adapter change message, and Rsp set to 1 indicates a response message.

Field Name	Length (Bits)	Description
General field	32	Refer to Section 9.2.1.1.
CRC32	32	Check code

9.2.4.4 Read DCCD

9.2.4.4.1 Request Message

The packet format of the DCCD read request message is shown in Figure 178, and the description of each field is shown in Table 131.

Figure 178 Packet format of a DCCD read request message

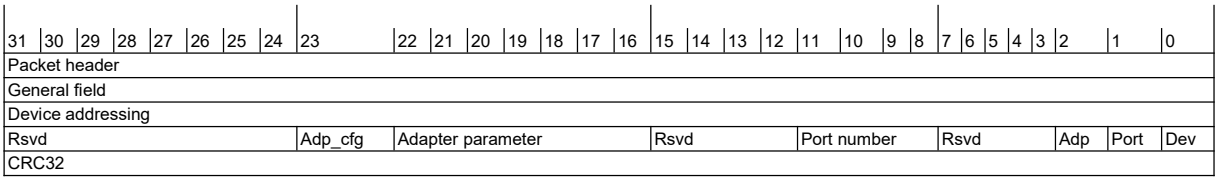


Table 131 Description of fields in a DCCD read request message packet

Field Name	Length (Bits)	Description
Packet header	32	Refer to Section 9.2.1.1, where Type set to 3 indicates a read DCCD message, and Rsp set to 0 indicates a request message.
General field	32	Refer to Section 9.2.1.1.
Device addressing	32	Refer to Section 9.2.2.
Dev (device description flag)	1	Obtain the device description flag: 0b: The description value of device capabilities does not need to be returned; 1b: The description value of device capabilities needs to be returned.
Port (port description flag)	1	Obtain the port description flag: 0b: The description value of port capabilities does not need to be returned; 1b: The description value of port capabilities needs to be returned.
Adp (adapter description flag)	1	Obtain the adapter description flag: 0b: The description value of adapter capabilities does not need to be returned;

Field Name	Length (Bits)	Description
		1b: The description value of adapter capabilities needs to be returned.
Rsvd	5	Reserved.
port number	4	Port number: 0b: indicates querying all port capability description values of the device; 1b–15b: indicates the port capability description value that the query number is the port number.
Rsvd	4	Reserved.
Adapter parameter	7	When Adp_cfg is 0b, the Adapter parameter is the specific adapter number; when Adp_cfg is 1b, the Adapter parameter is the adapter type. 0000000b: all adapters; 0000001b: audio and video transmission adapter; 0000010b: audio and video receiving adapter; Others: reserved.
Adp_cfg (adapter configure type)	1	Adapter configuration type: 0b: query by adapter number; 1b: query by adapter type.
Rsvd	8	Reserved.
CRC32	32	Check code

9.2.4.4.2 Response Message

The packet format of the DCCD read response message is shown in Figure 179, and the description of each field is shown in Table 132.

Figure 179 Packet format of a DCCD read response message

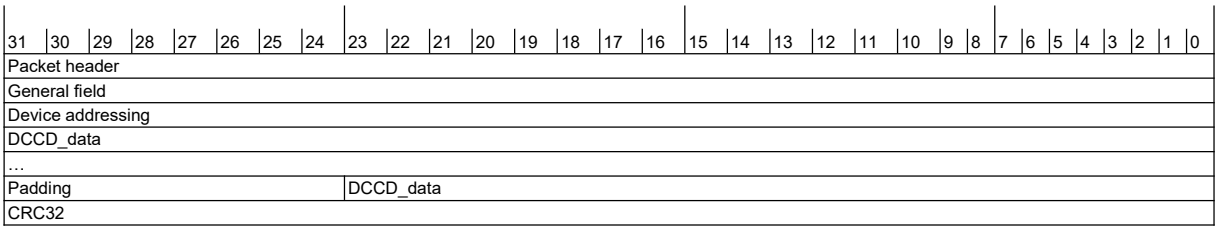


Table 132 Description of fields in a DCCD read response message packet

Field Name	Length (Bits)	Description

Field Name	Length (Bits)	Description
Packet header	32	Refer to Section 9.2.1.1, where Type set to 3 indicates a read DCCD message, and Rsp set to 1 indicates a response message.
General field	32	Refer to Section 9.2.1.1.
Device addressing	32	Refer to Section 9.2.2.
DCCD_data	32	DCCD data
…	32	DCCD data
DCCD_data	…	DCCD data
Padding	…	Filled with 0 to ensure 4-byte alignment
CRC32	32	Check code

9.2.4.5 Port Management

9.2.4.5.1 Capability Negotiation Request Message

The packet format of the capability negotiation request message is shown in Figure 180, and the description of each field is shown in Table 133.

Figure 180 Packet format of a capability negotiation request message

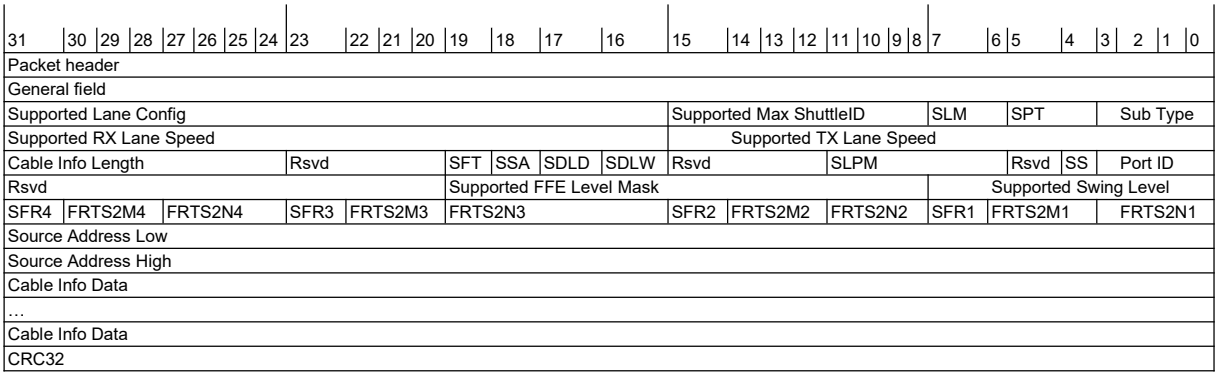


Table 133 Description of fields in a capability negotiation request message packet

DW	Bit	Name	Description
0	31:0	Packet header	Refer to Section 9.2.1.1, where Type set to 31 indicates a port management message, and Rsp set to 0 indicates a request message.
1	31:0	General field	Refer to Section 9.2.1.1.
2	3:0	Sub Type	Subtype: 0h: capability negotiation message

DW	Bit	Name	Description
	5:4	Supported Port Type (SPT)	Supported port types: 01b: LMP, main downlink port; 10b: LSP, main uplink port; 11b: DRP, dual role port; Others: reserved
	7:6	Supported Link Mode (SLM)	Supported link modes: Bit 0: whether to support the simplified-specification mode Bit 1: whether to support the full-specification mode
	15:8	Supported Max ShuttleID	Supported max ShuttleID:
	31:16	Supported Lane Config	Supported lane configurations: 00b: disable; 01b: RX; 10b: TX; 11b: TRX Each 2-bit field corresponds to a lane: B10[1:0] to lane 0, B10[3:2] to lane 1, B10[5:4] to lane 2, B10[7:6] to lane 3, B11[1:0] to lane 4, B11[3:2] to lane 5, B11[5:4] to lane 6, and B11[7:6] to lane 7.
3	15:0	Supported TX Lane Speed	Lane speed supported by TX, supporting any combination: Bit 0: whether to support 2 Gbps; Bit 1: whether to support 4 Gbps; Bit 2: whether to support 6 Gbps; Bit 3: whether to support 8 Gbps; Bit 4: whether to support 10 Gbps; Bit 5: whether to support 12 Gbps; Bit 6: whether to support 16 Gbps; Bit 7: whether to support 20 Gbps; Bit 8: whether to support 24 Gbps; Others: reserved
3	31:16	Supported RX Lane Speed	Lane speed supported by RX, supporting any combination: Bit 0: whether to support 2 Gbps; Bit 1: whether to support 4 Gbps; Bit 2: whether to support 6 Gbps; Bit 3: whether to support 8 Gbps; Bit 4: whether to support 10 Gbps; Bit 5: whether to support 12 Gbps;

DW	Bit	Name	Description
			Bit 6: whether to support 16 Gbps; Bit 7: whether to support 20 Gbps; Bit 8: whether to support 24 Gbps; Others: reserved
4	3:0	Port ID	The port ID that sends this capability negotiation request message.
	4	Standby State (SS)	Device standby state: 0b: the device not in the standby state; 1b: the device in the standby state
	5	Rsvd	Reserved.
	11:6	Supported Low Power Mode (SLPM)	Supported low power modes: Bit 0: whether to support LP0; Bit 1: whether to support LP1; Bit 2: whether to support LP2; Bit 3: whether to support LP3; Bit 4: whether to support LP0f; Bit 5: reserved
	15:12	Rsvd	Reserved.
	16	Supported Dynamic Link Width (SDLW)	Whether to support dynamic link width switching: 0b: link width switching not supported; 1b: link width switching supported
	17	Supported Dynamic Lane Direction (SDLD)	Whether to support dynamic lane direction switching: 0b: lane direction switching not supported; 1b: lane direction switching supported
	18	Rsvd	Reserved.
	19	Supported Fast Training (SFT)	Whether to support fast training: 0b: fast training not supported; 1b: fast training supported
	23:20	Rsvd	Reserved.
	31:24	Cable Info Length	Cable Info data length, in bytes, ranging from 0 to 255 0 indicates no Cable Info data is carried.
5	7:0	Supported Swing Level	Supported Swing levels Each bit indicates whether to support the corresponding Swing level, and bit_N represents whether to support Swing Level N. For example, 00111100b indicates that Swing levels 2, 3, 4, and 5 are supported, but Swing levels 0, 1, 6,

DW	Bit	Name	Description
			and 7 are not supported.
5	19:8	Supported FFE Level Mask	Supported FFE level masks. Bit [3:0]: PRE1 Mask; Bit [7:4]: POST1 Mask; Bit [11:8]: POST2 Mask. There are three FFE parameters, PRE1, POST1, and POST2, and each parameter occupies 4 bits. This field is defined as a mask, indicating the valid bit among the four bits of the FFE parameter. Examples: 0000b: gear 0 supported; 0011b: gears 0, 1, 2, and 3 supported; 0110b: gears 0, 2, 4, and 6 supported; 1100b: gears 0, 4, 8, and 12 supported; 1110b: gears 0, 2, 4, 6, 8, 10, 12, and 14 supported The FFE level supports up to 64 levels, that is, at most 6 bits of the 12 bits in this field are set to 1. The bits set to 1 in the mask of each parameter must be contiguous. For example, 0110b is a valid value, 1010b is an invalid value.
	31:20	Rsvd	Reserved.
6	3:0	Fast Recovery TS2 Num HS1 (FRTS2N1)	The number of rounds of transmitted LLCF_TS2 in fast recovery at HS1 (2 Gbps/4 Gbps) rate Note: Considering a link exception, if the link exception report message ERR_RM is transmitted through the sideband link, approximately 10 us is required for both sides of the port to synchronize the abnormal state. Therefore, based on the values of FRTS2N1 and FRTS2M1, at least 400 LLCF_TS2 shall be transmitted.
	6:4	Fast Recovery TS2 Magnitude HS1 (FRTS2M1)	The number of single rounds of transmitted LLCF_TS2 in fast recovery at HS1 (2 Gbps/4 Gbps) rate: 0h: 10h; 1h: 40h; 2h: 100h; 3h: 400h; 4h: 1000h; 5h: 4000h; 6h: 10000h; 7h: 40000h

DW	Bit	Name	Description
	7	Supported Fast Recovery HS1 (SFR1)	Whether to support the fast recovery feature at HS1 (2 Gbps/4 Gbps) rate: 0b: not support 1b: support
	11:8	Fast Recovery TS2 Num HS2 (FRTS2N2)	The number of rounds of transmitted LLCF_TS2 in fast recovery at HS2 (6 Gbps/8 Gbps) rate Note: Considering a link exception, if the link exception report message ERR_RM is transmitted through the sideband link, approximately 10 us is required for both sides of the port to synchronize the abnormal state. Therefore, based on the values of FRTS2N1 and FRTS2M1, at least 800 LLCF_TS2 shall be transmitted.
6	14:12	Fast Recovery TS2 Magnitude HS2 (FRTS2M2)	The number of single rounds of transmitted LLCF_TS2 in fast recovery at HS2 (6 Gbps/8 Gbps) rate: 0h: 10h; 1h: 40h; 2h: 100h; 3h: 400h; 4h: 1000h; 5h: 4000h; 6h: 10000h; 7h: 40000h
	15	Supported Fast Recovery HS2 (SFR2)	Whether to support the fast recovery feature at HS2 (6 Gbps/8 Gbps) rate: 0b: not support 1b: support
	19:16	Fast Recovery TS2 Num HS3 (FRTS2N3)	The number of rounds of transmitted LLCF_TS2 in fast recovery at HS3 (10 Gbps/12 Gbps/16 Gbps) rate Note: Considering a link exception, if the link exception report message ERR_RM is transmitted through the sideband link, approximately 10 us is required for both sides of the port to synchronize the abnormal state. Therefore, based on the values of FRTS2N1 and FRTS2M1, at least 1200 LLCF_TS2 shall be transmitted.
	22:20	Fast Recovery TS2 Magnitude HS3 (FRTS2M3)	The number of single rounds of transmitted LLCF_TS2 in fast recovery at HS3 (10 Gbps/12 Gbps/16 Gbps) rate: 0h: 10h;

DW	Bit	Name	Description
			1h: 40h; 2h: 100h; 3h: 400h; 4h: 1000h; 5h: 4000h; 6h: 10000h; 7h: 40000h
	23	Supported Fast Recovery HS3 (SFR3)	Whether to support the fast recovery feature at HS3 (10 Gbps/12 Gbps/16 Gbps) rate: 0b: not support 1b: support
	27:24	Fast Recovery TS2 Num HS4 (FRTS2N4)	The number of rounds of transmitted LLCF_TS2 in fast recovery at HS4 (20 Gbps/24 Gbps) rate Note: Considering a link exception, if the link exception report message ERR_RM is transmitted through the sideband link, approximately 10 us is required for both sides of the port to synchronize the abnormal state. Therefore, based on the values of FRTS2N1 and FRTS2M1, at least 2400 LLCF_TS2 shall be transmitted.
6	30:28	Fast Recovery TS2 Magnitude HS4 (FRTS2M4)	The number of single rounds of transmitted LLCF_TS2 in fast recovery at HS4 (20 Gbps/24 Gbps) rate: 0h: 10h; 1h: 40h; 2h: 100h; 3h: 400h; 4h: 1000h; 5h: 4000h; 6h: 10000h; 7h: 40000h
	31	Supported Fast Recovery HS4 (SFR4)	Whether to support the fast recovery feature at HS4 (20 Gbps/24 Gbps) rate: 0b: not support 1b: support
7	31:0	Source Address Low	Bit [31:0] of the source address that generates this message.
8	31:0	Source Address High	Bit [63:32] of the source address that generates this message.
9	31:0	Cable Info Data	Cable Info data This field is carried only when the Cable Info

DW	Bit	Name	Description
			Length field is greater than 0.
…	31:0	Cable Info Data	Same as above
N	31:0	Cable Info Data	Same as above
…	31:0	CRC32	Check code

9.2.4.5.2 Capability Negotiation Response Message

The packet format of the capability negotiation response message is shown in Figure 181, and the description of each field is shown in Table 134.

Figure 181 Packet format of a capability negotiation response message

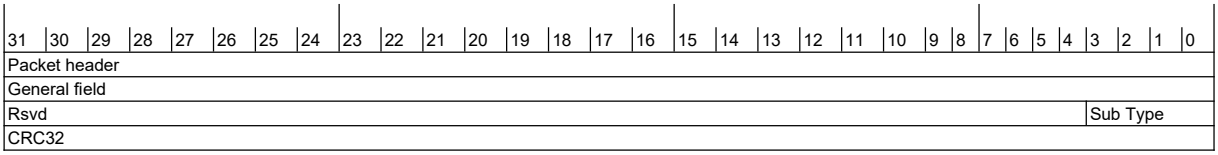


Table 134 Description of fields in a capability negotiation response message packet

Field Name	Length (Bits)	Description
Packet header	32	Refer to Section 9.2.1.1, where Type set to 31 indicates a port management message, and Rsp set to 1 indicates a response message.
General field	32	Refer to Section 9.2.1.1.
Sub Type	4	Subtype: 0h: capability negotiation message
Rsvd	28	Reserved.
CRC32	32	Check code

9.2.4.5.3 Link Configuration Request Message

The packet format of the link configuration request message is shown in Figure 182, and the description of each field is shown in Table 135.

Figure 182 Packet format of a link configuration request message

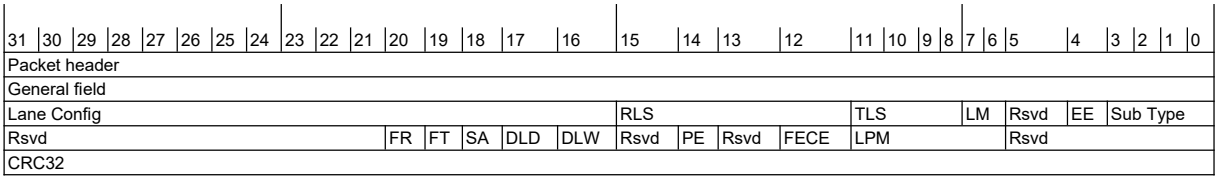


Table 135 Description of fields in a link configuration request message packet

DW	Bit	Name	Description
0	31:0	Packet header	Refer to Section 9.2.1.1, where Type set to 31 indicates a port management message, and Rsp set to 0 indicates a request message.
1	31:0	General field	Refer to Section 9.2.1.1.
2	3:0	Sub Type	Subtype: 1h: link configuration message
2	4	Enter Error (EE)	Enter an error and notify the link partner device whether to enter the error state: 0b: not enter the error state; 1b: enter the error state If this field is 1b, the subsequent fields in DWORD1–2 are invalid.
	5	Rsvd	Reserved.
	7:6	Link Mode (LM)	Link modes: 01b: simplified-specification mode; 10b: full-specification mode
	11:8	TX Lane Speed (TLS)	TX lane speed: 0h: 2 Gbps; 1h: 4 Gbps; 2h: 6 Gbps; 3h: 8 Gbps; 4h: 10 Gbps; 5h: 12 Gbps; 6h: 16 Gbps; 7h: 20 Gbps; 8h: 24 Gbps; Others: reserved
	15:12	RX Lane Speed (RLS)	RX lane speed: 0h: 2 Gbps; 1h: 4 Gbps; 2h: 6 Gbps;

DW	Bit	Name	Description
			3h: 8 Gbps; 4h: 10 Gbps; 5h: 12 Gbps; 6h: 16 Gbps; 7h: 20 Gbps; 8h: 24 Gbps; Others: reserved
	31:16	Lane Config	Lane configurations: 00b: disable; 01b: RX; 10b: TX; Others: reserved Each 2-bit field corresponds to a lane: B10[1:0] to lane 0, B10[3:2] to lane 1, B10[5:4] to lane 2, B10[7:6] to lane 3, B11[1:0] to lane 4, B11[3:2] to lane 5, B11[5:4] to lane 6, and B11[7:6] to lane 7.
3	5:0	Rsvd	Reserved.
	11:6	Low Power Mode (LPM)	Low power modes: Bit 0: whether to support LP0; Bit 1: whether to support LP1; Bit 2: whether to support LP2; Bit 3: whether to support LP3; Bit 4: whether to support LP0f; Bit 5: reserved
	12	FEC Enabled (FECE)	FEC enabled: 0b: FEC not enabled; 1b: FEC enabled
	13	Rsvd	Reserved.
	14	Precoding Enabled (PE)	Precoding enabled: 0b: Precoding not enabled; 1b: Precoding enabled
	15	Rsvd	Reserved.
	16	Dynamic Link Width (DLW)	Dynamic link width switching enabled: 0b: link width switching not enabled; 1b: link width switching enabled
	17	Dynamic Lane Direction (DLD)	Dynamic lane direction switching enabled: 0b: lane direction switching not enabled; 1b: lane direction switching enabled

DW	Bit	Name	Description
	18	Skew Adjust (SA)	Skew adjustment enabled: 0b: Skew adjustment not enabled; 1b: Skew adjustment enabled
	19	Fast Training (FT)	Fast training enabled: 0b: fast training not enabled; 1b: fast training enabled
	20	Fast Recovery (FR)	Fast recovery enabled: 0b: fast recovery not enabled; 1b: fast recovery enabled
	31:21	Rsvd	Reserved.
4	31:0	CRC32	Check code

9.2.4.5.4 Link Configuration Response Message

The packet format of the link configuration response message is shown in Figure 183, and the description of each field is shown in Table 136.

Figure 183 Packet format of a link configuration response message

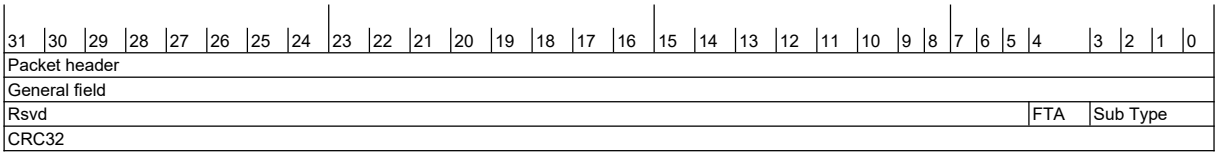


Table 136 Description of fields in a link configuration response message packet

Field Name	Length (Bits)	Description
Packet header	32	Refer to Section 9.2.1.1, where Type set to 31 indicates a port management message, and Rsp set to 1 indicates a response message.
General field	32	Refer to Section 9.2.1.1.
Sub Type	4	Subtype: 1h: link configuration message
FTA	1	Fast training response: 0b: fast training not received; 1b: fast training received
Rsvd	27	Reserved.
CRC32	32	Check code

9.2.4.5.5 Standby Request Message

The packet format of the standby request message is shown in Figure 184, and the description of each field is shown in Table 137.

Figure 184 Packet format of a standby request message

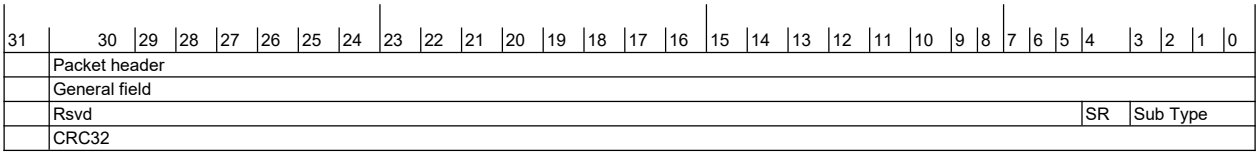


Table 137 Description of fields in a standby request message packet

Field Name	Length (Bits)	Description
Packet header	32	Refer to Section 9.2.1.1, where Type set to 31 indicates a port management message, and Rsp set to 0 indicates a request message.
General field	32	Refer to Section 9.2.1.1.
Sub Type	4	Subtype: 2h: standby message
SR (Standby Request)	1	Standby requests: 0b: exit standby request; 1b: enter standby request
Rsvd	27	Reserved.
CRC32	32	Check code

9.2.4.5.6 Standby Response Message

The packet format of the standby response message is shown in Figure 185, and the description of each field is shown in Table 138.

Figure 185 Packet format of a standby response message

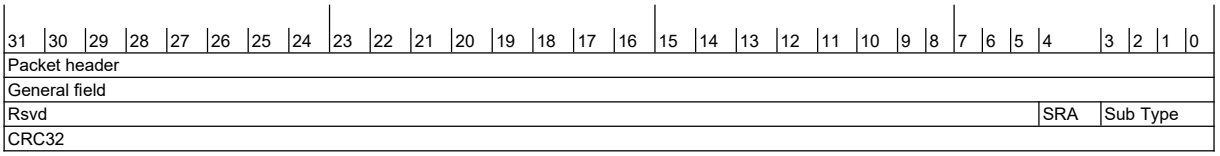


Table 138 Description of fields in a standby response message packet

Field Name	Length (Bits)	Description
Packet header	32	Refer to Section 9.2.1.1, where Type set to 31 indicates a port management message, and Rsp set to 1 indicates a response message.
General field	32	Refer to Section 9.2.1.1.
Sub Type	4	Subtype: 2h: standby message
SRA (Standby Request Ack)	1	Standby requests: 0b: standby request rejected; 1b: standby request accepted
Rsvd	27	Reserved.
CRC32	32	Check code

9.3 Device Management

9.3.1 Network Topology

The physical topology of the GPMI network is a network topology that supports interconnecting multiple GPMI device nodes into a network topology. A GPMI network supports access to up to 128 device nodes, and there are up to 127 levels of links between any two devices. A GPMI network is subject to distributed management by the management adapters of all devices in the network. Each device can construct a topology and a routing table (Table 84 Routing information table) with itself as the root. At the same time, a device may fall under the jurisdiction of multiple management adapters. This requires that the topology management mechanism must be able to adapt to this scenario and avoid conflicts between multiple management adapters.

Loops or parallel channels may exist in a network topology. The management adapter is responsible for identifying the loops and parallel channels by device addresses and forwarding lists. When there are multiple paths to access a GPMI device node, the management adapter selects which path shall be taken according to the service requirements.

Figure 186 Example of a GPMI network topology

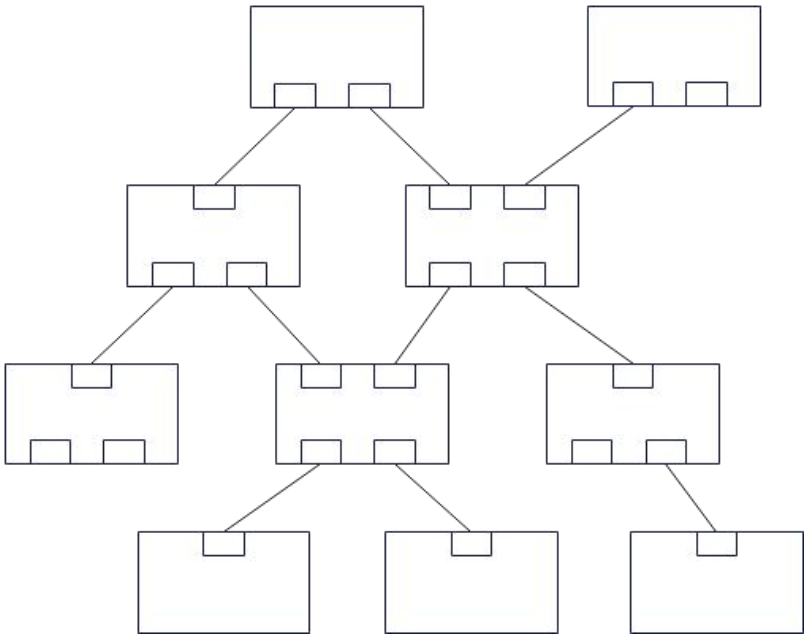
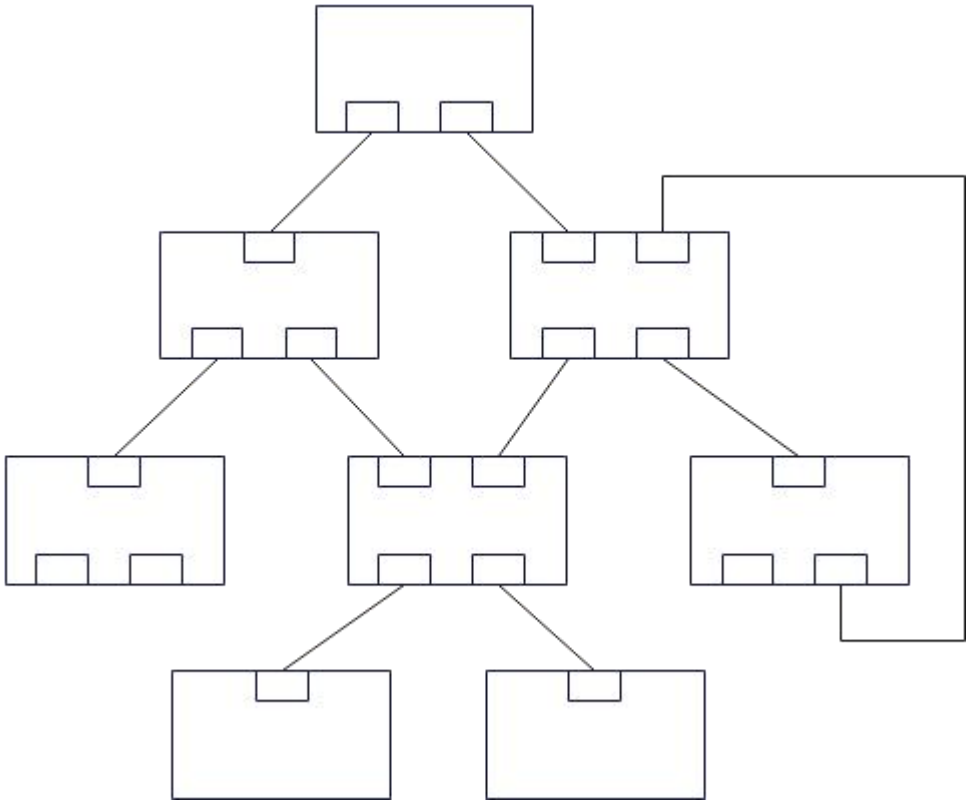


Figure 187 Example of a GPMI network topology with a loop



Loop Identification

The management adapter obtains the latest status information and connection relationship of each device in the GPMI network through topology discovery and topology update. Based on the saved connection status information, a topology with itself as the root can be maintained.

When a device is discovered, the management adapter identifies whether a loop or parallel link exists according to the forwarding list and the device address. If a destination address has two or more forwarding lists, there are multiple paths from the current device to the target device, that is to say, a loop exists.

9.3.2 Topology Management

9.3.2.1 Overview

Topology management is realized by using two functions, namely topology discovery and topology update. Management adapters implement the two functions by means of management messages. In this way, management adapters can discover devices that match their own service types, find optimal paths to destination devices, establish mapping tables between forwarding lists and device addresses, and graphically display all the devices and their interconnection relationships.

When any port of a device is connected to a new network, the management adapter of the device should conduct topology discovery to scan all the devices accessible through the port and their connection relationships, to construct the local topology.

After the topology is constructed, the management adapter should conduct topology update to trace the changes in the topology in real time and scan the changed status to update the topology. The topology update function focuses the changes in the topology, such as connection of any new device and disconnection of any existing device. In addition, when the status of any port of a device changes, the device shall send notification messages through all the other ports, to broadcast the latest status of the concerned port to the entire network, so the management adapters of the other devices in the network can complete topology update. Management adapters implement topology discovery by means of topology query requests and related response messages, and implement topology update by means of topology change requests and related response messages.

9.3.2.2 Topology Discovery

9.3.2.2.1 Overview

Each GPMI device will construct a local topology of the network where the device is located, and update it in real time. When two devices are connected, the networks where they are located will be connected to form a new and larger network. Each of the two connected devices will discover the topological structure of the network where the opposite device is located, and merge it with the existing local topology to obtain the topology of the merged network. This discovery process is completed by using topology query request messages and topology query response messages.

For example, as shown in Figure 188, Network 1 and Network 2 are originally two independent networks. In this example, Network 1 and Network 2 are connected by connecting Device A and Device E. After the port state machines of Device A and Device E switch to the Ready or Standby state after a series of jumps, topology discovery will begin.

- First, construct a topology query request message, and send it to the opposite device through the connected port.
- Then, after receiving the topology query response message from the opposite device, identify all the devices newly connected to the network and their connection relationships according to the device entries carried in the topology query response message and the port entries in

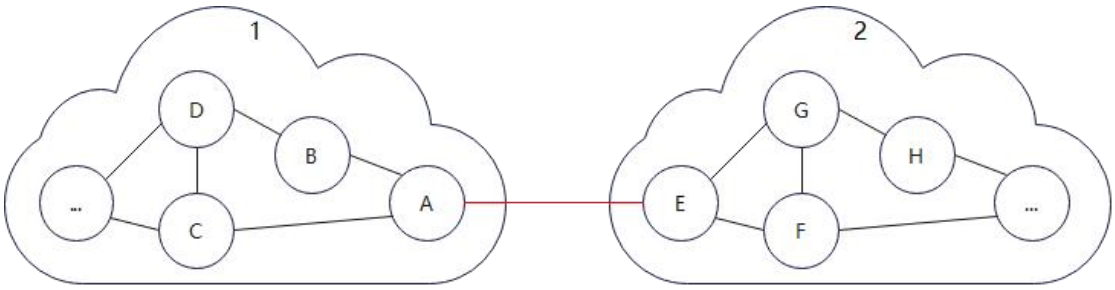
each device entry, that is to say, determine the NoDE devices in the network where the opposite device is located (the network newly accessed) and the connection relationships between the NoDE devices.

- Finally, obtain the topology of the merged network. The algorithm of this topological operation is not specified in this document.

Before a device conducts topology discovery, it shall complete the following two steps to ensure that the local topology is up to date:

- Wait until all the topology change request messages previously received are merged into the local topology, and are forwarded to other devices in the local topology.
- Wait until the other ports receive the topology query response messages corresponding to their topology query request messages and merge them into their topologies, and the topology change request messages generated by the topology query response messages are forwarded.

Figure 188 Example of topology management when two independent networks are connected



When the opposite device receives the topology query request message, it shall first merge all the topology change request messages previously received into its local topology, forward the messages, and then construct a corresponding topology query response message to reply. This topology query response message shall contain its own device entry (but not the port entry of the receiving port), as well as the device entries of all devices that can be connected to other ports except the port that received the topology query request message. Whether to carry the port entries of ports in the Disconnected state is not mandatory in this document.

Since the topology query response message constructed by the opposite device does not carry the port entry of the port receiving the topology query message, the device cannot obtain which port of the opposite device it is connected to according to the topology query response message from the opposite device. The device shall record the PortID in the capability negotiation request message from the opposite device, and merge the opposite device together with the other device entries into the topology after receiving the topology query response message of this time.

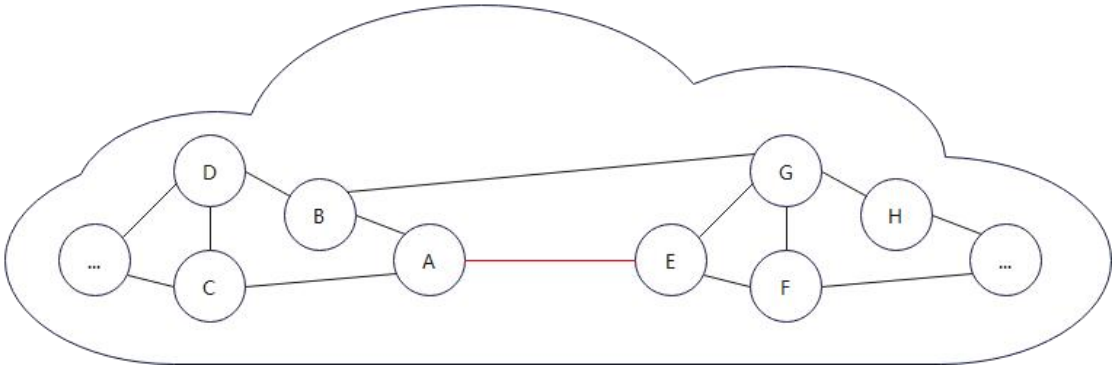
For the process related to topology changes mentioned in this section, refer to 9.3.2.3.

9.3.2.2.2 Loop Processing

Since the GPMI is of a network topology, a loop may exist between two devices. For example, as shown in Figure 189, before Device A and Device E are connected, Network 1 and Network 2 are already connected through Device B and Device G. Therefore, Device A and Device E locally maintain topologies that already contain Network 1 and Network 2 before they are connected, and they do not need to send any topology query request message to obtain the topology of each other after they are connected.

The device shall first determine if the opposite device is in the local topology, and if not, send a topology query request message for topology discovery. If the opposite device is already in the local topology, the device and the opposite device have been connected already, which means that there is one or more additional parallel channels between the two devices, that is to say, a loop exists. In this case, no topology query request message should be sent, and the new link can be directly merged into the topology.

Figure 189 Example of topology management when a new connection is added in the network



If the opposite device is not in the local topology but the device finds that its own device entry is already in the device entries carried in the topology query response message received, the device already exists in the topology of the opposite device. In this case, it is impossible to determine which topology is outdated, that is to say, it is impossible to determine whether a loop actually exists between the two devices. In this case, the device shall wait until any other port receives a topology change request message carrying the device entry of the opposite device, and then merge the opposite device into the local topology. If the topology of the device is outdated compared to that of the opposite device, that is to say, a loop exists, it will receive an expected topology change request message soon, and then merge the new link into the topology as the case that a loop exists. If no loop exists, that is to say, the topology of the opposite device is outdated compared to that of the device, no expected topology change request message will be received, and the device shall try to send a topology query request message again after tChannelTimeout. It is expected that the opposite device has completed topology update after the above period of waiting. After the device receives the topology query response message corresponding to the resent topology query request message, it will complete topology discovery as the case that no loop exists if it finds that its device entry is no longer in the topology of the opposite device, or start a new wait according to the above process if it finds that its device entry is still in the topology of the opposite device.

Note: In addition to being used during topology discovery as described in this section, topology query request messages and topology query response messages are also used in the process of handling topology change exceptions. For details, refer to 9.3.2.3.4.

9.3.2.3 Topology Update

9.3.2.3.1 Overview

This section is about the topology update process when the local topology changes due to events such as connection, disconnection, and adapter status change.

When two devices are connected so that their networks are connected, they will complete topology discovery first. At this moment, the other devices in the two networks have not yet perceived this connection, and their locally maintained topologies should also be updated. In this

case, topology change request messages and topology change response messages shall be used for topology update.

For example, as shown in Figure 188, no loop exists between Device A and Device E. After they have completed topology discovery normally as described in 9.3.2.2, they will immediately go on with topology update. The first step is to construct a topology change request message, whose device entries shall contain the device entry of the local device carrying the port entry of the port newly connected, and shall also contain all the device entries carried in the topology query response message received during topology discovery. Then the topology change request message is sent out through all the other ports in the Ready or Standby state.

For another example, as shown in Figure 189, a loop exists between Device A and Device E. Then, the device will also generate a topology change request message after topology discovery, and forward it according to the process described above. The difference is that this topology change request message only needs to indicate the change of this connection, with no new device entries added. That is to say, only the device entry of the local device is contained, and the port entry of the port newly connected this time should be contained in the device entry.

The two topology update scenarios described above are both subsequent processes of topology discovery triggered by port connection. When a port is disconnected or the adapter status changes, the device where this change occurs shall also immediately update the topology after completing the local topology update. That is to say, a corresponding topology change request message shall be generated and forwarded to all the ports in the Ready or Standby state. In the case that a port is disconnected, this message shall contain its own device entry, the value of the "CT" field shall be 2, and this message shall carry the port entry of the port disconnected. If no loop exists between the device and the device disconnected, this message shall also contain the device entries of the device disconnected and all devices that it can be connected to, and the value of the "CT" field shall be 0. In the case that the adapter status changes, this message shall contain its own device entry, the value of the "CT" field shall be 1, the value of the "NoPE" shall be 0, this message shall not carry any port entry, and the "AOF" field shall be filled in according to the current latest status.

The device that receives the topology change request message shall first construct a topology change response message and reply through the receiving port, then merge the device entries carried in the topology change request message into the local topology, and finally send the topology change request message through other ports in the Ready or Standby state. However, if the source device of this topology change request message does not exist in the local topology, the device will directly discard this message after replying to the topology change response message, but not perform the merging and forwarding process. Under normal circumstances, this message is forwarded, and the subsequent devices that receive this topology change request message will also process and forward it according to the above process, similar to the broadcast mechanism, until all the other devices in the network receive this topology change request message, and all devices in the entire network complete topology update.

9.3.2.3.2 Reliability of Topology Change Request Messages

Topology change request messages are sent to every device in the network in a way similar to the broadcast mechanism, and there is no end-to-end reply response message. To ensure their reliability, these messages should be retransmitted at the link level when an error occurs. For each topology change request message, if no topology change response request is received after tChannelTimeout (which shall be shorter than the timeout period for an end-to-end message), this message shall be resent up to 5 times. If the result is timeout for all the 5 retries, this message transmission is considered as failed.

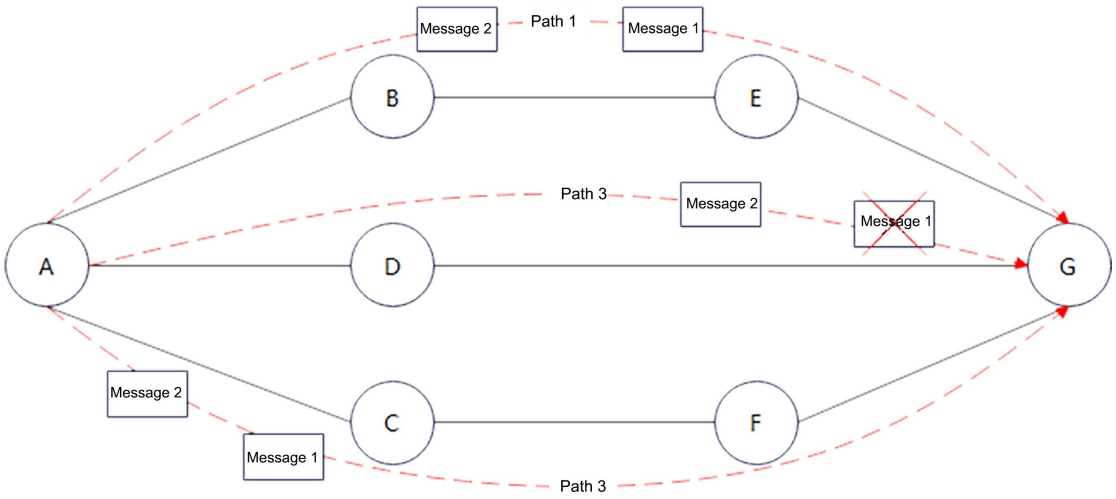
Topology change request messages support adaptive selection of high-speed or low-speed links for sending, and the transmission delays of these two links are different. In order to avoid out-of-order topology change request messages, when a port has sent a topology change request

message but receives no corresponding response message, it shall not send any other topology change request message to this port, but wait in line until the previous message is completed or fails.

9.3.2.3.3 Topology Change Identifiers

When a device's topology changes, it should notify all the devices in the entire network by means of topology change request message in a way similar to broadcast mechanism. In a GPMI network, one or more loops may exist between two devices, and the lengths of these channels may be different. In addition, topology change request messages support adaptive selection of high-speed or low-speed links for sending, and the transmission delays of these two links are different. For example, as shown in Figure 190, there are three channels between Device A and Device G, namely Channel 1, Channel 2, and Channel 3. Among them, high-speed link is adopted for Channel 1 and Channel 2, and low-speed link is adopted for Channel 3. The order of the three channels in terms of link delay from small to large is Channel 2, Channel 1, and Channel 3. In this example, Device A generates two topology change request messages consecutively, namely Message 1 and Message 2, which are broadcast through these 3 channels respectively. If packet loss occurs to Message 1 between Device D and Device G, the order of the messages that Device G finally receives is Message 2, Message 1, Message 2, Message 1, and Message 2. It can be seen that a device in a GPMI network may repeatedly receive old messages that have been processed, or receive multiple topology change request messages from another device out of order due to packet loss, that is to say, the topology change request message generated first arrives later than the one generated later.

Figure 190 Example of topology change



Topology change request messages carry the changed content in an incremental form, and each change message only carries the change compared to the previous state. Therefore, it is required that GPMI devices must continuously process all topology change request messages from a certain device in the order in which they are generated, and cannot directly process them in the order of reception. Otherwise, problems such as information exception, state rollback, and state oscillation may occur. For example, let us look at Figure 190 again and assume that Message 1 and Message 2 report a disconnection event and a connection event of a port of Device A respectively. The disconnection report, which is supposed to reach Device G first, is lost between Device D and Device G, so Device G receives the connection report first, and then Device G may think that Device A is in the exception state. Subsequently, Device G receives the event combination of "Disconnection - Connection - Disconnection - Connection". Although the final state is correct, the topology has unnecessary oscillation, increasing the burden on the system.

In order to avoid out-of-order or repeated processing of topology change request messages, the "Change ID (topology change identifier)" field is added to each message to indicate the order of the messages generated by a certain device. When the device receives a topology change request message, it can determine whether there is packet loss or repeated packets based on the topology change identifier carried in the message.

Each device shall maintain its own topology change identifier internally, which is initialized to 0 after power-on. Whenever it generates a topology change request message, it fills in the "Change ID" field with its current topology change identifier, and adds one to its topology change identifier. The value range of the topology change identifier is from 0 to 255. If the current value is 255, it will become 0 next time, that is to say, it changes by adding one by one in a cyclic manner. When a receiving device sees that the topology change identifier changes from 255 to 0, it shall be the two messages concerned consecutive.

Furthermore, each device shall record the topology change identifiers of the other devices in the topology. When a device is added to the topology through a topology query response message or a topology change request message, the value of the "Change ID" field in its device entry shall also be recorded as its topology change identifier. When a device is deleted from the topology, it will no longer be necessary to record its topology change identifier.

After a device receives a topology change request message and replies to the topology change response message, it shall compare the carried value (the value of the "Change ID" field carried by the message) with the recorded value (the topology change identifier of the message source device recorded locally), to determine what to do next:

— In one of the following cases, the device determines that a normal topology change has occurred as the topology change identifiers are consecutive, and therefore performs the merging and forwarding process as normal (to merge the topology of the device indicated in the topology change request message):

- (Carried value – Recorded value) == 1.
- Carried value == 0, and recorded value == 255.

— In one of the following cases, the device determines that a topology change exception has occurred as repeated topology change request messages have been received, and therefore it does not perform the merging and forwarding process but discards the messages directly, so as to avoid unnecessary topology oscillation and prevent topology change request messages from being forwarded endlessly in the GPMI network:

- Carried value == Recorded value.
- Carried value < Recorded value, and (Recorded value – Carried value) < 128.
- Carried value > Recorded value, and (Carried value – Recorded value) >= 128.

— In one of the following cases, the device determines that a topology change exception has occurred as a jump of topology change identifier has happened with packet loss, and therefore handles the topology change exception as described in 9.3.2.3.4:

- Carried value < Recorded value, and (Recorded value – Carried value) >= 128.
- Carried value > Recorded value, and (Carried value – Recorded value) < 128.

For example, assuming that a device has multiple ports in the Ready or Standby state, and the rates of different ports vary greatly. When it receives multiple topology change request messages from the same source device in succession, it is recommended to suspend forwarding the topology change request message when a faster port has forwarded more than N messages ahead of a slower port (for which N must be less than 128), to prevent other devices from misjudging repeated messages as an exception with packet loss.

9.3.2.3.4 Topology Change Exception

When a device determines that a packet loss has occurred to the topology change request message after comparing the topology change identifiers, the currently received topology change request message may not accurately reflect the actual state of the current topology, and this topology change exception request message cannot be merged into the topology. This situation is called a topology change exception. This section describes the method for solving topology change exceptions and the action requirements for devices before the exceptions are solved.

The management adapter of each device shall maintain a record of topology change exception state for every device in the topology to indicate whether a packet loss has occurred to the topology change request message from every device. When a device determines that an exception has occurred according to a topology change request message, it shall locally store the message but not perform the merging and forwarding process temporarily, to prevent this error from spreading to the network.

If any neighbor device connected through the port receiving the topology change request message can forward this message, no packet loss has occurred before. In order to solve this topology change exception, the neighbor device should send a topology query request message and perform a full topology synchronization with the device. After it receives the topology query response message from the neighbor device, it shall also ensure that the topology of the neighbor device is not outdated compared to the local one, and then merge this topology query response message into the local topology to avoid topology fallback after merging. Only when the value of the "Change ID" field in each device entry carried in the message is not less than the topology change identifier recorded locally for the neighbor device, the topology of the neighbor device is not outdated compared to the local one. If the topology of the neighbor device is outdated compared to the local one, the device shall resend the topology query request message after tChannelTimeout. If this is still the case after 10 retries, possibly the neighbor device is also in the topology change exception state, and is waiting for the device to solve the topology change exception first. In this case, the management adapter shall try to send a topology query request to other ports to solve this exception. If the topology of the neighbor device is not outdated compared to the local one, this topology query response message will be merged into the local topology and the device's topology change identifier recorded locally will be overwritten with the value of the "Change ID" field in each device entry.

Before a device sends a topology query request message to a neighbor device, it shall also complete the following steps to ensure that the local topology is up to date:

- Wait until all previously received topology change request messages have been merged into the local topology and forwarded.
- Wait until the other ports receive the topology query response messages corresponding to their topology query request messages and merge them into their topologies, and the topology change request messages generated by the topology query response messages are forwarded.
- After the above two steps, if the exception is still not solved, start to send topology query request messages to neighbor devices.

In order to avoid introducing more errors and to prevent the topologies of neighbor devices from always being outdated compared to the local one, if there is any topology change exception, the topology change request message received after the exception occurs will no longer be merged into the local topology, nor will a new topology discovery process be initiated. However, if a topology query request message is received, a topology query response message carrying the current topology shall be replied according to the rules.

If a device in the topology is in the topology change exception state and any port receives a topology change request message from the device, the message will not be merged or forwarded

but temporarily stored locally, until the topology change exception of the device is solved. This situation remains unless this message can solve the topology change exception of the device, that is to say, the value of the "Change ID" field it carries can restore the continuity of the device's topology change identifier.

If a device in the topology is in the topology change exception state and any port receives a topology change request message from any other devices that are not in the exception state, the message will not be merged into the process, but the forwarding process can still be performed as normal. In addition, it must be temporarily stored locally, and will be merged into the topology after all topology change exceptions before this message are solved. It shall be noted that the continuity of its topology change identifier shall also be determined before forwarding. If an exception occurs to a topology change request message, the subsequent topology change request messages of the device will be handled as the device is in the exception state, that is to say, they will not be merged or forwarded.

After the topology change exception of a device is solved by topology query response messages or topology change request messages, the topology change request messages temporarily stored locally shall be processed in sequence according to the order in which they are received. None of these messages were merged into the topology, but some have been forwarded and some have not. The decision to merge the message into the local topology shall be made based on the topology change identifier currently recorded locally. However, for any topology change request message that is not forwarded, even if the topology change identifier in the message is smaller than the current value locally recorded, the forwarding process shall be performed in sequence according to the rules. When these topology change request messages temporarily stored locally are handled, new exceptions that may occur can be solved in sequence according to the process described in this section.

- Note 1: When a topology query response message triggered by an exception is received, if the exception has been cleared, the topology query response message will be discarded directly.
- Note 2: When a topology change exception occurs, if the topology change request message is received at this port after sending the topology query request message and before receiving the response message, although the changes it carries must have been merged into the subsequent topology query response message, the device shall still temporarily store this topology change message locally and cannot discard it after forwarding. This is because topology change exceptions may be cleared in advance, and subsequent topology query response messages will be ignored.
- Note 3: When a topology change exception occurs, the topology change identifier used for forwarding and the topology change identifier merged into the topology may be temporarily inconsistent. The management adapter shall temporarily maintain two copies of topology change identifier for each source device.

9.3.2.4 Topology Management Process

9.3.2.4.1 Sending a Topology Query Request Message

The management adapter shall send a topology query request message when any of the following situations occurs:

(a) If the port state machine of any port (say Port A) of the device switches from the Opposite Device Detection state to the Ready or Standby state after a series of jumps, topology discovery shall be performed:

  (1) If any device in the topology is in the topology change exception state, wait until all exceptions are solved, and then go on with the next step.

(2) If any other port of the device has sent any topology query request message, wait until it receives the topology query response message and completes the merging and forwarding process, and then go on with the next step.

(3) If the address of the opposite device does not exist in the local topology, that is to say, no loop exists between the two devices, send a topology query request message to the opposite device through Port A; otherwise, do not send any topology query request message, but send a topology change request message to the other ports except Port A.

(b) If any port of the device receives a topology change request message, but an unexpected jump has occurred when comparing the topology change identifier carried in the message to that recorded locally.

After receiving the corresponding topology query response message, the management adapter shall perform the following actions:

(a) If the topology query request message is issued due to topology discovery:

(1) If the address of the device already exists in the topology of the opposite device, that is to say, a loop exists between the two devices:

- Wait for the topology change request messages from the other ports.
- If a topology change request message is received and the address of the opposite device has been merged into the local topology, send the topology change request message concerning this connection to the other ports except Port A, end the process, and take no further steps.
- If the timeout occurs (1 second), send the topology query request message again, and return to Step 1 after receiving the corresponding topology query response message.

(2) If no loop exists between the two devices:

- Merge the topology query response message into the topology.
- Inherit the topology change identifiers in all the device entries.
- Construct a topology change request message based on this response message and forward it to the other ports.

(b) If the topology query request message is issued due to a topology change exception:

(1) If the topology change exception of the device that triggered this topology query has been cleared, directly discard this topology query response message and take no further steps. Otherwise, go on with the next step.

(2) If the topology in the topology query response message is outdated compared to the local one:

- Resend the topology query request message after 1 second. If this is still the case after a total of 10 seconds, try to send the topology query request message to the other ports, and then return to step 1.

(3) If the topology in the topology query response message is not outdated compared to the local one:

- Merge the topology query response message into the local topology, and inherit the topology change identifiers in all the device entries.
- For the topology change request messages from this source device in the exception state that are temporarily stored locally and have not yet been forwarded, even if the

topology change identifier in the message is smaller than the one currently maintained locally, forward the messages in sequence according to the rules.

- Determine whether to merge all the topology change request messages temporarily stored locally into the local topology based on the latest topology change identifiers of the devices maintained locally.
- Clear the topology change exception of the source device of this topology change message.

9.3.2.4.2 Receiving a Topology Query Request Message

In order to cooperate with the opposite device to complete topology discovery, when a port (say Port A) receives a topology query request message, the device shall perform the following actions:

(a) Forward all the received topology change request messages, merge them into the local topology, and then go on with the next step. If any new topology change request message is received during this process, it is recommended to go on with the next step first to construct and send a topology query response message. Of course, if there is no topology change exception, you can also merge these topology change messages into the local topology first, forward them, and then go on with the next step, but it is not recommended to do so to avoid introducing more errors.

Note: If any other port of the device has sent any topology query request message, there is no need to wait for its topology query response message.

(b) Construct a topology query response message carrying the device entry of the device and the device entries of all the devices that can be connected to the other ports in the topology except Port A, and send it to the opposite end through Port A.

9.3.2.4.3 Sending a Topology Change Request Message

In order to cooperate with the other devices in the network to complete topology update, the management adapter shall send a topology change request message when any of the following situations occur:

— If the port state machine of any port of the device switches from the Opposite Device Detection state to the Ready or Standby state after a series of jumps:

- Complete topology discovery.
- If there is no loop with the opposite device, construct a topology change request message based on the topology query response message received during topology discovery and forward the message to the other ports.
- If there is a loop with the opposite device, construct a topology change request message that only carries the port entry of this port and forward the message to the other ports.

— If the status represented by any field in the device entry of this device changes, a topology change request message will be sent to all ports only when the adapter status changes according to the current version of the document.

If the status represented by any field in the device entry of this device changes, a topology change request message will be sent to the other ports only when the port is disconnected according to the current version of the document. This topology change request message shall contain the device entry carrying this port entry and the device entries of all failed devices, and the change type is "0: Delete Device".

9.3.2.4.4 Receiving a Topology Change Request Message

When a port receives a topology change request message, it shall perform the following actions:

(a) Reply to the topology change response message.

(b) Do not forward or merge the message but discard it directly if the source device of the message is not in the local topology or it is determined according to the topology change identifier that the message is a repeated one.

(c) If the source device of this message is in the topology change exception state:

(1) If the continuity of the topology change identifier carried in this message can be restored:

- Perform the merging and forwarding process.
- Clear the topology change exception state of the source device of the message.
- Merge all the topology change messages temporarily stored locally.
- If a topology query request message has been sent through a port due to the topology change exception of this source device, regard the process as completed, no longer wait for the topology query response message, and directly discard any corresponding topology query response message received later.

(2) If the continuity of the topology change identifier carried in this message cannot be restored:

- Do not merge or forward the message, but temporarily store it locally.

(d) If the source device of this message is not in the topology change exception state:

(1) If the topology change identifier in the message is consecutive:

- Forward this topology change request message to the other ports.
- If any other device is in the topology change exception state, do not merge the message into the local topology but temporarily store it locally.
- If no device is in the topology change exception state, merge the message into the local topology, and update the topology change identifier of this source device recorded locally.

(2) If it is determined according to the topology change identifier that a packet loss has occurred:

— If any other device is in the topology change exception state, wait until all previous topology change exceptions are solved, and then, if the unexpected jump is still not solved, go on with the next step.

— If no device is in the topology change exception state:

- If any topology change request message is being merged into the local topology, merge and forward it.
- If any other port of the device has sent any topology query request message, wait until it receives the topology query response message and merges the message into the topology, and then, if the unexpected jump is still not solved, go on with the next step, and otherwise, end the process.
- Do not forward or merge this topology change request message, but temporarily store it locally.

- Send a topology query request message to the opposite device through the receiving port.

9.3.2.5 Refreshing the Address Table

The management adapter shall generate an address table based on the topology, and refresh the address table immediately when the topology changes.

If multiple ports can communicate with a destination device and the loop shall be deleted, the management adapter shall use the port with the shortest distance to forward messages to the destination device, to avoid the address table addressing packet from forming an endless loop in the GPMI network and never reaching the destination device.

9.3.3 Equipment Resource Management

9.3.3.1 Overview

With the topology management function, management adapters can obtain brief information about every device in the network, such as the status of each port, the total bandwidth, and the types of adapters contained in every device. But they cannot obtain all the capacity information about the ports, the number of adapters of each type, the adapter list, the capacity of each adapter, or whether there is any USB3 hub inside. If a management adapter intends to establish a service relation with a device, it should further obtain detailed information on the device's capabilities and internal resources. In this case, the management adapter should actively read such information by reading the DCCD management message. In addition, if the online status of any adapter of a device changes, the device should actively send an adapter change request message to notify the other devices in the GPMI network.

9.3.3.2 DCCD

For the definition of DCCD, refer to Appendix F.

When a GPMI device is initialized, the corresponding DCCD shall be prepared. In addition, it is required to make sure that the capabilities declared in the DCCD match the capabilities of the GPMI device, that is to say, the GPMI shall be able to support the capabilities declared in the DCCD.

The data for the DCCD is obtained by means of DCCD read request and response messages. When a GPMI device needs to obtain the capabilities of another GPMI device as declared in the DCCD, it can initiate a DCCD read request message. When a GPMI device receives a DCCD read request message, it shall return the corresponding DCCD value according to the requirements of the request message.

When a GPMI device receives a DCCD read request message but its DCCD is not prepared yet, it can reply with a corresponding response message by returning the error code (0x1) Busy.

9.3.3.3 Adapter Management

9.3.3.3.1 Overview

Each device does not need to discover or maintain all the adapters of devices in the network, but just the adapters it is concerned with in the network, such as those that match its own service type.

9.3.3.3.2 Adapter Discovery

During topology discovery, the device entries of the devices in the network can be obtained. The "AOF" field in a device entry represents what types of adapters exist in the device. However, the "AOF" field only represents the presence or absence of each type of adapter, but not the quantity of adapters of each type, their adapter numbers, or capabilities. Therefore, the "AOF" field can only be used to preliminarily determine the service type supported by the device. If its service type matches itself, it is required to further obtain a detailed adapter list and capabilities by reading the DCCD.

9.3.3.3.3 Adapter Status Update

After the adapter lists of the other devices in the network are obtained, it is also required to trace the changes of the adapter lists in real time, such as the case when a new adapter goes online or an existing adapter goes offline. In this case, adapter change request messages and adapter change response messages are required to complete the adapter status update.

After a device has replied to a topology query request message on any port, if any further adapter status change occurs, it shall immediately construct an adapter change request message and forward the message through all the ports in the Ready or Standby state. To minimize the quantity of adapter change request messages, it is recommended that each device should enable the ports after all the adapters are initialized.

After a device receives an adapter change request message, it shall first construct an adapter change response message and reply through the port that receives the request message, then decide whether to merge the change carried in the request message into the local record according to its own needs, and finally forward the request message through the other ports in the Ready or Standby state. However, if the source device of this adapter change request message does not exist in the local topology, the device will directly discard this message after it replies to the adapter change response message, and it will not perform the merging and forwarding process. Under normal circumstances, the message is forwarded, and the subsequent devices that receive this adapter change request message will also process and forward it according to the above process, similar to the broadcast mechanism, until all the other devices in the network receive this adapter change request message, and all the devices in the entire network have completed adapter status update.

For the reliability of adapter change request messages, refer to 9.3.2.3.2.

9.3.3.3.4 Adapter Change Identifiers

The causes of adapter change identifiers, as well as the generation and processing rules, are introduced.

The difference between adapter change identifiers and topology change identifiers lies in that each device only maintains the adapter change identifiers of those devices it is concerned with. If an adapter change identifier does not match the service type of the device, it will not merge the adapter change request message from that device, but it should still forward the message to the other ports.

9.3.3.3.5 Adapter Change Exception

When a device receives an adapter change request message, it will determine whether it is consecutive, repeated, or with packet loss based on the adapter change identifier. For the judgment rule, refer to 9.3.2.3.3.

Unlike the method for handling topology change exceptions, if an adapter change exception occurs, the device cannot initiate full synchronization to the neighbor devices of the receiving port

because the neighbor devices do not necessarily maintain the adapter information of the source device where the exception occurs. Instead, a full query shall be made to the source device in the exception state by reading the DCCD. In addition, even if an exception occurs, the device should still forward this adapter change request message but not temporarily store it locally, because other devices can only perform full query on the source device in the exception state.

9.4 Port Management

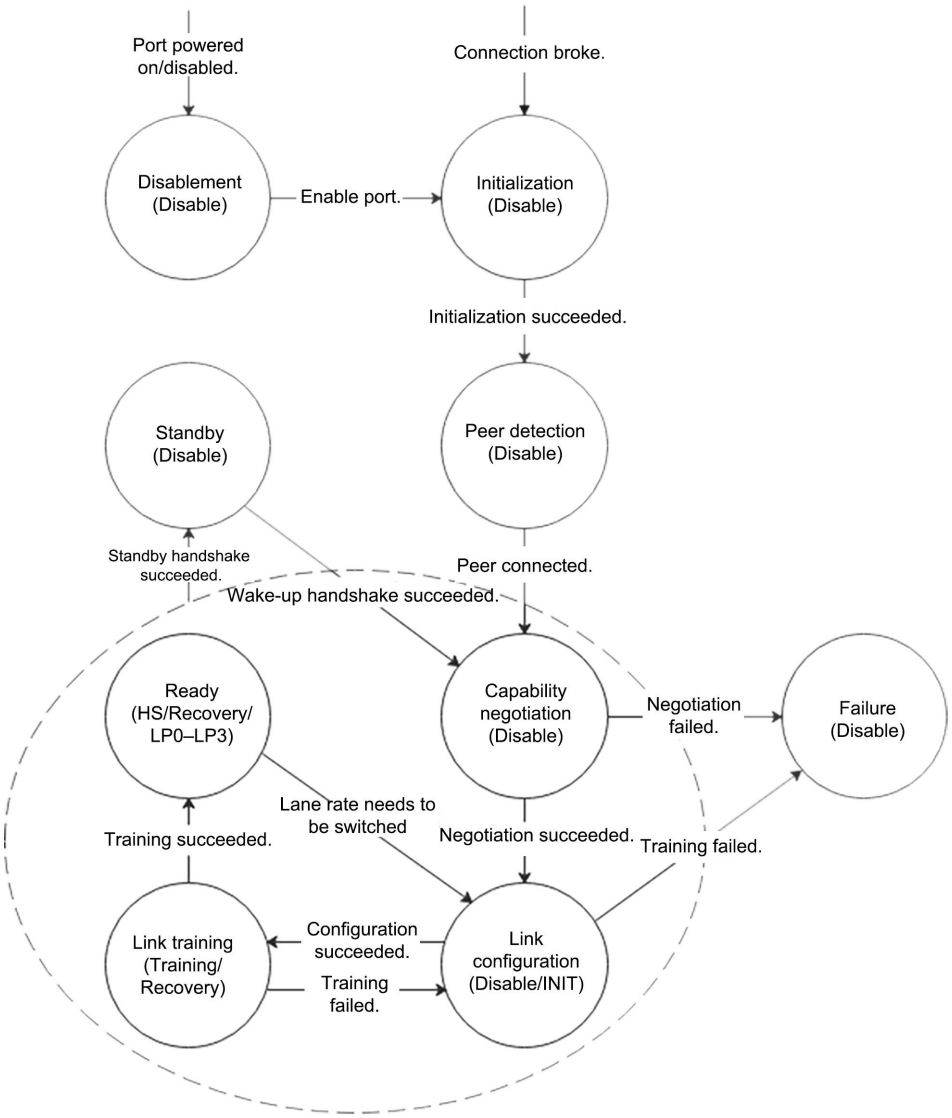
9.4.1 Overview

Management adapters shall implement the port management functions described in this section.

With the port management functions, management adapters can disable and enable ports, interact with peripheral interface controllers, negotiate port capabilities, configure link modes, switch ports to standby mode, and wake up ports. The functions such as port disconnection, main link lane number exchange, and sideband link polarity reversal in the Initialized state, link training, and state machine jump should be implemented in coordination with the logical layer.

Each management adapter maintains a port state machine for each physical port. The behavior and jump conditions in each state are as shown in Figure 191, where the first line in each circle is the port state, and the second line in brackets is the main link state corresponding to this port state.

Figure 191 Port state machine



9.4.2 Disabled State

9.4.2.1 Entry Conditions

When a device is powered on, all of its ports are in the Disabled state by default. Or, when a port receives the port disabling instruction in any state, it shall enter the Disabled state. The port disabling instruction is generally initiated by software.

9.4.2.2 Action

After a port enters the Disabled state, all transmissions on the main link and the sideband link of the port shall be stopped and any received data ignored. The main link remains in the Disabled state. The SBTX and SBRX of the sideband link are both in a high-impedance state and cannot be detected by the opposite device, nor do they have the ability to detect signals.

9.4.2.3 Exit Conditions

A port will exit the Disabled state and enter the Initialized state only after it receives the port enabling instruction. The port enabling instruction is generally initiated by the software after its own hardware and peripheral matching modules are initialized.

9.4.3 Initialized State

9.4.3.1 Entry Conditions

When a port receives the port enabling instruction in the Disabled state, it will exit the Disabled state and enter the Initialized state only after. Or, when a port detects a disconnection in any state, it will enter the Initialized state.

Note 1: For the judgment conditions of connection and disconnection, refer to 6.4.

Note 2: When a port is in the Disabled state, Initialized state, or Opposite Device Detection state, it cannot detect disconnection. Therefore, a port in any of these three states will not enter the Initialized state due to disconnection.

9.4.3.2 Action

After a port enters the Initialized state, all transmissions on that port shall be stopped and any data received on the port ignored. The main link remains in the Disabled state, and the sideband link remains in the Disconnected state as instructed. Then the following actions will be performed:

(a) Obtain the following information, and the specific method is not within the scope of the document:

- Whether the opposite device is a GPMI device.
- Whether the type of the local connector is USB Type-C or GPMI Type-B.
- Whether the cable is plugged reversely.
- Whether there is Cable Info.

(b) If the above information is not successfully obtained, or if the above information is obtained but the opposite device is not a GPMI device, the port will not go on with the subsequent steps, and the port state machine will remain in the Initialized state.

(c) If the cable is plugged reversely, the port will make adjustments according to the requirements of connector type, for example changing the lane number of the main link and reversing the polarity of the sideband link.

(d) If there is Cable Info, the port will start to obtain Cable Info. If this obtaining operation fails, the port will being regarded as having no Cable Info, and report an alarm to notify the software. It should be noted that even if there is no Cable Info or this obtaining operation fails, it does not affect the subsequent steps or the port state machine jump.

(e) SBTX has been brought down for a time longer than Tsltx_disconnect since the last disconnection.

(f) Initialization succeeded.

9.4.3.3 Exit Conditions

When a port is successfully initialized after the above steps, it will exit the Initialized state and enter the Opposite Device Detection state. Or, when a port receives the port disabling instruction, it will exit the Initialized state and enter the Disabled state.

9.4.4 Opposite Device Detection State

9.4.4.1 Entry Conditions

When a port in the Initialized state is successfully initialized, it will exit the Initialized state and enter the Opposite Device Detection state.

A certain period of time (Tsltx_disconnect) shall be reserved between unplugging and entering the next Opposite Device Detection state to ensure that the internal cached data packets have been cleared, to prevent the management adapter packet of the previous device from being sent to any newly connected device.

It is recommended that each management adapter identify scenarios where a port is frequently plugged and unplugged, to reduce the generation of topology change request messages without affecting the functions, and avoid network congestion. For example, a possible method is to limit the number of topology change request messages in 1 second generated by port plugging and unplugging. For another example, when multiple topology change request messages generated by plugging and unplugging are cached locally and have not been sent out, all other identical topology change request messages except the last one can be omitted, but it is necessary to make sure that the last topology change request message issued can reflect the current real port status.

If a port wants to exit the Error state, or to restart a capability negotiation with the opposite device, it can actively bring down the SBTX for a time longer than Tsltx_disconnect, and then bring up the SBTX to achieve an effect similar to resetting the port.

9.4.4.2 Action

After a port enters the Opposite Device Detection state, it will detect the status of the SBRX to determine whether the opposite device is connected, and will bring up the SBTX. The sideband link tends to ignore any data received on the SBRX and cannot send any management adapter packets until a connection is detected.

After a port detects a connection, it shall continue to maintain the SBTX high for a time longer than 1 ms to make sure that the opposite device has enough time to detect the device. If the SBRX detects a connection and immediately sends a packet through the SBTX, it may cause the opposite device to fail to detect the device because the SBTX is not always high when the device sends packets. However, if the SBRX receives a capability negotiation request message within 1 ms, it can enter the Capability Negotiation state in advance because the opposite device has successfully detected the device and entered the Capability Negotiation state.

9.4.4.3 Exit Conditions

After a port successfully detects the connection of the opposite device, if it maintains the SBTX high for 10 ms or receives a capability negotiation request message, it will exit the Opposite Device Detection state and enter the Capability Negotiation state. Or, when a port receives the port disabling instruction, it will exit the Opposite Device Detection state and enter the Disabled state.

9.4.5 Capability Negotiation State

9.4.5.1 Entry Conditions

When a port in the Opposite Device Detection state successfully detects the connection of the opposite device, it will exit the Opposite Device Detection state and enter the Capability Negotiation state. Or, when a port in the Standby state successfully wakes up the handshake, it will exit the Standby state and enter the Capability Negotiation state. For the conditions that a port enters and exits the Standby state and the processes that a port enters and exits handshake, refer to 9.4.9.

9.4.5.2 Action

After a port enters the Capability Negotiation state, it will enable the sideband link to send and receive packets, and then start to send capability negotiation request messages through the sideband link. If the port has received the capability negotiation response message corresponding to the sent capability negotiation request message, and in addition, it has received the capability negotiation request message of the opposite device and has replied to the corresponding capability negotiation response message, it is considered that a successful capability negotiation handshake is achieved. If a capability negotiation request message is received from the opposite device again after a successful handshake, the last message received shall prevail.

After a successful capability negotiation handshake is achieved, it is required to determine whether either of the two devices is in the Standby state. If the answer is yes, it will not be taken as a successful negotiation, that is to say, neither of the two ports will jump to the Link Configuration state. The "Standby State" field in the capability negotiation request message sent by the opposite device can be used to determine whether the opposite device is in the Standby state. The device in the Standby state shall actively initiate a port standby request message after a successful capability negotiation handshake, and enter the Standby state after a successful standby negotiation. Refer to 9.4.9.

For a port in the Capability Negotiation state, if it has exited the Standby state after sending a capability negotiation request message, in which the value of the "Standby State" field is 1, it shall resend a capability negotiation request message, in which the value of the "Standby State" field is 0.

After a successful capability negotiation handshake, it is required to determine whether any conflict exists in the capability items carried in the capability negotiation request messages of the two devices. If any of the following conflicts exist, the negotiation is considered as failure:

- The addresses of the two devices are identical.
- The port type supported by the two devices is LMP or LSP.
- A conflict exists between the lane modes supported by the two devices, that is to say, there is no matching lane in any direction. If successful lane matching is achieved in more than one direction, it is considered that no conflict exists.

If the negotiation fails, the LMP shall notify the opposite device to enter the Error state by sending a link configuration request message, in which the value of the EE field is 1. If neither of the two devices can decide the role of its own port due to a capability conflict, each of them shall notify the other to enter the Error state. For the handshake for entering the Error state, refer to 9.4.10.1.

After a successful capability negotiation handshake, if neither of the two devices is in the Standby state and no capability conflict exists between them, the port type (LMP/LSP) is determined based on the information obtained. If both devices are DRP, it is necessary to decide whether they are LMP based on their addresses. The device whose address is with a larger number is the LMP.

Then it is considered that a successful negotiation is achieved, and the ports enter the Link Configuration state.

A port in the Capability Negotiation state may receive any link configuration packet. If the value of the EE field in such a message is 1, the port shall reply the response message as normal and enter the Error state. Otherwise, it shall ignore the message and give no response.

If the addresses of the two devices are identical, it means that a device self-loop has occurred, that is to say, two ports of the device are connected. Except test scenarios, device self-loop data transmission is not supported in normal working scenarios. Therefore, in test scenarios, it may not be considered that the negotiation failed when the addresses of the two devices are identical.

9.4.5.3 Exit Conditions

A port will exit the Capability Negotiation state when it is in any of the following conditions:

- A successful capability negotiation is achieved, and the port enters the Link Configuration state.
- The capability negotiation fails, and the port enters the Error state after a successful handshake about entering the Error state is achieved.
- A successful negotiation about entering the Standby State is achieved, and the port enters the Standby state.
- The port receives a port disabling instruction, and enters the Disabled state.
- The port detects disconnection, and enters the Initialized state.

9.4.6 Link Configuration State

9.4.6.1 Entry Conditions

A port will enter the Link Configuration state from its current state when it is in any of the following conditions:

- A successful capability negotiation is achieved when the port is in the Capability Negotiation state.
- The port receives an instruction about a training failure in the logical layer when it is in the Link Training state.
- When the port is in the Ready state, the bandwidth management unit of the LMP management adapter believes that the lane rate in a certain direction should be switched.
- When the port is in the Ready state, the LSP receives a link configuration packet.

9.4.6.2 Action

9.4.6.2.1 Overview

After a port enters the Link Configuration state, it will start to perform the following steps:

(a) For the LMP:

(1) If the port entered the Link Configuration state from the Capability Negotiation state, it goes on with the next step. If the port entered the Link Configuration state from the Link Training state due to training failure, it should determine whether any error occurs in training. If the LMP fails to train with the same configuration for 3 consecutive times and

there is no other expected link establishment mode, it is considered that any error occurs in training. If any error occurs in training, the port sends a link configuration request message, in which the value of the EE field is 1, to notify the LSP to enter the Error state. For the handshake for entering the Error state, refer to 9.4.10.1. Otherwise, it goes on with the next step.

(2) The port configures the information that the link configuration request message should carry based on the information decision-making link obtained when it was in the Capability Negotiation state, and sends the link configuration request message through the sideband link. Then it waits for the corresponding link configuration response message. If the value of the "Fast Training Ack (FTA)" field in the link configuration response message is 0, it means that the opposite device refuses to use rapid link establishment, and then the LMP cannot enable rapid link establishment and does not need to resend the link configuration request message.

(3) The port configures the link mode, whether to enable rapid link establishment, and the information required for rapid link establishment to the logical layer and the physical layer, and notifies the logical layer to start establishing a link.

(4) Then the port considers that a successful configuration is achieved and enters the Link Training state.

(b) For LSP:

(1) It waits for a link configuration request message from the opposite device.

(2) If the value of the "Fast Training (FT)" field in the link configuration request message is 1, the port determines whether to accept rapid link establishment request based on whether it meets the conditions for rapid link establishment, which is reflected in the "Fast Training Ack (FTA)" field of the link configuration response message.

(3) The port configures the link mode and the information required for rapid link establishment to the logical layer and the physical layer, and notifies the logical layer to wait for the opposite device to establish a link.

(4) The port sends a link configuration response message, considers that a successful configuration is achieved, and enters the Link Training state.

Note: The LSP will not judge whether any error occurs during training, so the LSP in the Link Configuration state will not actively notify the opposite device to enter the Error state.

A port in the Link Configuration state may receive any capability negotiation request message, and it shall reply the capability negotiation response message as normal.

9.4.6.2.2 Link Mode

When the LMP makes a decision on the link mode, it is necessary to refer to Table 139 and Table 140 for the initial link establishment mode after plugging and unplugging, and start link establishment from the highest rate. After the port wakes up from the Standby state, it can start link establishment from the mode and rate before it entered the Standby state. If the link establishment fails, the management adapter of the LMP will choose to slow down or close certain lanes. The specific fallback strategy is not limited by this document.

Table 139 Optional initial link establishment modes for simplified-specification LMP

LMP	LSP	Recommended Initial Link Establishment Mode for LMP
DRP:4TR	LSP:4R	4T

LMP	LSP	Recommended Initial Link Establishment Mode for LMP
LMP:4T	DRP:4TR	4T
LMP:4T	LSP:4R	4T

Table 140 Optional initial link establishment modes for full-specification LMP

LMP	LSP	Recommended Initial Link Establishment Mode for LMP
DRP:8TR	LSP:8R	8T
LMP:8T	DRP:8TR	8T
LMP:8T	LSP:8R	8T

9.4.6.2.3 Rapid Link Establishment

Each management adapter shall first determine the link establishment rate, and then decide whether the conditions for rapid link establishment are met based on the historical information saved locally. For all the lanes that are to be used in this link establishment, if there are historical parameters saved locally that meet all of the following conditions, it is considered that the conditions for rapid link establishment are met, so rapid link establishment can be enabled:

- The addresses of opposite device for different channels are identical.
- The port numbers used by the device and the opposite device are identical.
- The VID, PID, and SN in Cable Info are identical.
- The rates and directions of the lanes are identical.

If the management adapter of the device supports the rapid link establishment ability, it is required to obtain the parameters currently used by each lane of the main link from the logical layer and the electrical layer after each successful link establishment, and then save them locally together with all the information required in the decision-making conditions. In addition, it can issue the locally saved historical parameters of the lanes of the main link matching this link establishment to the logical layer and electrical layer before starting the rapid link establishment.

9.4.6.3 Exit Conditions

A port will exit the Link Configuration state when it is in any of the following conditions:

- A successful link configuration is achieved, and the port enters the Link Training state.
- The LMP determines that any error occurs in link training and notifies the LSP to enter the Error state, and the port enters the Error state after a successful handshake.
- The LSP receives a link configuration request message indicating entering the Error state and replies with a link configuration response message, and the port enters the Error state.
- A successful negotiation about entering the Standby State is achieved, and the port enters the Standby state.
- The port receives a port disabling instruction, and enters the Disabled state.

- The port detects disconnection, and enters the Initialized state.

9.4.7 Link Training

9.4.7.1 Entry Conditions

If a port is successfully configured in the Link Configuration state, it will exit the Link Configuration state and enter the Link Training state.

9.4.7.2 Action

When a port is in the Link Training state, the main link enters the Recovery state if rapid link establishment is used. Otherwise, it enters the Training state.

Each management adapter shall wait for the link training results from the logical layer, and it will remain in this waiting state until it receives the results, with the timeout of this maintained by the logical layer.

A port in the Link Training state may receive a new link configuration request message, and it will enter the Link Configuration state to handle this request message after the current training process is completed, regardless of whether the training succeeds or fails. If the main link is in the Recovery or Training state, new link parameters cannot be issued directly to the logical layer, so it is necessary to make sure that the current training process ends before jumping to the Link Configuration state. The management adapter can choose to passively wait for the logical layer to indicate the end of training. If it does not want to wait, it can also choose to actively disable and then enable the main link to forcefully interrupt the current training process. If the value of the "EE" field in this link configuration request message is 1, there is no need to wait for the training to end, and the device can directly jump to the Link Configuration state.

9.4.7.3 Exit Conditions

A port will exit the Link Training state when it is in any of the following conditions:

- If the link training succeeds, it will enter the Ready state.
- If the link training fails, it will enter the Link Configuration state.
- A successful negotiation about entering the Standby State is achieved, and the port enters the Standby state.
- The port receives a port disabling instruction, and enters the Disabled state.
- The port detects disconnection, and enters the Initialized state.

9.4.8 Ready State

9.4.8.1 Entry Conditions

When a port in the Link Training state receives an indication of successful link training from the logical layer, it will exit the Link Training state and enter the Ready state.

9.4.8.2 Action

When a port is in the Ready state and the main link is ready to work, the management adapter can choose to send a management adapter packet through the main link.

In the Ready state, when it is necessary to adjust the link mode, and if the method is to adjust the link width but not the lane rate, the port will maintain the Ready state and send a link configuration request message. When the LSP receives a link configuration request message without adjusting the lane rate in the Ready state, it shall also remain in the Ready state and send a link configuration response message.

9.4.8.3 Exit Conditions

A port will exit the Ready state when it is in any of the following conditions:

- When the management adapter of the LMP decides that the lane rate should be switched, the port will enter the Link Configuration state.
- When the LSP receives a link configuration request message that requires switching the lane rate, the port will enter the Link Configuration state.
- A successful negotiation about entering the Standby State is achieved, and the port enters the Standby state.
- The port receives a port disabling instruction, and enters the Disabled state.
- The port detects disconnection, and enters the Initialized state.

9.4.9 Standby State

9.4.9.1 Entry Conditions

When a device initiates to enter or exit the Standby state through a port standby request message, the other device shall reply with a port standby response message to indicate whether it agrees or rejects. The conditions for a port to enter or exit the Standby state include:

(1) When one device initiates an entry request, the other device must agree.

(2) When one device initiates an exit request, the other device will decide to refuse or agree based on its own device status. For example, the port shall reject the request if the device is in the Standby state. Otherwise, it can agree to the request.

After the initiator of a port standby entry request message receives a port standby response message indicating an agreement, or the receiver of a port standby entry request message sends a port standby response message indicating an agreement, the port will enter the Standby state.

When the timeout is longer than 200 ms and the negotiation still fails more than 5 times, the port of the device will directly enter the Standby state. For example, the opposite device fails to reply to the port standby entry request message after a timeout, or refuses to reply to the port standby entry request message.

A port in the Capability Negotiation, Link Configuration, Link Training, or Ready state may receive a port standby request message. In this case, the port shall stop the current process, that is to say, it stops sending any new request messages, and ceases waiting for any response messages corresponding to sent request messages. The port performs the port standby negotiation process as normal, and enters the Standby state after a successful port standby negotiation is achieved.

9.4.9.2 Action

When a port is in the Standby state, the main link cannot work, and the sideband link can normally send, receive, and process various types of management adapter packets.

A port in the Standby state may receive any port standby request message, and it shall reply the port standby response message as normal.

When a port in the Standby state receives a capability negotiation request message or link configuration request message, it shall discard the message directly with no processing operation.

9.4.9.3 Exit Conditions

After the initiator of a port standby exit request message receives a port standby response message indicating an agreement, or the receiver of a port standby exit request message sends a port standby response message indicating an agreement, the port will exit the Standby state and enter the Capability Negotiation state.

9.4.10 Error State

9.4.10.1 Entry Conditions

When one of the following situations occurs, one device of the link will notify the opposite device to enter the Error state. After a successful handshake, both devices will enter the Error state:

- In the Capability Negotiation state, if the negotiation fails and the role of the port can be decided, the LMP shall notify the opposite device to enter the Error state.
- In the Capability Negotiation state, if neither of the two devices can decide the role of its own port due to a capability conflict, each of them shall notify the other to enter the Error state.
- In the Link Configuration state, if the LMP fails to train with the same configuration for 3 consecutive times and there is no other expected link establishment mode, it will notify the opposite device to enter the Error state.

In any of the above cases, the initiator of entering the Error state notifies the opposite device through a link configuration request message, in which the value of the EE field is 1, and enters the Error state after it receives the corresponding link configuration response message. When the timeout is longer than 200 ms and it is still the case after retrying more than 5 times, the port will directly enter the Error state, as the opposite device may have replied to the response message and has entered the Error state then, while the opposite device's response message is not successfully received due to other reasons such as poor link quality. The receiver that has entered the Error state will reply the link configuration response message when it receives the link configuration request message, in which the value of the EE field is 1, and enter the Error state after a successful reply.

If both devices initiate requests to enter the Error state at the same time, they will enter the Error state after a successful handshake is achieved, and ignore the link configuration request messages sent before.

9.4.10.2 Action

When a port is in the Error state, the main link cannot work, the sideband link maintains SBTX high, and SBRX ignores any received messages and maintains the ability to detect the unplugging of the opposite device.

9.4.10.3 Exit Conditions

A port will exit the Error state when it is in any of the following conditions:

- The port receives a port disabling instruction, and enters the Disabled state.

- The port detects disconnection, and enters the Initialized state.

9.5 Bandwidth Management

9.5.1 Overview

Bandwidth management messages are mainly used to establish a high-speed channel between two adapters to transmit audio, video, and data. Bandwidth management messages mainly contain messages for bandwidth query, bandwidth application, bandwidth adjustment, bandwidth release, channel removal, port bandwidth query, bandwidth negotiation, and bandwidth negotiation notification. Messages can be transmitted via the sideband link or the main link. Bandwidth management messages are received and sent by management adapters.

The types of bandwidth management messages are as shown in Table 141:

Table 141 Types of bandwidth management messages

Message Type	Message Name
0x4	Bandwidth query
0x5	Bandwidth application
0x6	Bandwidth adjustment
0x7	Bandwidth release
0x8	Channel removal
0x9	Reserved
0xA	Reserved
0xB	Reserved
0xC	Reserved

9.5.2 Bandwidth Query (Channel = 0)

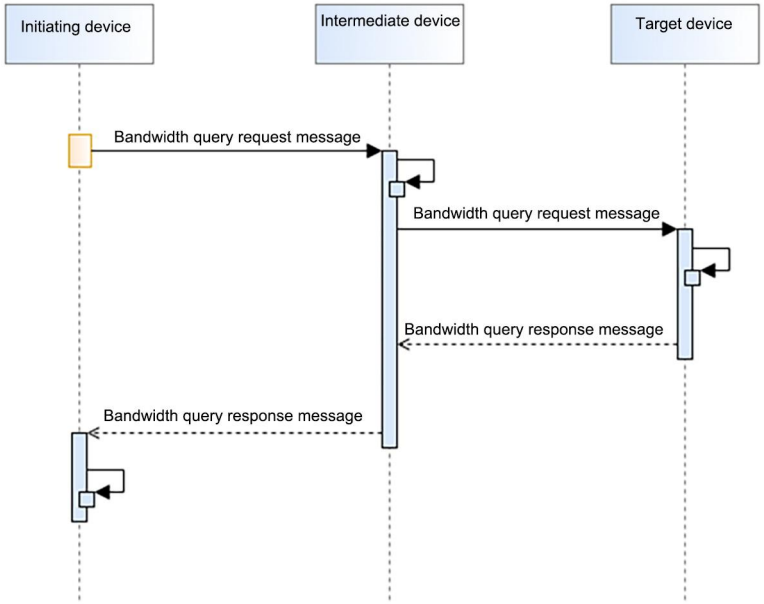
9.5.2.1 Introduction

Bandwidth query messages (Channel = 0) are used to query the minimum available bandwidth of each port on the channel between two devices in a GPMI network.

A bandwidth query message must be initiated by the source device of the corresponding stream. For example, for video stream, this message must be initiated by the device where the audio and video transmission adapter that sends the video is located.

The sequence of a bandwidth query is as shown in Table 192.

Figure 192 Sequence diagram of bandwidth query process



9.5.2.2 Processing Flow

9.5.2.2.1 Initiating Device

When a device generates a bandwidth query request message, the message will be processed as follows:

(a) Select the destination device to be queried and the corresponding channel according to the service needs.

(b) Assign 0 to Channel, assign the output stream bandwidth available at the output port to OutStreamBW, and assign 0xFFFFFFFF to InStreamBW, to generate a bandwidth query request message.

(c) Send a request message according to the forwarding list.

When a device receives a bandwidth query response message, the message will be processed as follows:

(a) Make a comprehensive decision according to the minimum bandwidth (OutStreamBW or InStreamBW) in the response message, the expected bandwidth, and the service requirements. The recommended decision-making strategy is as follows:

(1) When the minimum bandwidth is greater than or equal to the expected bandwidth, initiate the bandwidth application process.

(2) When the minimum bandwidth is smaller than the expected bandwidth:

- If the service is acceptable, for example, if it is possible to continue meeting the service needs by adjusting the resolution, color space, or bit depth, start the bandwidth application process.
- If the service is unacceptable, select another path to start the bandwidth query process again.

- If no path in the GPMI network can meet the bandwidth requirements of the current service, report an exception and let the application make a decision.

(b) For exception handling, refer to 9.2.3.3.

9.5.2.2.2 Intermediate Device

When a device receives a bandwidth query request message, the message will be processed as follows:

(a) Compare the available output stream bandwidth of the input port with InStreamBW in the message, and take the smaller value to update the InStreamBW field in the message.

(b) Extract the corresponding output port number OutputPortID from the forwarding list according to the value of FL Level.

(c) Compare the available output stream bandwidth of the output port with OutStreamBW in the message, and take the smaller value to update the OutStreamBW field in the message.

(d) Forward the request message according to the forwarding list.

(e) Exception handling:

(1) When a port receives a bandwidth query request message, if it is in the Link Configuration or Link Training state, it shall set ErrCode to 0x1 (Busy) and reply the response message directly.

(2) For the generation of other errors, refer to 9.2.3.3.

When a device receives a bandwidth query response message, it will directly forward the message according to the forwarding list with no processing operation.

9.5.2.2.3 Target Device

When a device receives a bandwidth query request message, the message will be processed as follows:

(a) Compare the available output stream bandwidth of the input port with InStreamBW in the message, and take the smaller value to update the InStreamBW field in the message.

(b) Determine whether the message comes from the destination device according to FL Level = FL Length + 1.

(c) Take the input port as the output port for the response message.

(d) Return a response message according to the forwarding list.

(e) Exception handling:

(1) When a port receives a bandwidth query request message, if it is in the Link Configuration or Link Training state, it shall set ErrCode to 0x1 (Busy) and reply the response message directly.

(2) For the generation of other errors, refer to 9.2.3.3.

Note: When the response message is returned in an exception situation error, it is necessary to assign 0 to PortID in the forwarding list corresponding to FL Level.

9.5.2.3 Message Description

The packet structure of bandwidth query request message is as shown in Figure 193, and the field description is as shown in Table 142.

Figure 193 Packet structure of bandwidth query request message

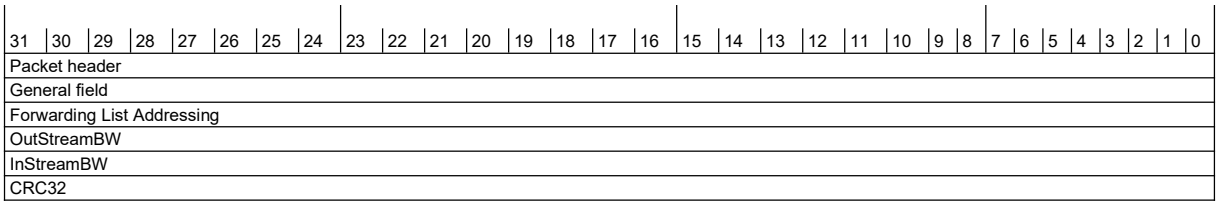


Table 142 Field description for packet of bandwidth query request message

Field Name	Length (Bits)	Description
Packet header	32	Refer to Section 9.2.1.1.
General field	32	Refer to Section 9.2.1.1.
Forwarding List Addressing	32	Refer to Section 9.2.2.
OutStreamBW (Out Stream BandWidth)	32	Indicates the output stream bandwidth, with a valid value range of [0, 0xFFFFFFFF). The initial value is 0xFFFFFFFF. For the update rules, refer to 9.5.2.2.
InStreamBW (In Stream BandWidth)	32	Indicates the input stream bandwidth, with a valid value range of [0, 0xFFFFFFFF). The initial value is 0xFFFFFFFF. For the update rules, refer to 9.5.2.2.
CRC32	32	Check code

The packet structure of the bandwidth query response message is as shown in Figure 194, and the field description is as shown in Table 143.

Figure 194 Packet structure of bandwidth query response message

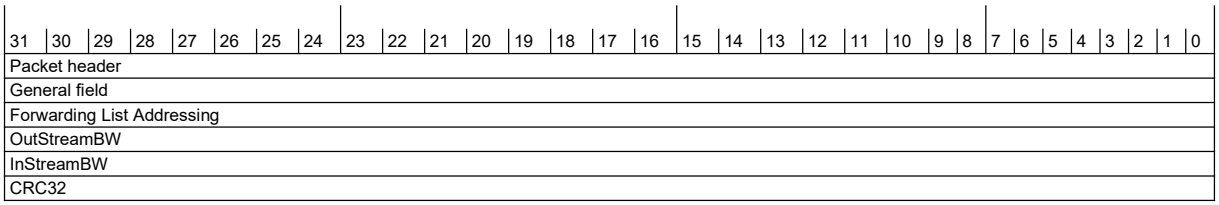


Table 143 Field description for packet of bandwidth query response message

Field Name	Length (Bits)	Description
Packet header	32	Refer to Section 9.2.1.1.
General field	32	Refer to Section 9.2.1.1.
Forwarding List Addressing	32	Refer to Section 9.2.2.
OutStreamBW (Out Stream BandWidth)	32	Indicates the minimum output stream bandwidth available to each port on the channel, with a valid value range of [0, 0xFFFFFFFF).
InStreamBW (In Stream BandWidth)	32	Indicates the minimum input stream bandwidth available to each port on the channel, with a valid value range of [0, 0xFFFFFFFF).
CRC32	32	Check code

9.5.3 Bandwidth Query (Channel = 1)

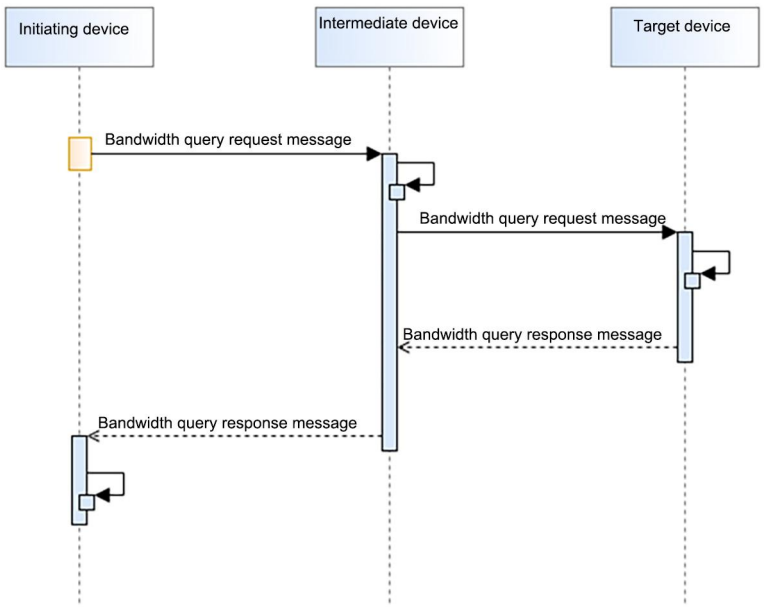
9.5.3.1 Introduction

Bandwidth query messages are used to query the available bandwidth of each port on the channel between two devices in a GPMI network.

A bandwidth query message must be initiated by the source device of the corresponding stream. For example, for video stream, this message must be initiated by the device where the audio and video transmission adapter that sends the video is located.

The sequence of a bandwidth query is as shown in Figure 195.

Figure 195 Sequence diagram of bandwidth query process



9.5.3.2 Processing Flow

9.5.3.2.1 Initiating Device

When a device generates a bandwidth query request message, the message will be processed as follows:

(a) Select the destination device to be queried and the corresponding channel according to the service needs.

(b) Assign 1 to Channel, assign the output stream bandwidth available at the output port to L0_OutStreamBW, and assign 0xFFFFFFFF (n = FL Length) to L1~n_OutStreamBW and L0~n_InStreamBW, to generate a bandwidth query request message.

(c) Send a request message according to the forwarding list.

When a device receives a bandwidth query response message, the message will be processed as follows:

(a) Make a comprehensive decision according to the available bandwidth (Ln_OutStreamBW or Ln_InStreamBW) of the Level n port in the response message, the expected bandwidth, and the service requirements. The recommended decision-making strategy is as follows:

(1) When the available bandwidth is greater than or equal to the expected bandwidth, start the bandwidth application process.

(2) When the available bandwidth is smaller than the expected bandwidth:

- If the service is acceptable, for example, if it is possible to continue meeting the service needs by adjusting the resolution, color space, or bit depth, start the bandwidth application process.
- If the service is unacceptable, select another path to start the bandwidth query process again.

- If no path in the GPMI network can meet the bandwidth requirements of the current service, report an exception and let the application make a decision.

(b) For exception handling, refer to 9.2.3.3.

9.5.3.2.2 Intermediate Device

When a device receives a bandwidth query request message, the message shall be processed as follows:

(a) Update the available output stream bandwidth of the input port to the Ln-1_InStreamBW field in the message.

(b) Extract the corresponding output port number Ln_PortID from the forwarding list according to the value of FL Level.

(c) Update the available output stream bandwidth of the output port to the Ln_OutStreamBW field in the message.

(d) Forward the request message according to the forwarding list.

(e) Exception handling:

(1) When a port receives a bandwidth query request message, if it is in the Link Configuration or Link Training state, it shall set ErrCode to 0x1 (Busy) and reply the response message directly.

(2) For the generation of other errors, refer to 9.2.3.3.

When a device receives a bandwidth query response message, it will directly forward the message according to the forwarding list with no processing operation.

9.5.3.2.3 Target Device

When a device receives a bandwidth query request message, the message will be processed as follows:

(a) Update the available output stream bandwidth of the input port to the Ln_InStreamBW field in the message.

(b) Determine whether the message comes from the destination device according to FL Level = FL Length + 1.

(c) Take the input port as the output port for the response message.

(d) Return a response message according to the forwarding list.

(e) Exception handling:

(1) When a port receives a bandwidth query request message, if it is in the Link Configuration or Link Training state, it shall set ErrCode to 0x1 (Busy) and reply the response message directly.

(2) For the generation of other errors, refer to 9.2.3.3.

Note: When the response message is returned in an exception situation error, it is necessary to assign 0 to PortID in the forwarding list corresponding to FL Level.

9.5.3.3 Message Description

The packet structure of the bandwidth query request message is as shown in Figure 196, and the field description is as shown in Table 144.

Figure 196 Packet structure of bandwidth query request message

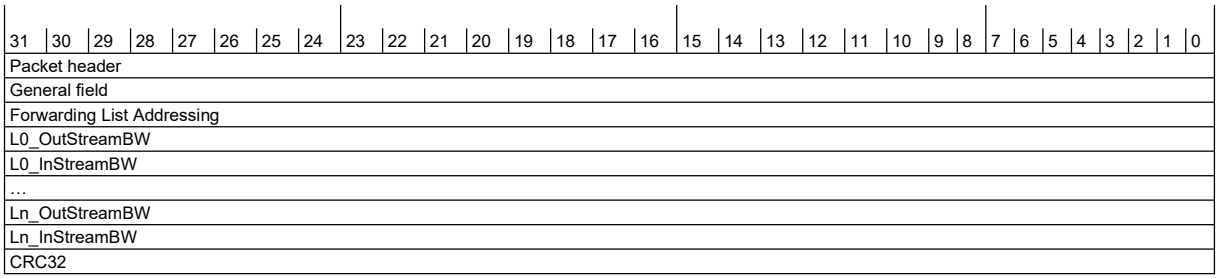


Table 144 Field description for packet of bandwidth query request message

Field Name	Length (Bits)	Description
Packet header	32	Refer to Section 9.2.1.1.
General field	32	Refer to Section 9.2.1.1.
Forwarding List Addressing	32	Refer to Section 9.2.2.
L0_OutStreamBW (Level 0 Out Stream BandWidth)	32	Indicates the output stream bandwidth of level n and the input stream bandwidth of level n, in Mbps. The value of n is FL Length, which means that there are FL Length + 1 pairs of Ln_OutStreamBW and Ln_InStreamBW. When n = 0, L0_OutStreamBW and L0_InStreamBW represent the output stream and input stream bandwidths of the output port of the initiating device respectively. The valid value range of Ln_OutStreamBW and Ln_InStreamBW is [0, 0xFFFFFFFF), and the default value is 0xFFFFFFFF. For the update rules of Ln_OutStreamBW and Ln_InStreamBW, refer to 9.5.3.2.
L0_InStreamBW (Level 0 In Stream BandWidth)	32	
…	32	
Ln_OutStreamBW (Level n Out Stream BandWidth)	32	
Ln_InStreamBW (Level n In Stream BandWidth)	32	
CRC32	32	Check code

The packet structure of the bandwidth query response message is as shown in Figure 197, and the field description is as shown in Table 145.

Figure 197 Packet structure of bandwidth query response message

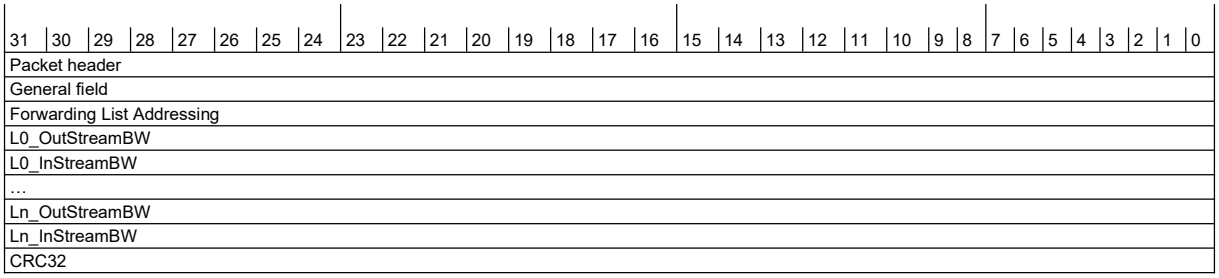


Table 145 Field description for packet of bandwidth query response message

Field Name	Length (Bits)	Description
Packet header	32	Refer to Section 9.2.1.1.
General field	32	Refer to Section 9.2.1.1.
Forwarding List Addressing	32	Refer to Section 9.2.2.
L0_OutStreamBW (Level 0 Out Stream BandWidth)	32	Indicates the available output stream bandwidth and available input stream bandwidth of the level n port on the channel, in Mbps, with a valid value range of [0, 0xFFFFFFFF).
L0_InStreamBW (Level 0 In Stream BandWidth)	32	
…	32	
Ln_OutStreamBW (Level n Out Stream BandWidth)	32	
Ln_InStreamBW (Level n In Stream BandWidth)	32	
CRC32	32	Check code

9.5.4 Bandwidth Application

9.5.4.1 Introduction

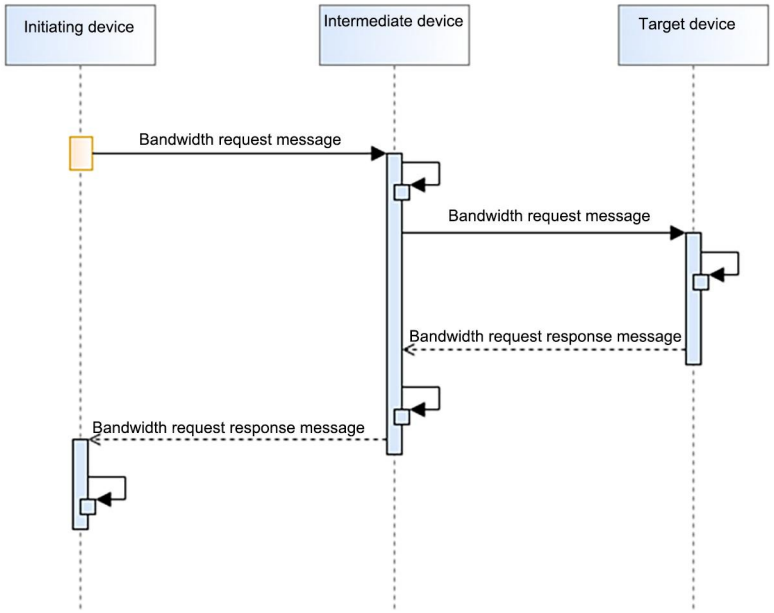
Bandwidth application messages are used to apply for and allocate bandwidth for each device port on the service channel, to establish the service channel.

For audio and video services, bandwidth application messages must be initiated by the source device of the stream (the device where the audio and video transmission adapter is). When a source device initiates a bandwidth application request, it must select an appropriate path based on the topology of the GPMI network. It is recommended to select according to the path length, and the shortest path is preferred.

Every device must record the information of all the service streams passing through the device, including input port number, source device address, source adapter ID, output port number, destination device address, destination adapter ID, bandwidth, and priority.

The bandwidth application process is as shown in Figure 198.

Figure 198 Sequence diagram of bandwidth application process



9.5.4.2 Processing Flow

9.5.4.2.1 Initiating Device

When a device generates a bandwidth application request message, the message will be processed as follows:

(a) Select the destination device and the corresponding channel as needed, taking one-way video service as an example.

(b) Assign the output stream bandwidth required by the service to OutStreamBW in the message, and assign 0xFFFFFFFF to InStreamBW. (Note: 0xFFFFFFFF indicates that it is not required to allocate bandwidth to the corresponding input stream direction, or to establish a service logical lane in the corresponding direction.)

(c) Update the information including source device address, source adapter ID, destination device address, destination adapter ID, service priority, and flow control mechanism to the message.

(1) The service priority is consistent with the priority of the corresponding service adapter. For details, refer to the adapter priority list.

(2) For the selection of flow control mechanism, refer to the description in "Explanation of Terms".

(3) For the calculation and allocation of the flow control cache, refer to the description of the transport layer.

(d) Query the output port number OutPortID according to the forwarding list, and check whether the output port number OutPortID exists in the forwarding list of the routing table according to the routing table.

(1) If it exists, use the existing ShuttleID and increase ReceiverCount by 1. If it is not required to allocate bandwidth for the node, directly go to step (f).

(2) Otherwise, assign a new ShuttleID to the output port OutPort_ID and set ReceiverCount to 1.

(e) Determine whether the current actual available bandwidth is greater than or equal to the output stream bandwidth (OutStreamBW) required by the service. If so, allocate an output stream bandwidth for the shuttle corresponding to ShuttleID. Otherwise, start the bandwidth increase management process.

(f) Record the information of the service stream, and update the information of the routing table.

(g) Compare the actually allocated bandwidth with OutStreamBW. If the actually allocated bandwidth is smaller than the value of OutStreamBW, assign the actually allocated bandwidth to OutStreamBW; otherwise, OutStreamBW remains unchanged.

(h) Issue the information in Table 146 to the transport layer.

Table 146 Information sent from bandwidth application initiating device to transport layer

Management Adapter	Transport Layer
Flow control mechanism	FC_STSH
Priority	Priority
Flow control cache	Credit Allocated Shuttle
Weight	Weight
Input ShuttleID	ShuttleID
Output port number	FwPort
Output ShuttleID	FwShuttleID

(i) Send a bandwidth application request message according to the forwarding list.

When a device receives a bandwidth application response message, the message will be processed as follows:

(a) Identify the path according to the CHShuttleID and Tag fields in the bandwidth application response message.

(b) Determine whether ReceiverCount is greater than 1. If it is greater than 1, go to step (e). Otherwise, go to step (c).

(c) Compare the bandwidth (OutStreamBW or InStreamBW) in the bandwidth application response message with the allocated bandwidth. If it is smaller than the allocated bandwidth, release the excess bandwidth, recalculate the weight and flow control cache, and issue the updated weight and flow control cache to the transport layer.

(d) Refresh the record of allocated bandwidth.

(e) Make a decision on the next action based on the obtained bandwidth, and the decision-making strategy is as follows:

(1) If the bandwidth in the response message is identical to the value that the service applies for, start the service transmission.

(2) If the bandwidth in the response message is smaller than the value that the service applies for and the service is acceptable, for example, if the service requirements can continue to be met by adjusting the resolution, color space, or bit depth, start the service transmission.

(3) If the bandwidth in the response message is smaller than the value that the service applies for, but the service is unacceptable, initiate the bandwidth release process, select the second shortest path, and then start the bandwidth application process.

(4) If no path in the GPMI network can meet the bandwidth requirements of the current service, report an exception to the application.

(f) Exception handling

(1) If ErrCode is not zero when the bandwidth response message is received, resend the bandwidth application request message based on the original Tag.

(2) If ErrCode is not zero when the bandwidth response message is received, first invoke the bandwidth release process, and then use a new Tag to send the bandwidth application request message.

(3) For exception handling, refer to 9.2.3.3.

9.5.4.2.2 Intermediate Device

When a device receives a bandwidth application request message, the message will be processed as follows:

(a) Receive a bandwidth application request message and record the service stream information.

(b) Query the output port number OutPortID according to the forwarding list, and check whether the output port number OutPortID exists in the forwarding list of the routing table according to the routing table.

(1) If it exists, use the existing ShuttleID and increase ReceiverCount by 1. If it is not required to allocate bandwidth for the node, directly go to step (d).

(2) Otherwise, assign a ShuttleID to the output port and set ReceiverCount to 1.

(c) Determine whether the current actual available bandwidth is greater than or equal to the output stream bandwidth (OutStreamBW) required by the service. If so, allocate an output stream bandwidth for the shuttle corresponding to ShuttleID. Otherwise, start the bandwidth increase management process.

(d) Record the information of the service stream, and update the information of the routing table.

(e) Compare the actually allocated bandwidth with OutStreamBW. If the actually allocated bandwidth is smaller than the value of OutStreamBW, assign the actually allocated bandwidth to OutStreamBW; otherwise, OutStreamBW remains unchanged.

(f) Issue the information in Table 146 to the transport layer.

(g) Send a bandwidth application request message according to the forwarding list.

(h) Exception handling

(1) If a bandwidth application message with repeated tags from the same source device is received and the bandwidth has been allocated, forward the message directly with no processing operation.

(2) For other exceptions, refer to 9.2.3.3.

When a device receives a bandwidth application response message, the message will be processed as follows:

(a) Identify the path according to the CHShuttleID and Tag fields in the bandwidth application response message.

(b) First determine whether ReceiverCount is greater than 1. If it is greater than 1, go to step 4. Otherwise, directly go to step 3.

(c) Compare the bandwidth (OutStreamBW or InStreamBW) in the bandwidth application response message with the allocated bandwidth. If it is smaller than the allocated bandwidth, release the excess bandwidth, recalculate the weight and flow control cache, and issue the updated weight and flow control cache to the transport layer.

(d) Refresh the record of allocated bandwidth.

(e) Forward the bandwidth application response message according to the forwarding list.

(f) Exception handling

(1) If a bandwidth application response message with repeated tags from the same source device is received and the bandwidth has been allocated, forward the message directly with no processing operation.

(2) For other exceptions, refer to 9.2.3.3.

9.5.4.2.3 Target Device

When a device receives a bandwidth application request message, the message will be processed as follows:

(a) Receive a bandwidth application request message and record the service stream information.

(b) Determine the destination device according to the forwarding list.

(c) Take the input port as the output port for the response message.

(d) Return a response message according to the forwarding list.

(e) Exception handling:

(1) If a bandwidth application response message with repeated tags from the same source device is received and the bandwidth has been allocated, forward the message directly with no processing operation.

(2) For other exceptions, refer to 9.2.3.3.

Notes: When it is necessary to apply for both the output and input stream bandwidths for the service, the intermediate device shall not only allocate the output stream bandwidth to the output port, but also allocate the output stream bandwidth to the input port, corresponding to InStreamBW in the bandwidth application message. The destination device shall allocate the output stream bandwidth to the input port, corresponding to InStreamBW in the bandwidth application message. The rules and procedures for bandwidth allocation are the same as above.

9.5.4.3 Message Description

The packet structure of the bandwidth application request message is as shown in Figure 199, and the field description is as shown in Table 147.

Figure 199 Packet structure of bandwidth application request message

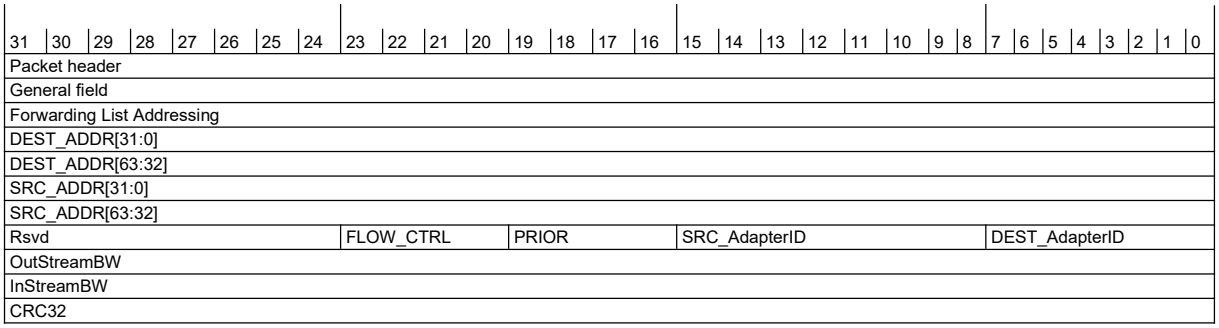


Table 147 Field description for packet of bandwidth query request message

Field Name	Length (Bits)	Description
Packet header	32	Refer to Section 9.2.1.1.
General field	32	Refer to Section 9.2.1.1.
Forwarding List Addressing	32	Refer to Section 9.2.2.
DEST_ADDR (Destination Address)	32	The destination device address is 32 bits low.
DEST_ADDR (Destination Address)	32	The destination device address is 32 bits high.
SRC_ADDR (Source Address)	32	The source device address is 32 bits low.
SRC_ADDR (Source Address)	32	The source device address is 32 bits high.
DEST_AdapterID (Destination Adapter)	8	Destination adapter ID.
SRC_AdapterID (Source Adapter Identifier)	8	Source adapter ID.
PRIOR (Priority)	4	Priority.
FLOW_CTRL	4	Flow control mechanism.
Rsvd	8	Reserved.
OutStreamBW (Output Stream BandWidth)	32	Output stream bandwidth, in Mbps: 0: indicates that the obtained output stream bandwidth is 0, and a service logical lane in the output stream direction should be established. 0xFFFFFFFF: indicates that no output

Field Name	Length (Bits)	Description
		stream bandwidth should be allocated, nor service channel in the output stream direction should be established. 0x0–0xEFFFFFFF: indicates that the output stream bandwidth should be applied for, and in addition, a service channel in the output stream direction should be established.
InStreamBW (Input Stream BandWidth)	32	Input stream bandwidth, in Mbps: 0: indicates that the obtained input stream bandwidth is 0, and a service logical lane in the input stream direction should be established. 0xFFFFFFFF: indicates that no input stream bandwidth should be allocated, nor service channel in the input stream direction should be established. 0x0–0xEFFFFFFF: indicates that the input stream bandwidth should be applied for, and in addition, a service channel in the input stream direction should be established.
CRC32	32	Check code

The packet structure of the bandwidth application response message is as shown in Figure 200, and the field description is as shown in Table 148.

Figure 200 Packet structure of bandwidth application response message

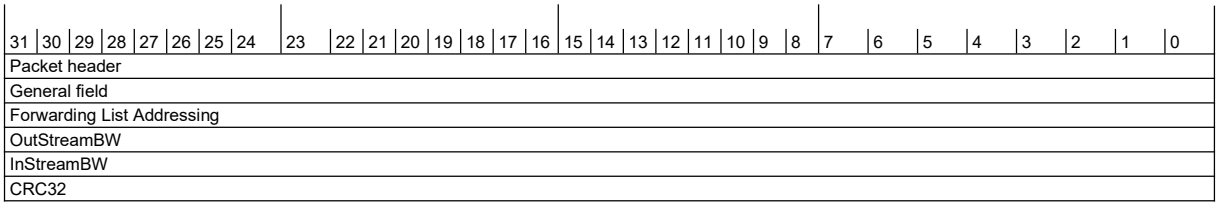


Table 148 Field description for packet of bandwidth application response message

Field Name	Length (Bits)	Description
Packet header	32	Refer to 9.2.1.1.
General field	32	Refer to 9.2.1.1.
Forwarding List Addressing	32	Refer to 9.2.2.
OutStreamBW	32	Output stream bandwidth, in Mbps:

Field Name	Length (Bits)	Description
(Output Stream BandWidth)		0: indicates that the obtained output stream bandwidth is 0. 0xFFFFFFFF: indicates that no output stream bandwidth should be allocated. 0x0–0xEFFFFFFF: indicates that the obtained output stream bandwidth is required.
InStreamBW (Input Stream BandWidth)	32	Input stream bandwidth, in Mbps: 0: indicates that the obtained input stream bandwidth is 0. 0xFFFFFFFF: indicates that no input stream bandwidth should be allocated. 0x0–0xEFFFFFFF: indicates that the obtained input stream bandwidth is required.
CRC32	32	Check code

9.5.5 Bandwidth Adjustment

9.5.5.1 Introduction

Bandwidth adjustment messages are used to adjust the bandwidth of each device on the service channel. Both upward (bandwidth increase) and downward (bandwidth decrease) adjustments are supported, for full bandwidth utilization and low power consumption.

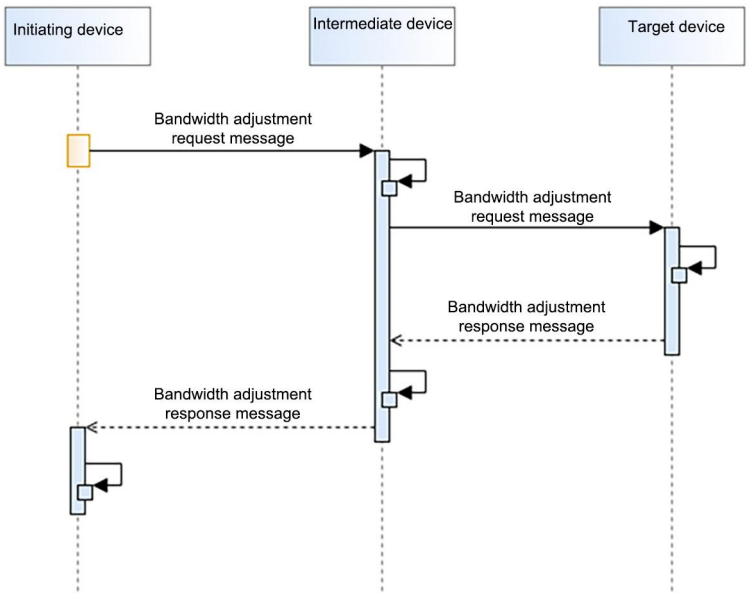
When the service changes and the demand for bandwidth increases, a bandwidth adjustment message (upward adjustment) can be initiated. When the demand for bandwidth decreases, a bandwidth adjustment message (downward adjustment) can also be initiated. The released bandwidth allows some main link lanes to enter the Low Power Consumption state or be used by other services.

Upward adjustment is the process to increase the bandwidth in the direction of the bandwidth adjustment request message, which is similar to bandwidth application, and downward adjustment is the process to decrease the bandwidth in the direction of the bandwidth adjustment response message.

A bandwidth adjustment message must be initiated by the source device of the corresponding stream. For example, for video stream, this message must be initiated by the device where the audio and video transmission adapter that sends the video is located.

The bandwidth adjustment process is as shown in Figure 201.

Figure 201 Sequence diagram of bandwidth adjustment process



9.5.5.2 Processing Flow

9.5.5.2.1 Initiating Device

When a device generates a bandwidth adjustment request message, the message will be processed as follows:

(a) Determine the target bandwidth to be adjusted according to the change of the service, and generate a bandwidth adjustment request message, taking one-way video service as an example.

(b) Assign the target bandwidth to OutSteamBW in the message, and assign 0xFFFFFFFF to InSteamBW.

(c) Determine whether upward adjustment or downward adjustment is needed according to the target bandwidth in the message. If the target bandwidth is larger than the bandwidth of the current path, upward adjustment is needed. If the target bandwidth is smaller than the bandwidth of the current path, downward adjustment is needed. If the target bandwidth is equal to the bandwidth of the current path, no adjustment is needed.

(d) If upward adjustment is needed, go to step (e). Otherwise, go to step (h).

(e) Determine whether the current available bandwidth is greater than the target bandwidth. If so, allocate the bandwidth according to the target bandwidth. If not, start the bandwidth increase management process. If the available bandwidth of the bandwidth management cannot be adjusted to the target value, return directly.

(f) Refresh the service stream information.

(g) Issue the following information to the transport layer.

Table 149 Information sent from bandwidth application initiating device to transport layer

Management Adapter	Transport Layer

Management Adapter	Transport Layer
Flow control cache	Credit Allocated Shuttle
Weight	Weight

(h) Send a bandwidth application request message according to the forwarding list.

When a device receives a bandwidth adjustment response message, the message will be processed as follows:

(a) Identify the path according to the CHShuttleID and Tag fields in the bandwidth adjustment response message.

(b) Determine whether an upward adjustment or a downward adjustment is needed according to the target bandwidth in the bandwidth adjustment response message. If downward adjustment is needed, go to step (c). Otherwise, go to step (e).

(c) Release more bandwidths and recalculate the flow control cache and weights based on the adjusted bandwidth.

(d) Issue the information in Table 149 to the transport layer, and wait for the transport layer to release the flow control cache.

(e) Make a decision to start service switching or enter low power consumption based on the adjusted bandwidth.

(f) For exception handling, refer to 9.2.3.3.

9.5.5.2.2 Intermediate Device

When a device receives a bandwidth adjustment request message, the message will be processed as follows:

(a) Compare the target bandwidth in the message with the bandwidth allocated to the current path, to determine whether upward adjustment or downward adjustment is needed. If upward adjustment is needed, go to step (b). Otherwise, directly go to step (e).

(b) Determine whether the current available bandwidth meets the requirements. If so, allocate it as required. If not, start the bandwidth increase management process.

(c) Refresh the service stream information.

(d) Recalculate the process cache and weights, and issue the information in Table 149 to the transport layer.

(e) Send a bandwidth application request message according to the forwarding list.

(f) Exception handling

(1) If the available bandwidth of the bandwidth management cannot be adjusted to the target value, return a response message that directly carries ErrCode as 0x6 (insufficient resources).

(2) For other exception handling, refer to 9.2.3.3.

When a device receives a bandwidth adjustment response message, the message will be processed as follows:

(a) Identify the path according to the CHShuttleID and Tag fields in the bandwidth adjustment response message.

(b) Determine whether upward adjustment or downward adjustment is needed according to the bandwidth in the adjustment response message. If downward adjustment is needed, go to step (c). Otherwise, go to step (e).

(c) Recalculate the process cache and weights, and issue the information in Table 149 to the transport layer. Wait for the transport layer to release the flow control cache, and then release the excess bandwidth.

(d) Refresh and record the actual bandwidth after adjustment.

(e) Forward the bandwidth application request message according to the forwarding list.

(f) For exception handling, refer to 9.2.3.3.

9.5.5.2.3 Target Device

When a device receives a bandwidth adjustment message, the message will be processed as follows:

(a) Determine the destination device according to the forwarding list.

(b) Take the input port as the output port for the response message.

(c) Return a response message according to the forwarding list.

(d) Exception handling

(1) If the available bandwidth of the bandwidth management cannot be adjusted to the target value, return a response message that directly carries ErrCode as 0x6 (insufficient resources).

(2) For other exception handling, refer to 9.2.3.3.

Note: When it is necessary to adjust both the output and input stream bandwidths for the service, the intermediate device shall not only adjust the output stream bandwidth to the output port, but also adjust the output stream bandwidth to the input port, corresponding to InStreamBW in the bandwidth adjustment message. The destination device shall adjust the output stream bandwidth to the input port, corresponding to InStreamBW in the bandwidth adjustment message. The rules and procedures for bandwidth adjustment are the same as above.

9.5.5.3 Message Description

The packet structure of the bandwidth adjustment request message is as shown in Figure 202, and the field description is as shown in Table 150.

Figure 202 Packet structure of bandwidth adjustment request message

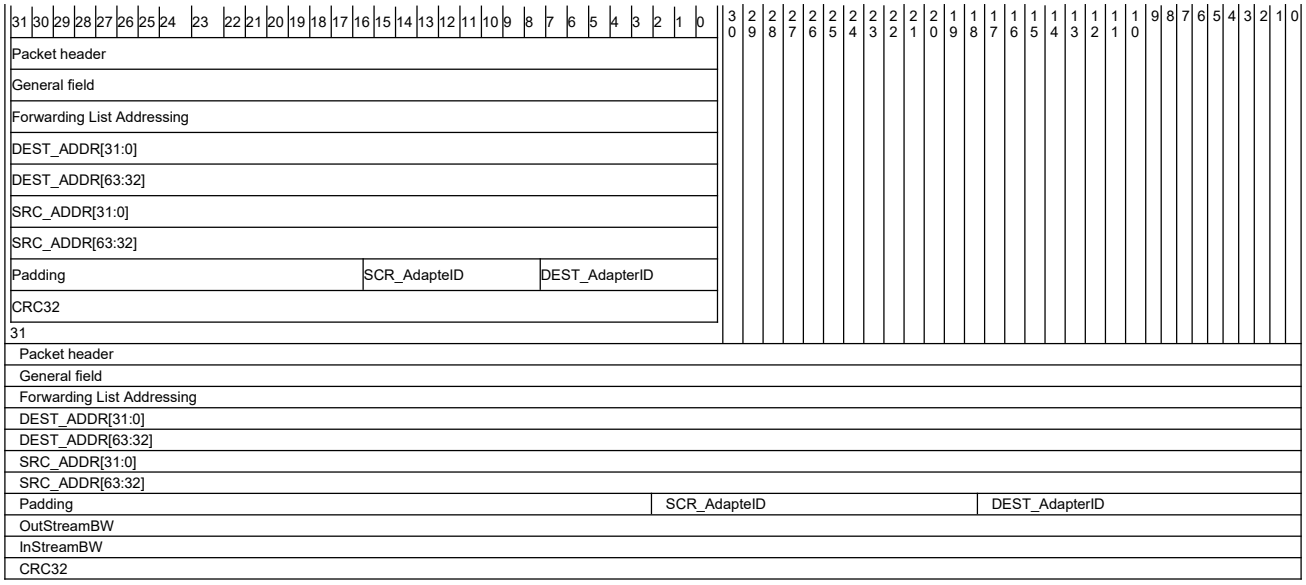


Table 150 Field description for packet of bandwidth query request message

Field Name	Length (Bits)	Description
Packet header	32	Refer to Section 9.2.1.1.
General field	32	Refer to Section 9.2.1.1.
Forwarding List Addressing	32	Refer to Section 9.2.2.
DEST_ADDR (Destination Address)	32	The destination device address is 32 bits low.
DEST_ADDR (Destination Address)	32	The destination device address is 32 bits high.
SRC_ADDR (Source Address)	32	The source device address is 32 bits low.
SRC_ADDR (Source Address)	32	The source device address is 32 bits high.
DEST_AdapterID (Destination Adapter)	8	Destination adapter ID.
SRC_AdapterID (Source Adapter Identifier)	8	Source adapter ID.
Padding	4	Filled with 0 to ensure 4-byte alignment
OutStreamBW (Out Stream BandWidth)	32	Adjust the target value of the output stream bandwidth, in Mbps: Valid value range [0, 0xFFFFFFFF).
InStreamBW	32	Adjust the target value of the input

Field Name	Length (Bits)	Description
(In Stream BandWidth)		stream bandwidth, in Mbps: Valid value range [0, 0xFFFFFFFF).
CRC32	32	Check code

The packet structure of the bandwidth adjustment response message is as shown in Figure 203, and the field description is as shown in Table 151.

Figure 203 Packet structure of bandwidth adjustment response message

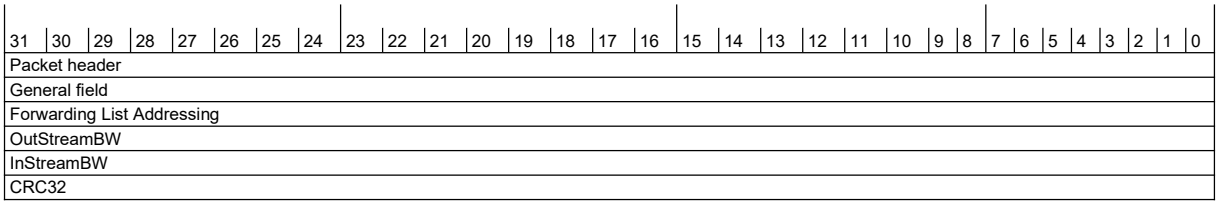


Table 151 Field description for packet of bandwidth query response message

Field Name	Length (Bits)	Description
Packet header	32	Refer to Section 9.2.1.1.
General field	32	Refer to Section 9.2.1.1.
Forwarding List Addressing	32	Refer to Section 9.2.2.
OutStreamBW (Out Stream BandWidth)	32	Adjust the target value of the output stream bandwidth, in Mbps: Valid value range [0, 0xFFFFFFFF).
InStreamBW (In Stream BandWidth)	32	Adjust the target value of the input stream bandwidth, in Mbps: Valid value range [0, 0xFFFFFFFF).
CRC32	32	Check code

9.5.6 Bandwidth Release

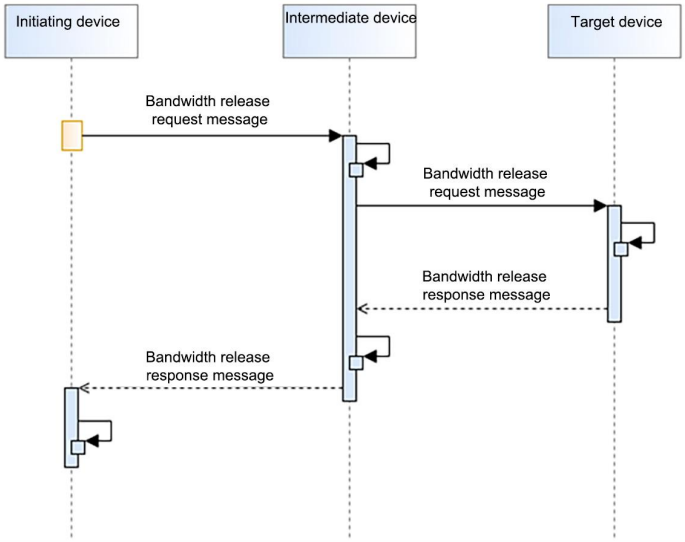
9.5.6.1 Introduction

Bandwidth release messages are used to release the bandwidth of each device port on the service channel. After the bandwidths of all the device ports on the service channel are released, the corresponding service channel will also be destroyed.

A bandwidth release message must be initiated by the source device of the corresponding stream. For example, for video stream, this message must be initiated by the device where the audio and video transmission adapter that sends the video is located.

The bandwidth release process is as shown in Figure 204.

Figure 204 Sequence diagram of bandwidth release process



9.5.6.2 Processing Flow

9.5.6.2.1 Initiating Device

When a device generates a bandwidth release request message, the message will be processed as follows:

(a) Generate the bandwidth release request message according to service changes. For example, when the service is stopped, or it is detected that a device on an established service channel in the GPMI network is lost, taking one-way video service as an example.

(b) Find the output port number OutPortID according to the forwarding list, look up the routing table according to the ShuttleID of the service stream, and determine whether the output port number exists in the forwarding list of the flow. If it exists, reduce the ReceiverCount of the output stream node by one, and set the FwVld field corresponding to the forwarding entry corresponding to the node to 0 to make it invalid. If the FwVld fields corresponding to all the forwarding entries corresponding to this node are 0, the Vld field of the service stream shall be set to 0 to make it invalid.

(c) Issue the following information to the transport layer.

Table 152 Information sent to transport layer for bandwidth release

Management Adapter	Transport Layer
Forward entry FwVld	FwVld
Service stream Vld	Vld

(d) Send a request message according to the forwarding list.

When a device receives a bandwidth release response message, the message will be processed as follows:

(a) Identify the path according to the CHShuttleID and Tag fields in the bandwidth adjustment response message.

(b) Determine whether ReceiverCount is 0. If it is 0, go to step (c). Otherwise, go to step (d).

(c) Start to release the flow control cache of the transport layer, and after it is completed, release the bandwidth.

(d) Clear the adapter unbinding information to remove the corresponding shuttle.

(e) For exception handling, refer to 9.2.3.3.

9.5.6.2.2 Intermediate Device

When a device receives a bandwidth release request message, the message will be processed as follows:

(a) Find the output port number OutPortID according to the forwarding list, look up the routing table according to the ShuttleID of the service stream, and determine whether the output port number exists in the forwarding list of the flow. If it exists, reduce the ReceiverCount of the output stream node by one, and set the FwVld field corresponding to the forwarding entry corresponding to the node to 0 to make it invalid. If the FwVld fields corresponding to all the forwarding entries corresponding to this node are 0, the Vld field of the service stream shall be set to 0 to make it invalid.

(b) Issue the information in Table 152 to the transport layer.

(c) Forward the bandwidth release request message according to the forwarding list.

(d) For exception handling, refer to 9.2.3.3.

When a device receives a bandwidth release response message, the message will be processed as follows:

(a) When a device receives a bandwidth release response message, it shall first determine whether ReceiverCount is 0. If it is 0, go to step (b). Otherwise, go to step (c).

(b) Start to release the flow control cache of the transport layer, and after it is completed, release the bandwidth.

(c) For exception handling, refer to 9.2.3.3.

9.5.6.2.3 Target Device

When a device receives a bandwidth release request message, the message will be processed as follows:

(a) Find the output port number OutPortID according to the forwarding list, look up the routing table according to the ShuttleID of the service stream, and determine whether the output port number exists in the forwarding list of the flow. If it exists, reduce the ReceiverCount of the output stream node by one, and set the FwVld field corresponding to the forwarding entry corresponding to the node to 0 to make it invalid. If the FwVld fields corresponding to all the forwarding entries corresponding to this node are 0, the Vld field of the service stream shall be set to 0 to make it invalid.

(b) Issue the information in Table 152 to the transport layer.

(c) Complete the refresh of the adapter unbinding information.

(d) Generate a bandwidth release response message and return the response message according to the forwarding list.

(e) For exception handling, refer to 9.2.3.3.

9.5.6.3 Message Description

The packet structure of the bandwidth release request message is as shown in Figure 205, and the field description is as shown in Table 153.

Figure 205 Packet structure of bandwidth release request message

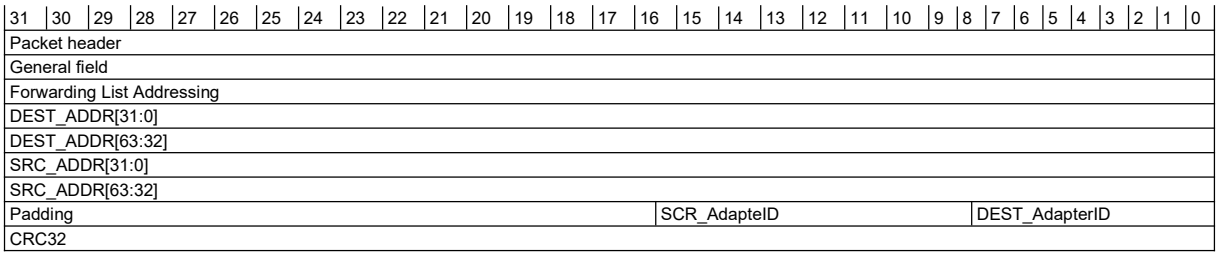


Table 153 Field description for packet of bandwidth release request message

Field Name	Length (Bits)	Description
Packet header	32	Refer to Section 9.2.1.1.
General field	32	Refer to Section 9.2.1.1.
Forwarding List Addressing	32	Refer to Section 9.2.2.
DEST_ADDR (Destination Address)	32	The destination device address is 32 bits low.
DEST_ADDR (Destination Address)	32	The destination device address is 32 bits high.
SRC_ADDR (Source Address)	32	The source device address is 32 bits low.
SRC_ADDR (Source Address)	32	The source device address is 32 bits high.
DEST_AdapterID (Destination Adapter)	8	Destination adapter ID.
SRC_AdapterID (Source Adapter Identifier)	8	Source adapter ID.
Padding	4	Filled with 0 to ensure 4-byte alignment
CRC32	32	Check code

The packet structure of the bandwidth release response message is as shown in Figure 206, and the field description is as shown in Table 154.

Figure 206 Packet structure of bandwidth release response message

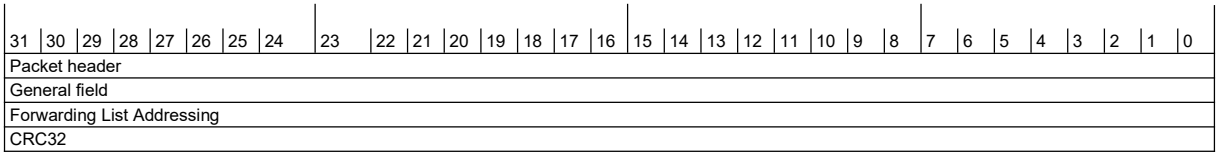


Table 154 Field description for packet of bandwidth release response message

Field Name	Length (Bits)	Description
Packet header	32	Refer to Section 9.2.1.1.
General field	32	Refer to Section 9.2.1.1.
Forwarding List Addressing	32	Refer to Section 9.2.2.
CRC32	32	Check code

9.5.7 Channel Removal

9.5.7.1 Introduction

Channel removal messages are used to remove service channels, release bandwidths, and destroy service channels. The scenario where a channel removal message will be generated: When a port is unplugged, the management adapter will generate a channel removal message for each forward stream passing through the port.

Note: A forward stream refers to a stream with which the port receives a bandwidth application request message.

The "channel removal process" initiated by the source device of the corresponding stream is the bandwidth release process.

9.5.7.2 Processing Flow

9.5.7.2.1 Source Device Port Unplugged

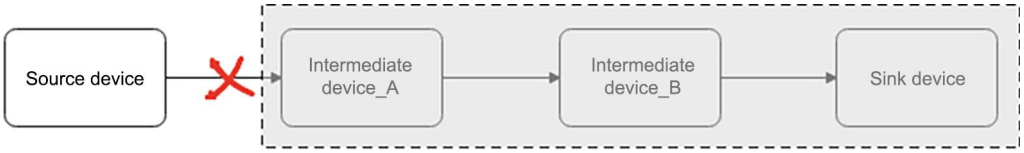
As shown in Figure 207, if the source device port is unplugged, it will be handled as follows (The gray part in the figure will be handled according to the process when an intermediate device port is unplugged):

(a) Receive a port unplugging event.

(b) Set the valid bit of the service stream (Vld) and the valid bit of the forwarded entry (FwVld) corresponding to the ShuttleID output passing through this port to 0, and issue them to the transport layer to invalidate them.

(c) Start the bandwidth release of the corresponding service stream and wait for the transport layer to complete the release of the flow control cache.

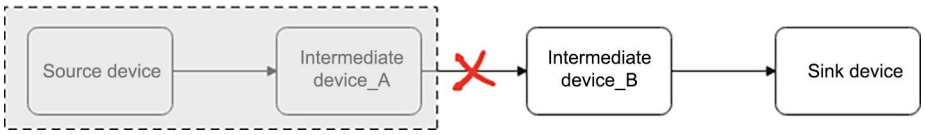
Figure 207 Schematic diagram of the scenario where the source device port is unplugged



9.5.7.2.2 Intermediate Device Port Unplugged

As shown in Figure 208, if an intermediate device port is unplugged, it will be handled as follows (The gray part in the figure will be handled according to the bandwidth release process as the source device identifies a topology change):

Figure 208 Schematic diagram of the scenario where an intermediate device port is unplugged



Initiating Device (Intermediate Device_B)

When a device generates a channel removal request message, the message will be processed as follows:

(a) Receive a port unplugging event.

(b) Extract the forward streams passing through this port.

(c) According to the port numbers InPortID and ShuttleID of the forward stream, query its output stream port numbers OutPortID and ShuttleID in the routing table, and generate a channel removal message.

(d) If the device is a multicast point, this message should be sent to all multicast output ports. If an output stream port has been unplugged, this message will be processed inside the management adapter.

(e) Send a request message according to the channel addressing result.

When a device receives a channel removal response message, the message will be processed as follows:

(a) Record the input port number InPortID.

(b) According to the port numbers InPortID and ShuttleID, query its output stream port numbers OutPortID and ShuttleID in the routing table.

(c) Determine whether it is the device initiating the channel removal message.

(d) Subtract the value of the ReceiverCount field carried in the message from the value of the ReceiverCount field of the corresponding stream of this port. When the value of the ReceiverCount field is reduced to 0, remove this path. If it is not reduced to 0, there is no need to remove the path nor perform steps (e) and (f).

(e) Start to release the flow control cache of the transport layer, and after it is completed, release the bandwidth.

(f) Issue the information in Table 155 to the transport layer.

Table 155 Information sent to transport layer for channel removal

Management Adapter	Transport Layer
Forward entry FwVld	FwVld
Service stream Vld	Vld

(g) For exception handling, refer to 9.2.3.3.

Destination Device (Sink Device)

When a device receives a channel removal request message, the message will be processed as follows:

(a) Record the input port number InPort_ID.

(b) According to the port numbers InPortID and ShuttleID, query its output stream port numbers OutPortID and ShuttleID in the routing table.

(c) Generate a response message that carries the ReceiverCount of the stream node.

(d) Return a response message according to the channel addressing result.

(e) For exception handling, refer to 9.2.3.3.

9.5.7.2.3 Sink Device Unplugged

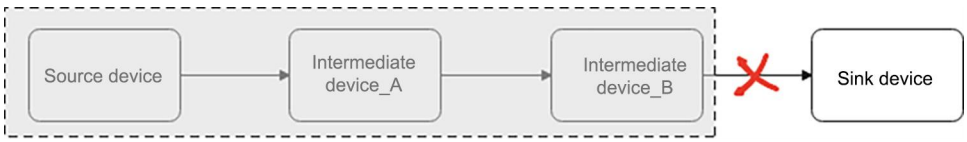
As shown in Figure 209, if the sink device port is unplugged, it will be handled as follows (The gray part in the figure will be handled according to the bandwidth release process as the source device identifies a topology change):

(a) Receive a port unplugging event.

(b) Set the valid bit of the service stream and the valid bit of the forwarded entry corresponding to the ShuttleID output passing through this port to 0, and issue them to the transport layer.

(c) For exception handling, refer to 9.2.3.3.

Figure 209 Schematic diagram of the scenario where the sink device port is unplugged



9.5.7.3 Message Description

The packet structure of the channel removal request message is as shown in Figure 210, and the field description is as shown in Table 156.

Figure 210 Packet structure of channel removal request message

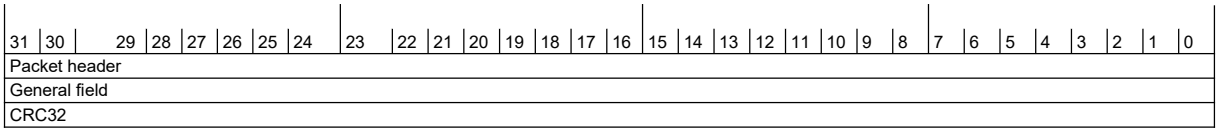


Table 156 Field description for packet of channel removal request message

Field Name	Length (Bits)	Description
Packet header	32	Refer to Section 9.2.1.1.
General field	32	Refer to Section 9.2.1.1.
CRC32	32	Check code

The packet structure of the channel removal response message is as shown in Figure 211, and the field description is as shown in Table 157.

Figure 211 Packet structure of channel removal response message

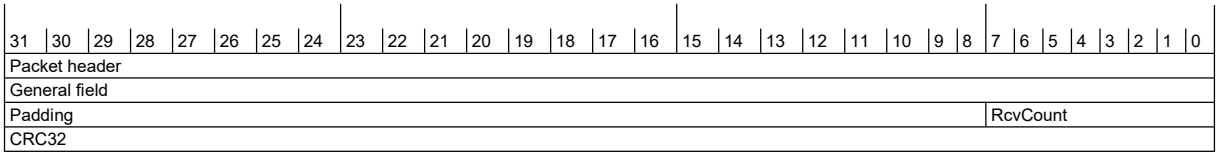


Table 157 Field description for packet of channel removal request message

Field Name	Length (Bits)	Description
Packet header	32	Refer to Section 9.2.1.1.
General field	32	Refer to Section 9.2.1.1.
RcvCount	8	Multicast count.
Padding	24	Fill in with 0 to ensure 4-byte alignment.
CRC32	32	Check code

9.5.8 Bandwidth Increase and Decrease Management

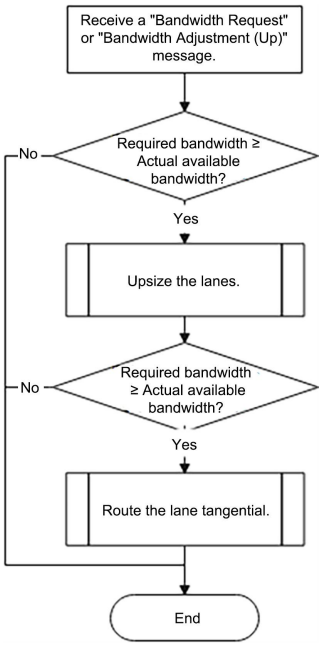
9.5.8.1 Overview

Management adapters complete bandwidth increase and decrease by means of LWAM messages and LDAM messages in the management packet of the logical layer, that is to say, lane increase or decrease is completed by means of LWAM messages, and lane direction switching is completed by means of LDAM messages.

9.5.8.2 Bandwidth Increase Management

When the bandwidth currently applied for or adjusted is greater than the actual available bandwidth, attempts may be made to increase the bandwidth through the lane increase process and the lane direction switching process. The reference control process is as shown in Figure 212.

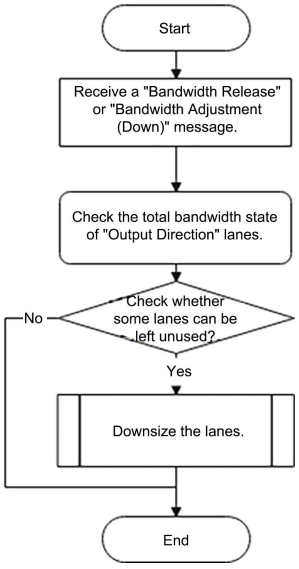
Figure 212 Reference lane increase process and lane direction switching process



9.5.8.3 Bandwidth Decrease Management

When the bandwidth currently released or adjusted is less than the actual available bandwidth, attempts may be made to reduce the bandwidth through the lane decrease process, and the idle lanes can enter the Low Power Consumption state. The reference control process is as shown in Figure 213.

Figure 213 Reference lane decrease process



9.6 Device Control

9.6.1 Overview

Device control messages are mainly used for information interaction and control between devices. Messages can be transmitted via the sideband link or the main link. The message type is fixed to 0x1B. The types and names of device control messages are as shown in Table 158. Device control messages are addressed by device address and only support unicast.

Table 158 Description of device control messages

Message Type	Command Type	Message Name	Whether It Must be Supported
0x1B	0x01	Status query	√
	0x02	Device wake-up	√
	0x03	Device standby	√
	0x04	Device name obtaining	
	0x05	Reserved	
	0x06	Adapter list	√
	0x07	Adapter binding	√
	0x08	Adapter unbinding	√
	0xEE	Vendor-defined	

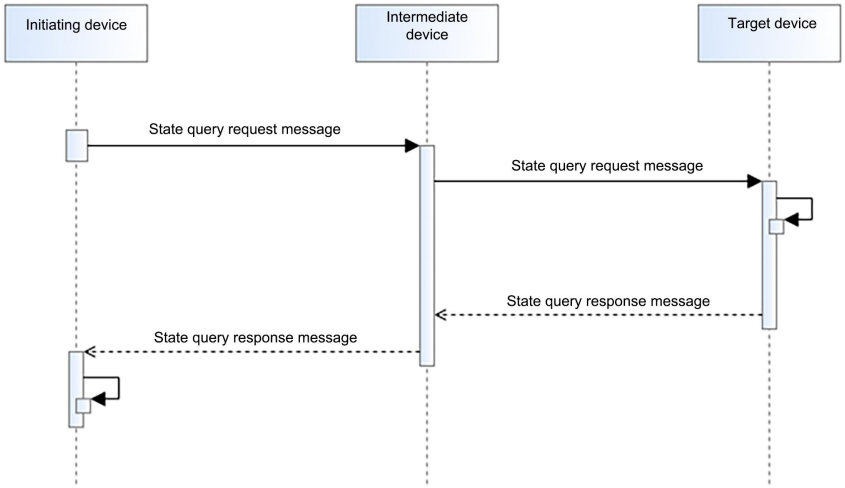
9.6.2 Device Status Query

9.6.2.1 Introduction

Device status query messages are used to query the status of a specific device in the GPMI network. These messages include status query request messages and status query response messages.

The status query process is as shown in Figure 214.

Figure 214 Status query process



9.6.2.2 Processing Flow

9.6.2.2.1 Initiating Device

When a device generates a device status query request message, the message will be processed as follows:

(a) When the device needs to query the status of a specific device in the GPMI network, generate a device status query request message.

(b) Send a request message based on the address table.

When a device receives a device status query response message, the message will be processed as follows:

(a) Extract device status information from the message.

(b) Report the acquired device status information to the application for unified processing by the application. Possible scenarios include: UI displaying the status of the remote device, waking up the remote destination device, and initiating a service transmission request to the remote destination device.

Exception handling: For error handling, refer to 9.2.3.3.

9.6.2.2.2 Intermediate Device

Send a forwarding request or response message according to the device address.

Exception handling: For error handling, refer to 9.2.3.3.

9.6.2.2.3 Target Device

When a device receives a device status query request message, the message will be processed as follows:

(a) Assign the current real state of the device to DevState, and generate the device status query response message. For status types, see the message description.

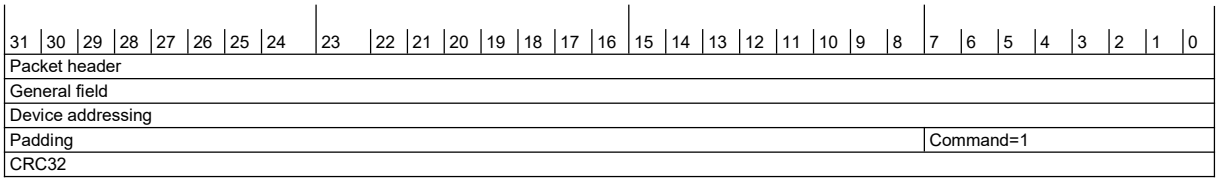
(b) Return a response message according to the address list.

Exception handling: For error handling, refer to 9.2.3.3.

9.6.2.3 Message Description

The structure of status query request message is as shown in Figure 215.

Figure 215 Structure of status query request message



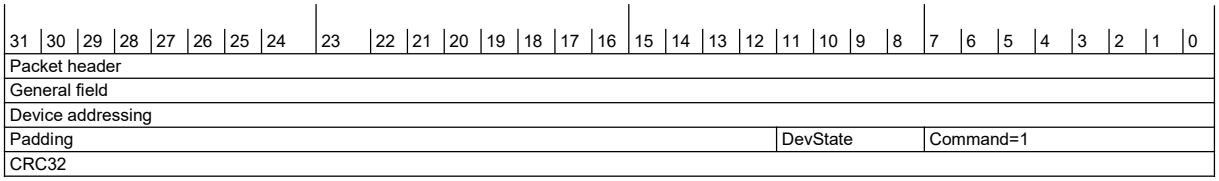
The variable description of status query request message is as shown in Table 159.

Table 159 Field description of status query request message

DW	Bit	Name	Description
0	31:0	Packet header	Refer to Section 9.2.1.1.
1	31:0	General field	Refer to Section 9.2.1.1.
2–5	31:0	Device addressing	Refer to Section 9.2.2.
	…	Device addressing	Refer to Section 9.2.2.
6	7:0	Command	The device control command number, which is 1.
	31:8	Padding	Pad to ensure 4-byte alignment.
7	31:0	CRC32	Check code

The structure of status query response message is as shown in Figure 216.

Figure 216 Structure of status query response message



The variable description of status query response message is as shown in Table 160.

Table 160 Field description of status query response message

DW	Bit	Name	Description
0	31:0	Packet header	Refer to Section 9.2.1.1.

DW	Bit	Name	Description
1	31:0	General field	Refer to Section 9.2.1.1.
2–5	31:0	Device addressing	Refer to Section 9.2.2.
	…	Device addressing	Refer to Section 9.2.2.
6	7:0	Command	The device control command number, which is 1.
	11:8	DevState (Device State)	Device state: 0x00: Working state. 0x01: Standby state. 0x02: Being in standby, that is, in the standby process. 0x03: Being woken up, that is, in the wake-up process. 0x04 to 0x0F: Reserved.
	31:12	Padding	Pad to ensure 4-byte alignment.
7	31:0	CRC32	Check code

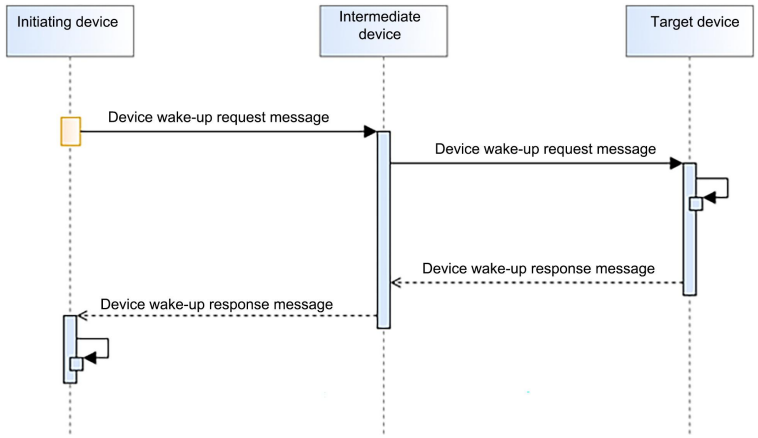
9.6.3 Device Wake-up

9.6.3.1 Introduction

Device wake-up messages are used to wake up a specific device in the GPMI network, and are often used in scenarios such as one-click playback. Device wake-up messages include device wake-up request messages and device wake-up response messages.

The device wake-up processing flow is as shown in Figure 217.

Figure 217 Device wake-up processing flow



9.6.3.2 Processing Flow

9.6.3.2.1 Initiating Device

When a device generates a device wake-up request message, the message will be processed as follows:

(a) When a device needs to wake up a destination device in the GPMI network, it will generate a device wake-up request message.

(b) Send a request message based on the address table.

When a device receives a device wake-up response message, the message will be processed as follows:

(a) Identify whether the destination device has entered the wake-up process.

(b) The device status query message can be used to confirm whether the destination device has been woken up.

(c) For exception handling, refer to 9.2.3.3.

9.6.3.2.2 Intermediate Device

Forward the request or response message according to the address table.

For exception handling, refer to 9.2.3.3.

9.6.3.2.3 Target Device

When a device receives a device wake-up request message, the message will be processed as follows:

(a) If the device is currently in the Standby state, perform the wake-up action, and generate a response message at the same time.

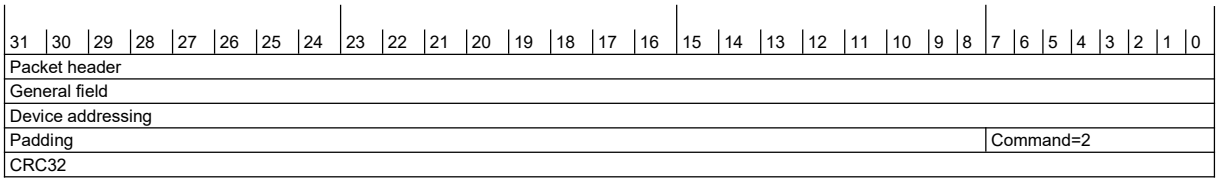
(b) Return a response message according to the address list.

Exception handling: If the current device is already in the working state, refresh ErrCode to 0x2 (device status error). For other error handling, refer to 9.2.3.3.

9.6.3.3 Message Description

The structure of device wake-up request message is as shown in Figure 218.

Figure 218 Structure of device wake-up request message



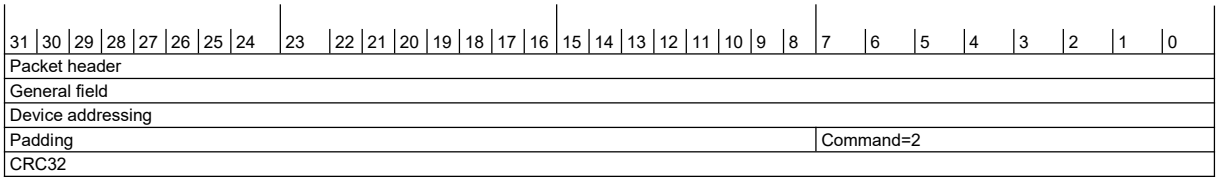
The variable description of device wake-up request message is as shown in Table 161.

Table 161 Field description of device wake-up request message

DW	Bit	Name	Description
0	31:0	Packet header	Refer to Section 9.2.1.1.
1	31:0	General field	Refer to Section 9.2.1.1.
2–5	31:0	Device addressing	Refer to Section 9.2.2.
	…	Device addressing	Refer to Section 9.2.2.
6	7:0	Command	The device control command number, which is 2.
	31:8	Padding	Fill in with data to ensure 4-byte alignment.
7	31:0	CRC32	Check code

The structure of device wake-up response message is as shown in Figure 219.

Figure 219 Structure of device wake-up response message



The variable description of device wake-up response message is as shown in Table 162.

Table 162 Field description of device wake-up response message

DW	Bit	Name	Description
0	31:0	Packet header	Refer to Section 9.2.1.1.
1	31:0	General field	Refer to Section 9.2.1.1.
2–5	31:0	Device addressing	Refer to Section 9.2.2.
	…	Device addressing	Refer to Section 9.2.2.
6	7:0	Command	The device control command number, which is 2.
	31:8	Padding	Fill in with data to ensure 4-byte alignment.
7	31:0	CRC32	Check code

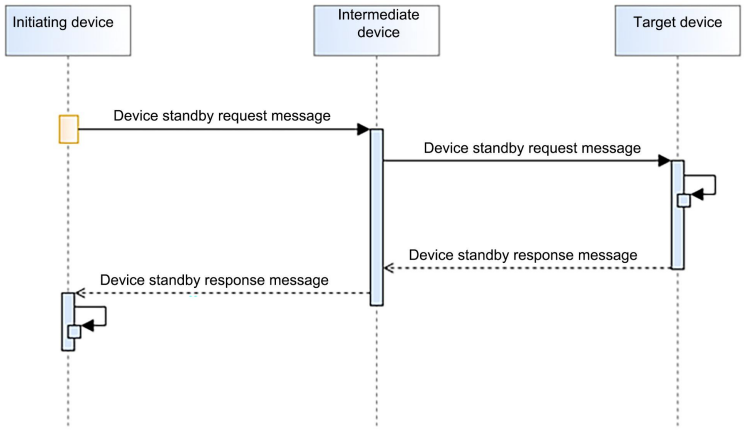
9.6.4 Device Standby

9.6.4.1 Introduction

Device standby messages are used to make a specific device in the GPMI network standby, and are often used in scenarios such as one-key standby. Device standby messages include device standby request messages and device standby response messages.

The device standby process is as shown in Figure 220.

Figure 220 Device standby process



9.6.4.2 Processing Flow

9.6.4.2.1 Initiating Device

When a device generates a device standby request message, the message will be processed as follows:

(a) When a device needs to make a destination device in the GPMI network to standby, it will generate a device standby request message.

(b) Send a request message based on the address table.

When a device receives a device standby response message, the message will be processed as follows:

(a) Identify whether the destination device has entered the standby process.

(b) The device status query message can be used to confirm whether the destination device has completed the standby process.

For exception handling, refer to 9.2.3.3.

9.6.4.2.2 Intermediate Device

Forward the request or response message according to the address table.

For exception handling, refer to 9.2.3.3.

9.6.4.2.3 Target Device

When a device receives a device standby request message, the message will be processed as follows:

(a) If the device is currently in the Working state, perform the standby action, and generate a response message.

(b) Return a response message according to the address list.

Exception handling: If the current device is already in the Standby state, refresh ErrCode to 0x2 (device status error). For other error handling, refer to 9.2.3.3.

9.6.4.3 Message Description

The structure of device standby request message is as shown in Figure 221.

Figure 221 Structure of standby request message

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
Packet header			
General field			
Device addressing			
Padding			Command=3
CRC32			

The variable description of device standby request message is as shown in Table 163.

Table 163 Field description of device standby request message

DW	Bit	Name	Description
0	31:0	Packet header	Refer to Section 9.2.1.1.
1	31:0	General field	Refer to Section 9.2.1.1.
2–5	31:0	Device addressing	Refer to Section 9.2.2.
	…	Device addressing	Refer to Section 9.2.2.
6	7:0	Command	The device control command number, which is 3.
	31:8	Padding	Fill in with data to ensure 4-byte alignment.
7	31:0	CRC32	Check code

The structure of device standby response message is as shown in Figure 222.

Figure 222 Structure of device standby response message

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
Packet header			
General field			
Device addressing			
Padding			Command=3
CRC32			

The variable description of device standby response message is as shown in Table 164.

Table 164 Field description of device standby response message

DW	Bit	Name	Description
0	31:0	Packet header	Refer to Section 9.2.1.1.
1	31:0	General field	Refer to Section 9.2.1.1.
2–5	31:0	Device addressing	Refer to Section 9.2.2.
	…	Device addressing	Refer to Section 9.2.2.
6	7:0	Command	The device control command number, which is 3.
	8	BD (Broadcast)	Whether it is a multicast message: 0: unicast message. 1: multicast message.
	31:9	Padding	Fill in with data to ensure 4-byte alignment.
7	31:0	CRC32	Check code

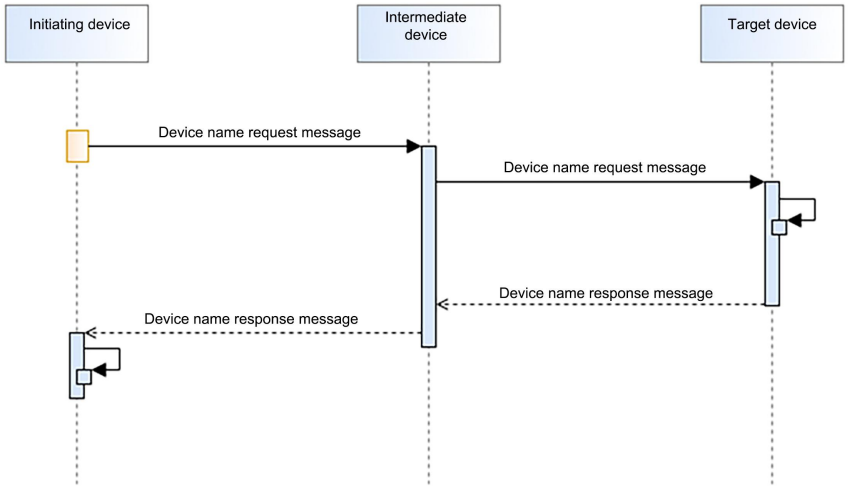
9.6.5 Device Name Obtaining

9.6.5.1 Introduction

Device name obtaining messages are used to obtain the name of a specific device in the GPMI network, and are often used in scenarios such as UI display or device management. Device name obtaining messages include device name obtaining request messages and device name obtaining response messages.

The processing flow of device name obtaining message is as shown in Figure 223.

Figure 223 Device name obtaining process



9.6.5.2 Processing Flow

9.6.5.2.1 Initiating Device

When a device generates a device name obtaining request message, the message will be processed as follows:

(a) When a user or device needs to obtain the name of a destination device in the GPMI network, it will generate a device name obtaining request message.

(b) Send a request message based on the address table.

When a device receives a device name obtaining response message, the message will be processed as follows:

(a) Extract the device name and report to the application.

(b) The application can display the device name on the UI.

For exception handling, refer to 9.2.3.3.

9.6.5.2.2 Intermediate Device

Forward the request or response message according to the address table.

For exception handling, refer to 9.2.3.3.

9.6.5.2.3 Target Device

When a device receives a device name obtaining request message, the message shall be processed as follows:

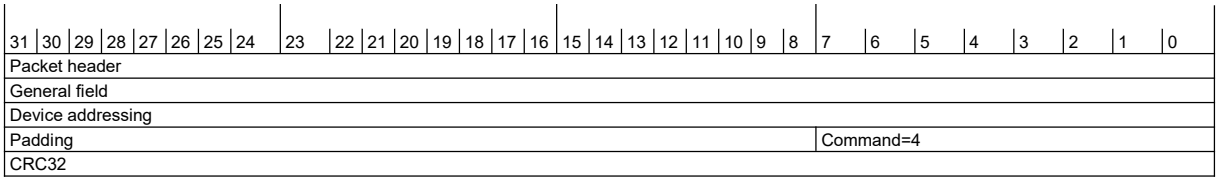
(a) Fill the device name into the message, and generate a response message. The device name has a maximum length of 32 characters. If it is longer than 32 characters, only the first 32 characters will be taken.

(b) Return a response message according to the address list.

9.6.5.3 Message Description

The structure of device name obtaining request message is as shown in Figure 224.

Figure 224 Structure of device name obtaining request message



The variable description of device name obtaining request message is as shown in Table 165.

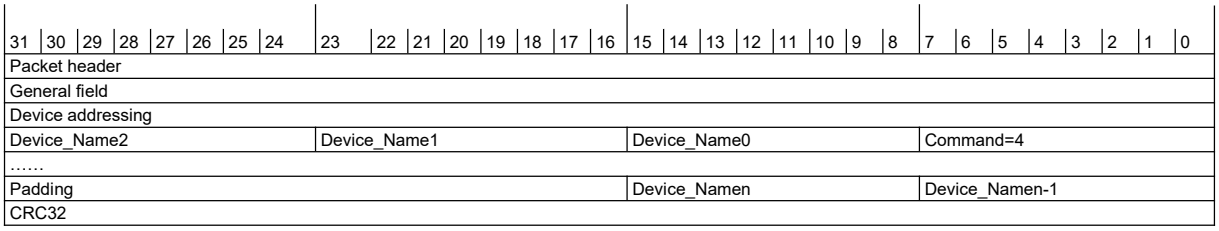
Table 165 Field description of device name obtaining request message

DW	Bit	Name	Description
0	31:0	Packet header	Refer to Section 9.2.1.1.

DW	Bit	Name	Description
1	31:0	General field	Refer to Section 9.2.1.1.
2–5	31:0	Device addressing	Refer to Section 9.2.2.
	…	Device addressing	Refer to Section 9.2.2.
6	7:0	Command	The device control command number, which is 4.
	31:8	Padding	Fill in with data to ensure 4-byte alignment.
7	31:0	CRC32	Check code

The structure of device name obtaining response message is as shown in Figure 225.

Figure 225 Structure of device name obtaining response message



The description of variables in device name obtaining response message is shown in Table 166.

Table 166 Field description of the device name obtaining response message

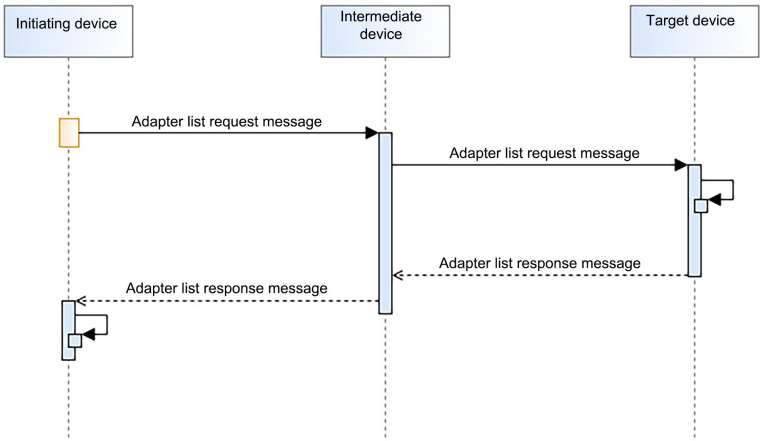
DW	Bit	Name	Description
0	31:0	Packet header	Refer to Section 9.2.1.1.
1	31:0	General field	Refer to Section 9.2.1.1.
2–5	31:0	Device addressing	Refer to Section 9.2.2.
	…	Device addressing	Refer to Section 9.2.2.
6	7:0	Command	The device control command number, which is 4.
	15:8	Device Name 0	The first character of the device name
	23:16	Device Name 1	The second character of the device name
	31:24	Device Name 2	The third character of the device name
…	…		The *n*th character of the device name. *n* is a maximum of 8 DWs and 32 characters.
N + n	31:0	CRC32	Check code

9.6.6 Adapter List Obtaining

9.6.6.1 Introduction

The adapter list obtaining request message is used to obtain the adapter list of a specific device in the GPMI network, and is often used to obtain detailed information about the device adapter when a channel is established. It consists of a request message and a response message for obtaining the adapter list. The process of obtaining the adapter list is shown in Figure 226.

Figure 226 Process of obtaining the adapter list



9.6.6.2 Processing Flow

9.6.6.2.1 Initiating Device

The device generates a request message for obtaining the adapter list, which is processed according to the following rules:

(a) Before a channel is established, when it is necessary to obtain the adapted list information, a request message for obtaining the adapter list is generated.

(b) Send a request message based on the address table.

The device receives the response message for obtaining the adapter list, which is processed according to the following rules

(a) Extract adapter information from the response message.

(b) This information can be used to support the planning of a channel.

For exception handling, refer to 9.2.3.3.

9.6.6.2.2 Intermediate Device

Forward the request or response message according to the address table.

For exception handling, refer to 9.2.3.3.

9.6.6.2.3 Target Device

When the device receives a request message for obtaining the adapter list, it processes the message according to the following rules:

(a) Fill the message body with all adapter information for which the device is online.

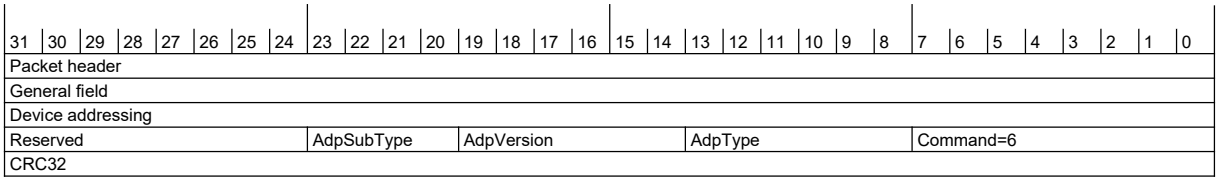
(b) Return a response message according to the address list.

For exception handling, refer to 9.2.3.3.

9.6.6.3 Message Description

The adapter list request message structure is described in Figure 227.

Figure 227 Adapter list request message structure



The variable description of the adapter list request message is shown in Table 167.

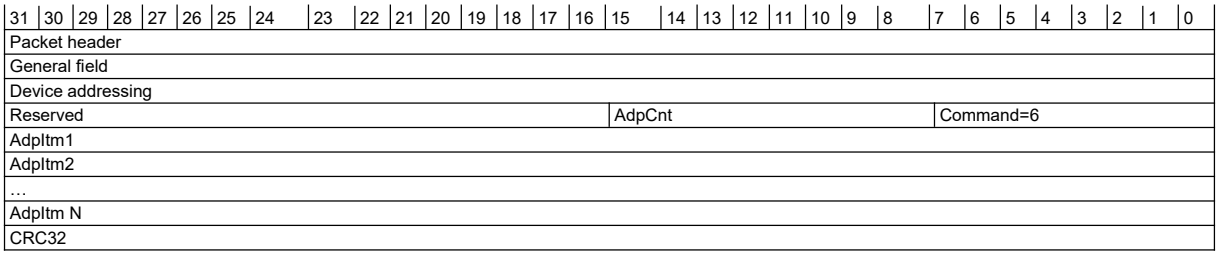
Table 167 Field description of the adapter list request message

DW	Bit	Name	Description
0	31:0	Packet header	Refer to Section 9.2.1.1.
1	31:0	General field	Refer to Section 9.2.1.1.
2–5	31:0	Device addressing	Refer to Section 9.2.2.
	…	Device addressing	Refer to Section 9.2.2.
6	7:0	Command	Device control command number: 6
	13:8	AdpType	Adapter type: 0: Query all other adapters except the management adapter. 1: video adapter Others: reserved
	19:14	AdpVersion	Adapter version: 0: all versions Other values: Query the corresponding version number.
	23:20	AdpSubType	Adapter sub-type: 0: all sub-types 1: downlink (transmitting) adapter 2: uplink (receiving) adapter Others: reserved

DW	Bit	Name	Description
	31:24	Reserved	Reserved.
7	31:0	CRC32	Check code

The structure of the adapter list obtaining response message is described in Figure 228.

Figure 228 Adapter list obtaining response message structure



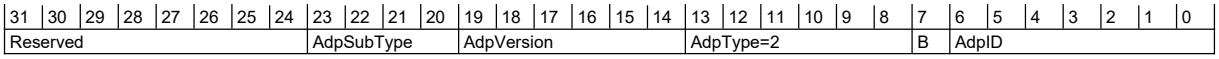
The variable description of the adapter list request response message is shown in Table 168.

Table 168 Field description of the adapter list request response message

DW	Bit	Name	Description
0	31:0	Packet header	Refer to Section 9.2.1.1.
1	31:0	General field	Refer to Section 9.2.1.1.
2–5	31:0	Device addressing	Refer to Section 9.2.2.
	…	Device addressing	Refer to Section 9.2.2.
6	7:0	Command	Device control command number: 6
	16:8	AdpCnt Adapter Count	Number of adapters
	31:16	Reserved	Reserved.
7	31:0	AdpItm 1 Adapter Item 1	Adapter description item 1
8	31:0	AdpItm 2 Adapter Item 2	Adapter description item 2
	…	…	
	31:0	AdpItm N Adapter Item 2	Adapter description item *N*
	31:0	CRC32	Check code

The format of the adapter description items for the video adapter is shown in Figure 229.

Figure 229 Adapter list obtaining response message structure



The fields of the video adapter description item are shown in Table 169.

Table 169 Field description of the video adapter description item

DW	Bit	Name	Description
6	6:0	AdpID	Device control command number: 6
	7	B (Binding)	Binding ID: 0: not bound 1: bound
	13:8	AdpType (Adapter Type)	Adapter type: 2: video adapter Others: reserved.
	19:14	AdpVersion (Adapter Version)	Adapter version: 1.
	23:20	AdpSubType (Adapter Sub-Type)	Adapter sub-type: 1: downlink (transmitting) adapter 2: uplink (receiving) adapter Others: reserved
	31:24	Reserved	Reserved.

9.6.7 Adapter Binding

9.6.7.1 Introduction

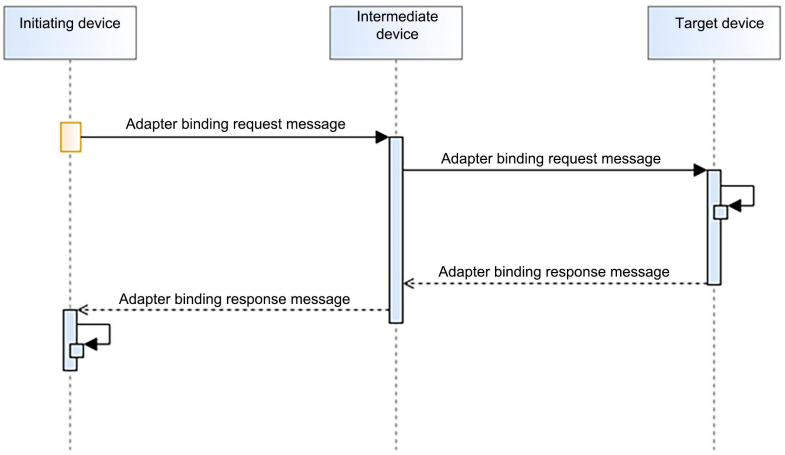
The binding operation is to pair two adapters to establish a channel between them. The two adapters to be bound shall follow the following rules: audio transmitting adapter and audio/ video receiving adapter.

The bandwidth allocation operation must be initiated by the audio and video source device. Therefore, when the video sink device establishes a channel it must initiate a binding request to the corresponding sink device. After receiving the binding request, the video source device initiates a bandwidth allocation request to establish the corresponding channel.

The adapter binding request must be initiated by the sink device (TV or monitor). For example, when the user selects a video source on the TV, the TV initiates adapter binding to the video source device. After receiving the adapter binding request, the video source device initiates a bandwidth allocation request to the TV to establish a channel for streaming data from the audio and video transmission adapter to the audio and video receiving adapter. In a complex network containing the GPMI, the source device cannot initiate an adapter binding request.

The adapter binding process flow is shown in Figure 230.

Figure 230 Adapter binding process



9.6.7.2 Processing Flow

9.6.7.2.1 Initiating Device

The device generates a request message for adapter binding, which is processed according to the following rules:

(a) According to service needs, select the target device and adapter to be bound, and generate an adapter binding request message.

(b) Send a request message based on the address table.

The device receives the adapter binding response message and processes it according to the following rules:

(a) The target adapter has been bound.

(b) The adapter binding information needs to be recorded synchronously, that is, the bound device address and adapter ID.

For exception handling, refer to 9.2.3.3.

9.6.7.2.2 Intermediate Device

Forward the request or response message according to the address table.

For exception handling, refer to 9.2.3.3.

9.6.7.2.3 Target Device

After receiving the adapter binding request message, the target device should handle it according to the following rules:

(a) Bind the adapter, record the adapter binding information, and generate an adapter binding response message.

(b) Return a response message according to the address list.

Exception handling: If the adapter is currently in a bound state, assign ErrCode to 0x5 (rejecting the request) and return a response message directly. For other exception handling, refer to 9.2.3.3.

9.6.7.3 Message Description

The adapter binding request message structure is described in Figure 231.

Figure 231 Adapter binding request message structure

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
Packet header			
General field			
Device addressing			
Reserved	DEST_AdapterID	SRC_AdapterID	Command=7
DEST_ADDR[31:0]			
DEST_ADDR[63:32]			
OutStreamBW			
InStreamBW			
CR32			

The description of the adapter binding request message variables is shown in Table 170.

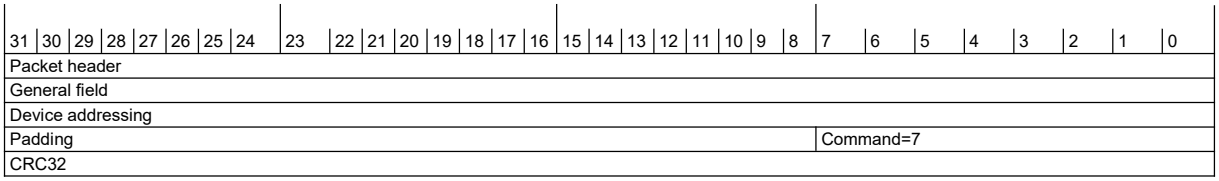
Table 170 Field description of the adapter binding request message

DW	Bit	Name	Description
0	31:0	Packet header	Refer to Section 9.2.1.1.
1	31:0	General field	Refer to Section 9.2.1.1.
2–5	31:0	Device addressing	Refer to Section 9.2.2.
	…	Device addressing	Refer to Section 9.2.2.
6	7:0	Command	Device control command number: 7
	15:8	SRC_AdapterID (Source Device AdapterID)	Source device adapter ID, which indicates the adapter ID of the device that initiates the binding request message. The device address corresponding to this adapter is SRC_ADDR.
	23:16	DEST_AdapterID (Destnation Device AdapterID)	Target device adapter ID, which indicates the adapter ID of the device that needs to be bound. The device address corresponding to this adapter is DEST_ADDR.
	31:24	Reserved	Reserved.
7	31:0	DEST_ADDR[31:0]	The target device address to be bound is 32 bits lower.
8	31:0	DEST_ADDR[63:32]	The target device address to be bound is 32 bits higher.
9	31:0	OutStreamBW (OutPut Stream BandWidth)	Outflow bandwidth: Only when the host initiates a USB adaptation binding message, the outflow bandwidth value of the corresponding downlink adaptation of the

DW	Bit	Name	Description
			host is carried. In other scenarios, an invalid value 0xFFFFFFFF is used by default.
10	31:0	InStreamBW (InPut Stream BandWidth)	Inflow bandwidth: Only when the host initiates a USB adaptation binding message, the inflow bandwidth value of the corresponding downlink adaptation of the host is carried. In other scenarios, an invalid value 0xFFFFFFFF is used by default.
11	31:0	CRC32	Check code

The adapter binding response message structure is described in Figure 232.

Figure 232 Adapter binding response message structure



The description of the adapter binding response message variables is shown in Table 171.

Table 171 Field description of the adapter binding response message

DW	Bit	Name	Description
0	31:0	Packet header	Refer to Section 9.2.1.1.
1	31:0	General field	Refer to Section 9.2.1.1.
2–5	31:0	Device addressing	Refer to Section 9.2.2.
	…	Device addressing	Refer to Section 9.2.2.
6	7:0	Command	Device control command number: 7
	31:8	Padding	Padding data. Pad four bytes for alignment.
7	31:0	CRC32	Check code

9.6.8 Adapter Unbinding

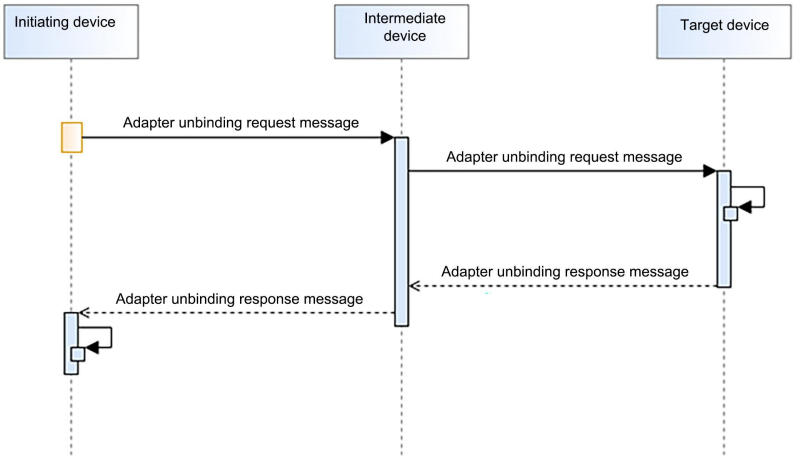
9.6.8.1 Introduction

Adapter unbinding refers to unbinding an adapter from the originally bound adapter and removing the channel from the source adapter to the target adapter. The actual removal of the channel needs to be completed through a bandwidth release message. When the bandwidth of the channel is released, it is considered that the unbinding is successful. It consists of an adapter unbinding request message and an adapter unbinding response message.

The adapter unbinding request shall be initiated by the sink device (TV or monitor).

The adapter unbinding process is shown in Figure 233.

Figure 233 Adapter unbinding process



9.6.8.2 Message Processing Flow

9.6.8.2.1 Initiating Device

The device generates a request message for adapter unbinding, which is processed according to the following rules:

(a) Generate an adapter unbinding request message according to service needs.

(b) Send a request message based on the address table.

The device receives the adapter unbinding response message and processes it according to the following rules:

(a) The target device has accepted the adapter unbinding request.

(b) If a service channel has been established, wait for the source device to initiate a bandwidth release request. After the bandwidth release request is completed, clear the binding information.

(c) If no channel is established, the binding information is cleared.

(d) For exception handling, refer to 9.2.3.3.

9.6.8.2.2 Intermediate Device

Forward the request or response message according to the address table.

For exception handling, refer to 9.2.3.3.

9.6.8.2.3 Target Device

When the device receives an adapter unbinding request message, it should be processed according to the following rules:

(a) The device decides whether to accept the unbinding request based on its own service conditions and generates an adapter unbinding response message.

(b) Return a response message according to the address list.

(c) If the unbinding request has been accepted and there is currently a channel, the bandwidth release of the channel needs to be initiated through a bandwidth release request message. After the bandwidth release is completed, clear the binding information.

(d) If the unbinding request has been accepted and there is no channel at present, the binding information is cleared directly.

For exception handling, refer to 9.2.3.3.

9.6.8.3 Message Description

The adapter unbinding request message structure is described in Figure 234.

Figure 234 Adapter unbinding request message structure

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
Packet header			
General field			
Device addressing			
Reserved	DEST_AdapterID	SRC_AdapterID	Command=8
DEST_ADDR[63:32]			
DEST_ADDR[31:0]			
CRC32			

The description of the adapter unbinding request message variables is shown in Table 172.

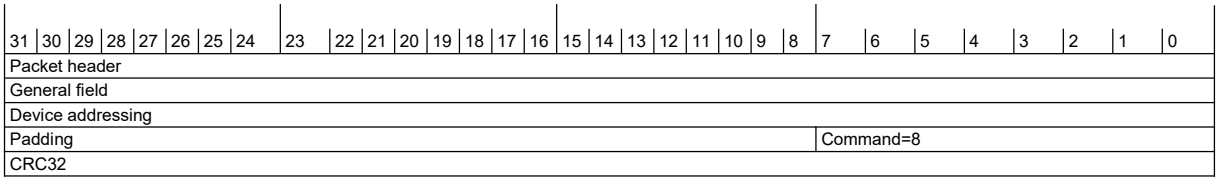
Table 172 Field description of the adapter unbinding request message

DW	Bit	Name	Description
0	31:0	Packet header	Refer to Section 9.2.1.1.
1	31:0	General field	Refer to Section 9.2.1.1.
2–5	31:0	Device addressing	Refer to Section 9.2.2.
	…	Device addressing	Refer to Section 9.2.2.
6	7:0	Command	Device control command number: 8
	15:8	SRC_AdapterID (Source Device AdapterID)	Source device adapter ID, which refers to the ID of the audio and video transmitting adapter of the message target device or the downlink adapter ID of the USB. The device address corresponding to this adapter pair is the target address of the "Device Address" sub-segment.
	23:16	DEST_AdapterID (Destnation Device AdapterID)	Target device adapter ID, which refers to the ID of the audio and video receiving adapter or USB uplink adapter ID of the target device that needs to be bound. The device address corresponding to this adapter pair is DEST_ADD.

DW	Bit	Name	Description
	31:24	Reserved	Reserved.
7	31:0	DEST_ADDR[63:32]	The target device address to be bound is 32 bits higher.
8	31:0	DEST_ADDR[31:0]	The target device address to be bound is 32 bits lower.
9	31:0	CRC32	Check code

The adapter unbinding response message structure is described in Figure 235.

Figure 235 Adapter unbinding response message structure



The description of the adapter unbinding response message variables is shown in Table 173.

Table 173 Field description of the adapter unbinding response message

DW	Bit	Name	Description
0	31:0	Packet header	Refer to Section 9.2.1.1.
1	31:0	General field	Refer to Section 9.2.1.1.
2–5	31:0	Device addressing	Refer to Section 9.2.2.
	…	Device addressing	Refer to Section 9.2.2.
6	7:0	Command	Device control command number: 8
	31:8	Padding	Padding data. Pad four bytes for alignment.
7	31:0	CRC32	Check code

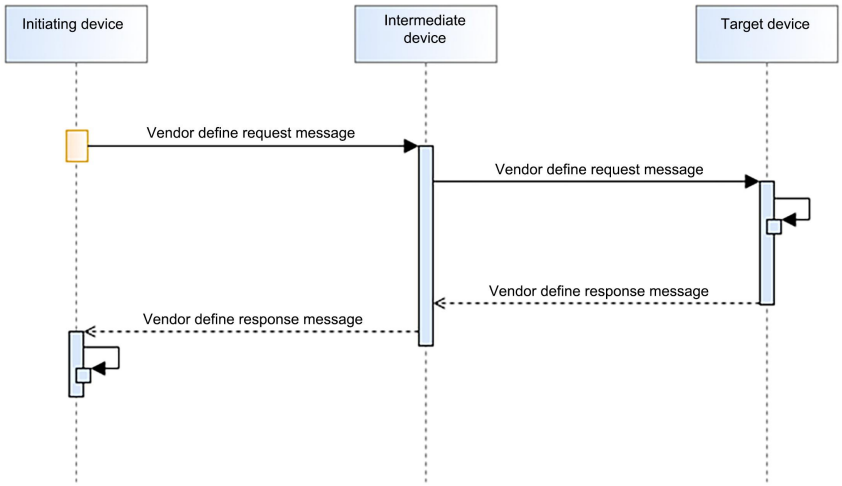
9.6.9 Vendor-defined

9.6.9.1 Introduction

Vendor-defined messages can support user customization and are easy to expand. When vendors or users customize messages, they need to clearly define the meaning and corresponding behavior of each field.

The processing flow of the vendor-defined message is shown in Figure 236.

Figure 236 Sequence diagram of vendor-defined message processing flow



9.6.9.2 Processing Flow

9.6.9.2.1 Initiating Device

When the device needs to generate vendor-defined custom messages according to service needs, the custom content must follow the corresponding custom content generated and the vendor ID must be accurately entered. Refer to Section 9.2.2.1.1 for more information.

When the state query response message fed back by the target device is received, it should be processed according to the content and processing rules of the vendor-defined custom message.

9.6.9.2.2 Intermediate Device

Forward the request or response message according to the address table.

For exception handling, refer to 9.2.3.3.

9.6.9.2.3 Target Device

When the target device receives a vendor-defined request message, it needs to process it according to the content and processing rules of the vendor-defined custom message.

9.6.9.3 Message Description

The vendor-defined request message structure is shown in Figure 237.

Figure 237 Vendor-defined request message structure

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
Packet header			
General field			
Device addressing			
Vendor Defined	Vendor ID		Command=238
Vendor Defined			
Padding			Vendor Defined
CRC32			

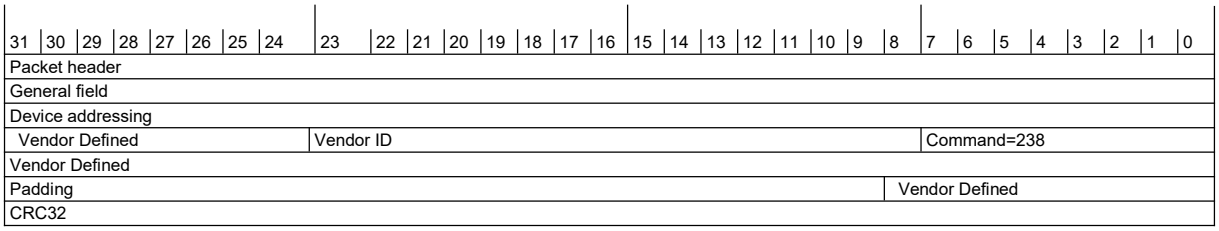
The variables of the vendor-defined request message are described in Table 174.

Table 174 Variables of the vendor-defined request message

DW	Bit	Name	Description
0	31:0	Packet header	Refer to Section 9.2.1.1.
1	31:0	General field	Refer to Section 9.2.1.1.
2–5	31:0	Device addressing	Refer to Section 9.2.2.
	…	Device addressing	Refer to Section 9.2.2.
6	7:0	Command	Device control command number: 238 (0xEE)
	15:8	Vendor ID	Device address, which is determined by the vendor identification code low byte
	24:16	Vendor ID	Device address, which is determined by the vendor identification code high byte
	31:25	Vendor Defined	Vendor-defined messages
…	…	Vendor Defined	Vendor-defined messages
		Padding	Pad to ensure 4-byte alignment.
…	31:0	CRC32	Check code

The structure of the vendor-defined response message is shown in Figure 238.

Figure 238 Vendor-defined response message structure



The variables of the vendor-defined response message are described in Table 175.

Table 175 Vendor-defined response message variables

DW	Bit	Name	Description
0	31:0	Packet header	Refer to Section 9.2.1.1.
1	31:0	General field	Refer to Section 9.2.1.1.
2–5	31:0	Device addressing	Refer to Section 9.2.2.
	…	Device addressing	Refer to Section 9.2.2.
6	7:0	Command	Device control command number: 238 (0xEE)

DW	Bit	Name	Description
	15:8	Vendor ID	Device address, which is determined by the vendor identification code low byte
	24:16	Vendor ID	Device address, which is determined by the vendor identification code high byte.
	31:25	Vendor Defined	Vendor-defined messages
…	…	Vendor Defined	Vendor-defined messages
		Padding	Pad to ensure 4-byte alignment.
…	31:0	CRC32	Check code

9.7 Content Protection

9.7.1 Overview

The GPMI uses ADCP to realize the content protection for audio and video flows.

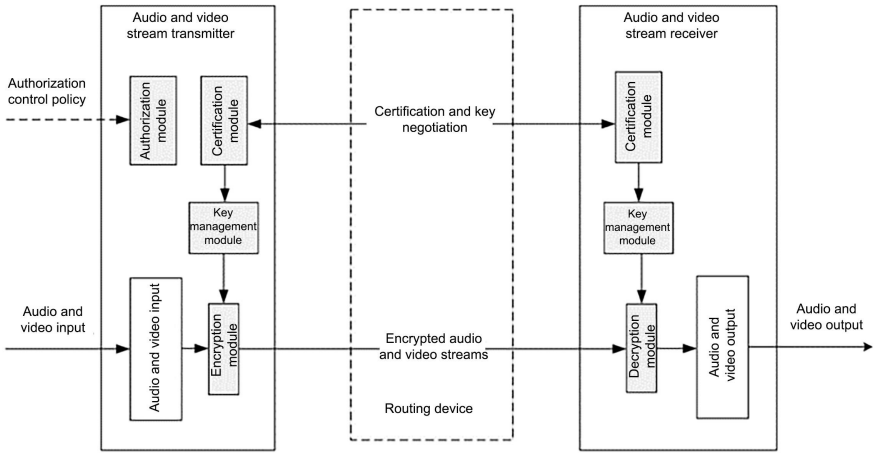
As shown in Figure 239, ADCP content protection consists of two parts:

- Authentication and key negotiation: The audio and video flow transmitter shall complete the authentication and key negotiation of the audio and video flow receiver before transmitting the encrypted audio and video flow to the receiver.
- Encryption and decryption information: The audio and video stream transmitter generates an encrypted description packet (EDP) and a key distribution packet (KDP) according to the authorization control policy of the current audio and video data flow, and transmits the EDP and KDP together with the audio and video flow to the receiver.

The encryption and decryption information of ADCP is transmitted through the EDP and KDP in the audio and video flows (see 8.7). The "Authentication and Key Negotiation" of ADCP needs to be completed through the management adapter. This chapter mainly introduces the processing of ADCP "Authentication and Key Negotiation" related messages in the management adapter.

The "Authentication and Key Negotiation" related messages of ADCP need to be encapsulated into management adapter packets before being transmitted and received through the management adapter.

Figure 239 Logical architecture of content protection



9.7.2 Request Message

The packet format of the content protection request message is shown in Figure 240, and the fields are described in Table 176.

Figure 240 Packet format of the content protection request message

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Packet header																															
General field																															
Destination Address Low																															
Destination Address High																															
Source Address Low																															
Source Address High																															
ADCP Message																															
CRC32																															

Table 176 Packet fields of the content protection request message

Field Name	Length (Bits)	Description
Packet header	32	Refer to Section 9.2.1.1.
General field	32	Refer to Section 9.2.1.1.
Destination Address Low	32	Bit [31:0] of the target address to which messages are received
Destination Address High	2	Bit [63:32] of the target address to which messages are received
Source Address Low	32	Bit [31:0] of the source address that generates this message.
Source Address High	32	Bit [63:32] of the source address that generates messages
ADCP Message	/	ADCP message

Field Name	Length (Bits)	Description
CRC32	32	Check code

ADCP messages are byte-aligned, while management adapter packets require 4-byte alignment. When the total length of the ADCP message is not 4-byte aligned, it is necessary to pad byte 0 at the end to achieve 4-byte alignment (the number of padding bytes is not more than 3 bytes).

For the list of ADCP authentication and key negotiation message types, see *Technical Specification of Advanced Digital Content Protection System*.

Note: The maximum length of the management adapter packet is 512 bytes, excluding the packet header (4 bytes), general field (4 bytes), addressing information (device addressing, 16 bytes), and CRC32 (4 bytes). The maximum message data can carry 484 bytes. When the total length of the ADCP message exceeds 484 bytes, it needs to be split into multiple content protection management adapter packets for transmission.

The ADCP module needs to encapsulate the ADCP message into a content protection management adapter packet and then hand it over to the management adapter for processing. The ADCP module needs to ensure the reliability and service integrity of ADCP message communication by itself. The management adapter is not responsible for the reliability of content protection management adapter packet transmission. Furthermore, it does not support retransmission of content protection management adapter packets after a timeout. The management adapter directly discards the content protection management adapter packets that fail CRC verification after receiving them.

For detailed ADCP message structure and interaction process, see *Technical Specification of Advanced Digital Content Protection System*.

9.7.3 Response Message

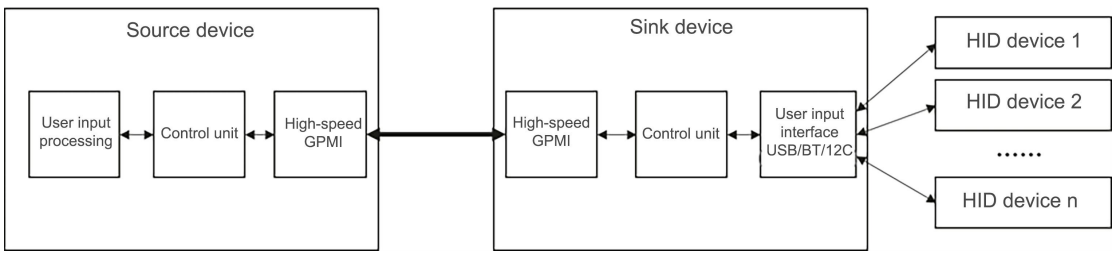
Content protection has no response message.

9.8 HID Pass-Through

9.8.1 Overview

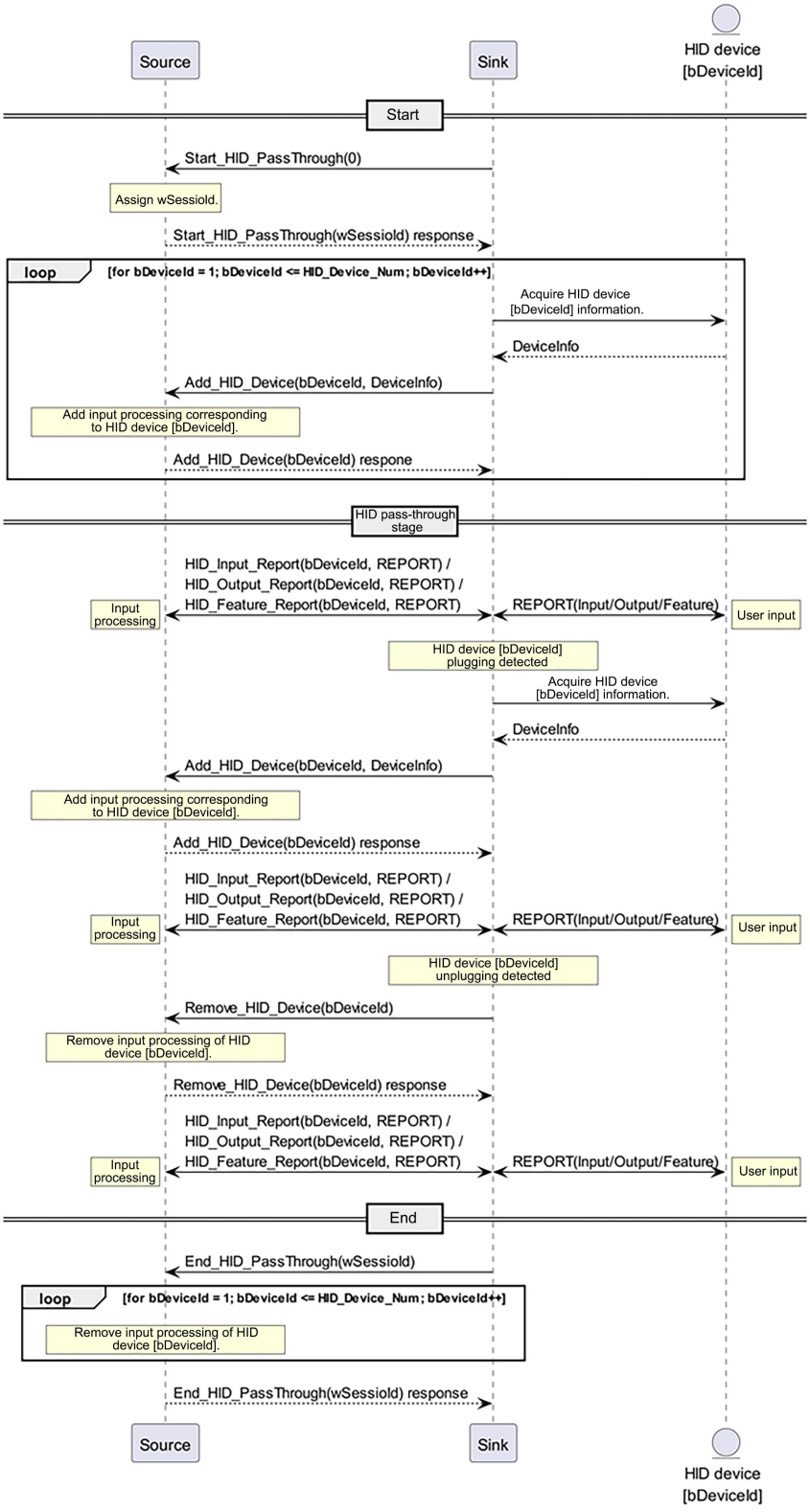
HID pass-through is a method in which the sink device transparently transmits the HID report generated by the local HID device to the source device through a high-speed interface (such as GPMI), and feeds back the processing result of the user input of the source device to the sink device HID device, thereby realizing the control and management of the source device through the sink device HID.

Figure 241 Schematic diagram of HID pass-through function



The HID pass-through interaction can be divided into three stages: the start stage, the HID pass-through stage, and the end stage. The overall interaction process is shown in Figure 242.

Figure 242 Interaction process of HID pass-through



In the HID pass-through startup stage, the sink device initiates an HID pass-through request to the source device through the Start_HID_PassThrough message.

After receiving the Start_HID_PassThrough message, if the source device agrees to this HID PassThrough request, it allocates a session identifier (wSessionId) for this HID pass-through session and transmits the session identifier to the sink device through a response.

When the source device accepts this HID pass-through session, the sink device sends the local HID device information to the source device through the Add_HID_Device message. After receiving the Add_HID_Device message, the source device loads the corresponding input processing module (such as HID driver) according to the HID device information in the message body. Each Add_HID_Device message carries the information of an HID device. If the sink device has multiple HID devices, multiple Add_HID_Device messages are required for transmission. Different HID devices are distinguished by the bDeviceId assigned to the sink device.

The HID pass-through stage follows the HID pass-through startup stage. At this stage, the sink device can transparently transmit the input report and function report generated by the HID device to the source device, and the source device can also configure the status of the HID device through the output report, or set the functional status of the HID device through the function report. The HID reports of different HID devices are distinguished by bDeviceId.

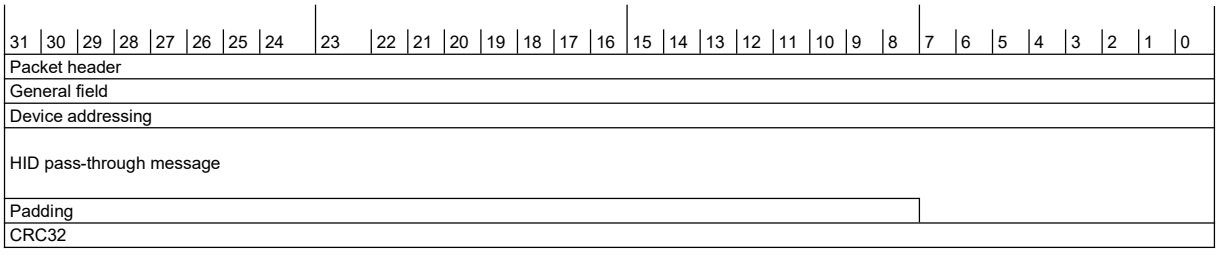
In the HID pass-through stage, the source device can communicate with the HID device in the sink device through GET_REPORT, SET_REPORT, GET_IDLE, SET_IDLE, GET_PROTOCOL and SET_PROTOCOL operations.

In addition, in the HID pass-through stage, when the sink device detects a new HID device, the new HID device can be added to the HID pass-through session through the Add_HID_Device message. When the sink device detects that a certain HID device has been removed, the HID device can be removed from the HID pass-through session through the Remove_HID_Device message.

When the sink device switches the active source device, or needs to end the HID PassThrough session with the source device for any reason, the sink device can notify the source device through the End_HID_PassThrough message. After receiving the End_HID_PassThrough message, the source device needs to log out of the HID device related to this HID pass-through session.

The HID pass-through message is transmitted through the address table addressing method of the management adapter. The management tunnel packet frame structure of HID pass-through is shown in Figure 243.

Figure 243 HID pass-through management tunnel message frame structure



- Packet header: ShuttleID is fixed to 0 (management adapter tunnel message), Type is fixed to 0x1D (HID pass-through message), and the meanings of other fields are described in 9.2.1.2 "Message Header."
- General fields: See 9.2.1.3 "General Fields."

- Device addressing: HID pass-through messages are addressed using an address table. See 9.2.2.1 "Table Addressing."
- HID pass-through message: carries the load of the HID pass-through message.
- Padding: Padding. The HID pass-through message is byte-aligned, while the management tunnel packet requires 4-byte alignment. When the total length of the HID pass-through message is not 4-byte aligned, the ending is padded with byte 0 to achieve 4-byte alignment (the number of padding bytes is not more than 3).
- CRC32: check information

This chapter will only describe the HID pass-through message in the following content. The message header, general fields, device addressing, padding, and CRC32 fields will not be detailed.

The HID pass-through message consists of two parts: the message header and the message body. The message header has a fixed 6 bytes, and its structure is shown in Table 177:

Table 177 Header structure of HID_PassThrough

Byte	Field	Number of Bytes	Type	Description
0	bDeviceId	1	BYTE	HID device identifier.
1	bOperation	1	BYTE	For HID pass-through message operation, see "HID_PassThrough Operation."
2	wSessionID	2	WORD	Session ID.
4	wLength	2	WORD	Message length, with a maximum of 5,120 bytes.

- bDeviceID: HID device identifier. The HID pass-through function supports pass-through of HID reports of multiple HID devices between the sink device and the source device. The HID device identification is used to identify HID reports generated by or sent to different HID devices. The HID device identifier is assigned by the sink device and synchronized to the source device via the Add_HID_Device message. The two messages Start_HID_PassThrough and End_HID_PassThrough have nothing to do with the HID device. In this case, fill 0 in the bDeviceID field.
- bOperation: HID pass-through message operation code, which is used to identify the operation type and operation code of the current HID pass-through message. For details, see Table 178 HID pass-through operation.
- wSessionID: session ID, assigned by the source device. For details, see 9.8.2 "Initiating HID Pass-Through." If a source device communicates with multiple sink devices, the source device can identify which sink device the HID report comes from through wSessionId, and then determine which HID device generated the HID report based on bDeviceId.
- wLength: length of the HID pass-through message, which includes the 6-byte header but does not include the padding bytes at the end. The maximum value shall not exceed 5,120.

Table 178 HID pass-through operation

Operation Type Higher 4 Bits	Operation Code Lower 4 Bits	Message Name	Source Device	Sink Device	Description
0x0	HID pass-through operation				
	0x0	Reserved			Reserved.
	0x1	Start_HID_PassThrough	M	M	Starts HID pass-through.
	0x2	Add_HID_Device	M	M	Adds an HID device.
	0x3	Remove_HID_Device	O	O	Removes an HID device.
	0x4	HID_Input_Report	M	M	HID input report
	0x5	HID_Output_Report	M	M	HID output report
	0x6	HID_Feature_Report	M	M	HID function report
	0x7	End_HID_PassThrough	M	M	Ends an HID pass-through.
	0x8–0xF	Reserved			Reserved.
0x1	GET_REPORT operation				
	0x0	GET_INPUT_REPORT	M	M	
	0x1	GET_OUTPUT_REPORT	M	M	
	0x2	GET_FEATURE_REPORT	M	M	
0x2	SET_REPORT operation				
	0x0	SET_INPUT_REPORT	M	M	
	0x1	SET_OUTPUT_REPORT	M	M	
	0x2	SET_FEATURE_REPORT	M	M	
0x3	0x0	GET_IDLE	O	O	
0x4	0x0	SET_IDLE	O	O	
0x5	0x0	GET_PROTOCOL	O	O	
0x6	0x0	SET_PROTOCOL	O	O	

9.8.2 Initiating HID Pass-Through

9.8.2.1 Introduction

When the sink device needs to start the HID pass-through function, it initiates a Start_HID_PassThrough message to the corresponding source device. After receiving the Start_HID_PassThrough message, the source device allocates wSessionID for the HID pass-through session and returns the wSessionID to the sink device through a response message.

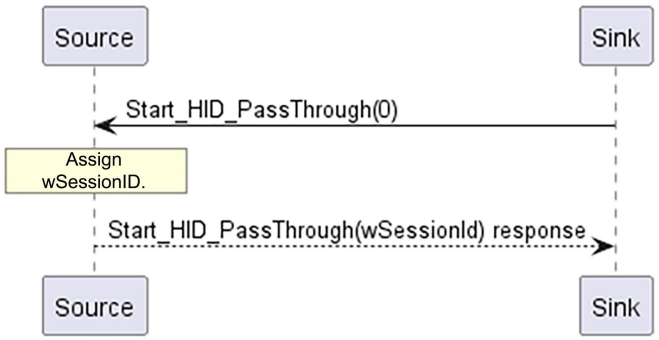
9.8.2.2 Processing Flow

The Start_HID_PassThrough message is initiated by the sink device. When the sink device needs to pass through the local HID device to a specific source device, it initiates a Start_HID_PassThrough request to the source device and carries the address of the device in the request message.

After the source device receives the Start_HID_PassThrough message, if this HID pass-through session is allowed, it assigns a wSessionID to it and returns the wSessionID to the sink device through the Start_HID_PassThrough response message. If the source device rejects this HID pass-through session, the wSessionID in the response message is set to 0.

After receiving the Start_HID_PassThrough response, the sink device checks whether wSessionID in the response message is 0. If wSessionID is 0, it indicates that the source device rejects the HID pass-through and the sink device ends this HID pass-through session. When the sink device receives a Start_HID_PassThrough response message in which wSessionID is not 0, it enters the next stage of HID pass-through.

Figure 244 HID pass-through process



Note:

- When the resources of the source device are insufficient, if the number of HID devices exceeds the limit of the operating system kernel, the HID pass-through session is rejected.
- A sink device can only initiate an HID pass-through session to a source device. If the sink device has initiated an HID pass-through session to a source device and needs to initiate an HID session to another source device, it must end the HID pass-through session with the old source device first.

9.8.2.3 Message Description

The Start_HID_PassThrough request message structure is detailed in Table 179.

Table 179 Structure of Start_HID_PassThrough request message

Byte	Field	Number of Bytes	Type	Description
0	bDeviceId	1	BYTE	HID device identifier. The HID device identifier of the Start_HID_PassThrough message is fixed to 0.
1	bOperation	1	BYTE	Operation code. The operation code to start HID pass-through is 0x01 (Start_HID_PassThrough).
2	wSessionID	2	WORD	Session ID. The wSessionID field in the Start_HID_PassThrough message sent by the sink device when starting HID pass-through is filled with 0. When the source device accepts this HID pass-through session, it allocates a wSessionID for the HID pass-through session and returns wSessionID to the sink device through a response message. If the source device rejects this HID pass-through session, it sets wSessionID in the response message to 0.
4	wLength	2	WORD	Message length. The HID PassThrough start message length is fixed to 14.
6	aSinkAddr	8	BYTE[]	Address of the sink device.

The structure of the Start_HID_PassThrough response message is shown in Table 180.

Table 180 Structure of Start_HID_PassThrough response message

Byte	Field	Number of Bytes	Type	Description
0	bDeviceId	1	BYTE	HID device identifier. The HID device identifier of the Start_HID_PassThrough message is fixed to 0.
1	bOperation	1	BYTE	Operation code. The operation code to start HID pass-through is 0x01 (Start_HID_PassThrough).
2	wSessionID	2	WORD	Session ID. The wSessionID field in the Start_HID_PassThrough message sent by the sink device when starting HID pass-through is filled with 0.

Byte	Field	Number of Bytes	Type	Description
				When the source device accepts this HID pass-through session, it allocates a wSessionID for the HID pass-through session and returns wSessionID to the sink device through a response message. If the source device rejects this HID pass-through session, it sets wSessionID in the response message to 0.
4	wLength	2	WORD	Message length. The HID PassThrough start message length is fixed to 14.
6	wValue	2	WORD	Return value: ● 0: indicates that wSessionID is successfully assigned. ● -1: indicates that the wSessionID allocation failed.

9.8.3 Adding an HID Device

9.8.3.1 Introduction

When the sink device needs to start HID report pass-through for a specific HID device, it transmits the identifier, device information, and report descriptor of the HID device to the source device through an Add_HID_Device (bDeviceID) message. The source device registers the HID input processing module (such as the HID driver) with the operating system according to the HID device information and report descriptor in the Add_HID_Device (bDeviceID) message.

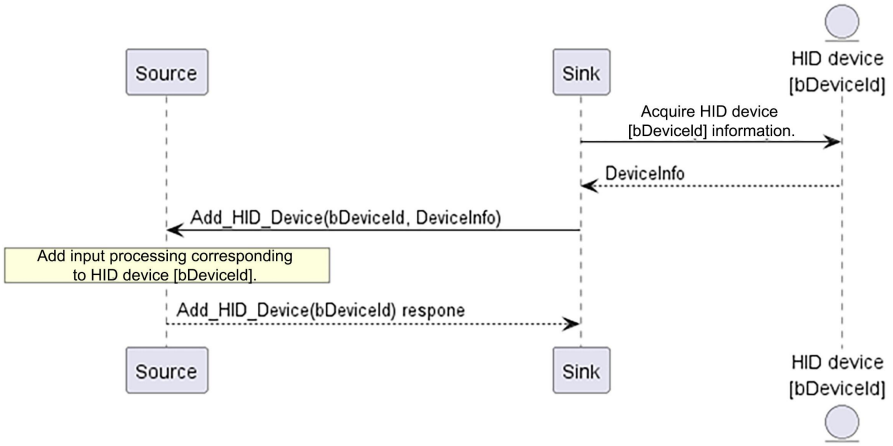
9.8.3.2 Processing Flow

After the sink device establishes an HID pass-through session through Start_HID_PassThrough, it sends its HID device information to the source device through the Add_HID_Device (bDeviceID) message. Wherein bDeviceID is the HID device identifier assigned by the sink device to the HID device at this end.

After receiving the Add_HID_Device (bDeviceID) message, the source device obtains the HID device information and the corresponding HID report descriptor from the message body, and registers the HID input processing with the operating system. When the source device successfully loads the HID input processing, it sends an Add_HID_Device (bDeviceID) response message to the sink device.

After the sink device receives the Add_HID_Device (bDeviceID) response message, it starts to pass through the HID report of the HID device corresponding to the bDeviceID to the source device.

Figure 245 Processing flow of adding an HID device



Note:

- An Add_HID_Device (bDeviceID) message carries only one HID device information. If the sink device has multiple HID devices to be passed through to the source device, multiple Add_HID_Device (bDeviceID) messages are required.
- In the HID pass-through stage, if the sink device detects that a new HID device is inserted, an HID report pass-through of the new HID device is added through the Add_HID_Device (bDeviceID) message.
- A maximum of 16 HID devices are supported for pass-through.

9.8.3.3 Message Description

The Add_HID_Device request message structure is detailed in Table 181.

Table 181 Add_HID_Device request message structure

Byte	Field	Number of Bytes	Type	Description
0	bDeviceId	1	BYTE	HID device identifier.
1	bOperation	1	BYTE	Operation code. The operation code for adding an HID device message is 0x02 (Add_HID_Device).
2	wSessionID	2	WORD	Session ID, negotiated by the Start_HID_PassThrough message
4	wLength	2	WORD	Message length Length of the Add_HID_Device message = Device name string length + Physical address string length + Serial number string length + Report descriptor length + 19

Byte	Field	Number of Bytes	Type	Description
6	wVendorId	2	WORD	Device supplier code Vendor ID of an HID device
8	wProductId	2	WORD	Product code This field is the product ID of an HID device, which identifies the model or product ID of the device.
10	wVersion	2	WORD	Version number of Binary-Coded Decimal (BCD) encoding The current version is fixed at 0x0100.
12	wUsagePage	2	WORD	
14	wUsage	2	WORD	
16	bNameLength	1	BYTE	The length of the device name string is less than or equal to 128.
17	cName		BYTE[]	Device name
	bPhysicalLength	1	BYTE	Physical address string length ≤ 64
	cPhysicalAddress		BYTE[]	Physical address string
	bSerialLength	1	BYTE	Serial number string length ≤ 64
	cSerial		BYTE[]	Serial number string
	wReportDescLength	2	WORD	HID report descriptor length ≤ 4,096
	wReportDescriptor		BYTE[]	HID report descriptor

The structure of the Add_HID_Device response message is detailed in Table 182.

Table 182 Add_HID_Device response message structure

Byte	Field	Number of Bytes	Type	Description
0	bDeviceId	1	BYTE	HID device identifier.
1	bOperation	1	BYTE	Operation code. The operation code for adding an HID device message is 0x02 (Add_HID_Device).
2	wSessionID	2	WORD	Session ID. It is the session ID negotiated by the Start_HID_PassThrough message.
4	wLength	2	WORD	Message length

Byte	Field	Number of Bytes	Type	Description
				The length of the Add_HID_Device response message is fixed to 8 bytes.
6	wValue	2	WORD	Return value: ● 0: indicates that a new HID device is successfully added. ● -1: indicates that a new HID device has failed to be added.

9.8.4 Removing an HID Device

9.8.4.1 Introduction

In the HID pass-through stage, if a certain HID device on the sink device is unplugged, the sink device informs the source device through the Remove_HID_Device(bDeviceId) message that the HID device corresponding to bDeviceID has been removed. After receiving the Remove_HID_Device(bDeviceId) message, the source device can uninstall the corresponding HID driver.

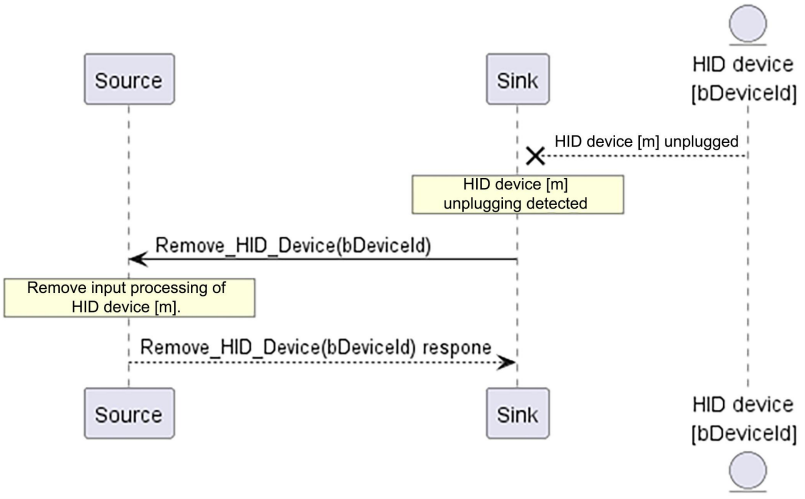
The Remove_HID_Device(bDeviceId) message is optional.

9.8.4.2 Processing Flow

If the sink device detects that the HID device is pulled out, or detects that a certain HID device fails, it initiates a Remove_HID_Device(bDeviceId) request to the source device.

After receiving the Remove_HID_Device(bDeviceId) request, the source device uninstalls the corresponding HID driver and then sends a Remove_HID_Device(bDeviceId) response message to the sink device.

Figure 246 Processing flow for removing an HID device



9.8.4.3 Message Description

The Remove_HID_Device request message structure is detailed in Table 183.

Table 183 Remove_HID_Device request message structure

Byte	Field	Number of Bytes	Type	Description
0	bDeviceId	1	BYTE	HID device identifier.
1	bOperation	1	BYTE	Operation code. The operation code for the Remove_HID_Device message is 0x03.
2	wSessionID	2	WORD	Session ID. Start_HID_PassThrough message negotiation session identifier.
4	wLength	2	WORD	Message length The length of the Remove_HID_Device message is fixed to 8.
6	wValue	2	WORD	Fixed to 0

The Remove_HID_Device response message structure is detailed in Table 184.

Table 184 Remove_HID_Device response message structure

Byte	Field	Number of Bytes	Type	Description
0	bDeviceId	1	BYTE	HID device identifier.
1	bOperation	1	BYTE	Operation code. The operation code for the Remove_HID_Device message is 0x03.
2	wSessionID	2	WORD	Session ID. It is the session ID negotiated by the Start_HID_PassThrough message.
4	wLength	2	WORD	Message length The length of the Remove_HID_Device message is fixed to 8.
6	wValue	2	WORD	Return value. 0 indicates that an HID device is removed successfully.

9.8.5 HID Input Report

9.8.5.1 Introduction

HID input reports typically travel from the HID device to the host, while in HID PassThrough, they are sent from the sink device to the source device.

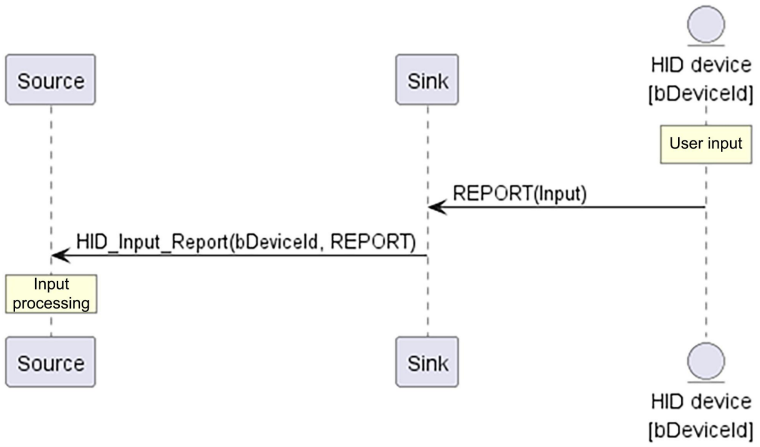
The input report is a data packet for communication between the HID device and the host system, transmitting instant input events generated by user interaction with the device.

9.8.5.2 Processing Flow

When the sink device receives the HID input report of the HID device corresponding to bDeviceId, the HID input report is sent to the source device through the HID_Input_Report(bDeviceId) message.

After receiving the HID_Input_Report(bDeviceId) message, the source device reports the HID input report in the message body to the operating system through the HID input processing corresponding to bDeviceId.

Figure 247 Processing flow of HID input reports



9.8.5.3 Message Description

The structure of the HID input report message is detailed in Table 185.

Table 185 Structure of HID input report message

Byte	Field	Number of Bytes	Type	Description
0	bDeviceId	1	BYTE	HID device identifier.
1	bOperation	1	BYTE	Operation code. The operation code of the HID input report message is 0x04 (HID_Input_Report).
2	wSessionID	2	WORD	Session ID.

Byte	Field	Number of Bytes	Type	Description
				It is the session ID negotiated by the Start_HID_PassThrough message.
4	wLength	2	WORD	Message length Its value is the actual HID input report data length plus 6.
6	aData		BYTE[]	Input report data.

HID input report messages require no response message.

9.8.6 HID Output Report

9.8.6.1 Introduction

Output reports are generated on the host, traveling from the host to the HID device. In HID pass-through, the output report is typically sent from source device to sink device.

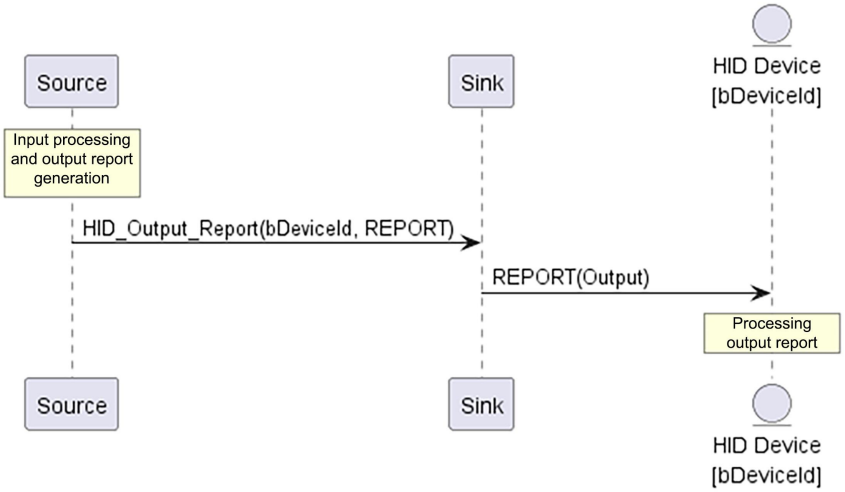
Output reports are data packets sent by the host system to the HID device to control device status, update display information, and trigger specific actions.

9.8.6.2 Processing Flow

After receiving the output report from the HID input processing corresponding to bDeviceId, the source device sends it to the sink device through the HID_Output_Report(bDeviceId) message.

The sink device receives the HID_Output_Report(bDeviceId) message and sends the HID output report in the message body to the HID device corresponding to bDeviceId.

Figure 248 Processing flow of HID output report



9.8.6.3 Message Description

The structure of the HID output report message is detailed in Table 186.

Table 186 Structure of HID output report message

Byte	Field	Number of Bytes	Type	Description
0	bDeviceId	1	BYTE	HID device identifier.
1	bOperation	1	BYTE	Operation code. The operation code of the HID output report message is 0x05 (HID_Output_Report).
2	wSessionID	2	WORD	Session ID. It is the session ID negotiated by the Start_HID_PassThrough message.
4	wLength	2	WORD	Message length Its value is the actual HID output report data length plus 6.
6	aData		BYTE[]	Output report data.

HID output report messages require no response message.

9.8.7 HID Feature Report

9.8.7.1 Introduction

The Feature Report is a two-way report, which can be sent from the HID device to the host or from the host to the HID device.

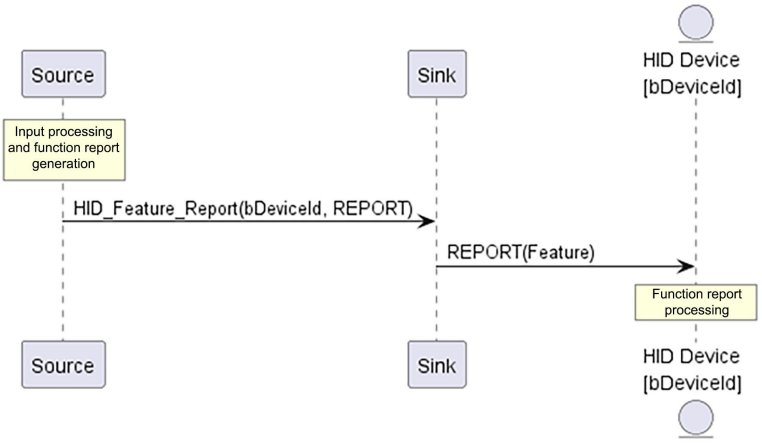
Feature reports are mainly used in computer systems to transmit data related to the device functional status.

9.8.7.2 Processing Flow

The feature report can be sent from the HID device of the source device to that of the sink device, or from the HID device of the sink device to that of the source device.

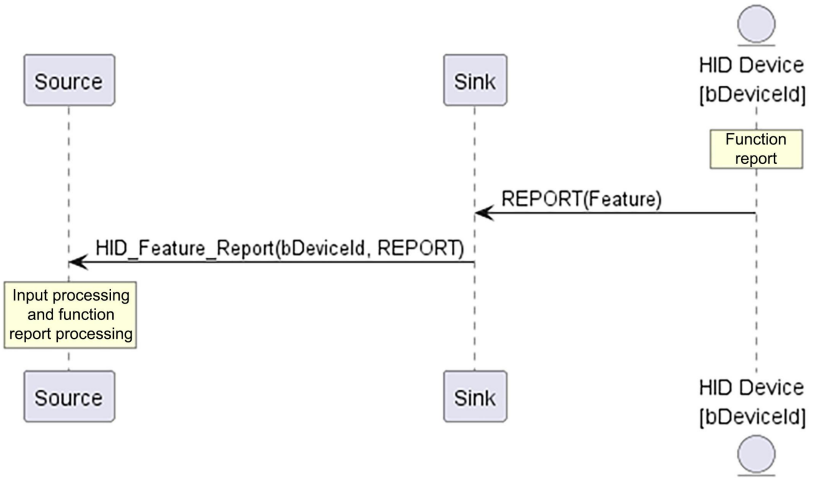
The feature report processing from the source device to the sink device is detailed in Figure 249.

Figure 249 Processing flow of HID feature report (from source device to sink device)



The feature report processing from the sink device to the source device is detailed in Figure 250.

Figure 250 Processing flow of HID feature report (from sink device to source device)



9.8.7.3 Message Description

The structure of HID feature report message is detailed in Table 187.

Table 187 Structure of HID feature report message

Byte	Field	Number of Bytes	Type	Description
0	bDeviceId	1	BYTE	HID device identifier.
1	bOperation	1	BYTE	Operation code. The operation code of the HID feature report message is 0x06 (HID_Feature_Report).

Byte	Field	Number of Bytes	Type	Description
2	wSessionID	2	WORD	Session ID, negotiated by the Start_HID_PassThrough message
4	wLength	2	WORD	Message length, whose value is the actual HID feature report data length plus 6.
6	aData		BYTE[]	Feature report data.

HID feature report messages require no response message.

9.8.8 Ending HID Pass-Through

9.8.8.1 Introduction

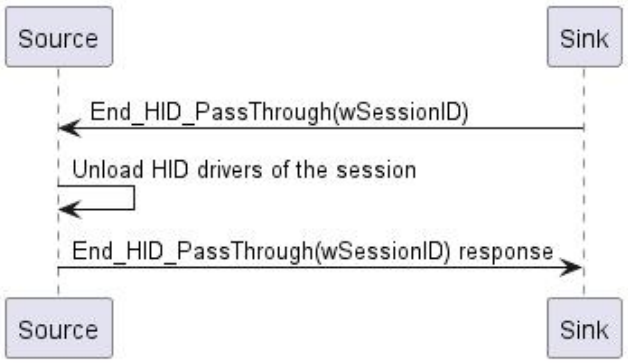
After the sink device switches the video source, the HID PassThrough function of the original video source device needs to be disabled. At this time, it is necessary to initiate an End_HID_PassThrough message to the source device.

9.8.8.2 Processing Flow

The sink device notifies the source device of the ending of the HID PassThrough function through the End_HID_PassThrough(wSessionId) message.

After receiving the End_HID_PassThrough(wSessionId) message, the source device uninstalls the HID input processing loaded by the HID PassThrough function, and then sends an End_HID_PassThrough(wSessionId) response message to the sink device.

Figure 251 Processing flow of ending HID pass-through



9.8.8.3 Message Description

The End_HID_PassThrough message structure is detailed in Table 188.

Table 188 End_HID_PassThrough request message structure

Byte	Field	Number of Bytes	Type	Description
0	bDeviceId	1	BYTE	HID device identifier.
1	bOperation	1	BYTE	Operation code. The operation code to end HID pass-through is 0x07 (End_HID_PassThrough).
2	wSessionID	2	WORD	Session ID. It is the session ID negotiated by the Start_HID_PassThrough message.
4	wLength	2	WORD	Message length This message has no message body, but only a message relationship. The message length is fixed to 6.

The End_HID_PassThrough response message structure is detailed in Table 189.

Table 189 End_HID_PassThrough response message structure

Byte	Field	Number of Bytes	Type	Description
0	bDeviceId	1	BYTE	HID device identifier.
1	bOperation	1	BYTE	Operation code. The operation code to end HID pass-through is 0x07 (End_HID_PassThrough).
2	wSessionID	2	WORD	Session ID. It is the session ID negotiated by the Start_HID_PassThrough message.
4	wLength	2	WORD	Message length This message has no message body, but only a message relationship. The message length is fixed to 6.
6	wValue	2	WORD	Return value. 0 indicates that End_HID_PassThrough is processed successfully. -1 indicates that End_HID_PassThrough processing failed.

9.8.9 GET_REPORT Message

9.8.9.1 Introduction

The GET_REPORT message is a control transmission request from the host to the HID device for specific report data.

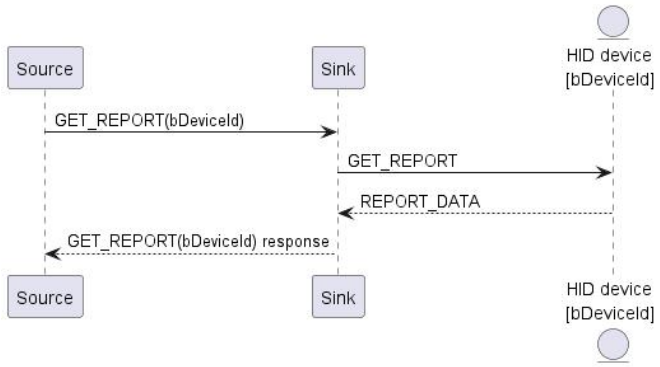
9.8.9.2 Processing Flow

The source device initiates a GET_REPORT(bDeviceId) request to the sink device.

The sink device receives the GET_REPORT(bDeviceId) request and initiates a GET REPORT request to the HID device corresponding to bDeviceId.

After receiving the GET_REPORT response from the HID device, the sink device returns the HID report returned by the HID device to the source device through a GET_REPORT (bDeviceId) response message.

Figure 252 Processing flow of GET_REPORT message



9.8.9.3 Information Description

The GET_REPORT message structure is detailed in Table 190.

Table 190 GET_REPORT message structure

Byte	Field	Number of Bytes	Type	Description
0	bDeviceId	1	BYTE	HID device identifier.
1	bOperation	1	BYTE	Operation. The higher four bits of the operation code indicate the operation type GET_REPORT, which is fixed to 0x1. The lower 4 bits indicate the report type obtained. The value 0 indicates the input report; value 1 indicates the output report; and value 2 indicates the feature report. 0x10: obtains the input report. 0x11: obtains the output report.

Byte	Field	Number of Bytes	Type	Description
				0x12: obtains the feature report. Others: reserved
2	wSessionID	2	WORD	Session ID. It is the session ID negotiated by the Start_HID_PassThrough message.
4	wLength	2	WORD	Message length
6	bReportId	1	BYTE	The first byte of the payload portion of GET_REPORT must be ReportId for the report to be obtained. If the HID device does not use a numbered report, the first byte is padded with 0.
7	aData		BYTE[]	Buffer for storing reports to be obtained.

Note:

- The GET_REPORT message is a mandatory request. Both the source and sink devices shall support the message.
- The GET_REPORT message is typically used to obtain a feature report and an input report of the HID device from the sink device. If the sink device receives a GET_REPORT request from the output report of the source device, it shall directly respond to an invalid GET_REPORT request and shall not initiate a GET_REPORT request to the HID device.
- The request message and the response message of GET_REPORT have the same structure.
- The first byte in the body of the GET_REPORT message sent from the source device to the sink device must be ReportId. If the target HID device uses an unnumbered HID report, this byte is padded with 0.
- The aData field in the GET_REPORT message body sent from the source device to the sink device serves as a receive buffer for reports and carries no meaning. After receiving the SET_REPORT message, the sink device initiates a GET_REPORT request to the HID device, fills in the field according to the actual report information returned by the HID device, and then returns it to the source device.

9.8.10 SET_REPORT Message

9.8.10.1 Introduction

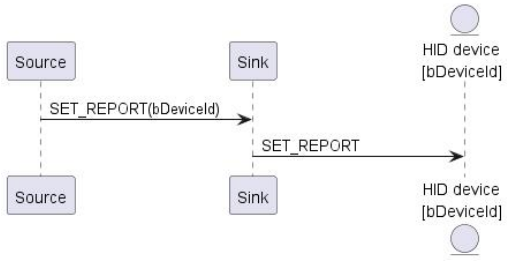
The SET_REPORT message is a control transmission request from the host to send specific report data.

9.8.10.2 Processing Flow

The source device initiates a SET_REPORT(bDeviceId) request to the sink device. The message body includes receiving an HID report from the HID input process.

The sink device receives the SET_REPORT(bDeviceId) request, obtains the HID report from the message body; and then initiates a SET REPORT request to the HID device corresponding to bDeviceId.

Figure 253 Processing flow of SET_REPORT message



9.8.10.3 Message Description

The SET_REPORT message structure is detailed in Table 191.

Table 191 SET_REPORT message structure

Byte	Field	Number of Bytes	Type	Description
0	bDeviceId	1	BYTE	HID device identifier.
1	bOperation	1	BYTE	Operation. The higher 4 bits of the operation code indicate the operation type SET_REPORT, which is fixed to 0x2. The lower 4 bits indicate the report type obtained. The value 0 indicates the input report; value 1 indicates the output report; and value 2 indicates the feature report. 0x20: sets the input report. 0x21: sets the output report. 0x22: sets the feature report. Others: reserved
2	wSessionID	2	WORD	Session ID. It is the session ID negotiated by the Start_HID_PassThrough message.
4	wLength	2	WORD	Message length
6	bReportId	1	BYTE	The first byte of the payload portion of SET_REPORT must be ReportId for the report to be set. If the HID device does not use a numbered report, the first byte is padded with 0.
7	aData		BYTE[]	Report data.

Note:

- The SET_REPORT message is a mandatory request. Both the source and sink devices shall support processing the message.
- The SET_REPORT message is typically used to set the feature report and output report of the HID device. The sink device shall ignore the SET_REPORT request if it receives one for the input report.
- The SET_REPORT message does not require a response and has no response message structure.

9.8.11 GET_IDLE Message

9.8.11.1 Introduction

The GET_IDLE message is a control transmission request from the host to the HID device for the current input idle state.

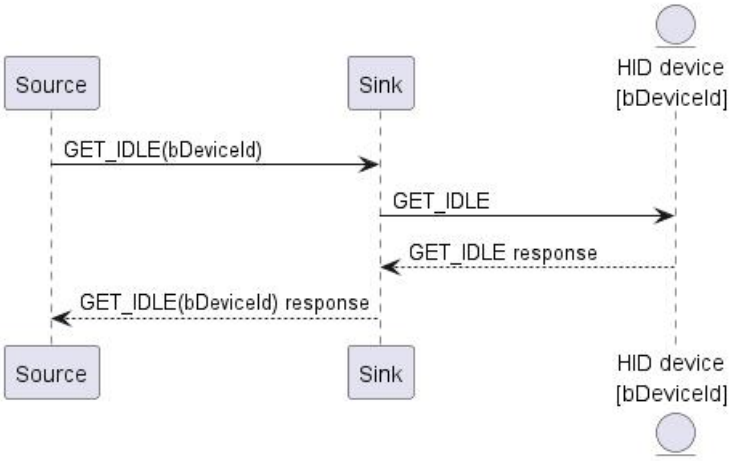
9.8.11.2 Processing Flow

After receiving the GET IDLE request from the HID input processing corresponding to bDeviceId, the source device initiates a GET_IDLE(bDeviceId) request to the sink device.

The sink device receives the GET_IDLE(bDeviceId) request and initiates a GET IDLE request to the HID device corresponding to bDeviceId. The bIdleRate returned by the HID device is then returned to the source device via a GET_IDLE(bDeviceId) response message.

After receiving the GET_IDLE(bDeviceId) response message, the source device returns bIdleRage in the message body to the bDeviceId corresponding HID input processing.

Figure 254 Processing flow of GET_IDLE message



9.8.11.3 Message Description

The GET_IDLE message structure is detailed in Table 192.

Table 192 GET_IDLE message structure

Byte	Field	Number of Bytes	Type	Description
0	bDeviceId	1	BYTE	HID device identifier.
1	bOperation	1	BYTE	Operation. The value of GET_IDLE is fixed to 0x30.
2	wSessionID	2	WORD	Session ID. It is the session ID negotiated by the Start_HID_PassThrough message.
4	wLength	2	WORD	Message length The value is fixed at 8.
6	bReportId	1	BYTE	The first byte of the payload portion of GET_IDLE must be ReportId to be queried. If the HID device does not use a numbered report, the first byte is padded with 0.
7	bIdleRate	1	BYTE	The idle time of the current specified input report to be returned For example, if the device has not received any input event for 4 milliseconds constantly, the return value is 0x01; if it has not received an input event for 8 milliseconds constantly, the return value is 0x02, and so on. If the device is always in active state (constantly receiving input events), the return value is 0x00.

Note:

- The request message and the response message of GET_IDLE have the same structure.
- The first byte in the body of the GET_IDLE message sent from the source device to the sink device must be ReportId. If the target HID device uses an unnumbered HID report, this byte is padded with 0.
- The bIdleRate field in the GET_IDLE message body sent from the source device to the sink device serves as a receive buffer for reports and carries no meaning. After receiving the GET_IDLE message, the sink device initiates a GET_IDLE request to the HID device, fills the field according to the actual bIdleRage returned by the HID device, and then returns it to the source device.

9.8.12 SET_IDLE Message

9.8.12.1 Introduction

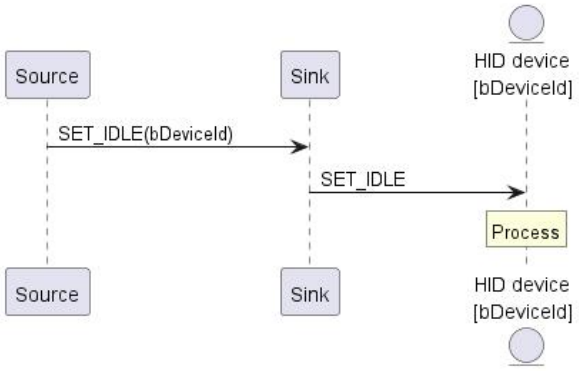
The SET_IDLE message is a request sent by the host to the HID device to set an idle state for a specified input report (that is, the time the device remains silent before receiving a user input event).

9.8.12.2 Processing Flow

After receiving the SET IDLE request from the HID input processing corresponding to bDeviceId, the source device initiates a SET_IDLE(bDeviceId) request to the sink device.

The sink device receives the SET_IDLE(bDeviceId) request, obtains bIdleRate from the message body; and then initiates a SET IDLE request to the HID device corresponding to bDeviceId.

Figure 255 Processing flow of SET_IDLE message



9.8.12.3 Message Description

The SET_IDLE message structure is detailed in Table 193.

Table 193 SET_IDLE message structure

Byte	Field	Number of Bytes	Type	Description
0	bDeviceId	1	BYTE	HID device identifier.
1	bOperation	1	BYTE	Operation. The value of SET_IDLE is fixed to 0x40.
2	wSessionID	2	WORD	Session ID. It is the session ID negotiated by the Start_HID_PassThrough message.
4	wLength	2	WORD	Message length The value is fixed at 8.
6	bReportId	1	BYTE	The first byte of the payload portion of SET_IDLE must be ReportId for the report to be set. If the HID device does not use a numbered report, the first byte is padded with 0.
7	bIdleRate	1	BYTE	Specifies the idle time for input reports. One byte, which indicates the time to remain idle (unit: 4 milliseconds). For example, to set the HID device to remain silent for 8 milliseconds before receiving the next input

Byte	Field	Number of Bytes	Type	Description
				event, set this field to 0x02.

SET_IDLE messages require no response message.

9.8.13 GET_PROTOCOL Message

9.8.13.1 Introduction

The GET_PROTOCOL message is a control transmission request from the host to the HID device for the current transmission protocol status.

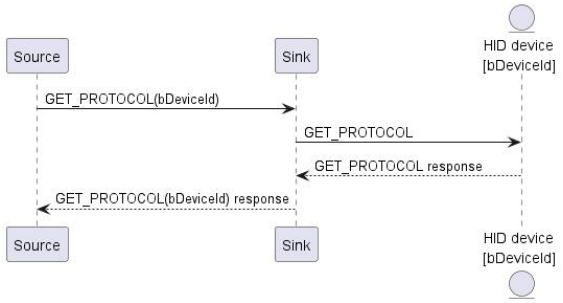
9.8.13.2 Processing Flow

After receiving the GET PROTOCOL request from the HID input processing corresponding to bDeviceId, the source device initiates a GET_PROTOCOL(bDeviceId) request to the sink device.

The sink device receives the GET_PROTOCOL(bDeviceId) request and initiates a GET_PROTOCOL request to the HID device corresponding to bDeviceId. The bProtocol returned by the HID device is then returned to the source device via a GET_IDLE(bDeviceId) response message.

After receiving the GET_PROTOCOL(bDeviceId) response message, the source device returns bProtocol in the message body to the bDeviceId corresponding HID input processing.

Figure 256 Processing flow of GET_PROTOCOL message



9.8.13.3 Message Description

The GET_PROTOCOL message structure is shown in Table 194.

Table 194 GET_PROTOCOL message structure

Byte	Field	Number of Bytes	Type	Description
0	bDeviceId	1	BYTE	HID device identifier.
1	bOperation	1	BYTE	Operation.

Byte	Field	Number of Bytes	Type	Description
				The value of GET_PROTOCOL is fixed to 0x50.
2	wSessionID	2	WORD	Session ID. It is the session ID negotiated by the Start_HID_PassThrough message.
4	wLength	2	WORD	Message length The value is fixed at 7.
6	bProtocol	1	BYTE	Stores the return value, which is the previous transmission protocol type: 0: Boot Protocol 1: Report Protocol

Note:

- The request message and the response message of GET_PROTOCOL have the same structure.
- In the GET_PROTOCOL message sent from the source device to the sink device, the bProtocol field in the message body is the protocol type expected for the HID input processing of the source device.

9.8.14 SET_PROTOCOL Message

9.8.14.1 Introduction

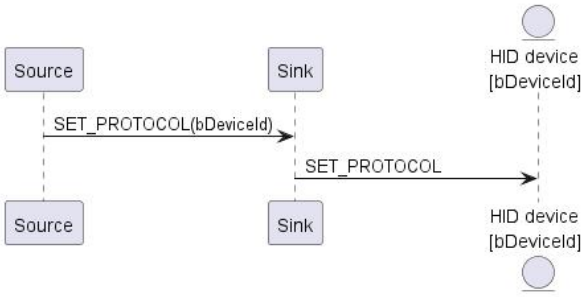
The SET_PROTOCOL message is a request sent by the host to the HID device for setting the HID transmission protocol (Boot Protocol or Report Protocol) currently used by the HID device.

9.8.14.2 Processing Flow

After receiving the SET_PROTOCOL request from the HID input process corresponding to bDeviceId, the source device parses bProtocol to be set and then initiates a SET_PROTOCOL (bDeviceId) request to the sink device.

The sink device receives the SET_PROTOCOL(bDeviceId) request, obtains bProtocol from the message body; and then initiates a SET_PROTOCOL request to the HID device corresponding to bDeviceId.

Figure 257 Processing flow of SET_PROTOCOL message



9.8.14.3 Message Description

The SET_PROTOCOL message structure is shown in Table 195.

Table 195 SET_PROTOCOL message structure

Byte	Field	Number of Bytes	Type	Description
0	bDeviceId	1	BYTE	HID device identifier.
1	bOperation	1	BYTE	Operation. The value of SET_PROTOCOL is fixed to 0x60.
2	wSessionID	2	WORD	Session ID. It is the session ID negotiated by the Start_HID_PassThrough message.
4	wLength	2	WORD	Message length The value is fixed at 7.
6	bProtocol	1	BYTE	Transmission protocol type: 0: Boot Protocol 1: Report Protocol

SET_PROTOCOL messages require no response message.

Appendix A (Informative) Reference Design of Electrical Layer

A.1 Lane Structure

Figures A.1 and A.2 show the circuit structure of an eight-lane port design. Figure A.1 illustrates the lane connection when the connector is in forward insertion, and Figure A.2 illustrates the lane connection when the connector is in reverse insertion.

The red dotted box shows the circuit structure of a four-lane design.

Figure A.1 Eight-lane structure in forward insertion

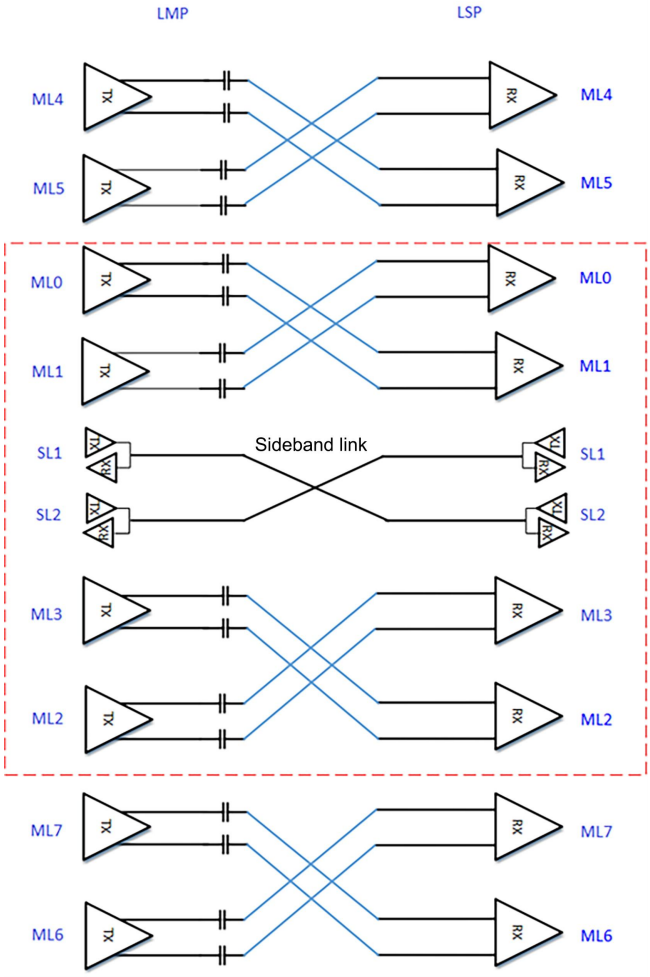
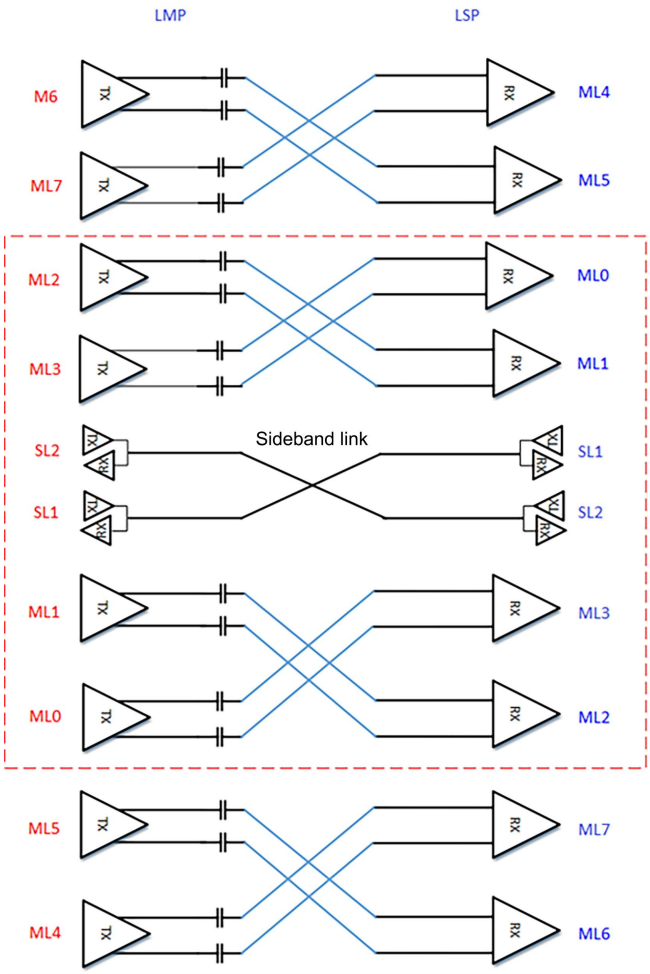


Figure A.2 Eight-lane structure in reverse insertion

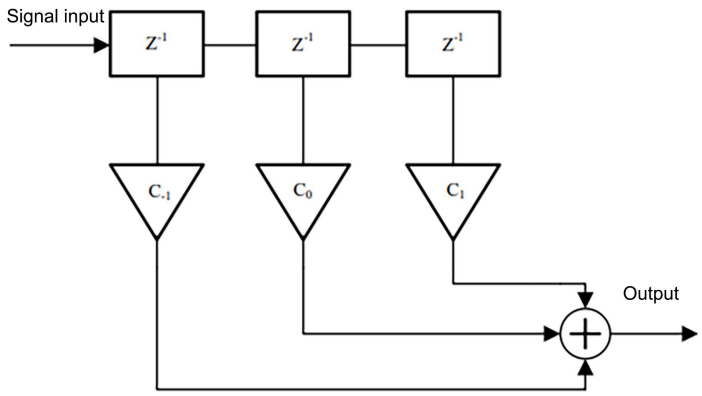


A.2 Equilibrium Settings

A.2.1 Reference TX Equalizer and Preset Gear

This section describes the reference equalizer model used by the TX, which supports a configurable FeedForward Equalizer (FFE) to compensate for lane frequency selective attenuation and improve link performance.

Figure A.3 FFE architecture of the TX



Where,

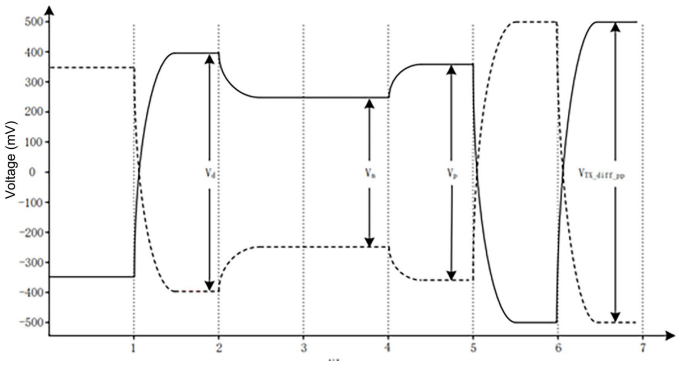
- Z^{-1} represents a delay unit through which a data delay of 1 UI is generated.
- C*n* is the weighting coefficient (tap coefficient), and the subscript *n* indicates the position relative to the current time. The weighting coefficient shall satisfy $0 < C_0 < 1$, $C_{-1} \leq 0$, $C_1 \leq 0$, and $C_0 - C_{-1} - C_1 = 1$.

The TX output waveform modified by the 3rd-order FFE has both de-emphasis and preshoot effects. The strength of de-emphasis and preshoot can be calculated by the following formula:

$$\text{Preshoot(dB)} = 20 \cdot log_{10}\frac{Vp}{Vn} = 20 \cdot log_{10}\frac{-C_{-1}+C_0+C_1}{C_{-1}+C_0+C_1} \quad \text{(A.1)}$$

$$\text{De} - \text{emphases(dB)} = 20 \cdot log_{10}\frac{Vn}{Vd} = 20 \cdot log_{10}\frac{C_{-1}+C_0+C_1}{C_{-1}+C_0-C_1} \quad \text{(A.2)}$$

Figure A.4 Reference waveform of 3rd-order TX equalizer



In the normal amplitude mode, the output waveform of the TX is shown in Figure A.4, and four voltage amplitudes exist. In the figure, $V_{TX\_diff\_pp}$ indicates the peak-to-peak value of the differential signal output by the TX; V*n* indicates the differential voltage value under the normal state; V*p* indicates the differential voltage value under the preshoot effect; and V*d* indicates the differential voltage value under the de-emphasis effect. The four voltage amplitudes shown in Figure A.4and the weighting coefficients in Figure A.3 satisfy the following:

$$V_n = (C_{-1} + C_0 + C_1) \cdot V_{\text{Tx\_diff\_pp}} \quad \text{(A.3)}$$

$V_p = (-C_{-1} + C_0 + C_1) \cdot V_{\text{Tx\_diff\_pp}}$ ..... (A.4)

$V_d = (C_{-1} + C_0 - C_1) \cdot V_{\text{Tx\_diff\_pp}}$ ..... (A.5)

$C_0 - C_{-1} - C_1 = 1$..... (A.6)

The main link electrical layer shall provide multiple FFE preset gears for reference based on the actual cable attenuation. Each preset gear needs to specify the following values: C_{-1}, C_0, C_1, Preshoot, and De-emphasis, where the units of Preshoot and De-emphasis are dB. Examples are shown in Table A.1.

Table A.1 Example of preset equilibrium gear at the TX

Preset Gear	C_{-1}	C_0	C_1	Preshoot (dB)	Deemphasis (dB)
TxFFE0	-0.05	0.8	-0.15	1.34	-3.52
TxFFE1	-0.05	0.90	-0.05	1	-1
TxFFE2	-0.05	0.87	-0.08	1.1	-1.7
TxFFE3	-0.075	0.85	-0.075	1.69	-1.69
TxFFE4	-0.1	0.8	-0.1	2.5	-2.5

A.2.2 Reference RX Equilibrium

A.2.2.1 Reference Continuous-Time Linear Equalizer

The main function of the Reference Continuous-Time Linear Equalizer (CTLE) is to compensate for the frequency selective attenuation of lanes. This section serves as a reference equilibrium to measure the TX signal eye diagram indicators.

CTLE transfer function:

$\mathrm{H(s)} = A_G \omega_{p2} \frac{s+\omega_{p1}/A_G}{(s+\omega_{p1})(s+\omega_{p2})}$.....(A.7)

In the formula:

S indicates the complex frequency of the Laplace transform;

ω_{p1} indicates the first pole, in rad/s;

ω_{p2} indicates the second pole, in rad/s;

A_G indicates the gain ratio.

The electrical layer shall specify the first pole (ω_{p1}), second pole (ω_{p2}), gain ratio (A_G), and peak gain at different rates. Examples are shown in Table A.2.

Table A.2 Reference CTLE parameters

Data Rate	First Pole ω_{p1} (rad/s)	Second Pole ω_{p2} (rad/s)	Gain Ratio A_G	Peak Gain (dB)

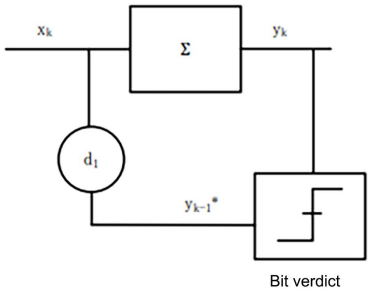
Data Rate	First Pole ω_{p1} (rad/s)	Second Pole ω_{p2} (rad/s)	Gain Ratio A_G	Peak Gain (dB)
6Gbps	$2\pi \cdot 2G$	$2\pi \cdot 5G$	1.8	2.8
			2.3	4.7
			2.7	6.0
			3.2	7.4
			3.9	9.0
			4.6	10.4
			5.5	12.0
8Gbps	$2\pi \cdot 3G$	$2\pi \cdot 6G$	2	3.1
			2.5	4.8
			2.9	6.0
			3.5	7.5
			4.2	9.1
			5	10.6
			6	12.1
10Gbps	$2\pi \cdot 4G$	$2\pi \cdot 7G$	2	2.8
			2.5	4.5
			3	5.9
			3.6	7.4
			4.3	8.9
			5.2	10.5
			6.2	12
12Gbps	$2\pi \cdot 5G$	$2\pi \cdot 8G$	2.1	2.9
			2.6	4.5
			3.1	5.9
			3.8	7.6
			4.5	9.0
			5.4	10.5
			6.4	12.0
16Gbps	$2\pi \cdot 7G$	$2\pi \cdot 10G$	2.2	2.9
			2.7	4.5

Data Rate	First Pole ω_{p1} (rad/s)	Second Pole ω_{p2} (rad/s)	Gain Ratio A_G	Peak Gain (dB)
			3.3	6.1
			4.0	7.6
			4.7	9.0
			5.7	10.6
			6.7	12.0
20Gbps	$2\pi \cdot 9G$	$2\pi \cdot 12G$	2.3	3.0
			2.8	4.5
			3.4	6.1
			4.1	7.6
			4.9	9.1
			5.8	10.5
			6.9	12.0
24Gbps	$2\pi \cdot 11G$	$2\pi \cdot 14G$	2.7	4.1
			3.3	5.6
			3.9	7.0
			4.7	8.6
			5.6	10.0
			7	12.0
			8	13.1

A.2.2.2 Reference Decision-Feedback Equalizer

Under high-speed transmission, the RX should be cascaded with a Decision-Feedback Equalizer (DFE) after CTLE equilibrium. Figure A.5 demonstrates a reference DFE with single-tap structure to standardize TX signal test. The equalizer of an actual RX device may not be designed in compliance with this parameter.

Figure A.5 Schematic diagram of a single-tap DFE



In the figure:

- y_k indicates the differential output signal of DFE;
- $y_{k-1}*$ indicates the output signal decision result, which is +1 or -1;
- d_1 indicates the feedback coefficient in mV;
- x_k indicates the input differential signal of DFE;
- k and k − 1 indicate the time indexes with 1 UI as the time unit.

The electrical layer shall specify the feedback factor d1 at different rates. d_1 should be 0−50 mV at data rates of 10 Gbps, 12 Gbps, 16 Gbps, 20 Gbps, and 24 Gbps.

Appendix B (Normative) Electrical Layer System Configurations

B.1 Overview

This section mainly describes the test point definition of the main link lane, impedance matching requirements in test, reference RX equilibrium settings, and jitter transfer function to provide a unified test environment for the main link TX and RX.

The test point defines the location for electrical parameter measurements. Impedance matching specifies the requirements for connection impedance and terminating resistor impedance during electrical parameter testing. Reference RX equilibrium settings define the equilibrium criteria required for TX and RX compliance testing.

The TX eye diagram and jitter index can only be tested after passing the jitter transfer function provided in this section.

B.2 Test Point

The test point (TP) is defined in Figure B.1. For details, see Table B.1.

Figure B.1 Definition of the test point

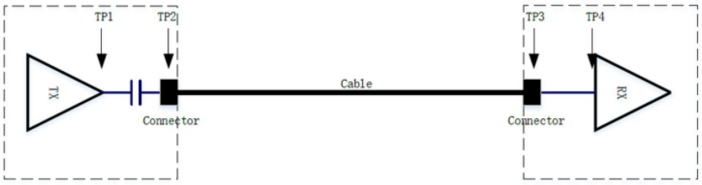


Table B.1 Description of the test point

No.	Description of Test Point	Technical Requirements
TP1	TX chip output point	This standard does not require the electrical characteristics of this test point.
TP2	TX connector output point	Measured on the male connector.
TP3	Output point from connector to RX	Measured on the female connector. Generally, the TP3 test results need to be processed through the CTLE or CTLE plus DFE settings of the instrument. Therefore, the subsequent test point referred to as TP3_EQ follows the same definition as TP3.
TP4	RX chip input point	This standard does not require the electrical characteristics of this test point.

B.3 Impedance Matching

The TX and RX should meet the impedance matching requirements during testing. Impedance is measured by time domain reflectometry (TDR). When measuring impedance, the connector should be included. Errors caused by the test fixture shall be compensated and shall not affect the test results.

TDR rise time, connection impedance, and terminating resistor impedance shall meet the technical requirements described in Table B.2.

Table B.2 Impedance requirements

Parameter Name	Value	Unit	Remarks
TDR rise time Note 1	<= 200	ps	
Connection impedance Note 2	80–100	Ω	
Terminating resistor impedance	701–10	Ω	
Note 1: The test point is TP2 for the TX and TP3 for the RX. Note 2: For the TX, it means the impedance of the TP2 to TX terminating resistor path; for the RX, it means the impedance of the TP3 to RX terminating resistor path. Note 3: A maximum of one impedance exception within a 150 ps window is permitted, limited to			

Parameter Name	Value	Unit	Remarks
the range of 90 Ω ±25%.			

B.4 Reference RX Equilibrium Settings

When testing TX signals, the reference RX equilibrium settings should be specified. When testing the TX signal in accordance with this standard, for data rates of 2 Gbps and 4 Gbps, measurements shall be performed at test point TP2 and no equilibrium shall be applied. At 6 Gbps and 8 Gbps, CTLE equilibrium shall be configured. For rates of 10 Gbps, 12 Gbps, 16 Gbps, 20 Gbps, and 24 Gbps, CTLE + DFE (optional) equilibrium shall be configured. See for description of the reference RX equilibrium.

B.5 Jitter Transfer Function

This standard provides a reference jitter transfer function (JTF). The eye diagram and jitter indicators of all TX signals shall be measured after the transmission signal passes through this model.

Figure B.2 shows the simplified clock and data recovery (CDR) circuit model. The closed-loop transfer function of the reference CDR is as follows:

$$H(\mathrm{s}) = \frac{2\zeta w_n s + {w_n}^2}{s^2 + 2\zeta w_n s + {w_n}^2} \quad \text{(B.1)}$$

The corresponding jitter transfer function is as follows:

$$JTF(\mathrm{s}) = 1 - \mathrm{H}(\mathrm{s}) = \frac{s^2}{s^2 + 2\zeta w_n s + {w_n}^2} \quad \text{(B.2)}$$

where,

ζ is the damping coefficient;

ωn is the natural angular frequency of the circuit system, in rad/s;

s is the complex frequency of the Laplace transform, in Hz.

Table B.3 specifies the technical requirements for CDR 3dB bandwidth ($f_{CDR\text{-}3dB}$), damping factor (ζ), and jitter transfer function 3dB bandwidth (*f*JTF-3dB) at various rates, while Figures B.3–B.6 specify the technical requirements that the jitter transfer function shall meet.

Figure B.2 Simplified CDR model

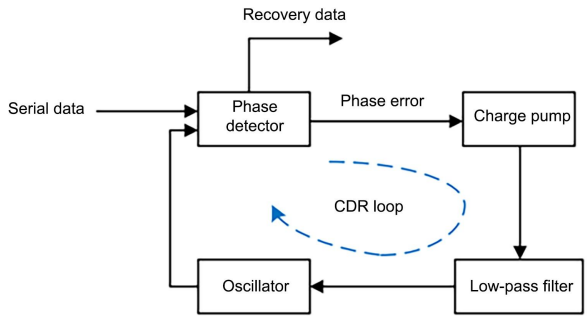


Table B.3 JTF parameters at various rates

Rate	CDR 3dB Bandwidth	Damping Factor	JTF 3dB Bandwidth
2/4 Gbps	4 MHz	1	2.51 MHz
6/8 Gbps	6 MHz	1	3.76 MHz
10/12/16 Gbps	8 MHz	1	5.02 MHz
20/24 Gbps	12 MHz	1	7.53 MHz

Figure B.3 Reference CDR transfer function and JTF at 2 Gbps and 4 Gbps

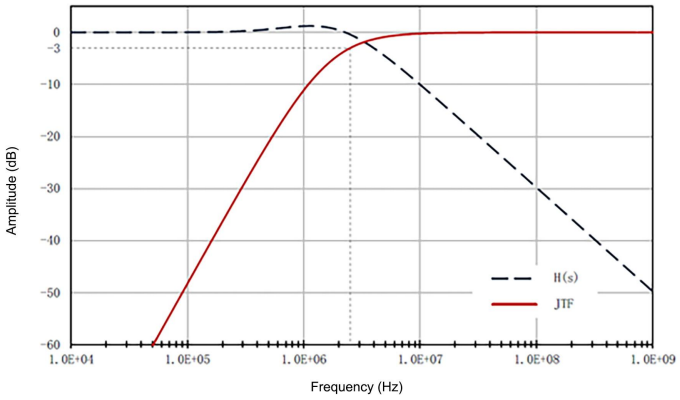


Figure B.4 Reference CDR transfer function and JTF at 6 Gbps and 8 Gbps

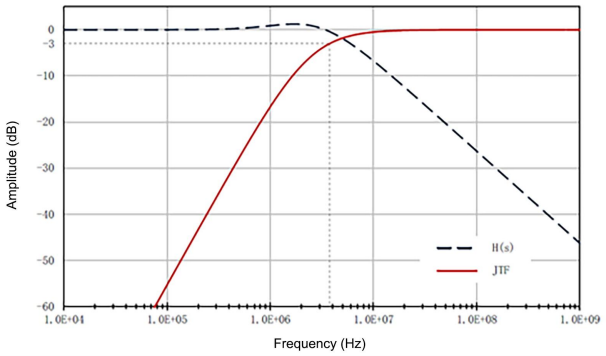


Figure B.5 Reference CDR transfer function and JTF at 10 Gbps, 12 Gbps, and 16 Gbps

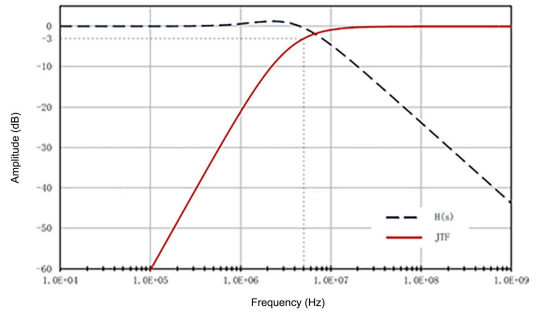
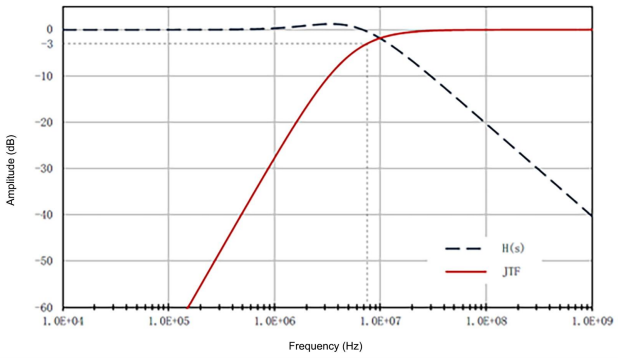


Figure B.6 Reference CDR transfer function and JTF at 20 Gbps and 24 Gbps



Appendix C (Normative) Agreements

C.1 Bit Order

As shown in Figure C.1, bit 0 is the lowest bit and bit 7 is the highest bit in a byte.

Figure C.1 Example of bit order

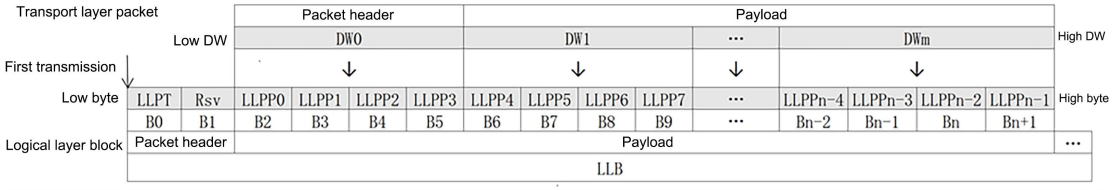
High bit (MSB)							Low bit (LSB)
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

When transmitting bits one by one, the bits are sent in little-endian order, meaning bit 0 is transmitted first and bit 7 is transmitted last.

C.2 Byte Order

The transport layer packets are mapped to the logical layer in a natural order, as shown in Figure C.2.

Figure C.2 Example of transport layer to logical layer mapping (LLPP*x* = LLPP0)



Note: B stands for byte.

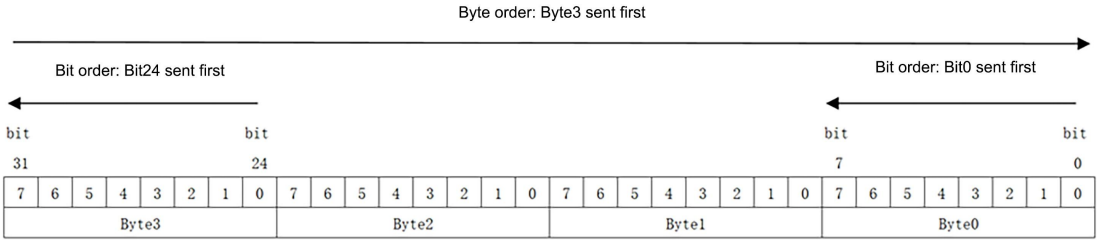
A four-byte DW[31:0] data in the transport layer is shown in Figure C.3, with Byte 0 being the lowest byte, Byte 1 being the second lowest byte, and so on.

Figure C.3 Example of four-byte bit order

High byte			Low byte
Byte 3[31:24]	Byte 2[23:16]	Byte 1[15:8]	Byte 0[7:0]

When the transport layer four-byte DW is mapped to the logical layer for transmission, Byte 3 is transmitted first and Byte 0 is transmitted last. As shown in Figure C.4, the sending order of four-byte data at the transport layer at the logical layer is:
bit24,bit25,bit26,bit27,bit28,bit29,bit30,bit31,bit16,bit17,bit18,bit19,bit20,bit21,bit22,bit23,bit8,bit9,bit10,bit11,bit12,bit13,bit14,bit15,bit0,bit1,bit2,bit3,bit4,bit5,bit6,bit7.

Figure C.4 Byte order and bit transmission order



C.3 CRC32

CRC32 generates verification results according to the following rules:

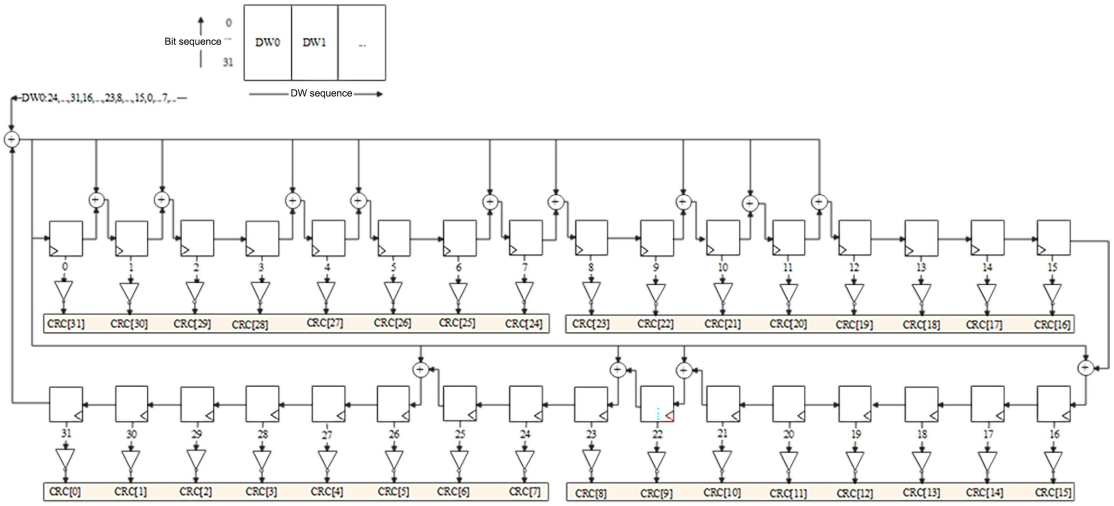
- CRC32 generator polynomial:

$$g(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1 \quad \text{(C.1)}$$

- Initial value: 0xFFFF_FFFFh
- Input data: reverse (True)
- Output data: reverse (True)
- Output check code exclusive OR: 0xFFFF_FFFFh

As shown in Figure C.5, the CRC32 generator circuit generates a check field CRC[31:0] after all bits to be checked are input.

Figure C.5 CRC32 generator circuit



C.4 CRC16

CRC16 generates verification results according to the following rules:

- CRC16 generator polynomial:

$$g(x) = x^{16} + x^{12} + x^5 + 1 \quad \text{(C.2)}$$

- Initial value: 0xFFFFh
- Input data: not reversed (False)
- Output data: not reversed (False)
- Output check code exclusive OR: 0x0000h

C.5 CRC8

CRC8 generates verification results according to the following rules:

- CRC8 generator polynomial:

$$g(x) = x^8 + x^2 + x + 1 \quad \text{(C.3)}$$

- Initial value: 0x00h
- Input data: not reversed (False)
- Output data: not reversed (False)
- Output check code exclusive OR: 0x55h

C.6 ECC

The ECC field contains 6 bits of check information (called ECC6) and uses an extended Hamming code (32, 26). The Hamming code (31, 26) is used to perform ECC error correction on bits 31 to 6

one by one. Bit 0 is used for two-bit error detection of the extended Hamming code. The 5-bit check field (called ECC5) of the Hamming code (31, 26) is calculated according to the following rules:

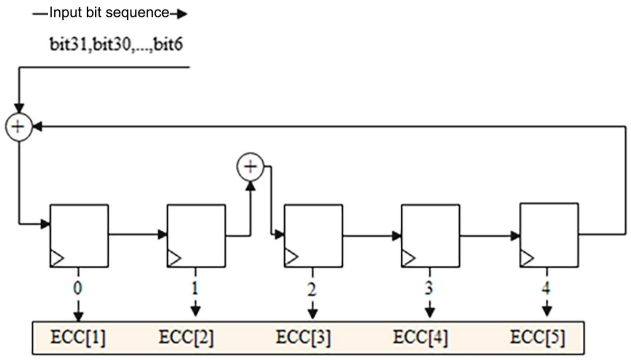
- Generator polynomial:

$$g(x) = x^5 + x^2 + 1 \quad \text{(C.4)}$$

- Check code bit width: 5 bits
- Initial value: 0x00h
- Input data: not reversed (False)
- Output data: not reversed (False)
- Output check code exclusive OR: 0x00h

As shown in Figure C.6, after all 31 bits of the ECC5 encoder circuit are input for verification, ECC[5:1] is the generated verification domain.

Figure C.6 ECC5: Hamming code (31, 26) encoder circuit



Each bit of the ECC[5:1] generated by the ECC5 encoder is exclusive-ORed with all input data to generate an extended field ECC[0], thereby obtaining the check field ECC[5:0] of ECC6.

Appendix D (Informative) Port Low Power Consumption

D.1 Port Entering Low Power Consumption

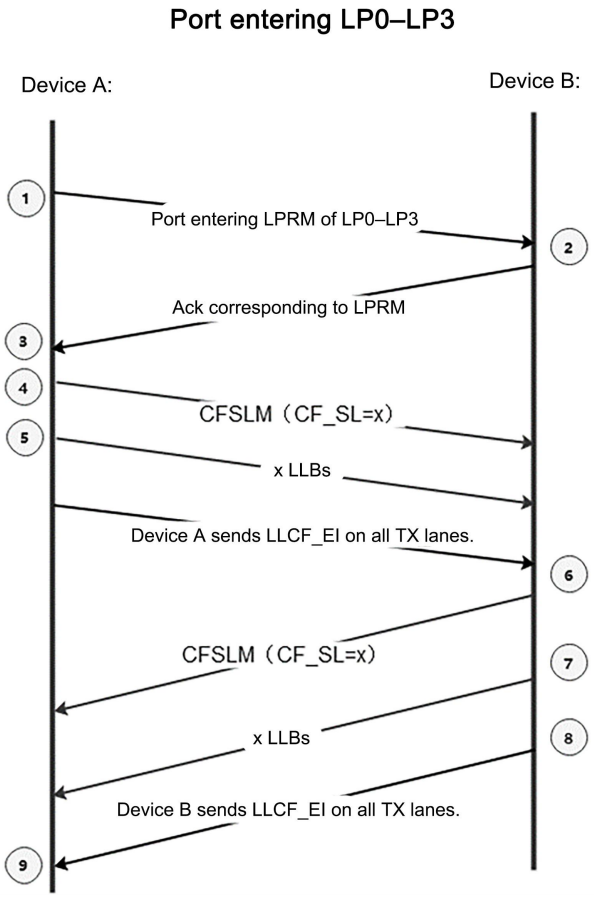
D.1.1 Overview

When the port enters low power consumption, it means that all lanes in use on this port enter a low power consumption state, including the TX link and RX link of this port. The port low power consumption supports LP0, LP1, LP2, and LP3 gears.

This document provides a port low power consumption solution design for reference.

D.1.2 Port Entering LP0–LP3 Low Power Consumption

Figure D.1 Port low power consumption process



(1) Device A sends a request LPRM for port entering LP0–LP3 low power consumption to device B.

(2) Device B feeds back an Ack message responding to the LPRM to device A.

(3) After device A receives the Ack corresponding to the LPRM, it sends one CFSLM to device B and specifies CF_SL as *x* in the CFSLM.

(4) Device A transmits x LLBs to device B.

(5) After all TX lanes send four LLCF_EI after sending service data, all TX lanes of device A enter a low power consumption state.

(6) After all RX lanes of device B receive the CFSLM sent by device A, device B sends one CFSLM to device A and specifies CF_SL as *x* in the CFSLM.

(7) Device B sends *x* LLBs to device A.

(8) After all TX lanes of device B send service data, they send four LLCF_EI, and then all TX and RX lanes of device B enter the low power consumption state.

(9) After all RX lanes of device A receive LLCF_EI, all RX lanes of device A then enter the low power consumption state.

D.1.3 Conditions for Refusing to Enter the Low Power State

When the peer device initiates a low power request, the local end shall decide whether it can enter the low power state according to the states of the audio and video adapter, management adapter, and other devices. If the rejection conditions are met, the local end shall feed back a Nack response to the peer end. Otherwise, it shall feed back an Ack response. The rejection conditions for various scenarios are as follows:

Table D.1 Conditions for denying entry to low power consumption state

Scenario	Conditions
Denying port to enter LP0 low power consumption	Any of the following conditions is true: ● The audio and video adapter denies port entry to LP0. ● The management adapter denies port entry to LP0. ● The current link does not support LP0.
Denying port to enter LP1 low power consumption	Any of the following conditions is true: ● The audio and video adapter denies port entry to LP1. ● The management adapter denies port entry to LP1. ● The current link does not support LP1.
Denying port to enter LP2 low power consumption	Any of the following conditions is true: ● The audio and video adapter denies port entry to LP2. ● The management adapter denies port entry to LP2. ● The current link does not support LP2.
Denying port to enter LP3 low power consumption	Any of the following conditions is true: ● The audio and video adapter denies port entry to LP3. ● The management adapter denies port entry to LP3. ● The current link does not support LP3.

Note: The audio/video adapter and management adapter must implement corresponding low-power entry and exit rejection conditions, based on the low-power entry delay (tLPxEntry) and exit delays (tLP0Exit, tLP1Exit, and tLP2Exit) specified for each low-power mode.

The maximum low power entry delay tLPxEntry specified in the Protocols is detailed in the training parameter requirements.

D.1.4 Troubleshooting Low Power Consumption Exception of Ports

Table D.2 Troubleshooting message loss in the process of entering low power consumption state

Exception	Handling Mechanism
The LPRM request for port entering LP0, LP1, LP2, and LP3 is lost/abnormal.	This type of request message requires the low power responder to feed back a response message. If the response message is lost/abnormal, the LPRM needs to be retransmitted. (See the description in 6.3.6)
The response message to the LPRM of port entering LP0,	The requester retransmits the LPRM, and the responder feeds back an Ack or a Nack message after receiving the LPRM

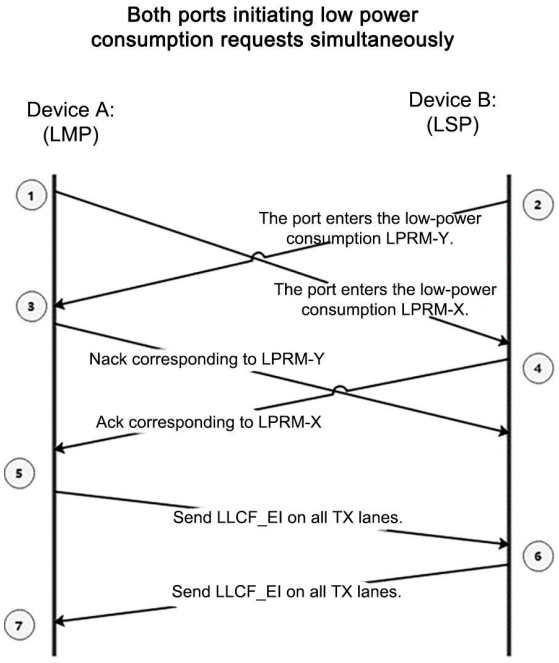
Exception	Handling Mechanism
LP1, LP2, and LP3 is lost/abnormal.	each time. After the requester fails to receive the corresponding response message after multiple times of retransmission, link retraining is initiated through ERR_RM.
LLCF_EI loss/exception	The low power responder (the port receiving the LPRM) detects a link error exception and initiates link retraining or link recovery through ERR_RM. Note: If the receiving end detects any frame header in the LLCF_EI format at the position where an LLCF_EI is received, the LLCF_EI has been recognized.

D.1.5 Troubleshooting Low Power Consumption Process Conflict of Ports

Figure D.2 shows both ends simultaneously initiating a port low power consumption request. The LMP initiates a low power consumption entry request LPRM but does not receive a response from the corresponding LPRM. To prevent process and gear conflict, if the LMP receives the LPRM sent by the peer end, the LMP needs to send a Nack corresponding to the LPRM to reject the low power consumption request from the peer end.

The process is detailed as follows:

Figure D.2 Low power consumption request processing of ports initiated simultaneously at both ends



(1) Device A (LMP) transmits a port low power request LPRM-X to device B (LSP).

(2) When device B has not received the LPRM sent by device A, device B sends a port low power consumption request LPRM-Y to device A.

(3) After device A receives the LPRM sent by device B, if it finds that both ends initiate a port low power consumption request LPRM at the same time, device A as an LMP needs to feed back the Nack corresponding to this LPRM to device B.

Note: In a conflict scenario, the LMP device must reject the request of the LSP device.

(4) After receiving the LPRM-X sent by device A, if device B finds that both ends initiate port low power consumption request LPRM at the same time, it needs to feed back the Ack or Nack corresponding to this LPRM to device A. Specific feedback (Ack or Nack) depends on service needs.

Note: Once the LMP finds that the current low-power process conflicts, it does not wait for the LPRM-Y handshake process to end. If the response message corresponding to LPRM-Y is not received, LPRM-Y will not be retransmitted.

(5) After device A receives the Ack corresponding to the LPRM, all TX lanes send four LLCF_EI after sending service data, and all TX lanes of device A enter a low power consumption state.

(6) After all RX lanes of device B receive LLCF_EI, all TX lanes send a LLCF_EI after sending service data, and then all TX and RX lanes of device B enter a low power consumption state.

(7) After all RX lanes of device A receive LLCF_EI, all RX lanes of device A then enter the low power consumption state.

D.2 Port Exiting Low Power Consumption

D.2.1 Overview

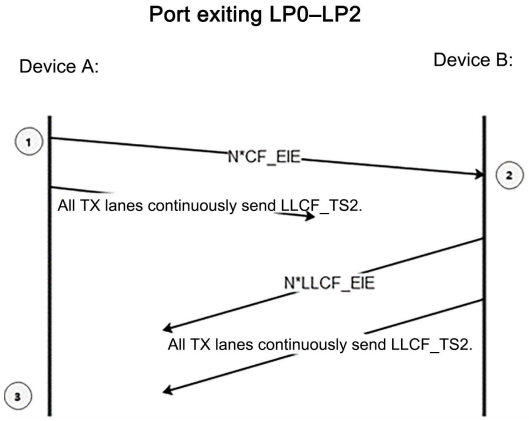
Port exiting low power consumption means that all lanes in use of this port exit the low power consumption state, including the TX link and RX link of this port.

If the current link is in the port low power state, the low power exit initiating device can only initiate a port low power exit request. If the low power exit initiating device initiates other low power exit requests, these requests will be ignored. The processing is shown in Table 44.

If the current link is in the low power consumption state of the port, and the peer device feeds back that the link is abnormal based on ERR_RM and needs to be retrained or restored, both the TX link and RX link of the port need to exit the low power consumption state.

D.2.2 Port Exiting LP0–LP2

Figure D.3 Port exiting LP0–LP2



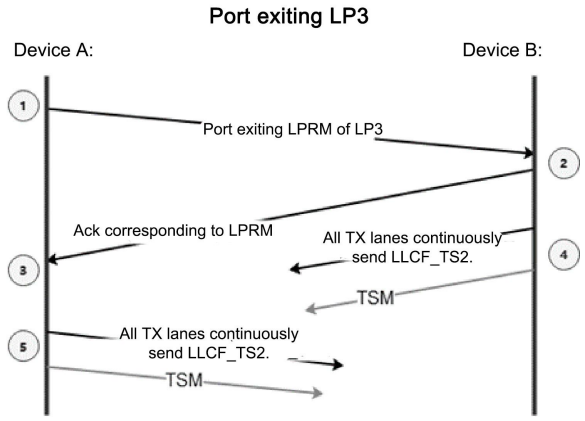
(1) The TX link of device A exits the low power state (the corresponding TXLKSM enters the Recovery state) and transmits N_EIE LLCF_EIE to device B. Then, it continues to transmit LLCF_TS2 for link recovery.

(2) When the device B senses that the RX is in the non-electrical idle state, the receiving link of the device B exits low power consumption (the corresponding RXLKSM enters the Recovery state), and at the same time, the transmitting link of the device B exits low power consumption (the corresponding TXLKSM enters the Recovery state). After the transmitting link of the device B sends N_EIE LLCF_EIE, the device B continues to send LLCF_TS2 for link recovery.

(3) When the receiving link of device A senses that the RX is in a non-electrical idle state, the receiving link of device A exits the low power consumption state (the corresponding RXLKSM enters the Recovery state).

D.2.3 Port exiting LP3

Figure D.4 Port exiting LP3



(1) Device A sends to device B a low power consumption request LPRM for the port to exit LP3.

(2) After receiving the LRRM, the device B feeds back the Ack corresponding to the LPRM to the device A, and the transmitting link and the receiving link of the device B exit low power consumption.

(3) After device A receives the Ack, its transmitting and receiving links exit low power consumption.

(4) After the device B finishes the low power consumption exit, it sends the TSM to indicate that the corresponding lane can start recovery training.

Note: The TSM message here requires that all lanes exiting LP3 start link training. If the number of training lanes indicated by TSM is incomplete, it will cause the untrained lane lock timeout, which will lead to training failure of all lanes exiting LP3. The TSM packet requirements in the next process are the same.

(5) After completing the low power consumption exit, device A continuously sends LLCF_TS2 for link recovery. Then, a TSM needs to be sent to indicate that the corresponding lane can start recovery training.

The port exiting LP3 can be initiated by device B. The process is similar to the port exiting LP3 process initiated by device A described in this section.

D.2.4 Exception Handling Mechanism for Exiting the Low Power Process

Table D.3 Troubleshooting message loss in the process of exiting the low power state

Exception	Handling Mechanism
The LPRM request for port exiting LP3 is lost/abnormal	Such request messages require the peer end to feedback a response message. Once the response message is lost/abnormal, the LPRM needs to be retransmitted. (See the description in 6.3.6)
The response message corresponding to the LPRM of port exiting LP3 is lost/abnormal	The requester retransmits the LPRM, and the responder feeds back an Ack or a Nack message after receiving the LPRM each time. After the requester fails to receive the corresponding response message after multiple retransmissions, link retraining is initiated through ERR_RM.

Appendix E (Informative) Transport Layer Flow Control Management

E.1 Flow Control Mechanism

Flow control ensures that the RBuff cache space of the link receiving unit does not overflow. Different types of service flow packets may use different mechanisms. The transport layer shall support the flow control management mechanism as shown in Table E.1.

Table E.1 Flow control management mechanism

Type	Description
Flow control disabling mechanism	When flow control is disabled, the link receiving unit allocates one flow control disabling buffer space for all service flows adopting the flow control disable mechanism. There is no flow control restriction when the link sending unit sends packets, and the link receiving unit discards the packets after the flow control disabling buffer overflows.
Exclusive flow control mechanism	The link receiving unit allocates an independent buffer space for each service flow adopting the exclusive flow control mechanism. Only when the independent buffer space is sufficient, the link sending unit can send packets.
Shared flow control mechanism	The link receiving unit allocates a shared buffer space for all service flows adopting the shared flow control mechanism. Only when the shared buffer space is sufficient, the link sending unit can send packets.

Note: The same service flow at both ends of a link needs to be configured with the same flow control mechanism.

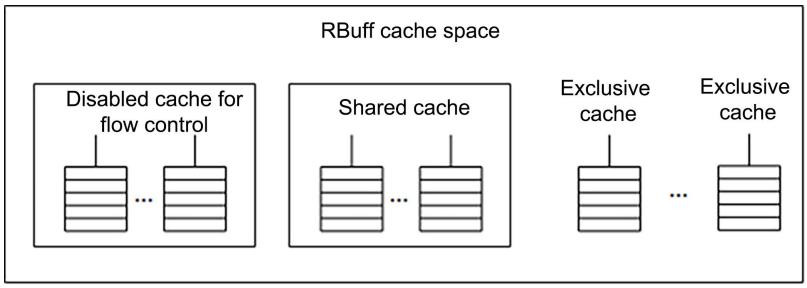
Each pair of the link sending unit and the link receiving unit independently performs flow control management of the link. The service flow packet received from the management adapter is fixed to adopt an exclusive flow control mechanism, and the unidirectional service flow and multicast service flow adopt a flow control disabling mechanism.

E.2 Buffer Space

As shown in Figure E.1, RBuff contains at least three types of buffer space:

- "Flow Control Disabling Buffer" space: Packets adopting the flow control disabling mechanism share the buffer space.
- "Shared Buffer" space: Packets adopting a shared flow control mechanism share the buffer space.
- "Exclusive Buffer" space: Packets adopting the exclusive flow control mechanism have independent buffer space.

Figure E.1 Schematic diagram of buffer space type



The GPMI uses a credit-based mechanism to allocate, track, and synchronize RBuff buffer space. One credit represents a specific byte length in the buffer space, and one credit contains 32 bytes. A 128-byte-long packet occupies four credits. If the credit value is not an integer, it is rounded up.

The management adapter configures three types of buffer space according to the buffer size of Rbuff. The buffer occupied by each service stream adopting the exclusive flow control mechanism cannot exceed the allocated "exclusive buffer space credit number", and the transport layer does not track the credit number. The buffer occupied by all service flows adopting the shared flow control mechanism cannot exceed the allocated "shared buffer space credit number", and the transport layer independently tracks the credit number of the buffer space. The buffer occupied by all service streams adopting the flow control disabling mechanism cannot exceed the allocated "flow control disabling buffer space credit number", and the transport layer independently tracks the credit number of the buffer space.

TLMP is generated and terminated at the transport layer, so it is not stored in RBuff and does not participate in flow control. When receiving TLDP, if the link receiving unit does not have enough credit buffer, the entire packet is discarded and a buffer exception alarm (RBuff_Err) is reported to the management adapter.

E.3 Credit Allocation

The link receiving unit informs the link sending unit of the number of credits allocated to each buffer space through a "transport layer credit allocation packet". The structure and field definition of the transport layer credit allocation packet are shown in Figure E.2 and Table E.2, respectively.

Figure E.2 Structure of the transport layer credit allocation packet

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Rsvd										TLMP Type=0						Length									D=0	ECC					
ST	Rsvd	Credits_Allocated														Rsvd	ShuttleID							Rsvd		ECC					
……																															
ST	Rsvd	Credits_Allocated														Rsvd	ShuttleID							Rsvd		ECC					

Table E.2 Definition of the payload field of the transport layer credit allocation packet

Bit	Field Name	Description
31	ST	Flags of shared and exclusive flow control mechanisms: Shared flow control mechanism: 0b; Exclusive flow control mechanism: 1b
30	Rsvd	Reserved
29:16	Credits_Allocated	Number of credits allocated
15	Rsvd	Reserved
14:8	ShuttleID	The lane identifier of the corresponding service flow packet. Valid when ST is 1; reserved when ST is 0.
7:6	Rsvd	Reserved
5:0	ECC	Checks bit [31:6].

After receiving the credit allocation packet, the link sending unit performs ECC check on each entry, corrects any single bit ECC error, and updates the "allocation packet ECC error correction counter". If an uncorrectable error is detected, the corresponding entry is discarded and the "allocation packet ECC error uncorrectable counter" is updated.

After receiving the credit allocation packet, the link sending unit shall clearly respond to each service flow through the transport layer credit allocation response packet. After extracting the credit allocation information from the credit allocation entry, if the number of allocated credits is more than that maintained locally, the Ack of the corresponding entry will be directly replied. Otherwise, the buffer has been reduced. An Ack is replied after the corresponding buffer space increases. The structure and field definition of the transport layer credit allocation response packet are shown in Figure E.3 and Table E.3, respectively.

Figure E.3 Structure of the transport layer credit allocation response packet

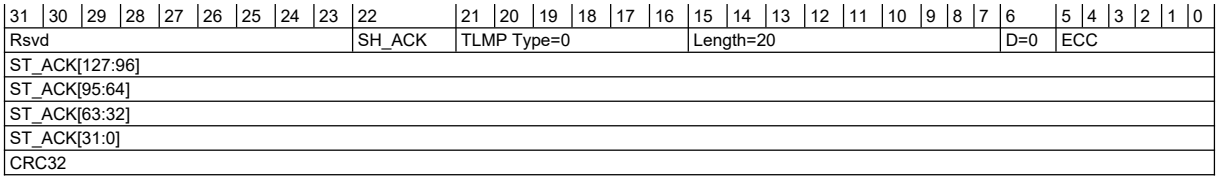


Table E.3 Field definition of the transport layer credit allocation response packet

Field Name	Description
SH_ACK	Responds to the credit allocation packet of the service flows adopting the shared flow control mechanism.
ST_ACK	Responds to the credit allocation packet of each service flow adopting the exclusive flow control mechanism. ST_ACK[0] indicates a response to the service flow with ShuttleID 0.
CRC32	Checks ST_ACK.

After receiving the credit allocation response packet, the link receiving unit performs a CRC check on the payload. If the CRC check fails, all ST_ACK carried are discarded and the "CRC error counter of the credit allocation response packet" is updated. CRC does not affect the response (SH_ACK) of the shared buffer.

E.4 Credit Consumption

The transport layer maintains the number of credits consumed by each service flow. The link sending unit synchronizes the credit consumption to the link receiving unit through a "credit consumption packet". The structure and field definition of the transport layer credit consumption packet are shown in Figure E.4 and Table E.4, respectively.

Figure E.4 Structure of the transport layer credit consumption packet

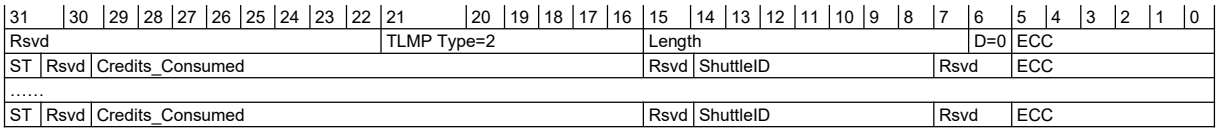


Table E.4 Definition of the payload field of the transport layer credit consumption packet

Bit	Field Name	Description
31	ST	Flags of shared and exclusive flow control mechanisms: Shared flow control mechanism: 0b; Exclusive flow control mechanism: 1b
30	Rsvd	Reserved
29:16	Credits_Consumed	Number of credits consumed.
15	Rsvd	Reserved
14:8	ShuttleID	The lane identifier of the corresponding service flow packet. Valid when ST is 1; reserved when ST is 0.
7:6	Rsvd	Reserved
5:0	ECC	Checks bit [31:6].

In order to avoid the discarding of credit consumption entries due to link errors, each entry is sent three times and included in three credit consumption packets. The interval between every two credit consumption packets is specified by tTxTLCCPItv. For the value of tTxTLCCPItv, see Section 7.11.

If the flow control time exceeds tTxErrFC, the service stream flow control is abnormal. In this case, the link sending unit sends a credit consumption packet corresponding to the service flow. If the flow control is abnormal, the link sending unit sends the flow control abnormality status to the link receiving unit through a notification packet, and then restarts the flow control timing. If the flow control abnormality state occurs for three consecutive times (the flow control has not been disabled during this period), a serious error of flow control abnormality is reported to the

management adapter. The structure and field definition of the flow control abnormality notification packet are shown in Figure E.5 and Table E.5, respectively.

Figure E.5 Structure of the transport layer flow control abnormality notification packet

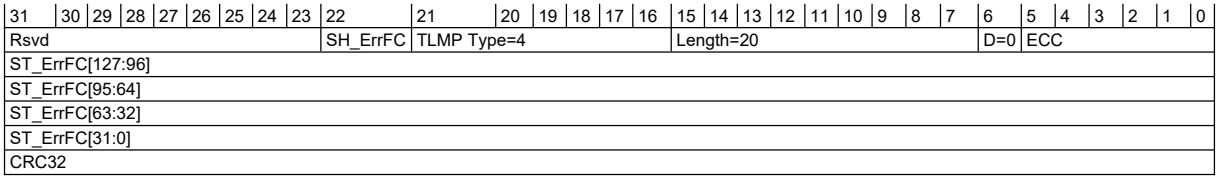


Table E.5 Definition of the payload fields of the transport layer flow control abnormality notification packet

Field Name	Description
SH_ErrFC	Indicates the flow control abnormality state of that service flow adopting the shared flow control mechanism.
ST_ErrFC	Indicates the flow control abnormality state of each service stream adopting the exclusive flow control mechanism. For example, the ST_ErrFC[0] indicates the flow control abnormality state of the service flow whose ShuttleID is 0.
CRC32	Checks ST_ACK.

After receiving the credit consumption packet, the link receiving unit performs ECC check on each entry, corrects any single bit ECC error, and updates the "consumption packet ECC error correction counter". If an uncorrectable error is detected, the corresponding entry is discarded and the "consumption packet ECC error uncorrectable counter" is updated.

E.5 Credit Recycle

After the RBuff forwards packets to the adapter or other port TBuff, the available buffer space of the corresponding service flow increases, and the link receiving unit shall maintain the number of credits recovered in time and synchronize them to the link sending unit through the "credit recycle packet". The structure and field definition of the transport layer credit recycle packet are shown in Figure E.6 and Table E.6, respectively.

Figure E.6 Structure of the transport layer credit recycle packet

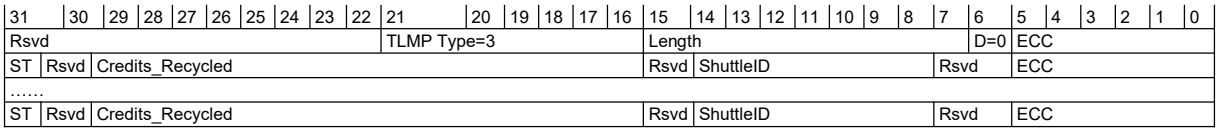


Table E.6 Definition of the payload field of the transport layer credit recycle packet

Bit	Field Name	Description
31	ST	Flags of shared and exclusive flow control mechanisms:

Bit	Field Name	Description
30	Rsvd	Reserved
29:16	Credits_Recycled	Number of credits recycled.
15	Rsvd	Reserved
14:8	ShuttleID	The lane identifier of the corresponding service flow packet. Valid when ST is 1; reserved when ST is 0.
7:6	Rsvd	Reserved
5:0	ECC	Checks bit [31:6].

After receiving the credit recycle packet, the link sending unit performs ECC check on each entry, corrects any single bit ECC error, and updates the "recycle packet ECC error correction counter". If an uncorrectable error is detected, the corresponding entry is discarded and the "recycle packet ECC error uncorrectable counter" is updated.

Appendix F (Normative) Device Comprehensive Capability Description

F.1 Overview

The device comprehensive capability description (DCCD) adopts a structure mode with variable length. The capabilities embodied by the DCCD are encapsulated in different data blocks, and users can expand the capability declaration by adding an extension data block.

This chapter describes the structure of the data content, excluding any discussion of devices or communication protocols.

F.2 Data Structure and Definition

F.2.1 Overview

The DCCD data structure uses four bytes to represent the start of the data structure: 0xD, 0xC, 0xC, and 0xD. The DCCD defines a system framework and data structure, which declares various capabilities supported by the display device, such as device parameters, audio and video capabilities, and others. The data structure is a continuous data structure consisting of a fixed-length basic field, a plurality of variable-length extension data blocks, and a final check, which is called DCCD structure for short. The data structure is shown in Figure F.1.

Table F.1 Data format

Attribute	Description	Remarks
Basic segment	Protocol identification	
	Version information	
	Device capability	
	Data length	

Attribute		Description	Remarks
Product information data block	Data block header	Field identifier = 0x1	
		Field version	
		Field length	
	Valid data of data block	Field valid data 1	
		Field valid data 2	
		…	
		Field valid data *N*	
Data block 1	Data block header	Field identifier	
		Field version	
		Field length	
	Valid data of data block	Field valid data 0	
		Field valid data 1	
		…	
		Field valid data *N*	
…			
Verification		Checksum	Currently, The checksum method is used to check the validity of the entire DCCD data. That is, the lower 8 bits of the sum of all DCCD data must be 0.

The basic segment is mainly used to declare the protocol version of the standard, as well as a summary description of device attributes and functions, and to pass the length information of the subsequent extended data. A DCCD can be composed of multiple data block fields, and each data block is used to describe the capability declaration of one of them. The maximum length of each data block cannot exceed 256 bytes.

Device capability is used to describe different types of device capabilities, for example, to describe the ability to receive audio and video. Subsequent extensions can be used to describe more different kinds of device capabilities.

A description device can declare one or more device capabilities. After acquiring its device capabilities, the peer device confirms the services that can be interacted between devices in combination with its own device capabilities.

F.2.2 Basic Field

The basic field consists of a fixed-length section (12 bytes) that mainly describes the version information and length of the DCCD and the basic device information. The structure is shown in Table F.2.

Table F.2 Basic field data format

Address	Bit	Description	Definition of Data Block
00h	7:0	0xD	The identification adopts the DCCD data structure.
01h	7:0	0xC	
02h	7:0	0xC	
03h	7:0	0xD	
04h	DCCD version information (current version: 1.0)		Describes the current DCCD version information, which is 10h by default.
	7:4	Major version number 1h = Version 1	
	3:0	Revision number 0h = revision 0	
05h	7:0	Reserved bit Default value: 0	
06h	7:4	Reserved bit Default value: 0	
	3	Device capability 1	BIT3: audio transmission capability
	2		BIT2: video transmission capability
	1		BIT1: audio receiving capability
	0		BIT0: video receiving capability
07h	7:0	Device capability 2	Reserved bit; default value: 0
08h	7:0	Device capability 3	Reserved bit; default value: 0
09h	7:0	Device capability 4	Reserved bit; default value: 0
0Bh to 0Ah	Data length Length: N		The data length is the length of all subsequent data (including data block and check value).
0Ch	7:0	Product information data block	
…		…	
0Bh+N	7:0	Check value Value range: 0–0xFF	

F.2.3 Definition of Data Block

The data block is mainly used to indicate the detailed description of the device capabilities, such as device parameters, audio and video capabilities, and others. The data block provides a more detailed description besides the device capabilities described in the basic segment. The data structure of the data block is shown in Table F.3.

Table F.3 Data block definition format

Attribute		Skew	Bit	Description	Function Description
Data block	Data block header	00h	7:0	Field identifier Value range: 1–0xFFF	Describes the functional characteristic identification of the data block. See Table F.4 for details. The field identifier length is 12 bits; the 7:4 bit in the 01h byte is 11:8 bit; and the 00h byte is 7:0 bit.
		01h	7:4		
			3:0	Field version	Version information of the data block; current default value: 1.
		02h	7:0	Field length Data block effective length (n) Value range: 0–0xFD	Indicates the valid data length in the data block.
	Valid data of data block	03h	7:0	Field valid data 1	The detailed description corresponds to the functional characteristics of the identifiers. Different identifiers have varying functional descriptions and differently arranged data declaration structures.
		04h	7:0	Field valid data 2	
				…	
				Field valid data *n*	

During DCCD parsing, if an unknown field identifier is identified (for example, a lower version device parses a new field identifier from a higher version device), the parsing cannot be terminated, and the corresponding content of the data segment shall be skipped according to the length of the data block. The data block shall be ignored, and the subsequent content shall be parsed.

Data blocks are used to describe various attributes and functions of the device. Different functions are distinguished by different identifiers. See Table F.4 for the content of the corresponding identifier. The length of each identified data block field is not fixed, and can be dynamically changed according to the content of the contained sub-data.

Table F.4 Definition of basic field identifiers

Field Identifier	Description	Remarks

Field Identifier	Description	Remarks
0x1	Product information field	
0x2	GPMI device description field	
0x3	GPMI port description field	
0x4	GPMI adapter description field	
0x5	ADCP capability description field	
0xF-0x6	Reserved bit	
0x10	Display parameter field	
0x11	CTA Timing field	
0x12	DMT Timing field	
0x13	CVT Timing field	
0x14	Reserved bit	Reserved identifier for video timing
0x15	Detailed timing description field	
0x16	Basic display capability field	
0x17	YCbCr 4:2:0 capability field	
0x18	Video capability field	
0x19	Advanced video capability field	
0x1A	Static HDR capability field	
0x1B	Dynamic HDR capability field	
0x1C	HDR vivid capability field	Reserved field
0x20	Video compression (PLLC) capability field	
0x3F-0x1C	Reserved bit	Video-related reserved field
0x40	Audio capability field	
0x41	Speaker capability field	
0x42	Advanced audio capability field	Reserved field
0xFC-0x42	Reserved bit	
0xFD	CTA extension field	
0xFE	Vendor-specific data field	

F.3 Product Information Field Requirements

This field is a required field of the DCCD and is mainly used to describe the basic information of a product. If it is declared, it must be the first data block. This field can only have one. Multiple product information fields are not allowed. Its structure description is shown in Table F.5.

Table F.5 Product information field

Skew	Bit	Description	Remarks
00h	7:0	Default value: 0x01h	The field identifier is 0x01.
01h	7:4	Default value: 0	
	3:0	Field version Default value: 1	
02h	7:0	Field length Valid data length; value range: 0x11–0x1E	
08h to 03h	47:0	Vendor name	See Table F.7.
0Ch to 09h	31:0	Product code	The product code is a 4-byte code number assigned by the vendor. The storage method follows a little-endian arrangement, meaning the least significant byte is stored at the lowest address. The vendor sets different codes for various products so that the peer device can effectively distinguish different device types by the same vendor.
10h to 0Dh	31:0	Product version number	The product version number is a 4-byte value used to distinguish between different versions of the same product. It is stored in hexadecimal format and supports up to 8 digits. The storage method follows a little-endian arrangement, meaning the least significant byte is stored at the lowest address.
11h	7:0	Week of production	The production week is recorded as the week number of the year and stored in hexadecimal format.
12h	7:0	Year of production	The production year is recorded as an offset from 1990 (where 1990 is represented as 0), with subsequent years increasing incrementally, and is stored in hexadecimal format.
13h	7:0	Product name length Effective data length of the product name (n); value range: 0–0x0D	13h describes the length of the product name. If the length is 0, the product name does not exist. The maximum effective length of the product name cannot exceed 13 bytes.
13h+N–14h	7:0	Product name	14h and later fields indicate the name

Skew	Bit	Description	Remarks
			description of a product. The product name is stored in ASCII format, that is, 14h is the first character of the product name, 15h is the second character, and so on.

08h to 03h indicates the "Vendor Name", which supports a maximum of six characters, each corresponding to an ASCII code. The encoding table corresponding to the ASCII code is shown in Table F.6.

Table F.6 Encoding table corresponding to the ASCII code

Description	A	B	C	D	E	F	G	H	I	J	K	L	M
ASCII Value	41h	42h	43h	44h	45h	46h	47h	48h	49h	4Ah	4Bh	4Ch	4Dh
Description	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
ASCII Value	4Eh	4Fh	50h	51h	52h	53h	54h	55h	56h	57h	58h	59h	5Ah
Description	0	1	2	3	4	5	6	7	8	9	-		
ASCII Value	30h	31h	32h	33h	34h	35h	36h	37h	38h	39h	2Dh		

The layout of vendor names is shown in Table F.7. If there are less than 6 letters, fill the field with 0.

Table F.7 Vendor name layout table

Skew	Description
03h	Character 1
04h	Character 2
05h	Character 3
06h	Character 4
07h	Character 5
08h	Character 6

F.4 GPMI Description Field

F.4.1 Overview

This field is mainly used to describe the capability declaration in GPMI devices, which mainly includes the description of the device, port, and adapter capabilities.

The GPMI description field is detailed in Table F.8.

Table F.8 GPMI description field

Field Identifier	Description
0x2	GPMI device description field
0x3	GPMI port description field
0x4	GPMI adapter description field
0x5	ADCP capability description field

F.4.2 GPMI Device Description Field

This field is mainly used to describe the parameter information of a GPMI device, such as the GPMI protocol version and ID.

The structural description is detailed in Table F.9.

Table F.9 GPMI device description field structure

Address	Bit	Description	Remarks
00h	7:0	Field identifier Default value: 0x2h	
01h	7:4	Default value: 0	
	3:0	Field version Default value: 1	
02h	7:0	Field length Valid data length fixed to Dh	
03h	7:4	GPMI protocol major version number	
	3:0	GPMI protocol sub-version number	
09h to 04h	47:0	DEVICE_ID	Device ID
0Bh to 0Ah	15:0	VENDOR_ID	Vendor identifier
0Ch	7:0	Number of ports Value range: 0x1–0xF	
0Dh	7:0	Number of adapters	Management adapter not included
0Eh	7:0	MAX_DEVICE_NUM Maximum number of devices supported	
0Fh	7:0	Maximum number of routing levels supported from source to sink devices for audio and video services	

Note: If the DCCD contains a GPMI device description field, the first data block in its DCCD shall be the product information field. If the GPMI device description field is not included, the first data block may not include the product information field.

F.4.3 GPMI Port Description Field

This field is mainly used to describe the parameter information of the GPMI port, such as the GPMI port type and supported rates, as shown in Table F.10.

Table F.10 GPMI port description field structure

Address	Bit	Description	Remarks
00h	7:0	Field identifier Default value: 0x3h	
01h	7:4	Default value: 0	
	3:0	Field version Default value: 1	
02h	7:0	Field length Valid data length fixed to 11h	
03h	7:6	Port Types	Supported port types: 1h: LMP, main downlink port 2h: LSP, main uplink port 3h: DRP, dual role port Others: reserved
	5:4	Link Mode	Supported link modes: Bit 0: whether to support the simplified-specification mode Bit 1: whether to support the full-specification mode
	3:0	Port number	
04h	7:6	Lane 3 lane mode	Corresponds to the lane mode supported by each lane. The definition is as follows: 00b: Disable 01b: RX; 10b: TX; 11b: TRX
	5:4	Lane 2 lane mode	
	3:2	Lane 1 lane mode	
	1:0	Lane 0 lane mode	
05h	7:6	Lane 7 lane mode	
	5:4	Lane 6 lane mode	
	3:2	Lane 5 lane mode	
	1:0	Lane 4 lane mode	
07h to 06h	15:0	Lane rate supported by TX	The lane rate supported by the sending/receiving lane supports any combination: Bit 0: whether to support 2 Gbps;
09h to 08h	15:0	Lane rate supported by RX	

Address	Bit	Description	Remarks
			Bit 1: whether to support 4 Gbps; Bit 2: whether to support 6 Gbps; Bit 3: whether to support 8 Gbps; Bit 4: whether to support 10 Gbps; Bit 5: whether to support 12 Gbps; Bit 6: whether to support 16 Gbps; Bit 7: whether to support 20 Gbps; Bit 8: whether to support 24 Gbps; Others: reserved

Table F.10 GPMI port description field structure (continued)

Address	Bit	Description	Remarks
0Ah	7:0	Supported max ShuttleID:	
0Bh	7:0	Supported low-power modes	Supported low power modes: Bit 0: whether to support LP0; Bit 1: whether to support LP1; Bit 2: whether to support LP2; Bit 3: whether to support LP3; Bit 4: whether to support LP0f; Others: reserved
0Ch	7:4	Reserved bit; default value: 0	
	3	Whether to support rapid training	0b: does not support fast training. 1b: supports fast training.
	2	Whether to support the Skew adjustment feature	0b: does not support dynamic Skew link width switch. 1b: supports dynamic Skew link width switch.
	1	Whether to support dynamic lane direction switch	0b: does not support lane direction switch. 1b: supports lane direction switch.
	0	Whether to support dynamic link width switch	0b: does not support link width switch. 1b: supports link width switch.
0Dh	7:0	Swing gear	Supported Swing levels Each bit indicates whether to

Address	Bit	Description	Remarks
			support the corresponding Swing level, and bit_N represents whether to support Swing Level N. For example, 00111100b indicates that Swing levels 2, 3, 4, and 5 are supported, but Swing levels 0, 1, 6, and 7 are not supported.
0Eh	7:4	FFE (POST1) gear mask	This field is defined as a mask, indicating the valid bit among the four bits of the FFE parameter. Examples: 0000b: gear 0 supported; 0011b: gears 0, 1, 2, and 3 supported 0110b: gears 0, 2, 4, and 6 supported; 1100b: gears 0, 4, 8, and 12 supported; 1110b: gears 0, 2, 4, 6, 8, 10, 12, and 14 supported The FFE level supports up to 64 levels, that is, at most 6 bits of the 12 bits in this field are set to 1. The bits set to 1 in the mask of each parameter must be contiguous. For example, 0110b is a valid value, 1010b is an invalid value.
	3:0	FFE (PRE1) gear mask	
0Fh	7:4	Reserved bit	
	3:0	FFE (POST2) gear mask	
10h	7	Whether to support the fast recovery feature at HS1 (2 Gbps/4 Gbps) rate	0b: does not support fast recovery. 1b: supports fast recovery.
	6:4	The number of single rounds of transmitted LLCF_TS2 in fast recovery at HS1 (2 Gbps/4 Gbps) rate	0h: 10h; 1h: 40h; 2h: 100h; 3h: 400h; 4h: 1000h; 5h: 4000h; 6h: 10000h; 7h: 40000h
	3:0	The number of rounds of transmitted LLCF_TS2 in fast recovery at HS1 (2 Gbps/4 Gbps) rate	
11h	7	Whether to support the fast recovery feature at HS2 (6 Gbps/8 Gbps) rate	0b: does not support fast recovery. 1b: supports fast recovery.
	6:4	The number of single rounds of transmitted LLCF_TS2 in fast	0h: 10h; 1h: 40h;

Address	Bit	Description	Remarks
		recovery at HS2 (6 Gbps/8 Gbps) rate	2h: 100h; 3h: 400h; 4h: 1000h; 5h: 4000h; 6h: 10000h; 7h: 40000h
	3:0	The number of rounds of transmitted LLCF_TS2 in fast recovery at HS2 (6 Gbps/8 Gbps) rate	
12h	7	Whether to support the fast recovery feature at HS3 (10 Gbps/12 Gbps/16 Gbps) rate	0b: does not support fast recovery. 1b: supports fast recovery.
	6:4	The number of single rounds of transmitted LLCF_TS2 in fast recovery at HS3 (10 Gbps/12 Gbps/16 Gbps) rate	0h: 10h; 1h: 40h; 2h: 100h; 3h: 400h; 4h: 1000h; 5h: 4000h; 6h: 10000h; 7h: 40000h
	3:0	The number of rounds of transmitted LLCF_TS2 in fast recovery at HS3 (10 Gbps/12 Gbps/16 Gbps) rate	
13h	7	Whether to support the fast recovery feature at HS4 (20 Gbps/24 Gbps) rate	0b: does not support fast recovery. 1b: supports fast recovery.
	6:4	The number of single rounds of transmitted LLCF_TS2 in fast recovery at HS4 (20 Gbps/24 Gbps) rate	0h: 10h; 1h: 40h; 2h: 100h; 3h: 400h; 4h: 1000h; 5h: 4000h; 6h: 10000h; 7h: 40000h
	3:0	The number of rounds of transmitted LLCF_TS2 in fast recovery at HS4 (20 Gbps/24 Gbps) rate	

F.4.4 GPMI Adapter Description Field

This field is mainly used to describe the parameter information of the GPMI adapter and provide an overview of the capabilities of various adapters, as shown in Table F.11.

Table F.11 GPMI adapter description field structure

Address	Bit #	Description	Remarks
00h	7:0	Field identifier Default value: 0x4h	
01h	7:4	Default value: 0	
	3:0	Field version Default value: 1	
02h	7:0	Field length Valid data length fixed to 5h	
03h	7:0	Adapter type	1: audio and video transmission 2: audio and video receiving; Others: reserved
04h	7:0	Adapter version Current default value: 1	
05h	7:0	Adapter number	
06h	7:0	Adapter Detailed Function Description Data Length	
07h	7:0		

This field can be used to describe various types of adapters. These adapters also need to declare the detailed capability description of their sub-functions. For example, if the "Adapter Type" in BIT3 is 1 "Audio Receiving Capability", it also needs to describe the audio and video related capabilities of the adapter.

BIT7_6 "Adapter Detailed Function Description Data Length" indicates the effective length of the subsequent adapter details description.

F.4.5 ADCP Capability Description Field

This field is mainly used to describe the parameter information of the ADCP capability, as shown in Table F.12. This field is bound to the adapter. If the adapter supports the ADCP capability, this field needs to be declared. If this field is not supported, no declaration is required.

Table F.12 ADCP capability description field structure

Address	Bit #	Description	Remarks
00h	7:0	Field identifier Default value: 0x5h	
01h	7:4	Default value: 0	
	3:0	Field version Default value: 1	
02h	7:0	Field length	

Address	Bit #	Description	Remarks
		Valid data length fixed to 1h	
03h	7:1	Reserved	
	0	Whether to support ADCP 1.0	

F.4.6 GPMI DCCD Declaration Strategies

F.4.6.1 Overview

The GPMI capability description is divided into three types: device, port, and adapter. The DCCD supports defining these capability types individually, in any combination, or as a single unified declaration.

The basic segment description of the "Device", "Port", and "Adapter" of the same port describes the capabilities of the port, that is, the "Device Capability" (BIT9:0) in the basic segment description of different types of capabilities of the same port must be consistent.

F.4.6.2 GPMI Device Description Only

In a field that contains the device description, the product information field is mandatory and must be placed in the first data block. The layout is shown in Table F.13.

Table F.13 Structure of GPMI device description only

Field Type	Overview
Basic field information	
Product information field	GPMI device description mandatory field
GPMI device description field	
Verification	

F.4.6.3 GPMI Port Description Only

In the field containing port description, there is no requirement for other fields, and their layout is shown in Table F.14.

Table F.14 GPMI port description structure only

Field Type	Overview
Basic field information	
GPMI port description field	Port 1 capability description
GPMI port description field	Port 2 capability description
Verification	

F.4.6.4 GPMI Adapter Description Only

A device may contain multiple adapters of different or the same type, so a DCCD describing an adapter may contain multiple GPMI adapter description fields.

Each adapter, according to its adapter attributes, also needs to declare the detailed functional characteristics related to the adapter. If the adapter is an "Audio and Video Receiving Adapter", it also needs to declare the capability parameters related to audio and video. Its layout is shown in Table F.15.

Table F.15 GPMI adapter description structure only

Field Type	Overview
Basic field information	
GPMI adapter description field	Capability description of adapter 1
Detailed function description field 1	
…	
Detailed function description field *n*	
GPMI adapter description field	Capability description of adapter 2
Detailed function description field 1	
…	
Detailed function description field *n*	
Verification	

The "Detailed Function Description Field" in the adapter capability description needs to be declared according to the corresponding adapter function type.

The "ADCP Capability Description Field" is also one of the detailed function description fields in the "GPMI Adapter Description Field". If the adapter supports ADCP, the "ADCP Capability Description Field" must be included. If the adapter does not support ADCP, the "ADCP Capability Description Field" cannot be included.

The "Adapter Detailed Content Description Data Length" (BIT7:6) in the "GPMI Adapter Description Field" is the sum of the lengths of all subsequent "Detailed Function Description Fields" of the adapter. If the detailed function description is not required to be declared in the adapter, the corresponding "Adapter Detailed Content Description Data Length" can be 0.

Note: During the DCCD parsing, the quick jump between different adapters can be realized by obtaining the information of "Adapter Detailed Content Description Data Length".

F.4.6.5 GPMI Multiple Capabilities Description

The GPMI capability description can be flexibly combined according to the type of capability requested.

The fields of different GPMI description types are not allowed to be mixed, that is, after the function description of one type is declared, the function description of the next type can be declared.

The order of precedence for the three GPMI description field types must be device, port, and adapter, as shown in Table F.16.

Table F.16 GPMI multiple capabilities description structure

Field Type	Overview
Basic field information	
Product information field	GPMI device description mandatory field
GPMI device description field	
GPMI port description field	GPMI Port Description Required Field
GPMI adapter description field	Capability Description of GPMI Adapter 1
Detailed function description field 1	
…	
Detailed function description field *n*	
GPMI adapter description field	Capability Description of GPMI Adapter 2
Detailed function description field 1	
…	
Detailed function description field *n*	
Verification	

If the three capabilities of the GPMI do not need to be fully reflected, the corresponding declaration capability field can be reserved according to the query requirements of the peer end.

F.5 Detailed Data Segment Requirements for Audio and Video Receiving Capability

F.5.1 Overview

The following description fields are valid only in "Audio and Video Receiving Adapter". Other adapter types cannot contain information of the following field.

F.5.2 Display Parameter Field Requirements

This field is mainly used to describe the parameter information of the product display. If the adapter supports "Video Receiving Capability", this field is mandatory; otherwise, the field is not selectable. This field can only have one. Multiple display parameter fields are not allowed. The structural description reference is detailed in Table F.17.

Table F.17 Display parameter fields

Skew	Bit	Description	Remarks
00h	7:0	Field identifier Default value: 0x10h	
01h	7:4	Default value: 0	
	3:0	Field version Default value: 1	
02h	7:0	Field length Default valid data length value: 0x12	
03h	7:0	Display horizontal width 7:0 bit Value range: 0–0xFF	Describes the size of the display, in cm. The horizontal width and vertical height support a maximum of 4,095 cm. If the field is 0, it means that no display size information is provided.
04h	7:0	Display vertical height 7:0 bit Value range: 0–0xFF	
05h	7:4	Display horizontal width 11:8 bit Value range: 0–0xF	
	3:0	Display vertical width 11:8 bit Value range: 0–0xF	
06h	7:0	Display horizontal pixel size 7:0 bit Value range: 0–0xFF	Describes the pixel size of the display, which supports a maximum of 65,535 horizontal and vertical pixels. If the field is 0, it means that no display pixel size information is provided.
07h	7:0	Display horizontal pixel size 15:8 bit Value range: 0–0xFF	
08h	7:0	Display vertical pixel size 7:0 bit Value range: 0–0xFF	
09h	7:0	Display vertical pixel size 15:8 bit Value range: 0–0xFF	
0Ah	7:0	Display conversion feature (gamma)	Describes information related to the gamma electro-optical transfer function (EOTF) supported by the display. The conversion formula between the field value and gamma value is: Gamma value = (Field value + 100)/100 The range of its gamma value is 1.00–3.54. If the input value is FFh, it means that there is no gamma information currently.

Skew	Bit	Description	Remarks
14h to 0Bh	79:0	Color feature	Table F.18

The color feature sub-field mainly describes the information related to color chromaticity and the color white. This field is described with reference to the "Chromaticity and Default White Point" in the VESA edid protocol, and its data structure is shown in Table F.18.

Table F.18 Color feature sub-field

Skew	Bit	Description
0Bh	1:0	Green y bits 1-0
	3:2	Green x bits 1-0
	5:4	Red y bits 1-0
	7:6	Red x bits 1-0
0Ch	1:0	White y bits 1-0
	3:2	White x bits 1-0
	5:4	Blue y bits 1-0
	7:6	Blue x bits 1-0
0Dh	7:0	Red x bits 9-2
0Eh	7:0	Red y bits 9-2
0Fh	7:0	Green x bits 9-2
10h	7:0	Green y bits 9-2
11h	7:0	Blue x bits 9-2
12h	7:0	Blue y bits 9-2
13h	7:0	White x bits 9-2
14h	7:0	White y bits 9-2

F.5.3 Video Timing Data Field

F.5.3.1 Overview

The following fields are mainly used to describe the Video Timing methods supported in the DCCD. If the adapter supports "Video Receiving Capability", at least one of the following fields must be included. If no video capability is supported, none of the following fields can be included.

The Video Timing data fields support four timing description methods, and users can freely declare the required field content as required.

- Timing Field of the CTA Standard for Consumer Devices

- Timing Field of the DMT Standard for Display Devices
- Timing Field of the CVT Standard
- Detailed Timing Description Field

F.5.3.2 CTA Timing Field

This field is used to describe the Video Timing information supporting CTA protocol in DCCD, and its data structure is shown in Table F.19.

Table F.19 CTA timing field

Address	Bit	Description
00h	7:0	Field identifier Default value: 0x11h
01h	7:4	Default value: 0
	3:0	Field version Default value: 1
02h	7:0	Field length Valid data length Value range: 0–0xFD
03h	7:0	CTA Timing Code 1
…		
02h+n	7:0	CTA Timing Code n

The CTA Timing Code refers to the VIC number in Table 1 Video Format Timings of the CTA-861-H protocol. If an unknown VIC is identified during peer parsing, the VIC Code can be ignored and subsequent parsing can be continued.

The VIC Code in the CTA Timing Field is arranged in a preferred order, that is, the first CTA Timing is the most recommended Timing in the CTA Timing Field, the second is the second recommended, and so on.

F.5.3.3 DMT Timing Field

This field is used to describe the Video Timing information supporting DMT protocol in DCCD, and its data structure is shown in Table F.20.

Table F.20 DMT timing field

Address	Bit	Description
00h	7:0	Field identifier Default value: 0x12h
01h	7:4	Default value: 0
	3:0	Field version Default value: 1

Address	Bit	Description
02h	7:0	Field lengthValid data lengthValue range: 0–0xFD
03h	7:0	DMT Timing Code 1
…		
02h+n	7:0	DMT Timing Code n

For DMT Timing Code, refer to the DMT ID Codes in Table 2-1 of DMT Protocol Version 1.0 and Revision 13. If unknown DMT ID Codes are identified during peer parsing, the Codes can be ignored and subsequent parsing can be continued.

The VIC Code in the DMT Timing Field is arranged in a preferred order, that is, the first DMT Timing is the most recommended Timing in the DMT Timing Field, the second is the second recommended, and so on.

F.5.3.4 CVT Timing Field

This field is used to describe the Video Timing information supporting CVT protocol in DCCD, and its data structure is shown in Table F.21.

Table F.21 CVT timing description field

Address	Bit	Description
00h	7:0	Field identifier Default value: 0x13h
01h	7:4	Default value: 0
	3:0	Field version Default value: 1
02h	7:0	Field length Valid data length, whose value must be a multiple of 7
09h to 03h	55:0	Detailed description of CVT Timing 1 Fixed length of 7 bytes
10h to 0Ah	55:0	Detailed description of CVT Timing 2 Fixed length of 7 bytes
…		

For a detailed description of CVT Timing, refer to Byte10-4 in Table 110 DisplayID Type X Video Timing Data Block (T10VTDB), T10_M=1 in the CTA-861-H protocol.

3D Timing is not currently supported, and the default value of 3D_Support is 0.

F.5.3.5 Detailed Timing Description Field

This field declares the video Timing information supported by a device by describing the details of Timing. The field is used only if the Timing required to be declared does not meet the declaration in the CTA, DMT, and CVT standard protocols. The data structure is shown in Table F.22.

Table F.22 Detailed timing description field

Skew	Bit	Description
00h	7:0	Field identifier Default value: 0x15h
01h	7:4	Default value: 0
	3:0	Field version Default value: 1
02h	7:0	Field length Valid data length, whose value must be a multiple of 21
17h to 03h	159:0	Detailed description of Timing 1
2Ch to 18h	159:0	Detailed description of Timing 2
…		

The field length must be a multiple of 21, and the detailed description structure of Timing is shown in Table F.23.

Detailed description of current Timing supports a maximum pixel clock of 4,294,967.295 MHz. The maximum width and height of Timing is 65,535.

The "Timing Details" in the Detailed Timing Description Field are arranged in order of preference, that is, the first Detailed Timing is the most recommended Timing, the second is the second recommended, and so on.

Table F.23 Timing details

Skew	Bit	Description
01h:00h	Number of active horizontal pixels	
00h	7:0	Number of horizontal effective pixels 7:0 bit Value range: 0–0xFF
01h	7:0	Number of horizontal effective pixels 15:8 bit Value range: 0–0xFF
03h:02h	Number of pixels in the horizontal blanking area	
02h	7:0	Number of pixels in the horizontal blanking area 7:0 bit Value range: 0–0xFF
03h	7:0	Number of pixels in the horizontal blanking area 15:8 bit Value range: 0–0xFF
05h:04h	Number of front porch pixels in the horizontal blanking area	

Skew	Bit	Description
04h	7:0	Horizontal blanking area front shoulder 7:0 bit Value range: 0–0xFF
05h	7:0	Horizontal blanking area front shoulder 15:8 bit Value range: 0–0xFF
07h:06h	HSync pixel number & horizontal synchronization polarity	
06h	7:0	Number of HSync pixels 7:0 bit Value range: 0–0xFF
07h	7	Horizontal synchronization polarity: ● 0 = negative polarity; ● 1 = positive polarity.
	6:0	Number of HSync pixels 14:8 bit Value range: 0–0x7F
09h:08h	Number of vertical effective lines	
08h	7:0	Number of vertical effective image lines 7:0 bit Value range: 0–0xFF
09h	7:0	Number of vertical effective lines 15:8 bit Value range: 0–0xFF
0Bh:0Ah	Number of lines in the vertical blanking area	
0Ah	7:0	Number of lines in the vertical blanking area 7:0 bit Value range: 0–0xFF
0Bh	7:0	Number of lines in the vertical blanking area 15:8 bit Value range: 0–0xFF
0Dh:0Ch	Number of front porch lines in the vertical blanking area	
0Ch	7:0	Vertical blanking area front shoulder 7:0 bit Value range: 0–0xFF
0Dh	7:0	Vertical blanking area front shoulder 15:8 bit Value range: 0–0x7F
0Fh:0Eh	VSync line number & vertical synchronization polarity	
0Eh	7:0	VSync line number 7:0 bit Value range: 0–0xFF
0Fh	7	Vertical synchronization polarity: ● 0 = negative polarity; ● 1 = positive polarity.
	6:0	VSync line number 14:8 bit Value range: 0–0xFF
13h:10h	Pixel clock (in kHz) For example, 297.000 MHz, the data is 28h 88h 04h 00h and a maximum of 4,294,967.295 MHz is supported.	

Skew	Bit	Description
10h	7:0	Pixel clock 7:0 bit Value range: 0–0xFF
11h	7:0	Pixel clock 15:8 bit Value range: 0–0xFF
12h	7:0	Pixel clock 23:16 bit Value range: 0–0xFF
13h	7:0	Pixel clock 31:24 bit Value range: 0–0xFF
14h	Timing mark	
	7:5	0h = reserved bit
	4	Video scanning mode: ● 0 = progressive scanning; ● 1 = interlaced scanning
	3:0	Ratio mode: ● 0h = the aspect ratio calculated based on pixel dimensions; ● 1h = 4:3 ● 2h = 16:9 ● 3h = 64:27 ● 4h = 256:135 ● 5h to Fh = reserved bit

F.5.4 Display Basic Capability Field

This field mainly describes the basic pixel encoding format and color depth information supported by a device. If the adapter supports "Video Receiving Capability", the following field type must be supported. If video capabilities are not supported, the following field cannot be included. The data structure is shown in Table F.24.

Table F.24 Display basic capability field

Skew	Bit	Description
00h	7:0	Field identifier Default value: 0x16h
01h	7:4	Default value: 0
	3:0	Field version Default value: 1
02h	7:0	Field length Valid data length fixed to 12h
06h to 03h	31:0	Maximum bandwidth supported by video (in kbps) If it is 0, it means there are no additional restrictions and the input video clock is only up to the input bandwidth.

Skew	Bit	Description
		If it is not 0, it means that the input video clock depends not only on the input bandwidth, but also cannot exceed this input value.
0Ah to 07h	31:0	The maximum pixel clock supported by video (in kHz) If it is 0, it means there are no additional restrictions on the pixel clock. If it is not 0, it means that the input video clock cannot exceed this input value.
0Bh	RGB bit width capability. If it is 0, it means that the RGB format is not supported.	
	7:4	Default value: 0; reserved bit
	3	Whether to support 16-bit width
	2	Whether to support 12-bit width
	1	Whether to support 10-bit width
	0	Whether to support 8-bit width
0Ch	YCbCr 4:4:4 bit width capability. If it is 0, it means that the YCbCr 4:4:4 format is not supported.	
	7:4	Default value: 0; reserved bit
	3	Whether to support 16-bit width
	2	Whether to support 12-bit width
	1	Whether to support 10-bit width
	0	Whether to support 8-bit width
0Dh	YCbCr 4:2:2 bit width capability. If it is 0, it means that the YCbCr 4:2:2 format is not supported.	
	7:4	Default value: 0; reserved bit
	3	Whether to support 16-bit width
	2	Whether to support 12-bit width
	1	Whether to support 10-bit width
	0	Whether to support 8-bit width
0Eh	Y Only bit width capability. If it is 0, it means that the Y Only format is not supported.	
	7:4	Default value: 0; reserved bit
	3	Whether to support 16-bit width
	2	Whether to support 12-bit width
	1	Whether to support 10-bit width

Skew	Bit	Description
	0	Whether to support 8-bit width
0Fh	ARGB bit width capability. If it is 0, it means that the ARGB format is not supported.	
	7:4	Default value: 0; reserved bit
	3	Whether to support 16-bit width
	2	Whether to support 12-bit width
	1	Whether to support 10-bit width
	0	Whether to support 8-bit width
10h	RAW bit width capability. If it is 0, it means that the RAW format is not supported.	
	7:4	Default value: 0; reserved bit
	3	Whether to support 16-bit width
	2	Whether to support 12-bit width
	1	Whether to support 10-bit width
	0	Whether to support 8-bit width
11h	Colorimetry support capability 1	
	7	AdobeYCC601 format support
	6	AdobeRGB format support
	5	sYCC601 format support
	4	xvYCC709 format support
	3	xvYCC601 format support
	2	ITU-R BT.709 format support
	1	ITU-R BT.601 format support
	0	sRGB format support
12h	Colorimetry support capability 2	
	7:5	Default value: 0; reserved bit
	4	DICOM Part 14 format support (Used in Y Only format)
	3	DCI-P3 format support
	2	BT2020YCC format support
	1	BT2020cYCC format support
	0	BT2020RGB format support

Skew	Bit	Description
13h	7:0	Default value: 0; reserved bit
14h	7:0	Default value: 0; reserved bit

06h–03h indicates the maximum bandwidth (in kbps) supported by video. If the value is not 0, the declared Timing information corresponding to the bit width information of all description Timing field information cannot exceed the value.

For example, the Timing information declares support for 4K 60Hz (RGB 8bit clock bandwidth: 14.256 Gbps), and the RGB format declares support for 8/10/12 bit width. If the maximum bandwidth supported by video is limited to 20 Gbps, which corresponds to 4K 60Hz Timing, its RGB format can only support 10 bits at most (10-bit clock bandwidth: 17.82 Gbps), but cannot support 12 bits (12-bit clock bandwidth: 21.384 Gbps).

07h–0Ch respectively describe the color bit width information supported by various basic pixel coding formats. If it is 0, it means that the pixel coding format is not supported. If the encoding format supports high bit rate width, it must support low bit rate width. For example, if RGB supports 10 bits of bit width, then RGB must support 8 bits of bit width.

Each bit from 0Ch to 0Dh describes the Colorimetry type supported by each pixel encoding format. If the bit value is 0, it means that the corresponding Colorimetry type is not supported.

F.5.5 YCbCr 4:2:0 Capability Field

F.5.5.1 Overview

This field mainly describes the device's capability to support receiving YCbCr 4:2:0. If an adapter supports "Video Receiving Capability" and the device supports YCbCr 4:2:0 pixel encoding capability, this field must be included. If YCbCr 4:2:0 is not supported, the following field must not be included. Its data structure is shown in Table F.25.

Table F.25 YCbCr 4:2:0 capability field

Skew	Bit	Description	Remarks
00h	7:0	Field identifier Default value: 0x17h	
01h	7:4	Default value: 0	
	3:0	Field version Default value: 1	
02h	7:0	Field length Valid data length Value range: 5–0xFD	
03h	YCbCr 4:2:0 bit width capability. If it is 0, it means that the YCbCr 4:2:0 format is not supported.		
	7:4	Default value: 0; reserved bit	
	3	Whether to support 16-bit width	
	2	Whether to support 12-bit width	
	1	Whether to support 10-bit width	

Skew	Bit	Description	Remarks
	0	Whether to support 8-bit width	
07h to 04h	31:0	YCbCr 4:2:0 supports the minimum pixel clock (in KHz)	Refer to 3.6.1.
08h+L1–08h	7:0	YCbCr 4:2:0 Timing description 1	Refer to 3.6.2.
09h+L1+L2–09h+L1	7:0	YCbCr 4:2:0 Timing description 2	
…			

If the "Maximum Bandwidth Supported by Video" (06h to 03h) in the "Display Basic Capability Field" is not 0, the Timing supported by YCbCr 4:2:0 is also limited by the "Maximum Bandwidth Supported by Video".

03h indicates the bit width capability currently supported by YCbCr 4:2:0. If it is 0, it indicates that the YCbCr 4:2:0 format is not currently supported.

F.5.5.2 YCbCr 4:2:0 Minimum Pixel Clock

This byte mainly describes the minimum pixel clock size required by Timing supported by YCbCr 4:2:0. The Timing supported by YCbCr 4:2:0 comes from all the Timing information declared in the Video Timing data field and the Video Data Block extended in the CTA extension field, such as the declaration that the Timing meets or exceeds the minimum pixel clock timing requirement. The Timing can support the YCbCr 4:2:0 format.

For example, the Timing information declares that 4K 60Hz is supported (YCbCR 4:2:0 8-bit clock bandwidth: 297 MHz), and YCbCR 4:2:0 declares that 8-bit/10-bit/12-bit widths are supported. If the minimum bandwidth supported by YCbCR 4:2:0 is 400,000 kHz, the corresponding 4K 60Hz Timing can only support 12 bits (YCbCR 420 12-bit clock bandwidth: 445.5 MHz). It does not support 10 bits (YCbCR 420 8-bit clock: 371.25 MHz) or 8 bits.

If the value is 0, it means that there is no minimum pixel clock limit for YCbCr 4:2:0, and all declared Timing can support the YCbCr 4:2:0 format.

F.5.5.3 YCbCr 4:2:0 Timing Description

This sub-field mainly describes the list that supports YCbCr 4:2:0 Timing separately. The Timing in the list only supports the timing of YCbCr 4:2:0 and does not support other pixel encoding formats. If it is not necessary to declare that the YCbCr 4:2:0 Timing function is supported separately, it may not be declared. Its structure is shown in Table F.26.

Table F.26 YCbCr 4:2:0 timing description

Skew	Bit	Description
00h	7:5	Timing protocol identifier. See Table F.27.
	4:0	Timing Code list length of YCbCr 4:2:0.
01h	7:0	YCbCr 4:2:0 Timing Code 1
02h	7:0	YCbCr 4:2:0 Timing Code 2

Skew	Bit	Description
…		
n	7:0	YCbCr 4:2:0 Timing Code *n*

The Timing type identifier is used to declare the protocol type of the Timing standard.

Table F.27 Timing protocol identifier

Address	Bit	Description
00h	7:5	Timing protocol identifier: ● 0: CTA protocol Timing ● 1: DMT protocol TimingDefault value: 0x18h ● Others: reserved

The 00h 4:0 bit is the list length that supports YCbCr 4:2:0 Timing, and the subsequent bytes are the CTA Timing Codes supported by YCbCr 4:2:0. A maximum of 32 Timing Codes are supported.

Codes for YCbCr 4:2:0 Only Timing are arranged in a preferred order, that is, the first Timing Code is the most recommended Timing in the protocol identifier, the second is the second recommended, and so on.

In addition, if the optimal Timing of a device is YCbCr 4:2:0 Only Timing, this field needs to be arranged at the forefront of all Timing description fields.

F.5.6 Video Capability Field

F.5.6.1 Overview

This field mainly describes the video-related capabilities supported by a device. If the adapter supports "Video Receiving Capability", this field must be included; otherwise, this field is not selectable. Its data structure is shown in Table F.28.

Table F. 28 Video capability field

Skew	Bit	Description	Remarks
00h	7:0	Field identifier Default value: 0x18h	
01h	7:4	Default value: 0	
	3:0	Field version Default value: 1	
02h	7:0	Field length Valid data length fixed to 04h	
03h	7:6	Reserved bit; default value: 0	
	5	S_PT1	
	4	S_PT0	

Skew	Bit	Description	Remarks
	3	S_IT1	
	2	S_IT0	
	1	S_CE1	
	0	S_CE0	
04h	7:4	Reserved bit; default value: 0	
	3	Whether the YCC Full quantization range is supported	
	2	Whether the YCC Limit quantization range is supported	
	1	Whether the RGB Full quantization range is supported	
	0	Whether the RGB Limit quantization range is supported	
05h	7:4	Reserved bit; default value: 0	
	3	Game Type Support	See F.5.6.2.
	2	Cinema Type Support	
	1	Photo Type Support	
	0	Graphics Type Support	
06h	7:2	Reserved bit; default value: 0	
	1	SR support	
	0	MEMC support	

The 03h byte mainly describes the overscan/underscan capability of a display, which can be separately declared for the Preferred, IT, and CE video formats. This field is described with reference to the function description of Video Capability Descriptor Data Byte 3 in 7.5.6 Video Capability Data Block of CTA-861-H protocol.

The 04h byte describes the quantization range capability supported by RGB and YCC. RGB or YCC can set the quantization range capability, respectively.

For example, if a device declares that it only supports the RGB Limit quantization range, the quantization range of the RGB video output by the transmitting end can only adopt the Limit mode.

At the same time, when the transmitting end sends a video, it needs to clearly inform the receiving end of the specific quantization range description of the video through the information frame.

F.5.6.2 Content Type

This sub-field is used to describe the video content information of a device, and each bit indicates the support of a specific content type. When the video content sent by the transmitting end is

graphics, photos, movies, or games, the receiving end has the processing method related to the corresponding type.

The supported function types refer to the function description of IT Content Type in CTA-861-H protocol.

F.5.7 Advanced Video Feature Capability Field

This field describes the advanced video capability information. If an adapter supports "Video Receiving Capability", the field can be declared. If video receiving capability is not supported, the following field cannot be included. Its data structure is shown in Table F.29.

Table F.29 Advanced video capability information structure

Skew	Bit	Description	Remarks
00h	7:0	Field identifier Default value: 0x19h	
01h	7:4	Default value: 0	
	3:0	Field version Default value: 1	
02h	7:0	Field length Valid data length fixed to 04h	
03h	7:4	Reserved bit; default value: 0	
	3	Whether smooth switching (SSW) is supported	
	2	Whether DFR/VRR is supported	DFR is the dynamic frame rate refresh feature of the GPMI.
	1	QVT support	
	0	ALLM support	
04h	7:0	Maximum refresh frame rate 7:0 bit	Refresh frame rate range supported by dynamic frame rate refresh If the dynamic frame rate refresh feature is not supported, the value shall be 0.
05h	7:2	Minimum refresh frame rate	
	1:0	Maximum refresh frame rate 9:8 bit	
06h	7:0	Smooth switching delay	Valid when smooth switching (SSW) is supported. The bit indicates that the switching of the sink device needs to be completed after Field Value (ms). When smooth switching (SSW) is not supported, the value shall be 0.

If "Yes" is selected for "QVT support" in 03h, the magnification video clock output by the device is also limited by the "Maximum Bandwidth Supported by Video (unit: KHz)" in the "Display Basic Capability Field".

F.5.8 Static HDR Capability Field

This field describes static HDR capability information. If the adapter supports "Video Receiving Capability", this field type is allowed to be declared. If video receiving capability is not supported, the following field cannot be included. Its data structure is shown in Table F.30.

Table F.30 Static HDR capability field

Skew	Bit	Description	Remarks
00h	7:0	Field identifier Default value: 0x1Ah	
01h	7:4	Default value: 0	
	3:0	Field version Default value: 1	
02h	7:0	Field length Valid data length fixed to 05h	
03h	7:0	TF	
04h	7:0	SM	
05h	7:0	Desired Content Max Luminance data (8 bits)	
06h	7:0	Desired Content Max Frame-average Luminance data (8 bits)	
07h	7:0	Desired Content Min Luminance data (8 bits)	

The 03h–07h field description information refers to 7.5.13 HDR Static MetaData Data Block in CTA861-H protocol.

F.5.9 Dynamic HDR Capability Field

This field describes dynamic HDR capability information. If the adapter supports "Video Receiving Capability", this field type is allowed to be declared. If video receiving capability is not supported, the following field cannot be included. Its data structure is shown in Table F.31.

Table F.31 Dynamic HDR capability field

Skew	Bit	Description	Remarks
00h	7:0	Field identifier Default value: 0x1Bh	
01h	7:4	Default value: 0	
	3:0	Field version Default value: 1	
02h	7:0	Field length	

Skew	Bit	Description	Remarks
		Valid data length (L)	
03h to 03h+L	7:0	HDR dynamic metadata type	

The 03h–03h+L field description information refers to 7.5.14 HDR Dynamic MetaData Byte (3–Lh) Data Block in CTA-861-H protocol.

F.5.10 HDR Vivid Capability Field

The field function is reserved.

F.5.11 Perceptual Lossless Compression (PLLC) Capability Field

This field describes the advanced video capability information. If an adapter supports "Video Receiving Capability", the field can be declared. If video receiving capability is not supported, the following field cannot be included. Its data structure is shown in Table F.32.

Table F.32 PLLC capability field

Address	Bit	Description	Remarks
00h	7:0	Field identifier Default value: 0x20h	
01h	7:4	Default value: 0	
	3:0	Field version Default value: 1	
02h	7:0	Field length Valid data length fixed to 14h	
03h	7:4	PLLC major version Indicates the major version of PLLC.	For PLLC v1.2 and PLLC v1.2a, the PLLC major version and minor version field values are 1h and 2h, respectively.
	3:0	PLLC minor version Indicates the minor version of PLLC	
04h	7:0	Bitstream level; bitstream of the reference bitstream level: ● 0x20: frame buffer level ● 0x30: port level ● Other value: reserved bit	Indicates the level of the bitstream
05h	7:3	Reserved.	
	2:0	RC buffer block size; bit rate control buffer block size: ● 000b = 1 KB ● 001b = 4 KB ● 010b = 16 KB	

Address	Bit	Description	Remarks
		● 011b = 64 KB ● Other value: reserved bit	
06h	7:0	Size of each RC buffer, in blocks	Buffer size (in blocks) = Register value + 1. The actual buffer size is calculated by multiplying the RC buffer size (in blocks) by the block size indicated by the RC buffer block size field in the PLLC RC buffer block size register. Note: It is the maximum algorithmic RC buffer size that the sink device may support and is used by the source device to ensure required latency. Example: If the RC buffer block size is programmed to 011b (64 KB) and the rate buffer size (in blocks) is 0111 1111 (128), then each RC buffer size = 64 KB × 128 = 8 MB.
07h	7	Each PLLC receiver device supports 16 cross-level slices.	The PLLC slice partitioning function is used to indicate the number of cross-level slice partitions that a PLLC receiver device can support. 0 = Number of cross-level slices that each PLLC receiver device does not support 1 = Number of cross-level slices that each PLLC receiver device supports
	6	Each PLLC receiver device supports 12 cross-level slices.	
	5	Each PLLC receiver device supports eight cross-level slices.	
	4	Each PLLC receiver device supports six cross-level slices.	
	3	Each PLLC receiver device supports four cross-level slices.	
	2	Each PLLC receiver device supports three cross-level slices.	
	1	Each PLLC receiver device supports two cross-level slices.	
	0	Each PLLC receiver device supports one cross-level slice.	
08h	7:0	bits_per_pixel7:0 The PLLC receiver device supports a maximum of 8 lower significant bits per pixel.	Maximum number of bits per pixel after compression supported by decompression is bits_per_pixel Maximum number of bits per pixel supported by the PLLC sink device. The format is U8.4 (unsigned, four decimal places) with a total of 12 bits, of which the higher 8 bits are integer parts and the lower 4 bits are decimal
09h	7:4	Reserved.	
	3:0	bits_per_pixel9:8 The PLLC receiver device	

Address	Bit	Description	Remarks
		supports a maximum of 4 higher significant bits per pixel.	parts. Actual bits_per_pixel = Higher 8 bits + Lower 4 bits × 1/16. Taking 0x081 as an example, Actual bits_per_pixel = 8 + 1 × 1/16 = 8.0625.
0Ah	7:4	Reserved.	PLLC decoder color format
	3	YCbCr 4:2:0 support: ● 0 = YCbCr 4:2:0 supported ● 1 = YCbCr 4:2:0 not supported	
	2	YCbCr 4:2:2 support: ● 0 = YCbCr 4:2:2 supported ● 1 = YCbCr 4:2:2 not supported	
	1	YCbCr 4:4:4 support: ● 0 = YCbCr 4:4:4 not supported ● 1 = YCbCr 4:4:4 supported	
	0	RGB444 support. The PLLC receiver device shall support the RGB44 color format: ● 0 = RGB444 not supported; ● 1 = RGB444 supported.	
0Bh	7:5	Reserved	PLLC decoder color depth
	4	16-bit color depth supports 16 bpc: ● 0 = 16 bpc not supported; ● 1 = 16 bpc supported.	
	3	12-bit color depth supports 12 bpc: ● 0 = 12 bpc not supported; ● 1 = 12 bpc supported.	
	2	10-bit color depth supports 10 bpc 0 = 10 bpc not supported; 1 = 10 bpc supported.	
	1	8-bit color depth supports 8 bpc 0 = 8 bpc not supported; 1 = 8 bpc supported.	
	0	Reserved	

Address	Bit	Description	Remarks
0Ch	7:0	The PLLC peak throughput (PLLC sink device) The decoder has four pixels per clock in megapixels per second (MP/s). The value is expressed in decimal.	0 = Not supported. Throughput = Configuration value × 50 MP/s.
0Dh	7:0	PLLC maximum image width	Maximum image width = Configuration value × 320
0Eh	7:0	PLLC maximum slice width	Maximum slice width = Configuration value × 320
0Fh	7:4	Reserved	
	3:0	The increment of bits per pixel supported by the decompressor, and the incremental accuracy of bits per pixel supported by the PLLC sink device are as follows: ● 0000b = 1/16 bpp; ● Other values are reserved.	
10h	7	Reserved	PLLC encoder configuration
	6	padding_type_flag (0: uniform padding; 1: padding on the right. Currently, only right padding is supported.)	
	5	padding_stuff_flag	
	4	bwq_enable_flag	
	3	pwq_enable_flag	
	2	ibc_enable_flag	
	1	brc_enable_flag	
	0	dp_enable_flag	
11h	7:0	substream_segment_size7:0 The lower 8 bits of substream_segment_size supported by the PLLC sink device	Sub-stream segment size supported by the PLLC decoder substream_segment_size supported by the PLLC sink device
12h	7:0	substream_segment_size9:8 The higher 8 bits of substream_segment_size supported by the PLLC sink device	
13h	7:0	chunk_size_in_cu7:0 7_0 of the largest chunk_size_in_cu supported by	Maximum chunk_size_in_cu supported by the PLLC decompressor

Address	Bit	Description	Remarks
		the PLLC sink device	Maximum chunk_size_in_cu supported by the PLLC sink device
14h	7:0	chunk_size_in_cu15:8 15_8 of the largest chunk_size_in_cu supported by the PLLC sink device	
15h	7:0	chunk_size_in_cu23:16 23_16 of the largest chunk_size_in_cu supported by the PLLC sink device	
16h	7:0	chunk_size_in_cu31:24 31_24 of the largest chunk_size_in_cu supported by the PLLC sink device	

F.5.12 Audio Capability Field

This field describes audio capabilities. If the adapter supports "Audio Receiving Capability", the following field types must be supported. If audio capabilities are not supported, the following field cannot be included. Its data structure is shown in Table F.33.

Table F.33 Audio capability field

Skew	Bit	Description	Remarks
00h	7:0	Field identifier Default value: 0x40h	
01h	7:4	Default value: 0	
	3:0	Field version Default value: 1	
02h	7:0	Field length Valid data length. The length must be a multiple of 3, and its value range is 0–0xFD.	
05h to 03h	23:0	CTA short Audio Descriptor 1	
08h to 06h	23:0	CTA short Audio Descriptor 2	
…			

CTA short Audio Descriptor refers to the description in 7.5.2 Audio Data Block of CTA-861-H protocol.

F.5.13 Speaker Capability Field

This field describes speaker capabilities. If the adapter supports "Audio Receiving Capability", the following field types must be supported. If audio capabilities are not supported, the following field cannot be included. Its data structure is shown in Table F.34.

Table F.34 Speaker capability field

Skew	Bit	Description	Remarks
00h	7:0	Field identifier Default value: 0x41h	
01h	7:4	Default value: 0	
	3:0	Field version Default value: 1	
02h	7:0	Field length Valid data length fixed to 3	
05h to 03h	23:0	Speaker allocation data block	

Currently, a maximum of 32 channels are supported. The speaker allocation field is described with reference to Table 69 in 7.5.3 Speaker Allocation Data Block of CTA-861-H protocol.

F.5.14 Advanced Audio Capability Field

The field function is reserved.

F.5.15 CTA Extension Field

This field describes how the CTA data block is encapsulated into the DCCD frame, and its corresponding field identifier is 0xFD. Its data structure is shown in Table F.35.

Table F.35 CTA extension field

Skew	Bit	Description	Remarks
00h	7:0	Field identifier Default value: 0xFDh	
01h	7:4	Default value: 0	
	3:0	Field version Default value: 1	
02h	7:0	Field length Value range: 0–0xFD	
03h	7:5	CTA identification code	
	4:0	CTA data block length (n)	
04h	7:0	CTA data block description 1	
05h	7:0	CTA data block description 2	
…			
03h+n	7:0	CTA data block description *n*	

03h–03h + n data is a data block located in a CTA-861-H, which can multiplex the data blocks defined in CTA861-H, such as audio and video, speakers, colorimetry, and others.

There can be multiple CTA extension fields in the protocol. Each extension field can only describe one CTA data block. If multiple CTA data blocks need to be declared, they can be declared in the protocol multiple times, and each extension field contains one CTA data block.

Note:

- When the field is parsed, if an unidentified CTA identification code is parsed, the data block containing the unidentified CTA identification can be directly ignored, and the parsing of other data fields cannot be affected.
- If the same content is declared multiple times, for example, the audio capability is declared in both the audio capability field and the CTA extension field, both of them need to be parsed, and the collection of the two functions is taken.

F.5.16 Vendor Specific Data Field

This field is used by the vendor to declare a privately customized capability. It can be customized for different vendors. Different vendors can use this field to pass the message they wish to convey.

When a private customization is declared, the device capability described by this field shall be declared. The vendor name of this private field shall be described by a three-byte Vendor ID, which indicates the unique vendor name.

The data structure shall be as shown in Table F.36.

Table F.36 Vendor-specific data field

Skew	Bit	Description	Remarks
00h	7:0	Default value: 0xFEh	Field identifier: 0xFEh
01h	7:4	Default value: 0	
	3:0	Field version Default value: 1	
02h	7:0	Field length Value range: 3–0xFD	
03h	7:0	Device capability type	The field describes the capability type mainly described by the customization field. See Table F.37 for details.
04h	7:0	Vendor ID byte 1	The 04h–06h Vendor ID code in the field is used to identify the specific codes of different vendors.
05h	7:0	Vendor ID byte 2	
06h	7:0	Vendor ID byte 3	
07h	7:0	Vendor-specific data 1	Vendor customization private information
08h	7:0	Vendor-specific data 1	
…			
06h+n	7:0	Vendor-specific data *n*	

The device capability types are described in Table F.37. For screen devices, the default value is fixed to 5.

Table F.37 Device capability type

Value	Description
1	General-purpose interface extension capability
2	Video capability
3	Audio capability

F.6 Strategies and Precautions

F.6.1 Display Receiving Capacity Data Block Strategies

If a device supports "Video Receiving Capability", the "Display Parameter Field", "Display Basic Capability Field", and "Video Capability Field" in the detailed data segment are required fields.

If a device supports "Video Receiving Capability", at least one field reflecting Video Timing shall be declared, and the field can be any one or more of the following:

(1) CTA Timing field

(2) DMT Timing Field

(3) CVT Timing field

(4) Detailed Timing Description Field

(5) Extended video data block in CTA extension field

If a device supports "Audio Receiving Capability", at least one of the "Audio Capability Field" in the data segment and the extended "Audio Data Block" in the "CTA Extension Field" shall be declared.

If a device supports "Audio Receiving Capability", at least one of the "Speaker Capability Field" in the data segment and the extended "Speaker Allocation Data Block" in the "CTA Extended Field" shall be declared.

If a device only supports receiving audio, it must not contain any video-related capability declaration. Through capability parsing, the peer end can obtain the pure audio receiving capability it currently supports. The same applies to the pure video receiving capability declaration.

F.6.2 Timing Optimization Strategies

The Video Timing supported by the device declaration currently supports four declaration methods as follows:

- CTA Timing field;
- DMT Timing Field;
- CVT Timing field;
- Detailed Timing description field;
- Extended video data block in CTA extension field.

In the protocol data structure, the first Video Timing parsed sequentially is the preferred Timing recommended by the device. The Video Timing parsed can be any of the four declarations mentioned above. If the sending device supports this preferred Timing, it should be output first by default.

For example, if the device is a TV display device, it is recommended that the CTA Timing Field be placed ahead of all Video Timings, and the CTA Timing is recommended first.

If a device is a display device such as a computer monitor, it is recommended that the DMT Timing Field be placed ahead of all Video Timings, and DMT Timing is recommended first.

In addition, the Video Timing in the above four declaration fields is also arranged according to the preferred internal order, that is, the first Timing in a field is the first preferred Timing of the field, the second is the second preferred Timing, and so on.

If the declared preferred Timing is not supported by the peer end, the peer can select the second recommended Timing in this field or the first recommended Timing of other fields as the optimal Timing for the device.

F.6.3 GPMI Data Block Strategies

For a GPMI device, its capability declaration constraint policy is as follows:

- The basic segment description of the "Device", "Port", and "Adapter" of the same port describes the capabilities of the port, that is, the capability overview (BIT9:0) in the basic segment description of different types of capabilities of the same port must be consistent.
- Each DCCD contains at least one or more of the three types of "GPMI Device Description Field", "GPMI Port Description Field", and "GPMI Adapter Description Field".
- The "GPMI Device Description Field" can have only one field. Multiple fields are not allowed. If the "GPMI Device Description Field" exists, the "Product Information Field" must also be declared, and only one is allowed. And the "Product Information Field" must be the first data segment after the basic segment.
- If the GPMI contains device, port, and adapter fields that are not allowed to be mixed, the layout order must be device, port, and adapter.
- If the GPMI adapter is an "Audio and Video Receiving Adapter", refer to the "Display Receiving Capability Data Block Strategies" in the above for its constraints.

F.6.4 Protocol Parsing Strategies

The DCCD protocol parsing strategies are as follows:

- The remaining protocol data can be obtained by reading the fixed-length (12 bytes) basic segment information and parsing the "Data Length" information in the basic segment.
- To split a data block, follow the steps: Parse the "Field Length" information of the "Data Block Header" (three bytes) to obtain the length of the data block, and then parse the content of the data block to obtain its position skew to the next data block. If the "Field Length" is greater than the remaining data length, the field is ignored, the content parsed before is retained, and the parsing is finished.
- If the parsed data block contains an unknown "Field Identifier", that block is ignored and the parsing of subsequent blocks cannot be affected. Processing continues directly with the next data block.
- For GPMI data description, GPMI device, GPMI port, and GPMI adapter data blocks shall be identified and parsed through "Field Identifier". Based on the guide of "Adapter Detailed

Function Description Data Length" in "GPMI Adapter Data Block", gradually parse the detailed function data block of the corresponding adapter.

- If there is an unknown ID Code in the "CTA Timing Field" or "DMT Timing Field", the ID Code can be directly ignored, but it cannot affect parsing subsequent ID Code.
- If similar capabilities are declared multiple times, for example, the audio capability is declared in both the audio capability field and the CTA extension field, both of them need to be parsed, and the collection of the two functions is taken.
- For the capability declaration of a GPMI device, different adapters can quickly jump to the next adapter description field through the "Adapter Detailed Function Description Data Length" in the "GPMI Adapter Description Field".

Bibliography

[1] T/SUCA 031-2022, Technical Specification of Advanced Digital Content Protection System Part I: Technical Specification of Content Protection for UMMI

[2] T/AI 129.1-2024, Information technology — Perceptual lossless compression — Part 1: Image