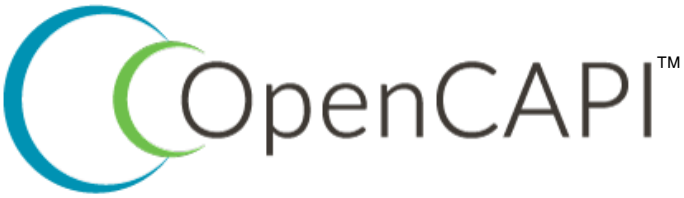


www.opencapi.org

OpenCAPI Data Link Layer (DL 3.0/3.1/4.0) Architecture Specification

Version 2.0
9 July 2020

Approved

Approved for Distribution to OpenCAPI Members
Approved for Distribution to Non-Members for Learning Purposes Only

OpenCAPI Data Link Layer (DL 3.0/3.1/4.0) Architecture

OpenCAPI DL Architecture Specification Work Group
OpenCAPI Consortium

Version 2.0 (9 July 2020)

Abstract

This document provides the architectural specification of the OpenCAPI data link layer (DL and DLX).
It is the work product of the OpenCAPI Consortium DL Architecture Specification Work Group.

This document is handled in compliance with the requirements outlined in the OpenCAPI Consortium Work Group (WG) process document. Comments, questions, etc. can be submitted to membership@opencapi.org.

Participants

Brian Allison, IBM, *Chair*

Lonny Lambrecht, IBM

Mike J Palmer, IBM

Sanjay Goyal, Microchip

Contents

List of figures

List of tables

Revision log

Each release of this document supersedes all previously released versions. The change history log lists all significant changes made to the document since its initial release. The use of change bars and mark up notation may be included and are noted in the revision log.

Revision date	Summary of changes
9 July 2020	Version 2.0. • Revised *Section 2.6 Endpoint link speed discovery* on page 24. • Revised *Section 5.1 DL idle flit* on page 39.
8 July 2019	Initial version 2.0 of the OpenCAPI Data Link Layer (DL 3.0/3.1/4.0) Architecture Specification in the OpenCAPI Consortium template.
6 December 2019	Initial version 2.0 release with the latest updates and book creation.

About this document

This preface contains some general format information (for example, headings) and document conventions.

Architecture compliance terminology

In architecture descriptions, certain terms carry meaning in addition to their normal use in English. The following terms are used within this architecture specification to describe the requirements an implementation must meet to be considered compliant.

Table 1. Architecture terms

Term	Description
invalid	Used for multi-bit fields where the contents are not reliable. The field or bus shall not be examined for any functional or error checking actions.
may	An architectural option indicating that an implementation is allowed to have this behavior or characteristic.
reserved	With respect to a field of a register or bus: • A reserved field shall be set to 0 by an implementation. • A reserved field shall not be examined by an implementation. With respect to a code point: • A reserved code point shall not be issued by a compliant implementation • A reserved code point shall cause a bounded undefined response (that is, it won't hang the system). • A reserved code point may be used in future revisions of the architecture. The architecture may specify that the use of a reserved code point is an error condition.
shall	An architectural requirement indicating a required behavior or characteristic.
uncertain	Used for single-bit fields where the contents are not reliable. The field or bus shall not be examined for any functional or error checking actions.
undefined	When the value of a field or a bus is undefined, the value may vary between implementations and may vary for a particular implementation for different actions. An implementation shall not examine a field when its value is undefined for functional purposes. However, the field may be checked for errors in those cases where an implementation includes error checking (that is, parity, ECC and so on).

Conventions

The OpenCAPI Consortium documentation uses several typesetting conventions.

Bit and byte numbering

Throughout this document, little-endian notation is used, which means that bits and bytes are numbered in descending order from left to right.

Thus, in the description of a 4-byte field, bit 31 is the most significant bit (MSb) and bit 0 is the least significant bit (LSb). The corresponding byte numbering is also shown.

MSb																													LSb
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
byte 3								byte 2								byte 1								byte 0							

Big-endian and little-endian byte ordering are shown in *Figure 1* and described in the *POWER ISA, version 3.0, Book I.*

Figure 1. Big- and little-endian comparisons

LE	7	6	5	4	3	2	1	0
	Bit numbering within a byte							
BE	0	1	2	3	4	5	6	7

4-byte field with character data shown

LE	3	2	1	0
Content:	M	I	K	E
BE	0	1	2	3

Illustrating the difference between little endian and big endian storing to memory of the 4-byte field shown to the left.

Memory offset	LE stored	BE stored	
0	E	M	
1	K	I	
2	I	K	
3	M	E	

Representation of numbers

The notation for bit encoding is as follows:

- Hexadecimal values are preceded by x and enclosed in single quotation marks. For example x‘0A00’. Bit numbering is little endian and, in this example, is 15 to 0.
- Binary values in sentences are shown in single quotation marks. For example ‘1010’. Bit numbering in is little endian and, in this example, is 3 to 0.
- ^{n}x means the replication of x, n times. That is, x is concatenated to itself n-1 times. $^{n}0$ and $^{n}1$ are special cases:
  - $^{n}0$ means a field of n bits with each bit equal to 0. For example, $^{5}0$ is equivalent to ‘00000’.
  - $^{n}1$ means a field of n bits with each bit equal to 1. For example, $^{5}1$ is equivalent to ‘11111’.

RTL notation

RTL notations are used to specify the architectural transformation performed by execution of a command.

Notation	Meaning
←	Assignment.
||	Concatenation.
=, ≠	Equal, not equal relations.
≥, ≤	Greater than or equal to, less than or equal to relations.
+	Two's complement addition.
-	Two's complement subtraction, unary minus
∨	Bitwise logical OR
∧	Bitwise logical AND
⊕	Bitwise logical exclusive OR
Max(x,y)	Returns x when x ≥ y; otherwise, returns y.
Min(x,y)	Returns x when x ≤ y; otherwise, returns y.
{x...y}	All integer values from x through y.
A = {x...y}	Returns true when A is a member of the set of integer values in the range of x through y.

Notes

This section describes engineering, developer, and editor notes.

Engineering notes

Engineering notes provide additional implementation details and recommendations not found elsewhere. The notes might include architectural compliance requirements. That is, the text might include *Architecture compliance terminology*. These notes should be read by all implementation and verification teams to ensure architectural compliance.

Engineering note

This is an example of an Engineering note. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin cursus hendrerit enim, vel tempus nibh ornare ut. Quisque ac augue eu augue convallis hendrerit. Mauris iaculis viverra ipsum nec dapibus. Nunc at porta libero. Curabitur luctus ultrices augue non pulvinar. Vestibulum mattis non ipsum at venenatis. Suspendisse euismod, neque et suscipit luctus, odio metus semper lectus, quis volutpat est libero quis nunc. Vivamus rutrum mauris sed tristique malesuada. Vivamus at augue vitae nisl cursus feugiat.

Developer notes

Developer notes are used to document the reasoning and discussions that led to the current version of the architecture. These notes might also include recommended changes for future versions of the architecture, or warnings of approaches that have failed in the past. These notes should be read by verification teams and contributors to the architecture.

Developer note

This is an example of a Developer note. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin cursus hendrerit enim, vel tempus nibh ornare ut. Quisque ac augue eu augue convallis hendrerit. Mauris iaculis viverra ipsum nec dapibus. Nunc at porta libero. Curabitur luctus ultrices augue non pulvinar. Vestibulum mattis non ipsum at venenatis. Suspendisse euismod, neque et suscipit luctus, odio metus semper lectus, quis volutpat est libero quis nunc.

Terms

The following terms are used in this document.

ACK	Acknowledgment or acknowledge. A transmission that is sent as an affirmative response to a data transmission.
AFU	Attached functional unit. Architecturally, this term refers to an endpoint unit or resource. Communication from the processor to the AFU goes through a protocol stack, transaction layer (TL), data link layer (DL), and physical medium layer (PHY). Command and data packets at the AFU interface are specified by the AFU command/data interface, which is the interface between the AFU protocol stack and the AFU.
AFU protocol	AFU protocol layer. This layer currently consists of: AFU_C protocol layer AFU_M protocol
ASIC	Application-specific integrated circuit
CAPI	Coherent Accelerator Processor Interface.
CDR	Clock and data recovery.
CRC	Cyclic redundancy check.
DL	OpenCAPI data link layer on the host processor.
DLX	OpenCAPI data link layer on the external OpenCAPI device
ECC	Error correction code. A code appended to a data block that can detect and correct bit errors within the block.
FIR	Fault Isolation Register. Register bits that show which piece of hardware failed.
Flit	An acronym for flow control digits. Typically used in networking to specify the smaller pieces that a larger network layer packet is broken into. See FLITs. In OpenCAPI version 3.0, a flit is associated with the specification of a DL packet and is defined as a 64-byte unit of data. Control and data flits are specified.
FPGA	Field-programmable gate array.
Frame	A control flit followed by 0 - 8 data flits.
Gbps	Gigabits per second.
GHz	Gigahertz.
Inbound	Specifies the direction from the attached OpenCAPI device towards the attached processor chip.
LFSR	Linear Feedback Shift Register.
LP	Low power.
LX	Transaction layer external.
MMIO	Memory-mapped input/output. Refers to the mapping of the address space required by an I/O device for Load or Store operations into the system's address space.
NACK	Negative acknowledgment.

OCDE	OpenCAPI device enable.
OL	Outside lane.
Outbound	Specifies the direction from the processor chip to the attached OpenCAPI device.
Packet	A TL or TLX unit of information. A command packet contains commands. A response packet contains response information. See the specification of these in the *OpenCAPI 3.0 Transaction Layer Specification*. Data is transferred in address-aligned 64-byte packets.
PHY	The Host PHY layer interfaces to the DL and the network. This is the bit stream level that specifies the electrical and optical transmission medium as well as the network interconnect topology. The current specification for the network is a point-to-point connection.
PHYX	On the OpenCAPI device, the PHYX layer interfaces to the DLX and the network. This is the bit stream level that specifies the electrical and optical transmission medium as well as the network interconnect topology. The current specification for the network is a point-to-point connection.
PLL	Phase-locked loop.
P/N	Positive/negative.
PRBS23	Pseudo-random binary sequence 23.
Reserved	A field that might be defined at a later date. Senders are not required to drive zeros, although that is good practice. Receivers must ignore reserved fields except for the CRC.
Response packet	The Host TL construct that contains response information to commands. Used for Host TL-to-Device TLX and Device TLX-to-Host TL communication.
RX	Receiver.
SerDes	Serializer/deserializer.
TL	OpenCAPI transaction layer found on the host processor. Interfaces to the DL and the protocol layer. Responsible for command-packet formation and response-packet handling and formation. Ensures that the order of data sent to the DL matches the command and response packet order sent to the DL. Manages data flits from the DL, and associates the data with the command or response packet that was received prior to the arrival of the data. The command and response packets contain data descriptors that enable this association. Provides flow control. Provides error handling and control. Manages virtual channels, virtual queues, and service queues associated with the virtual channels. Order is retained within virtual channels.

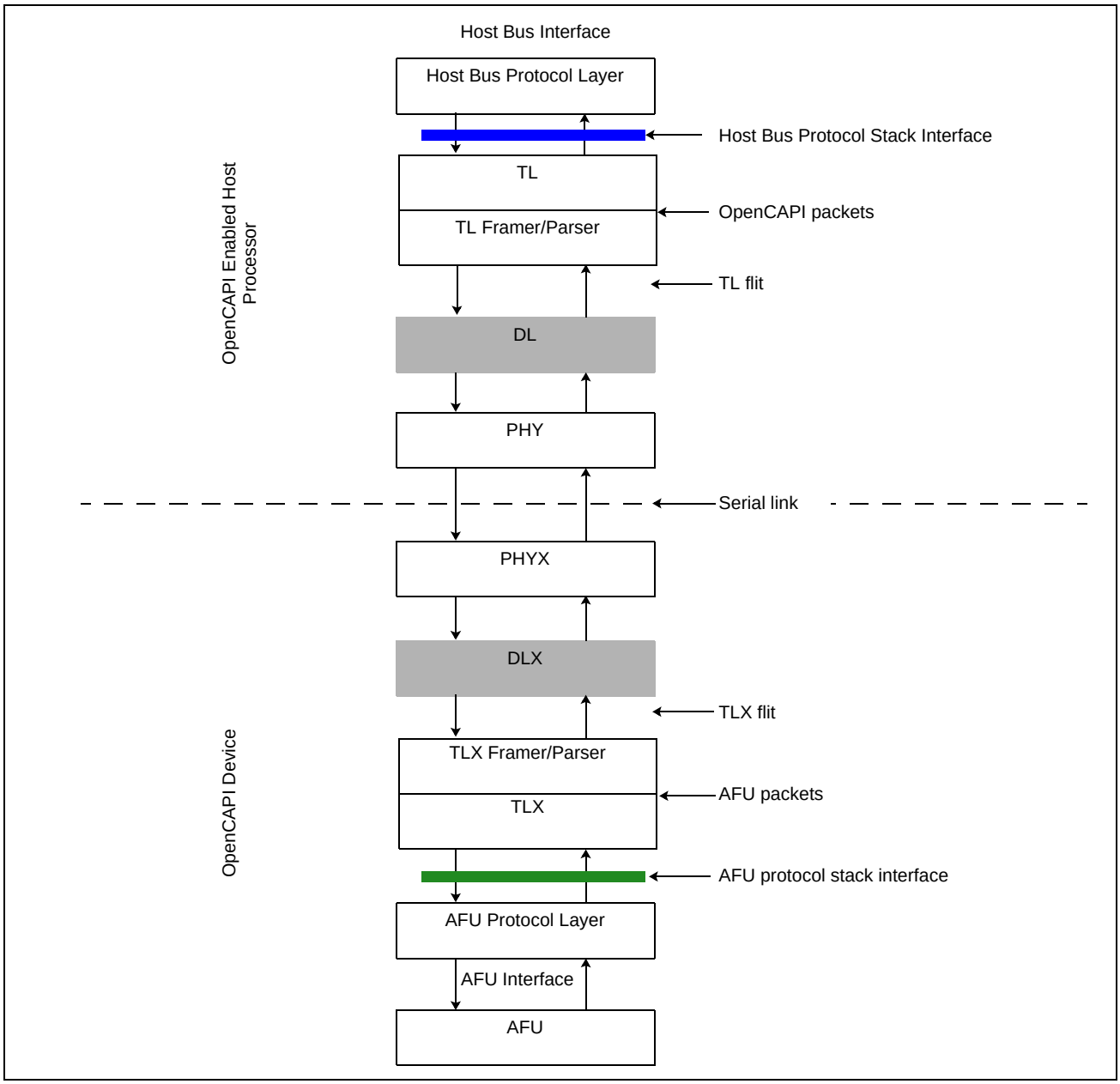
TLX	OpenCAPI transaction layer found on the OpenCAPI device. Interfaces to the DLX and the protocol layer. Responsible for command-packet formation and response-packet handling and formation. Ensures that the order of data sent to the DLX matches the command and response packet order sent to the DLX. Manages data flits from the DLX, and associates the data with the command or response packet that was received prior to the arrival of the data. The command and response packets contain data descriptors that enable this association. Provides flow control. Provides error handling and control.
TS1	Training set 1.
TS2	Training set 2.
TS3	Training set 3.
TX	Transmitter.

1. Overview

The OpenCAPI™ data link layer supports a serial data rate per lane that is consistent with the OIF CEI-28G-SR specification that connects any OpenCAPI-enabled processor, including the IBM® POWER® processor to an FPGA, ASIC, and other devices that contain an attached functional unit(s) [AFU(s)]. The base configuration uses eight lanes running at 25.78125 Gbps data rate. The supported SerDes rates are defined in *Table 2-7 Planned revisions and supported speeds* on page 26.

The data link layer that is implemented on the host is referred to as the DL. The data link layer that is implemented on the OpenCAPI device is referred to as the DLX. *Figure 1-1* shows where the DL and DLX fit in the OpenCAPI protocol layers. Any future references to DL apply to DL and the DLX.

Figure 1-1. OpenCAPI stack

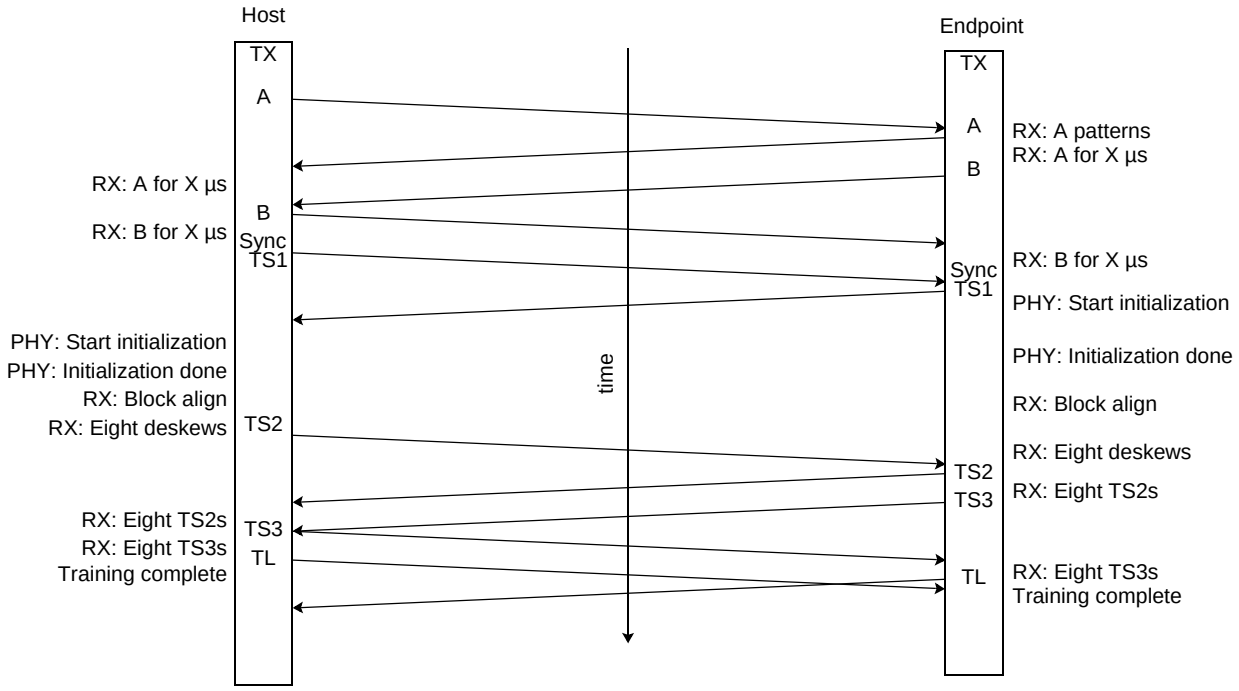


2. Link training

The endpoint shall be held in reset by an out-of-band OpenCAPI Device Enable (OCDE) signal. After the OCDE signal is enabled (active high), link training begins. The definition of the OCDE implementation is found in a host's platform architecture specification.

The link training is broken into three parts: PHY training, PHY initialization, and DL training. *Figure 2-1* illustrates the exchange of training steps. The left side is a host processor; the right side is an attached OpenCAPI device endpoint. The information that is inside the boxes is what the transmitters are sending.

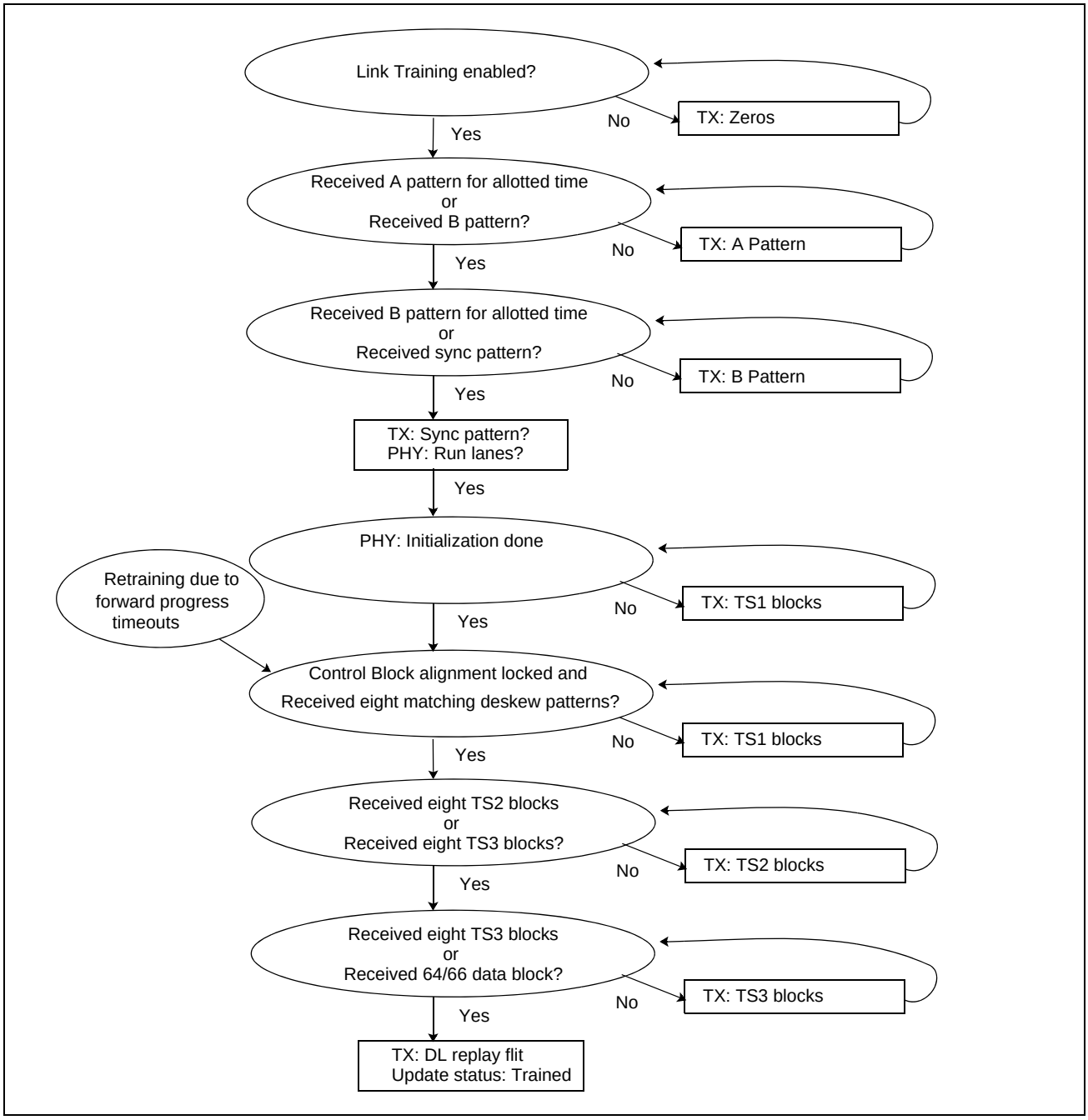
Figure 2-1. Training exchange between the host and endpoint



Note: The term "X µs" indicates a configurable number of microseconds as defined *Section 2.1 PHY training* on page 20.

Figure 2-2 on page 19 provides a training flow diagram for each side of the link.

Figure 2-2. Link training flow diagram



2.1 PHY training

When training is enabled, a slow-speed, non-scrambled signal shall be sent to reliably communicate across the link. All training shall be done on a per-lane basis. The first pattern transmitted shall be an A pattern, which is a repeating pattern of x‘FF00’. This pattern mimics a slow-speed clock, and shall be transmitted until the same pattern is received. Detecting the pattern shall be programmable to search for 4, 6, or 7 consecutive one values followed by the same number of zero values. The bits around the transition are not reliable and are ignored. For example, “1111----0000----” for the 4 consecutive case, where “-” signifies a don’t care. After receiving training pattern A for a programmable amount of time (1 µs - 16 ms), the lane switches to sending a B pattern. The B pattern is an A pattern with a pattern of x‘FFFF0000’ inserted after every 30 - 48, x‘FF00’ patterns. The receipt of an inverted B pattern adjusts the receiver polarity (P/N swap). After receiving the B pattern for the allotted time (the same time requirement as an A pattern), the transmitter of the lane shall send a sync pattern, x‘FF0000FF’, followed by the first DL training set. The A, B, and sync patterns are all non-scrambled and do not have any encoding on them. Again, the bits around the transition are not reliable and are ignored for the B pattern and the sync pattern.

Engineering note

Patterns in this section are transmitted from left to right. For example, Pattern A of x’FF00’ is sent down the physical wire as ‘1111 1111 0000 0000’.

Table 2-1 summarizes the DL PHY training patterns.

Table 2-1. PHY training patterns

Name	TX pattern	Frequency	Purpose
A pattern	x‘FF00’	Repeating	Device detection due to resetting of a link.
B pattern	x‘FFFF0000’ followed by 30 - 48 repeats of x‘FF00’	Repeating	Lane polarity (P/N swap).
Sync pattern	x‘FF0000FF’	Once	Start RX PHY initialization.

2.2 PHY initialization

When receiving the sync pattern on at least two lanes, the DL shall tell the PHY that it is about to receive random data by way of the “run lane” signal. It then waits for the “initialization done” signal from the PHY. Upon receiving the “run lane” signal, the PHY re-locks the clock data recovery (CDR), centers the eye, and obtains the high-speed bit lock. The DL ignores received data until PHY initialization is complete.

2.3 DL training sets

While the PHY is initializing and the DL is training, blocks that are associated with patterns TS1, TS2, TS3, and deskew are 64/66 encoded and scrambled (for more information, see *Section 10 Encoding and scrambling* on page 56). A block is a contiguous set of 64 bits following the sync header.

During PHY initialization, the DL sends control blocks with a TS1 pattern. After the PHY indicates that initialization completes, the RX searches for control block alignment and the receipt of eight deskew patterns with identical configuration information. When the receiver achieves block alignment and consistent deskew information (previously described eight deskew patterns with identical configuration information), the transmitter for this lane switches to sending TS2 blocks with valid receive lane information. After receiving eight TS2 blocks or eight TS3 blocks, it switches to sending TS3 blocks. After receiving eight TS3 blocks or a 64/66 data sync

header, the DL sends a minimum of nine DL-to-DL replay flits to sync up the sequence numbers, followed by TL data or DL-to-DL idle flits. *Table 2-2* describes the DL training sets. *Table 2-3* describes the DL3.0 TS2 and TS3 good lanes bit definitions, and *Table 2-4* on page 22 describes the DL 3.1/4.0 TS2 and TS3 good lanes bit definitions.

Table 2-2. DL training sets

Name	Bytes								Purpose
	7	6	5	4	3	2	1	0	
Training set 1 (TS1)	4A	4A	4A	4A	4A	4A	4A	4B	Block alignment and scrambler lock.
Training set 2 (TS2)	TS Byte 1	TS Byte 0	45	45	45	45	45	4B	Receipt of TS1 and valid lanes.
Training set 3 (TS3)	TS Byte 1	TS Byte 0	41	41	41	41	41	4B	Receipt of TS2 and valid lanes.
Deskew	Deskew Byte 2	Deskew Byte 1	Deskew Byte 0	1E	1E	1E	1E	4B	Every 32nd block during training sets. Lane-to-lane deskew and configuration. See *Section 2.4 Deskew markers* on page 23.

1. Bytes 5:0 of the TS1 are used to lock the unscrambler's LFSR.
2. Bytes 7 and 6 of the TS2 and TS3 training sets, the good lane field defined in Tables 2-3 and 2-4, indicate the configured lane width and which lanes are receiving good data. Bits (7:4) are one hot, and bits (3:0) may have multiple sets.

Engineering note

The bytes are sent in 0-1-2-3-4-5-6-7 order on a per lane basis.

Within each byte, the right most bit is sent first and the left most bit is sent last.

Engineering note

For example, Training set 1 (TS1) is sent down the physical wire as "1101_0010_0101_0010_0101_0010_0101_0010_0101_0010_0101_0010_0101_0010_0101_0010".

Table 2-3. [Category DL 3.0] TS2 and TS3 good lanes bit definitions

Bytes	Bits	Description of bytes 7 and 6
0	7:0	Reserved.
1	7:6	Reserved.
	5	Capable lane width of ×8.
	4	Reserved.
	3	All odd lanes trained (7,5,3,1).
	2	All even lanes trained (6,4,2,0).
	1	Lane 1 trained.
	0	Lane 0 trained.

[Category DL 3.1] *Table 2-4* describes the TS2 and TS3 good lanes bit descriptions.

Table 2-4. [Category DL 3.1/4.0] TS2 and TS3 good lanes bit definitions

Bytes	Bits	Description of bytes 7 and 6
0	7:0	Reserved.
1	7:6	Reserved.
	5	Capable lane width of ×8.
	4	Capable lane width of ×4 OL. Four outside lanes of a ×8 are capable in full function mode. Lanes may further degrade to ×2 and/or switch between ×2 and ×4 for power management.
	3	All inside lanes of the capable width trained: [×8] 6,4,3,1 or [×4 OL] 5,2.
	2	All outside lanes of the capable width trained: [×8] 7,5,2,0 or [×4 OL] 7,0.
	1	Reserved.
	0	Reserved.

Examples in byte 1:

- [Category DL 3.0] x‘2F’ = ×8 capable lane width, all lanes trained.
- [Category DL 3.0] x‘2A’ = ×8 capable lane width, but trained in ×4 degraded mode on odd lanes.
- [Category DL 3.1] x‘1C’ = ×4OL capable lane width, all 4 lanes trained.
- [Category DL 3.1] x‘14’ = ×4OL capable lane width, but trained to x2 degraded mode on lanes 7 and 0.
- [Category DL 3.1/4.0] x‘2C’ = ×8 capable lane width, all lanes trained.
- [Category DL 3.1/4.0] x‘28’ = ×8 capable lane width, but trained in ×4 degraded mode on inside lanes.
- [Category DL 3.1/4.0] x‘24’ = ×8 capable lane width, but trained in ×4 degraded mode on outside lanes.

Note: Bits 5 and 4 are mutually exclusive; therefore, only one of those bits may be set to a b‘1’.

2.4 Deskew markers

When the TS1, TS2, or TS3 training sets are transmitted, deskew markers are sent every 32 blocks. The last three bytes of the deskew marker shall contain the following configuration information:

- Lane number
- Version number
- Lane widths supported
- TX transmission order
- TX lane swap requested

During TS1 transmission, if the received lane numbers are reversed, the link attempts to the swap the lanes. Each lane shall receive eight identical deskew markers in a row to accept the configuration information.

[Category DL 3.0/4.0] *Table 2-5* defines the DL 3.0/4.0 deskew marker bits.

Table 2-5. [Category DL 3.0/4.0] Deskew marker bit definition

Bytes	Bits	Last three bytes of deskew definition
0	7:2	Reserved.
	1	×8 mode capable.
	0	Reserved.
1	7	Half width degraded mode capable.
	6	Reserved.
	5:0	Version number.
2	7	TX lane ordering for degraded modes: 1 Transmit lanes as FPGA. (This lane's data is transmitted first. Then, the neighbor's data is transmitted.) 0 Transmit lanes as host. (The neighbor's data is transmitted first. Then, this lane's data is transmitted.)
	6	TX lane swap requested (bus reversal). For more information, see *Section 2.9 Lane (bus) reversal* on page 32.
	5	Reserved.
	4:0	Lane number.

[Category DL 3.1] *Table 2-6* defines the DL 3.1 deskew marker bits. A device that supports both ×8 and ×4 OL shall handle the 4 lanes to be lanes 7, 5, 2, and 0.

Table 2-6. [Category DL 3.1] Deskew marker bit definition

Bytes	Bits	Last three bytes of deskew definition
0	7:4	Reserved.
	3	Reserved.
	2	Reserved.
	1	×8 mode capable.
	0	×4OL mode capable. Four outside lanes of a ×8 are capable in full function mode. Lanes may further degrade to ×2 and/or switch between ×2 and ×4 for power management.
1	7	Half-width degraded mode capable.
	6	Reserved.
	5:0	Version number.
2	7	Reserved.
	6	TX lane swap requested (bus reversal). For more information, see *Section 2.9 Lane (bus) reversal* on page 32.
	5	Power management capable.
	4:0	Lane number.

2.5 [Category DL 3.1] Alternative training step

The host shall implement a configuration mode with an enable/disable bit to send scrambled data to allow endpoint RX devices or cable re-timers to sync.

If enabled and before transmitting the first pattern A, the host shall send scrambled data for a configurable length of time between 256 µs - 4s. After the timer expires, the DL shall start the normal training sequence by sending pattern A. This alternative training step is defined in a host's platform architecture specification.

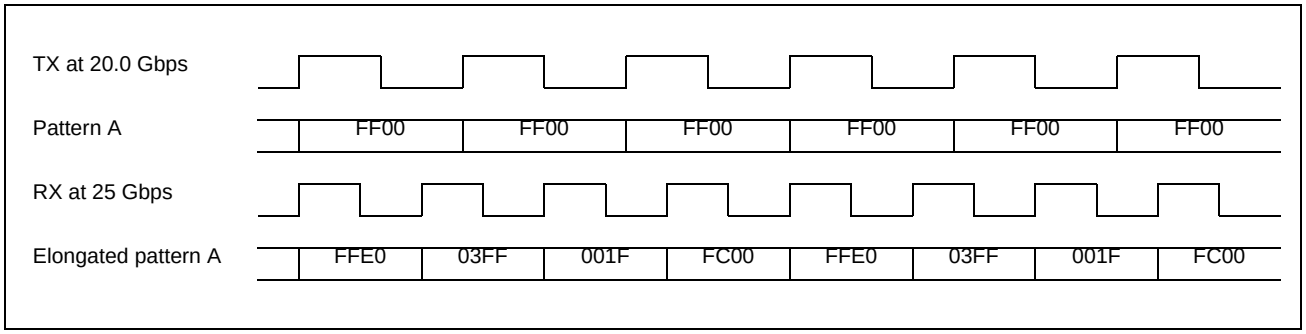
2.6 Endpoint link speed discovery

Both sides of the link initially run at their maximum link speed. The host DL initially transmits pattern A at its fastest supported speed. For more information, see *Table 2-7 Planned revisions and supported speeds* on page 26. It samples the periodicity of the received pattern A clock edges to determine if the RX is sampling a slower clock speed. If the host receiver detects an incoming periodicity that is slower than what its TX is sending, the RX sets a FIR. The exact definition of the FIR is defined by the host processor implementation specification. Software shall perform the following functions:

1. De-activate the OCDE to the endpoint (active high).
2. Reconfigure the host phase-locked loop (PLL) for the next supported slower speed. For more information, see *Table 2-7* on page 26.
3. Wait for the host's PLL to re-lock.
4. Activate the OCDE to the endpoint (active high).
5. Start the initialization procedure again. The number of iterations that is supported before an unrecoverable event is detected is defined by the host architecture specification.

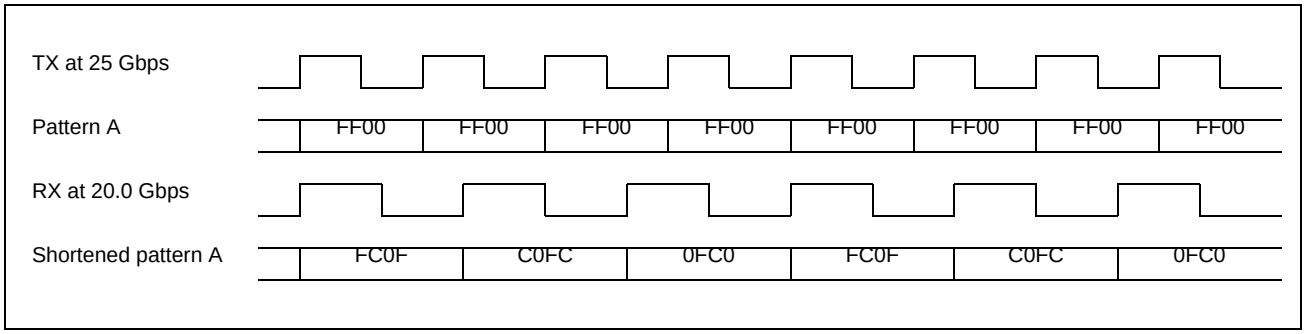
Figure 2-3 and *Figure 2-4* show how a faster or slower pattern A is decoded. (The speeds in the following figures are representative of mismatched frequencies as delineated in *Table 2-7* on page 26.)

Figure 2-3. Faster pattern A



Detection of an elongated pattern A by the host indicates that the endpoint is running slower, and the host PLL shall be reconfigured to the next supported slower speed.

Figure 2-4. Slower pattern A



Detection of a shortened pattern A by the host indicates that the endpoint is running faster, but no host action is required to be taken because the endpoint must be reconfigured to the next supported slower speed. For example, an endpoint FPGA must be reloaded with a new bit image that was created with the next supported slower speed and a new IPL must be re-initiated. Or, if an ASIC, the endpoint's PLL must be re-configured to the next supported slower speed during the IPL sequence. The ASIC IPL sequence is defined in the host's platform architecture specification.

Table 2-7 on page 26 shows the planned revisions and supported speeds. Endpoints are not required to support all speeds for the versions that it implemented. Hosts are required to support all speeds for the version that it implemented.

Table 2-7. Planned revisions and supported speeds

Revision	Speeds (Gbps)	Reference_Clock	Capable Link widths
DL 3.0	25.78125[1], 20.0[4]	156.25 (OIF Standard)	×8
DL 3.1	25.6[1], 21.33[2]	133.33 (JEDEC Standard)	×4OL, ×8
DL 4.0	31.875[1], 25.78125[3], 20.0[4] 32[1]	156.25 (OIF Standard) 133.33 (JEDEC Standard)	×8 x8

1. Plan of record.
2. Pre-IPL knowledge of link speeds, so no requirement to detect link speeds.
3. Backwards speed compatibility.
4. Lab testing mode only.

When both sides of the link detect that they are running the same bit rate, the PHY initialization starts counting the number of A patterns that it received.

2.7 Degraded modes

During the TS2 and TS3 training patterns, the trained lanes are transmitted back to the sender. If all of the lanes are trained, the link transmits across all lanes. If any lane of a group does not receive good training sets, that lane is marked as invalid and the link transmits on only the valid trained lanes (degraded mode). For more information, see *Table 2-3 [Category DL 3.0] TS2 and TS3 good lanes bit definitions* on page 21 or *Table 2-4 [Category DL 3.1/4.0] TS2 and TS3 good lanes bit definitions* on page 22.

2.8 Lane width detection and degraded mode flit transmission order

During the deskew training pattern, each side of the link transmits all of the lane widths that it supports. If more than one lane width is common between the two ends, the widest width is used. The lane-width information is used to configure the TL-to-DL ratio and the corresponding bytes-to-lane multiplexes.

Table 2-8 shows which table to use to determine the bytes-to-lane multiplexing.

Table 2-8. Transmit order bytes-to-lane multiplexing look up table (Page 1 of 2)

OpenCAPI Data Link Layer Version(s)	Width	Mode	*Table 2-9*	*Table 2-10*	*Table 2-11*	*Table 2-12*	*Table 2-13*	*Table 2-14*	*Table 2-15*	*Table 2-16*	*Table 2-17*	*Table 2-18*	*Table 2-19*	*Table 2-20*
0,1,2	×8	Full-width	X											
0	×8	Half-width degraded even lanes		X										
0	×8	Half-width degraded odd lanes			X									
3,4,5,6,8,9,10	×8	Full-width				X								
3,4,5,6,8,9,10	×8	Half-width degraded outside lanes					X							
3,4,5,6,8,9,10	×8	Half-width degraded inside lanes						X						

Note: See *Section 8* on page 52 for details on the OpenCAPI data link layer versions.

Table 2-8. Transmit order bytes-to-lane multiplexing look up table (Page 2 of 2)

OpenCAPI Data Link Layer Version(s)	Width	Mode	*Table 2-9*	*Table 2-10*	*Table 2-11*	*Table 2-12*	*Table 2-13*	*Table 2-14*	*Table 2-15*	*Table 2-16*	*Table 2-17*	*Table 2-18*	*Table 2-19*	*Table 2-20*
3,4,5,6,8,9,10	×8	Half-width power management					X							
8,9,10	×8	Quarter-width power management							X					
3,4,5,6,8,9,10	×4OL	Full-width					X							
8,9,10	×4OL	Half-width degraded inside lanes								X				
8,9,10	×4OL	Half-width degraded outside lanes							X					
8,9,10	×4OL	Half-width power management							X					
2	×8	Half-width degraded outside lanes store and forward lowest byte first									X			
2	×8	Half-width degraded inside lanes store and forward lowest byte first										X		
1	x8	Half-width degraded inside lanes											X	
1	x8	Half-width degraded outside lanes												X

Note: See *Section 8* on page 52 for details on the OpenCAPI data link layer versions.

Engineering note

In the following tables, flit bytes are sent in right-to-left order. For example, 63:62 indicates byte 62 is sent first followed by byte 63 next. Within each byte, the right most bit is sent first and the left most bit is sent last.

Table 2-9 shows the flit bytes to lane multiplexing on full-width mode (eight lanes) for OpenCAPI data link layer versions 0, 1, and 2.

Table 2-9. Flit bytes to lane multiplexing on full-width mode (×8) for versions 0, 1, 2

Cycle	Lane 7	Lane 6	Lane 5	Lane 4	Lane 3	Lane 2	Lane 1	Lane 0
3	63:62	55:54	47:46	39:38	31:30	23:22	15:14	7:6
2	61:60	53:52	45:44	37:36	29:28	21:20	13:12	5:4
1	59:58	51:50	43:32	35:34	27:26	19:18	11:10	3:2
0	57:56	49:48	41:40	33:32	25:24	17:16	9:8	1:0

Table 2-10 shows the flit bytes to lane multiplexing on half-width degraded even lanes for OpenCAPI data link layer version 0.

Table 2-10. Flit bytes to lane multiplexing on half-width degraded even lanes for version 0

Cycle	Lane 7	Lane 6	Lane 5	Lane 4	Lane 3	Lane 2	Lane 1	Lane 0
7		55:54		39:38		23:22		7:6
6		53:52		37:36		21:20		5:4
5		51:50		35:34		19:18		3:2
4		49:48		33:32		17:16		1:0
3		63:62		47:46		31:30		15:14
2		61:60		45:44		29:28		13:12
1		59:58		43:32		27:26		11:10
0		57:56		41:40		25:24		9:8

Table 2-11 shows the flit bytes to lane multiplexing on half-width degraded odd lanes for OpenCAPI data link layer version 0.

Table 2-11. Flit bytes to lane multiplexing on half-width degraded odd lanes for version 0

Cycle	Lane 7	Lane 6	Lane 5	Lane 4	Lane 3	Lane 2	Lane 1	Lane 0
7	63:62		47:46		31:30		15:14	
6	61:60		45:44		29:28		13:12	
5	59:58		43:32		27:26		11:10	
4	57:56		41:40		25:24		9:8	
3	55:54		39:38		23:22		7:6	
2	53:52		37:36		21:20		5:4	
1	51:50		35:34		19:18		3:2	
0	49:48		33:32		17:16		1:0	

Table 2-12 shows the flit bytes to lane multiplexing on full-width mode (eight lanes) for OpenCAPI data link layer versions 3, 4, 5, 6, 8, 9, and 10.

Table 2-12. Flit bytes to lane multiplexing on full-width mode (×8) for versions 3, 4, 5, 6, 8, 9, and 10

Cycle	Lane 7	Lane 6	Lane 5	Lane 4	Lane 3	Lane 2	Lane 1	Lane 0
3	63:62	61:60	59:58	57:56	55:54	53:52	51:50	49:48
2	47:46	45:44	43:42	41:40	39:38	37:36	35:34	33:32
1	31:30	29:28	27:26	25:24	23:22	21:20	19:18	17:16
0	15:14	13:12	11:10	9:8	7:6	5:4	3:2	1:0

Table 2-13 shows the flit bytes to lane multiplexing on half-width degraded outside lanes (×8 mode), half-width degraded lanes for power management (×8 mode), or full-width mode (×4OL) for OpenCAPI data link layer versions 3, 4, 5, 6, 8, 9, and 10.

Table 2-13. Flit bytes to lane multiplexing on half-width degraded outside lanes (×8), half-width degraded lanes for power management (×8), or full-width mode (×4OL) for versions 3, 4, 5, 6, 8, 9, and 10

Cycle	Lane 7	Lane 6	Lane 5	Lane 4	Lane 3	Lane 2	Lane 1	Lane 0
7	63:62		59:58			55:54		51:50
6	61:60		57:56			53:52		49:48
5	47:46		43:42			39:38		35:34
4	45:44		41:40			37:36		33:32
3	31:30		27:26			23:22		19:18
2	29:28		25:24			21:20		17:16
1	15:14		11:10			7:6		3:2
0	13:12		9:8			5:4		1:0

Table 2-14 shows the flit bytes to lane multiplexing on half-width degraded inside lanes for OpenCAPI data link layer versions 3, 4, 5, 6, 8, 9, and 10.

Table 2-14. Flit bytes to lane multiplexing on half-width degraded inside lanes (×8) for versions 3, 4, 5, 6, 8, 9, and 10

Cycle	Lane 7	Lane 6	Lane 5	Lane 4	Lane 3	Lane 2	Lane 1	Lane 0
7		63:62		59:58	55:54		51:50	
6		61:60		57:56	53:52		49:48	
5		47:46		43:42	39:38		35:34	
4		45:44		41:40	37:36		33:32	
3		31:30		27:26	23:22		19:18	
2		29:28		25:24	21:20		17:16	
1		15:14		11:10	7:6		3:2	
0		13:12		9:8	5:4		1:0	

Table 2-15 shows the flit bytes to lane multiplexing on quarter-width for power management (×2), half-width degraded outside lanes (×4OL), or half-width for power management (×4OL) for OpenCAPI data link layer versions 8, 9, and 10.

Table 2-15. Flit bytes to lane multiplexing on quarter-width for power management (×2), half-width degraded outside lanes (×4OL), or half-width for power management (×4OL) for versions 8, 9, and 10

Cycle	Lane 7	Lane 6	Lane 5	Lane 4	Lane 3	Lane 2	Lane 1	Lane 0
15	63:62							55:54
14	61:60							53:52
13	59:58							51:50
12	57:56							49:48
11	47:46							39:38
10	45:44							37:36
9	43:42							35:34
8	41:40							33:32
7	31:30							23:22
6	29:28							21:20
5	27:26							19:18
4	25:24							17:16
3	15:14							7:6
2	13:12							5:4
1	11:10							3:2
0	9:8							1:0

Table 2-16 shows the flit bytes to lane multiplexing on half-width degraded inside lanes (×4OL) for OpenCAPI data link layer versions 8, 9, and 10.

Table 2-16. Flit bytes to lane multiplexing on half-width degraded inside lanes (×4OL) for versions 8, 9, and 10
(Page 1 of 2)

Cycle	Lane 7	Lane 6	Lane 5	Lane 4	Lane 3	Lane 2	Lane 1	Lane 0
15			63:62			55:54		
14			61:60			53:52		
13			59:58			51:50		
12			57:56			49:48		
11			47:46			39:38		
10			45:44			37:36		
9			43:42			35:34		
8			41:40			33:32		
7			31:30			23:22		
6			29:28			21:20		
5			27:26			19:18		
4			25:24			17:16		
3			15:14			7:6		

Table 2-16. Flit bytes to lane multiplexing on half-width degraded inside lanes (×4OL) for versions 8, 9, and 10
(Page 2 of 2)

Cycle	Lane 7	Lane 6	Lane 5	Lane 4	Lane 3	Lane 2	Lane 1	Lane 0
2			13:12			5:4		
1			11:10			3:2		
0			9:8			1:0		

Table 2-17 Shows the flit bytes to lane multiplexing on half-width degraded outside lanes, store and forward, lowest byte first for OpenCAPI data link layer version 2.

Table 2-17. Flit bytes to lane multiplexing on half-width degraded outside lanes, store and forward, lowest byte first for version 2

Cycle	Lane 7	Lane 6	Lane 5	Lane 4	Lane 3	Lane 2	Lane 1	Lane 0
7	63:62		47:46			31:30		15:14
6	61:60		45:44			29:28		13:12
5	59:58		43:32			27:26		11:10
4	57:56		41:40			25:24		9:8
3	55:54		39:38			23:22		7:6
2	53:52		37:36			21:20		5:4
1	51:50		35:34			19:18		3:2
0	49:48		33:32			17:16		1:0

Table 2-18 shows the flit bytes to lane multiplexing on half-width degraded inside lanes, store and forward, lowest byte first for OpenCAPI data link layer version 2.

Table 2-18. Flit bytes to lane multiplexing on half-width degraded inside lanes, store and forward, lowest byte first for version 2

Cycle	Lane 7	Lane 6	Lane 5	Lane 4	Lane 3	Lane 2	Lane 1	Lane 0
7		63:62		47:46	31:30		15:14	
6		61:60		45:44	29:28		13:12	
5		59:58		43:32	27:26		11:10	
4		57:56		41:40	25:24		9:8	
3		55:54		39:38	23:22		7:6	
2		53:52		37:36	21:20		5:4	
1		51:50		35:34	19:18		3:2	
0		49:48		33:32	17:16		1:0	

Table 2-19 shows the flit bytes to lane multiplexing on half-width degraded inside lanes for OpenCAPI data link layer version 1.

Table 2-19. Flit bytes to lane multiplexing on half-width degraded inside lanes for version 1

Cycle	Lane 7	Lane 6	Lane 5	Lane 4	Lane 3	Lane 2	Lane 1	Lane 0
7		55:54		39:38	23:22		7:6	
6		53:52		37:36	21:20		5:4	
5		51:50		35:34	19:18		3:2	
4		49:48		33:32	17:16		1:0	
3		63:62		47:46	31:30		15:14	
2		61:60		45:44	29:28		13:12	
1		59:58		43:32	27:26		11:10	
0		57:56		41:40	25:24		9:8	

Table 2-20 shows the flit bytes to lane multiplexing on half-width degraded outside lanes for OpenCAPI data link layer version 1.

Table 2-20. Flit bytes to lane multiplexing on half-width degraded outside lanes for version 1

Cycle	Lane 7	Lane 6	Lane 5	Lane 4	Lane 3	Lane 2	Lane 1	Lane 0
7	63:62		47:46			31:30		15:14
6	61:60		45:44			29:28		13:12
5	59:58		43:32			27:26		11:10
4	57:56		41:40			25:24		9:8
3	55:54		39:38			23:22		7:6
2	53:52		37:36			21:20		5:4
1	51:50		35:34			19:18		3:2
0	49:48		33:32			17:16		1:0

2.9 Lane (bus) reversal

Lane reversal is independently determined on both of the RX and TX domains and shall be compatible with all supported lane widths. During training, the host DL detects reversed lane connections and attempts to swap the lanes. If incorrect lane connections are still present, the link does not train. OpenCAPI devices (DLX) are not required to support the multiplexers for lane reversal and shall indicate lane swap requests to the host in the deskew training sets. Then, the host reverses the lanes before transmitting the data.

The lane swap is handled as shown in *Table 2-21*.

Table 2-21. Lane reversal based on mode (Page 1 of 2)

×8 or ×4 Degraded	
Lane	Reversal
0	7
1	6
2	5

Table 2-21. Lane reversal based on mode (Page 2 of 2)

×8 or ×4 Degraded	
Lane	Reversal
3	4
4	3
5	2
6	1
7	0

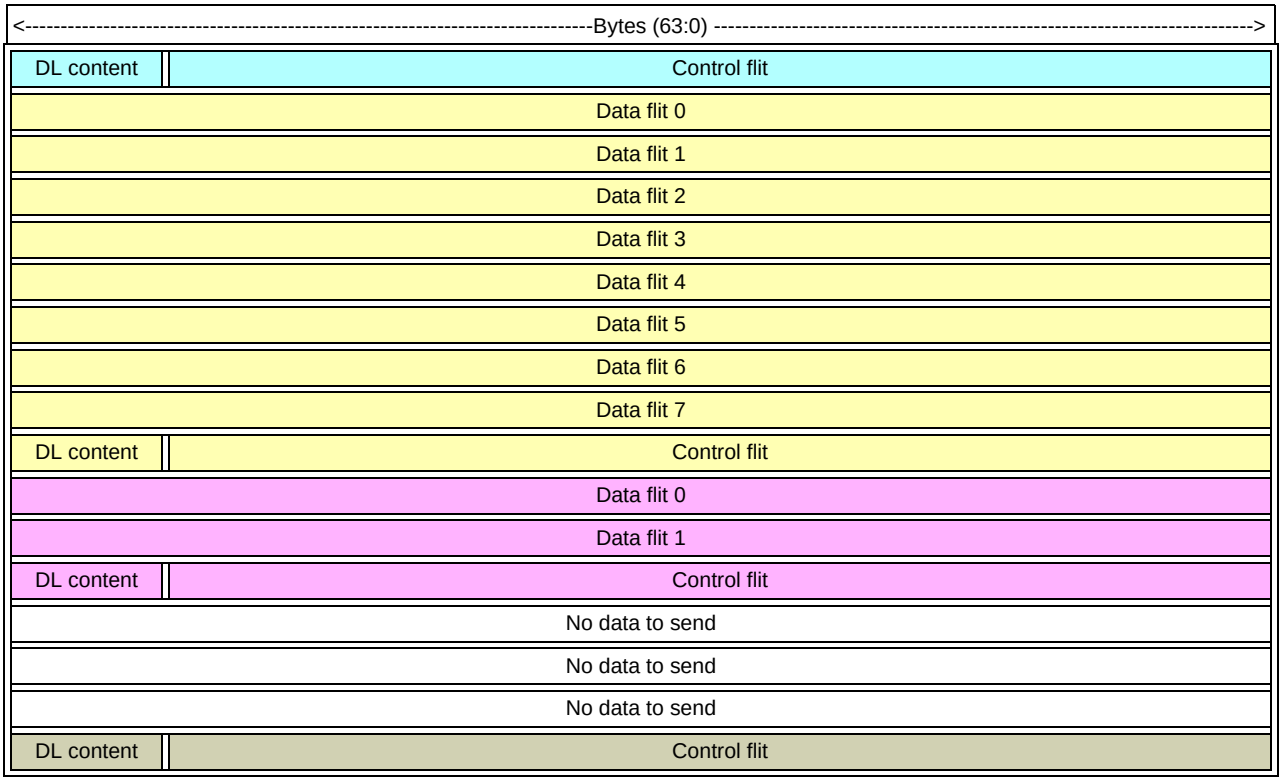
2.10 DL transmit configurations for link debug

Engineering note

The DL transmit may be configured to send different patterns for debug and calibration. In addition to being enabled for training, each lane may continuously send all zeros (disabled mode), A patterns, B patterns, DL TS1 patterns, control blocks with all zeros data, or a pseudo-random bit stream.

3. TL flits

Two types of 64-byte flits are sent from the TL: control flits and data flits. Control flits are defined by the *Open CAPI 3.0 Transaction Layer Specification*; 64 bits are reserved for DL content. Data flits contain 64 bytes of data. There shall be 0 - 8 data flits after each control flit. The last data flit shall be followed by a control flit for data validation. There shall be no gaps or delays in the stream from the TL to the DL, unless the DL informs the TL that it temporarily cannot accept more. For more information, see *Section 4.6 Data Stalled (19)* on page 37.



DL content	Control flit
Data flit 0	
Data flit 1	
Data flit 2	
Data flit 3	
Data flit 4	
Data flit 5	
Data flit 6	
Data flit 7	
DL content	Control flit
Data flit 0	
Data flit 1	
DL content	Control flit
No data to send	
No data to send	
No data to send	
DL content	Control flit

4. DL content fields

[Category DL 3.0/4.0] *Table 4-1* shows the DL content fields.

Table 4-1. [Category DL3 .0/4.0] DL content

Bits															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
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
CRC (35:20)															
CRC (19:4)															
CRC(3:0)				ACK count(4:0)					Reserved			Data stalled	Short flit next	TL template (5:4)	
TL template(3:0)				Bad data flit(7:0)								Data run length(3:0)			

[Category DL 3.1] *Table 4-2* shows the DL content fields.

Table 4-2. [Category DL 3.1] DL content

Bits															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
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
CRC (35:20)															
CRC (19:4)															
CRC(3:0)				ACK count(4:0)					Recal Info(1:0)		Rsv	Data stalled	Short flit next	TL template (5:4)	
TL template(3:0)				Bad data flit(7:0)								Data run length(3:0)			

Section 4.1 -Section 4.10 on page 38 describe each field.

4.1 CRC (63:28)

The CRC of a TL control flit shall cover that flit and all TL data flits since the last TL control flit. The CRC of a DL-to-DL flit shall cover only that flit. When a CRC error is detected, the RX shall send the CRC error signal to the TL. This indicates that all data flits since the last control flit are bad. The DL RX logic also requests the DL TX to send a replay flit stream.

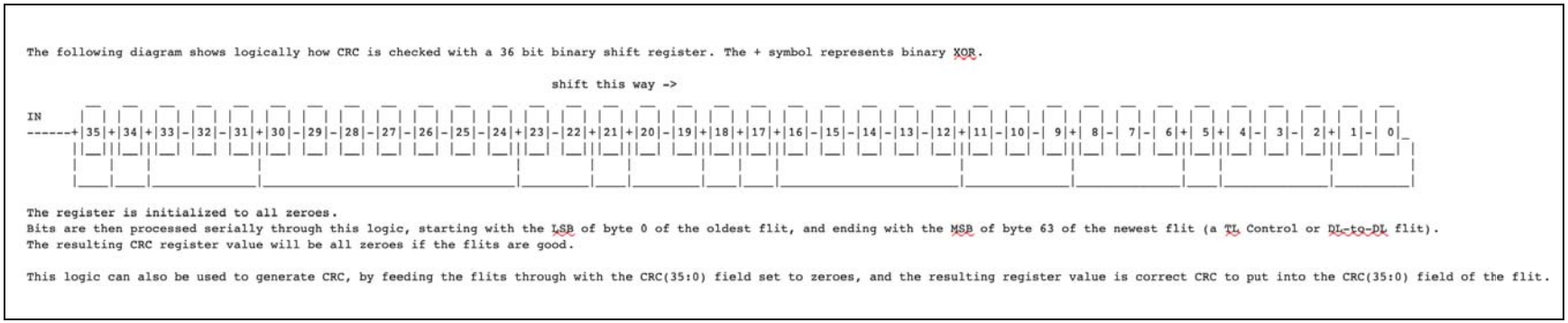
The 36-bit CRC uses the following polynomial, which ensures that all errors with an odd number of bits, up to 5 bits, or bursts of up to 36 bits, are detected. The mis-detect rate when the event is not covered is 1.46e-11:

$$x^{36} + x^{34} + x^{31} + x^{30} + x^{27} + x^{24} + x^{19} + x^{18} + x^{17} + x^{15} + x^{14} + x^{12} + x^{5} + x^{2} + x + 1$$

Engineering note

A 36-bit CRC polynomial is shown in *Figure 4-1*.

Figure 4-1. Logical checking of CRC with a 36-bit binary shift register



4.2 ACK count (27:23)

The ACK count bits indicate the number of new 64-byte TL flits (control or data) that were received successfully on the RX side since the last time a flit containing an ACK count field was transmitted. When the RX side detects a CRC error, it stops the TX from sending ACKs until the replayed data is received successfully.

Each TX and RX keeps a sequence count of the number of TL flits that are known to have been transferred successfully to the TL layer. Upon receiving a TL control flit with good CRC, the RX increments its sequence count (by the number of data flits preceding plus one) and passes this increment to the TX to be returned to the remote DL by way of the ACK count. The remote DL TX then uses it to increment its own TX sequence count. When a CRC error is detected on the link, the RX indicates to the TX to start a replay sequence of flits with a NACK to indicate to the remote side that a replay is required. All replay flits provide the sequence counts of the next TL flit to be transmitted (derived from the TX sequence count) and the sequence count of the last flit received successfully. Because the sequence number always points to the first flit after a control flit, the DL cannot send an ACK count that does not include a full frame. This ensures that the TX sequence number always points to the first flit after a control flit.

Implementation note: Flits that are received after a replay flit that were accepted with a good CRC the first time they were transmitted do not return ACKs.

4.3 [Category DL 3.1] Recal Info (22:21)

Recalibration information that is passed from DL-to-DL is used to recalibrate the RX lanes. *Table 4-3* defines the values that are used in this field.

Engineering note

Lane recalibration is driven by a configurable timer through a host DL's configuration register. The endpoint DLx responds to the recal messages in TL Control, DL idle, and replay flits.

Table 4-3. Recalibration Information

'00'	Recalibration state machine idle
'01'	Reset recalibration lane number to lane 0
'10'	Recalibrate next lane
'11'	Recalibrate next lane

4.4 [Category DL 3.1] Reserved (20)

This is a reserved field.

4.5 [Category DL 3.0/4.0] Reserved (22:20)

This is a reserved field.

4.6 Data Stalled (19)

When this bit is set, the next flit is a DL-to-DL flit, even though the data run length might indicate that there are data flits to follow before the next TL control flit.

Engineering note

Data stalled may be required if:

- The replay buffer is in a full condition
- It is necessary to send a power management message
- It is required to start a replay flit sequence (with NACK)

4.7 Short Flit Next (18)

When this bit is set, the next flit is a short idle flit, even though the data run length may have indicated that data flits were to follow. Therefore, the CRC of the next flit shall be calculated based on the short idle flit length. The normal flit length is 64 bytes. The length of the short idle flit is based on the version number of the endpoint. For more information, see *Table 8-1 Data link layer version numbers* on page 52.

4.7.1 [Category DL 3.1] Selectable Idle Flit Size

To reduce the latency at the end of an idle period, the DL 3.1 offers a 16-byte short idle flit size. This allows the DL to switch over to TL flits on the next cycle. The version number of both ends of the link shall be x‘8’- x‘F’ to support this mode.

Assuming that the DL 3.1 is configured to use the 16-byte idle flits, when the TL indicates that it has a valid flit, the DL 3.1 shall indicate that the next flit is a 64-byte control flit and allow the TL flit to be sent without delays.

When short idle flits are enabled, before transmitting any sequence of replay flits (excluding the initial replay sequence after training or retraining), the TX shall transmit three short idle flits with the short flit next bit asserted, and then one short idle flit with the short flit next bit de-asserted. The next flit to be sent by the TX shall be the first of at least nine replay flits. This ensures that the receiving side can find the starting beat of the replay sequence.

When short idle flits are enabled, the receiver must find the 4-beat replay alignment and shall look at the short flit next bit (bit 82) of each 16-byte beat. If bit 82 is asserted, the next beat is not the start of a replay flit. If bit 82 is de-asserted, the receiver speculates that this is a start of a replay flit and calculates CRC across the next four beats. If the CRC is clean and if the run length is a replay flit (run length of x’A’), the receiver accepts it as a replay and then starts counting the minimum count of nine replay flits. If less than nine consecutive replay flits with good CRC are received or if a CRC error is detected, the sequence of looking for bit 82 is started again with this current beat.

4.8 TL Template (17:12)

The TL templates are defined in the *OpenCAPI 3.0 Transaction Layer Specification*.

4.9 Bad Data Flit (11:4)

The bad data flits are defined in the *OpenCAPI 3.0 Transaction Layer Specification*.

4.10 Data Run Length (3:0)

The data run length indicates the number of TL data flits until the next control flit. Values x’0’ - x’8’ indicate the number of data flits that is associated with this control flit. The DL snoops this field to know when to expect the next control flit to insert the DL content. Run lengths of x‘9’ - x‘F’ from the transaction layer are invalid and shall be flagged as an error.

5. DL-to-DL flits

The DL shall communicate with the remote DL by sending special encodes in the run length field.

5.1 DL idle flit

When there are no TL control or data flits to send, the DL inserts an idle flit. These idle flits have a run length of x‘F’. When they are received, they shall be discarded. No data flits follow an idle flit unless the Short Flit Next (flit bit 18) is zero and stalled data run length (flit bits 7:4) is in the range 1 - 8, which indicates that data stalled completed. *Table 5-1* and *Table 5-2* show the last 8 bytes of the idle flit. All other bytes of the idle flit are reserved.

Table 5-1. [Category DL 3.0/4.0] Idle flit (last 8 bytes)

Bits															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
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
CRC(35:20)															
CRC(19:4)															
CRC(3:0)				ACK count(4:0)					Reserved			Data stalled	Short flit next	Reserved	
Reserved								Stalled data run length				Data run length(3:0) = x‘F’			

Table 5-2. [Category DL 3.1] Idle flit (last 8 bytes)

Bits															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
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
CRC(35:20)															
CRC(19:4)															
CRC(3:0)				ACK count(4:0)					Recal Info(1:0)		Rsv	Data stalled	Short flit next	Reserved	
Reserved				Power management message(3:0)				Stalled data run length				Data run length(3:0) = x‘F’			

Section 5.1.1 - Section 5.1.12 on page 41 describe each field.

5.1.1 CRC (63:28)

For more information, see *Section 4.1 CRC (63:28)* on page 35.

5.1.2 ACK Count (27:23)

For more information, see *Section 4.2 ACK count (27:23)* on page 36.

5.1.3 [Category DL 3.1] Recal Info (22:21)

For more information, see *Section 4.3 [Category DL 3.1] Recal Info (22:21)* on page 37.

5.1.4 [Category DL 3.1] Reserved (20)

This is a reserved field.

5.1.5 [Category DL 3./4.0] Reserved (22:20)

This is a reserved field.

5.1.6 Data Stalled (19)

When this bit is set and the Stalled Data Run Length is zero, the next flit is a DL-to-DL flit.

5.1.7 Short Flit Next (18)

For more information, see *Section 4.7 Short Flit Next (18)* on page 38.

5.1.8 [Category DL 3.0/4.0] Reserved (17:8)

This is a reserved field.

5.1.9 [Category DL 3.1] Reserved (17:12)

This is a reserved field.

5.1.10 [Category DL 3.1] Power Management Message (11:8)

When power management is enabled, this field indicates the current lane width and relays any special instructions to the DL on the other side of the link.

For more information about power management messages and how they are used, see *Section 7 [Category DL 3.1] Dynamic lane power down* on page 48

Table 5-3. [Category DL 3.1] Power management message (Page 1 of 2)

Field	Description
‘0000’	Power management is disabled.
‘0001’	Running in quarter-width mode.
‘0010’	Running in half-width mode.
‘0011’	Running in full-width mode.
‘0100’	Wake up lanes to run in half-width mode from quarter-width mode.
‘0101’	Wake up lanes to run in full-width mode from half-width mode.
‘0110’	Lanes are ready to run in half-width mode from quarter-width mode.

Table 5-3. [Category DL 3.1] Power management message (Page 2 of 2)

Field	Description
‘0111’	Lanes are ready to run in full-width mode from half-width mode.
‘1000’	Switch to half-width mode from full-width mode.
‘1001’	Switch to quarter-width mode from half-width mode.
‘1010’ - ‘1111’	Reserved.

5.1.11 Stalled Data Run Length (7:4)

When the previous flit indicated that data stalled (and Short Flit Next is not set), this is the number of data flits until the next control flit. A nonzero value indicates that the data flits from the control flit before the stall condition are coming next. Values of x’1’ - x’8’ indicate the number of data flits. Values x’9’x - x’F’ are invalid and bring down the link.

5.1.12 Data Run Length (3:0)

This field is set to x‘F’ to indicate an idle flit.

5.2 DL Replay Flit

When a replay is needed, the DL inserts a DL replay flit with a run length of x‘A’. Replay flits are repeated for a minimum of nine consecutive flits to ensure that a data flit does not falsely appear to be a replay flit and start an endless loop of replays.

The last 20 bytes of the replay flit are defined in *Table 5-4* on page 42. When short idles are enabled, four idle flits shall be transmitted before the first replay flit. The first three flits indicate that a short flit is next and the fourth idle flit indicates that a normal length flit shall be the next flit to be transmitted. The receiving logic shall use the short flit next indicator to determine where to start checking CRC for the replay flits.

Table 5-4. [Category DL 3.0/4.0] Replay flit (last 20 bytes)

Bits															
159	158	157	156	155	154	153	152	151	150	149	148	147	146	145	144
143	142	141	140	139	138	137	136	135	134	133	132	131	130	129	128
127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
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
CRC(35:20)															
CRC(19:4)															
CRC(3:0)				ACK count(4:0)				Reserved			NACK	Data stalled	Short flit next	Reserved	
Link errors(7:0)								Previous command run length(3:0)				Data run length(3:0) = x‘A’			
Starting sequence number(15:0)															
Acknowledge sequence number(15:0)															
DL link information(63:48)															
DL link information(47:32)															
DL link information(31:16)															
DL link information(15:0)															

Table 5-5. [Category DL 3.1] Replay flit (last 20 bytes)

Bits															
159	158	157	156	155	154	153	152	151	150	149	148	147	146	145	144
143	142	141	140	139	138	137	136	135	134	133	132	131	130	129	128
127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112
111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
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
CRC(35:20)															
CRC(19:4)															
CRC(3:0)				ACK count(4:0)				Recal info(1:0)		NACK	Data stalled	Short flit next	Reserved		
Link errors(7:0)								Previous command run length(3:0)			Data run length(3:0) = x‘A’				
Power management message(3:0)				Starting sequence number(11:0)											
Reserved				Acknowledge sequence number(11:0)											
DL link information(63:48)															
DL link information(47:32)															
DL link information(31:16)															
DL link information(15:0)															

The endpoint TL shall pass link information to the endpoint DL to transmit to the host. The DL concatenates the TL bits and information with its own and sends them in a replay flit. The link bits (7:0), and the associated link information (63:0), are stored in registers to aid in debug.

Section 5.2.1 -Section 5.2.18 on page 45 describe each field.

5.2.1 CRC (159:124)

For more information, see *Section 4.1 CRC (63:28)* on page 35.

5.2.2 ACK Count (123:119)

For more information, see *Section 4.2 ACK count (27:23)* on page 36.

Engineering note

This field is not used in a DL replay flit, because the Acknowledge Sequence Number field is used instead to notify what flits have been received successfully.

5.2.3 [Category DL 3.0/4.0] Reserved (118:117)

This is a reserved field.

5.2.4 [Category DL 3.1] Recal Info (118:117)

For more information, see *Section 4.3 [Category DL 3.1] Recal Info (22:21)* on page 37.

5.2.5 NACK (116)

NACK indicates that the Rx side of the link detected a CRC error, and it is being reported to the other end of the link.

5.2.6 Data Stalled (115)

When this bit is set, and Previous Command Run Length is zero, the next flit is a DL-to-DL flit.

5.2.7 Short Flit Next (114)

For more information, see *Section 4.7 Short Flit Next (18)* on page 38.

5.2.8 Reserved (113:112)

This is a reserved field.

5.2.9 Link Errors (111:104)

Link errors are detected by the remote side of the link. Bits 111:108 are DL link errors, bits 107:104 are TL link errors.

5.2.10 Previous Command Run Length (103:100)

When coming out of a replay condition (and Short Flit Next is not set), this is the number of data flits immediately following this replay flit. A value of zero indicates a control flit or a DL-to-DL flit is next. Values x‘9’ - x‘F’ are invalid and bring down the link.

5.2.11 Data Run Length (99:96)

This field is set to x‘A’ to indicate a replay flit.

5.2.12 [Category DL 3.0/4.0] Starting Sequence Number (95:80)

This field gives the sequence number of the next TL (control or data) flit that are transmitted across the link. The receiving logic uses this information to ignore duplicate flits. The replay flits cannot contain the same starting sequence number value. The starting sequence number is the penultimate. Also, the last replay flit must be the same to allow the receiver time to decode the flit and enable or disable processing of the immediately following TL flit.

5.2.13 [Category DL 3.0/4.0] Acknowledge Sequence Number (79:64)

This field defines the last sequence number that was received successfully by this side of the link. It may change between consecutive replay flits. The receiving logic passes this value to the transmit logic to indicate where it should start the data replay.

5.2.14 [Category DL 3.1] Power Management Message (95:92)

For more information, see *Section 5.1.11 Stalled Data Run Length (7:4)* on page 41.

5.2.15 [Category DL 3.1] Starting Sequence Number (91:80)

This field gives the sequence number of the next TL (control or data) flit which will be transmitted across the link. The receiving logic uses this information to ignore duplicate flits. The replay flits may not contain the same starting sequence number value. The starting sequence number in the penultimate and last replay flit shall be the same to allow the receiver time to decode the flit and enable or disable processing of the immediately following TL flit.

5.2.16 [Category DL 3.1] Reserved (79:76)

This is a reserved field.

5.2.17 [Category DL 3.1] Acknowledge Sequence Number (75:64)

This field defines the sequence number for the next TL flit that is required to be received. It may change between consecutive replay flits. The receiving logic passes this value to the transmit logic to indicate where it should start the data replay.

5.2.18 DL Link Information (63:0)

This field is a message that is passed across the link. The host transmits the contents of a register in this field. The host receives the message and stores it in a register that may snooped by software.

5.3 Other Run Lengths

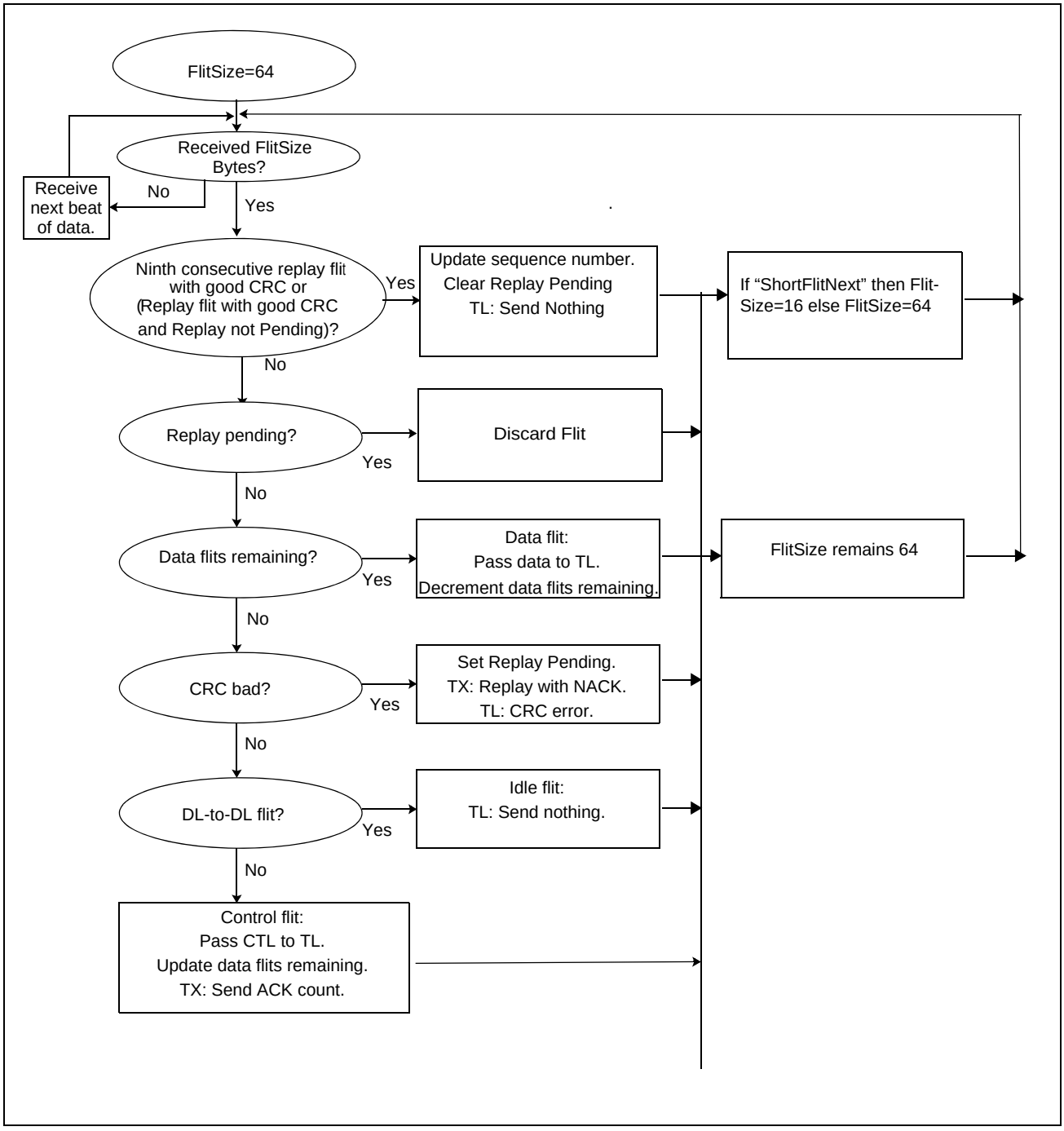
The other run lengths of x‘9’, x‘B’, x‘C’, x‘D’, and x‘E’ are reserved values for future DL-to-DL features.

6. Flit flow diagrams

This section contains receive and transmit flow diagrams.

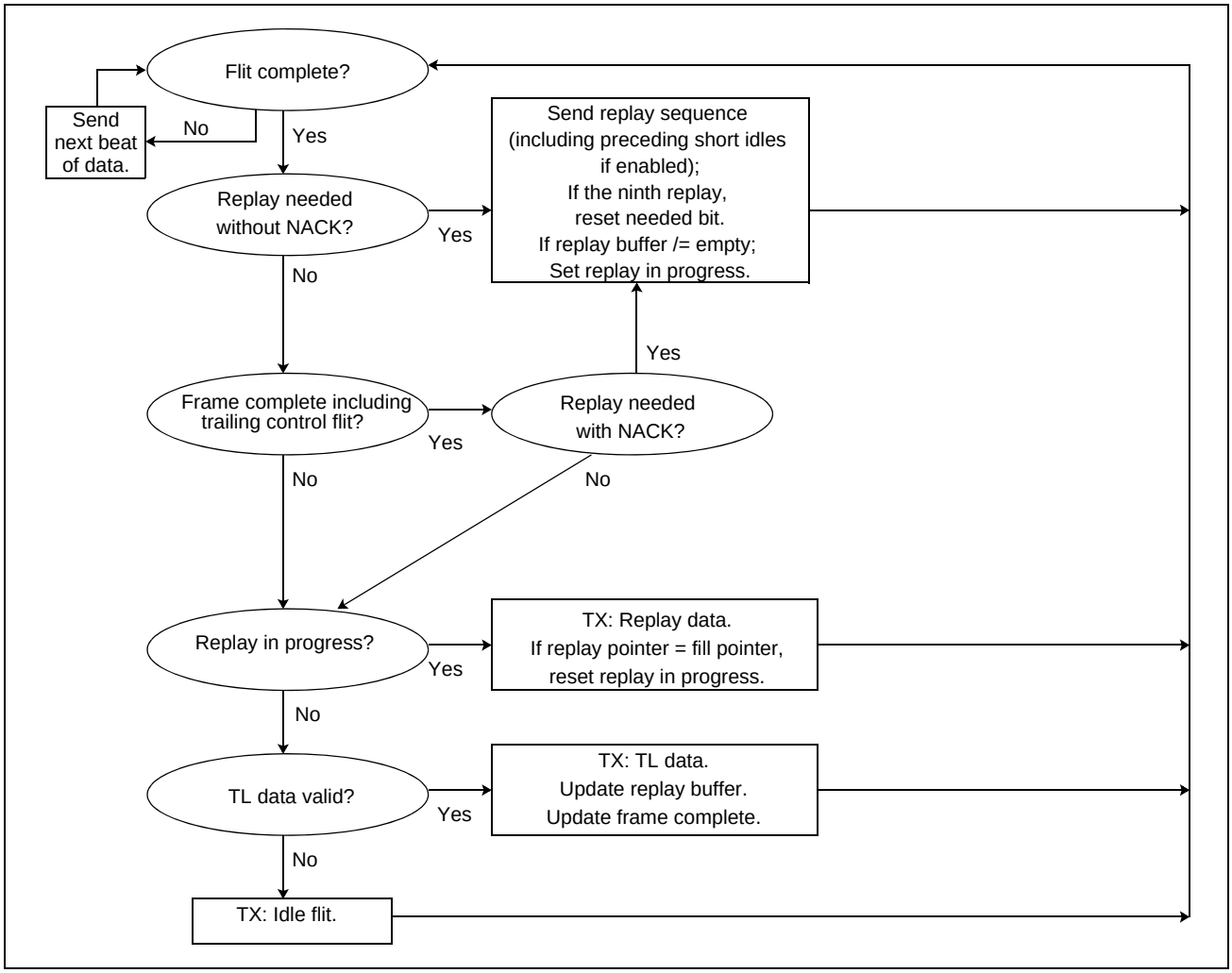
6.1 Receive flow

Figure 6-1. Receive flit flow diagram



6.2 Transmit flow

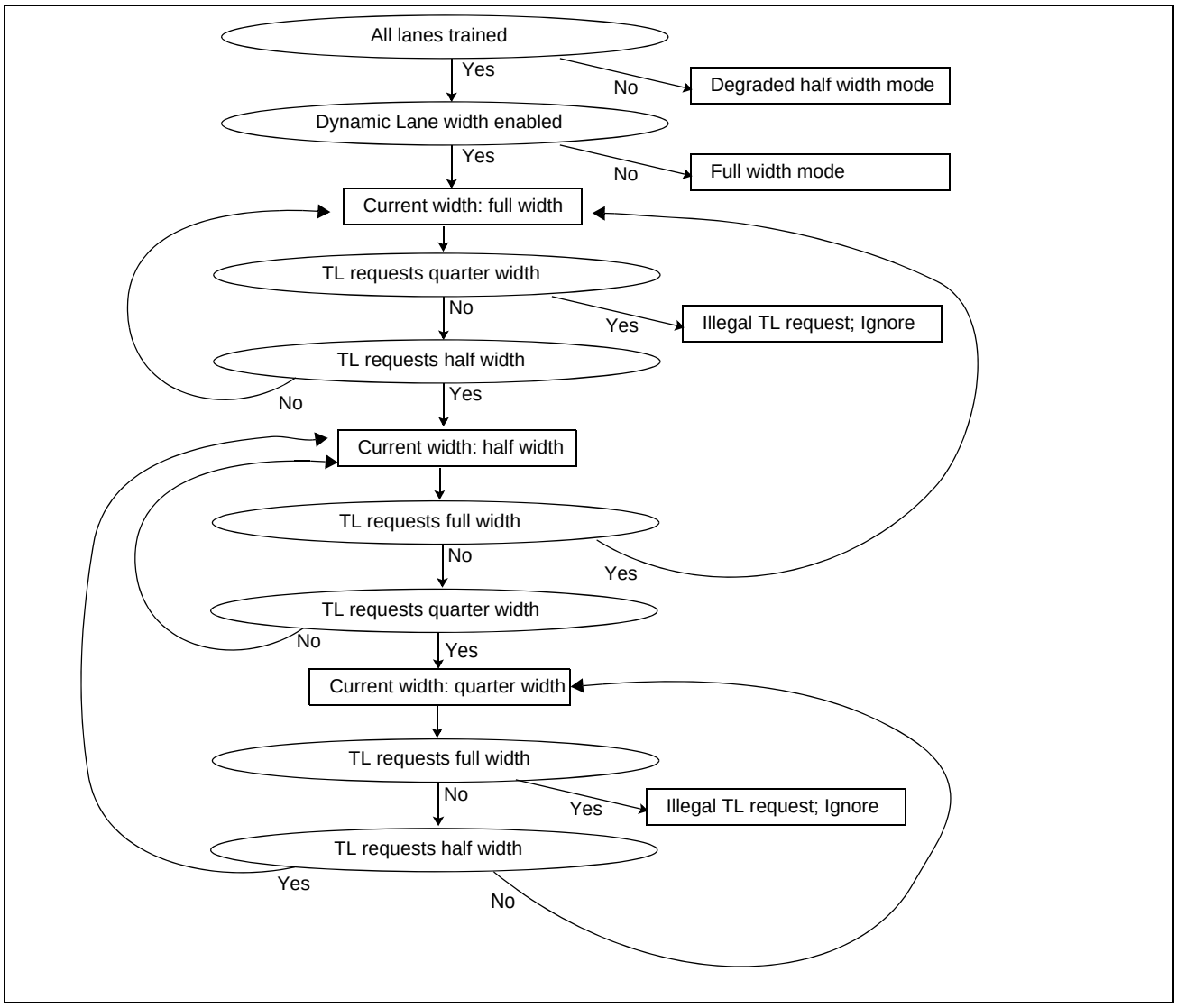
Figure 6-2. Transmit flit flow diagram



7. [Category DL 3.1] Dynamic lane power down

The host DL supports changing the number of lanes that are used to connect a device. These power management messages are contained in Idle and Replay flits. When the link is less than half utilized, the TL traffic may be routed onto half the lanes, and the extra lanes may be powered off to save power. When the device falls behind on its bandwidth, more lanes may be enabled to increase the bandwidth. The TL controls whether the link runs at quarter, half, or full width and the current link width request shall be completed before the TL makes a new link width request. The legal transitions are from full-to-half, half-to-quarter, quarter-to-half, or half-to-full width. Dynamic powering down of lanes is enabled only if all lanes were trained. If a lane fails while running in half- or quarter-width mode for power management, the link retrains in a degraded half-width mode. A powered-down lane shall be periodically powered on and calibrated, determined by a timer in the host device, to decrease the time that is required to increase the link width. If an unexpected retrain occurs while in a powered down state, all of the lanes shall be turned on and the link retrained at full width. An unexpected retrain is a retrain that is not associated with the powering on of more lanes (for example, changing from quarter-to-half width mode).

Figure 7-1. [Category DL 3.1] Dynamic Lane width states



7.1 Example 1: Lane width increase from ×4 half-width to ×8 full-width

1. Host is sending “0010” in Power Management Message (host currently running in half-width mode).
2. Device is sending “0010” in Power Management Message (device is currently running half-width mode).
3. Host TL decides that more bandwidth is needed (Power Management is always initiated by the host).
4. Host starts sending “0101” in Power Management Message (waking from half-width to full-width).
5. Host starts powering up lanes 1, 3, 4, and 6 and sending TS1 on them.
6. Device sees “0101” in Power Management Message and starts powering up lanes 1, 3, 4, and 6 and sending TS1 on them.
7. Device starts sending “0101” in Power Management Message (waking from half-width to full-width).
8. Device completes powering up lanes 1,3,4,6 and receives TS1 on them.
9. Device starts sending “0111” in Power Management Message (ready to switch from half-width to full-width).
10. Host sees “0111” in Power Management Message and when host also completes powering up lanes 1, 3, 4, and 6 and receiving TS1 on them, the host initiates a retraining sequence.
11. Device sees Control Sync Headers and goes into retraining itself.
12. Retraining proceeds at new full-width mode (x8). To speed up retraining, only two matching deskew patterns are required, rather than the typical 8 matching deskew patterns.
13. Link is now operating in full-width mode (x8).
14. Host is sending “0011” in Power Management Message (host is now running in full-width mode).
15. Device is sending “0011” in Power Management Message (device is now running in full-width mode).

7.2 Example 2: Lane width decrease from ×4 half-width to ×2 quarter-width

1. Host is sending “0010” in Power Management Message (host currently running in half-width mode)
2. Device is sending “0010” in Power Management Message (device currently running in half-width mode)
3. Host decides to save power (Power Management is always initiated by the host)
4. Host sends “1001” in Power Management Message (switch from half-width to quarter-width)
5. Host powers off Tx lanes 5 and 2. [Note 2]
6. Host sends “dead cycles” of arbitrary content for a period of time. [Note 1]
7. Host starts sending “0001” in Power Management Message (Tx running in quarter-width mode)
8. Device sees the “1001” in Power Management Message
9. Device powers off Rx lanes 5 and 2
10. Device ignores the receive data for the “dead cycles” time (device uses this time to reconfigure its Rx dataflow for quarter-width mode)
11. Device receives flits in quarter-width mode
12. Device sends “1001” in Power Management Message (switch from half-width to quarter-width)
13. Device powers off Tx lanes 5 and 2. [Note 2]
14. Device sends “dead cycles” of arbitrary content for a period of time. [Note 1]
15. Device starts sending “0001” in Power Management Message (Tx running in quarter-width mode)

16. Host sees the “1001” in Power Management Message
17. Host powers off Rx lanes 5 and 2
18. Host ignores the receive data for the “dead cycles” time (host uses this time to reconfigure Rx dataflow for quarter-width mode).
19. Host receives flits in quarter-width mode.

Note 1: The “dead cycles” duration is equal to the time it takes to send one flit at x4 width.
Note 2: When the Tx lanes are being powered off, rather than immediately turning off, four TS1s should be sent on the Tx first. This process triggers a retrain mechanism if the “1001” (or “1000”) in the Power Management Message encounters an error.

7.3 Recalibration

Periodic explicit recalibration is required by some High Speed Serdes implementations; for example, to keep a lane that is powered-off (by Power Management, and so on) to be brought back into functional use quickly. Data transfer across a lane that is powered on is not affected during its recalibration.

Recalibration is initiated by the Host periodically, where one lane is done (in both directions), per calibration request, ultimately wrapping back to lane 0 when all lanes completed. Consider the following points:

- For a link that is configured as a x8, the order of recalibration is 0, 1, 2, 3, 4, 5, 6, 7
- For a link that is configured as a x4OL, the order of recalibration is 0, 2, 5, 7

This numbering is logical lane number with the number appearing in the deskew markers.

Lanes that are not being actively used because of Power Management or degraded mode are still processed in the normal order.

During recalibration, scrambled data is transmitted on the associated lane. For lanes that Power Management does not require being powered on, scrambled TS1s shall be sent.

The time between successive recalibration requests to the same lane shall be programmable, including values of approximately 24ms, 48 ms, 200 ms, 500 ms, 1s, or disabled.

The two ends of the link coordinate recalibration by using the “Recal Info” field contained in the TL Control flits and DL-to-DL flits. A Device sends “00” in “Recal Info” field while not recalibrating. A Host sends “00” or “01” in the “Recal Info” field while not recalibrating; if the next lane that is to be recalibrated is lane 0, it sends “01”; else it sends “00”.

The procedure to recalibrate one lane is:

1. Host periodic timer expires, triggering a recalibrate.
2. If a Power Management width change is in progress, the Host waits for the Power Management to complete. If subsequent recalibrate requests are received, they are ignored.
3. If a Host Tx is currently powered off, the Host powers the Tx on and waits until it is sending scrambled TS1s.
4. Host starts sending “10” or “11” in “Recal Info” field. These two values are used alternately in successive recalibrations.
5. Device sees new value “10” or “11” in the incoming “Recal Info” field. If previous value was a “01”, then lane number is set to a ‘0’; otherwise, lane number advances from the lane number it used in the previous recalibration.

6. If Device Tx is powered off, the Device powers on the Tx and waits until it is sending scrambled TS1s.
7. Device starts sending “10” or “11” in the “Recal Info” field (same value as received from the Host).
8. If Device Rx is powered off, the Device Rx powers on and waits for the power-on to complete.
9. Device starts recalibrating its Receive High Speed Serdes logic on the associated lane.
10. Host sees new value “10” or “11” in the incoming “Recal Info” field.
11. If Host Rx is powered off, the Host powers the Rx on and waits for power-on to complete.
12. Host starts recalibrating its receive High Speed Serdes logic on the associated lane.

Device recalibration finish:

13. Device completes recalibration of its Rx.
14. Device powers off its Rx, unless Power Management needs it to be powered on.
15. Device starts sending “00” in the “Recal Info” field.
16. Host sees “00” in the incoming “Recal Info” field.
17. Host powers off its Tx, unless Power Management needs it to be powered on.

Host recalibration finish:

18. Host completes recalibration of its Rx.
19. Host powers off its Rx, unless Power Management needs it to be powered on.
20. Host starts sending “00” or “01” in “Recal Info” field. If the next lane to be recalibrated is lane 0, send “01”; else send “00”.
21. Device sees “00” or “01” in the incoming “Recal Info” field.
22. Device powers off its Tx, unless Power Management needs it to be powered on.

Recalibration is complete when both ends of the link are sending “00” or “01” in the “Recal Info” field. No Power Management change width is permitted until recalibration is complete.

In some error scenarios, the “Recal Info” field from the Device might change from “00” to “10”/”11” and back to “00” without the “10”/”11” ever being received successfully by the Host. The Host shall implement a timeout (two Recalibration periods) to detect this. Recovery is initiated by the Host sending to sending “01” and starting the next recalibrate on lane 0.

8. OpenCAPI data link layer versions

Based on the version number that is exchanged in the deskew blocks during the training process, the OpenCAPI DL may need to be configured to support different functions. *Table 8-1* lists the defined version numbers. Both sides transmit their version number and each side determines the trained mode based on the common settings. If both sides support the primary and secondary modes, the primary mode is used. The secondary method is used only if both sides support it, and at least one side does not support the primary mode. If the host cannot support the endpoint's version, the link does not train. No fail over occurs from primary to secondary modes because of error cases. Reference frequency, supported widths, and power management are not based on version number.

Table 8-1. Data link layer version numbers

Feature	Options	Version									
		0	1	2	3	4	5	6	8	9	10
Transmission order	8 consecutive bytes per lane (Store and Forward mode)	P	P	P		S	S	S			
	2 consecutive bytes per lane (Low-latency mode)				P	P	P	P	P	P	P
Degraded lanes	Odd/Even	P									
	Inside/Outside		P	P	P	P	P	P	P	P	P
Idle flit length	64 Bytes (Long idles)	P	P	P	P	P	P	S		S	P
	16 Bytes (Short idles)							P	P	P	
Error detection per lane	Enabled						P		P	P	P
	Disabled	P	P	P	P	P	S	P	S	S	S
Degraded transmit mode	Enabled (Neighbor first)	P	P								
	Disabled (Lowest byte first)			P	P	P	P	P	P	P	P

Note: P = primary mode; S = secondary mode

Version 0: DL 3.0 host; device/FPGA
Version 1: DL 4.0 device/FPGA (inside/outside; me/neighbor first)
Version 2: DL 4.0 device/FPGA (inside/outside; lowest byte first)
Version 3: DL 3.0 device/ASIC
Version 4: DL 4.0 host, device/ASIC
Version 5: Version 4 including error detection per lane
Version 6: Version 4 including short idle flits
Version 8: DL 3.1 memory buffer/ASIC
Version 9: DL 3.1 host memory slot; storage class memory/ASIC
Version 10: Version 8 but with long idles instead of short idles

The version number interoperability is listed in *Table 8-2*. The version number compliance are listed in *Table 8-3*.

Table 8-2. Version number interoperability

V# = Version number		Endpoints									
		V0	V1	V2	V3	V4	V5	V6	V8	V9	V10
Host	V0	X	L	L		L	L	L			
	V4	L	L	X	X	X	X	X		X	X
	V5	L	L	X	X	X	X	X		X	X
	V6	L	L	X	X	X	X	X	X	X	X
	V9				X	X	X	X	X	X	X

Note:

1. X = Supported
2. L = Limited support (full-width only, degrade lane ability is lost)
3. Empty cell/dark shaded = Not supported and the link will not train

Table 8-3. Version number compliance

V# = Version number		Endpoints									
		V0	V1	V2	V3	V4	V5	V6	V8	V9	V10
Host	V0	3									
	V4	O^3	O^4	4	O^3	4	O^4	O^4		31	O^4
	V5	O^3	O^4	4	O^3	4	O^4	O^4		31	O^4
	V6	O^3	O^4	4	O^3	4	O^4	O^4	O^{31}	31	O^4
	V9								31	31	

Note:

1. 3 = Required for a Host to be DL 3.0 compliant
2. 4 = Required for a Host to be DL 4.0 compliant
3. 31 = Required for a Host to be DL 3.1 compliant
4. O^3= Optional for a Host to be DL 3.0 compliant
5. O^4= Optional for a Host to be DL 4.0 compliant
6. O^{31}= Optional for a Host to be DL 3.1 compliant
7. An endpoint, is at a minimum, required to implement a single valid version number. The burden of compliance is maintained by the host.

9. Replay buffer

The Replay buffer holds TL flits that were sent, but for which the sending TX did not receive confirmation that they were received correctly on the other device's RX.

When an RX detects a CRC error, it shall send a replay request to the TX. After the next TL control flit or DL-to-DL flit, the TX shall send a minimum of nine replay flits with NACK set.

When the RX receives a replay flit with a NACK, it notifies the TX to initiate a replay flit stream from the last CRC good ACK. When the TX finishes its current flit, it sends a replay flit stream without a NACK. The stream includes the starting replay sequence number and the ACK sequence number, followed by data from the replay buffer.

When the RX detects any replay flit, it checks to see if its starting sequence number was already received successfully. If so, it drops the flits until it gets to the sequence number that it needs. The RX acknowledges only flits that are needed; duplicate flits are not acknowledged.

The CRC covers all bits that were transmitted excluding the sync headers. However, only the TL flits shall be stored in the replay buffer because the DL-to-DL special flits are exempt from being required to be replayed.

If forward progress is not made within a programmable amount of time, the link is retrained.

9.1 Replay buffer sizing

Each device shall determine its own replay buffer depth if all 12 bits of the sequence number are exchanged in the replay flits.

Table 9-1 lists some characteristics of the how to size a replay buffer.

Table 9-1. Replay buffer characteristics

Parameters	Description
Latency of cable = 20 ns	(5 ns per meter for optical cables) × (2 meter cable) = 10 ns of delay in the wire in each direction.
Latency of cable retimers = 28 ns	14 ns in each direction
Latency internal to the endpoint = 150 - 200 ns	Largest estimate from partners.
Latency to transmit a nine-flit frame = 22 ns	9 × 4 = 36 cycles
25.78125 Gbps per lane	16 bits every 1.6113 GHz cycle
Total cycles	270 ns × 1.611 GHz = 434 cycles for the round trip
×8 DL	16 bytes every cycle
Replay buffer minimum size	512 × 16 bytes (16 bytes data) which holds 128 flits (each 64 bytes in size)

Note: If a longer cable is used, the replay buffers are the limiting factor and performance is degraded. Therefore, a replay-buffer-full condition does not force a replay.

9.2 Control pointers

Three pointers control the replay buffer:

- The write pointer indicates where new entries are stored; it advances with each new entry.
- The replay pointer indicates the current read location; it is used only during replays.
- The ACK pointer advances when confirmation of successful reception by the other end of the link is confirmed

When a replay is requested, the replay pointer is set to the ACK pointer. The data from the replay buffer is resent across the link until the replay pointer equals the write pointer.

10. Encoding and scrambling

10.1 Encoding

The DL uses the 64/66 encoding across each lane of the link. For data blocks, the first two bits are ‘01’, and the next 64 bits are data. For control blocks, the first two bits are ‘10’, and the next 64 bits are control information. The control blocks are only used for training and retraining the link. All TL and DL flits use the data blocks across the link.

DL data block format (per lane):

- 2-bit sync header: ‘01’ if Error Detection per Lane is Disabled
- 2-bit sync header: ‘01’, ‘00’, 11’ if Error Detection per Lane is Enabled (see below)
- 64-bit data

Engineering note

For each 2-bit value in this section, the left bit is sent down the wire first.
For example, ‘10’ means send ‘1’ followed by a ‘0’.

DL control block format (per lane):

- 2-bit sync header: ‘10’
- 64-bit control information

10.2 Error detection per lane

When error detection per lane is enabled, after training, the data blocks calculate parity across the previous 64 bits (before scrambling) and use sync header encodes of ‘01’ to indicate an even number of bits were set and alternate between ‘00’ and ‘11’ to indicate odd number of bits were set. Also, the first header (after training or re-training) is always ‘01’ in the case where there were no data bits prior. The receiving side conducts the same parity check and reports an error if the threshold over a period of time is exceeded.

10.3 Scrambling

After PHY training, the transmitter scrambles all data by using a PRBS23 algorithm of polynomial $X^{23} + X^{21} + X^{16} + X^{8} + X^{5} + X^{2} + 1$. During DL initialization, the receiver captures the state of the PRBS23 and uses it to unscramble the data. Sync headers are not scrambled. A CRC error occurs if the receiver is ever out of synchronization with the transmitter. The link retrains, and the receiver recaptures the state of the PRBS23.

Engineering note

1. The transmit scrambler is not reset when the link retrains.
2. Each lane on the DLX TX may initialize the scrambler differently to reduce the effects of crosstalk.

Table 10-1 on page 57 is a logic example that how the scrambler bit stream is generated with a 23-bit binary shift register.

Figure 10-1. Generating a scrambler bit stream