Specification for Unified Protocol (UniPro®)

Version 1.8 – 13 September 2017

MIPI Board Adopted 11-Jan-2018

* CAUTION TO IMPLEMENTERS *

This document is a MIPI Specification. MIPI member companies' rights and obligations apply to this MIPI Specification as defined in the MIPI Membership Agreement and MIPI Bylaws.

The UniPro v1.8 Specification is a feature update succeeding the UniPro v1.61 Specification.

The UniPro v1.8 Specification references the following documents:

- *Specification for M-PHY®, Version 4.1*
- *Specification for Device Descriptor Block (DDB), Version 1.0*

Backwards Compatibility

The UniPro v1.8 Specification was designed to ensure interoperability with UniPro v1.6x. A few features defined in earlier versions of UniPro have been obsoleted in UniPro v1.8, mainly because they proved to be less useful or outdated. With the removal of features, the statement of interoperability relates only to those features that are common to UniPro v1.8 and predecessor versions. Backward compatibility to UniPro versions v1.41.00 and earlier is not maintained.

Changes

- Removed support for T-MPI (outdated)
- Deprecated hibernate entry while in LinkDown
- Revised LinkStartup
- Deprecated support for advanced Media Converters being programmed inline via M-PHY Line-Config state
- Added protocol support for HS-G4 transmission in M-PHY
- Started a path to deprecate the need for "Dummy Burst"

Known Issues

- No known issues.

NOTE: LICENSING OBJECTION RECEIVED

Implementers should be aware that a licensing objection was received from Intel in regard to version 0.80.00 of this Specification, MIPI Alliance Standard for Unified Protocol (UniPro), Version 0.80.00, dated 6 September 2006 and adopted by the MIPI Board of Directors on 26 February 2007 (UniPro v0.80.00). The UniPro v0.80.00 Specification is available at https://members.mipi.org/wg/UniPro/document/51407. Intel's licensing objection is available at https://members.mipi.org/wg/All-Members/document/9315.

Licensing objections are defined in Article X, Section 1 of the MIPI Bylaws. Member companies considering implementation or use of this standard are strongly encouraged to review this information.

UniPro v0.90.00 was drafted by the MIPI UniPro Working Group taking into account the Licensing Objection of Intel with respect to UniPro v0.80.00. UniPro v0.90.00 was renumbered to v1.00.00 upon approval by the MIPI Board as a MIPI Specification. The UniPro v1.00.00 Specification is available at https://members.mipi.org/wg/UniPro/document/51463.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Following the WG completion of UniPro v0.90.00, Intel expressed the following view:

> Intel notes that the same Necessary Claims identified in our prior License Objection are applicable to this latest Draft. However, Article X, Sec 1(3) of the Bylaws provides that since Intel properly submitted its Licensing Objection, “[Intel's] licensing obligations under the terms of its MIPI Membership Agreement shall terminate” with respect to those identified Necessary Claims. Accordingly, since those Necessary Claims are no longer subject to any MIPI licensing obligation for any subsequently adopted Specifications, Intel's prior License Objection stands and applies to this latest Draft.

The “latest Draft” referred to in the advice of Intel is UniPro v1.00.00, originally drafted by the UniPro Working Group as UniPro v0.90.00 and later renumbered to UniPro v1.00.00 during subsequent approval by the MIPI Board as a MIPI Specification. MIPI has no view as to the accuracy or validity of the statement made by Intel.

ALL MIPI MEMBERS AND ANY OTHER INTERESTED PARTIES ARE ADVISED THAT MIPI TAKES NO POSITION WITH REFERENCE TO, AND DISCLAIMS ANY OBLIGATION TO INVESTIGATE OR INQUIRE INTO, THE SCOPE OR VALIDITY OF ANY LICENSE OBJECTION OR CLAIM OR ASSERTION OF NECESSARY CLAIMS (AS DEFINED IN THE MIPI MEMBERSHIP AGREEMENT) WITH RESPECT TO UNIPRO V0.90.00 AND THE APPLICABILITY OF THE LICENSE OBJECTION OF INTEL TO UNIPRO V0.90.00.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Specification for Unified Protocol (UniPro®)

Version 1.8

13 September 2017

MIPI Board Adopted 11 January 2018

This document is a MIPI Specification. MIPI member companies' rights and obligations apply to this MIPI Specification as defined in the MIPI Membership Agreement and MIPI Bylaws.

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

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

NOTICE OF DISCLAIMER

The material contained herein is provided on an “AS IS” basis. To the maximum extent permitted by applicable law, this material is provided AS IS AND WITH ALL FAULTS, and the authors and developers of this material and MIPI Alliance Inc. (“MIPI”) hereby disclaim all other warranties and conditions, either express, implied or statutory, including, but not limited to, any (if any) implied warranties, duties or conditions of merchantability, of fitness for a particular purpose, of accuracy or completeness of responses, of results, of workmanlike effort, of lack of viruses, and of lack of negligence. ALSO, THERE IS NO WARRANTY OR CONDITION OF TITLE, QUIET ENJOYMENT, QUIET POSSESSION, CORRESPONDENCE TO DESCRIPTION OR NON-INFRINGEMENT WITH REGARD TO THIS MATERIAL.

IN NO EVENT WILL ANY AUTHOR OR DEVELOPER OF THIS MATERIAL OR MIPI BE LIABLE TO ANY OTHER PARTY FOR THE COST OF PROCURING SUBSTITUTE GOODS OR SERVICES, LOST PROFITS, LOSS OF USE, LOSS OF DATA, OR ANY INCIDENTAL, CONSEQUENTIAL, DIRECT, INDIRECT, OR SPECIAL DAMAGES WHETHER UNDER CONTRACT, TORT, WARRANTY, OR OTHERWISE, ARISING IN ANY WAY OUT OF THIS OR ANY OTHER AGREEMENT RELATING TO THIS MATERIAL, WHETHER OR NOT SUCH PARTY HAD ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

The material contained herein is not a license, either expressly or impliedly, to any IPR owned or controlled by any of the authors or developers of this material or MIPI. Any license to use this material is granted separately from this document. This material is protected by copyright laws, and may not be reproduced, republished, distributed, transmitted, displayed, broadcast or otherwise exploited in any manner without the express prior written permission of MIPI Alliance. MIPI, MIPI Alliance and the dotted rainbow arch and all related trademarks, service marks, tradenames, and other intellectual property are the exclusive property of MIPI Alliance Inc. and cannot be used without its express prior written permission. The use or implementation of this material may involve or require the use of intellectual property rights (“IPR”) including (but not limited to) patents, patent applications, or copyrights owned by one or more parties, whether or not members of MIPI. MIPI does not make any search or investigation for IPR, nor does MIPI require or request the disclosure of any IPR or claims of IPR as respects the contents of this material or otherwise.

Without limiting the generality of the disclaimers stated above, users of this material are further notified that MIPI: (a) does not evaluate, test or verify the accuracy, soundness or credibility of the contents of this material; (b) does not monitor or enforce compliance with the contents of this material; and (c) does not certify, test, or in any manner investigate products or services or any claims of compliance with MIPI specifications or related material.

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

MIPI Alliance, Inc.
c/o IEEE-ISTO
445 Hoes Lane, Piscataway New Jersey 08854, United States
Attn: Managing Director

Contents

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Figures

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Tables

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Release History

Date	Release	Description
2011-04-28	1.40.00	Board approved release.
2012-07-31	1.41.00	Board approved release.
2013-09-30	v1.6	Board approved release.
2015-12-17	v1.61	Board approved release
2018-01-11	v1.8	Board approved release

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

1 Introduction

The UniPro Specification is applicable to a wide range of Device types such as Application processors, co-processors, modems, storage subsystems including non-volatile memory modules, displays, camera sensors, 3D graphics and multimedia accelerators, to name the most obvious. It is also applicable to different types of data traffic such as control messages, bulk data transfer and packetized streaming.

Other MIPI Alliance Specifications are used for definition of physical layer and Application Layer protocols. As such, this Specification can be thought of as a member of a family of Specifications.

To aid in understanding the concepts presented in this Specification, the UniPro description is loosely based on the ISO OSI Reference Model (OSI/RM). See ***Section 4.2*** for more information on the similarities and differences to the OSI/RM.

1.1 Scope

This document defines the protocol used to transfer data between Devices that implement the UniPro Specification. This includes definitions of data structures, such as Packets and Frames, used to convey information across the Network. In addition, flow control, error handling, power and state management, and connection services are also within the scope of this document.

The PHY Adapter, Data Link, Network and Transport layer protocols, and the Device Management Entity (DME) are described in ***Section 5***, ***Section 6***, ***Section 7***, ***Section 8***, and ***Section 9***, respectively.

The electrical interface, physical layer timing and encoding, Application-specific protocols and command sets are out of scope for this document.

Figure 1 shows the scope of this document as it relates to the OSI/RM. Throughout this document, layer references are to the UniPro layering scheme unless otherwise indicated.

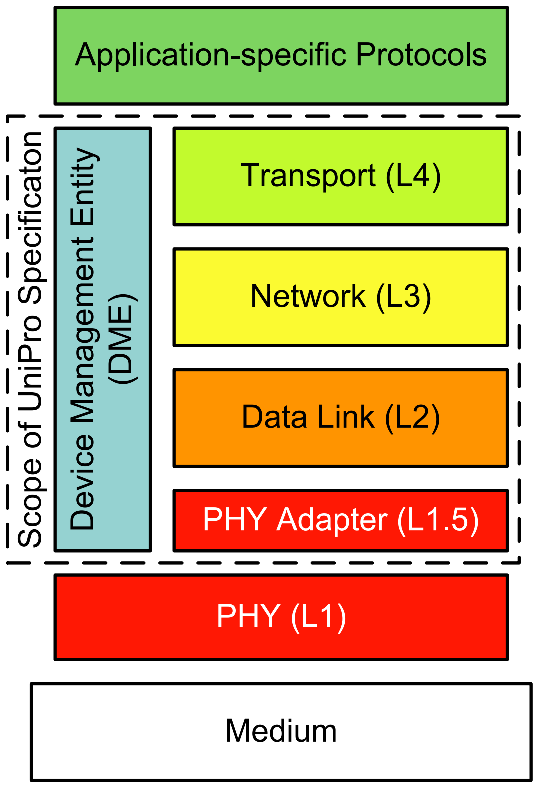

Application-specific Protocols
Scope of UniPro Specificaton
Device Management Entity (DME)
Transport (L4)
Network (L3)
Data Link (L2)
PHY Adapter (L1.5)
PHY (L1)
Medium

Figure 1 Scope of the UniPro Specification

1.2 Purpose

This Specification can be used by manufacturers to design products that adhere to MIPI Alliance Specifications for host processor and peripheral interfaces.

Implementing the UniPro Specification reduces the time-to-market and design cost of mobile Devices by simplifying the interconnection of products from different manufacturers. In addition, implementing new features is simplified due to the extensible nature of the UniPro Specification.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

2 Terminology

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

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

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

Numbers are decimal unless otherwise indicated. A prefix of 0x indicates a hexadecimal number, while a prefix of 0b indicates a binary number.

This document uses the C/Verilog representation for operators where bitwise AND is represented by ‘&’, bitwise OR is represented by ‘|’, bitwise XOR is represented by ‘^’ and 1’s complement (negation) is represented by ‘~’.

Named constants, e.g. FAST_STATE, and Service Primitive names (***Section 2.2***) are shown in all capital letters.

2.1 Definitions

Application (Layer): Logic or functionality within a Device that uses the services provided by the Transport Layer and DME of the UniPro stack.

Attribute: Atomic unit of information that can be read or written from an Application using DME_GET or DME_SET primitives, and DME_PEER_GET or DME_PEER_SET primitives. Attributes can be used to configure the behavior of a UniPro stack, to determine the current state of the interface or to determine the availability of interface options and interface capabilities.

Big-endian: This term describes the order in which a sequence of bytes are stored in computer memory. Big-endian is an order in which the “big end” (most significant value in the sequence) is stored at the lowest memory address.

Checksum: See Cyclic Redundancy Check (CRC).

Clock Lane: A unidirectional interconnect between two UniPorts which carries clock information needed for decoding the data on the Data Lanes. Some PHY technologies require this. The PHY Adapter Layer abstracts from whether or not a Clock Lane is required.

Component: A physical entity such as an integrated circuit containing at least one Device or switch.

Composition: The process of adding header, and possibly trailer, information to the PDU of an upper layer to build a lower layer PDU.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Connection: A Connection is a bidirectional, logical communication channel between two CPorts. It is a concept used for Connection-oriented data transmission, whereby two Devices need to be connected before data can be exchanged.

Connection-oriented data transmission: The transfer of a SDU from a source Service Access Point to a destination Service Access Point within the context of a Connection that has previously been established.

CPort: A CPort is a Service Access Point on the Transport Layer (L4) within a Device that is used for Connection-oriented data transmission.

CPort Safety Valve: A mechanism whereby a CPort discards received data whenever the CPort cannot deliver Segments at the rate at which they arrive. The mechanism is intended to manage Network congestion and avoid certain types of system-level deadlocks.

Critical (M-PHY®) Attributes: A set of control Attributes for M-PHY MODULEs that get special treatment within UniPro L1.5 because they are set using the UniPro PA Control Protocol (PACP).

Cyclic Redundancy Check (CRC): A coding scheme to detect errors in transferred data. The CRC is used to produce a checksum, which is typically a small number of bits, for a larger block of data.

Credit-based Flow Control: A method to control the speed of data transfer between two entities. The receiving side notifies the sending side that space is available in the receiving side buffer by issuing credits. The sending side may then transfer data up to the credit limit.

Data Lane: A physical interconnect between two UniPorts which carries data. A Link can consist of 1, 2, 3 or 4 Data Lanes per direction. A Data Lane in UniPro is used unidirectionally. A Lane with embedded clocking information is also considered a Data Lane.

Data Link (DL) Layer (L2): A Protocol Layer. The Data Link Layer provides the functional and procedural means to transfer data between two adjacent L2 instances. The Data Link Layer also detects, and possibly corrects, errors that may occur in the Physical Layer. In addition, the Data Link Layer maintains Flow Control between the L2 entities.

(Data Link Layer) Control Frame: A Layer 2 PDU, which is consumed by the Data Link Layer.

(Data Link Layer) Data Frame: A Layer 2 PDU whose SDU is passed on to upper layers.

Data Link Layer Flow Control: Flow Control implemented on Data Link Layer (L2) using credits. Data Link Layer Flow Control is also referred to as Link-Level Flow Control.

(Data Link Layer) Frame: A Layer 2 PDU for point-to-point transmission across an individual Link. Frames may be either Data Frames or Control Frames.

Data Link Layer Symbol: The DL Layer data stream is constructed of atomic 17-bit symbols. These symbols can be either control or data symbols.

Decomposition: The process of removing header, and possibly trailer, information from the PDU of a lower layer to extract an upper layer PDU.

Device: An addressable node on the Network. A Device needs to comply with Network Layer and Transport Layer (L3 and L4) requirements as defined in the UniPro Specification. An off-chip Link also needs to comply with PHY Adapter Layer and Data Link Layer (L1.5 and L2) requirements as defined in the UniPro Specification, and with a Physical Layer (L1) as defined in ***[MIPI02]***.

DeviceClass: A property of a UniPro Device that indicates what the UniPro Device does and what Application-level protocols it supports. Note that the DeviceClass concept defined here is intended to match that used in ***[MIPI01]***.

DeviceID: A number which uniquely identifies a Device on the Network.

Device Management Entity (DME): The Device Management Entity is a layer-independent entity interfacing with all UniPro Protocol Layers and the Application Layer. The DME may set or get layer Attributes and signal and control the resetting of individual layers.

Dual Simplex: Supporting independent communication in forward and reverse direction simultaneously.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Endpoint: Device that is a leaf of a Network.

End-to-End Flow Control (E2E FC): A credit-based flow control method that is implemented in the Transport Layer (L4) on a Connection.

Fragment: A portion of a Message that can be passed to, or received by, a CPort. Received Fragments are not generally identical to transmitted Fragments.

Flow Control: The process of managing the flow of data from one entity to another to ensure that the receiving entity can handle all of the incoming data.

Frame: see Data Link Layer Frame.

Gear: A mechanism defined in ***[MIPI02]*** that provides control over the speed of the Link in steps of 2x. Higher values of HS Gear or PWM Gear imply higher frequencies and higher bandwidths.

Get command: A command used to read the value of an Attribute.

Host: This is a Device that has computation and storage capabilities that allow it to control and manage other Devices on a network. A host is associated sometimes with the main CPU.

Lane: A unidirectional differential Data or Clock line pair within a Link.

Link: A bidirectional interconnection between two Devices or Switches.

Link Startup Sequence: The Link Startup Sequence is an L1.5 handshake to establish initial Link communication between two directly attached UniPorts.

Logical Lane Number (M-PHY): A number assigned by a Device to its outbound Data Lanes after the Link Startup Sequence. It is used for L1.5 Lane mapping whereby only the usable Physical Lanes are assigned a Logical Lane Number.

Long Header Packet: Packet that has an extended L3 header. Long Header packets are handled by the using the Long Header Trap feature.

Long Header Trap: A UniPro L3 feature that allows received Long Header Packets to be passed directly to the Application Layer as well as allowing the Application Layer to send Long Header Packets.

Master (entity): The master within a pair of peer entities is the side that takes the initiative and that selects the slave entity to communicate with. Note that for Notifications, however, the slave sends the Notification to the master.

Maximum Transfer Unit (MTU): A term for the size of the largest SDU that can be transferred in a single PDU of a given Protocol Layer.

Message: The PDU of the Application Layer (LA).

M-PHY: High-speed serial interconnect technology, based on clock embedding, as defined in ***[MIPI02]***.

Network: Communication structure allowing two or more Devices to exchange data via a communication protocol.

Network Layer (L3): A Protocol Layer. The part of the communication protocol that manages the Network and routes Packets to their destinations.

Operating Modes: UniPort-specific operating modes for power consumption and performance.

(OSI) Layer 2: See Data Link Layer.

(OSI) Layer 3: See Network Layer.

(OSI) Layer 4: See Transport Layer.

(PA) Symbol: The PDU of the PHY Adapter Layer (L1.5).

Packet: The PDU of the Network Layer (L3).

PACP (Control) frame: A PA Layer PDU used for power control purposes, peer Device Attribute setting and getting, and M-PHY testing.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Peer: In the definition of a protocol layer, two entities are considered Peers if the protocol can be defined as an interaction between both entities. Typically, Peers reside at the two ends of a Link (for lower OSI layers), or serve as communicating endpoints on the network (for higher OSI layers).

Peripheral: Any external Device that provides input or output services for the Host.

PHY Adapter Control Protocol (PACP): A control protocol within the PHY Adapter Layer (L1.5) used for setting power modes, peer Device Attribute setting and getting, and M-PHY testing handling.

PHY Adapter Layer (L1.5): A Protocol Layer. The PHY Adapter Layer presents a common interface to the Data Link Layer for the supported Physical Layer.

PHY-Protocol Interface (PPI): The Interface between the Physical Layer (PHY) and the PHY Adapter Layer.

Physical Lane Number (M-PHY): A static number assigned at the implementation phase of a Device to its outbound and inbound Data Lanes. The Physical Lane Number plays a role in the L1.5 Lane remapping to compensate for wiring topology.

Pin Count: Number of pins required to implement a physical port. Typically includes signal, power, and control pins.

Power Mode: The UniPro Power Mode is a configurable Attribute used to control the UniPro Power State. Various Power Modes have a one-to-one mapping to Power States while other Power Modes give the PHY Adapter Layer (L1.5) the freedom to autonomously switch between two pre-determined Power States.

PHY Power State: The PHY power state is defined in ***[MIPI02]*** for M-PHY. Please see ***[MIPI02]*** for further details.

UniPro Power State: The UniPro Power State is an abstracted representation within the PHY Adapter Layer (L1.5) of the current power state of the underlying Physical Layer.

Preemption: Nesting of higher priority DL Layer Data or DL Layer Control Frames into lower priority DL Layer Data Frames.

Protocol Control Information (PCI): Information exchanged between Layer-(x) instances to coordinate their joint-operation.

Protocol Data Unit (PDU): A unit of data consisting of Layer-(x) Protocol Control Information (PCI) and a Layer-(x) Service Data Unit (SDU).

Quality of Service (QoS): The ability to guarantee certain properties of communication. QoS is used to provide dedicated bandwidth and latency, and improved loss characteristics.

Reserved Field: All the bits in such a field are set to the reserved value.

Routing: The determination of the path within a Network along which a Packet is sent.

Segment: The PDU of the Transport Layer (L4).

Segmentation: The function of splitting a Message or Fragment into Segments within the Transport Layer transmitter.

Selector: An identifier of a range of Attributes associated with a given entity that can have multiple occurrence (i.e., CPort).

Service Data Unit (SDU): Data provided by the User of a given Layer that is transferred without interpretation or changes to a peer Layer.

Service Access Point (SAP): The point at which Layer-(x) services are provided by Layer-(x) to Layer-(x+1).

Service Primitive: An abstract, and implementation independent, representation of an interaction between a Layer(x)-service-user and a Layer-(x)-service-provider.

Service Provider: A layer that provides a SAP for use by another layer.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Service User: A layer that accesses a SAP provided by another layer.

Set command: A basic command used to change the value of an Attribute.

Short-header Packet: Packet that has a single-byte L3 header used for Connection-oriented data transfer.

Slave (entity): The Slave within a pair of peer entities is the passive entity that waits for a request from a Master entity before taking action or responding. Note that for Notifications, however, the Slave sends the Notification to the Master.

Switch: A switch routes Packets through the Network. It contains a UniPro-compliant L3 with routing functionality. A switch definition is not part of this version of the Specification and shall be made available in a future version of the Specification.

Switch Port: The physical interface of a Switch, needed to attach physically to a Network A switch definition is not part of this version of the Specification and shall be made available in a future version of the Specification.

Synchronization: A mechanism used by the PHY Adapter Layer (L1.5) to detect and compensate for timing skew between multiple M-PHY Data Lanes.

Tester: A Device running the UniPro test suite and used for testing a peer UniPro Device-Under-Test (DUT).

TstDst: A programmable analyzer of incoming Messages with the capability to check a stream of incoming Messages against the expected values and lengths and report errors.

TstSrc: A programmable traffic generator that creates sequences of Messages containing well-defined byte sequences of well-defined lengths.

Traffic Class: A priority equivalence class of Messages/Packets/Frames. Frames from a higher-priority Traffic Class are allowed to preempt lower-priority Traffic Classes.

Transport Layer (L4): A protocol layer that can transfer Segments between Devices. The layer's functionality and concepts include Message Segmentation, Connections, End-to-End Flow Control and in-order data delivery per Connection.

TPWM_TX: Unit Interval in LS_MODE. This is the duration of one transmitted bit when using M-PHY, see ***[MIPI02]***.

UniPort: Interface implementing the UniPro Specification (L4 to L1.5) and a physical layer (L1) Specification. The UniPort definition excludes the UniPro User or any Application Layer specifications.

UniPort-M: a UniPort that uses M-PHY ***[MIPI02]*** for the Physical Layer (L1).

UniPro: Unified Protocol.

UniPro User: An Application that resides above UniPro and uses it via the DME SAP and the Transport Layer SAP.

2.2 Service Primitives

This document uses an OSI-conforming name convention for Service Primitives. Service Primitive names are structured as follows:

```
<service-primitive> ::= <name-of-service-primitive> ( {<parameter>, }* )
<parameter> ::= <service control information> | <Service User data>
<name-of-service-primitive> ::= <layer-identifier>_<service-primitive-name>.<primitive>
<layer-identifier> ::= T | N | DL | PA
<service-primitive-name> ::= e.g. CONNECT | RELEASE | DATA | ACTIVITY_START …
<primitive> ::= req | ind | rsp | rsp_L | cnf | cnf_L
T: Transport        N: Network     DL: Data Link        PA: PHY Adapter
```

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

See ***Annex C*** for additional details.

2.3 Abbreviations

etc.	And so forth (Latin: et cetera)
e.g.	For example (Latin: exempli gratia)
i.e.	That is (Latin: id est)

2.4 Acronyms

ACK	Acknowledgment
AFC	Acknowledgment and Flow Control
A_PDU	Application Protocol Data Unit
CCITT	Comité Consultatif International Téléphonique et Télégraphique
CO	Connection Oriented
COF	Continuation of preempted Frame
CRC	Cyclical Redundancy Checking
CReq	Credit Transmit Request
CSD	Controlled Segment Dropping
DL	Data Link
DL_LM	Data Link Layer Management
DL_PDU	Data Link Protocol Data Unit
DL_SAP	Data Link Service Access Point
DL_SDU	Data Link Service Data Unit
DME	Device Management Entity
DT SH N_PDU	Data N_PDU with Short L3 Header
DT SH T_PDU	Data N_PDU with Short L4 Header
DUT	Device-Under-Test
E2E FC	End-to-End Flow Control
EFSM	Extended Finite State Machine
EMI	Electromagnetic Interference
EOF_EVEN	End of Frame for even L2 SDU
EOF_ODD	End of Frame for odd L2 SDU
EOM	End of Message
EoT	End of Transmission
ESC_DL	Data Link Layer Control Symbol Identifier
FC	Flow Control
FCT	Flow Control Token

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

HOL	Head of Line
HS	High Speed
HS_MODE	M-PHY mode of operation for High Speed transmission
ID	Identifier
ISO	International Organization for Standardization
ISTO	Industry Standards and Technology Organization
LA	Application Layer
LRST	Line Reset
LS	Least Significant
LSB	Least Significant Bit
LSS	LinkStartup Sequence
LS_MODE	M-PHY mode of operation for Low Speed transmission
Mbps	Megabit per second
MIB	Management Information Base
MIPI	MIPI Alliance, Inc.
MS	Most Significant
MSB	Most Significant Bit
MSC	Message Sequence Chart
MTU	Maximum Transfer Unit
NAC	Negative Acknowledgment Control
N_LM	Network Layer Management
N_PDU	Network Protocol Data Unit
N_SAP	Network Service Access Point
N_SDU	Network Service Data Unit
OSI	Open Systems Interconnection
OSI/RM	Open Systems Interconnection Reference Model
PA	PHY Adapter
PA_LM	PHY Adapter Layer Management
PA_PDU	PHY Adapter Protocol Data Unit
PA_SAP	PHY Adapter Service Access Point
PA_SDU	PHY Adapter Service Data Unit
PCI	Protocol Control Information
PDU	Protocol Data Unit
PHY	Physical Layer
PMC	PowerMode Change

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

POR	Power-on Reset
PPI	PHY-Protocol Interface
QoS	Quality of Service
RReq	Reset Link Request
RX	Receiver
SAP	Service Access Point
SDM	System Device Manager
SDU	Service Data Unit
SI	Symbol Interval. A 10-bit period for the transmission of one symbol. Symbol Interval scales with data rate. The Symbol Interval is applicable when using M-PHY. See ***[MIPI02]***.
SOF	Start of Frame
SOM	Start of Message
SoT	Start of Transmission
TC0	Traffic Class 0
TC1	Traffic Class 1
TF	Test Feature
TL	Transport Layer
T_CO_SAP	Transport Layer Connection-oriented Service Access Point
T_LM	Transport Layer Management
T_PDU	Transport Layer Protocol Data Unit
T_SAP	Transport Layer Service Access Point
T_SDU	Transport Layer Service Data Unit
TX	Transmitter
UI_{HS}	Unit Interval in HS_MODE. The duration of one bit when using M-PHY. See ***[MIPI02]***.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

3 References

[ITUT01] ITU-T Recommendation X.200, *Information technology – Open Systems Interconnection – Basic Reference Model: The basic model*, <http://www.itu.int/rec/T-REC-X/en>, International Telecommunication Union, July 1994.

[ITUT02] ITU-T Recommendation X.210, *Information technology – Open systems interconnection – Basic Reference Model: Conventions for the definition of OSI services*, <http://www.itu.int/rec/T-REC-X/en>, International Telecommunication Union, November 1993.

[ITUT03] ITU-T Recommendation X.25 (1996), *Interface between Data Terminal Equipment (DTE) and Data Circuit-terminating Equipment (DCE) for terminals operating in the packet mode and connected to public data networks by dedicated circuit*, <http://www.itu.int/rec/T-REC-X/en>, International Telecommunication Union, October 1996.

[MIPI01] *Specification for Device Descriptor Block (DDB)*, version 1.0, MIPI Alliance, Inc., adopted 12 October 2011.

[MIPI02] *Specification for M-PHY®*, version 4.1, MIPI Alliance, Inc., adopted 28 March 2017.

[TAN01] Tanenbaum, Andrew, *Computer Networks*, 4th Ed., <http://authors.phptr.com/tanenbaumcn4/>, Prentice Hall, 9 August 2002.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

This page intentionally left blank.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

4 Architecture Overview (informative)

UniPro is structured as a stack of protocol layers (see ***Figure 1***) that roughly follow the OSI Reference Model (OSI/RM) for networking as published in *ITU-T Recommendation X.200, Information technology – Open Systems Interconnection – Basic Reference Model: The basic model **[ITUT01]***. This means that any given protocol layer in the stack provides services that do not depend on higher layer protocols but do depend on the layers below it.

Each layer can be described conceptually as communicating with a peer layer at the other end of the Link (for lower protocol layers) or Network (for higher protocol layers). Communication between higher peer layers tends to be at a high level of abstraction, e.g. exchange of Application-specific units called Messages, while communication between lower peer layers tends to be at a low level of abstraction, e.g., in terms of byte or even bit streams. A specification of a layer thus also defines how a layer's abstract protocol behavior maps to the communication services provided by the layer directly below it (*Computer Networks, **[TAN01]***).

4.1 UniPro Layering

One way to visualize a protocol stack is to see it as a processing pipeline. In order to transmit data, each layer in the stack converts PDUs from the layer above it into PDUs it can process before passing the PDUs to the layer below it.

For example, in UniPro, the Transport Layer (L4) converts Application Level PDUs (Messages) into less abstract PDUs before passing them down to the next layer (L3). Each layer in the stack converts the PDUs from the previous layer into less abstract PDUs until finally, at the bottom layer of the stack (L1), the PDUs are converted into the electrical signals and low-level timing used by the Interconnect.

Similarly, when data is received from the interconnect at the bottom of the stack (L1), the electrical signals and low-level timing are converted into more abstract PDUs before being passed up to the next layer in the stack (L1.5). Each layer converts the PDUs into more abstract PDUs before passing them up to the next layer. Finally, the Transport Layer (L4) converts the PDUs back into Messages.

The UniPro Specification deliberately does not define which parts of the protocol should be implemented in hardware or in software, or which processing should take place concurrently as opposed to sequentially. For maximum performance, all UniPro layers have been designed to run efficiently as a fully pipelined hardware implementation. Implementers may however choose to implement some parts of the stack in software, thereby potentially serializing and slowing down parts of the processing. It is the implementer's responsibility to assess whether the implementation's architecture meets the temporal requirements, e.g. bandwidth and timers, found in the UniPro and PHY Specifications.

In this document, splitting the protocol stack into multiple layers provides a convenient and familiar structure to the protocol description without imposing one particular implementation. The interfaces between the protocol layers are defined at an abstract level known as Service Access Points (SAPs, *ITU-T Recommendation X.210, Information technology – Open systems interconnection – Basic Reference Model: Conventions for the definition of OSI services **[ITUT02]***). These internal interfaces are only intended to structure the Specification and are not intended to define interfaces that allow independently developed implementation modules to work together. A manufacturer may choose to merge two or more layers into a single layer, or subdivide layers, as long as the integral behavior of the defined protocol stack is unchanged.

4.2 Comparison of UniPro to the OSI/RM

Figure 2 compares UniPro to the OSI/RM. Each layer in UniPro provides roughly the same functionality as the corresponding layer in the OSI/RM.

Note that the OSI PHY (Physical) layer has been split into two sub-layers. The upper sub-layer, the PHY Adapter Layer (L1.5), exposes a PHY-independent interface to L2 and is part of the UniPro Specification. The lower PHY sub-layer (L1) is outside the scope of this document. UniPro supports the *Specification for M-PHY® **[MIPI02]***. The PHY Specification is essentially included in this document by reference.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

In addition, layers above the Transport Layer (L4) are outside the scope of this document. These Application-specific protocols may support Devices such as camera or display modules, high-speed modems, co-processors, etc., and may be implemented entirely in hardware or as software running on a general-purpose processor or any other combination of hardware and software.

The Device Management Entity (DME) is not typically shown in OSI protocol stacks, but serves as a control plane to initialize and control the layers involved in the actual data transport. The DME is not involved in data communication.

All references to protocol layers, by name or number, in this document are references to the UniPro stack unless explicitly stated otherwise.

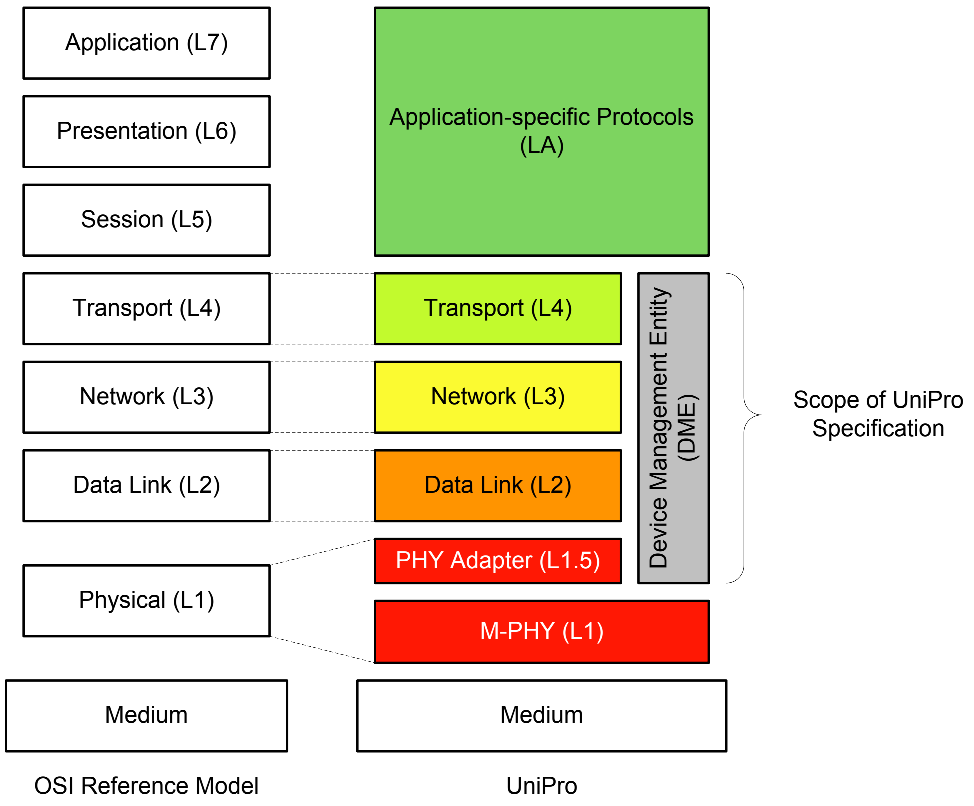


Figure 2 Comparison of UniPro and OSI/RM Layers

4.3 Service Access Points (SAP)

Upper layers use the services of lower layers by communicating through a conceptual interface known as a Service Access Point (SAP). As shown in ***Figure 3***, a SAP is named after the layer whose services it exposes. Thus, Applications access UniPro's services via the T_SAP where the T is shorthand for "Transport Layer".

4.3.1 Service Primitives

SAPs consist of multiple Service Primitives that resemble function calls in software. However, unlike function calls, Service Primitives abstract fully from the employed implementation choices, e.g. hardware versus software. Furthermore, Service Primitive are actually typed messages, with an optional parameter list, that are exchanged between state machines. Thus, unlike function calls, their invocation does not return a value or cause the transmitting state machine to block, waiting for a return value; instead of waiting, a Device might, or might not, get an asynchronous message back, depending on specification details.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

This document uses particular conventions for naming Service Primitives that are intended to highlight the direction in which the message is sent (transmit or receive) and whether the message initiates a transaction or is the consequence of a transaction requested elsewhere. These naming conventions for Service Primitive suffixes are explained in detail in ***Annex C***.

4.3.2 SAP Significance

SAPs defined in this document serve to modularize the Specification in a well-defined manner as shown in ***Figure 3***. SAPs at the top of the UniPro stack have special significance for users because they constitute the only external interfaces to services provided by the UniPro stack. Currently, two external interfaces for Application use are defined, CPorts (T_SAP) and DME_SAP. Additional interfaces for specialized purposes, such as security protocols, might be added in future versions of this Specification.

4.3.3 Data Flow Through the Stack

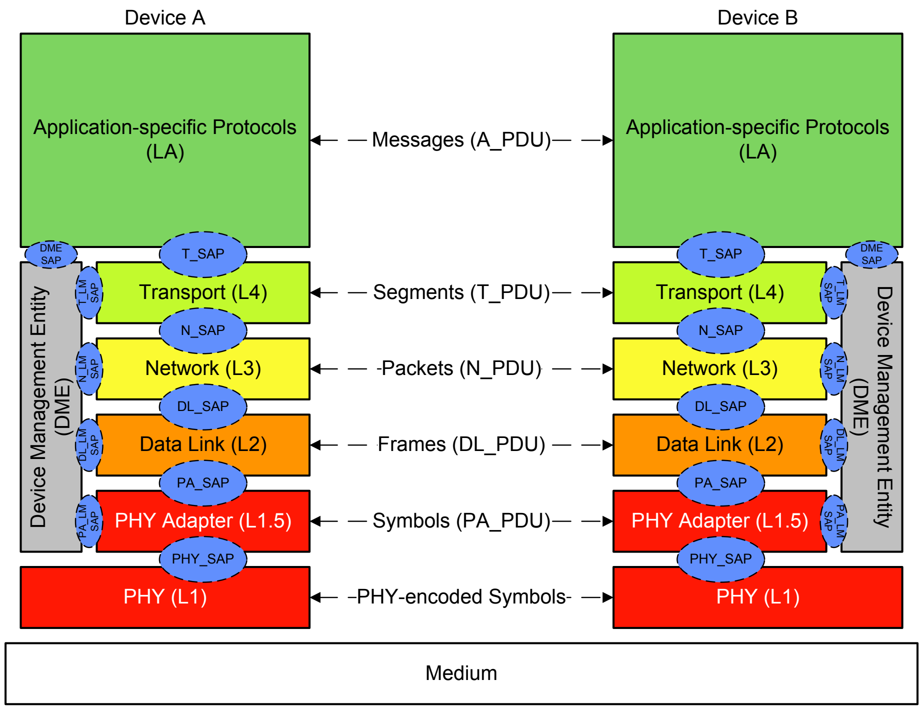


Figure 3 Simplified Model of a Single UniPro Link Connecting Two Devices

If the Application at Device A needs to send data to the Application at Device B, the data will pass via the T_SAP to the Transport Layer (L4), which passes the data via the N_SAP to the Network Layer (L3), which passes the data via the DL_SAP to the Data Link Layer (L2), which passes the data via the PA_SAP to the PHY Adapter Layer (L1.5), which passes the data via an interface belonging to the PHY to the PHY Layer (L1). The PHY communicates to the other PHY via the wiring (“medium”). At Device B, the data passes up the corresponding stack layers in reverse order.

Note that ***[MIPI02]*** defines an optional normative signal-level interface in its ***Annex A*** that corresponds to the PHY_SAP in ***Figure 3***.

4.4 Signal Interfaces

Although key interfaces are defined as SAPs, the external interfaces at the top and bottom of the UniPro stack are also provided as informative signal-level interfaces.

4.4.1 CPort Signal Interface

Annex D provides a description of an informative signal-level interface called the CPort Signal Interface. The interface provides one concrete approach to implementing the more abstract, normative T_SAP interface. The annex illustrates how the data sent and received by multiple CPorts can be multiplexed and how buffer overflow is prevented via a handshake. Applications defined on top of UniPro cannot generally assume that the informative CPort Signal Interface is available in actual UniPro implementations. Application-level specifications should instead reference the normative T_SAP interface.

4.4.2 Interface to the Physical Layer

As explained in the previous section, ***[MIPI02]*** provides signal level-interfaces in annexes defining external interfaces of the Physical Layer technology. An actual implementation is not mandated to provide these interfaces. This leeway allows a combined implementation of a UniPro stack and a Physical Layer to regard the PHY interface as an internal interface that can be optimized without being constrained by the interface described in an annex.

4.4.3 T-MPI Interface for Testing (M-PHY Oriented)

Starting from UniPro version 1.8, the FPGA SERDES M-PHY emulation technology is no longer supported by the Specification.

4.5 Protocol Data Units (PDU)

A list of Service Primitives and a full explanation of their usage would not be enough to define protocol behavior because it only defines what services are provided by a layer. SAPs do not define how this functionality is achieved using lower-level “over-the-medium” behavior. The data units that travel over the media are called Protocol Data Units (PDUs) and are described at multiple levels of abstraction (x_PDU, where “x” designates one of UniPro’s layers).

The Transport Layer, for example, by definition provides the functionality exposed by the T_SAP interface. At a conceptual level, the L4 logic at both ends implements a protocol that provides the services defined by the corresponding T_SAP interface pair. This L4 protocol obviously needs some kind of data exchange. At the L4 level these units of data exchange are known as L4 Protocol Data Units (or T_PDUs). So, part of the L4 specification maps T_SAP primitives to the exchange of T_PDU units between peer Devices.

Note that although the structure and behavior of T_PDUs (“Segments”) are a normative element of the specification of L4 behavior, L4 PDUs are abstract in the sense that they cannot be sent directly over the media or cannot be observed directly by monitoring the media. This is because transmission over the media involves using the services of lower protocol layers. Lower protocol layers may add additional header or trailer information or transform the data in other ways, e.g. symbol encoding in lower layers.

Thus, the UniPro Specification defines how L4 PDUs (“Segments”) are mapped using the L3 SAP to L3 PDUs (“Packets”), how Packets are mapped via the L2 SAP to L2 PDUs (“Frames”), and how Frames are mapped via the L1.5 SAP to L1.5 PDUs (“Symbols”). For M-PHY, the specification provides one more mapping within L1.5 from UniPro's 17-bit PHY-independent Symbols to the symbols passed to and from the PHY. Note that the M-PHY technology provides some form of line encoding and thus internally performs a final mapping.

On the one hand, the UniPro Specification thus makes extensive use of this OSI-based style for defining layered protocol specifications, which stresses rigorously decoupling of the behavior of individual layers. A special characteristic of the UniPro protocol stack is that the transformations between PDUs of the various protocol layers are relatively simple: L4, L3 and L2 merely add a few bytes of header and in one case trailer

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

data. This is because all layers were designed together, and were thus optimized to minimize total stack complexity. Nevertheless, the specification is still strictly described as independent layers coupled by SAPs in order to simplify its readability and its maintenance.

4.6 Physical Layer (L1)

Although Physical Layer (PHY) options for off-chip usage are defined within a separate MIPI Specification, ***[MIPI02]***, this is mainly for architectural reasons. The PHY is a technically critical element for the operation of UniPro and many basic properties of an off-chip UniPro implementation, such as bandwidth and power, are largely determined by the PHY.

4.6.1 M-PHY Technology

In UniPro version 1.8, the use of M-PHY ***[MIPI02]*** is mandated for chip-to-chip interconnections.

In the context of use with UniPro, M-PHY is used in a dual-simplex, high bandwidth serial link that employs a low-swing, differential signaling technique for high speed communication. Unidirectional PHY Links are not supported because various higher protocol layers require return information, e.g. for flow control.

4.6.1.1 Supported M-PHY Capabilities

M-PHY supports different configurations for the forward and reverse directions of the Link. The following configurations for an M-PHY-based Link are supported by UniPro:

- Both directions are assumed to have independent reference clocks (UniPro only supports Type-I M-PHY)
- Both directions can be in HS_MODE, both directions can be in PWM-MODE, or one direction is in HS_MODE and one direction in PWM-MODE
- Both directions may be used with different HS-GEAR or PWM-GEAR settings
- Both directions have to use the same HS RATE Series setting
- Both directions may have a different number of Data Lanes
- Basic Optical Media Converters are supported as optional
- Advanced Optical Media Converters are not supported. UniPro does not mandate the implementation of the M-PHY state LINE-CFG, nor does UniPro make use of the LINE-CFG state, should it be part of the M-PHY implementation.
- M-PHY line termination and drive strength can be controlled

4.6.2 L1 Symbol Encoding

In order to support UniPro, the PHY protocol layer needs to provide symbol encoding for the byte stream. This symbol encoding feature is needed to allow higher protocol layers to use special symbols, e.g. to denote the start of a Packet, that can be distinguished from payload symbols.

4.6.2.1 M-PHY and Symbol Encoding

The symbol encoding technique that UniPro uses for M-PHY is defined in ***Section 4.5*** of ***[MIPI02]***. In this well-known 8b10b coding scheme, every byte is converted to ten bits with a guaranteed maximum of five successive zero or one bits on the wire.

The M-PHY symbol encoding technique of the M-PHY is complemented by an additional symbol scrambling, which is located in L1.5. Data scrambling is used to mitigate the effects of EMR/EMC by spreading the information transmission energy of the Link over a possibly large frequency band using a data randomization technique.

Before being presented as PHY_SAP data symbols to the M-PHY, the L1.5 symbol scrambling process superposes L1.5 PHY data symbols with a pseudo-random bit sequence of a Pseudo-Random-Bit-Sequence generator. This superposition does not destroy information and can be reversed at the receiver side. Applying

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

this technique effectively avoids repetitive high frequency patterns on the Link and thus the emission of unduly high energy in a specific frequency band.

4.6.3 M-PHY Data Rates

[MIPI02] defines two RATE series (A and B) of pre-determined frequencies that are used for high-speed data exchange. The RATE-A series provides three frequencies or Gears defined at 1.25 (HS-G1 or HS-GEAR1), 2.50 (HS-G2), 5.0 (HS-G3), and 10.0 (HS-G4) Gbps. Expressed as a bandwidth that is actually available to the next-higher protocol layer, this respectively results in 1.0, 2.0, 4.0, and 8.0 Gbps due to 8b10b encoding. Frequencies for the RATE-B series are about 17% higher than RATE-A series and are primarily provided for EMI/EMC reasons. UniPro gives the controlling Application the choice of which series to use per Link (same setting for both directions) and which Gear to use per Link direction.

For Low-Speed mode (LS_MODE) data exchange, ***[MIPI02]*** also defines a series of Gears that give control of the trade-off between bandwidth and power. Thus, PWM-G1 support a range of raw bit-rates between 3 and 9 Mbps. This means that a TX may send at rates anywhere within this range, resulting in an effective bit-rate (due to 8b10 encoding) of 2.4 to 7.2 Mbps. For successive gears (PWM-G2 to PWM-G7), these speeds are multiplied by a factor of two for each higher Gear.

Note that for practical reasons, UniPro does not support PWM-G0, but uses for lowest speed transmissions PWM-G1.

The M-PHY's high-speed bandwidth can be increased by utilizing up to four lanes per direction. Using four Data Lanes behaves like a single PHY Lane with a 4x higher bandwidth. The M-PHY PWM-MODE can also run over multiple Lanes.

As UniPro does not mandate the same reference clocks for near end M-PHY and peer side M-PHY, the transmission data rates between sender and receiver may differ slightly. UniPro supports payload data rate reduction at the transmitter by inserting periodically skip symbols. Skip symbols are special M-PHY control symbols that are transmitted for the only purpose to consume Link bandwidth, they are skipped at the receiver without any functional interpretation. Thus, net payload data rate over the Link can be kept below a limit which can be handled by a receiver operating at the lowest possible reference clock frequency.

A UniPro PA Layer implements skip symbol insertion at the PA-TX. A UniPro PA-RX architecture may opt to exploit skip symbol insertion from the peer PA-TX. Skip symbol insertion at the transmitter may be disabled according to requirements of the receiver. A peer UniPro PA-RX may be implemented in a way not requiring skip symbol insertion for the purpose of plesiochronous clock adaptation between local and peer. Even in this case, the peer receiver should be able to receive and discard inserted skip symbols.

The peer PA-RX communicates the architectural need for skip symbol insertion to the local PA-TX during Link Startup capability exchange phase. Despite a peer PA-RX architecture requirement for skip symbol insertion, the local Application may overrule this request and disable PA-TX skip symbol insertion, if it has knowledge of relevant system clocking information. An example for such a case would be the use of a shared reference clock between local and peer or the knowledge of the Application that the local PA-TX always runs at lower symbol frequencies than the peer PA-RX receiver.

4.6.4 M-PHY Power States

The two directions of an M-PHY Link have independent L1 power states. States supported by M-PHY transmitters and receivers can be found in ***[MIPI02]***. The most relevant M-PHY states (and their mapping to Power State names used by UniPro) include the following:

- Unpowered (OFF_STATE); assumes the peer Device's RX is not powered. The Link cannot get out of this Power State on its own because the RX is unpowered.
- HIBERN8 (HIBERNATE_STATE); allows the Link to be put into low-power state for sustained periods of inactivity. The Link can become active again via in-band means.
- PWM-BURST (SLOW_STATE); enables communication in the multi-Mbps range.
- STALL (SLEEP_STATE); for periods of inactivity when operating in FAST_STATE.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

- HS-BURST (FAST_STATE); enables communication in the Gbps speed range.
- SLEEP (SLEEP_STATE); for periods of inactivity when operating in SLOW_STATE.

Note:

*The HIBERNATE_STATE receives special treatment within the UniPro stack compared to all the other Power States. The reasons why are mentioned in **Section 4.8**.*

Note:

UniPro does not allow an Application to put any of the directions of a Link into SLEEP_STATE. SLEEP_STATE is supported by allowing a UniPro stack to autonomously toggle M-PHY between SLEEP_STATE and either FAST_STATE or SLOW_STATE. Thus, if one direction of the Link happens to be in SLEEP_STATE, and some data needs to be sent, the UniPro stack has the capability to go to FAST or SLOW state, thus enabling bidirectional communication. This approach also explains why UniPro can hide the distinction between M-PHY's STALL and SLEEP states from Applications.

Controlling Power State transitions on M-PHY is quite complicated. Sometimes only a simple form of line-state signaling is enough. This could, for instance, involve asserting the lines to a negative differential state (DIF-N) for a longer period of time than can occur during normal operation. In such a case, the transmitter can signal a Power State transition and the attached receiver automatically recognizes the transition. Due to the limited options provided by line state signaling, various other transitions involve higher protocol layers.

Rather than expose this complexity to the Application that controls the UniPro Link, the UniPro stack abstracts many PHY related details.

M-PHY SLEEP and STALL states already require extremely low power consumption from the M-PHY as no data is transmitted. This contradicts with the requirement to be able to restart as quickly as possible the next burst of data transmission. UniPro helps local and peer M-PHY implementations to adjust their power consumption during STALL or SLEEP by a low level protocol signaling scheme at the end of a burst. When the burst ends on a single M-PHY control marker “End-of-Burst” (EoB), the UniPro protocol expects the following M-TX STALL or SLEEP to be of short duration. However, if the burst ends on a transmission of a pair of M-PHY EoB markers, the UniPro protocol expects a longer period of inactivity on the Link and grants an additional recovery time for the peer receiver to wake up from deeper power saving.

4.6.5 L1 Idles

One of the tasks of the PHY Layer is to transmit special L1 Idle symbol sequences on the line when the M_PHY is in PWM_BURST or HS_BURST state and there is no data to send. These Idle symbols are thus inserted in the gaps between Packets and are removed by the corresponding L1 receiver's decoder.

In the case of unscrambled data transmission over the M-PHY, Idle symbols map to pairs of special 10-bit M-PHY-defined FILLER control symbols. The use of pairs of PHY Idle symbols, when used with a UniPro stack, is convenient because it ensures that upper protocol layers can maintain 16-bit alignment on headers, trailers and payload. In the case of scrambled data transmission, an L1 Idle symbol sequence is transmitted as an even number of pseudo-randomly generated M-PHY data symbols marked by a leading MK3 marker symbol and a trailing M-PHY FILLER symbol.

4.7 PHY Adapter Layer (L1.5)

The PHY Adapter Layer is responsible for abstracting the details of the PHY technology, thus providing a PHY-independent interface (PA_SAP) to higher protocol layers.

4.7.1 L1.5 Multi-Lane Support

The PHY Adapter Layer provides bandwidth scalability by supporting up to four PHY Data Lanes per direction. The number of Data Lanes may differ between both directions.

When multiple Data Lanes are available, they are assumed to have identical capabilities. Although the user can opt to use a subset of available Data Lanes, active (used) Lanes are assumed to be in the same state, and L1.5 handles all details of how the Lanes are used autonomously.

Higher protocol layers see multiple Data Lanes simply as an increase in PHY bandwidth. For M-PHY this also applies to the SLOW_STATE PHY bandwidth.

4.7.1.1 L1.5 Multi-Lane Features for M-PHY

An Application can discover, via Attributes, how many Data Lanes are available, and can choose to activate fewer than all available Lanes.

During initialization of L1.5 however, a series of data exchanges are executed that perform the following actions:

- Determine which Data Lanes are attached to the peer Device.
- Determine how the physical Lanes are attached between the local Device and peer Device (topology).
- Assign Logical Data Lane numbering to the attached physical Lanes.

These actions are automatically handled during the Link Startup Sequence. Higher protocol layers are agnostic as to which physical Lanes are being used, or the topology of the wiring.

The discovery process accounts for unattached Lanes, or even Lanes with a hardware failure. It also provides the circuit board designer the freedom to route any TX physical Lane to any available RX physical Lane on the peer Device.

4.7.1.1.1 L1.5 Multi-Lane Deskewing for M-PHY

Streams of M-PHY symbols sent across multiple Data Lanes can have some degree of inter-Lane skew due to differences in trace length or design details. The amount of allowed inter-Lane skew is normatively bounded by L1.5 of the UniPro Specification. The L1.5 protocol has a feature to transmit a special deskew pattern. By sending the deskew pattern on every active Data Lane whenever a burst starts, the RX can detect and compensate for the amount of inter-Lane skew. The deskew pattern is transmitted unscrambled over the Link, but resets and synchronizes the scrambling process at both sides of the link. A protocol tester attached to the Link may detect deskew patterns to synchronize in flight onto a longer scrambled Message burst.

4.7.2 L1.5 Power States and Power Modes

The PHY Adapter Layer abstracts the power states provided by the PHY. Not only are the PHY-specific power states mapped to UniPro-defined power states, e.g. M-PHY's PWM states are mapped to the UniPro power state "SLOW_STATE", but the detailed PHY power management state machines are also abstracted to a simpler model.

Higher layer protocols can get the L1.5 power state, but can only impact it by setting the L1.5 Power Mode. In many cases, there is a one-to-one mapping of power modes to power states. Fast_Mode, Slow_Mode, Hibernate_Mode and Off_Mode correspond to FAST_STATE, SLOW_STATE, HIBERNATE_STATE and OFF_STATE, respectively, and thus to PHY-defined power states.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

However, L1.5 introduces two new power modes called FastAuto_Mode and SlowAuto_Mode. FastAuto_Mode tells L1.5 that the Link should be in FAST_STATE to transport data, but that L1.5 is allowed to put the Link into SLEEP_STATE when there is no data to send. In FastAuto_Mode, the Link behaves more or less like it were in Fast_Mode, but sometimes data arrives a bit later because there is some latency involved in getting from SLEEP_STATE to FAST_STATE. SlowAuto_Mode is the equivalent mechanism for SLOW_STATE.

So the difference between Power States and Power Modes are as follows:

- An Application can set only a Power Mode, but cannot set a Power State
- An Application can get a Power Mode and get a Power State
- The gettable value of a Power Mode simply reflects the value that was set
- The gettable value of a Power State may change spontaneously when in FastAuto_Mode or SlowAuto_Mode

4.7.2.1 L1.5 Controlling Power Modes for M-PHY

The L1.5 for M-PHY automates a significant part of the required steps used for Power Modes control, utilizing an L1.5-to-L1.5 communication protocol known as PACP (PHY Adapter Control Protocol). PDUs ("PACP frames") generated by L1.5 are multiplexed with the symbol stream received from L2 and can be recognized by a unique header pattern.

Providing logic and even a low-level protocol in L1.5 to control PHY state transitions reduces the risk of usage errors by exposing relatively simple interfaces.

The programming model used by the UniPro Specification to support M-PHY involves the following Critical Attributes at the local end (the end closest to the controlling Device) of the Link:

- Number of active Data Lanes per direction
- PWM-GEAR per direction
- HS-GEAR per direction
- HS RATE series (same value used for both directions)
- Line termination per direction
- Power Mode for each direction combined into a single PA_PWRMode Attribute

An identical set of Attributes at the "far" end of the Link is automatically synchronized with the "near" end set whenever PA_PWRMode is written, using the PA_LM_SET.req primitive, at either end of the Link.

A typical use-case is to set these Attributes to their desired values at the near end of the Link, followed by programming PA_PWRMode using the PA_LM_SET.req primitive. Setting the Power Mode then automatically causes PACP to transport all Attribute values, in a single PACP frame, to the Device at the far end of the Link, and thus to program the set of Critical Attributes there. During this interaction, the L1.5 Attributes at both ends of the Link are also automatically copied by L1.5 into the actual PHY Attributes, and the actual power state change, if applicable, takes place.

Note that programming PA_PWRMode, using the PA_LM_SET.req primitive, results in two separate acknowledgements. The first acknowledgement is almost instantaneous and indicates whether or not the request has been accepted (it might return an error code if, for example, a range is exceeded). The second acknowledgement is an asynchronous Notification that typically informs the originator that the Power Mode change request has been completed. This approach was taken because certain Power Modes transitions might take a relatively long time to execute, e.g. to start up and train Phase Locked Loop (PLL) circuitry.

Also note that the Application needs to modify only Lane, Gear and Series Attributes if a change is needed. However, analogous to ***[MIPI02]*** itself, any modifications to these Attributes are only effectuated on a write to PA_PWRMode, using the PA_LM_SET.req primitive.

The power mode change can fail because of a capability mismatch, a severe communication problem, or a concurrency problem. Therefore, the Application should check the returned error-code to determine if a retry is required.

4.7.3 L1.5 and Symbol Encoding

The PHY Adapter Layer also abstracts the symbol encoding technique provided by the PHY. At the PA_SAP interface, L1.5 uses 17-bit symbols. See ***Figure 4***.

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	L1.5 payload															
0	L1.5 payload															

Figure 4 17-bit PHY Adapter Layer Symbol Example

Each 17-bit symbol (the PA_PDU) can hold two bytes of data plus an associated control-or-data flag (Bit 16). This flag allows a UniPro stack to distinguish between special symbols used by the protocol and payload data. Note the color convention whereby Bit 16 is shown in red to correspond to the color convention used for L1.5 (see ***Figure 2***). In subsequent figures, color coding is used to indicate which field belongs to which layer in the UniPro protocol. Gray means "don't care".

Note that the use of 17-bit symbols at the PA_SAP interface is an abstract representation of the data going over the line. The 17-bit L1.5 symbols are thus translated within L1.5 to the symbols, or "code words" supported by the PHY. When Bit 16 equals zero, the two Bytes in the Symbol's L1.5 payload translate to two normal 10-bit (M-PHY) data codes. When Bit 16 equals one, bits [15:8] are used to identify a special control data code, while bits [7:0] directly map to one of the 256 normal data codes.

4.7.4 PACP frames as Used with M-PHY

The PACP protocol allows the L1.5 entity at one end of the Link to communicate with the L1.5 entity at the other end of the Link. This requires the exchange of messages known as PACP frames (see ***Figure 5***).

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	**ESC_PA**								**EscParam_PA=PACP_BEGIN**							
0	**PACP_FunctionID**															
0	**Parameters**															
0	**...**															
0	**CCITT CRC-16**															

Figure 5 PACP Frame Structure

The first 17-bit symbol contains a constant. Because Bit 16 is set to one ('1'), this symbol, at the PHY level, results in a PHY-level control symbol that is used only to signal the start of a PACP frame. The second Symbol indicates the type of PACP frame, and tells the receiving L1.5 how to interpret the remaining fields. Note that all fields are shown in "L1.5 red" because the data is defined, generated and absorbed by L1.5 in the UniPro stack.

PACP frames have very specialized usage (changes in L1.5 Power Modes and their confirmation, automatic exchange of capability information, forcing a form of low-level reset, peer Get/Set and M-PHY testing) and thus are sent infrequently.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

4.7.5 L1.5 Automatic Capability Discovery (M-PHY only)

When an M-PHY based Link is initialized, L1.5 automatically explores the capabilities of both directions of the Link. This results, for example, in L1.5 at both ends of the Link setting two gettable Attributes known as PA_MaxRxPWMGear and PA_MaxRxHSGear. These capability Attributes reside only at the RX end of the Link (they are not duplicated like the previously discussed Critical Attributes) and accurately portray the highest Gear settings in which the Link can operate.

As an example, assume that in one particular direction, the TX can handle PWM-G1 and PWM-G2 and the RX can in itself handle PWM-G1, PWM-G2 and PWM-G3. This fact is automatically discovered by L1.5, which then proceeds to degrade the value of PA_MaxRxPWMGear from an initial value of three (RX capability) down to two (TX capability, the lower of the two values). The resulting value of PA_MaxRxPWMGear may be optionally de-rated further by the Application, e.g. software, if it knows about other limitations such as Basic Optical Media Convertors. A similar discovery process also sets PA_MaxRxHSGear.

Although this Attribute pair is only located in a receiver (thus a total of four Attributes spread over two Devices), a request to put a Link in a specific state is checked for correctness. For example, if an attempt is made to put a Link in the following state:

TX: RATE Series A, Lanes=2, HS-GEAR=2, PWM-GEAR=1, PowerMode=HS

RX: RATE Series A, Lanes=1, HS-GEAR=1, PWM-GEAR=3, PowerMode=PWM

the request is automatically checked and, if the request is impossible to execute, L1.5 returns an error code and leaves the Link in a well-defined, and operable, state.

4.7.6 PHY Testing

The M-PHY ***[MIPI02]*** can be operated in PHY test-mode that differs from the normal operation mode. To be able to define a generic way to test the M-PHY, a controllable state machine, that can generate and send certain patterns is used. The test pattern are encapsulated into PACP frames and analyzed by the receiver side.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

4.8 Data Link Layer (L2)

The main responsibilities of the Data Link Layer are to provide reliable Links between a transmitter and a directly attached receiver and to multiplex and arbitrate multiple types of data traffic, e.g. priorities.

4.8.1 L2 Data Frames

The Data Link Layer clusters the 17-bit PA_PDU symbols into Data Frames. See ***Figure 6***. Every Data Frame consists of a 1-symbol header, a payload of up to 144 symbols and a 2-symbol trailer including a checksum (a 16-bit CRC).

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	ESC_DL								SOF			TC		Reserved		
0	L2 payload															
0	L2 payload															
0	L2 payload															
1	ESC_DL								EOF_EVEN			Frame Seq. Number				
0	CCITT CRC-16															

Figure 6 Example Data Frame with an Even Number of Payload Bytes

Payloads with an odd number of bytes are extended with an extra padding byte at the end of the L2 payload area. The presence of the padding byte is flagged in the Frame's trailer. See ***Figure 7***.

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	ESC_DL								SOF			TC		Reserved		
0	L2 payload															
0	L2 payload															
0	L2 payload								Padding Byte = 0x00							
1	ESC_DL								EOF_ODD			Frame Seq. Number				
0	CCITT CRC-16															

Figure 7 Example Data Frame with an Odd Number of Payload Bytes

4.8.2 L2 Control Frames

Several types of Control Frames are also used to allow L2 to talk to its peer L2, to enable L2 flow control and to handle transmission errors. Unlike Data Frames, Control Frames do not contain Application data. These Control Frames are known as AFC (for Acknowledgment and L2 Flow Control) and NAC (for signaling errors and triggering retransmission) Frames and can be identified using bits [7:5] in the first symbol of the L2 header. See ***Figure 8*** and ***Figure 9***.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	ESC_DL								AFC			TC		CReq	Reserved	
0	Frame Seq. Number					Reserved			Credit Value							
0	CCITT CRC-16															

Figure 8 AFC Frame Structure

AFC Frames are generally encountered as a result of Data Frame transmission: as Data Frames are sent, the receiver of the Data Frames will send AFC Frames back to the peer transmitter to acknowledge correct receipt (the Frame Sequence Number field) and to inform the transmitter about available L2 buffer space (the Credit Value field).

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	ESC_DL								NAC			Reserved				RReq
0	CCITT CRC-16															

Figure 9 NAC Control Frame Structure

NAC Frames are sent back to the transmitter after certain L2 error conditions are detected by the receiver. In contrast, a stream of AFC Frames generally indicates that things are going well.

4.8.3 L2 Retransmission on Errors

Occasional bit errors, e.g. due to thermal noise, or burst errors, e.g. due to EMI, occurring on the PHY are detected at the receiver. Data Frames contain Frame Sequence Numbers that are incremented in each successive Frame. Correctly received Data Frames are acknowledged by sending Acknowledgment Control Frames or AFC (Acknowledgment and Flow Control) Frames. AFC Frames inform the transmitter that all Data Frames up to a particular Frame Sequence Number have been properly received. This allows multiple Data Frames to be acknowledged by a single AFC Frame to improve bandwidth utilization. This feature is referred to as Group Acknowledgement.

A transmitter is also notified by the receiver of CRC errors or certain other errors via Negative Acknowledgment Control (NAC) Frames.

If the transmitter sends Data Frames, but receives neither an AFC nor a NAC Frame within a certain time interval, the transmitter will automatically retransmit, or “replay”, the unacknowledged Data Frames from a transmit-side buffer. Such timeouts can occur because AFC and NAC Frames can be lost or corrupted themselves, but are not covered by an acknowledgement or retransmission request scheme themselves.

4.8.4 L2 Flow Control

One key L2 feature is the provision of link-level flow control. L2 flow control ensures that the transmitter knows how much buffer space is available at L2 of the receiving end of the Link – thus preventing L2 buffer overflow and thus avoiding data loss. A credit-based flow control mechanism is used whereby the receiver sends credit information (in the form of AFC Frames) to update credit information maintained at the transmitter.

4.8.5 L2 Traffic Classes and Priorities

Another feature of the Data Link Layer is the multiplexing of Frames belonging to different Traffic Classes. UniPro currently supports two Traffic Classes called TC0 and TC1. Data Frames sent as TC1 have higher priority than Data Frames sent as TC0. Control Frames such as NAC have an even higher priority.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

4.8.5.1 Frame Preemption

The L2 transmitter can choose to interrupt or “preempt” an ongoing Data Frame transmission in order to send a higher priority Data Frame or Control Frame as soon as possible.

4.8.5.2 Single-TC Devices

Each Traffic Class requires a certain amount of L2 buffering at both the transmit side (for retransmission) and the receive side (for CRC checking), UniPro thus also supports Devices that support only TC0. The presence of such a single-TC Device is automatically detected by its peer Device during Data Link Layer initialization (see ***Section 6.7***).

In this Specification, the upper layers of the UniPro stack are described as if TC0 and TC1 are both present because this is the most general and likely the most common case. Support for the missing TC in higher protocol layers can be optimized away in an implementation.

A Single-TC Device only supports TC0. An Application running on top of a UniPro stack is responsible for using the appropriate TC.

Note that support for a traffic class is on a per-Device basis, and not on a per-direction basis. Therefore, a Device that supports TC1 supports both sending and receiving TC1 data.

4.9 Network Layer (L3)

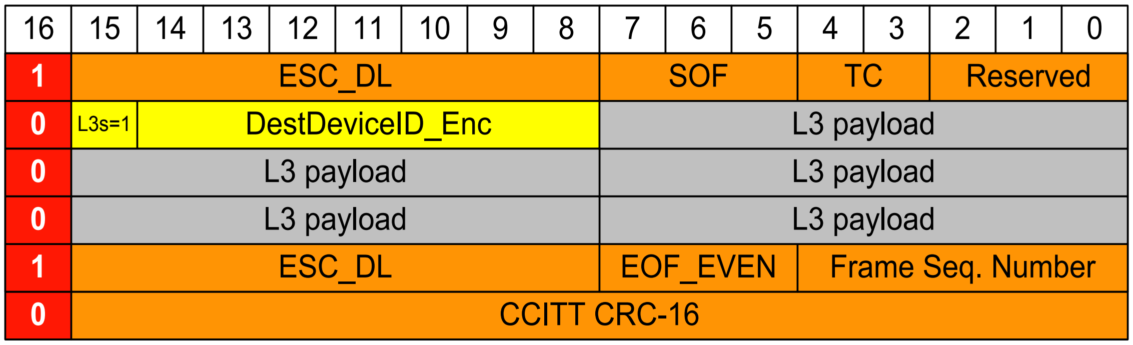
The purpose of the Network Layer is to allow data to be routed to the proper destination in a networked environment. The details of the Network Switches needed for this routing will be defined in a future version of the UniPro Specification.

4.9.1 L3 Packets

The Network Layer introduces a new PDU known as a Packet. When a Packet is passed down to L2, the Packet is encapsulated between an L2 header and an L2 trailer to form a single L2 Frame (see ***Figure 10***). There are two types of Packets, Long Header Packets, which will be defined in a future UniPro Specification, and Short Header Packets.

4.9.1.1 Short Header Packets

Most Packets on a UniPro Network are typically Short Header Packets. These Packets consist of a one byte Short Header followed by L3 payload bytes (see ***Figure 7***). This header byte contains a 7-bit destination address to enable Packet routing.



16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	ESC_DL								SOF			TC		Reserved		
0	L3s=1	DestDeviceID_Enc							L3 payload							
0	L3 payload								L3 payload							
0	L3 payload								L3 payload							
1	ESC_DL								EOF_EVEN			Frame Seq. Number				
0	CCITT CRC-16															

Figure 10 Example Short Header Packet Encapsulated within an L2 Data Frame

4.9.1.2 Long Header Packets

A future UniPro Specification will include support for Long Header Packets (L3s=0). Layer 3 provides a Long Header trap to allow software extensions to support certain future UniPro mechanisms. This Long Header trap feature allows Packets where the L3s=0 to be passed to a software entity (not covered by the

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

UniPro Specification). It also allows software to provide the entire payload of the L2 Data Frame. This feature is intended to allow Devices that support the UniPro v1.8 Specification to handle Packets with long headers via a software change only. This software entity is also known as the Network Layer extension.

4.9.2 L3 Devices and DeviceIDs

UniPro supports Networks of up to 128 Devices. Because Device identifiers (DeviceIDs) need to be unique within a Network they are either assigned at design time or when the Network is initialized.

Note that a Device is not necessarily a physical component: a single integrated circuit can contain multiple individually addressable Devices interconnected by a Network Switch. Similarly a multi-die package may at the package level have a single off-chip UniPro Link, but may be addressable as multiple Devices.

4.10 Transport Layer (L4)

The Transport Layer is the highest UniPro protocol layer involved in the transportation of data. It thus provides the data service interface (T_SAP or “CPorts”) that is used by hardware or software using UniPro. Unlike lower protocol layers, the Transport Layer tends to concentrate on relatively abstract mechanisms. These mechanisms allow a single physical Packet stream between two Devices to support multiple, independent, logical Packet streams or “Connections”.

4.10.1 L4 Connections

The Transport Layer supports multiple bidirectional Connections (T_CO_SAP) between Endpoint Devices. Connections can span a single hop or multiple hops using Switches. A pair of Endpoint Devices can communicate via multiple Connections and, on a Network, an Endpoint can have multiple simultaneous Connections to multiple Devices. Connections can be regarded as virtual private wires between Devices on the shared Network.

UniPro guarantees that data sent over a single Connection arrives in the order in which it was sent. All data sent over a single Connection has the same Traffic Class (TC0 or TC1). The latter is needed to ensure in-order delivery within a Connection.

4.10.2 L4 Segments

The PDUs of the Transport Layer are called Segments. When a Segment is passed down to L3, the Segment is simply prefixed by an L3 header to form a single L3 Packet which is then encapsulated between an L2 header and an L2 trailer to form a single L2 Frame (see ***Figure 11***).

Segments belonging to one Connection are guaranteed to arrive in order; the exact order of Segments received over multiple Connections is undefined. In addition, Segments transmitted over a Connection need to adhere to the same Application-level protocol.

For example, a UniPro-based display protocol may transmit its pixel stream and its command stream using two different Connections, each using a different Application-level protocol and potentially using a different Traffic Class. Segments belonging to both Connections can be distinguished at the destination (the display in this example) based on the DestCPortID_Enc field (see ***Section 4.10.3***).

4.10.2.1 Short Header Segments

A Short Header Segment (see ***Figure 11***, L4s=1) consists of a single-byte L4 header followed by up to 272 bytes of payload. Most Segments are Short Header Segments. A Short Header Segment is typically used to carry L4 payload after an L4 Connection has been established. A Short Header Segment is always shipped inside a Short Header Packet.

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	ESC_DL								SOF			TC		Reserved		
0	L3s=1	DestDeviceID_Enc							L4s=1	DestCPortID_Enc					FCT	EOM
0	L4 Payload								L4 Payload							
0	L4 Payload								L4 Payload							
1	ESC_DL								EOF_EVEN			Frame Seq. Number				
0	CCITT CRC-16															

Figure 11 Example of a Short Header Segment in a Short Header Packet in an L2 Frame

4.10.3 Communicating between L4 CPorts

The interface between the Transport Layer and the Application Layer is structured as a collection of CPorts (Connection-oriented Ports). Each CPort corresponds to a single SAP. CPorts within a Device are identified using integer values between 0 and 2047 (although values above 31 should be used sparingly because they reduce the number of addressable L3 Devices).

CPorts can be considered as sub-addresses within a Device: the Device is identified by an L3 DeviceID while its CPorts, e.g. CPort 0, CPort 3 and CPort 4, are internal sub-addresses that distinguish between the different Connections and thus different data streams to the same Device.

Every Connection connects exactly one CPort in one Device to a CPort in another Device. A Connection can therefore connect CPort 1 of Device 2 to CPort 3 of Device 7. Every CPort can be associated with at most one Connection at any time. In other words, a Device with four CPorts can support up to four simultaneous Connections. CPorts are thus in principle a scarce resource within an Endpoint because every CPort has an independent state which needs to be maintained.

4.10.4 The L4 SAP and Messages

An Application can submit data for transmission to L4 in the form of arbitrary-length data units known as Messages. L4 is responsible for splitting, or “segmenting”, the Messages into Segments.

UniPro thus does not impose a size limit on Messages; a Message can range from a command consisting of only a few bytes to an entire multi-MByte data stream transported over the Connection.

Although the Application data unit is a Message, the T_SAP offers the option to also transfer Message Fragments. This option can be useful when specifying Applications on top of a UniPro stack that need to interrupt the Message creation based on information from the UniPro stack, e.g., incoming Messages, or backpressure. There is no guarantee that the received Message Fragments have the same size as the transmitted Message Fragments or L4 Segments. In an implementation, the granularity at which Message (Fragments) are transmitted may have any arbitrary small granularity, as also illustrated in ***Annex D.*** The signaling of Message boundaries also provides options for resynchronizing the higher protocol layers if needed. This is possible because the End-of-Message flag always coincides with the end of a Segment, the end of a Packet and the end of a Frame. Message-based resynchronization can be relevant in special cases such as when Segments are discarded, or “dropped”, at the receiving Endpoint.

UniPro does not define the size of a data Fragment passed between UniPro Transport Layer (L4) and the Application that uses it. The data Fragment can be the size of a single T_PDU (Data Segment) received by L4 or an assembly of several Data Segments. Fragment size may be decided depending on the needs of UniPro L4 implementation or maybe depending on the needs of the Application.

4.10.5 End-to-End Flow Control in L4

In addition to providing the resources that are needed to support Connections and multiplex their traffic onto a shared Link, L4 also provides an end-to-end flow control (E2E FC) feature. This mechanism ensures that

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

the transmitting CPort never sends more data over a Connection than can be absorbed by the destination CPort's buffer.

Although E2E FC can be disabled, the feature is enabled on reset and disabling it is not generally recommended if the Application protocol using this CPort does not implement a E2E FC mechanism. If E2E FC is turned off, L4 may need to discard Segments, leading to data corruption. This loss of data occurs if the destination CPort does not have enough buffer space to store the incoming stream of Segments. With E2E FC disabled, the source CPort cannot know how much buffer space is free at the destination, and thus risks overflow and, therefore, data loss.

4.10.6 CSD and CSV Mechanisms within a CPort

A CPort is required to immediately absorb any data Segments that it receives from the Network. Failure to do so could lead to filling up of the L2 receive buffers for the associated Traffic Class which could in turn lead to blocking of Segments addressed to other CPorts or even (in a Network) to other Devices. This is undesirable for performance and system reliability reasons (deadlocks).

The solution is a pair of mechanisms within a CPort respectively known as Controlled Segment Dropping (CSD) and CPort Safety Valve (CSV). These ensure that incoming data that cannot be absorbed immediately by the CPort is automatically discarded.

CSD and CSV avoid one source of congestion and improve overall system integrity at the cost of introducing data corruption within the misbehaving Connection. Triggering of the CSD and CSV mechanisms typically indicates a Device malfunction such as a design error or a crashed software Application.

The CSV mechanism can be disabled if so desired.

4.10.7 Test Features

UniPro implementations can be tested using so-called Test Features. These are located at the top of the UniPro stack and, when enabled, behave like simple Applications. A Test Feature consists of a traffic generator that produces a well-defined sequence of bytes (structured as Messages). The Test Feature also contains a matching data analyzer that checks the correctness of the incoming data stream against the expected data traffic.

The built-in Test Features are mainly intended to allow a UniPro stack within a Device to be checked for design- and conformance problems in a well-defined and readily automated way.

The Test Features can also be useful for checking the operation of a prototype or production system as they provide a built-in functional self-test of the UniPro stack, the PHY Layers and associated wiring.

4.11 DME

The Device Management Entity (see ***Figure 3***) controls the layers in the UniPro stack. It provides access to various control parameters in all layers, manages the power mode transitions of the Link, and handles boot-up, hibernate and reset of the entire UniPro stack.

4.11.1 Control Attributes

The layers of the UniPro stack (L1.5, L2, L3 and L4) contain registers or "Attributes" that can be read ("get"), and in many cases written ("set") via the DME. Some Attributes are static and are defined at design time and describe the capabilities of the UniPro interface, e.g. number of CPorts. Others are dynamic and reflect the current (read-only) UniPro stack status, e.g. available space in L4 buffers. Others can be set to control parts of the protocol stack, e.g. to configure the number of active Data Lanes.

Persistent Attributes may be stored and provisioned in a form of Non Volatile Memory, eFuse, etc. Those values can be overwritten via the DME, if desired, using the SET operation with AttrSetType = STATIC. After a value is overwritten it becomes the new persistence value the specific Attribute.

For the Transport Layer, for example, configuration is currently done via a control SAP named T_LM_SAP (“Layer Management”). Attributes have both a symbolic name, such as T_PeerDeviceID, a 16-bit identifier, e.g. 0x4021, and in some cases an extra index or “selector”. The symbolic name is referenced in the text when the Attribute impacts particular protocol behavior. Thus, in the example of T_PeerDeviceID, this Attribute determines the peer Device connected to a given CPort, and is copied into L3 Packet headers. The “selector” index is used to distinguish multiple CPorts as each CPort has its own state Attributes.

Attributes have both a symbolic name, such as PA_ActiveTxDataLanes, and a 16 bit identifier, e.g. 0x1560. In some cases, an additional index or “selector” is required to distinguish multiple CPorts or Test Features because each has its own set of Attributes.

4.11.2 Control Interfaces

Layers 1.5, 2, 3, and 4 are controlled from the DME by “Layer Management” SAPs. Thus, for example, the Transport Layer is configured via an SAP named T_LM_SAP (see ***Figure 3***).

4.12 Backward Compatibility

UniPro v1.8 is a feature upgrade to UniPro v1.61. Every Device architecture representing a UniPro v1.61 implementation will be able to communicate with any architecture representing a UniPro v1.8 implementation based on the feature set that is defined in common between both versions. However, downgrading UniPro v1.8 to communicate with earlier versions of UniPro is not always automated, and may inflict configuration restrictions and/or require application layer intervention.

This version of the UniPro specification gives up backward compatibility to UniPro Versions v1.41.00 and earlier.

4.13 Supported Topologies

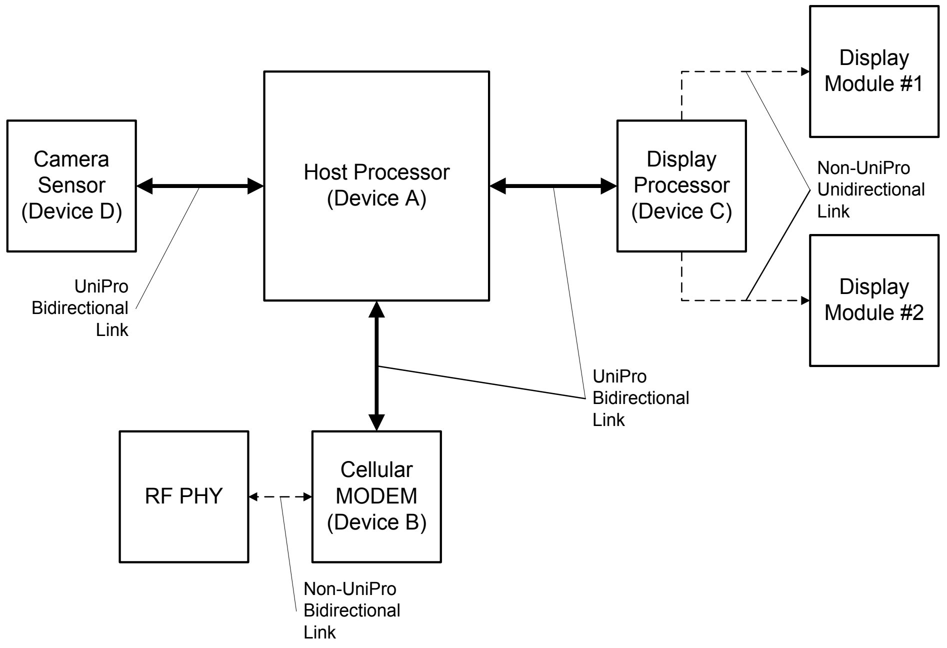
Although Switches for routing in Networks are not supported in this version of UniPro, UniPro is designed to support Networks of Devices.

UniPro Networks are symmetrical in that Endpoints are peer Devices during normal operation. In principle, any Device can take the initiative to start sending data to any other Device on the Network. There is thus no Master/Slave distinction at the UniPro stack level.

4.13.1 Point-to-Point Links

At the physical level, UniPro only supports point-to-point Links between pairs of Devices as these support the required high data rates. As shown in ***Figure 12***, a Device such as a host processor can use a UniPro Link to connect to multiple other Devices. However, the UniPro Links are independent and do not work together to form a Network as, in this example, because there is no provision in UniPro to route data from one Link to another. In the figure, this means that Device D cannot communicate with Device C using only UniPro protocols.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential


Display Module #1
Camera Sensor (Device D)
Host Processor (Device A)
Display Processor (Device C)
Non-UniPro Unidirectional Link
UniPro Bidirectional Link
Display Module #2
UniPro Bidirectional Link
RF PHY
Cellular MODEM (Device B)
Non-UniPro Bidirectional Link

923

Figure 12 Point-to-Point Configuration Example

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

4.13.2 UniPro Networks

For more complex systems, particularly those requiring a true Network, a Switch can be added to route data between the Devices. The Switch is not covered by this version of UniPro, but is relevant to understand the overall architecture.

Networked topologies might be desirable to reduce the cabling in systems where the components are located in physically separate chambers such as in a mobile flip-phone.

In the example shown in ***Figure 13***, two independent Links connect Device A to other Devices in the system. While Device B can only communicate using UniPro protocols with Device A, Device D can communicate directly with Device A, Device C and Device E.

Note that ***Figure 13*** essentially shows two independent Networks. One Network consists of the Link between Devices A and B. The other Network consists of the Links connecting Devices A, C, D, E, and the intermediate Switch. Any communication between these two Networks is outside the scope of this document. A Device on one Network, e.g. Device B, may have the same DeviceID as a Device on the other Network, e.g. Device D.

Figure 13 shows the Switch as a logically separate component for clarity. Although Switches can be implemented as discrete components on separate integrated circuits, they may alternatively be integrated onto the same die as Devices to avoid increasing the component count. Since the PHY Layer is designed for off-chip communication, the connection between a Switch and a Device on the same die might likely avoid using a high-speed serial PHY Layer, and might instead use, for example, a parallel CMOS interface.

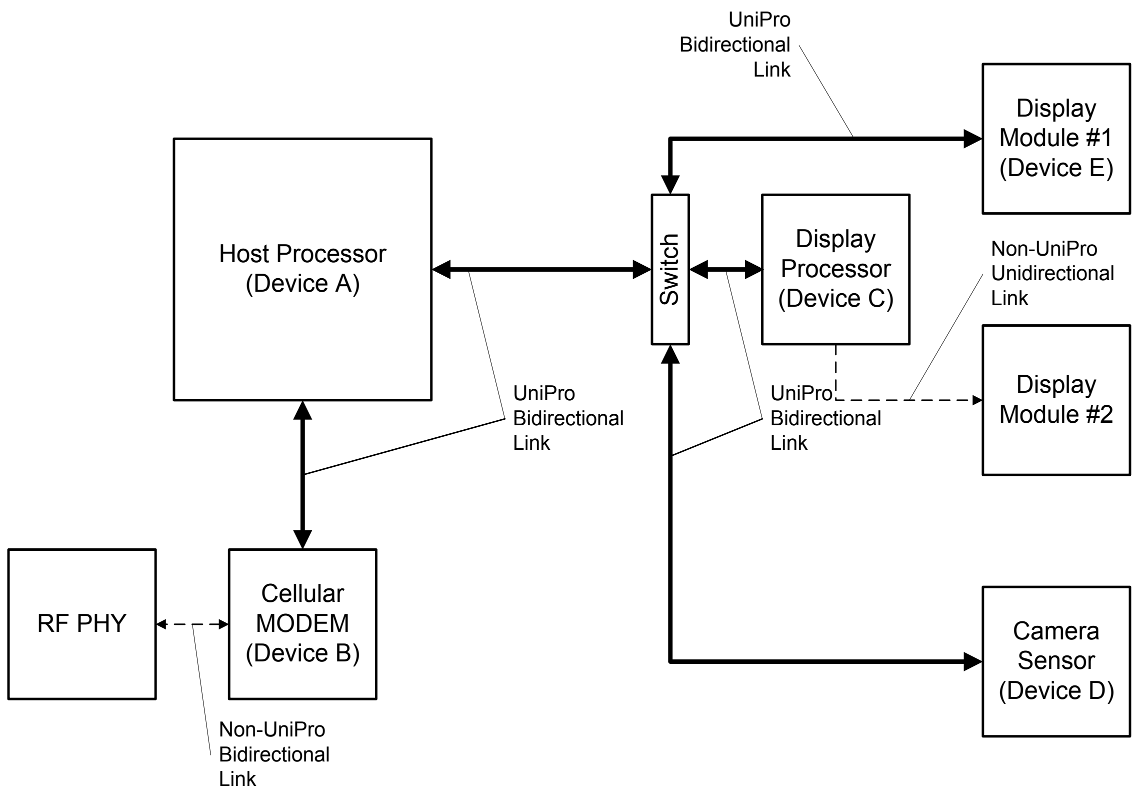

UniPro Bidirectional Link
Display Module #1 (Device E)
Host Processor (Device A)
Switch
Display Processor (Device C)
Non-UniPro Unidirectional Link
UniPro Bidirectional Link
UniPro Bidirectional Link
Display Module #2
RF PHY
Cellular MODEM (Device B)
Camera Sensor (Device D)
Non-UniPro Bidirectional Link

Figure 13 UniPro Network Configuration Example

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5 PHY Adapter Layer (L1.5)

The PHY Adapter (PA) Layer is an intermediate layer between L1 and L2 (see ***Figure 3***).

Figure 14 shows the SAP model used by the PHY Adapter Layer. Note that this figure only shows PHY Data Lanes; potential PHY Clock Lanes are not shown. The PA Layer service to the Data Link Layer is provided by means of the PHY Adapter Service Access Point (PA_SAP). The PA Layer in turn relies on the service provided by the PHY Service Access Points (PHY_SAPs). The PHY Adapter Layer Management SAP (PA_LM_SAP) is provided to the Device Management Entity (DME) for configuration and control purposes.

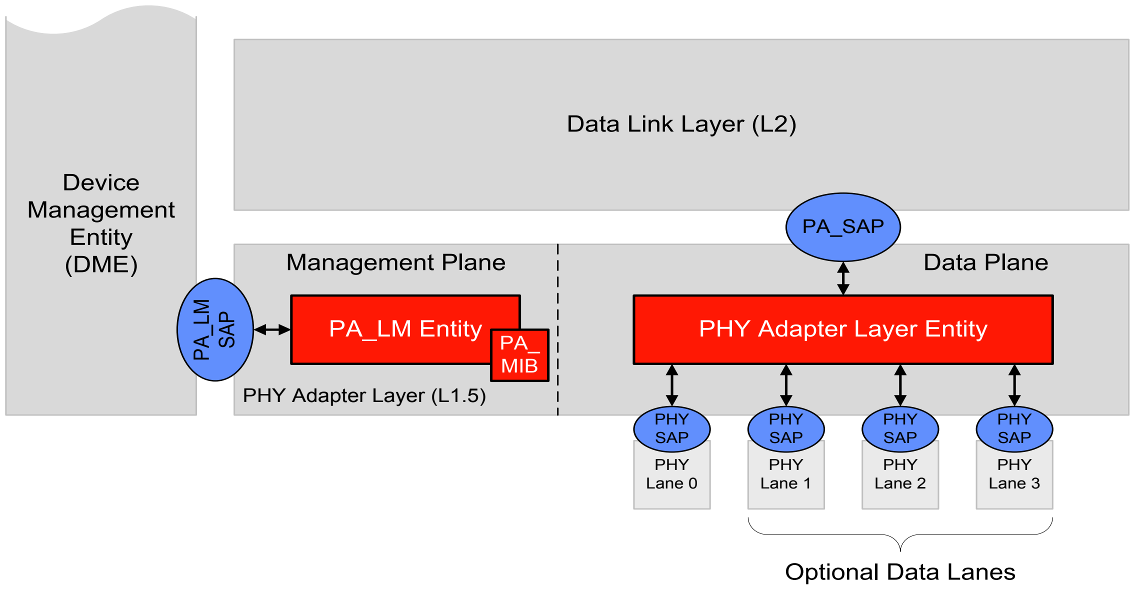


Figure 14 PHY Adapter Layer SAP Model

The PHY Adapter ensures that the UniPro protocol is independent of the PHY technology. To do so, it hides internal implementation details of the PHY from the Data Link Layer. The PHY Adapter Layer for M-PHY is called PA M-PHY.

For off-chip communication, a UniPro implementation shall use a physical layer compliant with ***[MIPI02]***. Future versions of the UniPro Specification may support and allow alternative physical layers for off-chip usage. Note that an on-chip implementation of UniPro is not restricted to any specific set of physical layer technologies.

5.1 PHY Adapter Layer Service Functionality and Features

The PA Layer service defined in this Specification provides:

- Transmission and reception of Data Link Layer control symbols and data symbols via underlying PHY
- Lane distribution and merging in multi-Lane ports
- Provision of UniPro power management operating modes
- (Re-)Initialization of the PHY TX path
- Access to all PHY Attributes of all available Lanes
- Mandatory implementation of skip symbol insertion at PA-TX
- Optional exploitation in a PA-RX of skip symbol insertion at the peer PA-TX

These services are offered independently from the PHY technology. Just a basic set of features is required from the PHY, as described in ***Section 5.2***.

5.2 Features Assumed from the PHY Layer

A PA Layer is associated with each PHY entity via its PHY_SAP. The PHY_SAP is defined by the relevant MIPI PHY Specification.

The PA Layer requires the following features provided by the PHY:

- Transmission and reception of encoded PHY symbols, including access to PHY escape symbols
- Transmission of PHY IDLE symbol sequences when no data is supplied.
- Indication that PHY IDLE symbol sequences were received.
- A method to re-initialize the forward Link to overcome error situations.
- Provision of different Power Modes and a method to signal them from transmitter to receiver.

When a PHY provides these basic features, the PHY Adapter will be able to implement the required PA Layer services for the Data Link Layer.

5.3 PA_SAP

The PA_SAP offers three groups of Service Primitives: Data Transfer primitives, Control primitives and Status primitives.

The Data Transfer primitives enable data transmission and reception, while the Control primitives are needed for (re-)initialization of the Link. Status primitives are used to indicate PA Layer status information, such as identified errors, to the Data Link Layer.

The primitives on this SAP are used by the Data Link (DL) Layer.

5.3.1 Data Transfer Primitives

The primitives covered in this section are listed in ***Table 1***.

Table 1 PA_SAP Data Transfer Primitives

Name	Request	Indication	Local Response	Response	Local Confirm	Confirm
PA_DATA	***5.3.1.1***	***5.3.1.3***	***5.3.1.4***	–	***5.3.1.2***	–
PA_ESCDATA	***5.3.1.5***	***5.3.1.7***	***5.3.1.8***	–	***5.3.1.6***	–

The parameters used for these primitives are defined in ***Table 2***.

Table 2 PA_SAP Data Transfer Primitive Parameters

Name	Type	Valid Range	Description
Data	Integer	0 to 65535	Normal payload data
EscType	Enumeration	ESC_DL	Escaped data type (See ***Table 39***).
EscParam	Integer	0 to 255	Escaped payload data

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.3.1.1 PA_DATA.req

This primitive requests the transmission of payload data.

The semantics of this primitive are:

PA_DATA.req(Data)

The primitive parameter is defined in ***Table 2***.

When Generated

The DL Layer shall generate a PA_DATA.req primitive to transmit a DL Layer data symbol over the Link.

Effect on Receipt

The PA Layer shall transfer the payload over the Link via the underlying PHY.

5.3.1.2 PA_DATA.cnf_L

This primitive informs the PA Service User that the Service Provider, L1.5 in this case, is ready to accept a new PA_DATA.req or PA_ESCDATA.req primitive.

The semantics of this primitive are:

PA_DATA.cnf_L( )

When Generated

This primitive shall be generated by the PA Service Provider after the receipt of a PA_DATA.req primitive, when the PA Service Provider can accept a new request to transfer a new DL Layer symbol.

Effect on Receipt

Following the emission of a PA_DATA.req primitive and prior to the reception of a PA_DATA.cnf_L primitive, the PA Service User shall not emit a new PA_DATA.req or PA_ESCDATA.req primitive.

The PA Service User may emit a new PA_DATA.req primitive immediately following a reset or after the reception of the PA_DATA.cnf_L or PA_ESCDATA.cnf_L primitive corresponding to a previously emitted PA_DATA.req or PA_ESCDATA.req primitive, respectively.

5.3.1.3 PA_DATA.ind

This primitive reports the reception of a DL data symbol over the Link.

The semantics of this primitive are:

PA_DATA.ind(Data)

The primitive parameter is defined in ***Table 2***.

When Generated

This primitive shall be generated by the PA Layer when a Data PA_PDU is received over the Link.

Effect on Receipt

When the DL Layer receives this primitive, it shall consume the Data PA_PDU.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.3.1.4 PA_DATA.rsp_L

This primitive informs the Service Provider that the PA Service User, L2 in this case, is ready to accept a new PA_DATA.ind or PA_ESCDATA.ind primitive.

The semantics of this primitive are:

PA_DATA.rsp_L( )

When Generated

This primitive shall be generated by the PA Service User after the receipt of a PA_DATA.ind primitive, when the PA Service User can accept a new PA_DATA.ind or PA_ESCDATA.ind primitive.

Effect on Receipt

Following the emission of a PA_DATA.ind primitive and prior to the reception of a PA_DATA.rsp_L primitive, the PHY Adapter Layer shall not emit a new PA_DATA.ind or PA_ESCDATA.ind primitive. The PHY Adapter Layer may emit a new PA_DATA.ind primitive only just after reset, or after reception of the PA_DATA.rsp_L or PA_ESCDATA.rsp_L primitive corresponding to a previously emitted PA_DATA.ind or PA_ESCDATA.ind primitive, respectively.

5.3.1.5 PA_ESCDATA.req

This primitive requests the transmission of escaped payload data.

The semantics of this primitive are:

PA_ESCDATA.req(EscType, EscParam)

The primitive parameters are defined in ***Table 2***.

When Generated

The DL Layer shall generate a PA_ESCDATA.req primitive to transmit an escaped DL Layer data symbol over the Link. The EscType parameter defines the type of escaped data. The corresponding escaped payload data is given with the EscParam parameter.

Effect on Receipt

The PA Layer shall transfer the EscType and EscParam information over the Link via the underlying PHY.

5.3.1.6 PA_ESCDATA.cnf_L

This primitive informs the PA Service User that the Service Provider, L1.5 in this case, is ready to accept a new PA_DATA.req or PA_ESCDATA.req primitive.

The semantics of this primitive are:

PA_ESCDATA.cnf_L( )

When Generated

This primitive shall be generated by the PA Service Provider after the receipt of a PA_ESCDATA.req primitive, when the PA Service Provider can accept a new request to transfer a new DL Layer symbol.

Effect on Receipt

Following the emission of a PA_ESCDATA.req primitive and prior to the reception of a PA_ESCDATA.cnf_L primitive, the PA Service User shall not emit a new PA_DATA.req or PA_ESCDATA.req primitive.

The PA Service User may emit a new PA_ESCDATA.req primitive immediately following a reset or after the reception of the PA_DATA.cnf_L or PA_ESCDATA.cnf_L primitive corresponding to a previously emitted PA_DATA.req or PA_ESCDATA.req primitive, respectively.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.3.1.7 PA_ESCDATA.ind

This primitive reports the reception of a DL Control symbol.

The semantics of this primitive are:

PA_ESCDATA.ind(EscType, EscParam)

The primitive parameters are defined in ***Table 2***.

When Generated

This primitive shall be generated by the PA Layer when an Escaped Data PA_PDU is received over the Link via the underlying PHY and the EscType field of the PA_PDU is equal to ESC_DL.

Effect on Receipt

When the DL Layer receives this primitive, it shall consume the EscType and EscParam information.

5.3.1.8 PA_ESCDATA.rsp_L

This primitive informs the Service Provider that the PA Service User, L2 in this case, is ready to accept a new PA_DATA.ind or PA_ESCDATA.ind primitive.

The semantics of this primitive are:

PA_ESCDATA.rsp_L( )

When Generated

This primitive shall be generated by the PA Service User after the receipt of a PA_ESCDATA.ind primitive, when the PA Service User can accept a new indication PA_DATA.ind or PA_ESCDATA.ind.

Effect on Receipt

Following the emission of a PA_ESCDATA.ind primitive and prior to the reception of a PA_ESCDATA.rsp_L primitive, the PHY Adapter Layer shall not emit a new PA_DATA.ind or PA_ESCDATA.ind primitive. The PHY Adapter Layer may emit a new PA_DATA.ind primitive only just after reset, or after reception of the PA_DATA.rsp_L or PA_ESCDATA.rsp_L primitive corresponding to a previously emitted PA_DATA.ind or PA_ESCDATA.ind primitive, respectively.

5.3.2 Control Primitives

The Control primitives of the PA_SAP are used by the DL Layer to control the PHY Adapter Layer.

The INIT primitive is represented as request with associated confirmation. This primitive is prefixed by "PA" for the PA_SAP. The primitive is summarized in ***Table 3***.

Table 3 PA_SAP Control Primitives

Name	Request	Indication	Local Response	Response	Local Confirm	Confirm
PA_INIT	*5.3.2.1*	*5.3.2.8*	–	–	*5.3.2.2*	–
PA_DL_PAUSE	–	*5.3.2.3*	*5.3.2.4*	–	–	–
PA_DL_RESUME	–	*5.3.2.5*	–	–	–	–
PA_LANE_ALIGN	*5.3.2.6*	–	–	–	*5.3.2.7*	–

The parameters used for these primitives are defined in ***Table 4***.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Table 4 PA_SAP Control Primitive Parameters

Name	Type	Valid Range	Value	Description
PAResult	Boolean	FALSE	0	Result of the operation
		TRUE	1	

5.3.2.1 PA_INIT.req

This primitive requests to (re-)initialize the PA Layer and underlying PHY. For M-PHY, both transmit and receive paths are (re-)initialized. For a description of the (re-)initialization process on M-PHY see ***Section 5.7.10***.

The semantics of this primitive are:

PA_INIT.req( )

When Generated

The DL Layer generates this request to (re-)initialize the PA Layer and underlying PHY. Refer to ***Section 6*** for details.

Effect on Receipt

The PA Layer (re-)initializes its transmit path. Further, it performs the necessary actions to (re-)initialize the underlying PHY transmit path. In the case of M-PHY, the PA Layer and PHY receive paths are also (re-)initialized.

Following the emission of a PA_INIT.req primitive, and prior to the reception of a PA_INIT.cnf_L primitive, the PA Service User shall not emit a new PA_INIT.req primitive.

5.3.2.2 PA_INIT.cnf_L

This primitive reports the completion of the (re-)initialization procedure.

The semantics of this primitive are:

PA_INIT.cnf_L(PAResult, PAReverseLinkInitialized)

When Generated

This primitive shall be generated by the PA Layer as response to a PA_INIT.req, when it has performed the (re-)initialization procedure. PAResult shall be set to TRUE if the operation completed successfully, otherwise PAResult shall be set to FALSE.

For an M-PHY-based UniPro stack, the initialization procedure is only successful, if both, forward Link and reverse Link, are initialized. Therefore, PAReverseLinkInitialized shall always be set to TRUE. Setting PAReverseLinkInitialized to FALSE is reserved for potential future PHY technologies that do not support synchronized Line-Reset between local TX and peer RX, which may require bilateral synchronized PA Layer operation for re-initialization.

Effect on Receipt

The DL Layer is informed about the finalization of the (re-)initialization procedure.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.3.2.3 PA_DL_PAUSE.ind

This primitive informs the PA Service User, the DL Layer in this case, that the PA Layer was requested to execute a operation that requires the usage of the Link, e.g. Power Mode change or PACP frame transmission. This primitive is one in the group of primitives defining the handshake procedure used between PA Layer and DL Layer to coordinate the Link usage. After generating this primitive, the PA Layer shall wait for the reception of the PA_DL_PAUSE.rsp_L before taking any further action.

The semantics of this primitive are:

PA_DL_PAUSE.ind( )

When Generated

This primitive shall be generated by the PA Layer when an operation requiring PACP transmission is requested and shall precede any other actions necessary to execute the operation.

Effect on Receipt

When the DL Layer receives this primitive, it shall take the necessary action to reach a state where the Link is temporarily not used (see ***Section 6.6.5***).

5.3.2.4 PA_DL_PAUSE.rsp_L

This primitive informs the Service Provider that the PA Service User, the DL Layer in this case, has reached a state where the Link may be used by the PA Layer.

The semantics of this primitive are:

PA_DL_PAUSE.rsp_L( )

When Generated

This primitive shall be generated by the DL Layer after receipt of a PA_DL_PAUSE.ind and only when the DL Layer has reached a state where the PA Layer can use the Link (see ***Section 6.6.5***).

Effect on Receipt

When the PA Layer receives this primitive, it shall continue the execution of the requested operation.

5.3.2.5 PA_DL_RESUME.ind

This primitive informs the PA Service User, the DL Layer in this case, that the PA Layer has completed its operation and the DL Layer may continue to use the Link.

The semantics of this primitive are:

PA_DL_RESUME.ind(PAResult)

When Generated

This primitive shall be generated by the PA Layer when it has completed its operation. The parameter is set to TRUE if the operation completed successfully, otherwise the parameter is set to FALSE.

Effect on Receipt

When the DL Layer receives this primitive, it shall resume normal operation knowing that the UniPro Link is once more in a state where Frames can be sent (see ***Section 6.6.5***).

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.3.2.6 PA_LANE_ALIGN.req

The DL Layer generates this request to force the PA Layer to execute a Lane Synchronization, which results in the PA Layer sending a deskew pattern as described in ***Section 5.7.11***.

The semantics of this primitive are:

PA_LANE_ALIGN.req( )

When Generated

The DL Layer shall generate a PA_LANE_ALIGN.req primitive to ensure that all Lanes are synchronized (deskewed) and shall not generate a PA_ESCDATA.req or PA_DATA.req primitive before completion of the Lane Synchronization.

Effect on Receipt

PA M-PHY shall execute Lane Synchronization as defined in ***Section 5.7.11***.

5.3.2.7 PA_LANE_ALIGN.cnf_L

This primitive informs the PA Service User, the DL Layer in this case, that the Service Provider, the PA Layer in this case, has completed the Lane Synchronization previously requested by the primitive PA_LANE_ALIGN.req.

The semantics of this primitive are:

PA_LANE_ALIGN.cnf_L( )

When Generated

This primitive shall be generated by the PA Service Provider after receipt of a PA_LANE_ALIGN.req primitive, after the Lane Synchronization has completed.

Effect on Receipt

The DL Layer shall be allowed again to generate a PA_ESCDATA.req or PA_DATA.req primitive.

5.3.2.8 PA_INIT.ind

This primitive informs the PA Service user, in this case the DL Layer, about the processing of a PA_INIT requested by the peer PA Layer.

The semantics of this primitive are:

PA_INIT.ind( )

When Generated

This primitive shall be generated by the PA Layer when it receives a PACP_PWR_req frame with the UserDataValid Flag set to b0 and both parameters, TxMode and RxMode, unequal to b11 (Enter Hibernate).

Effect on Receipt

When the DL Layer receives this indication, it shall notify the DME accordingly.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.3.3 Status Primitives

The ERROR primitive is represented as an indication and is prefixed by ,"PA" for the PA_SAP. The primitive is summarized in ***Table 5***.

Table 5 PA_SAP Status Primitive

Name	Request	Indication	Local Response	Response	Local Confirm	Confirm
PA_ERROR	–	***5.3.3.1***	–	–	–	–

The parameter used for this primitive is defined in ***Table 6***.

Table 6 PA_SAP Status Primitive Parameters

Name	Type	Valid Range	Value	Description
PAErrorCode	Enumeration	BAD_PHY_SYMBOL	1	Error types as identified by the PA Layer and flagged to the L2.
		UNMAPPED_PHY_ESC_SYMBOL	2	
		UNEXPECTED_PHY_ESC_SYMBOL	3	
		BAD_PA_PARAM	4	

5.3.3.1 PA_ERROR.ind

This primitive reports errors identified by the PA Layer to the Data Link Layer.

The semantics of this primitive are:

PA_ERROR.ind(PAErrorCode)

The primitive parameter is defined in ***Table 6***.

When Generated

This primitive shall be generated by the PA Layer when an error is detected in the incoming stream of symbols.

With M-PHY, PAErrorCode identifies the detected error as follows:

- BAD_PHY_SYMBOL: a coding error is detected by the PHY (see ***Section 5.7.5***)
- UNMAPPED_PHY_ESC_SYMBOL: a reserved symbol is detected by the PHY (see ***Section 5.7.5)***
- UNEXPECTED_PHY_ESC_SYMBOL: a Marker or a FILLER is received for PA_PDU[7:0] (see ***Section 5.7.5)***
- BAD_PA_PARAM: ESC_PA with EscParam_PA other than PACP_BEGIN, TRG_UPR0, TRG_UPR1, TRG_UPR2 (see ***Section 5.7.5)***

The PA_ERROR.ind primitive is sent when a PA symbol (formed out of two M-PHY symbols) has an error that the PA Layer can detect.

Note:

Two consecutive M-PHY symbol errors might cause one or two PA_ERROR.ind primitives, depending on their position relative to the PA symbol.

Effect on Receipt

Refer to the Data Link Layer definitions for details.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.4 PA_LM_SAP

The PHY Adapter Layer Management (PA_LM) SAP offers three groups of Service Primitives: Configuration primitives, Control primitives and Status primitives. The primitives on the PA_LM_SAP are used by the Device Management Entity (DME) to configure and control the layer and receive layer status information.

The Configuration primitives enable access to configuration information specific to the PA Layer. This information is represented as a PHY Adapter Layer-specific Management Information Base (PA_MIB). The PA_MIB is regarded as "contained" within the PA_LM entity. Multiple accesses to the PA_MIB via the Configuration primitives shall not occur concurrently. The available configuration Attributes are defined in ***Section 5.8.***

The Control primitives provide direct control of the PA Layer. Control primitives are generated by the DME and can occur concurrently.

The Status primitives indicate status information of the PA Layer. Status primitives are generated by the PA Layer and can occur concurrently.

5.4.1 Configuration Primitives

The PA_LM configuration primitives, GET and SET, are used by the DME to retrieve and store values, respectively, for the configuration Attributes in the PA_MIB.

The GET and SET primitives are represented as requests with associated confirm primitives. These primitives are prefixed by PA_LM. The primitives are summarized in ***Table 7***.

Table 7 PA_LM Configuration Primitives

Name	Request	Indication	Local Response	Response	Local Confirm	Confirm
PA_LM_GET	***5.4.1.1***	–	–	–	***5.4.1.2***	–
PA_LM_SET	***5.4.1.3***	–	–	–	***5.4.1.4***	–
PA_LM_PWR_MODE	–	***5.4.1.5***	***5.4.1.6***	–	–	–
PA_LM_PWR_MODE_ CHANGED	–	***5.4.1.7***	–	–	–	–
PA_LM_PEER_GET	***5.4.1.12***	***5.4.1.8***	–	***5.4.1.9***	–	***5.4.1.13***
PA_LM_PEER_SET	***5.4.1.14***	***5.4.1.10***		***5.4.1.11***		***5.4.1.15***

The parameters used for these primitives are defined in ***Table 8***.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Table 8 Configuration Primitive Parameters

Name	Type	Valid Range	Value	Description
MIBattribute	Integer	Any MIB Attribute as defined in ***Section 5.8*** (range 0x1000 to 0x1FFF), or a PHY Attribute (range 0x0000-0x0FFF)	–	The Attribute identifier of the MIB Attribute
SelectorIndex	Integer	0 to 2*PA_MaxDataLanes-1	–	Indicates the targeted M-PHY lane when relevant
GenMIBattribute	Integer	Any MIB Attribute	–	The Attribute identifier of the MIB Attribute
GenSelectorIndex	Integer	M-PHY: 0 to 2*PA_MaxDataLanes - 1 CPort: 0 to T_NumCPorts - 1 Test Feature: 0 to T_NumTestFeatures - 1	–	Indicates the targeted M-PHY Lane or CPort Test Feature when relevant
MIBvalue	AttrValueType	As defined in ***Section 5.8***	–	The value of the MIB Attribute
AttrSetType	Enum	NORMAL	0	Select whether the actual value (NORMAL) or the Attribute's non-volatile value (STATIC) is addressed
		STATIC	1	
ConfigResultCode	Enum	SUCCESS	0	Indicates result of the request
		INVALID_MIB_ATTRIBUTE	1	
		INVALID_MIB_ATTRIBUTE_VALUE	2	
		READ_ONLY_MIB_ATTRIBUTE	3	
		WRITE_ONLY_MIB_ATTRIBUTE	4	
		BAD_INDEX	5	
		LOCKED_MIB_ATTRIBUTE	6	
		PEER_COMMUNICATION_FAILURE	8	
		BUSY	9	
PAPowerModeUserData	Data	–	–	24-byte data payload
PowerChangeResultCode	Enum	–	–	Result code of a Power Mode change operation (see ***Table 9***)
GenericErrorCode	Enum	SUCCESS	0	Result of the operation
		FAILURE	1	

Table 9 PowerChangeResultCode Values

PowerChangeResultCode	Value	Condition
PWR_OK	0	The request was accepted
PWR_LOCAL	1	The local request was successfully applied
PWR_REMOTE	2	The peer request was successfully applied
PWR_BUSY	3	The request was aborted due to concurrent requests
PWR_ERROR_CAP	4	The request was rejected because the requested configuration exceeded the Link's capabilities
PWR_FATAL_ERROR	5	The request was aborted due to a communication problem. The Link may be inoperable

5.4.1.1 PA_LM_GET.req

This primitive requests information about a given PHY Attribute or the PA_MIB Attribute identified by MIBattribute. The SelectorIndex shall be interpreted as a data PHY Lane index for the targeted PHY Attribute. The SelectorIndex shall be ignored for the targeted PA_MIB Attribute.

The semantics of this primitive are:

PA_LM_GET.req(MIBattribute, SelectorIndex)

The primitive's parameters are defined in ***Table 8***.

When Generated

This primitive is generated by the DME to obtain information from the PA_MIB or the underlying PHY.

Effect on Receipt

The PA_LM entity attempts to retrieve the requested MIB Attribute value from PA_MIB or the underlying PHY and responds with PA_LM_GET.cnf_L that gives the result.

5.4.1.2 PA_LM_GET.cnf_L

This primitive reports the result of an information request about a given PA_MIB Attribute or a PHY Attribute.

The semantics of this primitive are:

PA_LM_GET.cnf_L(ConfigResultCode, MIBvalue)

The primitive's parameters are defined in ***Table 8***.

When Generated

The PA Layer shall generate the PA_LM_GET.cnf_L primitive in response to a PA_LM_GET.req from the DME.

Effect on Receipt

The DME is informed about the result of the operation, and in case ConfigResultCode is SUCCESS, MIBvalue carries the MIB Attribute value. For any other value of ConfigResultCode, MIBvalue is undefined.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.4.1.3 PA_LM_SET.req

This primitive attempts to set the PHY Attribute or a PA_MIB Attribute indicated by MIBattribute, to the MIBvalue value. The SelectorIndex shall be interpreted as a data PHY Lane index for the targeted PHY Attribute. The SelectorIndex shall be ignored for the targeted PA_MIB Attribute. The AttrSetType parameter is applicable for PA_MIB Attributes and ignored for PHY Attributes.

The semantics of this primitive are:

PA_LM_SET.req(AttrSetType, MIBattribute, MIBvalue, SelectorIndex)

The primitive's parameters are defined in ***Table 8***.

When Generated

This primitive is generated by the DME to set the indicated PA_MIB Attribute.

Effect on Receipt

The PA_LM attempts to set the specified MIB Attribute in its database or in the PHY. If the MIB Attribute implies a specific action, then this primitive requests that the action be performed. For example, setting PA_PWRMode triggers the Link Configuration procedure (see ***Section 5.7.12)***.The PA_LM responds with PA_LM_SET.cnf_L that gives the result.

5.4.1.4 PA_LM_SET.cnf_L

This primitive reports the results of an attempt to set the value of an Attribute in the PA_MIB or in the underlying PHYs.

The semantics of this primitive are:

PA_LM_SET.cnf_L(ConfigResultCode)

The primitive's parameter is defined in ***Table 8***.

When Generated

The PA Layer shall generate the PA_LM_SET.cnf_L primitive in response to a PA_LM_SET.req from the DME. ConfigResultCode shall be set to INVALID_MIB_ATTRIBUTE if AttrSetType of the request was STATIC and the Attribute indicated by MIBattribute and SelectorIndex does not support setting its reset value.

Effect on Receipt

The PA_LM_SET.cnf_L confirms the success or failure of the PA_LM_SET.req at the PA Layer, and should have no further effects at the DME.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.4.1.5 PA_LM_PWR_MODE.ind

This primitive passes the PAPowerModeUserData that was received from the peer Device during the Link Configuration procedure.

The semantics of this primitive are:

PA_LM_PWR_MODE.ind(PAResult, PAPowerModeUserData)

The first parameter is set to TRUE, when the Power Mode change request was created by the local DME, otherwise it is set to FALSE.

The second parameter is a 24-byte long data payload containing private data from the peer DME.

When Generated

See ***Section 5.7.12.***

Effect on Receipt

DME shall respond with the PA_LM_PWR_MODE.rsp_L primitive.

5.4.1.6 PA_LM_PWR_MODE.rsp_L

This primitive acknowledges PA_LM_PWR_MODE.ind. The PAPowerModeUserData is to be sent to the peer Device if the Power Mode change request was created by the peer DME.

The semantics of this primitive are:

PA_LM_PWR_MODE.rsp_L(PAPowerModeUserData)

The parameter is defined in ***Table 8***.

When Generated

The DME shall generate the PA_LM_PWR_MODE.rsp_L primitive in response to a PA_LM_PWR_MODE.ind from the PA Layer.

Effect on Receipt

The PA Layer continues the on-going Power Mode change request. If the Power Mode change was requested by the peer Device, then the contents of PAPowerModeUserData is to be transmitted to the peer Device via a PACP_PWR_cnf frame. If not, then the PA Layer begins to finish the Power Mode change by closing down the burst.

See ***Section 5.7.12*** for more information.

5.4.1.7 PA_LM_PWR_MODE_CHANGED.ind

This primitive reports the results of the Link Configuration procedure.

The semantics of this primitive are:

PA_LM_PWR_MODE_CHANGED.ind(PowerChangeResultCode)

The primitive's parameter is defined in ***Table 9***.

When Generated

See ***Section 5.7.12.***

Effect on Receipt

The DME is notified that the local request or a peer request has been completed with the status reported by the parameter.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.4.1.8 PA_LM_PEER_GET.ind

This primitive indicates the value of the Attribute identified by GenMIBattribute, and, if relevant, the GenSelectorIndex. The GenSelectorIndex shall be interpreted either as a data PHY Lane index, CPort index or Test Feature index, depending on the particular Attribute. For Attributes not associated with GenSelectorIndex, the GenSelectorIndex parameter shall be ignored.

The semantics of this primitive are:

PA_LM_PEER_GET.ind(GenMIBattribute, GenSelectorIndex)

The primitive's parameter is defined in ***Table 8***.

When Generated

This primitive is generated by the local PA Layer on the reception of a PACP_GET_req frame requesting to obtain local Attribute information.

Effect on Receipt

The DME attempts to retrieve the requested Attribute value from the UniPro stack. The DME shall respond with PA_LM_PEER_GET.rsp that gives the result.

5.4.1.9 PA_LM_PEER_GET.rsp

This primitive reports the results of an information request about the Attribute given in the received PA_LM_PEER_GET.ind.

The semantics of this primitive are:

PA_LM_PEER_GET.rsp(ConfigResultCode, MIBvalue)

The primitive's parameters are defined in ***Table 8***.

When Generated

The DME shall generate the PA_LM_PEER_GET.rsp primitive in response to a PA_LM_PEER_GET.ind from the PA Layer.

Effect on Receipt

The PA Layer is informed about the result of the operation, and in case ConfigResultCode is SUCCESS, MIBvalue carries the MIB Attribute value. For any other value of ConfigResultCode, MIBvalue is undefined. The PA Layer shall use PACP to forward the result of the operation to the peer Device.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.4.1.10 PA_LM_PEER_SET.ind

This primitive attempts to set the MIBvalue value to an Attribute of the local UniPro stack identified by GenMIBattribute, and, if relevant, the GenSelectorIndex. The GenSelectorIndex shall be interpreted either as a data PHY Lane index or CPort index or Test Feature index depending of the Attribute. For Attributes not associated with the GenSelectorIndex, the GenSelectorIndex shall be ignored.

The semantics of this primitive are:

PA_LM_PEER_SET.ind(AttrSetType, GenMIBattribute, MIBvalue, GenSelectorIndex)

The primitive's parameters are defined in ***Table 8***.

When Generated

This primitive is generated by the local PA Layer on the reception of a PACP_SET_req frame requesting to set a local Attribute.

Effect on Receipt

The DME attempts to set the specified Attribute. The DME responds with PA_LM_PEER_SET.rsp that gives the result.

5.4.1.11 PA_LM_PEER_SET.rsp

This primitive reports the results of an attempt to set the value of an Attribute in the UniPro stack or PHYs.

The semantics of this primitive are:

PA_LM_PEER_SET.rsp(ConfigResultCode)

The primitive's parameter is defined in ***Table 8***.

When Generated

The DME shall generate the PA_LM_PEER_SET.rsp primitive in response to a PA_LM_PEER_SET.ind from the PA Layer.

Effect on Receipt

The PA_LM_PEER_SET.rsp confirms the success or failure of the PA_LM_PEER_SET.ind at the DME, and the PA Layer shall use the PACP protocol to forward the result to the peer Device.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.4.1.12 PA_LM_PEER_GET.req

Support for this primitive is optional. However, if an implementation supports this primitive, it shall implement it as described in this section.

This primitive requests information about a given Attribute of the peer Device, this Attribute being identified by GenMIBattribute, and, if relevant, the GenSelectorIndex. The GenSelectorIndex shall be interpreted either as a data PHY Lane index or CPort index or Test Feature index depending of the Attribute. For Attributes not associated with the GenSelectorIndex, the GenSelectorIndex shall be ignored.

The semantics of this primitive are:

PA_LM_PEER_GET.req(GenMIBattribute, GenSelectorIndex)

The primitive's parameter is defined in ***Table 8***.

When Generated

This primitive is generated by the DME to obtain information about an Attribute of the peer Device.

Effect on Receipt

The PA Layer entity attempts to retrieve the requested MIB Attribute value from the peer Device by using PACP protocol and responds with PA_LM_PEER_GET.cnf that gives the result.

5.4.1.13 PA_LM_PEER_GET.cnf

Support for this primitive is optional. However, if an implementation supports this primitive, it shall implement it as described in this section.

This primitive reports the results of an information request about an Attribute of the peer Device.

The semantics of this primitive are:

PA_LM_PEER_GET.cnf(ConfigResultCode, MIBvalue)

The primitive's parameters are defined in ***Table 8***.

When Generated

The PA Layer shall generate the PA_LM_PEER_GET.cnf primitive in response to a PA_LM_PEER_GET.req from the DME.

Effect on Receipt

The DME is informed about the result of the operation, and in case ConfigResultCode is SUCCESS, MIBvalue carries the MIB Attribute value. For any other value of ConfigResultCode, MIBvalue is undefined.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.4.1.14 PA_LM_PEER_SET.req

Support for this primitive is optional. However, if an implementation supports this primitive, it shall implement it as described in this section.

This primitive attempts to set the Attribute of the peer Device indicated by GenMIBattribute and, if relevant, the GenSelectorIndex, to the MIBvalue value. The GenSelectorIndex shall be interpreted either as a data PHY Lane index or CPort index or Test Feature index depending of the Attribute. For Attributes not associated with the GenSelectorIndex, the GenSelectorIndex shall be ignored.

The semantics of this primitive are:

PA_LM_PEER_SET.req(AttrSetType, GenMIBattribute, MIBvalue, GenSelectorIndex)

The primitive's parameters are defined in ***Table 8***.

When Generated

This primitive is generated by the DME to set the indicated Attribute of the peer Device.

Effect on Receipt

By using the PACP protocol, the PA Layer shall attempt to set the specified MIB Attribute of the peer Device. The PA Layer responds with PA_LM_PEER_SET.cnf that gives the result.

5.4.1.15 PA_LM_PEER_SET.cnf

Support for this primitive is optional. However, if an implementation supports this primitive, it shall implement it as described in this section.

This primitive reports the results of an attempt to set the value of an Attribute of the peer Device.

The semantics of this primitive are:

PA_LM_PEER_SET.cnf(ConfigResultCode)

The primitive's parameter is defined in ***Table 8***.

When Generated

The PA Layer shall generate the PA_LM_PEER_SET.cnf primitive in response to a PA_LM_PEER_SET.req from the DME.

Effect on Receipt

The PA_LM_PEER_SET.cnf confirms the success or failure of the PA_LM_PEER_SET.req at the PA Layer, and should have no further effects at the DME.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.4.2 Control Primitives

The following control primitives of the PA_LM_SAP are used for controlling the FastAuto_Mode and SlowAuto_Mode (see ***Table 12***), and for reset and boot purposes (see ***Section 9.11)***.

The RESET, ENABLE_LAYER, LINKSTARTUP and ENDPOINTRESET primitives are prefixed by PA_LM for the PA Layer management SAP.

The primitives are summarized in ***Table 10***.

Table 10 PA_LM Control Primitives

Name	Request	Indication	Local Response	Response	Local Confirm	Confirm
PA_LM_RESET	***5.4.2.1***	–	–	–	***5.4.2.2***	–
PA_LM_ENABLE_LAYER	***5.4.2.3***	–	–	–	***5.4.2.4***	–
PA_LM_LINKSTARTUP	***5.4.2.5***	***5.4.2.7***	–	–	***5.4.2.6***	–
PA_LM_ENDPOINTRESET	***5.4.2.8***	***5.4.2.10***	–	–	***5.4.2.9***	–
PA_LM_HIBERNATE_ENTER	***5.4.2.11***	***5.4.2.13***	–	–	***5.4.2.12***	–
PA_LM_HIBERNATE_EXIT	***5.4.2.14***	***5.4.2.16***	–	–	***5.4.2.15***	–
PA_LM_PRE_HIBERNATE_EXIT	–	***5.4.2.17***	***5.4.2.18***	–	–	–
PA_LM_TEST_MODE	***5.4.2.20***	***5.4.2.19***	–	–	***5.4.2.21***	–
PA_LM_PHY_RESET	–	***5.4.2.22***	–	–	–	–
PA_LM_RX_SYMBOL_CNT	–	***5.4.2.23***	–	–	–	–
PA_LM_TX_SYMBOL_CNT	–	***5.4.2.24***	–	–	–	–

The parameters used for these primitives are defined in ***Table 11***.

Table 11 PA_LM Control Primitive Parameters

Name	Type	Valid Range	Value	Description
AutoState	Enum	FAST_STATE	1	Defines the UniPro Power State when the Device is in FastAuto_Mode or SlowAuto_Mode
		SLOW_STATE	2	
		SLEEP_STATE	4	
ResetLevel	Enum	COLD	0	Defines the reset level
		WARM	1	
GenericErrorCode	Enum	SUCCESS	0	Indicates the result of the request
		FAILURE	1	
PHYDirection	Enum	TX	0	Indicates the target of the reset
		RX	1	

5.4.2.1 PA_LM_RESET.req

This primitive requests the PA_LM to reset the PA Layer and PHY Layer. See ***Section 9.3*** for a description of reset.

The semantics of this primitive are:

PA_LM_RESET.req(ResetLevel)

When Generated

This primitive shall be generated by the DME in order to reset the PA Layer and PHY Layer.

Effect on Receipt

The PA Layer shall reset transmit and receive state machines to the reset power-on states and reset values. The reset values of the PA Layer Attributes, as defined in ***Section 5.8,*** shall take effect.

The PA Layer shall reset the PHY entities and shall put them into a state according to the Hibernate_Mode Power Mode, scheduling the next transition into SlowAuto_Mode Power Mode.

The ResetLevel COLD shall reset the complete PA Layer including any statistics and all Attributes. The ResetLevel WARM shall reset the PA Layer without any statistics, if they exist. During a RESET of the PA Layer or PHY, the PA Layer shall not forward any received data to the DL Layer.

Detailed effects on the physical layer are described in the PHY-specific sections later in this section.

5.4.2.2 PA_LM_RESET.cnf_L

This primitive is used during the UniPro reset procedure (see ***Section 9.11.1***).

The semantics of this primitive are:

PA_LM_RESET.cnf_L( )

When Generated

The PA Layer uses the PA_LM_RESET.cnf_L primitive during the boot procedure to indicate to the DME that the PA Layer came out of reset. PA_LM_RESET.cnf_L is generated in response to PA_LM_RESET.req. The PA Layer is considered reset only, if also all underlying PHY entities have confirmed coming out of reset to the PA Layer.

Effect on Receipt

The DME is informed that the PA Layer came out of reset.

5.4.2.3 PA_LM_ENABLE_LAYER.req

This primitive enables the PA Layer.

The semantics of this primitive are:

PA_LM_ENABLE_LAYER.req( )

When Generated

The PA_LM_ENABLE_LAYER.req primitive is part of the UniPro boot procedure (see ***Section 9.11.2***) and is generated by the DME after the PA Layer comes out of reset.

Effect on Receipt

The PA Layer shall begin monitoring the inbound Lanes for incoming trigger events and respond with PA_LM_ENABLE_LAYER.cnf_L.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.4.2.4 PA_LM_ENABLE_LAYER.cnf_L

This primitive reports that PA Layer has been enabled.

The semantics of this primitive are:

PA_LM_ENABLE_LAYER.cnf_L( )

When Generated

This primitive shall be generated by the PA Layer in response to a PA_LM_ENABLE_LAYER.req primitive from the DME.

Effect on Receipt

The DME is informed that the PA Layer has been enabled and is actively monitoring the inbound Lanes for incoming Link Startup Sequences.

5.4.2.5 PA_LM_LINKSTARTUP.req

This primitive attempts to establish a Link to the peer Device by starting the Link Startup Sequence (L1.5 initialization protocol). See ***Section 5.6.3***.

The semantics of this primitive are:

PA_LM_LINKSTARTUP.req( )

When Generated

The DME shall generate this when it wants to establish a Link to the peer Device.

Effect on Receipt

The PA Layer enters the Link Startup Sequence. ***Section 5.6.3*** defines the order of triggers sent and received during this sequence.

During the Link Startup Sequence, the PA Layer shall not forward any received data to the DL Layer. Also, the PA Layer shall not accept any data from the DL Layer for transmission.

Once the Link Startup Sequence has finalized, the PA Layer shall enter normal operation. This is indicated to the DME with the PA_LM_LINKSTARTUP.cnf_L primitive.

5.4.2.6 PA_LM_LINKSTARTUP.cnf_L

This primitive is used during the UniPro boot procedure (see ***Section 9.11.2)*** to indicate to the DME that the PA Layer completed the Link Startup Sequence and the PA Layer is ready to be used by the Data Link Layer.

The semantics of this primitive are:

PA_LM_LINKSTARTUP.cnf_L(GenericErrorCode)

When Generated

This primitive is generated by the PA Layer after the Link Startup Sequence has completed. GenericErrorCode is set to SUCCESS when the operation has completed successfully, and to FAILURE when the operation failed, e.g. due to timeout.

Effect on Receipt

The DME is informed about the completion of the PA Layer’s Link Startup Sequence.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.4.2.7 PA_LM_LINKSTARTUP.ind

This primitive indicates the reception of the first phase of the Link Startup Sequence.

The semantics of this primitive are:

PA_LM_LINKSTARTUP.ind( )

When Generated

This primitive shall be generated by the PA Layer when it receives a 'TRG_UPR0' trigger in PWM-G1 and the PA Layer is not executing the Link Startup Sequence. Thus, PA_LM_LINKSTARTUP.ind shall not be generated by the PA Layer in the time between PA_LM_LINKSTARTUP.req and PA_LM_LINKSTARTUP.cnf_L. This primitive shall not be generated if the PA Layer has not been enabled (see ***Section 5.4.2.3)***.

The trigger is defined in ***Section 5.7.6***.

Effect on Receipt

The DME issues DME_LINKSTARTUP.ind or DME_LINKLOST.ind depending on the Link state. See ***Section 9.8.2.14*** and ***Section 9.8.2.15***.

5.4.2.8 PA_LM_ENDPOINTRESET.req

This primitive is used to transmit the EndPointReset trigger ('TRG_EPR') over the Link to the peer PA Layer. See ***Section 9.3.3*** for details.

The semantics of this primitive are:

PA_LM_ENDPOINTRESET.req( )

When Generated

The DME shall generate this primitive when it receives an DME_ENDPOINTRESET.req from the DME_SAP user (generally the Application Layer).

Effect on Receipt

Once the EndPointReset trigger is sent, the PA Layer shall indicate it to the DME with the PA_LM_ENDPOINTRESET.cnf_L primitive (see ***Section 9.3.3)***.

5.4.2.9 PA_LM_ENDPOINTRESET.cnf_L

This primitive reports the completion of a request to send a EndPointReset trigger ('TRG_EPR').

The semantics of this primitive are:

PA_LM_ENDPOINTRESET.cnf_L( )

When Generated

This primitive is generated by the PA Layer in response to a PA_LM_ENDPOINTRESET.req primitive, after having sent the EndPointReset trigger ('TRG_EPR').

Effect on Receipt

When the DME receives this primitive from the local PA Layer it forwards a confirmation (Endpoint Reset) to the DME User (generally the Application Layer).

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.4.2.10 PA_LM_ENDPOINTRESET.ind

This primitive indicates the reception of an EndPointReset trigger (‘TRG_EPR’) over the Link.

The semantics of this primitive are:

PA_LM_ENDPOINTRESET.ind( )

When Generated

The PA Layer shall generate this primitive when it receives an EndPointReset trigger (‘TRG_EPR’).

Effect on Receipt

When the DME receives this primitive from the local PA Layer it forwards a notification (Endpoint Reset) to the DME User (generally the Application Layer).

5.4.2.11 PA_LM_HIBERNATE_ENTER.req

This primitive attempts to put the PA Layer and the Link to Hibernate_Mode.

The semantics of this primitive are:

PA_LM_HIBERNATE_ENTER.req( )

When Generated

This primitive is generated by the DME to put the PA Layer and the Link into Hibernate.

Effect on Receipt

The PA Layer responds with PA_LM_HIBERNATE_ENTER.cnf_L to indicate whether or not the request was accepted.

5.4.2.12 PA_LM_HIBERNATE_ENTER.cnf_L

This primitive reports the result of a hibernate request.

The semantics of this primitive are:

PA_LM_HIBERNATE_ENTER.cnf_L(GenericErrorCode)

When Generated

The PA Layer shall generate this primitive in response to a PA_LM_HIBERNATE_ENTER.req. If GenericErrorCode is set to SUCCESS, then the PA Layer begins to hibernate the Link as specified in ***Section 5.7.13.1.*** If GenericErrorCode is set to FAILURE, the PA Layer rejected the request, possibly because the PA Layer is executing a Power Mode change or a Link Startup Sequence.

Effect on Receipt

If GenericErrorCode is set to SUCCESS, the DME shall wait until it receives PA_LM_HIBERNATE_ENTER.ind to know when the operation has been completed.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.4.2.13 PA_LM_HIBERNATE_ENTER.ind

This primitive reports the completion of a hibernate request.

The semantics of this primitive are:

PA_LM_HIBERNATE_ENTER.ind(PowerChangeResultCode)

The primitive parameters are defined in ***Table 9***.

When Generated

The PA Layer shall generate this primitive when the enter hibernate procedure has completed. The parameter carries the result of the operation.

Effect on Receipt

The DME is informed of the completion of the enter hibernate procedure and the final status of the operation.

5.4.2.14 PA_LM_HIBERNATE_EXIT.req

This primitive attempts to put the PA Layer and the Link into Active Mode.

The semantics of this primitive are:

PA_LM_HIBERNATE_EXIT.req( )

When Generated

This primitive is generated by the DME to request the PA Layer and the Link to exit hibernate.

Effect on Receipt

The PA Layer responds with PA_LM_HIBERNATE_EXIT.cnf_L to indicate whether or not the request was accepted.

5.4.2.15 PA_LM_HIBERNATE_EXIT.cnf_L

This primitive reports the results of a hibernate exit request.

The semantics of this primitive are:

PA_LM_HIBERNATE_EXIT.cnf_L(GenericErrorCode)

When Generated

The PA Layer shall generate this primitive in response to a PA_LM_HIBERNATE_EXIT.req. If GenericErrorCode is set to SUCCESS, then the PA Layer begins to exit hibernate as specified in ***Section 5.7.13.3.*** If GenericErrorCode is set to FAILURE, the PA Layer rejected the request.

Effect on Receipt

The DME is informed of the result of the operation. If GenericErrorCode is set to SUCCESS, the DME shall wait until it receives PA_LM_HIBERNATE_EXIT.ind to know when the operation has been completed.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.4.2.16 PA_LM_HIBERNATE_EXIT.ind

This primitive reports the completion of a hibernate exit request.

The semantics of this primitive are:

PA_LM_HIBERNATE_EXIT.ind(PowerChangeResultCode)

The primitive parameters are defined in ***Table 9***.

When Generated

The PA Layer shall generate this primitive when the exit hibernate procedure has completed.

Effect on Receipt

The DME is informed of the completion of the exit hibernate procedure.

Note:

The exit hibernate procedure might have been requested by a peer Device as well as the DME. At this point, the PA Layer and the Link are in the same state as before hibernation.

5.4.2.17 PA_LM_PRE_HIBERNATE_EXIT.ind

This primitive reports reception of a hibernate exit request.

The semantics of this primitive are:

PA_LM_PRE_HIBERNATE_EXIT.ind( )

When Generated

The PA Layer shall generate this primitive when it receives a M-LANE-HIBERN8Exit.indication primitive from the PHY Layer.

Effect on Receipt

The DME shall prepare the other Layers for hibernate exit, see ***Section 9.5.2***. The DME shall respond with PA_LM_PRE_HIBERNATE_EXIT.rsp_L once it has completed the task.

5.4.2.18 PA_LM_PRE_HIBERNATE_EXIT.rsp_L

This primitive reports the completion of a pre-hibernate exit request by the DME.

The semantics of this primitive are:

PA_LM_PRE_HIBERNATE_EXIT.rsp_L( )

When Generated

The DME shall generate the PA_LM_PRE_HIBERNATE_EXIT.rsp_L primitive in response to a PA_LM_PRE_HIBERNATE_EXIT.req from the PA Layer.

Effect on Receipt

The PA Layer shall continue the hibernate exit procedure by issuing a hibernate exit request on the outbound Link.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.4.2.19 PA_LM_TEST_MODE.ind

This primitive reports the reception of a request to put the PA Layer in a given test mode.

The semantics of this primitive are:

PA_LM_TEST_MODE.ind( )

When Generated

The PA Layer shall generate this primitive when a PACP_TEST_MODE_req frame is received before PA_LM_LINKSTARTUP.ind and the PA_LogicalLaneMap attribute is updated to assign RX Logical Lane Number 0 to the Lane, which received the PACP_TEST_MODE_req frame.

Effect on Receipt

The DME is informed that the PA Layer has been put in the testing mode and shall inform the user of the DME with DME_TEST_MODE.ind primitive.

5.4.2.20 PA_LM_TEST_MODE.req

Support for this primitive is optional. If the DME_TEST_MODE.req primitive is supported by a DME implementation, the PA Layer shall support the PA_LM_TEST_MODE.req primitive as described in this section.

This primitive attempts to put the PA Layer of the peer Device to the test mode.

The semantics of this primitive are:

PA_LM_TEST_MODE.req( )

When Generated

This primitive is generated by the DME to request the PA Layer to put the PA Layer of the peer Device in to a given test mode. The primitive is only accepted before PA_LM_LINKSTARTUP.ind.

Effect on Receipt

The PA Layer shall send the PACP_TEST_MODE_req frame.

The PA Layer responds with PA_LM_TEST_MODE.cnf_L to inform the completion of the request.

5.4.2.21 PA_LM_TEST_MODE.cnf_L

Support for this primitive is optional. If the PA_LM_TEST_MODE.req primitive is supported by an implementation, it shall support the PA_LM_TEST_MODE.cnf_L primitive as described in this section.

This primitive reports the completion of a test mode request.

The semantics of this primitive are:

PA_LM_TEST_MODE.cnf_L( )

When Generated

The PA Layer shall generate this in response to a PA_LM_TEST_MODE.req from the DME, after having sent the PACP_TEST_MODE_req frame.

Effect on Receipt

The DME is informed about the completion of the operation.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.4.2.22 PA_LM_PHY_RESET.ind

This primitive reports that the PHY was reset due to a received or generated LINE-RESET.

The semantics of this primitive are:

PA_LM_PHY_RESET.ind(PHYDirection)

When Generated

The PA Layer shall generate this when it detects an incoming LINE-RESET or it drives an outgoing LINE-RESET. PHYDirection shall be set to TX when the PA Layer generates the LINE-RESET (i.e. M-TX is affected), and to RX when the PA Layer receives the LINE-RESET (i.e. M-RX is affected).

Effect on Receipt

The DME is informed that the PHY has been reset and shall inform the user of the DME with DME_ERROR.ind primitive (see ***Section 9.8.2.28)***. Since the PHY reset clears all Attributes to their default values, the user has to re-apply all custom settings.

5.4.2.23 PA_LM_RX_SYMBOL_CNT.ind

This primitive reports every chunk of 1024 received valid PHY symbols, independent of the number of M-PHY Lanes in use and disregarding any PA frame or DL Frame structure.

The semantics of this primitive are:

PA_LM_RX_SYMBOL_CNT.ind( )

When Generated

The PA Layer shall generate this indication every time it has received 1024 valid PHY symbols. Valid PHY symbols include any PHY data symbols or PHY control Markers between M-PHY HOB and UniPro EOB, even those received with PHY Errors.

Effect on Receipt

The DME is informed that the PHY has received valid PHY symbols and updates its Quality of Service Counters accordingly.

5.4.2.24 PA_LM_TX_SYMBOL_CNT.ind

This primitive reports every chunk of 1024 transmitted PHY symbols, independent of the number of M-PHY Lanes in use and disregarding any PA frame or DL Frame structure.

The semantics of this primitive are:

PA_LM_TX_SYMBOL_CNT.ind( )

When Generated

The PA Layer shall generate this indication every time it has transmitted 1024 PHY symbols. PHY symbols include any PHY data symbols or PHY control Markers between M-PHY HOB and UniPro EOB.

Effect on Receipt

The DME is informed that the PHY has transmitted valid PHY symbols and updates its Quality of Service Counters accordingly.

5.5 Structure and Encoding of Protocol Data Units

The PA Layer communicates with its peer PA Layer via PHY Adapter Protocol Data Units (PA_PDUs). There are two types of PA_PDUs, one for normal data and one for escaped data. These are defined in the following sections.

All PA_PDUs have a fixed length of 17-bits.

5.5.1 Data PA_PDU

The Data PA_PDU carries DL Layer and PACP frame data symbols as payload. It can be initiated by a PA_DATA.req from the DL Layer, or autonomously by the PA Layer.

This PDU has the structure shown in ***Figure 15***.

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Data															

Figure 15 Data PA_PDU

The header bit (bit 16) is fixed to '0' in this PDU. The remaining 16-bits carry the DL Layer or PACP frame data symbols in a way that bits [15:0] of the data symbol map directly to bits [15:0] of the Normal Data PA_PDU.

5.5.2 Escaped Data PA_PDU

The Escaped Data PA_PDU carries escaped data as payload. It can be initiated by the DL Layer or autonomously by the PA Layer.

Figure 16 shows the general structure of this PDU.

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	EscType								EscParam							

Figure 16 Escaped Data PA_PDU

In the Escaped Data PA_PDU, the header bit (bit 16) is set to '1'. The payload of the Escaped Data PA_PDU is partitioned into the EscType field (bits [15:8]) and EscParam (bits [7:0]). The only valid values for EscType are ESC_DL and ESC_PA, as described in ***Section 5.5.2.1*** and ***Section 5.5.2.2***, respectively.

A transmitter shall not transmit an Escaped Data PA_PDU with EscType set to any value other than ESC_DL or ESC_PA.

5.5.2.1 DL Escaped Data PA_PDU

The DL Layer uses the PA_ESCDATA.req primitive to request the transmission of an Escaped Data PA_PDU. In this case, the PDU transports the primitive parameters EscType and EscParam over the Link. This information shall be passed to the peer DL Layer by a PA_ESCDATA.ind. ***Figure 17*** shows the generic structure of a DL Escaped Data PA_PDU.

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	ESC_DL								EscParam_DL							

Figure 17 DL Escaped Data PA_PDU

UniPro defines only one EscType for the higher protocol layers, namely the ESC_DL.

For the PA_PDU composition and CRC calculation, the ESC_DL is encoded to '0b00000001'. Therefore, an Escaped Data PA_PDU that is initialized by the DL Layer with EscType=ESC_DL looks like in ***Figure 18***.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Note that the ESC_DL encoding in the PA_PDU is different from the ESC_DL encoding for the physical layer, which is described in ***Section 5.7.3***.

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0	1	EscParam_DL							

Figure 18 DL Escaped Data PA_PDU with EscType = ESC_DL

5.5.2.2 PA Escaped Data PA_PDU

An Escaped Data PA_PDU can also be generated by the PA Layer autonomously without request from the DL Layer. In this case it transports PA Layer control information to the peer PA Layer and the PDU is called 'PA Escaped Data PA_PDU'. Its information shall not be passed to the peer DL Layer with a PA_ESCDATA.ind.

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	ESC_PA								EscParam_PA							

Figure 19 PA Escaped Data PA_PDU

For PDU composition and CRC calculation, the ESC_PA is encoded to '0b11111110'.

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	0	EscParam_PA							

Figure 20 PA Escaped Data PA_PDU with EscType = ESC_PA

Note that the ESC_PA encoding in the PA_PDU is different from the ESC_PA encoding for the physical layer, which is described in ***Section 5.7.3.***

Receiving a PA Escaped Data PA_PDU shall be reported to L2 using the PA_ERROR.ind primitive with PAErrorCode set to BAD_PA_PARAM (see ***Section 5.3.3.1)*** when the following condition occurs:

- An ESC_PA symbol is received with an undefined value of EscParam_PA.

5.6 Protocol Features

This section defines the generic PHY Adapter Layer protocol features such as Lane distribution. The remaining PHY-specific protocol features are described in ***Section 5.6.3.3.***

5.6.1 UniPro Link Management

An available Lane is a Lane that is implemented in the PHY. A Lane can be either connected or unconnected, depending of the interconnection between the local PHY and the peer PHY. The connection state is automatically discovered during the Link Startup Sequence. An active Lane is used for data transfers while an inactive or unused Lane is not. The number of active Lanes is set by PA_ActiveTxDataLanes and PA_ActiveRxDataLanes.

5.6.1.1 Power Modes

UniPro defines six Power Modes that are abstracted from the Power Modes provided by the underlying PHY.

Each of the UniPro Power Modes Fast_Mode, Slow_Mode, Hibernate_Mode and Off_Mode map to exactly one UniPro Power State. Each of the UniPro Power Modes Fast_AutoMode and Slow_AutoMode map to two UniPro Power States, which can be switched autonomously.

UniPro supports up to four PHY Data Lanes per direction in all modes. Unused Lanes shall be kept in HIBERNATE_STATE. Unconnected Lanes shall be kept in OFF_STATE.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Table 12 summarizes the different UniPro Power Modes, their properties and their mapping to UniPro Power States.

Table 12 UniPro Power Modes

Power Mode	Description	UniPro Power State
Fast_Mode	The Fast_Mode is capable of the highest data transmission rate of all Power Modes. The Link is always ready to send and receive data while providing a well-defined latency, which is the lowest of any UniPro Power Modes. The actual data rate per Data Lane in the Fast_Mode is PHY-specific.	FAST_STATE
Slow_Mode	The Slow_Mode is also capable of transmitting data. It should be less than, or equal to, the data rate in Fast_Mode and consume less, or equal, power. The provided latency is well-defined but might be higher than in the Fast_Mode. The actual data rate in the Slow_Mode is PHY-specific.	SLOW_STATE
FastAuto_Mode	The FastAuto_Mode allows the UniPro stack to autonomously switch between the FAST_STATE and the SLEEP_STATE, depending whether or not the DL Layer has data to send. By switching between states, the FastAuto_Mode might save power compared to the Fast_Mode, but at the cost of a possibly less well-defined latency. UniPro does not dictate a certain method for the autonomous switching. In any case, a PA Layer shall be able to transmit DL Layer data when in FastAuto_Mode.	FAST_STATE, SLEEP_STATE
SlowAuto_Mode	The SlowAuto_Mode allows the UniPro stack to autonomously switch between the SLOW_STATE and the SLEEP_STATE, depending on whether or not the DL Layer has data to send. By switching between states, the SlowAuto_Mode might save power compared to the Slow_Mode, but at the cost of a possibly less well-defined latency. UniPro does not dictate a certain method for the autonomous switching. In case a UniPro stack does not provide any method for autonomous switching, its SlowAuto_Mode shall be strictly mapped to the SLOW_STATE. In any case, a PA Layer shall be able to transmit DL Layer data when in SlowAuto_Mode.	SLOW_STATE, SLEEP_STATE
Hibernate_Mode	The Hibernate_Mode is a low Power Mode in which no data transmission is possible. There may be a long latency returning from this mode to one of the other modes. The Hibernate_Mode should consume less power than any of the other modes in this table, except Off_Mode.	HIBERNATE_STATE
Off_Mode	The Off_Mode prepares for the removal of power. When a Device is ready to have its power removed it enters the OFF_STATE.	OFF_STATE

A UniPro Power Mode is assigned per Link direction, thus the Power Mode of the forward Link may be different from the Power Mode of the reverse Link. Note that all active Lanes per direction shall have the same power configuration. In M-PHY, a UniPro stack is able to set the Power Mode of its forward and reverse Links via the Link Configuration Procedure described in ***Section 5.7.12***.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.6.2 Lane Distribution and Merging

UniPro allows transmitting data over multiple PHY Data Lanes, when the bandwidth of a single Lane is not sufficient. The PHY Adapter Layer distributes the transmitted data over multiple PHY Data Lanes on the sending side and merges the data again on the receiving side.

Figure 21 shows the single Data Lane case. In terms of Lane distribution this is the trivial case; the PHY Adapter Protocol Data Units (PA_PDUs) shall be delivered in order over the only existing PHY Data Lane (PHY Data Lane 0). In the following figures, PA_PDU #0 is the earliest generated PDU.

When multiple Lanes are used, PA symbols shall be transmitted in order, starting from Lane 0 through Lane (PA_ActiveTxDataLanes - 1). Similarly, PA symbols shall be received in order starting from Lane 0 through Lane (PA_ActiveRxDataLanes - 1). When a single Lane is used, Lane 0 shall be used to transmit and receive PA symbols. Note that a L2 Frame may begin from any active Lane N given that Lanes 0 through Lane N-1 contain data and the Lane distribution order is followed.

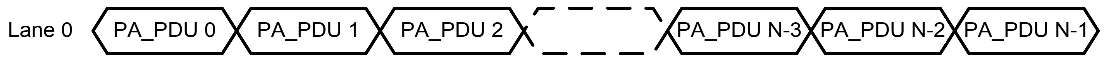


Figure 21 PHY Adapter PDU Ordering in a Single Lane Configuration

Figure 22 defines the PHY Adapter Layer PDU ordering for two Data Lanes. The diagram shows that the PDU #0 and PDU #1 are transferred simultaneously on the two Data Lanes.

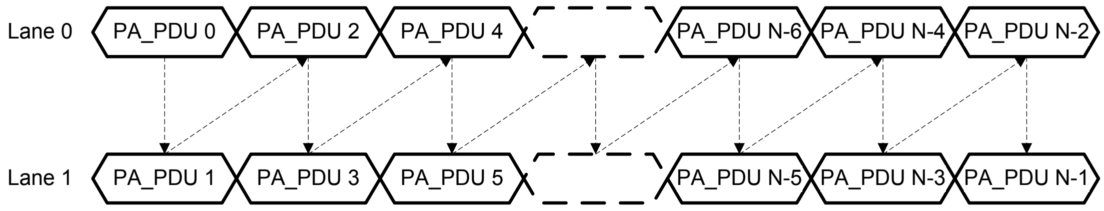


Figure 22 PHY Adapter PDU Ordering in a Dual Lane Configuration

Figure 23 defines the PDU ordering for a three Data Lane configuration. Here, the PDU #0, PDU #1 and PDU #2 are transferred simultaneously on the three Data Lanes.

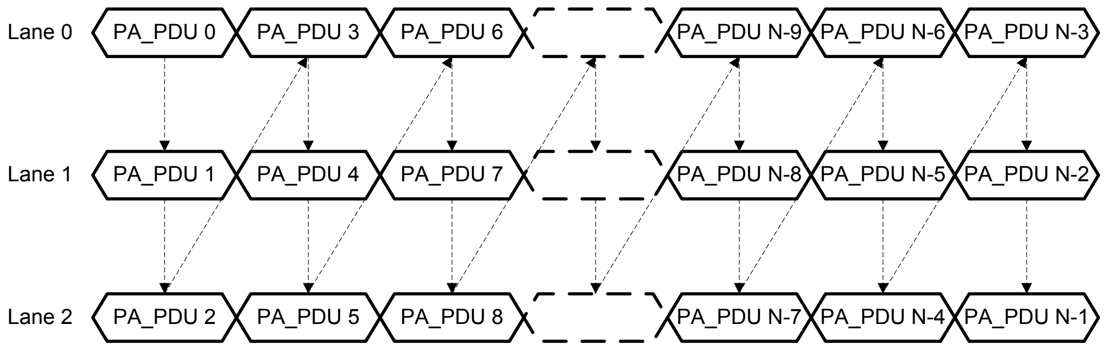


Figure 23 PHY Adapter PDU Ordering in a Three Lane Configuration

The PDU ordering for a four Data Lane configuration is depicted in ***Figure 24***. Here, the PDU #0, PDU #1, PDU #2 and PDU #3 are transferred simultaneously on the four Data Lanes.

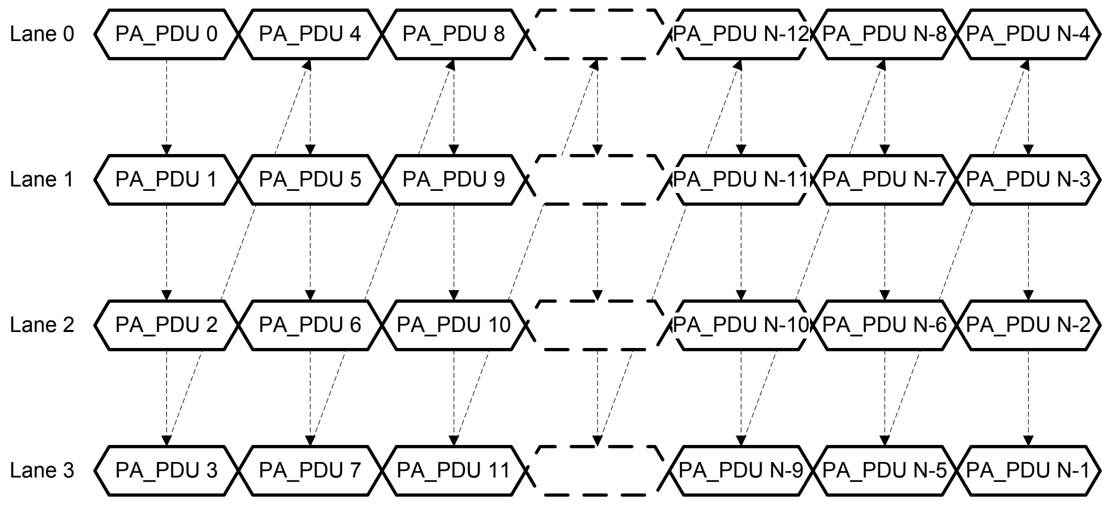


Figure 24 PHY Adapter PDU Ordering in a Four Lane Configuration

The described Lane distribution scheme is independent of the actual PHY type. The mapping of PHY Adapter PDUs to PHY bits is defined in ***Section 5.7.3.***

5.6.3 Link Startup Sequence

The Link Startup Sequence is a multi-phase handshake, which exchanges UniPro trigger events to establish initial Link communication, in both directions, between two directly attached UniPro Devices. Mapping of the UniPro trigger events to M-PHY trigger events is defined in ***Section 5.7.6.***

Before initial or repeated Link Startup Sequence, all available M-TXs and M-RXs shall be (re-)powered and reset into M-PHY HIBERN8 state, and the PA Layer shall be reset and enabled using the primitives described in ***Section 5.4.2.1*** to ***Section 5.4.2.4.***

Link Startup shall be initiated by the DME using the PA_LM_LINKSTARTUP.req primitive. The Link Startup Sequence might be initiated at the same time on both ends of the Link or at different times. The UniPro stack shall not autonomously generate a Link Startup as part of DL Layer error recovery.

The Link Startup Sequence is protected with a timeout mechanism. If the sequence is not completed within a specified time (PHY-specific), the sequence is aborted and the DME is notified with a PA_LM_LINKSTARTUP.cnf_L primitive having the status parameter set to FAILURE. A successful Link Startup is indicated with the status parameter set to SUCCESS.

5.6.3.1 Phases of the Link Startup Sequence

Both ends of a Link might enter the Link Startup Sequence at the same time. Alternatively, one end might start a Link Startup and the other end might detect this, reset itself and then enter into Link Startup.

Table 13 defines the five phases of the Link Startup Sequence. When the PA Layer receives the PA_LM_LINKSTARTUP.req primitive from the DME, it shall enter the sequence at the initialization phase (see ***Section 5.7.8.1***). When the PA Layer finalizes the Link Startup Sequence in phase 5, it shall confirm this to the DME with the PA_LM_LINKSTARTUP.cnf_L primitive.

The Link Startup Sequence is very robust. It uses robust PHY trigger events and transmits each trigger multiple times. If a Link Startup fails continuously it probably indicates that the peer Device is not present, not powered, or has some fatal error. The failure is reported to the DME with a PA_LM_LINKSTARTUP.cnf_L primitive with the parameter set to FAILURE.

Each Device shall incorporate a timeout to know when it is appropriate to abort the attempt to startup a Link (see ***Section 5.7.8.1***).

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Table 13 Link Startup Sequence

Phase	PA Transmitter	PA Receiver
0	Continue to send TRG_UPR0 (all available Lanes)	Wait for TRG_UPR0 reception. Lane discovery (see ***Section 5.7.8.2)***
0b	Send two additional TRG_UPR0 (all available Lanes) Then proceed with phase 1	Ignore all data
1	Continue to send TRG_UPR1	Wait for TRG_UPR1 reception on all Lanes Re-align Lane numbering (***Section 5.7.8.3)***
2	Send two additional TRG_UPR1 Then proceed with phase 3	Ignore all data
3	Continue to send TRG_UPR2 on all Lanes	Wait for TRG_UPR2 reception on PHY RX Data Lane 0 • TRG_UPR2 (-> advance to phase 4) • TRG_UPR1 (-> ignore) • Others (-> ignore)
4	Send two additional TRG_UPR2 on all Lanes Then proceed with Phase 5	Ignore all data
5	Transfer Capabilities	Receive Capabilities and apply down grading
	Report to the DME using PA_LM_LINKSTARTUP.cnf_L that the Link Startup Sequence succeeded. Exit the Link Startup Sequence and enter SlowAuto_Mode.	

5.6.3.2 Terminating a Link Startup

A UniPro Link Startup sequence shall be aborted without reporting PA_LM_LINKSTARTUP.cnf_L to the DME by either of the following conditions:

- Local Application setting Power Mode to Off_Mode (via DME_POWEROFF.req or implementation specific procedures)
- Local Assertion of UniPro Cold Reset or UniPro Warm Reset

5.6.3.3 Error Processing during Link Startup

During Link Startup, the PA layer shall only advance the Link Startup Phase, but never revert to an earlier Phase, even when receiving errors on the incoming Link.

The PA Receiver should advance from Phase 0, Phase 1 or Phase 3 under the sufficient criteria of a correct reception of the corresponding expected trigger symbols.

The PA Receiver should otherwise ignore received errors during Phase 0 up to Phase 4.

The PA Receiver should ignore errors in the received M-PHY burst structure during Phase 0 up to Phase 5, see ***Figure 44*** and ***Figure 45.***

The PA Receiver shall follow the error handling defined in ***Section 5.7.7***, when in Phase 5. The PA Receiver shall remain in Phase 5 and generate a timeout, if any of the expected PACP capability indication frames is not received.

A PA_LM_LINKSTARTUP.cnf_L(FAILURE) shall be generated only from timeout. The timeout may happen during any Phase of the Link Startup sequence and PA Layer and M-PHY settings between local and peer Device may be out of synchronization. In particular, timeout in local and peer PA Layer happen at different times. The PA Layer shall be reset and enabled again, the M-PHYs shall be powered and reset into

HIBERN8 immediately after PA_LM_LINKSTARTUP.cnf_L(FAILURE), to prepare for a new Link Startup attempt from the local DME or the peer Device.

5.7 PHY Adapter to M-PHY Interaction

This section defines the interaction between the PHY Adapter and an M-PHY physical layer ***[MIPI02]***. An M-PHY transmitter and an M-PHY receiver are called M-TX and M-RX, respectively.

The M-PHY primitives and Attributes referred in this section are defined in ***[MIPI02]***.

5.7.1 Data Transmission

An M-TX is capable of transmitting eight bit data symbols and a few control symbols. A UniPro stack shall use the following M-PHY control symbols: MARKER0 (MK0), MARKER1 (MK1), MARKER2 (MK2), MARKER3 (MK3), MARKER4 (MK4) and FILLER (FLR).

PA TX shall use the following M-PHY M-TX-DATA primitives to transmit M-PHY symbols and to control the Link's state:

- M-LANE-SYMBOL
- M-LANE-PREPARE
- M-LANE-BurstEnd
- M-LANE-SaveState
- M-LANE-AdaptStart
- M-LANE-AdaptComplete

PA RX shall use the following M-PHY M-RX-DATA SAP primitives to receive M-PHY symbols, to detect errors, and to detect changes in the Link's state:

- M-LANE-SYMBOL
- M-LANE-PREPARE
- M-LANE-BurstEnd
- M-LANE-HIBERN8Exit
- M-LANE-MRXSaveState
- M-LANE-AdaptComplete

5.7.2 PHY Control and Attributes

Each M-TX and M-RX has a separate set of Attributes as defined in ***[MIPI02]***. These Attributes are accessed using the following M-PHY M-TX-CTRL-SAP and M-RX-CTRL-SAP primitives:

- M-CTRL-CFGGET
- M-CTRL-CFGSET

In addition, the following M-PHY primitives shall also be used for controlling and for monitoring the PHY:

- M-CTRL-CFGREADY
- M-CTRL-RESET
- M-CTRL-LINERESET

The PA Layer shall control M-TX Tactivate timing generation (TActivateControl = "ProtocolControlled" with M-TX RSE_PO_TX control), if required by the M-TX architecture.

The PA Layer shall provide an access to all local M-PHY MODULE Attributes via the PA_LM_GET.req and PA_LM_SET.req primitives. The peer M-PHY MODULE Attributes are accessible via the PA_LM_PEER_GET.req and PA_LM_PEER_SET.req primitives. The M-PHY Attributes (0x00 to 0xFF, see ***[MIPI02]***) shall be linearly mapped to UniPro Attributes 0x0000 to 0x00FF. For example, M-PHY's TX_Amplitude Attribute (0x25) can be accessed using UniPro Attribute ID 0x0025. A particular M-TX or

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

M-RX shall be identified using a selector where the values 0 to 3 are mapped to TX logical Lanes 0 to 3 and the values 4 to 7 are mapped to RX logical Lanes 0 to 3.

After UniPro Cold Reset or UniPro Warm Reset, the Logical Lane map is undefined until attribute PA_LogicalLaneMap is initially set. During this time the selector shall signify the Physical Lane Number instead of the Logical Lane Number. The Logical Lane map is defined after reception of a PA_LM_LINKSTARTUP.cnf_L, reception of a PA_LM_TEST_MODE.ind or after local confirmation PA_LM_SET.cnf_L from setting attribute PA_LogicalLaneMap.

The following Attributes are write-protected after PA_LM_LINKSTARTUP.cnf_L(SUCCESS) due to their critical nature in the PA Layer's Power Mode control:

- TX_MODE
- TX_PWMGEAR
- TX_HSGEAR
- TX_HSRATE_Series
- TX_HIBERN8_Control
- TX_LCC_Sequencer
- TX_LCC_Enable
- TX_HS_Unterminated_LINE_Drive_Enable
- MC_HS_Unterminated_LINE_Drive_Enable
- MC_HS_Unterminated_Enable
- TX_LS_Terminated_LINE_Drive_Enable
- MC_LS_Terminated_LINE_Drive_Enable
- MC_LS_Terminated_Enable
- RX_MODE
- RX_PWMGEAR
- RX_HSGEAR
- RX_HSRATE_Series
- RX_Enter_HIBERN8
- RX_HS_Unterminated_Enable
- RX_LS_Terminated_Enable

Trying to set these Attributes shall be rejected with a READ_ONLY_MIB_ATTRIBUTE error. An access to an Attribute of an unavailable Lane shall be rejected with a BAD_INDEX error.

The UniPro protocol determines PA_SaveConfigTime as the maximum of local TX_Min_SAVE_Config_Time_Capability and peer RX_Min_SAVE_Config_Time_Capability. In order to calculate PA_SaveConfigTime correctly, the start of both time periods denoted by the two Min_SAVE_Config_Time_Capabilities needs to be synchronized to the entry into M-PHY SAVE state (ideally the time of M-PHY RCT). After elapsed PA_SaveConfigTime, the UniPro protocol may start immediately requesting a new M-TX Burst and hence requires a peer M-RX being able to receive the Burst.

The UniPro Protocol requires that TX_Min_SAVE_Config_Time_Capability advertised in the M-TX corresponds to a time period starting with the entry into M-TX SAVE state and finishing with the M-TX being able to transmit the next Burst.

The UniPro Protocol requires that RX_Min_SAVE_Config_Time_Capability advertised in the M-RX corresponds to a time period starting with the entry into M-RX SAVE state and finishing with the M-RX being ready to receive the next Burst. In case of ambiguities resulting from implementation or detection, both TX_Min_SAVE_Config_Time_Capability and peer RX_Min_SAVE_Config_Time_Capability shall reflect the worst case minimum period required in SAVE state. These periods defined by TX_Min_SAVE_Config_Time_Capability and RX_Min_SAVE_Config_Time_Capability shall include all timings required to enter M-PHY SAVE state and to exit from SAVE state, including those timings not accounted for in ***[MIPI02]***, such as power ramping and clock ramping.

An M-PHY may exhibit an implementation specific delay between the entry into SAVE state and the execution of the RCT. Capabilities shall always advertise their values based on the assumption of zero delay between RCT and entry into SAVE state.

5.7.2.1 LINE-RESET

The PA Layer issues a LINE-RESET to the PHY in three cases:

1. At the beginning of phase 0 of the Link Startup Sequence
2. At the recovery step in the Power Mode change operation
3. On a request from the DL Layer with a PA_INIT.req primitive

The LINE-RESET shall be issued on all connected TX Lanes if the Link Startup Sequence has been completed. Otherwise, the LINE-RESET shall be issued on all available TX Lanes. Before issuing the LINE-RESET, the PA Layer shall close an existing TX burst, wake up all available or connected TX Lanes, and wait for PA_TActivate. After completing the LINE-RESET operation, the PA Layer shall wait for PA_SaveConfigTime.

The PA Layer shall indicate to the DME via a PA_LM_PHY_RESET.ind primitive if the PHY has been reset due to a received LINE-RESET or due to a transmitted LINE-RESET. Since LINE-RESET causes PHY to reset many of its Attributes to their default values, the Application needs to SET the optimized values after each reset.

5.7.3 Symbol Translation

The translation process for UniPro PA_PDU Symbols into M-PHY Symbols involves for every Lane the following mapping process:

- Mapping of the PA_PDU to M-PHY symbol pairs
- Insertion of IDLE sequences and M-PHY related control markers.
- Data Scrambling
- Insertion of skip symbols

The logical process order at the PA TX follows the order of this list, the logical process order at the PA RX follows the inverse order of this list.

5.7.3.1 Mapping of PA_PDU to PHY_SAP Symbols

Each PA_PDU is translated to exactly two M-PHY symbols. The formed M-PHY symbol pair is shown using the notation <high, low>. The "high" symbol shall be transmitted before the "low" symbol. For data symbols, PA_PDU[15:8] and PA_PDU[7:0], represented as eight bit binary bit sequence "HGFEDCBA" the highest numbered bit is the most significant bit with the capital letter "H", and the lowest numbered bit is the least significant bit with the capital letter "A".

The eight bit M-PHY data symbol is mapped onto parameter "DataValue" of the PHY_SAP primitive M-LANE-SYMBOL with "DataNCtrl" set to 0.

For all M-PHY symbols transmitted over the link the M-PHY 8b10b encoding/decoding services shall be applied.

PA_PDU symbols require mapping to:

- A data PA_PDU is mapped to <PA_PDU[15:8], PA_PDU[7:0]>.
- A DL escaped PA_PDU (see ***Figure 18***) is mapped to M-PHY symbols <MK0, PA_PDU[7:0]>
- A PA escaped PA_PDU (see ***Figure 20***) is mapped to M-PHY symbols <MK1, PA_PDU[7:0]>.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.7.3.2 Burst Start and Deskew Pattern

An M-PHY burst shall begin by transmitting a deskew pattern <MK0, MK1>, in which MK0 functions as an M-PHY HEAD-OF-BURST marker. The deskew pattern is also used when resynchronizing Lanes (see ***Section 5.7.10)***. The deskew pattern shall be transmitted simultaneously on all active Lanes. The deskew pattern may be transmitted at any point in time, but the PA Transmitter shall satisfy a proximity rule, keeping a distance of 8 PA_PDUs (which is equal to 16 SI) on every Lane between transmission of consecutive deskew patterns. The PA Receiver shall be able to receive deskew patterns at any point in time.

For the purpose of potential error recovery, a deskew pattern shall be transmitted in the following scenarios:

1. When requested from the PA Service User via PA_LANE_ALIGN.req, while the PA TX is not in a SAVE state, see ***Section 5.7.10.***
2. When requested from the PA Service User via PA_INIT.req and before the outbound Link transmits the associated PACP_PWR_req.
3. During the time interval between reception of a PACP_PWR_req and transmission of a PAC_PWR_cnf frame, if the received PACP_PWR_req frame triggers a PA_INIT.ind
4. During the time interval between two identically transmitted PACP request frames, during which the PACP_Request_Timer timed out.
5. During the time interval between two transmitted PACP confirm frames, when the last PACP confirm frame is sent in response to a replayed PACP request frame. The replay condition is recognized by comparing the PACP CCITT-CRC16 fields of the last two successively received PACP request frames.

The deskew pattern at the start of the burst may be exploited to satisfy the above five rules, if additional insertion of a deskew pattern violates the proximity rule.

Deskew pattern transmission shall respect the 16-bit PA_PDU protocol alignment at the transmitter.

5.7.3.3 Dummy Burst

A dummy burst is an M-PHY burst that is opened temporarily on otherwise inactive M-PHY Lanes in a Multi-Lane scenario during Link configuration. The purpose of a dummy burst is to synchronize the activation of configured M-PHY Lane attributes across both, inactive and activated Lanes, to the end of the Burst (***[MIPI02]*** RCT event). A dummy burst does not carry any payload and is not used in Lane distribution and merging (see ***Section 5.6.2***). The PA receiver ignores the data transferred via a dummy burst. A dummy burst is issued only on Lanes with a logical Lane number greater than 0.

The dummy burst shall begin by transmitting a special pattern <MK0, FLR>, which distinguishes the dummy burst from a normal burst. During the dummy burst, the PA Layer shall transmit only FILLERs. The PA Layer shall end the dummy burst in the same manner as a normal burst (see ***Section 5.7.4)***.

5.7.3.4 Burst End

When ending a burst or dummy burst, the PA Layer TX shall perform the following steps on all Lanes with open bursts:

1. Transmit one MK2 symbol, i.e. an M-PHY End-of-Burst marker
2. Optionally, transmit a second MK2 symbol.
3. Transmit FILLER symbols (FLR) or IDLE sequence for PA_TxTrailingClocks
4. Issue an M-LANE-BurstEnd.request

The PA Layer shall align MK2 symbols across Lanes using <FLR, FLR> symbol pairs, if necessary (see ***Figure 25***).

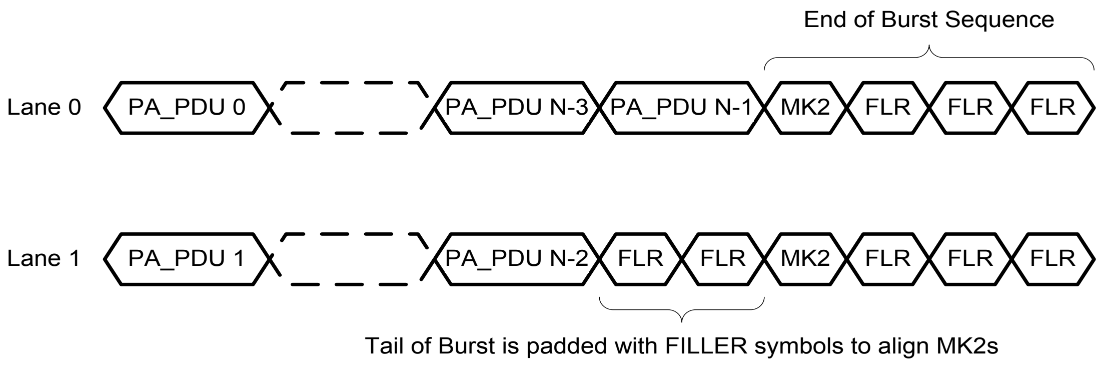


Figure 25 End of Burst Sequence with PA_TxTrailingClocks = 3, PA_ActiveTxDataLanes = 2, and Odd Number of PA_PDUs (informative)

The PA RX ignores all received M-PHY symbols and M-PHY error messages prior to the Head-of-Burst <MK0, MK1> sequence and after reception of the last MK2.

Instead of a single <MK2> M-PHY symbol, the transmitter may close the burst by a PA_PDU aligned M-PHY symbol pair of <MK2, MK2> (End-of-Burst) markers. When starting the next burst, the local PA Layer shall add a guard time between an M-LANE-PREPARE.confirm and the M-LANE-SYMBOL.request for transmission of the <MK0>, Head-of-Burst. When starting a HS-BURST or a PWM-BURST in PWM-G6 or PWM-G7, this additional guard time results in an extended period of M-TX SYNC symbol transmission.

When starting a PWM-BURST without M-TX SYNC Symbol transmission, the additional guard time results in an extended period of M-TX $T_{PWM\text{-}PREPARE}$, see ***[MIPI02]***. In this case, the guard time extension shall not extend $T_{PWM\text{-}PREPARE}$ beyond $T_{Line\text{-}Reset\text{-}Detect}$ time defined by ***[MIPI02]***. Detection of a symbol pair of <MK2, MK2>, together with the extended time between M-LANE-PREPARE.confirm and M-LANE-SYMBOL.request may be exploited by an M-RX or PA-Layer for advanced Power saving mechanisms.

The additional guard time is defined by the attribute PA_MK2ExtensionGuardBand and is settable by Application layer. When this attribute is set to '0' (default), the additional guard time shall follow the attribute PA_TActivate.

The PA Layer of the receiver shall treat the first occurrence of an MK2 marker as the EoB and the second MK2 marker as the indication to expect an additional guard time from the transmitter. Legacy receivers ignore the second MK2 marker as it is received outside the payload frame.

In Multi-Lane configurations, the PA TX shall transmit <MK2, MK2> on all activated Lanes or Lanes transmitting a dummy burst at the same Symbol Interval. The Receiver shall observe <MK2, MK2> only from Logical Lane 0.

The PA TX shall not transmit an <MK2, MK2> when the M-TX is configured to enter HIBERN8. The decision on generation of a <MK2, MK2> M-PHY symbol pair is implementation-specific and depends on Application data transmission statistics. It is not within the scope of this specification.

If the PA Layer receives an M-LANE-BurstEnd.indication from the M-RX without having previously received an End-of-Burst Marker, it shall treat the M-LANE-BurstEnd.indication at the same time as an End-of-Burst Marker.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.7.3.5 Mapping of IDLE Sequences

If the PA Layer does not have anything to transmit, and the burst is active, the PA Layer shall insert IDLE sequences. Within a burst, between the initial deskew pattern (<MK0>, <MK1>) and the Tail-of-Burst, IDLE sequences may be inserted anywhere into the PA_PDU symbol stream. IDLE sequences may be inserted into a DL Frame. It is permitted, but discouraged, to insert IDLE sequences into a PACP frame. When inserted into a CRC protected frame structure, inserted IDLE sequences shall not be considered for CRC-16 calculation or PACP frame length checking.

IDLE sequence insertion shall be carried out before the Lane distribution described in ***Section 5.6.2***. IDLE sequence mapping onto PHY symbol sequences described later in this chapter shall be carried out after the Lane distribution process. The receiver at the PA Layer shall remove IDLE sequences after Lane merging.

If scrambling is disabled or not implemented, IDLE sequences are mapped to <FLR, FLR> M-PHY symbol pairs in order to guarantee PA symbol alignment. In other words, the 2-M-PHY symbol alignment is followed throughout the burst, including any idle periods. The receiver at the PA Layer shall remove <FLR, FLR> M-PHY symbol pairs.

If scrambling is enabled, the PA Layer shall encode, in every Lane independently, an existing IDLE sequence of N successive PA_PDU aligned M-PHY symbol pairs consisting of <FLR,FLR>, into a sequence starting with a PA_PDU aligned <MK3>, followed by (N-1) M-PHY symbol pairs carrying <D.07.3, D.07.3> and ending with <FLR>. During the subsequent scrambling process, the PA TX shall scramble M-PHY data <D.07.3, D.07.3> into pseudo-random M-PHY data symbols. The PA RX shall de-scramble the received sequence again into a sequence of N successive PA_PDU aligned M-PHY symbol pairs consisting of <FLR,FLR>. The shortest possible encoded IDLE sequence consists of <MK3,FLR> with no <D.07.3, D.07.3> symbol encoding in between. Due to transmission errors, the PA RX might decode a badly formed IDLE sequence or an IDLE sequence carrying PHY errors. In such a case, the receiver shall stop unrolling the ongoing IDLE sequence upon the reception of any PHY control symbol. In a multi-Lane Link, IDLE sequence encoding and decoding shall be performed independently on all M-PHY Lanes, if scrambling is enabled.

5.7.3.6 M-PHY Data Scrambling

Data Scrambling is a technique used to mitigate problems related to Electromagnetic Interference. The scrambling feature described in this chapter is optional normative. If a UniPro implementation includes support for scrambling, the scrambling feature shall be implemented as described in this chapter.

Data being transmitted shall be scrambled with a Pseudo-Random-Bit-Stream (PRBS) to reduce the likelihood of repetitive pattern in the Link.

Scrambling support shall be implemented in inbound and outbound direction of the PA Layer. Scrambling shall be enabled or disabled at every Power Mode Change request. The new scrambling settings apply to the Link at the first burst following the completed Power Mode change procedure.

After UniPro Cold or Warm Reset, the scrambling process is disabled. Scrambling shall be requested only, if the peer Device has flagged the ability to descramble data during a PACP_CAP_EXT1_ind exchange (PA_PeerScrambling). A scrambling request always involves scrambling on inbound and outbound Link. However, scrambling shall be applied only to Links in Fast_Mode or FastAuto_Mode. Links in Slow_Mode or SlowAuto_Mode are not scrambled, despite an active request for scrambling.

The PA-Layer shall scramble at the transmitter after PA_PDU mapping to 8-bit M-PHY symbols and before 8b10b encoding inside M-TX. The PA-Layer shall de-scramble at the receiver after 10b8b decoding inside the M-RX and before matching 8-bit M-RX symbols to PA_PDUs. M-PHY skip symbols, if used, are always inserted at the PA TX between scrambler and 8b10b encoder and removed at the PA RX between 10b8b decoder and de-scrambler.

Data Scrambling shall apply on a per-Lane base. In Multi-Lane M-PORT Links, scrambling shall appear as independent between the Lanes. All active Lanes, and Lanes transmitting a dummy burst, going into the same

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

direction shall be scrambled. M-PHY Marker symbols shall never be scrambled. M-PHY Data symbols shall be scrambled, if the scrambling process is enabled.

The PRBS shall be generated using the Galois form of a Linear Feedback Shift Register (LFSR) implementing the generator polynomial:

$$G(x) = x^{16} + x^5 + x^4 + x^3 + 1$$

The LFSR shall be initialized dependent on the logical Lane number to the following 16-bit seed values:

- Logical Lane 0: seed = 0x0040
- Logical Lane 1: seed = 0x0080
- Logical Lane 2: seed = 0x00C0
- Logical Lane 3: seed = 0x0100

In active Lanes, initialization shall be triggered during transmission or reception of a deskew sequence <MK0, MK1> and the following M-PHY Data Symbol shall be scrambled starting with the (re-) initialized LFSR output G(x). Lanes transmitting a dummy Burst shall initialize the LFSR at the start of the Burst. A Tester Equipment attempting in-flight Line snooping may use the deskew-pattern to synchronize its own descrambler.

The LFSR shall generate an eight-bit sequence at G(x) for every M-PHY Symbol to be scrambled, starting from its reset value. The LFSR shall generate new bit sequences of G(x) for every M-PHY symbol, the LFSR polynomial generation shall also advance for eight-bit cycles, when M-PHY marker symbols are presented to the scrambler. However, skip symbol insertion shall not advance the LFSR as skip symbol insertion logically happens after the scrambling process.

Scrambling shall be achieved by bit-wise addition (X-OR) of a sequence of eight bits G(x) with the M-PHY data Symbol bit to be scrambled, starting from bit 0 up to bit 7 of the 8-bit M-PHY data symbol. The scrambler scrambles the high symbol of the <high, low> symbol pair first. Refer to ***Figure 26*** for an example of scrambling in Lane #0.

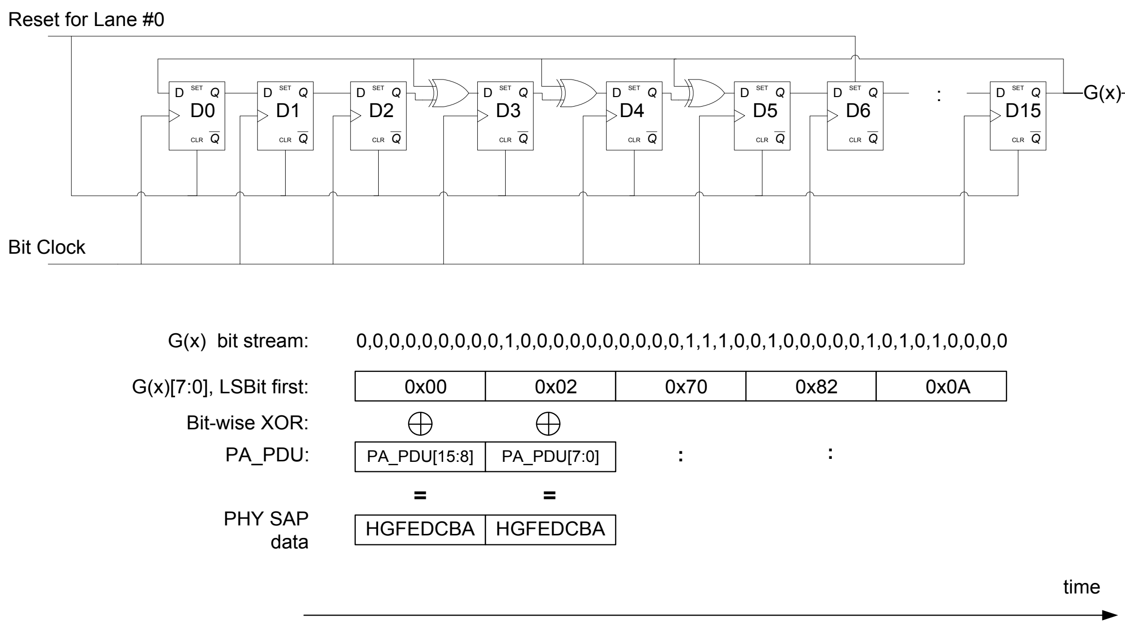


Figure 26 Example for Scrambler and PA_PDU Data Scrambling (Informative)

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.7.3.7 Skip Symbol Insertion

The PA Layer shall insert skip symbol pairs <MK4, MK4> in intervals not exceeding PA_TxSkipPeriod, if skip symbol insertion is enabled. The purpose of skip symbol pair insertion is to reduce protocol payload bandwidth at the transmitter in order to compensate for symbol interval tolerances between fast transmitters and slow receivers. The receiver shall discard received skip symbol pairs before the descrambling process.

During Link Startup Capability exchange, the peer PA Layer informs the local PA Layer about its architectural requirements for skip symbol reception by setting Status[2] in PACP_CAP_EXT1_ind. On reception of PACP_CAP_EXT1_ind, this Flag is copied into the local PA_TxSkip. PA_TxSkip may be reset by the Application after Link Startup under the following conditions:

- It is known to the Application that the Link uses a Shared Reference Clock.
- It is known to the Application that the TX transmission bandwidth is lower than the RX sampling bandwidth for any period in time.

Prior to this version, UniPro has no capability to insert or receive skip symbols. An implementation of this version of UniPro shall ensure compatibility to earlier versions of UniPro, where skip symbols cannot be used. Skip symbol insertion shall be disabled after a UniPro Cold Reset or Warm Reset. The Application shall not enable skip symbol insertion when the peer Device has not sent a PACP_CAP_EXT1_ind frame during Link Startup. The Application may not enable skip symbol insertion when the peer Device has confirmed via PACP_CAP_EXT1_ind frame that it does not intend to make use of the received skip symbols.

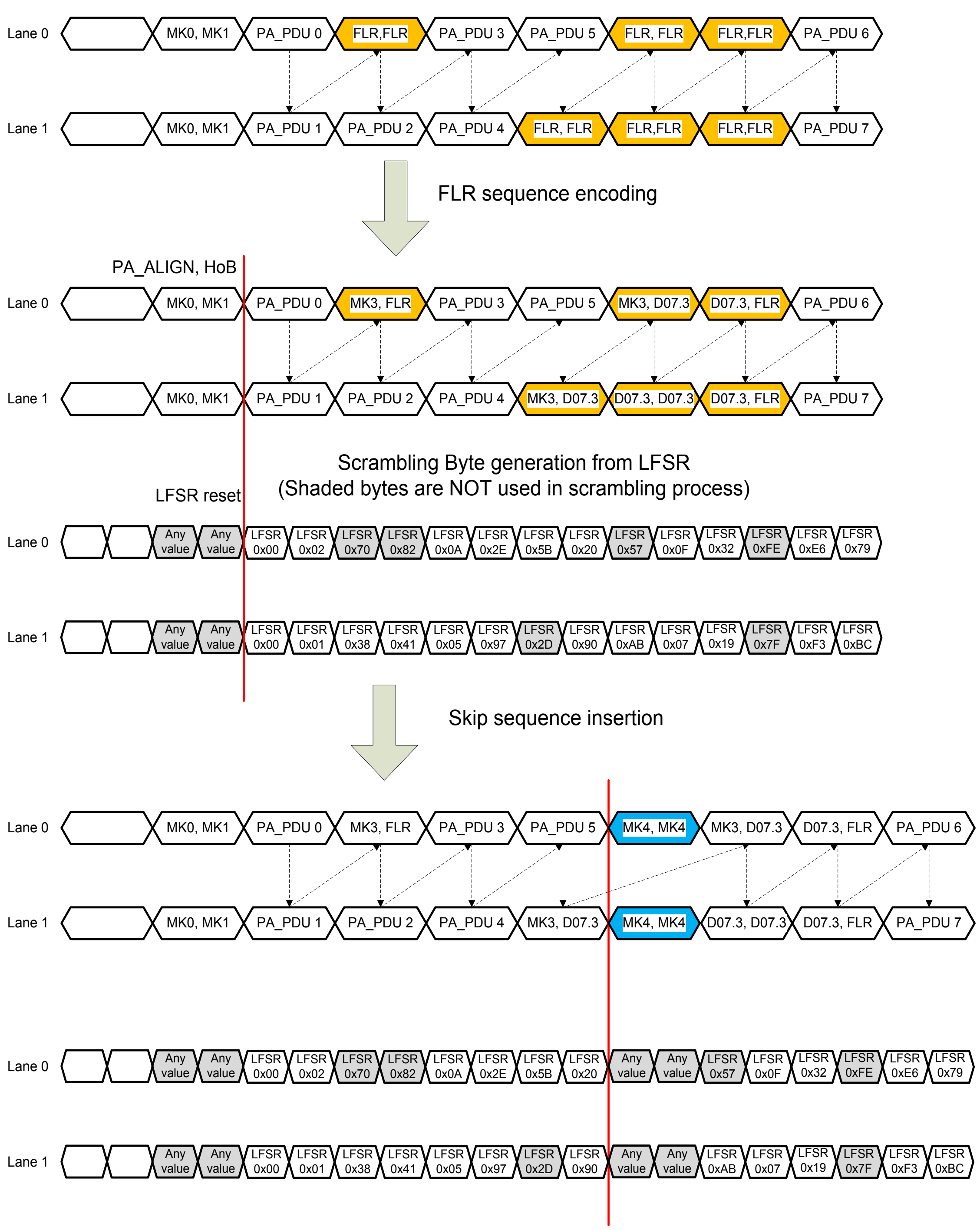
Within one Lane and during any arbitrary time interval between two M-PHY skip symbol pairs <MK4, MK4>, the difference between the number of 8b10b encoded M-PHY symbols transmitted at the actual transmission rate and the number of 8b10b encoded M-PHY symbols for a hypothetical transmission at the lowest possible defined transmission rate shall never exceed two 8b10b encoded M-PHY symbols. Valid M-PHY transmission rates are defined in ***[MIPI02]***.

Skip symbol pairs shall be inserted only during Fast_Mode or FastAuto_Mode bursts. Skip symbol insertion shall be performed on active Lanes only after the <MK0, MK1> deskew pattern at the start of a burst and before the first <MK2> (EoB). Skip symbol insertion shall be performed during a dummy burst only after the <MK0, FLR> pattern at the start of the dummy burst and only before the first <MK2> (EoB).

With scrambling enabled, the PA layer processes skip symbols in the following logical order, independent of the actual implementation: local PA TX scrambling, followed by local PA TX skip symbol insertion, followed by local M-TX Line encoding, followed by peer M-RX Line decoding, followed by peer PA RX skip symbol removal, followed by peer PA RX descrambling.

M-PHY skip symbol pairs shall be inserted on all active, or inactive connected Lanes with opened dummy burst, at the same symbol time interval, skip symbol shall be removed on a per Lane decision in the receiver. Skip symbol insertion shall not break the 16-bit PA_PDU protocol alignment, but are not correlated in any other way to the symbol stream to be transmitted.

Figure 27 depicts a two Lane example for scrambling and skip sequence insertion. LFSR encoder bytes are generated per Lane and bit-wise X-OR'ed to the data payload PA_PDU[15:8] and PA_PDU[7:0]. LFSR scrambler bytes shown in ***Figure 27*** are those composed of the bit sequence delivered by G(x).


FLR sequence encoding
PA_ALIGN, HoB
Scrambling Byte generation from LFSR
(Shaded bytes are NOT used in scrambling process)
LFSR reset
Skip sequence insertion
Lane 0
Lane 1

2084 **Figure 27 Scrambling and Skip Sequence Insertion (Two-Lane Example)**

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.7.4 Power State Mapping

The mapping between UniPro Power State and M-PHY power states is defined in ***Table 14***.

Table 14 UniPro Power State to M-PHY State Mapping

UniPro Power Mode	UniPro Power State	M-PHY state
Fast_Mode	FAST_STATE	HS-BURST
FastAuto_Mode		
Slow_Mode	SLOW_STATE	PWM-BURST
SlowAuto_Mode		
FastAuto_Mode	SLEEP_STATE	STALL
SlowAuto_Mode		SLEEP
Hibernate_Mode	HIBERNATE_STATE	HIBERN8
Off_Mode	OFF_STATE	UNPOWERED

A UniPro Link may use any number of Lanes in SLOW_STATE and FAST_STATE.

After the end of the burst, the PA Layer TX guarantees a minimum re-configuration time for the M-PHY before beginning a new burst. The minimum time values are obtained in the Link Startup Sequence (see ***Section 5.7.8.5)***, and cover both the local M-TX and the peer M-RX. The following three cases shall be detected:

1. HS burst ending with no configuration applied. The PA Layer shall wait for PA_StallNoConfigTime, starting from reception of a M-LANE-SaveState.indication primitive, before beginning a new burst (M-LANE-PREPARE.request).
2. PWM burst ending with no configuration applied. The PA Layer shall wait for PA_SleepNoConfigTime, starting from reception of a M-LANE-SaveState.indication primitive, before beginning a new burst (M-LANE-PREPARE.request).
3. HS burst or PWM burst ending with a new configuration applied to one, or both, Link directions. The PA Layer shall wait for PA_SaveConfigTime, starting from the moment when all MODULEs are in SAVE state, before beginning a new burst (M-LANE-PREPARE.request) or before issuing M-LANE-AdaptStart.request. PA_SaveConfigTime also applies in those cases where a new configuration for one or both Link directions does not differ from the old configuration.

In all three cases above, the PA TX may end a burst on an extended EoB sequence <MK2, MK2> and add an extra guard time defined by PA_MK2ExtensionGuardBand between the next M-LANE-PREPARE.confirm and the M-LANE-SYMBOL.request for the <MK0>, Head-of-Burst.

The PA Layer shall have a timer, HIBERNATE_TIMER, to ensure the minimum hibernate time, T_{HIBERN8}, (see ***[MIPI02]***) is observed. The PA Layer shall set HIBERNATE_TIMER to PA_Hibern8Time when a M-TX enters HIBERN8. The PA Layer shall wait for HIBERNATE_TIMER to expire before performing an operation that would cause a M-TX to exit HIBERN8. A value of zero in PA_Hibern8Time shall disable HIBERNATE_TIMER.

When waking-up a Lane from HIBERNATE_STATE, the PA Layer shall keep the M-TX in SLEEP_STATE for the period set in PA_TActivate. The value of PA_TActivate takes into account the peer M-RX's T_{ACTIVATE}, as defined in ***[MIPI02]***, and possibly other system-specific timing issues.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.7.5 Error Mapping

The 3b4b_Error, 5b6b_Error, and RD_Error ***[MIPI02]*** errors reported by M-RX shall be indicated to the DL Layer with a PA_ERROR.ind primitive having PAErrorCode set to BAD_PHY_SYMBOL.

The Res_Error error reported by M-RX shall be indicated to the DL Layer with a PA_ERROR.ind primitive having PAErrorCode set to UNMAPPED_PHY_ESC_SYMBOL.

If the PA Layer receives an illegal PHY symbol or a sequence of two aligned PHY symbols not forming a valid PA_PDU, the PA Layer shall pass a PA_ERROR.ind primitive with PAErrorCode set to UNEXPECTED_PHY_ESC_SYMBOL to the DL Layer. A Lane receiving a dummy burst shall never report a PA_ERROR.ind. A PA_Error.ind shall be reported only after the PA Layer has acquired the Lane synchronization and until the last MK2 was received.

Valid PHY symbol pairs after deskewing are listed below:

<MK0, MK1>, <MK0, PA_PDU[7:0]>, <MK1, PA_PDU[7:0]>, <PA_PDU[15:0]>, <MK2, FLR>, <MK2, MK2>, <FLR, FLR>

If scrambling is active:

<MK3, FLR>, <MK3, PA_PDU[7:0]>, <PA_PDU[15:8], FLR>

If skip pattern insertion has been requested from the peer:

<MK4, MK4>

Each received PA escaped PA_PDU with EscParam_PA other than PACP_BEGIN, TRG_UPR0, TRG_UPR1, or TRG_UPR2 shall be indicated to the DL Layer with a PA_ERROR.ind primitive having the PAErrorCode set to BAD_PA_PARAM.

5.7.6 Trigger Mapping

UniPro trigger TRG_UPR0 has a 2-bit parameter representing the physical Lane number that the trigger was transmitted on. The mapping to ESC_PA and further to M-PHY symbols is shown in ***Table 15***. The trigger is transmitted on all available Lanes with a unique parameter on each Lane.

Table 15 TRG_UPR0 Mapping

Physical Lane	EscParam	M-PHY Symbol Pair
0	0x40	<MK1, 0x40>
1	0x41	<MK1, 0x41>
2	0x42	<MK1, 0x42>
3	0x43	<MK1, 0x43>

UniPro trigger TRG_UPR1 has a 4-bit parameter containing information regarding connected Lanes in a bitmap format. The bit n of the parameter shall be set to ‘1’ if a TRG_UPR0 has been received on Physical Lane n. TRG_UPR1 is mapped to an escaped PA_PDU with EscParam parameter set to 0x80 + C, where C is the 4-bit parameter of TRG_UPR1. Minimum and maximum values for C are 1 and 15, respectively. ***Table 16*** shows examples for TRG_UPR1 mappings. The trigger is transmitted on all available Lanes with the same value of C.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Table 16 TRG_UPR1 Mapping Examples (Informative)

Lane information	Value of C	EscParam	M-PHY Symbol Pair
Lane 0	0b0001	0x81	<MK1, 0x81>
Lane 0 and 1	0b0011	0x83	<MK1, 0x83>
Lane 0, 1, 2, and 3	0b1111	0x8F	<MK1, 0x8F>
Lane 3 and 1	0b1010	0x8A	<MK1, 0x8A>

UniPro trigger TRG_UPR2 is mapped to an escaped PA_PDU with EscParam set to 0xC0. This translates to <MK1, 0xC0>. The trigger is transmitted on all available Lanes.

UniPro trigger TRG_EPR is mapped to PACP_EPR_ind (see ***Section 5.7.7.5)***. The trigger is transmitted as any PACP frame, i.e. following the normal Lane distribution rules.

5.7.7 PA Control Protocol (PACP)

The PA Layer uses PA Control Protocol (PACP) to communicate with the peer PA Layer. A PACP frame is transmitted as any other DL Layer or PA Layer data on active Lanes with the exception that the start of a PACP frame shall be aligned to Logical Lane 0. The PA Layer may transmit FILLERs, if necessary, to guarantee this alignment. The PACP frame may be interleaved into a DL Frame transmission, either respecting the DL Frame boundary or completely being nested into a single DL Frame. PACP frames themselves cannot be nested with other PACP frames or be preempted by DL Frames.

Before transmitting a PACP frame, the PA Layer shall get permission to use the Link from the DL Layer. This is accomplished by issuing a PA_DL_PAUSE.ind primitive to the DL Layer. When the DL Layer has reached to a state where it can pause its transmission, it replies with a PA_DL_PAUSE.rsp_L primitive. Once the PA Layer has finished the operation, e.g. a Power Mode change, it shall indicate the DL Layer via the PA_DL_RESUME.ind primitive that the DL Layer can continue its transmission.

The PACP frame structure is shown in ***Figure 28***. All bits that are marked as Reserved shall be set to '1'. All fields of a PACP frame shall be encoded in Big-endian format.

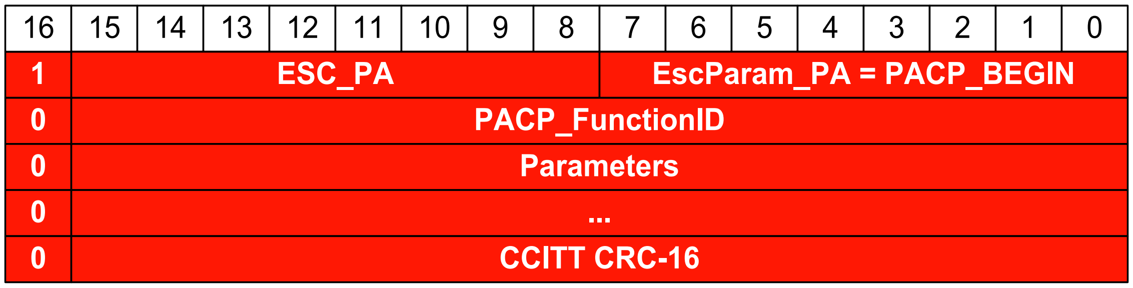


Figure 28 PACP Frame Structure

The receiving PA Layer detects the beginning of a PACP frame from the unique start symbol, an escaped PA_PDU with EscParam set to PACP_BEGIN, which is equal to 0x01. The 16-bit field PACP_FunctionID identifies the function and also determines how many 16-bit parameters, if any, follow. The PACP frame ends with a CCITT CRC-16 field. The CRC shall cover all symbols, including bit 16, from the start symbol coding the PACP_BEGIN up to, but not including the CCITT CRC-16 symbol of the frame. Also, bit 16 of the PA Layer data symbol that carries the CCITT CRC-16 checksum, which is always set to 0, shall not be covered. The CRC-16 checksum shall be calculated over the whole PACP frame using the same algorithm as in the DL Layer (see ***Section 6.6.11)***. The receiving PA Layer shall silently ignore all PACP frames that fail the checksum check, use an invalid PACP_FunctionID, or fail the length check against the lower 8 bits of

PACP_FunctionID. The reception of a new PACP start symbol shall cause the PA Layer to discard any PACP frame that has not been received completely.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Table 17 defines all valid values for PACP_FunctionID and are described in detail in other sections.

Table 17 PACP Functions

PACP_FunctionID	Name	Parameter Count	Description
0x010E	PACP_PWR_req	14	Power Mode change request
0x020D	PACP_PWR_cnf	13	Power Mode change confirmation
0x0306	PACP_CAP_ind	6	Capability information.
0x0309	PACP_CAP_EXT1_ind	9	Capability information extension 1
0x030C	PACP_CAP_EXT2_ind	12	Capability information extension 2
0x0400	PACP_EPR_ind	0	EndPointReset request
0x0500	PACP_TEST_MODE_req	0	To set PA Layer in PHY testing mode
0x0602	PACP_GET_req	2	Request the value of an Attribute
0x0703	PACP_GET_cnf	3	Return the value of an Attribute
0x0805	PACP_SET_req	5	Set the value of an Attribute
0x0901	PACP_SET_cnf	1	Return result of setting the value of an Attribute
0x0A3F	PACP_TEST_DATA_0	63	Encapsulated test pattern for PHY testing, four different payload lengths.
0x0A7D	PACP_TEST_DATA_1	125	
0x0ABD	PACP_TEST_DATA_2	189	
0x0AFD	PACP_TEST_DATA_3	253	

The PA Layer has two counters for counting received PACP frames, PA_PACPFrameCount and PA_PACPErrorCount. PA_PACPFrameCount shall be incremented by one if the received frame has a valid FunctionID and passes the checksum verification. PA_PACPErrorCount shall be incremented by one for each PACP start symbol that is not part of a PACP frame that incremented PA_PACPFrameCount. The counters shall automatically roll-over to zero on an overflow. A PA_LM_SET request to PA_PACPFrameCount or PA_PACPErrorCount shall set the counter to zero.

5.7.7.1 PACP_PWR_req

A PACP_PWR_req frame is used when changing a Link's power configuration (see ***Section 5.7.12.2)***. A complete frame is shown in ***Figure 29***.

The frame parameters contain the requested power configuration for both directions and a data payload (PAPowerModeUserData) for DME. Note that in this frame "TX" and "RX" refer to the outbound and inbound directions of the requestor, respectively.

The fields shall be as follows (related PA Layer Attribute given in parenthesis):

- DevID: Local Device ID, set to 0x80 when N_DeviceID_valid is FALSE, otherwise set to the value of N_DeviceID
- Flags

  Flags[5]: Scrambling request: set to '1' if scrambling is requested. The value of Flags[5] is based on Attribute PA_Scrambling, when the Power Mode Change is initiated by the Application. During PA_INIT, Flags[5] value is based on the current Power Mode Configuration. Flags[5] shall be considered as a scrambling request only if

PA_PeerScrambling is set to 1 (peer indicated scrambling capability in PACP_CAP_EXT1_ind frame).
Flags[4]: UserDataValid, set to '1' if the frame contains valid PAPowerModeUserData
Flags[3]: HS Series, set to '0' for Series A and to'1' for Series B (PA_HSSeries)
Flags[2]: LINE-RESET request
Flags[1]: TX-direction termination enable (PA_TxTermination)
Flags[0]: RX-direction termination enable (PA_RxTermination)

- Adapt: This field indicates the presence of ADAPT and type of ADAPT range selected for the current Power Mode Change (PA_TxHsAdaptType).
- TxMode: UniPro Power Mode for TX-direction (PA_PWRMode[b2:b0]). Set to 0x3 when entering hibernate.
- TxLane: Active Lane count for TX-direction (PA_ActiveTxDataLanes)
    - set to '0' when PA_ActiveTxDataLanes=4, else set to PA_ActiveTxDataLanes
- TxGear: PWM or HS gear for TX-direction (PA_TxGear)
- RxMode: UniPro Power Mode for RX-direction (PA_PWRMode[b6:b4]). Set to value 0x3 when entering hibernate.
- RxLane: Active Lane count for RX-direction, set to '0' when count = 4 (PA_ActiveRxDataLanes)
    - set to '0' when PA_ActiveRxDataLanes=4, else set to PA_ActiveRxDataLanes
- RxGear: PWM or HS gear for RX-direction (PA_RxGear)
- PAPowerModeUserData: user-data for peer DME (PA_PWRModeUserData)

The direction-specific parameters shall be ignored if the direction's Mode is set to UNCHANGED (Refer footnote 4 in ***Table 28***). However, UniPro configures the M-PHY with the current active configuration and applies M-CTRL-CFGREADY. If the LINE-RESET flag is asserted, the receiving PA Layer shall issue a LINE-RESET before processing the received frame.

If the UserDataValid flag is set to '1', the receiving PA Layer shall pass the data received in the PAPowerModeUser field to the DME via the PA_LM_PWR_MODE.ind primitive (see ***Section 5.7.12.4)***.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	ESC_PA								EscParam_PA = PACP_BEGIN							
0	PACP_FunctionID = PACP_PWR_req															
0	DevID								Adapt		Flags					
0	TxMode			TxLane		TxGear			RxMode			RxLane		RxGear		
0	PAPowerModeUserData[0]															
0	PAPowerModeUserData[1]															
0	PAPowerModeUserData[2]															
0	PAPowerModeUserData[3]															
0	PAPowerModeUserData[4]															
0	PAPowerModeUserData[5]															
0	PAPowerModeUserData[6]															
0	PAPowerModeUserData[7]															
0	PAPowerModeUserData[8]															
0	PAPowerModeUserData[9]															
0	PAPowerModeUserData[10]															
0	PAPowerModeUserData[11]															
0	CCITT CRC-16															

Figure 29 PACP_PWR_req

5.7.7.2 PACP_PWR_cnf

The PACP_PWR_cnf frame is a response to a PACP_PWR_req frame. The frame structure is shown in ***Figure 30***.

The frame's fields shall be set as follows:

- Status:
  - The Status field shall contain the status of the requested Power Mode change. The possible values of the Status field are listed in ***Table 18***.
- FC:
  - Shall be set to '0'. if a '1' is received, then the PACP_PWR_cnf definition follows legacy UniPro version.
- Adapt: The definition of this field is identical to the field defined in PACP_PWR_req, ***Section 5.7.7.1***. The value of the field shall be exactly the same as what is received in the corresponding PACP_PWR_req frame.
- PAPowerModeUserData [0..9]:
  - The PAPowerModeUserData [0..9] parameters carry the data payload delivered from DME via PA_LM_PWR_MODE.rsp_L (PAPowerModeUserData[0..9]) primitive.
- DevID, Flags, TxMode, TxLane, TxGear, RxMode, RxLane, and RxGear:
  - The definition of these fields are identical to the fields defined in PACP_PWR_req, ***Section 5.7.7.1***. The value of the fields shall be exactly the same as what is received in the corresponding PACP_PWR_req frame.
  - These parameters are only used for debug and testing thus the parameters shall be ignored and silently discarded by the receiver.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	ESC_PA								EscParam_PA = PACP_BEGIN							
0	PACP_FunctionID = PACP_PWR_cnf															
0	Reserved							FC	Status							
0	PAPowerModeUserData[0]															
0	PAPowerModeUserData[1]															
0	PAPowerModeUserData[2]															
0	PAPowerModeUserData[3]															
0	PAPowerModeUserData[4]															
0	PAPowerModeUserData[5]															
0	PAPowerModeUserData[6]															
0	PAPowerModeUserData[7]															
0	PAPowerModeUserData[8]															
0	PAPowerModeUserData[9]															
0	DevID								Adapt		Flags					
0	TxMode			TxLane		TxGear			RxMode			RxLane		RxGear		
0	CCITT CRC-16															

Figure 30 PACP_PWR_cnf

Table 18 PACP_PWR_cnf Status Values

Status	Enumeration	Description
PWR_OK	0	Request was accepted and executed.
PWR_BUSY	3	Request was rejected due to concurrent access.
PWR_ERROR_CAP	4	Request was rejected due to capability mismatch.

5.7.7.3 PACP_CAP_ind

The PACP_CAP_ind frame is shown in ***Figure 31***. It shall be used in the last state of the Link Startup Sequence to notify the peer PA Layer of the local M-TX and M-RX capabilities (see ***Section 5.7.8)***.

The frame's fields shall be set as follows (source of the information given in parenthesis):

- Flags
    - Flags[1]: HS_MODE untermination capability (TX_HS_Unterminated_LINE_Drive_Capability)
    - Flags[0]: LS_MODE termination capability (TX_LS_Terminated_LINE_Drive_Capability)
- MaxHS: maximum HS gear, zero if HS mode unavailable (TX_HSGEAR_Capability, TX_HSMODE_Capability)

  This field shall be ignored by the PA receiver if PACP_CAP_EXT2_ind is received. Instead, the MaxHS field of PACP_CAP_EXT2_ind shall be used.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

If TX_HSGEAR_Capability is returned with value 4 or above, the PA transmitter shall set this field with value 2’b11, in all other cases, the PA transmitter shall set this field with the value retrieved from TX_HSGEAR_Capability[1:0].

- MaxPWM: Maximum PWM gear (TX_PWMGEAR_Capability)
- TSleepNoConfig: M-PHY timing information (RX_Min_SLEEP_NoConfig_Time_Capability)
- TStallNoConfig: M-PHY timing information (RX_Min_STALL_NoConfig_Time_Capability)
- TSaveConfig: M-PHY timing information (RX_Min_SAVE_Config_Time_Capability)
- VersionInfo: Local UniPro version (PA_LocalVerInfo)

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	ESC_PA								EscParam_PA = PACP_BEGIN							
0	PACP_FunctionID = PACP_CAP_ind															
0	TSleepNoConfig				Reserved					Flags		MaxHS		MaxPWM		
0	TStallNoConfig								TSaveConfig							
0	VersionInfo															
0	Reserved															
0	Reserved															
0	Reserved															
0	CCITT CRC-16															

Figure 31 PACP_CAP_ind

5.7.7.4 PACP_CAP_EXT1_ind

The PACP_CAP_EXT1_ind frame is shown in ***Figure 32***. It gathers all evolutionary introduced capabilities defined from M-PHY specification revision 2.0 onwards. It shall be used in the last state of the Link Startup Sequence directly before transmission of the PACP_CAP_ind to notify the peer PA Layer of the local M-TX, M-RX and PA Layer capabilities (see ***Section 5.7.8)***.

Legacy Devices prior to UniPro version 1.6 not being able to recognize PACP_CAP_EXT1_ind shall silently discard the reception of those PACP frames and proceed to PACP_CAP_ind reception.

The frame’s fields shall be set as follows (source of the information given in parenthesis):

- Status
    - Status[0]: MK2 Extension Support
        - 1: PA exploits EoB extension <MK2, MK2>
        - 0: PA does not exploit EoB extension <MK2, MK2>
    - Status[1]: scrambling support
        - 1: PA implements scrambling
        - 0: PA does not implement scrambling
    - Status[2]: PA RX skip symbol requirement
        - 0: PA RX does not require skip symbol insertion
        - 1: PA RX requires skip symbol insertion
    - Status[3]: copy from attribute PA_Local_TX_LCC_Enable, fixed to 0b0
        - 0: request to skip M-PHY Line Configuration states
- THibern8: M-PHY timing information (RX_Hibern8Time_Capability)
- TMinActivate: M-PHY timing information (RX_Min_ActivateTime_Capability)
- TAdvHibern8: M-PHY timing information (RX_Advanced_Hibern8Time_Capability)

- TAdvMinActivate: M-PHY timing information (RX_Advanced_Min_ActivateTime_Capability)
- RxAdvGranularity: M-PHY timing information (RX_Advanced_Granularity_Capability)
- MinRxTrailingClocks: (PA_MinRxTrailingClocks)
- RxPwmBurstClosureLength: (RX_PWM_BURST_Closure_Length_Capability)
- RxPwmG6G7SyncLength: M-PHY timing information (RX_PWM_G6_G7_SYNC_LENGTH_Capability)
- RxLsPrepareLength: M-PHY timing information (RX_LS_PREPARE_LENGTH_Capability)
- RxHsG1SyncLength: M-PHY timing information (RX_HS_G1_SYNC_LENGTH_Capability)
- RxHsG1PrepareLength: M-PHY timing information (RX_HS_G1_PREPARE_LENGTH_Capability)
- RxHsG2SyncLength: M-PHY timing information (RX_HS_G2_SYNC_LENGTH_Capability)
- RxHsG2PrepareLength: M-PHY timing information (RX_HS_G2_PREPARE_LENGTH_Capability)
- RxHsG3SyncLength: M-PHY timing information (RX_HS_G3_SYNC_LENGTH_Capability)
- RxHsG3PrepareLength: M-PHY timing information (RX_HS_G3_PREPARE_LENGTH_Capability)
- Unsupported or non-existing M-PHY capability attributes are reported with every bit set to '1'.

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	ESC_PA								EscParam_PA = PACP_BEGIN							
0	PACP_FunctionID = PACP_CAP_EXT1_ind															
0	THibern8								TMinActivate				Status			
0	TAdvHibern8								TAdvMinActivate				RxAdvGranularity			
0	MinRxTrailingClocks								RxPwmBurstClosureLength							
0	RxLsPrepareLength								RxPwmG6G7SyncLength							
0	RxHsG1PrepareLength								RxHsG1SyncLength							
0	RxHsG2PrepareLength								RxHsG2SyncLength							
0	RxHsG3PrepareLength								RxHsG3SyncLength							
0	Reserved															
0	Reserved															
0	CCITT CRC-16															

Figure 32 PACP_CAP_EXT1_ind

5.7.7.5 PACP_CAP_EXT2_ind

The PACP_CAP_EXT2_ind frame is shown in ***Figure 33***. It gathers all evolutionarily introduced capabilities defined from M-PHY specification revision 4.0 onwards. It shall be used in phase 5 of Link Startup Sequence before transmission of the PACP_CAP_EXT1_ind to notify the peer PA Layer of the local M-TX, M-RX, and PA Layer capabilities (see ***Section 5.7.8)***.

Legacy Devices prior to UniPro version 1.8 not being able to recognize PACP_CAP_EXT2_ind shall silently discard the reception of those PACP frames and proceed to PACP_CAP_EXT1_ind or PACP_CAP_ind reception.

The frame's fields shall be set as follows (source of the information given in parenthesis):

- MaxHS: Maximum HS gear, or zero if HS mode is unavailable (TX_HSGEAR_Capability, TX_HSMODE_Capability)

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

This field overrides the field with the same name that is found in PACP_CAP_ind frame.

- RxHsG4SyncLength: M-PHY timing information (RX_HS_G4_SYNC_LENGTH_Capability)
- RxHsG4PrepareLength: M-PHY timing information (RX_HS_G4_PREPARE_LENGTH_Capability)
- RxHsAdaptInitial: M-PHY timing information (RX_HS_ADAPT_INITIAL_Capability)
- RxHsAdaptRefresh: M-PHY timing information (RX_HS_ADAPT_REFRESH_Capability)
- Unsupported or non-existing M-PHY capability attributes are reported with every bit set to '1'.

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	ESC_PA								EscParam_PA = PACP_BEGIN							
0	PACP_FunctionID = PACP_CAP_EXT2_ind															
0	Reserved													MaxHS		
0	RxHsG4PrepareLength								RxHsG4SyncLength							
0	RxHsAdaptRefresh								RxHsAdaptInitial							
0	Reserved															
0	Reserved															
0	Reserved															
0	Reserved															
0	Reserved															
0	Reserved															
0	Reserved															
0	Reserved															
0	Reserved															
0	CCITT CRC-16															

Figure 33 PACP_CAP_EXT2_ind

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.7.7.6 PACP_EPR_ind

The PACP_EPR_ind frame is shown in ***Figure 34***. It does not have any parameters.

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	ESC_PA								EscParam_PA = PACP_BEGIN							
0	PACP_FunctionID = PACP_EPR_ind															
0	CCITT CRC-16															

Figure 34 PACP_EPR_ind

5.7.7.7 PACP_TEST_MODE_req

The PACP_TEST_MODE_req frame is shown in ***Figure 35***. It is used to set the peer Device in PHY testing mode. The frame does not have any parameters.

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	ESC_PA								EscParam_PA = PACP_BEGIN							
0	PACP_FunctionID = PACP_TEST_MODE_req															
0	CCITT CRC-16															

Figure 35 PACP_TEST_MODE_req

5.7.7.8 PACP_GET_req

The PACP_GET_req frame is shown in ***Figure 36***. This PACP frame shall be sent when a PA_LM_PEER_GET.req primitive is generated by the DME or when the PHY Test Feature requests it.

The frame's fields shall be set as follows:

- MIBattribute: defines the Attribute to be accessed in the peer Device
- GenSelectorIndex: the index required for certain Attributes

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	ESC_PA								EscParam_PA = PACP_BEGIN							
0	PACP_FunctionID = PACP_GET_req															
0	MIBattribute															
0	GenSelectorIndex															
0	CCITT CRC-16															

Figure 36 PACP_GET_req

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.7.7.9 PACP_GET_cnf

The PACP_GET_cnf frame is shown in ***Figure 37***. When the DME generate the PA_LM_PEER_GET.rsp primitive or when the PHY Test Feature requests it, the PA Layer shall send the PACP_GET_cnf frame.

The frame's fields shall be set as follows:

- ConfigResultCode: returns the result of the operation. The values taken by this field are defined in the context of the PA_LM_PEER_GET.rsp primitive in ***Table 8***
- MIBvalue: holds the value of the Attribute

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	ESC_PA								EscParam_PA = PACP_BEGIN							
0	PACP_FunctionID = PACP_GET_cnf															
0	ConfigResultCode								Reserved							
0	MIBvalue (high word)															
0	MIBvalue (low word)															
0	CCITT CRC-16															

Figure 37 PACP_GET_cnf

5.7.7.10 PACP_SET_req

The PACP_SET_req frame is shown in ***Figure 38***. This PACP frame shall be sent when a PA_LM_PEER_SET.req primitive is generated by the DME or when the PHY Test Feature requests it.

The frame's fields shall be set as follows:

- Cnf: defines the behavior of the receiving PA Layer
  - 0: no PACP_SET_cnf frame shall be sent
  - 1: a PACP_SET_cnf frame shall be sent to acknowledge the reception of the PACP_SET_req frame and to return the result of the operation
- T: defines the target Attribute type (AttrSetTypeType)
  - 0: NORMAL
  - 1: STATIC
- MIBattribute: defines the Attribute to be accessed in the peer Device
- GenSelectorIndex: the index required for certain Attributes
- MIBvalue: holds the value which the Attribute shall take

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	ESC_PA								EscParam_PA = PACP_BEGIN							
0	PACP_FunctionID = PACP_SET_req															
0	Cnf	T	Reserved													
0	MIBattribute															
0	GenSelectorIndex															
0	MIBvalue (high word)															
0	MIBvalue (low word)															
0	CCITT CRC-16															

Figure 38 PACP_SET_req

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.7.7.11 PACP_SET_cnf

The PACP_SET_cnf frame is shown in ***Figure 39***. When the DME generates the PA_LM_PEER_SET.rsp primitive, the PA Layer shall send the PACP_SET_cnf frame.

The frame's fields shall be set as follows:

- ConfigResultCode: returns the result of the operation. The values taken by this field are defined in the context of the PA_LM_PEER_SET.rsp primitive in ***Table 8***

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	ESC_PA								EscParam_PA = PACP_BEGIN							
0	PACP_FunctionID = PACP_SET_cnf															
0	ConfigResultCode								Reserved							
0	CCITT CRC-16															

Figure 39 PACP_SET_cnf

5.7.7.12 PACP_TEST_DATA

The PACP_TEST_DATA frame is shown in ***Figure 40***. There are four variations of the basic PACP_TEST_DATA frame, each having a different FunctionID and different payload length. The frame is used in PHY testing as explained in ***Section 5.7.15.***

The TestPattern field carries the PHY test pattern. The transmitted patterns are defined in ***Section 5.7.15***. The received patterns are undefined since the receiver shall only check the header, the length, and the checksum while ignoring the content of TestPattern.

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	ESC_PA								EscParam_PA = PACP_BEGIN							
0	PACP_FunctionID = PACP_TEST_DATA															
0	TestPattern[0]								TestPattern[1]							
0	TestPattern[2]								TestPattern[3]							
0	...								...							
0	TestPattern[n-2]								TestPattern[n-1]							
0	CCITT CRC-16															

Figure 40 PACP_TEST_DATA

5.7.8 Link Startup Sequence

The overall view of the Link Startup Sequence is covered in ***Section 5.6.3.*** This part describes the M-PHY-specific details of the sequence including the Data Lane discovery, realignment, and capability exchange parts.

During this sequence, the transceivers are initialized to the default configuration. Then the connected M-PHY Lanes are identified and enumerated to produce a mapping between physical Lanes and logical Lanes. Finally, the peer PA Layers exchange capability information with each other.

After the successful completion of the sequence, the Link capabilities (PA_MaxRxHSGear and PA_MaxRxPWMGear), the number of connected Lanes (PA_ConnectedTxDataLanes and PA_ConnectedRxDataLanes), and the logical Lane mapping (PA_LogicalLaneMap) are updated.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.7.8.1 Initialization Phase

A PA_LM_LINKSTARTUP.req triggers the following preparations:

- All available M-TX shall be requested to exit M-PHY HIBERN8 state.
- LINE-RESET shall be issued on all TX Lanes to reset any misconfigured basic MCs and peer M-RXs.

After this, all available M-TXs are in PWM-G1 and local PA Layer shall move to phase 0. The local M-RX is in HIBERN8 state, requiring M-LANE-HIBERN8Exit followed by M-CTRL-LINE-RESET from the peer in order to enter PWM-G1. A LINE-RESET received on M-RX shall not disrupt the Link Startup Sequence. The PA Layer shall be sensitized to listen to TRG_UPR0 reception in PWM-G1 mode an all available Lanes. A timer, PA_LINKSTARTUP_TIMER, shall be started upon entering phase 0 to provide a timeout signal if the procedure does not complete within a specified time, 100 ms with an accuracy of +/-10%. The PA_LINKSTARTUP_TIMER shall be restarted upon receiving the first TRG_UPR0 from the peer Device following the local PA_LM_LINKSTARTUP.req. If the timeout is triggered, the PA Layer shall abort the Link Startup Sequence immediately and terminate any active burst.

5.7.8.2 Data Lane Discovery (Phases 0 and 0b)

The Link Startup Sequence starts with the discovery of the connected M-PHY Data Lanes. In the example shown in ***Figure 41***, both Devices have four TX Data Lanes and four RX Data Lanes, but not all available Data Lanes are connected, and no assumptions are made regarding which TX Data Lane is connected to which RX Data Lane of the peer Device. The only constraint is that at least one Data Lane shall be connected in each direction (because UniPro does not support unidirectional Links).

In the first phase, both Devices shall repeatedly send TRG_UPR0 triggers on all available Data Lanes. The TRG_UPR0 trigger data structure is defined in ***Section 5.7.6***, but the data structure contains a field that shall be set by every transmitting Data Lane to hold that Data Lane's fixed, internal “Physical Data Lane number” (PL# in ***Figure 41***).

The peer Device shall update its local PeerTxConnectedLanesMask register to reflect the Data Lanes on which triggers have been received. This information is used in subsequent phases of this sequence.

A Device shall continue transmitting TRG_UPR0 Messages until it receives a TRG_UPR0 Message from the peer Device, after which it shall send two extra TRG_UPR0 Messages before proceeding to the next phase.

The following steps describe Initialization Phase in detail:

1. Request all available M-TX to exit from HIBERN8.
2. Wait for a period of PA_TActivate to wake-up peer M-RXs from HIBERN8 (see ***[MIPI02]***).
3. Issue a LINE-RESET on all available TX Lanes and wait for PA_SaveConfigTime

The following steps describe Phase 0 and Phase 0b operation in detail:

4. Begin burst (includes the transmission of the deskew pattern) on all available TX Lanes.
5. Send a TRG_UPR0 on all available TX Lanes.
6. End burst on all available TX Lanes.
7. Wait for a period of PA_TActivate to wake-up M-RXs from HIBERN8 (see ***[MIPI02]***).
8. If a TRG_UPR0 has been received then continue from step 9, else continue from step 4.
9. Begin burst (includes the transmission of the deskew pattern) on all available TX Lanes.
10. Send two additional TRG_UPR0s on all available TX Lanes.
11. Exit to Phase 1.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

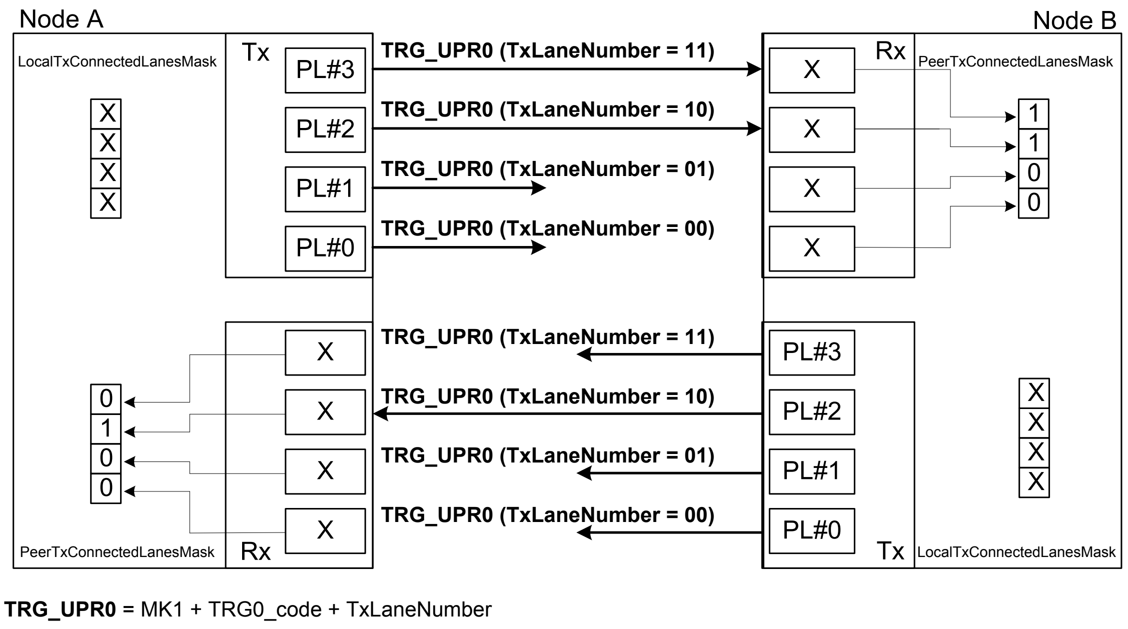

Node A
Node B
LocalTxConnectedLanesMask
Tx
PL#3
PL#2
PL#1
PL#0
TRG_UPR0 (TxLaneNumber = 11)
TRG_UPR0 (TxLaneNumber = 10)
TRG_UPR0 (TxLaneNumber = 01)
TRG_UPR0 (TxLaneNumber = 00)
Rx
PeerTxConnectedLanesMask
X
1
1
0
0
0
1
0
0
PeerTxConnectedLanesMask
Rx
Tx
LocalTxConnectedLanesMask
TRG_UPR0 = MK1 + TRG0_code + TxLaneNumber

Figure 41 M-PHY Lanes Discovery

5.7.8.3 Data Lane Realignment (Phases 1 and 2)

In these phases, both Devices shall repeatedly send TRG_UPR1 triggers on all available Data Lanes. The TRG_UPR1 trigger structure is defined in ***Section 5.7.6***, but the data structure contains a field that shall be set on each available Data Lane to the PeerTxConnectedLanesMask value set in the previous phase.

Once the peer Device receives a TRG_UPR1 trigger, it shall use the PeerTxConnectedLanesMask information to update its LocalTxConnectedLanesMask register. This register assigns a Logical Data Lane number (LL#) to each connected Physical Data Lane (PL#).

A Device shall continue transmitting TRG_UPR1 Messages until it receives a TRG_UPR1 Message, after which it shall send two extra TRG_UPR1 Messages before proceeding to the next phase.

The following steps describe Phase 1 and Phase 2 operation in detail:

1. Send a TRG_UPR1 on all available TX Lanes.
2. If a TRG_UPR1 has been received then continue from Step 4.
3. Continue from Step 1.
4. Send two additional TRG_UPR1s on all available TX Lanes.
5. Reconfigure all available M-TXs and M-RXs to use single-Lane PWM-G1 Gear on future logical Lane #0. Unconnected or inactive Lanes are configured to enter HIBERN8. The new configuration does not become effective until the end of Phase 4, when the current burst is closed.
6. Exit to Phase 3.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

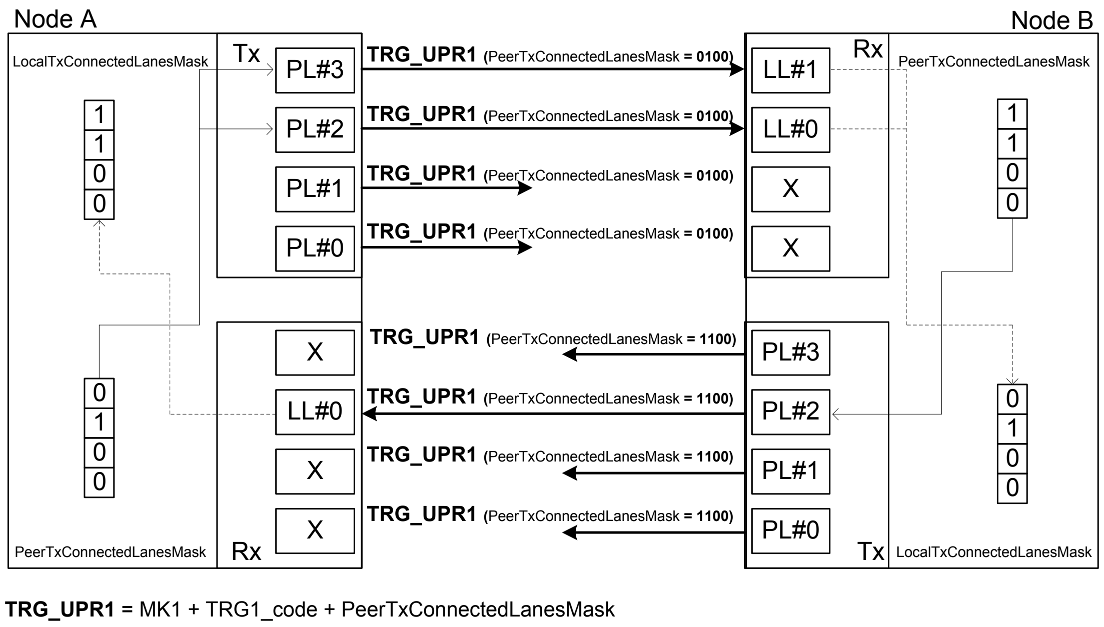

Node A
Node B
LocalTxConnectedLanesMask
Tx
PL#3
PL#2
PL#1
PL#0
TRG_UPR1 (PeerTxConnectedLanesMask = 0100)
LL#1
LL#0
X
Rx
PeerTxConnectedLanesMask
1 1 0 0
X
LL#0
TRG_UPR1 (PeerTxConnectedLanesMask = 1100)
0 1 0 0
PeerTxConnectedLanesMask
Rx
Tx
LocalTxConnectedLanesMask
TRG_UPR1 = MK1 + TRG1_code + PeerTxConnectedLanesMask

Figure 42 Repeated Transmission of TRG_UPR1 Messages

5.7.8.4 Link Startup Sequence Termination (Phases 3 and 4)

In the final phases, both ends of connected PHY Data Lanes have matching Logical Data Lane numbers. Each Device shall update its PA_ConnectedTxDataLanes and PA_ConnectedRxDataLanes Attributes to reflect how many connected Data Lanes were discovered.

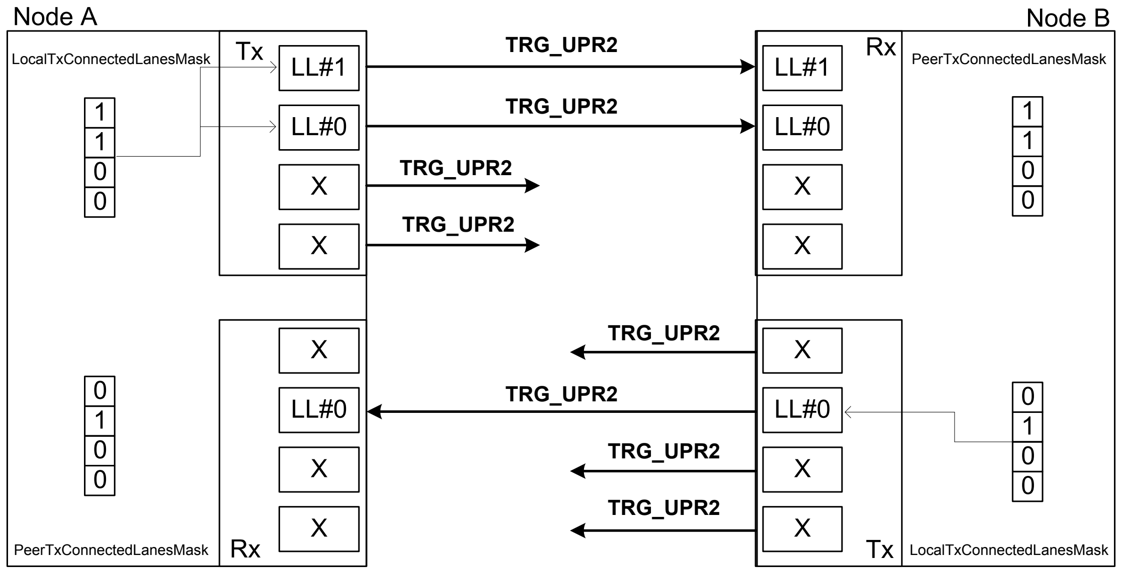
Note:

> *Because the discovery process is handled autonomously, and considered to be highly robust, information about which physical Data Lanes are connected and the interconnect topology are not made available to the Application in a normative way. In fact, both Physical and Logical Data Lane numbers as described in this Link Startup Sequence are invisible to higher protocol layers and to the Application.*

In these phases, TRG_UPR2 Messages (with no special payload) shall be repeatedly transmitted over all available Data Lanes until a TRG_UPR2 Message is received by the peer Device, after which the Device shall send two more TRG_UPR2 Messages over all available Data Lanes. These phases essentially serve to terminate previous phases in a robust and orderly manner.

The following steps describe Phase 3 and Phase 4 operation in detail:

1. Send a TRG_UPR2 on all available TX Lanes
2. If a TRG_UPR2 has been received then continue from Step 4
3. Continue from Step 1
4. Send two additional TRG_UPR2s on all available TX Lanes
5. Close the burst (which activates the new Lane settings applied during configuration in Phase 2)
6. Update PA_LogicalLaneMap, and begin using the logical Lane numbering
7. M-TXs and M-RXs on unconnected Lanes shall be configured to the UNPOWERED state
8. Exit to Capability Phase



TRG_UPR2 = MK1 + TRG2_code

Figure 43 Repeated Transmission of TRG_UPR2 Messages

5.7.8.5 Capability Exchange

After finishing Phase 4 of the Link Startup Sequence, the PA Layer shall start a Burst on logical Lane #0 and transmit its capabilities and the local version information to the peer Device using PACP_CAP_EXT2_ind (see ***Section 5.7.7.5***), PACP_CAP_EXT1_ind (see ***Section 5.7.7.4),*** and PACP_CAP_ind (see ***Section 5.7.7.3)*** in this order, using the PHY configuration described in Step 5 of ***Section 5.7.8.3.***

If PACP_CAP_EXT1_ind frame is received, the PA Layer shall configure M-PHY attribute TX_LCC_Enable = “NO” on the activated M-TX (logical Lane #0), and ignore a potential reception of a M-CTRL-LCCReadStatus.indication primitive from the M-RX. When the PA Layer receives a PACP_CAP_ind frame it shall perform capability downgrading as described in ***Section 5.7.9.*** using information from PACP_CAP_ind, PACP_CAP_EXT1_ind and, if received, from PACP_CAP_EXT2_ind. The OMC related M-PHY MC capability attributes MC_*_Capability remain undefined and will not be factored in during the subsequent downgrading process. Capability Management as described in ***Section 5.7.9*** shall be performed assuming a direct electrical connection on inbound and outbound Link.

If PACP_CAP_EXT2_ind frame is not received while peer version info indicates that the peer supports UniPro v1.8, then the PA Layer shall time out the LINK-STARTUP procedure.

If PACP_CAP_EXT1_ind frame is not received before the reception of PACP_CAP_ind, then the PA Layer shall time out the LINK-STARTUP procedure.

Once receiving PACP_CAP_ind, the PA Layer shall close the Burst.

The peer Device’s version information received in PACP_CAP_ind is stored in PA_RemoteVerInfo. This finishes the Link Startup Sequence. The M-PHY Link is now in the default Power Mode, i.e. PWM-G1, 1-Lane.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.7.9 Capability Management

Local and peer M-TX and M-RX may differ in capabilities. To simplify capability checking, the PA Layer forms a common capability value for the whole inbound Link beginning from the peer M-TX and ending with the local M-RX. Equations for calculating the common capability values are listed later in this section. This capability downgrading is automatically performed at the end of the Link Startup Sequence (see ***Section 5.7.8.5)***.

The following capabilities shall be collected to the PA Layer Attributes and downgraded based on the local and peer information (M-PHY MODULE Attributes prefixed with "peer" are received in the Link Startup Sequence via a PACP_CAP_ind frame):

```
if (RX_HSMODE_Capability == TRUE) {
   PA_MaxRxHSGear = min(RX_HSGEAR_Capability, peer_TX_HSGEAR_Capability);
}
else
{
   PA_MaxRxHSGear = 0;
}
PA_MaxRxPWMGear = min(RX_PWMGEAR_Capability, peer_TX_PWMGEAR_Capability);
PA_RxHSUnterminationCapability = RX_HS_Unterminated_Capability
      && peer_TX_HS_Unterminated_LINE_Drive_Capability;
PA_RxLSTerminationCapability = RX_LS_Terminated_Capability
      && peer_TX_LS_Terminated_LINE_Drive_Capability;
PA_SleepNoConfigTime >= max(TX_Min_SLEEP_NoConfig_Time_Capability,
      peer_RX_Min_SLEEP_NoConfig_Time_Capability);
PA_StallNoConfigTime >= max(TX_Min_STALL_NoConfig_Time_Capability,
      peer_RX_Min_STALL_NoConfig_Time_Capability);
PA_SaveConfigTime >= max(TX_Min_SAVE_Config_Time_Capability,
      peer_RX_Min_SAVE_Config_Time_Capability);
TX_Min_SLEEP_NoConfig_Time = max(TX_Min_SLEEP_NoConfig_Time_Capability,
      peer_RX_Min_SLEEP_NoConfig_Time_Capability);
TX_Min_STALL_NoConfig_Time = max(TX_Min_STALL_NoConfig_Time_Capability,
      peer_RX_Min_STALL_NoConfig_Time_Capability);
```

Derived values for PA_SleepNoConfigTime, PA_StallNoConfigTime, and PA_SaveConfigTime constitute a lower bound. An implementation may add margin to this timing.

Since the automatic downgrading process cannot take into account a Basic OMC or the restrictions from the physical interconnection, the Application has the ability to overwrite the calculated capabilities through these Attributes.

While the capability downgrading is done only once, at the end of the Link Startup Sequence, the capability checking is performed whenever there is a request to change the Power Mode. The PA Layer shall verify the inbound Link capabilities. For example, if the local Device wants to change parameters of the outbound Link, the peer PA Layer is responsible for verifying the Link capabilities.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

A PA Layer that receives a PACP_CAP_EXT1_ind shall extend PA Layer attribute collection and downgrading of the following attributes:

```
PA_TxMk2Extension = Status[0];
PA_PeerScrambling = Status[1];
PA_TxSkip = Status[2];
PA_Peer_TX_LCC_Enable: Status[3], fixed to 0;
if ((peer_RX_Advanced_Granularity_Capability[0] == 1) &&
   (TX_Advanced_Granularity_Capability[0] == 1))
{
   Scale = (2^(peer_RX_Advanced_Granularity_Capability[2:1])) /
      (2^(TX_Advanced_Granularity_Capability[2:1]));
   if (Scale >= 1)
   {
      PA_Granularity = TX_Advanced_Granularity_Capability[2:1]+2;
      PA_TActivate = peer_ RX_Advanced_Min_ActivateTime_Capability * Scale;
      PA_Hibern8Time = max(TX_Advanced_Hibern8_Capability,
            peer_RX_Advanced_Hibern8_Capability * Scale);
   } else {
      PA_Granularity = peer_RX_Advanced_Granularity_Capability[2:1]+2;
      PA_TActivate = peer_RX_Advanced_Min_ActivateTime_Capability;
      PA_Hibern8Time = max(TX_Advanced_Hibern8_Capability / Scale,
            peer_RX_Advanced_Hibern8_Capability);
   }
} else { //compatibility mode
   PA_TActivate = peer_ RX_Min_ActivateTime_Capability;
   PA_Hibern8Time = max (TX_Hibern8Time_Capability,
         peer_RX_Hibern8Time_Capability);
   PA_Granularity = 6 (100us;)
}
PA_TxTrailingClocks = peer_MinRxTrailingClocks;
PA_TxHsG1SyncLength = peer_RX_HS_G1_SYNC_LENGTH;
PA_TxHsG1PrepareLength = peer_RX_HS_G1_PREPARE_LENGTH;
PA_TxHsG2SyncLength = peer_RX_HS_G2_SYNC_LENGTH;
PA_TxHsG2PrepareLength = peer_RX_HS_G2_PREPARE_LENGTH;
PA_TxHsG3SyncLength = peer_RX_HS_G3_SYNC_LENGTH;
PA_TxHsG3PrepareLength = peer_RX_HS_G3_PREPARE_LENGTH;
```

A PA Layer that receives a PACP_CAP_EXT2_ind shall extend PA Layer attribute collection and downgrading of the following attributes:

```
PA_TxHsG4SyncLength = peer_RX_HS_G4_SYNC_LENGTH;
PA_TxHsG4PrepareLength = peer_RX_HS_G4_PREPARE_LENGTH;
PA_PeerRxHsAdaptRefresh = peer_RX_HS_ADAPT_REFRESH:
PA_PeerRxHsAdaptInitial = peer_RX_HS_ADAPT_INITIAL;
```

The parameters RxPwmG6G7SyncLength, RxPwmPrepareLength, and RxPwmBurstClosureLength received with PACP_CAP_EXT1_ind are not mapped into the PA Layer attribute space as they do not need to be maintained within UniPro. Instead, they are statically copied to the relevant M-TX attributes TX_PWM_G6_G7_SYNC_LENGTH, TX_LS_PREPARE_LENGTH, and TX_PWM_BURST_Closure_Extension, and downgraded there if necessary (existence of the M-PHY attributes shall be ensured before setting them). If PACP_CAP_EXT1_ind is not received, configuration of the above M-TX attributes is left to the Application.

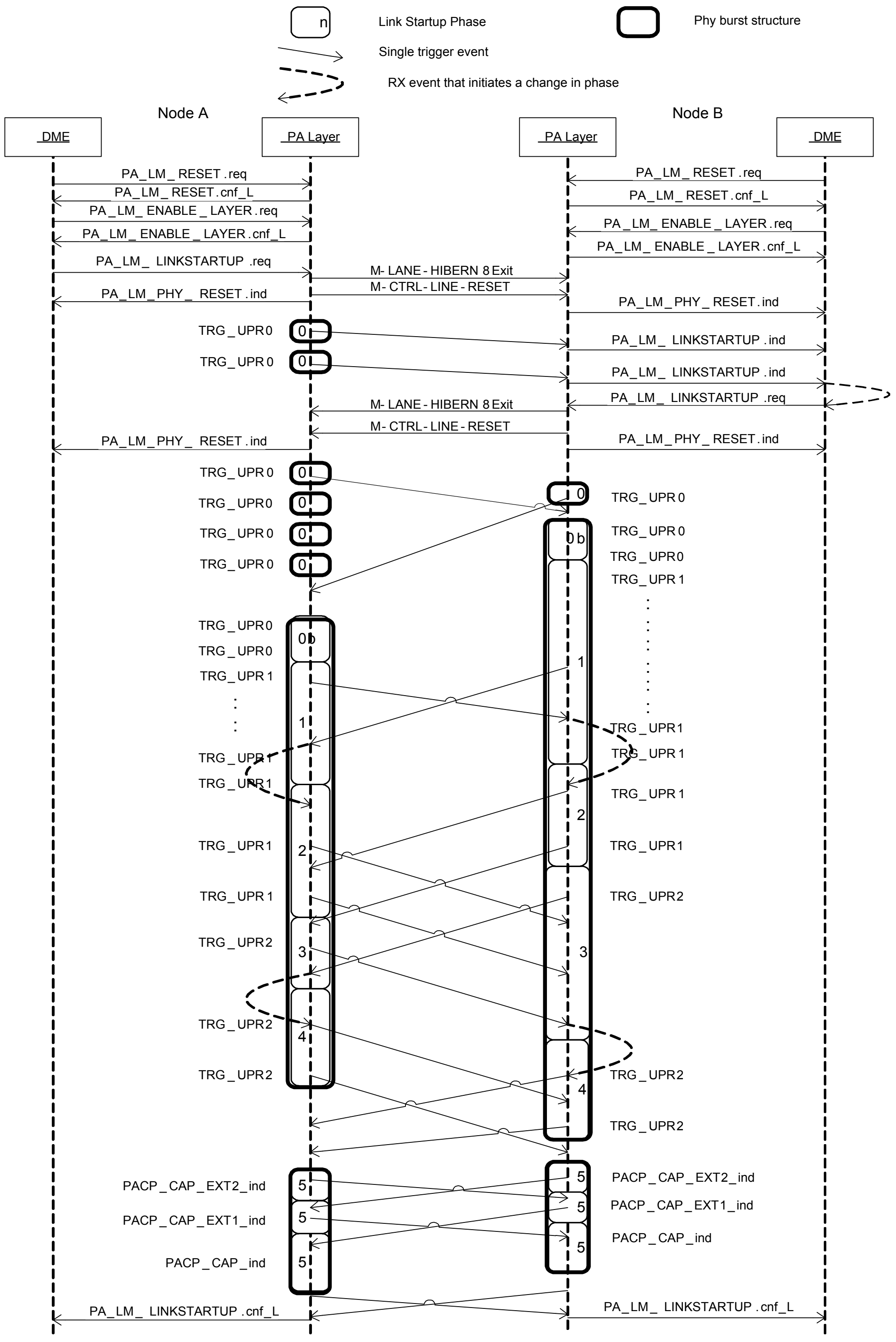
Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

When receiving a PACP_CAP_EXT1_ind, the PA Layer shall automate the programming of M-PHY timing attributes TX_HS_SYNC_LENGTH and TX_HS_PREPARE_LENGTH dynamically with every Power Mode change from the set of parameters PA_TxHsG<x>SyncLength and PA_TxHsG<x>PrepareLength, where <x> denotes the TxHsGear to be set.

The Application handles remaining M-PHY Attributes (see ***Section 5.7.2)*** which are not covered by the automated capability management procedure.

Figure 44 and ***Figure 45*** summarize phase 0 to 5 of the Link Startup sequence initiated from one side or from both sides of the Link.

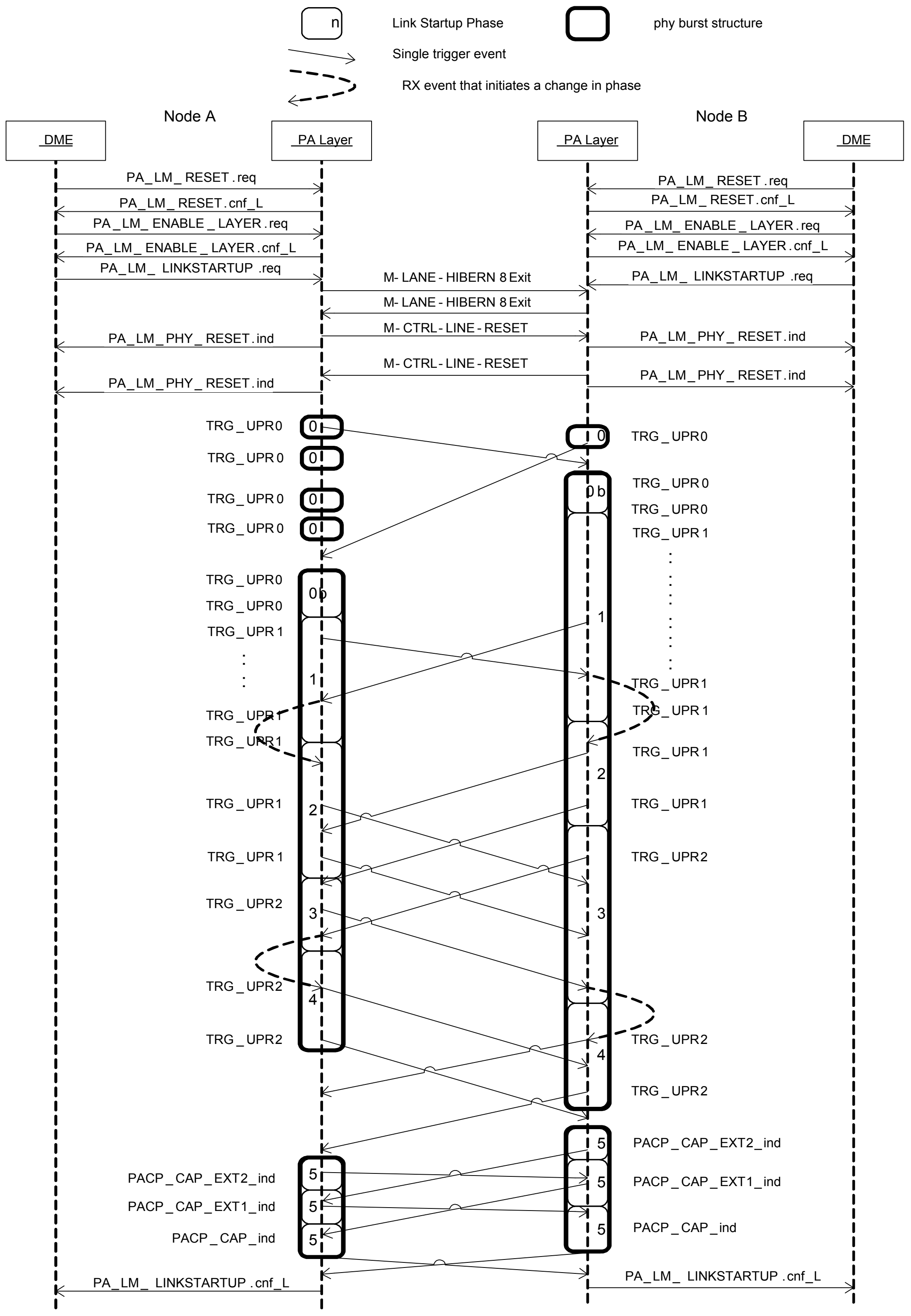
Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential



2574

Figure 44 Single Ended Link Startup Initiation (Informative)

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential



2575

Figure 45 Link Startup Initiation from Both Ends (Informative)

5.7.10 (Re-)Initialization

The PA Layer offers two mechanisms to reinitialize the Link. The less severe mechanism, deskew pattern injection, inserts additional deskew patterns to all active Lanes in order to fix any M-PHY symbol or PA Symbol synchronization issues. The insertion adds a one-symbol delay to the transmission, and may be requested at any point. The mechanism may be triggered by the DL Layer via the PA_LANE_ALIGN primitive (see ***Section 5.3.2.6)***. This request is ignored in SLEEP_STATE since there is no active burst on-going.

The more severe mechanism involves reinitializing the power configuration of both inbound and outbound Links. The DL Layer can request this mechanism via a PA_INIT primitive (see ***Section 5.3.2.1)***. This re-initialization utilizes the Link Configuration procedure beginning with the first retry (see step 2 in ***Section 5.7.12.6)***, using the current power mode and does not exchange the PowerModeUserData. From the time the DL Layer is paused, all detailed steps of the Link Configuration, like handling communication errors, and performing M-PHY configuration shall also be done for PA-INIT. A summary of the procedure is as follows:

1. Begin a Burst, if one is not already existing, on the active Lanes; otherwise, else transmit a deskew pattern on an existing Burst. If communicating to peer Devices implementing a version of UniPro earlier than v1.8, begin a dummy Burst on inactive connected Lanes.
2. Send a PACP_PWR_req frame containing the Power Mode configuration used before PA_INIT was issued. The UserDataValid flag is set to '0' as valid PAPowerModeUserData is not exchanged. The LINE-RESET request flag is set to '0'.
3. PACP_REQUEST_TIMER is set to PA_PACPReqTimeout and started.
4. Wait for a PACP_PWR_cnf frame from the peer Device, or for PACP_REQUEST_TIMER to timeout. If the PACP_PWR_cnf frame was received correctly, go to Step 10.
5. Issue LINE-RESET to reset outbound M-TX and M-RX to PWM-G1.
6. Begin a burst on logical Lane 0, and a dummy burst on other connected Lanes.
7. Send a PACP_PWR_req frame containing the Power Mode configuration used before PA_INIT was issued. The UserDataValid flag is set to '0' as valid PAPowerModeUserData is not exchanged.

   If either outbound or inbound direction is configured to HS-G4, then the Adapt Field shall be configured with PA_AdaptAfterLRSTInPA_INIT. In any other case the Adapt field is set to 2'b11 (No ADAPT).

   The LINE-RESET request flag is set to request the peer Device issue LINE-RESET on the inbound Link.
8. PACP_REQUEST_TIMER is set to PA_PACPReqTimeout + $T_{LINE\text{-}RESET}$ and started
9. Wait for a PACP_PWR_cnf frame from the peer Device, or for PACP_REQUEST_TIMER to timeout.
10. Configure M-TX and M-RX.
11. End burst to activate the new PHY configuration.
12. Reply to the DL Layer via a PA_INIT.cnf_L primitive.

If there is a timeout at Step 9, the Link is considered unusable and the reinitialization is aborted. The PA Layer replies to the DL Layer with a PA_INIT.cnf_L primitive, even though the reinitialization failed. The PA Layer shall be ready for a possible retry by the DL Layer.

If the reinitialization succeeds, the Power Mode is reinitialized using the active settings, and a misconfigured M-RX receives correct settings.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.7.11 Lane Synchronization

PA Layer RX shall lock the PA Layer symbol synchronization on each Lane every time the deskew pattern is received. M-RX handles the M-PHY symbol synchronization by locking to MK0, which is the high part of the deskew pattern.

In the multi-Lane usage, the PA Layer shall synchronize the multiple incoming Lanes because of the skew between the Lanes and because of the independent RX clocks of the M-RXs. The Lane synchronization happens by aligning the incoming deskew patterns that are sent in parallel from the transmitter. Depending of the deskew requirements, this may require, for example, creating shallow deskew-FIFOs within the PA Layer RX.

The principle operation of the aligner shall be as follows:

1. When the burst starts, Lanes are not synchronized
2. Per each active Lane not having detected the deskew pattern, PA Layer RX discards PA symbols until it detects the deskew pattern
3. Once all active Lanes have received the deskew pattern, Lane synchronization is successful and PA Layer RX begins to consume PA symbols
4. If PA Layer RX detects a new deskew pattern, the Lane synchronization for all Lanes is lost and the operation returns to Step 2.

In case a deskew pattern is lost and the PA Layer RX cannot acquire the Lane synchronization, the deskew-FIFOs of the synchronized Lanes overflow, which is the intended behavior. The PA Layer RX shall wait until the peer Device transmits another deskew pattern and shall lock on to this second pattern.

At a minimum, the PA Layer shall be able to operate correctly with a Lane-to-Lane skew of four PA Symbols at its PHY_SAP, see ***Table 22*** for details.

5.7.12 Link Configuration Procedure

Either Device can change the Power Mode of a Link by assigning a new power configuration. The configuration includes UniPro Power Mode, M-PHY-specific Attributes, e.g. GEARs, and Lane count information. The configuration may have separate settings for forward, reverse, or both directions. The Link Configuration procedure, as specified in this section and illustrated in ***Figure 46***, covers all M-PHY-specific details. The enter hibernation and exit hibernation procedures are covered in ***Section 5.7.13.***

Power Mode and Lane-count configuration are set via a PA_LM_SET.req primitive to the related Attributes (see ***Section 5.8***). Attributes may be set in any order since the actual Link configuration procedure begins only after setting PA_PWRMode. However, all Attributes shall be set before activating the Power Mode change procedure.

An Application shall change PA_HSSeries only when a Link is configured to Slow_Mode or SlowAuto_Mode.

The configuration procedure shall be activated by setting PA_PWRMode via a PA_LM_SET.req primitive. The Attribute contains fields for both directions.

The PA Layer shall first check that the given configuration is possible and then replies with the PA_LM_SET.cnf_L primitive where the parameter indicates if the configuration change can proceed. Note, that a reception of a successful PA_LM_SET.cnf_L primitive does not mean that the configuration has been applied, only that the request was accepted by the PA Layer. The PA Layer shall later indicate the completion of the configuration change via the PA_LM_PWR_MODE_CHANGED.ind primitive, where the parameter reports the actual result of the operation.

After accepting the Power Mode change request, the PA Layer shall first request permission from the DL Layer via the PA_DL_PAUSE.ind primitive. Once the DL Layer responds with a PA_DL_PAUSE.rsp_L primitive, the PA Layer may begin changing the configuration. When the configuration has been changed, or the request has been cancelled, the PA Layer shall notify the DL Layer via a PA_DL_RESUME.ind primitive and then shall signal the DME with PA_LM_PWR_MODE_CHANGED.ind (PWR_LOCAL, PWR_BUSY, or PWR_ERROR_CAP). This last primitive completes the Link configuration and makes the PA Layer ready for the next configuration request. A successful request that originates from the peer Device is signaled to the DME with PA_LM_PWR_MODE_CHANGED.ind (PWR_REMOTE). Please refer also to ***Section 5.7.12.6*** for error handling induced by Link errors.

5.7.12.1 Error Handling

A configuration request may be cancelled due to the following reasons.

- A race condition between two requests (see ***Section 5.7.12.1.1)***.
- The requested configuration is invalid and cannot be applied (see ***Section 5.7.12.1.2)***.
- An unrecoverable error in the communication (see ***Section 5.7.12.1.3)***

5.7.12.1.1 Concurrency Resolution

The PA Layer has to arbitrate requests because both Devices might try to change a Link configuration at the same time. As a result, the PA Layer may reject one or both requests depending on the relative timing of those requests. The failure notification is sent to the DME either via the return parameter of the PA_LM_SET.cnf_L primitive or via the PA_LM_PWR_MODE_CHANGED.ind primitive depending upon when the concurrency was detected. In every case, the end Power Mode configuration, after a concurrency issue, shall be either the previous configuration, i.e. both requests were rejected, or a new configuration, i.e. one request was rejected and one request was accepted.

The resolution rules are:

1. A local request shall be rejected when the PA Layer is processing a local request or a peer request from the peer Device. In this case, the conflict is detected when the DME issues a PA_LM_SET.req primitive to PA_PWRMode. The DME is notified with a PA_LM_SET.cnf_L primitive having a parameter BUSY.
2. A peer request shall be answered with PACP_PWR_cnf(BUSY) or alternatively be left unconfirmed, when the PA Layer is processing a local request, and the DevID field of the peer request is larger than or equal to X, where X is N_DeviceID if N_DeviceID_valid is TRUE, otherwise X is 0x80.

Note:

The second rule causes concurrent requests from both Devices to be rejected when both local and peer DeviceIDs have not yet been assigned. If at least one of the DeviceIDs have been set, then the equation guarantees that one of two on-going requests is accepted.

5.7.12.1.2 Invalid Configuration Detection

If the requested configuration is invalid, e.g. requires a higher speed than is supported, the request may be cancelled in the following phases:

1. PA_LM_SET.req to a configuration Attribute. If the value exceeds the defined range, the PA Layer shall respond with PA_LM_SET.cnf_L (INVALID_MIB_ATTRIBUTE_VALUE).
2. PA_LM_SET.req to PA_PWRMode. If the given configuration is determined to be impossible, e.g. due to conflicting configuration parameters, the PA Layer shall respond with PA_LM_SET.cnf_L (INVALID_MIB_ATTRIBUTE_VALUE).
3. If the peer Device determines the requested configuration is invalid, the Status field of the PACP_PWR_cnf frame shall be set to PWR_ERROR_CAP.
4. If the local PA Layer receives the PACP_PWR_cnf frame before Step 3 is taken (see ***Section 5.7.12.6)***, the local PA Layer shall pass PA_LM_PWR_MODE_CHANGED.ind (PWR_ERROR_CAP) to the DME. In this case, the Link configuration is not changed.
5. If the local PA Layer receives the PACP_PWR_cnf frame after Step 3 is taken (see ***Section 5.7.12.6***), the local PA Layer shall pass PA_LM_PWR_MODE_CHANGED.ind (PWR_FATAL_ERROR) to the DME. In this case, the Link configuration is lost.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.7.12.1.3 Communication Errors and Retries

The Link Configuration Procedure uses the Link to exchange PACP frames (see ***Section 5.7.7)*** with the peer Device and is thus vulnerable to bit errors. The requesting Device is responsible for retransmitting the request after a timeout. For details, see ***Section 5.7.12.*** In case of a failure, the end configuration of the Link is indeterminate, possibly even inoperable. However, the PA Layer shall be in a state where it can accept new Power Mode change requests from the DME.

5.7.12.2 Detailed Link Configuration Operation

The detailed Link configuration operation depends on the UniPro version(s) used by the Devices in the Link.

5.7.12.2.1 With UniPro Version 1.8 Only

If neither of the Devices in the Link are legacy UniPro Versions with version numbers lower than v1.8, this section describes the detailed operation of the Link configuration. In any other case, please refer to ***Section 5.7.12.2.2***.

The configuration is set via PA_HSSeries, PA_TxGear, PA_TxTermination, PA_Scrambling, PA_TxHsAdaptType, PA_RxGear, PA_RxTermination, PA_ActiveTxDataLanes, and PA_ActiveRxDataLanes. In addition to the M-PHY MODULE configuration, the PA Layer can transmit user-data, e.g. L2 timer values, to the peer DME. This can be supplied via PA_PWRModeUserData.

The basic principle of the Power Mode change is depicted in ***Figure 46***. Attributes are assumed to be set before changing the Power Mode. The local PA Layer first checks the request against the inbound Link capabilities, in this case from peer to local. After replying to the DME that the request was accepted, the local PA Layer requests permission from the DL Layer. Then the local PA Layer shall send a PACP_PWR_req frame on the active TX Lanes.

When the peer PA Layer receives a valid PACP_PWR_req frame, it shall first check the request against its inbound Link capabilities, in this case from local to peer. The peer PA layer shall then save a copy of the PACP_PWR_req frame fields, DevID, Adapt, Flags, TxMode, TxLane, TxGear, RxMode, RxLane, and RxGear, to add to the PACP_PWR_cnf. Then the peer PA Layer shall request permission from the DL Layer. Next, the peer PA Layer shall exchange PAPowerModeUserData, e.g., L2 timer values, with its DME. The last 2 parameters of PAPowerModeUserData received from the DME shall be replaced with the saved fields, DevID, Adapt, Flags, TxMode, TxLane, TxGear, RxMode, RxLane, and RxGear.

If the request was accepted, the peer PA Layer shall configure the PHY Layer with the parameters of the request.

After completing these steps, the peer PA Layer shall send the PACP_PWR_cnf frame. When the local PA Layer receives a valid PACP_PWR_cnf frame, it shall first check the Status field. If the Status field contains PWR_OK, PAPowerModeUserData is passed to the local DME. The last two parameters of the PACP_PWR_cnf frame can be passed to the DME with the remaining ten PAPowerModeUserData parameters such that a total of 24 bytes are passed to the DME. Passing this data has no affect as the DME silently discards the PAPowerModeUserData. The local PHY Layer shall be configured with the requested parameters. The local PA Layer shall close the burst on the outbound Link.

If the Power Mode Change handshake is successful and the number of configured active outbound Lanes exceeds the number of currently active Lanes, then inactive Lanes will be taken out of the M-PHY Hibern8 state. Those M-TX Modules need to be in the SAVE state for a period of PA_Tactivate. In this kind of Power Mode Change request, it is required that UniPro waits for PA_Tactivate and the PA_SaveConfigTime timer to expire before issuing an M-LANE-PREPARE.request or an M-LANE-AdaptStart.request. After the timers expire, the PA Layer shall request the subsequent burst or Adapt simultaneously across all Lanes within a given sub-Link.

The peer PA Layer shall close the burst on the other Link when detecting the end of burst on its inbound Link. Consequently, both directions of the Link have the new configuration activated. Both sides of the Link shall notify their respective DL Layers about the completion of the procedure. The local and peer PA Layer shall notify their respective DMEs with a PA_LM_PWR_MODE_CHANGED.ind primitive with the PowerChangeResultCode parameter containing PWR_LOCAL and PWR_REMOTE, respectively. If the Status field does not contain PWR_OK, the local PA Layer shall close the burst on the outbound Link, and notify the local DME with a PA_LM_PWR_MODE_CHANGED.ind primitive with the PowerChangeResultCode parameter containing the value of the Status field of the PACP_PWR_cnf frame. In both cases, when the DMEs have been notified, the Power Mode change is concluded, and both PA Layers are ready for subsequent operations.

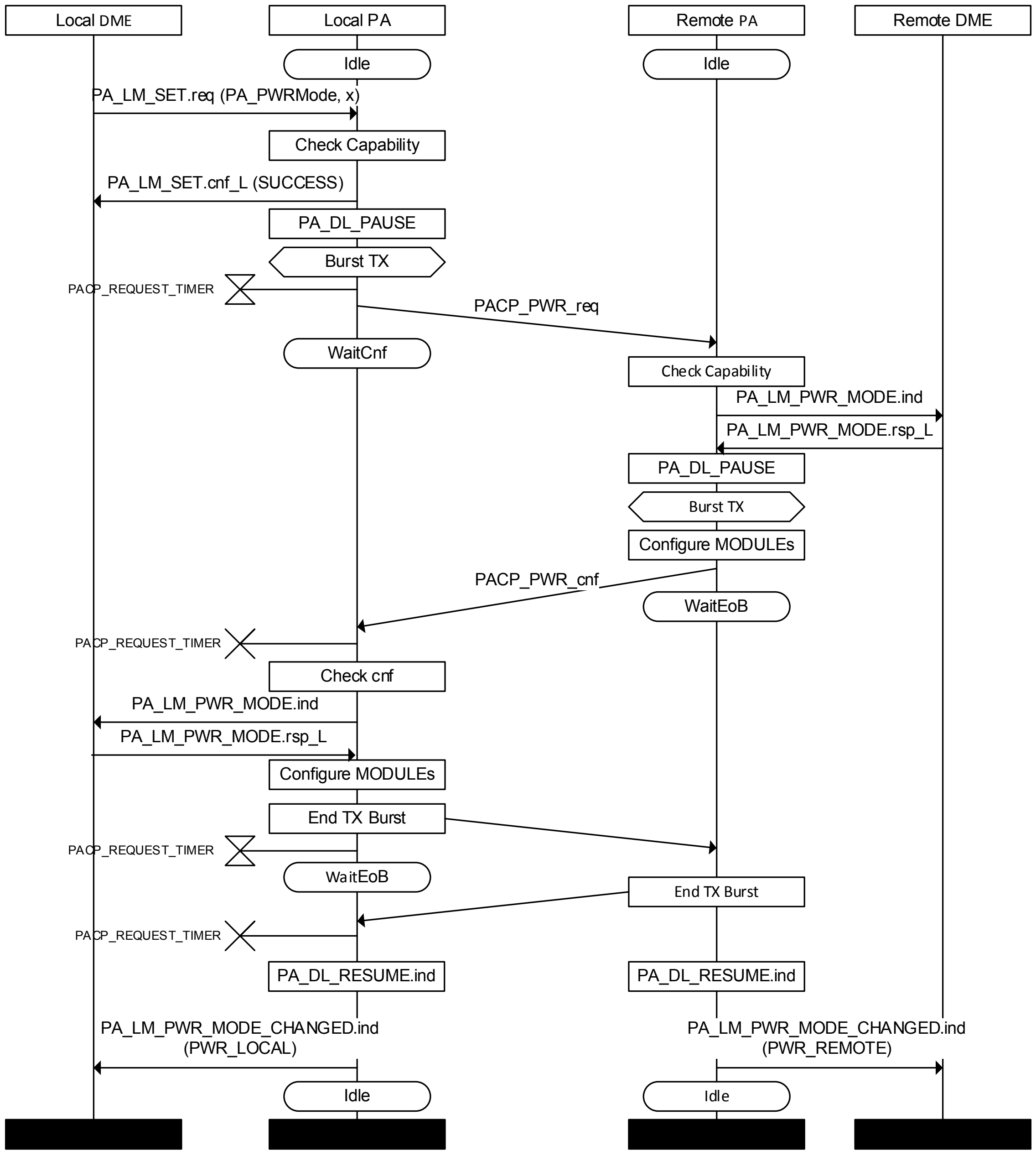


Figure 46 Power Mode Change Using PACP_PWR_req and PACP_PWR_cnf

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.7.12.2.2 With UniPro Versions Prior to 1.8

If one of the Devices in the Link is a UniPro version earlier than v1.8, the detailed operation differs from ***Section 5.7.12.2.1***.

The configuration is set via PA_HSSeries, PA_TxGear, PA_TxTermination, PA_Scrambling, PA_TxHsAdaptType, PA_RxGear, PA_RxTermination, PA_ActiveTxDataLanes, and PA_ActiveRxDataLanes. In addition to the M-PHY MODULE configuration, the PA Layer can transmit user-data, e.g. L2 timer values, to the peer DME. This can be supplied via PA_PWRModeUserData.

The basic principle of the Power Mode change is again depicted in ***Figure 46***. Attributes are assumed to be set before changing the Power Mode. The local PA Layer first checks the request against the inbound Link capabilities, in this case from peer to local. After replying to the DME that the request was accepted, the local PA Layer requests permission from the DL Layer. If the PHY Layer supports activating Lanes while other Lanes are in BURST, the local PA Layer may use an existing burst on the active TX Lanes to send the PACP_PWR_req frame. Otherwise, the local PA Layer shall begin a burst on the active TX Lanes to send the PACP_PWR_req frame.

In both cases, the local PA Layer shall begin a dummy burst on the inactive connected TX Lanes. When the last Lane has entered BURST, the local PA Layer shall send the PACP_PWR_req frame.

When the peer PA Layer receives a valid PACP_PWR_req frame, it shall first check the request against its inbound Link capabilities, in this case from local to peer. The peer PA layer shall then save a copy of the PACP_PWR_req frame fields, DevID, Adapt, Flags, TxMode, TxLane, TxGear, RxMode, RxLane, and RxGear, to add to the PACP_PWR_cnf. Then the peer PA Layer shall request permission from the DL Layer. Next, the peer PA Layer shall exchange PAPowerModeUserData, e.g., L2 timer values, with its DME. The last two parameters of PAPowerModeUserData received from the DME shall be replaced with the saved fields, DevID, Adapt, Flags, TxMode, TxLane, TxGear, RxMode, RxLane, and RxGear.

If the PHY Layer supports activating Lanes while other Lanes are in BURST, the peer PA Layer may use an existing burst on the active TX Lanes before sending the PACP_PWR_cnf frame. Otherwise, the peer PA Layer shall begin a burst on the active TX Lanes before sending the PACP_PWR_cnf frame. In addition, if the request was accepted, the peer PA Layer shall begin a dummy burst on the inactive connected TX Lanes, and shall configure the PHY Layer with the parameters of the request. If the request was not accepted, the peer PA Layer shall configure the inactive connected inbound Lanes to enter HIBERN8. After completing these steps, the peer PA Layer shall send the PACP_PWR_cnf frame.

When the local PA Layer receives a valid PACP_PWR_cnf frame, it shall first check the Status field. If the Status field contains PWR_OK, PAPowerModeUserData is passed to the local DME. The last two parameters of the PACP_PWR_cnf frame can be passed to the DME with the remaining ten PAPowerModeUserData parameters such that a total of 24 bytes are passed to the DME. Passing this data has no affect as the DME silently discards the PAPowerModeUserData. The local PHY Layer shall be configured with the requested parameters. The local PA Layer shall close the burst on the outbound Link.

The peer PA Layer shall close the burst on the other Link when detecting the end of burst on its inbound Link. Consequently, both directions of the Link have the new configuration activated. Both sides of the Link shall notify their respective DL Layers about the completion of the procedure. The local and peer PA Layer shall notify their respective DMEs with a PA_LM_PWR_MODE_CHANGED.ind primitive with the PowerChangeResultCode parameter containing PWR_LOCAL and PWR_REMOTE, respectively. If the Status field does not contain PWR_OK, then the local PA Layer shall configure the outbound Lanes with open dummy burst to enter HIBERN8, close the burst on the outbound Link, and notify the local DME with a PA_LM_PWR_MODE_CHANGED.ind primitive with the PowerChangeResultCode parameter containing the value of the Status field of the PACP_PWR_cnf frame. In both cases, when the DMEs have been notified, the Power Mode change is concluded, and both PA Layers are ready for subsequent operations.

The power configuration of a MODULE is always set during an active burst. In order to keep the configuration among all connected Lanes identical, the inactive, but connected, Lanes are temporarily put into a dummy burst for the duration of the Power Mode change procedure.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.7.12.3 Configuring M-PHY MODULEs

A PA Layer implementation following this Specification version 1.8 can configure MODULEs without Dummy Burst, and can configure inactive M-PHY MODULEs during M-PHY SAVE or HIBERN8 states.

When operating against a PA Layer implementing UniPro legacy version 1.61, the PA Layer of a version 1.8 Device acts like the PA Layer of a version 1.61 Device. During such legacy operation, except for the exit from HIBERN8, the power configuration of a MODULE is always set during an active M-PHY Burst. In order to keep the configuration among all connected Lanes identical, the inactive, but connected, Lanes are temporarily put into a dummy Burst for the duration of the Power Mode change procedure.

5.7.12.3.1 Preparation of M-PHY MODULE Configuration

If the connection is established between two UniPro stacks implementing version 1.8, this section shall be disregarded. If connected to a PA Layer implementing UniPro version 1.61, the following preparation for M-PHY MODULEs is required:

All connected Lanes that are not yet activated shall be woken up and put into the dummy Burst before applying the configuration. For example, when two Lanes are connected and the Lane count is increased from one to two, the second Lane is in HIBERN8 and thus needs to be woken up. However, the inactive Lanes running a dummy Burst shall never be used for data communication.

Before creating a dummy Burst, the PA Layer shall wake-up the Lane from HIBERNATE_STATE (see ***Section 5.7.4)***. A dummy Burst shall begin by transmitting a special pattern <MK0, FLR>, which distinguishes the dummy Burst from a normal Burst. During the dummy Burst, the PA Layer shall transmit only FILLERs. The PA Layer shall end the dummy Burst in the same manner as a normal Burst (see ***Section 5.7.4)***.

The Lane wake-up is performed as follows:

```
For all connected Lanes:
TX_HIBERN8_Control = EXIT;
Inform M-TX that configuration is ready via M-CTRL-CFGREADY.request primitive
Wait for PA_TActivate
Start a burst or a dummy burst on all connected Lanes
```

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.7.12.3.2 Applying the new M-PHY Configuration

The following Attributes are set on all connected M-TX:

```
TX_LCC_Enable = NO;
if ((outbound_mode == Fast_Mode) || (outbound_mode == FastAuto_Mode))
{
   TX_MODE = HS_MODE;
   TX_HSRATE_Series = hs_series;
   TX_HSGEAR = outbound_gear;
   if (PACP_CAP_EXT2_ind was detected) {
      switch (outbound_gear) {
         case 1:   TX_HS_SYNC_LENGTH = PA_TxHsG1SyncLength;
                   TX_HS_PREPARE_LENGTH = PA_TxHsG1PrepareLength;
         case 2:   TX_HS_SYNC_LENGTH = PA_TxHsG2SyncLength;
                   TX_HS_PREPARE_LENGTH = PA_TxHsG2PrepareLength;
         case 3:   TX_HS_SYNC_LENGTH = PA_TxHsG3SyncLength;
                   TX_HS_PREPARE_LENGTH = PA_TxHsG3PrepareLength;
         case 4:   TX_HS_SYNC_LENGTH = PA_TxHsG4SyncLength;
                   TX_HS_PREPARE_LENGTH = PA_TxHsG4PrepareLength;
      }
   }
   else if (PACP_CAP_EXT1_ind was detected) {
      switch (outbound_gear) {
         case 1:   TX_HS_SYNC_LENGTH = PA_TxHsG1SyncLength;
                   TX_HS_PREPARE_LENGTH = PA_TxHsG1PrepareLength;
         case 2:   TX_HS_SYNC_LENGTH = PA_TxHsG2SyncLength;
                   TX_HS_PREPARE_LENGTH = PA_TxHsG2PrepareLength;
         case 3:   TX_HS_SYNC_LENGTH = PA_TxHsG3SyncLength;
                   TX_HS_PREPARE_LENGTH = PA_TxHsG3PrepareLength;
      }
   }
   TX_HS_Unterminated_LINE_Drive_Enable = (1 - outbound_termination);
}
if ((outbound_mode == Slow_Mode) || (outbound_mode == SlowAuto_mode))
{
   TX_MODE = LS_MODE;
   TX_PWMGEAR = outbound_gear;
   TX_LS_Terminated_LINE_Drive_Enable = outbound_termination;
}
```

The following Attribute is set additionally on any connected M-TX that becomes inactive again after the burst. This includes those Lanes transferring a dummy burst against Devices implementing the version v1.61 of this specification:

```
TX_HIBERN8_Control = ENTER;
```

When operating against a v1.8 peer without dummy burst, for all M-TX Modules to be activated from HIBERN8 the following Attribute is set:

```
TX_HIBERN8_Control = EXIT;
```

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

The following Attributes are set on all connected M-RX (all connected lanes are woken up and in burst or dummy burst when local attribute programming is performed):

```
if ((inbound_mode == Fast_Mode) || (inbound_mode == FastAuto_Mode))
{
   RX_MODE = HS_MODE;
   RX_HSRATE_Series = hs_series;
   RX_HSGEAR = inbound_gear;
   RX_HS_Unterminated_Enable = (1- inbound_termination);
}
if ((inbound_mode == Slow_Mode) || (inbound_mode == SlowAuto_mode))
{
   RX_MODE = LS_MODE;
   RX_PWMGEAR = inbound_gear;
   RX_LS_Terminated_Enable = inbound_termination;
}
```

The following Attribute is set additionally on any connected M-RX that becomes inactive again after the Burst. This includes those Lanes transferring a dummy Burst against Devices implementing UniPro version 1.61:

```
RX_Enter_HIBERN8 = YES;
```

If the power mode change request includes an ADAPT request, then there is a need for additional configuration on M-TX and M-RX which still stay active after Burst closure (see ***Section 5.7.12.4***). After setting Attributes, the configuration is marked as ready via an M-CTRL-CFGREADY primitive. These settings become active on all Lanes not later than when logical Lane #0 is reconfigured after exit from the Burst. ADAPT should not be expected or invoked for a Lane that will enter HIBERN8 or remain in HIBERN8.

The above used constants ENTER, EXIT, HS_MODE, LS_MODE, and YES as defined by ***[MIPI02]***.

In case the local Device requests the Power Mode change, the parameters outbound_mode, inbound_mode, hs_series, outbound_gear, inbound_gear, outbound_termination, and inbound_termination correspond to the PACP_PWR_req parameter (attribute) set to be transmitted:

```
outbound_mode = TxMode (PA_PWRMode[b2:b0]);  // UniPro Power Mode for TX direction
inbound_mode = RxMode (PA_PWRMode[b6:b4]);   // UniPro Power Mode for RX direction
hs_series = Flags[3] +1;                     // local PA_HsSeries
outbound_gear = TxGear;                      // local PA_TxGear
inbound_gear = RxGear;                       // local PA_RxGear
outbound_termination=Flags[1];               // local PA_TxTermination
inbound_termination=Flags[0];                // local PA_RxTermination
```

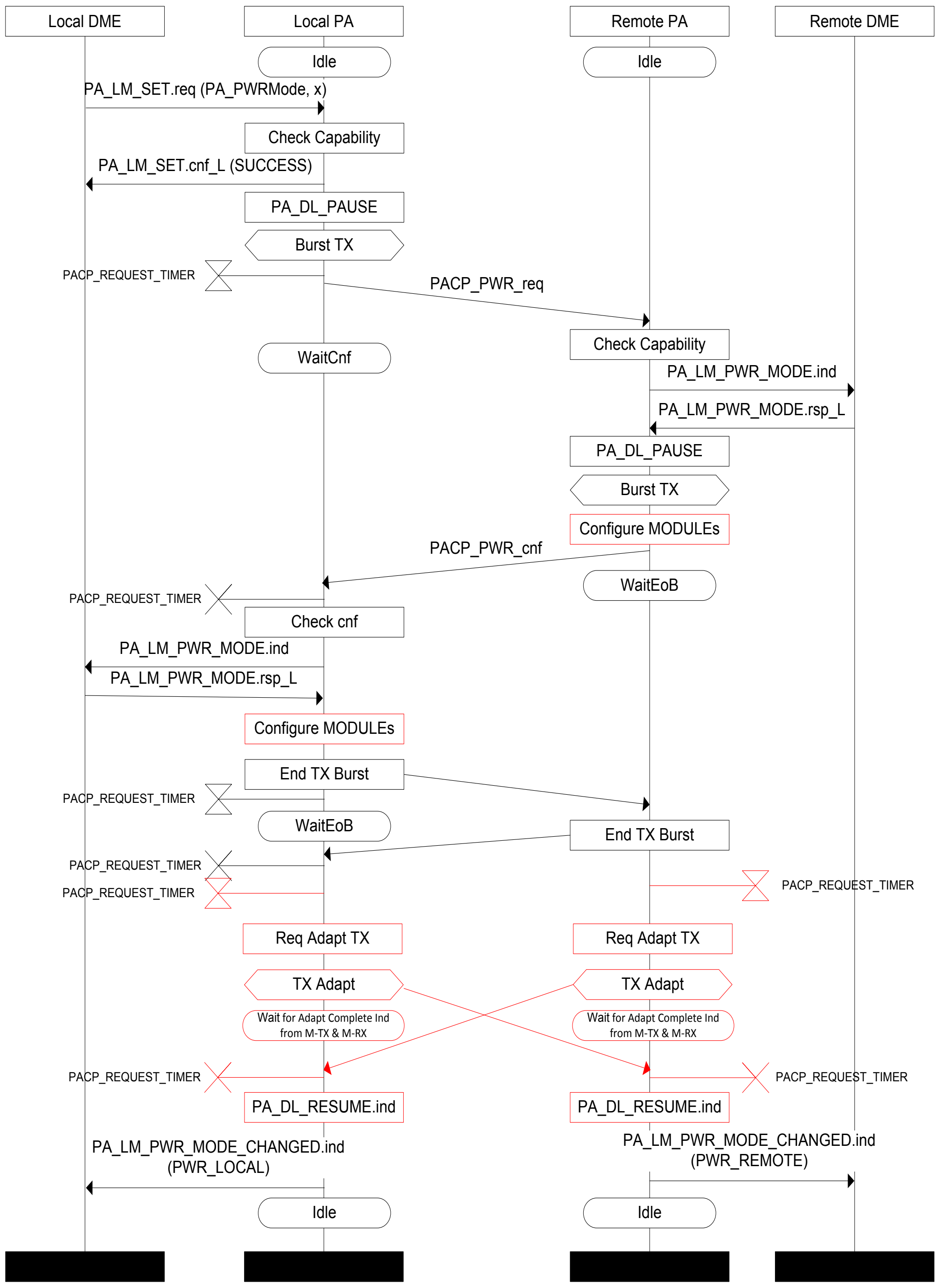
In case the peer Device requests the Power Mode change, the parameters outbound_mode, inbound_mode, hs_series, outbound_gear, inbound_gear, outbound_termination and inbound_termination are determined from received PACP_PWR_req parameters:

```
outbound_mode = RxMode;          // UniPro Power Mode for RX direction of
                                 // requesting peer Device
inbound_mode = TxMode;           // UniPro Power Mode for TX direction of
                                 // requesting peer Device
hs_series = Flags[3] + 1;
outbound_gear =RxGear;
inbound_gear = TxGear;
outbound_termination =Flags[0];   // RX-direction termination enable of peer Device
inbound_termination =Flags[1];    // TX-direction termination enable of peer Device
```

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.7.12.4 Power Mode Change Process with ADAPT

2923 This Section describes handling of ADAPT state on top of the existing Power Mode Change process. ***Figure***
2924 ***47*** illustrates the basic principle of the Power Mode change with ADAPT, where both the forward and reverse
2925 Links experience the ADAPT process. ***Figure 47*** is an extension of ***Figure 46*** adding ADAPT.



2926

Figure 47 Power Mode Change Using PACP_PWR_req and PACP_PWR_cnf with ADAPT State

Figure 48 describes the Power Mode Change process in which only the forward link is experiencing ADAPT. It also shows the Power Mode Change process when PACP_PWR_req has asymmetric settings.

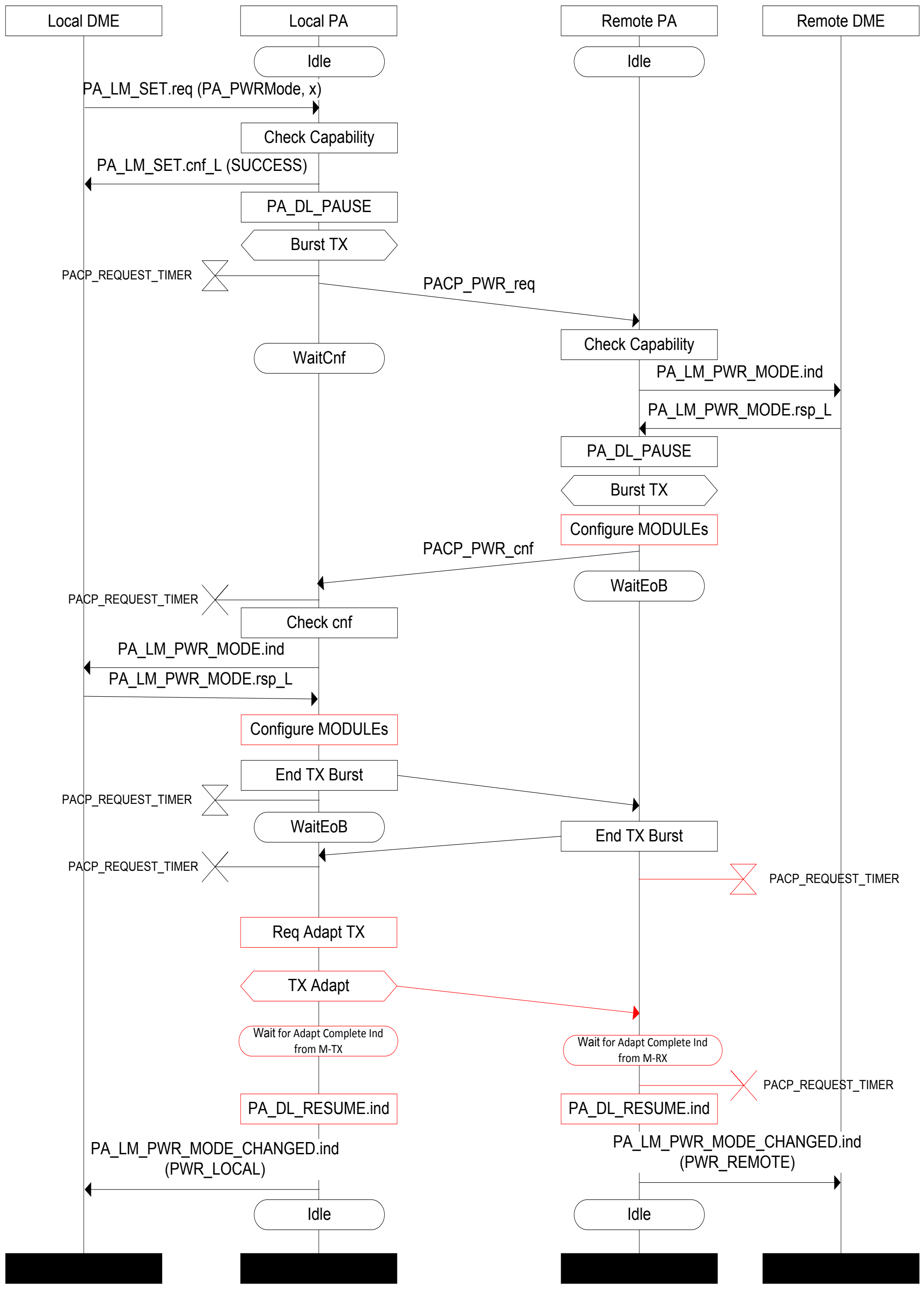


Figure 48 Power Mode Change Using PACP_PWR_req and PACP_PWR_cnf with ADAPT State Only for Forward Link

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Figure 49 shows an example of how the Power Mode Change process can be visualized on serial lines when both forward and reverse links experience ADAPT.

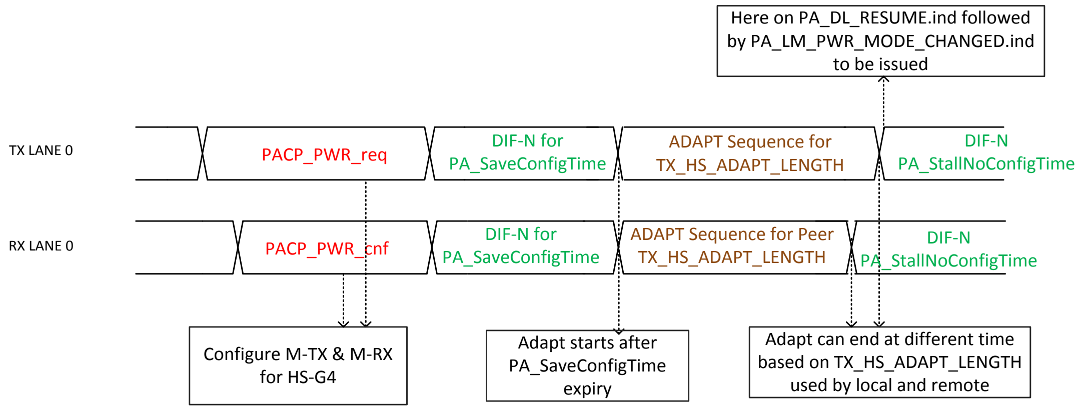

Here on PA_DL_RESUME.ind followed by PA_LM_PWR_MODE_CHANGED.ind to be issued
TX LANE 0
PACP_PWR_req
DIF-N for PA_SaveConfigTime
ADAPT Sequence for TX_HS_ADAPT_LENGTH
DIF-N PA_StallNoConfigTime
RX LANE 0
PACP_PWR_cnf
DIF-N for PA_SaveConfigTime
ADAPT Sequence for Peer TX_HS_ADAPT_LENGTH
DIF-N PA_StallNoConfigTime
Configure M-TX & M-RX for HS-G4
Adapt starts after PA_SaveConfigTime expiry
Adapt can end at different time based on TX_HS_ADAPT_LENGTH used by local and remote

Figure 49 Serial Line View of Power Mode Change Using PACP_PWR_req and PACP_PWR_cnf with ADAPT State

The Adapt field of the PACP_PWR_req frame is controlled through the PA_TxHsAdaptType or PA_AdaptAfterLRSTInPA_INIT attribute.

The value programmed in the PA_AdaptAfterLRSTInPA_INIT attribute will be used as the Adapt field of the PACP_PWR_req that is generated after LINE-RESET as part of the PA_INIT process (see ***Section 5.7.10).***

The Application initiating the Power Mode Change request configures the PA_TxHSAdaptType attribute, and the value of this attribute will be used in the Adapt field. The UniPro transmitter initiating the PACP_PWR_req shall check this attribute has the correct value setting. If it is not set correctly, then the UniPro receiver shall reject the Power Mode Change request. It is a responsibility of the Application to configure this attribute with the correct value, based on the Power Mode Change request type.

Currently ***[MIPI02]*** supports ADAPT only for HS-G4. As a result, PA_TxHSAdaptType can only be set to the value 2'00 or 2'b01 if the Power Mode Change request has a HS-G4 speed request on at least one direction in inbound or outbound data Lanes.

> **Example:** If the PACP_PWR_req has TxMode = RxMode = FastAuto_Mode or Fast_Mode, and if the requested gear is not HS-G4 on both sub-Links, then it is incorrect to configure PA_TxHSAdaptType to 2'b00 or 2'b01.

The Adapt field in the PACP_PWR_req frame determines the presence of ADAPT and the type of ADAPT Range for the current Power Mode Change process.

Interpretation of the Adapt field:

- **00:** ADAPT is present, and the selected ADAPT Type is REFRESH
- **01:** ADAPT is present, and the selected ADAPT Type is INITIAL
- **10:** Reserved
- **11:** ADAPT is not present in this PowerModeChange

The Local Device can initiate the PACP_PWR_req with Adapt = 2'11 in cases where it intends to skip the ADAPT process. This value can be used when the Link partners have already achieved equalization.

The selected ADAPT Range applies to both inbound and outbound data Lanes.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

If the ADAPT is involved in the PowerModeChange, and the sub-Link is not already in HS-G4 speed, then PACP_PWR_req may satisfy the following condition in order to equalize all lanes before entering HS-G4.

```
TxLane = PA_ActiveTxDataLanes = PA_ConnectedTxDataLanes
RxLane = PA_ActiveRxDataLanes = PA_ConnectedRxDataLanes
```

Example: If the ADAPT is only applicable to inbound communication, then the RxLane field of PACP_PWR_req may have the RxLane field value set to RxLane = PA_ActiveRxDataLanes = PA_ConnectedRxDataLanes

The PA receiver shall only treat the Adapt field as valid if PACP_PWR_req has a HS-G4 request on any one sub-Link (either inbound or outbound), otherwise it shall ignore the Adapt field.

If the PACP_PWR_req has a HS-G4 request on any one sub-Link (either inbound or outbound), and the Adapt field is set to 2'b10, then the PA receiver shall reject the power mode change request and respond with the value PWR_ERROR_CAP in the status field of the PACP_PWR_cnf.

If the Power Mode Change handshake is successful and inbound data Lanes are expected to receive ADAPT, then all active M-RX Lanes shall be configured as shown below:

```
if (Adapt == REFRESH)
{
   RX_ADAPT_Control = {REFRESH, ADAPT};
}
else if (Adapt == INITIAL)
{
   RX_ADAPT_Control = {INITIAL, ADAPT};
}
```

If the Power Mode Change handshake is successful and outbound data Lanes are expected to transmit ADAPT, then all activated M-TX Lanes shall be configured as shown below:

```
if (Adapt == REFRESH)
{
   TX_HS_ADAPT_LENGTH = PA_PeerRxHsAdaptRefresh;
}
else if (Adapt == INITIAL)
{
   TX_HS_ADAPT_LENGTH = PA_PeerRxHsAdaptInitial;
}
```

No PA Attribute is defined to store the Adapt field value that is received in the PACP_PWR_req frame. UniPro can internally store this value. However, the Adapt field value that is received in PACP_PWR_req that occurs after LINE-RESET in the PA_INIT process shall not be stored, and shall not be used as the Adapt value in a successive PACP_PWR_req frame.

Refer to ***Figure 47*** and ***Figure 48*** while going through below description.

- According to ***[MIPI02]***, RX_HS_ADAPT_INITAL_Capability or RX_HS_ADAPT_REFRESH_Capability exist only if RX_HS_Equalizer_Setting_Capability has the value 1. If RX_HS_Equalizer_Setting_Capability has the value 0, then RX_HS_ADAPT_INITAL_Capability and RX_HS_ADAPT_REFRESH_Capability do not exist and the M-RX does not support ADAPT.
- The local or peer Device can indicate that its inbound Link does not support ADAPT. If RX_HS_ADAPT_INITAL_Capability and RX_HS_ADAPT_REFRESH_Capability do not exist for the local M-RX then the inbound Link does not support ADAPT.
- A value of 255 in PA_PeerRxHSAdaptInitial or PA_PeerRxHsAdaptRefresh means that the peer indicated that its M-RX does not support ADAPT via a PACP_CAP_EXT2_ind frame, and the outbound Link does not support ADAPT.
- If the outbound Link does not support ADAPT, the PA transmitter shall not send the M-LANE-AdaptStart.request primitive to its M-TX.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

- If the inbound Link does not support ADAPT, the PA shall not configure the M-RX's RX_ADAPT_Control attribute for ADAPT.
- If the Power Mode Change request is for HS-G4 on one sub-Link, and the Adapt field is set to the value 2'b00 or 2'b01 while the corresponding sub-Link does not support ADAPT, then the Power Mode Change process will proceed without ADAPT on that sub-Link.
- If the Power Mode Change request is for HS-G4 on both sub-Links, and the Adapt field is set to the value 2'b00 or 2'b01, while neither sub-Link supports ADAPT then the Power Mode Change process will proceed with no ADAPT.
- The PA receiver shall treat the Adapt field as valid only if the PACP_PWR_req has a HS-G4 request on any one sub-Link (either inbound or outbound), otherwise it shall ignore the Adapt field.
- As mentioned above, the PA receiver shall also ignore the Adapt field and ADAPT procedure if there is no ADAPT support for a given sub-Link.
- If a valid ADAPT request is present in at least one direction's (inbound or outbound) data Lanes, the PA layer shall postpone issuing PA_DL_RESUME.ind to the DL layer until the ADAPT sequence is complete.
- Once the PACP_PWR_req and PACP_PWR_cnf handshake is complete and Status is PWR_OK, both the local Device and the peer Device shall wait for the period PA_SaveConfigTime to elapse.
- After this, the PA transmitter initiates the ADAPT request to the M-TX when its outbound data Lanes need to perform ADAPT.
- If the ADAPT request is present on inbound data Lanes, then the PA layer starts the PACP_REQUEST_TIMER at the beginning of the PA_SaveConfigTime period following the PACP_PWR_req and the PACP_PWR_cnf handshake.
- If ADAPT is only requested on outbound data Lanes, the PA layer waits for an M-LANE-AdaptComplete.indication from the M-TX. If ADAPT is only requested on inbound data Lanes, the PA layer waits for an M-LANE-AdaptComplete.indication from the M-RX. If ADAPT is requested on both inbound and outbound data Lanes, the PA layer waits for M-LANE-AdaptComplete.indications from both the M-TX and the M-RX.
- The PA Layer loads the PACP_REQUEST_TIMER with a value of 10ms.
- If the Application intends to set TX_HS_ADAPT_LENGTH with a value higher than PA_PeerRxHsAdaptRefresh/Initial (based on fine/coarse selection), then the value shall also be less than PACP_REQUEST_TIMER.

  This requirement is necessary because this version of UniPro has no provision for passing the local TX_HS_ADAPT_LENGTH to the peer UniPro. If this requirement were not followed, then the peer UniPro PA layer would not properly predict the maximum ADAPT length on its M-RX. As a result, the peer PA layer PACP_REQUEST_TIMER might expire before the peer M-RX issues its M-LANE-AdaptComplete.indication to the peer PA layer.

  The PACP_REQUEST_TIMER helps the PA layer to exit from ADAPT when the M-RX fails to issue an M-LANE-AdaptComplete.indication. For ADAPT to be able to complete successfully, the value configured in the PACP_REQUEST_TIMER must be chosen to ensure that the M-LANE-AdaptComplete.indication is received before the timer expires.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.7.12.4.1 Use of ADAPT Service Primitives

This Section describes how UniPro uses the ADAPT service primitives (see ***[MIPI02]*** for further details on all ADAPT service primitives). Based on the Power Mode Change request type, if the outbound data Lanes of the local Device need to perform ADAPT then the PA transmitter performs the following steps:

- The local PA transmitter requests the M-TX to transmit the ADAPT sequence via the M-LANE-AdaptStart.request primitive.
- The M-TX confirms the start of the ADAPT sequence with M-LANE-AdaptStart.confirm.
- The ADAPT sequence runs to completion. The local PA transmitter shall not send a new ADAPT request before the previously requested ADAPT sequence completes.
- When the ADAPT sequence is complete, the M-TX confirms this status through the M-LANE-AdaptComplete.indication.
- Based on the Power Mode Change request type, if the local Device's inbound data Lanes need to experience ADAPT then the PA receiver starts the PACP_REQUEST_TIMER.
- When ADAPT reception is complete the M-RX indicates this through the M-LANE-AdaptComplete.indication.
- Once the M-LANE-AdaptComplete.indication is received, the PA layer issues the PA_DL_RESUME.ind to the DL layer and issues PA_LM_PWR_MODE_CHANGED.ind to the DME layer.
- If a valid ADAPT request is present on outbound Link then the PA transmitter initiates the PA_StallNoConfigTime once the M-LANE-AdaptComplete.indication is received, and then waits for the timer to expire before initiating the next Burst.
- If the M-RX does not indicate ADAPT completion by issuing M-LANE-AdaptComplete.indication, this will lead to expiration of the PACP_REQUEST_TIMER. In such a case the PA layer shall pass the PA_LM_PWR_MODE_CHANGED.ind(PWR_FATAL_ERROR) to the DME.

5.7.12.5 User Data Within PACP_PWR Frames

The PA Layer provides a method for the local and peer DME to exchange information that is needed during the Power Mode change. The information is transmitted in PAPowerModeUserData in the PACP_PWR_req and PACP_PWR_cnf frames. PAPowerModeUserData is just a 24-byte payload for the PA Layer. The structure of PAPowerModeUserData, as used by the DME, is defined in ***Table 102***. The length of PAPowerModeUserData is enough to store all L2 timer values for all traffic classes.

The local DME, which creates the Power Mode change request, supplies the information with the value of PA_PWRModeUserData before setting PA_PWRMode. The local PA Layer shall attach this information to the PACP_PWR_req frame and transmit it. On reception, the peer PA Layer shall deliver the payload to the peer DME via the PA_LM_PWR_MODE.ind(FALSE, payload_req) primitive. The first parameter tells if the request was originated from this side. The peer DME responds with the PA_LM_PWR_MODE.rsp_L(payload_cnf) primitive. The peer PA Layer shall then attach the payload_cnf to the PACP_PWR_cnf frame and transmit it. The local PA Layer shall deliver the payload back to the local DME with the PA_LM_PWR_MODE.ind(TRUE, payload_cnf). The PA Layer shall wait until the local DME responds with PA_LM_PWR_MODE.rsp_L(null) before finishing the Power Mode change operation.

While the re-initialization procedure (see ***Section 5.7.10)*** and the hibernation procedure (see ***Section 5.7.13.1)*** exploit functionalities of the Link Configuration procedure, they do not support the user data exchange described above. In those operations, the UserDataValid flag of a PACP_PWR_req frame shall be set to '0' to indicate that PAPowerModeUserData does not carry valid data. UserDataValid set to '0' instructs the receiving PA Layer to skip the data delivery to the DME completely. Similarly, when the requesting PA Layer receives the response in a PACP_PWR_cnf frame, it skips the data delivery to the DME.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.7.12.6 Error Recovery

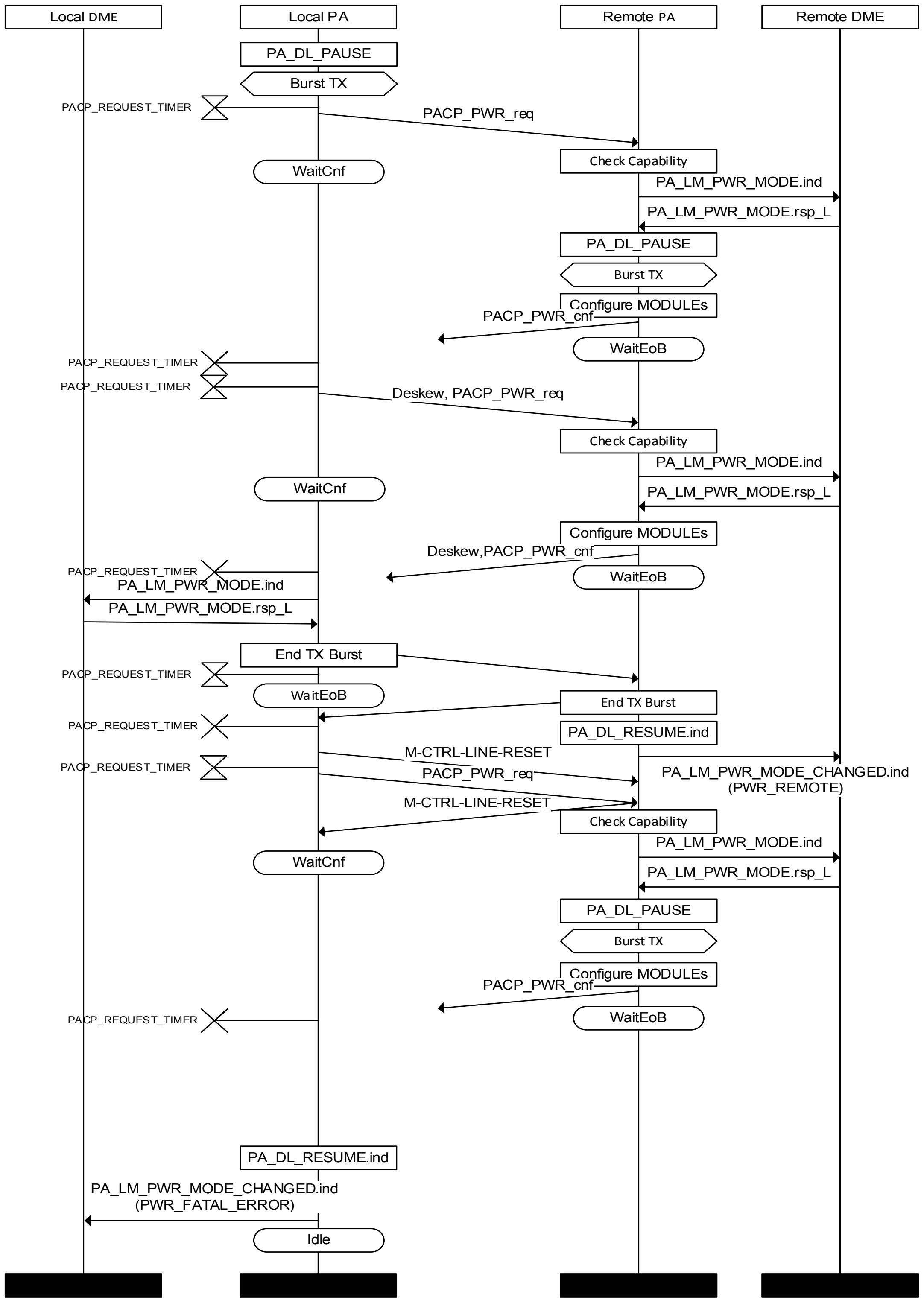
This section describes recovery mechanisms that are used to cope with an unreliable communication path that might introduce bit-errors or symbol loss. ***Figure 50*** shows the relevant part for one example of an unsuccessful case of a PowerMode change attempt, and it is used here as the basis for the description.

The local PA Layer shall have a timer, PACP_REQUEST_TIMER, to detect a missing PACP_PWR_cnf frame, and also to detect missing end-of-burst on the inbound Link. The initial timer value shall be set through PA_PACPReqTimeout and PA_PACPReqEoBTimeout, respectively.

The following list includes the steps related to error recovery. Only the local PA Layer shall take recovery actions while the peer PA Layer only follows them:

1. A PACP_PWR_req frame is transmitted. PACP_REQUEST_TIMER shall be set to PA_PACPReqTimeout.
2. If a valid PACP_PWR_cnf frame is not received before PACP_REQUEST_TIMER expires, the PA Layer shall transmit a deskew pattern and shall send the PACP_PWR_req frame again. PACP_REQUEST_TIMER shall be set to PA_PACPReqTimeout.
3. If a valid PACP_PWR_cnf frame is not received before PACP_REQUEST_TIMER expires, the PA Layer shall first issue a LINE-RESET, and then shall send the PACP_PWR_req frame again, but this time asserting the LINE-RESET request flag. The LINE-RESET request flag instructs the peer PA Layer to do a LINE-RESET on the opposite direction. PACP_REQUEST_TIMER shall be set to PA_PACPReqTimeout + $T_{LINE\text{-}RESET}$ to take into account the time consumed at the LINE-RESET issued by the peer.
4. If a valid PACP_PWR_cnf frame is not received before PACP_REQUEST_TIMER expires, or if the Status field of the PACP_PWR_cnf frame does not contain PWR_OK, the PA Layer shall abort the Power Mode request, and pass PA_LM_PWR_MODE_CHANGED.ind(PWR_FATAL_ERROR) to the DME. The PA Layer shall close the Burst (see ***Figure 50***).
5. If a PACP_PWR_cnf frame is received successfully with PWR_OK, the local PA Layer closes the burst and shall set PACP_REQUEST_TIMER to PA_PACPReqEoBTimeout. If the peer PA Layer does not close the burst within this period, the local PA Layer shall abort the Power Mode request and pass PA_LM_PWR_MODE_CHANGED.ind(PWR_FATAL_ERROR) to the DME.

Transfers after LINE-RESET use the PWM-G1 1-Lane configuration until the PA layer can reestablish the current Power Mode.



3124
Figure 50 Unsuccessful PowerMode Change with PWR_FATAL_ERROR

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.7.13 PA Hibernate

Hibernate is separated from the normal Power Mode change operation. The hibernate enter and hibernate exit procedures are described in this section. When entering hibernate, the current Power Mode configuration, including M-PHY settings and Lane count information, are stored. They are automatically restored when exiting hibernate.

The PA Layer delays the exit hibernate operation until HIBERNATE_TIMER has expired.

5.7.13.1 Entering Hibernate

The DME issues a PA_LM_HIBERNATE_ENTER.req primitive to the PA Layer to put the PA Layer into hibernate. If the PA Layer is not busy with another Power Mode change operation, it shall respond with the PA_LM_HIBERNATE_ENTER.cnf_L primitive with the parameter set to SUCCESS. Otherwise, it shall respond with the parameter set to FAILURE.

When the procedure has completed successfully, the PA Layer shall issue a PA_LM_HIBERNATE_ENTER.ind primitive with the parameter set to PWR_LOCAL to the requesting DME. The peer PA Layer shall issue a PA_LM_HIBERNATE_ENTER.ind primitive with the parameter set to PWR_REMOTE to the peer DME. In error cases, the primitive has other status, e.g. PWR_BUSY or PWR_FATAL_ERROR.

After successful completion of the procedure, the Link in both directions is in HIBERNATE_STATE and the M-PHY MODULEs are in HIBERN8. The squelch detection of an M-RX may be turned off on all Lanes, except Logical Lane 0, to save power.

The procedure is depicted in ***Figure 51***. Internally, the PA Layer shall use the same Link Configuration procedure as specified in ***Section 5.7.12,*** including capability check, handling of errors and retries and M-PHY configuration, with the following modifications:

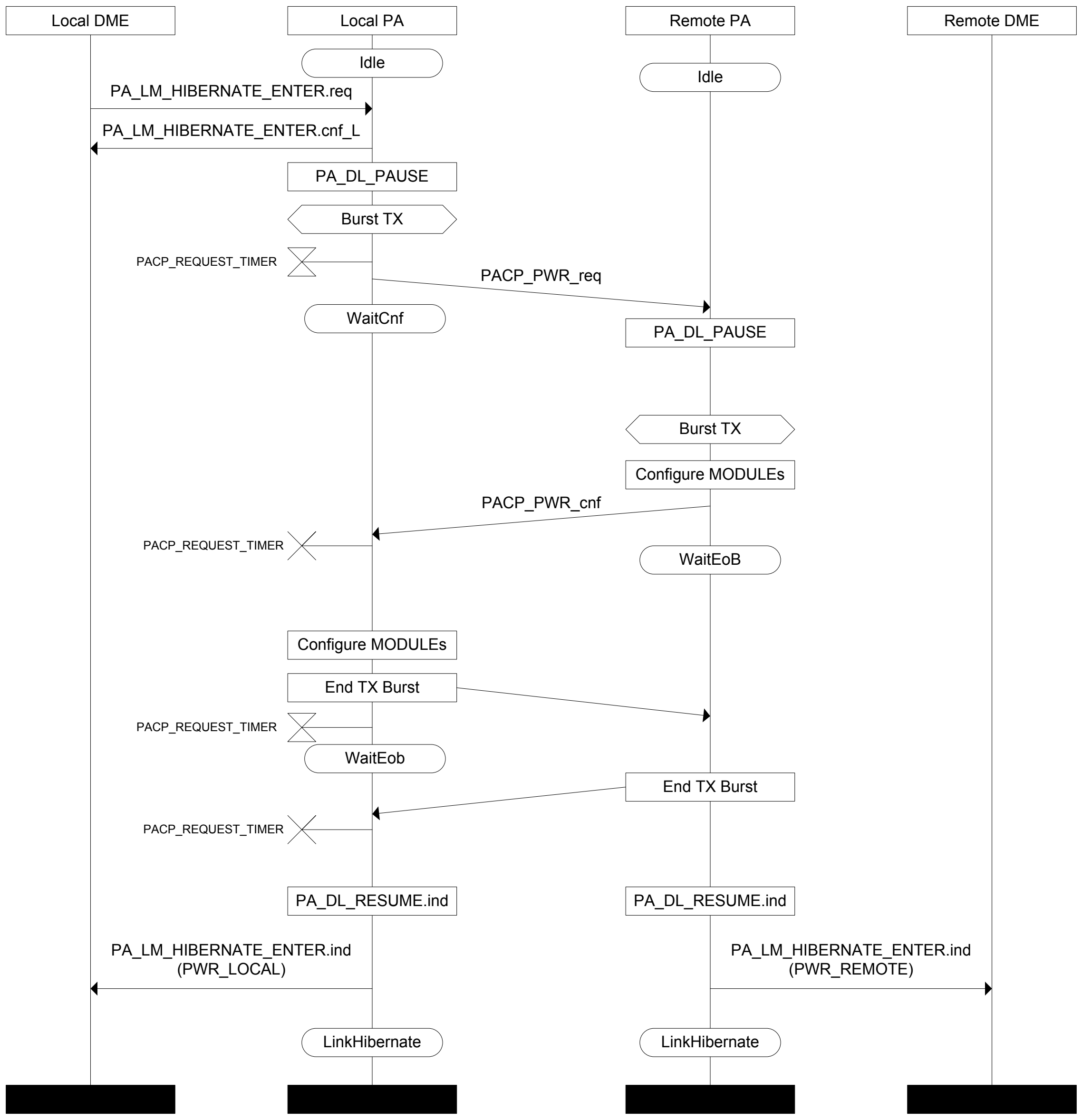
- DME uses the PA_LM_HIBERNATE_ENTER.req primitive instead of using Attributes
- The PA Layer uses PA_LM_HIBERNATE_ENTER.cnf_L to indicate acceptance of the hibernate request
- The PA Layer indicates the status with the PA_LM_HIBERNATE_ENTER.ind primitive
- User data in PACP_PWR_req and PACP_PWR_cnf frames is not used, and is not forwarded to the DME (see ***Section 5.7.12.4***)
- Entering hibernate never fails due to invalid configuration or capability checking
- For all active M-TXs, TX_HIBERN8_Control shall be set to ENTER, other Attributes shall not be set
- For all active M-RXs, RX_Enter_HIBERN8 shall be set to YES, other Attributes shall not be set

The concurrency and error recovery methods are as in the normal Link Configuration procedure.

If the Link Startup Sequence has not been started, a PA_LM_HIBERNATE_ENTER.req is not executed and the PA Layer shall return PA_LM_HIBERNATE_ENTER.cnf_L(FAILURE).

The PA Layer shall generate the PA_LM_HIBERNATE_ENTER.cnf_L and PA_LM_HIBERNATE_ENTER.ind primitives as described previously.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential



3160

Figure 51 Entering Hibernate (After Link Startup)

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.7.13.2 Retained State Information

While in HIBERNATE_STATE, the following information shall be retained:

- PHY configuration (retained by M-PHY MODULEs)
- Information gathered during Link Startup Sequence:
    - PA_ConnectedTxDataLanes, PA_ConnectedRxDataLanes
    - PA_LogicalLaneMap
    - PA_MaxRxPWMGear, PA_MaxRxHSGear
    - PA_RemoteVerInfo
    - PA_SleepNoConfigTime, PA_StallNoConfigTime, PA_SaveConfigTime
    - PA_RxHSUnterminationCapability, PA_RxLSTerminationCapability
    - PA_TActivate
    - PA_Granularity
    - PA_TxTrailingClocks
    - PA_PACPReqTimeout, PA_PACPReqEoBTimeout
    - PA_PeerScrambling
    - PA_TxSkip
    - PA_TxSkipPeriod
    - PA_Local_TX_LCC_Enable
    - PA_Peer_TX_LCC_Enable
    - PA_TxMk2Extension
    - PA_Hibern8Time
    - PA_TxHsG1SyncLength
    - PA_TxHsG1PrepareLength
    - PA_TxHsG2SyncLength
    - PA_TxHsG2PrepareLength
    - PA_TxHsG3SyncLength
    - PA_TxHsG3PrepareLength
    - PA_TxHsG4SyncLength
    - PA_TxHsG4PrepareLength
    - PA_PeerRxHsAdaptRefresh
    - PA_PeerRxHsAdaptInitial
    - PA_MK2ExtensionGuardBand

In addition, the PA Layer shall retain the Link configuration active before entering HIBERNATE_STATE which is gettable through the following Attributes (see ***Section 5.8)***:

- PA_PWRMode
- PA_ActiveTxDataLanes, PA_ActiveRxDataLanes
- PA_TxGear, PA_RxGear
- PA_HSSeries
- PA_TxTermination, PA_RxTermination
- PA_Scrambling
- PA_TxHsAdaptType
- PA_AdaptAfterLRSTInPA_INIT

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.7.13.3 Exiting Hibernate

The DME may request the PA Layer exit hibernate by issuing a PA_LM_HIBERNATE_EXIT.req primitive. The PA Layer shall confirm the request with a PA_LM_HIBERNATE_EXIT.cnf_L primitive. When the exit hibernate procedure has completed, the local PA Layer shall issue a PA_LM_HIBERNATE_EXIT.ind primitive to the local DME with the parameter set to PWR_LOCAL. Similarly, when the peer PA Layer detects the end of hibernate, it shall issue a PA_LM_HIBERNATE_EXIT.ind primitive to the peer DME with the parameter set to PWR_REMOTE. The PA_LM_HIBERNATE_EXIT.ind primitive indicates that the Link is restored and is using the configuration that was active before hibernation.

The PA Layer uses M-PHY signaling to exit HIBERNATE_STATE. First, all active TX Lanes shall be woken up by setting TX_HIBERN8_Control to EXIT. This causes the M-TXs to begin driving DIF-N to the Lanes. The local PA Layer shall wait for PA_TActivate, and then start a timer WAIT_HIBERN8EXIT_TIMER set to PA_PeerWakeTimeout.

When the peer M-RX detects DIF-N, it wakes up and informs the peer PA Layer with a M-LANE-HIBERN8Exit.indication primitive. The M-LANE-HIBERN8Exit.indication primitive shall cause the peer PA Layer to wake up and issue PA_LM_PRE_HIBERNATE_EXIT.ind to the DME. The DME shall perform hibernate exit preparations on DL, N and T Layer according to ***Section 9.5.2*** and issue PA_LM_PRE_HIBERNATE_EXIT.rsp_L to the peer PA Layer. When the peer PA Layer receives PA_LM_PRE_HIBERNATE_EXIT.rsp_L, it shall wake up its M-TXs by setting TX_HIBERN8_Control to EXIT on all active Lanes. The peer PA TX shall start a timer set to PA_TActivate. If the timer expires, indicating that the hibernate has ended, the PA Layer shall send PA_LM_HIBERNATE_EXIT.ind(PWR_REMOTE) to the peer DME. After reception of PA_LM_PRE_HIBERNATE_EXIT.rsp_L, the peer PA RX shall be able to receive symbol data from the M-RX and forward it to the DL Layer.

When the local M-RX detects DIF-N, it wakes up and sends the local PA Layer a M-LANE-HIBERN8Exit.indication primitive. The local PA Layer shall stop WAIT_HIBERN8EXIT_TIMER and indicate that the hibernate has ended by sending the local DME PA_LM_HIBERNATE_EXIT.ind(PWR_LOCAL).

If WAIT_HIBERN8EXIT_TIMER expires, the hibernate exit has failed and the local PA Layer shall issue a PA_LM_HIBERNATE_EXIT.ind(PWR_FATAL_ERROR) to the local DME. At this point, the Link is in unknown, possibly inoperable state.

Immediately after the local PA Layer issues PA_LM_HIBERNATE_EXIT.ind(PWR_LOCAL), the local DL layer may request transmission from the local PA Layer. This transmission granted by the local PA may cause M-LANE-PREPARE.indication and M-LANE-SYMBOL.indication arriving at the peer PA Layer during the time the peer PA TX still waits for PA_TActivate. Provisions initiated by the peer DME between PA_LM_PRE_HIBERNATE_EXIT.ind and PA_LM_PRE_HIBERNATE_EXIT.rsp_L shall ensure that the peer DL Layer shall correctly receive this data. The peer DL Layer shall, however, only request transmission from the peer PA after PA_LM_HIBERNATE_EXIT.ind(PWR_REMOTE).

If the Link Startup Sequence has not been started, a PA_LM_HIBERNATE_EXIT.req primitive shall cause the PA Layer to skip the wake-up procedure previously described, i.e. driving DIF-N and waiting for incoming DIF-N.

Similarly, if the PA Layer receives a M-LANE-HIBERN8Exit.indication primitive while the Link Startup Sequence has not begun, the PA Layer shall skip the wake-up procedure previously described, i.e. driving DIF-N. The PA Layer shall generate the PA_LM_HIBERNATE_EXIT.cnf_L and PA_LM_HIBERNATE_EXIT.ind primitives as described previously.

Exiting hibernate is depicted in ***Figure 52***.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

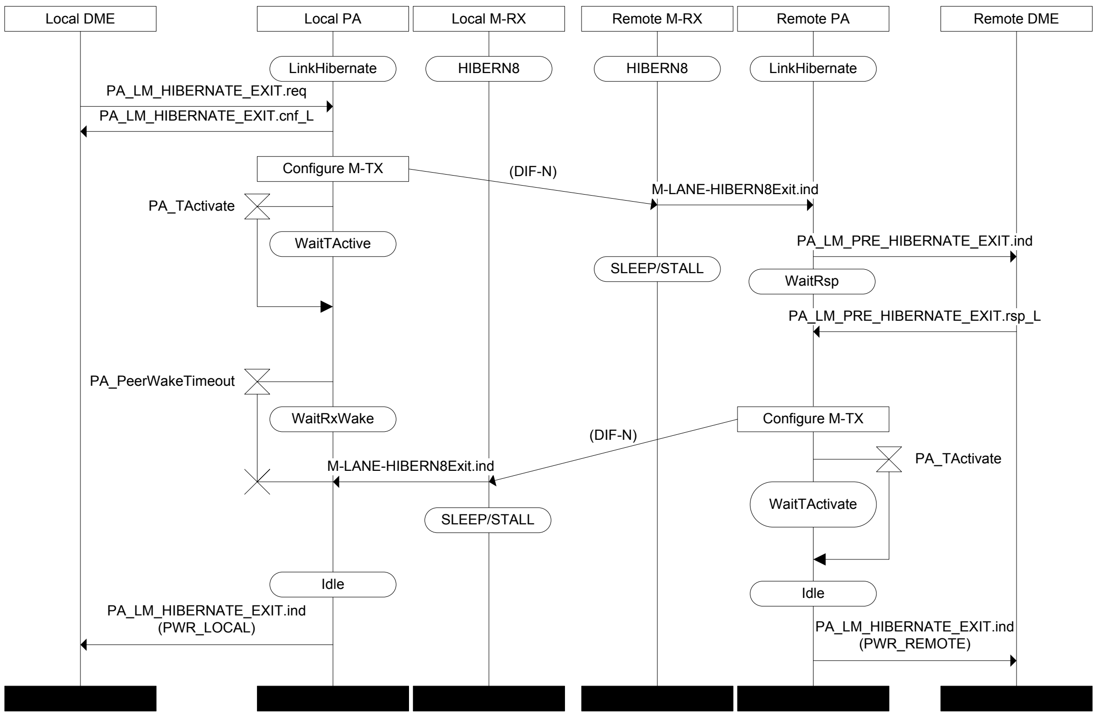


Figure 52 Exiting Hibernate (After Link Startup)

5.7.14 PEER Attribute GET and SET

The PA Layer provides a method to set and get peer Device's Attributes. This is accomplished by using PACP_GET_req, PACP_GET_cnf, PACP_SET_req and PACP_SET_cnf frames. The PA Layer is just a transport while the actual processing of the requests is implemented in the peer Device's DME. Note that the same PACP frames are used also in the PHY testing.

While the PA Layer is executing a Power Mode change or processing a PA_LM_PEER_GET.req, PA_LM_PEER_SET.req, PA_LM_GET.req, or PA_LM_SET.req primitive, the PA Layer shall respond to a PA_LM_PEER_GET.req or PA_LM_PEER_SET.req request with a confirmation where ConfigResultCode is set to BUSY.

While the PA Layer is executing a Power Mode change, it shall not transmit PACP_SET_req, PACP_GET_req, PACP_SET_cnf or PACP_GET_cnf frames. The initiator of the Power Mode Change may still receive a PACP_SET_req or PACP_GET_req frames that have been issued by the peer via PA_LM_PEER_SET.req or PA_LM_PEER_GET.req before reception of a PACP_PWR_req frame. In such cases the initiator of the Power Mode change discards processing and confirmation of received PACP_SET_req or PACP_GET_req frames. If the peer receives a PACP_PWR_req frame, it cancels processing of pending expected incoming PACP_SET_cnf or PACP_GET_cnf frames and confirms the pending PA_LM_PEER_SET.req or PA_LM_PEER_GET.req with parameter ConfigResultCode set to BUSY.

The logic for transmitting PACP_GET_req and PACP_SET_req frames is optional, and depends whether the PA_LM_PEER_GET.req and PA_LM_PEER_SET.req primitives are implemented.

5.7.14.1 GET Operation

When the PA Layer receives a PA_LM_PEER_GET.req primitive from the DME, the PA Layer shall transmit a PACP_GET_req frame containing the same information that the primitive had. The PA Layer shall wait until it receives the confirmation or the timer expires (see ***Section 5.7.13.3)***. When the PA Layer receives the PACP_GET_cnf frame, it shall forward the results to the DME via the PA_LM_PEER_GET.cnf.

When the peer PA Layer receives the PACP_GET_req frame, it shall forward the Attribute information to the DME via the PA_LM_PEER_GET.ind primitive. The PA Layer shall wait for a PA_LM_PEER_GET.rsp primitive. When the PA Layer receives the primitive, it shall transmit a PACP_GET_cnf frame containing the results given in the primitive.

5.7.14.2 SET Operation

When the PA Layer receives a PA_LM_PEER_SET.req primitive from the DME, the PA Layer shall transmit a PACP_SET_req frame containing the same information that the primitive had. If the PA Layer is not in PHY test mode, the PA Layer shall set the Cnf flag of the PACP_SET_req frame to ‘1’. If the PA Layer is in PHY test mode, the PA Layer shall set the Cnf flag of the PACP_SET_req frame to ‘0’. A PA Layer shall still be able to interpret the Cnf flag of a received PACP_SET_req and return a PACP_SET_cnf packet, if requested.

If the Cnf flag was set to ‘1’, the PA Layer shall wait until it receives the confirmation or the timer expires (see ***Section 5.7.14.3)***. When the PA Layer receives the PACP_SET_cnf frame, it shall forward the results to the DME via a PA_LM_PEER_SET.cnf primitive.

If the Cnf flag is set to '0', the PA Layer does not expect a PACP_SET_cnf frame and shall ignore PACP_SET_cnf frames received. ***Section 5.7.14.3*** does not apply as no PACP_REQUEST_TIMER is started. The minimum guard time between PACP_SET_req frame with Cnf set to '0' and any subsequent PACP frame depends on implementation specific process capabilities of local and peer Device. The Application sending out DME_PEER_SET.req is in charge of control for this minimum guard time. The PA Layer shall forward a PA_LM_PEER_SET.cnf primitive with parameter "ConfigResultCode = SUCCESS" to the DME upon completion of its local operation.

When the peer PA Layer receives the PACP_SET_req frame, it shall forward the Attribute information to the DME via a PA_LM_PEER_SET.ind primitive. The PA Layer shall wait for a PA_LM_PEER_SET.rsp primitive. When the PA Layer receives the primitive, and if the Cnf flag in the PACP_SET_req frame was set to ‘1’, the PA Layer shall transmit a PACP_SET_cnf frame containing the status given in the primitive.

5.7.14.3 Replay Mechanism

The PA Layer has a replay mechanism to protect against random communication errors. When transmitting a PACP_GET_req frame or a PACP_SET_req frame with the Cnf flag set to ‘1’, the PA Layer shall set PACP_REQUEST_TIMER to PA_PACPReqTimeout. If the timer expires before receiving the confirmation frame, the PA Layer shall transmit a deskew pattern and retransmit the request frame. Likewise, at the peer, when the two most recently received PACP request frames carry an identical CCITT-CRC16 field, the PA Layer shall transmit a deskew pattern before it transmits the PACP confirm frame. If the PA Layer fails to receive a response after three transmissions (i.e. two retries), the PA Layer shall abort the DME’s request by issuing a PA_LM_PEER_GET.cnf or PA_LM_PEER_SET.cnf primitive, depending of the request, with the parameter ConfigResultCode set to PEER_COMMUNICATION_FAILURE.

Note, the replay mechanism of PACP_GET_req and PACP_SET_req is identical to the one specified in ***Section 5.7.12.6*** for PACP_PWR_req, with the exception that the LINE-RESET phase is not used here. Therefore, an implementation may share the replay functionality (including the timer) between these two features.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.7.15 PHY Testing

This section describes the PA Layer's operation in the PHY test-mode. The behavior and functionalities may differ from the normal operation.

To be able to define a generic way to test the M-PHY, a specific M-PHY Test Feature is required. This Test Feature is essentially a controllable state machine, which can generate and send certain patterns of M-PHY symbols to the M-PHYs, which can be received and analyzed by a specialized tool. These test patterns are encapsulated into PACP frames, so that in receiver testing, the Test Feature shall use the frame verification of a PACP frame. The number of passed and failed test patterns can be read from the PA_PACPFrameCount and PA_PACPErrorCount counters (see ***Section 5.7.7)***.

Since the PA Layer and the M-PHY Test Feature need to be both in direct contact with the M-PHY, the PA Layer shall support two modes of operation: the normal operating mode and a dedicated mode for testing the M-PHY. To put the PA Layer of a DUT in a testing mode, the Tester shall send a PACP_TEST_MODE_req frame. When a UniPro Device, the DUT, receives the PACP_TEST_MODE_req frame, the PA Layer of the DUT shall go to the testing mode. The PA Layer of the DUT shall stop to forward any PA Symbols to the Data Link Layer and shall not accept any PA Symbols from the Data Link Layer. The PA Layer shall be able to receive a PACP_TEST_MODE_req frame until the end of Phase 0 of the Link Startup Sequence. This capability provides a mechanism for test equipment to bypass the Link Startup Sequence of the UniPro stack. In addition, the PA Layer may be able to receive the frame after Phase 0, but shall not receive a PACP_TEST_MODE_req frame after completing the Link Startup Sequence.

A UniPro Device shall implement the PHY test mode as DUT, and may implement the PHY test mode acting as Tester.

A DUT shall implement the following operations in the PHY test mode:

- Lane merging, distribution, and synchronization
- Reception and decoding of the following PACP frames:
    - PACP_TEST_MODE_req
    - PACP_GET_req and PACP_SET_req
    - PACP_TEST_DATA
- Transmission and encoding of the following PACP frames:
    - PACP_GET_cnf
    - PACP_SET_cnf in response to a PACP_SET_req with the Cnf flag set to `1´
    - PACP_TEST_DATA

A Tester shall implement the following operations in addition to above list:

- Reception and decoding of the following PACP frames:
    - PACP_GET_cnf
    - PACP_SET_cnf
- Transmission and encoding of the following PACP frames:
    - PACP_GET_req
    - PACP_SET_req (mandated: Cnf flag = `0´)
    - PACP_TEST_MODE_req

The Tester prepares entry into PHY test mode by applying a reset to the DUT and a PA_LM_RESET.req followed by a PA_LM_ENABLE_LAYER.req. Then, the Tester drives the outbound M-TXs out of HIBERN8 by applying PA_LM_SET.req(, TX_HIBERN8_Control, "EXIT",) to all available M-TX Lanes, followed by a PA_LM_SET.req(, PA_PHYTestControl, CfgReady=1,). The Tester waits for the M-TX to complete the exit from HIBERN8, then continues requesting test mode entry via PA_LM_TEST_MODE_req. The PA Layer shall send the PACP_TEST_MODE_req frame on a single Lane before or during the Link Startup Sequence initiated by a DUT. The reception of the frame shall cause the PA

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Layer to cancel an active Link Startup Sequence immediately and enter into test mode. The PA Layer maps the inbound Logical Lane number 0 to the single Lane at which the PACP_TEST_MODE_req is received.

The DUT may signal successful entry into test mode by waking up its own outbound M-TXs, if it has not yet started the Link Startup sequence.

The PA Layer shall exit from the test-mode with power-on reset or with PA_LM_RESET.req.

The following information, which is normally gathered during the Link Startup Sequence, shall be manually set via PACP_SET_req frames:

- PA_ConnectedTxDataLanes
- PA_ConnectedRxDataLanes
- PA_ActiveTxDataLanes
- PA_ActiveRxDataLanes
- PA_LogicalLaneMap
- PA_SleepNoConfigTime
- PA_StallNoConfigTime

In the test mode, the M-PHY MODULE's power configuration is not handled by the PA Layer's normal Power Mode change operation. Instead, the M-PHY MODULE's configuration shall be set via PACP_SET_req frames directly to the M-PHY MODULE's Attributes. Therefore, the PA Layer shall allow all access to M-PHY MODULE's Attributes, including those that are specified to be write-protected in the normal mode. The M-CTRL-CFGREADY.request primitive is controlled via PA_PHYTestControl.

M-TXs shall be controlled as follows:

- If ContBurst in PA_PHYTestControl is set to '0', the PA Layer TX shall create a new burst for each transmitted PACP frame. When there is no data to be sent, the M-TX shall be in SAVE state.
- If ContBurst is set to '1' in PA_PHYTestControl, the PA Layer TX shall keep the M-TX in BURST mode. When there is no data to be sent, the PA Layer TX shall send FILLERs. PACP frames comprising payload data with repetitive character, like CRPAT or CJTPAT, should be sent back to back with a minimum amount of FILLERs being inserted between successive Frames.
- When LineReset is set to '1' in PA_PHYTestControl, the PA Layer TX shall issue an M-CTRL-LINERESET.request on all PA_ConnectedTxDataLanes. Note, the bit is automatically cleared.
- When CfgReady is set to '1' in PA_PHYTestControl, the PA Layer TX shall issue an M-CTRL-CFGREADY.request to all M-TXs and M-RXs. Note, the bit is automatically cleared.
- When Adapt is set to '1' in PA_PHYTestControl, the PA Layer TX shall issue a M-LANE-AdaptStart.request becoming effective on the next outbound BURST on all PA_ActiveTxDataLanes. Before applying this setting, all Adapt related attributes in M-PHY must be configured in local M-TX and peer M-RX. Adapt condition checks for Mode and Gear prerequisites are not automated. Note that the bit is automatically cleared after the M-TX has finished Adapt.
- Scrambling should always be switched off.

In the case that multiple bits in PA_PHYTestControl are set, the bits should be processed in the above order. An implementation may restrict the setting of certain multiple bit combinations.

In test mode, Link timing is no longer controlled by the PA Layer, but by the Tester Application. For this reason, the Tester Application shall respect necessary processing latencies inferred by the Device Under Test, applying gap times between subsequent PACP frames.

The PA Layer shall send test patterns when TestPatternTransmit is set in PA_PHYTestControl. The PA Layer shall encode the test pattern data into a PACP_TEST_DATA frame as shown in ***Table 19***. The PA Layer supports two Test Patterns, CJTPAT and CRPAT with four variations for different lane counts. The generated test pattern sequence is selected with the TestPatternSelect parameter in PA_PHYTestControl. The PA Layer shall generate the CJTPAT sequence when TestPatternSelect is set to'0' and the CRPAT sequence when TestPatternSelect is set to '1'. The PA Layer shall select the variation based on the value in

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

PA_ActiveTxDataLanes. The CJTPAT and CRPAT test sequences are given in ***Section 5.7.15.1*** and ***Section 5.7.15.2,*** respectively.

Table 19 PHY Test Pattern PACP frames

Pattern	Active Lanes	PACP frame	Payload	CRC-16
CJTPAT	1	PACP_TEST_DATA_0	CJTPAT_1	0xAE43
CRPAT			CRPAT_1	0xFFA6
CJTPAT	2	PACP_TEST_DATA_1	CJTPAT_2	0x9AE7
CRPAT			CRPAT_2	0xE543
CJTPAT	3	PACP_TEST_DATA_2	CJTPAT_3	0x6A6C
CRPAT			CRPAT_3	0xBBBE
CJTPAT	4	PACP_TEST_DATA_3	CJTPAT_4	0x68D4
CRPAT			CRPAT_4	0x43E6

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.7.15.1 CJTPAT Sequences

The payload of the CJTPAT Test Pattern PACP frames shall be as specified in ***Table 20***. The left column specifies the PA Layer Symbol while the value in the CJTPAT_n column specifies how many times that symbol is transmitted. Empty cell equals zero (i.e. the symbol is not transmitted at all). These symbols shall be transmitted in the order shown in the table from top to bottom.

Table 20 CJTPAT Test Patterns

PA Symbol	Repeat Count			
	CJTPAT_1	CJTPAT_2	CJTPAT_3	CJTPAT_4
0x0ABD	–	–	1	–
0x0AFD	–	–	–	2
0x7E7E	21	42	63	84
0x7E74	1	2	3	4
0x7EAB	1	2	3	4
0xB5B5	6	12	18	24
0xB55E	1	2	3	4
0x4A7E	1	2	3	4
0x7E7E	21	42	63	84
0x7E6B	1	2	3	4
0x7E54	1	2	3	4
0x4A4A	6	12	18	24
0x4ABE	1	2	3	4
0xB57E	1	2	3	4
0x9AAB	1	1	1	2
0x9AE7	–	–	1	–
0x0AAB	–	–	–	1

5.7.15.2 CRPAT Sequences

The payload of the CRPAT Test Pattern PACP frames shall be as follows:

- CRPAT_1: 10*(ABCDEF) + 1*(ABG)
- CRPAT_2: 10*(ABBCCDDEEFFA) + 1*(ABBCG)
- CRPAT_3: 1*(H) + 10*(ABCBCDCDEDEFEFAFAC) + 1*(ABCBCDGG)
- CRPAT_4: 2*(K) + 10*(ABCDBCDECDEFDEFAEFABFABC) + 1*(ABCDBCDEGLG)

In the above sequences, the number defines the repeat count of the symbol vectors given in the parenthesis. The symbols are abbreviated to capital letters for editorial reason. The mapping of letters to PA Symbols is given in ***Table 21***.

Table 21 CRPAT Test Pattern PA Symbols

A	B	C	D	E	F	G	H	K	L
0xB314	0x5EFB	0x3559	0xBED7	0x2347	0x6B8F	0xAAB8	0x0ABD	0x0AFD	0xAAB1

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.7.16 Physical Layer Requirements

UniPro requires an M-PHY-based Physical Layer to fulfill the following requirements:

- All MODULEs are M-PHY Type-I MODULEs
- PWM-G1 is mandatory
- A Link has a minimum of one Lane per direction, and a maximum of four Lanes per direction
- All capabilities of all M-RXs are equal
- All capabilities of all M-TXs are equal
- All M-TXs within an M-PORT shall share the same reference clock
- The maximum duration of TLINE-RESET generated inside M-TX does not exceed 20ms
- The PA Layer shall permanently use TActivateControl = "ProtocolControlled"
- The capability PA_SaveConfigTime indicates the time that the peer requires in order to be able to receive the next Burst. This includes M-PHY configuration and implementation specific timings induced by peer M-RX re-configuration.

An M-PHY-based Physical Layer shall implement the Lane-to-Lane skew values defined in ***Table 22***.

Table 22 Lane-to-Lane Skew Values for an M-PHY-Based Physical Layer

Parameter	Values		Unit	Description
	Min.	Max.		
$T_{L2L\_SKEW\_HS\_TX}$	0	10	UI_{HS}	Lane-to-Lane Skew for an M-TX in HS_MODE and HS-G1 to HS-G3 (see ***[MIPI02]***).
$T_{L2L\_SKEW\_HSG4\_TX}$	0	20	UI_{HS}	Lane-to-Lane Skew for an M-TX in HS_MODE and HS-G4
$T_{L2L\_SKEW\_PWM\_TX}$	0	10	T_{PWM_TX}	Lane-to-Lane Skew for an M-TX in PWM-MODE (see ***[MIPI02]***).
$T_{L2L\_SKEW\_HS\_RX}$	0	30	UI_{HS}	Lane-to-Lane Skew for an M-RX in HS_MODE and HS-G1 to HS-G3 (see ***[MIPI02]***).
$T_{L2L\_SKEW\_HSG4\_RX}$	0	60	UI_{HS}	Lane-to-Lane Skew for an M-RX in HS_MODE and HS-G4.
$T_{L2L\_SKEW\_PWM\_RX}$	0	30	T_{PWM_TX}	Lane-to-Lane Skew for an M-RX in PWM-MODE (see ***[MIPI02]***).

5.8 Management Information Base and Protocol Constants

Table 24, ***Table 25***, and ***Table 26*** show PHY Adapter Layer Attributes that may be accessed via PA_LM_GET and PA_LM_SET primitives.

For the current version of UniPro, settable PHY Adapter Layer Attributes shall be set prior to normal operation, e.g. during system initialization.

All PHY Adapter Layer Attributes should be readable.

A PA_LM_GET.req to PA_ActiveTxDataLanes, PA_ActiveRxDataLanes, PA_PWRMode, PA_TxGear, PA_RxGear, PA_HSSeries, PA_Scrambling, PA_TxHsAdaptType, PA_TxTermination, and PA_RxTermination Attributes shall return the value of the currently active parameter of the Link. Therefore, the value returned from a PA_LM_GET.req of these Attributes is not necessarily the value that was previously set.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.8.1 PHY Adapter Common Attributes

3441 **Table 23 PHY Adapter Protocol Constants**

Constant	Description	Type	Unit	Value
PA_MaxDataLanes	The maximum number of PHY data Lanes supported by the PA Layer	Integer	Lane	4
PA_PeerWakeTimeout	The maximum time allowed for the peer to complete hibernate exit	Integer	ms	100

3442 **Table 24 PHY Adapter (gettable, settable) Common Attributes**

Attribute	Attribute ID	Description	Type	Unit	Valid Attribute Value(s)	Mandatory Value(s)	Reset Value
PA_ActiveTxDataLanes	0x1560	Active TX Data Lanes	Integer	Lane	1 to PA_AvailTxDataLanes	1	1
PA_TxTrailingClocks	0x1564	Number of PHY byte clock cycles forced without data before a mode change from FAST_STATE or SLOW_STATE to SLEEP_STATE	Integer	Symbol	0 to 255	255	255
PA_ActiveRxDataLanes	0x1580	Active RX Data Lanes	Integer	Lane	1 to PA_AvailRxDataLanes	1	1

3443 **Table 25 PHY Adapter (gettable, static) Common Attributes**

Attribute	Attribute ID	Description	Unit	Valid Attribute Value(s)
PA_PHY_Type	0x1500	PHY Type	Enum	M-PHY = 1
PA_AvailTxDataLanes	0x1520	Available TX Data Lanes	Integer	1 to PA_MaxDataLanes
PA_AvailRxDataLanes	0x1540	Available RX Data Lanes	Integer	1 to PA_MaxDataLanes
PA_MinRxTrailingClocks	0x1543	Minimum number of PHY byte clock cycles without data to receive before a mode change from FAST_STATE or SLOW_STATE to SLEEP_STATE, or before a pause in data transmission while in FAST_STATE or SLOW_STATE	Integer	0 to 255

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

3444 **Table 26 PHY Adapter (gettable, dynamic) Common Attributes**

Attribute	Attribute ID	Description	Unit	Valid Attribute Value(s)

PA_TxPWRStatus	0x1567	TX Power State status	Enum	OFF_STATE = 0
				FAST_STATE = 1
				SLOW_STATE = 2
				HIBERNATE_STATE = 3
				SLEEP_STATE = 4

PA_RxPWRStatus	0x1582	RX Power State status	Enum	OFF_STATE = 0
				FAST_STATE = 1
				SLOW_STATE = 2
				HIBERNATE_STATE = 3
				SLEEP_STATE = 4

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.8.2 PHY Adapter M-PHY-Specific Attributes

3445 **Table 27 PHY Adapter (gettable, dynamic) M-PHY-Specific Attributes**

Attribute	Attribute ID	Description	Unit	Valid Attribute Value(s)
PA_RemoteVerInfo	0x15A0	Peer Device version information. Available after a successful Link StartUp Sequence.	16-bit word	See ***Section 9.9***

3446 **Table 28 PHY Adapter (gettable, settable) M-PHY-Specific Attributes**

Attribute	Attribute ID	Description	Type	Unit	Valid Attribute Value(s)	Mandatory Value(s)	Reset Value
PA_TxHsG1SyncLength	0x1552	Start of Burst: High Speed Synchronization pattern Length in HS-G1 Follows ***[MIPI02]*** definition for TX_HS_SYNC_LENGTH	Integer	–	Bit [7:6]: 0 to 1 Bit [5:0]: 0 to 15 A value of 255 is also valid, but indicates non-existence of this Gear at the peer Device		Bit [7:6]: 1 Bit [5:0]: 15
PA_TxHsG1PrepareLength	0x1553	Start of Burst: Minimum Prepare time in HS-G1 Follows ***[MIPI02]*** definition for TX_HS_PREPARE_LENGTH	Integer	–	0 to 15 A value of 255 is also valid, but indicates non-existence of this Gear at the peer Device		15
PA_TxHsG2SyncLength	0x1554	Start of Burst: High Speed Synchronization pattern Length in HS-G2 Follows ***[MIPI02]*** definition for TX_HS_SYNC_LENGTH	Integer	–	Bit [7:6]: 0 to 1 Bit [5:0]: 0 to 15 A value of 255 is also valid, but indicates non-existence of this Gear at the peer Device		Bit [7:6]: 1 Bit [5:0]: 15
PA_TxHsG2PrepareLength	0x1555	Start of Burst: Minimum Prepare time in HS-G2 Follows ***[MIPI02]*** definition for TX_HS_PREPARE_LENGTH	Integer	–	0 to 15 A value of 255 is also valid, but indicates non-existence of this Gear at the peer Device		15
PA_TxHsG3SyncLength	0x1556	Start of Burst: High Speed Synchronization pattern Length in HS-G3 Follows ***[MIPI02]*** definition for TX_HS_SYNC_LENGTH	Integer	–	Bit [7:6]: 0 to 1 Bit [5:0]: 0 to 15 A value of 255 is also valid, but indicates non-existence of this Gear at the peer Device		Bit [7:6]: 1 Bit [5:0]: 15

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Attribute	Attribute ID	Description	Type	Unit	Valid Attribute Value(s)	Mandatory Value(s)	Reset Value
PA_TxHsG3PrepareLength	0x1557	Start of Burst: Minimum Prepare time in HS-G3 Follows ***[MIPI02]*** definition for TX_HS_PREPARE_LENGTH	Integer	–	0 to 15 A value of 255 is also valid, but indicates non-existence of this Gear at the peer Device		15
PA_TxMk2Extension	0x155A	Peer supports MK2 signaling (inbound direction)	Enum	–	0=unsupported 1=supported	0	0= unsupported
PA_PeerScrambling	0x155B	Peer supports scrambling	Enum	–	0=unsupported 1=supported	0,1	0= unsupported
PA_TxSkip	0x155C	Peer indicates requirement for skip symbol insertion (Set during Link Startup)	Enum	–	0=not required 1=required	0,1	not required
PA_TxSkipPeriod	0x155D	A skip symbol <MK4,MK4> shall be inserted at least every interval with a number of PA_PDUs equal to PA_TxSkipPeriod The value 250 should correspond to +/-2000ppm difference between local and peer	Integer	–	200-10000	–	250
PA_Local_TX_LCC_Enable	0x155E	Local M-PHY Line Configuration Enable	Boolean	–	Fixed to 0=NO 1=YES is deprecated	0	0=NO
PA_Peer_TX_LCC_Enable	0x155F	Peer M-PHY Line Configuration Enable Updated on reception of PACP_CAP_EXT1_ind frame	Boolean	–	0=NO 1=YES is deprecated	0	0=NO
PA_ConnectedTxDataLanes	0x1561	Number of TX Data Lanes connected	Integer	–	0 to PA_Max DataLanes	0 to PA_AvailTx DataLanes	0

Attribute	Attribute ID	Description	Type	Unit	Valid Attribute Value(s)	Mandatory Value(s)	Reset Value
PA_TxGear	0x1568	TX Gear in PWM or HS Mode	Enum	–	PWM_G1 =1 PWM_G2 =2 PWM_G3 =3 PWM_G4 =4 PWM_G5 =5 PWM_G6 =6 PWM_G7 =7 HS_G1=1 HS_G2=2 HS_G3=3 HS_G4=4	PWM_G1	PWM_G1
PA_TxTermination	0x1569	TX Termination	Enum	–	OFF=0 ON=1	OFF ON	OFF
PA_HSSeries	0x156A	TX and RX Frequency Series in High Speed Mode	Enum	–	A=1 B=2		A
PA_PWRMode[4]	0x1571	TX/RX Power Mode: TX[b3:b0] RX[b7:b4] Setting of this Attribute triggers the Link Configuration procedure (see ***Section 5.7.12***).	Enum	–	Fast_Mode=1 Slow_Mode=2 FastAuto_Mode =4 SlowAuto_Mode =5	Slow_Mode, SlowAuto_ Mode	SlowAuto_ Mode
PA_ConnectedRxDataLanes[1]	0x1581	Number of RX Data Lanes connected	Integer	–	0 to PA_Max DataLanes	0 to PA_AvailRxD ataLanes	0

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Attribute	Attribute ID	Description	Type	Unit	Valid Attribute Value(s)	Mandatory Value(s)	Reset Value
PA_RxGear	0x1583	RX Gear in PWM or HS Mode	Enum	–	PWM_G1 =1 PWM_G2 =2 PWM_G3 =3 PWM_G4 =4 PWM_G5 =5 PWM_G6 =6 PWM_G7 =7 HS_G1=1 HS_G2=2 HS_G3=3 HS_G4=4	PWM_G1	PWM_G1
PA_RxTermination	0x1584	RX Termination	Enum	–	OFF=0 ON=1	OFF, ON	OFF
PA_Scrambling	0x1585	Scrambling Request	Enum	–	OFF=0 ON=1	OFF,ON	OFF
PA_MaxRxPWMGear	0x1586	Maximum RX Low Speed Gears (PWM)	Integer	–	PWM_G1 =1 PWM_G2 =2 PWM_G3 =3 PWM_G4 =4 PWM_G5 =5 PWM_G6 =6 PWM_G7 =7	PWM_G1 to RX_PWM GEAR_ Capability	PWM_G1
PA_MaxRxHSGear	0x1587	Maximum RX High Speed Gears (HS) 0 means no HS available	Integer	–	NO_HS=0 HS_G1=1 HS_G2=2 HS_G3=3 HS_G4=4	NO_HS to RX_HS GEAR_ Capability	NO_HS

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Attribute	Attribute ID	Description	Type	Unit	Valid Attribute Value(s)	Mandatory Value(s)	Reset Value
PA_PACPReqTimeout	0x1590	Expiration value of the PACP_REQUEST_TIMER when the PA Layer is waiting for a cnf Message (see ***Section 5.7.12.6*** & ***Section 5.7.14.3***). The actual duration of the timeout period shall be the value set in the Attribute ±10%	Integer	ms	1 to 63		63
PA_PACPReqEoBTimeout	0x1591	Expiration value of the PACP_REQUEST_TIMER when the PA Layer is waiting for the end of burst (see ***Section 5.7.12.6)*** The actual duration of the timeout period shall be the value set in the Attribute ±10%	Integer	ms	1 to 15		15
PA_LogicalLaneMap[1]	0x15A1	Logical to Physical Lane mapping Attribute contains a valid value after a successful Link StartUp Sequence **TX Lanes:** Bits [1:0]: Physical Lane of Logical Lane 0 Bits [3:2]: Physical Lane of Logical Lane 1 Bits [5:4]: Physical Lane of Logical Lane 2 Bits [7:6]: Physical Lane of Logical Lane 3 **RX Lanes:** Bits [9:8]: Physical Lane of Logical Lane 0 Bits [11:10]: Physical Lane of Logical Lane 1 Bits [13:12]: Physical Lane of Logical Lane 2 Bits [15:14]: Physical Lane of Logical Lane 3	16-bit word	–	Implementation-specific		Any
PA_SleepNoConfigTime[1]	0x15A2	Minimum time to wait between bursts in PWM mode when no new configuration was performed	Integer	SI	1 to 15		15
PA_StallNoConfigTime[1]	0x15A3	Minimum time to wait between bursts in HS mode when no new configuration was performed	Integer	SI	1 to 255		255
PA_SaveConfigTime[1]	0x15A4	Minimum time to wait between bursts when a new configuration was performed	Integer	40 ns	1 to 250[3]		250

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Attribute	Attribute ID	Description	Type	Unit	Valid Attribute Value(s)	Mandatory Value(s)	Reset Value
PA_RxHSUnterminationCapability	0x15A5	Specifies whether or not the inbound Link supports unterminated line in HS mode	Boolean	–	FALSE=0 TRUE=1	FALSE	FALSE
PA_RxLSTerminationCapability	0x15A6	Specifies whether or not the inbound Link supports terminated line in PWM mode	Boolean	–	FALSE=0 TRUE=1	FALSE	FALSE
PA_Hibern8Time	0x15A7	Minimum time to wait in HIBERN8 before allowing a M-TX to exit HIBERN8	Integer	PA_ Granularity	0 to 10000		128
PA_TActivate	0x15A8	Time to wait before activating a burst in order to wake-up peer M-RX	Integer	PA_ Granularity	1 to 10000		16
PA_LocalVerInfo	0x15A9	Local version info Delivered during Link Startup Sequence to the peer Device and stored in PA_RemoteVerInfo in the peer Device.	16-bit word	–	See ***Section 9.9***		0
PA_Granularity	0x15AA	Granularity for PA_TActivate and PA_Hibern8Time	Integer	μs	1=1 μs 2=4 μs 3=8 μs 4=16 μs 5=32 μs 6=100 μs		6
PA_MK2ExtensionGuardBand	0x15AB	MK2 extension time	Integer	PA_ Granularity	0 to 10000	0	0
PA_PWRModeUserData [0 to 11]	0x15B0 to 0x15BB	Data to be sent within PACP_PWR_req and delivered to the peer DME	12x 16-bit word	–	See ***Table 102***		0
PA_PACPFrameCount[2]	0x15C0	Number of valid PACP frames received	Integer	–	0 to 4294967295		0
PA_PACPErrorCount[2]	0x15C1	Number of erroneous PACP frames received	Integer	–	0 to 4294967295		0

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Attribute	Attribute ID	Description	Type	Unit	Valid Attribute Value(s)	Mandatory Value(s)	Reset Value
PA_PHYTestControl	0x15C2	PHY Test Feature control register b0 : ContBurst b1 : CfgReady b2 : LineReset b3 : TestPatternTransmit b4 : TestPatternSelect b5 : Adapt Setting this Attribute has side-effects (see ***Section 5.7.15)***.	6-bit word	–	All		0
PA_TxHsG4SyncLength	0x15D0	Start of Burst: High Speed Synchronization pattern Length in HS-G4 Follows ***[MIPI02]*** definition for TX_HS_SYNC_LENGTH	Integer	–	Bit [7:6]: 0 to 1 Bit [5:0]: 0 to 15 A value of 255 is also valid, but indicates non-existence of this Gear at the peer Device		Bit [7:6]: 1 Bit [5:0]: 15
PA_TxHsG4PrepareLength	0x15D1	Start of Burst: Minimum Prepare time in HS-G4 Follows ***[MIPI02]*** definition for TX_HS_PREPARE_LENGTH	Integer	–	0 to 15 A value of 255 is also valid, but indicates non-existence of this Gear at the peer Device		15
PA_PeerRxHsAdaptRefresh	0x15D2	Start of Burst: High Speed adaption pattern Length Follows ***[MIPI02]*** definition for TX_HS_ADAPT_LENGTH when M-PHY is configured for ADAPT state.	Integer	–	Bit [7]: 0 to 1 Bit [6:0]: 0 to 127 A value of 255 is also valid, but indicates non-existence of ADAPT support at the peer device receiver		0
PA_PeerRxHsAdaptInitial	0x15D3	Start of Burst: High Speed adaption pattern Length Follows ***[MIPI02]*** definition for TX_HS_ADAPT_LENGTH when M-PHY is configured for ADAPT state.	Integer	–	Bit [7]: 0 to 1 Bit [6:0]: 0 to 127 A value of 255 is also valid, but indicates non-existence of ADAPT support at the peer device receiver		0
PA_TxHsAdaptType	0x15D4	Adapt Type selector (See ***Section 5.7.12.4***)	Enum	–	00 = REFRESH 01 = INITIAL 10 = Reserved 11 = No ADAPT		11

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Attribute	Attribute ID	Description	Type	Unit	Valid Attribute Value(s)	Mandatory Value(s)	Reset Value
PA_AdaptAfterLRSTInPA_INIT[5]	0x15D5	Adapt Type selector This value is used in PACP_PWR_req which is transmitted in SlowAuto_Mode as part of PA_INIT (see ***Section 5.7.10***)	Enum	–	00 = REFRESH 01 = INITIAL 10 = Previous ADAPT value 11 = No ADAPT		10

Note:

1. *Attribute should be set only when needed by PHY Testing (see **Section 5.7.15**).*
2. *Statistic. See **Section 9.3** for reset behavior.*
3. *For backward compatibility with UniPro v1.6x the range is limited to 250. Implementers may choose to extend the range to account for application specific requirements.*
4. *Starting from UniPro v1.8, it is deprecated for the Application to issue power mode change requests with PWR_MODE set to 7 (UNCHANGED). The UniPro PA layer shall reject such requests locally. This means that if the PA layer receives PA_LM_SET.req(AttrSetType, MIBattribute, MIBvalue, SelectorIndes) with MIBAttribute = PA_PWRMode and MIBvalue= UNCHANGED, then the PA layer shall generate PA_LM_SET.cnf_L() with ConfigResultCode=INVALID_MIB_ATTRIBUTE_VALUE. However, for backward compatibility in cases where UniPro v1.8 is able to communicate with an earlier UniPro version, the PA receiver must support Power Mode Change requests with PWR_MODE set to UNCHANGED.*
5. *If PA_AdaptAfterLRSTInPA_INIT == 00, then the 'Adapt' field in PACP_PWR_req is set to 00 (REFRESH)*
   if PA_AdaptAfterLRSTInPA_INIT == 01, then the 'Adapt' field in PACP_PWR_req is set to 01 (INITIAL)
   if PA_AdaptAfterLRSTInPA_INIT == 11, then the 'Adapt' field in PACP_PWR_req is set to 11 (No ADAPT)
   if PA_AdaptAfterLRSTInPA_INIT == 10, then the following values shall be used:
      'Adapt' = 00 in PACP_PWR_req if previous PMC had REFRESH as 'Adapt'.
      'Adapt' = 01 in PACP_PWR_req if previous PMC had INITAL as 'Adapt'.
      'Adapt' = 11 in PACP_PWR_req if previous PMC had 'NO ADAPT' as 'Adapt'

This page intentionally left blank.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

6 Data Link Layer (L2)

The principal objective of this section is to specify the characteristics and behavior of a conceptual Data Link Layer service and Data Link Layer protocol.

This section does not specify individual implementation or products, nor does it constrain the implementation of Data Link Layer and its interfaces within a system.

Figure 53 shows the SAP model used for the Data Link Layer. The DL Layer service to the Network Layer is provided through two Traffic Class-specific Service Access Points (DL_TCx_SAP). The DL Layer in turn relies on the service provided by the PA_SAP. The Data Link Layer Management SAP (DL_LM_SAP) is provided to the Device Management Entity (DME) for configuration and control purposes.

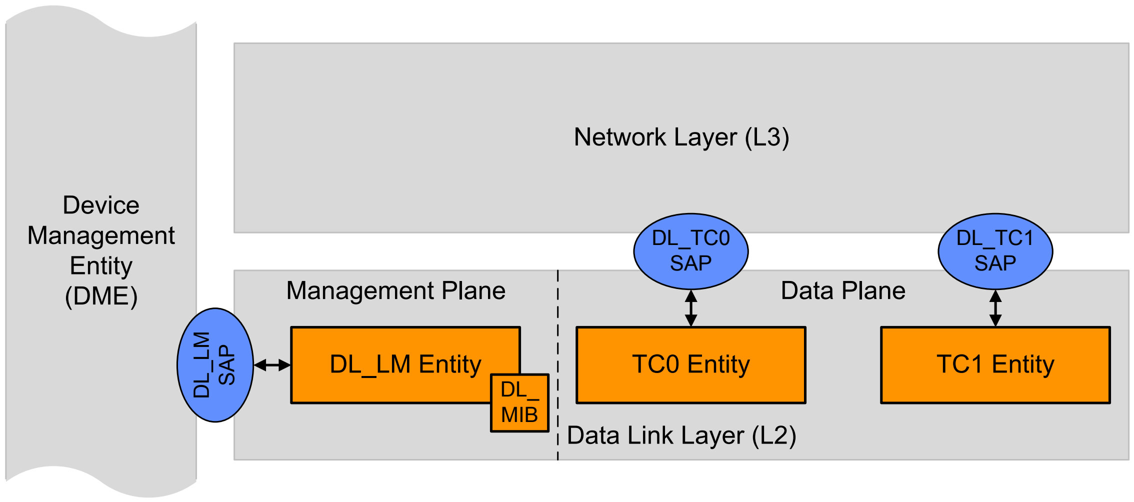


Figure 53 Data Link Layer SAP Model

6.1 Data Link Layer Service Functionality and Features

The DL Layer provides services to assure the transparent and reliable transfer of user-data between DL Service Users.

The way in which the supporting communication resources are utilized to achieve this transfer is invisible to the DL Service Users. The DL Layer provides following key features.

Key features for transmit direction:

- Frame composition
- Optional Frame preemption
- Triggering of PHY initialization
- Flow control
- Support for up to two Traffic Classes by priority-based arbitration, Traffic Class 0 shall always be present
- CRC generation
- Retransmission of a Frame in case of errors

Key features for receive direction:

- Frame decomposition
- Capability to receive preempted Frames
- Generation of flow control credit information
- Support for two Traffic Classes reception
- CRC verification

- Detection of various errors and autonomous reaction to them
- Autonomous acknowledgment of all unacknowledged Frames

6.2 Services Assumed from PHY Adapter Layer

The DL Layer assumes following services from PHY Adapter Layer to fulfill its services to DL Service User:

- Sending and receiving a control symbol
- Sending and receiving a data symbol
- PHY initialization

6.3 DL_TCx_SAP

Services to a DL Service User are provided via the DL_TCx_SAP. Each Traffic Class uses a dedicated Service Access Point (SAP) for transferring data. The two defined SAPs are DL_TC0_SAP and DL_TC1_SAP. The Traffic Class identification is performed based on the used SAP.

At the sending side, data is passed via the DL_TCx_SAP in DL_SDUs to the DL Layer to transfer it to a peer DL Layer. At the recipient side, the DL Layer delivers received data in DL_SDUs to the upper layer.

6.3.1 Data Transfer Primitives

The primitives covered in this section are listed in ***Table 29***.

Table 29 DL_TCx_SAP Data Transfer Primitives

Name	Request	Indication	Local Response	Response	Local Confirm	Confirm
DL_DATA	***6.3.1.1***	***6.3.1.3***	***6.3.1.4***	–	***6.3.1.2***	–

Table 30 lists the parameters that appear in the DL_TCx_SAP primitives.

Table 30 DL_TCx_SAP Primitive Parameters

Name	Type	Valid Range	Value	Description
DL_SDU	byte stream	1 to DL_MTU		Specifies the length of the data passing through the DL_TCx_SAP before transmission or after reception
L2ResultCode	Enumeration	SUCCESS	0	Indicates the result of the DL_DATA.req request.
		NO_PEER_TC	1	

6.3.1.1 DL_DATA.req

This primitive is used to send a DL_SDU over a dedicated TC of the DL Layer. The TC is given by the SAP used. The user may transmit any integral number of bytes greater than zero up to the DL_MTU.

The semantics of this primitive are:

DL_DATA.req(DL_SDU)

When Generated

The DL Service User shall generate a DL_DATA.req primitive to send a DL_SDU.

Effect on Receipt

The reception of a DL_DATA.req primitive shall cause DL Service Provider to transfer DL_SDU to its peer DL Layer.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

6.3.1.2 DL_DATA.cnf_L

This primitive informs the DL Service User that the Service Provider, L2 in this case, is ready to accept a new DL_DATA.req primitive.

The semantics of this primitive are:

DL_DATA.cnf_L(L2ResultCode)

When Generated

This primitive shall be generated by the Data Link Service Provider after the receipt of a DL_DATA.req primitive, when the DL Service Provider can accept a new request to transfer a DL_SDU.

If DL_PeerTCxPresent is TRUE, i.e. the peer Device implements TCx, the Data Link Service Provider shall set L2ResultCode to SUCCESS.

If DL_PeerTCxPresent is FALSE, i.e. the peer Device does not implement TCx, the Data Link Service Provider shall set L2ResultCode to NO_PEER_TC. The DL_SDU provided by DL_DATA.req is ignored, meaning the DL_SDU is not put in the transmitter logical buffer, and is not sent on the Link.

Effect on Receipt

Following the emission of a DL_DATA.req primitive and prior to the reception of a DL_DATA.cnf_L primitive, the DL Service User shall not emit a new DL_DATA.req primitive.

The DL Service User may emit a new DL_DATA.req primitive immediately following a reset or after the reception of the DL_DATA.cnf_L primitive corresponding to a previously emitted DL_DATA.req primitive.

6.3.1.3 DL_DATA.ind

At the local RX the DL Layer Service Provider shall pass the received DL_SDU to the DL Layer Service User using this primitive via the SAP of appropriate Traffic Class. The DL_SDU may consist of any integral number of bytes greater than zero up to DL_MTU.

The semantics of this primitive are:

DL_DATA.ind(DL_SDU)

When Generated

The primitive shall be generated when the DL Layer Service Provider receives a DL_PDU at the local RX. The DL_SDU shall only be provided to the DL Layer Service User after the completion of the Data Link Layer Initialization (see ***Section 6.7***).

Effect on Receipt

Upon reception of a DL_DATA.ind primitive, the DL Layer Service User shall consume DL_SDU on a Traffic Class associated with the SAP.

6.3.1.4 DL_DATA.rsp_L

This primitive informs the Service Provider that the DL Service User, L3 in this case, is ready to accept a new DL_DATA.ind or DL_ESCAPED_DATA.ind primitive.

The semantics of this primitive are:

DL_DATA.rsp_L( )

When Generated

The DL Service User shall generate DL_DATA.rsp_L in response to a DL_DATA.ind to indicate that the DL Service User can accept and process a new DL_PDU.

Effect on Receipt

Following the emission of a DL_DATA.ind primitive and prior to the reception of a DL_DATA.rsp_L primitive, the Data Link Layer shall not emit a new DL_DATA.ind primitive. The Data Link Layer may emit a new DL_DATA.ind primitive only just after reset, or after reception of the DL_DATA.rsp_L primitive corresponding to a previously emitted DL_DATA.ind primitive.

6.4 DL_LM_SAP

The Data Link Layer Management (DL_LM) SAP offers three groups of Service Primitives: Configuration primitives, Control primitives and Status primitives. The primitives on the DL_LM_SAP are used by the Device Management Entity (DME) to configure and control the layer and receive layer status information.

The Configuration primitives enable access to configuration information specific to the DL Layer. This information is represented as a DL Layer-specific Management Information Base (DL_MIB). The DL_MIB is regarded as "contained" within the DL_LM entity. Multiple accesses to the DL_MIB via the Configuration primitives shall not occur concurrently. The available Data Link Layer Attributes are defined in ***Section 6.8.***

The Control primitives provide direct control of the DL Layer. Control primitives are generated by the DME and can occur concurrently.

The Status primitives indicate status information of the DL Layer. Status primitives are generated by the DL Layer and can occur concurrently.

6.4.1 Configuration Primitives

The DL_LM configuration primitives, GET and SET, are used by the DME to retrieve and store values, respectively, for the configuration Attributes in the DL_MIB.

The GET and SET primitives are represented as requests with associated confirm primitives. These primitives are prefixed by DL_LM. The primitives are summarized in ***Table 31***.

Table 31 DL_LM Configuration Primitives

Name	Request	Indication	Local Response	Response	Local Confirm	Confirm
DL_LM_GET	***6.4.1.1***	–	–	–	***6.4.1.2***	–
DL_LM_SET	***6.4.1.3***	–	–	–	***6.4.1.4***	–

The parameters used for these primitives are defined in ***Table 32***.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Table 32 DL_LM Configuration Primitive Parameters

Name	Type	Valid Range	Value	Description
MIBattribute	Integer	0x2000 to 0x2FFF and the MIB Attribute within that range are defined in ***Section 6.8***	–	The address of the MIB Attribute
MIBvalue	Variable	As defined in ***Section 6.8***	–	The value of the MIB Attribute
ConfigResultCode	Enum	SUCCESS	0	Indicates the result of the request
		INVALID_MIB_ATTRIBUTE	1	
		INVALID_MIB_ATTRIBUTE_VALUE	2	
		READ_ONLY_MIB_ATTRIBUTE	3	
AttrSetType	Enum	NORMAL	0	Select whether the actual value (NORMAL) or the reset value (STATIC) of the Attribute is set
		STATIC	1	

6.4.1.1 DL_LM_GET.req

This primitive requests information about a given DL_MIB Attribute identified by MIBattribute.

The semantics of this primitive are:

```
DL_LM_GET.req( MIBattribute )
```

The primitive parameter is defined in ***Table 32***.

When Generated

This primitive is generated by the DME to obtain information from the DL_MIB.

Effect on Receipt

The DL_LM entity attempts to retrieve the requested MIB Attribute value from DL_MIB and responds with DL_LM_GET.cnf_L that gives the result.

6.4.1.2 DL_LM_GET.cnf_L

This primitive reports the results of an information request about the DL_MIB.

The semantics of this primitive are:

DL_LM_GET.cnf_L(ConfigResultCode, MIBvalue)

The primitive parameters are defined in ***Table 32***.

The DL Layer shall set ConfigResultCode in DL_LM_GET.cnf_L to one of the values shown in ***Table 33*** for the condition given in the table. The DL Layer should not set ConfigResultCode to INVALID_MIB_ATTRIBUTE_VALUE or READ_ONLY_MIB_ATTRIBUTE.

Table 33 DL_LM_GET.cnf_L ConfigResultCode Values

ConfigResultCode	Condition
SUCCESS	The request succeeds, i.e. the MIB Attribute indicated by DL_LM_GET.req MIBattribute is gettable. The DL Layer shall set MIBvalue in DL_LM_GET.cnf_L to the MIB Attribute value.
INVALID_MIB_ATTRIBUTE	The MIB Attribute indicated by DL_LM_GET.req MIBattribute is invalid or not implemented. The value of MIBvalue in DL_LM_CC_GET.cnf_L is undefined.

When Generated

The DL Layer shall generate the DL_LM_GET.cnf_L primitive in response to a DL_LM_GET.req from the DME.

Effect on Receipt

The DME is informed about the result of the operation, and in case ConfigResultCode is SUCCESS, MIBvalue carries the MIB Attribute value. For any other value of ConfigResultCode, MIBvalue is undefined.

6.4.1.3 DL_LM_SET.req

This primitive attempts to set the indicated DL_MIB Attribute indicated by MIBattribute to the MIBvalue value.

The semantics of this primitive are:

DL_LM_SET.req(AttrSetType, MIBattribute, MIBvalue)

The primitive parameters are defined in ***Table 32***.

When Generated

This primitive is generated by the DME to set the value of indicated DL_MIB Attribute.

Effect on Receipt

The DL_LM attempts to set the value of the specified MIB Attribute in its database. The DL_LM responds with DL_LM_SET.cnf_L that gives the result.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

6.4.1.4 DL_LM_SET.cnf_L

This primitive reports the results of an attempt to set the value of an Attribute in the DL_MIB.

The semantics of this primitive are:

DL_LM_SET.cnf_L(ConfigResultCode)

The primitive parameter is defined in ***Table 32***.

The DL Layer shall set ConfigResultCode in DL_LM_SET.cnf_L to one of the values shown in ***Table 34*** for the condition given in the table.

Table 34 DL_LM_SET.cnf_L ConfigResultCode Values

ConfigResultCode	Condition
SUCCESS	The request succeeds, i.e. the MIB Attribute indicated by DL_LM_SET.req MIBattribute is set to the value of DL_LM_SET.req MIBvalue.
INVALID_MIB_ATTRIBUTE	The MIB Attribute indicated by DL_LM_SET.req MIBattribute is invalid or not implemented. If AttrSetType is STATIC, the Attribute indicated by MIBattribute and SelectorIndex does not support setting its reset value.
READ_ONLY_MIB_ATTRIBUTE	The MIB Attribute indicated by DL_LM_SET.req MIBattribute exists, but is not settable.
INVALID_MIB_ATTRIBUTE_VALUE	The MIB Attribute indicated by DL_LM_SET.req MIBattribute exists and is settable, but the value of DL_LM_SET.req MIBvalue is outside of the implemented range, or outside of the valid value range, for the MIB Attribute.

When Generated

The DL Layer shall generate the DL_LM_SET.cnf_L primitive in response to a DL_LM_SET.req from the DME.

Effect on Receipt

DL_LM_SET.cnf_L confirms the success or failure of the DL_LM_SET.req at the Data Link Layer, and should have no further effects at the DME.

6.4.2 Control Primitives

The Service Primitives in this section are provided for controlling the Data Link Layer.

The primitives covered in this section are listed in ***Table 35***.

Table 35 DL_LM_SAP Control Primitives

Name	Request	Indication	Local Response	Response	Local Confirm	Confirm
DL_LM_RESET	***6.4.2.1***	–	–	–	***6.4.2.2***	–
DL_LM_ENABLE_LAYER	***6.4.2.3***	–	–	–	***6.4.2.4***	–
DL_LM_LINKSTARTUP	***6.4.2.5***	–	–	–	***6.4.2.6***	–
DL_LM_HIBERNATE_ENTER	***6.4.2.7***	–	–	–	***6.4.2.8***	–
DL_LM_HIBERNATE_EXIT	***6.4.2.9***	–	–	–	***6.4.2.10***	–
DL_LM_PRE_HIBERNATE_ EXIT	***6.4.2.11***	–	–	–	***6.4.2.12***	–

Table 36 lists the parameters that appear in the DL_LM_SAP control primitives.

Table 36 DL_LM_SAP Control Primitive Parameters

Name	Type	Valid Range	Value	Description
ResetLevel	Enum	COLD	0	Defines the reset level of the requested reset
		WARM	1	

6.4.2.1 DL_LM_RESET.req

This primitive requests to reset the Data Link Layer.

The semantics of this primitive are:

DL_LM_RESET.req(ResetLevel)

When Generated

This primitive is generated by the DME when it needs to reset the Data Link Layer.

Effect on Receipt

The DL Layer resets transmitter and receiver to the power-on reset states and values. The TCx entities discard all DL_SDUs currently processed and located in any logical buffer. Until the reset/boot completes, the Data Link Layer does not perform data transmit or receive operations.

The ResetLevel COLD resets the complete DL Layer including the statistics. The ResetLevel WARM resets the DL Layer without the statistics, if they exist.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

6.4.2.2 DL_LM_RESET.cnf_L

The DL_LM_RESET.cnf_L primitive is used during the UniPro reset procedure (see ***Section 9.11.1***).

The semantics of this primitive are:

DL_LM_RESET.cnf_L( )

When Generated

The DL uses the DL_LM_RESET.cnf_L primitive during the boot procedure to indicate to the DME that the DL came out of reset, thus being ready to exercise the L2 initialization protocol. See ***Section 6.7***.

Effect on Receipt

The DME is informed that the DL came out of reset.

6.4.2.3 DL_LM_ENABLE_LAYER.req

The DL_LM_ENABLE_LAYER.req primitive starts the L2 initialization protocol (see ***Section 6.7***) as required by the UniPro boot procedure.

The semantics of this primitive are:

DL_LM_ENABLE_LAYER.req( )

When Generated

The DL_LM_ENABLE_LAYER.req primitive is part of the UniPro boot procedure (see ***Section 9.11.2***) and is generated by the DME after the Data Link Layer comes out of reset and the PHY Adapter Layer is ready to be used.

Effect on Receipt

The DL shall reach the state where it is ready to receive the AFCs as part of the L2 initialization protocol (see ***Section 6.7***), as required by the UniPro boot procedure.

When this state is reached, this is indicated to the DME with the DL_LM_ENABLE_LAYER.cnf_L primitive.

6.4.2.4 DL_LM_ENABLE_LAYER.cnf_L

The DL_LM_ENABLE_LAYER.cnf_L primitive is used during the UniPro boot procedure (see ***Section 9.11.2***) to indicate to the DME that the DL is ready for the L2 initialization protocol.

The semantics of this primitive are:

DL_LM_ENABLE_LAYER.cnf_L( )

When Generated

The DL_LM_ENABLE_LAYER.cnf_L primitive is generated by the DL after it reached the state where the L2 initialization protocol may be started.

Effect on Receipt

The DME is informed about the readiness of the Data Link Layer for the L2 initialization protocol during the boot procedure.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

6.4.2.5 DL_LM_LINKSTARTUP.req

This primitive attempts to establish a Link to the peer Device by starting the Data Link Layer Initialization. See ***Section 6.7***

The semantics of this primitive are:

DL_LM_LINKSTARTUP.req( )

When Generated

The DME shall generate this when it wants to establish a Link to the peer Device.

Effect on Receipt

The Data Link Layer Initialization shall start.

Once the Data Link Layer Initialization is finalized, the Data Link Layer shall enter normal operation. This is indicated to the DME with the DL_LM_LINKSTARTUP.cnf_L primitive.

6.4.2.6 DL_LM_LINKSTARTUP.cnf_L

This primitive is used during the UniPro boot procedure (see ***Section 9.11.2***) to indicate to the DME that the Data Link Layer completed the Initialization and the Data Link Layer is ready to be used by the Network Layer.

The semantics of this primitive are:

DL_LM_LINKSTARTUP.cnf_L( )

When Generated

This primitive is generated by the Data Link Layer after the Data Link Layer Initialization is completed.

Effect on Receipt

The DME is informed about the completion of the Data Link Layer Initialization.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

6.4.2.7 DL_LM_HIBERNATE_ENTER.req

This primitive requests the Data Link Layer to go to Hibernate.

The semantics of this primitive are:

DL_LM_HIBERNATE_ENTER.req( )

When Generated

The DME shall generate this primitive when it wants to hibernate the Data Link Layer.

Effect on Receipt

The Data Link Layer shall first reach the Frozen state, where all its timers shall be stalled, and shall retain the value of the following Attributes:

- DL_TC0TXFCThreshold
- DL_FC0ProtectionTimeOutVal
- DL_TC0ReplayTimeOutVal
- DL_AFC0ReqTimeOutVal
- DL_AFC0CreditThreshold
- DL_TC0OutAckThreshold
- DL_TC1TXFCThreshold
- DL_FC1ProtectionTimeOutVal
- DL_TC1ReplayTimeOutVal
- DL_AFC1ReqTimeOutVal
- DL_AFC1CreditThreshold
- DL_TC1OutAckThreshold

Just before hibernating, the Data Link Layer shall indicate its intentions to the DME with the DL_LM_HIBERNATE_ENTER.cnf_L primitive. Then the Data Link Layer is hibernated.

6.4.2.8 DL_LM_HIBERNATE_ENTER.cnf_L

This primitive is used to indicate that the Data Link Layer is about to hibernate.

The semantics of this primitive are:

DL_LM_HIBERNATE_ENTER.cnf_L( )

When Generated

This primitive is generated by the Data Link Layer in response to a DL_LM_HIBERNATE_ENTER.req primitive and it indicates that the Data Link Layer is hibernating.

Effect on Receipt

The DME is informed about the completion of all necessary actions require of the Data Link Layer to hibernate.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

6.4.2.9 DL_LM_HIBERNATE_EXIT.req

This primitive is used to request the Data Link Layer to stop hibernating and return to normal operation.

The semantics of this primitive are:

DL_LM_HIBERNATE_EXIT.req( )

When Generated

The DME shall generate this primitive when it wants to unhibernate the Data Link Layer.

Effect on Receipt

The Data Link Layer shall transition to its reset state, restore the value of any Attributes retained when entering the Hibernate State (see ***Section 6.4.2.7***), then enable itself and finally shall start the Data Link Layer Initialization. When the Data Link Layer Initialization is completed, the Data Link Layer shall generate the DL_LM_HIBERNATE_EXIT.cnf_L to indicate to the DME its return to normal operation.

6.4.2.10 DL_LM_HIBERNATE_EXIT.cnf_L

This primitive is used to indicate to the DME that the Data Link Layer has returned to normal operation after hibernating.

The semantics of this primitive are:

DL_LM_HIBERNATE_EXIT.cnf_L( )

When Generated

This primitive is generated by the Data Link Layer in response to a DL_LM_HIBERNATE_EXIT.req primitive. It indicates that the Data Link Layer is not hibernating and has returned to normal operation.

Effect on Receipt

The DME is informed of the completion of all necessary actions required by the Data Link Layer to return to normal operation after hibernating.

6.4.2.11 DL_LM_PRE_HIBERNATE_EXIT.req

Support for this primitive is optional. However, if an implementation supports this primitive it shall implement it as described in this section.

This primitive is used to request the Data Link Layer to stop, or prepare to stop, hibernating, and prepare to return to normal operation. Hibernate exit is requested with the DL_LM_HIBERNATE_EXIT.req primitive.

The semantics of this primitive are:

DL_LM_PRE_HIBERNATE_EXIT.req( )

When Generated

The DME shall generate this primitive when the Data Link Layer needs to prepare to exit Hibernate prior to exiting Hibernate.

Effect on Receipt

The Data Link Layer shall enable PA_ESCDATA.ind and PA_DATA.ind processing and prepare to exit Hibernate State.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

6.4.2.12 DL_LM_PRE_HIBERNATE_EXIT.cnf_L

Support for this primitive is optional. However, if an implementation supports this primitive it shall implement it as described in this section.

This primitive is used to indicate to the DME that the Data Link Layer has been prepared to return to normal operation after hibernating and is enabled to process PA_ESCDATA.ind and PA_DATA.ind from the PA Layer correctly.

The semantics of this primitive are:

DL_LM_PRE_HIBERNATE_EXIT.cnf_L( )

When Generated

This primitive is generated by the Data Link Layer in response to a DL_LM_PRE_HIBERNATE_EXIT.req primitive. It indicates that the Data Link Layer has been prepared for return to normal operation.

Effect on Receipt

The DME is informed of the completion of all necessary actions required by the Data Link Layer to return to normal operation after preparation for exiting Hibernate.

6.4.3 Status Primitives

The Service Primitives in this section are provided for status provision by the Data Link Layer.

The primitives covered in this section are listed in ***Table 37***.

Table 37 DL_LM_SAP Status Primitives

Name	Request	Indication	Local Response	Response	Local Confirm	Confirm
DL_LM_ERROR	–	***6.4.3.1***	–	–	–	–
DL_LM_CTRL_NOFRAME	–	***6.4.3.2***	–	–	–	–
DL_LM_TC1_NOFRAME	–	***6.4.3.3***	–	–	–	–
DL_LM_TC0_NOFRAME	–	***6.4.3.4***	–	–	–	–
DL_LM_CTRL_FRAME	–	***6.4.3.5***	–	–	–	–
DL_LM_TC1_FRAME	–	***6.4.3.6***	–	–	–	–
DL_LM_TC0_FRAME	–	***6.4.3.7***	–	–	–	–

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Table 38 lists the parameters that appear in the DL_LM_SAP status primitives.

Table 38 DL_LM_SAP Primitive Parameters

Name	Type	Valid Range	Value	Description
DLErrorCode	Enum	NAC_RECEIVED	1	Indicates to the DME in case an error event occurred in the Data Link Layer
		TCx_REPLAY_TIMER_EXPIRED	2	
		AFCx_REQUEST_TIMER_EXPIRED	3	
		FCx_PROTECTION_TIMER_EXPIRED	4	
		CRC_ERROR	5	
		RX_BUFFER_OVERFLOW	6	
		MAX_FRAME_LENGTH_EXCEEDED	7	
		WRONG_SEQUENCE_NUMBER	8	
		AFC_FRAME_SYNTAX_ERROR	9	
		NAC_FRAME_SYNTAX_ERROR	10	
		EOF_SYNTAX_ERROR	11	
		FRAME_SYNTAX_ERROR	12	
		BAD_CTRL_SYMBOL_TYPE	13	
		PA_INIT_ERROR	14	
		PA_ERROR_IND_RECEIVED	15	
		PA_INIT	16	

6.4.3.1 DL_LM_ERROR.ind

The Service Primitive indicates to the DME an error event in the Data Link Layer.

The semantics of this primitive are:

DL_LM_ERROR.ind(DLErrorCode)

The primitive parameter is defined in ***Table 38***.

When Generated

This primitive is generated by the Data Link Layer when an error-related event is detected.

There are four types of events that are reported:

- Events causing a Frame retransmission
    - NAC_RECEIVED shall indicate a NAC Frame has been received
    - TCx_REPLAY_TIMER_EXPIRED shall indicate the TCx_REPLAY_TIMER has expired
- Events caused by a possible failure of AFC transmission
    - AFCx_REQUEST_TIMER_EXPIRED shall indicate the AFCx_REQUEST_TIMER has expired
    - FCx_PROTECTION_TIMER_EXPIRED shall indicate the FCx_PROTECTION_TIMER has expired
- Events caused by an error detected on the RX Link, which cause a NAC to be reported to the other side

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

    - CRC_ERROR shall indicate a CRC error has been detected on either a Data or a Control Frame
    - RX_BUFFER_OVERFLOW shall indicate that the RX buffer overflows, likely because of an erroneous SOF symbol or because an attempt was made to send data over a Traffic Class which is not implemented.
    - MAX_FRAME_LENGTH_EXCEEDED shall indicate that a Frame with a payload longer than the DL_MTU has been received
    - WRONG_SEQUENCE_NUMBER shall indicate a correct Frame with a wrong Frame Sequence Number has been received
    - AFC_FRAME_SYNTAX_ERROR shall indicate that an AFC symbol not followed by two data symbols has been received
    - NAC_FRAME_SYNTAX_ERROR shall indicate that a NAC symbol not followed by one data symbol has been received
    - EOF_SYNTAX_ERROR shall indicate that an EOF_EVEN or an EOF_ODD symbol not followed by one data symbol has been received
    - FRAME_SYNTAX_ERROR shall indicate that an unexpected framing sequence has been received
    - BAD_CTRL_SYMBOL_TYPE shall indicate if an unsupported Data or Control Symbol is received
    - PA_ERROR_IND_RECEIVED shall indicate that PA_ERROR.ind has been received
- Attempt or failure to (re-)initialize the PHY using PA_INIT.req
    - PA_INIT_ERROR shall indicate that the PA_INIT.req has failed
    - PA_INIT shall indicate that the DL Layer has issued a PA_INIT.req to the PA Layer, or that the DL Layer has received a PA_INIT.ind from the PA Layer.

Effect on Receipt

This indication may be used to take action for statistical procedures.

6.4.3.2 DL_LM_CTRL_NOFRAME.ind

DL_LM_CTRL_NOFRAME.ind indicates to the DME that the Data Link Layer transmitter does not have any Control Frames queued for transmission.

The semantics of this primitive are:

DL_LM_CTRL_NOFRAME.ind( )

When Generated

The Data Link Layer generates this primitive when there are no new Control Frames to transmit.

Effect on Receipt

This indication may be used to take action for power management procedures.

6.4.3.3 DL_LM_TC1_NOFRAME.ind

The Service Primitive indicates to the DME that there are no new or unacknowledged Data Frames in the Data Link Layer transmitter for Traffic Class 1.

The semantics of this primitive are:

DL_LM_TC1_NOFRAME.ind( )

When Generated

The Data Link Layer generates this primitive when there is no TC1 data to send and all transmitted TC1 Frames are acknowledged.

Effect on Receipt

This indication may be used to take action for power management procedures.

6.4.3.4 DL_LM_TC0_NOFRAME.ind

The Service Primitive indicates to the DME that there are no new or unacknowledged Data Frames in the Data Link Layer transmitter for Traffic Class 0.

The semantics of this primitive are:

DL_LM_TC0_NOFRAME.ind( )

When Generated

The Data Link Layer generates this primitive when there is no TC0 data to send and all transmitted TC0 Frames are acknowledged.

Effect on Receipt

This indication may be used to take action for power management procedures.

6.4.3.5 DL_LM_CTRL_FRAME.ind

DL_LM_CTRL_FRAME.ind indicates to the DME that the Data Link Layer has to transmit a Control Frame (AFC0, AFC1 or NAC Frame).

The semantics of this primitive are:

DL_LM_CTRL_FRAME.ind( )

When Generated

The Data Link Layer generates this primitive when there is a Control Frame to transmit.

Effect on Receipt

This indication may be used to take action for power management procedures.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

6.4.3.6 DL_LM_TC1_FRAME.ind

The Service Primitive indicates to the DME that there is at least one new or unacknowledged Data Frame in the Data Link Layer transmitter for Traffic Class 1.

The semantics of this primitive are:

DL_LM_TC1_FRAME.ind( )

When Generated

The Data Link Layer generates this primitive when there is at least one TC1 Data Frame to send or not all transmitted TC1 Frames are acknowledged.

Effect on Receipt

This indication may be used to take action for power management procedures.

6.4.3.7 DL_LM_TC0_FRAME.ind

The Service Primitive indicates to the DME that there is at least one new or unacknowledged Data Frame in the Data Link Layer transmitter for Traffic Class 0.

The semantics of this primitive are:

DL_LM_TC0_FRAME.ind( )

When Generated

The Data Link Layer generates this primitive when there is at least one TC0 Data Frame to send or not all transmitted TC0 Frames are acknowledged.

Effect on Receipt

This indication may be used to take action for power management procedures.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

6.5 Structure and Encoding of Protocol Data Units

DL Layer PDUs (DL_PDUs), or Frames, consist of a series of 17-bit symbols encoded as Data symbols or Control symbols as defined in ***Section 6.5.1*** and ***Section 6.5.2***, respectively.

The most significant bit of a symbol is used to distinguish Data symbols from Control symbols. These symbols use different PA Layer Service Primitives to cause the PA Layer to use the appropriate encoding scheme suited to the underlying PHY.

Note that this does not mean that these 17-bit symbols pass across the Link. The PHY Adapter Layer or PHY Layer can translate the symbols using a suitable encoding scheme.

The DL Layer supports two categories of Frames: Data Frames and Control Frames. Data Frames are used to transfer data between two DL Layers located on different UniPorts. Control Frames are used for flow control and reliability.

Figure 54 shows the supported Frames and also the derived Frames, where the highlighted boxes each represent a family of Frames.

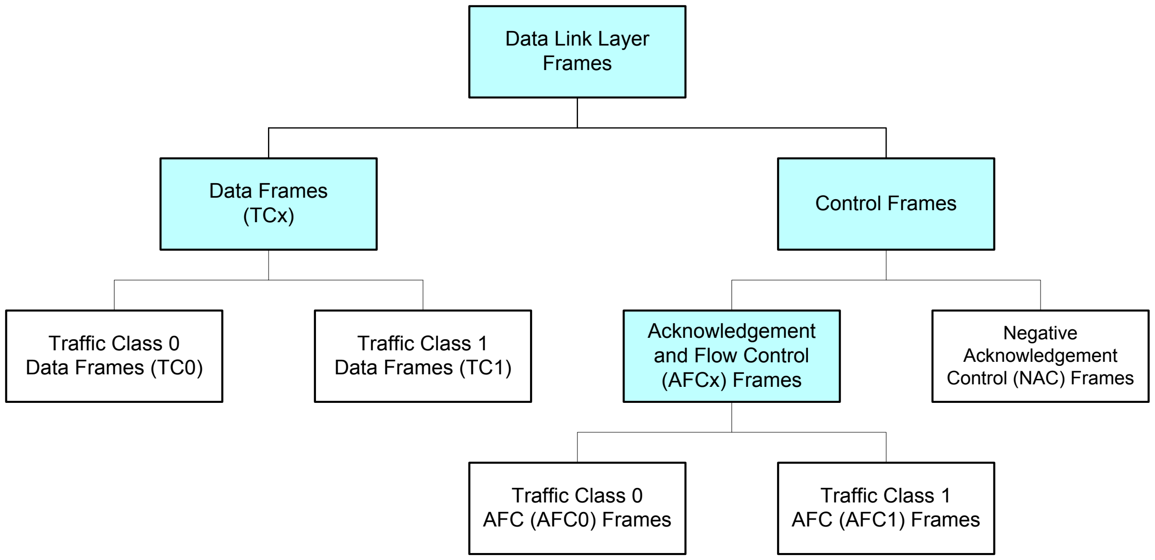

Data Link Layer Frames
Data Frames (TCx)
Control Frames
Traffic Class 0 Data Frames (TC0)
Traffic Class 1 Data Frames (TC1)
Acknowledgement and Flow Control (AFCx) Frames
Negative Acknowledgement Control (NAC) Frames
Traffic Class 0 AFC (AFC0) Frames
Traffic Class 1 AFC (AFC1) Frames

Figure 54 Control and Data Frames Taxonomy

The DL Layer shall support Data Frames of at least one Traffic Class. If both TCs are supported, the DL Layer shall support Data Frames of two Traffic Classes with different priorities. These Frames always have a Start of Frame (SOF) symbol, one or more data bytes, possibly a padding byte, an End of Frame (EOF_EVEN or EOF_ODD) and an error protection symbol. The Frame length is flexible for each Traffic Class (see ***Section 6.8***), but the maximum Frame length is limited to DL_SYMBOL_MTU symbols (excluding SOF symbol, EOF_EVEN or EOF_ODD symbol, COF and CRC symbols).

Additionally, each Traffic Class possesses a separate AFC Frame and, for all Traffic Classes, a common NAC Frame. These Control Frames are different from the Data Frames as they do not have SOF and EOF_EVEN or EOF_ODD symbols. ESC_DL together with Control Symbol Type acts as start of a Frame for respective Control Frame. Each Control Frame is of fixed length, which depends on its type. These Control Frames may be transmitted during Data Frames, i.e. they can preempt Data Frames, depending on the priority rule (see ***Section 6.6.4***).

Transmission gaps within Data Link Layer Frames resulting in the PHY inserting PHY Idle symbols while in HS mode should be avoided.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

6.5.1 Data Symbol

The DL Layer shall use data symbol to transmit or receive data information. ***Figure 55*** shows the DL Layer data symbol structure. Bit 16 shall be set to '0' for a data symbol in the DL Layer.

DL Layer shall use the PA_DATA.req Service Primitive, which is provided by the PHY Adapter Layer to send a data symbol and shall consider PA_SDU received from PA_DATA.ind Service Primitive of the PHY Adapter Layer as a data symbol. The identification bit (bit16) is not passed via the Service Primitive.

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Data															

Figure 55 Data Link Layer Data Symbol

6.5.2 Control Symbol

The control symbol shall be used by DL Layer to receive or transmit control information. Bits 15 to 8 form the MS byte, and bits 7 to 0 form the LS byte, of a control symbol. Bit 16 shall be set to '1' for a control symbol in the DL Layer.

The DL Layer shall use the PA_ESCDATA.req Service Primitive, which is provided by the PHY Adapter Layer to send a control symbol and shall consider PA_SDU received from PA_ESCDATA.ind Service Primitive that is provided by the PHY Adapter Layer as a control symbol. The identification bit (bit16) is not passed via the Service Primitive.

The ESC_DL character is used as a DL Layer control symbol identifier as shown in ***Figure 56***. The LS byte is partitioned into a Control Symbol Type field and a parameter field. The Control Symbol Type field refers to a control symbol identity and the parameter field specifies parameter values that have different meanings for different control symbols.

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	ESC_DL								Ctrl Symbol Type			Parameter				

Figure 56 Control Symbol Definition

Note that data and control symbols can be translated to an encoding scheme suitable for the PHY by the PA Layer. If the underlying PHY supports character coding the control symbol identifier (MS byte of a control symbol) can be translated to a special PHY character. The number of such special PHY characters is limited, and, therefore, Data Link Layer minimizes the number of control symbol identifiers.

Table 39 shows the currently defined MS byte of control symbol. The undefined values are reserved for future use. The DL Layer shall not use the reserved values. If received they shall be discarded.

Table 39 Control Symbol MS Byte Encoding

Control Symbol MS Byte	Tx_Data Bit[16]	Tx_Data, Bits[15:8]	Description
ESC_DL	1	00000001	DL Layer control symbol identifier

Table 40 shows all DL Layer control symbols with the corresponding Control Symbol Type and a short description.

Table 40 Control Symbol Type Field Definition

Control Symbol	Type	Description
SOF	0b000	Start Of Frame Parameter includes Traffic Class identifier
EOF_EVEN	0b001	End of Frame for even L2 payload Parameter indicates Frame Sequence Number
EOF_ODD	0b010	End Of Frame for odd L2 payload Parameter indicates Frame Sequence Number
COF	0b011	Continuation Of preempted Frame Parameter includes Traffic Class identifier
	0b100	Reserved
NAC	0b101	Start of a Negative Acknowledgment Control Frame Parameter includes a flag to request the peer end reinitialize its TX PHY
AFC	0b110	Start of an Acknowledgment and Flow Control Frame Parameter includes Traffic Class identifier and AFC type identification
	0b111	Reserved

The remaining values are reserved for future use. The transmitter shall not use reserved Control Symbol Type values. A control symbol received with a reserved Control Symbol Type shall be dropped.

Table 41 shows definition of the parameter fields of the DL Layer control symbols.

Table 41 Control Symbols Parameter Field Definition

Control Symbol	Parameter Field				
	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SOF	TC		Reserved		
EOF_EVEN	Frame Sequence Number				
EOF_ODD	Frame Sequence Number				
COF	TC		Reserved		
AFC	TC		CReq	Reserved	
NAC	Reserved				RReq

The DL Layer transmitter shall set the reserved bits to one. The DL Layer receiver shall ignore the reserved bits.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Bit 4 and bit 3 are used for identification of Traffic Class for SOF, COF and AFC control symbols. The usage is defined in ***Table 42***.

Table 42 Traffic Class Identification

TC	Traffic Class
11	Reserved
10	Reserved
01	TC1
00	TC0

The DL Layer transmitter shall not use reserved Traffic Class values. A SOF, COF or AFC control symbol received with a reserved TC value shall be dropped.

6.5.3 Data Frames

All Traffic Classes in DL Layer shall use the same format of user Data Frames, which shall encapsulate upper layer PDU. Each upper layer PDU shall be encapsulated in one Data Link Layer Frame, and one Data Frame shall only encapsulate one upper layer PDU. A Data Frame shall always start with a SOF symbol. After the SOF symbol, the Data Frame shall, as payload, include all the data symbols composing the Frame's SDU, i.e., the PDU provided by L3. In case the Frame payload contains an odd number of bytes, the last data symbol shall carry the last payload byte in the symbol's most significant byte, and the least significant byte shall contain 0x00. In case of preemption, COF symbols shall be included in between data symbols when Frame transmission is resumed. The Data Frame shall end with either an EOF_EVEN symbol or an EOF_ODD symbol when it encapsulates an even or an odd number of payload bytes, respectively. The EOF_EVEN or EOF_ODD symbol shall be followed by a data symbol carrying the Frame checksum using the CCITT CRC-16 ***[ITUT03]*** standard as described in ***Section 6.6.11***.

For all Data Link Layer Data Frames, the following rules shall apply:

- The Data Link Layer Data Frames are protected by a CCITT CRC-16 ***[ITUT03]*** checksum.
- The Data Link Layer Data Frame information shall be used only when the CRC checksum is correct.

Figure 57 shows a Data Link Layer Data Frame with an even number of DL_SDU bytes. ***Figure 58*** shows a Data Link Layer Data Frame with an odd number of DL_SDU bytes plus a padding byte. The maximum length of DL_SDU shall not exceed DL_SYMBOL_MTU symbols (including the padding byte, but excluding SOF symbol, EOF_EVEN or EOF_ODD symbol, COF and CRC symbols).

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	ESC_DL								SOF			TC		Reserved		
0	DL_SDU – Byte 0								DL_SDU – Byte 1							
0																
.								.								
.								.								
.								.								
0																
0	DL_SDU – Byte n-2 (n <= DL_MTU)								DL_SDU – Byte n-1 (n <= DL_MTU)							
1	ESC_DL								EOF_EVEN			Frame Seq. Number				
0	CCITT CRC-16															

Figure 57 Data Frame with Even Number of DL_SDU Bytes

The padding is always placed in the least significant byte of the last data symbol before the EOF_ODD symbol of a Data Frame. The transmitter shall set the padding byte to zero and at the receiver this padding byte shall be discarded after it has been fed through the CRC checker. The padding byte shall not be passed to the upper layer.

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	ESC_DL								SOF			TC		Reserved		
0	DL_SDU – Byte 0								DL_SDU – Byte 1							
0																
.								.								
.								.								
.								.								
0																
0	DL_SDU – Byte n-1 (n <= DL_MTU)								Padding Byte = 0x00							
1	ESC_DL								EOF_ODD			Frame Seq. Number				
0	CCITT CRC-16															

Figure 58 Data Frame with Odd Number of DL_SDU Bytes

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Figure 59 shows a Data Link Layer Data Frame with an even number of DL_SDU bytes that is preempted by a NAC Control Frame. The continuation of the preempted Frame is marked by the COF control symbol.

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	ESC_DL								SOF			TC		Reserved		
0	DL_SDU – Byte 0								DL_SDU – Byte 1							
0																
.								.								
.								.								
.								.								
0																
0	DL_SDU – Byte x-2								DL_SDU – Byte x-1							
1	ESC_DL								NAC			Reserved				RReq
0	CCITT CRC-16															
1	ESC_DL								COF			TC		Reserved		
0	DL_SDU – Byte (x)								DL_SDU – Byte (x+1)							
0																
.								.								
.								.								
.								.								
0																
0	DL_SDU – Byte n-2 (n <= DL_MTU)								DL_SDU – Byte n-1 (n <= DL_MTU)							
1	ESC_DL								EOF_EVEN			Frame Seq. Number				
0	CCITT CRC-16															

Figure 59 Data Frame with Preemption

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

6.5.4 Control Frames

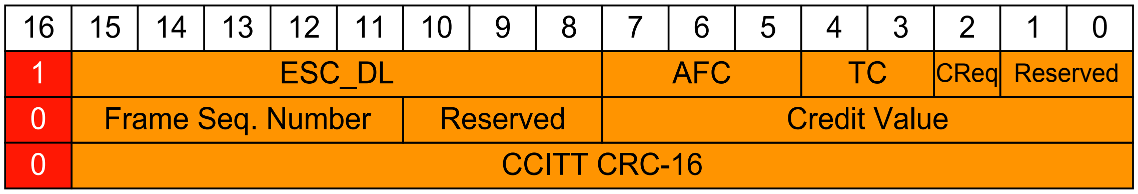
For all Data Link Layer Control Frames, the following rules shall apply:

- The Data Link Layer Control Frames are protected by a CCITT CRC-16 ***[ITUT03]*** checksum.
- The Data Link Layer Control Frame information shall be used only when the CRC checksum is correct.
- The Data Link Layer Control Frames shall not be preempted.

6.5.4.1 Data Link Layer Acknowledgment and Flow Control Frame

The Data Link Layer Acknowledgment and Flow Control (AFC) Frame is used for acknowledging correctly received Data Frames and for exchanging flow control information of the corresponding Traffic Class. The AFC Frame shall always start with an AFC control symbol, which is followed by two data symbols. The AFC Frame includes the Traffic Class identification, Credit Transmit Request (CReq) bit, Frame Sequence Number and the flow control credit value.

The acknowledgment (Frame Sequence Number) and flow control (Credit Value) information are assigned to the Traffic Class identified by the TC field. The AFC Frame fields shall be structured as shown in ***Figure 60***.



16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	ESC_DL								AFC			TC		CReq	Reserved	
0	Frame Seq. Number					Reserved			Credit Value							
0	CCITT CRC-16															

Figure 60 AFC Frame

The AFC Frame is used to exchange credit information and acknowledgments with the peer. The CReq bit in the AFCx symbol is used for requesting flow control information for the corresponding Traffic Class from the peer. The rules for requesting AFC Frame transmission are defined in ***Section 6.6.9***.

The Frame Sequence Number is defined with 5-bits that allows acknowledgments either individually or in a group (an acknowledgment to more than one Data Frame, but limited to sixteen, i.e. grouped acknowledgment) by the receiver per Traffic Class. See ***Section 6.6.8.1.***

The credit unit size in bytes is defined by DL_CreditUnitSize and the Credit Value field in AFC Frame is defined by DL_CreditValSizeBits. See ***Table 46***. Therefore, an AFC Frame can convey credit information for a receiver buffer size up to maximum of 4 KB due to the required roll over functionality of the credit accumulator. The flow control credit does represent the total number of credits available since boot time and does not represent a relative number. During the Data Link Layer initialization phase the free space in buffer is communicated via the AFC Frames. More details are listed in ***Section 6.7***.

The reserved bits shall be set to '1' when transmitting and shall be ignored at the receiver.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

6.5.4.2 Data Link Layer Negative Acknowledgment Control Frame

The Data Link Layer Negative Acknowledgment Control (NAC) Frame shall be sent when the receiver detects an error (see ***Section 6.6.10***) in any Frame or if a Data Frame is received with a wrong Frame Sequence Number or if the reverse Link needs to be reinitialized. The NAC Frame length shall be two symbols. The NAC Frame fields shall be structured as shown in ***Figure 61***. The NAC Frame contains a RReq bit to request the peer to reinitialize its TX PHY.

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	ESC_DL								NAC			Reserved				RReq
0	CCITT CRC-16															

Figure 61 NAC Frame

The reserved bits shall be set to '1' when transmitting and shall be ignored at the receiver.

6.6 Protocol Features

The following sections describe the control and data flow in the DL Layer for the transmitter and receiver.

6.6.1 PDU Composition Feature

The upper layer PDU is encapsulated always in one DL Layer Frame. Each upper layer PDU shall be encapsulated in a separate DL Layer Frame. The upper layer shall provide a PDU of size that fits within the maximum Frame length allowed for TX in the DL Layer.

The DL Layer sender adds a SOF control symbol as header for each upper layer PDU and an EOF_EVEN or EOF_ODD control symbol and the CRC symbol as trailer. The composition is shown in ***Figure 62*** for an even DL_SDU length.

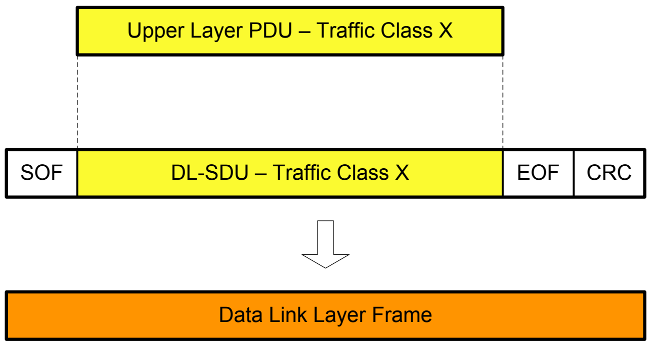


Figure 62 Composition of PDU with Even Length DL_SDU

6.6.1.1 Preemption

In order to reduce delays in the transmission of high priority Frames and to improve QoS in conjunction with upper layers, the DL Layer can insert high priority Frames within a low-priority Data Frame if the latter one is already in transmission. This functionality is known as preemption of a Frame. Support of preemption functionality is optional for the transmitter, whereas the receiver shall always be able to receive preempted and non-preempted Frames.

DL_TxPreemptionCap defines if the transmit side of the Data Link Layer has preemption capability. If TRUE, the transmit side shall support preemption. If FALSE, the transmit side shall not support any case of preemption: Data Frames shall not be preempted by AFC Frames or NAC Frames; Data Frames of lower priorities shall not be preempted by Data Frames of higher priorities.

The transmission of a preempted Frame shall be continued with a COF symbol, followed by a data symbol or an EOF_EVEN or EOF_ODD symbol for that Traffic Class.

In addition, another pre-emption can occur back-to-back to the previous preempting Frame.

Examples are shown as follows:

… [TC0 SOF+DATA][AFC DATA+CRC][TC0 COF][TC1 SOF+DATA+EOF+CRC]

[TC0 COF+EOF+CRC] …

… [TC0 SOF+DATA][AFC DATA+CRC][TC1 SOF+DATA+EOF+CRC][TC0 COF+EOF+CRC] …

… [TC0 SOF][TC1 SOF+DATA+EOF+CRC][TC1 SOF+DATA+EOF+CRC]

[TC0 COF+DATA+EOF+CRC] …

Figure 63 shows the case, where preemption is carried out in the middle of the low priority Data Frame for the quick transmission by the high-priority Frame. Both Frames have even DL_SDU lengths.

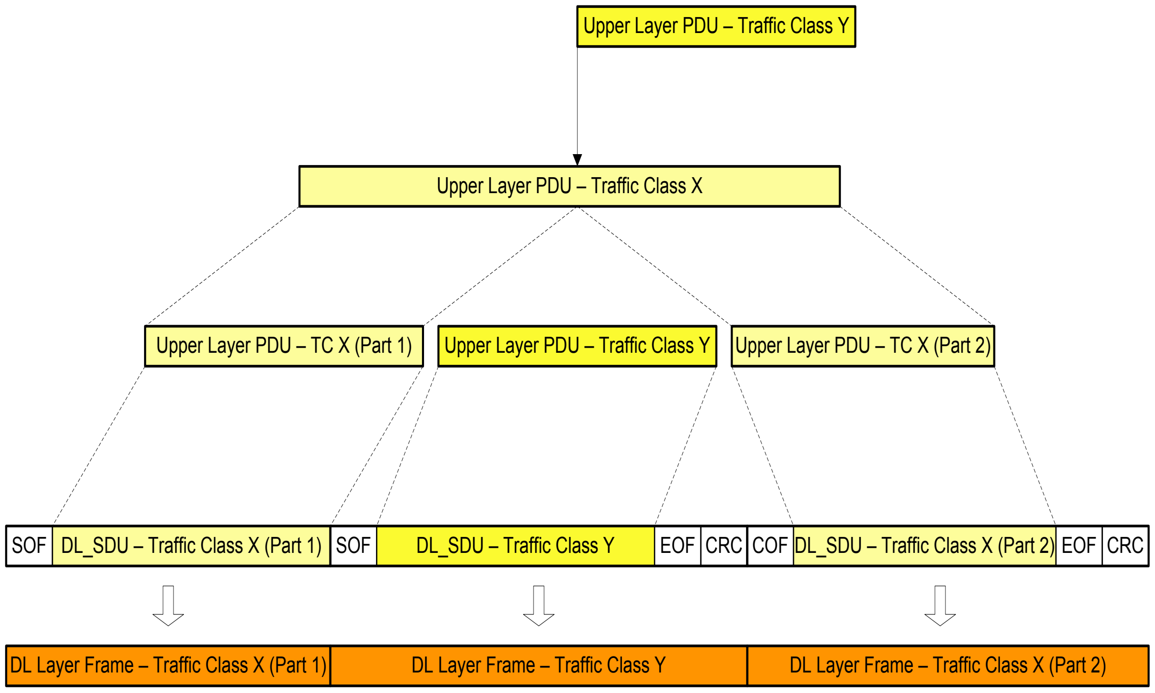

Upper Layer PDU – Traffic Class Y
Upper Layer PDU – Traffic Class X
Upper Layer PDU – TC X (Part 1)
Upper Layer PDU – Traffic Class Y
Upper Layer PDU – TC X (Part 2)
SOF
DL_SDU – Traffic Class X (Part 1)
SOF
DL_SDU – Traffic Class Y
EOF
CRC
COF
DL_SDU – Traffic Class X (Part 2)
EOF
CRC
DL Layer Frame – Traffic Class X (Part 1)
DL Layer Frame – Traffic Class Y
DL Layer Frame – Traffic Class X (Part 2)

Figure 63 Composition with Preemption (Traffic Class Y > X)

When implemented, the preemption allows a faster delivery of the high-priority Frames and prevents the Frames from being blocked by a low priority Frame. In contrast to that a system without preemption causes a higher latency to the high-priority Frame. ***Section B.2*** highlights the effects of preemption in more detail.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

The priority list for preemption is given in ***Section 6.6.4***.

The following rules shall be considered during preemption:

- The preemption shall not occur:
    - between EOF_EVEN or EOF_ODD and CRC symbols
    - within AFC and NAC Frames (including CRC symbols)
- The preemption may happen at any symbol boundary except where the above rule is valid

6.6.2 PDU Decomposition Feature

The DL receiver removes the header (SOF symbol) and the trailer (EOF_EVEN or EOF_ODD symbol and CRC symbol) that have been added by the peer DL Layer from the incoming Frame and passes the DL_SDU to the upper layer if the CRC check is passed for the received Frame, the Frame Sequence Number is in the expected order and the received Frame is not a Frame that was already previously passed to the upper layer (retransmitted Frame). The decomposition without preemption is shown in ***Figure 64***.

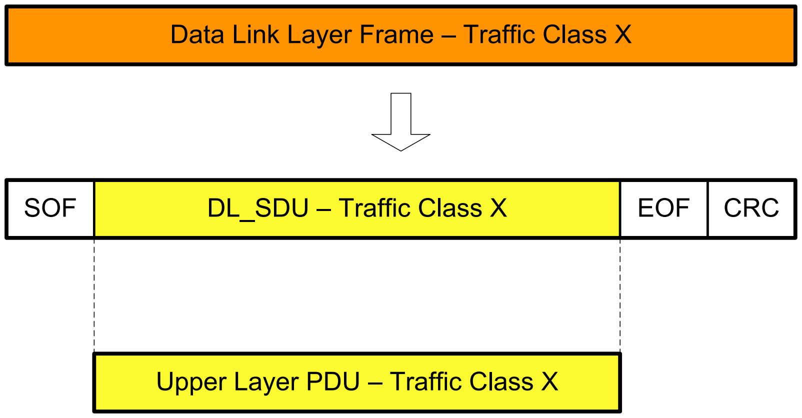

Data Link Layer Frame – Traffic Class X
SOF
DL_SDU – Traffic Class X
EOF
CRC
Upper Layer PDU – Traffic Class X

Figure 64 Decomposition with Even Length DL_SDU

In case of preemption also COF control symbols are removed before passing the Frame to the Network Layer. The correct and complete payload (without COF symbol) of a Frame is passed to the upper layer if the above mentioned rules for correctness are fulfilled. ***Figure 65*** shows the decomposition with preemption.

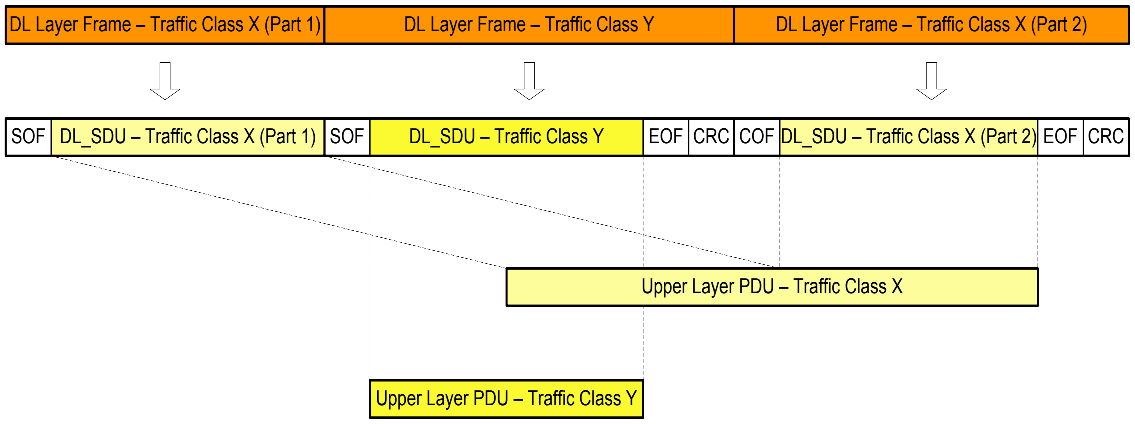

DL Layer Frame – Traffic Class X (Part 1)
DL Layer Frame – Traffic Class Y
DL Layer Frame – Traffic Class X (Part 2)
SOF
DL_SDU – Traffic Class X (Part 1)
SOF
DL_SDU – Traffic Class Y
EOF
CRC
COF
DL_SDU – Traffic Class X (Part 2)
EOF
CRC
Upper Layer PDU – Traffic Class X
Upper Layer PDU – Traffic Class Y

Figure 65 Decomposition with Preemption

6.6.3 Buffering Mechanism

A retransmission capability shall be provided in the transmitter per Traffic Class to retain Frames after transmission until they are acknowledged for that Traffic Class. This retention allows the Frames to be retransmitted if not successfully transmitted earlier.

The following rules apply for all TC buffers:

- The upper layer PDU shall be available to DL Layer completely prior to Frame start.
- The DL Layer shall accept upper layer PDU only if it is capable of retransmitting it.

For any Traffic Class, the DL Layer implementation shall be able to retransmit any Frame it sent until the Frame is acknowledged. In addition, the DL Layer implementation shall be able to receive a Frame of the maximum payload size (DL_MTU) for any Traffic Class and shall not deliver to the upper layers data from a Frame with any of the errors defined in ***Section 6.4.3.1***.

6.6.4 Arbitration Scheme

Table 43 lists the DL Layer priority in descending order of priority for all Frame types.

Table 43 Supported Priorities

Priority	DL Layer Frame Name	Frame Type	Description
Highest Priority	NAC	Control	Negative Acknowledgment Control (NAC) Frame
–	AFC1 (promoted)	Control	Acknowledgment and Flow Control (AFC) Frame for Traffic Class 1 triggered by expiry of AFC1_REQUEST_TIMER, or by expiry of FC1_PROTECTION_TIMER, or due to reception of an AFC1 with CReq bit set to ‘1’, or when AFC1_REQUEST_TIMER is running, and a PA_DL_PAUSE.ind is received
–	AFC0 (promoted)	Control	Acknowledgment and Flow Control (AFC) Frame for Traffic Class 0 triggered by expiry of AFC0_REQUEST_TIMER, or by expiry of FC0_PROTECTION_TIMER, or due to reception of an AFC0 with CReq bit set to ‘1’, or when AFC0_REQUEST_TIMER is running, and a PA_DL_PAUSE.ind is received
–	AFC1	Control	Acknowledgment and Flow Control (AFC) Frame for Traffic Class 1 triggered by conditions other than the conditions that trigger AFC1 (promoted)
–	TC1	Data	Traffic Class 1 Data Frames
–	AFC0	Control	Acknowledgment and Flow Control (AFC) Frame for Traffic Class 0 triggered by conditions other than the conditions that trigger AFC0 (promoted)
Lowest Priority	TC0	Data	Traffic Class 0 Data Frames that can be used for traffic for which the latency is less critical

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

6.6.5 Change of Certain Link Properties

When there is a change of certain properties of the Link, i.e. power mode, number of active Lanes, etc. coordination between the PHY Adapter Layer (L1.5) and the Data Link Layer (L2) is necessary to ensure proper operation before, during and after the property is changed. Such coordination is orchestrated by the PHY Adapter Layer, when the need occurs, through the PA_DL_PAUSE.ind primitive.

When PA_DL_PAUSE.ind is received, all Data Link Layer timers shall be stopped. The Data Link Layer may complete transmission of the current Frame. The Data Link Layer shall transmit pending, promoted AFC or NAC Frames before issuing PA_DL_PAUSE.rsp_L. The Frozen state shall be reached no later than after the transmission of any ongoing Frames. When the Frozen state is reached, the PA_DL_PAUSE.rsp_L shall be generated. After generating the PA_DL_PAUSE.rsp_L, the Data Link Layer shall not send any new Symbols to the PHY Adapter Layer. If Frames are not being transmitted, and there are no pending, promoted AFC or NAC Frames, the Data Link Layer shall immediately generate a PA_DL_PAUSE.rsp_L.

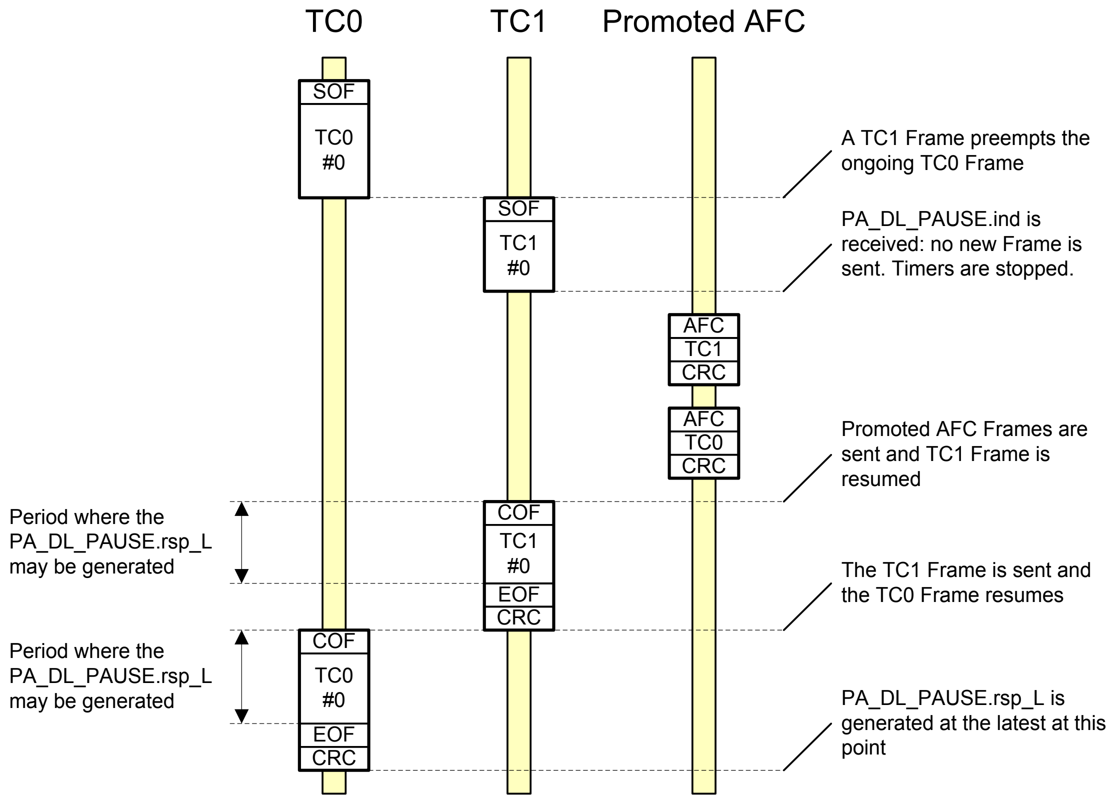

TC0
TC1
Promoted AFC
SOF
TC0
#0
SOF
TC1
#0
AFC
TC1
CRC
AFC
TC0
CRC
COF
TC1
#0
EOF
CRC
COF
TC0
#0
EOF
CRC
A TC1 Frame preempts the ongoing TC0 Frame
PA_DL_PAUSE.ind is received: no new Frame is sent. Timers are stopped.
Promoted AFC Frames are sent and TC1 Frame is resumed
The TC1 Frame is sent and the TC0 Frame resumes
PA_DL_PAUSE.rsp_L is generated at the latest at this point
Period where the PA_DL_PAUSE.rsp_L may be generated
Period where the PA_DL_PAUSE.rsp_L may be generated

Figure 66 Generation of PA_DL_PAUSE.rsp_L Example

Figure 66 exemplifies the behavior previously described in the case where the PA_DL_PAUSE.ind is received when a TC1 Frame preempting a TC0 Frame is transmitted, and promoted AFCs are pending transmission. New TCx Frames are not transmitted, and the TC1 Frame is preempted by two promoted AFC Frames. Once the promoted AFC Frames have been transmitted, the TC1 Frame followed by the TC0 Frame are transmitted to completion. The completion times of TC0 and TC1 are shown in the figure. PA_DL_PAUSE.rsp_L can be transmitted any time between the start of the PA_DL_PAUSE.ind primitive and reception of the AFC Frame with CReq bit set to ‘1’, or after the transmission of the promoted AFC Frame, but not between the EOF and CRC of a Frame. Depending on implementation, the actual transmission of the promoted AFC Frames might be delayed after the reception of an AFC Frame with the CReq bit set to ‘1’.

The transmission of the currently preempted TC0 Frame is completed and finally this marks the latest point in time where the PA_DL_PAUSE.rsp_L primitive is generated, if it was not already generated.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

If the Frozen state is entered during the transmission of a Frame, when exiting the Frozen state and resuming normal operation that Frame shall be continued from the point of interruption.

The PHY Adapter Layer indicates the operation has completed with a PA_DL_RESUME.ind primitive sent to the Data Link Layer. Upon reception of the primitive PA_DL_RESUME.ind, the Data Link Layer shall leave the Frozen state, shall resume normal operation and shall send any pending Frames according to the arbitration scheme. The DL Layer shall re-enable Frame transmission, reset and restart all timers. The DL Layer shall not generate a COF symbol, when resuming operation upon reception of the primitive PA_DL_RESUME.ind.

6.6.6 TCx Data Frame Transmission

The following rules shall be applied for the transmission of a Data Frame for any Traffic Class:

- Frames shall be transmitted according to the arbitration scheme (see ***Section 6.6.4***). Frames of higher priority may preempt (see ***Section 6.6.1.1***) a Data Frame of lower priority.
- A Data Frame shall only be started if the following conditions are satisfied:
    - Enough credits are available to send the complete DL_SDU to the peer
    - Number of unacknowledged Frames is less than sixteen
- The transmission of a preempted (see ***Section 6.6.1.1***) Data Frame shall be continued with a COF symbol.

6.6.7 Flow Control

The Data Link Layer utilizes a dedicated credit flow control scheme per Traffic Class. The transmitter shall keep track of the amount of free logical buffer space in the receiver. This allows the transmitter to start a Data Frame of a Traffic Class if the peer receiver has free logical buffer space for this TC for the complete Frame. The credits are used only for Data Frame payload and padding byte and shall not be used for transmitting SOF, EOF_EVEN, EOF_ODD, COF and CRC symbols, and Control Frames.

The credit flow control scheme used in DL Layer requires that the DL transmitter shall keep track of the space available in the peer RX logical buffer per Traffic Class (logical buffer to store Frames). The system is based on registers that represent the total number of credits that have been available since boot up/reset. This amount of credits is then compared to the total amount of credit used since power-up or reset to determine if a Frame shall or shall not be transmitted. Note that the number of credits received is often pessimistic. It is delayed in time and often does not reflect the actual amount of credit available at the receiver, but a slightly older version of this number. This creates a conservative view of the space available. This 'error' may actually impact system performance, as data may be held in the transmitter when space exists in the receiver. However, it will always be robust, as this error only understates the space available and never overstates it. Credits are self-healing. That is, if a credit Frame is dropped due to either CRC checksum failure of AFC Frame or AFC Frame is not detected, it will be corrected by the transmission of a next, or new, more up-to-date AFC Frame. This works because the credits represent the total number of credits since boot time.

During the boot procedure, each side exchanges, for each Traffic Class, AFC Frames with CReq set to '1' and a Credit Value field equal to the size of their receiving logical buffer in credit units (see ***Section 6.7***).

Each node shall maintain, for each TC, the following register values:

- R; Stores the number of credits received
- U; Stores the total number of credits used
- S; Stores the total number of credits sent
- A; Stores the total number of credits available
- Part_U; Stores the number of partial credits of U
- Part_A; Stores the number of partial credits of A

The transmitter part of a node handles the registers R and U, whereas the receiver block of a node handles the registers S and A. The initial values of R, U, and S are 0. A is initialized to the size of the receiving logical

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

buffer in credit units (DL_CreditUnitSize). If the logical buffer size is not an integer multiple of the credit unit size then the available logical buffer space to be specified shall be rounded down to the nearest integer multiple of credit unit size, e.g., if the logical buffer size is 1000 bytes, then A shall be set to 31. The size of credit registers R, S, U and A shall be the same as the credit value size of an AFC Frame (see ***Figure 60***), which is 8-bits, whereas the granularity of their representation is one credit unit size. Note that the 8-bit R, U, S, A registers actually contain the number of total credits received, used, sent or available, respectively, modulo 256. Overflow of the 8-bit registers shall be ignored.

When a node receives an error-free AFC Frame for a Traffic Class, the value received in credit value field of the AFC Frame shall replace the value of R.

When an L2 transmitter sends payload or padding bytes to the L1.5 layer, the node shall increment U by the number of whole credit units sent, i.e. the integer quotient of the number of bytes sent divided by DL_CreditUnitSize. The node shall internally track the number of partial credits sent, i.e. data sent to the other end which is less than DL_CreditUnitSize bytes, in Part_U. One partial credit represents one byte. Whenever the sum of partial credits sent in Part_U exceeds DL_CreditUnitSize bytes, U shall be incremented by one and Part_U shall be decremented by DL_CreditUnitSize, i.e. the Part_U value is replaced by the number of partial credits sent modulo DL_CreditUnitSize. The partial credit sum shall be less than one credit unit size at any time.

The payload bytes fetched by the upper layer from the DL Layer and the padding bytes discarded by the DL Layer represent the logical buffer empty space created in the DL Layer receiver's logical buffer. The DL Layer shall increment A by the integer quotient of the newly created logical buffer empty space in bytes divided by DL_CreditUnitSize. The remainder of the division shall be accumulated in bytes as partial credits in Part_A. Whenever the sum of partial credits in Part_A exceeds DL_CreditUnitSize bytes, A shall be incremented by one and Part_A shall be decremented by DL_CreditUnitSize.

The AFC sending node shall set the AFC credit value field with the contents of A when an AFC Frame is sent. At the same time, the value of S shall be replaced by the value of A.

For the original transmission of a Data Frame, a node is allowed to send at most the number of bytes given by ***Equation 1.***

Equation 1:

$$[((R + 2^{DL\_CreditValSizeBits} - U) \bmod 2^{DL\_CreditValSizeBits}) \times DL\_CreditUnitSize - Part\_U] \text{ bytes}$$

where,

DL_CreditValSizeBits is the number of bits used to represent Credit Value field in AFC Frame and
DL_CreditUnitSize is the size of one credit unit in bytes.

Note, the term '$(R + 2^{DL\_CreditValSizeBits} - U) \bmod 2^{DL\_CreditValSizeBits}$' addresses wrap-around issues, and avoids negative numbers in the result.

In the case of the original transmission of a Data Frame that belongs to a given TC, the DL Layer shall not start transmission if the size of the Frame payload, including the padding byte, is larger than the value given by ***Equation 1***.

For each transmitted Frame, its associated credit value shall be counted only once in the U and Part_U credit registers, independent of the number of times the Frame is retransmitted.

Since credits were consumed during the original transmission of a Data Frame that belongs to a given TC, the DL Layer shall not check for credits in case of a retransmission of the Data Frame. That is, the DL Layer transmits a Data Frame that is part of a retransmission, even if the Frame payload including the padding byte is larger than the value given by ***Equation 1***.

A node shall transmit an AFC Frame due to flow control when the following condition occurs:

$$[(A + 2^{DL\_CreditValSizeBits} - S) \bmod 2^{DL\_CreditValSizeBits}] > DL\_AFCxCreditThreshold$$

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

where,

x represents the Traffic Class number, 0 or 1,

A and S belong to the same Traffic Class as DL_AFCxCreditThreshold.

If DL transmitter has a Data Frame to send and fewer credits than DL_TCxTXFCThreshold, it shall send an AFCx Frame with CReq bit set to '1' after expiration of FCx_PROTECTION_TIMER to request the peer DL Layer to send flow control information. The following formula is used to compare the number of credits available to DL_TCxTXFCThreshold:

$$[((R + 2^{DL\_CreditValSizeBits} – U) \bmod 2^{DL\_CreditValSizeBits}) \times DL\_CreditUnitSize – Part\_U] < DL\_TCxTXFCThreshold \times DL\_CreditUnitSize$$

The FCx_PROTECTION_TIMER is used for protecting against loss of credits due to the loss of the AFCx Frame. The timeout value of FCx_PROTECTION_TIMER is denoted as DL_FCxProtectionTimeOutVal.

FCx_PROTECTION_TIMER shall be reset when at least one of the following conditions is valid:

- With the expiration of FCx_PROTECTION_TIMER
- With the reception of an error-free AFCx Frame

FCx_PROTECTION_TIMER shall run when at least one of the following conditions is valid:

- The TCx of the peer node is present (see ***Section 6.7***), TX of a TCx has less credits than DL_TCxTXFCThreshold and data to send
- The Data Link Layer is in the initialization phase, and no AFCx Frame was received for TCx

The FCx_PROTECTION_TIMER shall be stopped when all of the above run conditions are not satisfied or if the DL Layer is waiting for PA_INIT.cnf_L as a result of a request for PHY (re-)initialization (PA_INIT.req).

After sending an AFCx Frame with CReq bit set to '1', if the FCx_PROTECTION_TIMER expires without receiving the AFCx Frame, then PA_INIT.req shall be triggered and NAC transmission shall be enabled.

After receiving PA_INIT.cnf_L with PAReverseLinkInitialized set to FALSE, a NAC Frame with the RReq bit set to '1' shall be transmitted before the transmission of an AFCx Frame with CReq bit set to '1'.

After receiving PA_INIT.cnf_L with PAReverseLinkInitialized set to TRUE, a NAC Frame with the RReq bit set to '0' shall be transmitted before the transmission of an AFCx Frame with CReq bit set to '1'.

Note:

*The priority of the AFCx Frame is raised when responding to an AFCx Frame with CReq bit set to '1'. See **Section 6.6.4.***

More details are given in ***Section 6.6.8.2***. All AFC Frame transmission rules are explained in ***Section 6.6.9***.

Table 44 provides an example of the changes in the various credit-tracking registers after boot up. In the example, the receiver's initial credit value, e.g. for TC0, is given by DL_TC0RxInitCreditVal which is assumed to be sixteen, and the DL_AFCxCreditThreshold is set to fourteen. The local node transmits AFC0 Frames to the peer node, and the peer node transmits Data Frames of TC0 to the local node. ***Figure 67*** illustrates the same example as ***Table 44*** using a Message Sequence Chart. ***Figure 68*** depicts the difference between R and U credits at peer node TX. The graph shows the variation of (R – U) credits during data transmission and AFC Frame reception.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

4185 **Table 44 Example of Flow Control Register Changes after Boot Up**

Time	Comments / Event	Local Node						Peer Node					
		RX			TX			RX			TX		
		Avail (A)	Partial Credits of A	Sent (S)	Rcvd (R)	Used (U)	Partial Credits of U	Avail (A)	Partial Credits of A	Sent (S)	Rcvd (R)	Used (U)	Partial Credits of U
T0	Init	16	0	0	0	0	0	16	0	0	0	0	0
T1	Local node sends an AFC Frame with a value of 16	16	0	16	0	0	0	16	0	0	0	0	0
T2	Peer node receives the AFC Frame and updates its value	16	0	16	0	0	0	16	0	0	16	0	0
T3	Peer node sends 256 bytes	16	0	16	0	0	0	16	0	0	16	8	0
T4	Local node receives 256 bytes	16	0	16	0	0	0	16	0	0	16	8	0
T5	Local node L3 reads 256 bytes Local node could send an AFC Frame but in this example, it is considered that it must wait for 15 new available credits, i.e. it must read 480 bytes	24	0	16	0	0	0	16	0	0	16	8	0
T6	Peer node sends 218 bytes	24	0	16	0	0	0	16	0	0	16	14	26
T7	Local node receives 218 bytes	24	0	16	0	0	0	16	0	0	16	14	26
T8	Local node reads 212 bytes	30	20	16	0	0	0	16	0	0	16	14	26
T9	Peer node sends 32 bytes	30	20	16	0	0	0	16	0	0	16	15	26
T10	Local node receives 32 bytes	30	20	16	0	0	0	16	0	0	16	15	26
T11	Local node reads 38 bytes. Local node can send an AFC Frame now	31	26	16	0	0	0	16	0	0	16	15	26
T12	Local node sends an AFC Frame with a value of 31 (adding 15 credits = 480 bytes)	31	26	31	0	0	0	16	0	0	16	15	26

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Time	Comments / Event	Local Node						Peer Node					
		RX			TX			RX			TX		
		Avail (A)	Partial Credits of A	Sent (S)	Rcvd (R)	Used (U)	Partial Credits of U	Avail (A)	Partial Credits of A	Sent (S)	Rcvd (R)	Used (U)	Partial Credits of U
T13	Peer node receives the AFC Frame	31	26	31	0	0	0	16	0	0	31	15	26
T14	Peer node sends 6 bytes	31	26	31	0	0	0	16	0	0	31	16	0
T15	Local node receives 6 bytes	31	26	31	0	0	0	16	0	0	31	16	0
T16	Local node reads 6 bytes	32	0	31	0	0	0	16	0	0	31	16	0
…	…	…	…	…	…	…	…	…	…	…	…	…	…

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

4186

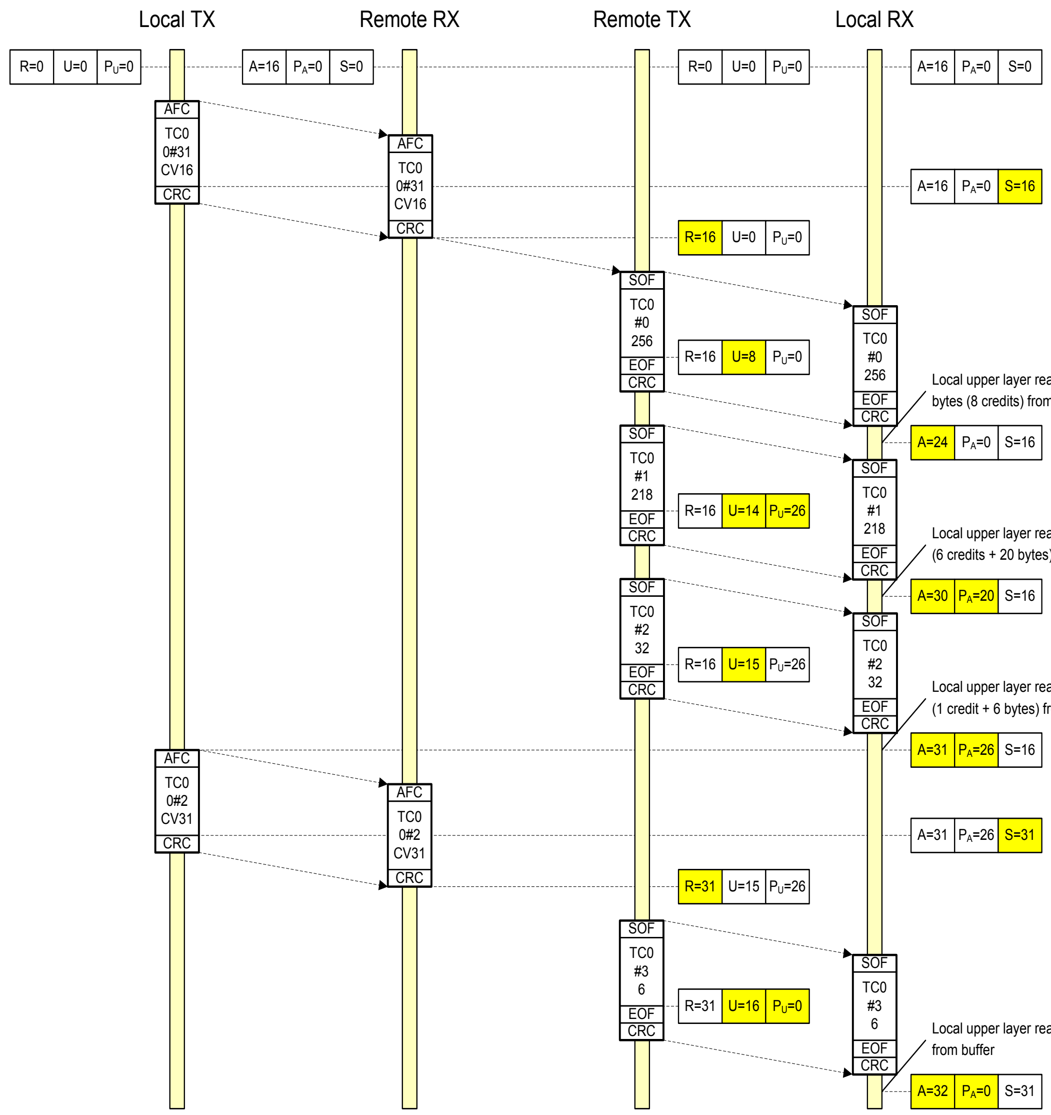


Figure 67 Message Sequence Chart for Flow Control Register Changes After Boot Up Example

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

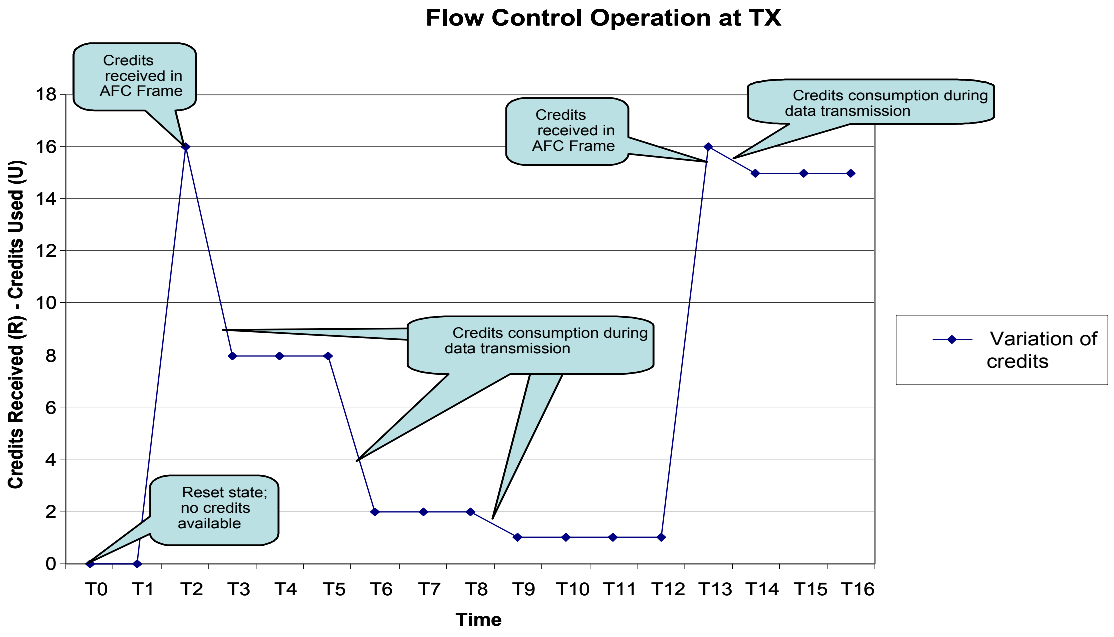


Figure 68 Variation of Credits During Data Transmission Example

The padding byte, if it is used in a Data Frame at transmitter side, shall consume a partial credit (partial credits of U).

At the receiver the padding byte shall be discarded and available partial credits (partial credits of A) incremented by one.

An alternative way to handle credits during retransmission is known as the roll-back credit method. Using the roll-back credit method, in the case of retransmission, the number of credits of all Frames to be replayed is evaluated, and the total amount is decremented from the U and Part_U credit registers. In this case, the DL Layer shall not transmit a Data Frame that belongs to a given TC if the size of the Frame payload, including the padding byte, is larger than the value given by ***Equation 1.*** The U and Part_U credit registers shall be updated according to the number of credits consumed during retransmission.

6.6.8 Acknowledgment Operation

Each DL Layer Data Frame sent by the local transmitter shall be acknowledged either individually or in a group (an acknowledgment to more than one Frame, i.e., grouped acknowledgment) by the peer receiver. The peer receiver shall acknowledge Frames only when it receives Frames in good condition (see ***Section 6.6.10***). The DL Layer shall drop a Frame in its entirety and not process the Frame in any way if the CRC checksum is incorrect or has an unexpected Frame Sequence Number and shall transmit a Negative Acknowledgment Control (NAC) Frame. If a Data Frame is received error-free (correct CRC symbol and with expected Frame Sequence Number), the receiver shall schedule an acknowledgment with the Frame Sequence Number of this Frame. As the DL Layer communicates flow control and acknowledgment in the same Frame (AFC Frame, see ***Section 6.5.4***) the generation of AFC Frame for the pending acknowledgments may not be immediate and follow the AFC Frame generation rules (see ***Section 6.6.9***). The AFC Frame contains current credit information of the receiver logical buffer and an acknowledgment with Frame Sequence Number of the last correctly received Frame for that Traffic Class. When a Frame Sequence Number is received in an AFC Frame, all Frames sent up to and including the Frame referenced by the Frame Sequence Number shall be considered acknowledged. During retransmission all flow control information shall be accepted (see ***Section 6.6.8.2***). If a Frame is corrupted, i.e. CRC error, it is unreliable to depend on it to make any decision (to identify if it is a TC0 or TC1, Data Frame or a Control Frame). There is a dedicated AFC Frame for each Traffic Class (AFC0 for TC0 and AFC1 for TC1).

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

The following sections provide information on Frame Sequence Number and acknowledgment timeout operation.

6.6.8.1 Frame Sequence Number Identification

A 5-bit Frame Sequence Number shall be used with each Data Frame. The Frame Sequence Number is a part of EOF_EVEN/EOF_ODD symbol. Each Traffic Class (TC) shall use its own Frame Sequence Numbers independently of other TCs, i.e., the Frame Sequence Numbers shall not be shared among TCs. The range of Frame Sequence Numbers is from 0 to 31 for each TC. The Frame Sequence Number shall be transmitted within an AFC Frame (see ***Section 6.5.4.1***) of the corresponding TC to acknowledge correctly received Frames. A wrap around mechanism shall be used so that once the Frame Sequence Number of a Data Frame that belongs to a given TC reaches thirty-one it shall roll over and repeat the Frame Sequence Number from 0 for the next Data Frame for that TC. Frame Sequence Numbers shall have the following characteristics:

- A separate Frame Sequence Number is available for each Traffic Class.
- The valid range is from 0 to 31.
- The Frame Sequence Number is incremented with transmission of a Frame of that Traffic Class. The Frame Sequence Number wrap around mechanism shall be ensured.
- When reset, the first transmitted Data Frame in every traffic class shall start with Frame Sequence Number 0.
- When reset, the last good Frame Sequence Number received shall be set to 31. The next expected receive Frame Sequence Number shall be 0.
- When a Data Frame of a Traffic Class with an expected Frame Sequence Number is correctly received, the Frame Sequence Number shall be incremented. The Frame Sequence Number wrap around mechanism shall be ensured.

The incrementing Frame Sequence Number with its wrapping mechanism supports a maximum of 16 outstanding Frames for grouped acknowledgment at any time for each Traffic Class. The limit of sixteen outstanding Frames is due to the fact that the DL Layer shall guarantee in order delivery and shall detect replayed Frames.

6.6.8.2 Retransmission and Frame Acknowledgment Timeout Mechanism

A dedicated replay timer (TCx_REPLAY_TIMER) shall be provided for each Traffic Class (TC0 and TC1) to identify unacknowledged transmitted Frames. This replay timer shall not be reset if AFCx Frames are received for previously acknowledged Frames and there is at least one unacknowledged Frame in the logical buffer. The timer guards against the possibility of an AFC or a NAC Frame encountered errors and were discarded.

During Data Frame retransmission, the payload and Frame Sequence Numbers shall be identical to the original transmission per Traffic Class. The retransmission shall be processed according to the arbitration scheme. When Frames are being retransmitted, they shall be all transmitted sequentially, even when an acknowledgment for a Frame under a retransmission or still to be retransmitted is received.

If an AFCx (where x is 0 for TC0, 1 for TC1) Frame is received, then the Frame Sequence Number in the AFCx Frame is compared to the last sent Data Frames or up to the previously retransmitted Frame (considering wrap around mechanism and allowed maximum number of outstanding Frames) in order to accept it. If the Frame Sequence Number in the AFCx Frame is either not in the outstanding Frames, or greater than the last (re)transmitted Data Frame of the corresponding TC, then flow control information shall be accepted but the acknowledgment information in the AFCx Frame shall be considered as outdated and discarded. The following example illustrates this case for TC0:

- Local RX receives AFC0#3 Frame
- Local TX sends Frames TC0Frame#4, TC0Frame#5, TC0Frame#6 and TC0Frame#7
- Local TX triggers retransmission due to timeout (last acknowledged Frame was TC0Frame#3)
- Local TX triggers PA_INIT.req

- After confirmation by PA_INIT.cnf_L Service Primitive, the local TX starts transmission of a NAC Frame with the RReq bit set according to parameter PAReverseLinkInitialized, and then sends an AFC Frame (current information on flow control and acknowledgment status) for TC1.
- As there are no outstanding TC1 Data Frames, the local TX do not send any TC1 Data Frames
- Local TX sends AFC (current information on flow control and acknowledgment status) for TC0
- Local TX starts replaying Frame from TC0Frame#4 to TC0Frame#7
- During TC0Frame #4 replay, local RX receives AFC0#7 Frame
- The local RX accepts flow control information but discards acknowledgment information from AFC0#7 Frame and continue with the replay (TC0Frame#4, TC0Frame#5, TC0Frame#6, and TC0Frame#7)

If the TCx_REPLAY_TIMER expires for a given TCx Data Frame without receiving an acknowledgment, i.e., the Frame is assumed not successfully transmitted, then the DL Layer shall trigger PA_INIT.req Service Primitive, and enable NAC transmission. After receiving PA_INIT.cnf_L with the parameter PAReverseLinkInitialized set to FALSE, the DL Layer shall transmit a NAC Frame with the RReq bit set to ‘1’. After receiving PA_INIT.cnf_L with the parameter PAReverseLinkInitialized set to TRUE, the DL Layer shall transmit a NAC Frame with the RReq bit cleared to ‘0’, then an AFC Frame (current information on flow control and acknowledgment status) for TC1, then all unacknowledged Data Frames, if any, of TC1, then an AFC Frame (with current flow control and acknowledgment status) for TC0, and then all unacknowledged Data Frames, if any, of TC0.

If a NAC Frame is detected during the transmission of any Data Frame then the transmitter shall stop current Data Frame transmission if the EOF_EVEN or EOF_ODD symbol was not sent yet. If the EOF_EVEN or EOF_ODD symbol is sent this Frame shall not be stopped. If a NAC Frame is detected during the transmission of any Control Frame then the transmitter shall not stop the Control Frame under transmission.

The receiver of a NAC Frame shall perform the following sequence: The receiver shall trigger PA_INIT.req of the PHY Adapter Layer if the RReq bit in the NAC Frame is set to ‘1’. If PA_INIT.req is triggered, then the TX shall wait for PA_INIT.cnf_L before continuing with the sequence. The DL Layer shall trigger PA_LANE_ALIGN.req Service Primitive of the PHY Adapter Layer if the RReq bit in the NAC Frame is set to ‘0’. Only after the reception of the primitive PA_LANE_ALIGN.cnf_L will the sequence continue. After PA_INIT or PA_LANE_ALIGN, AFC Frames and all unacknowledged Data Frames, if any, for all TCs shall be transmitted according to the priority scheme. Retransmitted lower priority Frames can be preempted by higher-priority non-replayed Frames.

The retransmission of a Frame shall be detected in the receiver by comparing the received Frame Sequence Number with the expected Frame Sequence Number. If the received Frame Sequence Number is not an expected one, then it shall be compared with last sixteen Frame Sequence Numbers prior to the expected Frame Sequence Number. If any one of that is matching to the correctly received Frame Sequence Number, then the received Frame shall be identified as a retransmitted Frame and shall be discarded. In any case, whether the Frame is identified as a retransmitted Frame or not, an acknowledgment shall be scheduled either individually or in a group for each correctly received Data Frame.

The replay timer (TCx_REPLAY_TIMER, where x = 0 or 1) guards against losing AFC or NAC Frame detection. The Frame acknowledge timer (AFCx_REQUEST_TIMER) shall guarantee that the peer transmitter schedules an AFC Frame before the transmitter TCx_REPLAY_TIMER expires. The DL_TCxReplayTimeOutVal and the DL_AFCxReqTimeOutVal for a TC shall be programmed such that this is the case. An illustration of the Frame acknowledge timer and replay timer for TC0 is given in ***Figure 69***. Initially both timers are in a reset state, they are stopped. The local TX sends TC0 Frame#0 to peer RX and starts TC0_REPLAY_TIMER. The peer RX starts AFC0_REQUEST_TIMER after reception of TC0 Frame#0. After some time, the peer TX sends AFC0#0 acknowledging TC0 Frame#0, resets and stops AFC0_REQUEST_TIMER. The timer operations for TC1 are similar to TC0.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

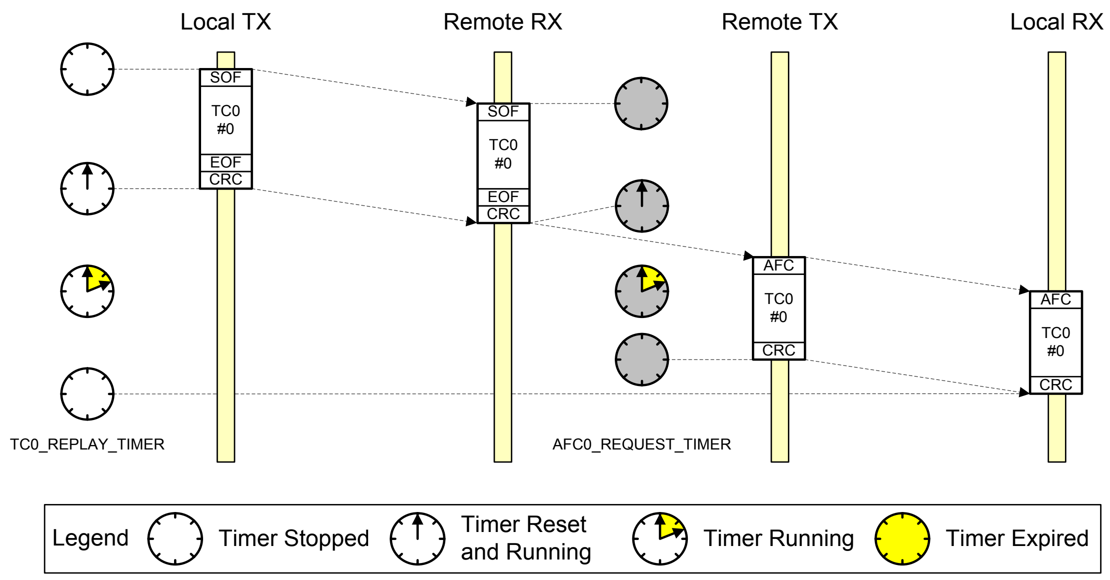


Figure 69 Definition of Frame Acknowledgment Timer and Replay Timer for TC0

TCx_REPLAY_TIMER shall be reset when at least one of the following conditions is valid:

- with the transmission of the last symbol of a TCx Data Frame (CRC symbol)
- with the reception of a correct AFCx Frame which acknowledges at least one unacknowledged Data Frame
- with the reception of a correct NAC Frame
- with the expiration of TCx_REPLAY_TIMER

TCx_REPLAY_TIMER shall run when all of the following conditions are valid:

- there exists at least one unacknowledged transmitted TCx Data Frame
- the DL Layer is not waiting for PA_INIT.cnf_L as a result of a request for PHY (re-)initialization (PA_INIT.req)

The TCx_REPLAY_TIMER should be stopped when any of the above TCx_REPLAY_TIMER run rules is not satisfied.

AFCx_REQUEST_TIMER shall be reset when at least one of the following conditions is valid:

- with the reception of a correct TCx Data Frame
- with the reception of a correct NAC Frame
- with the transmission of an AFCx Frame
- with the expiration of AFCx_REQUEST_TIMER

This AFCx_REQUEST_TIMER shall run when the following condition is valid:

- there are unacknowledged received TCx Data Frames and the DL Layer is not waiting for PA_INIT.cnf_L as a result of a request for PHY (re-)initialization (PA_INIT.req)

The AFCx_REQUEST_TIMER should be stopped when the above AFCx_REQUEST_TIMER run rules are not satisfied.

NOTE: With the expiration of the AFCx_REQUEST_TIMER the priority of the AFCx Frame is raised. See ***Section 6.6.4***.

6.6.9 AFCx Frame Transmission

All AFC Frame transmission rules shall be enabled only after the first AFCx Frame with the CReq bit set to '1' is transmitted as a result of a DL_LM_LINKSTARTUP.req or DL_HIBERNATE_EXIT.req. In all other cases, AFC Frame transmission shall be disabled for TC1 after the initial AFC exchange if TC1 is not present in the peer Device, which is the case when DL_PeerTC1Present is FALSE.

The following rules shall be used to trigger the transmission of AFCx Frames with the CReq bit set to '0':

- After reception of a NAC Frame
- Before transmitting a NAC Frame, if that NAC Frame is not triggered by FCx_PROTECTION_TIMER or TCx_REPLAY_TIMER expiry. L2 shall be able to receive AFC Frames before a NAC Frame. The cases dealing with timer expiry are explained in ***Section 6.6.7*** and ***Section 6.6.8.2***.
- After TCx_REPLAY_TIMER has expired
- After AFCx_REQUEST_TIMER has expired. In this case, the AFCx Frame's priority shall be promoted.
- When the difference between the current received Frame Sequence Number that needs to be acknowledged (currentTCxFrSeqNum) and the last acknowledged Frame Sequence Number (lastAFCxFrSeqNum) exceeds the DL_TCxOutAckThreshold threshold:

  Equation 2:

  $$[(32 + \text{currentTCxFrSeqNum} - \text{lastAFCxFrSeqNum}) \bmod 32] > \text{DL\_TCxOutAckThreshold}$$

- After reception of a retransmitted Data Frame, when the Frame Sequence Number of that frame equals the last acknowledged Frame Sequence Number. See ***Line 4291***.
- When the difference between the available credits (the A credit accumulator) and the sent credits (the S credit register) exceeds the DL_AFCxCreditThreshold threshold (see ***Section 6.6.7***):

  Equation 3:

  $$[(2^{\text{DL\_CreditValSizeBits}} + A - S) \bmod 2^{\text{DL\_CreditValSizeBits}}] > \text{DL\_AFCxCreditThreshold}$$

- After reception of an AFCx Frame with the CReq bit set to '1'. The response to this case shall be transmission of AFCx with priority promoted, even when DL_PeerTCxPresent is FALSE.
- After reception of PA_DL_PAUSE.ind while the AFCx_REQUEST_TIMER is running. The response to this case shall be transmission of AFCx with priority promoted.

The following rules shall be used to trigger transmission of AFCx Frames with CReq bit set to '1':

- After FCx_PROTECTION_TIMER has expired. The response to this case shall be transmission of AFCx with priority promoted.
- For first AFC sent during initial AFC exchange (see ***Section*** 6.7)

6.6.10 NAC Frame Transmission

Prior the transmission of a NAC Frame, the primitive PA_LANE_ALIGN.req shall be triggered.

A NAC Frame (may have RReq bit set to '1') shall be transmitted when at least one of the following conditions occurs:

- A CRC error in an incoming Frame
- A RX buffer overflow of any TC
- Reception of a Frame with a payload length more than DL_SYMBOL_MTU symbols in any TC
- An incorrect Frame Sequence Number in the received Data Frame for any TC (see following description)
- An AFCx symbol not followed by two data symbols

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

- A NAC symbol not followed by one data symbol
- An EOF_EVEN or EOF_ODD symbol not followed by a data symbol, i.e. CRC symbol
- Reception of a PA_ERROR.ind
- Reception of a COF, EOF_EVEN or EOF_ODD symbol when a Frame has not been started
- Reception of a SOF symbol when a Data Frame of the same Traffic Class is already ongoing and the Data Frame is not currently preempted
- Reception of a SOF symbol with TC=0 when a TC1 Data Frame is already ongoing
- Reception of a COF symbol during a Data Frame of the same Traffic Class, when that Data Frame has not been preempted
- Reception of a COF symbol continuing a Data Frame of a different TC
- Reception of an EOF_EVEN, EOF_ODD, or a data symbol after the CRC of a preempting Frame
- Reception of a DL control symbol with invalid values for defined fields, e.g. undefined Control Symbol Type or TC
- Reception of an unexpected framing sequence; data symbols received between Frames

When any of the above errors are detected at the receiver, any partially received Data Frame shall be discarded. After transmitting a NAC Frame, NAC transmission shall be disabled until the DL Layer receives one Data or Control (AFC or NAC) Frame without any error, e.g. CRC error, wrong Frame Sequence Number, etc., and in the case of Data or AFC, they can be of any TC. NAC Frame transmission shall be disabled immediately after scheduling a NAC as a response to any of the above errors. I.e., even if after scheduling a NAC, the DL Layer has not yet received the corresponding Service Primitives PA_ESCDATA.cnf_L() and PA_DATA.cnf_L() from the PA Layer, NAC transmission shall be disabled until the DL Layer detects an event that re-enables NAC transmission.

Although NAC Frame transmission may be disabled, a NAC Frame shall be transmitted with the RReq bit set according to parameter PAReverseLinkInitialized of the previously received PA_INIT.cnf_L, when at least one of the following conditions occurs:

- The FCx_PROTECTION_TIMER expires while waiting to receive an AFCx Frame in response to a previously sent AFCx Frame with CReq bit set to ‘1’.
- The TCx_REPLAY_TIMER expires.

In this context, an incorrect Frame Sequence Number in a received Data Frame for any TC means a Sequence Number that appears to be from a future Frame. If the expected Frame Sequence Number is ExpSeq and the received Frame Sequence Number is RcvSeq, RcvSeq is incorrect if the following equation is satisfied:

Equation 4:

$$0 < (32 + \mathrm{RcvSeq} - \mathrm{ExpSeq}) \bmod 32 < 16$$

A Data Frame is considered preempted when at least one Frame has preempted it and no COF symbol for the same Traffic Class has been received yet to resume that Frame. Examples of symbol sequences when this NAC generation condition holds are (SOF symbol with TC=0, DATA, SOF symbol with TC=0), when no preemption has occurred for the started Frame, or (SOF symbol with TC=0, DATA, AFC symbol, DATA, DATA, COF symbol with TC=0, DATA, SOF symbol with TC=0), when an earlier preemption has completed and the ongoing TC0 Data Frame is no longer preempted.

A SOF symbol during an ongoing Frame of the same TC is permitted when a replay starts, but only when the Frame is preempted, as a replay always generates an AFC Frame before any replayed Data Frame. For example, the symbol sequence (SOF symbol with TC=0, DATA, AFC symbol, DATA, DATA, SOF symbol with TC=0) is valid and does not generate a NAC Frame, since the second SOF symbol with TC=0 represents the beginning of a replayed Data Frame.

6.6.11 Error Detection Mechanism

The Data Link Layer shall detect transmission errors using CCITT CRC-16 ***[ITUT03]*** for all Data Frames and Control Frames (AFC and NAC). The CRC shall cover all symbols, including any COF symbols, from the first control symbol of the Frame (SOF, AFC or NAC symbol) up to, but not including, the CRC symbol of the Frame, as shown in ***Figure 70***. Also, bit 16 of the DL Layer data symbol that carries the CRC checksum, which is always set to zero, shall not be covered by CRC.

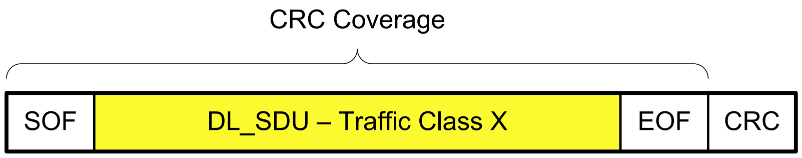


Figure 70 CRC Coverage

The CRC-16 has the following polynomial:

$$G(x) = x^{16} + x^{12} + x^5 + 1$$

To calculate the CRC-16 at the transmitter, the CRC-16 value shall be initialized to 0b1111 1111 1111 1111 (D[15], D[14], … D[1], D[0]). In addition, the first bit of the CRC-16 calculation shall be the most significant bit (bit 16) of the DL symbol. Finally, the calculation bit sequence shall be complemented to obtain the CRC-16 checksum.

A DL Layer data symbol (bit 16 = 0) shall carry the 16-bit CRC value with X^{15} through X^0 of the CRC-16 value corresponding to bit 15 through bit 0 of the symbol, as shown in ***Figure 71***.

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	X^{15}	. . .						X^8	X^7	. . .						X^0

Figure 71 Bit Allocation for CRC-16 Checksum in DL Layer Data Symbol

At the receiver, the initial value of the CRC-16 calculation shall be 0b1111 1111 1111 1111 (D[15], D[14], … D[1], D[0]). If a Frame payload (including DL Layer control symbols and DL Layer data symbols) are sent to the CRC-16 calculator at the receiver, it shall result, in the absence of transmission errors, in the same checksum that has been received.

An example illustration of CRC calculation is given in ***Annex B.3***.

6.6.12 PHY Initialization Condition

The PA_INIT.req shall be triggered if at least one of the following conditions is fulfilled:

- after reception of NAC Frame with the RReq bit set to '1'
- when FCx_PROTECTION_TIMER expires after being triggered by an AFCx Frame with CReq bit set to '1'
- after TCx_REPLAY_TIMER expires

The PA_INIT.req may be triggered if at least one of the following conditions is fulfilled:

- before transmission of a NAC Frame with the RReq bit set to '0'
- after reception of a NAC Frame with the RReq bit set to '0'

If PA_INIT.req is successful, then a NAC Frame shall be sent, and the Data Link Layer continues operations.

If PA_INIT.cnf_L with PAResult set to FALSE is received, autonomous error recovery is exhausted and a fatal error is assumed to have occurred. In this case, the DL should notify the DME using DL_LM_ERROR.ind(PA_INIT_ERROR). Once this notification has been issued, the Data Link Layer is blocked and a NAC Frame cannot be sent.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Upon reception of the DL_LM_ERROR.ind the DME should notify the Application using DME_ERROR.ind(DL, PA_INIT_ERROR).

Upon reception of DME_ERROR.ind(DL, PA_INIT_ERROR) by the Application, the Application should reset the UniPro stack as this condition is a fatal error.

6.7 Data Link Layer Initialization

The Data Link Layer initialization consists of the initial credit exchange, and is the process of preparing the Link to execute its normal operation. The initial credit exchange is started when the DME generates the DL_LM_LINKSTARTUP.req or the DL_LM_HIBERNATE_EXIT.req primitive. Both primitives shall be called only when the DL has been previously enabled by the DME via DL_LM_ENABLE_LAYER.req and enabling has been confirmed via DL_LM_ENABLE_LAYER.cnf_L.

After reception of the DL_LM_LINKSTARTUP.req or the DL_LM_HIBERNATE_EXIT.req primitive, the DL shall first transmit the AFC Frame for Traffic Class 1 (AFC1) and then transmit the AFC Frame for Traffic Class 0 (AFC0). Both AFC Frames shall have the CReq bit set. The Frame Sequence Number and the credit value carried by these two AFC Frames are the reset values of the last acknowledged Frame Sequence Number (31 as defined in ***Section 6.6.8***) and number of accumulated credits A (the size of the receiving logical buffer in credit units as defined in ***Section 6.6.7***) for the two Traffic Classes, respectively.

As long as the first AFCx Frame was not received, the FCx_PROTECTION_TIMER runs (see ***Section 6.6.7***). The FCx_PROTECTION_TIMER behavior (see ***Section 6.6.7***), rules for AFC Frame transmission (see ***Section 6.6.9***) and NAC Frame transmission (see ***Section 6.6.10***) ensure reliable exchange of credits during initialization.

If the DL Layer receives the first AFCx Frame with a flow control credit value of zero, DL_PeerTCxPresent shall be set to FALSE. If the DL Layer receives flow control credits greater than, or equal to, ten in the first received AFCx Frame, DL_PeerTCxPresent shall be set to TRUE. All other values of flow control credits received during the initial AFC exchange are not supported and results in a undefined behavior.

When one AFCx Frame is received for each Traffic Class, the initial credit exchange and thus DL Layer initialization are considered complete and L2 shall be reported to the DME as operational by sending the DL_LM_LINKSTARTUP.cnf_L primitive.

The number of credits received in the first AFC Frame for TCx shall be stored in DL_PeerTCxRxInitCreditVal. This value minus one shall define the upper bound of the valid value of DL_TCxTXFCThreshold as shown in ***Table 45***.

A UniPro implementation supporting only one Traffic Class shall implement TC0.

Figure 72 illustrates the initial credit exchange procedure for TC0 and TC1 for the case of a receiver logical buffer size of sixteen credits and a value of nine in both DL_TC0TXFCThreshold and DL_TC1TXFCThreshold.

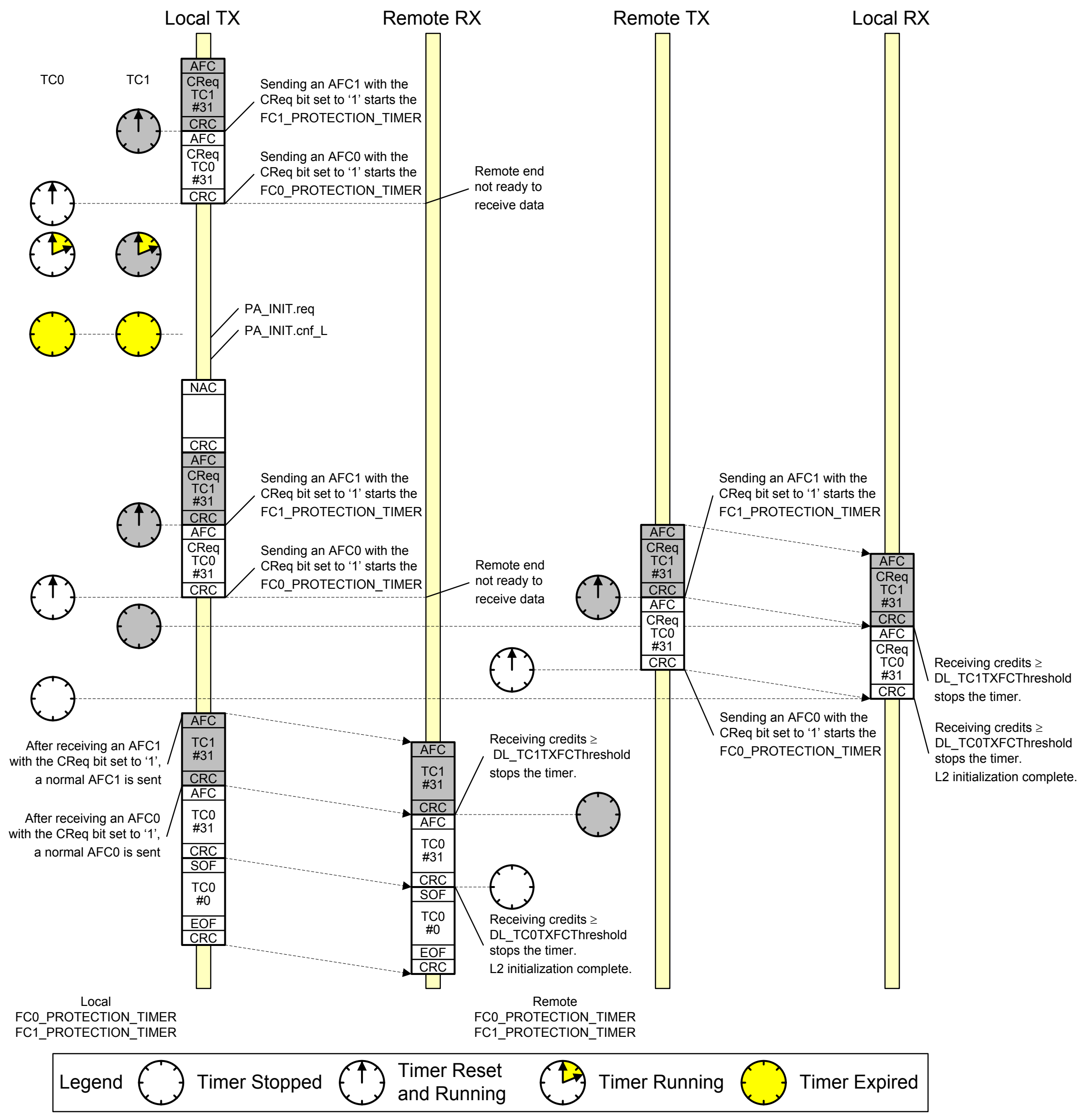

Local TX
Remote RX
Remote TX
Local RX
TC0
TC1
AFC
CReq
TC1
#31
CRC
AFC
CReq
TC0
#31
CRC
Sending an AFC1 with the CReq bit set to '1' starts the FC1_PROTECTION_TIMER
Sending an AFC0 with the CReq bit set to '1' starts the FC0_PROTECTION_TIMER
Remote end not ready to receive data
PA_INIT.req
PA_INIT.cnf_L
NAC
CRC
Sending an AFC1 with the CReq bit set to '1' starts the FC1_PROTECTION_TIMER
Sending an AFC0 with the CReq bit set to '1' starts the FC0_PROTECTION_TIMER
Remote end not ready to receive data
Receiving credits ≥ DL_TC1TXFCThreshold stops the timer.
Receiving credits ≥ DL_TC0TXFCThreshold stops the timer.
L2 initialization complete.
After receiving an AFC1 with the CReq bit set to '1', a normal AFC1 is sent
After receiving an AFC0 with the CReq bit set to '1', a normal AFC0 is sent
SOF
TC0
#0
EOF
Receiving credits ≥ DL_TC1TXFCThreshold stops the timer.
Receiving credits ≥ DL_TC0TXFCThreshold stops the timer.
L2 initialization complete.
Local
FC0_PROTECTION_TIMER
FC1_PROTECTION_TIMER
Remote
FC0_PROTECTION_TIMER
FC1_PROTECTION_TIMER
Legend
Timer Stopped
Timer Reset and Running
Timer Running
Timer Expired

Figure 72 Initial Credit Exchange Example

6.8 Management Information Base and Protocol Constants

Table 45 and ***Table 47*** show the Data Link Layer Attributes.

Table 46 lists the Protocol Constants used by the protocol specification and defines valid values for the Attributes.

In the tables in this section all Attributes should be readable.

A Data Link Layer Attribute that has more than one mandatory value should be settable.

All UniPro Attributes shall reset to a default value specified in the ***Table 45***. Some UniPro Attributes shall allow Attributes to load a persistent value instead. Persistent values may be stored and provisioned in a form of Non Volatile Memory, eFuse, etc. storage.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

4492 **Table 45 Data Link Layer (gettable, settable) Attributes**

Attribute	Attribute ID	Description	Type	Unit	Valid Attribute Value(s)	Mandatory Value(s)	Reset Value
TC0 Parameters							
DL_TC0TXFCThreshold[1]	0x2040	Threshold for triggering an AFC0 Frame with CReq bit set to '1' to request flow control update from peer This Attribute is retained during hibernate	Integer	DL_CreditUnitSize	1 to DL_PeerTC0RxInitCreditVal – 1	9	9
DL_FC0ProtectionTimeOutVal	0x2041	Expiration value of the FC0_PROTECTION_TIMER The actual duration of the timeout period shall be the value set in the Attribute ±10% A value of zero means "OFF" This Attribute is retained during hibernate	Integer	µs	0 to 65535		8191
DL_TC0ReplayTimeOutVal	0x2042	Expiration value of the TC0_REPLAY_TIMER The actual duration of the timeout period shall be the value set in the Attribute ±10% A value of zero means "OFF" This Attribute is retained during hibernate	Integer	µs	0 to 65535		65535
DL_AFC0ReqTimeOutVal	0x2043	Expiration value of the AFC0_REQUEST_TIMER The actual duration of the timeout period shall be the value set in the Attribute ±10% A value of zero means "OFF" This Attribute is retained during hibernate	Integer	µs	0 to 65535		32767

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Attribute	Attribute ID	Description	Type	Unit	Valid Attribute Value(s)	Mandatory Value(s)	Reset Value
DL_AFC0CreditThreshold	0x2044	Threshold for AFC0 triggering based on available TC0 credits at receiver. This Attribute is retained during hibernate.	Integer	DL_CreditUnitSize	0 to 127	0 to 8	0
DL_TC0OutAckThreshold	0x2045	Number of outstanding acknowledgments for TC0 (number of TC0 Data Frames received before an AFC0 Frame is sent). This Attribute is retained during hibernate.	Integer	Frame	0 to 15		0
TC1 Parameters							
DL_TC1TXFCThreshold[1]	0x2060	Threshold for triggering an AFC1 Frame with CReq bit set to '1' to request flow control update from peer. This Attribute is retained during hibernate.	Integer	DL_CreditUnitSize	1 to DL_PeerTC1RxInit CreditVal – 1	9	9
DL_FC1ProtectionTimeOutVal	0x2061	Expiration value of the FC1_PROTECTION_TIMER The actual duration of the timeout period shall be the value set in the Attribute ±10%. A value of zero means "OFF" This Attribute is retained during hibernate.	Integer	μs	0 to 65535		8191

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Attribute	Attribute ID	Description	Type	Unit	Valid Attribute Value(s)	Mandatory Value(s)	Reset Value
DL_TC1ReplayTimeOutVal	0x2062	Expiration value of the TC1_REPLAY_TIMER The actual duration of the timeout period shall be the value set in the Attribute ±10%. A value of zero means “OFF” This Attribute is retained during hibernate.	Integer	µs	0 to 65535		65535
DL_AFC1ReqTimeOutVal	0x2063	Expiration value of the AFC1_REQUEST_TIMER The actual duration of the timeout period shall be the value set in the Attribute ±10%. A value of zero means “OFF” This Attribute is retained during hibernate.	Integer	µs	0 to 65535		32767
DL_AFC1CreditThreshold	0x2064	Threshold for AFC1 triggering based on available TC1 credits at receiver. This Attribute is retained during hibernate.	Integer	DL_CreditUnitSize	0 to 127	0 to 8	0
DL_TC1OutAckThreshold	0x2065	Number of outstanding acknowledgments for TC1 (number of TC1 Data Frames received before an AFC1 Frame is sent). This Attribute is retained during hibernate.	Integer	Frames	0 to 15		0

4493 **Table 46 Data Link Layer Protocol Constants**

Attribute	Description	Type	Unit	Value
DL_MTU	Maximum Payload Length	Integer	Byte	288
DL_SYMBOL_MTU	Maximum number of symbols in a DL_MTU size payload (DL_SYMBOL_MTU = DL_MTU/2)	Integer	Symbol	144
DL_CreditValSizeBits	Number of bits used to represent Credit Value field in an AFC Frame	Integer	Bit	8
DL_CreditUnitSize	Size of one credit unit	Integer	Byte	32

4494 **Table 47 Data Link Layer (gettable, static) Attributes**

Attribute	Attribute ID	Description	Type	Unit	Valid Attribute Value(s)
DL_TxPreemptionCap	0x2000	Preemption capability on transmit side for both Traffic Classes	Bool		TRUE = 1, FALSE = 0
TC0					
DL_TC0TxMaxSDUSize	0x2001	TC0 Transmitter Maximum SDU Size	Integer	Symbol	1 to DL_SYMBOL_MTU
DL_TC0RxInitCreditVal[1]	0x2002	TC0 Receiver Initial Credit Value (initial register A value)	Integer	DL_CreditUnitSize	10 to 128
DL_TC0TxBufferSize	0x2005	TC0 Transmitter Buffer Size	Integer	DL_CreditUnitSize	10 to 128
DL_PeerTC0Present	0x2046	Peer Device supports TC0	Bool		TRUE = 1, FALSE = 0
DL_PeerTC0RxInitCreditVal[1]	0x2047	Peer TC0 Receiver Initial Credit Value (initial register A value)	Integer	DL_CreditUnitSize	10 to 128
TC1					
DL_TC1TxMaxSDUSize	0x2003	TC1 Transmitter Maximum SDU Size	Integer	Symbol	1 to DL_SYMBOL_MTU
DL_TC1RxInitCreditVal[1]	0x2004	TC1 Receiver Initial Credit Value (initial register A value)	Integer	DL_CreditUnitSize	0, 10 to 128
DL_TC1TxBufferSize	0x2006	TC1 Transmitter Buffer Size	Integer	DL_CreditUnitSize	0, 10 to 128

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Attribute	Attribute ID	Description	Type	Unit	Valid Attribute Value(s)
DL_PeerTC1Present	0x2066	Peer Device supports TC1	Bool		TRUE = 1, FALSE = 0
DL_PeerTC1RxInitCreditVal[1]	0x2067	Peer TC1 Receiver Initial Credit Value (initial register A value)	Integer	DL_CreditUnitSize	0, 10 to 128

4495 1. *The maximum value depends on the receiver buffer size.*

This page intentionally left blank.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

7 Network Layer (L3)

This section defines the service provided by the Network Layer, its protocol behavior and the Network Protocol Data Units (N_PDUs) used by the Network Layer. It aims at providing as much implementation flexibility as possible given the restrictions needed to achieve a high degree of interoperability.

Figure 73 shows the SAP model used for the Network Layer. The Network service to the Transport Layer is provided through two Traffic Class-specific Service Access Points (N_TCx_SAP). The Network Layer in turn relies on the service provided by the appropriate DL_TCx_SAP. The Network Layer Management SAP (N_LM_SAP) is provided to the Device Management Entity (DME) for configuration and control purposes. The N_TRAPx_SAP is provided as an entry to divert or inject Long Header Packets from or to software such that an existing UniPro implementation can be extended in software to support future UniPro features when they become available.

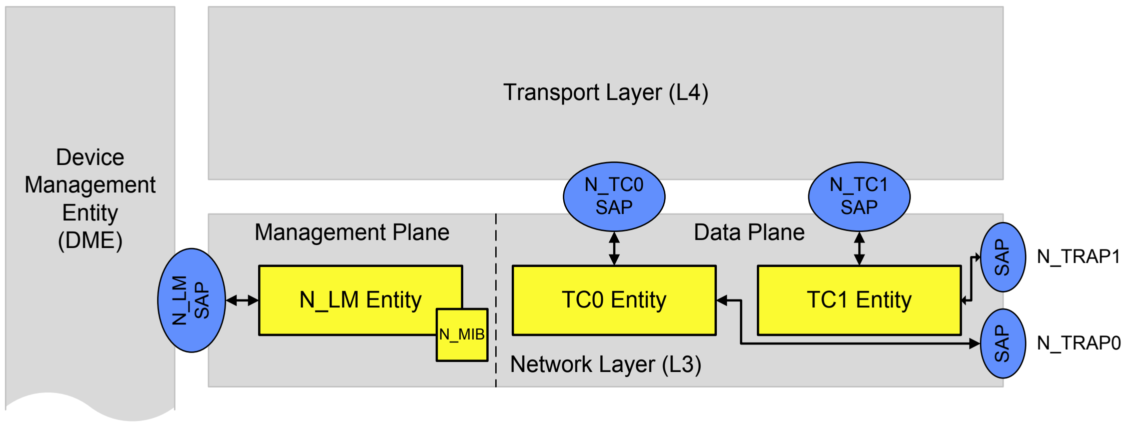


Figure 73 Network Layer SAP Model

7.1 Network Layer Service Functionality and Features

The Network Layer shall provide the following features and services to provide for the transfer of user-data between Network Service Users.

- Addressing
- Packet Composition and Packet Decomposition
- Packet format recognition
- Error handling
- Support for at least one Traffic Class
- Long Header trap to enable L3 and L4 extensions to future versions of UniPro

The Network Layer shall not limit the user-data with regard to its content and its representation.

7.2 Network Layer Addressing

This section defines the abstract syntax and semantics of the Network address. Although the current version of UniPro only provides limited capabilities in the Network Layer, a DeviceID concept is provided for compatibility with future extensions which provide full Network routing capability.

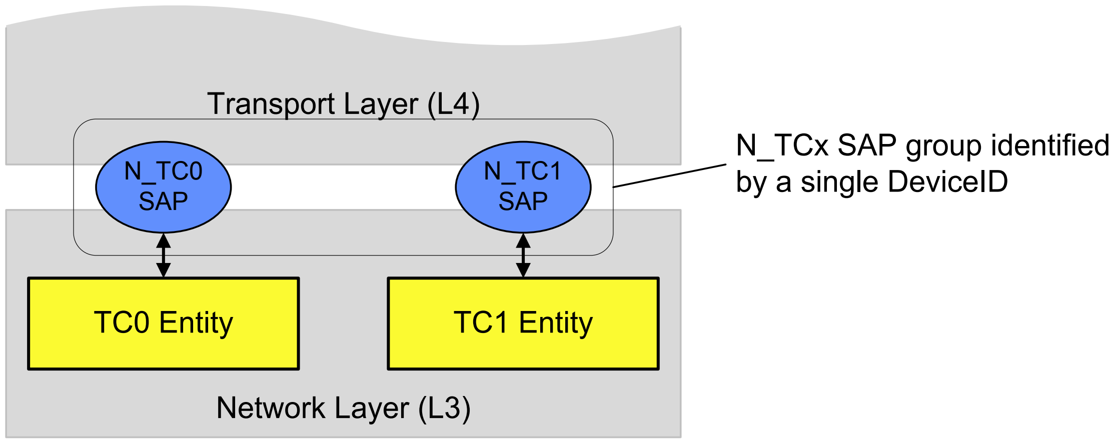


Figure 74 N_SAPs, Network Address and DeviceID

In general a Network address, defined as DeviceID, identifies one Network Layer SAP group (N_TCx_SAP) (refer to ***Figure 74***). By means of the used entity, namely TC0 or TC1, and addressed by the Network Service User, e.g. Transport Layer – L4, the DeviceID identifies uniquely any individual Network Layer SAP, i.e. N_TC0_SAP or N_TC1_SAP. The uniqueness of DeviceIDs within a UniPro Network shall be ensured.

The DeviceID, uniquely identifying a specific Device, is a 7-bit value, ranging from 0 to N_MaxDeviceID (see ***Table 65***), which is the highest possible DeviceID supported in the current version of UniPro.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

7.3 Data Communication Between Devices

Data is passed between the Network Layer and its users in Network Service Data Units (N_SDUs) qualified by certain parameters via SAPs by means of Service Primitives, defined later on. An N_SDU is the only unit, which can be exchanged between Network Service Users. The transport is performed by means of Network Protocol Data Units (N_PDUs), which are also referred to as Packets. Every single Packet is transmitted independently from each other Packet. Packets shall only be exchanged between SAPs belonging to the same Traffic Class as depicted in ***Figure 75***.

In addition, N_SDUs may be discarded upon any occurrence of certain conditions, e.g. unsupported header type, specified in ***Section 7.9.4***. The order of Packets shall not be changed and Packets shall not be duplicated during transmission.

The maximum size of a Packet depends on the N_MaxPacketSize value defined in ***Table 65***. The maximum size of user-data is limited by the Network Layer Maximum Transfer Unit (N_MTU) size, also defined in ***Table 65***. These values represent the largest sizes defined for the protocol. A UniPro stack may support a smaller user-data size than N_MTU.

The maximum transmit payload size per Packet for TC0 supported by a UniPro stack is held in T_TC0TxMaxSDUSize. Similarly, T_TC1TxMaxSDUSize holds the maximum transmit payload size per Packet for TC1. The values of N_TC0TxMaxSDUSize and N_TC1TxMaxSDUSize shall not exceed (DL_TC0TxMaxSDUSize – N_MaxHdrSize) and (DL_TC1TxMaxSDUSize – N_MaxHdrSize), respectively. See ***Table 47*** and ***Table 65*** for the Attribute and constant values

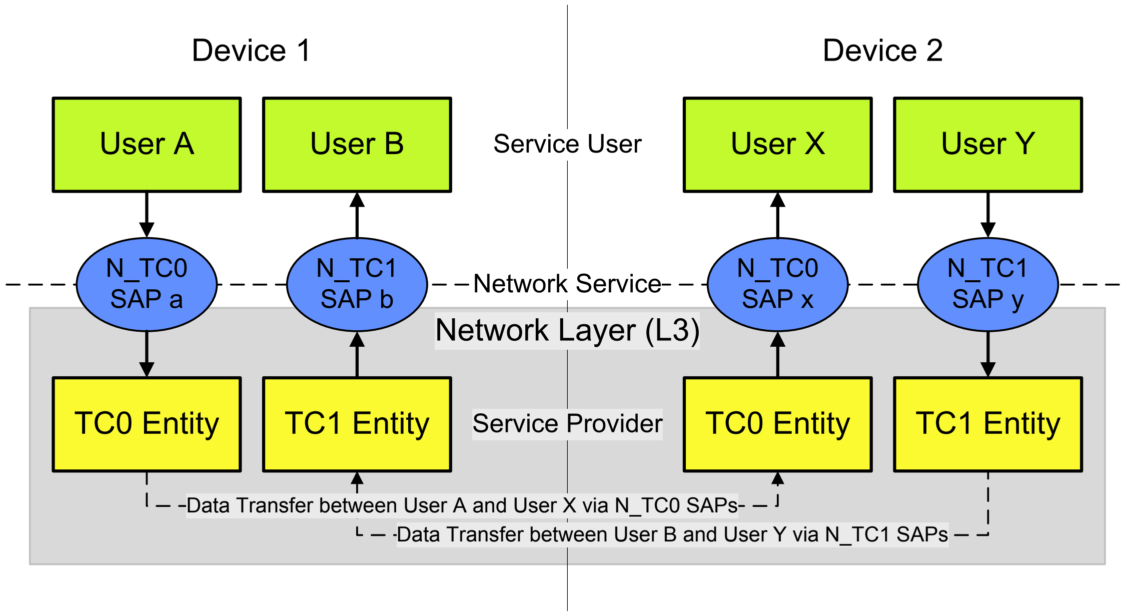


Figure 75 Network Layer Data Transfer Using Different Traffic Classes

7.4 Services Assumed From the Data Link Layer

The Data Link Layer provides its services to the Network Layer via DL_TCx_SAP. The DL_TCx_SAP is defined in ***Section 6.3***.

The Network Layer requires the following services and properties provided by the Data Link Layer:

- Reliable data transmission
- In-order delivery of data belonging to the same TC

Data transmission and reception by means of Packets are supported by the exchange of parameters and indications between the Network Layer and the Data Link Layer. These parameters and indications allow the Network Layer to control and be informed of the data transmission and reception.

The Network Layer inherits characteristics from the underlying Data Link Layer services as shown in ***Figure 76*** to enhance the basic Network Service. Refer to ***Section 6*** for detailed property descriptions.

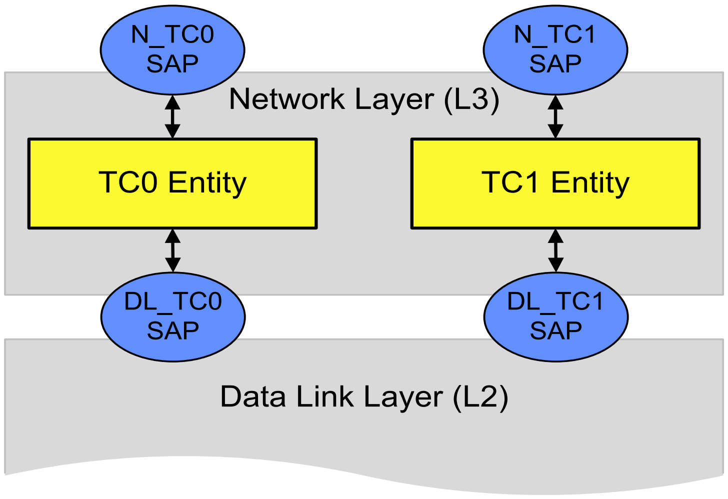


Figure 76 Relationship of Network Service Characteristics and Underlying Services

In order to support the given characteristics and to keep them independent, the Network Layer model consists of two identical TCx entities.

Note, although there are two separate entities TC0 and TC1 drawn, this does not imply any implementation choice.

7.5 Limitations and Compatibility Issues

As already specified, the Network Layer's responsibility is to provide the transport of Packets from the transmitting resource through the Network to the receiving resource, i.e. routing Packets based on logical Device identifiers, namely DeviceID. In the current version of UniPro only point-to-point Links are supported, that means no routing and switching is necessary and therefore these Network Layer-specific features and protocols are not specified here.

To ensure interoperability between the different versions of UniPro, the Network Layer services supported by the current version of UniPro are targeted to be a sub-set of any forthcoming UniPro Network Layer services.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

7.6 N_TCx_SAP

The N_TCx_SAP provides the services for basic data transmission.

At the sending side, data is passed via the N_TCx_SAP in N_SDUs to the Network Layer to transfer it to the peer Network Layer. At the recipient side, the Network Layer delivers received data in N_SDUs to the upper layer. The Traffic Class identification is done based on the used SAP.

The N_TCx_SAP shall support the transmission of user-supplied data between users, which shall not be modified and shall not be re-ordered by the provider. The user may transmit any integral number of bytes greater than zero up to the N_MTU.

7.6.1 Data Transfer Primitives

The primitives covered in this section are listed in ***Table 48***.

Table 48 N_TCx_SAP Primitives

Name	Request	Indication	Local Response	Response	Local Confirm	Confirm
N_DATA_SHDR	***7.6.1.1***	***7.6.1.3***	***7.6.1.4***	–	***7.6.1.2***	–
N_DATA_LHDR_TRAP	***7.6.1.5***	***7.6.1.7***	***7.6.1.8***	–	***7.6.1.6***	–

Table 49 lists the parameters that appear in the N_TCx_SAP primitives.

Table 49 N_TCx_SAP Primitive Parameters

Name	Type	Valid Range	Value	Description
DestPeerDeviceID	Integer	0 to N_MaxDeviceID		Specifies the DeviceID of the destination Device of the N_SDU
DeviceIDOffset	Integer	0 to N_MaxDeviceIDOffset		Specifies the DeviceID offset that is used for the encoding and decoding of CPortIDs. See ***Section 8.2***
Payload	byte string			Specifies the data passed by N_DATA_SHDR.req from Layer 4 and by N_DATA_SHDR.ind to Layer 4. The length of the payload shall be in the range from 1 to N_MTU
L3ResultCode	Enumeration	SUCCESS	0	Indicates the result of the request
		NO_PEER_TC	1	
L2Payload	byte string			Specifies the data passed by N_DATA_LHDR_TRAP.req and N_DATA_LHDR_TRAP.ind between the L3 extension and Layer 3. The length of the payload shall be in the range from 1 to DL_MTU
LhdrTrapStatus	Enumeration	OK	0	Indicates whether received data has been lost in Layer 3 due to a slow Application
		LOST_DATA	1	

7.6.1.1 N_DATA_SHDR.req

This primitive requests the transfer of user payload to a peer Network Layer by means of a DT SH N_PDU (Refer to ***Section 7.8.1.1***), i.e. no source DeviceID or other L3-specific information will be transferred.

The provided DestPeerDeviceID shall identify the recipient of the payload. The DeviceIDOffset parameter shall be in the range [0, N_MaxDeviceIDOffset], which is derived from Transport Layer protocol constants, and is defined in ***Table 65***.

The user may transmit any integral number of bytes that shall be greater than zero up to the N_TCxTxMaxSDUSize. The value of N_TCxTxMaxSDUSize depends on the value of DL_TCxTxMaxSDUSize and the value of N_MaxHdrSize.

The semantics of this primitive are:

N_DATA_SHDR.req(DestPeerDeviceID, DeviceIDOffset, Payload)

When Generated

The Network Service User shall generate a N_DATA_SHDR.req primitive to send user payload to the specified recipient.

Effect on Receipt

The instructed Network Layer shall initiate the transmission of the given payload.

7.6.1.2 N_DATA_SHDR.cnf_L

This primitive informs the Service User that the Service Provider, so L3 in this case, is ready to accept a new N_DATA_SHDR.req primitive.

The semantics of this primitive are:

N_DATA_SHDR.cnf_L(L3ResultCode)

When Generated

This primitive shall be generated when the Network Service Provider is ready to accept a new request to transfer user payload.

If the Network Service Provider is ready to accept a new N_DATA_SHDR.req primitive, which implies that the TCx Traffic Class is present in the peer Device, the returned L3ResultCode shall be SUCCESS. All other L3ResultCode values should indicate a failure.

If the peer Device does not implement the TCx Traffic Class, the value of the L3ResultCode shall be NO_PEER_TC. The Network Layer learns about the peer Device not implementing the TCx Traffic Class when it receives DL_DATA.cnf_L with L2ResultCode set to NO_PEER_TC.

Effect on Receipt

Following the emission of a N_DATA_SHDR.req primitive and prior to the reception of a N_DATA_SHDR.cnf_L primitive, the Network Layer Service User shall not emit a new N_DATA_SHDR.req primitive.

The Network Service User may emit a new N_DATA_SHDR.req primitive immediately following a reset or after the reception of the N_DATA_SHDR.cnf_L primitive corresponding to a previously emitted N_DATA_SHDR.req primitive.

This primitive is invoked by the L3_TC_tx process, prior to return to the Idle state, when leaving the WaitDLCnfL state. The L3_TC_tx process defines the behavior associated with the handshake between N_DATA_SHDR.req and N_DATA_SHDR.cnf_L.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

7.6.1.3 N_DATA_SHDR.ind

This primitive shall report the reception of user payload, carried by a DT SH N_PDU; no sender is identified. The DeviceIDOffset shall be provided to the Service User for Transport Layer decoding purposes (see ***Section 8.10.4***). The payload may consist of any integral number of bytes greater than zero up to the N_MTU.

The semantics of this primitive are:

N_DATA_SHDR.ind(DeviceIDOffset, Payload)

When Generated

This primitive shall be generated when the Network Service Provider has successfully processed a received DL SH N_PDU.

Effect on Receipt

Upon reception of a N_DATA_SHDR.ind primitive, the Network Layer Service User should consume the payload.

7.6.1.4 N_DATA_SHDR.rsp_L

This primitive informs the Network Layer that the Transport Layer is ready to accept a new N_DATA_SHDR.ind primitive.

The semantics of this primitive are:

N_DATA_SHDR.rsp_L( )

When Generated

This primitive shall be generated when the Transport Layer is ready to accept and process a new N_PDU.

Effect on Receipt

Following the emission of a N_DATA_SHDR.ind primitive and prior the reception of a N_DATA_SHDR.rsp_L primitive, the Network Layer shall not emit a new N_DATA_SHDR.ind primitive. The Network Layer may emit a new N_DATA_SHDR.ind primitive only just after reset, or after the reception of N_DATA_SHDR.rsp_L primitive corresponding to a previously emitted N_DATA_SHDR.ind primitive.

This primitive is invoked by the L3_TC_rx process, prior to return to the Idle state, when leaving the WaitDLRspL state. The L3_TC_rx process defines the behavior associated with the handshake between N_DATA_SHDR.ind and N_DATA_SHDR.rsp_L.

7.6.1.5 N_DATA_LHDR_TRAP.req

The N_DATA_LHDR_TRAP.req primitive is used to provide a Long Header Packet for transmission to the Data Link Layer.

The semantics of this primitive are:

N_DATA_LHDR_TRAP.req(L2Payload)

When Generated

A Network Layer extension shall generate a N_DATA_LHDR_TRAP.req primitive to send a Long Header Packet to Data Link Layer. The specification of the Network Layer extension is outside the scope of this document.

Effect on Receipt

The received Data Link Layer payload shall be transmitted to the Data Link Layer without any processing.

7.6.1.6 N_DATA_LHDR_TRAP.cnf_L

This primitive informs the Network Layer extension that Network Layer is ready to accept a new N_DATA_LHDR_TRAP.req primitive.

The semantics of this primitive are:

N_DATA_LHDR_TRAP.cnf_L(L3ResultCode)

When Generated

This primitive shall be generated when the Network Layer is ready to accept a new request from the Network Layer extension. If the Network Layer is ready and TCx is present in the peer Device, the returned L3ResultCode shall be SUCCESS. All other L3ResultCode values should indicate a failure.

If the peer Device does not implement the TCx Traffic Class, the value of the L3ResultCode shall be NO_PEER_TC. The Network Layer learns about the peer Device not implementing the TCx Traffic Class when it receives DL_DATA.cnf_L with L2ResultCode set to NO_PEER_TC.

Effect on Receipt

Following the emission of a N_DATA_LHDR_TRAP.req primitive and prior to the reception of a N_DATA_LHDR_TRAP.cnf_L primitive, the Network Layer extension shall not emit a new N_DATA_LHDR_TRAP.req primitive.

The Network Layer extension may emit a new N_DATA_LHDR_TRAP.req primitive immediately following a reset or after the reception of the N_DATA_LHDR_TRAP.cnf_L primitive corresponding to a previously emitted N_DATA_LHDR_TRAP.req primitive.

7.6.1.7 N_DATA_LHDR_TRAP.ind

This N_DATA_LHDR_TRAP.ind primitive reports the reception of user-data to the Network Layer extension.

The semantics of this primitive are:

N_DATA_LHDR_TRAP.ind(L2Payload, LhdrTrapStatus)

When Generated

The primitive shall be generated when the Data Link Layer payload received from the Data Link Layer has the L3s bit set to ‘0’ indicating a Long Header. Long Headers cannot be interpreted in the current version of UniPro, and, therefore, are forwarded to the Network Layer extension, which implements future features of UniPro. The Data Link Layer payload shall be forwarded without modification to the Network Layer extension.

The primitive shall indicate in LhdrTrapStatus whether any data has been dropped from the previous generation of the primitive. An LhdrTrapStatus value of OK shall indicate that no data has been dropped, and a value of LOST_DATA shall indicate that data was dropped.

Effect on Receipt

The Network Layer extension processes the Long Header Packet.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

7.6.1.8 N_DATA_LHDR_TRAP.rsp_L

The N_DATA_LHDR_TRAP.rsp_L primitive indicates that the Network Layer extension is ready to accept a new N_DATA_LHDR_TRAP.ind primitive.

The semantics of this primitive are:

N_DATA_LHDR_TRAP.rsp_L( )

When Generated

This primitive shall be generated when the Network Layer extension is ready to accept and process a new Data Link Layer payload.

Effect on Receipt

Following the emission of a N_DATA_LHDR_TRAP.ind primitive and prior to reception of a N_DATA_LHDR_TRAP.rsp_L primitive, the Network Layer shall not emit a new N_DATA_LHDR_TRAP.ind primitive. The Network Layer may emit a new N_DATA_LHDR_TRAP.ind primitive only just after reset, or after reception of the N_DATA_LHDR_TRAP.rsp_L primitive corresponding to a previously issued N_DATA_LHDR_TRAP.ind primitive.

7.7 N_LM_SAP

The Network Layer Management (N_LM_SAP) offers three groups of Service Primitives: Configuration primitives, Control primitives and Status primitives. The primitives on the N_LM_SAP are used by the Device Management Entity (DME) to configure and control the layer and receive layer status information.

The Configuration primitives enable access to Network Layer Attributes. This information is represented as a Network Layer-specific Management Information Base (N_MIB). The N_MIB is regarded as “contained” within the N_LM entity. Multiple accesses to the N_MIB via the Configuration primitives shall not occur concurrently. The available Network Layer Attributes are defined in ***Section 7.14***.

The Control primitives provide direct control of the Network Layer. Control primitives are generated by the DME and can occur concurrently.

The Status primitives indicate status information of the Network Layer. Status primitives are generated by the Network Layer and can occur concurrently.

7.7.1 Configuration Primitives

The N_LM configuration primitives, GET and SET, are used by the DME to retrieve and store values, respectively, for the configuration Attributes in the N_MIB.

The GET and SET primitives are represented as requests with associated confirm primitives. These primitives are prefixed by N_LM. The primitives are summarized in ***Table 50***.

Table 50 N_LM_SAP Configuration Primitives

Name	Request	Indication	Local Response	Response	Local Confirm	Confirm
N_LM_GET	***7.7.1.1***	–	–	–	***7.7.1.2***	–
N_LM_SET	***7.7.1.3***	–	–	–	***7.7.1.4***	–

The parameters used for these primitives are defined in the next table.

Table 51 N_LM_SAP Configuration Primitive Parameters

Name	Type	Valid Range	Value	Description
MIBattribute	Integer	MIBattribute values fall between 0x3000 and 0x3FFF The valid MIBattribute values are defined in ***Section 7.14***		The address of the MIB Attribute
MIBvalue	Variable	As defined in ***Section 7.14***		The value of the MIB Attribute
SetType	Enum	NORMAL	0	Select whether the actual value (NORMAL) or the Attribute's non-volatile value (STATIC) is addressed
		STATIC	1	
ConfigResultCode	Enum	SUCCESS	0	Indicates the result of the request
		INVALID_MIB_ATTRIBUTE	1	
		INVALID_MIB_ATTRIBUTE_VALUE	2	
		READ_ONLY_MIB_ATTRIBUTE	3	
		WRITE_ONLY_MIB_ATTRIBUTE	4	

7.7.1.1 N_LM_GET.req

This primitive requests the value of the Attribute identified by MIBattribute.

The semantics of this primitive are:

N_LM_GET.req(MIBattribute)

When Generated

This primitive is generated by the DME to obtain the value of the Attribute identified by MIBattribute.

Effect on Receipt

The Network Layer entity shall attempt to retrieve the requested Attribute value and respond with N_LM_GET.cnf_L that gives the retrieved value.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

7.7.1.2 N_LM_GET.cnf_L

This primitive indicates the success or failure of N_LM_GET.req, and, is successful, returns the Attribute value.

The semantics of this primitive are:

N_LM_GET.cnf_L(ConfigResultCode, MIBvalue)

The primitive parameters are defined in ***Table 51***.

The Network Layer shall set ConfigResultCode in N_LM_GET.cnf_L to one of the values shown in ***Table 52*** for the condition given in the table. The Network Layer should not set ConfigResultCode to INVALID_MIB_ATTRIBUTE_VALUE or READ_ONLY_MIB_ATTRIBUTE.

Table 52 N_LM_GET.cnf_L ConfigResultCode Values

ConfigResultCode	Condition
SUCCESS	The request succeeds, i.e. the MIB Attribute indicated by N_LM_GET.req MIBattribute is gettable The Network Layer shall set MIBvalue in N_LM_GET.cnf_L to the Attribute value
INVALID_MIB_ATTRIBUTE	The Attribute indicated by MIBattribute is invalid or not implemented The value of MIBvalue in N_LM_GET.cnf_L is undefined.
WRITE_ONLY_MIB_ATTRIBUTE	The Attribute indicated by MIBattribute exists, but is not gettable The value of MIBvalue in N_LM_GET.cnf_L is undefined

When Generated

The Network Layer entity shall generate the N_LM_GET.cnf_L primitive in response to the most recent N_LM_GET.req from the DME.

Effect on Receipt

The DME is informed about the result of the operation, and in case ConfigResultCode is SUCCESS, MIBvalue carries the Attribute value. For any other value of ConfigResultCode, MIBvalue is undefined.

7.7.1.3 N_LM_SET.req

This primitive attempts to set the value (SetType = NORMAL) or the reset value (SetType = STATIC) of the Attribute indicated by MIBattribute to the MIBvalue.

The semantics of this primitive are:

N_LM_SET.req(SetType, MIBattribute, MIBvalue)

The primitive parameters are defined in ***Table 51***.

When Generated

This primitive is generated by the DME to set the value or reset value of the indicated Attribute.

Effect on Receipt

If SetType is set to NORMAL, the Network Layer shall attempt to set the Attribute addressed by MIBattribute to the MIBvalue. If SetType is set to STATIC, the Network Layer shall attempt to set the reset value of the Attribute addressed by MIBattribute to the MIBvalue.

The Network Layer uses N_LM_SET.cnf_L to inform DME of the result of N_LM_SET.req.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

7.7.1.4 N_LM_SET.cnf_L

This primitive reports the results of an attempt to set the value of an Attribute or its reset value.

The semantics of this primitive are:

N_LM_SET.cnf_L(ConfigResultCode)

The primitive parameter is defined in ***Table 51***.

The Network Layer entity shall set ConfigResultCode in N_LM_SET.cnf_L to one of the values shown in ***Table 53*** for the conditions given in the table. The N_LM entity should not set ConfigResultCode to WRITE_ONLY_MIB_ATTRIBUTE.

Table 53 N_LM_SET.cnf_L ConfigResultCode Values

ConfigResultCode	Condition
SUCCESS	The request succeeds, i.e. the value or reset value of the Attribute indicated by MIBattribute is set to the value of MIBvalue.
INVALID_MIB_ATTRIBUTE	The MIBattribute value is invalid, or otherwise The Attribute indicated by MIBattribute and SelectorIndex is not implemented, or otherwise If SetType is STATIC, the Attribute indicated by MIBattribute and SelectorIndex does not support setting its reset value
READ_ONLY_MIB_ATTRIBUTE	The Attribute indicated by MIBattribute exists, but is not settable This ConfigResultCode value shall only be used if SetType is NORMAL
INVALID_MIB_ATTRIBUTE_VALUE	The Attribute indicated by MIBattribute exists and is settable (SetType = NORMAL) or allows its reset value to be set (SetType = STATIC), but the value of MIBvalue is outside of the implemented range, or outside of the valid value range, for the Attribute

When Generated

The Network Layer shall generate the N_LM_SET.cnf_L primitive in response to the most recent N_LM_SET.req from the DME.

Effect on Receipt

N_LM_SET.cnf_L confirms the success or failure of the N_LM_SET.req at the Network Layer, and should have no further effects at the DME.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

7.7.2 Control Primitives

The Service Primitives in this section are provided for controlling the Network Layer.

The primitives covered in this section are listed in ***Table 54***.

Table 54 N_LM_SAP Control Primitives

Name	Request	Indication	Local Response	Response	Local Confirm	Confirm
N_LM_RESET	***7.7.2.1***	–	–	–	***7.7.2.2***	–
N_LM_ENABLE_LAYER	***7.7.2.3***	–	–	–	***7.7.2.4***	–
N_LM_HIBERNATE_ENTER	***7.7.2.5***	–	–	–	***7.7.2.6***	–
N_LM_HIBERNATE_EXIT	***7.7.2.7***	–	–	–	***7.7.2.8***	–
N_LM_PRE_HIBERNATE_EXIT	***7.7.2.9***	–	–	–	***7.7.2.10***	–

Table 55 lists the parameters that appear in the N_LM_SAP control primitives.

Table 55 N_LM_SAP Control Primitive Parameters

Name	Type	Valid Range	Value	Description
ResetLevel	Enum	COLD	0	Defines the reset level of the requested reset
		WARM	1	

7.7.2.1 N_LM_RESET.req

This primitive requests reset of the Network Layer.

The semantics of this primitive are:

N_LM_RESET.req(ResetLevel)

When Generated

This primitive is generated by the DME when it needs to reset the Network Layer. The Network Layer is reset in combination with resetting of all the other UniPro layers as described in the reset procedure.

Effect on Receipt

The Network Layer sets itself to the reset states and Attribute values, and discards all N_SDUs currently processed. Then, the Network Layer shall generate N_LM_RESET.cnf_L to the DME. After issuing N_LM_RESET.cnf_L, the Network Layer shall not react to any Network Layer SAP primitive other than N_LM_RESET.req and N_LM_ENABLE_LAYER.req

The ResetLevel COLD resets the complete Network Layer including the statistics and configuration Attributes. The ResetLevel WARM resets the Network Layer without the statistics.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

7.7.2.2 N_LM_RESET.cnf_L

The N_LM_RESET.cnf_L primitive is used during the UniPro reset procedure (see ***Section 9.11.1***).

The semantics of this primitive are:

N_LM_RESET.cnf_L( )

When Generated

The N_LM_RESET.cnf_L primitive is issued to the DME in response to N_LM_RESET.req to indicate that the Network Layer came out of reset.

Effect on Receipt

The DME is informed that the Network Layer came out of reset.

7.7.2.3 N_LM_ENABLE_LAYER.req

The N_LM_ENABLE_LAYER.req primitive is used during the UniPro boot procedure (see ***Section 9.11.2***).

The semantics of this primitive are:

N_LM_ENABLE_LAYER.req( )

When Generated

The N_LM_ENABLE_LAYER.req primitive is part of the UniPro boot procedure (see ***Section 9.11.2***), and is generated by the DME after the Network Layer came out of reset and the Data Link Layer was enabled.

Effect on Receipt

The Network Layer shall respond with N_LM_ENABLE_LAYER.cnf_L to the DME. After issuing N_LM_ENABLE_LAYER.cnf_L, all Network Layer functionality and SAP primitives shall be enabled.

7.7.2.4 N_LM_ENABLE_LAYER.cnf_L

The N_LM_ENABLE_LAYER.cnf_L primitive is used during the UniPro boot procedure (see ***Section 9.11.2***) to indicate to the DME that the Network Layer is enabled.

The semantics of this primitive are:

N_LM_ENABLE_LAYER.cnf_L( )

When Generated

The N_LM_ENABLE_LAYER.cnf_L primitive is issued to the DME in response to N_LM_ENABLE_LAYER.req to indicate that the Network Layer was enabled.

Effect on Receipt

The DME is informed that the Network Layer is enabled.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

7.7.2.5 N_LM_HIBERNATE_ENTER.req

The N_LM_HIBERNATE_ENTER.req primitive requests the Network Layer enter hibernation.

The semantics of this primitive are:

N_LM_HIBERNATE_ENTER.req( )

When Generated

The DME generates the N_LM_HIBERNATE_ENTER.req primitive to request the Network Layer enter hibernation.

Effect on Receipt

The Network Layer shall issue the N_LM_HIBERNATE_ENTER.cnf_L primitive and then enter hibernation. During hibernation, the Network Layer shall retain N_DeviceID and N_DeviceID_valid, and may lose all other state information.

7.7.2.6 N_LM_HIBERNATE_ENTER.cnf_L

The N_LM_HIBERNATE_ENTER.cnf_L primitive is used to indicate that the Network Layer is about to hibernate.

The semantics of this primitive are:

N_LM_HIBERNATE_ENTER.cnf_L( )

When Generated

The N_LM_HIBERNATE_ENTER.cnf_L primitive is generated by the Network Layer in response to a N_LM_HIBERNATE_ENTER.req primitive and it indicates that the Network Layer is hibernating.

Effect on Receipt

The DME is informed about the Network Layer entering hibernate.

7.7.2.7 N_LM_HIBERNATE_EXIT.req

The N_LM_HIBERNATE_EXIT.req primitive is used to request the Network Layer to exit hibernation and return to normal operation.

The semantics of this primitive are:

N_LM_HIBERNATE_EXIT.req( )

When Generated

The DME generates the N_LM_HIBERNATE_EXIT.req primitive to request the Network Layer to exit hibernation.

Effect on Receipt

The Network Layer shall go to its reset state and restore the N_DeviceID and N_DeviceID_valid from their retained values. Then, the Network Layer shall issue the N_LM_HIBERNATE_EXIT.cnf_L primitive to the DME.

7.7.2.8 N_LM_HIBERNATE_EXIT.cnf_L

The N_LM_HIBERNATE_EXIT.cnf_L primitive is used to indicate to the DME that the Network Layer has exited hibernation and is operating normally.

The semantics of this primitive are:

N_LM_HIBERNATE_EXIT.cnf_L( )

When Generated

The N_LM_HIBERNATE_EXIT.cnf_L primitive is generated by the Network Layer in response to a N_LM_HIBERNATE_EXIT.req primitive and it indicates that the Network Layer has exited hibernation and is operating normally.

Effect on Receipt

The DME is informed about the Network Layer exiting hibernate.

7.7.2.9 N_LM_PRE_HIBERNATE_EXIT.req

Support for this primitive is optional. However, if an implementation supports this primitive it shall implement it as described in this section.

This primitive is used to request the Network Layer to stop, or prepare to stop, hibernating, and prepare to return to normal operation. Hibernate exit is requested with the N_LM_HIBERNATE_EXIT.req primitive.

The semantics of this primitive are:

N_LM_PRE_HIBERNATE_EXIT.req( )

When Generated

The DME shall generate this primitive when the Network Layer needs to prepare to exit Hibernate prior to exiting Hibernate.

Effect on Receipt

The Network Layer shall prepare to exit Hibernate State.

7.7.2.10 N_LM_PRE_HIBERNATE_EXIT.cnf_L

Support for this primitive is optional. However, if an implementation supports this primitive it shall implement it as described in this section.

This primitive is used to indicate to the DME that the Network Layer has been prepared to return to normal operation after hibernating.

The semantics of this primitive are:

N_LM_PRE_HIBERNATE_EXIT.cnf_L( )

When Generated

This primitive is generated by the Network Layer in response to a N_LM_PRE_HIBERNATE_EXIT.req primitive. It indicates that the Network Layer has been prepared for return to normal operation.

Effect on Receipt

The DME is informed of the completion of all necessary actions required by the Network Layer to return to normal operation after preparation for exiting Hibernate.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

7.7.3 Status Primitives

The Service Primitives in this section are provided for gathering status information of the Network Layer. The primitives covered in this section are listed in ***Table 56***.

Table 56 N_LM_SAP Status Primitives

Name	Request	Indication	Local Response	Response	Local Confirm	Confirm
N_LM_DISCARD	–	*7.7.3.1*	–	–	–	–

Table 57 lists the parameters that appear in the N_LM_SAP status primitives.

Table 57 N_LM_SAP Status Primitive Parameters

Name	Type	Valid Range	Value	Description
L3DiscardReasonCode	Enum	UNSUPPORTED_ HEADER_TYPE	1	A N_PDU whose header contains unsupported values was received
		BAD_DEVICEID_ENC	2	N_DeviceID_valid is TRUE, and either an N_PDU was received with a DestDeviceID_Enc less than N_DeviceID or the computed DeviceIDOffset is greater than N_MaxDeviceIDOffset
		LHDR_TRAP_ PACKET_DROPPING	3	Indicates that a Long Header Packet has been dropped because the L3 extension was not available to accept the Packet
		MAX_N_PDU_ LENGTH_EXCEEDED	4	Indicates that a N_PDU with a payload longer than the N_MTU has been received

7.7.3.1 N_LM_DISCARD.ind

This primitive indicates that the discard N_PDU feature has been performed (refer to ***Section 7.9.4***) and reports the reason for triggering the operation.

The semantics of this primitive are:

N_LM_DISCARD.ind(L3DiscardReasonCode)

When the N_LM_DISCARD.ind is generated the provided L3DiscardReasonCode shall be set according to the cases described in ***Section 7.9.4***.

When Generated

This primitive shall be generated by the Network Layer as a result of performing the discard N_PDU feature.

Effect on Receipt

The DME is notified of the cause of the discard. This information may be used to take action upon or for statistical procedures.

7.8 Structure and Encoding of Network Protocol Data Units

This section defines the Network Layer Protocol Data Units (usually referred to as Packets) and shows the header fields and their meaning.

A Packet (N_PDU) shall contain an integral number of bytes greater than zero and shall be limited to N_MaxPacketSize. This restriction forces the Packet to always fit entirely into a DL Frame.

7.8.1 Data N_PDUs

Table 58 lists the Data Network Protocol Data Units (DT N_PDU) types that shall be supported.

Table 58 User-Data Network Layer PDU Types

N_PDU Name	Mnemonic	Overall Header Size	Remarks/Limitations	Section
Data N_PDU with short L3 header	DT SH N_PDU	8-bit	Payload length shall be in the range from 1 to N_MTU	***7.8.1.1***

7.8.1.1 DT SH N_PDU

If the L3s bit is set to ‘1’, the N_PDU is known as DT SH N_PDU, or a Data N_PDU with a short L3 Header. The DT SH N_PDU is used to transfer Network Service User-data (N_SDU) to a peer Network Layer.

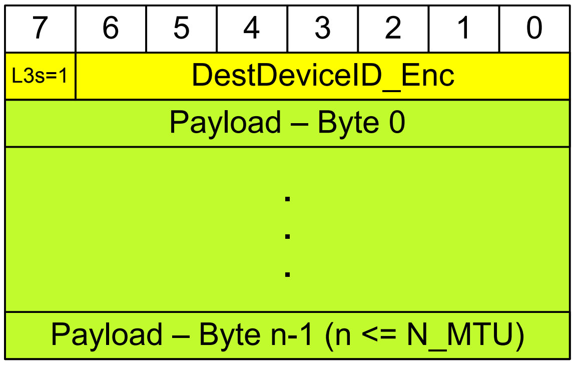


Figure 77 Network Layer Data N_PDU with Short Header (DT SH N_PDU)

The natural DT SH N_PDU granularity is bytes, i.e. any integral number of data bytes, i.e. N_SDU, in the range from 1 to N_MTU can be transferred.

7.8.2 N_PDU Fields

Table 59 lists the N_PDU fields.

Table 59 Network Layer N_PDU Header Fields

Field Name	Field Description	Field Size	Remarks/Limitations	Section
L3s	L3 Header type field	1-bit	Identifies the N_PDU	***7.8.2.1***
DestDeviceID_Enc	Address identifying the destination of a Packet	7-bit	Note, value also encodes parts of L4 DestPeerCPortID, refer to ***Section 8.10.4***	***7.8.2.2***
Payload (N_SDU)	User-data (Network Service Data Unit)	1 to N_MTU bytes	Carries the payload of a Packet	***7.8.2.3***

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

7.8.2.1 L3 Header Type Field (L3s)

The L3s header field serves as a header type bit. The value is determined by the Service Primitive used. The N_DATA_SHDR Service Primitive shall set the L3s header field to ‘1’. The N_DATA_LHDR_TRAP Service Primitive carries an existing N_PDU in which the L3s bit is set to '0' (see ***Section 7.12***). Upon reception of an N_PDU with the L3s bit set to ‘1’, the N_PDU is processed by the Network Layer and the payload forwarded to Transport Layer via N_DATA_SHDR.ind (see ***Section 7.6.1.3***). A received N_PDU with L3s set to '0' is forwarded unmodified to the Network Layer extension via N_DATA_LHDR_TRAP.ind (see ***Section 7.12***).

Table 60 shows the supported settings.

Table 60 Settings of the L3s Field

Field Name	Size	Value	Meaning	Remarks
L3s	1-bit	0	Long Header Data N_PDU is used	The Long Header Data N_PDU will be defined in a future version of UniPro Currently, N_PDUs with L3=0 are received from or forwarded to the Network Layer extension via the Long Header Trap (see ***Section 7.12***)
		1	Short Header Data N_PDU (DT SH N_PDU) is used	Caused by N_DATA_SHDR primitive use

7.8.2.2 Destination Device ID

The DestDeviceID_Enc field shall encode the DestPeerDeviceID and DeviceIDOffset parameters provided via the N_DATA_SHDR.req primitive. The encoding shall follow the formula below:

DestDeviceID_Enc = DestPeerDeviceID + DeviceIDOffset

Any integral number in the range from 0 to N_MaxDeviceID shall be supported.

Table 61 shows the supported settings.

Table 61 Settings of the DestDeviceID_Enc Field

Field Name	Size	Value	Meaning	Remarks
DestDeviceID_Enc	7-bit	0 to N_MaxDeviceID	Encoded DeviceID value of targeted Device	–

7.8.2.3 Payload – Network Service Data Unit

The N_SDU field carries the payload data given by the Service User and can therefore carry any value.

Table 62 Settings of the Payload Field

Field Name	Size	Value	Meaning	Remarks
Payload	1 to N_MTU bytes	Any byte string	Payload	Size depends on the configured maximum payload size of the sending N_TCx entity (N_TCxTxMaxSDUSize)

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

7.8.3 Relationship of N_PDU Type Selection and Service Primitive Usage

The correct N_PDU type for transmission and upon reception shall be determined by the association in ***Table 63***.

Table 63 Service Primitive to N_PDU Association Table

Used Service Primitive	Selected N_PDU Type	Remarks
N_DATA_SHDR	DT SH N_PDU	Data Short Header Network Protocol Data Unit

7.9 Protocol Features

7.9.1 N_PDU Composition Feature

This feature is responsible for the composition of Network Protocol Data Units (N_PDUs). As depicted in ***Figure 78***, this feature constructs and adds a L3 header to the provided N_SDU to form an N_PDU. The header consists of different fields specified in ***Section 7.8.2***. For the construction of the N_PDU additional control information is needed, called Protocol Control Information (PCI). The PCI is obtained from local layer-specific state information and parameters given by means of Service Primitive requests.

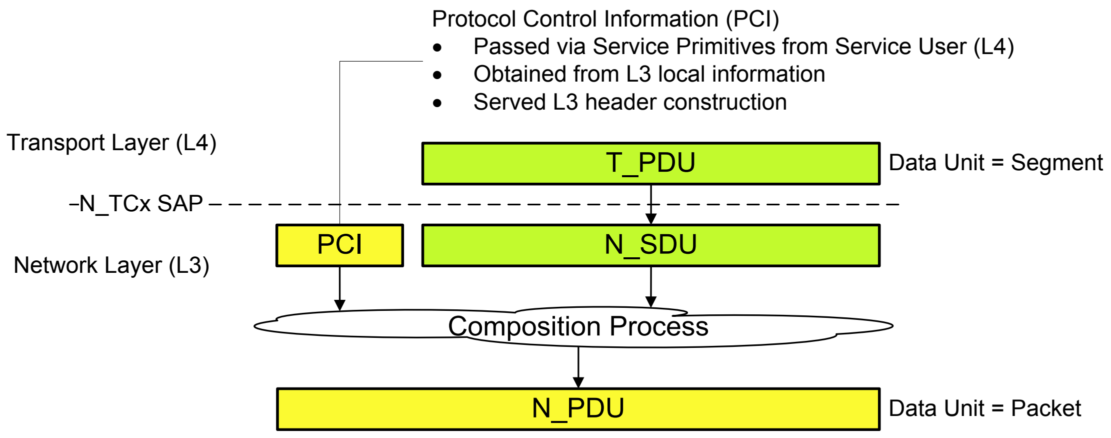

Protocol Control Information (PCI)
Passed via Service Primitives from Service User (L4)
Obtained from L3 local information
Served L3 header construction
Transport Layer (L4)
T_PDU
Data Unit = Segment
N_TCx SAP
PCI
N_SDU
Network Layer (L3)
Composition Process
N_PDU
Data Unit = Packet

Figure 78 Network Layer Composition Process

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

7.9.2 N_PDU Decomposition Feature

This feature is responsible for the decomposition of the N_PDUs. The decomposition feature extracts header information of received Packets and uses additionally layer-specific local state information to form the Protocol Control Information.

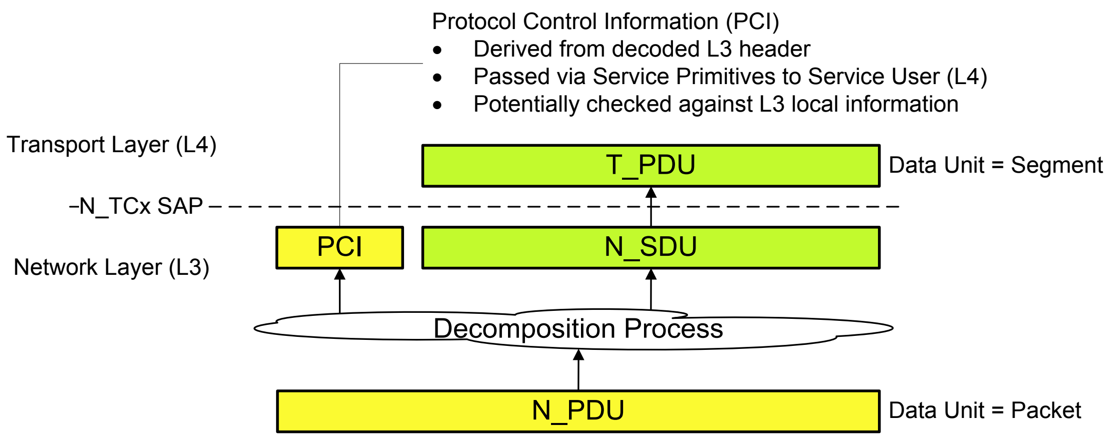

Protocol Control Information (PCI)
• Derived from decoded L3 header
• Passed via Service Primitives to Service User (L4)
• Potentially checked against L3 local information
Transport Layer (L4)
T_PDU
Data Unit = Segment
–N_TCx SAP
PCI
N_SDU
Network Layer (L3)
Decomposition Process
N_PDU
Data Unit = Packet

Figure 79 Network Layer Decomposition Process

The user-data is obtained from the N_SDU field. The user-data and parts of the PCI shall be provided to the Network Service User by means of the appropriate Service Primitive indications.

For the construction of the PCI, the Header Format Analysis feature is used.

7.9.3 Header Format Analysis Feature

The feature shall determine the incoming N_PDU type and shall perform decoding of the DestDeviceID_Enc field.

The N_PDU type is determined by the L3s field and identifies the appropriate Service Primitive indication to call, either N_DATA_SHDR.ind (L3s = 1) or N_DATA_LHDR_TRAP.ind (L3s =0). An N_PDU with L3s=0 is forwarded unmodified to the Layer 3 extension as described in ***Section 7.12***. The fields of an N_PDU with L3s=1 are extracted as described in ***Section 7.8.2***, and the N_PDU is processed as follows:

- If N_DeviceID_valid is set to TRUE, the DeviceIDOffset parameter shall be computed from the DestDeviceID_Enc field specified by the following formula:

  DeviceIDOffset = DestDeviceID_Enc – N_DeviceID

- If N_DeviceID_valid is set to FALSE, the DeviceIDOffset parameter shall be set to '0'.

The resulting value of DeviceIDOffset shall be provided to the Transport Layer via the N_DATA_SHDR.ind primitive.

7.9.4 Discard N_PDU Feature

In case of any occurrences of the situations defined by L3DiscardReasonCode listed in ***Table 57***, the Network Layer shall perform all actions necessary to free up any resources used for the particular affected process.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

7.10 Packet Transmission

A source Device shall transmit N_SDUs in the order in which they arrived at the local TCx N_SAP.

For the transmission of Packets the following operations shall be performed in the order listed below:

- The correct N_PDU type shall be determined according to the rules defined in ***Section 7.8.3***
- The selected N_PDU shall be composed by means of the N_PDU composition feature and in accordance to the encoding rules defined in ***Section 7.8.2***
    - The present header fields shall be computed and shall be set accordingly
    - The Payload field shall be filled with the given user-data

7.11 Packet Reception

Upon the reception of a Packet the following operations shall be performed in the following order:

- The received N_PDU shall be decomposed by means of the N_PDU decomposition feature involving the header format analysis feature and in accordance to ***Section 7.8.2***
    - The type of the N_PDU shall be recovered
    - The present header fields shall be recovered
    - The DeviceIDOffset shall be decoded according to the formula given in ***Section 7.9.3***
    - The user-data shall be obtained from the Payload field
- If N_DeviceID_valid is TRUE, the DestDeviceID_Enc shall be checked against the N_DeviceID to ensure that the Packet arrived at the correct destination. If the DestDeviceID_Enc is greater than or equal to N_DeviceID, the Packet shall be accepted, otherwise it shall be discarded. If N_DeviceID_valid is FALSE, this check is not performed.
- The user-data shall be indicated to the Network Service User by means of the corresponding Service Primitives according to ***Section 7.6.1*** and with accordingly set parameters.

Upon any exception listed in ***Section 7.9.4*** the Discard N_PDU feature shall be performed.

7.12 Long Header Trap

Currently, UniPro provides support for Layer 3 short headers. In future releases, Layer 3 Long Header support will be added. The Long Header Trap is a feature that allows a UniPro implementation to be extended, e.g., in software, to support future Layer 3 Long Header Packets and the features based on them, such as Configuration. The Long Header Trap will be discontinued when Layer 3 Long Header support is added to the UniPro Specification.

The Long Header Trap transmitter forwards Packets from the Layer 3 extension to Layer 2 without modification. A Packet received from a Layer 3 extension shall have the L3s bit set to ‘0’, and shall conform to the Layer 3 Long Header Packet format as defined in future versions of UniPro.

The arbitration process between Short Header Packets and Long Header Packets is unspecified. However, the process shall guarantee Short Header progress.

When the Long Header Trap receiver receives a L2 payload from Layer 2 with the L3s bit set to ‘0’, it shall forward the unmodified L2 payload to the Layer 3 extension.

The flow of incoming Short Header Packets shall not be affected by the Long Header Trap. That is, if Layer 3 waits for the N_DATA_LHDR_TRAP.rsp_L from the Layer 3 extension, any incoming Short Header Packet, i.e., with L3s bit set to ‘1’, shall be processed immediately. Any incoming Long Header Packet, i.e., with L3s bit set to ‘0’, shall be either dropped or stored outside the Short Header Packet processing path. When a Long Header Packet is dropped, the N_LM_DISCARD.ind primitive is issued with L3DiscardCode equal to LHDR_TRAP_PACKET_DROPPING.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

7.13 Network Layer and Data Link Layer Interaction

The generation of a N_DATA_SHDR.req results in the generation of a corresponding Data Link Layer-specific DL_DATA.req by the accordant Network Layer TCx Entity, in case:

- The Packet processing for transmission has been performed successfully
- The Service Primitives are called correctly, i.e. the parameters are in the defined valid range

The receipt of a Data Link Layer-specific DL_DATA.ind associated with delivery of a data unit shall cause the involved Network Layer TCx Entity, i.e. TC0 or TC1, to generate an N_DATA_SHDR.ind, in case:

- The Packet processing for reception has been performed successfully
- The discard N_PDU feature has not been performed

7.14 Management Information Base and Protocol Constants

Table 64 and ***Table 66*** show the Network Layer Attributes.

Table 65 lists the Protocol Constants used by the protocol specification and defines valid values for the Attributes.

All Network Layer Attributes should be readable.

At reset, a settable Attribute shall be reset to its corresponding static value, if one exists, or to its Reset Value otherwise. See ***Section 9*** for more details.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Table 64 Network Layer (gettable, settable) Attributes

Attribute	Attribute ID	Description	Type	Unit	Valid Attribute Value(s)	Mandatory Value(s)	Reset Value	Notes
N_DeviceID	0x3000	Local DeviceID used for checking the DestDeviceID_Enc field in the Network Layer Header (see ***Section 7.9.3*** and ***Section 7.9.4*** for more details). This Attribute is retained during hibernation.	Integer	–	0 to N_MaxDeviceID		0	–
N_DeviceID_valid	0x3001	Specifies whether N_DeviceID Attribute value is valid, and turns on the checking of the DestDeviceID_Enc field in the Network Layer Header (see ***Section 7.9.3*** and ***Section 7.9.4*** for more details). This Attribute is retained during hibernation.	Boolean	–	TRUE=1 FALSE=0		FALSE	–

Table 65 Network Layer Protocol Constants

Attribute	Description	Type	Unit	Value	Notes
N_MaxPacketSize	Maximum Packet Size (PDU size)	Integer	Byte	N_MTU + N_MaxHdrSize	–
N_MaxHdrSize	Maximum L3 header size	Integer	Byte	1	–
N_MTU	Network Layer Maximum Transfer Unit	Integer	Byte	T_MaxSegmentSize	–
N_MaxDeviceIDOffset	Maximum DeviceID offset	Integer	–	63	–
N_MaxDeviceID	Maximum DeviceID	Integer	–	127	–

Table 66 Network Layer (gettable, static) Attributes

Attribute	Attribute ID	Description	Type	Unit	Valid Attribute Value(s)
N_TC0TxMaxSDUSize	0x3020	Maximum transmit payload (SDU) size per Packet for TC0.	Integer	Byte	1 to N_MTU
N_TC1TxMaxSDUSize	0x3021	Maximum transmit payload (SDU) size per Packet for TC1.	Integer	Byte	1 to N_MTU

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

8 Transport Layer (L4)

This section defines the service provided by the Transport Layer, its protocol behavior and the Transport Protocol Data Units (T_PDUs) used by the Transport Layer (TL). It aims to provide as much implementation flexibility as possible given the restrictions needed to achieve a high degree of interoperability.

Figure 80 shows the SAP model used for the Transport Layer. The Transport service to the Application Layer is provided by means of the Transport Layer Connection-oriented Service Access Point (T_CO_SAP). The Transport Layer in turn relies on the service provided by the N_TCx_SAP. The Transport Layer Management SAP (T_LM_SAP) is provided to the Device Management Entity (DME) for configuration and control purposes.

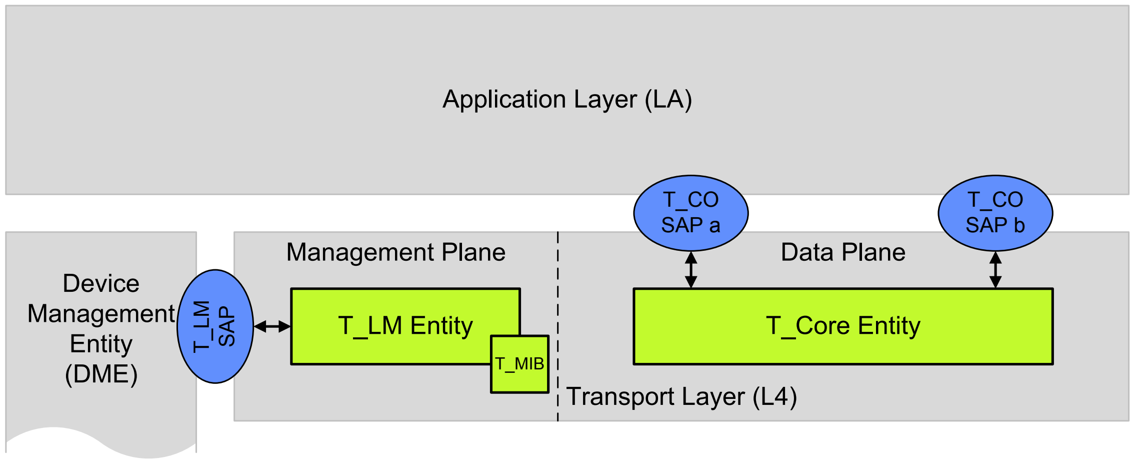


Figure 80 Transport Layer SAP Model

8.1 Transport Layer Service Functionality and Features

The Transport Layer shall provide the following features and services for the transfer of user-data between Transport Layer users.

- Addressing
- Segmentation and Reassembly
- Segment Composition and Segment Decomposition
- Segment format recognition
- Connections
- End-to-End flow-control
- Error handling

The Transport Layer shall not limit the user-data with regard to its content and its representation.

8.2 Transport Layer Addressing

This section defines the abstract syntax and semantics of the Transport address.

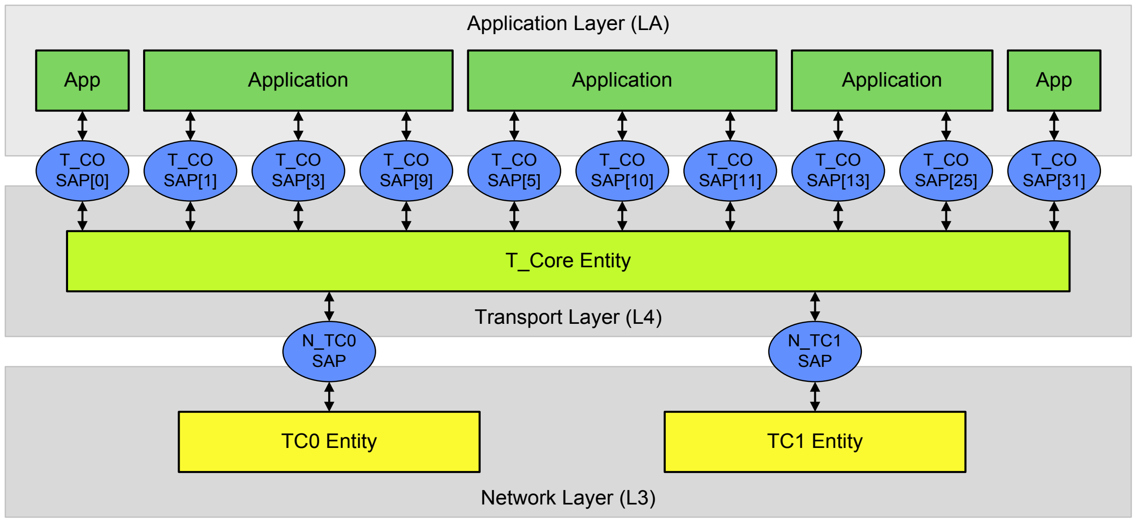


Figure 81 T_SAPs, Transport Address and CPortID

A Transport address identifies one Transport Layer Connection-oriented SAP (T_CO_SAP[x]) by means of a unique address (refer to ***Figure 81***) which is referred to as CPortID. The CPortID number ('x' in the SAP notation) shall range from 0 to T_MaxCPortID (defined in ***Table 93***).

Note:

*CPortID values above 31 should only be used if necessary. This is because each destination Device's use of CPortID values above 31 will reduce the number of Network Layer Device addresses: if the highest CPortID value is greater or equal to 32 and less than x*32-1 (with x = 1 to 64), the number of available Network Layer Device addresses is reduced by x-1. For example, if a Device uses CPortID values up to 100 (implying x = 4), this will reduce the maximum number of addressable Devices by three units. The corresponding address translation process is described in **Section 8.10.4**.*

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

8.3 Connection-Oriented (CO) Data Communication

Connections in UniPro are used to distinguish data streams to, for example, qualify Connections with various properties such as QoS information.

Data is passed between the Transport Layer and its users by means of Transport Service Data Units (T_SDUs) qualified by certain parameters via SAPs by means of Service Primitives, defined later on.

The Transport Service provides and maintains Connections for the data transmission between two CPorts, respectively between two CPorts Users (see ***Figure 82***).

A Connection describes a bidirectional, logical communication channel between two CPorts. Each Connection owns a set of certain properties qualifying different service characteristics, e.g. Traffic Class and End-to-End Flow-control, and parameters identifying the peer CPort, e.g. the T_PeerDeviceID and T_PeerCPortID. A CPort shall not be able to support multiple simultaneous Connections. A CPort is not bound to a single Traffic Class, but the association is made on a per-connection basis.

The transport is performed by means of Transport Protocol Data Units (T_PDUs), which are also referred to as Segments.

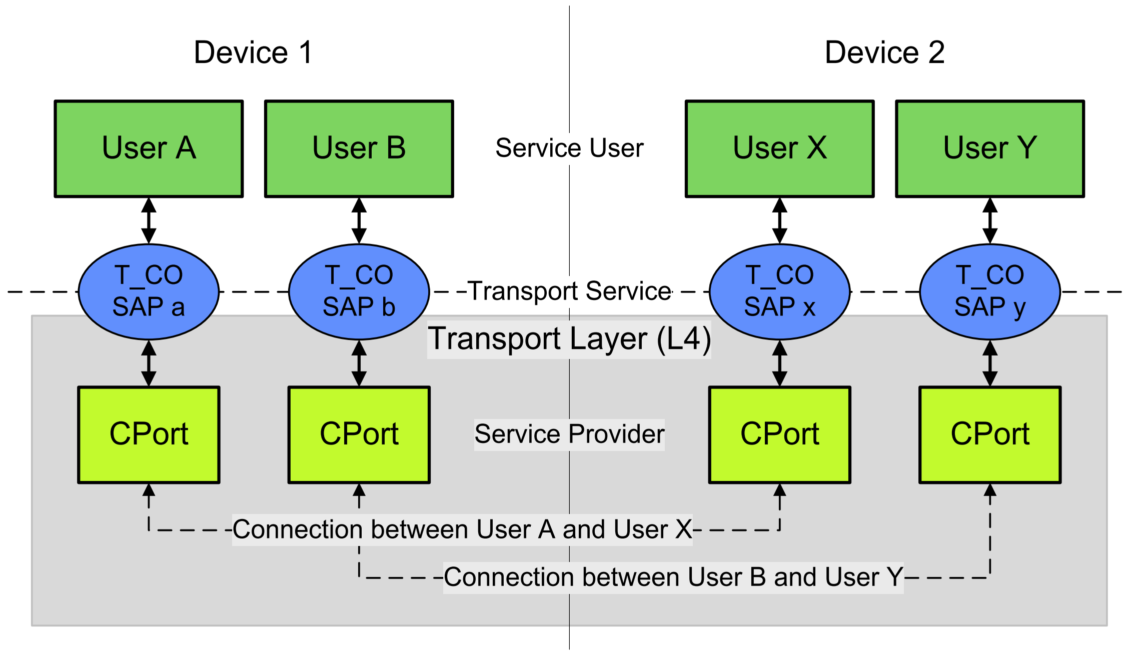


Figure 82 Concept of a Transport-Level Connection

8.4 Segmentation and Reassembly

In UniPro, T_SDUs, also referred to as Messages, are used by the CPort Users to communicate. The Transport Layer supports T_SDUs of arbitrary length. Due to payload length, multiple T_PDUs might be needed to carry one T_SDU from peer-to-peer over a Connection. For that reason, segmenting of T_SDUs into T_PDUs and reassembling of T_PDUs into T_SDUs shall be supported. For controlling the reassembly process at the receiving side, an End-Of-Message (EOM) indication bit is carried within every T_PDU.

The maximum payload a Segment can carry is referred to as Transport Layer Maximum Transfer Unit (T_MTU), and is defined in ***Table 93***. The maximum Segment size (T_MaxSegmentSize) is also defined in ***Table 93***. These values represent the largest sizes defined for the protocol. A UniPro stack may support a smaller Segment payload size than T_MTU.

The maximum transmit payload size per Segment for TC0 supported by a UniPro stack is held in T_TC0TxMaxSDUSize. Similarly, T_TC1TxMaxSDUSize holds the maximum transmit payload size per Segment for TC1. The values of T_TC0TxMaxSDUSize and T_TC1TxMaxSDUSize shall not exceed (N_TC0TxMaxSDUSize – T_MaxHdrSize) and (N_TC1TxMaxSDUSize – T_MaxHdrSize), respectively. See ***Table 66*** and ***Table 93*** for the Attribute and constant values.

It is assumed by the Transport Layer that Segments, which are carried by means of L3 Packets, belonging to the same Traffic Class and going to the same Device are not re-ordered.

8.5 End-to-End Flow Control (E2E FC)

The E2E FC feature is intended to regulate the flow of T_PDUs independently of the flow control of other layers.

It is assumed by the Transport Layer that the underlying services provide reliable link-level data transmission.

The Transport Layer also assumes that the underlying Data Link Layer provides link-level flow control. An additional configurable End-to-End Flow Control (E2E FC) mechanism shall be provided and enabled upon reset, but can be disabled after reset on a per-Connection basis. This E2E Flow Control is intended to prevent interactions between Connections due to congestion caused by shared L2 resources, also known as Head of Line blocking (HOL). The E2E FC can also prevent certain types of deadlock situations.

Note:

> *The E2E FC feature described in this chapter is normative optional. If a UniPro implementation chooses to support E2E FC, it shall implement it as described in this chapter.*

8.6 Services Assumed From the Network Layer

The Network Layer provides its services to the Transport Layer via the N_SAP (refer to ***Figure 81***). The N_SAP is defined in ***Section 7.6***.

The Transport Layer requires the following services and properties provided by the Network and lower layers:

- Reliable data transmission
- In-order delivery of data belonging to the same Traffic Class

Data transmission and reception by means of Segments are supported by the exchange of requests and indications between the Transport Layer and the Network Layer. These parameters and indications allow the Transport Layer to control, and be informed of, the data transmission and reception.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

8.7 Limitations and Compatibility Issues

Conceptually, the Transport Layer supports T_SDUs (Messages) of arbitrary length.

In the current version of UniPro a standardized Connection management protocol is not provided. Instead, Connections shall be statically opened by setting CPort Attributes, either during initial configuration using the DME or at design-time. The DME-based Connection configuration is described in ***Section 8.10.3***.

8.8 T_CO_SAP

The T_CO_SAPs provide the services for basic data transmission.

The data transfer Service Primitives shall be used to transfer user-data called Transport Service Data Units (T_SDUs) or Messages, in either direction or in both directions simultaneously on a Connection. The Transport Service shall preserve the content, the order and the boundaries of the T_SDUs passed through the T_CO_SAP. The T_CO_SAP offers primitives to transfer Message Fragments. There is no guarantee that the received Message Fragment boundaries are the same as the transmitted Message Fragment boundaries. The user may transmit a zero size Message. The user shall not transmit a zero size Message Fragment if the EOM is set to FALSE. The user may transmit a zero size Message Fragment if the EOM is set to TRUE. Note that internally, the Transport Layer may transmit a T_PDU with zero length payload to transmit a FCT token or an EOM. A receiver shall be able to correctly process zero length payload Segments.

Even though the T_CO_SAP allows the transfer of Message Fragments, a CPort User may still communicate using full messages by providing the whole Message to the T_CO_DATA.req and setting the EOM to TRUE.

Data transmission is supported by flow-control services governing the exchange of Segments, either locally or in an end-to-end manner.

Prior to any data-transmission between two users, a Connection needs to be opened as described in ***Section 8.10.3***.

T_CO_SAP primitives shall not be called when the UniPro stack is in Hibernate. While the UniPro stack is in Hibernate, T_CO_SAP primitives have undefined behavior.

8.8.1 Data Transfer Primitives

The primitives covered in this section are listed in ***Table 67***.

Table 67 T_CO_SAP Primitives

Name	Request	Indication	Local Response	Response	Local Confirm	Confirm
T_CO_DATA	***8.8.1.1***	***8.8.1.3***	***8.8.1.4***	–	***8.8.1.2***	–
T_CO_FLOWCONTROL	***8.8.1.5***	–	–	–	***8.8.1.6***	–

Note:

The identifier "CO" in the primitives describes the Connection-oriented service type.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Table 68 lists the parameters that appear in the T_CO_SAP primitives.

Table 68 T_CO_SAP Primitive Parameters

Name	Type	Valid Range	Value	Description
Message Fragment	byte string	Any byte string		Zero or more bytes specifying the data passing the T_CO_SAP before transmission or after reception
FragmentStatus	Enum	FRAGMENT_CORRECT	0	Indicates a correct Message Fragment
		FRAGMENT_CORRUPT	1	Indicates that the Message Fragment is incomplete due to activation of CSD or CSV
		FRAGMENT_AFTER_GAP	2	Indicates that data dropping due to CSD or CSV preceded the Message Fragment
		FRAGMENT_CORRUPT_AFTER_GAP	3	Indicates that the Message Fragment is incomplete due to activation of CSD or CSV and data dropping preceded the Message Fragment
EOM	Bool	FALSE	0	Indicates no end of Message at the end of the Message Fragment
		TRUE	1	Indicates that the end of the Message Fragment is a Message boundary
SOM	Bool	FALSE	0	Indicates the Message Fragment is not the first Fragment of Message
		TRUE	1	Indicates the Message Fragment is the first Fragment of Message
Credits	Integer	1 to T_MaxE2EFCCredits		Specifies the amount of data in bytes the particular Transport Layer SAP (CPort) is permitted to pass
CreditsAccepted	Integer	0 to T_MaxE2EFCCredits		Returns the number of accepted Credits
L4CPortResultCode	Enum	SUCCESS	0	Indicates the result of the request
		NO_CONNECTION	1	
		CREDITS_EXCEEDED	2	
		NO_CPORT	3	
		NO_PEER_TC	6	
		INVALID_FRAGMENT	7	

L4CPortResultCode shall be set to NO_CPORT when an Application attempts to access a nonexistent CPort using the T_CO_SAP. This might be the case when a UniPro stack multiplexes several CPorts over a single signal interface as described in ***Annex D***.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

8.8.1.1 T_CO_DATA.req

This primitive requests the transfer of a Message Fragment to a peer CPort via the associated Connection.

The semantics of this primitive are:

T_CO_DATA.req(MessageFragment, EOM)

When Generated

The CPort User generates a T_CO_DATA.req primitive to send a Message Fragment via the associated Connection. The Message Fragment may have any positive size. A Message Fragment may have a zero size only if EOM is set to TRUE. The EOM parameter shall be set to TRUE when the Message Fragment is the last Fragment of the Message, and shall be set to FALSE otherwise.

Effect on Receipt

The instructed CPort shall transmit the given Message Fragment on the associated Connection.

8.8.1.2 T_CO_DATA.cnf_L

This primitive reports the result of a Message Fragment transfer attempt to a specified recipient.

The semantics of this primitive are:

T_CO_DATA.cnf_L(L4CPortResultCode)

When Generated

This primitive shall be generated by the CPort as a result of a T_CO_DATA.req. The confirmation indicates the success or failure of an attempted transmission. An attempted transmission is interpreted as the successful or unsuccessful service usage of the underlying layer.

When the T_CO_DATA.req succeeds, the returned L4CPortResultCode shall be SUCCESS. All other L4CPortResultCode values should indicate a failure, in which case the Message Fragment shall not be transmitted. In the absence of a Connection, the L4CPortResultCode shall be NO_CONNECTION. If the Message Fragment provided with T_CO_DATA.req has a size of zero and the EOM flag is set to FALSE, the L4CPortResultCode shall be INVALID_FRAGMENT. In case the E2E FC is enabled and the Message Fragment length exceeds the T_PeerBufferSpace, the L4CPortResultCode shall be CREDITS_EXCEEDED. In case the corresponding CPort does not exist, the L4CPortResultCode shall be set to NO_CPORT.

If the peer Device does not implement the TCx Traffic Class, the value of the L4CPortResultCode shall be NO_PEER_TC. The Transport Layer learns about the peer Device not implementing the TCx Traffic Class when it receives N_DATA_SHDR.cnf_L or N_DATA_LHDR_TRAP.cnf_L with L3ResultCode set to NO_PEER_TC.

Effect on Receipt

The Transport user is notified of the results of the attempt by the CPort to transfer a Message Fragment provided with the most recent T_CO_DATA.req. On a given T_CO_SAP, following the emission of a T_CO_DATA.req primitive and prior to the reception of a T_CO_DATA.cnf_L primitive, the CPort User shall not emit a new T_CO_DATA.req primitive.

The CPort User may emit a new T_CO_DATA.req primitive immediately following a reset or after the reception of the T_CO_DATA.cnf_L primitive corresponding to a previously emitted T_CO_DATA.req primitive on this T_CO_SAP.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

8.8.1.3 T_CO_DATA.ind

This primitive reports the reception of a complete or incomplete Message Fragment. No sender is identified by the indication; instead this has to be determined by the associated Connection.

The semantics of this primitive are:

T_CO_DATA.ind(MessageFragment, EOM, SOM, FragmentStatus)

When Generated

The T_CO_DATA.ind primitive shall be generated by the affected CPort to deliver to the Transport user Message Fragment resulting from data received at that CPort by means of a Connection. The Message Fragments delivered using T_CO_DATA.ind may be different in size compared to the transmitted Message Fragments or the received Segments. The EOM shall be set to TRUE if the Message Fragment is the last Fragment of the Message, and shall be set to FALSE otherwise. The SOM shall be set to TRUE if the Message Fragment is the first Fragment of the Message, and shall be set to FALSE otherwise.

If an incomplete Message Fragment is delivered, but data was not dropped between the last received and the current Message Fragments, FragmentStatus shall be set to FRAGMENT_CORRUPT. If a complete Message Fragment is delivered, and data has been dropped between the last received and the current Message Fragment, FragmentStatus shall be set to FRAGMENT_AFTER_GAP. If an incomplete Message Fragment is delivered, and data was dropped between the last received and the current Message Fragment, FragmentStatus shall be set to FRAGMENT_CORRUPT_AFTER_GAP. If none of the above errors has occurred, FragmentStatus shall be set to FRAGMENT_CORRECT.

Typically, FragmentStatus indicates an error when parts of the data inside or before Message Fragment were discarded due to the CPort Safety Valve feature (***Section 8.10.11***) or by the Controlled Segment Dropping feature (***Section 8.10.10.2***).

Effect on Receipt

Upon reception of a T_CO_DATA.ind primitive, the CPort User shall consume the Message Fragment.

8.8.1.4 T_CO_DATA.rsp_L

This primitive informs the CPort that the CPort User is ready to accept a new T_CO_DATA.ind primitive.

The semantics of this primitive are:

T_CO_DATA.rsp_L( )

When Generated

This primitive shall be generated when the CPort User can accept and process a new T_PDU.

Effect on Receipt

On a given T_CO_SAP, following the emission of a T_CO_DATA.ind primitive and prior to the reception of a T_CO_DATA.rsp_L primitive, the CPort shall not emit a new T_CO_DATA.ind primitive. The CPort may emit a T_CO_DATA.ind primitive on a SAP only just after reset, or after the reception of a T_CO_DATA.rsp_L primitive corresponding to a previously emitted T_CO_DATA.ind on that SAP.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

8.8.1.5 T_CO_FLOWCONTROL.req

This primitive is the service request primitive for the Connection flow control service. Refer to ***Section 8.10.10***.

The semantics of this primitive are:

T_CO_FLOWCONTROL.req(Credits)

When Generated

The CPort User sends a T_CO_FLOWCONTROL.req primitive to the CPort to request control of the flow of user-data associated with a Connection from the Transport Layer. When T_CPortFlags.E2EFC is '1', or T_CPortFlags.E2EFC is '0' and T_CPortFlags.CSD_n is '0', the CPort User shall issue the T_CO_FLOWCONTROL.req to signal its ability to consume more data. When T_CPortFlags.E2EFC is '0' and T_CPortFlags.CSD_n is '1', the use of T_CO_FLOWCONTROL.req is not necessary.

Effect on Receipt

When receiving the T_CO_FLOWCONTROL.req primitive, the CPort shall increment T_LocalBufferSpace with the value of Credits supplied by T_CO_FLOWCONTROL.req. The maximum value of T_LocalBufferSpace is limited to T_MaxE2EFCCredits. If by adding Credits T_LocalBufferSpace would exceed T_MaxE2EFCCredits, the amount of accepted credits (CreditsAccepted) shall be set to (T_MaxE2EFCCredits – T_LocalBufferSpace) when issuing the T_CO_FLOWCONTROL.cnf_L primitive, and T_LocalBufferSpace shall be set to T_MaxE2EFCCredits.

In case the end-to-end flow control feature is enabled for the particular Connection, i.e. T_CPortFlags.E2EFC equals ‘1’, the T_CO_FLOWCONTROL.req primitive shall also transfer Flow Control Tokens (FCTs) to the peer CPort using the associated Connection by means of a DT T_PDU (see ***Section 8.10.1***).

One FCT corresponds to a specific number of Credits in bytes, which is defined per Connection and per direction (T_TxTokenValue, T_RxTokenValue; see ***Table 92***). T_TxTokenValue is used for calculating the amount of FCTs to be sent to the connected peer CPort. T_RxTokenValue is used to calculate the equivalent value of the received FCT in bytes to increment the T_PeerBufferSpace counter for the transmitting peer CPort.

Since the primitive can accept more Credits than one Token is worth, while only one FCT can be signaled to the peer CPort at a time, the number of outstanding Credits to be sent (T_CreditsToSend, see ***Table 92***) shall be tracked. When E2E FC is enabled, the T_CreditsToSend counter shall be incremented by the value in the parameter Credits, limited to the maximum number of E2E FC Credits (T_MaxE2EFCCredits, see ***Table 93***), and shall be decremented by T_TxTokenValue each time a FCT is sent.

The request is locally completed with a T_CO_FLOWCONTROL.cnf_L primitive. This does not necessarily mean that the transmission of FCTs to a peer CPort was successfully completed.

Additional Comments

Control of the flow of data on a Connection is independent of control of the flow on other Connections. The CPort User shall ensure that the Credits given represent less than or equal the amount of data the user can accept and that the amount of Credits shall not exceed T_MaxE2EFCCredits.

8.8.1.6 T_CO_FLOWCONTROL.cnf_L

This primitive is the service confirm primitive for the Connection flow control service.

The semantics of this primitive are:

T_CO_FLOWCONTROL.cnf_L(CreditsAccepted, L4CPortResultCode)

When Generated

This primitive shall be generated by the CPort as a result of a T_CO_FLOWCONTROL.req. It confirms the completion of a T_CO_FLOWCONTROL.req and indicates success or failure of the local operation, i.e. the given Credits are recognized and the sending of corresponding FCTs is scheduled.

When the T_CO_FLOWCONTROL.req succeeds, the returned L4CPortResultCode shall be SUCCESS and the value of CreditsAccepted should carry the value of the corresponding T_CO_FLOWCONTROL.req. All other values of L4CPortResultCode should indicate a failure. In case the maximum number of Credits is exceeded, the L4CPortResultCode shall be CREDITS_EXCEEDED and the value of CreditsAccepted shall be set to the number of Credits that have been accepted by the Transport Layer. In the absence of a Connection, the L4CPortResultCode of the confirm Service Primitive shall be NO_CONNECTION.

Effect on Receipt

The Transport user is notified of the results of the service request completion.

8.9 T_LM_SAP

The Transport Layer Management (T_LM) SAP offers three groups of Service Primitives: Configuration primitives, Control primitives and Status primitives. The primitives on the T_LM_SAP are used by the Device Management Entity (DME) to configure and control the layer and receive layer status information.

The Configuration primitives enable access to the Transport Layer Attributes defined in ***Section 8.14.3.6, Section 8.14.4.5, and Section 8.16***.

The Control primitives provide direct control of the Transport Layer. Control primitives are generated by the DME and can occur concurrently.

The Status primitives indicate status information of the Transport Layer. Status primitives are generated by the Transport Layer and can occur concurrently.

8.9.1 Configuration Primitives

The T_LM configuration primitives, GET and SET, are used by the DME to retrieve and store values, respectively, for the Transport Layer Attributes. The invocation of a SET.req primitive may require the Transport Layer to perform certain defined actions.

The GET and SET primitives are represented as requests with associated confirm primitives. These primitives are prefixed by T_LM. The primitives are summarized in ***Table 69***.

Table 69 T_LM_SAP Configuration Primitives

Name	Request	Indication	Local Response	Response	Local Confirm	Confirm
T_LM_GET	***8.9.1.1***	–	–	–	***8.9.1.2***	–
T_LM_SET	***8.9.1.3***	–	–	–	***8.9.1.4***	–

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

The parameters used for these primitives are defined in ***Table 70***.

Table 70 T_LM_SAP Configuration Primitive Parameters

Name	Type	Valid Range	Value	Description
MIBattribute	Integer	MIBattribute values fall between 0x4000 and 0x4FFF The valid MIBattribute values are defined in ***Section 8.14.3.6, Section 8.14.4.5, and Section 8.16***	–	The address of the MIB Attribute
MIBvalue	defined by MIBattribute	As defined in ***Section 8.14.3.6***, ***Section 8.14.4.5***, and ***Section 8.16***	–	The value of the MIB Attribute
SelectorIndex	Integer	0 to T_NumCPorts – 1	–	Indicates the targeted CPort or Test Feature when relevant
SetType	Enum	NORMAL	0	Select whether the actual value (NORMAL) or the Attribute's non-volatile value (STATIC) is addressed.
		STATIC	1	
ConfigResultCode	Enum	SUCCESS	0	Indicates the result of the request
		INVALID_MIB_ATTRIBUTE	1	
		INVALID_MIB_ATTRIBUTE_VALUE	2	
		READ_ONLY_MIB_ATTRIBUTE	3	
		WRITE_ONLY_MIB_ATTRIBUTE	4	
		BAD_INDEX	5	
		LOCKED_MIB_ATTRIBUTE	6	
		BAD_TEST_FEATURE_INDEX	7	

8.9.1.1 T_LM_GET.req

This primitive requests the value of the Attribute identified by MIBattribute and, if relevant, the SelectorIndex. The SelectorIndex shall be interpreted either as a CPort index or a Test Feature index based on the Attribute ID. For Attributes not associated with the SelectorIndex, the SelectorIndex shall be ignored.

The semantics of this primitive are:

T_LM_GET.req(MIBattribute, SelectorIndex)

The primitive parameters are defined in ***Table 70***.

When Generated

This primitive is generated by the DME to obtain the value of the Attribute identified by MIBattribute.

Effect on Receipt

The Transport Layer shall attempt to retrieve the requested Attribute value and respond with T_LM_GET.cnf_L that gives the retrieved value.

8.9.1.2 T_LM_GET.cnf_L

This primitive indicates the success or failure of T_LM_GET.req, and, is successful, returns the Attribute value.

The semantics of this primitive are:

T_LM_GET.cnf_L(ConfigResultCode, MIBvalue)

The primitive parameters are defined in ***Table 70***.

The Transport Layer shall set ConfigResultCode in T_LM_GET.cnf_L to one of the values shown in ***Table 71*** for the condition given in the table. The Transport Layer should not set ConfigResultCode to INVALID_MIB_ATTRIBUTE_VALUE or READ_ONLY_MIB_ATTRIBUTE.

Table 71 T_LM_GET.cnf_L ConfigResultCode Values

ConfigResultCode	Condition
SUCCESS	The request succeeds, i.e. the MIB Attribute indicated by T_LM_GET.req MIBattribute is gettable. The Transport Layer shall set MIBvalue in T_LM_GET.cnf_L to the Attribute value.
INVALID_MIB_ATTRIBUTE	The Attribute indicated by MIBattribute is invalid or not implemented. The value of MIBvalue in T_LM_GET.cnf_L is undefined.
WRITE_ONLY_MIB_ATTRIBUTE	The Attribute indicated by T_LM_GET.req MIBattribute exists, but is not gettable. The value of MIBvalue in T_LM_GET.cnf_L is undefined.
BAD_INDEX	The value of SelectorIndex is greater than, or equal to, T_NumCPorts.
BAD_TEST_FEATURE_INDEX	The value of SelectorIndex is greater than, or equal to, T_NumTestFeatures

When Generated

The Transport Layer shall generate the T_LM_GET.cnf_L primitive in response to the most recent T_LM_GET.req from the DME.

Effect on Receipt

The DME is informed about the result of the operation, and in case ConfigResultCode is SUCCESS, MIBvalue carries the Attribute value. For any other value of ConfigResultCode, MIBvalue is undefined.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

8.9.1.3 T_LM_SET.req

This primitive attempts to set the value (SetType = NORMAL) or the reset value (SetType = STATIC) of the Attribute indicated by MIBattribute to the MIBvalue and, if relevant, the SelectorIndex. The SelectorIndex shall be interpreted either as a CPort index or a Test Feature index based on the Attribute ID. For Attributes not associated with the SelectorIndex, the SelectorIndex shall be ignored.

The semantics of this primitive are:

T_LM_SET.req(SetType, MIBattribute, MIBvalue, SelectorIndex)

The primitive parameters are defined in ***Table 70***.

When Generated

This primitive is generated by the DME to set the value or reset value of the indicated Attribute.

Effect on Receipt

If SetType is set to NORMAL, the Transport Layer shall attempt to set the Attribute addressed by MIBattribute to the MIBvalue. If SetType is set to STATIC, the Transport Layer shall attempt to set the reset value of the Attribute addressed by MIBattribute to the MIBvalue.

The Transport Layer uses T_LM_SET.cnf_L to inform the DME of the result of T_LM_SET.req.

8.9.1.4 T_LM_SET.cnf_L

This primitive reports the results of an attempt to set the value of an Attribute or its reset value.

The semantics of this primitive are:

T_LM_SET.cnf_L(ConfigResultCode)

The primitive parameter is defined in ***Table 70***.

The Transport Layer shall set ConfigResultCode in T_LM_SET.cnf_L to one of the values shown in ***Table 72*** for the conditions given in the table. The T_LM entity should not set ConfigResultCode to WRITE_ONLY_MIB_ATTRIBUTE.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Table 72 T_LM_SET.cnf_L ConfigResultCode Values

ConfigResultCode	Condition
SUCCESS	The request succeeds, i.e. the value or the reset value of the Attribute indicated by MIBattribute and SelectorIndex is set to the value of MIBvalue.
INVALID_MIB_ATTRIBUTE	The MIBattribute value is invalid, or otherwise the Attribute indicated by MIBattribute and SelectorIndex is not implemented, or otherwise If SetType is STATIC, the Attribute indicated by MIBattribute and SelectorIndex does not support setting its reset value.
READ_ONLY_MIB_ATTRIBUTE	The Attribute indicated by MIBattribute and SelectorIndex exists, but is not settable. This ConfigResultCode value shall only be used if SetType is NORMAL.
INVALID_MIB_ATTRIBUTE_VALUE	The Attribute indicated by MIBattribute and SelectorIndex exists and is settable (SetType = NORMAL) or allows its reset value to be set (SetType = STATIC), but the value of MIBvalue is outside of the implemented range, or outside of the valid value range, for the Attribute.
LOCKED_MIB_ATTRIBUTE	The Attribute indicated by MIBattribute and SelectorIndex exists and is settable, but it belongs to a CPort that is in the CONNECTED state. This ConfigResultCode value shall only be used if SetType is NORMAL.
BAD_INDEX	The CPort indicated by T_LM_SET.req SelectorIndex is outside of the range of available CPorts.
BAD_TEST_FEATURE_INDEX	The value of T_LM_GET.req SelectorIndex is greater than, or equal to, T_NumTestFeatures.

When Generated

The Transport Layer shall generate the T_LM_SET.cnf_L primitive in response to the most recent T_LM_SET.req from the DME.

Effect on Receipt

T_LM_SET.cnf_L confirms the success or failure of the T_LM_SET.req at the Transport Layer, and should have no further effects at the DME.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

8.9.2 Control Primitives

The Service Primitives in this section are provided for controlling the Transport Layer.

The primitives covered in this section are listed in ***Table 73***.

Table 73 T_LM_SAP Control Primitives

Name	Request	Indication	Local Response	Response	Local Confirm	Confirm
T_LM_RESET	***8.9.2.1***	–	–	–	***8.9.2.2***	–
T_LM_ENABLE_LAYER	***8.9.2.3***	–	–	–	***8.9.2.4***	–
T_LM_HIBERNATE_ENTER	***8.9.2.5***	–	–	–	***8.9.2.6***	–
T_LM_HIBERNATE_EXIT	***8.9.2.7***	–	–	–	***8.9.2.8***	–
T_LM_PRE_HIBERNATE_EXIT	***8.9.2.9***	–	–	–	***8.9.2.10***	–

Table 74 lists the parameters that appear in the T_LM_SAP control primitives.

Table 74 T_LM_SAP Control Primitive Parameters

Name	Type	Valid Range	Value	Description
ResetLevel	Enum	COLD	0	Defines the reset level of the requested reset
		WARM	1	

8.9.2.1 T_LM_RESET.req

This primitive requests the reset of the Transport Layer.

The semantics of this primitive are:

T_LM_RESET.req(ResetLevel)

When Generated

This primitive is generated by the DME when it needs to reset the Transport Layer. The Transport Layer is reset in combination with resetting of all the UniPro protocol layers as described in the reset procedure.

Effect on Receipt

The Transport Layer sets itself to the reset states and Attribute values, and discards all Segments and/or Messages currently processed. Then, the Transport Layer shall generate T_LM_RESET.cnf_L to the DME. After issuing T_LM_RESET.cnf_L, the Transport Layer shall not react to any Transport Layer SAP primitive other than T_LM_RESET.req and T_LM_ENABLE_LAYER.req

The ResetLevel COLD resets the complete Transport Layer including the statistics and configuration Attributes. The ResetLevel WARM resets the Transport Layer without the statistics.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

8.9.2.2 T_LM_RESET.cnf_L

The T_LM_RESET.cnf_L primitive is used during the UniPro reset procedure (see ***Section 9.11.1***).

The semantics of this primitive are:

T_LM_RESET.cnf_L( )

When Generated

The T_LM_RESET.cnf_L primitive is issued to the DME in response to T_LM_RESET.req to indicate that the Transport Layer came out of reset.

Effect on Receipt

The DME is informed that the Transport Layer came out of reset.

8.9.2.3 T_LM_ENABLE_LAYER.req

The T_LM_ENABLE_LAYER.req primitive is used during the UniPro boot procedure (see ***Section 9.11.2***).

The semantics of this primitive are:

T_LM_ENABLE_LAYER.req( )

When Generated

The T_LM_ENABLE_LAYER.req primitive is part of the UniPro boot procedure (see ***Section 9.11.2***), and is generated by the DME after the Transport Layer came out of reset, and the Network Layer was enabled.

Effect on Receipt

The Transport Layer shall respond with T_LM_ENABLE_LAYER.cnf_L to the DME. After issuing T_LM_ENABLE_LAYER.cnf_L, all Transport Layer functionality and SAP primitives shall be enabled.

8.9.2.4 T_LM_ENABLE_LAYER.cnf_L

The T_LM_ENABLE_LAYER.cnf_L primitive is used during the UniPro boot procedure (see ***Section 9.11.2***) to indicate to the DME that the Transport Layer is enabled.

The semantics of this primitive are:

T_LM_ENABLE_LAYER.cnf_L( )

When Generated

The T_LM_ENABLE_LAYER.cnf_L primitive is issued to the DME in response to T_LM_ENABLE_LAYER.req to indicate that the Transport Layer was enabled.

Effect on Receipt

The DME is informed that the Transport Layer is enabled.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

8.9.2.5 T_LM_HIBERNATE_ENTER.req

The T_LM_HIBERNATE_ENTER.req primitive is used to request the Transport Layer to enter hibernation.

The semantics of this primitive are:

T_LM_HIBERNATE_ENTER.req( )

When Generated

The DME generates the T_LM_HIBERNATE_ENTER.req primitive to request the Transport Layer to enter hibernation.

Effect on Receipt

The Transport Layer shall issue the T_LM_HIBERNATE_ENTER.cnf_L primitive to the DME and then enter hibernation. During hibernation, the Transport Layer should not retain any Attribute, and may lose all its state.

8.9.2.6 T_LM_HIBERNATE_ENTER.cnf_L

The T_LM_HIBERNATE_ENTER.cnf_L primitive is used to indicate to the DME that the Transport Layer is about to hibernate.

The semantics of this primitive are:

T_LM_HIBERNATE_ENTER.cnf_L( )

When Generated

The T_LM_HIBERNATE_ENTER.cnf_L primitive is generated by the Transport Layer in response to a T_LM_HIBERNATE_ENTER.req primitive and it indicates that the Transport Layer is hibernating.

Effect on Receipt

The DME is informed about the Transport Layer entering hibernate.

8.9.2.7 T_LM_HIBERNATE_EXIT.req

The T_LM_HIBERNATE_EXIT.req primitive is used to request the Transport Layer to exit hibernation and return to normal operation.

The semantics of this primitive are:

T_LM_HIBERNATE_EXIT.req( )

When Generated

The DME generates the T_LM_HIBERNATE_EXIT.req primitive to request the Transport Layer to exit hibernation.

Effect on Receipt

The Transport Layer shall go to its reset state. Then the Transport Layer shall set its Attributes to their reset values and then issue the T_LM_HIBERNATE_EXIT.cnf_L primitive to the DME.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

8.9.2.8 T_LM_HIBERNATE_EXIT.cnf_L

The T_LM_HIBERNATE_EXIT.cnf_L primitive is used to indicate to the DME that the Transport Layer has exited hibernation and is operating normally.

The semantics of this primitive are:

T_LM_HIBERNATE_EXIT.cnf_L( )

When Generated

The T_LM_HIBERNATE_EXIT.cnf_L primitive is generated by the Transport Layer in response to a T_LM_HIBERNATE_EXIT.req primitive and it indicates that the Transport Layer has exited hibernation and is operating normally.

Effect on Receipt

The DME is informed about the Transport Layer exiting hibernate.

8.9.2.9 T_LM_PRE_HIBERNATE_EXIT.req

Support for this primitive is optional. However, if an implementation supports this primitive it shall implement it as described in this section.

This primitive is used to request the Transport Layer to stop, or prepare to stop hibernating, and prepare to return to normal operation. Hibernate exit is later requested with the T_LM_HIBERNATE_EXIT.req primitive.

The semantics of this primitive are:

T_LM_PRE_HIBERNATE_EXIT.req( )

When Generated

The DME shall generate this primitive when the Transport Layer needs to prepare to exit Hibernate prior to exiting Hibernate.

Effect on Receipt

The Transport Layer shall prepare to exit Hibernate State.

8.9.2.10 T_LM_PRE_HIBERNATE_EXIT.cnf_L

Support for this primitive is optional. However, if an implementation supports this primitive it shall implement it as described in this section.

This primitive is used to indicate to the DME that the Transport Layer has been prepared to return to normal operation after hibernating.

The semantics of this primitive are:

T_LM_PRE_HIBERNATE_EXIT.cnf_L( )

When Generated

This primitive is generated by the Transport Layer in response to a T_LM_PRE_HIBERNATE_EXIT.req primitive. It indicates that the Transport Layer has been prepared for return to normal operation.

Effect on Receipt

The DME is informed of the completion of all necessary actions required by the Transport Layer to return to normal operation after preparation for exiting Hibernate.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

8.9.3 Status Primitives

The Service Primitives in this section are provided for status provision by the Transport Layer.

The primitives covered in this section are listed in ***Table 75***.

Table 75 T_LM_SAP Status Primitives

Name	Request	Indication	Local Response	Response	Local Confirm	Confirm
T_LM_DISCARD	–	***8.9.3.1***	–	–	–	–

Table 76 lists the parameters that appear in the T_LM_SAP status primitives.

Table 76 T_LM_SAP Status Primitive Parameters

Name	Type	Valid Range	Value	Description
L4DiscardReasonCode	Enum	UNSUPPORTED_HEADER_TYPE	1	Indicates the reason for discarding a T_PDU
		UNKNOWN_CPORTID	2	
		NO_CONNECTION_RX	3	
		CONTROLLED_SEGMENT_DROPPING	4	
		BAD_TC	5	
		E2E_CREDIT_OVERFLOW	6	
		SAFETY_VALVE_DROPPING	7	
		MAX_T_PDU_LENGTH_EXCEEDED	8	

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

8.9.3.1 T_LM_DISCARD.ind

This primitive indicates that the Discard T_PDU feature has been performed (refer to ***Section 8.10.13***) and reports the reason for triggering the operation. The semantics of this primitive are:

T_LM_DISCARD.ind(CPortID, L4DiscardReasonCode)

When T_LM_DISCARD.ind is generated for the CPort given in CPortID, the provided L4DiscardReasonCode shall be set according to the cases summarized in ***Table** 77*, and described in ***Section 8.10.13***.

Table 77 T_LM_DISCARD.ind Reason Codes

L4DiscardReasonCode	Value	Description
UNSUPPORTED_HEADER_TYPE	1	A T_PDU was received whose header contains unsupported values
UNKNOWN_CPORTID	2	The CPort a received T_PDU is targeting does not exist
NO_CONNECTION_RX	3	A user-data carrying T_PDU targeting a CPort is received that has no established Connection, i.e. is not in the CONNECTED state
CONTROLLED_SEGMENT_DROPPING	4	The Segment was dropped at the receiver due to insufficient buffer space This can occur only when End-to-End Flow Control is disabled
BAD_TC	5	A T_PDU was received with a Traffic Class that is different than the T_TrafficClass Attribute of the destination CPort
E2E_CREDIT_OVERFLOW	6	The size of a received Segment is bigger than the space left in the RX buffer, given by the value of T_LocalBufferSpace
SAFETY_VALVE_DROPPING	7	CPort Safety Valve mechanism has discarded data
MAX_T_PDU_LENGTH_EXCEEDED	8	Indicates that a T_PDU with a payload longer than the T_MTU has been received

When Generated

This primitive shall be generated by the T_LM entity as a result of performing the discard T_PDU feature.

Effect on Receipt

The DME is notified of the discard.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

8.10 Structure and Encoding of Transport Protocol Data Units

This section defines the Transport Layer Protocol Data Units (usually referred to as Segments) and shows the header fields and their meaning.

A Segment (T_PDU) shall contain an integral number of bytes greater than zero and shall be limited to T_MaxSegmentSize. This guarantees that any Segment shall always fit in exactly one Packet (N_PDU).

8.10.1 Data T_PDUs

Table 78 lists the Data Transport Protocol Data Unit (DT T_PDU) types that shall be supported.

Table 78 Transport Layer PDU Types

T_PDU Name	Mnemonic	Overall Header Size	Remarks/Limitations	Section
Data T_PDU with short header	DT SH T_PDU	8-bit	Payload is limited to T_MTU	***8.10.1.1***

8.10.1.1 DT SH T_PDU

The current version of UniPro only defines a DT T_PDU with a short header, referred to as DT SH T_PDU and identified by the L4s bit set to '1'. The DT SH T_PDU is used to transfer User data between peer CPorts. DT SH T_PDUs are sent using the N_DATA_SHDR.req primitive.

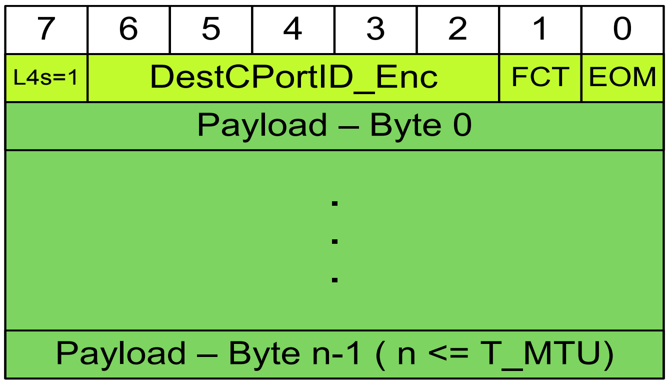


Figure 83 Data T_PDU with Short Header (DT SH T_PDU)

The natural T_PDU granularity is bytes, i.e. any integral number of data bytes in the range from 0 to T_MTU can be transferred.

8.10.2 T_PDU Fields

Table 79 describes the T_PDU fields.

Table 79 T_PDU Header Fields

Field Name	Field Description	Field Size	Remarks/Limitations
L4s	L4 Header type field	1-bit	Identifies the used T_PDU type; this field is fixed to ‘1’ indicating a DT SH T_PDU
DestCPortID_Enc	Least significant bits of destination CPortID	5-bit	Contains the five least significant bits of the addressed CPort
FCT	Flow Control Token	1-bit	Signals a new Flow Control Token
EOM	End Of Message	1-bit	Identifies the last Segment of a Message
Payload	Message Fragment	0 to T_MTU bytes	Carries a Message chunk with a maximum size that depends on the configured maximum payload size of the used Traffic Class x (T_TC[x]TxMaxSDUSize) There is no relation enforced between the Segment Payload and the TX and RX Message Fragments as transferred through T_CO_DATA SAP

8.10.2.1 L4 Header Type Field (L4s)

The L4s header field serves as a header type bit. In the current version of UniPro, a Segment created as a result of T_CO_DATA.req Service Primitive shall have the L4s header bit set to ‘1’. Upon reception of a Segment with the L4s bit set to ‘0’, the T_PDU shall be discarded (see ***Section 8.10.13***) and a notification according to ***Section 8.9.3.1*** shall be given.

Table 80 shows the supported settings.

Table 80 Settings of the L4s Field

Field Name	Size	Value	Meaning	Remarks
L4s	1-bit	0	Reserved	Not allowed; no corresponding Service Primitive supported
		1	Short Header Data T_PDU (DT SH T_PDU) is used	Caused by the use of the T_CO_DATA primitive

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

8.10.2.2 DestCPortID_Enc

The DestCPortID_Enc field shall contain the five least significant bits of the CPortID of the targeted CPort. The destination CPort value is a fixed property of a Connection and all T_PDUs sent within a Connection shall have the same value for DestCPortID_Enc.

In various cases a 5-bit CPortID may be enough. If a Device supports more than thirty-two CPorts, only the least significant bits of the destination CPortID shall be encoded into the DestCPortID_Enc header field according to the definitions in ***Section 8.10.4***.

Table 81 shows the supported settings.

Table 81 Settings of the DestCPortID_Enc Field

Field Name	Size	Value	Meaning	Remarks
DestCPortID_Enc	5-bit	0 to 31	Least significant bits of PeerCPortID	Least Significant bits and remaining most significant bits are determined by the formulas given in ***Section 8.10.4***

8.10.2.3 Flow Control Token (FCT)

The FCT field is used to signal a Flow Control Token to support the E2E Flow-control service (refer to ***Section 8.10.10.1***).

Table 82 Settings of the FCT Field

Field Name	Size	Value	Meaning	Remarks
FCT	1-bit	0	No Flow Control Token received/sent	–
		1	Flow Control Token is carried by T_PDU	–

This bit is set to ‘1’ by the sending CPort in case the end-to-end control feature is enabled and the CPort has to signal a Flow Control Token to the receiving peer CPort. The conditions when a FCT is signaled are specified in ***Section 8.10.10.1***.

8.10.2.4 End of Message (EOM)

The EOM field is used to mark the final Segment of a Message. Note that short Messages may be carried by a single Segment. When the EOM bit is set to ‘1’, the first payload byte of any subsequent Segment shall be considered the first byte of a new Message.

Table 83 Settings of the EOM Field

Field Name	Size	Value	Meaning	Remarks
EOM	1-bit	0	The End of Message has not been reached	–
		1	The last byte of the most recent received or transmitted Fragment contains the last byte of the potentially segmented Message	–

As already introduced in ***Section 8.4***, the segmentation process takes care of splitting Messages so that each resulting Segment does not exceed the T_MTU size. The Transport Layer sender ensures that the reassembly process on the receiving side merges the received Segments into exactly the same Message by asserting the End of Message (EOM) flag to indicate the end of a Message. The order of the intermediate Segments is preserved by means of the required Network properties, i.e. in-order delivery.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Upon reception of a Segment with the EOM bit set to '1', the Transport Layer shall notify the receiving Application after the last Segment has been processed by means of a T_CO_DATA.ind. The Transport Layer may also notify the receiving Application when parts of a Message are available, e.g., one or more Segments with the EOM bit set to '0' have been received.

8.10.2.5 Payload

The payload field carries the Message given by the CPort User and is potentially segmented by the CPort. The payload may carry any value.

Table 84 Settings of the Payload Field

Field	Size	Value	Meaning	Remarks
Payload	0 to T_MTU bytes	Any byte string	A complete Message or a portion of a Message	–

Protocol Features

8.10.3 Connection Management Feature

Prior to any data transmission, Connections shall be opened to ensure that the CPorts defining a Connection are correctly configured. When the Connection is not needed anymore, the Connection may be closed to allow the involved CPorts to be used by a different Connection

As already stated, UniPro does not yet support a standardized Connection management protocol, i.e. Connection opening and closing via protocol data units (PDUs) across the Link. In the current Specification, Connection opening is supported either based on a non-volatile Attribute reset values or using DME-based local and peer Attribute setting.

A Connection is opened between two peer CPorts. The Connection parameter Attributes listed in ***Table 92*** shall be set during Connection opening. The Attributes are set via the DME, using the DME_SET and DME_PEER_SET primitives. The Connection parameter Attributes shall be set according to the following rules:

- The T_PeerDeviceID Attribute of each CPort shall be set to the N_DeviceID of the peer CPort.
- The T_PeerCPortID Attribute of each CPort shall be set to the CPortId of the peer CPort
- The T_TrafficClass Attribute shall be set to the same value for both CPorts.
- The T_ProtocolID Attribute shall be set to the same value for both CPorts.
- The T_CPortFlags.E2EFC shall be set to the same value for both CPorts.
- If E2E FC is enabled (T_CPortFlags.E2EFC = '1'), the T_TxTokenValue Attribute of each CPort and the T_RxTokenValue Attribute of the peer CPort shall be set to the same value.
- The T_CPortMode Attribute for each CPort shall be set to CPORT_APPLICATION.
- If E2E FC is enabled (T_CPortFlags.E2EFC = '1') or the E2E FC is disabled and CSD is enabled (T_CPortFlags.E2EFC = '0'and T_CPortFlags.CSD_n = '0'), each of the two CPorts shall set the T_PeerBufferSpace Attribute of the local CPort to the same value as T_LocalBufferSpace Attribute of the peer CPort. These values represent the amount of data that the CPort where T_PeerBufferSpace is set can initially transmit, and, hence, its peer CPort can receive.
- The T_CreditsToSend Attribute for each CPort shall be set to 0.
- The T_ConnectionState Attribute for each CPort shall be set to CONNECTED.
- The T_ConnectionState Attribute shall be the last Attribute to be set at each CPort.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

8.10.4 Address Translation Feature

The Address translation feature provides the means to handle CPortIDs with more than 5-bits to enable addressing Devices with more than thirty-two CPorts. The remaining most significant bits of the T_PeerCPortID are translated to form the DeviceIDOffset, which is passed to the Network Layer, where, together with the T_PeerDeviceID, is used to compute the DestDeviceID_Enc field in the Network Layer Header. For received Segments the DestCPortID_Enc, which is extracted from the header, and the DeviceIDOffset, which is provided by the Network Layer, shall be translated to determine the targeted CPort.

For the transmission the destination CPortID and destination DeviceIDOffset shall be determined as follows:

DestCPortID_Enc = T_PeerCPortID & 0x1F

DeviceIDOffset = T_PeerCPortID >> 5

The T_PeerDeviceID parameter shall be passed unchanged to the Network Layer:

DestPeerDeviceID = T_PeerDeviceID

For the reception the targeted CPortID shall be determined as follows:

CPortID = DestCPortID_Enc + (DeviceIDOffset << 5)

8.10.5 Segmentation Feature

Segmenting is the process of splitting Messages, or Message Fragments, into one or more ordered data chunks that are then composed into Segments on the transmitting side (refer to ***Section 8.10.7***). Message boundaries are preserved as provided by the user of the Transport Layer, whereas Message Fragment boundaries, if any, may change.

It shall be ensured that T_SDUs, i.e. Messages, are sent in sequence, i.e. a new segmenting process shall only take place after a previous one has finished. The EOM parameter of a DT T_PDU indicates whether or not there are subsequent DT T_PDUs in the sequence carrying parts of the T_SDU.

The Segment payload may have any length, but shall not exceed T_TCxTxMaxSDUSize, where TCx corresponds to the Segment traffic class. If the Message, or Message Fragment, length is greater than T_TCxTxMaxSDUSize, the segmentation scheme should create Segments of T_TCxTxMaxSDUSize length except when special conditions occur, such as the last Segment of a Message, or when the Segment is sent due to a Flow Control Token transmitted request. If the Message, or Message Fragment, length is less than, or equal to, T_TCxTxMaxSDUSize, the segmentation scheme shall create only one Segment.

The two scenarios for segmentation on the TX side are shown in ***Figure 84***.

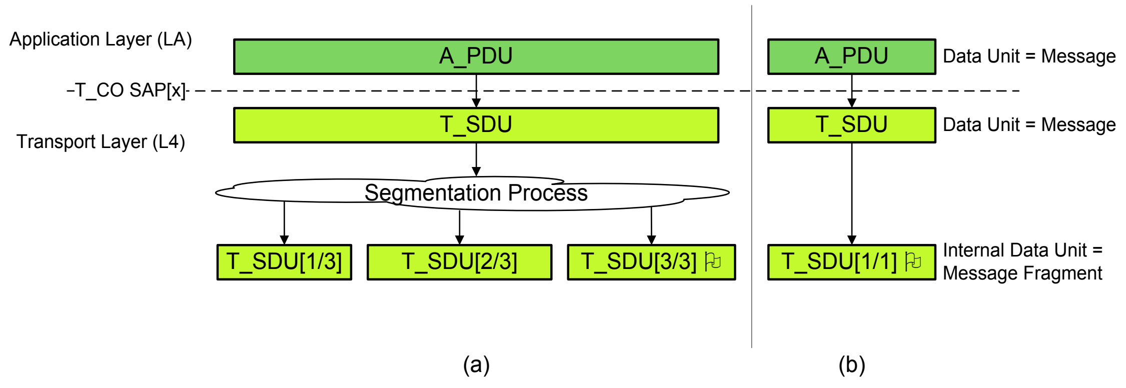


Figure 84 Transport Layer Segmentation Process

In ***Figure 84***, the Application hands over a Message (an A_PDU that then becomes a T_SDU) to the Transport Layer. The Transport Layer performs the segmenting process and the Segment containing the last portion of a Message shall be marked with EOM set to '1', which is visualized in the figure by a flag symbol. The segmenting process is invoked on a given Message when the Message is larger than the payload a Segment can carry, as exemplified in ***Figure 84a***. In contrast, in ***Figure 84b*** the original structure is retained since the T_SDU fits entirely into a single T_PDU and only the EOM bit is set to '1'. The segmenting state is maintained for each Connection.

In exceptional cases, mainly encountered during error recovery, it might be necessary to signal an End of Message even after the last valid portion of the Message has been sent and there is no more data left to transmit. In this case, a Segment without any payload may be sent with the EOM bit set to '1'. The first payload byte of any subsequent Segment shall be considered the first byte of a new Message.

Note that the preceding description does not imply that an implementation of the Transport Layer performs segmenting by buffering entire Messages so that they can be split into multiple Segments, it is merely a conceptual model. An implementation of the Transport Layer may accept Message Fragments or any other sized units of data from the Application Layer and thus avoid Message-sized buffers that are owned by the Transport Layer. The Transport Layer is however responsible for ensuring that a Segment payload never exceeds the T_MTU limit.

8.10.6 Reassembly Feature

After the transmission to the destination the reassembly process takes place on the receiving side. This is the exact counterpart of the segmentation process and is depicted in ***Figure 85a***.

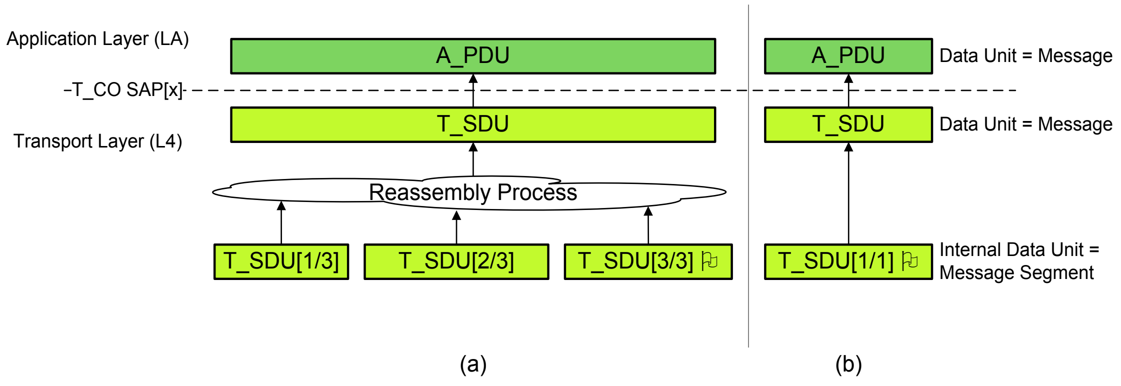


Figure 85 Transport Layer Reassembly Process

On reception of the first Segment that is determined either by reception of the very first Segment or by the first Segment following a completed reassembling process, the reassembly process starts and is continued as long as no further data T_PDUs with EOM bit set to '1' are received. Whenever an EOM is received, the reassembling process ends and the receiving CPort User shall be notified by the CPort about that occurrence by setting the EOM parameter to TRUE when calling the T_CO_DATA.ind Service Primitive. The T_CO_DATA.ind Service Primitive offers the flexibility to provide either entire Messages or Message Fragments to the Application. As stated in ***Section 8.8.1.3***, no relationship is specified between the received Message Fragment size and the transmitted Message Fragment size or the received Segment size. If the Segment contains a complete T_SDU the reassembly process is not required as shown in ***Figure 85b***. The reassembling state shall be maintained for each Connection.

Note that this description does not imply that an implementation of the Transport Layer performs reassembly by buffering multiple Segment payloads until an entire Message is available; it is merely a conceptual model. An implementation of the Transport Layer may hand off Message Fragments, or any other sized units of data, to the Application Layer and thus avoid Message-sized buffers that are owned by the Transport Layer.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

8.10.7 T_PDU Composition Feature

This feature is responsible for the composition of Transport Protocol Data Units (T_PDUs). As depicted in ***Figure 86***, this feature constructs and adds a L4 header to the provided T_SDU to form a T_PDU. The header consists of different fields specified in ***Section 8.10.2***. For the construction of the T_PDU additional control information is needed, called Protocol Control Information (PCI). The PCI is obtained from local layer-specific state information and parameters given by means of Service Primitive requests.

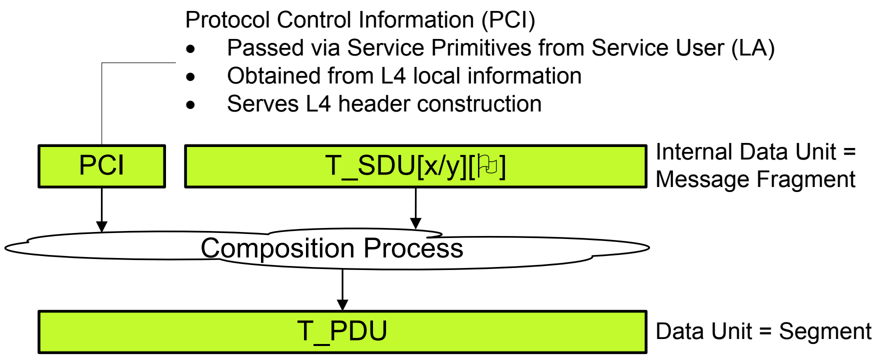

Protocol Control Information (PCI)
• Passed via Service Primitives from Service User (LA)
• Obtained from L4 local information
• Serves L4 header construction
PCI
T_SDU[x/y][🏳]
Internal Data Unit = Message Fragment
Composition Process
T_PDU
Data Unit = Segment

Figure 86 Transport Layer Composition Process

In the Transport Layer, the composition process is performed on complete or portions of T_SDUs (expressed by T_SDU[x/y] in ***Figure 86***).

The current segmenting state shall set the EOM parameter in the accordant data T_PDU.

8.10.8 T_PDU Decomposition Feature

This feature is responsible for the decomposition of the T_PDUs. The decomposition feature extracts header information of received Packets and uses additionally layer-specific local state information to form the Protocol Control Information.

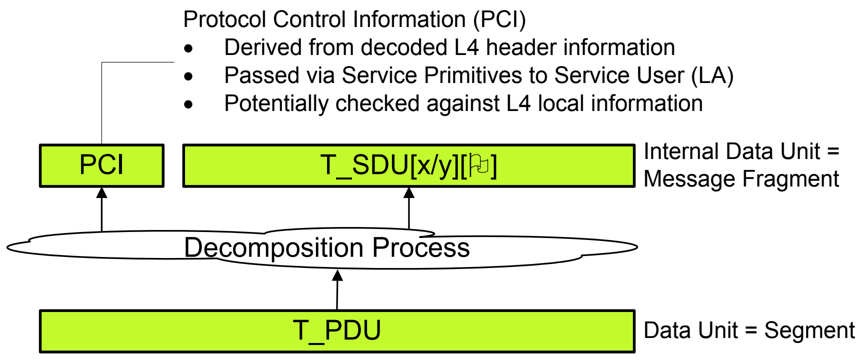

Protocol Control Information (PCI)
• Derived from decoded L4 header information
• Passed via Service Primitives to Service User (LA)
• Potentially checked against L4 local information
PCI
T_SDU[x/y][🏳]
Internal Data Unit = Message Fragment
Decomposition Process
T_PDU
Data Unit = Segment

Figure 87 Transport Layer Decomposition Process

The user-data is obtained from the T_SDU field. The user-data and parts of the PCI shall be provided to the Transport Service User by means of the appropriate Service Primitive indications.

For the construction of the PCI the header format analysis feature is used.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

8.10.9 Header Format Analysis Feature

The feature determines the incoming T_PDU type and extracts the T_PDU header fields present in the correspondent T_PDU structure. In addition it is identified whether or not a received T_PDU targets an existing CPort with an associated established Connection.

The T_PDU type is determined by the L4s field and identifies the T_PDU header structure.

In the current version of UniPro the L4s field shall be set to ‘1’. In the case where a T_PDU is received with the L4s bit set to '0', the T_PDU is discarded.

Depending on the determined T_PDU structure, the header fields are extracted and the actual values are provided either to the corresponding subsequent protocol features or to the Service User by means of Service Primitive indications or confirmations. The fields to be extracted are specified in ***Section 8.10.2***.

The destination CPortID shall be determined by the Address translation protocol feature (***Section 8.10.4***). First, it shall be checked whether the CPort exists. If not, the discard T_PDU feature (***Section 8.10.13)*** shall be performed. After that, it shall be checked whether a Connection exists for that CPort, i.e. T_ConnectionState of the particular CPort must be CONNECTED. If not, the discard T_PDU feature shall be performed.

If T_ConnectionState is CONNECTED, the FCT shall be examined and if this parameter is set to ‘1’ this value shall be signaled to the E2E Flow Control protocol feature specified in ***Section 8.10.10.1***. If T_ConnectionState is not CONNECTED, FCT shall not be processed. In this situation consistency between the T_PeerBufferSpace Attribute of the local CPort and the T_LocalBufferSpace of the peer CPort can no longer be guaranteed. Application should re-initialize following steps in ***Section 8.10.3***, before re-opening the connection.

If T_ConnectionState is CONNECTED, the EOM shall be extracted and the content shall be signaled to the Reassembling protocol feature. If T_ConnectionState is not CONNECTED, EOM shall not be processed.

8.10.10 Explicit Flow Control Features

The UniPro L4 Explicit Flow Control is a feature used to regulate the flow of T_PDUs between two CPorts on one Connection. A Connection between a pair of CPorts can be configured either with the End-to-End Flow Control (E2E FC) feature enabled or with the E2E FC feature disabled.

E2E FC ensures the transmitting CPort knows how much buffer space at the receiving CPort is currently available for its T_PDUs. The transmitting CPort uses E2E FC to prevent overflowing the receiving CPort buffer (see ***Section 8.10.10.1***). If a CPort provides support for E2E FC, it shall default to this mode after reset.

The E2E FC feature provides an effective means to prevent Head of Line (HOL) blocking and potential deadlock situations. Any Connection-oriented T_PDU arriving at the destination Device should be able to progress to a buffer that is reserved for that Connection. A CPort should use E2E FC for all Connections.

Alternatively, E2E FC may be disabled, allowing the transmitting CPort to send data without knowing how much buffer space at the receiving CPort is currently available for its T_PDUs. To prevent congestion when the receiving CPort buffer is full, incoming data Segments should be discarded (dropped) using the Controlled Segment Dropping (CSD) feature (see ***Section 8.10.10.2***).

Note:

> *The E2EFC and CSD features described in this section are normative optional. If a UniPro implementation chooses to support E2EFC and CSD, it shall implement it as described in this section.*

When the E2E FC feature is disabled, there is no guarantee that CPort-specific buffer space is available. The CPort User shall ensure that at any time a Message or Message Fragment indicated by the CPort is immediately consumed. This requirement serves to prevent HOL blocking and resulting congestion or deadlocks. The CSD feature is provided to achieve this goal. Although CSD helps ensure system-level integrity, using CSD leads to data corruption of the received data stream. This corruption is signaled to the Application.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

A receiving CPort shall also support the CPort Safety Valve (CSV) feature (see ***Section 8.10.11***). The CSV feature protects system-level integrity by avoiding HOL blocking. CSV discards one or more Segments when an Application stops accepting data.

CSV works independently of the E2E FC or CSD. E2E FC and CSD prevent Application RX buffer overflow, while CSV protects against an Application stopping receiving data, even when Application RX buffer space is known to be available for the data supplied by the CPort.

8.10.10.1 End-to-End Flow Control Feature

The point-to-point (P2P) reliability provided by the Data Link Layer guarantees that every Segment injected into the Network is correctly and reliably delivered to the destination. This does not automatically provide E2E reliability as there is a possibility to lose data at the destination, due to lack of resources for processing incoming data. Use of E2E FC ensures that the transmitting CPort does not send more data than the Transport user at the receiving CPort is capable of accepting and provides E2E reliability by that means.

T_CPortFlags, T_TxTokenValue and T_RxTokenValue are associated with a particular CPort and in turn apply to the single Connection this CPort is involved in. These Attributes shall be set during Connection setup. T_CPortFlags consists of 3 bits (E2EFC, CSD_n and CSV_n). T_CPortFlags.E2EFC shall indicate whether E2E FC is enabled ('1') or not ('0'). T_CPortFlags.CSD_n and T_CPortFlags.CSV_n are specified in ***Section 8.10.10.2*** and ***Section 8.10.11***, respectively. T_RxTokenValue shall determine the value of a Token received, measured in bytes. T_TxTokenValue shall determine the value of a Token transmitted, also measured in bytes. T_TxTokenValue shall be equal to T_RxTokenValue of the connected peer CPort and vice versa. T_RxTokenValue and T_TxTokenValue shall not exceed the size of the receiver buffer as this would prevent the receiver from sending a token to the transmitter.

T_PeerBufferSpace and T_CreditsToSend shall track the current values related to E2E FC.

T_PeerBufferSpace holds the actual number of bytes a transmitter is allowed to send. It shall be incremented by T_RxTokenValue upon any received Flow Control Token.

A Token is signaled by the FCT header field of a DT T_PDU. Whether a Token was received shall be determined and signaled by the header format analysis feature. The CPort shall transmit Segments if and only if the contained payload size is smaller or equal to the T_PeerBufferSpace value. The T_PeerBufferSpace counter shall be decremented by the amount of data sent.

T_CreditsToSend identifies how many Credits are to be sent to the peer CPort by means of FCTs. This counter shall be incremented with the parameter Credits provided with the T_CO_FLOWCONTROL.req Service Primitive and shall be decremented by T_TxTokenValue whenever a DT T_PDU with FCT header field set to ‘1’ has been transmitted.

The amount of T_PeerBufferSpace shall not exceed the amount of data the Transport user at the peer CPort can consume.

The following sections and the subsequent Message Sequence Charts (MSCs) exemplify the E2E FC procedure. The MSCs provided are not meant to be exhaustive.

Table 85 E2E Flow Control Steps

Step	Description
1	The receiving CPort User should indicate that it can process new portions of data by invoking the T_CO_FLOWCONTROL.req Service Primitive with the appropriate parameter value set, which causes incrementing the T_CreditsToSend counter by the value confirmed by T_CO_FLOWCONTROL.cnf_L.
2	In case the T_CreditsToSend counter value is greater than or equal to T_TxTokenValue, the receiving CPort shall either: Trigger the transmission of one or more DT T_PDUs with FCT set to '1' to the peer CPort if no data transfer is scheduled by the transmitter and decrement the T_CreditsToSend counter by T_TxTokenValue per FCT sent or Set the FCT field to one ('1') for the next, and potentially subsequent, DT T_PDUs scheduled by the transmitter and decrement the T_CreditsToSend by T_TxTokenValue per FCT sent Step 2 shall be repeated until the T_CreditsToSend counter value is less than T_TxTokenValue.
3	Upon reception of a DT T_PDU with FCT header field set to '1', the transmitting CPort shall increment the T_PeerBufferSpace counter by T_RxTokenValue.
4	In case T_PeerBufferSpace is greater than zero, the sending CPort shall send as much data as indicated by T_PeerBufferSpace. The CPort shall ensure that any Segment transmission can be completed without running out of credits. The amount of data sent shall be subtracted from T_PeerBufferSpace.

Note:

Step 1 to step 3 may be performed at any time in parallel to the execution of step 4.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

In ***Figure 88*** the principal usage of E2E FC and a subsequent data transmission is outlined. Note the CPortID parameters are omitted in the figure to enhance readability.

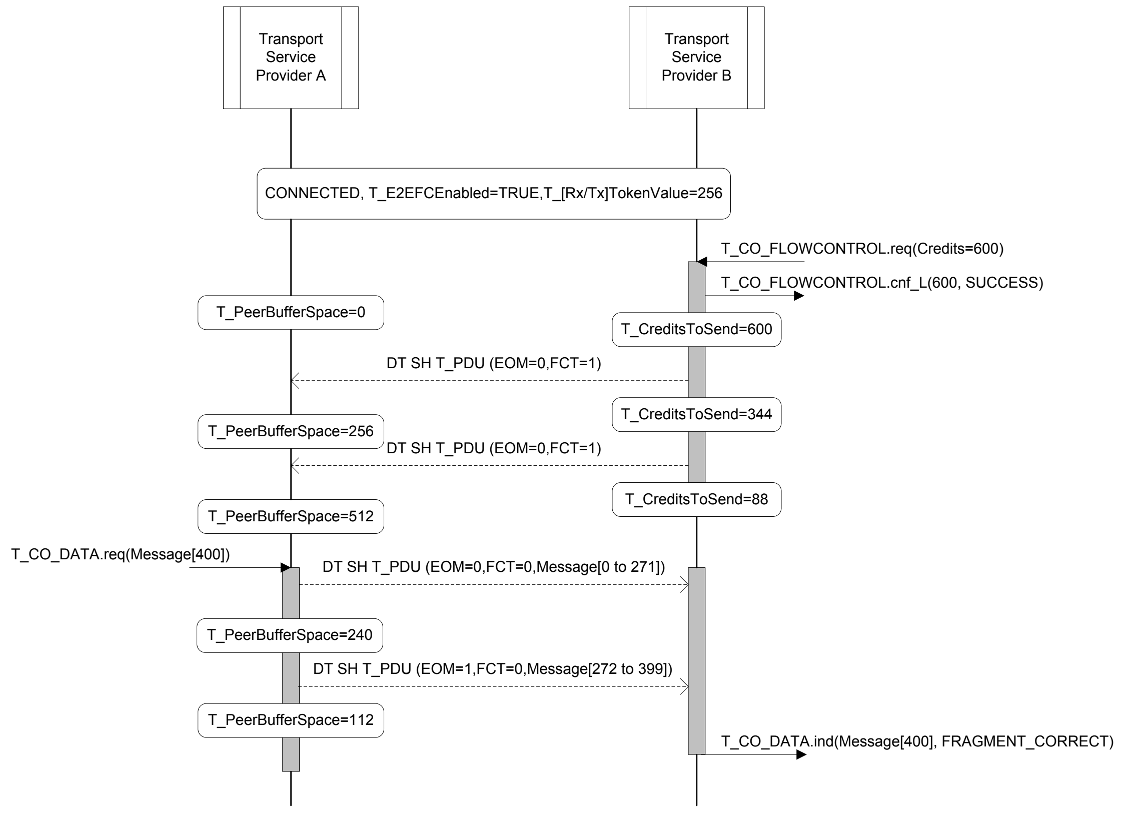

Transport Service Provider A
Transport Service Provider B
CONNECTED, T_E2EFCEnabled=TRUE,T_[Rx/Tx]TokenValue=256
T_CO_FLOWCONTROL.req(Credits=600)
T_CO_FLOWCONTROL.cnf_L(600, SUCCESS)
T_PeerBufferSpace=0
T_CreditsToSend=600
DT SH T_PDU (EOM=0,FCT=1)
T_CreditsToSend=344
T_PeerBufferSpace=256
DT SH T_PDU (EOM=0,FCT=1)
T_CreditsToSend=88
T_PeerBufferSpace=512
T_CO_DATA.req(Message[400])
DT SH T_PDU (EOM=0,FCT=0,Message[0 to 271])
T_PeerBufferSpace=240
DT SH T_PDU (EOM=1,FCT=0,Message[272 to 399])
T_PeerBufferSpace=112
T_CO_DATA.ind(Message[400], FRAGMENT_CORRECT)

Figure 88 Data Transmission using E2E FC

Figure 89 depicts a data transmission procedure where the sending side experiences a situation where the remaining credits are not enough to continue with the transmission of the next scheduled Segment. The sending Transport Layer interrupts the process for this particular CPort and Connection until new credits are given from the receiving side. Note the CPortID parameters are omitted in the figure to enhance readability.

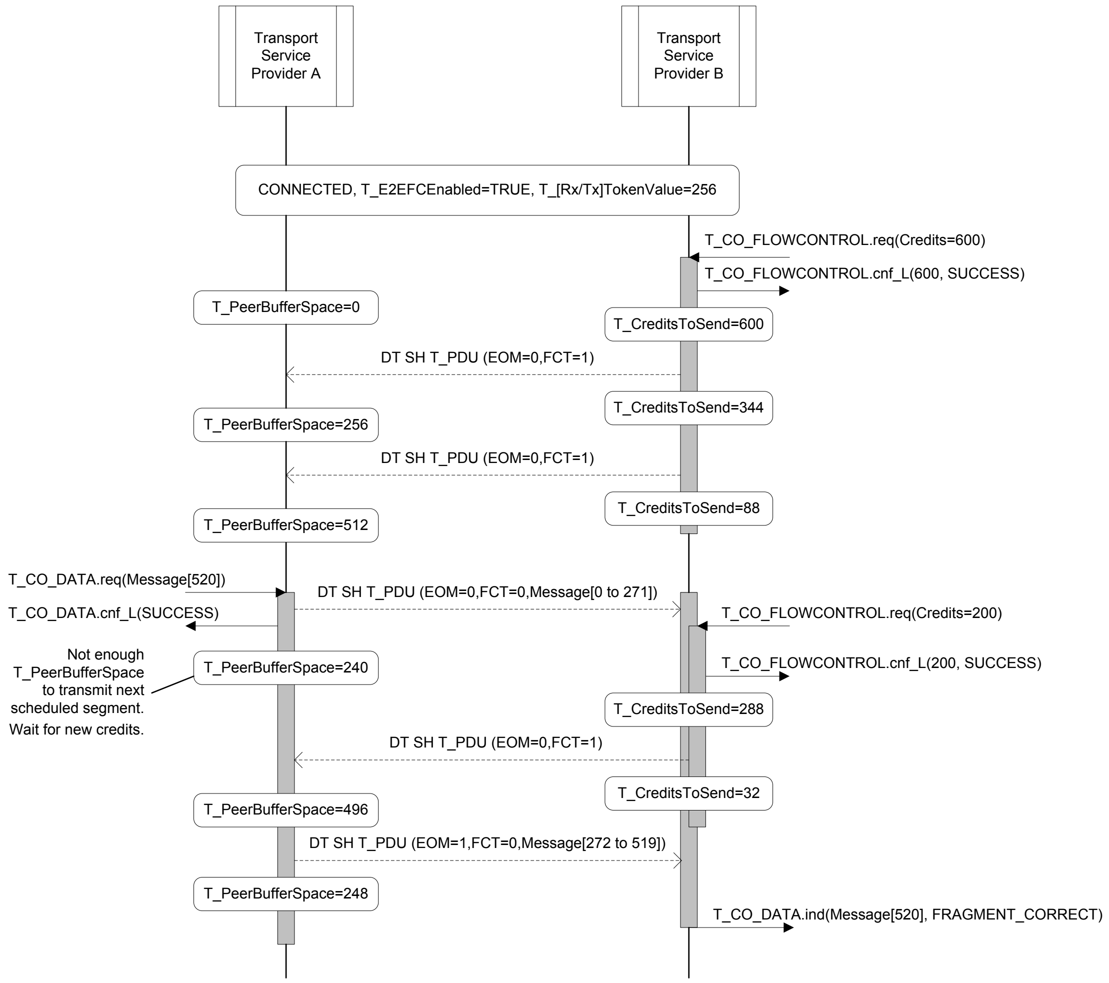

Transport Service Provider A
Transport Service Provider B
CONNECTED, T_E2EFCEnabled=TRUE, T_[Rx/Tx]TokenValue=256
T_CO_FLOWCONTROL.req(Credits=600)
T_CO_FLOWCONTROL.cnf_L(600, SUCCESS)
T_PeerBufferSpace=0
T_CreditsToSend=600
DT SH T_PDU (EOM=0,FCT=1)
T_CreditsToSend=344
T_PeerBufferSpace=256
DT SH T_PDU (EOM=0,FCT=1)
T_CreditsToSend=88
T_PeerBufferSpace=512
T_CO_DATA.req(Message[520])
DT SH T_PDU (EOM=0,FCT=0,Message[0 to 271])
T_CO_DATA.cnf_L(SUCCESS)
T_CO_FLOWCONTROL.req(Credits=200)
Not enough T_PeerBufferSpace to transmit next scheduled segment. Wait for new credits.
T_PeerBufferSpace=240
T_CO_FLOWCONTROL.cnf_L(200, SUCCESS)
T_CreditsToSend=288
DT SH T_PDU (EOM=0,FCT=1)
T_CreditsToSend=32
T_PeerBufferSpace=496
DT SH T_PDU (EOM=1,FCT=0,Message[272 to 519])
T_PeerBufferSpace=248
T_CO_DATA.ind(Message[520], FRAGMENT_CORRECT)

Figure 89 Interrupted Data Transmission Using E2E FC

Note: the E2E FC procedure does not need to acknowledge FCT signals, and FCT signals do not contain synchronization information, which is possible due to P2P reliability of the underlying Network.

8.10.10.2 Controlled Segment Dropping Feature

In case the E2E FC is disabled (T_CPortFlags.E2EFC = '0'), T_CPortFlags.CSD_n shall indicate whether Controlled Segment Dropping (CSD) is enabled ('0') or not ('1'). The Transport Layer does not use CSD on a connection with E2E FC enabled. As a result, when T_CPortFlags.E2EFC is '1', T_CPortFlags.CSD_n shall be ignored.

When CSD is enabled, the CPort User uses the. T_CO_FLOWCONTROL.req Service Primitive to indicate the RX buffer space associated to the CPort.

CSD support may be omitted for cases where E2E FC cannot be disabled.

When CSD is enabled, T_LocalBufferSpace shall hold the actual number of bytes a receiver is allowed to receive and forward to the CPort User. T_LocalBufferSpace shall be incremented by the value of the Credits

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

parameter upon any invocation of the T_CO_FLOWCONTROL.req Service Primitive and decremented by the amount of data received and forwarded to the receiving CPort User.

If CSD is disabled, the CPort shall do no checks on the Segment payload size.

If CSD is enabled, the CPort shall process and forward the received Segment payload only if the T_LocalBufferSpace value is greater or equal than the size of the payload of the T_PDU currently indicated by the underlying Network service. Otherwise, the received Segment payload shall be discarded after ensuring that the T_PDU header is processed to be able to decode and indicate a potentially signaled EOM.

The Reassembly process shall be informed in case a Segment payload was discarded to be able to determine whether a Message could be delivered completely or incompletely.

The receiving User of a CSD-enabled CPort indicates that it can process new portions of data by invoking the T_CO_FLOWCONTROL.req Service Primitive with the appropriate parameter value set, which causes incrementing the T_LocalBufferSpace counter with that value.

In case T_LocalBufferSpace is greater than zero, the receiving CSD-enabled CPort is entitled to receive as much payload data as indicated by T_LocalBufferSpace, provided that it is ensured that any Segment received can be completed in order to forward it to the receiving CPort User without running out of credits. Otherwise, the payload of the current Segment is discarded. The amount of data received is subtracted from T_LocalBufferSpace.

The MSC in ***Figure 90*** shows the situation where the payload of all incoming Segments are discarded due to a lack of Credits. Loss of Segment payload is marked with a black circle at the end of an arrow. CPortID parameters are omitted in the figure to enhance readability.

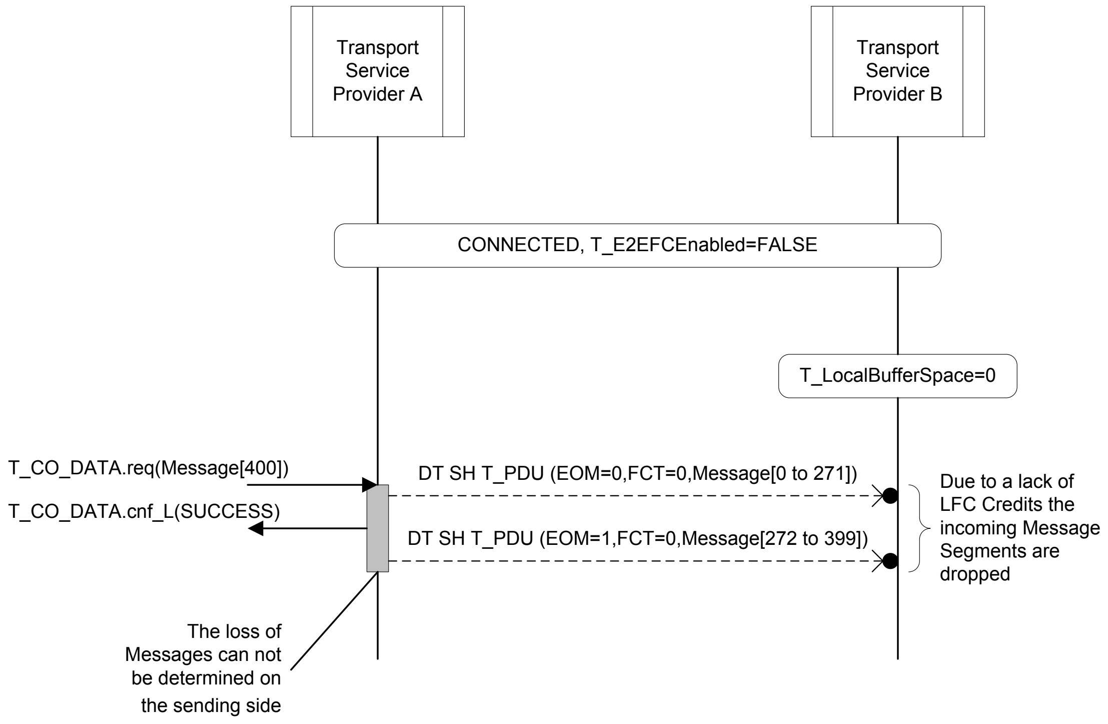

Transport Service Provider A
Transport Service Provider B
CONNECTED, T_E2EFCEnabled=FALSE
T_LocalBufferSpace=0
T_CO_DATA.req(Message[400])
T_CO_DATA.cnf_L(SUCCESS)
DT SH T_PDU (EOM=0,FCT=0,Message[0 to 271])
DT SH T_PDU (EOM=1,FCT=0,Message[272 to 399])
Due to a lack of LFC Credits the incoming Message Segments are dropped
The loss of Messages can not be determined on the sending side

Figure 90 Discard of Segment Payload Due to Lack of Credits

Figure 91 illustrates the data transmission procedure using Controlled Segment Dropping to allow reception of incoming Segment payload. CPortID parameters are omitted in the figure to enhance readability.

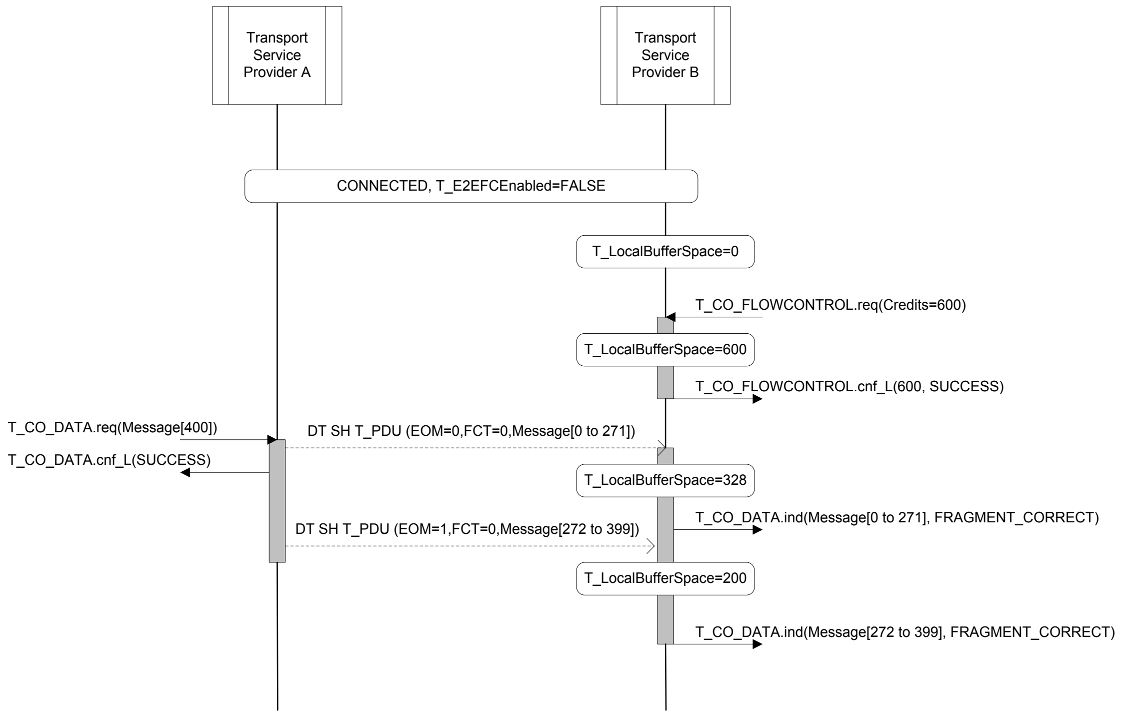

Transport Service Provider A
Transport Service Provider B
CONNECTED, T_E2EFCEnabled=FALSE
T_LocalBufferSpace=0
T_CO_FLOWCONTROL.req(Credits=600)
T_LocalBufferSpace=600
T_CO_FLOWCONTROL.cnf_L(600, SUCCESS)
T_CO_DATA.req(Message[400])
DT SH T_PDU (EOM=0,FCT=0,Message[0 to 271])
T_CO_DATA.cnf_L(SUCCESS)
T_LocalBufferSpace=328
T_CO_DATA.ind(Message[0 to 271], FRAGMENT_CORRECT)
DT SH T_PDU (EOM=1,FCT=0,Message[272 to 399])
T_LocalBufferSpace=200
T_CO_DATA.ind(Message[272 to 399], FRAGMENT_CORRECT)

Figure 91 Successful Data Transmission With Controlled Segment Dropping

The MSC depicted in ***Figure 92*** exemplifies a situation where, due to interim lack of credits, parts of the overall Message are lost. According to the Service Primitive definitions and the reassembling feature this is signaled to the receiving CPort User by tagging the delivered Message with FRAGMENT_AFTER_GAP. Loss of Segment payload is marked with a black circle at the end of an arrow. CPortID parameters are omitted in the figure to enhance readability.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

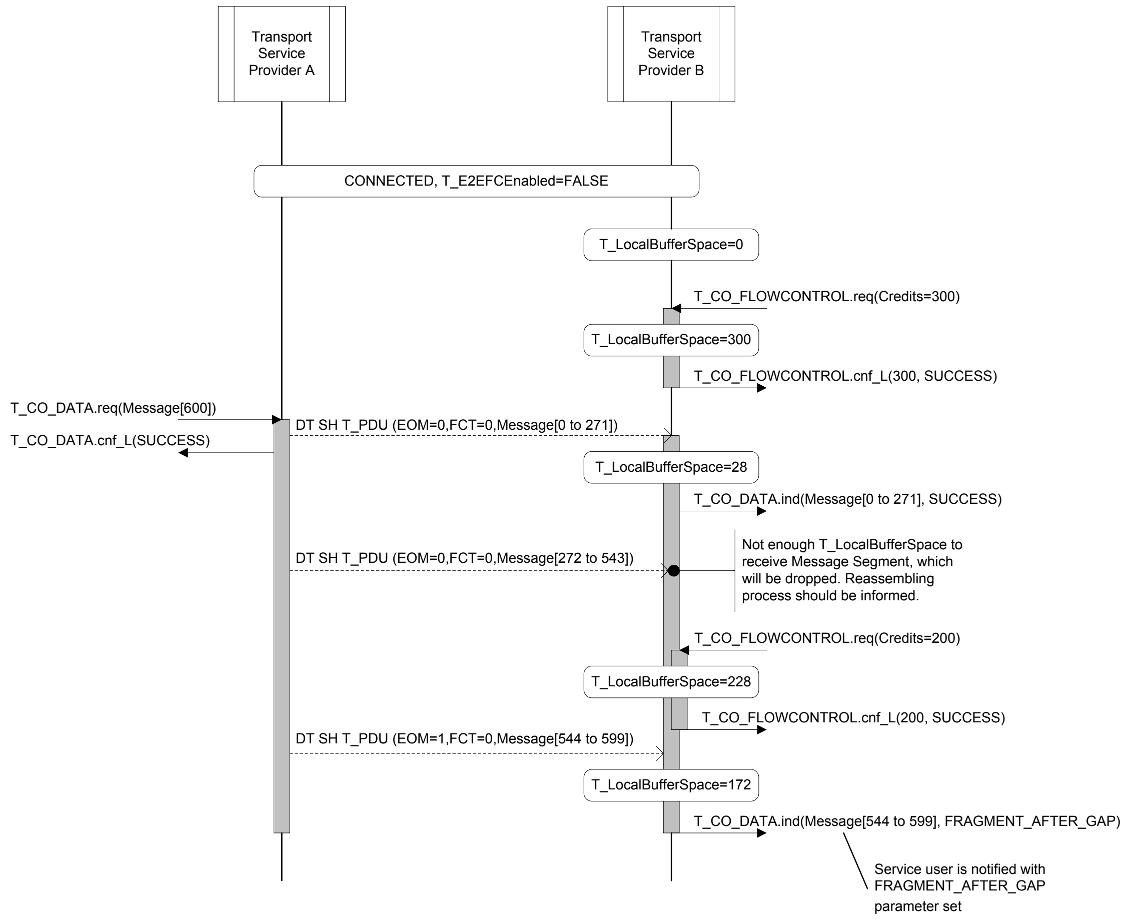

Transport Service Provider A
Transport Service Provider B
CONNECTED, T_E2EFCEnabled=FALSE
T_LocalBufferSpace=0
T_CO_FLOWCONTROL.req(Credits=300)
T_LocalBufferSpace=300
T_CO_FLOWCONTROL.cnf_L(300, SUCCESS)
T_CO_DATA.req(Message[600])
T_CO_DATA.cnf_L(SUCCESS)
DT SH T_PDU (EOM=0,FCT=0,Message[0 to 271])
T_LocalBufferSpace=28
T_CO_DATA.ind(Message[0 to 271], SUCCESS)
DT SH T_PDU (EOM=0,FCT=0,Message[272 to 543])
Not enough T_LocalBufferSpace to receive Message Segment, which will be dropped. Reassembling process should be informed.
T_CO_FLOWCONTROL.req(Credits=200)
T_LocalBufferSpace=228
T_CO_FLOWCONTROL.cnf_L(200, SUCCESS)
DT SH T_PDU (EOM=1,FCT=0,Message[544 to 599])
T_LocalBufferSpace=172
T_CO_DATA.ind(Message[544 to 599], FRAGMENT_AFTER_GAP)
Service user is notified with FRAGMENT_AFTER_GAP parameter set

Figure 92 Unsuccessful Data Transmission Due to Interim Lack of Credits

8.10.11 CPort Safety Valve

The receiver part of a CPort has a "CPort Safety Valve" mechanism that becomes active when the receiving Application, for whatever reason, cannot accept the payload of incoming Segments at the rate at which the corresponding Data Frames are received across the Network. For example, if a software Application crashes, it would be undesirable if the "misbehaving" Application would cause backpressure and impact other (generally unrelated) traffic on the UniPro Network, which otherwise could impact performance and possibly cause deadlocks. In a well-designed system, the CPort Safety Valve mechanism (CSV) should never trigger. This means that when E2E FC is turned on, Connections are still reliable.

The price one pays for the CSV mechanism is that the CPort's received data stream will be corrupted due to dropped data whenever the CSV mechanism is triggered. This loss of data integrity is signaled to the Application, which can decide how to handle this. Full recovery is unlikely because the problem is essentially a design issue within the receiving Device.

A correct operation of the CPort Safety Valve implies that a CPort shall accept a stream of Segments without causing any throttling of the incoming stream through backpressure – regardless of how slowly the Application accepts the incoming data. The CPort shall discard any incoming Segment payload that cannot be delivered immediately.

T_CPortFlags.CSV_n shall indicate whether CPort Safety Valve is enabled ('0') or not ('1').

CSV works independently of E2E FC and CSD, and can be used in conjunction with both E2E FC and CSD, or without any of them being turned on.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

8.10.12 Error Reporting for CSD and CSV

When the payload of one or more data Segments is discarded due to Controlled Segment Dropping or the CPort Safety Valve mechanisms, this shall be indicated to the Application when a next Message Fragment is passed up to the Application by setting the FragmentStatus parameter to a non-zero value. This indicates that the Message Fragment and the Message the Fragment belongs to are corrupt, that one or more Message Fragments were lost, or both.

The dropping of one or more Segment payloads due to the CSV mechanism shall be signaled to the DME using the T_LM_DISCARD indication with L4DiscardReasonCode set to SAFETY_VALVE_DROPPING.

The dropping of one or more Segment payloads due to the CSD mechanism shall also be signaled to the DME using the T_LM_DISCARD indication with L4DiscardReasonCode set to CONTROLLED_SEGMENT_DROPPING.

If, with E2E FC enabled, one or more Segment payloads had to be dropped because the CPort received more data than could fit into T_LocalBufferSpace, this shall be signaled to the DME using the T_LM_DISCARD indication with L4DiscardReasonCode set to E2E_CREDIT_OVERFLOW. This is not supposed to happen in a well-designed system and might indicate a conformance issue within either endpoint, or a system configuration error.

8.10.13 Discard T_PDU Feature

In case of any occurrences of the situations listed in ***Table 77***, the Transport Layer shall perform all actions necessary to free up any resources used for the particular affected process.

On every invocation of the Discard T_PDU feature the T_LM_DISCARD.ind shall be given with the corresponding L4DiscardReasonCode.

8.11 Segment Transmission

A source Device shall transmit T_SDUs in the order in which they arrived at the local T_CO_SAP.

For the transmission of Segments the following operations shall be performed in the order listed below:

- The correct T_PDU type shall be determined according to the rules defined in ***Section 8.10.2.1***.
- The T_SDU shall be segmented according to the procedure defined in ***Section 8.10.5***
- The selected T_PDU shall be composed by means of the T_PDU composition feature (***Section 8.10.7***) and in accordance to the encoding rules defined in ***Section 8.10.2***
    - The present header fields shall be set with the values
        - provided and maintained by the protocol features involved
        - provided by the Service Primitive invoked
    - The payload field shall be filled with the current Message Segment of the given Message or Message Fragment
- The resulting T_PDU shall be transmitted in accordance to the E2E FC procedure defined in ***Section 8.10.10.1*** if the E2E FC is enabled for the particular Connection

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

8.12 Segment Reception

Upon the reception of a Segment the following operations shall be performed in the order listed below:

- The received T_PDU shall be decomposed by means of the T_PDU decomposition feature (***Section 8.10.8***) involving the header format analysis feature (***Section 8.10.9***) and in accordance to ***Section 8.10.2***
    - The type of the T_PDU shall be recovered
    - The present header fields shall be recovered and the resulting values shall be made available to the corresponding protocol features
    - The user-data (or Segments of user-data) shall be obtained from the payload field
- The T_SDU shall be reassembled according to the procedure defined in ***Section 8.10.6***
- The T_PDU should be received and processed in accordance to the Controlled Segment Dropping procedure defined in ***Section 8.10.10.2*** if the E2E FC is disabled for the particular Connection
- If the receiving Application cannot accept the data, the Segment is dropped according to the Safety Valve procedure defined in ***Section 8.10.11***.
- The user-data shall be indicated to the CPort User by means of the corresponding Service Primitives according to ***Section 8.8.1*** and with accordingly set parameters after the T_SDU has been reassembled. In case of interim Segment loss, this error shall be indicated to the receiving CPort User.

Upon any exception listed in ***Section 8.10.13*** the Discard T_PDU feature shall be performed.

8.13 CPort Arbitration

The Transport Layer provides support for multiple CPorts. As a result, the Transport Layer may have multiple connections simultaneously open. When opened, the connections are mapped to one of the available Traffic Classes, which is then used for the entire connection duration.

For segment transmission, the Transport Layer shall use segment-level round robin as a default arbitration scheme between CPorts mapped to the same Traffic Class. Additional arbitration schemes may also be provided.

For an implementation that supports multiple Traffic Classes, and requires Transport Layer arbitration across CPorts mapped on different Traffic Classes, the CPorts mapped to the higher-priority Traffic Classes shall have precedence over CPorts mapped on lower-priority Traffic Classes. The Traffic Class priority shall be determined according to the Data Link Layer definition in ***Section 6.6.4***.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

8.14 UniPro Test Feature

The UniPro Test Feature consists of two components, TstSrc and TstDst, as described in ***Table 86***. TstSrc is a traffic generator and TstDst is a traffic analyzer. Both components are configurable and can be controlled via Attributes manipulated by the DME. TstSrc and TstDst shall be used together and thus any usage of the term "Test Feature" refers to a TstSrc and TstDst pair as shown in ***Figure 93***.

Table 86 Test Feature Overview

Component	Description	Section
TstSrc	Programmable traffic generator that creates sequences of Messages (T_SDUs) containing well-defined byte sequences of well-defined lengths	***8.14.3***
TstDst	Analyzer of incoming Messages (T_SDUs) with the capability to report errors	***8.14.4***

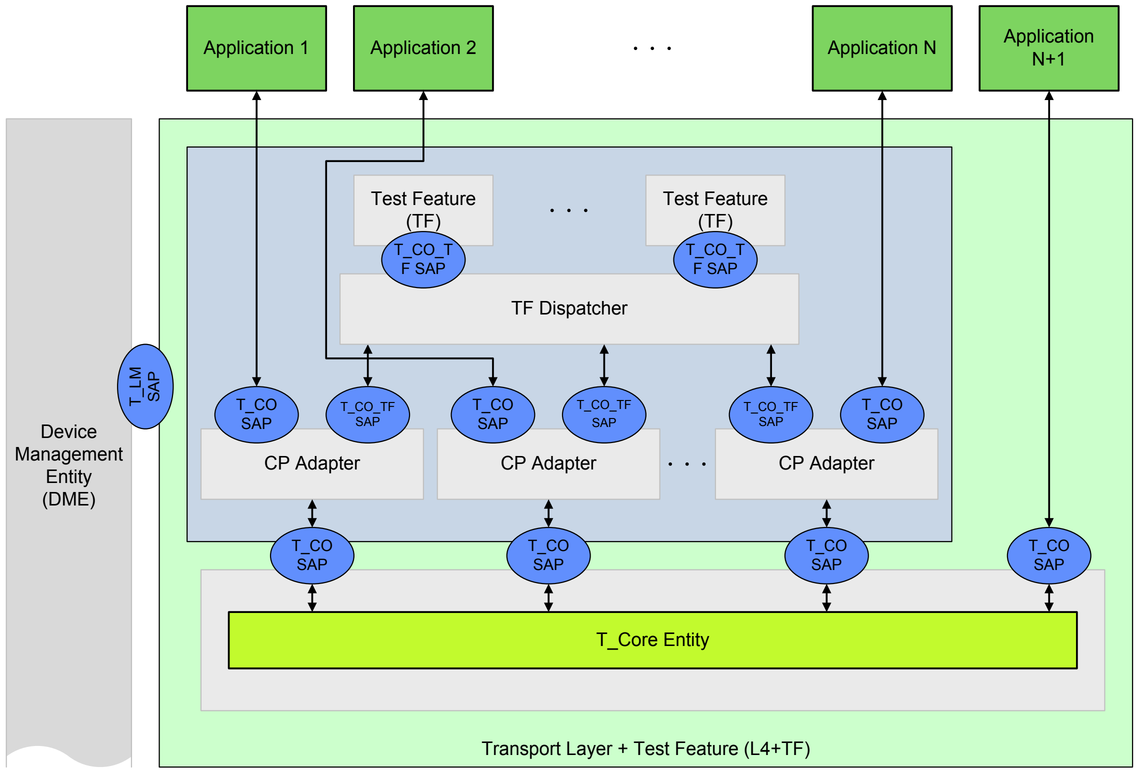


Figure 93 Test Feature Location Within the Transport Layer

The Test Feature is located below the LA and within the L4 as shown in ***Figure 93***. A Device can contain multiple Test Feature instances. The number of instances depends on the number of implemented CPorts in the Device. If the Device contains one CPort, then the Device shall contain at least one instance of the Test Feature. If the Device contains two or more CPorts, then the Device shall contain at least two instances of the Test Feature. These requirements are summarized in ***Table 87***. The previous conditions are needed to be able to test arbitration at L4.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Table 87 Determining the Number of Test Feature Instances in a DUT

Number of CPorts in the DUT	Minimum Number of Test Feature instances in the DUT	Minimum Number of CPorts Available for Testing
1	1	1
2 or more	2	2

The number of Test Feature instances in a Device shall be available through a static, gettable-only Attribute at L4 DME called T_NumTestFeatures. The interconnection between the Test Feature instances and the CPorts shall be implemented via two intermediate components, TF_Dispatcher and CP_Adapter.

TF_Dispatcher specifies the actual interconnection between the CPorts and Test Feature instances. It is worth noting that the TF_Dispatcher is not configurable and thus it is an implementation-specific block.

CP_Adapter performs the multiplexing and demultiplexing of the SAP primitives directed to and received from the CPort. The CP_Adapter shall be controlled via T_CPortMode, which sets the CPort either in the Application mode or test mode. T_CPortMode shall be maintained by every CPort regardless of whether or not it can be tested by the Test Features.

The assignment of a specific Test Feature instance to a given CPort is controlled via T_TstCPortID, which shall be maintained by every Test Feature instance. T_TstCPortID specifies the CPort which shall be tested by the Test Feature instance. If the Tester attempts to set T_TstCPortID to a CPort ID which cannot be associated with the Test Feature instance for any reason, e.g. the CP_Adapter is not connected to the TF_Dispatcher, then the Test Feature instance shall report this as an error to the requester.

The CP_Adapter shall be locked, i.e. not settable, when the CPort is connected to the Test Feature instance. Similarly, the Test Feature instance Attributes shall be locked if the TstSrc or TstDst are enabled.

The Application Layer (LA) is expected to be configured into a safe state before closing the Connection and establishing the Connection with the Test Feature instance.

8.14.1 Algorithm for Discovering the Test Feature Instances in a DUT (informative)

The following algorithm may be used by the Tester to discover the testable CPorts in the DUT.

```
Declare ConnectionMode as Enumeration of {CONNECTED, NO_CPORT, UNKNOWN};
Declare connectivity_matrix as Matrix of ConnectionMode[T_NumTestFeatures][T_NumCPorts];

for (tf_index = 0; tf_index < T_NumTestFeatures; tf_index++) {
    for (cport_id_value = 0; cport_id_value < T_NumCPorts; cport_id_value++) {
        request_primitive T_LM_SET.req(T_TstCPortID, cport_id_value, tf_index);
        receive_confirmation T_LM_SET.cnf_L(response);
        if (response == SUCCESS) {
            connectivity_matrix[tf_index][cport_id_value] = CONNECTED;
        }
        else if (response == (INVALID_MIB_ATTRIBUTE |
                              INVALID_MIB_ATTRIBUTE_VALUE |
                              LOCKED_MIB_ATTRIBUTE) {
            connectivity_matrix[tf_index][cport_id_value] = NO_CPORT;
        }
        else {
            connecitvity_matrix[tf_index][cport_id_value] = UNKNOWN;
        }
    }
}
```

8.14.2 Configuration Primitives

All Attributes defined for the Test Feature shall be accessible through the T_LM primitives defined in ***Section 8.9.1***.

8.14.3 Traffic Generator (TstSrc)

8.14.3.1 Activation

TstSrc shall be activated and start generating Messages via an Attribute called T_TstSrcOn, which shall take two values only, either TRUE or FALSE. If T_TstSrcOn is FALSE, then TstSrc shall be deactivated and does not generate any Messages. If T_TstSrcOn is TRUE, TstSrc is active and shall generate Messages. TstSrc shall use T_CO_DATA.req with the EOM parameter set to TRUE for transmitting entire Messages. When a Message is being transmitted, the TstSrc shall wait for the corresponding T_CO_DATA.cnf_L Service Primitive.

TstSrc shall be deactivated if any of the following conditions occurs:

- T_TstSrcOn is set to FALSE.
- TstSrc is configured to generate a finite number of Messages (see ***Section 8.14.3.2***) and that number of Messages has been generated.

8.14.3.2 Number of Generated Messages

TstSrc shall be configured to generate either a finite or infinite stream of Messages via an Attribute called T_TstSrcMessageCount. If T_TstSrcMessageCount is set to zero, then TstSrc shall generate an infinite stream of Messages. Otherwise, TstSrc shall generate a number of Messages equal to the non-zero value assigned to T_TstSrcMessageCount. After generating that number of Messages, TstSrc shall turn itself off by setting T_TstSrcOn to FALSE.

8.14.3.3 Message Length

The length of the generated Messages by TstSrc shall be configured via an Attribute called T_TstSrcMessageSize, which shall be able to hold any value in the range 1 to 65535, inclusively. TstSrc shall generate Messages with a length in bytes equal to the value of this Attribute.

8.14.3.4 Byte Pattern

TstSrc generates a byte stream which is used to fill Messages according to the pattern specified in T_TstSrcPattern. Currently, the Test Feature specification only defines one pattern for the generated traffic which is sawtooth pattern. The following rules shall apply for the sawtooth pattern:

- The first byte in the pattern shall have a zero value.
- The byte pattern is unique and endless as long as TstSrc is activated, i.e. the pattern shall start at the first byte of the first generated Message and it crosses the boundaries of the generated Messages. The pattern shall be reset whenever TstSrc is re-activated, i.e. by setting T_TstSrcOn to TRUE.
- TstSrc shall add T_TstSrcIncrement to the value of each subsequent byte in the pattern and shall roll-over at the maximum value, which is set to 255.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

8.14.3.5 Inter-Message Gap

TstSrc, as long as it is on, shall introduce gap delays between consecutive Messages equal to the value, in microseconds, in T_TstSrcInterMessageGap. The gap delay is measured from T_CO_DATA.cnf_L to the next T_CO_DATA.req. ***Figure 94*** shows an example in which TstSrc is configured to transmit two Messages with Inter-message gap enabled.

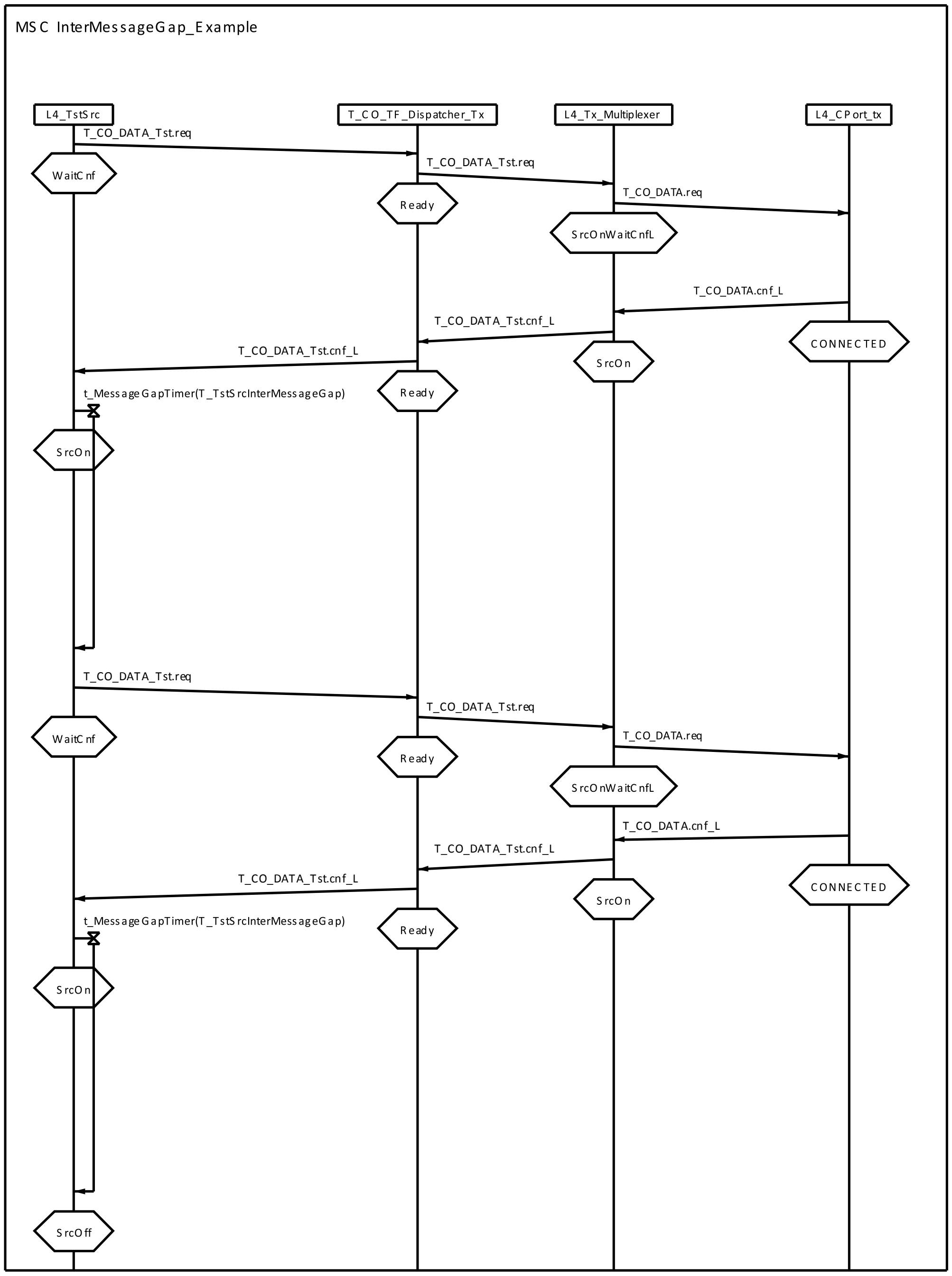


Figure 94 Inter-Message Gap Example

TstSrc shall only insert the gap delay between two Messages. As a result, TstSrc does not precede the first Message with a gap delay and does not succeed the last Message with a gap delay.

8.14.3.6 Summary of the General Test Feature and TstSrc Attributes

At reset, a settable Attribute shall be reset to its corresponding static value, if one exists, or to its Reset Value otherwise. See ***Section 9*** for more details.

Table 88 TstSrc (settable, gettable) Attributes

Attribute	Attribute ID	Description	Type	Units	Valid Attribute Value(s)	Mandatory Value(s)	Reset Value
T_TstSrcOn	0x4081	Message generation state	Boolean	–	FALSE=0 TRUE=1	FALSE=0	FALSE
T_TstSrcPattern	0x4082	Pattern used for Message generation	Enum	–	Sawtooth=0		Sawtooth
T_TstSrcIncrement	0x4083	The increment value between two consecutive bytes	Integer	–	0 to 255		1
T_TstSrcMessageSize	0x4084	The size of each generated Message	Integer	Byte	1 to 65535		256
T_TstSrcMessageCount	0x4085	The number of Messages generated (Non-zero: finite Message stream, zero: infinite Message stream)	Integer	Message	0 to 65535		256
T_TstSrcInterMessageGap	0x4086	If non-zero, the gap time between two Messages The actual duration of the timeout period shall be the value set in the Attribute ±10%	Integer	µs	0 to 65535		0

Table 89 General Test Feature (settable, gettable) Attributes

Attribute	Attribute ID	Description	Type	Units	Valid Attribute Value(s)	Mandatory Value(s)	Reset Value
T_TstCPortID	0x4080	The ID of the CPort under test	Integer	–	0 to T_MaxCPortID		Any

Copyright © 2007-2016 MIPI Alliance, Inc.
All rights reserved.
Confidential

8.14.4 Traffic Analyzer (TstDst)

TstDst acts as a consumer and analyzer of the incoming T_SDUs. TstDst sits on top of the T_Core as shown in ***Figure 93***. The analysis capability of TstDst is configurable and can be enabled and disabled as desired (see ***Section 8.14.4.4***).

8.14.4.1 Activation

TstDst shall be activated to start consuming the Messages from the associated CPort via an Attribute called T_TstDstOn. If T_TstDstOn is set to FALSE, then TstDst shall be deactivated and does not consume any Messages. If T_TstDstOn is TRUE, TstDst is active and shall consume Messages.

TstDstOn shall automatically deactivate itself if any of the following conditions occur:

- T_TstDstOn is set to FALSE.
- A first error is discovered in the incoming Message stream.

8.14.4.2 End-to-End Flow Control

TstDst, as long as it is on, shall use the Flow Control primitives to allow the associated CPort to track the local available buffer space. Three Attributes are available for controlling the flow control Service Primitives.

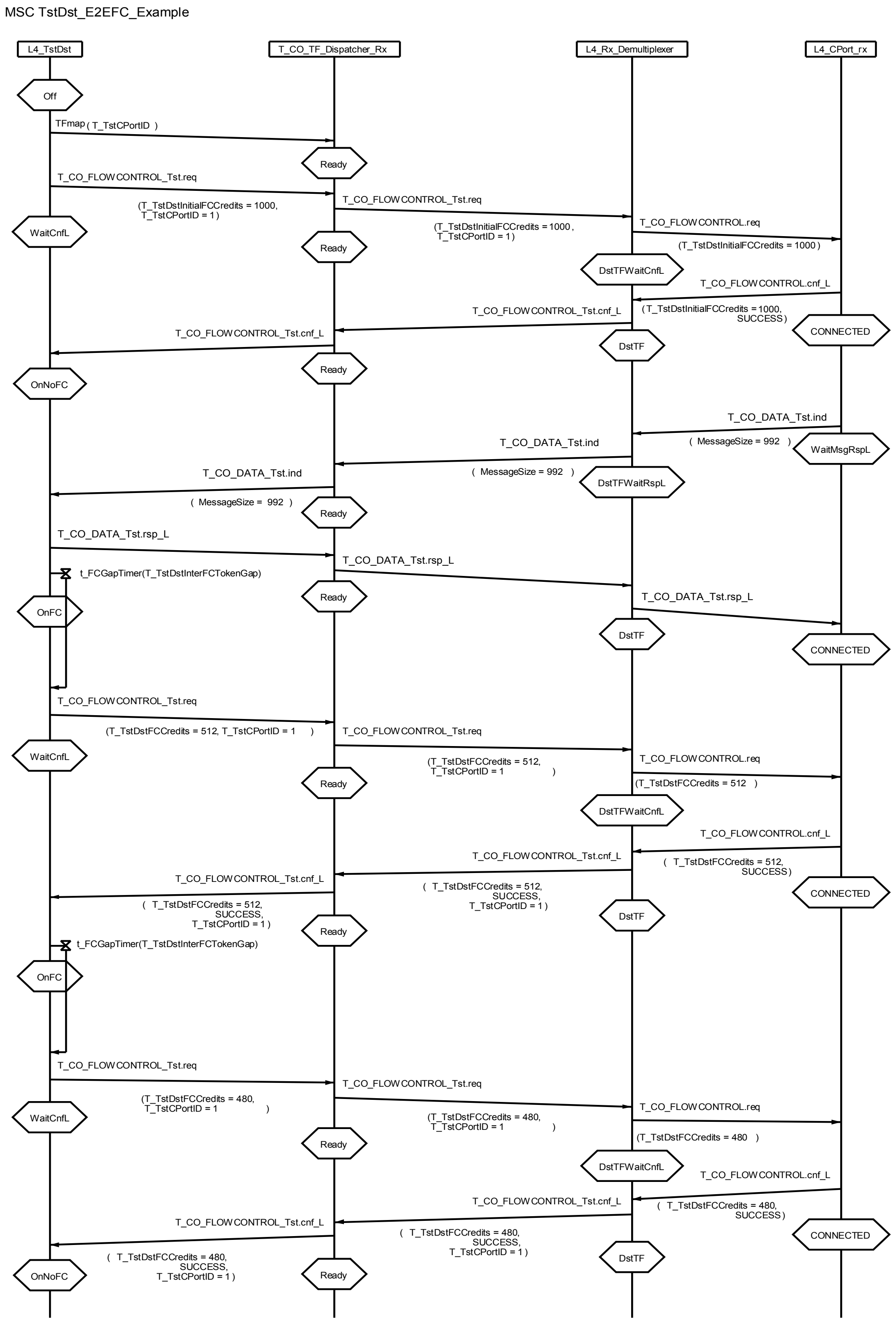
The first Attribute is called T_TstDstFCCredits. It indicates the maximum number of credits that shall be passed by TstDst to the T_CO_FLOWCONTROL.req Service Primitive. The number of credits that the TstDst sends shall be less than or equal to the amount of received data. This means that the TstDst emulates normal Application behavior. If TstDst does not receive any incoming data, then it does not request a T_CO_FLOWCONTROL.req Service Primitive.

TstDst tracks the amount of received data and generates an equal amount of end-to-end credits back. Initially, the TstDst end-to-end credits shall be set to 0. When data is received, the amount of data in bytes shall be added to the TstDst end-to-end credits. When T_CO_FLOWCONTROL.req is issued, its Credits parameter value shall be subtracted from the TstDst end-to-end credits.

The second Attribute, called T_TstDstInterFCTokenGap defines the period in microseconds at which the T_CO_FLOWCONTROL.req Service Primitive may be issued. The T_CO_FLOWCONTROL.req Service Primitive shall be issued only when TstDst end-to-end credits are available. The Credits parameter of the issued T_CO_FLOWCONTROL.req Service Primitive shall be equal to the lesser of T_TstDstFCCredits and the available TstDst end-to-end credits.

The third Attribute is called T_TstDstInitialFCCredits. It indicates the number of credits that shall be passed by the TstDst to T_CO_FLOWCONTROL.req Service Primitive immediately after enabling the TstDst. This Attribute is intended to ensure that the associated CPort tracks the local available buffer space after reset in the test mode which can be used either for issuing End-to-End Flow Control (E2EFC) tokens or Controlled Segment Dropping (CSD).

5920 An example which illustrates the E2EFC behavior within TstDst is shown in ***Figure 95***.


MSC TstDst_E2EFC_Example
L4_TstDst
T_CO_TF_Dispatcher_Rx
L4_Rx_Demultiplexer
L4_CPort_rx
Off
TFmap (T_TstCPortID)
Ready
T_CO_FLOWCONTROL_Tst.req
(T_TstDstInitialFCCredits = 1000, T_TstCPortID = 1)
T_CO_FLOWCONTROL_Tst.req
WaitCnfL
(T_TstDstInitialFCCredits = 1000 , T_TstCPortID = 1)
T_CO_FLOWCONTROL.req
Ready
(T_TstDstInitialFCCredits = 1000)
DstTFWaitCnfL
T_CO_FLOWCONTROL.cnf_L
T_CO_FLOWCONTROL_Tst.cnf_L
(T_TstDstInitialFCCredits = 1000, SUCCESS)
T_CO_FLOWCONTROL_Tst.cnf_L
CONNECTED
DstTF
Ready
OnNoFC
T_CO_DATA_Tst.ind
T_CO_DATA_Tst.ind
(MessageSize = 992)
WaitMsgRspL
T_CO_DATA_Tst.ind
(MessageSize = 992)
DstTFWaitRspL
(MessageSize = 992)
Ready
T_CO_DATA_Tst.rsp_L
T_CO_DATA_Tst.rsp_L
t_FCGapTimer(T_TstDstInterFCTokenGap)
Ready
T_CO_DATA_Tst.rsp_L
OnFC
DstTF
CONNECTED
T_CO_FLOWCONTROL_Tst.req
(T_TstDstFCCredits = 512, T_TstCPortID = 1)
T_CO_FLOWCONTROL_Tst.req
WaitCnfL
T_CO_FLOWCONTROL.req
(T_TstDstFCCredits = 512, T_TstCPortID = 1)
Ready
(T_TstDstFCCredits = 512)
DstTFWaitCnfL
T_CO_FLOWCONTROL.cnf_L
T_CO_FLOWCONTROL_Tst.cnf_L
(T_TstDstFCCredits = 512, SUCCESS)
T_CO_FLOWCONTROL_Tst.cnf_L
CONNECTED
(T_TstDstFCCredits = 512, SUCCESS, T_TstCPortID = 1)
DstTF
(T_TstDstFCCredits = 512, SUCCESS, T_TstCPortID = 1)
Ready
t_FCGapTimer(T_TstDstInterFCTokenGap)
OnFC
T_CO_FLOWCONTROL_Tst.req
T_CO_FLOWCONTROL_Tst.req
(T_TstDstFCCredits = 480, T_TstCPortID = 1)
T_CO_FLOWCONTROL.req
WaitCnfL
(T_TstDstFCCredits = 480, T_TstCPortID = 1)
Ready
(T_TstDstFCCredits = 480)
DstTFWaitCnfL
T_CO_FLOWCONTROL.cnf_L
T_CO_FLOWCONTROL_Tst.cnf_L
(T_TstDstFCCredits = 480, SUCCESS)
T_CO_FLOWCONTROL_Tst.cnf_L
CONNECTED
(T_TstDstFCCredits = 480, SUCCESS, T_TstCPortID = 1)
DstTF
(T_TstDstFCCredits = 480, SUCCESS, T_TstCPortID = 1)
OnNoFC
Ready

5921
Figure 95 E2EFC Behavior in TstDst Example

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

8.14.4.3 Message Counting

When TstDst is on, it shall increment T_TstDstMessageCount every time T_CO_DATA.ind is called with the EOM flag set to TRUE, regardless of the correctness of the Message. When T_CO_DATA.ind is called with the EOM flag set to FALSE, T_TstDstMessageCount shall not be changed.

The TstDst shall never automatically clear T_TstDstMessageCount. It is the responsibility of the Tester to set the T_TstDstMessageCount to a known value before TstDst is turned on.

8.14.4.4 Message Analysis

Message analysis functionality is configurable and is controlled via the five Attributes T_TstDstErrorDetectionEnable, TstDstPattern, T_TstDstMessageSize, T_TstDstIncrement and T_TstDstMessageOffset.

If T_TstDstErrorDetectionEnable is TRUE, TstDst shall check the size and payload of the incoming Messages based on T_TstDstIncrement and T_TstDstMessageSize. Otherwise, TstDst shall accept incoming Messages without generating any error Messages.

T_TstDstPattern indicates the expected pattern for the incoming Messages.

T_TstDstMessageSize specifies the expected Message size for the incoming Messages. If there is a mismatch between the incoming Message size and the value of this Attribute and T_TstDstErrorDetectionEnable is set to TRUE, then TstDst shall consider this an error, overwrite the Attribute value with the incoming Message length and follow the error handling procedure explained in ***Section 8.14.4.4.1***.

In the case of the sawtooth pattern, T_TstDstIncrement represents the expected increment between two consecutive bytes in the incoming Messages. If there is a mismatch between the actual increment between consecutive bytes and the value of this Attribute and T_TstDstErrorDetectionEnable is set to TRUE, then TstDst shall consider this an error, overwrite the Attribute value with the byte which has the incorrect increment, and follow the error handling procedure explained in ***Section 8.14.4.4.1***.

In the case of the sawtooth pattern, T_TstDstMessageOffset represents the index, starting from 1, of the byte which has the incorrect increment. This Attribute shall be set to zero after analyzing every incoming Message and finding no errors within it. If an error is found, then the index of the byte carrying the incorrect increment shall overwrite the Attribute.

8.14.4.4.1 Error Handling Procedure

There are three sources of errors at TstDst as listed in ***Table 90***.

Table 90 Possible Errors at TstDst

Error	Error Code	Description
FRAGMENT_CORRUPT	1	The incoming Message Fragment is labeled as “CORRUPT” by T_CO_DATA.ind
INVALID_MSG_SIZE	2	Mismatch between the incoming Message length and T_TstDstMessageSize
UNEXPECTED_BYTE_VALUE	3	In case of the Sawtooth pattern, this error code indicates a mismatch in the increment between the consecutive bytes in the incoming Message and T_TstDstIncrement

When the TstDst detects any of the errors shown in ***Table 90***, it shall perform the following actions in the given order:

- The TstDst de-activates itself by setting T_TstDstOn to FALSE.
- T_TstDstErrorDetectionEnable is set to FALSE, i.e. Message analyzing is disabled.
- The TstDst sets the T_TstDstErrorCode with the error code that generated the error.

T_TstDstMessageCount shall reflect the index of the faulty Message within the Message stream since T_TstDstMessageCount is incremented on every incoming Message with, or without, error.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

8.14.4.5 Summary of TstDst Attributes

At reset, a settable Attribute shall be reset to its corresponding static Attribute, if one exists, and to the Reset Value otherwise. See ***Section 9*** for more details.

Table 91 TstDst (settable, gettable) Attributes

Attribute	Attribute ID	Description	Type	Unit	Valid Attribute Value(s)	Mandatory Value(s)	Reset Value
T_TstDstOn	0x40A1	The state of TstDst	Boolean	–	FALSE=0 TRUE=1	FALSE=0	FALSE
T_TstDstErrorDetectionEnable	0x40A2	Enable analyzing incoming Messages	Boolean	–	FALSE=0, TRUE=1		FALSE
T_TstDstPattern	0x40A3	The expected pattern in the incoming Messages	Enum	–	Sawtooth=0		Sawtooth
T_TstDstIncrement	0x40A4	The expected increment value between two consecutive bytes	Integer	–	0 to 255		1
T_TstDstMessageCount	0x40A5	Number of received Messages	Integer	Message	0 to 65535		0
T_TstDstMessageOffset	0x40A6	Offset of error from start of Message	Integer	Byte	0 to 65535		0
T_TstDstMessageSize	0x40A7	The expected size of incoming Message	Integer	Byte	0 to 65535		256
T_TstDstFCCredits	0x40A8	The amount of credits passed to the flow control Service Primitive	Integer	Byte	1 to T_MaxE2EFCCredits		256
T_TstDstInterFCTokenGap	0x40A9	The interval in microseconds for which requesting T_CO_FLOWCONTROL.req is skipped The actual duration of the timeout period shall be the value set in the Attribute ±10%	Integer	μs	0 to 65535		0
T_TstDstInitialFCCredits	0x40AA	The initial number of credits passed by the TstDst to T_CO_FLOWCONTROL.req after enabling the TstDst	Integer	Byte	1 to T_MaxE2EFCCredits		256
T_TstDstErrorCode	0x40AB	Error code that generated the TstDst disconnection	Enum	–	NO_ERROR=0 FRAGMENT_CORRUPT=1 INVALID_MSG_SIZE=2 UNEXPECTED_BYTE_VALUE=3		NO_ERROR

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

8.15 Transport Layer and Network Layer Interaction

The generation of a T_CO_DATA.req results in the generation of one or multiple Network Layer-specific N_DATA_SHDR.req Service Primitives by the Transport Layer, in case:

- The Segment processing for transmission has been performed successfully
- The Service Primitives are called correctly, i.e. the parameters are in the defined valid range

The receipt of one or multiple Network Layer-specific N_DATA_SHDR.ind Service Primitives associated with delivery of a data unit causes the Transport Layer to generate a T_CO_DATA.ind, in case:

- The Segment processing for reception has been performed successfully (even with a potentially incomplete Message)

The Message Sequence Chart in ***Figure 96*** illustrates a possible interaction scenario between a Transport Layer and a Network Layer. For simplicity, certain parameters, e.g. DestDeviceID_Enc), are left out. When appropriate, settings of header fields are embedded for a better understanding.

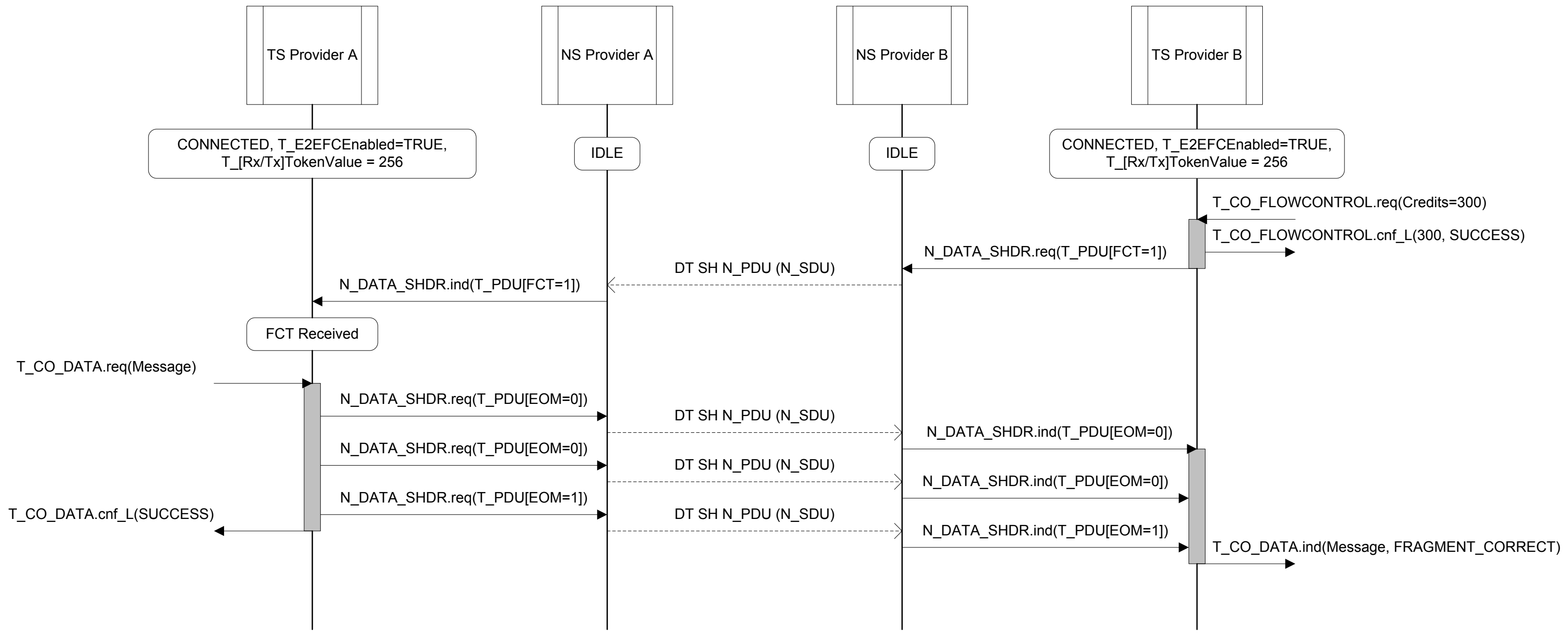

TS Provider A
NS Provider A
NS Provider B
TS Provider B
CONNECTED, T_E2EFCEnabled=TRUE, T_[Rx/Tx]TokenValue = 256
IDLE
IDLE
CONNECTED, T_E2EFCEnabled=TRUE, T_[Rx/Tx]TokenValue = 256
T_CO_FLOWCONTROL.req(Credits=300)
T_CO_FLOWCONTROL.cnf_L(300, SUCCESS)
N_DATA_SHDR.req(T_PDU[FCT=1])
DT SH N_PDU (N_SDU)
N_DATA_SHDR.ind(T_PDU[FCT=1])
FCT Received
T_CO_DATA.req(Message)
N_DATA_SHDR.req(T_PDU[EOM=0])
DT SH N_PDU (N_SDU)
N_DATA_SHDR.ind(T_PDU[EOM=0])
N_DATA_SHDR.req(T_PDU[EOM=0])
DT SH N_PDU (N_SDU)
N_DATA_SHDR.ind(T_PDU[EOM=0])
N_DATA_SHDR.req(T_PDU[EOM=1])
T_CO_DATA.cnf_L(SUCCESS)
DT SH N_PDU (N_SDU)
N_DATA_SHDR.ind(T_PDU[EOM=1])
T_CO_DATA.ind(Message, FRAGMENT_CORRECT)

Figure 96 MSC of Network Layer and Transport Layer Interaction

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

8.16 Management Information Base and Protocol Constants

Table 92 and ***Table 94*** show the Transport Layer Attributes.

Table 93 lists the Protocol Constants used within the protocol specification and within the Configuration and Capability Attributes.

All Transport Layer Attributes should be readable.

At reset, a settable Attribute shall be reset to its corresponding static Attribute, if one exists, and to the Reset Value otherwise. See ***Section 9*** for more details.

Table 92 Transport Layer (gettable, settable) Attributes

Attribute	Attribute ID	Description	Type	Units	Valid Attribute Value(s)	Mandatory Value(s)	Reset Value
Needed per CPort (Connection Parameters)							
T_PeerDeviceID	0x4021	DeviceID of the peer CPort, which is used to compute the DestDeviceID_Enc field in the Network Layer Header	Integer	–	0 to N_MaxDeviceID		Any
T_PeerCPortID	0x4022	CPortID of the peer CPort, which is used to compute the DestCPortID_Enc of the Transport Layer and the DestDeviceID_Enc field in the Network Layer Header	Integer	–	0 to T_MaxCPortID		Any

Attribute	Attribute ID	Description	Type	Units	Valid Attribute Value(s)	Mandatory Value(s)	Reset Value
Needed per CPort (Connection Parameters)							
T_ConnectionState	0x4020	State of the Connection IDLE: The connection is inactive in both directions CONNECTED: The connection is active in both directions PREPARE: The L4 behaves as if T_ConnectionState==IDLE for service requests from the Application Layer, and acts as if T_ConnectionState==CONNECTED for Service processing from the L3 to the Application CPort-related MIBattributes other than T_ConnectionState can only be changed during T_ConnectionState == IDLE, they are locked and cannot be changed, when T_ConnectionState==PREPARE or T_ConnectionState == CONNECTED	Enum	–	IDLE=0 CONNECTED =1 PREPARE=2	IDLE=0 CONNECTED =1	IDLE
T_TrafficClass	0x4023	Traffic Class of current Connection, which determines the value of the TC field in the Data Link Layer Header.	Integer	–	0, 1	Supported Traffic Class(es)	Any
T_ProtocolID[1]	0x4024	ProtocolID value of the current Connection	Integer	–	0 to 65535		0
T_CPortFlags	0x4025	CPort control register: b0: End-to-End Flow Control (E2EFC) b1: Controlled Segment Dropping (CSD_n) b2: CPort Safety Valve (CSV_n)	3-bit word	–	All	6, 2	Any
T_TxTokenValue	0x4026	Value of a E2E FC Token transmitted	Integer	Bytes	32, 64, 128, … 16777216 (powers of 2 only)		32
T_RxTokenValue	0x4027	Value of a E2E FC Token received	Integer	Bytes	32, 64, 128, … 16777216 (powers of 2 only)		32
T_LocalBufferSpace[2]	0x4028	Amount of local buffer space as tracked by the local CPort	Integer	Bytes	0 to T_MaxE2EFCCredits		0
T_PeerBufferSpace[2]	0x4029	Conservative value of the amount of buffer space at the peer CPort	Integer	Bytes	0 to T_MaxE2EFCCredits		0

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Attribute	Attribute ID	Description	Type	Units	Valid Attribute Value(s)	Mandatory Value(s)	Reset Value
Needed per CPort (Connection Parameters)							
T_CreditsToSend[2]	0x402A	Amount of space in the local buffer that has not been communicated to the peer destination port yet	Integer	Bytes	0 to T_MaxE2EFCCredits		0
T_CPortMode	0x402B	CPort Mode When set to CPORT_APPLICATION, the CPort is used by the Application. When set to CPORT_UNDER_TEST, the CPort is used by the Transport Layer Test Feature.	Enum		CPORT_APPLICATION = 1 CPORT_UNDER_TEST = 2		CPORT_APPLICATION

1. Reserved for future use.

2. The value of this Attribute is dynamic.

Table 93 Transport Layer Protocol Constants

Attribute	Description	Type	Units	Value
T_MaxSegmentSize	Maximum Segment Size (T_PDU size)	Integer	Byte	T_MTU + T_MaxHdrSize
T_MaxHdrSize	Maximum L4 header size	Integer	Byte	1
T_MTU	Transport Layer Maximum Transfer Unit	Integer	Byte	272
T_MaxCPortID	Maximum value of CPortID	Integer	–	2047
T_MaxE2EFCCredits	Maximum number of E2E FC credits	Integer	Byte	4294967295

Table 94 Transport Layer (gettable, static) Attributes

Attribute	Attribute ID	Description	Type	Unit	Valid Attribute Value(s)
T_NumCPorts	0x4000	Number of available CPorts	Integer	CPorts	1 to T_MaxCPortID+1
T_NumTestFeatures[1]	0x4001	Number of available Test Feature instances	Integer	Test Features	1 or 2 to T_NumCPorts
T_TC0TxMaxSDUSize	0x4060	Maximum transmit payload (SDU) size per Segment for TC0	Integer	Byte	1 to T_MTU
T_TC1TxMaxSDUSize	0x4061	Maximum transmit payload (SDU) size per Segment for TC1	Integer	Byte	1 to T_MTU

*1. The starting value depends on T_NumCPorts (see **Table 87**).*

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

9 Device Management Entity (DME)

This section defines the services provided by the DME. It aims to provide as much implementation flexibility as possible given the restrictions needed to achieve a high degree of interoperability.

The DME is the Service Provider with respect to Control (see ***Section 9.1***) and Configuration (see ***Section 9.2***) services described below. In turn, in order to fulfill the service provision, the DME assumes certain services from the Transport, Network, Data Link and PHY Adapter Layers. The DME service usage in relation to these layer services is explained in more detail in this section.

Figure 97 shows the SAP model used for the Device Management Entity.

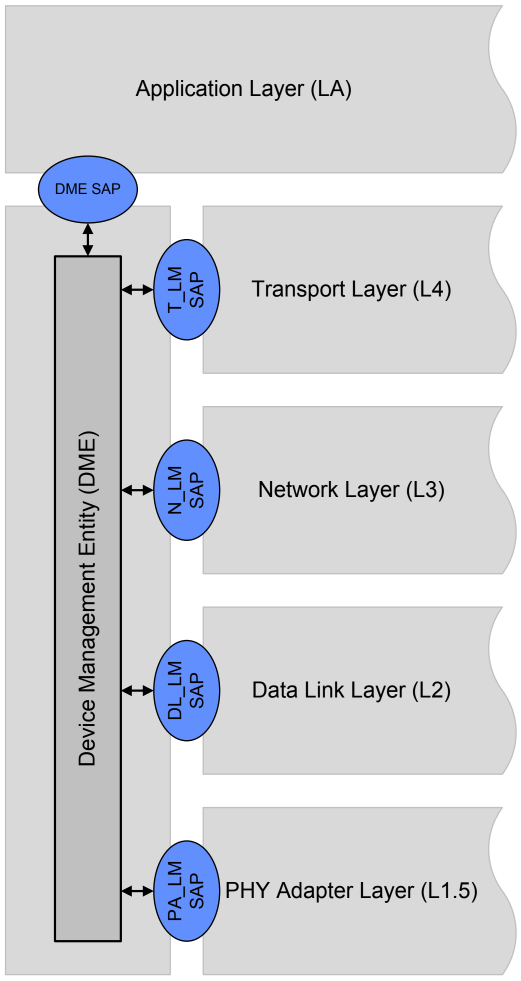


Figure 97 Device Management Entity SAP Model

The functionality of the DME can be divided into two entities, Control and Configuration.

9.1 DME Control

The DME Control entity is responsible for power-on, power-off and reset for the entire UniPro stack. In addition, it manages the Link Startup Sequence, power mode changes, EndPointReset and the hibernate entry and exit sequences for the lower UniPro stack layers.

9.2 DME Configuration

The DME Configuration entity routes Get and Set requests to the appropriate layer.

9.2.1 Attributes

The DME Configuration uses Attributes to access configurable parameters in various UniPro protocol layers. ***Figure 98*** provides the format of the Attribute address. The Attribute address is a 16-bit MIBAttributeID that has three sub-fields: an implementation-specific bit (denoted as "I"), a 3-bit layer identifier (denoted "LayerID", see ***Table 95***) and a 12-bit layer-specific Offset (per layer). When Bit 15 of the MIBAttributeID, the implementation-specific bit, is set to '0', the layer-specific Offset is defined in the relevant section of the appropriate specification, either this document or ***[MIPI02]***. When bit 15 is set to ‘1’, the layer-specific Attribute ID is implementation-specific.

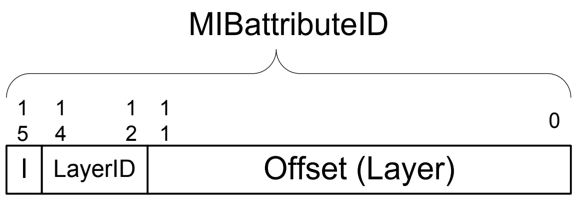


Figure 98 Attribute Address Format

9.2.2 Attribute Routing

The DME Configuration entity shall route requests based on the Attribute ID to UniPro stack layers as shown in ***Table 95***. If the destination layer does not have an Attribute corresponding to Attribute ID, it shall respond with the INVALID_MIB_ATTRIBUTE error code (ConfigResultCode, see ***Table 98***).

Table 95 DME Attributes Routing Rules

Destination Layer	LayerID (Bits 14:12)	DME Action
PHY (L1)	000	Route to PHY via PA (L1.5)
PA (L1.5)	001	Route to PA (L1.5)
DL (L2)	010	Route to DL (L2)
N (L3)	011	Route to N (L3)
T (L4)	100	Route to T (L4)
DME	101	Access local DME Attributes
–	110 and 111	DME shall respond with INVALID_MIB_ATTRIBUTE

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

9.3 DME Service Functionality and Features

Three types of reset are defined in this section, UniPro Cold Reset, UniPro Warm Reset and EndPointReset. The provision of these resets is mandatory for a UniPro implementation.

Cold Reset and Warm Reset affect the entire UniPro stack, Transport Layer (L4) through PHY Adapter Layer (L1.5), and the respective Attributes, on the local side of a Link. The difference between Warm Reset and Cold Reset is that Cold Reset clears statistics, while Warm Reset does not.

EndPointReset is a method to trigger a reset of the peer Device Application over the Link. The effect of the EndPointReset trigger on the Application, if any, is endpoint-specific and out of scope for this document.

The different types of reset are summarized in ***Table 96***.

Table 96 Reset Scope and Triggers

Reset Type	Reset Scope			Local Trigger	Remote Trigger	Results in Boot Procedure
	UniPro Stack, including Attributes	UniPro Statistics	Endpoint Application			
UniPro Cold Reset	YES	YES	NO	POR, Application	–	YES
UniPro Warm Reset	YES	NO	NO	Application	–	YES
EndPointReset	NO	NO	YES	–	TRG_EPR	NO

At the end of a UniPro Cold Reset or UniPro Warm Reset procedure, the UniPro Link is disabled and it can either be instructed to enter the Off State (DME_POWEROFF.req), or a Boot process shall be initiated (DME_ENABLE.req). For further information see ***Figure 105*** and ***Section 9.11.2***.

The three types of reset are described in ***Section 9.3.1***, ***Section 9.3.2*** and ***Section 9.3.3***. The assumed features of other layers are defined in ***Section 9.7***.

9.3.1 UniPro Cold Reset

UniPro Cold Reset is the fundamental UniPro reset. It shall be triggered by any power-on reset (POR) event, or by the local Endpoint Application. It cannot be directly triggered by a Link Startup Sequence.

The Cold Reset shall return all layers to their initial condition. All protocol state machines, timers, all Attributes including statistics, error counters and error flags shall be reset to their reset states and all data buffers shall be cleared.

When the DME receives a DME_RESET.req from the DME User it shall reset the entire UniPro stack using the <layer-identifier>_LM_RESET.req primitives. The DME shall address the layers in bottom up order (L1.5 to L4). At the end of the Reset procedure the Link reaches the Disabled state. In the Disabled state the DME User can invoke the Boot procedure using the DME_ENABLE.req primitive (see ***Section 9.11.2***), or move to the Off State using the DME_POWEROFF.req primitive.

The Cold Reset is not intended to reset the local Application.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

9.3.2 UniPro Warm Reset

UniPro Warm Reset has a very similar effect as the Cold Reset, but shall not be triggered by a POR event. A UniPro Warm Reset may be triggered by the local Endpoint Application when an unrecoverable UniPro stack failure was detected. It may also be triggered by the DME User on reception of a DME_LINKLOST.ind, indicating that the peer Device has undergone a reset and has initiated the Link Startup Sequence (see ***Section 5.4.2.7***).

The Warm Reset shall return all layers to their initial condition. All protocol state machines, timers, and Attributes shall be reset to their reset states and all data buffers shall be cleared. The difference from Cold Reset is that status fields including statistics, error counters and error flags shall not be cleared.

The Warm Reset is not intended to reset the local Application.

9.3.3 EndPointReset

The EndPointReset is a means for the local Application to request the peer Endpoint Application to reset itself. It is essentially a remote trigger event (TRG_EPR) transferred across a UniPro Link. Sending this trigger event may be requested by the local Endpoint Application using the DME_ENDPOINTRESET.req primitive.

The peer UniPort shall indicate the reception of this trigger event to its Endpoint Application. The resulting action performed by the peer Endpoint Application is optional. It may reset itself or may refuse to react to the trigger. If the Endpoint Application resets itself, it shall also apply a Warm Reset to its UniPort. If this action is not performed, the EndPointReset has no further effect on the UniPro stack.

Mapping of 'TRG_EPR' to the PHY is defined in the PHY-specific ***Section 5.7.6***.

9.4 Initializing UniPro Attributes after Reset

All UniPro Attributes shall reset to a default value specified in the MIB Attribute tables of each layer section (L1.5 to L4 and the DME). Some UniPro Attributes can load a persistent value instead. Persistent values may be stored and provisioned in a form of NVM/eFuse storage. Attributes that can load a persistent value are described in each Attribute section of the relevant layer section.

An Application may update or override the default or persistent value on-demand. The default retention policy (required for Hibernate) for each Attribute in a UniPro layer is determined by implementation. However, UniPro shall mandate a subset of Attributes to be retained. The retention polices for these Attributes are described in each Attribute section of the relevant layer section.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

9.5 Hibernate

Hibernate is a UniPro state in which the PHY is in the HIBERNATE_STATE, and the UniPro stack keeps only a minimal set of features active. The UniPro stack retains a minimal state as defined by each stack layer. During hibernation, the UniPro stack is not available to carry Application-level data transfer/communication, and the primitives of the Application-level interface, T_CO_SAP, have undefined behavior. During hibernation, the DME continues to be available for the DME User.

Figure 99 describes Link hibernation where one Device, typically a Host Device, initiates Link hibernation with a peer Device.

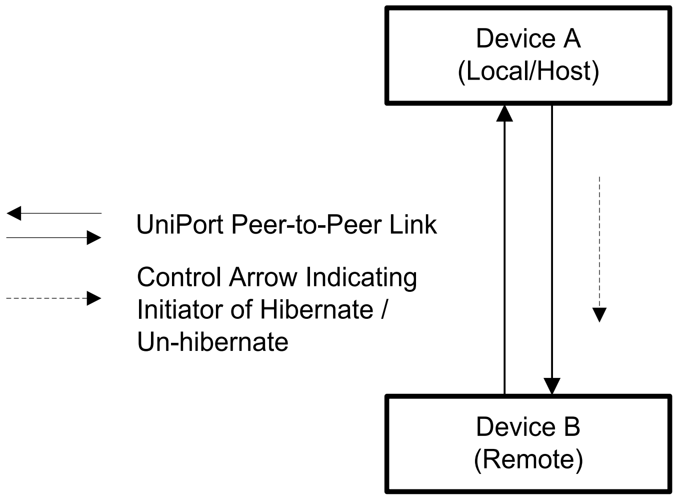

Device A
(Local/Host)
UniPort Peer-to-Peer Link
Control Arrow Indicating Initiator of Hibernate / Un-hibernate
Device B
(Remote)

Figure 99 Hibernating a Peer-to-Peer Link

The following sections describe the hibernate entry and exit flow.

9.5.1 Entering Hibernate

Figure 100 shows a high level MSC flow for entering hibernate. A detailed flow showing SAP primitives is described in the following sections. The color coding in ***Figure 100*** divides the hibernate entry flow down into steps that require exchanging data at the Application Layer, PA Layer and DME.

The Hibernate Entry Flow is explained in two key phases:

- Phase 1, which describes the entry flow up until the Application is ready to start the enter hibernation procedure.
- Phase 2, which describes the entry flow after the Application has instructed the UniPro stack to enter hibernation.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

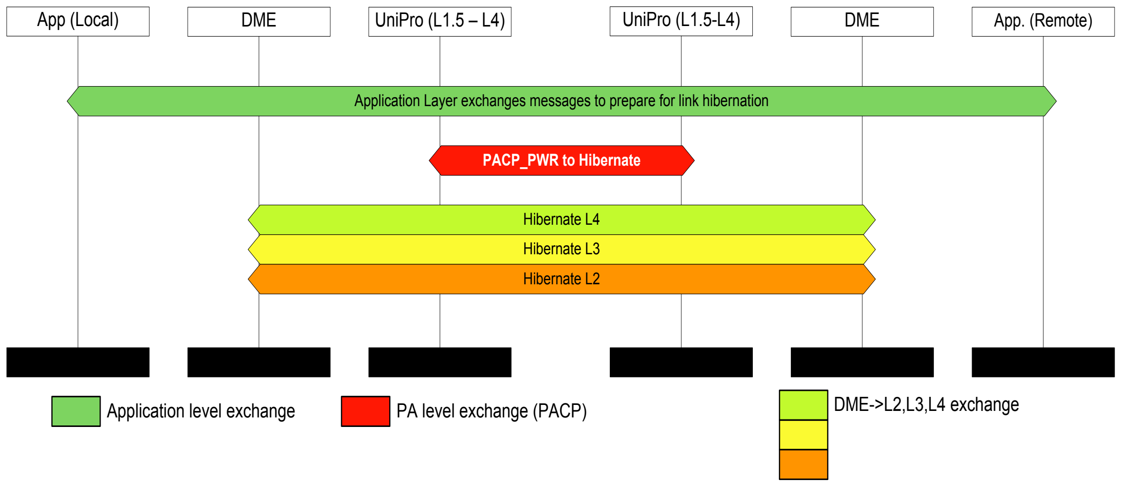


Figure 100 Hibernate High Level Entry Flow

9.5.1.1 Hibernate Entry Phase 1 Flow for Peer-to-Peer (P2P) Devices

Figure 101 depicts the Hibernate Entry Phase 1 MSC flow for a peer Device. The Application shall exchange pre-condition Messages for hibernate entry at the Application level before a UniPro stack can enter hibernate. ***Annex F*** provides an informative description of these pre-condition Messages. These recommended pre-conditions guarantee that there is no outstanding data in the UniPro stack. Once the Application level pre-conditions successfully complete, the Application is ready to hibernate the UniPro stack.

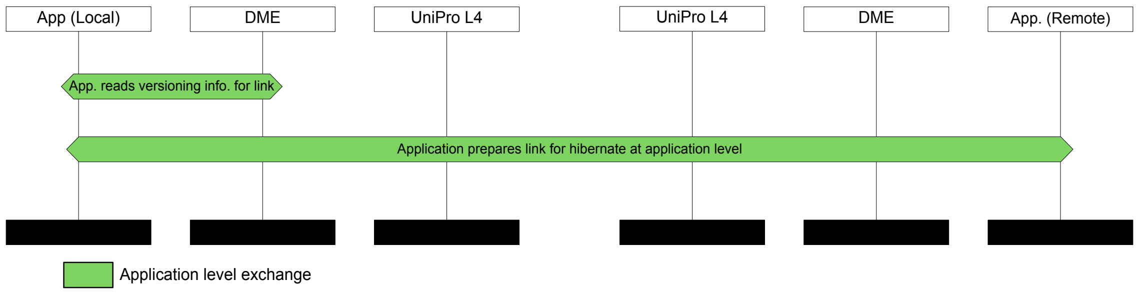

App (Local)
DME
UniPro L4
UniPro L4
DME
App. (Remote)
App. reads versioning info. for link
Application prepares link for hibernate at application level
Application level exchange

Figure 101 Hibernate Entry Flow Phase 1

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

9.5.1.2 Hibernate Entry Phase 2 Flow

Figure 102 depicts the Hibernate Entry Phase 2 flow in UniPro. Once the Application level pre-condition is completed, the Application shall initiate phase 2 sending a DME_HIBERNATE_ENTER.req to the DME. The DME in turn initiates UniPro stack hibernate by sending PA_LM_HIBERNATE_ENTER.req SAP primitive to the local PA Layer. The local PA Layer receives this request and places the M-PHY Link into hibernate using the PACP protocol (detailed description of PACP protocol can be found in ***Section 5.7.7***).

Once the PA Layer has successfully hibernated the M-PHY Link, subsequent layers of the local and peer UniPro stack (L4 to L2) shall be hibernated by the DME by sending a <layer-identifer>_LM_HIBERNATE_ENTER.req SAP primitive to the respective layers.

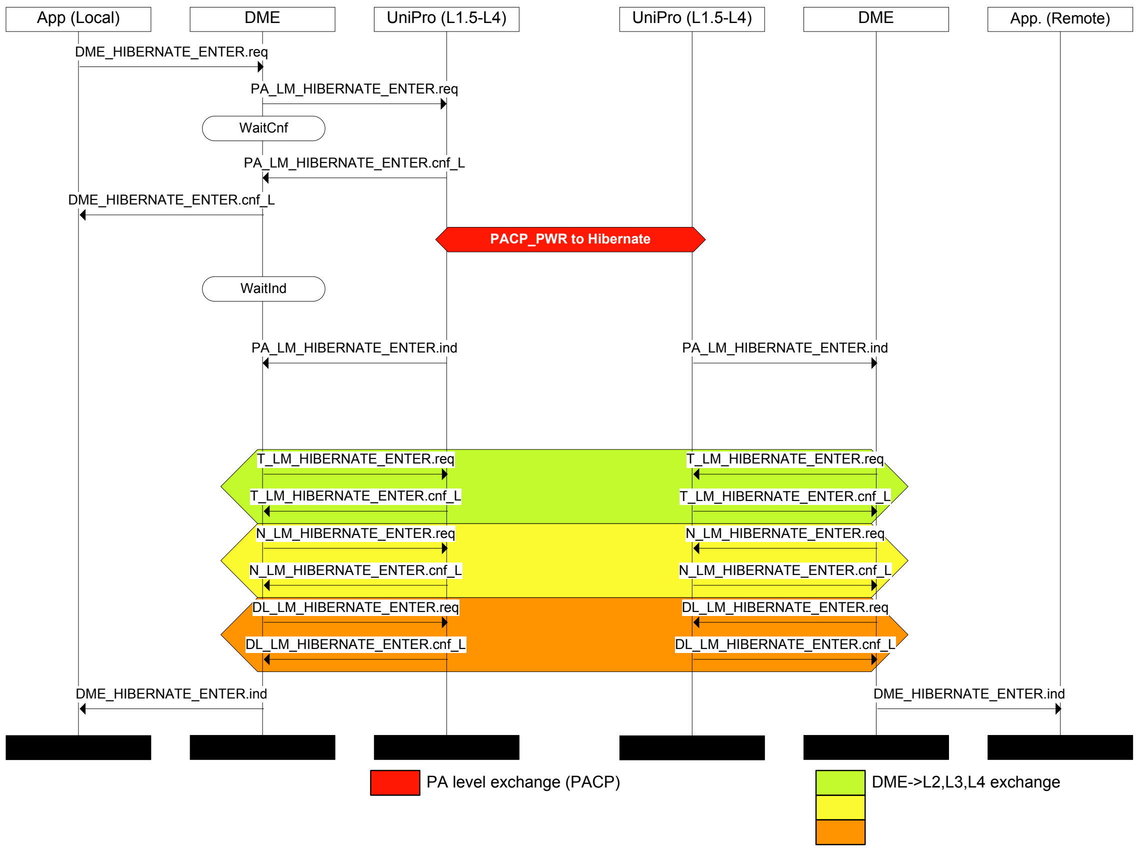

App (Local)
DME
UniPro (L1.5-L4)
UniPro (L1.5-L4)
DME
App. (Remote)
DME_HIBERNATE_ENTER.req
PA_LM_HIBERNATE_ENTER.req
WaitCnf
PA_LM_HIBERNATE_ENTER.cnf_L
DME_HIBERNATE_ENTER.cnf_L
PACP_PWR to Hibernate
WaitInd
PA_LM_HIBERNATE_ENTER.ind
PA_LM_HIBERNATE_ENTER.ind
T_LM_HIBERNATE_ENTER.req
T_LM_HIBERNATE_ENTER.req
T_LM_HIBERNATE_ENTER.cnf_L
T_LM_HIBERNATE_ENTER.cnf_L
N_LM_HIBERNATE_ENTER.req
N_LM_HIBERNATE_ENTER.req
N_LM_HIBERNATE_ENTER.cnf_L
N_LM_HIBERNATE_ENTER.cnf_L
DL_LM_HIBERNATE_ENTER.req
DL_LM_HIBERNATE_ENTER.req
DL_LM_HIBERNATE_ENTER.cnf_L
DL_LM_HIBERNATE_ENTER.cnf_L
DME_HIBERNATE_ENTER.ind
DME_HIBERNATE_ENTER.ind
PA level exchange (PACP)
DME->L2,L3,L4 exchange

Figure 102 Hibernate Entry Flow Phase 2

9.5.2 Exiting Hibernate (Un-hibernate)

Figure 103 shows the hibernate exit flow. The Application shall be responsible for un-hibernating a UniPro Link.

The Application shall initiate hibernate exit by sending DME_HIBERNATE_EXIT.req to the DME. The DME in turn initiates UniPro stack un-hibernate by sending PA_LM_HIBERNATE_EXIT.req SAP primitive to the local PA Layer. The local PA Layer receives this request and un-hibernates the M-PHY Link using the PACP protocol (detailed description of PACP protocol can be found in ***Section 5***).

Once the PA Layer has successfully un-hibernated the M-PHY Link, the DME shall un-hibernate subsequent layers of the local and peer UniPro stacks (L4 to L2) by sending a <layer-identifier>_LM_HIBERNATE_EXIT.req primitive to the respective layers.

Hibernate PRE-exit Service Primitives, <layer-identifier>_LM_PRE_HIBERNATE_EXIT, may be added from the DME to the appropriate layers in order to allow the layers more time to wake-up from Hibernate state (see ***Figure 103***). In general, <layer-identifier>_LM_PRE_HIBERNATE_EXIT primitives should be used for waking up the layers (receive path) from the Hibernate state and <layer-identifier>_LM_HIBERNATE_EXIT primitives for activating the layers (transmit path) after wake-up. PA_LM_PRE_HIBERNATE_EXIT.ind primitive is sent from the PA Layer to the DME, and is used to indicate to the DME the correct time to perform <layer-identifier>_LM_PRE_HIBERNATE_EXIT during a peer Device initiated Hibernate Exit Procedure. The layers might receive data between PA_LM_PRE_HIBERNATE_EXIT.rsp_L and <layer-identifier>_LM_HIBERNATE_EXIT requests, and they shall be able to handle that data correctly, independent of the implementation of <layer-identifier>_LM_PRE_HIBERNATE_EXIT.

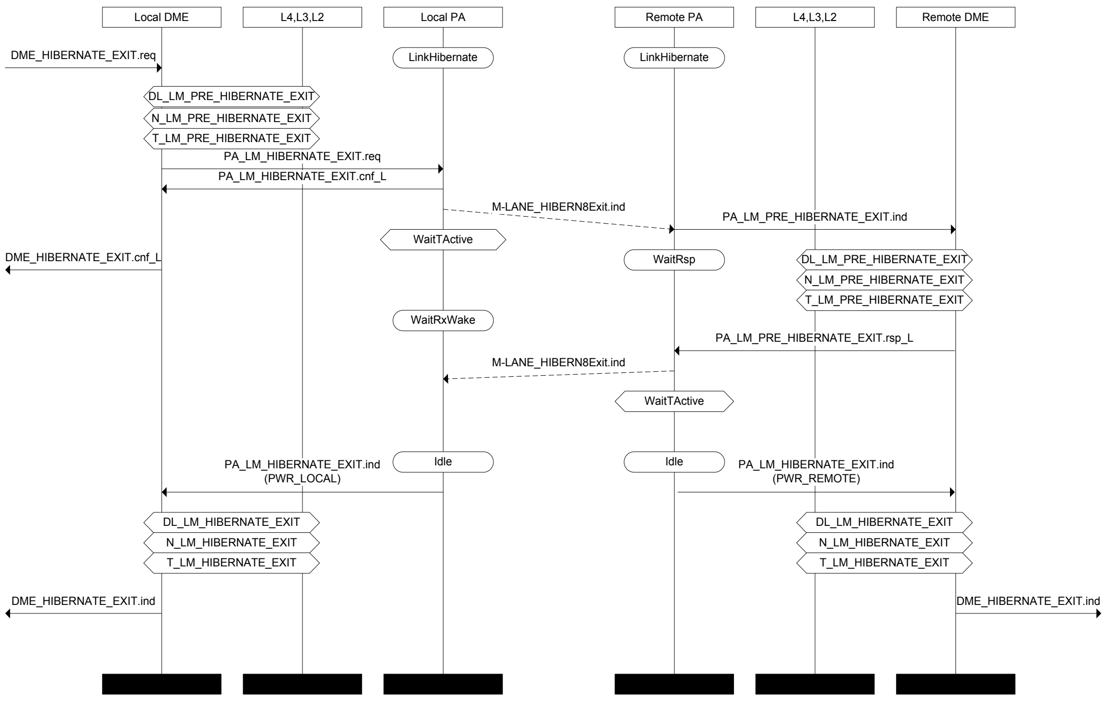


Figure 103 Hibernate Exit Flow

9.5.3 Failing to Enter or Exit Hibernate

The DME sends confirmation of a hibernate or un-hibernate request to the Application. The Application is responsible for handling failures to hibernate or un-hibernate the Link. Upon failure, the Application should retry the request, or retry the request then perform a Link initialization. The Application can also only perform a Link initialization. In all cases, the DME should report persistent failure to the Application.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

9.6 DME Power Mode Change

The power mode change from the local PA Layer's perspective is depicted in ***Figure 46***. Detailed description is specified in ***Section 5.7.12***. The power mode change from the DME perspective is provided in ***Figure 104***.

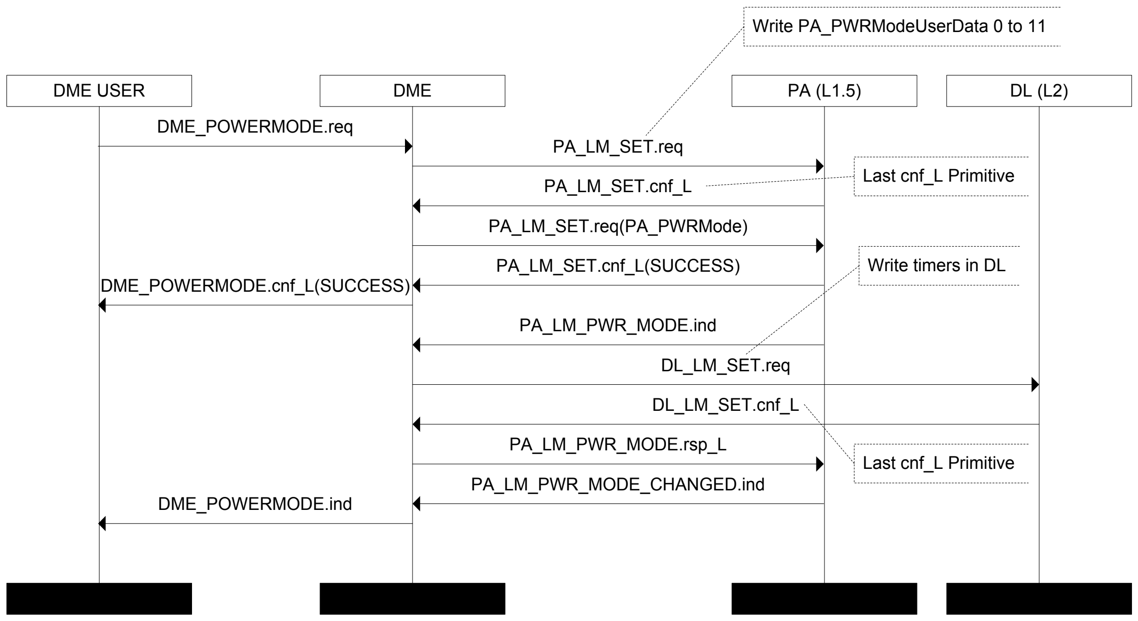


Figure 104 Power Mode Change (DME Perspective)

The power mode change in the DME level may be split into the following three distinctive phases:

- Issue a request to the local PA Layer
- Update the local L2 timer values
- Wait until the request (to the local PA Layer) has been completed

Note:

The first phase is skipped when the request is made by the peer DME. The second phase is missing if the power mode change request fails.

The local PA Layer provides a method to transmit data between two peer DME entities. The local data is sent to the peer DME and the peer DME's response is reported back. The payload is twenty-four bytes of data (see ***Table 8***) to fit all L2 timer values for all TCs.

9.6.1 Issuing the Request

The power mode configuration shall be given to the local PA Layer via LM SAP. All power mode related Attributes shall be set before activating the power mode change request, which is done by setting PA_PWRMode.

The procedure in brief:

- Set the parameter values of the PA Layer Attributes using the PA_LM_SET.req primitive as follows:
    - for TX: PA_ActiveTxDataLanes, PA_TxGear, and PA_TxTermination
    - for RX: PA_ActiveRxDataLanes, PA_RxGear, and PA_RxTermination
    - PA_HSSeries
- If the return value of any PA_LM_SET.cnf_L primitive was not SUCCESS, the request has failed, e.g. due to capability mismatch

- Set the PA_PWRMode using the PA_LM_SET.req primitive, where parameter TxPowerMode is mapped to PA_PWRMode[b3:b0] and parameter RxPowerMode is mapped to PA_PWRMode[b7:b4].
- If the return values is SUCCESS, then the local PA Layer has accepted the request
- Wait for the PA_LM_PWR_MODE_CHANGED.ind primitive (see ***Section 5.4.1.7***)

9.6.2 Updating L2 Timer Values

The local PA Layer is informed with the PA_LM_PWR_MODE.ind primitive when the timer values in the L2 Layer can be updated. At that point, the L2 timers are stopped due to a request by the local PA Layer.

There are two parameters in the PA_LM_PWR_MODE.ind primitive, PAResult and PAPowerModeUserData. Two scenarios are defined by the value of PAResult:

- PAResult is set to TRUE. This indicated that the power mode change request was created by the local DME. In this case the second parameter is discarded, and the LocalL2TimerData parameter of the DME_POWERMODE.req primitive shall be used to program the L2 timers.
- PAResult is set to FALSE. This indicates that the power mode change request was created by a peer DME. In this case the second parameter, PAPowerModeUserData, contains the content of the peer DME's PA_PWRModeUserData Attribute and shall be used to program the L2 timers.

The DME shall update the local L2 timer values using the DL_LM_SET.req primitives.

When the DME has finished updating the local L2 timer values, it informs the local PA Layer to resume using the PA_LM_PWR_MODE.rsp_L primitive. This primitive contains a parameter, which carries the PA_PWRModeUserData information to the peer DME. In the current version of UniPro, the PAPowerModeUserData parameter in the PA_LM_PWR_MODE.rsp_L primitive shall be set to the reserved value.

The local PA Layer informs the local L2 layer internally when the L2 timers can be re-started.

9.6.3 Completion

If the power mode change request was accepted, the DME shall wait until the local PA Layer returns with the PA_LM_PWR_MODE_CHANGED.ind primitive. When the peer DME initiates the request, the local DME shall receive the primitive even though it did not issue the power mode change request.

The PA_LM_PWR_MODE_CHANGED.ind primitive has a parameter indicating the result of the power mode change operation. Parameter values are defined in ***Table 9***.

9.7 Features Assumed From Other Layers

The DME is associated with:

- the local Transport layer via the layer's T_LM_SAP as defined in ***Section 8.9***
- the local Network layer via the layer's N_LM_SAP as defined in ***Section 7.7***
- the local Data Link layer via the layer's DL_LM_SAP as defined in ***Section 6.4***
- the local PHY Adapter layer via the layer's PA_LM_SAP as defined in ***Section 5.4***

The DME requires each associated layer to support the following features:

- The GET, SET, RESET (COLD, WARM), ENABLE and HIBERNATE (ENTER, EXIT) primitives

The DME further requires the PHY Adapter layer to support the following features:

- Methods to send and receive the Link Startup Sequence
- Methods to send and receive the EndPointReset trigger

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

9.8 DME_SAP

The primitives on this Service Access Point are intended to be used by an Application Layer, that is, a component that is higher in the reset hierarchy than the UniPro stack.

9.8.1 Configuration Primitives

The configuration primitives, GET and SET, of the DME provides the means to retrieve and store values, respectively, for the configuration Attributes found in the UniPro stack and in the underlying PHY Layer.

Table 97 DME_SAP Configuration Primitives

Name	Request	Indication	Local Response	Response	Local Confirm	Confirm
DME_GET	***9.8.1.1***	–	–	–	***9.8.1.2***	–
DME_SET	***9.8.1.3***	–	–	–	***9.8.1.4***	–
DME_PEER_GET	***9.8.1.5***	–	–	–	–	***9.8.1.6***
DME_PEER_SET	***9.8.1.7***	–	–	–	–	***9.8.1.8***

After successful completion of the Link Startup procedure, or after entering PHY test mode, the DME shall have remote access to Attributes of the peer Device on the UniPro Link via the local PA Layer.

Even if the capability to request the access to peer Device Attributes is optional for a single Device, at least one Device on each UniPro Link, usually the Host, needs this capability. Hence, a Host shall support the DME_PEER_GET.xxx and the DME_PEER_SET.xxx primitives. These primitives are optional for a Peripheral. Both Host and Peripheral shall support DME_GET.xxx and DME_SET.xxx. primitives.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

6190 **Table 98 DME_SAP Configuration Primitive Parameters**

Name	Type	Valid Range	Value	Description
AttrSetType	Enum	NORMAL	0	Select whether the actual value (NORMAL) or the Attribute's non-volatile (STATIC) value is addressed
		STATIC	1	
MIBattribute	Integer	0x0000 to 0xFFFF	–	The address of the MIBattribute
GenSelectorIndex	Integer	M-PHY: 0 to 2*PA_MaxDataLanes – 1 CPort: 0 to T_NumCPorts – 1 Test Feature: 0 to T_NumTestFeatures – 1	–	Indicates the targeted M-PHY data lane or CPort or Test Feature when relevant
ConfigResultCode	Enum	SUCCESS	0	Indicates the result of the request (see further explanation in ***Table 99***).
		INVALID_MIB_ATTRIBUTE	1	
		INVALID_MIB_ATTRIBUTE_VALUE	2	
		READ_ONLY_MIB_ATTRIBUTE	3	
		WRITE_ONLY_MIB_ATTRIBUTE	4	
		BAD_INDEX	5	
		LOCKED_MIB_ATTRIBUTE	6	
		BAD_TEST_FEATURE_INDEX	7	
		PEER_COMMUNICATION_FAILURE	8	
		BUSY	9	
		DME_FAILURE	10	
MIBvalue	Variable	–	—	The value of the MIB Attribute. The MIB value size cannot exceed thirty-two bits

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Table 99 provides further explanation to the various Return Codes.

Table 99 ConfigResultCode Values

ConfigResultCode	Condition
SUCCESS	The request succeeds, i.e. the value or the reset value of the Attribute indicated by MIBattribute and SelectorIndex is set to the value of MIBvalue
INVALID_MIB_ATTRIBUTE	The MIBattribute value is invalid, or otherwise the Attribute indicated by MIBattribute and SelectorIndex is not implemented, or otherwise If AttrSetType is STATIC, the Attribute indicated by MIBattribute and SelectorIndex does not support setting its reset value
INVALID_MIB_ATTRIBUTE_VALUE	The Attribute indicated by MIBattribute and SelectorIndex exists and is settable (AttrSetType = NORMAL) or allows its reset value to be set (AttrSetType = STATIC), but the value of MIBvalue is outside of the implemented range, or outside of the valid value range, for the Attribute
READ_ONLY_MIB_ATTRIBUTE	The Attribute indicated by MIBattribute and SelectorIndex exists, but is not settable. This ConfigResultCode value shall only be used if AttrSetType is NORMAL
WRITE_ONLY_MIB_ATTRIBUTE	–
BAD_INDEX	The CPort indicated by T_LM_SET.req SelectorIndex is outside of the range of available CPorts
LOCKED_MIB_ATTRIBUTE	The Attribute indicated by MIBattribute and SelectorIndex exists and is settable, but it belongs to a CPort that is in the CONNECTED state. This ConfigResultCode value shall only be used if AttrSetType is NORMAL
BAD_TEST_FEATURE_INDEX	The value of T_LM_GET.req SelectorIndex is greater than, or equal to, T_NumTestFeatures
PEER_COMMUNICATION_FAILURE	PACP peer configuration protocol has encountered error
BUSY	Previous operation related to setting Attribute has not been completed, i.e. setting power mode or peer Device Attribute access
DME_FAILURE	This value shall be used if the DME fails to perform a GET or SET operation because the layer being addressed by the DME is in an inaccessible state

The DME shall use a request/confirmation handshake for the GET and SET primitives. The DME User shall not issue a new request primitive before the previous one has terminated. Only the UniPro stack may delay a primitive processing and cause a BUSY condition.

If not in PHY test mode, the Application shall complete DME configuration access into the local or peer PHY layer by issuing a DME Power Mode Change. If in PHY test mode, configuration follows the manual procedure described in ***Section 5.7.15***.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

9.8.1.1 DME_GET.req

This primitive shall be used to request information from a specific Attribute identified by MIBattribute and, if relevant, the GenSelectorIndex. The GenSelectorIndex shall be interpreted either as a data PHY Lane index or CPort index or Test Feature index depending on the Attribute. For Attributes not associated with a GenSelectorIndex, the GenSelectorIndex shall be ignored.

The semantics of this primitive are:

DME_GET.req(MIBattribute, GenSelectorIndex)

When Generated

This primitive is generated by the DME User to obtain information from a local Attribute of the UniPro stack.

Effect on Receipt

The DME determines which layer should be accessed and forwards the GET request to that layer. Addressing is done based on the MIBattribute and the GenSelectorIndex if relevant.

9.8.1.2 DME_GET.cnf_L

This primitive reports the results of a GET request.

The semantics of this primitive are:

DME_GET.cnf_L(ConfigResultCode, MIBvalue)

When Generated

This primitive is generated by the DME in response to the DME_GET.req primitive when the GET request is completed. If the ConfigResultCode is not SUCCESS, the MIBvalue shall be ignored, otherwise MIBvalue hold the value of the requested Attribute.

The ConfigResultCode field includes a result code from the layer that was previously addressed using the DME_GET.req primitive (see ***Section 9.2.2)***.

Effect on Receipt

The DME User may generate a new DME_GET.req or DME_SET.req primitive.

9.8.1.3 DME_SET.req

This primitive shall be used to set the value of a specific Attribute identified by MIBattribute and, if relevant, the GenSelectorIndex. The GenSelectorIndex shall be interpreted either as a data PHY Lane index or CPort index or Test Feature index depending of the Attribute. For Attributes not associated with a GenSelectorIndex, the GenSelectorIndex shall be ignored.

The semantics of this primitive are:

DME_SET.req(AttrSetType, MIBattribute, MIBvalue, GenSelectorIndex)

When Generated

This primitive is generated by the DME User to set the value of a local Attribute of the UniPro stack or underlying PHY.

Effect on Receipt

The DME determines which layer should be accessed and forwards the SET request to that layer. Addressing is done based on the MIBattribute and GenSelectorIndex if relevant.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

9.8.1.4 DME_SET.cnf_L

This primitive shall be used to report the results of a SET request.

The semantics of this primitive are:

DME_SET.cnf_L(ConfigResultCode)

When Generated

This primitive is generated when the layer has finished processing the SET request. The corresponding.cnf_L primitive serves to provide information on any errors that might have occurred, and also serves as a synchronizing handshake.

The ConfigResultCode field includes a result code from the layer that was previously addressed using the DME_SET.req primitive (see ***Section 9.2.2***).

Effect on Receipt

The DME User may generate a new DME_GET.req or DME_SET.req primitive.

9.8.1.5 DME_PEER_GET.req

This primitive shall be supported by a Host and is optional for a Peripheral. However, if an implementation supports this primitive, it shall implement it as described in this section.

This primitive shall be used to request information from a specific Attribute, in the peer Device, identified by MIBattribute and, if relevant, the GenSelectorIndex. The GenSelectorIndex shall be interpreted either as a data PHY Lane index or CPort index or Test Feature index depending of the Attribute. For Attributes not associated with a GenSelectorIndex, the GenSelectorIndex shall be ignored.

The semantics of this primitive are:

DME_PEER_GET.req(MIBattribute, GenSelectorIndex)

When Generated

This primitive is generated by the DME User to obtain information from an Attribute located in the UniPro stack of the peer Device.

Effect on Receipt

The DME will forward the request to the local PA Layer by generating the PA_LM_PEER_GET.req primitive.

9.8.1.6 DME_PEER_GET.cnf

This primitive shall be supported by a Host and is optional for a Peripheral. If the DME_PEER_GET.req primitive is supported by an implementation, it shall also support the DME_PEER_GET.cnf primitive as described in this section.

This primitive reports the results of a GET request.

The semantics of this primitive are:

DME_PEER_GET.cnf(ConfigResultCode, MIBvalue)

When Generated

This primitive is generated by the DME in response to the DME_PEER_GET.req primitive when the GET request is completed. If the ConfigResultCode is not SUCCESS, the MIBvalue shall be ignored, otherwise the MIBvalue hold the value of the requested Attribute.

Effect on Receipt

The DME User may generate a new DME_PEER_GET.req or DME_PEER_SET.req primitive.

9.8.1.7 DME_PEER_SET.req

This primitive shall be supported by a Host and is optional for a Peripheral. However, if an implementation supports this primitive, it shall implement it as described in this section.

This primitive shall be used to set the MIBvalue value of a specific Attribute, of the peer Device, identified by MIBattribute and, if relevant, the GenSelectorIndex. The GenSelectorIndex shall be interpreted either as a data PHY Lane index or CPort index or Test Feature index depending of the Attribute. For Attributes not associated with a GenSelectorIndex, the GenSelectorIndex shall be ignored.

The semantics of this primitive are:

DME_PEER_SET.req(AttrSetType, MIBattribute, MIBvalue, GenSelectorIndex)

When Generated

This primitive is generated by the DME User to set the value of an Attribute located in the UniPro stack or underlying PHY of the peer Device.

Effect on Receipt

The DME will forward the request to the local PA Layer by generating the PA_LM_PEER_SET.req primitive.

9.8.1.8 DME_PEER_SET.cnf

This primitive shall be supported by a Host and is optional for a Peripheral. If the DME_PEER_SET.req primitive is supported by an implementation, it shall also support the DME_PEER_SET.cnf primitive as described in this section.

This primitive shall be used to report the results of a PEER SET request.

The semantics of this primitive are:

DME_PEER_SET.cnf(ConfigResultCode)

When Generated

This primitive is generated after the reception of the PA_LM_PEER_SET.cnf primitive.

Effect on Receipt

The DME User may generate a new DME_PEER_GET.req or DME_PEER_SET.req primitive.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

9.8.2 Control Primitives

The Service Primitives in this section are provided for controlling the reset and run mode of the entire UniPro protocol stack.

The primitives covered in this section are listed in ***Table 100***.

Table 100 DME_SAP Control Primitives

Name	Request	Indication	Local Response	Response	Local Confirm	Confirm
DME_POWERON	***9.8.2.1***	–	–	–	***9.8.2.2***	–
DME_POWEROFF	***9.8.2.3***	–	–	–	***9.8.2.4***	–
DME_ENABLE	***9.8.2.5***	–	–	–	***9.8.2.6***	–
DME_RESET	***9.8.2.7***	–	–	–	***9.8.2.8***	–
DME_ENDPOINTRESET	***9.8.2.9***	***9.8.2.11***	–	–	***9.8.2.10***	–
DME_LINKSTARTUP	***9.8.2.12***	***9.8.2.14***	–	–	***9.8.2.13***	–
DME_LINKLOST	–	***9.8.2.15***	–	–	–	–
DME_HIBERNATE_ENTER	***9.8.2.16***	***9.8.2.18***	–	–	***9.8.2.17***	–
DME_HIBERNATE_EXIT	***9.8.2.19***	***9.8.2.21***	–	–	***9.8.2.20***	–
DME_POWERMODE	***9.8.2.22***	***9.8.2.24***	–	–	***9.8.2.23***	–
DME_TEST_MODE	***9.8.2.25***	***9.8.2.27***	–	–	***9.8.2.26***	–
DME_ERROR	–	***9.8.2.28***	–	–	–	–
DME_QoS	–	***9.8.2.29***	–	–	–	–

Table 101 lists the parameters that appear in the DME_SAP control primitives.

Table 101 DME_SAP Control Primitive Parameters

Name	Type	Valid Range	Value	Description
GenericErrorCode	Enum	SUCCESS	0	Report the outcome of the operation as SUCCESS or FAILURE
		FAILURE	1	
ResetLevel	Enum	COLD	0	Select whether a Warm or a Cold reset should be performed
		WARM	1	
TxPowerMode, RxPowerMode	Enum	Fast	1	Specifies the requested power mode
		Slow	2	
		FastAuto	4	
		SlowAuto	5	
LocalL2TimerData	Struct	N/A	N/A	Data structure containing required local timer values for power mode change (see ***Table 102***)
RemoteL2TimerData	Struct	N/A	N/A	Data structure containing required peer timer values for power mode change (see ***Table 102***)
PowerChangeResultCode	Enum	See ***Table 9***		
LayerNumber	Enum	DME	0x5	The layer that reported the error indicated by LayerErrorCode
		T	0x4	
		N	0x3	
		DL	0x2	
		PA (and PHY)	0x1	
LayerErrorCode	Integer	N/A		A layer specific error code reported back to the DME (See relevant layer chapter for the list of error codes and also ***Table 103***)
QoSSource	Enum	TX	0x1	Outbound Link quality degradation expected
		RX	0x2	Inbound Link quality degradation expected
		PA_INIT	0x3	PA_INIT count exceeded, indicating a problem of either inbound or outbound Link (or both)

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Table 102 defines the LocalL2TimerData and RemoteL2TimerData organization. Values in the “Order” column correspond to the elements of the PAPowerModeUserData parameter in the PACP_PWR_req frame (see ***Figure 29***) and the PACP_PWR_cnf frame (see ***Figure 30***)

Unused and reserved bits should be set to ‘1’.

Table 102 L2 Timer Data Structure

Order	Element	Type[1]	Description
0	DL_FC0ProtectionTimeOutVal	Integer	Flow control protection timeout value for TC0
1	DL_TC0ReplayTimeOutVal	Integer	Replay timeout value for TC0
2	DL_AFC0ReqTimeOutVal	integer	AFC request timeout value for TC0
3	DL_FC1ProtectionTimeOutVal	Integer	Flow control protection timeout value for TC1
4	DL_TC1ReplayTimeOutVal	Integer	Replay timeout value for TC1
5	DL_AFC1ReqTimeOutVal	integer	AFC request timeout value for TC1
6	reserved for TC2	Integer	–
7	reserved for TC2	Integer	–
8	reserved for TC2	integer	–
9	reserved for TC3	Integer	–
10	reserved for TC3	Integer	–
11	reserved for TC3	integer	–

1 16-bit integer

Table 103 provides further references to the specific definition of the LayerErrorCode (see ***Table 101***) in each layer.

Table 103 LayerErrorCode Mapping

Layer	Layer Indicator	LayerErrorCode	Reference
PA	1.5	PHYDirection	***Table 11***
DL	2	DLErrorCode	***Table 38***
N	3	L3DiscardReasonCode	***Table 57***
T	4	L4DiscardReasonCode	***Table 76***

9.8.2.1 DME_POWERON.req

Support for this primitive is optional. However, if an implementation supports this primitive, it shall implement it as described in this section.

The DME_POWERON.req primitive is used to power on the layers L1.5 through L4 connected to the DME.

The semantics of this primitive are:

```
DME_POWERON.req( )
```

When Generated

This primitive is generated when the DME User powers on the UniPro stack.

Effect on Receipt

The UniPro stack layers are powered on by the DME. The UniPro stack, via DME, shall be able to receive and respond to DME_POWERON.req, even in the POWEREOFF state.

9.8.2.2 DME_POWERON.cnf_L

Support for this primitive is optional. If the DME_POWERON.req primitive is supported by an implementation, it shall also support the DME_POWERON.cnf_L primitive as described in this section.

The DME_POWERON.cnf_L primitive is used as a local confirm for the DME_POWERON.req primitive.

The semantics of this primitive are:

```
DME_POWERON.cnf_L( GenericErrorCode )
```

When Generated

This primitive is generated when powering on all the UniPro stack layers has been completed.

Effect on Receipt

The DME User knows that the DME has powered on all the UniPro stack layers.

9.8.2.3 DME_POWEROFF.req

Support for this primitive is optional. However, if an implementation supports this primitive, it shall implement it as described in this section.

The DME_POWEROFF.req primitive is used to power off the layers L1.5 through L4 connected to the DME.

The semantics of this primitive are:

```
DME_POWEROFF.req( )
```

When Generated

This primitive is generated when the DME User powers off the UniPro stack.

Effect on Receipt

The UniPro stack layers are powered off by the DME.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

9.8.2.4 DME_POWEROFF.cnf_L

Support for this primitive is optional. If the DME_POWEROFF.req primitive is supported by an implementation, it shall also support the DME_POWEROFF.cnf_L primitive as described in this section.

The DME_POWEROFF.cnf_L primitive is used as a local confirm for the DME_POWEROFF.req primitive.

The semantics of this primitive are:

DME_POWEROFF.cnf_L(GenericErrorCode)

When Generated

This primitive is generated when the powering off all the UniPro stack layers has been completed.

Effect on Receipt

The DME User knows that the DME has powered off the UniPro stack layers.

9.8.2.5 DME_ENABLE.req

The DME User uses this primitive to enable the UniPro stack.

The semantics of this primitive are:

DME_ENABLE.req( )

When Generated

This primitive is generated when the DME User wants to enable the UniPro stack.

Effect on Receipt

The layers are enabled in order from bottom to top by the DME.

9.8.2.6 DME_ENABLE.cnf_L

This primitive is generated when the enable procedure has been completed.

The semantics of this primitive are:

DME_ENABLE.cnf_L(GenericErrorCode)

When Generated

This primitive is generated when the enable procedure has been completed.

Effect on Receipt

The DME User knows that the DME has enabled all the layers.

9.8.2.7 DME_RESET.req

The DME_RESET.req primitive shall be used to reset the DME and the UniPro stack layers L1.5 through L4 connected to it.

The semantics of this primitive are:

DME_RESET.req(ResetLevel)

When Generated

This primitive is generated when the DME User resets the UniPro stack through the DME.

Effect on Receipt

The UniPro stack is reset and changes state to the Disabled state.

9.8.2.8 DME_RESET.cnf_L

The DME_RESET.cnf_L primitive shall be used as a local confirm for the DME_RESET.req primitive.

The semantics of this primitive are:

DME_RESET.cnf_L( )

When Generated

This primitive is generated when reset process has been completed.

Effect on Receipt

The DME User knows that the DME has completed the reset process.

9.8.2.9 DME_ENDPOINTRESET.req

The DME_ENDPOINTRESET.req primitive shall be used to send an EndPointReset request command to the Link Endpoint.

The semantics of this primitive are:

DME_ENDPOINTRESET.req( )

When Generated

This primitive is generated when the DME User request the DME to perform a reset of the Endpoint.

Effect on Receipt

The EndPointReset command is sent by the local PA Layer to the Link Endpoint.

9.8.2.10 DME_ENDPOINTRESET.cnf_L

The DME_ENDPOINTRESET.cnf_L primitive shall be used as a local confirmation for the reception of the DME_ENDPOINTRESET.req primitive by the DME.

The semantics of this primitive are:

DME_ENDPOINTRESET.cnf_L(GenericErrorCode)

When Generated

This primitive is generated when EndPointReset command has been sent by the local PA Layer.

Effect on Receipt

The DME User knows that the EndPointReset command has been sent by the local PA Layer.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

9.8.2.11 DME_ENDPOINTRESET.ind

The DME_ENDPOINTRESET.ind primitive shall be used to indicate that the EndPointReset request has been received.

The semantics of this primitive are:

DME_ENDPOINTRESET.ind( )

When Generated

This primitive is generated to the DME User when the EndPointReset response is received by the local PA Layer.

Effect on Receipt

No automatic reset shall be performed, but it is up to DME User to respond accordingly.

9.8.2.12 DME_LINKSTARTUP.req

The DME_LINKSTARTUP.req primitive shall be used to start the UniPro Link up.

The semantics of this primitive are:

DME_LINKSTARTUP.req( )

When Generated

This primitive is generated when the DME User requests the DME to start the UniPro Link up process.

Effect on Receipt

First the DME starts the local PA Layer and then the local DL Layer with dedicated primitives in the PA_LM and DL_LM SAPs.

9.8.2.13 DME_LINKSTARTUP.cnf_L

The DME_LINKSTARTUP.cnf_L primitive shall be used as a local confirm for the DME_LINKSTARTUP.req primitive.

The semantics of this primitive are:

DME_LINKSTARTUP.cnf_L(GenericErrorCode)

When Generated

This primitive is generated when the start-up process has been completed successfully or error is encountered. The success or failure is reported in the primitive parameter. the DME passes this confirmation to the DME User.

Effect on Receipt

If the Link start-up process was successful, the DME User knows that the UniPro Link is in the LinkUp state. Otherwise the Link remains in the LinkDown state.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

9.8.2.14 DME_LINKSTARTUP.ind

The DME_LINKSTARTUP.ind primitive shall be used as an indication that Link start-up process has been initiated by the peer of the Link.

The semantics of this primitive are:

DME_LINKSTARTUP.ind( )

When Generated

This primitive is generated when the Link is in LinkDown state and the DME receives the PA_LM_LINKSTARTUP.ind primitive from the local PA Layer, which indicates the peer is trying to establish a Link.

Effect on Receipt

The local DME User responds with DME_LINKSTARTUP.req primitive to complete the Link start-up process.

9.8.2.15 DME_LINKLOST.ind

The DME_LINKLOST.ind primitive shall be used as an indication that Link is lost.

The semantics of this primitive are:

DME_LINKLOST.ind( )

When Generated

This primitive is generated when the Link is in LinkUp state or Hibernate state and the DME receives the PA_LM_LINKSTARTUP.ind primitive from the local PA Layer. This indicates a condition where the peer is trying to re-establish a Link and the Link is lost.

Effect on Receipt

The Link is put in the LinkLost state and the DME User must apply a DME_RESET to redo the boot sequence and get to the LinkUp state.

9.8.2.16 DME_HIBERNATE_ENTER.req

The DME_HIBERNATE_ENTER.req primitive shall be used in LinkUp to put the Link and all the UniPro layers into the Hibernate state to save power.

The semantics of this primitive are:

DME_HIBERNATE_ENTER.req( )

When Generated

This primitive is generated when the DME User requests the Link to move into the Hibernate state.

Effect on Receipt

The Hibernate state entering process, which is carried out by the DME, layer by layer, is started. In LinkUp state, the local PA Layer communicates the Hibernate state change with the peer. The local PA Layer indicates the change of state (to the hibernate state) to the DME.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

9.8.2.17 DME_HIBERNATE_ENTER.cnf_L

The DME_HIBERNATE_ENTER.cnf_L primitive shall be used as a local confirmation for the DME_HIBERNATE_ENTER.req primitive.

The semantics of this primitive are:

DME_HIBERNATE_ENTER.cnf_L(GenericErrorCode)

When Generated

This primitive is generated as a local confirmation for the hibernate enter request.

Effect on Receipt

If no error is returned the process to go into hibernate state continues.

9.8.2.18 DME_HIBERNATE_ENTER.ind

The DME_HIBERNATE_ENTER.ind primitive shall be used to indicate the completion of the Hibernate Enter sequence. The PowerChangeResultCode parameter shall indicate if the procedure has succeeded or not. The procedure has succeeded if the returned PowerChangeResultCode value is either PWR_LOCAL or PWR_REMOTE. In this case, the Link is in the Hibernate state. Both ends of the Link receive this primitive.

The semantics of this primitive are:

DME_HIBERNATE_ENTER.ind(PowerChangeResultCode)

When Generated

This primitive is generated when the hibernate entering process has been completed and the Link state is changed to the Hibernate state if the process was successful.

Effect on Receipt

The DME User knows that the DME has completed its part of the hibernate enter process successfully or if an error was encountered during the process.

9.8.2.19 DME_HIBERNATE_EXIT.req

The DME_HIBERNATE_EXIT.req primitive shall be used to get the Link out of the Hibernate state.

The semantics of this primitive are:

DME_HIBERNATE_EXIT.req( )

When Generated

This primitive is generated when the DME User requests the DME to get the Link out of the Hibernate state.

Effect on Receipt

The Hibernate exit process, which is carried out by the DME, layer by layer, is started. When exiting from Hibernate into LinkUp state, the local PA Layer communicates the Hibernate exit with the peer. The local PA Layer indicates the change of state (out of the hibernate state) back to the DME.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

9.8.2.20 DME_HIBERNATE_EXIT.cnf_L

The DME_HIBERNATE_EXIT.cnf_L primitive shall be used as a local confirmation for the DME_HIBERNATE_EXIT.req primitive.

The semantics of this primitive are:

DME_HIBERNATE_EXIT.cnf_L(GenericErrorCode)

When Generated

This primitive is generated as a local confirmation for the hibernate exit request.

Effect on Receipt

If no error is returned the process to exit the Hibernate state proceeds.

9.8.2.21 DME_HIBERNATE_EXIT.ind

The DME_HIBERNATE_EXIT.ind primitive shall be used to indicate the completion of the Hibernate Exit sequence. The PowerChangeResultCode parameter shall indicate if the procedure has succeeded or not. The procedure has succeeded if the returned PowerChangeResultCode value is either PWR_LOCAL or PWR_REMOTE. In this case the Link has left the Hibernate state. Both ends of the Link receive this primitive.

The semantics of this primitive are:

DME_HIBERNATE_EXIT.ind(PowerChangeResultCode)

When Generated

This primitive is generated when the hibernate exit process has been completed successfully by the DME and the Link state is changed to the LinkUp state. Otherwise an error shall indicate failure of the exit process.

Effect on Receipt

The DME User knows that the DME has completed its part of the hibernate exit process successfully and the Link is now in the LinkUp state. If the hibernate process has failed an error shall be reported.

9.8.2.22 DME_POWERMODE.req

This primitive shall be used to change the power mode of the Link (See ***Section 5*** and ***Section 6*** for more information about power mode change). The requested UniPro power mode and all the Data Link Layer timer required for the power mode change are provided as parameters. Attributes defining the PHY parameters for the new power mode change shall be configured by the DME User before issuing the DME_POWERMODE.req primitive.

The semantics of this primitive are:

DME_POWERMODE.req(TxPowerMode, RxPowerMode, LocalL2TimerData, RemoteL2TimerData)

When Generated

This primitive is generated when the DME User wants the DME to change the power mode of the Link.

Effect on Receipt

The DME completes the power mode change process.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

9.8.2.23 DME_POWERMODE.cnf_L

The DME_POWERMODE.cnf_L primitive shall be used as a local confirmation for the DME_POWERMODE.req primitive.

The semantics of this primitive are:

DME_POWERMODE.cnf_L(GenericErrorCode)

When Generated

This primitive is generated as a local confirmation for the DME_POWEMODE.req primitive including possible error code if the power mode change is impossible due to an invalid set of parameters provided or for any other reason.

Effect on Receipt

The DME User knows that the power mode change process will take place.

9.8.2.24 DME_POWERMODE.ind

This indication shall be used to indicate that the DME, PA Layer, or DL Layer part of the power mode change has been completed. Both ends receive this indication.

The semantics of this primitive are:

DME_POWERMODE.ind(PowerChangeResultCode)

When Generated

This primitive is generated when the power mode has been changed.

Effect on Receipt

The DME User knows that the power mode change has been completed.

9.8.2.25 DME_TEST_MODE.req

This primitive shall be supported by a Host and is optional for a Peripheral. However, if an implementation supports this primitive, it shall implement it as described in this section.

This primitive is used to put the peer Device into test mode.

The semantics of this primitive are:

DME_TEST_MODE.req( )

When Generated

This primitive is generated by the DME User to put the peer Device in to a given test mode.

Effect on Receipt

The DME forwards the request to the local PA Layer by generating the PA_LM_TEST_MODE.req primitive.

9.8.2.26 DME_TEST_MODE.cnf_L

This primitive shall be supported by a Host and is optional for a Peripheral. If the DME_TEST_MODE.req primitive is supported by an implementation, it shall also support the DME_TEST_MODE.cnf_L primitive as described in this section.

This primitive inform the DME User that the sequence to put the peer Device in to a given test mode was sent.

The semantics of this primitive are:

DME_TEST_MODE.cnf_L(GenericErrorCode)

When Generated

This primitive is generated after reception of the PA_LM_TEST_MODE.cnf_L primitive indicating that the request was sent to the peer Device.

Effect on Receipt

The DME User knows that peer Device is in the given test mode.

9.8.2.27 DME_TEST_MODE.ind

This primitive is used to indicate to the DME User, that the UniPro stack has been put in to a certain test mode.

The semantics of this primitive are:

DME_TEST_MODE.ind( )

When Generated

This primitive is generated when the DME receives the PA_LM_TEST_MODE.ind primitive.

Effect on Receipt

The DME User knows that the UniPro stack is not in a normal operation mode anymore, but in a given test mode.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

9.8.2.28 DME_ERROR.ind

This primitive is used to indicate to the DME User, that a layer in the UniPro stack has encountered an error condition. The information forwarded to the DME user differs by layer:

- For the Transport layer (L4), the T_LM_DISCARD.ind information is forwarded to the DME User.
- For the Network layer (L3), the N_LM_DISCARD.ind information is forwarded to the DME User.
- For the Data Link layer (L2), the DL_LM_ERROR.ind information is forwarded to the DME User.
- For the PA Layer (L1.5), the PA_LM_PHY_RESET.ind information is forwarded to the DME User.

The semantics of this primitive are:

DME_ERROR.ind(LayerNumber, LayerErrorCode)

LayerNumber follows the definition in ***Table 101***. The LayerErrorCode is defined in ***Table 103***.

When Generated

This primitive is generated when the DME receives an error indication primitive from one of the UniPro layers.

Effect on Receipt

The DME User knows that the UniPro stack has encountered an error condition and acts accordingly.

9.8.2.29 DME_QoS.ind

This primitive is used to indicate to the DME User that a PHY Link degradation has been monitored. The Link degradation may not be severe enough to trigger a fatal Link error, but the DME User might need to take corrective action.

The semantics of this primitive are:

DME_QoS.ind(QoSSource)

QoSSource follows the definition in ***Table 101***.

When Generated

This primitive is generated when one of the QoS thresholds being overstepped.

Effect on Receipt

The DME User knows that the UniPro stack has encountered a Quality of Service degradation and acts accordingly.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

9.9 UniPro Versioning

The DME shall keep the version information of the UniPro stack. The version information is passed to the local PA Layer version Attribute by the DME using the PA_LM_SET.req primitive before the Link startup operation. The version information mapping to a 16-bit Attribute field shall be as defined in ***Table 104***.

Table 104 UniPro Version Information Mapping

Field	Bits	Values	Description
Unused	[15:10]	0	N/A
Connection Management Protocol Present	9	1	Connection Management Protocol present.
		0	No Connection Management Protocol present.
Config Protocol Present	8	1	Config Protocol present.
		0	No Config Protocol present.
Device Type	[7:4]	0	Endpoint
UniPro Version	[3:0]	6 to 15	Above version 1.8
		5	UniPro version 1.8
		4	UniPro version 1.61
		3	UniPro version 1.6
		2	UniPro version 1.41.00
		1	UniPro version 1.40.00
		0	Reserved value

The Fields Connection Management Protocol Present and Config Protocol Present shall be set to zero.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

9.10 UniPro States

Figure 105 shows an abstracted and simplified state diagram of the UniPro stack after power on and DME_RESET.req. After Power On, the stack is in the Disabled state until it gets a local request to start-up the Link (DME_ENABLE.req). It then moves to the LinkDown state and waits until it receives a confirmation of the LinkStartUp procedure (PA_LM_LINKSTARTUP.cnf_L). If the LinkStartUp sequence is successful, the stack moves to the LinkUp state and is ready to send and receive data. The stack can enter hibernate mode from LinkUp state upon receipt of a request from the DME. During Link configuration (such as a power mode change) the stack enters the LinkCfg state temporarily, during which time no data exchange is possible. Once the change in configuration is successfully applied, or it fails, the stack returns to the LinkUp state.

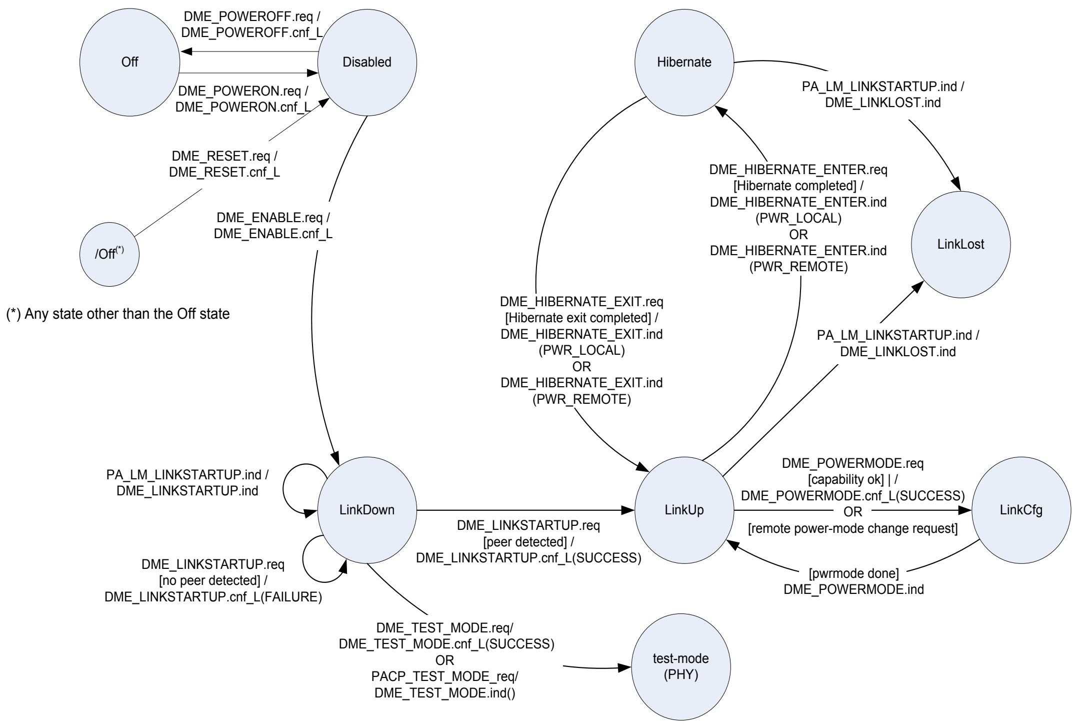

Off
DME_POWEROFF.req / DME_POWEROFF.cnf_L
Disabled
DME_POWERON.req / DME_POWERON.cnf_L
DME_RESET.req / DME_RESET.cnf_L
/Off(*)
DME_ENABLE.req / DME_ENABLE.cnf_L
(*) Any state other than the Off state
Hibernate
PA_LM_LINKSTARTUP.ind / DME_LINKLOST.ind
DME_HIBERNATE_ENTER.req [Hibernate completed] / DME_HIBERNATE_ENTER.ind (PWR_LOCAL) OR DME_HIBERNATE_ENTER.ind (PWR_REMOTE)
LinkLost
DME_HIBERNATE_EXIT.req [Hibernate exit completed] / DME_HIBERNATE_EXIT.ind (PWR_LOCAL) OR DME_HIBERNATE_EXIT.ind (PWR_REMOTE)
PA_LM_LINKSTARTUP.ind / DME_LINKLOST.ind
PA_LM_LINKSTARTUP.ind / DME_LINKSTARTUP.ind
LinkDown
DME_LINKSTARTUP.req [no peer detected] / DME_LINKSTARTUP.cnf_L(FAILURE)
DME_LINKSTARTUP.req [peer detected] / DME_LINKSTARTUP.cnf_L(SUCCESS)
LinkUp
DME_POWERMODE.req [capability ok] | / DME_POWERMODE.cnf_L(SUCCESS) OR [remote power-mode change request]
LinkCfg
[pwrmode done] DME_POWERMODE.ind
DME_TEST_MODE.req/ DME_TEST_MODE.cnf_L(SUCCESS) OR PACP_TEST_MODE_req/ DME_TEST_MODE.ind()
test-mode (PHY)

Figure 105 UniPro States

Based on the UniPro state diagram shown in ***Figure 105***, some restrictions are enforced on all DME_<service-primitive-name>.req and DME_<service-primitive-name>.ind primitives (see ***Section 9.8***). These restrictions specify when a primitive is going to be executed, or when it shall be ignored by an implementation (see ***Table 105***).

Table 105 DME_SAP Restrictions

Primitive	When Restricted (request ignored, indication not generated)
DME_SET.req	Restricted in the OFF and Disabled states. The Application or the implementation may additionally restrict use of DME_SET.req and DME_GET.req during Hibernate state.
DME_GET.req	
DME_PEER_SET.req	In the OFF, Disabled, LinkDown, Hibernate and LinkLost states
DME_PEER_GET.req	
DME_POWERON.req	In all states except the OFF state
DME_POWEROFF.req	In all states except the Disabled state
DME_ENABLE.req	In all states except the Disabled state
DME_ENDPOINTRESET.ind	In all states except the LinkUp state
DME_ENDPOINTRESET.req	In all states except the LinkUp state
DME_LINKLOST.ind	In all states except the LinkUp and Hibernate states
DME_LINKSTARTUP.ind	In all states except the LinkDown state
DME_LINKSTARTUP.req	In all states except the LinkDown state
DME_HIBERNATE_ENTER.ind	In all states except the LinkUp states
DME_HIBERNATE_ENTER.req	In all states except the LinkUp states
DME_HIBERNATE_EXIT.ind	In all states except the Hibernate state
DME_HIBERNATE_EXIT.req	In all states except the Hibernate state
DME_POWERMODE.ind	In all states except the LinkUp and LinkCfg states
DME_POWERMODE.req	In all states except the LinkUp state
DME_TEST_MODE.ind	In all states except the LinkDown state
DME_TEST_MODE.req	In all states except the LinkDown state
DME_QoS.ind	In all states except the LinkUp and LinkCfg state

If a DME_SAP request control primitive is issued in a restricted state, the DME shall set the GenericErrorCode field in the local confirmation primitive to FAILURE.

If a DME_SAP configuration primitive is issued in a restricted state, the DME shall set the ConfigResultCode field in the confirmation primitive to DME_FAILURE.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

9.11 Reset and Boot Procedures

This section describes the UniPro Reset and Boot procedures.

9.11.1 Reset Procedure

The reset procedure is described in ***Section 9.3***.

9.11.2 Boot Procedure

The UniPro boot procedure follows a Cold Reset, including a power-on reset, or a Warm Reset. The boot procedure is invoked by the DME User using the DME_ENABLE.req primitive.

The UniPro boot procedure is controlled by the DME, which ensures that the UniPro layers are initialized in order, beginning with L1.5 up to L4. Thus, the DME enables a layer only after the layer below it has reported that it is enabled. For example, the UniPro boot procedure ensures that L2 does not start its AFC exchange until the L1.5 exercises the Link Startup Sequence and reports that the PHY is operational.

Upon receiving the DME_ENABLE.req primitive, each UniPro layer shall be enabled by the DME using the <layer-identifier>_LM_ENABLE_LAYER.req primitive. After being enabled, each layer shall perform its own initialization. Each layer shall use the <layer-identifier>_LM_ENABLE_LAYER.cnf_L primitive to indicate to the DME that the layer has completed initialization.

At the end of this phase the Link reached the LinkDown state and the DME shall report this condition to the DME User using the DME_ENABLE.cnf_L primitive.

The DME User shall proceed with the boot sequence and instruct the DME to transition the Link to the LinkUp state using the DME_LINKSTARTUP.req primitive.

Upon receiving the DME_LINKSTARTUP.req primitive the DME shall startup L1.5 followed by L2 using the <layer-identifier>_LM_LINKSTARTUP.req primitive. Each layer shall use the <layer-identifier>_LM_STARTUP.cnf_L primitive to indicate to the DME that the layer has completed the respective layer startup procedure. The UniPro boot procedure ensures that L2 does not start its AFC exchange until the L1.5 exercises the Link Startup Sequence and reports that the PHY is operational.

At the end of this phase, the Link reaches the LinkUp state and the DME shall report this condition to the DME User using the DME_LINKSTARTUP.cnf_L primitive.

To be able to communicate through a UniPro stack, Network and Transport Layers also need to be configured. In particular, N_DeviceID and N_DeviceID_valid need to be configured in the Network Layer, and the Attributes for connection setup need to be set in the Transport Layer as described in ***Section 8.10.3***. These Attributes are either automatically set to the defined reset values, or set by the Application before the Link is used. In the latter case, these Attributes should be set immediately after receiving the DME_LINKSTARTUP.cnf_L.

After receiving the DME_LINKSTARTUP.cnf_L and configuring the Network Layer and Transport Layer Attributes, the boot process is complete and the Link is ready for data transfers.

After reception of a failure confirmation DME_LINKSTARTUP.cnf_L(FAILURE), the Link is typically left in an unknown state with set-up inconsistencies between the local Device M-PHY and the peer Device M-PHY. Before a new Link start-up is attempted, the DME User should consider resetting and re-enabling the stack using the primitive DME_RESET.req followed by the primitive DME_ENABLE.req, to enable a LinkStartup re-attempt.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

9.11.3 Boot Procedure Example (informative)

The following three types of boot sequence are provided for reference:

- Default, which is shown in ***Figure 106***.
- Peer initiated, which is shown in ***Figure 107***.
- ***Figure 108*** describes a special case where a peer Device-initiated sequence resulted in a Link reaching LinkUp state, while receiving a DME_LINKLOST.ind primitive.

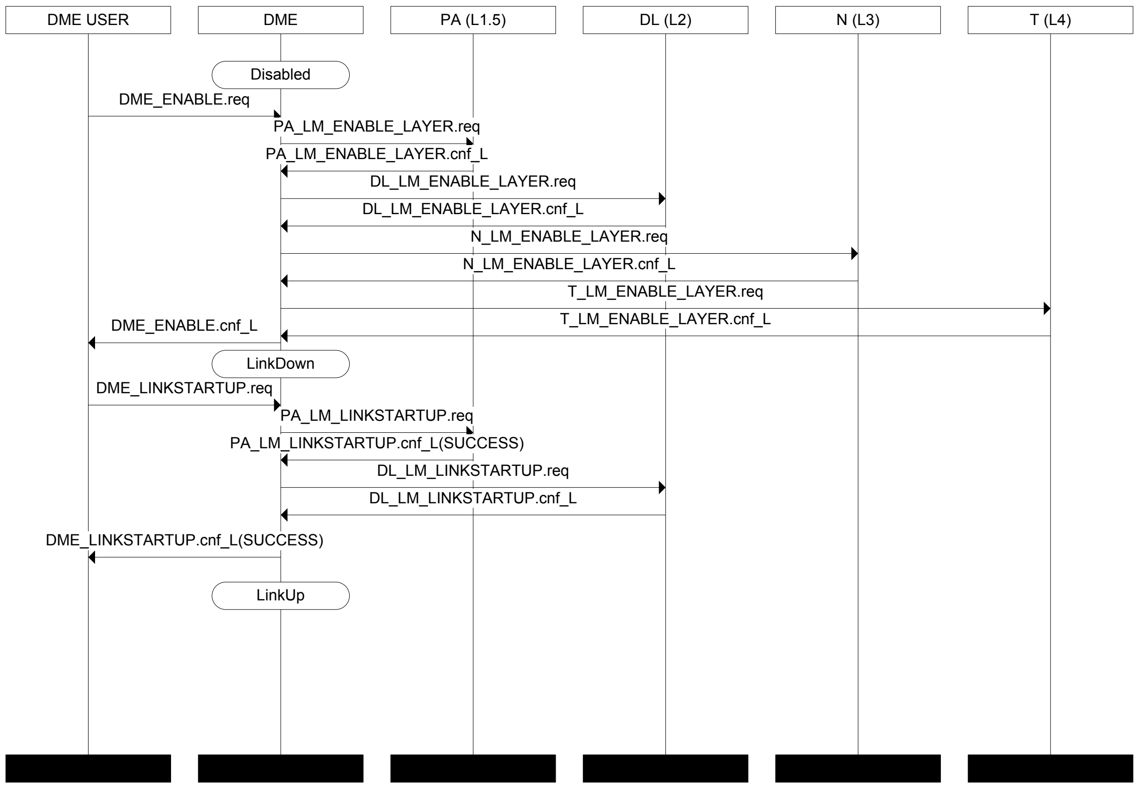
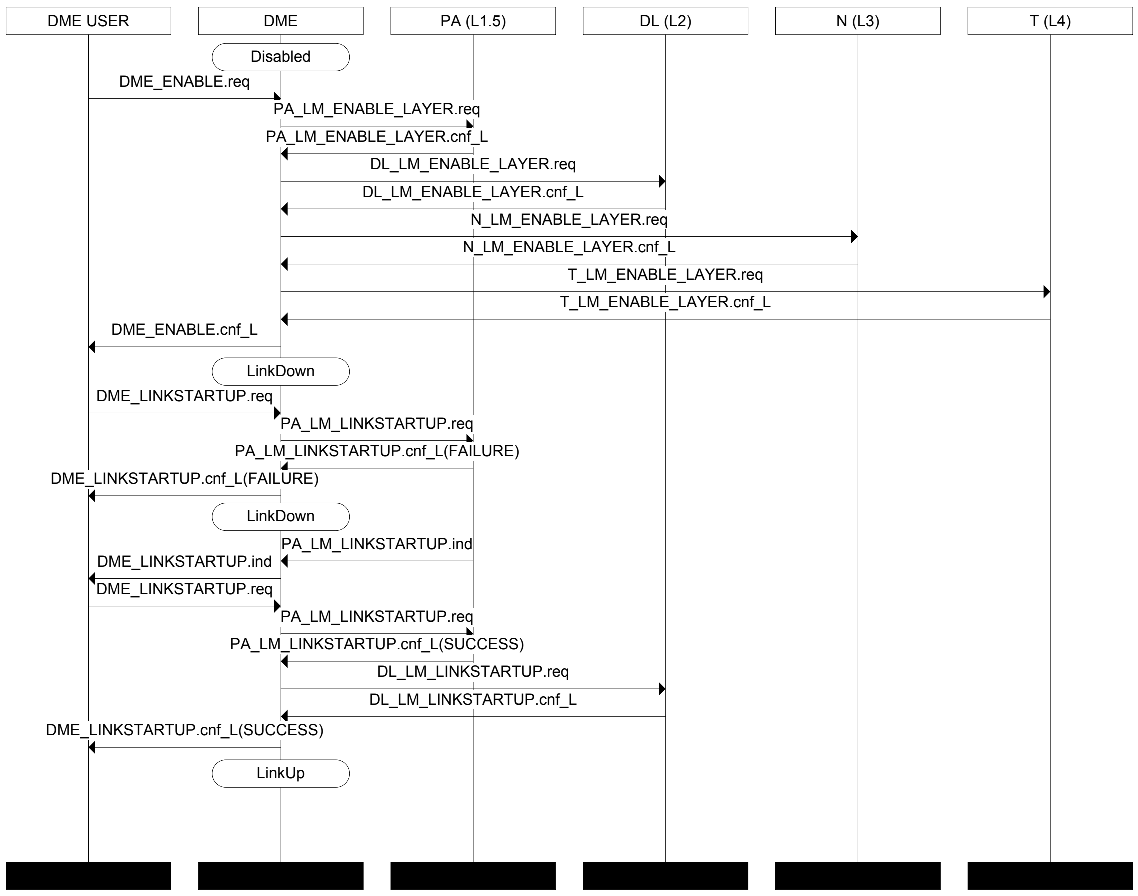


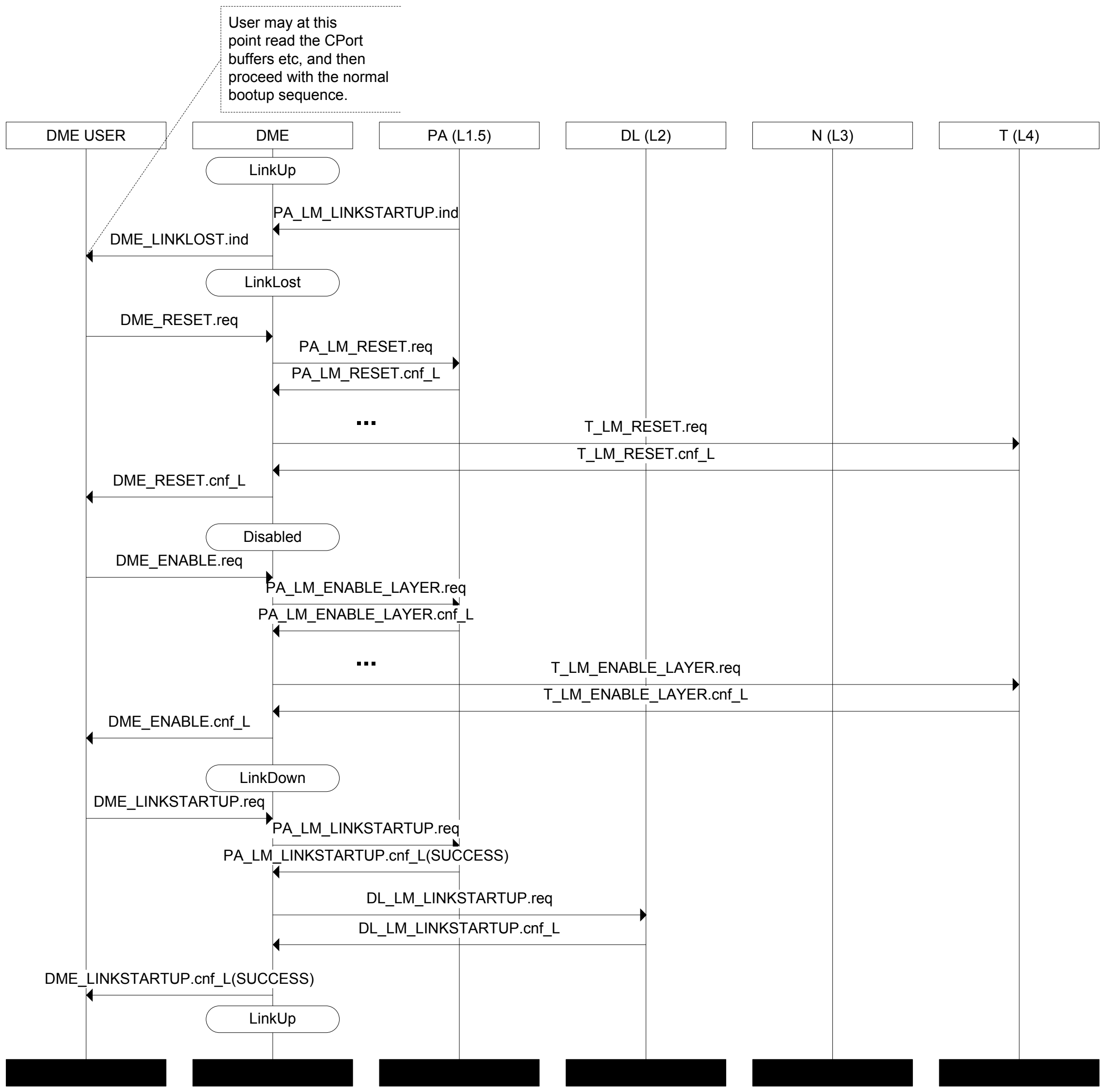
Figure 106 Default Boot Procedure

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential



6652

Figure 107 Peer Initiated Boot Procedure



6653

Figure 108 Link Lost Triggered Boot Procedure

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

9.12 DME DDB L1

Table 106 shows the DME DDB L1 Attributes, which are copies of the DME User's DDB L1 parameters. See ***[MIPI01]*** for the definition of these parameters.

All DME DDB L1 Attributes shall be gettable, and may be settable. Parameters should be settable if functionality sharing is desired.

If DME_DDBL1_Revision is read as zero, the DME_DDBL1 Attributes are unpredictable and invalid. If DME_DDBL1_Revision is read as a non-zero value, all DME_DDBL1 Attributes are valid and read-only.

At reset, each settable DME Attribute shall be reset to its corresponding static value, if one exists or to its Reset Value otherwise.

Table 106 DDB L1 Attributes

Attribute	Attribute ID[1]	Unit	Description	Reset Value
DME_DDBL1_Revision	0x5000	Int	The revision of the DDB Specification supported by this Device, encoded as major-version * 0x10 + minor-version The value shall be 0x10 (DDB v1.0)	0x00
DME_DDBL1_Level	0x5001	Int	Eight-bit field indicating the DDB support: b0: DDB Level 1 support. Shall be 0b1 b1: DDB Level 2 get support. Shall be 0b1 if, and only if, the GET-DDB-LEVEL2 Service is supported. If b1 is set to '1', then b0 shall also be set to '1' b2: DDB Level 2 set support. Shall be 0b1 if, and only if, the SET-DDB-LEVEL2 Service is supported. If b2 is set to '1', both b0 and b1 shall also be set to '1' b[7:3]: Reserved: Shall be 0b00000	0x1 (no support for DDB L2) Allow also the DDB L2 indications, with the limitation that it is up to the function-specific driver to fetch DDB L2.
DME_DDBL1_DeviceClass	0x5002	Int	The Device Class ID of the DME User, as specified by the MIPI Alliance If the Device does not conform to a specified Device class, the value shall be zero	N/A
DME_DDBL1_ManufacturerID	0x5003	Int	Manufacturer ID of the DME User, as specified by the MIPI Alliance	N/A
DME_DDBL1_ProductID	0x5004	Int	The Manufacturer's internal product ID of the DME User	N/A
DME_DDBL1_Length	0x5005	Int	The length of any DDB Level 2 data (see ***[MIPI01]***) For Devices supporting only DDB Level 1, the value shall be zero	N/A

1 Attributes 0x5000 to 0x5005 correspond to the DDB Level 1 fields and their offsets defined in ***[MIPI01]***.

The DME Attributes listed in ***Table 106*** should be retained during Hibernation. Those Attributes are simple and do not cause any implicit action when set.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

9.13 Quality of Service Monitoring

Typically, the UniPro stack manages to repair many potential Link problems without the need to notify the Application Layer. Repair mechanisms of the stack include L2 re-transmission and successful handling of PA_INIT.req. Nevertheless, any successful repair sequence requires Link bandwidth that is no longer available for the Application. Therefore, it may be useful for an Application to understand how often such repair attempts are made.

The DME implements Quality of Service monitoring using a simple counting scheme, counting error events and comparing them against the number of correctly received or transmitted bytes. When the error counter exceeds a programmed threshold before the byte counter overflows, a DME_QoS.ind is issued to the Application and both counters are reset. When the byte counter overflows before the error counter has reached the programmed threshold, both counters are reset without triggering a DME_QoS.ind.

The DME provides Link quality monitoring for the following purposes:

- Detection of re-occurring repaired fatal error conditions on the Link (PA_INIT loop). This kind of detection is useful if capabilities exchanged between local and peer permit a potential operation at a higher M-PHY Gear, but the physical interconnect between local and peer Device does not, or, after Line quality degradation, no longer satisfies channel characteristics defined in [MIPI02].
- Detection of degraded inbound or outbound Link quality, to allow an Application to issue an ADAPT sequence for a Link running in HS-G4. This kind of detection is used to monitor a slowly degrading Link quality, e.g. one being affected by temperature and voltage variations, against the expected M-PHY bit error rate.

Inbound and Outbound Links may be monitored from both peers of the Link. An Application is responsible for devising a QoS monitoring scheme that avoids conflicts resulting from local and peer Device monitoring of the same Link. Applications that define Link control from only one side of the Link should apply QoS monitoring from the Device that controls the Link's power modes. The implementation of a QoS monitoring is mandatory for any Device that actively controls the power mode of the Link.

QoS monitoring is only available during LinkUp and LinkCfg. During all other states, QoS monitoring is inactive. During Hibernate QoS status is retained, and QoS monitoring is resumed when exiting Hibernate.

9.13.1 PA_INIT Monitoring

The number of PA_INIT.req or PA_INIT.ind [as notified by DL_LM_ERROR.ind(PA_INIT)] is measured against the sum of received valid PHY symbols (as notified by PA_LM_RX_SYMBOL_CNT.ind, plus transmitted PHY symbols notified by PA_LM_TX_SYMBOL_CNT.ind and counted by the TXRX QoS counter).

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

9.13.2 Inbound Link Monitoring

The Link quality is measured by comparing the number of received valid PHY symbols (as notified by PA_LM_RX_SYMBOL_CNT.ind and counted by the RX QoS counter) with the number of detected DL Layer CRC errors [from DL_LM_ERROR.ind(CRC_ERROR)]. PACP frames are not factored into this error count, as PACP errors are already counted and can be monitored in the PA Layer.

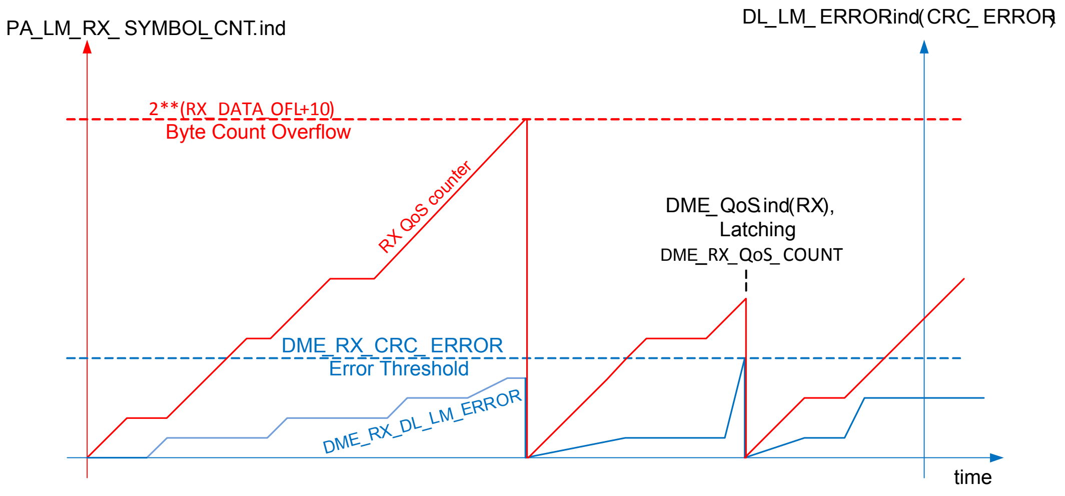


Figure 109 DME_QoS.ind Generation for Inbound Link Monitoring

9.13.3 Outbound Link Monitoring

Link quality is measured by comparing the number of transmitted PHY symbols (as notified by PA_LM_TX_SYMBOL_CNT.ind and counted by the TX QoS counter) with the number of received NAC errors [as notified by DL_LM_ERROR.ind(NAC_RECEIVED)]. The number of received L2 NAC errors is related in first approximation to the number of packet reception errors at the peer. PACP frames are not factored into this error count.

9.14 DME Attributes for QoS Measurement

Table 107 shows the DME QoS Attributes.

All DME QoS Attributes shall be gettable, and some shall be settable.

At Cold Reset or Warm Reset, each DME Attribute shall be reset to its corresponding static value, if one exists or to its Reset Value otherwise.

During Hibernate, all QoS Attributes shall be retained.

Table 107 DME QoS Attributes

Attribute	Attribute ID	Unit	Description	Reset Value
DME_TX_DATA_OFL (settable, gettable)	0x5100	Int	[5:0] TX_DATA_OFL Defines upper count range for TX QoS counter counting transmitted PHY symbols to 2**(TX_DATA_OFL+10) Valid range: 0 to 32	0x00
DME_TX_NAC_RECEIVED (settable, gettable)	0x5101	Int	[5:0] TX_NAC_RECEIVED Threshold for number of NAC_RECEIVED to trigger DME_QoS.ind(TX) during TX QoS counter counts from 0 to 2**(TX_DATA_OFL+10) Valid range: 0x01 to 0x3F	0x3F
DME_TX_QoS_COUNT (gettable)	0x5102	Int	[31:0] TX QoS counter value latched at time of last DME_QoS.ind(TX) Indicates the number of transmitted PHY symbols until QoS problem is flagged Resolution is 2**10 Symbols A value of 1 means that between 1024 and 2047 PHY symbols have been transmitted	0
DME_TX_DL_LM_ERROR (gettable)	0x5103	Int	[5:0] Actual counter value of DL_LM_ERROR.ind (NAC_RECEIVED) events	0x00
DME_RX_DATA_OFL (settable, gettable)	0x5110	Int	[5:0] RX_DATA_OFL Defines upper count range for RX QoS counter counting received valid PHY symbols to 2**(RX_DATA_OFL+10) Valid range: 0 to 32	0x00
DME_RX_CRC_ERROR (settable, gettable)	0x5111	Int	[5:0] RX_CRC_ERROR Threshold for number of CRC_ERRORs required to trigger DME_QoS.ind(RX) during RX QoS counter counts from 0 to RX_DATA_OFL Valid range: 0x01 to 0x3F	0x3F
DME_RX_QoS_COUNT (gettable)	0x5112	Int	[31:0] RX QoS counter value latched at time of last DME_QoS.ind(RX) Indicates the number of received PHY Symbols until QoS problem is flagged Resolution is 2**10 Symbols	0

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

			A value of 1 means that between 1024 and 2047 PHY symbols have been received	
DME_RX_DL_LM_ERROR (gettable)	0x5113	Int	[5:0] Actual counter value of DL_LM_ERROR.ind (CRC_ERROR) events	0x00
DME_TXRX_DATA_OFL (settable, gettable)	0x5120	Int	[5:0] TXRX_DATA_OFL Defines upper count range for TXRX counter counting the sum of transmitted and received symbols to 2**(TXRX_DATA_OFL+10) Valid range: 0 to 32	0x00
DME_TXRX_PA_INIT_REQUEST (settable, gettable)	0x5121	Int	[5:0] Threshold for number of PA_INIT requests required to trigger DME_QoS.ind(PA_INIT) during TXRX QoS counter counts from 0 to TXRX_DATA_OFL Valid range: 0x01 to 0x3F	0x3F
DME_TXRX_QoS_COUNT (gettable)	0x5122	Int	[31:0] TXRX QoS counter value latched at time of last DME_QoS.ind(PA_INIT) Indicates the sum of transmitted and received PHY symbols until QoS problem is flagged Resolution is 2**10 PHY Symbols	0
DME_TXRX_DL_LM_ERROR (gettable)	0x5123	Int	[5:0] Actual counter value of DL_LM_ERROR.ind (PA_INIT) events	0x00
DME_QoS_ENABLE (settable, gettable)	0x5130	3-bit word	[0]: TX QoS monitoring enabled [1]: RX QoS monitoring enabled [2]: PA_INIT monitoring enabled Only enabled bits shall generate a DME_QoS.ind QoS counters are reset when the corresponding enable bit is set to b0	0x00
DME_QoS_STATUS (gettable)	0x5131	3-bit word	[0]:DME_QoS.ind(TX) detected [1]:DME_QoS.ind(RX) detected [2]:DME_QoS.ind(PA_INIT) detected Reading the QoS_Status resets this attribute	0x00

6715

This page intentionally left blank.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Annex A Lane Management and Link Errors

The following sections depict informative examples for the Lane Management of an attached M-Port in case of a Link Error. It is assumed that two Lanes are connected in every direction of the Link. The nomenclature of the PACP_PWR_req shown in all diagram is defined as

PACP_PWR_req (X,n;Y,m)

where

X represents the new Link Gear, Mode, Termination, etc. request of the PowerMode change in the outbound direction,

n is the number of Lanes in outbound direction,

Y is the Link Gear, Mode, Termination, etc. request of the PowerMode change in the inbound direction, and

m is the number of Lanes in inbound direction.

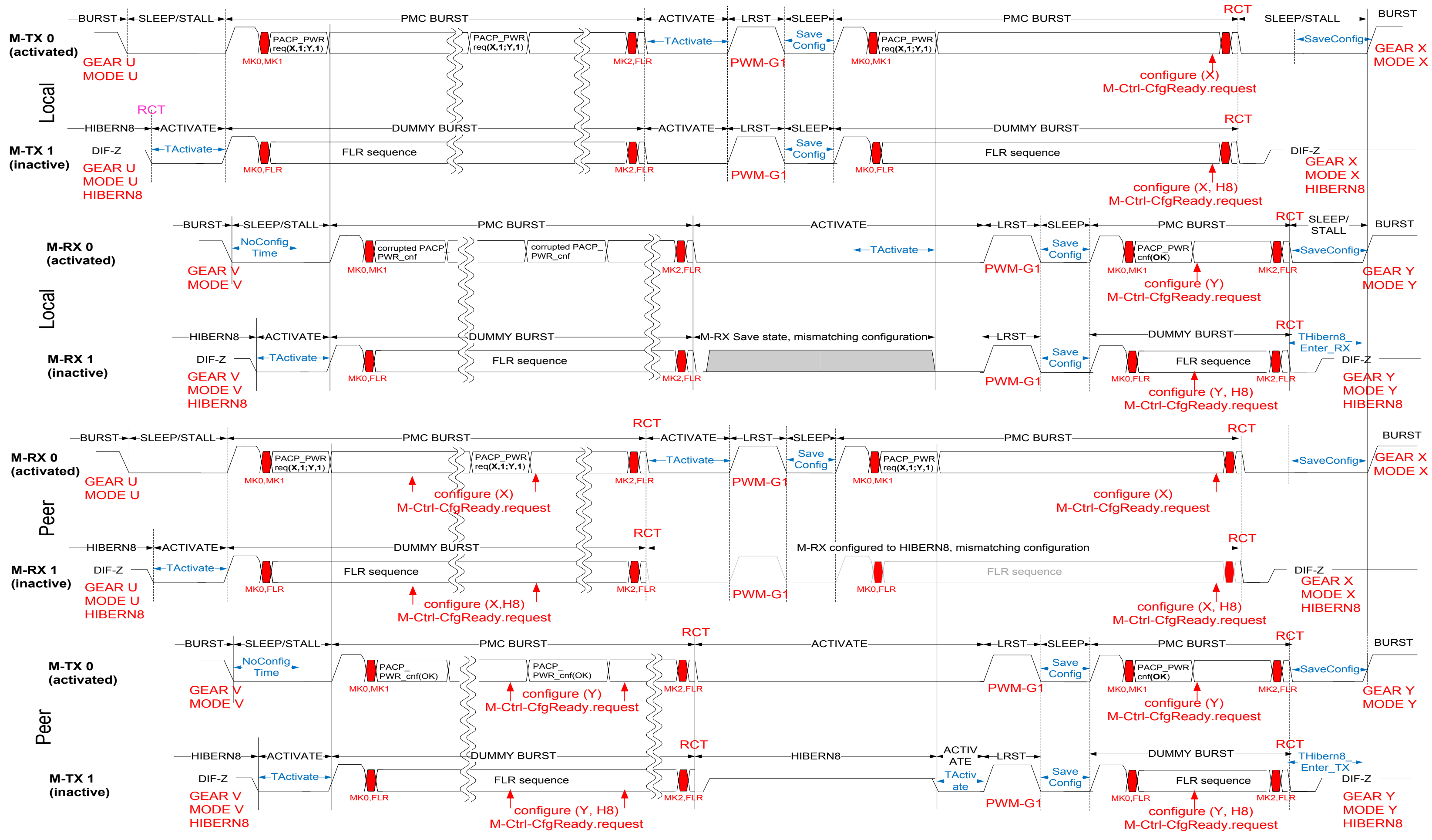
Please note that for ***Figure 116***, the Gear and Mode parameters and the number of Lanes set by PACP_PWR_req are identical to the initial Link state before PA_INIT.

In all demonstrated cases, the assumed underlying Link Error is failed reception of a PACP_PWR_cnf frame from the peer.

After a first M-PHY Burst with a failing PACP_PWR_req or PACP_PWR_cnf exchange, the local M-TX and the peer M-RX of a Link may leave the Burst being configured into different Modes or/and Gears. The subsequent Line-Reset exchange aims to break up that deadlock and repair the link. However, since M-PHY only responds to Line-Reset during M-PHY ACTIVE states, and not during HIBERN8, there is no guarantee that the Link can be recovered. The ability to recover depends in general on the robustness of the peer M-RX implementation to respond to unmatched local M-TX line states or unexpected M-TX Line signaling.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

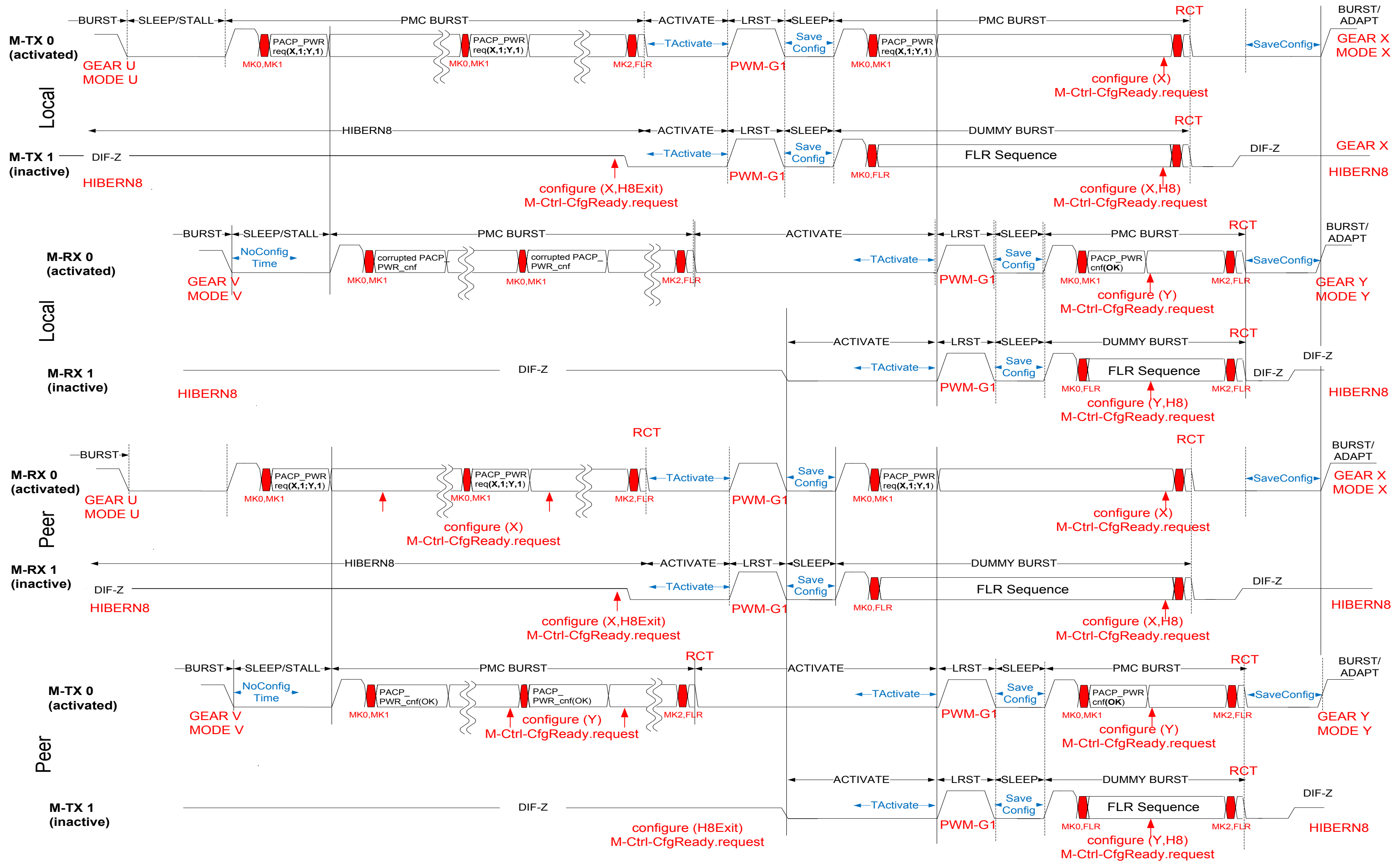
A.1 PowerMode Change



6737 **Figure 110 Single Lane PACP_PWR Exchange Error Recovery (UniPro v1.8 <-> v1.61)**

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

6738



6739

Figure 111 Single Lane PACP_PWR Exchange Error Recovery (UniPro v1.8 <-> v1.8)

6740

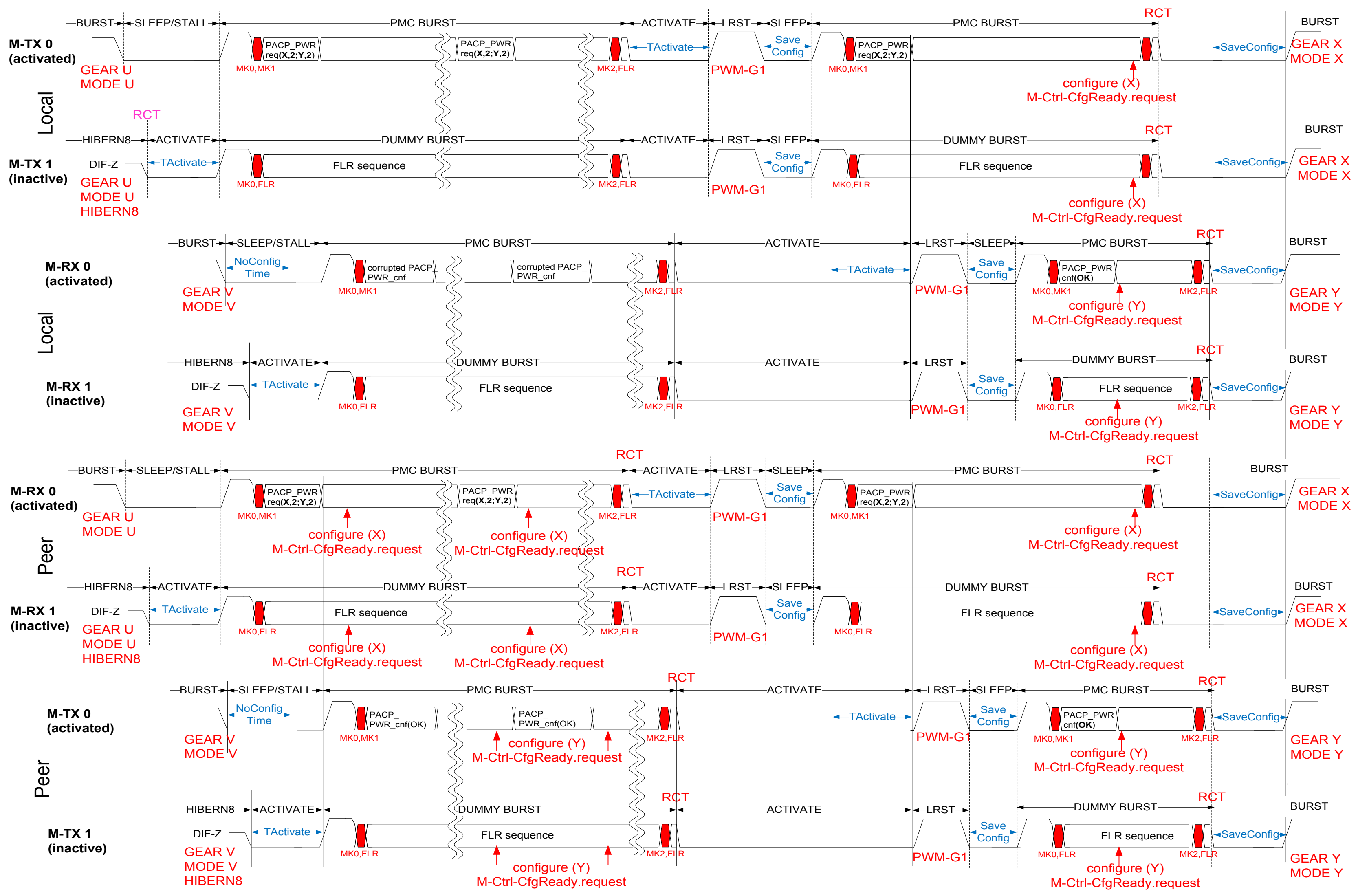


Figure 112 Single Lane to Dual Lane PACP_PWR Exchange Error Recovery (UniPro v1.8 <-> v1.61)

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

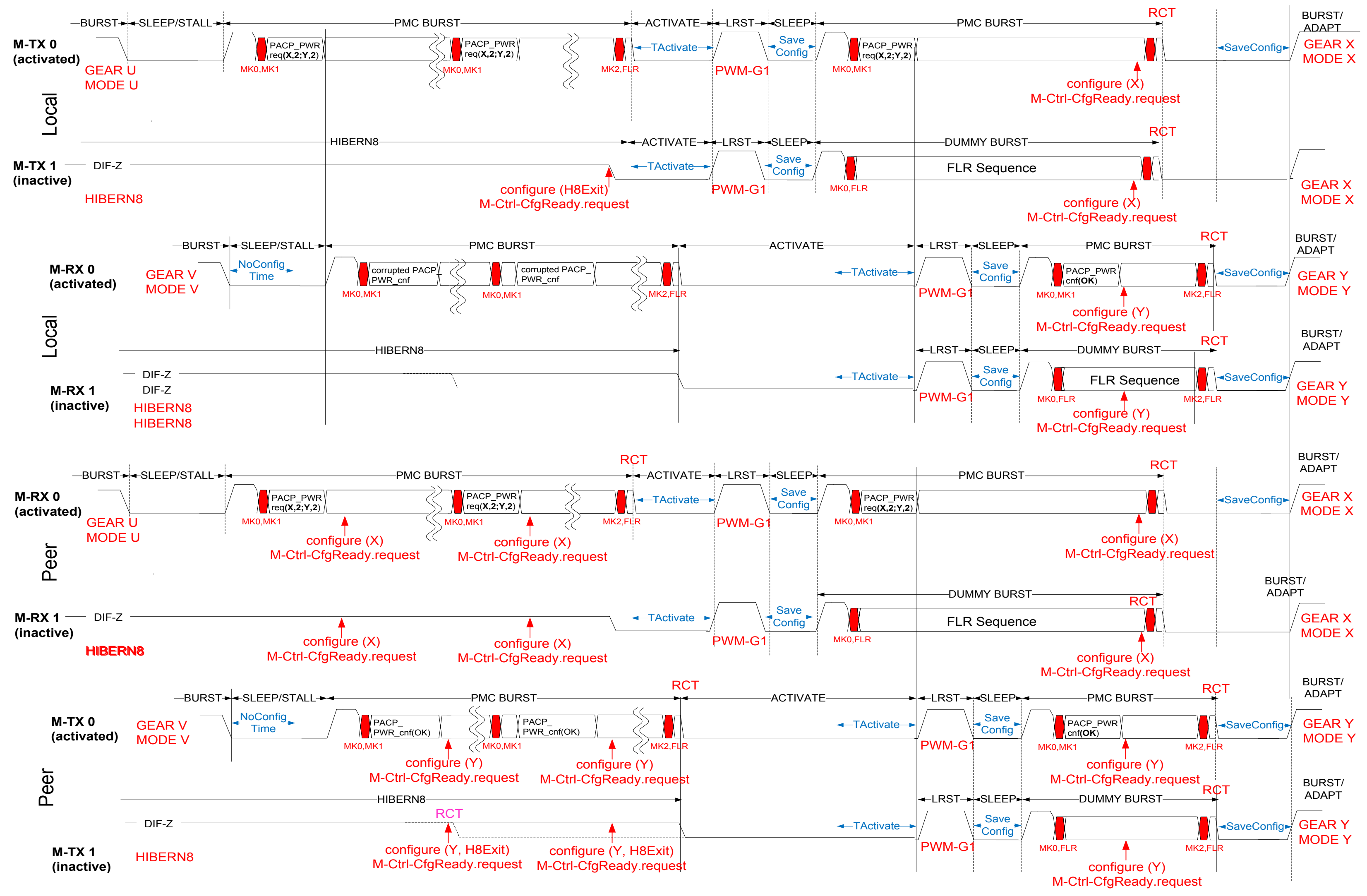


Figure 113 Single Lane to Dual Lane PACP_PWR Exchange Error Recovery (UniPro v1.8 <-> v1.8)

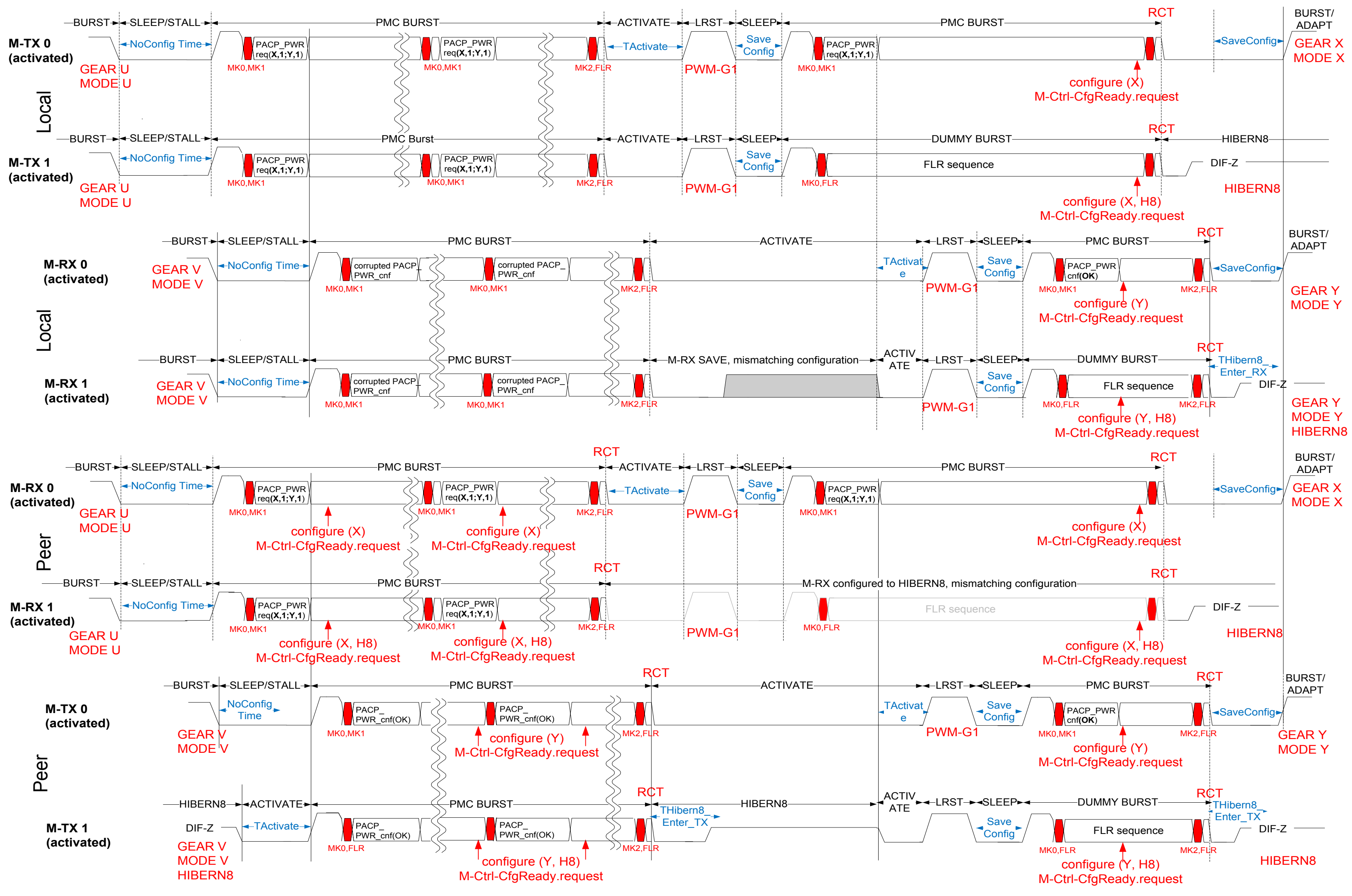


Figure 114 Dual to Single Lane PACP_PWR Exchange Error Recovery (UniPro v1.8 <-> v1.8)

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

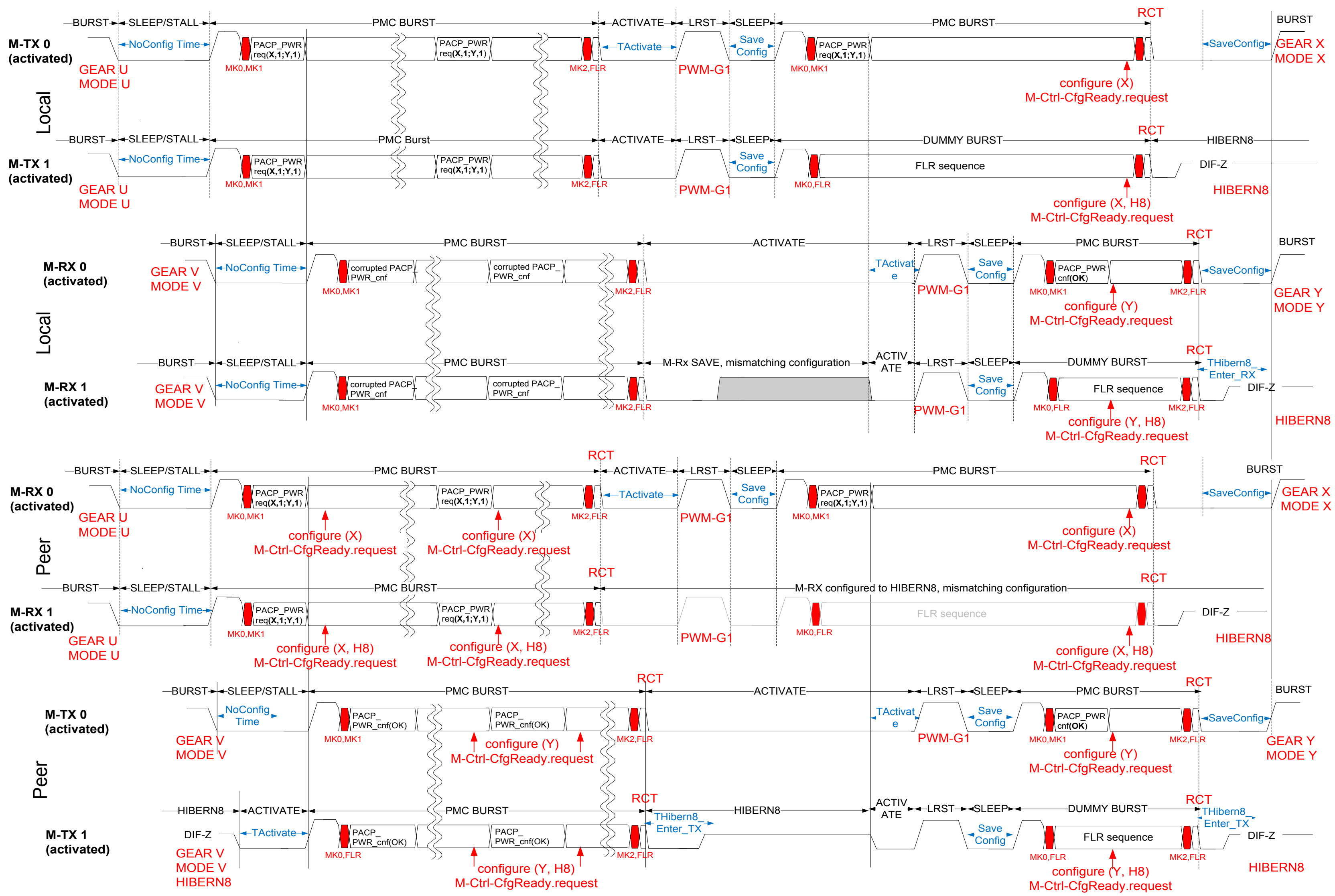
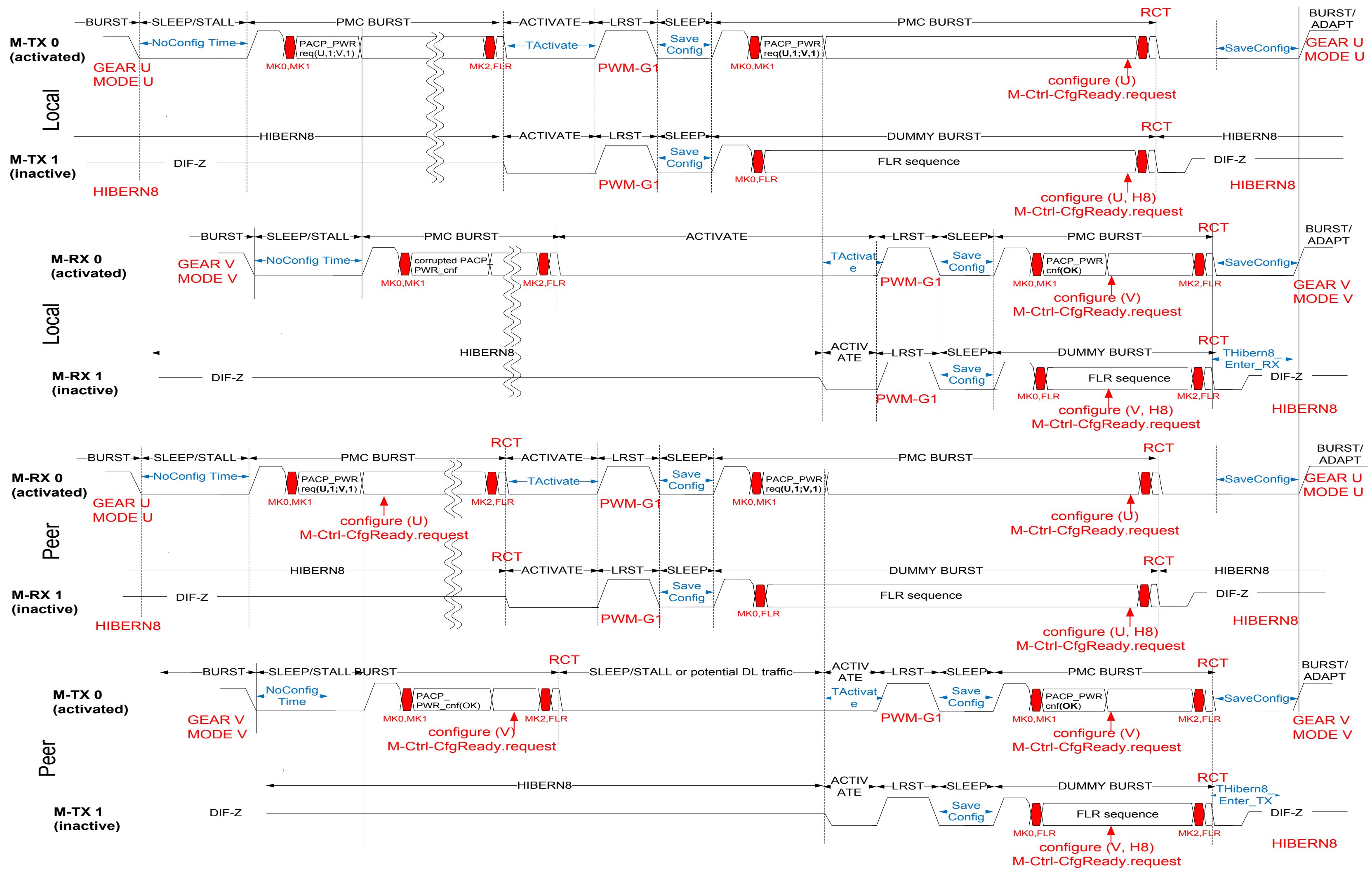


Figure 115 Dual to Single Lane PACP_PWR Exchange Error Recovery (UniPro v1.8 <-> v1.61)

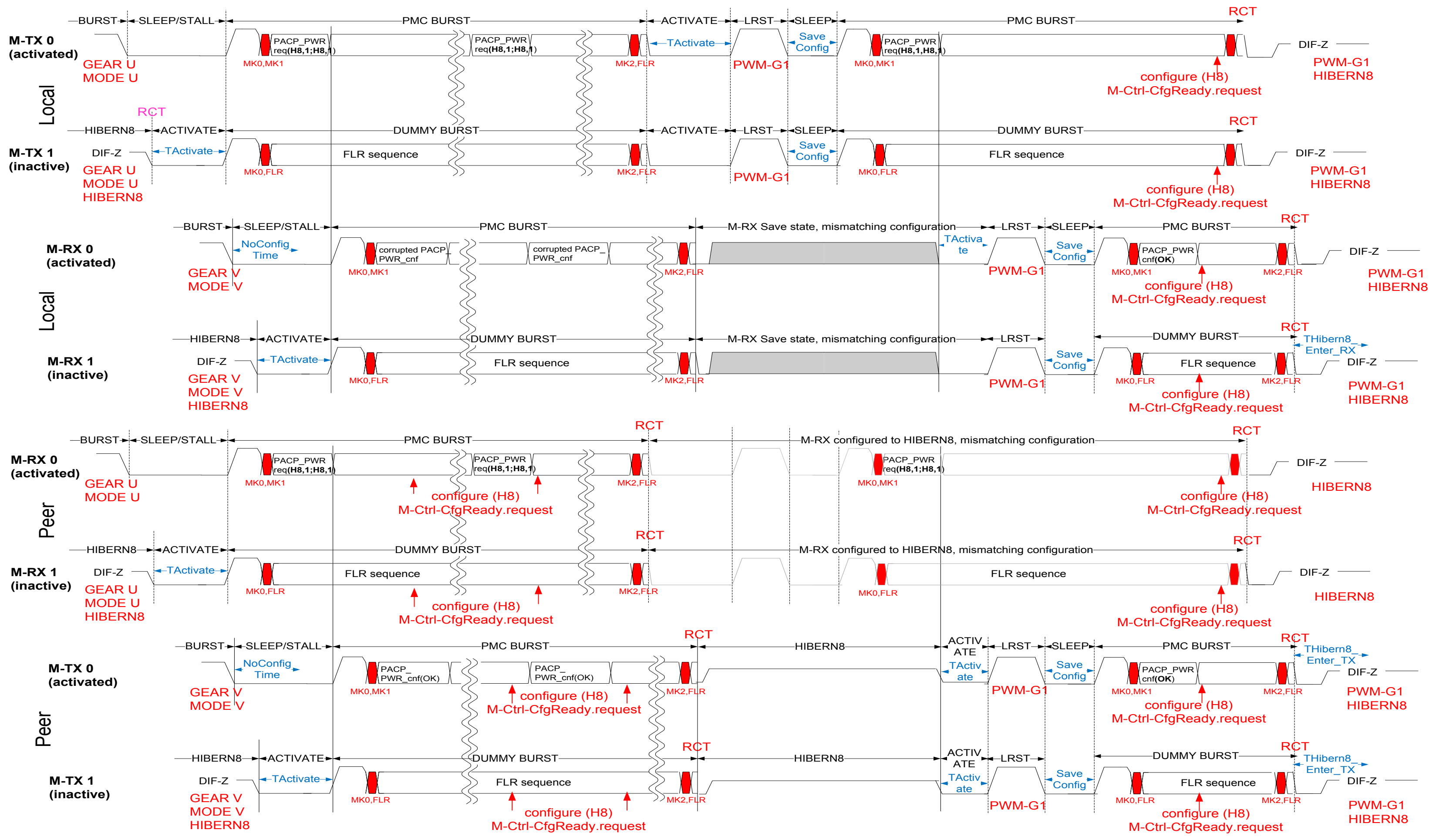


6744

Figure 116 PA_INIT Single Lane (UniPro v1.8 <-> v1.8)

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

A.2 Hibernate



6745

Figure 117 Single Lane HIBERNATE Error Recovery (UniPro v1.8 <-> v1.61)

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

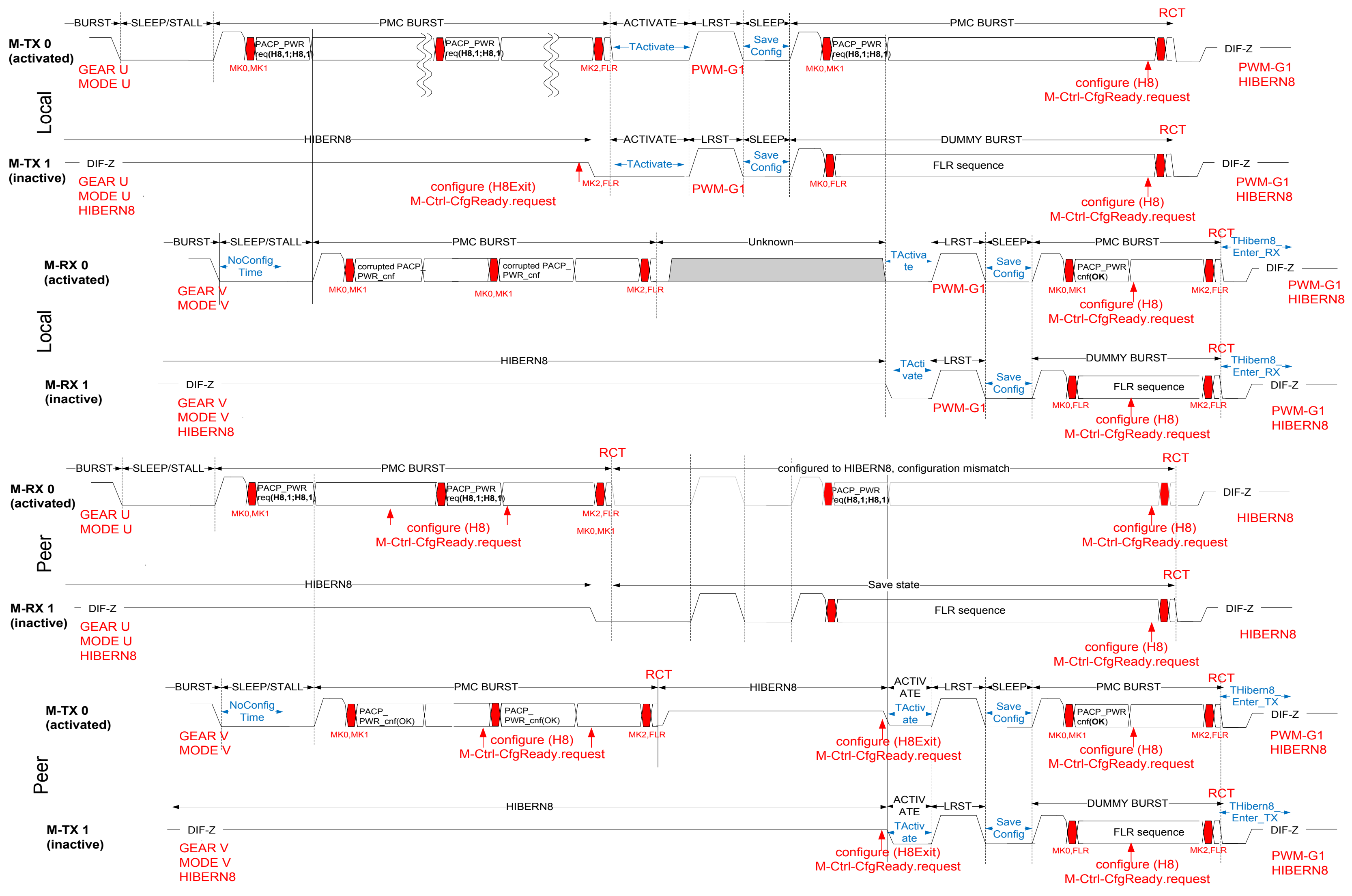


Figure 118 Single Lane HIBERNATE Error Recovery (UniPro v1.8 <-> v1.8)

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Annex B Data Link Layer (informative)

B.1 Reliability Scenarios

B.1.1 Timeout Mechanism

The TCx_REPLAY_TIMER operation is illustrated in ***Figure 119*** for Traffic Class 0 without grouped acknowledgment. (DL_TC0OutAckThreshold set to zero).

- Here, local TX uses Traffic Class 0 for transmitting a Frame with Frame Sequence Number 0 (Frame#0) to peer RX and TC0_REPLAY_TIMER is started after sending the last symbol of Frame#0 (CRC symbol).
- Peer RX receives the Frame#0 and triggers peer TX to send AFC0#0 for Frame#0.
- Local RX receives the AFC0#0 Frame.
- TC0_REPLAY_TIMER is in running mode until the AFC0#0 is received (without an error). The successful reception of AFC0#0 Frame resets the TC0_REPLAY_TIMER. Then the timer is in not running state (stop state) as there are no pending acknowledgments.

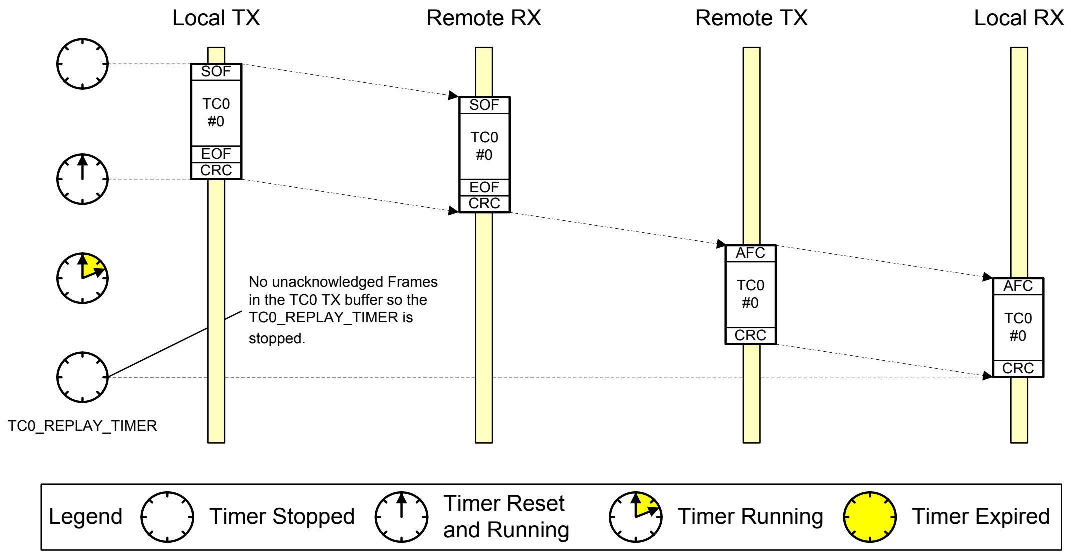


Figure 119 TC0 Replay Timer Operation for Simple ACK (No Grouping ACK)

The TC0_REPLAY_TIMER behavior for grouped acknowledgment is shown in ***Figure 120***. Traffic Class 0 is used for illustration purposes though the same behavior is applicable to other Traffic Classes.

- TC0_REPLAY_TIMER is reset and started after sending the last symbol of Frame#0.
- While Frame #1 is being transmitted, TC0_REPLAY_TIMER is allowed to run.
- TC0_REPLAY_TIMER is reset and allowed to run after sending the last symbol of Frame#1.
- While Frame #2 is being transmitted, TC0_REPLAY_TIMER is allowed to run.
- TC0_REPLAY_TIMER is reset and allowed to run after sending the last symbol of Frame#2.
- Successful reception of AFC0#1 Frame (grouped acknowledgment for Frame#0 and Frame#1) TC0_REPLAY_TIMER is reset and allowed to run since acknowledgment for Frame#2 is not yet received.
- Reception of AFC0#2 Frame by local RX resets and stops TC0_REPLAY_TIMER as there are no unacknowledged Frames.

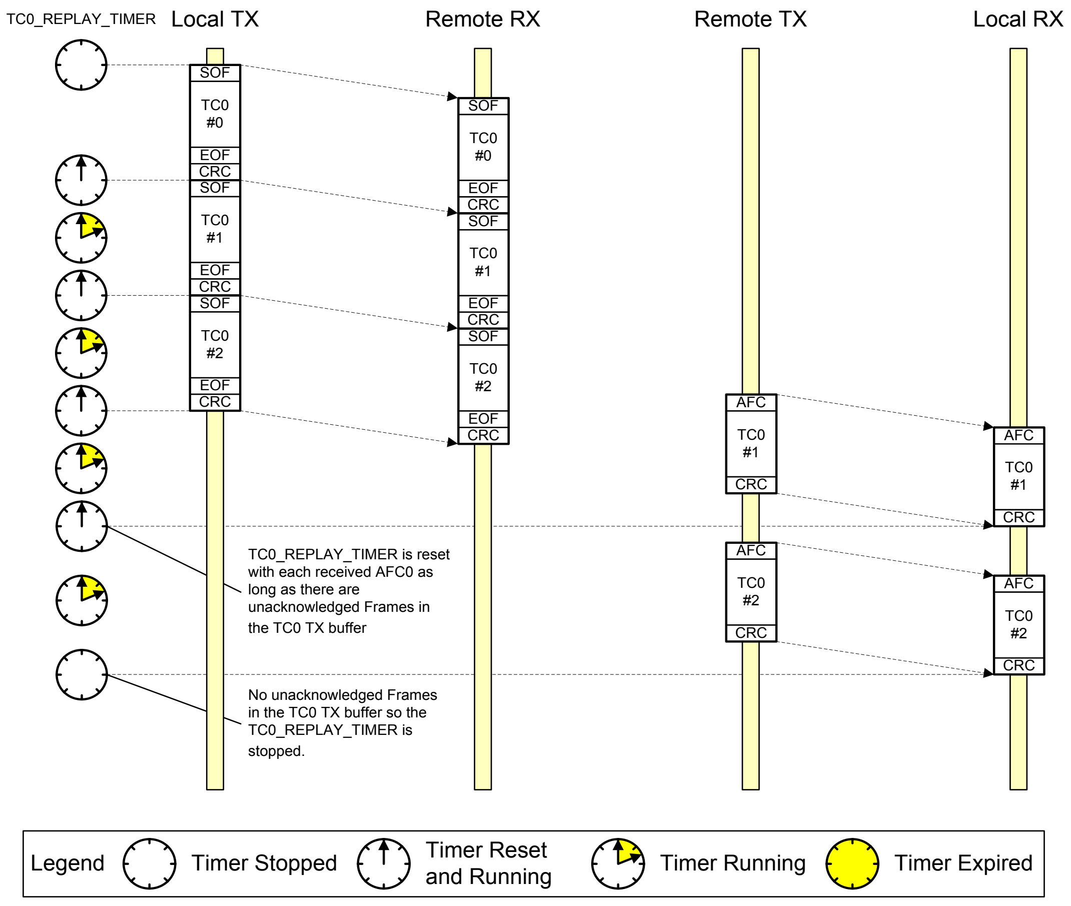

TC0_REPLAY_TIMER
Local TX
Remote RX
Remote TX
Local RX
SOF
TC0 #0
EOF
CRC
TC0 #1
TC0 #2
AFC
TC0_REPLAY_TIMER is reset with each received AFC0 as long as there are unacknowledged Frames in the TC0 TX buffer
No unacknowledged Frames in the TC0 TX buffer so the TC0_REPLAY_TIMER is stopped.
Legend
Timer Stopped
Timer Reset and Running
Timer Running
Timer Expired

Figure 120 TC0 Replay Timer Operation for Grouped ACK

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Figure 121 shows TCx_REPLAY_TIMER behavior in case of preemption (TC0 is preempted by TC1).

- TC0_REPLAY_TIMER is reset and started after sending the last symbol of TC0 Frame#0.
- While TC0 Frame #1 is being transmitted, TC0_REPLAY_TIMER is allowed to run.
- The local TX preempts TC0 Frame #1 with a higher priority Frame, TC1 Frame #0.
- Since TC0 Frame #1 has not been acknowledged, TC0_REPLAY_TIMER continues to run.
- TC1_REPLAY_TIMER is reset and allowed to run after sending the last symbol of TC1 Frame#0.
- Since TC1 has no more Frames to send, TC0 resumes transmitting TC0 Frame#1 from its preempted position.
- TC0_REPLAY_TIMER is reset and allowed to run after sending the last symbol of TC0 Frame#1.
- After successful reception of AFC1#0 Frame by the local RX, TC1_REPLAY_TIMER is stopped as there are no unacknowledged Frames for TC1.
- After successful reception of AFC0#1 Frame by the local RX, TC0_REPLAY_TIMER is stopped as there are no unacknowledged Frames for TC0.

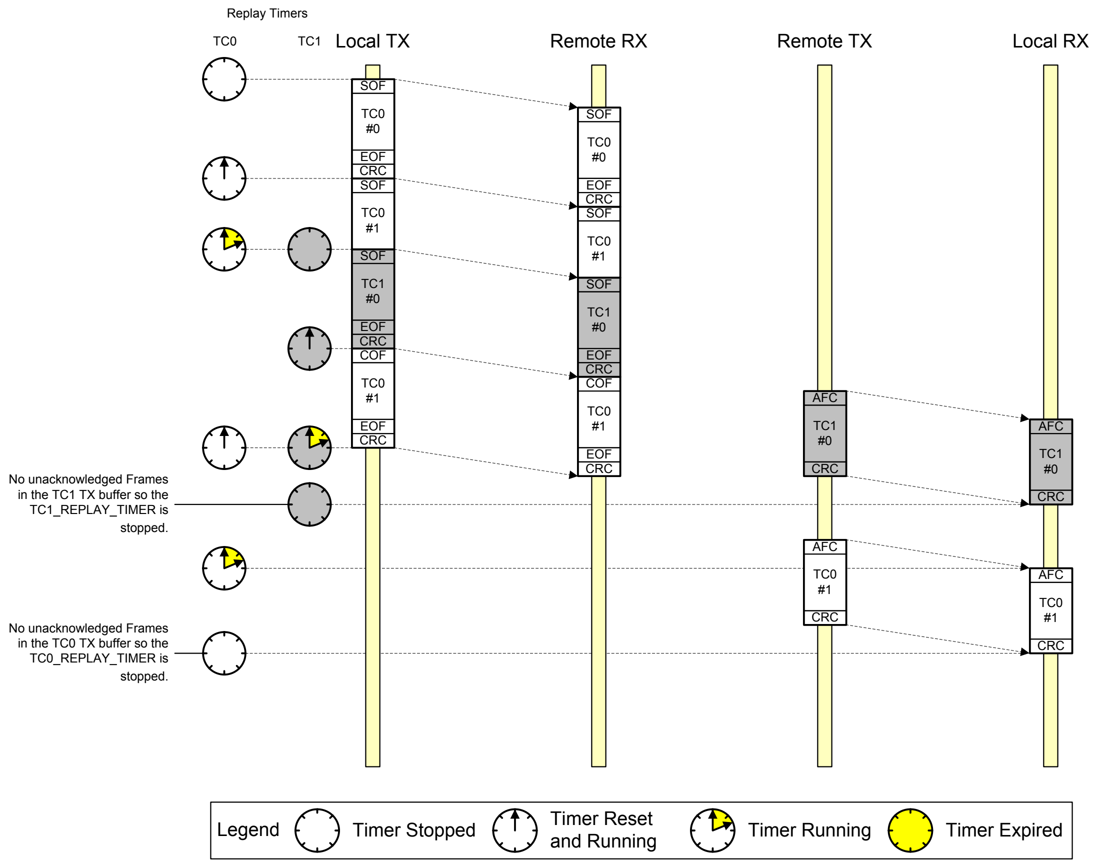


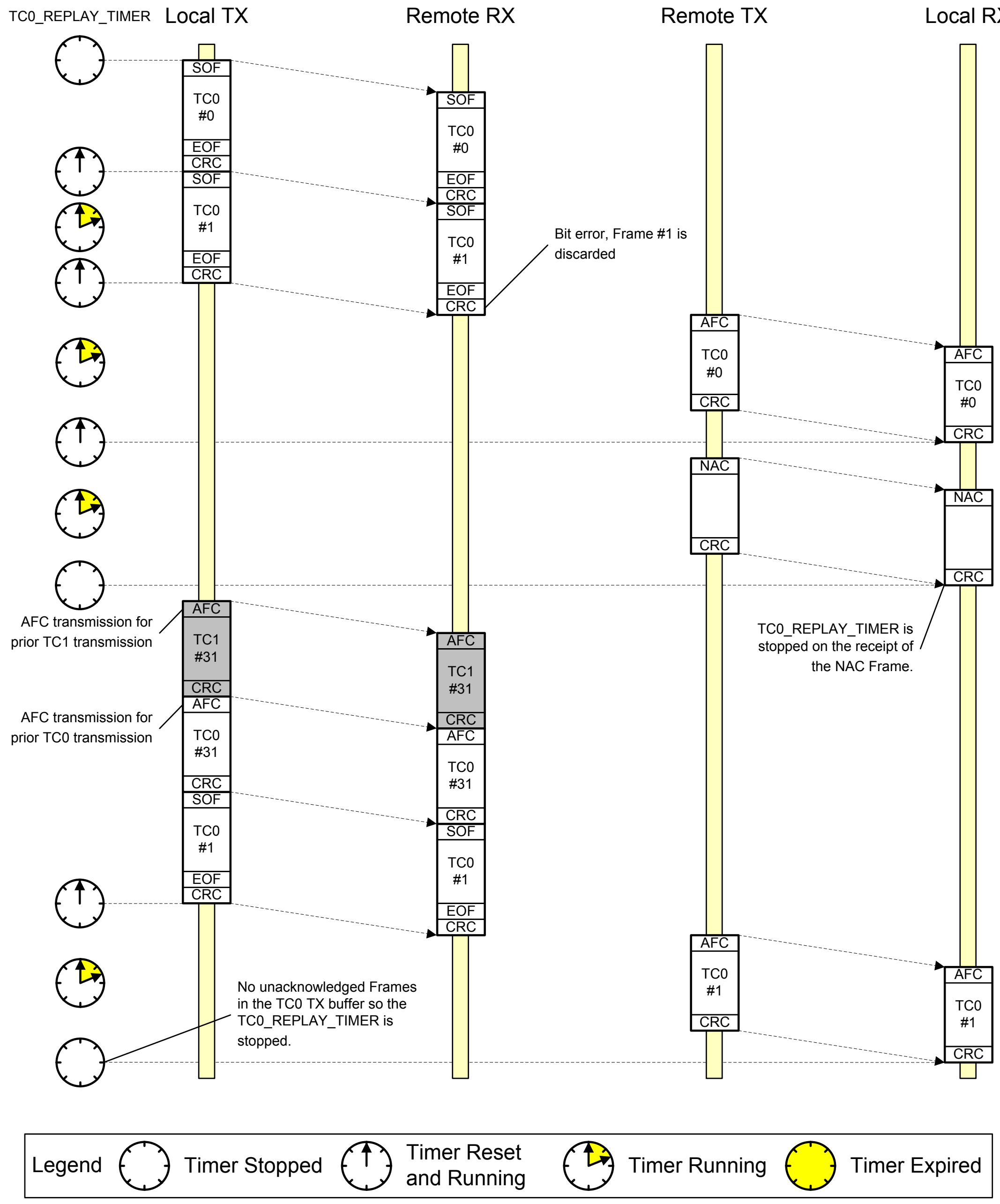
Figure 121 TCx Replay Timer Behavior During Preemption

B.1.2 Retransmission Mechanism

Figure 122 illustrates retransmission due to the received NAC Frame by local RX, where TC0 and TC1 Frames are depicted respectively in white and gray color.

- The local TX sends TC0 Frame#0 and Frame#1. TC0_REPLAY_TIMER is reset and started after sending the last symbol of Frame#1. The Forward Link carries no traffic from the local TX after sending TC0 Frame#1.
- The peer RX receives Frame#0 in good condition, and Frame#1 with an error (detected during CRC checking). It discards the last Frame, i.e. Frame#1, and schedules an AFC Frame to acknowledge TC0 Frame#0, and a NAC Frame with the RReq bit set to '0' (see ***6.5.4.2*** for NAC Frame details), which acknowledges correct reception of TC0 Frame#0.
- After reception of the AFC0 Frame by the local RX, the local RX resets and starts the TC0_REPLAY_TIMER
- After reception of the NAC Frame by the local RX, the local RX resets and freezes the TC0_REPLAY_TIMER, transmits AFC Frames for TC1 and TC0 and retransmits TC0 Frame#1. After retransmitting TC0 Frame#1 it resets and starts TC0_REPLAY_TIMER.
- Peer RX receives TC0 Frame#1 in good condition, this time, and sends an acknowledgment for the Frame with AFC0#1 Frame.
- The local RX detects AFC0#1 Frame and stops the TC0_REPLAY_TIMER as there are no pending acknowledgments.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential


TC0_REPLAY_TIMER
Local TX
Remote RX
Remote TX
Local RX
SOF
TC0 #0
EOF
CRC
TC0 #1
Bit error, Frame #1 is discarded
AFC
NAC
AFC transmission for prior TC1 transmission
TC1 #31
AFC transmission for prior TC0 transmission
TC0 #31
TC0_REPLAY_TIMER is stopped on the receipt of the NAC Frame.
No unacknowledged Frames in the TC0 TX buffer so the TC0_REPLAY_TIMER is stopped.
Legend
Timer Stopped
Timer Reset and Running
Timer Running
Timer Expired

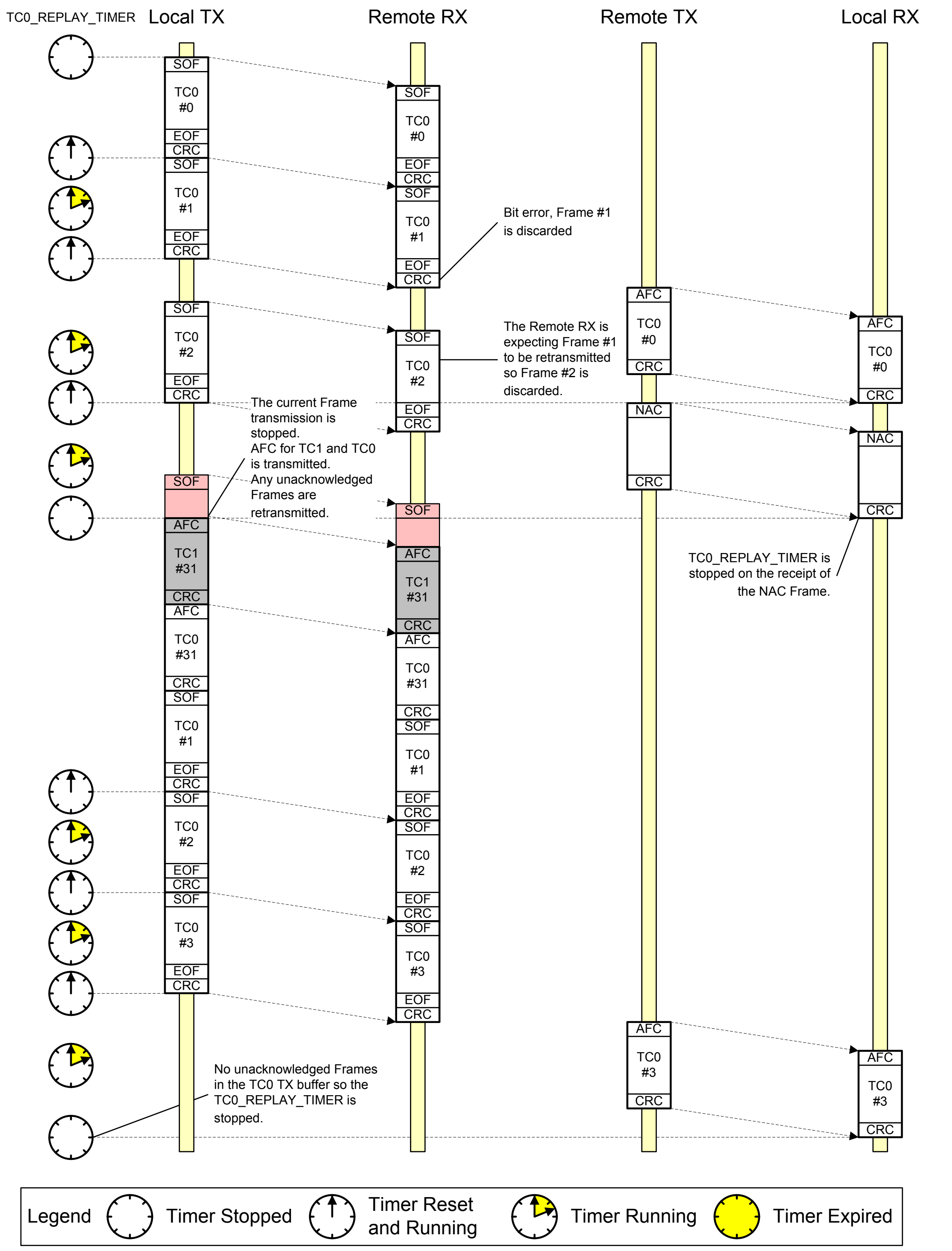
6803

Figure 122 Retransmission Triggered by NAC Frame

Figure 123 shows a scenario with two Traffic Classes, and depicts the case when a NAC Frame is received by the local RX due to bad Frame reception by the peer RX while the local TX is sending a Data Frame on the forward Link.

- The local TX sends TC0 Frame#0, Frame#1, Frame#2 and Frame#3. TC0_REPLAY_TIMER is reset and started after finalization of each Frame
- The peer RX receives Frame#0 in good condition, and Frame#1 with an error (detected during CRC checking). It schedules the transmission of the AFC#1 Frame and a NAC Frame with the RReq bit set to '0' and discards all TC0 Data Frames until it receives TC0 Frame#1 error free.
- The reception of NAC Frame by the local RX stops the TC0 Frame #3 transmission (without concluding the current Frame).
- Local TX transmits AFC Frames for TC1 and TC0 and starts retransmission of all unacknowledged Data Frames in the priority order. In this illustration only TC0 traffic is depicted, hence it starts sending TC0 Frames from the oldest unacknowledged Frame (TC0 Frame#1) to the latest Frame (TC0 Frame#3) available in buffer.
- Peer RX receives the AFC1 Frame in good condition and activates the NAC Frame generation. See ***Section 6.6.10*** for more details.
- Peer RX receives TC0 Frame#2 and TC0 Frame#3 in good condition and sends an acknowledgment with AFC0#3 Frame.
- The local RX detects the AFC0#3 Frame and stops the TC0_REPLAY_TIMER as there are no pending acknowledgments.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential


TC0_REPLAY_TIMER
Local TX
Remote RX
Remote TX
Local RX
SOF
TC0 #0
EOF
CRC
TC0 #1
TC0 #2
Bit error, Frame #1 is discarded
The Remote RX is expecting Frame #1 to be retransmitted so Frame #2 is discarded.
AFC
NAC
The current Frame transmission is stopped. AFC for TC1 and TC0 is transmitted. Any unacknowledged Frames are retransmitted.
TC1 #31
TC0 #31
TC0_REPLAY_TIMER is stopped on the receipt of the NAC Frame.
TC0 #3
No unacknowledged Frames in the TC0 TX buffer so the TC0_REPLAY_TIMER is stopped.
Legend
Timer Stopped
Timer Reset and Running
Timer Running
Timer Expired

6824

Figure 123 Retransmission Triggered by NAC Frame while Stopping Current Frame

Figure 124 depicts the case where retransmission is triggered by expiration of TC0_REPLAY_TIMER.

- After sending TC0 Frame#16 the TC0_REPLAY_TIMER is reset and started waiting for an AFC0 Frame from peer TX by the local RX.
- The peer RX could not detect an EOF_EVEN or EOF_ODD symbol and CRC symbol, so it did not send an AFC0 Frame.
- The local TX TC0_REPLAY_TIMER expires.
- The local TX triggers PHY initialization (not shown in the figure).
- After reception of PA_INIT.cnf_L with PAReverseLinkInitialized set to TRUE (not shown in the figure), the local TX transmits NAC Frame with the RReq bit set to ‘0’.
- The local TX starts transmission first with the AFC Frames for TC1 and TC0 and then retransmits TC0 Frame#16 as there are no unacknowledged TC1 Data Frames.
- The peer TX transmits AFC Frames for TC1 and TC0. It has no unacknowledged Data Frames for TC1 and TC0 to send.
- The peer RX receives Frame#16.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

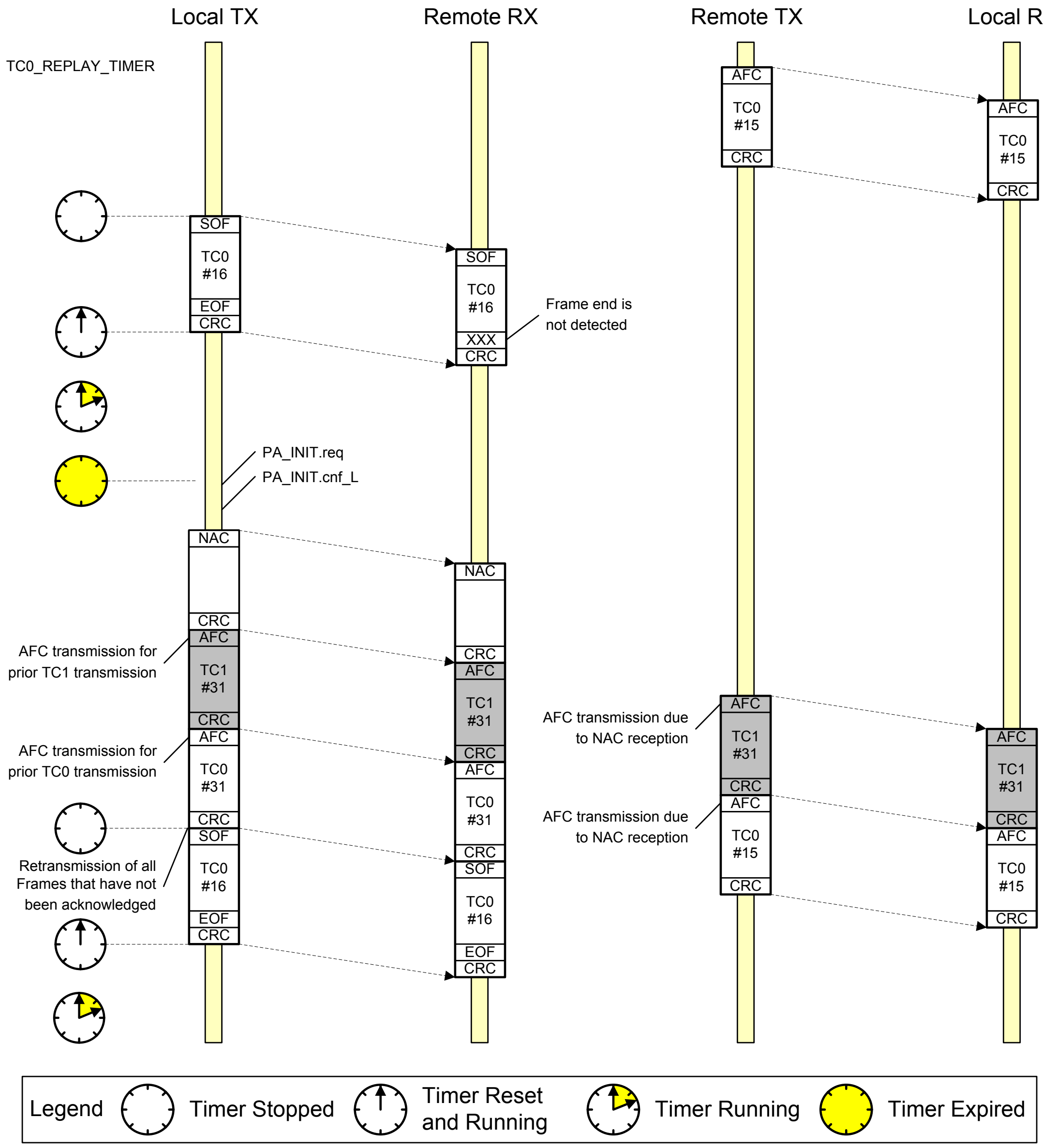
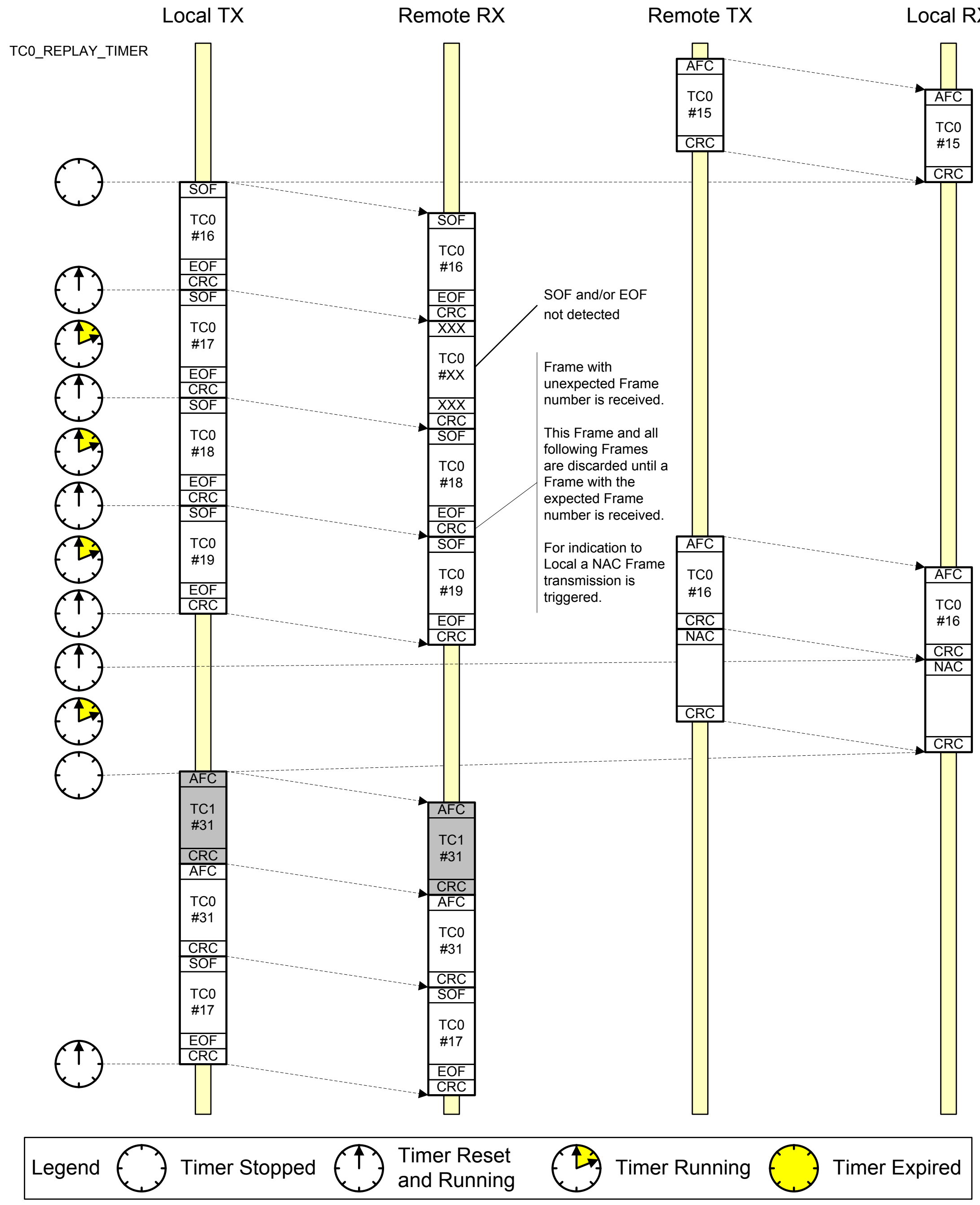

Local TX
Remote RX
Remote TX
Local RX
TC0_REPLAY_TIMER
AFC
TC0 #15
CRC
SOF
TC0 #16
EOF
CRC
XXX
Frame end is not detected
PA_INIT.req
PA_INIT.cnf_L
NAC
AFC transmission for prior TC1 transmission
TC1 #31
AFC transmission for prior TC0 transmission
TC0 #31
Retransmission of all Frames that have not been acknowledged
AFC transmission due to NAC reception
Legend
Timer Stopped
Timer Reset and Running
Timer Running
Timer Expired

Figure 124 Retransmission Triggered by TC0_REPLAY_TIMER Expiration

Figure 125 depicts the scenario where the retransmission is caused by a wrong Frame Sequence Number.

- The local TX transmits TC0 Data Frames starting from Frame #16. An acknowledgment is received for all Frames up to Frame#15 (sending of these Frames are not shown in the figure) that stops TC0_REPLAY_TIMER before Frame#16 is transmitted.
- The peer RX detects Frame#16 and Frame#18 but could not detect Frame#17. It recognizes the wrong Frame Sequence Number when it received Frame#18 (while Frame#17 was expected).
- The peer RX discards all TC0 Data Frames from Frame#18 until it receives TC0 Frame#17 and peer TX sends an AFC0#16 Frame and a NAC Frame with the RReq bit set to ‘0’.
- The local RX receives the AFC0#16 and the NAC Frame, transmits AFC Frames for TC1 and TC0 then starts retransmission of TC0 Frame#17 and the Frames that follow it (not shown in the figure).


Local TX
Remote RX
Remote TX
Local RX
TC0_REPLAY_TIMER
AFC
TC0 #15
CRC
SOF
TC0 #16
EOF
CRC
TC0 #17
TC0 #18
TC0 #19
XXX
TC0 #XX
SOF and/or EOF not detected
Frame with unexpected Frame number is received.
This Frame and all following Frames are discarded until a Frame with the expected Frame number is received.
For indication to Local a NAC Frame transmission is triggered.
NAC
TC1 #31
TC0 #31
Legend
Timer Stopped
Timer Reset and Running
Timer Running
Timer Expired

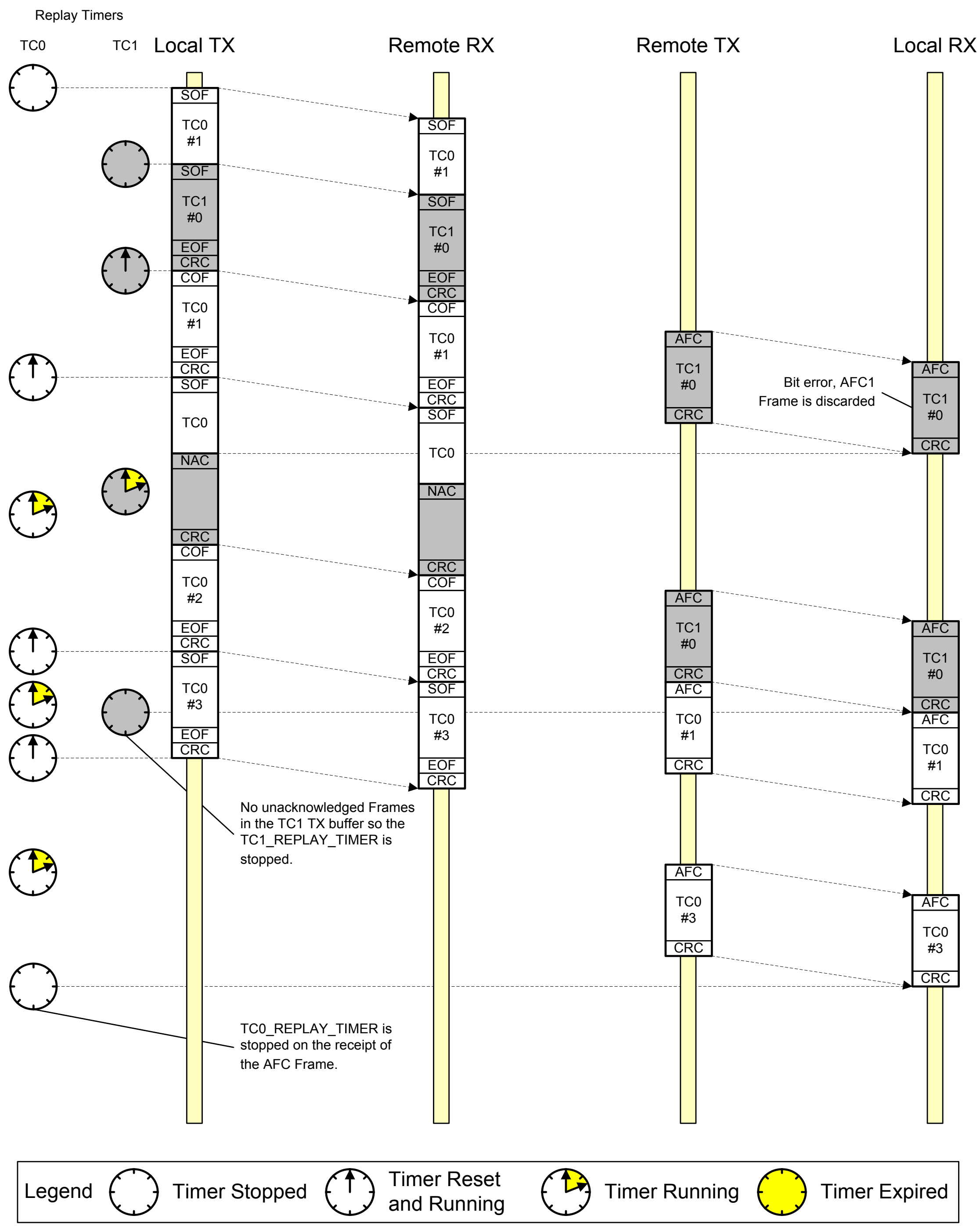
6851

Figure 125 Retransmission Due to Wrong Frame Sequence Number

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Figure 126 depicts the scenario where NAC Frame transmission is triggered due to errors in AFC reception.

- The local TX starts sending TC0 Frame#1. TC0 Frame#1 is preempted by TC1 Frame#0 while the former was in transmission.
- After transmitting TC1 Frame#0 completely, TC1_REPLAY_TIMER is reset and started and the local TX continues with sending TC0 Frame#1 from its preempted position and then TC0 Frame#2.
- Meanwhile, the peer RX receives TC1 Frame#0 and peer TX sends acknowledgment for it with the AFC1#0 Frame.
- The local RX detects this AFC1#0 Frame in error. Then the local TX sends a NAC Frame (RReq bit set to ‘0’). The NAC Frame preempts TC0 Frame#2. After the NAC Frame is sent, the transmission of TC0 Frame#2 is continued from its preempted position.
- The peer RX detects the NAC Frame and peer TX transmits only the AFC1#0 and AFC0#1 Frames. There is no Data Frame retransmission from remote TX because there are no outstanding Frames at the peer TX.
- With the reception of the AFC1#0 Frame, the TC1_REPLAY_TIMER is stopped as there are no unacknowledged Frames in the buffer. The reception of the AFC0#3 Frame (acknowledgment for TC0 Frame#3) stops the TC0_REPLAY_TIMER. The illustration also shows resetting and running of TC0_REPLAY_TIMER after sending each TC0 Frame.


Replay Timers
TC0
TC1
Local TX
Remote RX
Remote TX
Local RX
SOF
TC0 #1
TC1 #0
EOF
CRC
COF
NAC
AFC
TC0 #2
TC0 #3
Bit error, AFC1 Frame is discarded
No unacknowledged Frames in the TC1 TX buffer so the TC1_REPLAY_TIMER is stopped.
TC0_REPLAY_TIMER is stopped on the receipt of the AFC Frame.
Legend
Timer Stopped
Timer Reset and Running
Timer Running
Timer Expired

6870

Figure 126 No Retransmission Due to Error in AFC Reception

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

B.2 Preemption Scenarios

B.2.1 Forward Link

Figure 127 explains the forward Link use-case. The TC0 is requested to transmit a bulk of data. Just after the TC0 Frame started on the Link a small Message, i.e. high priority request like IRQ, is requested to send over TC1.

Figure 128 shows the further behavior of the Link without preemption. The transmission of the IRQ Message is delayed until the TC0 Frame is finalized.

Figure 129 and *Figure 130* show the behavior of the Link with preemption. *Figure 129* shows that the IRQ Message preempts TC0 Frame transmission as it has a higher priority. After the IRQ Message is finalized the TC0 Frame transmission is finalized. The IRQ Message is delivered faster with preemption.

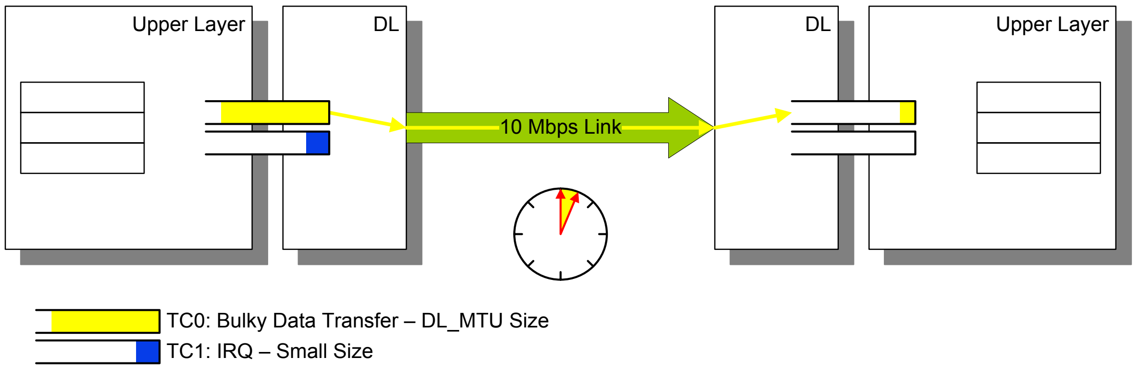


Figure 127 Forward Link Use-Case

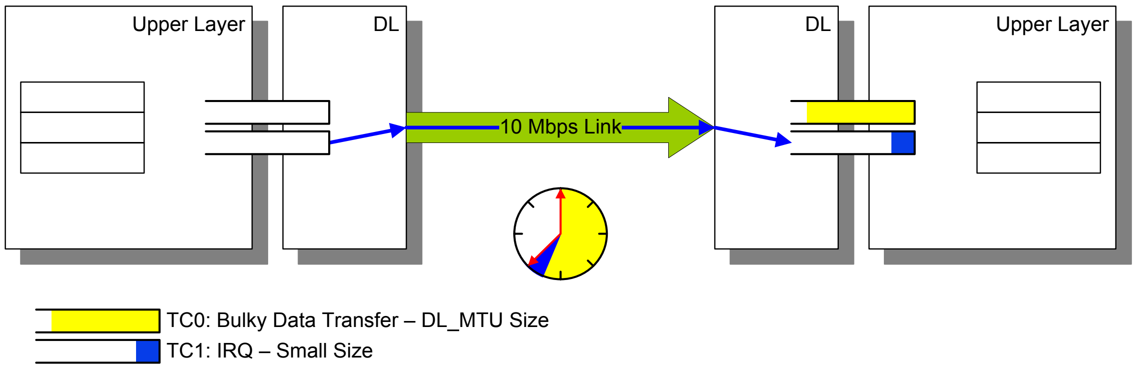


Figure 128 Forward Link Use-Case Without Preemption

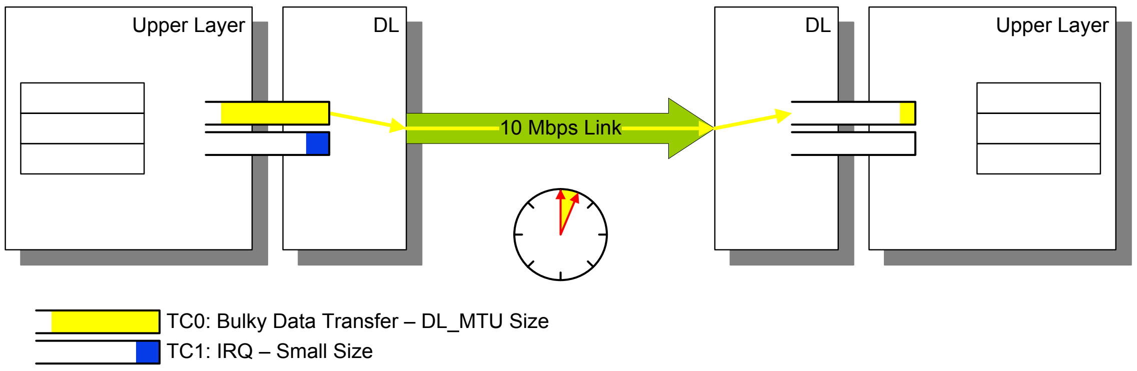


Figure 129 Forward Link Use-Case with Preemption – Start of Preemption

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

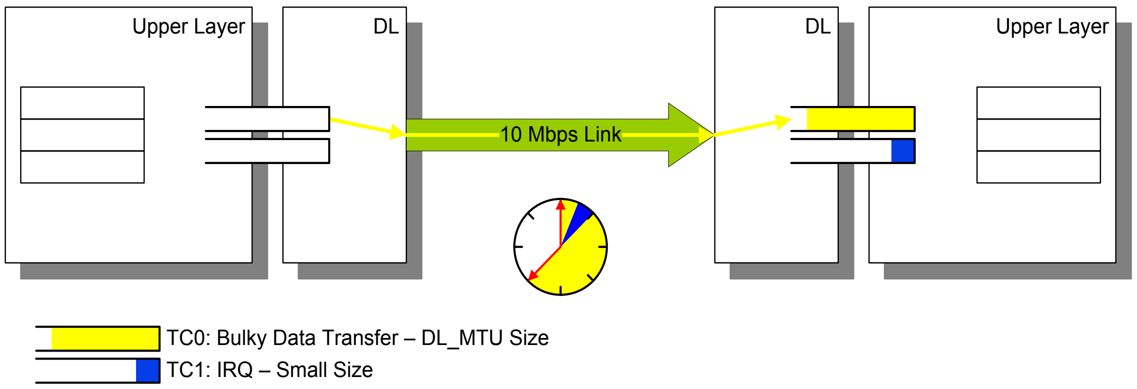


Figure 130 Forward Link Use-Case with Preemption – End of Preemption

B.2.2 Reverse Link

Besides the forward Link use-case explained earlier, the reverse Link use case (see ***Figure 131***) also benefits from the preemption. Without preemption the reverse traffic can block the forward Link by a significant factor compared to preemption case. Because an AFC Frame can only be sent after a current transmitted Frame is finalized and on the forward Link the Frame cannot be transmitted when the forward Link is waiting for an AFC Frame (no credits available or max outstanding acknowledgments (DL_TCxOutAckThreshold) are reached).

With preemption the AFC Frame is transmitted immediately in consideration of the arbitration scheme and the IDLE time of the forward Link is reduced to a minimum if no higher priorities traffic is ongoing on the reverse Link.

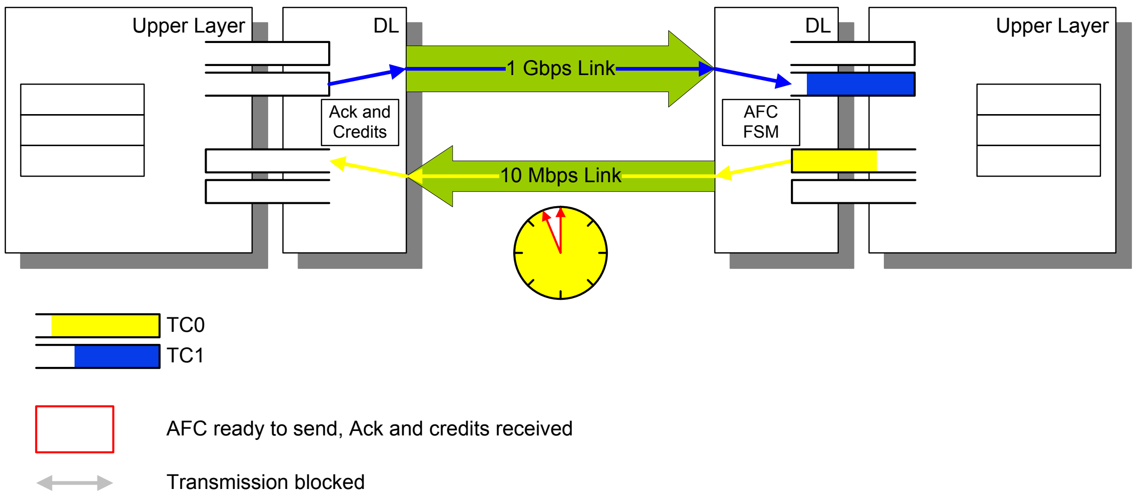


Figure 131 Reverse Link Use-Case

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

The behavior of the use-case is described here without preemption. ***Figure 132*** shows that TC0 Frame is started on reverse Link just before the completion of the TC1 data on forward Link. The received TC1 data is consumed by the upper layer and the buffer is empty now. AFC1 Frame cannot be sent immediately as TC0 occupies the reverse Link. In ***Figure 133*** the TC1 buffer at the receiver is empty and it is not yet communicated to the sender side. The sender has new TC1 data to send. It cannot be sent due to the undelivered AFC1 Frame from receiver side. This results in an IDLE state of the forward Link. After completion of TC0, the AFC1 Frame is transmitted on the reverse Link (see ***Figure 134***). After the reception of the flow control the forward Link is able to transmit new TCx Data Frame on the forward Link.

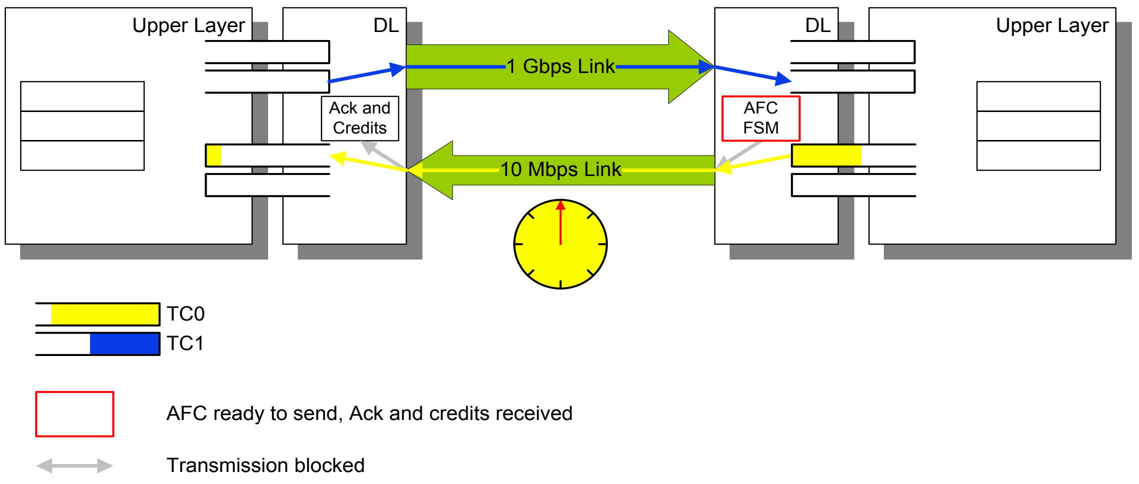


Figure 132 Reverse Link Use-Case Without Preemption – AFC1 is Blocked

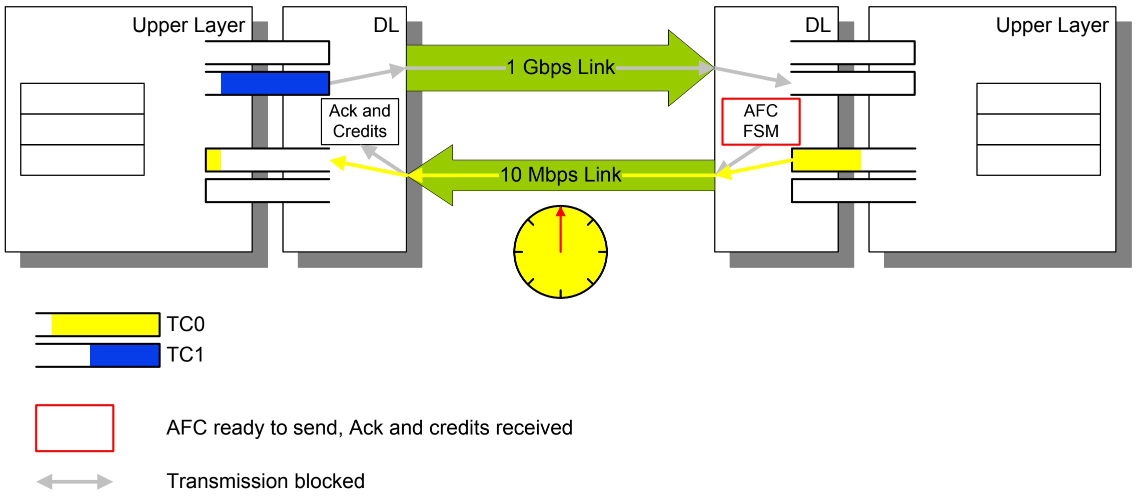


Figure 133 Reverse Link Use-Case Without Preemption – TC1 Data is Blocked

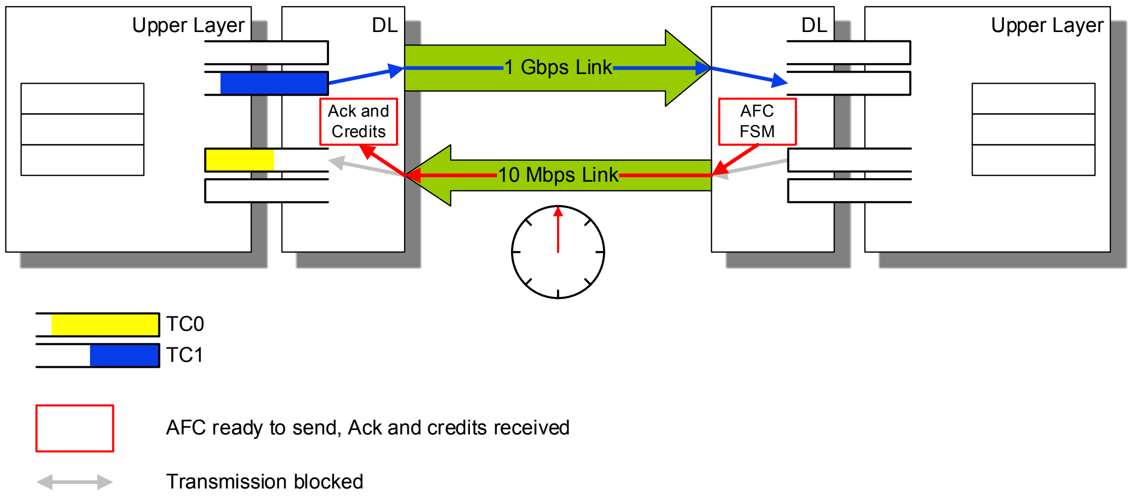


Figure 134 Reverse Link Use-Case Without Preemption – AFC1 Transmission

With preemption the behavior is different. As the AFC1 is higher priority than the TC0, the AFC1 preempts immediately the ongoing TC0 transmission (see ***Figure 135***). The sender can serve TC1 Data Frame transmission after the AFC1 is received (see ***Figure 136***). The preempted Frame is continued after the AFC Frame transmission is finalized (see ***Figure 137***). IDLE time of the forward Link is reduced to a minimum.

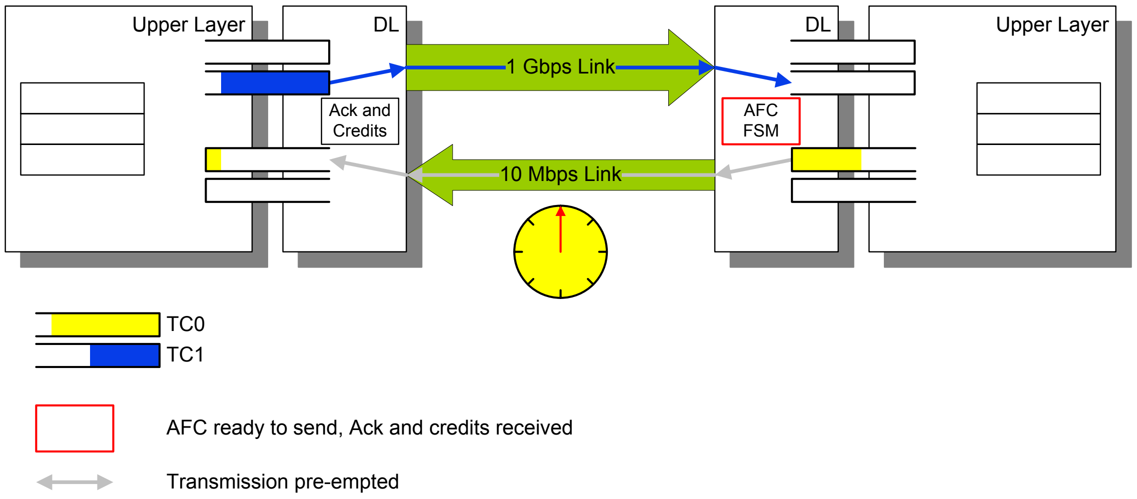
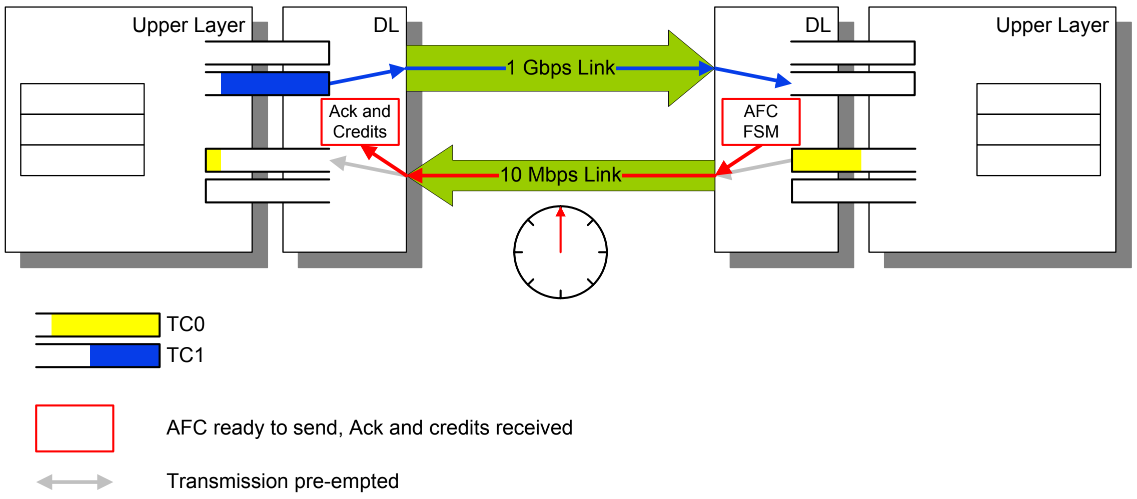
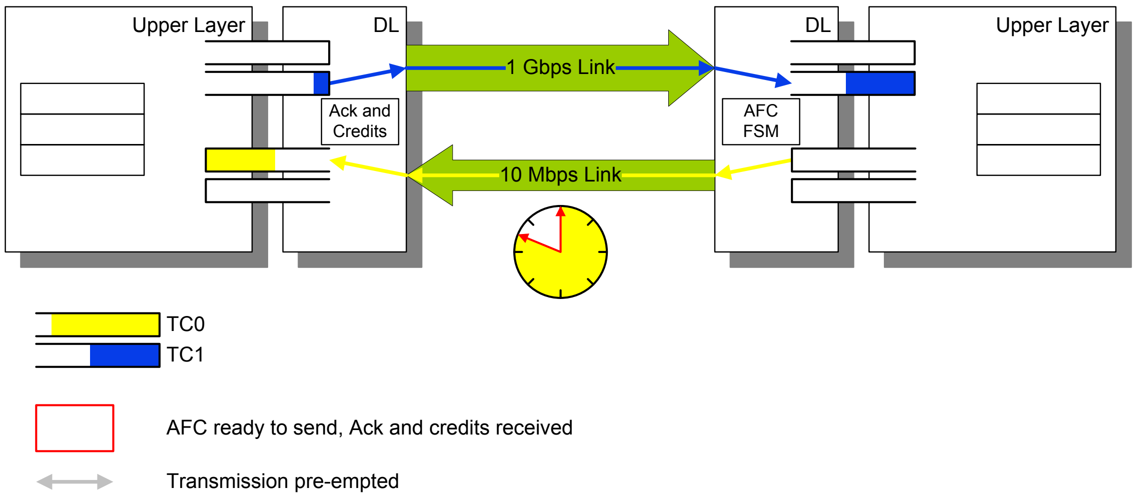


Figure 135 Reverse Link Use-Case With Preemption – AFC1 Transmission



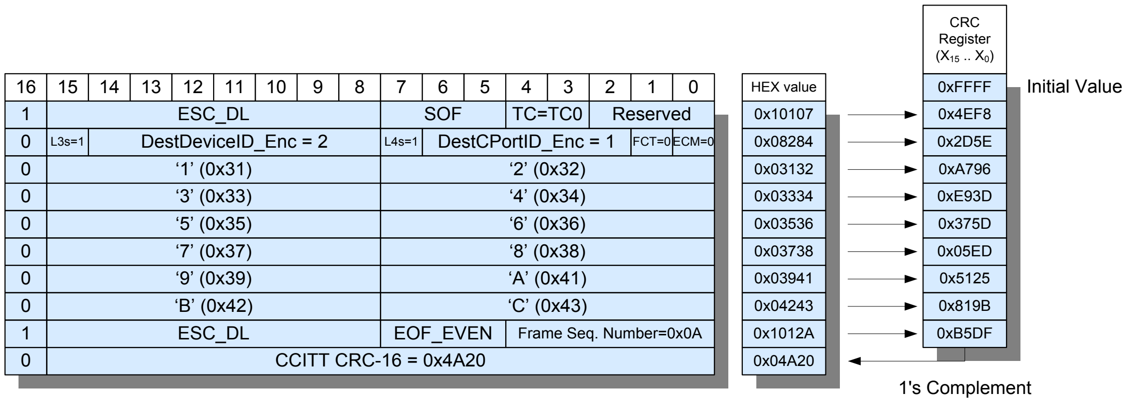
6910
Figure 136 Reverse Link Use-Case With Preemption – AFC1 Reception



6911
Figure 137 Reverse Link Use-Case with Preemption – End of Preemption

B.3 CCITT CRC-16 Example

An illustrative example of the CCITT CRC-16 is given in ***Figure 138***. The example shows a TC0 Frame with ASCII characters ‘1’ through ‘9’, ‘A’, ‘B’ and ‘C’.



16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	HEX value	CRC Register (X_{15} .. X_0)
																		0xFFFF Initial Value
1	ESC_DL								SOF			TC=TC0		Reserved			0x10107	0x4EF8
0	L3s=1	DestDeviceID_Enc = 2							L4s=1	DestCPortID_Enc = 1					FCT=0	ECM=0	0x08284	0x2D5E
0	‘1’ (0x31)								‘2’ (0x32)								0x03132	0xA796
0	‘3’ (0x33)								‘4’ (0x34)								0x03334	0xE93D
0	‘5’ (0x35)								‘6’ (0x36)								0x03536	0x375D
0	‘7’ (0x37)								‘8’ (0x38)								0x03738	0x05ED
0	‘9’ (0x39)								‘A’ (0x41)								0x03941	0x5125
0	‘B’ (0x42)								‘C’ (0x43)								0x04243	0x819B
1	ESC_DL								EOF_EVEN			Frame Seq. Number=0x0A					0x1012A	0xB5DF
0	CCITT CRC-16 = 0x4A20																0x04A20	1's Complement

Figure 138 CCITT CRC Example

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Annex C SAP Primitive Formalism (informative)

This Specification uses the OSI protocol formalism developed in ***[ITUT01]*** and ***[ITUT02]*** to describe the UniPro protocol stack. However, a few additional concepts are needed to fully describe the UniPro stack and its behavior.

Section 4.3 provides an overview of the SAP concept used by the UniPro stack. The following summary of the four classical Service Primitives is provided for convenience:

- The Request (.req) Service Primitive is used by a Service User to request a Service Provider execute a particular action. The action may cause local and peer Device state changes.
- The Indication (.ind) Service Primitive is used by a Service Provider to indicate to a Service User that a particular action was executed. The action may have been triggered remotely or locally.
- The Response (.rsp) Service Primitive is used by a Service User to carry out an action in response to an Indication Service Primitive. The action may cause remote and local state changes.
- The Confirm (.cnf) Service Primitive is used by a Service Provider to signal that a particular action was executed or to provide a response to a request. The action may have been triggered remotely or locally.

In most cases, these Service Primitives relate to each other as shown in ***Figure 139***.

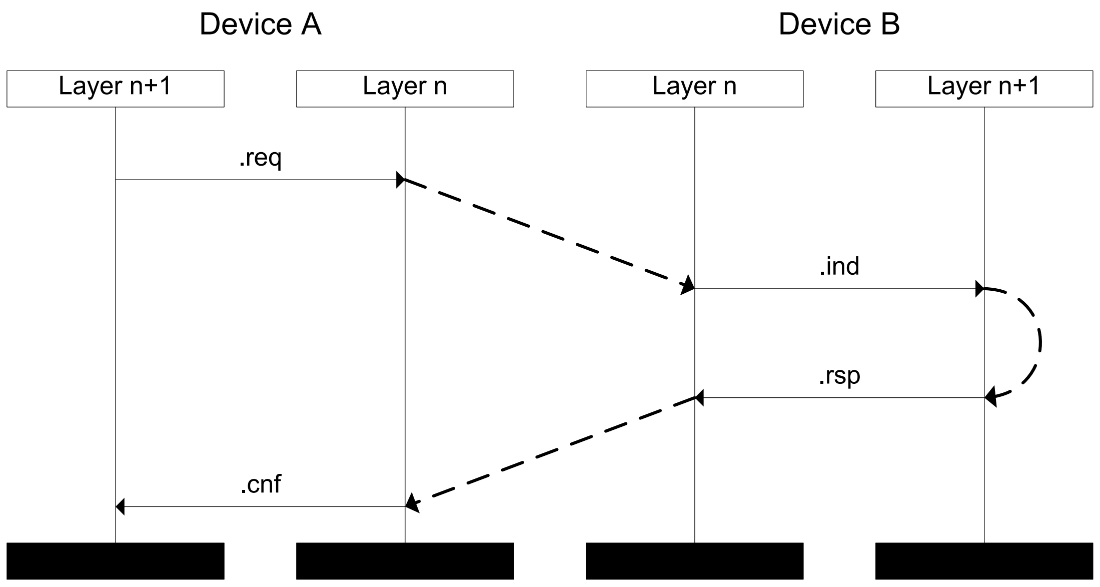


Figure 139 Common Usage of SAP Primitives

C.1 Additional SAP Primitives

The Message Sequence Chart shown in ***Figure 139*** is typical for certain types of transactions. However, there are situations where the relationship of a Response or Confirm Service Primitive to another Service Primitive or action is ambiguous as shown in ***Figure 140***. In this example, it is not clear if a Confirm Service Primitive is the result of a local or remote action. A Response Service Primitive has the same ambiguity.

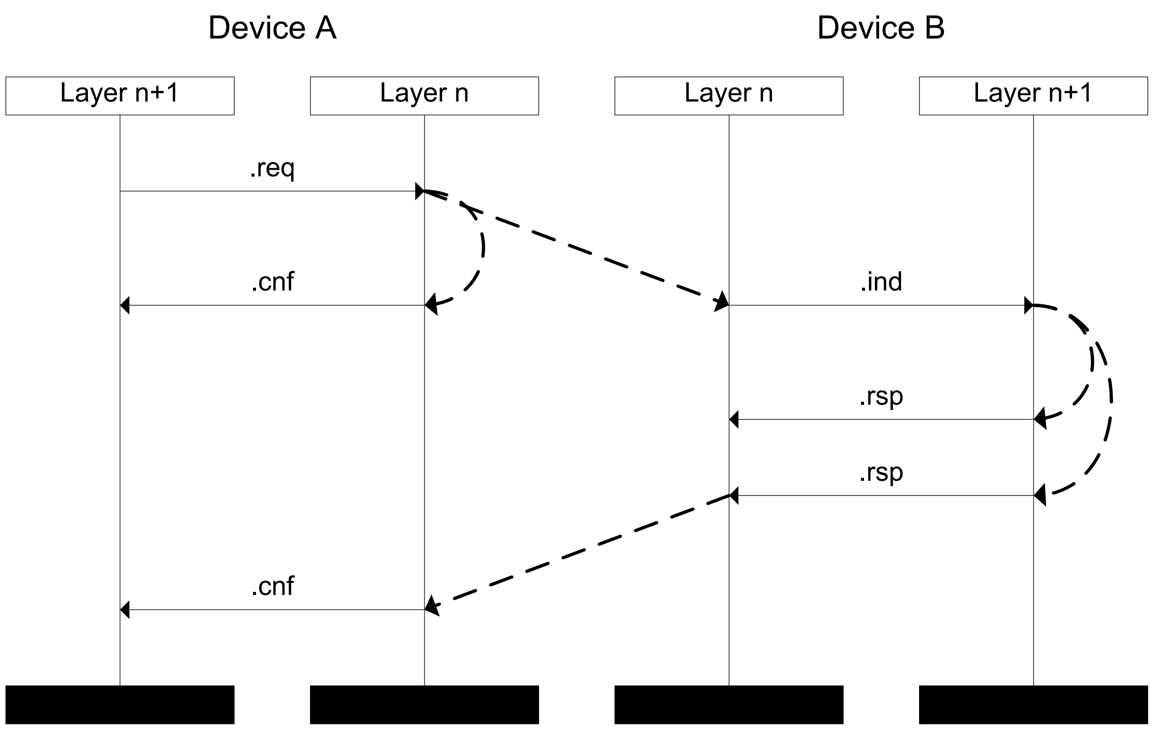


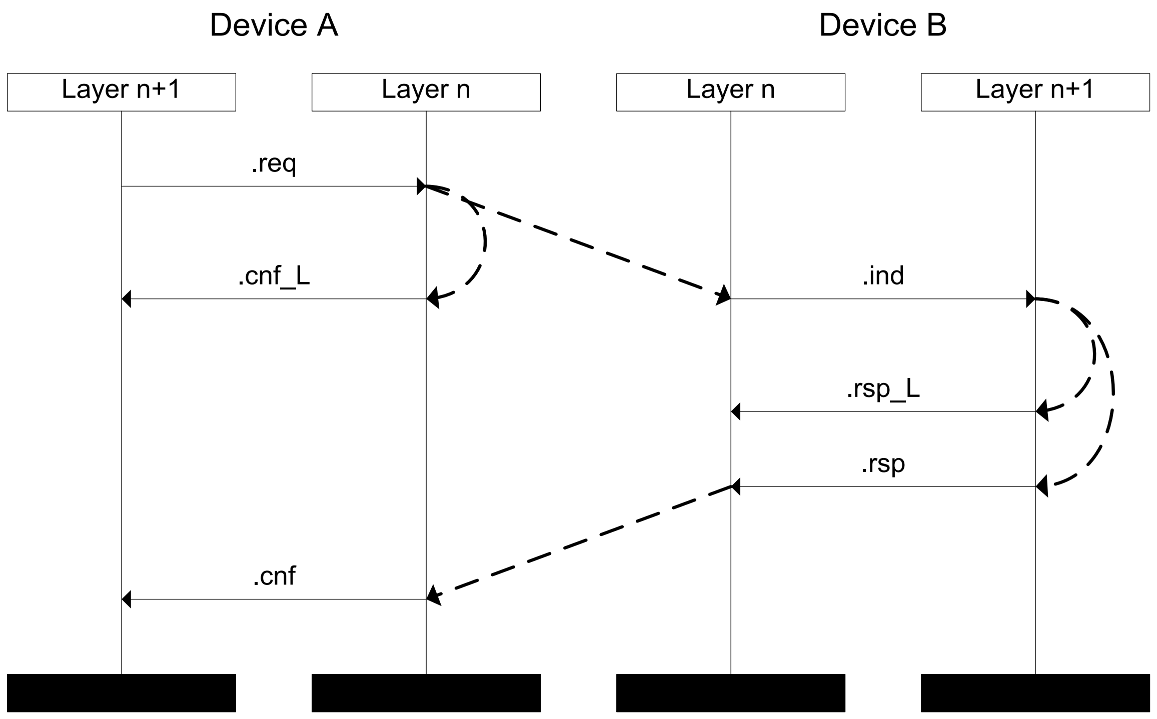
Figure 140 Ambiguous Usage of SAP Primitives

The following additional Service Primitives are introduced to avoid this ambiguity:

- The Local Confirm (.cnf_L) Service Primitive is issued following a Request Service Primitive directed at the local Device. Typically, a new Request Service Primitive cannot be issued before reception of a Local Confirm Service Primitive, thus regulating the rate of Messages sent between the Service User and local Service Provider.
- The Local Response (.rsp_L) Service Primitive is issued following an Indication Service Primitive that was directed at the local Device. Typically, a new Indication Service Primitive cannot be issued before reception of a Local Response Service Primitive, thus regulating the rate of Messages sent between the Service User and local Service Provider.

With the addition of these two new Service Primitives, the relationships between Service Primitives are modified as shown in ***Figure 141***. The Response (.rsp) and Confirm (.cnf) Service Primitives are now reserved for Messages that result in a Message sent to a peer Device.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential



6950

Figure 141 Usage of the Local SAP Primitives

This page intentionally left blank.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Annex D CPort Signal Interface (informative)

The CPort signal interface defines a generic signal interface used to connect UniPro CPorts to Applications. The CPort signal interface is the signal-interface equivalent of the T_CO_SAP.

The CPort signal interface is informative only. Conformance to the UniPro Specification does not depend on any portion of the signal interface defined herein. This interface is meant as an example of how the more generic T_CO_SAP could be instantiated in an implementation.

Arbitration between different CPorts is performed by the UniPro stack using a round-robin scheme. An optional arbitration scheme might be implemented in the Application layer in addition to the arbitration scheme in the UniPro stack. Either option is a design choice that leads to a particular CPort signal interface design. Interfaces for each option are presented in the following sections. Signals can be added or removed from these interfaces as needed.

For an implementation that includes the optional arbitration scheme in the Application layer, a multiplexed CPort signal interface is described in ***Section D.1***. In this case, sending more than one Fragment simultaneously is not possible. This interface includes status and arbitration signals, which provide relevant information for the Application.

For an implementation that includes the arbitration scheme in the UniPro stack, a non-multiplexed CPort signal interface is described in ***Section D.2***. In this case, more than one Fragment can be sent simultaneously since the T_CO_DATA.req interface is duplicated for each CPort.

The CPort signal interface is optimized for a local on-chip interface. As a result, there is no attempt to minimize the number of signals.

D.1 Multiplexed CPort Signal Interface

A multiplexed CPort signal interface is presented in ***Figure 142***. If only a single multiplexed CPort interface is implemented, the UniPro L4 arbitration scheme collapses and a UniPro L4 arbiter is no longer required. However, the multiplexed CPort interface requires arbitration in the Application Layer.

All CPort streams from the Application are multiplexed into a single physical CPort signal interface at the UniPro boundary. An arbiter is used to select the active CPort stream and CPort ID based on the selected arbitration policy favored by the Application.

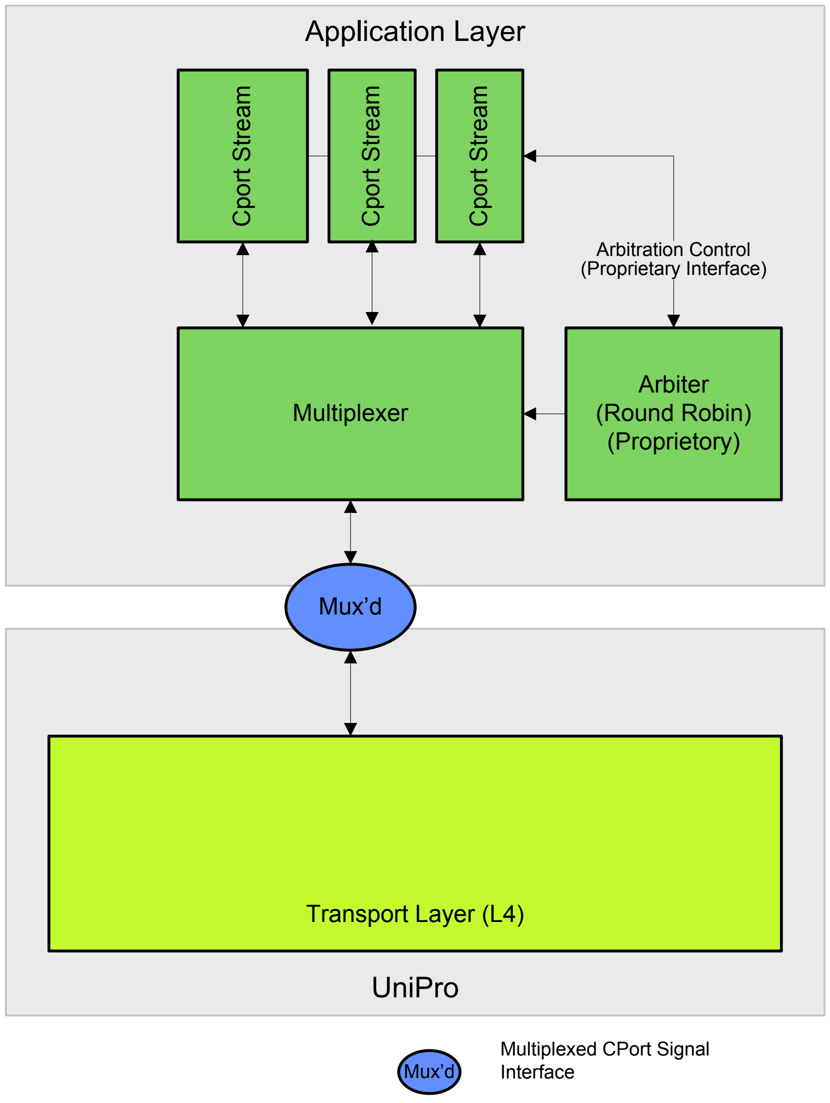


Figure 142 Top Level Multiplexed CPort Signal Interface Diagram

CPort arbitration is Segment based. An Application controls which Message Fragment should be sent first, and therefore there is only one set of data lines. The arbiter selects the priority and order of Segments of different CPorts, and can implement any kind of priority scheme such as round-robin or priority encoded.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

D.1.1 Signal Definition

The multiplexed CPort signal interface, depicted in ***Figure 143***, consists of five signal groups:

- Global, which includes the clock
- T_CO_DATA.req, which includes the short-header data transmission signals
- T_CO_DATA.ind, which includes the short-header data reception signals
- T_CO_FLOWCONTROL.req, which includes the flow-control transmission signals
- TX Status and Arbitration, which includes optional signals for external TX CPort arbitration

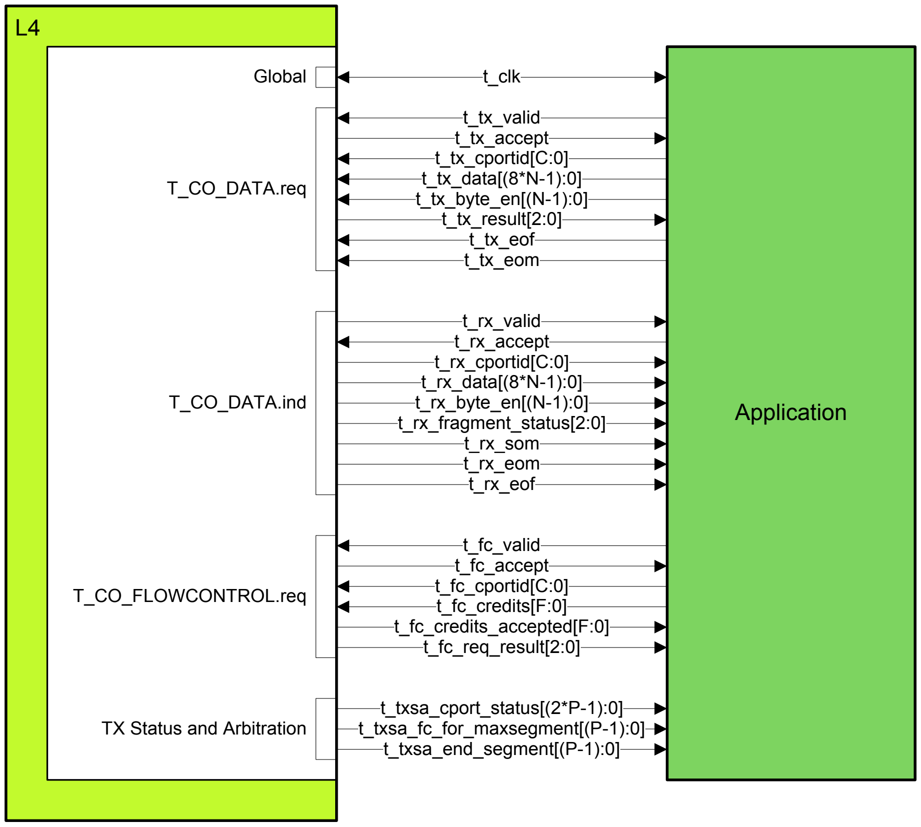


Figure 143 Multiplexed CPort Signal Interface Example

All signals of the CPort signal interface are synchronous to the rising edge of t_clk.

The T_CO_DATA.req, T_CO_DATA.ind and T_CO_FLOWCONTROL.req reflect the UniPro T_CO_SAP.

The Application has a TX CPort arbiter that controls which Fragment is sent to which CPort, and at what time. If the interface consists of multiple CPorts, this Application controlled arbitration cannot control nor effect the UniPro Transport Layer CPort arbitration between multiple physically implemented multiplexed CPorts. For example, if there are a few Fragments queued at more than one UniPro CPort, only the Application layer defines the order of transmission of these Fragments.

The CPort signal interface multiplexes CPort data transfers in order to limit the number of signals, and because data is serialized by the UniPro stack. The UniPro stack can assist the arbitration through the Status and Arbitration signal group. There is no need for similar arbitration signals at the RX side, as the data is serialized before being transferred over the UniPro Link.

A detailed description of the CPort interface signals is shown in ***Table 108***.

Table 108 Multiplexed CPort Interface Signal Definition

Signal Name	Driver	Description
Global Group		
t_clk	Clock source	Main Clock All the signals of the CPort signal interface are synchronous to the rising edge of t_clk.
T_CO_DATA.req Group		
t_tx_valid	Application	Transmitter Valid When '1', this signal indicates that t_tx_cportid, t_tx_data, t_tx_byte_en, t_tx_eof and t_tx_eom are valid, and will remain valid, until they are accepted by the CPort by setting t_tx_accept to '1'. Data is transferred when t_tx_valid and t_tx_accept are both '1'.
t_tx_accept	CPort	Transmitter Accept When '1', this signal indicates that the CPort is ready to accept data. The UniPro stack may set t_tx_accept to '1' before t_tx_valid is set to '1' by Application.
t_tx_cportid[C:0]	Application	Transmitted CPort ID This signal indicates the local CPort ID to which the data is destined. This signal is sampled when t_tx_valid is '1', and is driven with each data element. The signal width depends on the number of CPorts in the UniPro stack, e.g., if sixteen CPorts are implemented, C = 3. The t_tx_cportid value remains constant for all valid clocks in a Message Fragment, and only changes after the t_tx_eof has indicated an End Of Fragment to the UniPro Stack.
t_tx_data[(8*N-1):0]	Application	Transmitter Data This signal width is a multiple of 8, and is implementation-specific. The byte transmission order is t_tx_data[(8*N-1):(8*N-8)] (MS byte), t_tx_data[(8*N-9):(8*N-16)], … t_tx_data[7:0] (LS byte).
t_tx_byte_en[(N-1):0]	Application	Transmitter Byte Enable In the case of a multi-byte data interface, this signal contains one bit per data byte. The t_tx_byte_en[i] values of '1' and '0' indicate the presence and absence of valid data for the i-th byte (t_tx_data[(i*8+7):(i*8)]), respectively. The t_tx_byte_en bits for a Fragment shall be contiguous. Thus, in the last data transfer of a Fragment, the permissible values for t_tx_byte_en are only those values with adjacent valid bytes starting from the MS byte. For example with N = 4, the permissible t_tx_byte_en values are 0b1111, 0b1110, 0b1100 and 0b1000. The Application may defer termination of a Message transmitting an isolated End Of Message without the need to present payload at t_tx_data. In this case t_tx_eom is set to 0b1 together with t_tx_byte_en being set to 0b0000. A Message carries no payload at all, if such an isolated End Of Message transfer is directly preceded by another transfer with t_tx_eom being set.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Signal Name	Driver	Description
t_tx_result[2:0]	CPort	Transmitter Request Result This is the CPort response to the Application's T_CO_DATA.req. t_tx_result is valid one cycle after both t_tx_valid and t_tx_accept were set to ‘1’. t_tx_result uses the values for L4CPortResultCode as shown in ***Table 68***. The following values are valid: 0: SUCCESS; The data was received by the CPort successfully. 1: NO_CONNECTION; The given CPort number is not connected. 3: NO_CPORT; The given CPort number does not exist. 6: NO_PEER_TC; The peer Device does not support the given TC. 7: INVALID_FRAGMENT; The given Fragment is corrupted.
t_tx_eof	Application	Transmitter End Of Fragment When '1', this signal indicates that the last byte of the Fragment is sent on the t_tx_data lines. The least significant-enabled byte is the final byte of the Message Fragment. The current Fragment is finalized, and any new data to the same CPort starts a new Fragment. This signal can be used to control the size of Segments on the Link.
t_tx_eom	Application	Transmitter End Of Message When '1', this signal indicates the end of the Message. The last Message byte is the least significant enabled byte transferred by t_tx_data. The last Message byte is either transferred in the same clock cycle as t_tx_eom or in a previous clock cycle with t_tx_eom = 0b0 followed by the transfer of t_tx_eom = 0b1 together with t_tx_byte_en = 0b0000). Setting t_tx_eom to '1' results in the EOM flag being set to '1' for the Segment containing the last Message byte. A Segment with EOM flag being set is also transmitted, if the Message contains zero payload bytes.
T_CO_DATA.ind Group		
t_rx_valid	CPort	Receiver Valid When '1', this signal indicates that t_rx_cportid, t_rx_data, t_rx_byte_en, t_tx_eof, t_rx_som and t_rx_eom are valid, and will remain valid, until they are accepted by the Application by setting t_rx_accept to '1'. The data is transferred when t_rx_valid and t_rx_accept are both '1'.
t_rx_accept	Application	Receiver Accept When '1', this signal indicates that the Application is ready to accept data. The Application may set t_rx_accept to '1' before t_rx_valid is set to '1'.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Signal Name	Driver	Description
t_rx_cportid[C:0]	CPort	Receiver CPort ID This signal indicates the CPort ID to which the data belongs. This signal is sampled when t_rx_valid is '1', and is driven with each data element. The signal width depends on the number of CPorts in the UniPro stack, e.g., if sixteen CPorts are implemented, C = 3. A CPORT 'N' that receives a message containing no payload is indicated by t_rx_valid, t_rx_som, t_rx_eof and t_rx_eom being set while t_rx_byte_en = 0b0000 and t_rx_cportid = N
t_rx_data[(8*N-1):0]	CPort	Receiver Data This signal width is a multiple of 8, and is implementation-specific. The byte transmission order is t_rx_data[(8*N-1):(8*N-8)] (MS byte), t_rx_data[(8*N-9):(8*N-16)], … t_rx_data[7:0] (LS byte).
t_rx_byte_en[(N-1):0]	CPort	Receiver Byte Enable In the case of a multi-byte data interface, this signal contains one bit per data byte. The t_rx_byte_en[i] values of '1' and '0' indicate the presence and absence of valid data for the i-th byte (t_rx_data[(i*8+7):(i*8)]), respectively. The t_rx_byte_en bits for a Fragment shall be contiguous. Thus, in the last data transfer of a Fragment, the permissible values for t_rx_byte_en are only those values with adjacent valid bytes starting from the MS byte. For example with N = 4, the permissible t_rx_byte_en values are 0b1111, 0b1110, 0b1100 and 0b1000. The value t_rx_byte_en = 0b0000 is permissible only, when t_rx_eom is set.
t_rx_som	CPort	Receiver Start Of Message When '1', this signal informs the Application Layer that the MS byte of this access is the first byte of a new Message.
t_rx_eom	CPort	Receiver End Of Message When '1', this signal indicates that the last valid byte in t_rx_byte_en is the LS byte of the Fragment that is also the last valid byte of the Message. A Message containing no payload is indicated by t_rx_som and t_rx_eom being set while t_rx_byte_en = 0b0000.
t_rx_eof	CPort	Receiver End Of Fragment When '1', this signal indicates reception of the last byte of a Message Fragment, which is the least significant enabled-byte transferred by t_rx_data in the same clock cycle. ***Note:*** *A Message Segment in L4 might be mapped to a single Message Fragment in the Application Layer.*

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Signal Name	Driver	Description
t_rx_fragment_status[1:0]	CPort	Receiver Fragment Status This signal indicates whether the currently received Fragment is valid or corrupt due to Fragment(s) being dropped in the CPort (by the CPort Safety Valve or Controlled Segment Dropping). t_rx_fragment_status uses the values for FragmentStatus as shown ***Table 68***. The following values are valid: 0: FRAGMENT_CORRECT: The current Fragment transfer is correct 1: FRAGMENT_CORRUPT: The current Fragment transfer is corrupted as a CSD or CSV occurred after the start of current Fragment transfer 2: FRAGMENT_AFTER_GAP: The current Fragment transfer is correct but one or more Fragments were dropped due to a CSD or CSV before the start of current Fragment 3: FRAGMENT_CORRUPT_AFTER_GAP: The current Fragment is corrupted as a CSD or CSV occurred after the start of current Fragment transfer. In addition, one or more Fragments were dropped due to a CSD or CSV before the start of current Fragment The signal is set to '0' or '2' at the start of current Fragment depending on whether the previous Fragment was correctly delivered or not. Once the signal changes from '0' to '1', it stays constant until t_rx_eof is set to '1'. Once the signal changes from a value '2' to '3', it stays constant until t_rx_eof is set to '1'.
T_CO_FLOWCONTROL.req Group		
t_fc_valid	Application	Flow Control Valid When '1', this signal indicates that t_fc_cportid and t_fc_credits are valid, and will remain valid, until they are accepted by the CPort by setting t_fc_accept to '1'. The data is transferred when t_fc_valid and t_fc_accept are both '1'.
t_fc_accept	CPort	Flow Control Accept When '1', this signal indicates that the CPort is ready to accept credits. The CPort may set t_fc_accept to '1' before t_fc_valid is set '1'.
t_fc_cportid[C:0]	Application	Flow Control CPort ID This signal indicates the CPort ID to which the credits are destined. The signal width depends on the number of CPorts in the UniPro stack, e.g., if sixteen CPorts are implemented, C = 3.
t_fc_credits[F:0]	Application	Flow Control Credits This signal indicates the number of credits for a CPort. The credits represent the free buffer space in bytes that became available from the last time t_fc_credits was asserted for the same CPort. The signal width, F + 1, is implementation-specific, and relates to the number of credits that can be issued at one time. When more credits than the t_fc_credits capacity needs to be transferred, the t_fc_credits signal must be asserted multiple times.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Signal Name	Driver	Description
t_fc_credits_accepted[F:0]	CPort	Flow Control Credits Accepted This signal is valid one cycle after t_fc_credits was transferred, and indicates the number of credits accepted by the CPort. All credits are transferred when t_fc_credits_accepted is equal to t_fc_credits from the previous cycle (equivalent to the SUCCESS value of L4CPortResultCode of T_CO_FLOWCONTROL.cnf_L). Only a partial number of credits are transferred when t_fc_credits_accepted is less than t_fc_credits from the previous cycle (equivalent to the CREDITS_EXCEEDED value of L4CPortResultCode of T_CO_FLOWCONTROL.cnf_L). The width of the t_fc_credits_accepted signal is identical to the t_fc_credits signal width.
t_fc_result[2:0]	CPort	Flow Control Result This is the CPort response to the Application's T_CO_FLOWCONTROL.req. This result code is valid one cycle after the t_fc_valid and t_fc_accept were asserted. t_fc_result uses the values for L4CPortResultCode as shown in ***Table 68***. The following values are valid: 0: SUCCESS: The data was received by the CPort successfully 1: NO_CONNECTION: The given local CPort number is not connected 2: CREDITS_EXCEEDED: The given credits exceed the maximum number of allowed credits for the given local CPort number 3: NO_CPORT: The given local CPort number does not exist. 4 to 7: Invalid values

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Signal Name	Driver	Description
TX Status / Arbitration Group		
t_txsa_cport_status[(2*P-1):0]	CPort	CPort Status This signal reflects the status of each CPort, with t_txsa_cport_status[2*i+1:2*i] reflecting the status of the i-th CPort (P is the total number of CPorts). 0b00: SUCCESS; Indicates that the CPort is configured and usable for data transmission and reception. 0b01: NO_CONNECTION; Indicates that the CPort is not connected 0b11: NO_PEER_TC; Indicates that the CPort is connected to a Traffic Class which is not present in the peer UniPro node . 0b10: RESERVED; Value is reserved.
t_txsa_fc_for_maxsegment[(P-1):0]	CPort	Credits for Maximum Segment If the t_txsa_fc_for_maxsegment[i] signal is '1', the i-th CPort has enough end-to-end credits to send a maximum Segment (as determined by T_TCxTxMaxSDUSize), otherwise the i-th CPort has less credits (P is the total number of CPorts).
t_txsa_end_segment[(P-1):0]	CPort	End Of Segment The t_txsa_end_segment[i] signal is set to '1' for a single cycle every time Layer 4 introduces an end of Segment.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

D.1.2 Timing Diagrams

This section illustrates the functionality of the multiplexed CPort signal interface with timing diagrams.

An example of a Message Fragment transfer from the Application to the UniPro stack is shown in ***Figure 144***. The UniPro stack indicates that it is ready to accept data by setting t_tx_accept to '1'. t_tx_byte_en is set to 0b1111, which indicates that all four bytes of the interface contain valid data. The data in this Message Fragment is transferred to CPort 0 as indicated by t_tx_cportid.

As t_tx_valid and t_tx_accept remain set to'1', the Message Fragment continues to be transferred until t_tx_eof is set to '1' in Cycle N. The data transfer in Cycle N contains only three valid bytes (Dn-2, Dn-1 and Dn) as indicated by 0b1110 on t_tx_byte_en. These three bytes are transferred on the most significant byte lines of t_tx_data. In this example, t_tx_eom is also set to '1' in Cycle N, indicating that the Fragment is the last Fragment in the Message.

In Cycle N+1, the transfer continues with a second Message Fragment for CPort 1 as indicated by t_tx_cportid. In this transfer, the UniPro stack uses t_tx_accept to delay the data transfer in Cycle N+2. A transmitter transfer can be delayed in this way when, for example, the core inserts a header symbol.

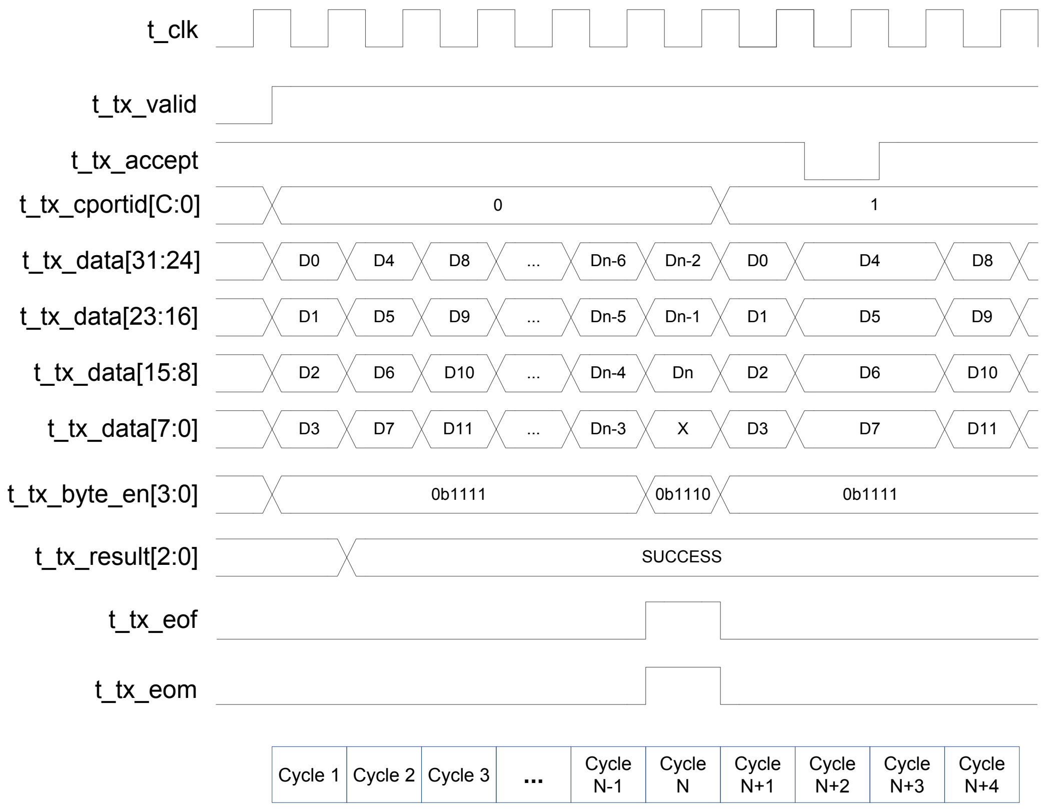


Figure 144 Transmitter Data Transfer Example for Multiplexed CPort Signal Interface

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

An example of a Message transferred as multiple Message Fragments from an Application to the UniPro stack is shown in ***Figure 145***. This example shows a Message being transferred as two Message Fragments. Hence, the second Message Fragment is the last Fragment of the Message.

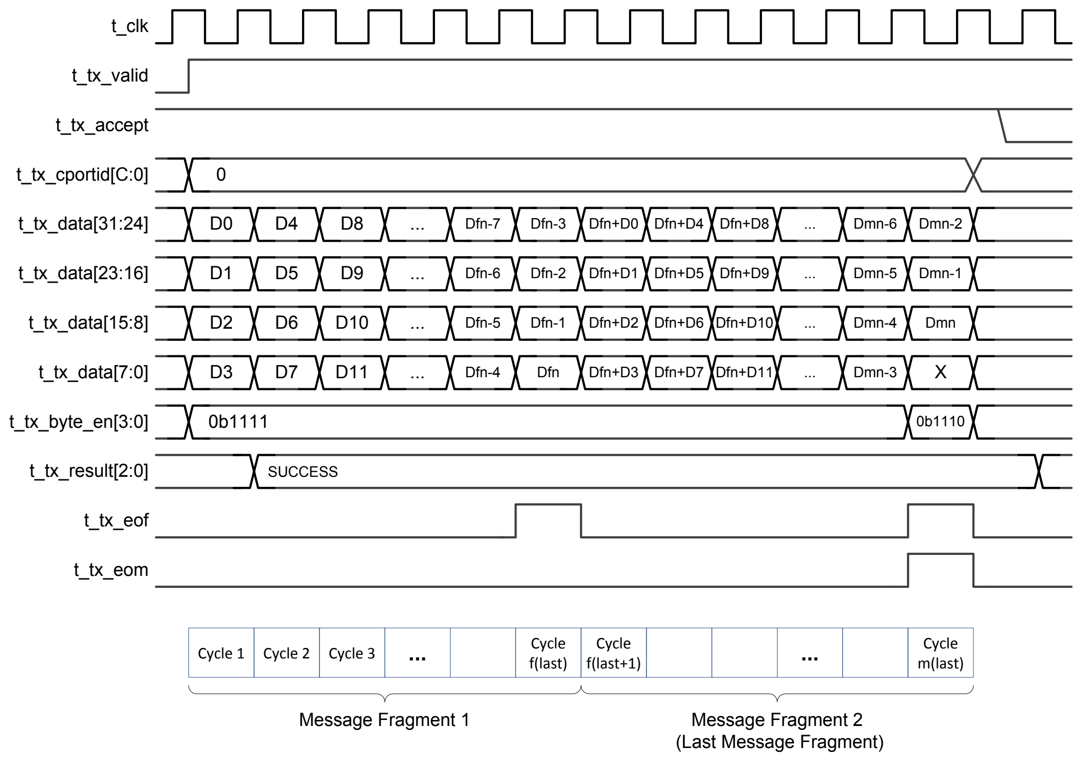

t_clk
t_tx_valid
t_tx_accept
t_tx_cportid[C:0]
0
t_tx_data[31:24]
D0 D4 D8 ... Dfn-7 Dfn-3 Dfn+D0 Dfn+D4 Dfn+D8 ... Dmn-6 Dmn-2
t_tx_data[23:16]
D1 D5 D9 ... Dfn-6 Dfn-2 Dfn+D1 Dfn+D5 Dfn+D9 ... Dmn-5 Dmn-1
t_tx_data[15:8]
D2 D6 D10 ... Dfn-5 Dfn-1 Dfn+D2 Dfn+D6 Dfn+D10 ... Dmn-4 Dmn
t_tx_data[7:0]
D3 D7 D11 ... Dfn-4 Dfn Dfn+D3 Dfn+D7 Dfn+D11 ... Dmn-3 X
t_tx_byte_en[3:0]
0b1111
0b1110
t_tx_result[2:0]
SUCCESS
t_tx_eof
t_tx_eom
Cycle 1
Cycle 2
Cycle 3
...
Cycle f(last)
Cycle f(last+1)
...
Cycle m(last)
Message Fragment 1
Message Fragment 2 (Last Message Fragment)

Figure 145 Transmission of a Message as Multiple Fragments

Upon transmitting the last word of the first Message Fragment, the Application asserts t_tx_eof to indicate the end of Fragment (Cycle f(last)). Upon transmitting the second (last) Message Fragment the Application asserts both t_tx_eof and t_tx_eom to indicate end of Fragment, as well as end of Message transmission (Cycle m(last)).

Note that t_tx_byte_en is set to 0b1111 throughout the first Message Fragment transmission; it can take other values (see ***Table 108***) only at the final cycle of the last Message Fragment transmission (Cycle m(last)).

Other than the aforementioned signal behavior, all signal behavior remains the same as described in ***Figure 144***.

A receiver data transfer is shown in ***Figure 146***, first on CPort 0 (Cycle 1 through Cycle N), then CPort 1 (Cycle N+1 onwards) as indicated by t_rx_cportid. The data is transferred when both t_rx_valid and t_rx_accept are '1'. Thus, data is transferred in all cycles, except Cycle 2. In Cycle N, the end of Fragment is indicated by setting t_rx_eof to '1'. In Cycle N, only the two most significant bytes of t_rx_data are enabled, as indicated by 0b1100 on t_rx_byte_en. FRAGMENT_CORRECT on t_rx_fragment_status indicates that the Fragment was correctly received.

In Cycle N+1, the transfer continues with a second Message Fragment to CPort 1, as indicated by t_tx_cportid. In Cycle N+2 of this Message, t_rx_fragment_status has a value of FRAGMENT_CORRUPT, indicating that the data transfer was corrupted due to a CSD or CSV.

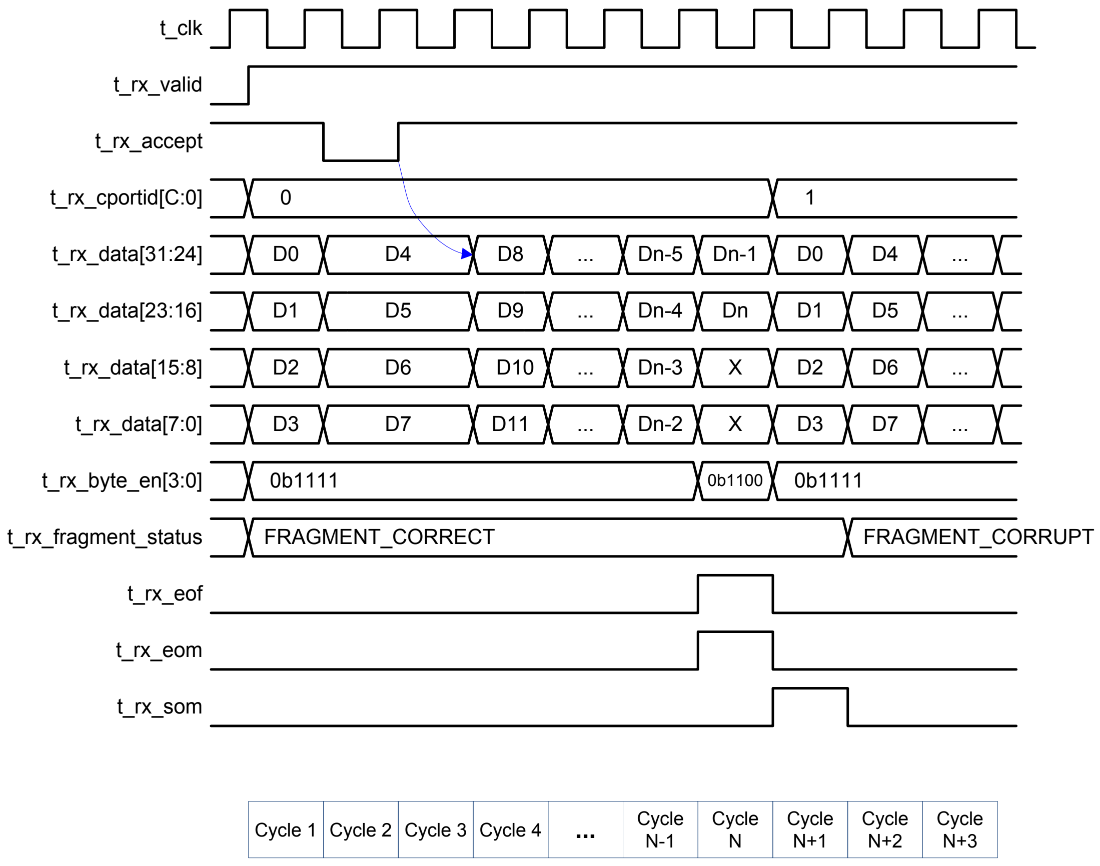


Figure 146 Receiver Data Transfer Example

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

A flow control credit transfer is shown in ***Figure 147***. The first flow control credit C0 is made available by the Application to CPort 0 on Cycle 1 by setting t_fc_valid to '1'. The credit is accepted in Cycle 2 by setting t_fc_accept to '1'. As a result, in the following Cycle, the UniPro stack indicates that the credits have been correctly processed by setting t_fc_credits_accepted to the same credit value, C0, as the value of transferred credits, and by setting t_fc_result to SUCCESS.

In Cycle 3 and Cycle 4, t_fc_valid is '0', which indicates t_fc_credits does not contain meaningful value. As shown in Cycle 4, the UniPro stack can set the t_fc_accept to '1' indicating it can accept a new credit value even if credit is not available.

In Cycle 5 and Cycle 6, new credit values C1 and C2 are transferred for CPort 1 and CPort 2, respectively. The credits are acknowledged by t_fc_credits_accepted on Cycle after the credit transfer, in Cycle 6 and Cycle 7, respectively.

The user of a CPort might transfer credits on t_fc_credits, while the UniPro stack acknowledges the credits from the previous Cycle on t_fc_credits_accepted, as shown in Cycle 6.

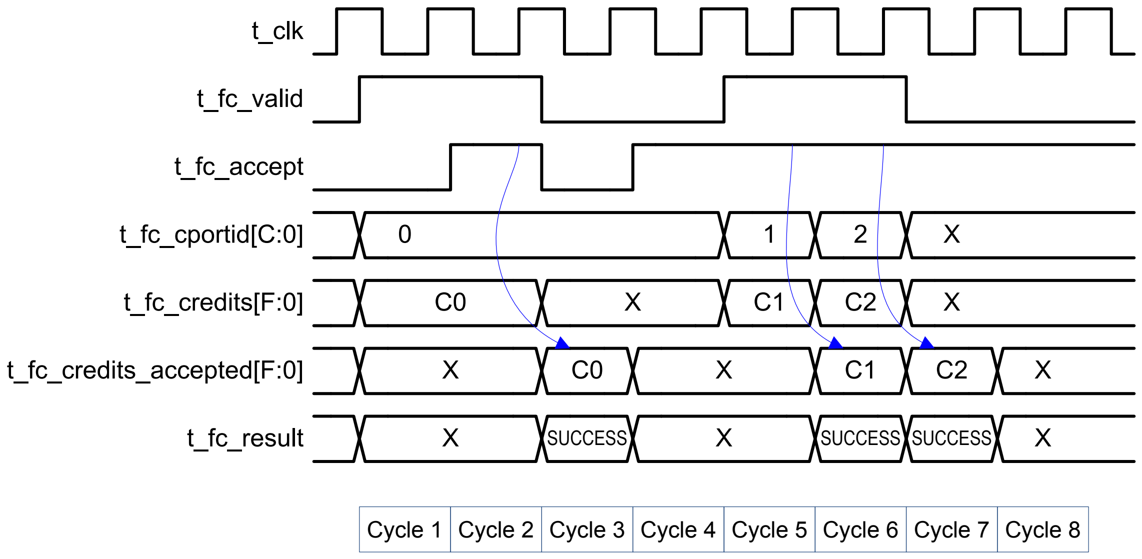


Figure 147 Flow Control Credit Transfer Example

D.2 Non-Multiplexed CPort Signal Interface

A CPort signal interface supporting internal arbitration is in ***Figure 148***. Internal arbitration is Segment based. This version of the CPort signal interface is dedicated to a single CPort. If more CPorts are needed, more CPort signal interface instances are used. Some of the signals can, however, be shared among many instances of the CPort signal interface.

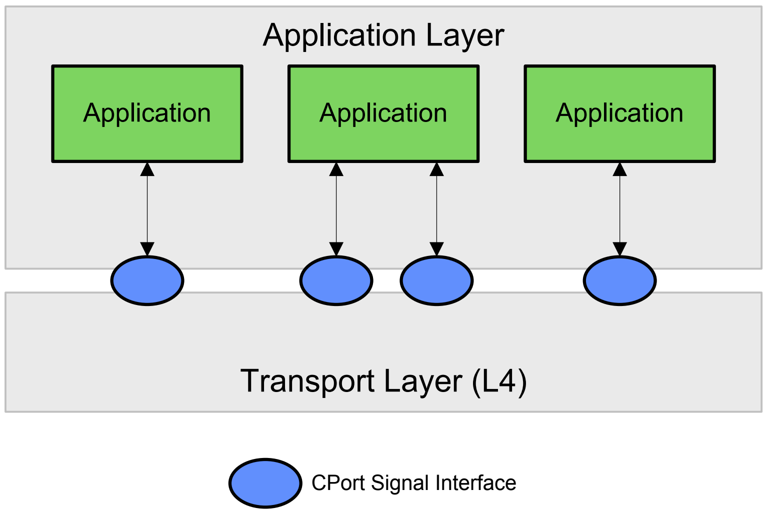


Figure 148 Top Level Non-multiplexed CPort Signal Interface Diagram

D.2.1 Signal Definition

The non-multiplexed CPort signal interface, depicted in ***Figure 149***, consists of four signal groups:

- Global, which includes the clock
- T_CO_DATA.req, which includes the short-header data transmission signals
- T_CO_DATA.ind, which includes the short-header data reception signals
- T_CO_FLOWCONTROL.req, which includes the flow-control transmission signals

All signals of the CPort signal interface are synchronous to the rising edge of t_clk.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

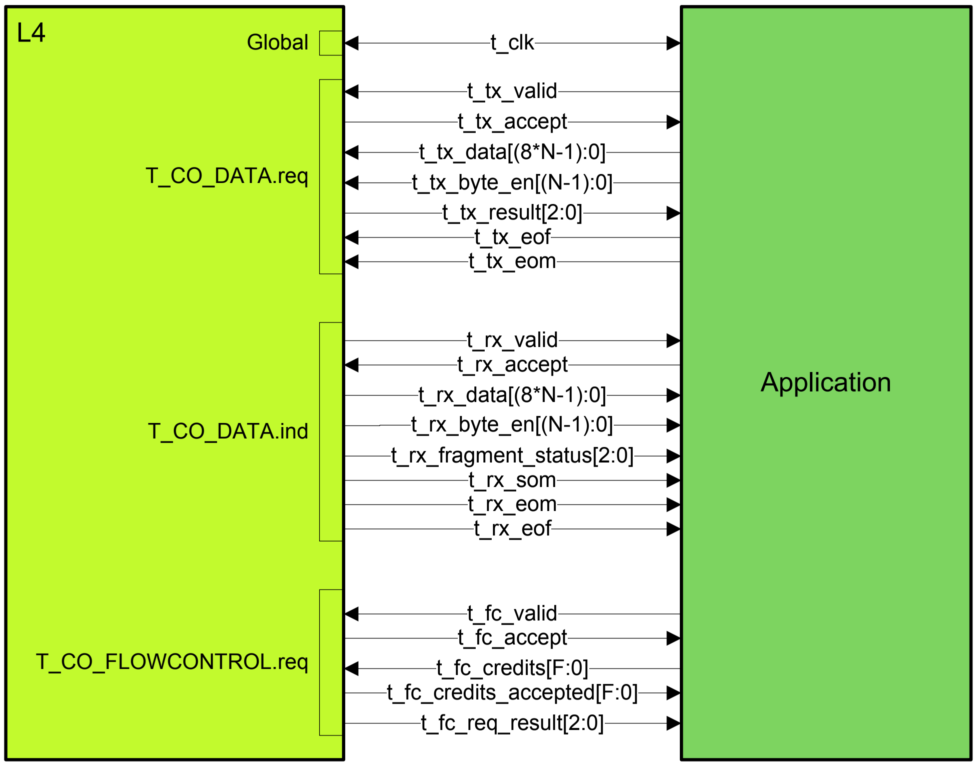


Figure 149 Non-Multiplexed CPort Signal Interface Example

The T_CO_DATA.req, T_CO_DATA.ind and T_CO_FLOWCONTROL.req reflect the UniPro T_CO_SAP.

The CPort signal interface shown is for single CPort. If more CPorts are needed, each additional CPort requires one physical signal interface. Some signal can be shared between separate CPort interfaces. They are marked as “global” signals. The shown configuration is based on particular design choices, which lead to this particular CPort signal interface design. Other choices are obviously possible. Moreover, signals can be added or removed as needed.

There is no assumption on the amount of knowledge in the Application level. The Application can try to drive data to each CPort without the need to know the Segment size of the internal arbitration algorithm or how busy each CPort is. UniPro can handle all traffic using the t_tx_accept signal. (see ***Section D.2.2***).

A detailed description of all the CPort interface signals is shown in ***Table 109***.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

7074 **Table 109 Non-multiplexed CPort Interface Signal Definition**

Signal Name	Driver	Description
Global Group		
t_clk	Clock source	Main Clock All the signals of the CPort signal interface are synchronous to the rising edge of t_clk.
T_CO_DATA.req Group		
t_tx_valid	Application	Transmitter Valid When '1', this signal indicates that t_tx_cportid, t_tx_data, t_tx_byte_en and t_tx_eom are valid, and will remain valid, until they are accepted by the CPort by setting t_tx_accept to '1'. Data is transferred when t_tx_valid and t_tx_accept are both '1'.
t_tx_accept	CPort	Transmitter Accept When '1', this signal indicates that the CPort is ready to accept data. t_tx_accept is intended for byte level transmit control. The CPort may set t_tx_accept to ‘1’ before t_tx_valid is set to ‘1’.
t_tx_data[(8*N-1):0]	Application	Transmitter Data This signal width is a multiple of 8, and is implementation-specific. The byte transmission order is t_tx_data[(8*N-1):(8*N-8)] (MS byte), t_tx_data[(8*N-9):(8*N-16)], … t_tx_data[7:0] (LS byte).
t_tx_byte_en[(N-1):0]	Application	Transmitter Byte Enable In the case of a multi-byte data interface, this signal contains one bit per data byte. The t_tx_byte_en[i] values of '1' and '0' indicate the presence and absence of valid data for the i-th byte (t_tx_data[(i*8+7):(i*8)]), respectively. The t_tx_byte_en bits for a Fragment shall be contiguous. Thus, in the last data transfer of a Fragment, the permissible values for t_tx_byte_en are only those values with adjacent valid bytes starting from the MS byte. For example with N = 4, the permissible t_tx_byte_en values are 0b1111, 0b1110, 0b1100 and 0b1000. The Application may defer termination of a Message transmitting an isolated End Of Message without the need to present payload at t_tx_data. In this case t_tx_eom is set to 0b1 and t_tx_byte_en is set to 0b0000. A Message carries no payload at all if such an isolated End Of Message transfer is directly preceded by another transfer with t_tx_eom set.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Signal Name	Driver	Description
t_tx_result[2:0]	CPort	Transmitter Result This is the CPort response to the Application's T_CO_DATA.req. t_tx_result is valid one cycle after t_tx_valid was asserted. t_tx_result uses the values for L4CPortResultCode as shown ***Table 68***. The following values are valid: 0: SUCCESS: The data was received by the CPort successfully 1: NO_CONNECTION: The given CPort number is not connected 2: CREDITS_EXCEEDED: The maximum number of Credits is exceeded 7: INVALID_FRAGMENT: The given Fragment is corrupted
t_tx_eof	Application	Transmitter End Of Fragment When '1', this signal indicates that the last byte of the Fragment is sent on the t_tx_data lines. The least significant-enabled byte is the final byte of the Message Fragment. The current Fragment is finalized, and any new data to the same CPort starts a new Fragment. This signal can be used to control the size of Segments on the Link.
t_tx_eom	Application	Transmitter End Of Message When '1', this signal indicates the end of the Message. The last Message byte is the least significant enabled byte transferred by t_tx_data. The last Message byte is either transferred in the same clock cycle as t_tx_eom or in a previous clock cycle with t_tx_eom = 0b0 followed by the transfer of t_tx_eom = 0b1 together with t_tx_byte_en = 0b0000). Setting t_tx_eom to '1' results in the EOM flag being set to '1' for the Segment containing the last Message byte. A Segment with the EOM flag set is also transmitted if the Message contains zero payload bytes.
T_CO_DATA.ind Group		
t_rx_valid	CPort	Receiver Valid When '1', this signal indicates that t_rx_cportid, t_rx_data, t_rx_byte_en, and t_rx_eom are valid, and will remain valid, until they are accepted by the Application by setting t_rx_accept to '1'. The data is transferred when t_rx_valid and t_rx_accept are both '1'.
t_rx_accept	Application	Receiver Accept When '1', this signal indicates that the Application is ready to accept data. The Application may set t_rx_accept to '1' before t_rx_valid is set to '1'.
t_rx_data[(8*N-1):0]	CPort	Receiver Data This signal width is a multiple of 8, and is implementation-specific. The byte transmission order is t_rx_data[(8*N-1):(8*N-8)] (MS byte), t_rx_data[(8*N-9):(8*N-16)], … t_rx_data[7:0] (LS byte).

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Signal Name	Driver	Description
t_rx_byte_en[(N-1):0]	CPort	Receiver Byte Enable In the case of a multi-byte data interface, this signal contains one bit per data byte. The t_rx_byte_en[i] values of '1' and '0' indicate the presence and absence of valid data for the i-th byte (t_rx_data[(i*8+7):(i*8)]), respectively. The t_rx_byte_en bits for a Fragment shall be contiguous. Thus, in the last data transfer of a Fragment, the permissible values for t_rx_byte_en are only those values with adjacent valid bytes starting from the MS byte. For example with N = 4, the permissible t_rx_byte_en values are 0b1111, 0b1110, 0b1100, and 0b1000. The value t_rx_byte_en = 0b0000 is only permissible when t_rx_eom is set.
t_rx_som	CPort	Receiver Start Of Message When '1', this signal informs the Application Layer that the MS byte of this access is the first byte of a new Message.
t_rx_eom	CPort	Receiver End Of Message When '1', this signal indicates that the last valid byte in t_rx_byte_en is the LS byte of the Fragment that is also the last valid byte of the Message. A Message containing no payload is indicated by t_rx_som and t_rx_eom both being set, and t_rx_byte_en = 0b0000.
t_rx_eof	CPort	Receiver End Of Fragment When ‘1’, this signal indicates reception of the last byte of a Message Fragment, which is the least significant enabled byte transferred by t_rx_data in the same clock cycle. ***Note:*** *A Message Segment in L4 might be mapped to a single Message Fragment in the Application Layer.*
t_rx_fragment_status[1:0]	CPort	Receiver Fragment Status This signal indicates whether the currently received Fragment is valid or corrupt due to Fragments being dropped in the CPort (by the CPort Safety Valve or Controlled Segment Dropping). t_rx_fragment_status uses the values for FragmentStatus as shown ***Table 68***. The following values are valid: 0: FRAGMENT_CORRECT: The current Fragment transfer is correct 1: FRAGMENT_CORRUPT: The current Fragment transfer is corrupted as a CSD or CSV occurred after the start of current Fragment transfer 2: FRAGMENT_AFTER_GAP: The current Fragment transfer is correct but one or more Fragments were dropped due to a CSD or CSV before the start of current Fragment 3: FRAGMENT_CORRUPT_AFTER_GAP: The current Fragment is corrupted as a CSD or CSV occurred after the start of current Fragment transfer. In addition, one or more Fragments were dropped due to a CSD or CSV before the start of current Fragment. The signal is set to '0' or '2' at the start of current Fragment depending on whether the previous Fragment was correctly delivered or not. Once the signal changes from '0' to '1', it stays constant until t_rx_eof is set to '1'. Once the signal changes from a value '2' to '3', it stays constant until t_rx_eof is set to '1'.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Signal Name	Driver	Description
T_CO_FLOWCONTROL.req Group		
t_fc_valid	Application	Flow Control Valid When '1', this signal indicates that t_fc_credits is valid, and will remain valid, until it is accepted by the CPort by setting t_fc_accept to '1'. The data is transferred when t_fc_valid and t_fc_accept are both '1'.
t_fc_accept	CPort	Flow Control Accept When '1', this signal indicates that the CPort is ready to accept credits. The CPort may set t_fc_accept to '1' before t_fc_valid is set '1'.
t_fc_credits[F:0]	Application	Flow Control Credits This signal indicates the number of credits for a CPort. The credits represent the free buffer space in bytes that became available from the last time t_fc_credits was asserted for the same CPort. The signal width, F + 1, is implementation-specific, and relates to the number of credits that can be issued at one time. When more credits than the t_fc_credits capacity needs to be transferred, the t_fc_credits signal must be asserted multiple times.
t_fc_credits_accepted[F:0]	CPort	Flow Control Credits Accepted This signal is valid one cycle after t_fc_credits was transferred, and indicates the number of credits accepted by the CPort. The credit transfer is correct when t_fc_credits_accepted is equal to t_fc_credits from the previous cycle (equivalent to the SUCCESS value of L4CPortResultCode of T_CO_FLOWCONTROL.cnf_L). The credit transfer is erroneous when t_fc_credits_accepted is less than t_fc_credits from the previous cycle (equivalent to the CREDITS_EXCEEDED value of L4CPortResultCode of T_CO_FLOWCONTROL.cnf_L). The width of the t_fc_credits_accepted signal is identical to the t_fc_credits signal width.
t_fc_result[2:0]	CPort	Flow Control Result This is the CPort response to the Application's T_CO_FLOWCONTROL.req. This result code is valid one cycle after the t_fc_valid and t_fc_accept were asserted. t_fc_result uses the values for L4CPortResultCode as shown ***Table 68***. The following values are valid: 0: SUCCESS: The data was received by the CPort successfully 1: NO_CONNECTION: The given local CPort number is not connected 2: CREDITS_EXCEEDED: The given credits exceed the maximum number of allowed credits for the given local CPort number 4 to 7: Invalid values

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

D.2.2 Timing Diagrams

This section illustrates the functionality of the non-multiplexed CPort signal interface with timing diagrams.

Figure 150 shows an example of data flow to two different CPorts. In this case, two instances of the T_CO_SAP are used. All T_CO_DATA.req group signals are duplicated, with one set for each CPort. The UniPro stack has a dedicated buffer for each CPort, and can accept data from the Application simultaneously for both CPorts.

From Cycle 3 the Application is ready to send data to CPort 1 on t_tx_data(1). The Application sets t_tx_valid(1) and t_tx_data(1). The UniPro stack in this example could have accepted data for CPort 1 during Cycle 1, if it set t_tx_accept(1) to ‘1’ at the start of the Cycle. On Cycle 3, both t_tx_valid(1) and t_tx_accept(1) are set to ‘1’, and data can be passed between the Application and CPort 1 until Cycle 9, when t_tx_eof(1) is set to ‘1’, indicating the end of the Fragment.

The second interface, for CPort 2, sets t_tx_valid(2) to ‘1’ on Cycle 6, and transmits data to CPort(2) from Cycle 6 to Cycle 10 (as the CPort sets t_tx_accept(2) to ‘1’ on Cycle 6). Note that t_tx_accept(2) stops only the Fragment on t_tx_data(2) for one cycle.

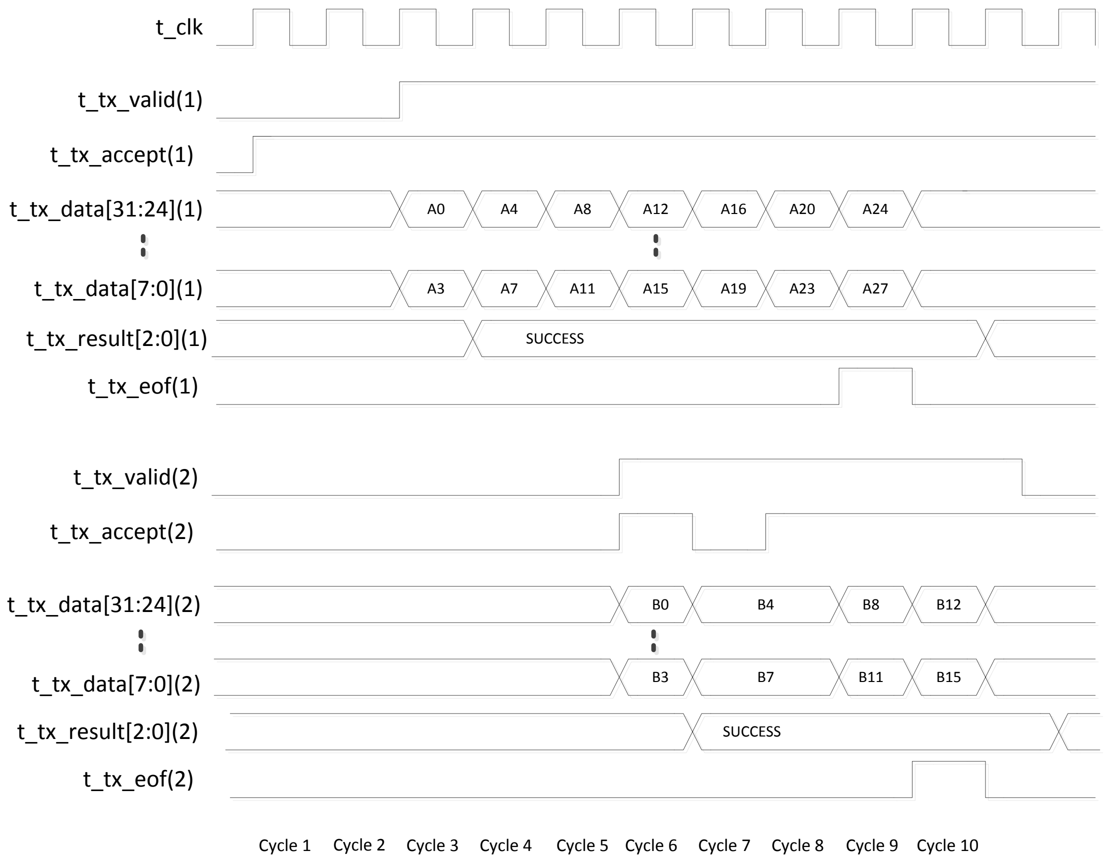


Figure 150 Example of Data Transmitted Simultaneously From Two CPorts

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Figure 151 shows an example of a data flow to two different CPorts. The Application sets t_tx_valid(1) to '1' on Cycle 1 and t_tx_valid(2) to '1' on Cycle 2, to start sending data on t_tx_data(1) and t_tx_data(2) from Cycle 1 and Cycle 2, respectively.

In this case, the UniPro stack can accept data for one CPort at a time. To control data flow, the UniPro stack can set only one t_tx_accept signal to '1' at a time; the remaining t_tx_accept signal is set to '0'. At Cycle 3 t_tx_accept(2) is set to '1' until the end of t_tx_data(2), t_tx_eof(2) Fragment (Cycle 6). At that time t_tx_accept(2) is set to '0' and t_tx_accept(1) is set to '1' to allow data transfer for t_tx_data(1).

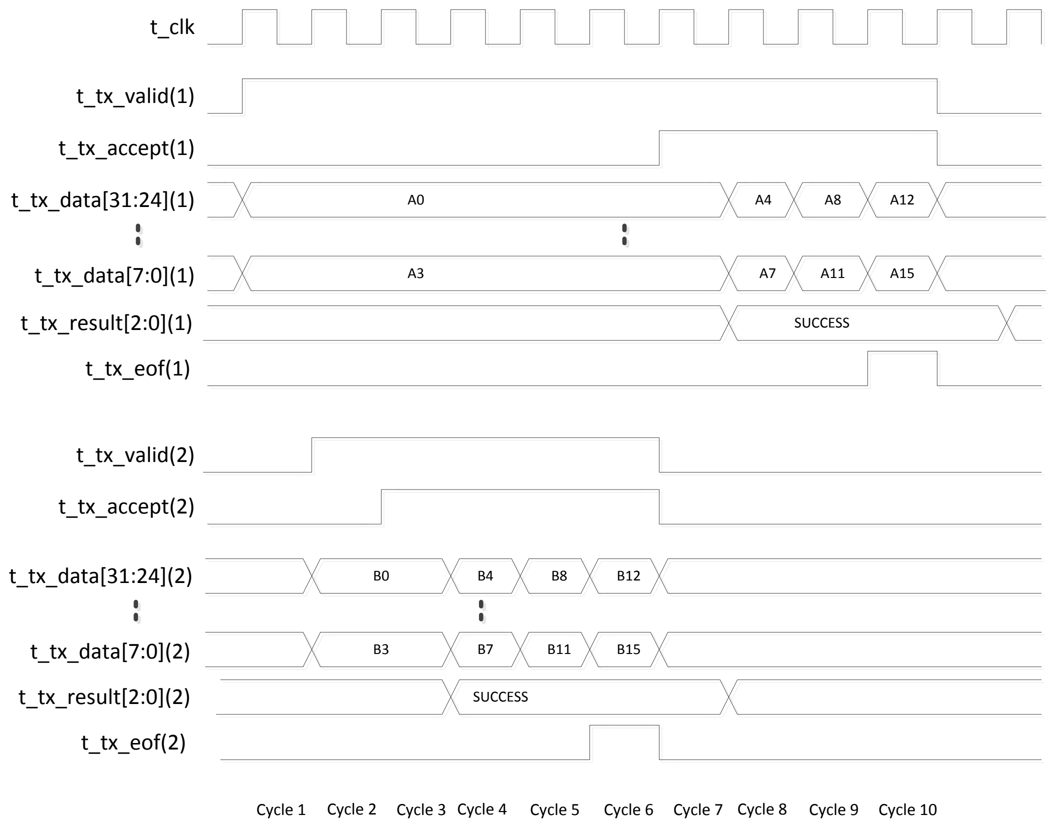


Figure 151 Example of Data Transmitted from One CPort at a Time

For T_CO_DATA.ind and T_CO_FLOWCONTROL.req interfaces example, refer to ***Figure 146*** and ***Figure 147***. These interfaces do not depend on the method of arbitration.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

This page intentionally left blank.

7099

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Annex E Options and Tunable Parameters (informative)

This annex is targeted to writers of a specification using UniPro. It collects the most significant "design time" options and parameters, that can be used, tuned or restricted by a specification using UniPro, but also gives guidelines for IP suppliers or users. Those options and parameters are constant for a given UniPro implementation and cannot be changed at run-time. Nothing new is added by this annex to the UniPro behavior since it merely list points already presented in other sections of this document. The intent of this annex is to make it easier for users of the UniPro Specification to verify that no major decision point was overlooked when defining a specification on top of UniPro, or that no essential point was overlooked when considering the design, use or re-use of IPs.

Note that only options or tunable parameters found in the UniPro Specification are covered in this section since the targeted audience is writers of specifications defined on top of UniPro.

E.1 PHY Adapter Layer (L1.5)

For the PHY Adapter Layer, the following options and tunable parameters are defined:

- The PHY Type, defined by PA_PHY_Type
- Number of supported Lanes for TX and RX, defined by PA_AvailTxDataLanes and PA_AvailRxDataLanes, respectively
- Access to Attributes of the peer Device
- PHY Layer test mode

E.1.1 Access to Attributes of a Peer Device

With the PACP protocol, the DME can access, via the local PHY Adapter Layer, Attributes of the peer Device. Even if the capability to request access of peer Device Attributes is optional for a single Device, at least one Device on each UniPro Link, usually the Host, needs this capability. Thus, a Device is always capable of receiving a request from a peer Device to access local Attributes, process the request and return a result.

E.1.2 PHY Test Mode

With the PACP protocol, the PA Layer can instruct the peer Device to be in a given test mode. The capability to send this request to the peer Device is optional, however the capability of receiving such a request from the peer Device is mandatory.

E.2 Data Link Layer (L2)

For the Data Link Layer, the following options and tunable parameters are defined:

- Number of, and which, Traffic Classes are supported
- Size of TX and RX buffers
- The maximum supported Frame size
- Support for preemption

E.2.1 Supported Traffic Classes

UniPro leaves the choice to Applications to use one or two Traffic Classes. However, if an Application implements only one Traffic Class, the Traffic Class is TC0. The value of DL_TC1RxInitCreditVal defines if TC1 is implemented. If this Attribute has a value of zero, TC1 is not implemented.

Note that TC1 is intended for low-latency, low-bandwidth traffic. This could be restricted and enforced by Switches to use, at maximum, a certain percentage of a UniPro Link. The percentage could be a system level configurable parameter. Therefore, to ensure compatibility with future networks, Applications needing high bandwidth for certain traffic are advised to use TC0 for such traffic.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

E.2.2 TX and RX Buffer Sizes

The size of the RX buffer is defined by DL_TC0RxInitCreditVal and DL_TC1RxInitCreditVal for TC0 and TC1, respectively. A specification using UniPro could be more restrictive in what values are expected from a UniPro implementation it uses. Note that reducing the RX buffer size might reduce cost, but there might be a non-negligible impact on performance at high bandwidth, especially if the average Frame size is small.

The size of the TX buffer is defined by DL_TC0TxBufferSize and DL_TC1TxBufferSize for TC0 and TC1, respectively. Those values do not impact the protocol behavior of UniPro, but might have an impact on the performance, depending on hardware and software partitioning, Application behavior, etc.

E.2.3 Maximum Supported Frame Size

A UniPro implementation can choose to support a smaller Frame size than the maximum allowed Frame size. The Frame size is indicated by DL_TC0TxMaxSDUSize and DL_TC1TxMaxSDUSize for TC0 and TC1, respectively. Note there does not seem to be any practical reason to mandate a smaller Frame size, but a specification could mandate support for at least a specific size Frame or support for the full Frame size indicated by DL_MTU.

E.2.4 Support for Preemption

An implementation can choose to support preemption in the TX side. Support for preemption is indicated by DL_TxPreemptionCap. An Application can mandate a specific value of this Attribute.

E.3 Network Layer (L3)

For the Network Layer the following options and tunable parameters are defined:

- Maximum supported Packet size

E.3.1 Maximum Supported Packet Size

A UniPro implementation can choose to support a smaller Packet size than the maximum allowed Packet size. The Packet size is indicated by N_TC0TxMaxSDUSize and N_TC1TxMaxSDUSize for TC0 and TC1, respectively. Note there does not seem to be any practical reason to mandate a smaller Packet size, but a specification could mandate support for at least a specific Packet size or support for the full Packet size indicated by N_MTU.

E.4 Transport Layer (L4)

For the Transport Layer, the following options and tunable parameters are defined:

- Number of CPorts
- Number of Test Features
- CPort arbitration algorithm
- Maximum supported Segment size
- Static Connections

E.4.1 Number of CPorts

The number of CPorts present in an implementation is given by the value of T_NumCPorts. This is one of the critical parameters that any Application designed on top of UniPro should define clearly, since it defines the limits of concurrent L4 Connections possible at any given time.

To allow UniPro Devices with multiple Functions, a specification should not mandate a maximum number of CPorts allowed for a given UniPro implementation, since it artificially limits the possible reuse of such a specification for building multiple Function Devices. However, a specification using UniPro should define the minimum number of CPorts expected from a UniPro implementation.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

E.4.2 Number of Test Features

The number of Test Features present in an implementation is given by the value of T_NumTestFeatures. This value is not critical for the protocol behavior of an Application on top of a UniPro stack, but is a very important parameter impacting the level of testability of a UniPro implementation. Conformance tests defined for UniPro rely heavily on the availability of Test Features.

E.4.3 CPort Arbitration Algorithm

Round Robin arbitration is mandated by the UniPro Specification, but there is the option to add any other arbitration algorithm of the CPorts. An Application on top of a UniPro stack could mandate the addition of another scheme, without violating the UniPro Specification, as long as Round Robin arbitration is also provided.

E.4.4 Maximum Supported Segment Size

A UniPro implementation can choose to support a smaller Segment size than the maximum allowed Segment size. The Segment size is indicated by T_TC0TxMaxSDUSize and T_TC1TxMaxSDUSize for TC0 and TC1, respectively. Note that as for the maximum Packet and Frame size, there does not seem to be any practical reason to mandate a smaller Segment size, but a specification could mandate support for at least a specific Segment size or support the full Segment size indicated by T_MTU.

E.4.5 Static Connections

The static configuration of a UniPro Device (see ***Annex E.6***) can be used to define static L4 Connections.

E.5 Device Management Entity

The Device Management Entity has optional Service Primitives to power on and power off the UniPro stack and access Attributes of a peer Device.

E.5.1 Powering On or Off the UniPro Stack

DME_POWERON and DME_POWEROFF Service Primitives are optional. DME_POWERON.req and DME_POWERON.cnf_L are used to power on the UniPro stack, while DME_POWEROFF.req and DME_POWEROFF.cnf_L are used to power off the UniPro stack.

E.5.2 Access to Attributes of a Peer Device

DME_PEER_GET and DME_PEER_SET Service Primitives are optional. Note that the same condition as stated in ***Annex E.1.1*** applies here as well.

E.5.3 Testing Mode

DME_TEST_MODE.req and DME_TEST_MODE.ind are specifically used in testing mode and are not intended for normal Application use.

E.6 Static Configuration of a UniPro Device

A UniPro Device can be statically configured, utilizing the non-volatile reset values, e.g., to configure a static L4 Connection. This means that after reset the value of these Attributes is automatically set, without requiring any action from an Application on top of the UniPro stack, to their corresponding reset value, which can be different from their corresponding default value defined for the Attribute. For a gettable or settable Attribute that is implemented with a non-volatile reset value, an Application on top of a UniPro stack can set and change the reset value as desired by the Application.

Note that UniPro provides a standard method to write a non-volatile reset value through the DME_SET.req and DME_PEER_SET.req primitives.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

This page intentionally left blank.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Annex F Recommended Pre-condition Messages for Hibernate Entry and Exit (informative)

This annex provides an informative description of the pre-condition Messages that are exchanged at the Application level during Hibernate Entry Phase 1 flow. The pre-condition Messages ensure that no additional Messages are transmitted and received by the UniPro stack and all potential Links are identified to hibernate between the two UniPro Devices.

There are two types of pre-condition Messages initiated by an Application, inactivity Messages and Link information Messages.

Inactivity pre-condition Messages allow Applications to exchange inactivity states of the local and peer Applications. The protocol for detecting inactivity or idleness should be defined by the Application. This Message might be detected by using an inactivity timer to detect idleness of the Application. When the inactivity timer expires, an Application Attribute is set to TRUE indicating that the Application is in an idle state. The inactivity pre-condition Message can read this Attribute and use it to detect inactivity in the Application. The initial value of an inactivity timer might be restored once an Application exits hibernate. An inactivity pre-condition Message is intended to allow easier entry into hibernate. Once inactivity of a Link is detected, the Application strengthens its confidence that no additional Segments, Packets or Frames are outstanding in the UniPro stack.

Link information-related pre-condition Messages might be used to communicate which Links need to be hibernated for hibernate Links between two Devices. For example, Device B in ***Section 9.5*** sent a pre-condition Message commanding Device A to initiate hibernate between Device A and Device B. The protocol for sharing Link information should be defined by the Application.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

This page intentionally left blank.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Annex G Alternative High Level Power Mode Change Control (informative)

This annex provides an informative description of an alternative way for implementing high level Power Mode Change using only the DME_SET primitive.

An implementation that wants to change the power mode of the UniPro stack using DME_SET, rather than DME_POWERMODE, requires additional DME Attributes to hold the L2 timer values for the new power mode. These Attributes are used to set the L2 timer values during the power mode change process at the time PA_LM_PWR_MODE.ind is received.

If DME_POWERMODE is used as described in ***Section9***, the parameters for DME_POWERMODE need to be provided as configurable items at the Application Layer.

The solution presented in this annex is for the case where changing the power mode is done using only DME_SET Service Primitives in order to simplify the controlling Application. In such a case, the proposed solution can be utilized to avoid proprietary DME_SET-based mechanisms.

The Attributes shown in ***Table 110*** are used for L2 timers of the local UniPro stack in a locally initiated power mode change. In a remote power mode change, the value of the Attributes are received through the PA Layer during the power mode change process.

Table 110 DME Attributes for DME_SET-based High Level Power Mode Control

Attribute	Attribute ID[1]	Description	Type	Unit	Valid Attribute Values	Mandatory Values	Reset Value
DME_LocalFC0ProtectionTimeOutVal	0xD041	Stores the FC protection timeout value for TC0 applied with the next power mode change	Integer	µs	0 (OFF), 1 to 65535		8191
DME_LocalTC0ReplayTimeOutVal	0xD042	Stores the Replay timeout value for TC0 applied with the next power mode change	Integer	µs	0 (OFF), 1 to 65535		65535
DME_LocalAFC0ReqTimeOutVal	0xD043	Stores the request timeout value for TC0 applied with the next power mode change	Integer	µs	0 (OFF), 1 to 65535		32767
DME_LocalFC1ProtectionTimeOutVal	0xD044	Stores the FC protection timeout value for TC1 applied with the next power mode change	Integer	µs	0 (OFF), 1 to 65535		8191
DME_LocalTC1ReplayTimeOutVal	0xD045	Stores the Replay timeout value for TC1 applied with the next power mode change	Integer	µs	0 (OFF), 1 to 65535		65535
DME_LocalAFC1ReqTimeOutVal	0xD046	Stores the request timeout value for TC1 applied with the next power mode change	Integer	µs	0 (OFF), 1 to 65535		32767

1. *Recommended vendor-specific values.*

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

A power mode change is structured in several accesses to the UniPro stack. As a first step, the local PHY Adapter Layer Attributes shown in ***Table 111*** are set.

Table 111 PHY Adapter Layer Attributes for Alternative Power Mode Change

Attribute	Attribute ID	Reference
PA_TxGear	0x1568	***Table 28***
PA_TxTermination	0x1569	
PA_HSSeries	0x156A	
PA_ActiveTxDataLanes	0x1560	***Table 24***
PA_RxGear	0x1583	***Table 28***
PA_RxTermination	0x1584	
PA_ActiveRxDataLanes	0x1580	***Table 24***

These Attributes are stored in specific registers as they are not directly applied to the current configuration. These Attributes are also stored locally in the PA Layer, but not forwarded to the PHY Layer until the power mode change procedure is successful.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

After initializing the PA Layer, it is necessary to program the Data Link Layer timer values for the local Device and the peer Device using the Attributes in ***Table 112***. These values become effective after a successful power mode change request. It is not necessary to set Attributes for TC1 if only one traffic class is supported.

Table 112 Data Link Layer Attributes for Alternative Power Mode Change

Attribute	Attribute ID	Location	Description
DME_LocalFC0ProtectionTimeOutVal	0xD041	Local	–
DME_LocalTC0ReplayTimeOutVal	0xD042	Local	–
DME_LocalAFC0ReqTimeOutVal	0xD043	Local	–
DME_LocalFC1ProtectionTimeOutVal	0xD044	Local	Not used if only a single Traffic Class is supported.
DME_LocalTC1ReplayTimeOutVal	0xD045	Local	Not used if only a single Traffic Class is supported.
DME_LocalAFC1ReqTimeOutVal	0xD046	Local	Not used if only a single Traffic Class is supported.
DL_FC0ProtectionTimeOutVal	0x15B0	Peer	Set using PAPowerModeUserData[0] of a PACP_PWR_req frame.
DL_TC0ReplayTimeOutVal	0x15B1	Peer	Set using PAPowerModeUserData[1] of a PACP_PWR_req frame.
DL_AFC0ReqTimeOutVal	0x15B2	Peer	Set using PAPowerModeUserData[2] of a PACP_PWR_req frame.
DL_FC1ProtectionTimeOutVal	0x15B3	Peer	Set using PAPowerModeUserData[3] of a PACP_PWR_req frame.
DL_TC1ReplayTimeOutVal	0x15B4	Peer	Set using PAPowerModeUserData[4] of a PACP_PWR_req frame.
DL_AFC1ReqTimeOutVal	0x15B5	Peer	Set using PAPowerModeUserData[5] of a PACP_PWR_req frame.

Finally, the power mode change request is started by accessing PA_PWRMode (L1.5, 0x1571). This Attribute contains the requested transmission mode, fast or slow.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

Participants

The list below includes those persons who participated in the Working Group that developed this Specification and who consented to appear on this list.

Rachel Aseriel, Cadence Design Systems, Inc.
Björn Behrens, Toshiba Corporation
Rajesh Bhaskar, Intel Corporation
Tony Carosa, Protocol Insight, LLC
Roy Chestnut, Teledyne LeCroy
Hsien-Ho Chuang, MediaTek Inc.
Marek Cieplucha, Cadence Design Systems, Inc.
Asher Druck, SanDisk Corporation
Freeda Fernando, Arasan Chip Systems, Inc.
Naveen Gangaiah, Synopsys, Inc.
Gordon Getty, Teledyne LeCroy
Ramesh Hanchinal, Synopsys, Inc.
Edward Hsieh, Silicon Motion, Inc.
Shiuan Jing Yeh, MediaTek Inc.
Lokesh Kabra, Synopsys, Inc.
Seongyoon Kim, Samsung Electronics, Co.
Hendrik Lange, Toshiba Corporation
Sangheon Lee, Samsung Electronics, Co.
Larry Madar, Google, Inc.
Vishnuvardhan Reddy Mandala, Synopsys, Inc.
Ofir Michaeli, Cadence Design Systems, Inc.
Aarthi Nadarajan, Intel Corporation
Shyama S Nair, Intel Corporation
Ross Nelson, Protocol Insight, LLC
Laura Nixon, MIPI Alliance
Josh Pan, MediaTek Inc.
Rajeev Ramanath, Google, Inc.
Roland Scherzinger, Keysight Technologies Inc.
Yoni Shternhell, SanDisk Corporation
Olin Sibert, Google Inc.
Jürgen Urban, Toshiba Corporation
Hung Vuong, Qualcomm Incorporated

Past contributors to UniPro version 1.61 and earlier:

Radha Atukula, NVIDIA
Rajesh Bhaskar, Intel Corporation
Hicham Bouzekri, STMicroelectronics
Tony Carosa, Protocol Insight LLC
Insiya Challawala, Synopsys, Inc.
Roy Chestnut, LeCroy Corporation
Hsien-Ho Chuang, MediaTek Inc.
Marek Cieplucha, Cadence Design Systems, Inc.
Gary Debes, Texas Instruments Incorporated
Michel Gillet, Nokia Corporation
Glen Hambro, MIPI Alliance Program Manager
Ramesh Hanchinal, Synopsys, Inc.
Lokesh Kabra, Synopsys, Inc.
Sanjay Karambale, NVIDIA
Pekka Korpinen, Nokia Corporation
Manoz Krovvidy, NVIDIA
Ariel Lasry, Toshiba Corporation
Tim Leuchter, Toshiba Corporation
Kumaravel Manickam, GDA Technologies
Ofir Michaeli, Cadence Design Systems, Inc.
Shyama S Nair, Intel Corporation
Ross Nelson, Protocol Insight LLC
Laura Nixon, MIPI Alliance Program Manager
Josh Pan, MediaTek Inc.
Chang-Jae Park, Samsung Electronics, Co.
Juha Rakkola, Nokia Corporation
Rajeev Ramanath, Google, Inc.
Yoram Rimoni, Qualcomm Incorporated
Roland Scherzinger, Keysight Technologies Inc.
Olin Sibert, Google, Inc.
Tom Sidle, MIPI Alliance Technical Editor
Chad Slaugh, Keysight Technologies Inc.
Dong Hyun Song, SK Hynix
Juergen Urban, Toshiba Corporation
John Wiedemeier, LeCroy Corporation

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

This page intentionally left blank.

Copyright © 2007-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential