ANSI/IEEE Std 802.12, 1998 Edition

(Incorporating ANSI/IEEE Stds 802.12-1995 and
ANSI/IEEE Std 802.12d-1997)

(Adopted by ISO/IEC and redesignated as
ISO/IEC 8802-12:1998)

IEEE Standard for Information technology—

Telecommunications and information exchange between systems—

Local and metropolitan area networks—

Specific requirements

Part 12: Demand-Priority access method, physical layer and repeater specifications

**Adopted by the ISO/IEC and redesignated as
ISO/IEC 8802-12:1998**

Sponsor

LAN/MAN Standards Committee
of the
IEEE Computer Society

Abstract: The media access control characteristics for the demand-priority access method are specified. The layer management, physical layers, and media that support this access method are also specified. Layer and sublayer interface specifications are aligned to the ISO Open Systems Interconnection Basic Reference Model and ISO/IEC 8802 models. Specifications for 100 Mb/s operation over 100 Ω balanced cable (twisted-pair) Categories 3 through 5, 150 Ω shielded balanced cable, and fibre-optic media are included. Optional implementation of redundant links to facilitate automatic recovery of network connectivity in case of link or repeater failure anywhere in the network path is specified. Rules for connecting redundant links within a network are defined.
Keywords: balanced cable, data processing, demand-priority access method, fault tolerance, fibre optic, information interchange, local area network, media access control, models, mode of data transmission, network interconnection, redundant link, repeater, star-topology local area network, twisted pair, network interconnection

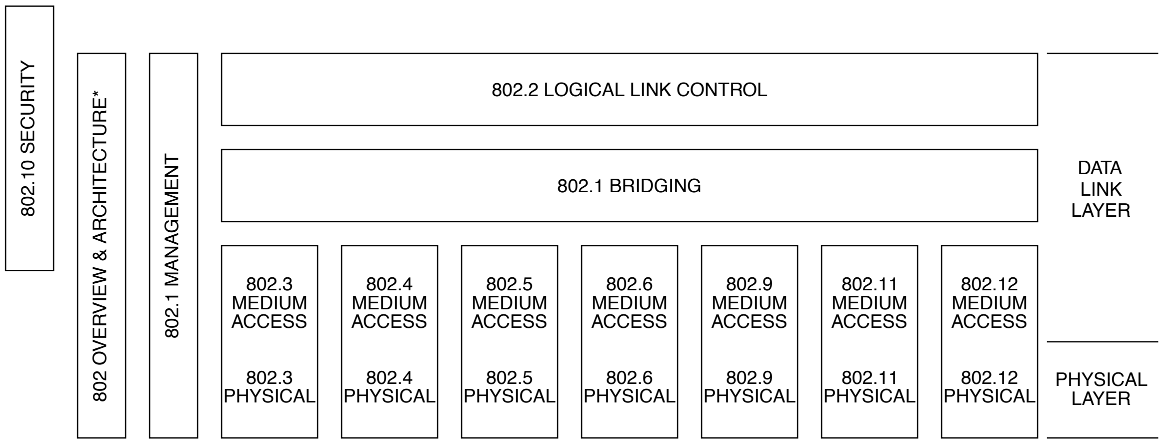
The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA

ISBN 0-7381-1487-1

Introduction to ANSI/IEEE Std 802.12, 1998 Edition

[This is not part of ANSI/IEEE Std 802.12, 1998 Edition, Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 12: Demand-Priority access method, physical layer and repeater specifications.]

This standard is part of a family of standards for local and metropolitan area networks. The relationship between the standard and other members of the family is shown below. (The numbers in the figure refer to IEEE standard numbers.)



* Formerly IEEE Std 802.1A.

This family of standards deals with the Physical and Data Link layers as defined by the International Organization for Standardization (ISO) Open Systems Interconnection (OSI) Basic Reference Model (ISO/IEC 7498-1 : 1994). The access standards define seven types of medium access technologies and associated physical media, each appropriate for particular applications or system objectives. Other types are under investigation.

The standards defining the technologies noted above are as follows:

• IEEE Std 802	*Overview and Architecture*. This standard provides an overview to the family of IEEE 802 Standards.
• ANSI/IEEE Std 802.1B and 802.1k [ISO/IEC 15802-2]	*LAN/MAN Management*. Defines an OSI management-compatible architecture, and services and protocol elements for use in a LAN/MAN environment for performing remote management.
• ANSI/IEEE Std 802.1D [ISO/IEC DIS 15802-3]	*Media Access Control (MAC) Bridges*. Specifies an architecture and protocol for the interconnection of IEEE 802 LANs below the MAC service boundary.
• ANSI/IEEE Std 802.1E [ISO/IEC 15802-4]	*System Load Protocol*. Specifies a set of services and protocol for those aspects of management concerned with the loading of systems on IEEE 802 LANs.
• ANSI/IEEE Std 802.1F	*Common Definitions and Procedures for IEEE 802 Management Information*
• ANSI/IEEE Std 802.1G [ISO/IEC 15802-5]	*Remote Media Access Control (MAC) bridging*. Specifies extensions for the interconnection, using non-LAN communication technologies, of geographically separated IEEE 802 LANs below the level of the logical link control protocol.
• ANSI/IEEE Std 802.2 [ISO/IEC 8802-2]	*Logical link control*

• ANSI/IEEE Std 802.3 [ISO/IEC 8802-3]	*CSMA/CD access method and physical layer specifications*
• ANSI/IEEE Std 802.4 [ISO/IEC 8802-4]	*Token passing bus access method and physical layer specifications*
• ANSI/IEEE Std 802.5 [ISO/IEC 8802-5]	*Token ring access method and physical layer specifications*
• ANSI/IEEE Std 802.6 [ISO/IEC 8802-6]	*Distributed Queue Dual Bus (DQDB) access method and physical layer specifications*
• ANSI/IEEE Std 802.9 [ISO/IEC 8802-9]	*Integrated Services (IS) LAN Interface at the Medium Access Control (MAC) and Physical (PHY) Layers*
• ANSI/IEEE Std 802.10	*Interoperable LAN/MAN Security*
• ANSI/IEEE Std 802.11 [ISO/IEC 8802-11]	*Wireless LAN Medium Access Control (MAC) and physical layer specifications*
• ANSI/IEEE Std 802.12 [ISO/IEC 8802-12]	*Demand-priority access method, physical layer and repeater specifications*

In addition to the family of standards, the following is a recommended practice for a common Physical Layer technology:

• IEEE Std 802.7	*IEEE Recommended Practice for Broadband Local Area Networks*

The following additional working group has authorized standards projects under development:

• IEEE 802.14	*Standard Protocol for Cable-TV Based Broadband Communication Network*

Conformance test methodology

An additional standards series, identified by the number 1802, has been established to identify the conformance test methodology documents for the 802 family of standards. Thus the conformance test documents for 802.3 are numbered 1802.3.

ANSI/IEEE Std 802.12, 1998 Edition (ISO/IEC 8802-12: 1998)

This standard defines the protocol and compatible interconnection of data communication equipment via a repeater-controlled, star-topology Local Area Network (LAN) using the demand-priority access method. It includes a 100 Mb/s implementation with Physical Layers for four-pair 100 Ω balanced cable Categories 3 through 5; two-pair 150 Ω shielded balanced cable; and fibre optic cable. Work is currently under way in the following areas: higher speed operation; two-pair Category 5 100 Ω balanced cable; and full-duplex operation. Optional implementation of redundant links to provide automatic recovery of network connectivity in case of link or repeater failure anywhere in the network path is specified. Rules and recommendations for connecting redundant links are provided.

Participants

The original working group included the following members at the time IEEE Std 802.12 was approved:

Pat Thaler, *Chair* **William G. Lane,** *Technical Editor*

Alan Albrecht
Ian Atkinson
Steve Barilovits
Robert Baumert
John Bestel
Heinz C. Blennemann
Warren Burnett
Kiwon Chang
Samuel Chang
Alistair Coles
Ronald J. Cooper*
Simon Crouch
David Cunningham
Joe Curcio
Charles J. Daniels
Sanchaita Datta
Remi Despres
Thuyen Dinh
Jonathan Edney
Dean M. Edwards
Bob Faulk
Juan Figueroa
Andrew Frank
Steve Goody
John Griesing
John Grinham
Karunakar Gulukota
Del Hanson
Terrence Harte
Susan Hennenfent
Kenneth Huang
Mike Hughes
Allen Jett
Dieter W. Junkers
Jayant Kadambi
Allen Kasey
Duane Kuang
Hans Lackner
Robert H. Leonowich*
Aaron Lepold
Chen-De Lin
James Little
Jonathan Lo
Andy Luque
Peter Martini
John Messenger*
John E. Montague
Teruo Moriguchi
Henry Ngai
Paul Nikolich
Jerry Pate
Peter Rautenberg
Everett O. Rigsbee
Dan Scavezze
James Schooler
Anthony Seaman
Koichiro Seto
Amit Shah
K. Karl Shimada
Som Sikdar
Ramesh Sivakolundu
Michael Spratt
Andre Szczepanek
Sadry Tavana
Mark Thompson
Zbigniew (Bish) Turlej
Bob Watson
Greg Watson
Alan Weissberger
James Welch
Kathleen M. Wilhelm
Edward Wong
James Young

*Task Force Chair

The following persons also made significant contributions to the development of the original edition:

Alan Chambers
Jacques Christ
Brice Clark
Daniel Dove
Robert Gudz
Lee Haas
Jonathan Jedwab
Clarence Joh
William Kind
Josef Kozilek
Sam Madani
Tim McShane
Steve Methley
Jörg Ottensmeyer
Dinah Sloan

The following persons were on the balloting committee for IEEE Std 802.12:

Don Aelmore
Bernhard Albert
Alan Albrecht
Abe Ali
Hasan S. Alkhatib
Kit Athul
William E. Ayen
Kendall F. Barney
Simon Black
Kwame Boakye
Kathleen L. Briggs
Peter K. Campbell
James Carlo
Alan J. Chwick
Alistair Coles
Ian Crayford
Robert S. Crowder
Joe Curcio
Ibibia K. Dabipi
Robert Donnan
Daniel Dove
Edward A. Dunlop
John E. Emrich
Alvin W. Eng
Philip H. Enslow, Jr.
Changxin Fan
Robert L. Faulk
John W. Fendrich
David Fifield
Michael Fischer
Christian G. Folting
Howard M. Frazier
Harvey A Freeman
Robert J. Gagliano
D. G. Gan
Harry Gold
Patrick Gonia
Jacob J. Hsu
Howard Johnson
Henry D. Keen
Gary C. Kessler
Mladen Kezunovic
Yongbum Kim
Mikio Kiyono
Kenneth C. Kung
David Law
Lanse M. Leach
Randolph S. Little
Donald C. Loughry
Robert D. Love
William C. Lynch
Peter Martini
William C. McDonald
Tim J. McShane
Bennett Meyer
Colin K. Mick
Ann Miller

IEEE Std 802.12-1995 was approved by the American National Standards Institute on 14 March 1996. IEEE Std 802.12d-1997 was approved by the American National Standards Institute on 20 August 1997.

Contents

Information technology— Telecommunications and information exchange between systems— Local and metropolitan area networks— Specific requirements—

Part 12: Demand-Priority access method, physical layer and repeater specifications

1. Overview

1.1 Scope

This International Standard defines the protocol and compatible interconnection of data communication equipment via a repeater-controlled, star-topology Local Area Network (LAN) using the demand-priority access method.

1.2 Purpose

The purpose of this protocol is to provide a higher speed LAN with deterministic access, priority, and optional filtering. Pursuant to this, the protocol will

a) Provide a minimum data rate of 100 Mb/s.

b) Provide smooth migration from ISO/IEC 8802-3 and ISO/IEC 8802-5 LANs.

c) Support either ISO/IEC 8802-3 or ISO/IEC 8802-5 frame format and MAC service interface to the LLC.

d) Support a cascaded star topology over twisted pair and fibre-optic generic building wiring.

e) Allow topologies of 2.5 km and greater with three levels of cascading.

f) Provide a Physical Layer Bit Error Rate (BER) of less than 10^{-8}.

g) Provide fair access and bounded latency.

h) Provide two priority levels: normal and high.

i) Provide a low-latency service through high priority for support of multimedia applications over extended networks.

j) Support an option for filtering individually addressed packets at the repeater to enhance privacy.

k) Support network management to monitor network performance, isolate faults, and control network configuration.
l) Enable low-cost implementation and high levels of integration.
m) Provide for robust operation by testing the Physical Layer connection before allowing an end node to enter the network, and by removing disruptive nodes.
n) Support an option for redundant links that will enable automatic recovery of network connectivity in case of link or repeater failure anywhere in the network path.

1.3 Document organization

The clauses and annexes of this International Standard provide an overview and define the requirements of the demand-priority protocol and its supporting physical layers.

This International Standard is organized as follows:

Clauses 1 through 4 contain the overview, references, definitions, and abbreviations and acronyms, respectively.

Clause 5 (Notation) defines the various notations that are used within this International Standard.

Clause 6 (Introduction to the protocol) shows the relationship of the protocol to the IEEE 802 family of standards and defines the architectural structure of the network end nodes and repeaters. An overview of the operation of a demand-priority LAN is also provided.

Clause 7 (MAC service specifications) defines the interface services provided by the Medium Access Control (MAC) sublayer to the Logical Link Control (LLC) sublayer.

Clause 8 (PMI service specifications) defines the interface services provided by the Physical Medium Independent (PMI) sublayer to the MAC and Repeater Medium Access Control (RMAC) entities.

Clause 9 (PMD service specifications) defines the interface services provided by the Physical Medium Dependent (PMD) sublayer to the PMI.

Clause 10 (MAC frame format structure) defines the MAC frame formats, including addressing conventions, data-length restrictions, Frame Check Sequence (FCS) generation, and training frame requirements.

Clause 11 (MAC protocol) defines the MAC protocol. Finite state machines and pseudo code descriptions are supplemented with prose annotations. MAC-based operational counters, error counters, and timers are described. Link initiation and training are specified.

Clause 12 (RMAC protocol) defines the functional control requirements and services of the RMAC sublayer in the network repeater. Fault detection, training, and recovery procedures for operation in single and multiple repeater networks are described.

Clause 13 (Layer management functions and services) defines the functional requirements and recommendations for the optional Layer Management Entity (LME), including direct and indirect interrogation of operation and error counters, and maintenance of operational and error statistics.

Clause 14 (PMI specifications) specifies the PMI portion of the Physical Layer. This includes data-symbol scrambling, descrambling, encoding, and decoding requirements for the PMI. Four-channel bit-stream formats are defined.

Clause 15 (MII specifications) defines the functional characteristics of the Medium Independent Interface (MII) and, additionally, provides electrical specifications for an optional, physically exposed MII.

Clause 16 (4-UTP PMD, MDI, and link specifications) defines the 4-UTP PMD functional requirements, including timing, signal conditioning, and control signal recognition for link configurations with 100 Ω balanced cable. The electrical and mechanical requirements of the physically exposed 4-UTP Medium Dependent Interface (MDI) are defined.

Clause 17 (Dual simplex STP PMD, MDI, and link specifications) defines the STP PMD functional requirements, including timing, signal conditioning, and control signal recognition for link configurations with two-pair 150 Ω balanced cable. The electrical and mechanical requirements of the physically exposed STP MDI are defined.

Clause 18 (Dual simplex fibre-optic PMD, MDI, and link specifications) defines the fibre-optic PMD functional requirements, including timing, signal conditioning, and control signal recognition for link configurations with two optical fibre links. The optical, electrical, and mechanical requirements of the physically exposed fibre-optic MDI are defined.

Clause 19 (2-TP PMD, MDI, and link specifications), the specification of a PMD to support two twisted pair links, is reserved for future study.

Annex A (PICS proforma) provides a Protocol Implementation Conformance Statement (PICS) proforma in compliance with the relevant requirements, and in accordance with the relevant guidance, given in ISO/IEC 9646-2:1994[1] (normative).

Annex B (General environmental and safety specifications) defines the general environmental and safety requirements for the various network configurations (normative).

Annex C (GDMO specifications for demand-priority managed objects) defines the Guidelines for the Definition of Managed Objects (GDMO) notation used for layer management objects (normative if the optional LME is implemented).

Annex D (Allocation of object identifier values) contains a summary of all object identifier values that have been allocated for layer management purposes (normative if the optional LME is implemented).

Annex E (Network topology rules) defines topology rules for ISO/IEC 8802-12 networks (normative).

Annex F (Use of cabling systems with a differential characteristic impedance of 120 Ω nominal for 4-UTP links) provides information on an alternate balanced cable (informative).

Annex G (Bibliography) contains a bibliography that can supply additional information regarding specific subclauses of this standard (informative).

1.4 Application areas

The applications environment for the demand-priority LAN is intended to be commercial and light industrial.

[1]Information on references can be found in Clause 2.

1.5 Conformance requirements

Annex A contains PICS proforma definitions for demand-priority network components.

1.5.1 End node PICS proforma

The supplier of an end-node implementation that is claimed to conform to this standard shall complete a copy of the relevant PICS proforma in Annex A, including the information necessary to identify both the supplier and the implementation.

1.5.2 Repeater PICS proforma

The supplier of a repeater implementation that is claimed to conform to this standard shall complete a copy of the relevant PICS proforma in Annex A, including the information necessary to identify both the supplier and the implementation.

1.5.3 PMD PICS proforma

The supplier of a PMD implementation that is claimed to conform to this standard shall complete a copy of the relevant PICS proforma in Annex A, including the information necessary to identify both the supplier and the implementation.

2. Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this International Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of ISO and IEC maintain registers of currently valid International Standards.

CISPR 22:1993, Limits and methods of measurement of radio disturbance characteristics of information technology equipment.[2]

IEC 60060-1:1989, High-voltage test techniques—Part 1: General definitions and test requirements.[3]

IEC 60874-1:1993, Connectors for optical fibres and cables—Part 1: Generic specification.

IEC 60950:1991, Safety of information technology equipment, including electrical business equipment.

IEEE Std 802-1990, IEEE Standards for Local and Metropolitan Area Networks: Overview and Architecture (ANSI).[4]

ISO 4902:1989, Information technology—Data communication—37-pole DTE/DCE interface connector and contact number assignments.[5]

ISO TR 8509:1987, Information processing systems—Open Systems Interconnection—Service conventions.

ISO/IEC 7498-1:1984, Information technology—Open Systems Interconnection—Basic Reference Model—Part 1: The Basic Model.

ISO/IEC 7498-4:1989, Information processing systems—Open Systems Interconnection—Basic Reference Model—Part 4: Management framework.

ISO/IEC 8802-2:1998 [ANSI/IEEE Std 802.2, 1998 Edition], Information technology—Telecommunications and information exchange between systems—Local and Metropolitan area networks—Specific requirements—Part 2: Logical link control.[6]

ISO/IEC 8802-3:1996 [ANSI/IEEE Std 802.3, 1996 Edition], Information technology—Telecommunications and information exchange between systems—Local and Metropolitan area networks—Part 3:

[2]CISPR documents are available from the International Electrotechnical Commission, 3, rue de Varembé, Case Postale 131, CH-1211, Genève 20, Switzerland/Suisse. CISPR documents are also available in the United States from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA.

[3]IEC publications are available from IEC Sales Department, Case Postale 131, 3, rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse. IEC publications are also available in the United States from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA.

[4]IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA.

[5]ISO and ISO/IEC publications are available from the ISO Central Secretariat, Case Postale 56, 1 rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse. ISO publications are also available in the United States from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA.

[6]This publication and other ISO/IEC [ANSI/IEEE] publications are available from the ISO Central Secretariat as well as from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA.

Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications.

ISO/IEC 8802-5:1998 [ANSI/IEEE Std 802.5, 1998 Edition], Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Part 5: Token ring access method and physical layer specifications.

ISO/IEC 8824:1990, Information technology—Open Systems Interconnection—Specification of Abstract Syntax Notation One (ASN.1).

ISO/IEC 8877:1992, Information technology—Telecommunications and information exchange between systems—Interface connector and contact assignments for ISDN Basic Access Interface located at reference points S and T.

ISO/IEC 9646-1:1994, Information technology—Open Systems Interconnection—Conformance testing methodology and framework—Part 1: General concepts.

ISO/IEC 9646-2:1994, Information technology—Open Systems Interconnection—Conformance testing methodology and framework—Part 2: Abstract Test Suite specification.

ISO/IEC 10040:1992, Information technology—Open Systems Interconnection—Systems management overview.

ISO/IEC 10165-2:1992, Information technology—Open System Interconnection—Structure of management information: Definition of management information.

ISO/IEC 10165-4:1992, Information technology—Open System Interconnection—Structure of management information—Part 4: Guidelines for the definition of managed objects.

ISO/IEC 11801:1995, Information technology—Generic cabling for customer premises.

ISO/IEC 15802-1:1995 Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Common specifications—Part 1: Medium Access Control (MAC) service definition.

ISO/IEC 15802-2:1995 [ANSI/IEEE Std 802.1B, 1995 Edition], Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Common specifications—Part 2: LAN/MAN management.

ISO/IEC DIS 15802-3 [ANSI/IEEE Std 802.1D, 1998 Edition[7]], Information technology—Telecommunications and information exchange between systems—Local area networks—Media access control (MAC) bridges.

ISO/IEC TR 11802-2:1995, Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Technical reports and guidelines—Part 2: Standard Group MAC Addresses.

[7] IEEE Std 802.1D-1998 was approved by the IEEE Standards Board on 30 June 1998 but was not yet published at the time this standard went to press. The draft standard is, however, available from the IEEE. The anticipated publication date is October 1998, when it will be published as ISO/IEC 15802-3:1998 [ANSI/IEEE Std 802.1D, 1998 Edition]. For status information, contact the IEEE Standards Department at 1 (732) 562-3800.

Note—Footnotes 8 and 9 are intentionally left blank in order to maintain consistency with footnote numbering in IEEE 802.12c:1998, the recently published full-duplex supplement to this standard.

3. Definitions

For purposes of this International Standard, the following definitions apply:

3.1 4-UTP: Four-pair 100 Ω balanced cable meeting or exceeding the Category 3 specifications in ISO/IEC 11801:1995.

3.2 5B6B encoding: A method whereby data quintets are mapped (encoded) as code sextets.

3.3 balanced cable: A cable consisting of one or more metallic symmetrical cable elements (twisted pairs or quads).

3.4 baud: A unit of signaling speed equal to the number of discrete conditions or signal events per second, or the reciprocal of the time of the shortest signal element in a character.

3.5 broadcast address: A special address consisting of all 1's indicating all end nodes on the network.

3.6 BT: The time required for one data bit to cross the Medium Independent Interface (MII)—Bit Time = 1/TxClk.

3.7 bundled cable: A cable consisting of multiple twisted pairs. Bundled cable in this standard indicates a cable with 25 unshielded copper twisted pairs, Category 3 or better, that may contain up to six 4-UTP links.

3.8 cascade: A multilevel repeater topology in which higher-level repeaters are connected through their local ports to the cascade port of lower-level repeaters.

3.9 cascade port: The repeater port that enables a cascade connection to a higher-level repeater.

3.10 clear: The action that removes the outgoing signal from a link (i.e., clears the link) and prepares the Physical Medium Dependent (PMD) to receive a packet.

3.11 common-mode voltage: The instantaneous algebraic mean of two signals applied to a balanced circuit, where both signals are referred to a common reference.

3.12 cross-connect: A group of connection points, often wall- or rack-mounted in a wiring closet, used to mechanically terminate and interconnect twisted-pair building wiring.

3.13 demand priority: A round-robin arbitration method to provide LAN access based on message priority level.

3.14 destination address (DA): A field in the packet format identifying the end node(s) to which the packet is being sent.

3.15 differential mode voltage: The instantaneous algebraic difference of two signals applied to a balanced circuit, where both signals are referred to a common reference.

3.16 downlink: The transmission medium between a repeater and a connected end node or lower level repeater, as viewed from the local repeater. *Contrast with:* **uplink**.

3.17 dual simplex: A link segment configuration containing two simultaneous signal paths, one in each direction. (STP and fibre optic links are configured as dual simplex.)

3.18 elasticity buffer: A first-in-first-out (FIFO) buffer in the network repeater that can provide temporary storage for a message packet during retransmission delays. The buffer acts as a shift register or delay line, and does not need to hold an entire, full-length packet. *See also:* **store-and-forward buffer**.

3.19 enable high only: A link control signal from an upper repeater to a lower repeater pre-empting a lower repeater's normal-priority round-robin control cycle.

3.20 end node: A physical device that may be attached to a LAN link segment for the purpose of transmitting and receiving information on that link medium. For example, an end node may be a user station, a bridge, or a LAN analyzer. It is identified by a unique 48-bit address.

3.21 end of stream delimiter (esd): Patterns that identify the end of an MII data stream.

3.22 extension bits: One to four bits that are appended to the last octet of a MAC frame to complete the last data quintet.

3.23 fibre-optic cable: A cable containing one or more optical fibres.

3.24 fibre-optic link: A link segment configured from fibre optic cables and two attached Medium Dependent Interface (MDI) connectors.

3.25 fibre pair: Optical fibres interconnected to provide two continuous light paths terminated at both ends in an optical connector.

3.26 filler: Three- or six-bit reserved code patterns that are appended to the end of individual MII channel data streams to equalize the stream lengths in all four channels.

3.27 frame: The logical organization of control and data fields (e.g., addresses, data, error check sequences) defined for a MAC sublayer. In this standard, the term *frame* refers to a MAC frame unless otherwise indicated. *See also:* **MAC frame**.

3.28 frame check sequence (FCS): A Cyclic Redundancy Check (CRC) used by the transmit and receive algorithms to detect errors in the bit sequence of a MAC frame.

3.29 full duplex: A link segment capable of transferring signals in both directions simultaneously.

3.30 functional address (FA): A bit-significant address used in the ISO/IEC 8802-5 MAC format to identify well-known functional groups.

3.31 grant: A link control signal or link condition indicating that the receiving entity has been given permission to send a packet.

3.32 half duplex: A link segment capable of transferring signals in either direction along the link, but not in both directions simultaneously. Requires line turnaround to change signal direction. (4-UTP links are full duplex in control mode, but only half duplex in data mode.)

3.33 idle: (Idle_Up, Idle_Down) A link control signal indicating that the sending entity currently has no traffic pending for the entity connected to the other end of the link.

3.34 incoming: A link control signal indicating that a packet may soon be sent to the receiving entity.

3.35 individual address: The unique address identifying an individual end node.

3.36 invalid frame: A frame that is marked with an Invalid Packet Marker (IPM) or that has been identified by the MAC, Repeater Medium Access Control (RMAC), or lower sublayers as containing errors.

3.37 invalid packet marker (IPM): A pattern used by a repeater that is substituted for the end of stream delimiter (esd) in the MII channel transmission frames to identify a packet that was received with transmission errors.

3.38 layer management entity (LME): The logical portion of the repeater responsible for collection of operational and error statistics and for management of the network.

3.39 level n repeater: A repeater that is (n–1) link segments below the root repeater in a cascade.

3.40 link is clear: A condition in which there is no incoming energy on the link. *See also*: **clear**.

3.41 link segment: The physical interconnection between two repeaters or between a repeater and an end node. A link segment includes the link medium (twisted pairs or optical fibres) and its two attached Medium Dependent Interface (MDI) connectors.

3.42 local port: The repeater port that allows connection to a lower-level entity (e.g., end node, repeater, bridge, LAN analyzer).

3.43 logical link control (LLC) sublayer: That part of the Data Link Layer that supports medium independent data link functions, and uses the services of the MAC to provide services to the Network Layer.

3.44 MAC frame: The logical organization of control and data fields (e.g., addresses, data, error check sequences) defined for the MAC sublayer. The MAC frame may be constructed in either ISO/IEC 8802-3 or ISO/IEC 8802-5 format. *See also*: **frame**.

3.45 medium: The physical material from which the link is constructed.

3.46 medium access control (MAC) sublayer: The portion of the Data Link Layer that controls access to the medium. The MAC sublayer is required in end nodes.

3.47 medium dependent interface (MDI): The physically exposed interface between the link segment medium and the PMD of the end node or repeater, for which all mechanical, electrical or optical, and transmitted signal requirements are specified.

3.48 medium independent interface (MII): The logical interface between the Physical Medium Independent (PMI) and PMD in an end node or repeater. Optionally, the MII may be implemented as a physically exposed interface with specified signaling timing and electrical characteristics.

3.49 multicast address: A special address indicating a specific group of end nodes.

3.50 null address: An all zero address that does not identify any network end node. The null address may be used as the destination address in training packets being sent from an end node or lower-level repeater to an upper-level repeater. The null address may be used in the Source Address (SA) field of training frames to verify link operation between an end node and the connected repeater. The null address may also be used in void frames. Packets with the null address are forwarded by the receiving repeater to all promiscuous ports.

3.51 octet: A group of eight adjacent bits.

3.52 packet: The total information transmitted over the link medium, including the preamble, the MAC frame, and the start of stream and end of stream delimiters. *See also:* **frame**.

3.53 pad: Any combination of octets used to extend the end of the data field of the ISO/IEC 8802-3 MAC frame so that it will meet minimum length requirements.

3.54 physical medium dependent (PMD) sublayer: The medium dependent portion of the Physical Layer.

3.55 physical medium independent (PMI) sublayer: The medium independent portion of the Physical Layer.

3.56 primitive: A definition of a service provided by a sublayer to the sublayer immediately above. Primitives may be initiated by either the upper or lower sublayer.

3.57 privacy mode: A mode in which an end node receives only those packets specifically addressed to it.

3.58 promiscuous mode: A mode in which a repeater port or an end node receives all message traffic transmitted on the network.

3.59 Protocol Implementation Conformance Statement (PICS): A statement of which capabilities and options have been implemented for a given interconnection protocol.

3.60 quintet: A contiguous string of five bits.

3.61 redundant link: A second link from an end node or from the cascade port of a repeater that provides an alternative path to maintain network connectivity in case of a repeater or link failure.

3.62 repeater: A device used to extend the length, topology, and interconnectivity of the physical medium beyond that imposed by a single segment. Demand-priority repeaters perform the functions of restoring signal amplitude, waveform, and timing. They also arbitrate access to the network from connected end nodes and optionally collect statistics regarding network operations.

3.63 repeater medium access control (RMAC) sublayer: The sublayer in the repeater that arbitrates packet sequencing and controls packet routing.

3.64 request: (Request_Normal, Request_High) A link control signal indicating that a lower entity has traffic pending for the network.

3.65 return: A secondary link control signal indicating that the pre-empted normal-priority round-robin cycle in a lower repeater is not complete.

3.66 root repeater: The level 1 (topmost) repeater in a cascade.

3.67 service: The capabilities and action provided by one layer for another.

3.68 sextet: A contiguous string of six bits.

3.69 simplex: A link segment configuration capable of transferring signals in one direction only.

3.70 source address (SA): A field in the message packet format identifying the sending end node.

3.71 start of stream delimiter (ssd): Reserved code patterns that identify the beginning of the MII channel transmission frame. The ssd indicates the transmission priority of the packet.

3.72 store-and-forward buffer: A first-in-first-out (FIFO) buffer in the network repeater that can provide temporary storage for an entire message packet prior to retransmission. The buffer acts as a shift register and must hold an entire, full-length packet. *See also:* **elasticity buffer**.

3.73 STP: A 150 Ω shielded balanced cable meeting the specifications in ISO/IEC 11801:1995.

3.74 symbol rate: The number of symbols transmitted per second and expressed in baud (e.g., 1 Mbd = 1 000 000 symbols per second).

3.75 training: (Training_Up, Training_Down) A link control signal indicating that the sending entity is either requesting or giving permission to train (initialize) the link.

3.76 twisted pair: Two continuous, insulated copper conductors twisted around each other in a helical manner. Twisted-pair cable may be unshielded (UTP) or shielded (STP).

3.77 twisted-pair cable: A group of two or four twisted pairs within a single sheath.

3.78 twisted-pair cable binder group: A group of twisted pairs within a cable that are bound together.

3.79 twisted-pair link: A link segment consisting of a twisted-pair cable and two attached MDI connectors.

3.80 unicast address: An individual address identifying an individual end node.

3.81 uplink: The transmission medium between an end node or repeater and a connected higher-level repeater, as viewed from the local entity. *Contrast with:* **downlink**.

3.82 UTP: A 100 Ω unshielded balanced cable meeting the specifications in ISO/IEC 11801:1995.

3.83 weight 2/4 code alternation: A rule requiring successive unbalanced-code symbols in a data stream to be alternately chosen between weight-2 and weight-4 code groups.

3.84 weight-2 code: An unbalanced code sextet containing exactly two 1’s.

3.85 weight-4 code: An unbalanced code sextet containing exactly four 1’s.

4. Abbreviations and acronyms

The following abbreviations and acronyms are used in this standard:

AC	ISO/IEC 8802-5 format Access Control field
AI	Active Optical Input
AO	Active Optical Output
BER	Bit error rate
BT	Bit Time
CRC	Cyclic Redundancy Check
CRS	Configuration Report Server
CS	Control Signal
DA	Destination Address
esd	end of steam delimiter
FA	Functional Address
FAI	Functional Address Indicator
FC	ISO/IEC 8802-5 format Frame Control field
FCS	Frame Check Sequence
FIFO	First In First Out
FWHM	Full Width at Half Maximum
GDMO	Guidelines for the Development of Managed Objects
HCS	Half Control Signal
I/G	Individual/Group
IPM	Invalid Packet Marker
L	Length of ISO/IEC 8802-3 format data
LAN	Local Area Network
LED	Light Emitting Diode
LLC	Logical Link Control

LME	Layer Management Entity
LSB	Least Significant Bit
LTH	ISO/IEC 8802-5 format RI field Length
MAC	Medium Access Control
MDI	Medium Dependent Interface
mesd	multiplexed end of stream delimiter
MII	Medium Independent Interface
mpreamble	multiplexed preamble
MSB	Most Significant Bit
msdu	MAC service data unit
mssd	multiplexed start of stream delimiter
NEXT	Near-End Crosstalk
NRZ	Non-Return-to-Zero
OSI	Open System Interconnection
PDU	Protocol Data Unit
PICS	Protocol Implementation Conformance Statement
PMD	Physical Medium Dependent
PMI	Physical Medium Independent
PVC	Polyvinyl Chloride
RCS	Receiver Control State
REM	Ring Error Monitor
RI	ISO/IEC 8802-5 format Routing Information field
RII	Routing Information Indicator
RMAC	Repeater Medium Access Control
RPS	Ring Parameter Server
RxClk	Receive Clock
RxEn	Receive Enable

SA	Source Address
SAP	Service Access Point
ssd	start of stream delimiter
TCS	Transmit Control State
TPIO	Twisted Pair Input/Output circuit
TxClk	Transmit Clock
TxEn	Transmit Enable
Vdd	power supply voltage
Vih	Voltage, input high
Vil	Voltage, input low
Voh	Voltage, output high
Vol	Voltage, output low

5. Notation

This standard uses service primitives, finite state machines, state tables, and pseudo code, supplemented by prose descriptions and illustrative diagrams, to define the requirements of the protocol.

5.1 Service definition method and notation

Services in this specification are described in an abstract manner using the method and naming conventions of the ISO/IEC Open Systems Interconnection Service Conventions defined in ISO TR 8509:1987. The services of an (n-1) layer are the capabilities that it offers to an (n)-user in the next higher (n) layer. In order to provide its service, the (n-1) layer builds its functions on the services that it requires from the next lower layer. Figure 5-1 illustrates the layer service model.

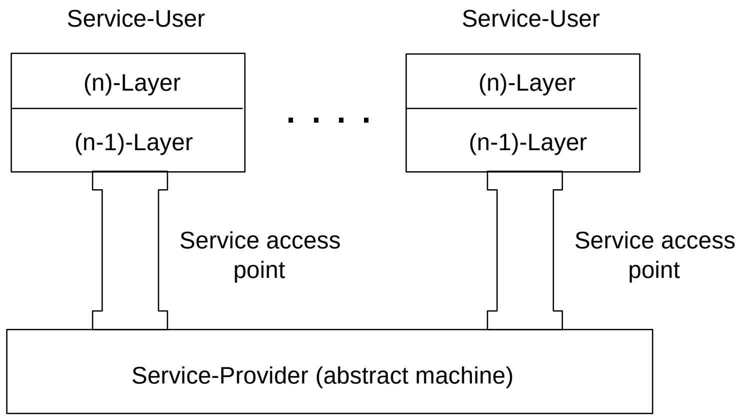


Figure 5-1—Layer service model

Services are specified by describing the information flow between the service user and the next lower layer. This information flow is modeled by discrete, instantaneous events that characterize the provision of service. Each event consists of passing a service primitive from one layer to another. User layers communicate with peer user layers by passing primitives through one or more Service Access Points (SAPs) to service providers associated with the identified peer user(s).

Services are described through the use of service primitives and parameters that characterize each service. A service may have one or more related primitives that constitute the activity related to the particular service. Each service primitive may have zero or more parameters that convey the information required to provide the service.

Service primitives convey the information required to provide a particular service. These service primitives are an abstraction in that they specify only the service provided rather than the means by which it is provided. Further, this definition is independent of any particular interface implementation. There is no direct conformance of equipment to these service definitions. Clauses 10 through 19 and the associated normative annexes specify the requirements for the functional behavior of equipment.

Two generic types of primitives are used in this specification:

a) *Request primitive.* This primitive is passed by the (n)-layer to the (n-1)-layer to request that a service be initiated. Request primitives generated by a layer may result in one or more primitives being generated by the next lower layer.

b) *Indication primitive.* This primitive is passed from the (n-1)-layer to the (n)-layer to indicate an event that is significant to the (n)-layer. This event may be logically related to a remote service request, or may be caused by an event internal to the (n-1)-layer.

Some of the possible relationships among primitives are illustrated by the time-sequence diagrams in Figure 5-2. Primitives that occur earlier in time and that are connected by dotted lines are the logical antecedents of subsequent primitives. Figures 5-2a) through 5-2d) show possible relationships within an individual device. Figure 5-2e) shows a typical relationship between the connected entities at each end of a link. If there is no specific relationship between events, where it is impossible to predict which will occur first, but where both must occur within a finite period of time, then a tilde (~) is used. This possibility is shown in Figure 5-2f). An example might be the indication of a failure in the link between a repeater and the end node—it would be impossible to know which entity would be the first to detect the failure.

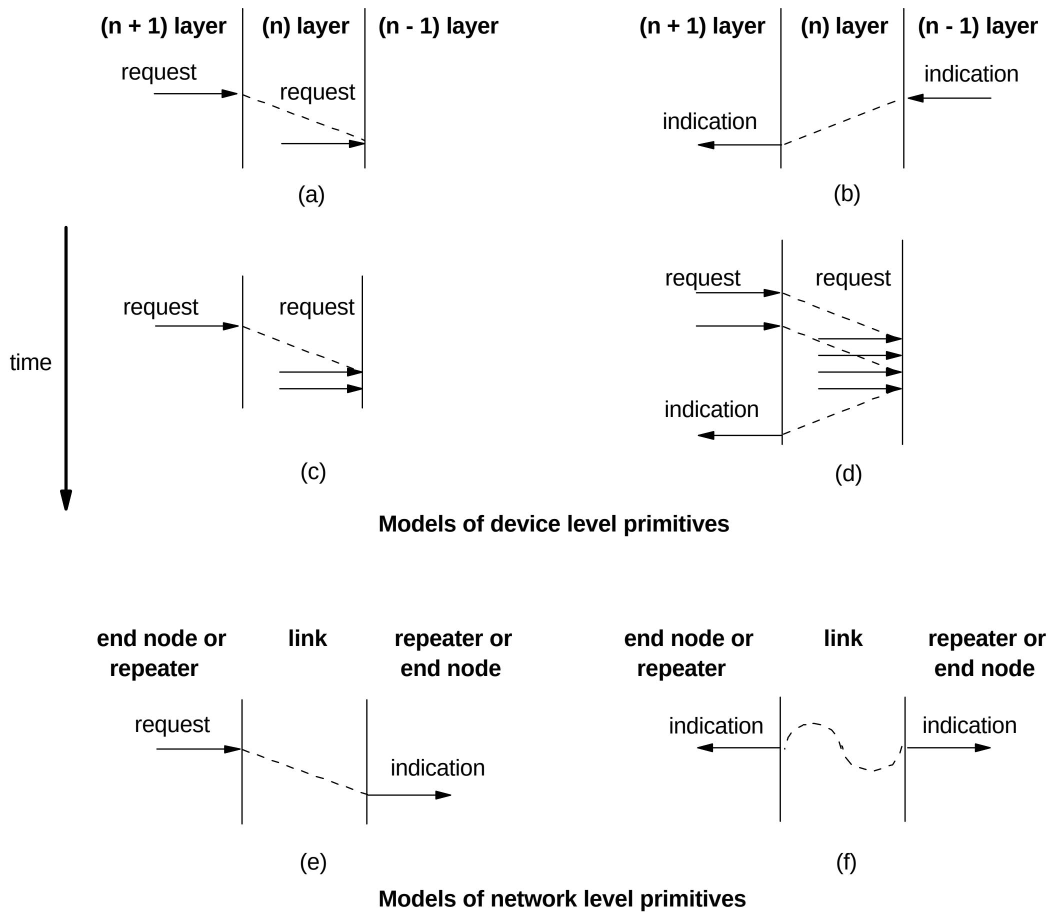


Figure 5-2—Primitive time-sequence examples

5.2 State diagram notation

The operation of a protocol can be described by subdividing the protocol into a number of interrelated functions. The operation of a function can be described by state machines. Each machine represents the domain of a function and consists of a group of connected, mutually exclusive states. Only one state of a function may be active at any one time; however, several functions may be active at the same time.

Each state that a function can assume is represented by a rectangle as shown in Figure 5-3. State names are shown above the dotted lines in the rectangle; actions or operations to be accomplished in the state are shown below the double line.

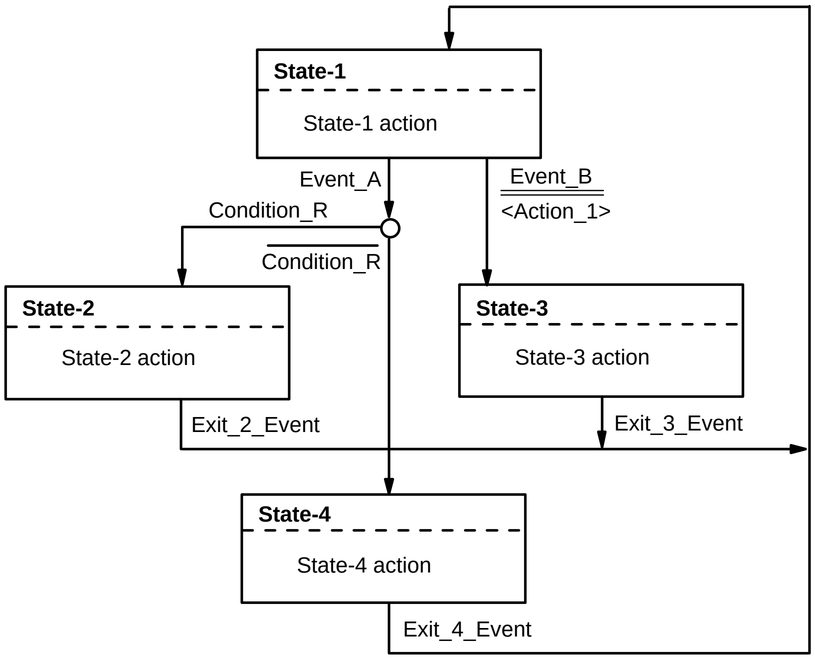


Figure 5-3—Example state machine diagram

State machines are event driven and can be in any defined state. They remain in the current state until an event occurs that would cause them to change states. Transitions are presumed to be instantaneous.

The possible transitions between the states are represented by arrows from one state to the next. If multiple states have transition paths to a common destination (States 2, 3, and 4 in the example all have transition paths to State-1), the path may be shown as a transition from the state to a common path. Transition paths that cross each other without joining arrows are presumed to be independent.

Events or conditions that cause the particular transition are shown next to the transition path at the exit point of the state. Transition arrows that fork (indicated by a small circle) require the logical AND of the labeled transition conditions. For example, transition from State-1 to State-4 would require that Event_A occur AND that Condition_R be false.

Actions or operations that are to be performed during a transition are enclosed in < > and are shown below a double line under the condition required for the transition. For example, the transition from State-1 to State-3 would require that Event_B occur and that Action_1 be taken.

5.3 Pseudo code state description

The structure of a state machine is commonly defined in the state machine diagram. The operation of the state machine may be defined either by prose annotation or by coded constructs, such as a pseudo code. For some state machines (e.g., MAC, RMAC), this standard uses prose annotation to provide an overview of each state of a state machine, and pseudo code to define the operations accomplished by that state. If there are any perceived differences in the interpretation of the prose annotation and the pseudo code description, the pseudo code shall take precedence.

5.3.1 Pseudo code constructs and key words

The pseudo code used in this standard is a variant of PASCAL. It uses PASCAL constructs and key words; however, the operation differs somewhat from PASCAL. Execution of any pseudo code construct or code segment is considered to be instantaneous, regardless of the number of statements or the number of iterations involved.

=	Is equal to -- For example, the statement: If (A = B) would be interpreted as testing for the condition: is A equal to B? If A equals B, the statement would be set to true, otherwise it would be false.
<>	Is not equal to -- For example, the statement: If (A <> B) would be interpreted as testing for the condition: is A not equal to B? If A does not equal B, the statement would be set to true, otherwise it
would	be false.
:=	The replacement operator --For example, the statement: A := B means replace the current value of A with the value of B.
{ }	Defines a non-executable comment. When used within the body of the code, it encloses an annotation or comment. Pseudo code com-
ments	in this standard are shown in bold face type (for example, see 5.3.2).
[ ]	Refers to elements with multiple instantiations, for example, ADDRESS_MATCH [X] refers to the Xth element of the array ADDRESS_MATCH.
And	The logical AND operator.
Begin End	Marks the BEGINning and END of a code section
Case Of	A concise form of multiple If statements (see example in 5.3.2)
For Do	A method of defining bounds for a code loop. For example, the con-
struct:	For variable vvv = value_1 to value_2, Do statement bbb would execute statement bbb for all integer values of variable vvv beginning with value_1 and ending with value_2.
Forever Do	A method of defining a code segment that executes continuously, for example, every clock cycle.
If Then Else	A method of testing for a condition and then executing alternate statements. For example, the construct:

```
If (statement aaa)
    Then
        statement bbb
    Else
        statement ccc
```

	would TEST the value of the statement aaa, and THEN, if aaa were true, bbb would be executed; ELSE, if aaa were false, ccc would be executed.
Or	The logical OR operator.
Repeat Until	A method for defining a code segment that is executed repeatedly until some defined condition is true. For example, the code segment:

```
Repeat
    statement aaa
    statement bbb
        ...
    statement nnn
Until (condition sss)
```

would cause statements aaa through nnn to be repeatedly executed until the condition sss became true.

Wait	Suspend execution for the period of time specified. For example, the construct:

```
Wait ( duration )
```

would suspend operation for a period of time equal to the value of duration.

Wait_Until	Suspend execution until the named event occurs. For example, the construct:

```
Wait_Until (some_event)
```

would suspend execution until the event: some_event occurred.

With var Do	Execute the statements following this construct using the contents of the field specified by var. For example, the code segment:

```
With ALLOWED_CONFIGURATION_FIELD Do

    Statement aaa

    Statement bbb

End
```

would cause the contents of ALLOWED_CONFIGURATION_FIELD to be subjected to the operations specified in statements aaa and bbb.

5.3.2 Pseudo code example

Consider an example of how the pseudo code is interpreted. Suppose that the state machine in Figure 5-3 is in STATE_1, that both condition-1 and condition-2 can never be true at the same time, and that the pseudo code for STATE_1 is:

```
STATE_1: INITIAL_STATE
   Begin {execute the initial operations that are to be performed on entering the state}
      aaa;
      bbb;
   End;  {end of initial operations}

   Case TRANSITION_CONTROL of: {poll for possible state transition conditions}
```

```
        CONDITION_1:   {condition-1 is true; test for condition-3 to determine the next state}
            Begin
               IF (CONDITION_3 = TRUE) then  {condition-3 is true; transition to state-2}
                  NEXT_STATE := STATE_2
               else {condition-3 is false; transition to state-4}
                  NEXT_STATE := STATE_4;
            End {condition-1}

        CONDITION_2 : {condition-2 is true; execute action-1 and transition to state-3}
            Begin
               ACTION_1
               NEXT_STATE := STATE_3
            End {condition-2}

    End;  {case of transition control}
```

The operations defined under the current state that can be performed are performed immediately. Then operations that test for the existence of transition conditions (defined here in a case construct) are polled continuously.

a) If condition-1 occurs before condition-2, transition is made to either state-2 (if condition-3 is true) or state-4 (if condition-3 is false).
b) If condition-2 occurs first, action-1 is performed and then transition is made to state-3.

These actions and transitions are conditional on the existence of the indicated event being detected, and are not carried out if some other event occurs.

NEXT_STATE indicates the destination state of the transition. It may be a global variable and may be available to several state machines. Changing the value of NEXT_STATE may cause state transitions and/or actions to be taken in other state machines as well.

5.3.3 Pseudo code functions and procedures

The pseudo code state definitions may call special, commonly used routines known as functions or procedures. Functions and procedures are distinguished by the function or procedure name, followed by either:

a) Parameters in brackets, such as in: CHANGE_CASCADE_PORT_TONE (REQ_H), or
b) Brackets without parameters, such as in: POWER_ON ( ).

5.3.3.1 Functions

Functions are frequently used code segments that may carry out actions and that return values to the calling routine. The function may or may not utilize parameters to transfer data between itself and the calling routine. For example, a pseudo code function call might be:

```
        VARIABLE := EXAMPLE_FUNCTION (parameter_1, parameter_2)
```

where parameter_1 and parameter_2 are values that the calling routine wishes EXAMPLE_FUNCTION to act upon or supply (they may be either constants or variables).

The function EXAMPLE_FUNCTION(v_in, v_out) is defined as a separate routine that accepts one value from, and returns two values to, the calling routine.

```
EXAMPLE_FUNCTION(v_in, v_out)
    Begin;
        If v_in <> some_value
            then
                EXAMPLE_FUNCTION := return_value_1
             else
                EXAMPLE_FUNCTION := return_value_2;
        mmm
        nnn
        v_out := computed_value
    End;
```

In this case, the calling routine supplies the value of parameter_1, which is accepted as the value of v_in in the function. Inclusion of EXAMPLE_FUNCTION as the destination variable of a replacement statement in the function definition allows it to return a direct replacement value for VARIABLE in the calling statement. The value of v_out (computed_value in this example) is returned to parameter_2 in the calling routine.

5.3.3.2 Procedures

Procedures are frequently used routines that carry out actions, but do not return a value to the calling routine. The procedure may or may not utilize parameters to transfer data from the calling routine to the procedure.

5.4 Numerical representation

Numerical values may be represented in binary, decimal, or hexadecimal notation.

Binary values are represented by the binary digits 0, 1, or x, and are indicated by the letter B followed by the binary value enclosed in apostrophes wherever it is not obvious from the context (e.g., a logical value). The lowercase x represents a digit that may be either 0 or 1. An example binary number is B'001xxx'. An example logical value is RCS [2:0] = 010.

Hexadecimal values are represented by the hexadecimal digits 0–9, A–F, or x, and are indicated by the uppercase letter X followed by the hexadecimal value enclosed in apostrophes. The lowercase x represents a digit that may be any hexadecimal value. An example hexadecimal value is X'01 6F xx'.

The actual numerical value of an octet within a MAC frame field depends on the particular format being used (see 10.1.1).

5.5 Time-space diagrams

Operational sequences are often depicted in time-space diagrams as shown in Figure 5-4. The space between the horizontal lines represents the link between the repeater and the connected end nodes (or repeaters). Control signals between the repeater and the end node are shown as single arrows indicating the direction of the signal. (For clarity, signals sent by the end node are shown solid; signals sent by the repeater are shown dashed.) The specific control signal is identified by the indicated signal name; the duration of the signal is identified by the horizontal arrow. The different spacing between the repeater and the end nodes indicates the possibility of different link distances. The slope of the transmitted control signals and data packets depicts the propagation delay in the link.

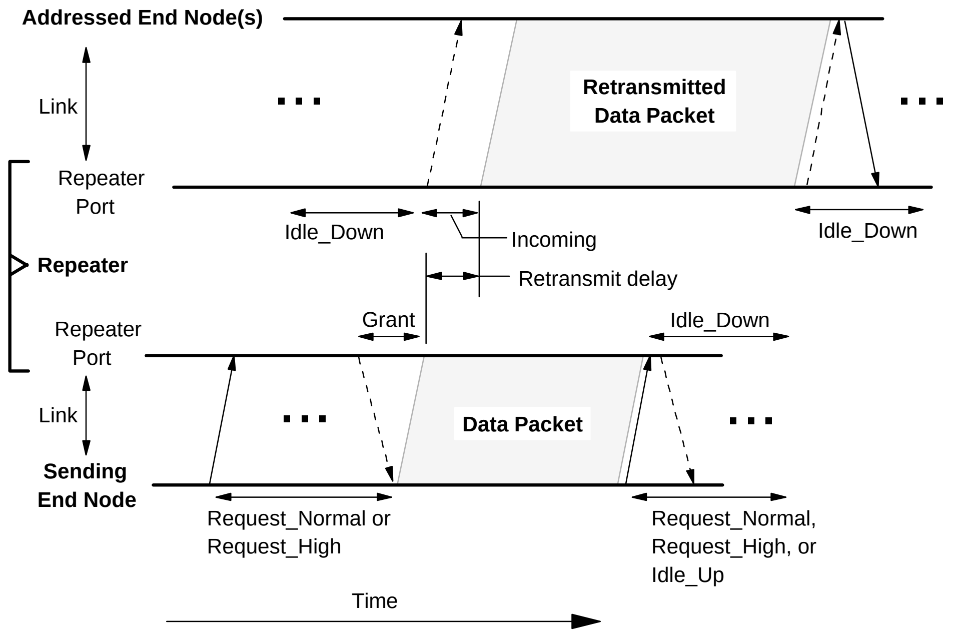


Figure 5-4—Example time-space diagram

6. Introduction to the protocol

The demand-priority medium access protocol provides a means by which stations (end nodes) can communicate with each other over a centrally controlled LAN that offers a choice of several different link media including: 100 Ω balanced cable (4-UTP and 2-TP), 150 Ω shielded balanced cable (STP), and optical fibres.

6.1 Network topology

The demand-priority access method has been defined to allow considerable flexibility in the network topology.

6.1.1 Network components

The demand-priority LAN architecture utilizes three principal structural components: end nodes, repeaters, and network links.

a) End nodes are typically personal or larger computers, user workstations, peripherals, or file servers. They might also be special devices (such as LAN analyzers), or they might be bridges to other LANs.

b) Repeaters are the network controllers and are configured with two or more local ports to allow connection to end nodes or other repeaters. They can also be configured with an optional cascade port, which is reserved for connection to repeaters only.

c) Link segments provide the interconnection medium between a repeater and its connected end nodes or other repeaters.

6.1.2 Network structure

The network can be structured with single or multiple repeater levels.

6.1.2.1 Single repeater networks

The simplest network structure contains one repeater and two or more end nodes as shown in Figure 6-1.

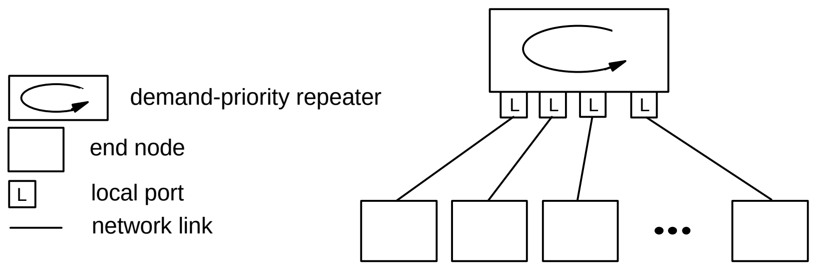


Figure 6-1—A single-repeater network topology

6.1.2.2 Cascaded networks

Larger topologies can contain several levels of repeaters interconnected in a cascade as shown in Figure 6-2. Each repeater will typically be connected to one or more end nodes, and can be connected to one or more repeaters. Lower-level repeaters and end nodes are connected to local ports. Higher-level repeaters must be connected to a cascade port. Interconnection between two repeaters using only local ports is not allowed.

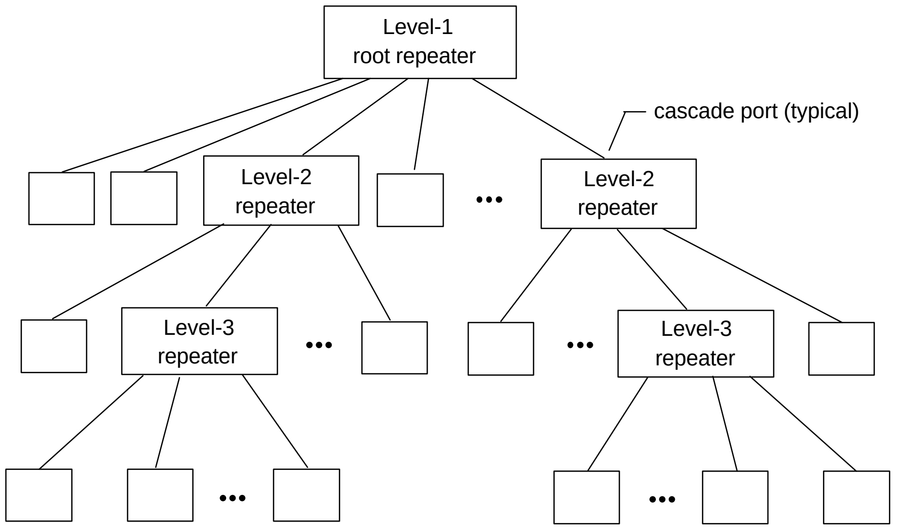


Figure 6-2—A multilevel cascaded topology

The topmost repeater in the cascade is designated as the Level-1 repeater. Repeaters in each succeeding lower level in the cascade are designated by the number of links between them and the root repeater by the equation:

$$\text{repeater level} = (\text{number of link segments away from the root repeater}) + 1 \qquad (1)$$

All repeaters on the same level are designated with the same level number.

6.1.2.3 Connection to other LANs

Connection of a demand-priority network to other LANs is shown in Figure 6-3. Each bridge is connected to a local port on the demand-priority repeater, and is treated as though it were an end node on the network.

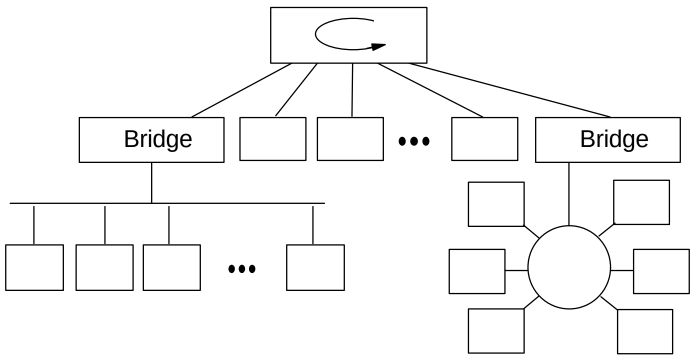


Figure 6-3—A demand-priority network interconnected with other LANs

6.1.3 Link configuration

Packets can be sent to an individual end node, to groups of end nodes (group addressing), or to all end nodes on the network (broadcast addressing). Repeater ports connected to end nodes can be configured to support either of two addressing modes:

a) Privacy mode, in which packets are sent to addressed end nodes only, or
b) Promiscuous mode, in which the port transmits all traffic.

Local ports connected to lower-level repeaters, bridges, or network analyzers should always be configured for promiscuous-transmission mode. Packets with broadcast and multicast addresses are sent to all ports.

End nodes can request the repeater port to be configured to receive packets in either privacy or promiscuous mode; however, promiscuous-mode requests can be denied by the network manager. Repeater support for privacy mode is optional.

6.1.4 Link media

The links between repeater local ports and their connected end nodes can be 4-pair 100 Ω balanced cable (4-UTP and 2-TP), 2-pair 150 Ω balanced cable (STP), or two optical fibres.

For 4-UTP repeater-to-end node links, the recommended configuration is 4-pair cable. Bundled cable (25-pair binder-groups) is allowed, but requires repeaters with privacy mode and store-and-forward capability because excessive crosstalk could occur with simultaneous transmission of a packet to multiple ports on the same bundled cable as the sending end node.

Bundled cable must not be used in repeater-to-repeater, repeater-to-bridge, or repeater-to-end node links where the end node is allowed to receive all network traffic. See Annex E for additional information and topology rules and recommendations for ISO/IEC 8802-12 networks.

6.1.5 Redundant links

Both end nodes and repeater cascade ports may optionally be equipped with a second (redundant) link to maintain network connectivity in case of individual link or repeater failure in the network connection path. Specific requirements for implementing redundant links are given in Clause 11 for end nodes and in Clause 12 for repeaters. Rules for connecting redundant links and recommendations for interconnecting redundant-link repeaters and end nodes in a network containing one or more repeaters that do not have redundant-link capability are defined in Annex E.

6.2 Basic operational concepts

Demand-priority access is a priority-based, round-robin arbitration method in which the central network controller (the repeater) regularly polls[10] its connected ports to determine which have transmission requests pending, and whether the transmission request is normal priority (e.g., for data files) or high priority (e.g., for real-time voice, video, or data). The following overview concentrates on the operation of the protocol between the repeater and its connected end nodes. The full description of round-robin arbitration and port selection in single and multiple repeater networks is provided in Clause 12.

[10]The particular method of determining whether or not a repeater port has a transmission request pending is an implementation decision. The requirement is for each port to be interrogated at least once each packet time.

6.2.1 Round-robin polling

Round-robin polling provides all end nodes with access to the network during each round-robin cycle. The repeater maintains two next-port pointers, one for normal-priority and one for high-priority requests, that keep track of the next ports to be serviced. All ports are polled at least once per packet transmission to determine which have requests pending.

A high-priority request allows an end node to obtain permission to send sooner than its normal-priority cycle position. High-priority requests are serviced before normal-priority requests, but do not cause a normal-priority transmission in progress to be interrupted.

In the repeater, timers monitor the normal-priority requests at each port to ensure that abnormally large high-priority traffic levels do not preclude normal-priority traffic. Normal-priority requests that have been pending for approximately 250 ms are automatically elevated to high priority and are serviced in port order as indicated by the high-priority next-port pointer. Priority promotion is not intended to occur during normal network operation. When an abnormal situation occurs (e.g., an end node continuously making high-priority requests), priority promotion allows normal-priority requests to be serviced until network management locates and isolates the source of the problem.

6.2.2 Data transmission

Packet transmission follows a request/grant sequence in which the request is initiated by the sending end node and the issuance of the grant is controlled by the repeater. When packet transmission is occurring:

a) The repeater polls all local ports to determine which end nodes or lower-level repeaters are requesting to send a packet and whether the request is normal priority or high priority. If an end node has a packet ready to send, it transmits either a Request_Normal or Request_High control signal. Otherwise, the end node transmits the Idle_Up control signal.

b) Once the repeater has finished transmitting the packet, it selects the next end node with a request pending (the next port closest to the next-port-pointer value) by sending Grant to the selected end node. Idle_Down is sent to all other end nodes. Packet transmission begins when the end node detects the grant.

c) The repeater then alerts all end nodes (other than the sender) on the network of a possible incoming packet (by transmitting an Incoming control signal) and decodes the destination address as it is being received.

d) When an end node receives the Incoming control signal, it prepares to receive a packet.

e) After the repeater has decoded the destination address, the packet is sent to the addressed end node(s) and to any promiscuous ports. Communication to nonaddressed, nonpromiscuous ports is inhibited during packet transmission (the repeater sends Idle_Down to these ports during packet transmission).

Figure 6-4 shows a conceptual time-space diagram of a transmission (from an end node connected to port 8 to an end node connected to port 3) in a single-level repeater topology. Elasticity buffering of the packet is provided in the repeater to compensate for delays required to decode the destination address and to begin packet retransmission to the designated end node.

6.2.3 Round-robin transmission example

Figure 6-5 shows an example that would result with round-robin polling in an 8-station, single-repeater network, assuming that the network was idle at t = 0. The normal-priority request from port 2 is selected by the repeater’s RMAC and packet transmission begins. During this time, high-priority requests are received from ports 5 and 1, but they are not acted upon until the current packet transmission is complete. The repeater then selects and processes the high-priority request with the closest port in port order to the high-priority next-port pointer (port 1 in this example). The high-priority request from port 5 is serviced next. Since there are no additional high-priority requests pending after port 5, the repeater returns to normal-priority requests and processes these, in port order (ports 3, 6, and 7), as indicated by the normal-request next-port pointer. Another high-priority request is received from port 1 and is serviced before the remaining normal-priority requests can be satisfied. These are then processed in the order indicated by the normal-priority next-port pointer.

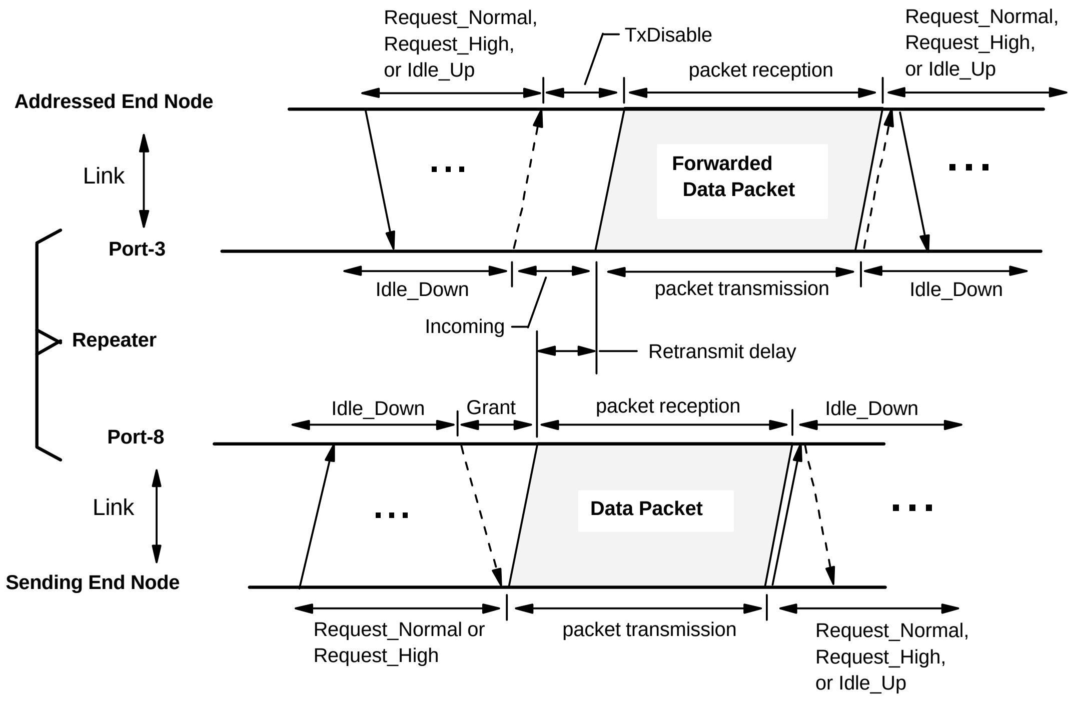
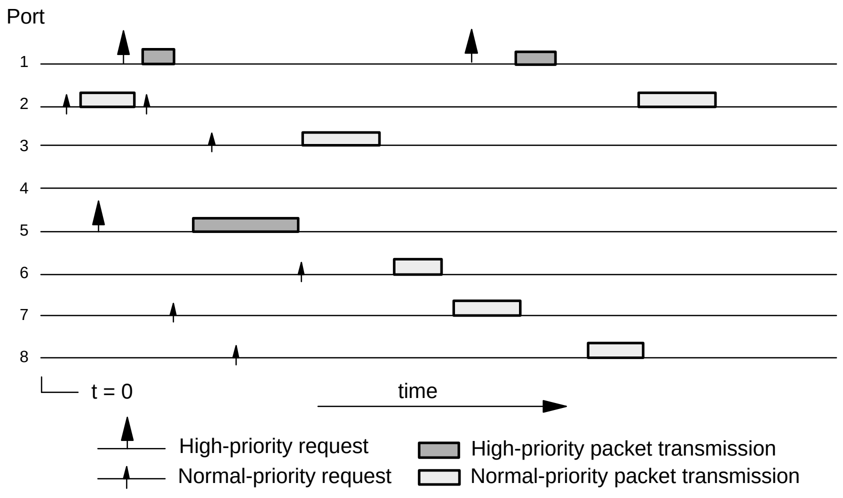


Figure 6-4—Packet transmission time-space diagram


Port
1
2
3
4
5
6
7
8
t = 0
time
High-priority request
Normal-priority request
High-priority packet transmission
Normal-priority packet transmission

Request sequence = n-2, H-5, H-1, n-2, n-7, n-3, n-8, n-6, H-1
Packet sequence = n-2, H-1, H-5, n-3, n-6, n-7, H-1, n-8, n-2

Figure 6-5—An example packet sequence in a single-repeater network

6.3 Operations in cascaded networks

Transmission in cascaded networks follows a similar procedure to operations in a single-repeater network. The interconnected repeaters act essentially the same as a single large repeater. All traffic is sent to each repeater, and each repeater polls its active ports for requests at least once per packet transmission.

The Level-1 (root) repeater has primary control of the packet sequencing. Lower-level repeaters with requests pending issue requests to the next higher-level repeater in the same manner as an end node.

Request selection begins at the root repeater. If the next port to be serviced is connected to an end node, the request is handled essentially in the same way as described in 6.2. All repeaters and other end nodes on the network are alerted of a possible incoming packet. The destination address is decoded as the packet is being received. If the address indicates an end node on a repeater's port address list, the packet is forwarded to that end node and to any promiscuous ports. The packet is always forwarded to all other repeaters on the network.

If the next port (with a pending request) in the root-repeater selection sequence is connected to a lower-level repeater, the request selection responsibility is granted to the lower-level repeater, and that repeater keeps the selection authority for one complete round-robin cycle. Requests from succeeding ports in the root-repeater result in a single packet transmission if the link is connected to an end node, or in a complete round-robin cycle if the link is connected to a repeater. The effect is a network that acts as though it has a single large repeater with port ordering as shown in the example of Figure 6-6.

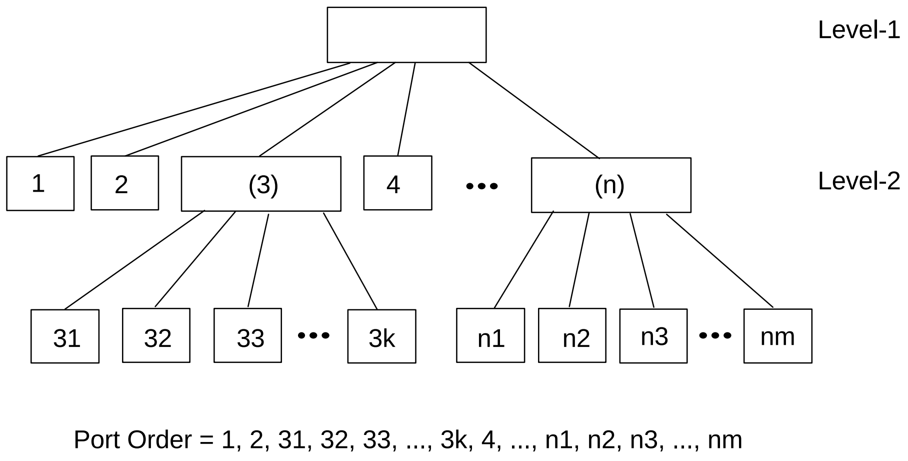

Level-1
Level-2
1
2
(3)
4
(n)
31
32
33
3k
n1
n2
n3
nm
Port Order = 1, 2, 31, 32, 33, ..., 3k, 4, ..., n1, n2, n3, ..., nm

Figure 6-6—Effective port ordering in a network with two repeater levels

High-priority requests from any port in a network with more than one repeater level are processed before normal-priority requests. If a high-priority request is received before completion of a normal-priority round-robin in a lower-level repeater, the root repeater pre-empts the remainder of the round-robin cycle by sending the Enable_High_Only control signal to the lower repeater. After the high-priority request has been satisfied, the lower-level repeater is allowed to complete the normal-priority round-robin cycle. Figure 6-7 shows an example request sequence and the resulting packet order for the network of Figure 6-6, assuming that a normal-priority packet is being transmitted at time = t, and that no high-priority requests are pending.

6.4 Link training

A link training sequence is required to verify the cable quality for data transmission, to allow the receiver to adapt to the link, and to establish the end node's address. It is performed each time a link is logically established (e.g., power-up, cable connection). It is also performed when some error conditions are detected.

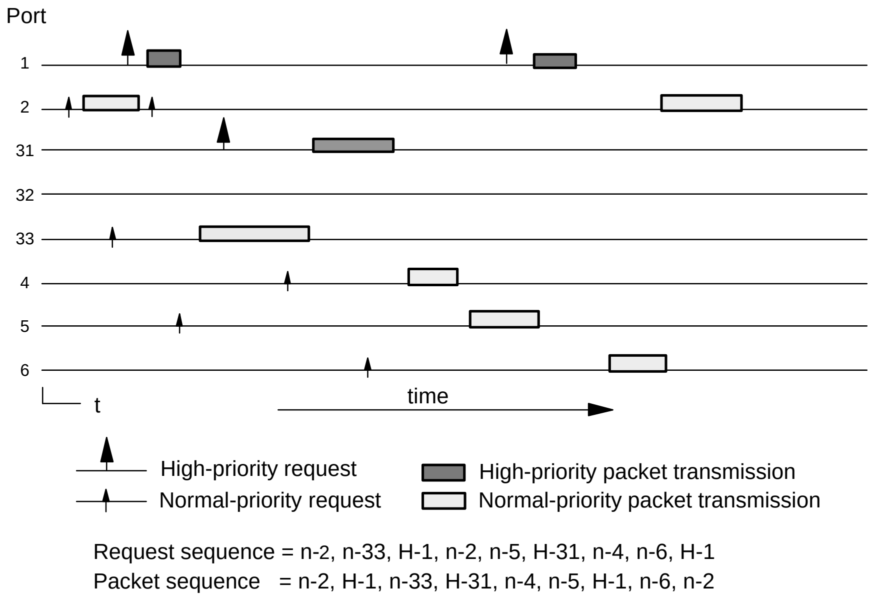

Port
1
2
31
32
33
4
5
6
t
time
High-priority request
Normal-priority request
High-priority packet transmission
Normal-priority packet transmission
Request sequence = n-2, n-33, H-1, n-2, n-5, H-31, n-4, n-6, H-1
Packet sequence = n-2, H-1, n-33, H-31, n-4, n-5, H-1, n-6, n-2

Figure 6-7—Example packet transmission sequence in a cascaded network

Link training is always initiated by the lower entity (the training initiator), which can be either an end node or a lower repeater. The upper repeater can force training to be initiated by disabling the link. Training is accomplished by sending a series of special training packets in each direction between the end node and the repeater. Training frames are sent to all repeaters to alert them that training is in progress somewhere on the network. During training, the end node sends Training_Up in place of Idle_Up and the repeater sends Training_Down in place of Idle_Down.

6.5 Transmission errors

Transmission problems such as excessive noise can cause reception errors in a packet. Error detection mechanisms in the data code and the CRC appended to each packet enable repeaters and end nodes to detect such errors. Invalid packets received by the repeater are marked by substituting an Invalid Packet Marker (IPM) for the end of stream delimiter (esd) in the outgoing packet's end of frame sequence. Invalid packets received by the destination end node can be discarded. Appropriate error counters are incremented at all receiving entities.

6.6 Error recovery procedures

Recovery from transmission errors, including notification of the sending end node, is the responsibility of upper layers in the Open Systems Reference Model protocol stack. The repeater detects and recovers from protocol errors by isolating faulty links to protect the network. The LME provides tools to support identification of deteriorating or defective links.

6.7 Architectural relationship to the ISO/IEC and IEEE standards

Local network design can be viewed from either of two important perspectives: architectural, which emphasizes the logical division of the system and how these divisions fit together; or implementational, which describes the actual components, their packaging, and their interconnection. This standard adopts the architectural perspective, separating the system into four logical sublayers that are defined to fit within the ISO/IEC 8802 and IEEE 802 families of standards.

6.7.1 End node architecture

The four sublayers of the reference model for the demand-priority end node correspond to the two lower layers of the ISO/IEC 7498-1:1994 model for Open Systems Interconnection (OSI). A depiction of this relationship is given in Figure 6-8.

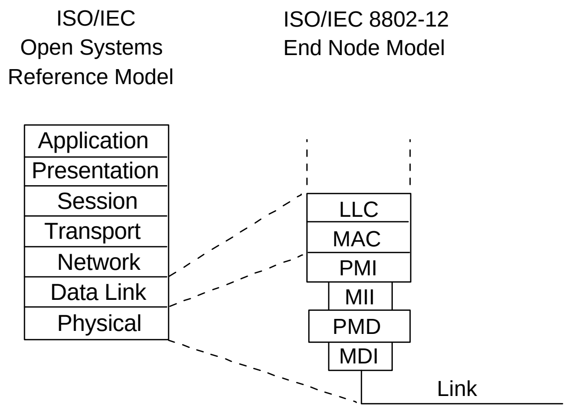


Figure 6-8—Demand-priority end node model

The upper sublayer in the end node is typically an ISO/IEC 8802-2 Class I LLC supporting Type 1 unacknowledged, connectionless-mode transmission, or an ISO/IEC 8802-2 Class II LLC supporting Type 2, connection-mode transmission, or similar protocol instance.

The MAC sublayer is compatible with two MAC formats and interfaces:

a) The MAC-frame format and MAC/LLC interface of ISO/IEC 8802-3:1996
b) The MAC-frame format and MAC/LLC interface of ISO/IEC 8802-5:1995

The MAC will also support an Ethernet frame within the ISO/IEC 8802-3 compatible MAC-frame format, where a type field is used in place of the length field.

The functions intended for the ISO Physical Layer are encompassed by the PMI and PMD sublayers. The MII is defined as a logical interface between the PMI and the PMD. An optional physical definition of the MII allows the interchange of PMDs supporting different physical links.

The MDI is usually a fully specified physically exposed connection between the PMD and the link medium. However, in some cases, a MAC is embedded in another device without an exposed MDI. For instance, a MAC supporting network management protocols in a repeater might not have an exposed MDI. In that case, the Physical Layer functions are logically, but not physically, present.

6.7.2 Repeater architecture

The architecture of the network repeater, shown in Figure 6-9, is similar to the architecture of the end node.

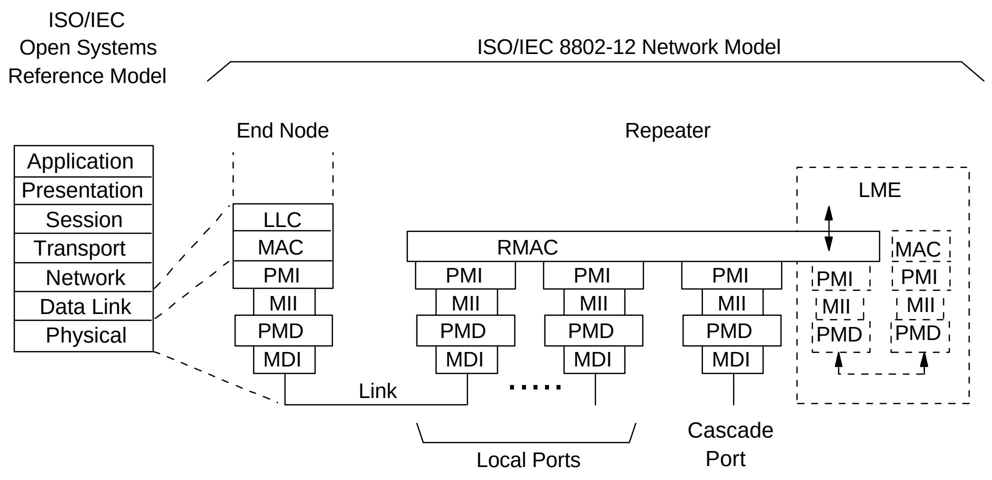


Figure 6-9—Demand-priority network model

The RMAC sublayer provides arbitration and control of packet sequencing through the various ports of the repeater.

The PMI and PMD functions are the same as in an end node. In the architectural model of this standard, there is a separate PMI/PMD pair for each port. In an implementation, it is possible to meet the requirements of this standard while sharing all or part of the PMI circuitry between multiple ports. The optional cascade port is only used for connecting (cascading) the local repeater to an upper-level repeater.

The LME is an optional unit that can be functionally integrated with the repeater for network management purposes. It can be viewed as a special end node that has both direct and indirect access to the RMAC:

a) The direct path (represented by a solid arrow) allows immediate access to status and events in the RMAC to update the LME’s operation and error counters.
b) An optional indirect path (represented by a dotted arrow) supports packet traffic between the local LME and either remote LMEs or end nodes through the LME’s own MAC sublayer.

The optional path allows for in-band management whereby management information is passed over the LAN. Management information can also be accessed through out-of-band management, which passes the information over other communications links or locally through a console on the repeater.

The PMI/PMD to PMD/PMI path between the RMAC and the LME is shown for illustrative purposes only. The MAC, PMIs, and PMDs in the LME might not necessarily be present as separate physical entities. The depiction only indicates that a link exists for frame interchange between the RMAC and the local LME. Communication over this path provides the logical functions of the MAC and physical sublayers. No particular implementation of this function is implied. The means of accomplishing a communication path between the RMAC and the LME’s MAC is an implementation issue and is beyond the scope of this standard.

6.7.3 MAC functions

The MAC accepts transmit requests and provides receive indications for packet traffic between the end node and the repeater. The MAC controls frame format composition and decomposition, checks for transmission errors in received frames, and initiates control requests to the PMI. The MAC service specifications, formats, and protocol are defined in Clauses 7, 10, and 11, respectively.

6.7.4 RMAC functions

The RMAC is the primary control sublayer of the network repeater. The RMAC accepts transmit requests from the end node, arbitrates packet transfer sequences, interprets destination addresses, and routes incoming packets to the proper outbound ports. It also supplies a subset of the functions (such as packet error checking) provided by the MAC in an end node. The RMAC protocol is defined in Clause 12.

6.7.5 PMI functions

The PMI supplies Transmit Control State (TCS) indications to the PMD, provides data scrambling and encoding for outgoing frames, and adds preambles plus the start and end of stream sequences to the data streams prior to transmission. In the receive mode, the PMI accepts Receiver Control State (RCS) indications from the PMD, strips the preamble and the start and end of stream sequences, and provides data descrambling and decoding for incoming frames. PMI functions are defined in Clause 14.

6.7.6 MII

The logical medium independent interface (MII) is between the PMI and the PMD. PMD service primitives are passed across this interface. The optional physical implementation of the MII provides interchangeability of PMDs and allows easy reconfiguration of end nodes and repeater ports to support multiple link media. MII specifications are given in Clause 15.

6.7.7 PMD functions

The PMD provides control signal generation and recognition, data stream signal conditioning, clock recovery, and channel multiplexing appropriate to the link medium.

6.7.8 MDI

The MDI is the physically exposed connection between the PMD and the link segment. The PMD and MDI must both support the same link medium. Clauses 16 through 18 define PMD and MDI functions for 4-UTP, STP, and LED fibre-optic link media, respectively.

6.7.9 LME functions

The LME, if provided, acts as a specialized end node. It collects operational and error statistics and provides the operational management support for the network. LME functions are described in Clause 13.

6.8 Items for future study

Areas being considered for future development include:

a) A burst-mode capability, allowing end nodes to send multiple short packets per grant.
b) Higher speed operation.
c) Full-duplex capability (on switched or two-node networks).

7. Medium Access Control (MAC) service definitions

7.1 Scope

This clause specifies the services provided by the MAC sublayer to the local LLC sublayer in an end node, to the internal relay entity in a bridge, and to the repeater's local LME entity (see Figure 7-1). The MAC services to the LLC are defined in ISO/IEC 15802-1:1995. MAC services to a bridge are defined in ISO/IEC DIS 15802-3. The MAC interfaces to an LLC are described in 7.3; the MAC interfaces to a bridge are described in 7.4. These descriptions indicate MAC operation; no particular implementation of the MAC sublayer is implied.

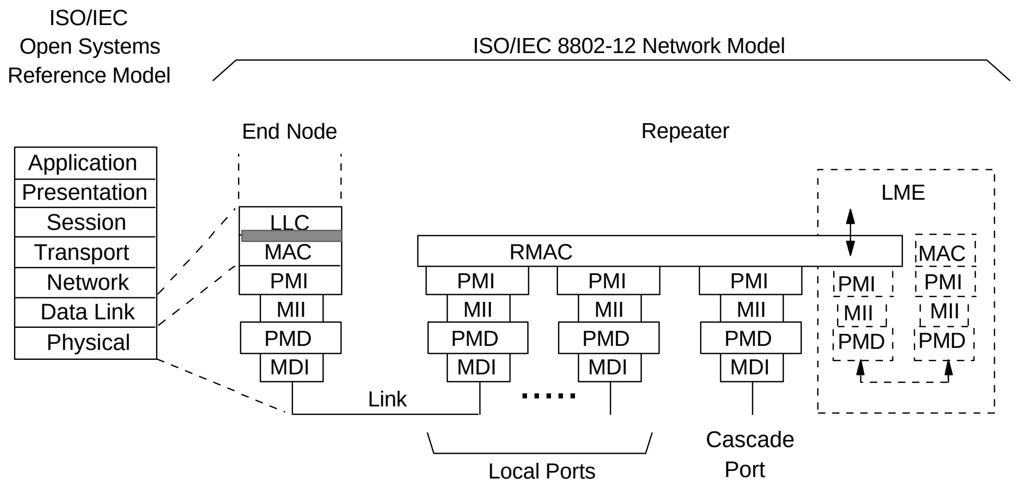


Figure 7-1—Relationship to the LAN model

7.2 Overview of MAC services

The services provided by the MAC sublayer allow:

a) The local LLC sublayer in an end node to exchange data with peer LLC sublayer entities (see 7.3).
b) The relay entity in a bridge to exchange data with local MAC entities in the bridge (see 7.4).

7.3 MAC services to the LLC

Two service primitives are defined for the MAC to LLC interface.

a) MA_UNITDATA.request
b) MA_UNITDATA.indication

The ISO/IEC 8802-12 protocol provides compatibility modes for both ISO/IEC 8802-3 and ISO/IEC 8802-5 frame structures (see 10.2 and 10.3).

7.3.1 ISO/IEC 8802-3 compatibility mode

Figure 7-2 shows the mapping of the MA_UNITDATA.request primitive and parameters to the ISO/IEC 8802-3 frame fields, and the mapping of ISO/IEC 8802-3 frame fields to the MA_UNITDATA.indication primitive and parameters. The ISO/IEC 8802-12 frame structure for the ISO/IEC 8802-3 compatibility mode is defined in 10.2.

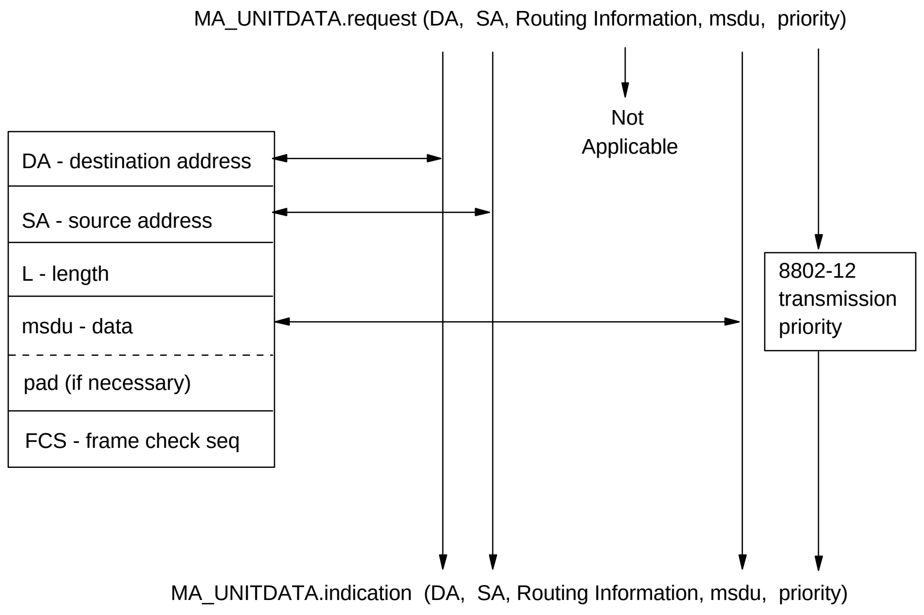

MA_UNITDATA.request (DA, SA, Routing Information, msdu, priority)
Not Applicable
DA - destination address
SA - source address
L - length
msdu - data
pad (if necessary)
FCS - frame check seq
8802-12 transmission priority
MA_UNITDATA.indication (DA, SA, Routing Information, msdu, priority)

Figure 7-2—Mapping MAC service primitives to/from ISO/IEC 8802-3/ISO/IEC 8802-12 MAC frames

Upon receipt of an MA_UNITDATA.request primitive from the local LLC entity, the MAC will compose a frame using the parameters supplied to create the DA, SA, L, data, and pad (if necessary) fields, and to calculate the FCS field of the ISO/IEC 8802-3 frame. The ISO/IEC 8802-12 protocol provides two levels of service: “normal” and “high.” Service priority is not included as a field in the ISO/IEC 8802-3 frame, but is encoded in the ISO/IEC 8802-12 packet start frame delimiter. If the priority is not indicated in the MA_UNITDATA.request, the default value should be “normal.”

When a valid ISO/IEC 8802-3 frame is received, an MA_UNITDATA.indication is made to the LLC entity. The value for the User_priority parameter will be the value of the priority level extracted from the received ISO/IEC 8802-12 packet start frame delimiter.

7.3.2 ISO/IEC 8802-5 compatibility mode

Figure 7-3 shows the mapping of the MA_UNITDATA.request primitive and parameters to the ISO/IEC 8802-5 frame fields, and the mapping of ISO/IEC 8802-5 frame fields to the MA_UNITDATA.indication primitive and parameters. The ISO/IEC 8802-12 frame structure for the ISO/IEC 8802-5 compatibility mode is defined in 10.3.

Upon receipt of an MA_UNITDATA.request primitive from the local LLC entity, the MAC will compose an ISO/IEC 8802-5 frame using the parameters supplied to create the AC, FC, DA, SA, RI (if present), and INFO (user data) fields, and to calculate the FCS field. The RII bit in the SA field will be set to “1” if and only if routing information is present. The priority is encoded in the YYY bits of the FC field. The ISO/IEC 8802-12 protocol provides two levels of service: “normal” and “high.” The recommended default service is “normal” for ISO/IEC 8802-5 priority levels “000” through “100,” and “high” for ISO/IEC 8802-5 priority levels “101” through “111.” The service priority level is encoded in the ISO/IEC 8802-12 packet start of stream delimiter.

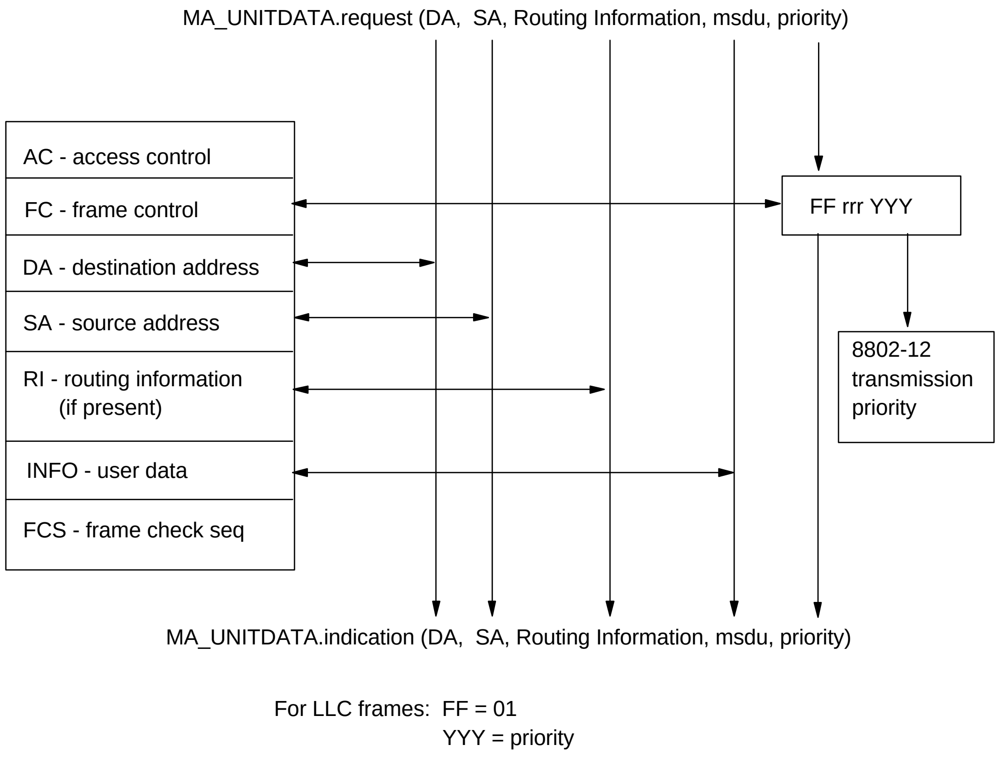

MA_UNITDATA.request (DA, SA, Routing Information, msdu, priority)
AC - access control
FC - frame control
DA - destination address
SA - source address
RI - routing information (if present)
INFO - user data
FCS - frame check seq
FF rrr YYY
8802-12 transmission priority
MA_UNITDATA.indication (DA, SA, Routing Information, msdu, priority)
For LLC frames: FF = 01
YYY = priority

Figure 7-3—Mapping MAC service primitives to/from ISO/IEC 8802-5/ISO/IEC 8802-12 frames

When a valid ISO/IEC 8802-5 frame is received, an MA_UNITDATA.indication is made to the LLC entity. If the frame type is LLC, then the value of the priority parameter will be the value of the YYY bits in the FC field.

7.4 MAC services to a bridge

Two service primitives are defined for the bridge interfaces.

a) M_UNITDATA.request
b) M_UNITDATA.indication

The formats for M_UNITDATA.indications and M_UNITDATA.requests are the same as formats for MA_UNITDATA.indications and MA_UNITDATA.requests, except for the addition of an optional parameter for the FCS and two parameters that indicate the frame_type and the requested mac_action. The FCS parameter may be used to preserve the FCS value when bridging between LANs using like formats. The frame_type and mac_action parameters are not explicitly encoded in the MAC frames, but are included as constants in bridging service requests and indications as specified in ISO/IEC DIS 15802-3, subclause 2.4.

8. Physical Medium Independent (PMI) service definitions

8.1 Scope

This clause describes the services provided by the PMI sublayer for the MAC sublayer in an end node or repeater LME, and for the RMAC sublayer in a repeater (see Figure 8-1). The services are described in an abstract manner assuming the logical model of one PMI/PMD pair per port and a clock function (TxClk[11]) for synchronization of primitive generation. The model is used for definition only and no particular implementation for the PMI is implied. Also, there is not necessarily a one-to-one correspondence between the primitives defined here and the PMI coding and control functions described in Clause 14.

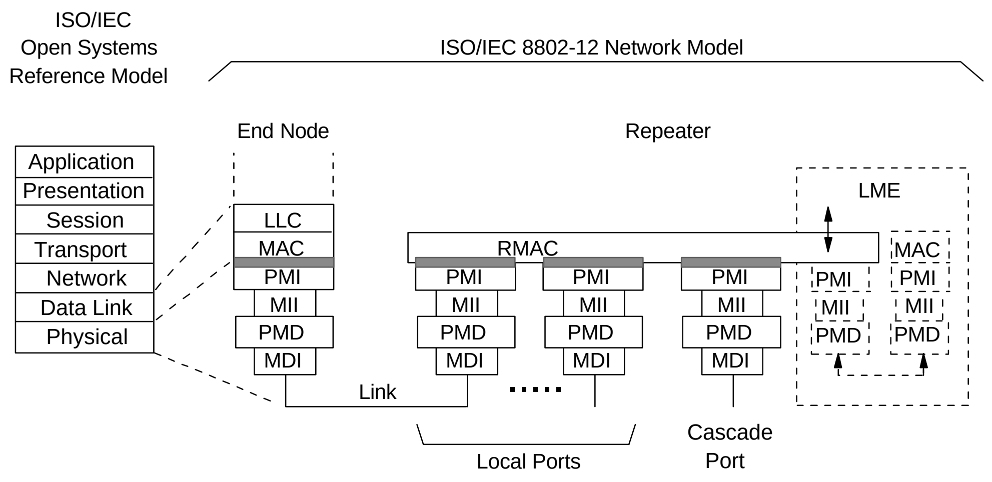


Figure 8-1—Relationship to the LAN model

8.2 Services provided by the PMI sublayer

The services provided by the PMI sublayer allow:

a) The local MAC sublayer entity to exchange MAC-frame data with peer MAC sublayer entities.
b) The local RMAC sublayer entity in the repeater to arbitrate packet sequencing, routing, and re-transmission of MAC-frame data on the network.

8.3 PMI service primitives

The PMI service primitives define the interface between the MAC and the PMI sublayers in an end node and between the RMAC and PMI sublayers in a repeater. The primitives associated with the MAC/PMI and the RMAC/PMI interfaces fall into two basic categories:

a) Primitives that have local significance and that provide the information required by either the MAC or RMAC sublayer to perform its control functions:

   1) PMI_CONTROL.request

[11] TxClk is generated by the PMI.

   2) PMI_CONTROL.indication
   3) PMI_TRANSMIT.indication
   4) PMI_CONFIGURATION.indication
   5) PMI_RECEIVE_ENABLE.request

b) Primitives that support the transfer of data from a MAC or RMAC sublayer entity to other peer MAC or RMAC sublayer entities contained within the same LAN:

   1) PMI_UNITDATA.request
   2) PMI_UNITDATA.indication

Generation and reception of PMI_CONFIGURATION.indication is optional.

8.4 PMI_CONTROL.request

This primitive defines the control signals to be transmitted over the link.

PMI_CONTROL.request (transmit_control_state)

transmit_control_state (TCS). The transmit_control_state parameter specifies the TCS that will be transferred through the PMI to the PMD. The transmit_control_state parameter values are defined in 11.2.1.2 for MAC sublayers, and in 12.3.1 for RMAC sublayers. Conversion of the parameter values to TCS encoding is defined in 14.3.1.

8.4.1 When generated

This primitive is generated by the MAC or RMAC sublayer every cycle of the transmit clock (TxClk).

8.4.2 Effect of receipt

The receipt of this primitive will cause the PMI to generate a PMD_CONTROL.request to the PMD (see 14.3.1).

8.5 PMI_CONTROL.indication

This primitive indicates the current receiver state in the PMD.

```
PMI_CONTROL.indication (grant_state,
                        receiver_control_state)
```

grant_state: The grant_state parameter allows the PMD to indicate both a current and previous receiver control state. The grant_state parameter can be either one or zero. The interpretation of a grant_state value of one is defined in 11.2.1.4 for the MAC, and in 12.3.3 for the RMAC. A value of zero indicates that the receiver_control_state parameter contains the currently decoded receive control state on the line. A value of one indicates that the receiver control state on the line is grant and that the receiver_control_state decoded before grant arrived is indicated in the receiver control state parameter. Conversion of the MII signals to grant_state coding is defined in 14.3.3.

receiver_control_state (RCS): The receiver_control_state parameter indicates the RCS. The receiver_control_state parameter values are defined in 11.2.1.3 for MAC sublayers, and in 12.3.2 for RMAC sublayers. Conversion of RCS encoding to the PMI_CONTROL.indication parameter values is defined in 14.3.2.

8.5.1 When generated

This primitive is generated every cycle of TxClk in response to a PMD_CONTROL.indication primitive.

8.5.2 Effect of receipt

The receipt of this primitive will cause the MAC or the RMAC to take the action defined for the particular receiver_control_state.

8.6 PMI_TRANSMIT.indication

This primitive indicates the completion of a packet transmission.

PMI_TRANSMIT.indication (transmit_complete)

transmit_complete: The transmit_complete parameter is set by the PMI during packet transmission to let the MAC or the RMAC know when the last data-stream bits have been transferred across the MII. This parameter is set to "more" as long as there are still data-stream bits to be transferred. A value of "end" indicates that packet transmission is complete.

8.6.1 When generated

This primitive is generated every cycle of TxClk during packet transmission. The transmit_complete parameter will be set to "end" on the next cycle of TxClk after the PMI has deasserted TxEn.[12]

8.6.2 Effect of receipt

Receipt of this primitive with a transmit_complete parameter value of "more" will have no effect on either the MAC or the RMAC. A transmit_complete parameter value of "end" will cause the MAC and the RMAC to take the actions defined for the particular parameter value.

8.7 PMI_CONFIGURATION.indication (optional)

This primitive defines the connected PMD/Link configuration.

PMI_CONFIGURATION.indication (configuration)

configuration: When implemented, the configuration parameter indicates the connected PMD/Link configuration. It can be any of the values defined in 15.2.11.

8.7.1 When generated

The primitive is generated every cycle of TxClk.

8.7.2 Effect of receipt

Receipt of this primitive will allow the MAC or the RMAC to inform network management what type of link is present.

[12]TxEn is an internal control signal that crosses the MII.

8.8 PMI_RECEIVE_ENABLE.request

This primitive establishes the data receive operation mode for the PMI and the PMD.

PMI_RECEIVE_ENABLE.request (receive_enable_state)

receive_enable_state: The receive_enable_state parameter can be either “assert” or “deassert.” The value of “deassert” provides a way for the MAC or the RMAC to terminate the data receive mode in the PMI and the PMD when the MAC or the RMAC detects that the data receive operation mode should cease.

8.8.1 When generated

The PMI_RECEIVE_ENABLE.request primitive is generated by the MAC, or, in the case of the cascade port of a repeater, by the RMAC, after receipt of Incoming from a connected higher-level repeater. The PMI_RECEIVE_ENABLE.request is generated to a local port of a repeater when grant is asserted in response to a request.

8.8.2 Effect of receipt

Receipt of this primitive will cause the PMI either to assert (or deassert) RxEn[13] and to switch to the data receive (or control) mode.

8.8.3 Additional comments

The PMI is normally responsible for deasserting RxEn following packet reception. The PMI monitors the RCS code during packet reception and when RCS = 000, or when the PMI detects a complete end of stream sequence, RxEn is deasserted. The MAC or the RMAC overrides the PMI’s responsibility only when the incoming packet alert has been cancelled, or when a packet length error or another fault is detected.

8.9 PMI_UNITDATA.request

This primitive defines the transfer of MAC frame data from the frame buffer in the MAC or the RMAC to the PMI.

```
PMI_UNITDATA.request (priority,
                      data_octet,
                      mac_frame,
                      error_indication)
```

priority: The priority parameter indicates the transmission-priority level. It can be either “high” or “normal.” The priority parameter informs the PMI which priority start delimiter to use.

data_octet: The data_octet parameter is the octet being transferred to the PMI from the MAC or RMAC frame buffer. All possible combinations of eight bits are valid.

mac_frame: The mac_frame parameter indicates when octets have been transferred from the MAC or the RMAC to the PMI. The mac_frame parameter associated with the first octet will be “begin.” It will then be “more” so long as there are still data octets in the frame buffer. The mac_frame parameter associated with the last octet will be set to “end” to indicate that the frame transfer is complete.

[13]RxEn is an internal control signal that crosses the MII.

error_indication: The error_indication parameter defines the error status of the frame being retransmitted by the RMAC. This parameter is null if the primitive is generated by a MAC sublayer. The value of this parameter is determined by the error_status parameter of the last PMI_UNITDATA.indication primitive for the received frame. The error_indication will be:

a) “Valid” if the value of the received error_status parameter is “valid.”
b) “IPM” if the value of the received error_status parameter is either “IPM” or “error.”

8.9.1 When generated

The PMI_UNITDATA.request primitive is generated by the MAC or RMAC sublayer for each data octet to be transmitted. The mac_frame parameter is set to indicate whether the octet is the first octet in the frame, an intermediate octet in the frame, or the last octet in the frame.

8.9.2 Effect of receipt

The PMI will convert, scramble, and encode the data octets into four data-bit streams for delivery to the MII as defined in 14.4.2.

8.9.3 Additional comments

The MAC frame in the demand-priority protocol does not contain start of stream and end of stream sequences. The PMI must account for timing differences between the transfer of the single octet stream from the MAC or RMAC sublayer and its subsequent conversion to scrambled and encoded bit streams at the MII.

8.10 PMI_UNITDATA.indication

This primitive defines the transfer of data from the PMI sublayer to the MAC-frame buffer in the MAC or RMAC sublayers.

```
PMI_UNITDATA.indication (data_octet,
                         priority,
                         start_mac_frame_status,
                         mac_frame,
                         error_status)
```

data_octet: The data_octet parameter indicates the octet being transferred from the PMI to the frame buffer in the MAC or the RMAC. All possible combinations of eight bits are valid.

priority: The priority parameter indicates the priority of the ssd in the received packet. It can be either “high” or “normal.” The priority will be “normal” if start_mac_frame_status is “invalid.”

start_mac_frame_status: The start_mac_frame_status parameter indicates that a possible ssd has been detected at the beginning of the packet. This parameter can be either “valid” or “invalid.”

mac_frame: The mac_frame parameter indicates when octets are being transferred from the PMI to the MAC or the RMAC. The mac_frame parameter associated with the first octet will be “begin.” It will then be “more” so long as there are still data octets to be transferred into the frame buffer. The mac_frame parameter associated with the last octet will be set to “end” to indicate that the frame transfer is complete.

error_status: The error_status parameter indicates the received error status of the total received packet. It can be “valid,” “IPM,” or “error.” For packets that contain both detected errors and an IPM, the status parameter will be set to “error.”

8.10.1 When generated

This primitive is generated for each octet being transferred from the PMI to the MAC or RMAC frame buffer. The PMI_UNITDATA.indication with the mac_frame value equal to “end” must be sent on the next cycle of TxClk after the PMI has deasserted RxEn.

8.10.2 Effect of receipt

The receipt of this primitive will cause the MAC or RMAC sublayer to accept or stop accepting incoming octets from the PMI.

9. Physical Medium Dependent (PMD) service definitions

9.1 Scope

This clause describes the services provided by the PMD sublayer to the PMI sublayer (see Figure 9-1). The services provided are described in an abstract manner; and no particular implementation is implied.

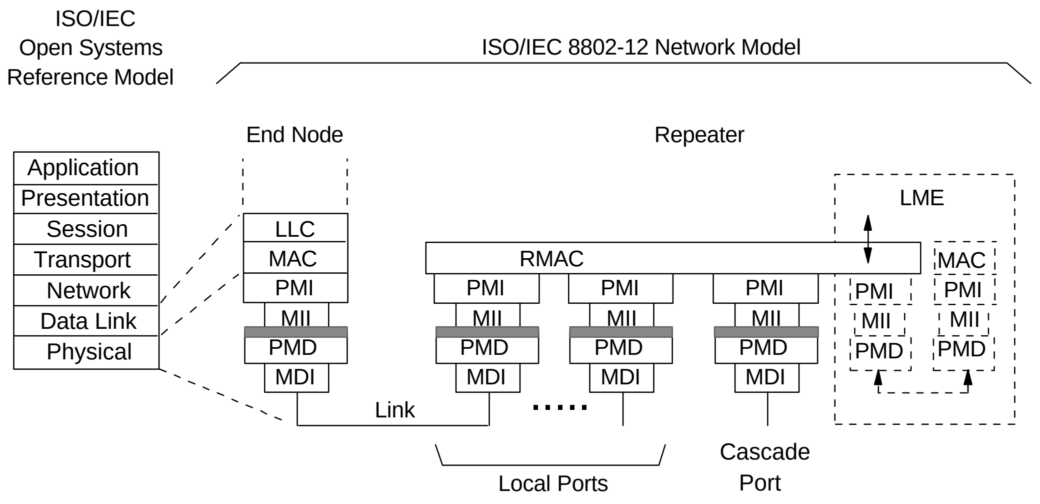


Figure 9-1—Relationship to the LAN model

9.2 Services provided by the PMD sublayer

The services provided by the PMD sublayer allow the local PMI sublayer entity to exchange data with peer PMI sublayer entities.

9.3 PMD service primitives

The primitives associated with the PMD/PMI interface fall into two basic categories:

a) Primitives that have local significance and that provide the information required by the local MAC or RMAC sublayer to perform its local control functions:

  PMD_CONTROL.request
  PMD_CONTROL.indication
  PMD_CONFIGURATION.indication
  PMD_RECEIVE_ENABLE.request

b) Primitives that support the transfer of data from a PMI sublayer entity to other peer PMI sublayer entities contained within the same LAN:

  PMD_DATA.request
  PMD_DATA.indication

The generation and reception of PMD_CONFIGURATION.indication is optional.

9.4 PMD_CONTROL.request

This primitive defines the TCS to the PMD.

PMD_CONTROL.request (transmit_control_state)

transmit_control_state: The transmit_control_state parameter specifies the control signal(s) to be sent. This parameter can be binary 000 through 111. Specific transmit control states and their meanings are described in 14.3.1. The specific control signals for each PMD are defined in Clauses 16 through 18.

9.4.1 When generated

This primitive is generated by the PMI sublayer every cycle of TxClk in response to a PMI_CONTROL.request from the MAC or RMAC sublayers.

9.4.2 Effect of receipt

The receipt of this primitive will cause the PMD to clear the link of outgoing control signals or to transmit the control signal(s) appropriate to the link medium for the transmit_control_state requested.

9.4.3 Additional comments

If an optional, physically exposed MII is implemented, the realization of this primitive is the transfer of the TCS code value through the MII circuits: TCS [2:0].

9.5 PMD_CONTROL.indication

This primitive indicates the grant and receiver control states of the PMD.

PMD_CONTROL.indication (grant_state,
receiver_control_state)

grant_state: The grant_state parameter indicates whether a grant is being decoded at the PMD. It can be either zero (grant not received) or one (grant received).

receiver_control_state: The receiver_control_state parameter indicates the current RCS. This parameter can be binary 000 through 111. Specific receiver control states and their meanings are described in 14.3.2. The specific input signals for each PMD are defined in Clauses 16 through 18.

9.5.1 When generated

This primitive is generated every cycle of TxClk.

9.5.2 Effect of receipt

Receipt of this primitive will cause the PMI to generate a PMI_CONTROL.indication to the MAC or RMAC sublayer.

9.5.3 Additional comments

If an optional, physically exposed MII is implemented, the realization of this primitive is the transfer of the grant_state and receiver_control_state code values through the MII circuits: RxGrant and RCS [2:0].

9.6 PMD_CONFIGURATION.indication (optional)

This primitive defines the PMD/Link configuration.

PMD_CONFIGURATION.indication (configuration)

configuration: When implemented, the configuration parameter indicates the connected PMD. Meanings of each code value are given in 15.3.11.

9.6.1 When generated

The primitive is generated every cycle of TxClk.

9.6.2 Effect of receipt

Receipt of this primitive will cause the PMI to generate a PMI_CONFIGURATION.indication to the MAC or the RMAC.

9.6.3 Additional comments

If an optional, physically exposed MII is implemented, the realization of this primitive is the transfer of the configuration code value through the optional MII circuits: Config [3:0].

9.7 PMD_RECEIVE_ENABLE.request

This primitive establishes the data receive operation mode for the PMD.

PMD_RECEIVE_ENABLE.request (receive_enable_state)

receive_enable_state: The receive_enable_state parameter can be either “assert” or “deassert.” The value of “deassert” provides a way for the PMI to terminate the data receive mode in the PMD.

9.7.1 When generated

The PMD_RECEIVE_ENABLE.request primitive is generated by the PMI either upon receipt of a PMI_RECEIVE_ENABLE.request, or upon detection of an end of packet condition.

9.7.2 Effect of receipt

Receipt of this primitive will cause the PMD to switch to the data receive (or control) mode.

9.7.3 Additional comments

The PMI is normally responsible for deasserting RxEn following packet reception. The PMI monitors the RCS code during packet reception and when RCS = 000, or when the PMI detects a complete end of stream sequence, RxEn is deasserted. The MAC or the RMAC overrides the PMI’s responsibility only when the incoming packet alert has been canceled, or when a packet length error or another fault is detected.

9.8 PMD_DATA.request

This primitive supports the transfer of four data streams from the PMI sublayer to the PMD.

PMD_DATA.request (data_bits [3:0])

data_bits: The data_bits parameter represents a single bit in each of the four data streams being transferred from the PMI to the PMD.

9.8.1 When generated

This primitive is generated by the PMI sublayer every cycle of TxClk while TxEn is asserted.

9.8.2 Effect of receipt

Receipt of this primitive will cause the PMD to multiplex the four bit-streams as required by the link medium and to transmit the packet data over the link.

9.8.3 Additional comments

If an optional, physically exposed MII is implemented, the realization of this primitive is the assertion of transmit enable and the transfer of outgoing data streams through the MII circuits: TxEn, TxClk, and TData [3:0].

9.9 PMD_DATA.indication

This primitive supports the transfer of data from the PMD sublayer to the PMI sublayer.

PMD_DATA.indication (data_bits [3:0])

data_bits: The data_bits parameter represents a single bit in each of the four data streams being transferred from the PMD to the PMI.

9.9.1 When generated

The PMD will begin monitoring the incoming circuits for preamble upon assertion of RxEn. This primitive will be generated by the PMD after preamble detection and acquisition of the receive clock (RxClk[14]). The PMD shall continue generating this primitive on every cycle of RxClk until receipt of a PMD_RECEIVE_ENABLE.request with the receive_enable_state parameter value of "deassert."

9.9.2 Effect of receipt

The receipt of this primitive will cause the PMI sublayer to accept an individual data bit in each of the four bit streams.

9.9.3 Additional comments

If an optional, physically exposed MII is implemented, the realization of this primitive is the transfer of the incoming data streams through the MII circuits: RxClk and RData [3:0].

[14]RxClk is recovered from the incoming data stream by the PMD.

10. Medium Access Control (MAC) frame format structure

10.1 MAC frame formats

The demand-priority access method provides compatibility support for two MAC-frame formats: one based on the ISO/IEC 8802-3 frame format, and the other based on the ISO/IEC 8802-5 frame format[15]. The MAC frames are used for peer-to-peer data exchange between end nodes. A special ISO/IEC 8802-12 frame format is used for training packet construction during link initialization, and a void frame format is defined for use when the start of the incoming frame can not be validly detected.

a) The ISO/IEC 8802-3 compatible frame format is described in 10.2.
b) The ISO/IEC 8802-5 compatible frame format is described in 10.3.
c) The ISO/IEC 8802-12 training frame format is described in 10.6.
d) The void frame format is described in 10.7.

The existence of both ISO/IEC 8802-3 and ISO/IEC 8802-5 frame formats does not imply that individual ISO/IEC 8802-12 LANs should support communication with both formats simultaneously. Rather, it is intended that individual LANs be homogeneous and that they support either format, but not both at the same time. Conversion between ISO/IEC 8802-3 and ISO/IEC 8802-5 compatible formats is a bridging function defined in ISO/IEC 10038:1993.

10.1.1 MAC frame ordering principles

The protocol data units (PDUs) in the MAC and RMAC sublayers are described as a sequence of fields in specific order. Each figure in Clause 10 depicts the fields as they appear in the MAC frame formats, and in the order in which they are transferred, the leftmost field first.

The PDU in the PMI sublayer is an octet. Therefore, the sequence of octets in the fields of the MAC frame forms an octet stream at the MAC/PMI and RMAC/PMI sublayer boundaries. The leftmost octet in each field of the MAC frame is passed across the MAC/PMI or RMAC/PMI boundary first.

The order of bits in each octet of the MAC frame is as depicted in the figures in this clause, and the order of transfer is always leftmost bit first. However, because of the way the MAC frame is constructed, the bit order in the octets of the ISO/IEC 8802-3 MAC frame differs from the bit order in the octets of the ISO/IEC 8802-5 frame:

a) In ISO/IEC 8802-3 MAC frame formats, the leftmost bit of each octet (except those in the FCS field) is the least significant.
b) In ISO/IEC 8802-5 MAC frame formats, the leftmost bit of each octet is the most significant.
c) In ISO/IEC 8802-12 training frames, the leftmost bit of each octet (except those in the FCS field) is the least significant.

Figure 10-1 shows an example of how octets received at the MAC/LLC interface would be represented after conversion by the MAC format interpreters.

[15]ISO/IEC 8802-12 networks operating in the ISO/IEC 8802-5 mode transfer MAC frames only. ISO/IEC 8802-5 tokens are replaced by the request/grant handshake sequence of the ISO/IEC 8802-12 protocol, and hence, are not defined for ISO/IEC 8802-12 use.

Octet hexadecimal values as received from the LLC	A3	8F	D4	
Binary nibble values	1010 0011	1000 1111	1101 0100	
Octets converted to 8802-3 format	11000101	11110001	00101011	···

(a) LLC octet to ISO/IEC 8802-3 format octet conversion

Octet hexadecimal values as received from the LLC	A3	8F	D4	
Binary nibble values	1010 0011	1000 1111	1101 0100	
Octets converted to 8802-5 format	10100011	10001111	11010100	···

(b) LLC octet to ISO/IEC 8802-5 format octet conversion

Figure 10-1—LLC to MAC format octet interpreter examples

Figure 10-2 begins with an existing MAC frame and shows the extracted octet values that would result from ISO/IEC 8802-3 and ISO/IEC 8802-5 format interpreters.

8802-3 hex values	A3	8F	D4	
8802-3 nibbles	1010 0011	1000 1111	1101 0100	
MAC-frame octets	11000101	11110001	00101011	···
8802-5 nibbles	1100 0101	1111 0001	0010 1011	
8802-5 hex values	C5	F1	2B	

Figure 10-2—MAC to LLC format octet interpreter example

MAC addresses are assigned as ordered sequences of bits. The bit order is not modified in either format, and is not changed in bridging. Therefore, the numerical value of each octet in a MAC address is different in the two formats; the Individual/Group (I/G) bit, which is always transferred first, is the least significant bit of the first octet in ISO/IEC 8802-3 format and the most significant in ISO/IEC 8802-5 format. When bridging from one format to another, numerical values of MAC addresses are converted by a bit-reversal operation on each octet.

10.2 Elements of the ISO/IEC 8802-12 MAC frame format—ISO/IEC 8802-3 compatibility mode

This subclause defines the ISO/IEC 8802-12 MAC-frame format and identifies the fields required for the ISO/IEC 8802-3 compatibility mode (see Figure 10-3). Field descriptions are specified by ISO/IEC 8802-3:1996 and are included here for information only. All fields are fixed size, except for the data and pad fields which can contain any integral number of octets between the minimum and maximum defined values.

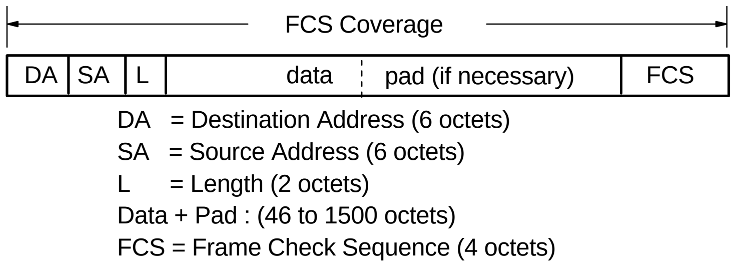

FCS Coverage
DA
SA
L
data
pad (if necessary)
FCS
DA = Destination Address (6 octets)
SA = Source Address (6 octets)
L = Length (2 octets)
Data + Pad : (46 to 1500 octets)
FCS = Frame Check Sequence (4 octets)

Figure 10-3—ISO/IEC 8802-3 MAC frame format

10.2.1 Address fields

Each frame contains two address fields: the Destination Address and the Source Address, in that order. The format of the address fields is shown in Figure 10-4. Each address field will contain 48 bits.

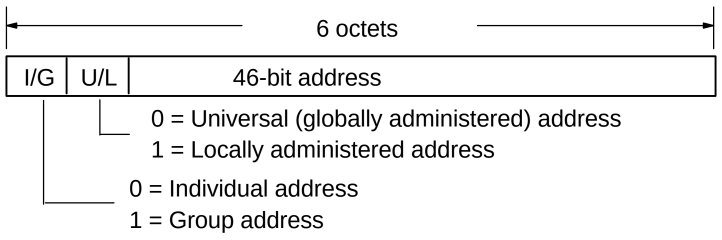

6 octets
I/G
U/L
46-bit address
0 = Universal (globally administered) address
1 = Locally administered address
0 = Individual address
1 = Group address

Figure 10-4—ISO/IEC 8802-3 MAC frame address field format

10.2.1.1 Destination Address (DA) field

The DA field specifies the destination for which the frame is intended. Destination Addresses are furnished by the originating LLC as part of a data transfer command. A Destination Address can be either an individual or a group address, and can be either locally or globally administered.

individual address: The unique address associated with an individual end node on the network. The address of an end node must be distinct from all other individual end node addresses on the same LAN (in the case of local administration) or from the individual addresses of other LAN end nodes on a global basis (in the case of universal administration). Individual addresses can be either unicast addresses or the null address.

unicast address: An individual address identifying an individual end node.

null address: An individual address containing all zeros. The null address indicates that the frame is not addressed to any end node. The null address is used as the Destination Address in training frames and can be used as the Destination Address in void frames. End nodes will not be assigned the null address.

group address: A multidestination address, associated with zero or more end nodes on a given network. In general, group addresses are associated by higher level convention with a group of logically related end nodes. It has been observed that several protocols in general use today require support for one or more group addresses, and that failure to provide end node support for recognition of multiple group addresses can be a serious impediment to the use of that end node in multiprotocol environments. It is recommended that end nodes support recognition of multiple group addresses.

broadcast address: A predefined group address that always denotes the set of all end nodes on a given LAN. All 1's in the DA field is predefined to be the broadcast address.

multicast address: A group address associated with several logically related end nodes.

10.2.1.2 Source Address (SA) field

The SA field identifies the end node from which the frame was initiated. The SA is supplied by the sending MAC sublayer. The I/G bit in the SA field is reserved and is set to 0.

10.2.2 Length (L) field

The L field is a two-octet field whose value indicates the number of LLC data octets in the data field.[16]

10.2.3 Data and pad fields

The data field contains 46–1500 octets. If the LLC data is less than 46 octets, the data field is extended by appending extra octets (a pad) at the end of the LLC data to bring the length of the LLC data plus pad to 46.

Full data transparency is provided in the sense that any arbitrary sequence of octet values can appear in the data field. Octets in the pad are treated as place holders only and can be any arbitrary value. Pads are included in FCS calculations, and are removed by the receiving MAC prior to passing the data to a receiving LLC.

10.2.4 Frame Check Sequence (FCS) field

A CRC procedure is used to generate a CRC value for insertion into the FCS field prior to transmission. The CRC value is computed as a function of the DA, SA, L, data, and pad fields. The encoding is defined by the generating polynomial:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1 \qquad (2)$$

Mathematically, the CRC value corresponding to a given frame is defined by the following procedure:

a) The first 32 bits of the frame are complemented.
b) The n bits of the frame are then considered to be the coefficients of a polynomial M(x) of degree n–1. (The first bit of the DA field corresponds to the $x^{(n-1)}$ term and the last bit of the data field corresponds to the x^0 term.)
c) M(x) is multiplied by x^{32} and divided by G(x), producing a remainder R(x) of degree ≤ 31.
d) The coefficients of R(x) are considered to be a 32-bit sequence.
e) The bit sequence is complemented and the result is the CRC (see Annex G, [B5]).

The 32 bits of the CRC value are placed in the FCS field such that the x^{31} term is the leftmost bit of the first octet, and the x^0 term is the rightmost bit of the last octet, as shown in Figure 10-5. The FCS is transferred commencing with the coefficient of the x^{31} term and proceeding in sequence to the coefficient of the x^0 term.

[16]Frames with a length field value greater than 1500 may be ignored, discarded, or used in a private manner. Use of such frames is beyond the scope of this standard. A length field greater than 1500 may be interpreted as an Ethernet type field.

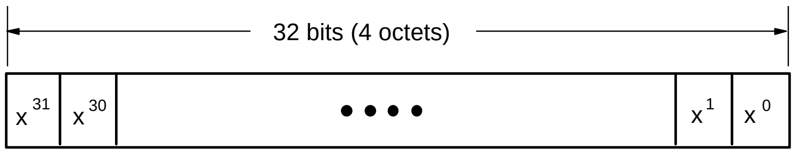
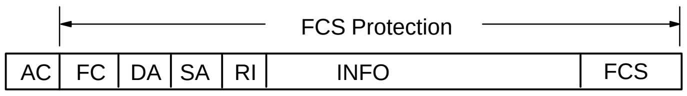


Figure 10-5—ISO/IEC 8802-3 FCS field format

10.3 Elements of the ISO/IEC 8802-12 MAC frame format—ISO/IEC 8802-5 compatibility mode

This subclause defines the ISO/IEC 8802-12 MAC-frame format and identifies the fields required for the ISO/IEC 8802-5 compatibility mode. Field descriptions are specified by ISO/IEC 8802-5:1998 and are included here for information only.

The ISO/IEC 8802-5 MAC frame format is shown in Figure 10-6. It may or may not contain an information (INFO) field. It may or may not contain a Routing Information (RI) field.



AC	= Access Control (1 octet)
FC	= Frame Control (1 octet)
DA	= Destination Address (6 octets)
SA	= Source Address (6 octets)
RI	= Routing Information (0 to 30 octets)
INFO	= Information (0 or more octets)
FCS	= Frame Check Sequence (4 octets)

Note—The combined length of the RI field plus the INFO field must not exceed 4502 octets.

Figure 10-6—ISO/IEC 8802-5 MAC frame format

10.3.1 Access Control (AC) field

The AC field, shown in Figure 10-7, is not used by the ISO/IEC 8802-12 protocol, but is a required part of the ISO/IEC 8802-5 format.

P	P	P	T	M	R	R	R

PPP	= Priority bits
T	= Token bit
M	= Monitor bit
RRR	= Reservation bits

Figure 10-7—ISO/IEC 8802-5 AC field format

10.3.1.1 Priority (PPP) bits

The PPP bits are used in ISO/IEC 8802-5 protocols to indicate the priority of a token and, by implication, when stations are allowed to use the token. The eight levels of priority increase from the lowest (B'000') to the highest (B'111') priority. This bit is reserved, will be transmitted as zero, and will be ignored by the receiver.

10.3.1.2 Token (T) bit

The T bit is used in ISO/IEC 8802-5 protocols to differentiate a token (indicated by a “0”) from a frame (indicated by a “1”). Since ISO/IEC 8802-12 transmits frames only, this bit will be set to 1. The equivalent bit in training and void frames will be set to zero.

10.3.1.3 Monitor (M) bit

The M bit is used in ISO/IEC 8802-5 protocols to prevent a token that has priority greater than B'000' or any frame from continuously circulating on the ring. All frames and tokens are initially transferred with the M bit set to zero. This bit is reserved, will be transmitted as zero, and will be ignored by the receiver.

10.3.1.4 Reservation (RRR) bits

The RRR bits allow ISO/IEC 8802-5 stations with high-priority PDUs to request (in frames or tokens as they are being repeated) that the next token be issued at the requested priority. These bits are reserved, will be transmitted as zero, and will be ignored by the receiver.

10.3.2 Frame Control (FC) field

The FC field, shown in Figure 10-8, identifies the type of ISO/IEC 8802-5 mode frame and user priority.

F	F	Z	Z	Z	Z	Z	Z

FF = Frame type bits
ZZZZZZ = Reserved bits

Figure 10-8—ISO/IEC 8802-5 FC field format

10.3.2.1 Frame type (FF) bits

The FF bits indicate the type of frame to be transferred:

B'00' = undefined format (reserved for future standardization)

B'01' = LLC frame (contains an LLC PDU)

B'1x' = undefined format (reserved for future standardization)

undefined frame format: The values of B'00' and B'1x' are reserved for frame types that may be defined in the future. To ensure compatibility, any future frame formats should conform to the following conditions:

a) The position of the AC and FC fields is unchanged.
b) The AC and FCS fields of the format are separated by an integral number of octets. This number is at least one (the FC field).
c) All bits between the AC and FCS fields can be either zeros or ones.

10.3.2.2 ZZZZZZ bits

LLC frames: If the FF bits indicate an LLC frame, the ZZZZZZ control bits are designated as “rrrYYY.” The “rrr” bits are reserved for future standardization and will be transmitted as B'000' in all transmitted frames, and can be ignored upon reception. The “YYY” bits are used to carry the user priority of the PDU from the source LLC entity to the target LLC entity or entities.

undefined frames: The ZZZZZZ bits are undefined and reserved for future standardization.

10.3.3 Address fields

Each frame contains two address fields: the Destination Address and the Source Address, in that order. Each address field will contain 48 bits.

10.3.3.1 Destination Address (DA) field

The DA field specifies the destination for which the frame is intended. Destination Addresses are furnished by the originating LLC as part of a data transfer command. The format of the DA field is shown in Figure 10-9.

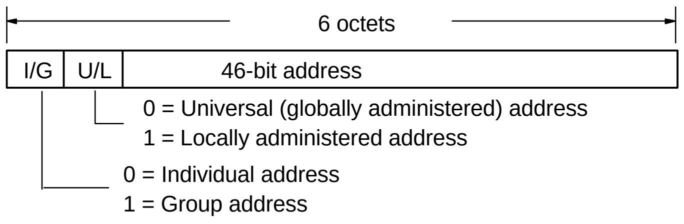

6 octets
I/G
U/L
46-bit address
0 = Universal (globally administered) address
1 = Locally administered address
0 = Individual address
1 = Group address

Figure 10-9—ISO/IEC 8802-5 MAC frame DA field format

A Destination Address can be either an individual or a group address, and can be either locally or globally administered, as indicated in Figure 10-9.

individual address: The unique address associated with an individual end node on the network. The address of an end node must be distinct from all other individual end node addresses on the same LAN (in the case of local administration) or from the individual addresses of other LAN end nodes on a global basis (in the case of universal administration). Individual addresses can be either unicast addresses or the null address.

unicast address: An individual address identifying an individual end node.

null address: An individual address containing all zeros. The null address indicates that the frame is not addressed to any end node. The null address is used as the Destination Address in training frames and can be used as the Destination Address in void frames. End nodes will not be assigned the null address.

group address: A multidestination address, associated with zero or more end nodes on a given network. In general, group addresses are associated by higher level convention with a group of logically related end nodes. It has been observed that several protocols in general use today require support for one or more group addresses, and that failure to provide end node support for recognition of multiple group addresses can be a serious impediment to the use of that end node in multiprotocol environments. It is recommended that end nodes support recognition of multiple group addresses.

broadcast address: A predefined group address that always denotes the set of all end nodes on a given LAN. All 1's in the DA field is predefined to be the broadcast address.

functional addresses (FAs): FAs are bit-significant addresses that are contained within a subset of locally administered group addresses, and that are used to identify well known functional entities. FAs are differentiated from general group addresses and are identified when the 17 highest order bits indicate functional addressing (when the first two octets of the DA are X'C0 00' and the first bit of the third octet is 0). The remaining 31 bits of the address field each represents a separate functional address. Figure 10-10 shows the DA field functional address format, and Table 10-1 lists the functional addresses that have been defined and reserved for a particular purpose. The numerical values of the addresses shown in both Figure 10-10 and Table 10-1 are computed using the MSB-first interpretation of the bit sequence.

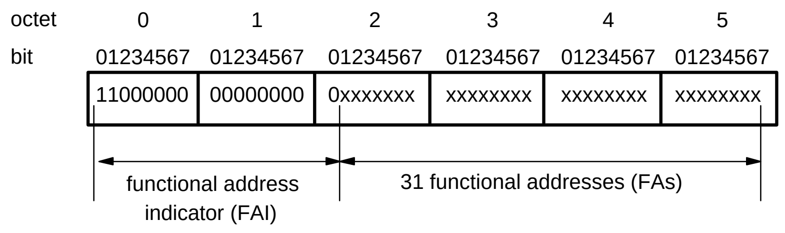


Figure 10-10—ISO/IEC 8802-5 FA format

Table 10-1—ISO/IEC 8802-5 FAs defined for use in MAC frames

Functional Address (FA)	Function name	6-octet address
xxx xxxx xxxx xxxx xxxx xxxx xxxx xxx1	Active monitor	X'C0 00 00 00 00 01'
xxx xxxx xxxx xxxx xxxx xxxx xxxx xx1x	Ring Parameter Server (RPS)	X'C0 00 00 00 00 02'
xxx xxxx xxxx xxxx xxxx xxxx xxxx 1xxx	Ring Error Monitor (REM)	X'C0 00 00 00 00 08'
xxx xxxx xxxx xxxx xxxx xxxx xxx1 xxxx	Configuration Report Server (CRS)	X'C0 00 00 00 00 10'

NOTE—Functional address X'C0 00 00 00 00 80' is widely used by higher-layer protocols (which are sometimes referred to as Netbios®).

The 6-octet address given in Table 10-1 indicates the address for that single function. Multiple functions are addressed by combining functional addresses. An example combined address is X'C0 00 00 00 00 12' which is addressed to both the Ring Parameter Server (RPS) and Configuration Report Server (CRS) functions.

When an end node "sets" a functional address, it will recognize all frames sent to that function and attempt to copy all frames whose DA contains the Functional Address Indicator (FAI) and has the corresponding FA bit set to B'1'. All functional addresses are set by network management.

Use of functional addresses octet 5 bit 1 through octet 5 bit 7 which are not defined above are reserved. Functional addresses octet 2 bit 1 through octet 5 bit 0 are available for general use.

Refer to ISO/IEC TR 11802-2:1995 (Standard Group MAC Addresses) for locally administered MAC group (functional) addresses used by ISO/IEC 8802-5. Also note that functional address X'C0 00 00 00 00 80' is widely used by higher-layer protocols and that functional address X'C0 00 00 00 01 00' is widely used for bridging. Because a functional address can be used simultaneously by multiple protocols, a user can not assume that it designates a single function. Other frame fields should be used to identify the destination function.

10.3.3.2 Source Address (SA) field

The SA field identifies the end node from which the frame was initiated. The Source Address is supplied by the sending MAC sublayer.

Because the Source Address is always an individual address, the implied value for the I/G bit is zero. This bit is used as the Routing Information Indicator (RII) to indicate the presence or absence of a Routing Information (RI) field in the frame. The format of the SA field is shown in Figure 10-11.

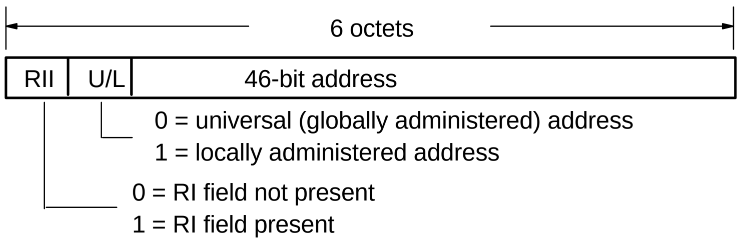

6 octets
RII
U/L
46-bit address
0 = universal (globally administered) address
1 = locally administered address
0 = RI field not present
1 = RI field present

Figure 10-11—ISO/IEC 8802-5 MAC frame SA field format

10.3.4 Routing Information (RI) field

When a frame’s RII bit in the SA field is 1, the RI field will be included in the MAC frame. The format shown in Figure 10-12 provides sufficient information to allow the MAC entity to determine the size of the RI field and to parse the frame properly. For detailed information regarding the structure and contents of the RI field, see ISO/IEC DIS 15802-3 and ISO/IEC 8802-2:1998.

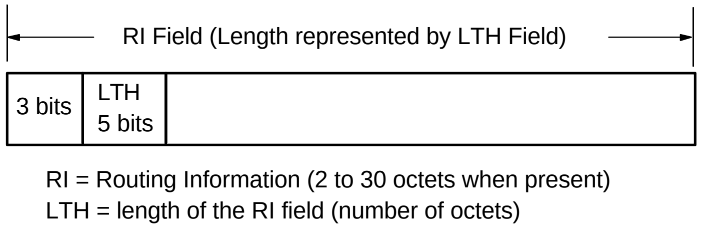

RI Field (Length represented by LTH Field)
3 bits
LTH
5 bits
RI = Routing Information (2 to 30 octets when present)
LTH = length of the RI field (number of octets)

Figure 10-12—ISO/IEC 8802-5 RI field format

10.3.4.1 Length (LTH) bits

These five bits indicate the length (in octets) of the RI field. Length field values will be even values between 2 and 30, inclusive.

10.3.5 Information (INFO) field

The INFO field contains zero, one, or more octets of LLC information. The sum of the lengths of the RI field and the INFO field will be less than or equal to 4502 octets. The format of the INFO field is indicated by the frame type (FF) bits in the FC field.

10.3.6 Frame Check Sequence (FCS) field

A CRC procedure is used to generate a CRC value for insertion into the FCS field prior to transmission. The CRC value is computed as a function of the FC, DA, SA, RI, and INFO fields. The encoding is defined by the generating polynomial:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1 \qquad (3)$$

Mathematically, the CRC value corresponding to a given frame is defined by the following procedure:

a) The first 32 bits of the frame are complemented.

b) The n bits of the frame are then considered to be the coefficients of a polynomial M(x) of degree n–1. (The first bit of the DA field corresponds to the $x^{(n-1)}$ term and the last bit of the data field corresponds to the x^0 term.)

c) M(x) is multiplied by x^{32} and divided by G(x), producing a remainder R(x) of degree ≤ 31.

d) The coefficients of R(x) are considered to be a 32-bit sequence.

e) The bit sequence is complemented and the result is the CRC (see Annex G, [B5]).

The 32 bits of the CRC value are placed in the FCS field such that the x^{31} term is the leftmost bit of the first octet, and the x^0 term is the rightmost bit of the last octet, as shown in Figure 10-13. The FCS will be transferred commencing with the coefficient of the x^{31} term and proceeding in sequence to the coefficient of the x^0 term.

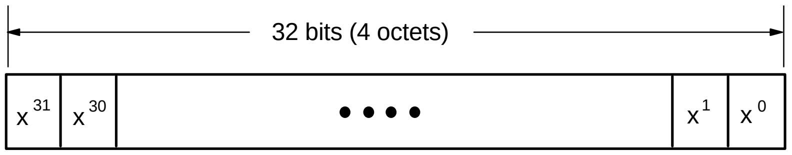


Figure 10-13—ISO/IEC 8802-5 FCS field format

10.4 Address administration

10.4.1 Universal address administration

The structure of universal addresses is defined in IEEE Std 802-1990, Clause 5. All universally administered addresses are globally unique, and are administered by a global authority. The procedure for administration of these addresses is not specified in this standard. Information concerning the registration authority and its procedures can be obtained on request from the Secretary General, ISO Central Secretariat, Case Postale 56, CH-1211 Genève, Switzerland. For information on global address administration, contact the Registration Authority for ISO 8802-5, c/o The Institute of Electrical and Electronics Engineers Inc., 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA.

An end node shall be provided with a universally administered address. An end node may have provision for replacing the universally administered address with a locally administered address.

10.4.2 Local address administration

Local addresses are administered by a local (to the LAN) authority. Network administrators should be careful when assigning locally administered addresses due to the structure of hierarchical addressing. The 14 low-order bits of the first two octets of the address had a special meaning (ring ID). Two values of ring

ID were assigned for use with all stations (Null Ring = all zeros and Any Ring = all ones). Because of this, the following sets of addresses (computed using the MSB-first interpretation of the bit sequence) should be used with care: X'40 00 xx xx xx xx', X'7F FF xx xx xx xx', X'C0 00 xx xx xx xx', and X'FF FF xx xx xx xx'. In addition, locally administered group addresses with the last four octets X'FF FF FF FF' can also be recognized by hierarchical addressing as a broadcast address.

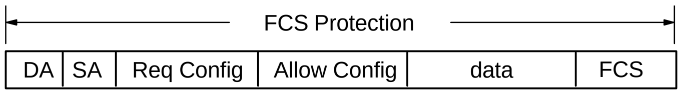
Hierarchical addressing for locally administered addressees is neither required nor specified by this standard. Users are discouraged from implementing hierarchical addressing.

10.5 Invalid MAC frames

An invalid MAC frame is a frame that contains detected errors or an IPM (see 14.4.4). The contents of an invalid MAC frame are not normally passed to an LLC but shall be forwarded by a repeater. The occurrence of invalid MAC frames may be communicated to the LME.

10.6 Elements of the ISO/IEC 8802-12 training frame

Training frames are special MAC frames that are used only during link initialization. Training frames are initially constructed by the MAC (or the RMAC) in the end node (or repeater) at the lower end of a link. Training frames shall be forwarded to all network repeaters. The ISO/IEC 8802-12 training frame format is shown in Figure 10-14.



DA = destination address (6 octets)
SA = source address (6 octets)
Req Config = requested configuration (2 octets)
Allow Config = allowed configuration (2 octets)
Data = data (594 to 675 octets)
FCS = frame check sequence (4 octets)

Figure 10-14—ISO/IEC 8802-12 training frame format

10.6.1 Address fields

Each frame contains two address fields: the Destination Address and the Source Address, in that order. The format of the address fields is shown in Figure 10-15. Each address field shall contain 48 bits.

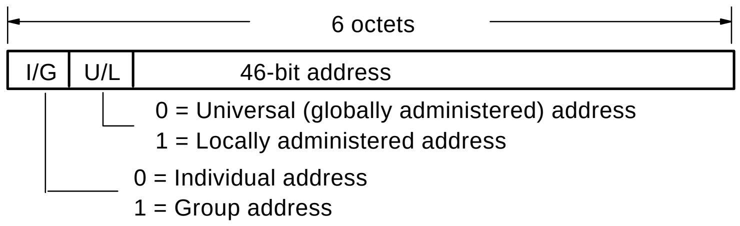

6 octets
I/G
U/L
46-bit address
0 = Universal (globally administered) address
1 = Locally administered address
0 = Individual address
1 = Group address

Figure 10-15—ISO/IEC 8802-12 training frame address field format

10.6.1.1 Destination address

Because training is a procedure that directly involves only the repeater and the lower entity, and because the destination of the training packet is the repeater at the upper end of the link, the destination address shall be the null address.

10.6.1.2 Source address

Part of the link initialization process is to establish the network address of the connected lower entity initiating the training session:

a) If the lower entity is an end node, the Source Address shall be the individual address of the end node unless a non-null individual address has not yet been assigned to the end node. In the latter case, the link may be tested by training with the null address. The end node shall not be allowed to join the network until a non-null address is assigned.
b) If the lower entity is a repeater, the Source Address shall normally be the null address. If the repeater contains an integrated LME with an assigned non-null individual address, the repeater may train with that address.

An end node may train with a null address to verify link operability, but training will not be successfully complete until the end node has trained with an assigned individual address.

10.6.2 Requested configuration field format

The requested configuration field allows the lower entity to inform the connected higher-level repeater about itself and to request a port configuration. Figure 10-16 shows the requested configuration field format. The training response packet from the RMAC shall contain the lower entity's requested configuration.

R	P	P	F	F	r	r	r	r	r	r_n	r_c	D	v	v	v

R	=	repeater bit
PP	=	promiscuous bits
FF	=	format bits
r	=	reserved bit (set to 0)
r_n	=	not used–this bit corresponds with the access not allowed bit in the allowed configuration field (set to 0)
r_c	=	not used–this bit corresponds to the configuration bit in the allowed configuration field (set to 0)
D	=	duplicate (redundant) uplink bit
vvv	=	version bits

Figure 10-16—ISO/IEC 8802-12 training frame requested configuration field format

10.6.2.1 Repeater (R) bit

The R bit allows the training initiator to inform the connected higher-level repeater whether it is a repeater or an end node. An end node may be any MAC-based device such as a user station, a bridge, or a LAN analyzer.

a) R = 0, the training initiator is an end node
b) R = 1, the training initiator is a repeater

10.6.2.2 Promiscuous (PP) bits

The PP bits provide a means for a lower entity to request the class of unicast addressed packets it wishes to receive:

a) PP = 00, receive only unicast packets specifically addressed to this end node
b) PP = 01, reserved for future standardization
c) PP = 10, receive all packets forwarded by the local repeater (promiscuous mode)
d) PP = 11, reserved for future standardization

Repeaters, bridges, and network analyzers shall always request that all packets be forwarded by the local repeater (promiscuous mode). Group-addressed packets shall be forwarded to all active ports regardless of the PP configuration.

10.6.2.3 Format (FF) bits

The FF bits allow the training initiator to request the operational format it wishes to use.

a) FF = 00, ISO/IEC 8802-3 format is requested
b) FF = 01, reserved for future standardization
c) FF = 10, ISO/IEC 8802-5 format is requested
d) FF = 11, either ISO/IEC 8802-3 or ISO/IEC 8802-5 format is acceptable

10.6.2.4 r bits

The r bits are reserved for future standardization. They shall be transmitted as zero and ignored by the receiver.

10.6.2.5 Duplicate uplink (D) bit

The duplicate uplink bit indicates whether or not a lower entity is training this link as the secondary link.

a) D = 0, either the link is being trained as the primary link, or the lower entity does not have redundant-link capability.
b) D = 1, the link is being trained as the secondary link.

The D bit is used by the training initiator to request that the higher-level entity disable duplicate-address checking for that local port. A value of one in the D bit of the requested configuration field shall cause a repeater with local-port-redundant-link capability to ignore the MAC address attribute of this port when performing duplicate-address checks. A value of one shall also always cause a repeater with local-port-redundant-link capability to generate a value of zero for the D bit in the allowed configuration field of the training response frame, regardless of the value of the aCentralMgmtDetectedDupAddr network management attribute. The D bit in the requested configuration field is informational for network management.

10.6.2.6 vvv bits

The vvv bits are used to identify the version of the ISO/IEC 8802-12 MAC/RMAC training protocol with which the training initiator is compliant. The current version shall be numbered 001.

10.6.3 Allowed configuration field format

The allowed configuration field allows the connected higher-level repeater to respond with the allowed configuration. Figure 10-17 shows the allowed configuration field format. The training initiator (the lower entity) shall set the contents of the allowed configuration field to zero. The RMAC shall set the bits of the allowed configuration field as defined in 12.9.8.

R	P	P	F	F	r	r	r	r	r	N	C	D	v	v	v

R = repeater bit	N = access not allowed bit
PP = promiscuous bits	C = configuration bit
FF = format bits	D = duplicate address bit
r = reserved (set to 0)	vvv = version bits

Figure 10-17—ISO/IEC 8802-12 Allowed configuration field format

10.6.3.1 Repeater (R) bit

The R allows the repeater to indicate how the lower entity will be treated.

a) R = 0, requested access as an end node is allowed
b) R = 1, requested access as a repeater is allowed

10.6.3.2 Promiscuous (PP) bits

The PP bits provide a means for the repeater to indicate the class of unicast addressed packets the lower entity will receive.

a) PP = 00, the lower entity will receive only unicast packets specifically addressed to it
b) PP = 01, reserved for future standardization
c) PP = 10, the lower entity will receive all packets forwarded by the local repeater
d) PP = 11, reserved for future standardization

10.6.3.3 Format (FF) bits

The FF bits allow the repeater to indicate the operational format that is allowed.

a) FF = 00, ISO/IEC 8802-3 format will be used
b) FF = 01, reserved for future standardization
c) FF = 10, ISO/IEC 8802-5 format will be used
d) FF = 11, reserved for future standardization

10.6.3.4 r bits

The r bits are reserved for future standardization and shall be set to zero.

10.6.3.5 Access not allowed (N) bit

The N bit is provided for private use to allow the repeater to indicate that the lower entity will not be allowed to join the network for reasons other than configuration (e.g., because of security restrictions).

a) N = 0, access will be allowed only if the configuration is compatible with the network (when C = 0)
b) N = 1, access will not be allowed (even if the configuration is compatible)

Use of the N bit is optional. If the N bit is not used, it shall be set to zero.

10.6.3.6 Configuration (C) bit

The C bit indicates whether the requested configuration is compatible with the network.

a) C = 0, the configuration is compatible with the network
b) C = 1, the configuration is not compatible with the network[17] [in this case, the FF, PP, and R bits shall indicate the configuration that would be allowed (providing N = 0)]

10.6.3.7 Duplicate address (D) bit

The duplicate address bit indicates whether or not a duplicate address has been detected in the repeater's port address list.

a) D = 0, no duplicate address has been detected.
b) D = 1, a duplicate address has been detected.

If both links from a redundant-link training initiator are connected to local ports on the same higher-level repeater and that repeater is not equipped with local-port-redundant-link training capability, the D bit in the requested configuration field will be ignored and a duplicate address may be detected. In this case, it will be necessary for the lower entity to train at least one link with the null address so that the primary link can train with this repeater without duplicate address error.

10.6.3.8 vvv bits

The vvv bits identify the version of the ISO/IEC 8802-12 MAC/RMAC training protocol with which the repeater is compliant. The current version shall be numbered 001.

10.6.4 Training frame data field

The training frame data field shall contain between 594 and 675 octets as constructed by the training initiator. The first 55 octets may be used for private protocol information. The remaining octets shall all be set to zero. The last four octets of the data field in the RMAC's response frame may be any value and the data field may be four octets longer than the received data field in the lower entity's request frame.

a) If some or all of the first 55 octets are used for private protocol information:
    1) The first five octets shall be a protocol identifier assigned in accordance with 5.3 of IEEE Std 802-1990.[18]
    2) Any unused octet in the remaining 50 shall be set to zero.
b) If the first 55 octets are not used for private purposes, all shall be set to zero.

Because there is a scrambler in the Physical Layer, the all zeros portion of this data field provides pseudo-random output from the transmitter. Transceivers can use this input to characterize the frequency response of the cable. This information can be used to adjust circuits such as equalizers and phase-locked loops when the link is initialized. The private protocol field is limited in length so that most of the data field will be zeros. The length of the training frame combined with the requirement that 24 consecutive

[17]An incompatible configuration may be caused by a mismatch between the requested and allowed R, PP, FF, or vvv fields, or by a duplicate or null address in an end node.

[18]At least one of the first three octets in the protocol identifier will be nonzero.

training frames be received without error to complete training ensures that noisy links will not complete training.

10.6.5 Training frame FCS field

A CRC procedure is used to generate a CRC value for insertion into the FCS field prior to transmission. The CRC value is computed as a function of the DA, SA, requested configuration, allowed configuration, and data fields. The encoding is defined by the generating polynomial:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1 \quad (4)$$

Mathematically, the CRC value corresponding to a given frame is defined by the following procedure:

a) The first 32 bits of the frame are complemented.
b) The n bits of the frame are then considered to be the coefficients of a polynomial M(x) of degree n–1. (The first bit of the DA field corresponds to the $x^{(n-1)}$ term and the last bit of the data field corresponds to the x^0 term.)
c) M(x) is multiplied by x^{32} and divided by G(x), producing a remainder R(x) of degree ≤ 31.
d) The coefficients of R(x) are considered to be a 32-bit sequence.
e) The bit sequence is complemented and the result is the CRC (see Annex G, [B5]).

The 32 bits of the CRC value are placed in the FCS field such that the x^{31} term is the leftmost bit of the first octet, and the x^0 term is the rightmost bit of the last octet, as shown in Figure 10-5. The FCS shall be transferred commencing with the coefficient of the x^{31} term and proceeding in sequence to the coefficient of the x^0 term.

10.7 Elements of the ISO/IEC 8802-12 void frame

The repeater shall generate a void frame when the acknowledged device does not send a packet within the expected time window or if the start of a packet is erred such that the PMI can not decode the data values:

a) If an invalid start frame delimiter is detected in an incoming frame, the void frame shall consist entirely of zeros and shall be approximately the same length as the incoming frame.
b) If no incoming frame is received, the void frame shall consist of 550–600 zero octets.
c) If the incoming frame is delayed, the void frame may consist of one or more zeros followed by incoming frame data. The frame length shall be the length of the incoming frame plus the generated leading zeros, but shall not exceed 4520 octets.

If an error is detected in the start of frame sequence of an incoming packet, or if a packet is delayed or fails to arrive from a selected lower entity, the repeater generates a void frame to maintain network activity and prevent time-outs in other repeaters on the network. Void frames are always marked with IPMs (Physical Layer delimiters that identify packets transmitted with errors), are forwarded by receiving repeaters, and are ignored if received by an end node. The minimum void frame length ensures that, before the end of the void frame, the repeater will receive a void frame from the lower repeater if the lower repeater is not the source of the problem. This allows the repeaters to locate and isolate a defective link. The maximum void frame length prevents the concatenated void frames in a maximum size network from exceeding a maximum packet size.

11. Medium Access Control (MAC) protocol

11.1 Scope

The MAC is the primary control layer in an end node. This clause describes the data functions and control operations of the MAC sublayer (see Figure 11-1). The descriptions are provided as state diagrams, supplemented by prose annotations, with state definitions in pseudo code. All are normative. Should differences be perceived in interpretation, pseudo code definitions shall take precedence. No particular physical implementation is implied.

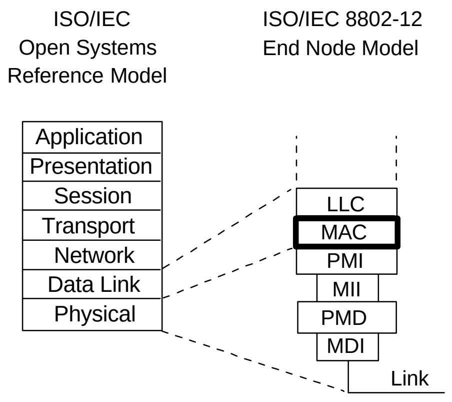


Figure 11-1—Relationship to the LAN model

An end node may be equipped with a second MAC and a second (redundant) link to maintain network connectivity in case of link or repeater failure. Recommendations for connecting redundant-link end nodes in an ISO/IEC 8802-12 network are given in E.5. The definition of redundant links for end nodes and the manner in which reception of duplicate packets is resolved if both links are simultaneously active are implementation issues and are beyond the scope of this standard.

11.2 Overview of the MAC operations

The MAC sublayer provides end node access control to the link medium for data transmission and reception and for control signal exchange with the repeater's RMAC. It accepts MA_UNIT-DATA.requests (see 7.3) from the LLC, or M_UNITDATA (see 7.4) from a bridge, and constructs an appropriate MAC frame for transmission in either ISO/IEC 8802-3 or ISO/IEC 8802-5 format. The MAC:

a) Adds its own address as the source address if the source address is not provided by the LLC.
b) Computes a CRC value and inserts it into the FCS field.[19]
c) Issues a control signal request for permission to send, based on the priority level indicated by the LLC.
d) Initiates packet transmission after a Grant (to send) has been received from the repeater.

The MAC also reacts to control signals from the repeater indicating a possible incoming packet by requesting the physical sublayers to switch to the receive mode. When data packets are received, the MAC verifies the destination address, performs an FCS check on the received frame, and checks whether any other transmission errors have been detected, or if an IPM has been received. If all error checks are

[19]When bridging between like formats, the CRC value may be included with the received frame.

negative in an end node, the MAC frame is converted back to LLC frame format and an MA_UNITDATA.indication (DA, SA, Routing Information, msdu, priority)[20] is made to the LLC. If all error checks are negative in a bridge, the MAC makes an M_UNITDATA.indication (frame_type, mac_action, DA, SA, Routing Information, msdu, user_priority, FCS)[21] to the bridge entity.

The definition of the MAC sublayer in the remainder of this clause is based on service to an LLC sublayer. If the upper sublayer is a bridge entity, the MA_UNITDATA.request and MA_UNITDATA.indication primitives identified in Autonomous process 1 : Wait for MA_UNITDATA.request (11.4.1.1), in the PROCEDURE Process_Received_MAC_Frame (11.5.6), and in the FUNCTION Build_Frame (11.5.7) is replaced by the appropriate M_UNITDATA.request or M_UNITDATA.indication primitive with appropriate additional parameters as indicated in 7.4.

11.2.1 Link control

Arbitration and control of data traffic are accomplished through peer level exchange of control request and indication primitives between the MAC sublayer in the end node and the RMAC sublayer in the repeater as shown in Figure 11-2:

a) Outgoing control requests are specified in the transmit_control_state parameter value of the PMI_CONTROL.request primitives generated by the MAC.

b) Incoming control signals are indicated in the receiver_control_state and grant_state parameters of the PMI_CONTROL.indication primitives generated by the PMI.

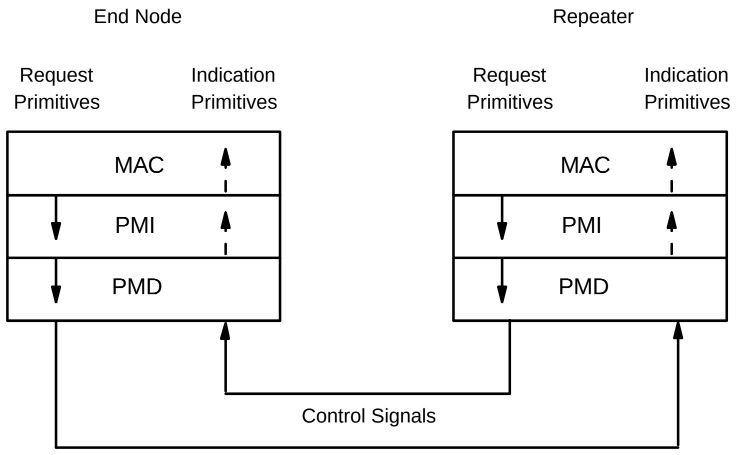


Figure 11-2—Link control

11.2.1.1 Control state definitions

The transmit control states provide a means for the MAC to request, via the PMI, a particular control action to be taken (or control signal to be sent) by the PMD sublayer. The grant state and receiver control states provide a means for the PMD to indicate that a link condition or control signal has been received. The grant state indicates when packet transmission may proceed.

[20] The Routing Information field in an MA_UNITDATA.indication is only applicable to the ISO/IEC 8802-5 format (see 7.3, figures 15 and 16).

[21] The frame_type and mac_action parameters are encoded as constants. The Routing Information field in an M_UNITDATA.indication is only applicable to the ISO/IEC 8802-5 format (see 7.4).

11.2.1.2 Transmit control states

A new TCS may be requested by the MAC while the outgoing frame is being transferred to the PMI, but the equivalent control signals will not be sent by the PMD until after packet transmission is fully complete. The TCS definitions are:

Idle_Up: Idle_Up indicates that the end node is in the idle state and is not requesting to send a packet.

Request_High, Request_Normal: The TCSs used by the end node to inform the repeater that it wishes to send a packet:

a) Request_High indicates that a high-priority packet needs to be sent.
b) Request_Normal indicates that a normal-priority packet needs to be sent.

Training_Up: Training_Up indicates that link training is desired or is continuing.

TxDisable: The TCS that allows the MAC to request that the PMD's transmitters be disabled.

11.2.1.3 Receiver control states

RCS values indicate either the current or previous RCS, depending on the grant state:

a) If grant_state = 0, the RCS represented by the RCS code value indicates the current control state received at the MDI.
b) If grant_state = 1, the RCS represented by the RCS code value indicates the control state received at the MDI prior to grant_state being decoded.

Data_on_Link: The RCS used by the PMD to indicate that data streams (rather than control signals) are being received from the link.

Grant: The control state that indicates that the repeater has granted the end node permission to send one packet. Grant is a separate code used in conjunction with the RCS.

Idle_Down: Idle_Down indicates that the repeater currently has no packet that is destined for the end node.

Incoming: The RCS used by a repeater to alert the end node that a packet may be incoming.

Link_Warning: The RCS used by the PMD to indicate a condition that may require the link to be retrained.

Mode Transition: The RCS used by the PMD to indicate that the signals on the link are in transition between data and control modes.

Training_Down: The RCS that indicates that training may proceed.

11.2.1.4 Grant state coding

A grant_state = "1" indicates that the end node may send one packet. A grant_state = "0" indicates that a packet may not be sent.

11.2.2 Internal control

Control requests for, and indications of, actions that are internal to the end node are also defined by PMI primitives:

a) PMI_TRANSMIT.indication allows the PMI to inform the MAC when packet transmission is complete[22] (see 8.6).
b) PMI_RECEIVE_ENABLE.request defines when the physical sublayers are to switch to the data reception mode (see 8.8).

11.3 MAC operational modes

At the conceptual level, the MAC can be viewed as being in any of the three meta-states shown in Figure 11-3:

a) Training (initializing) the link in preparation to join the network.
b) Active and able to communicate with peer entities elsewhere on the network.
c) Not yet able to complete training and waiting to join the network.

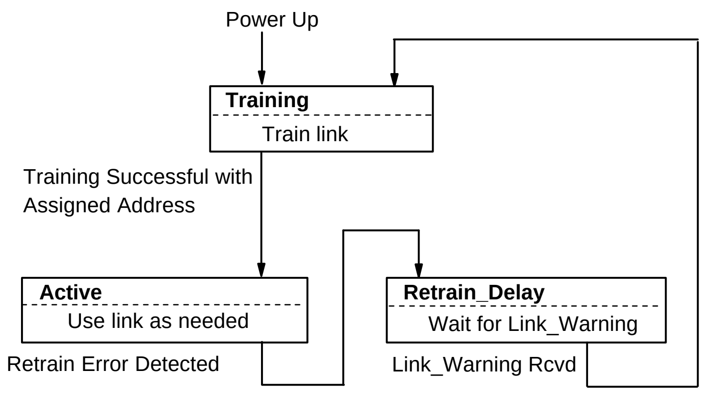
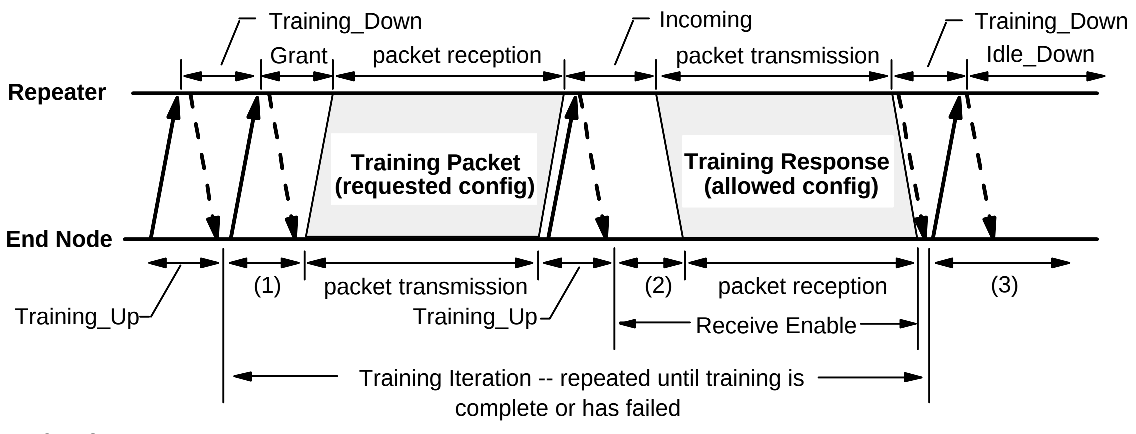


Figure 11-3—Meta-state diagram

11.3.1 Training mode

The link must be initialized each time the end node is powered up or whenever an error condition indicates that the link may not be operating correctly. Training is initiated by the end node and is accomplished by the successful exchange of training frames with the repeater as depicted in the time-space diagram in Figure 11-4.

[22]Because start of stream and end of stream sequences are generated by the PMI rather than the MAC, the PMI_ TRANSMIT.indication (transmit_complete) occurs a number of clock cycles after the transfer of octets from the MAC to the PMI is complete.



NOTES

1— Request_Normal

2— The control signal sent after receipt of Incoming depends on the PMD/Link configuration:

a) In half duplex links: the link is cleared of out-going control signals to prepare for incoming data

b) In dual simplex links: Training_Up shall continue to be sent during the entire time RxEn is asserted.

3— Idle_Up if training is complete, Training_Up if training has failed.

Figure 11-4—Link training time-space diagram

The training process begins with the end node requesting training by generating a PMI_CONTROL.request (Training_Up) primitive. Training begins with the receipt of a PMI_CONTROL.indication (Training_Down) primitive and continues until a series of consecutive training frames (training iterations in Figure 11-4) have been successfully exchanged between the end node's MAC and the repeater's RMAC sublayers:

a) If 24 consecutive packets are received without error, the C and N bits are zero after the last packet, and the MAC has trained with an assigned address, then the MAC indicates success by generating a PMI_CONTROL.request (Idle_Up) primitive [rather than PMI_CONTROL.request (Request_ Normal)] after the last received training packet. It transitions to the active state upon receipt of a PMI_CONTROL.indication (Idle_Down or Incoming) primitive.

b) If training fails, the MAC waits for a 1–2 s delay and then initiates a new training sequence.

The end node shall neither send nor receive normal data traffic while training is in progress.

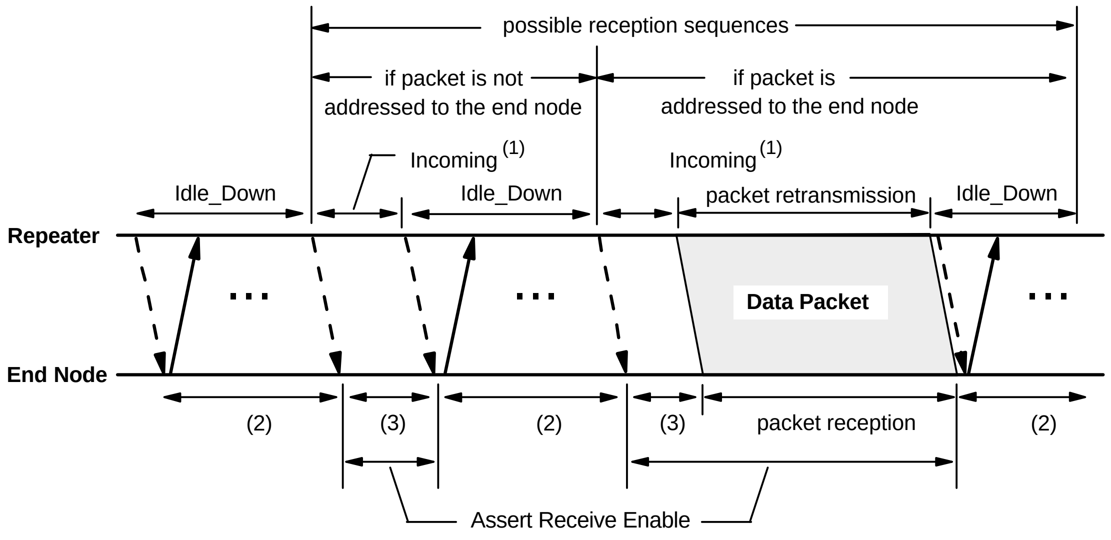
11.3.2 Active mode

The end node may both send and receive data traffic while in the active mode. The possible reception sequences are shown in Figure 11-5.

a) Receipt of a PMI_CONTROL.indication (Incoming) primitive alerts the MAC that a data packet may be incoming. Because the repeater does not know the destination address of the packet at the time it sends Incoming, the end node may or may not actually receive a packet at this time.

b) The MAC generates a PMI_RECEIVE_ENABLE.request (assert) and switches to the data receive mode.

c) After packet reception or receipt of Idle_Down, the MAC generates a PMI_CONTROL.request (Idle_Up, Request_High, or Request_Normal) to indicate whether it has transmission requests pending and, if so, the highest priority level.



NOTES

1—Incoming takes precedence over requests for permission to send.

2—Request_Normal, Request_High, or Idle_Up

3—The control signal sent by the PMD after receipt of Incoming depends on the PMD/Link configuration and is transparent to the MAC:

a) In half duplex links: the link is cleared of out-going control signals to prepare for incoming data.

b) In dual simplex links: Request_Normal, Request_High, or Idle_Up shall continue to be sent during the entire time RxEn is asserted.

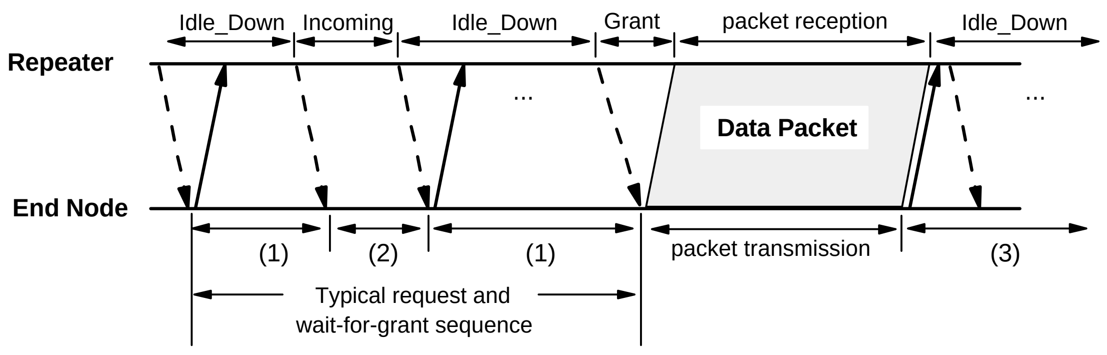
Figure 11-5—Packet reception time-space diagram

The transmission sequence is shown in Figure 11-6. Since the end node must wait its turn in the repeater's round-robin cycle, several control sequences of Incoming followed by Idle_Down (the request, wait for grant sequence in Figure 11-5) may occur before the repeater grants permission to send:

a) Receipt of a PMI_CONTROL.indication primitive with a grant_state parameter equal to “one” indicates permission to send one data packet.

b) The MAC switches to the data transmit mode and generates a PMI_UNITDATA.request.

c) After packet transmission is complete, the MAC generates a PMI_CONTROL.request primitive showing Idle_Up, Request_High, or Request_Normal to indicate whether it has additional transmission requests pending and, if so, the priority level.



NOTES

1—Request_Normal or Request_High

2—The control signal sent by the PMD after receipt of Incoming depends on the PMD/Link configuration and is transparent to the MAC:

a) In half duplex links: the link is cleared of out-going control signals to prepare for incoming data.

b) In dual simplex links: Request_Normal, Request_High, or Idle_Up shall continue to be sent during the entire time RxEn is asserted.

3—Request_Normal, Request_High, or Idle_Up

Figure 11-6—Packet transmission time-space diagram

11.3.3 Retrain_Delay mode

The MAC shall enter the Retrain_Delay mode whenever a retraining error has been detected. The MAC enters this mode because an error has occurred that requires the link to be retrained. While in this state the MAC will disable its transmitter. The MAC remains in this mode until it sees Link_Warning, which indicates that the RMAC is aware that the link must be retrained.

11.3.3.1 Link errors

During operation, the state machine may encounter errors that could be caused, for example, by a noisy or damaged link. When errors requiring link retraining occur, the MAC sets the variable RETRAIN_LINK to True, which causes the autonomous process Monitor Retrain_Link to generate a RETRAIN_LINK_ EVENT. The RETRAIN_LINK_EVENT will cause the MAC to transition first to the Retrain_Delay mode, and then to the Training mode.

11.4 The MAC logical structure

The MAC state machine consists of nine states supported by four autonomous processes as shown in Figure 11-7. It must support two logical interfaces: to the LLC (defined by the service primitives described in Clause 7), and to the PMI (defined by the service primitives described in Clause 8).

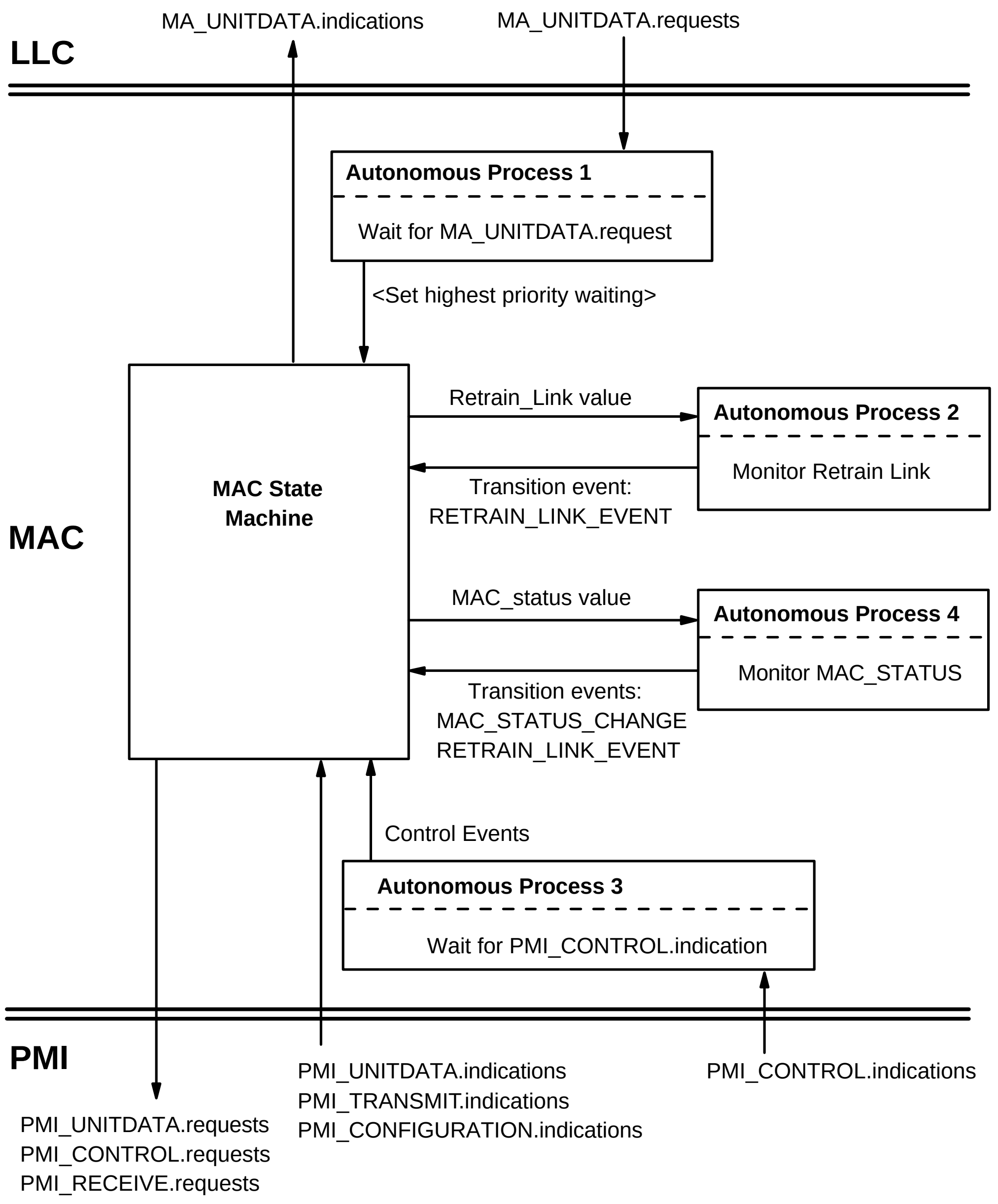


Figure 11-7—MAC state machine and autonomous processes

11.4.1 Autonomous processes

All autonomous processes shall generate the appropriate event on each cycle of the PMI transmit clock (TxClk).

11.4.1.1 Wait for MA_UNITDATA.request

This process waits for an MA_UNITDATA.request and updates the MAC state machine variable HIGHEST_PRIORITY_WAITING to indicate the priority of the highest priority packet that is waiting to be transmitted. The parameters of the MA_UNITDATA.request are used to build a frame as described in the function Build_Frame (see 11.5.7).

The ISO/IEC 8802-12 protocol provides two levels of priority: "normal" and "high." The MA_UNITDATA.request parameter priority is replaced by the formal parameter LLC_PRIORITY in order to emphasize that the LLC and MAC may use different sets of values for priority. This autonomous process must map the LLC_PRIORITY parameter to a value that the MAC can use:

a) For ISO/IEC 8802-3 frames, the transmission priority shall be either "normal" or "high." If no priority level is indicated in the MA_UNITDATA.request priority parameter, the default priority level should be "normal."
b) For ISO/IEC 8802-5 frames, the priority parameter range is 0 to 7. The default levels should be:
    1) Normal-priority for priority parameter values of 0 to 4.
    2) High-priority for priority parameter values of 5 to 7.

11.4.1.2 Monitor RETRAIN_LINK

This process examines the variable RETRAIN_LINK and generates RETRAIN_LINK_EVENT whenever the variable RETRAIN_LINK is found to be true. RETRAIN_LINK_EVENT is a case variable used in MAC3_IDLE_OR_REQUEST.

11.4.1.3 Monitor MAC_STATUS

This process examines the variable MAC_STATUS and generates an event MAC_STATUS_CHANGE whenever the variable MAC_STATUS is set to "closed" or whenever it changes from "closed" to some other value. In addition, this process must generate an event RETRAIN_LINK_EVENT whenever MAC_STATUS is set to "closed" by a network management **acClose** action—this will force the link to go to the Retrain_Delay mode.

11.4.1.4 Wait for PMI_CONTROL.indication

This process waits for a PMI_CONTROL.indication and then issues the appropriate event to the MAC state machine. At each indication received, the process generates the appropriate event:

```
Case (grant_state) Of
    true : RCS_IS_GRANT;
End {Case}
Case (receiver_control_state) Of
    Mode_Transition : RCS_IS_MODE_TRANSITION;
    Idle_Down   : RCS_IS_IDLE_DOWN;
    Incoming    : RCS_IS_INCOMING;
    Training_Down  : RCS_IS_TRAINING_DOWN;
    Data_on_Link   : {not used};
    Link_Warning   : RCS_IS_LINK_WARNING;
    default  : RCS_IS_ILLEGAL;
End {Case}
```

This process detects illegal RCS values passed up from the PMI and indicates such a value as RCS_IS_ILLEGAL. This process also sets the RCS variable to the value of receiver_control_state. RCS is examined in the MAC state machine state 3: MAC3_IDLE_OR_REQUEST.

11.4.2 MAC state machine structure

The structure of the MAC state machine is shown in Figure 11-8. States 3 through 8 are involved in two modes: training and active.

a) Training is initiated in States 1 and 2. The MAC will alternate between these states until the repeater indicates that it will allow the training process. The actual training process is accomplished in States 3 through 8.
b) The active mode supports three activities (packet transmission, packet reception, and idle) and involves only States 3 through 8:
    1) When the MAC receives a frame, it cycles through the state sequence:
        i) MAC3_IDLE_OR_REQUEST
        ii) MAC4_WAIT_FOR_START_MAC_FRAME.
        iii) MAC5_READ_FRAME.
        iv) MAC8_ACQUIRE_CONTROL_SIGNAL
    2) When the MAC sends a frame, it cycles through the sequence:
        i) MAC3_IDLE_OR_REQUEST
        ii) MAC6_TRANSMIT_FRAME
        iii) MAC7_WAIT_FOR_PMI_TRANSMIT_INDICATION
        iv) MAC8_ACQUIRE_CONTROL_SIGNAL
    3) When the MAC is idle, it remains in State 3.
c) The Retrain_Delay mode is defined by State 9. This state is used when an error occurs.

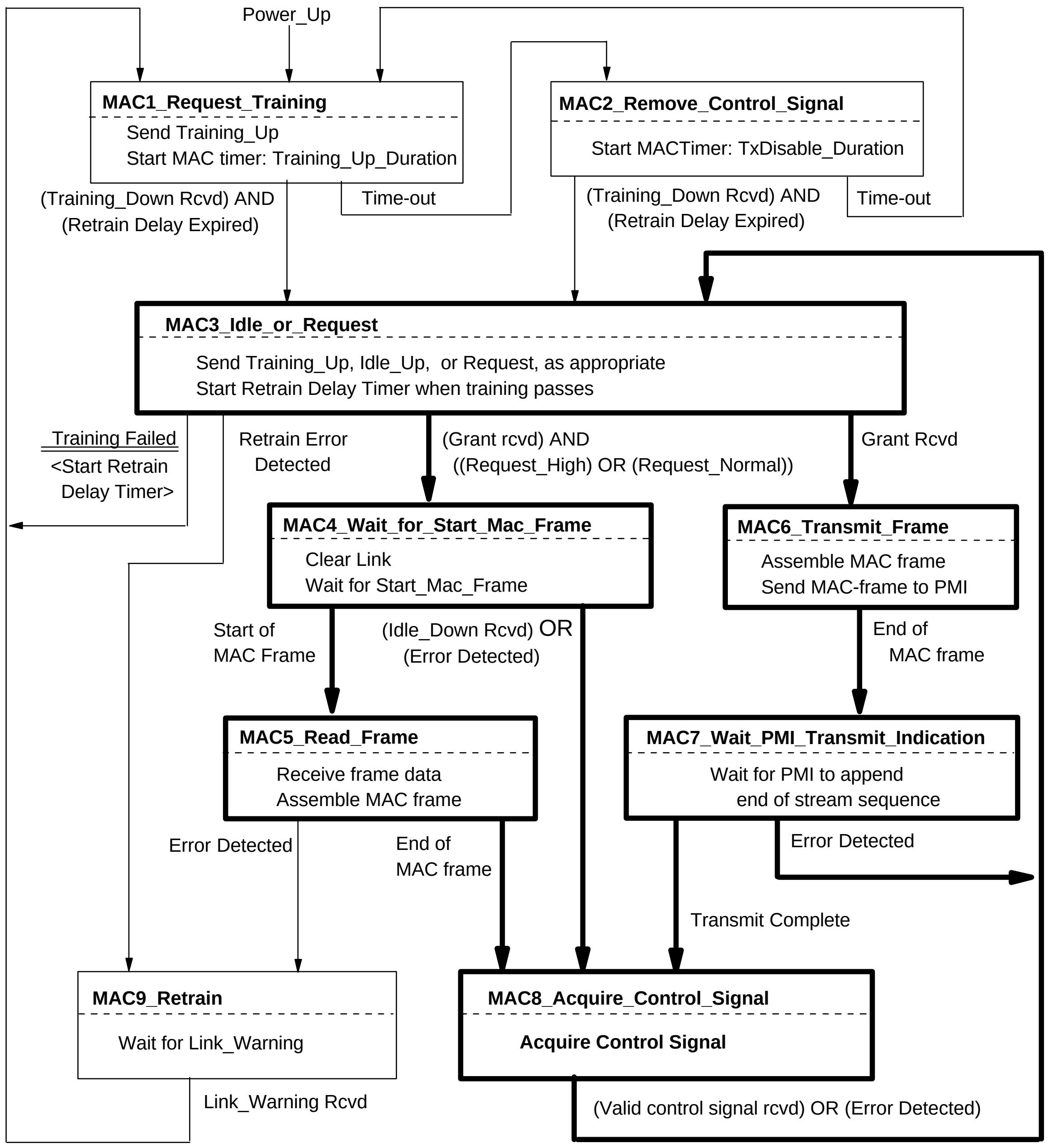

Power_Up
MAC1_Request_Training
Send Training_Up
Start MAC timer: Training_Up_Duration
MAC2_Remove_Control_Signal
Start MACTimer: TxDisable_Duration
(Training_Down Rcvd) AND (Retrain Delay Expired)
Time-out
(Training_Down Rcvd) AND (Retrain Delay Expired)
Time-out
MAC3_Idle_or_Request
Send Training_Up, Idle_Up, or Request, as appropriate
Start Retrain Delay Timer when training passes
Training Failed
<Start Retrain Delay Timer>
Retrain Error Detected
(Grant rcvd) AND ((Request_High) OR (Request_Normal))
Grant Rcvd
MAC4_Wait_for_Start_Mac_Frame
Clear Link
Wait for Start_Mac_Frame
MAC6_Transmit_Frame
Assemble MAC frame
Send MAC-frame to PMI
Start of MAC Frame
(Idle_Down Rcvd) OR (Error Detected)
End of MAC frame
MAC5_Read_Frame
Receive frame data
Assemble MAC frame
MAC7_Wait_PMI_Transmit_Indication
Wait for PMI to append end of stream sequence
Error Detected
End of MAC frame
Error Detected
Transmit Complete
MAC9_Retrain
Wait for Link_Warning
MAC8_Acquire_Control_Signal
Acquire Control Signal
Link_Warning Rcvd
(Valid control signal rcvd) OR (Error Detected)

Figure 11-8—MAC state machine structure

11.5 MAC state machine general definitions

11.5.1 Counters

The MAC state machine uses two counters:

SUCCESSFUL_PACKET_PAIRS_COUNTER. Integer; This counter counts the number of training frame pairs that have been exchanged in succession without error.

TRAINING_FRAMES_SENT. Integer; This counter counts the number of training frames sent during the current training attempt.

11.5.2 Timers

The MAC state machine uses three timers:

MAC_Retraining_Delay_Timer. This timer is used to ensure that the MAC does not attempt a training sequence more frequently than a set rate. This timer is started either if training fails, or at the moment that training succeeds. It is checked in states MAC1 and MAC2.

MAC_Timer. This is a general-purpose timer.

MAC_Train_Response_Timer. This timer is started when the MAC is training and has just sent a training frame. This timer expires if the MAC does not receive an allowed frame within a certain period. This timer is examined in state MAC3.

This timer is also used when the end node decides that training is successful. At the instant the end node sends Idle_Up to the repeater, the MAC starts this timer with a time-out value defined by Training_Final_Idle_or_Incoming_Duration. The end node expects to see either Incoming or Idle_Down before this timer expires; otherwise, the end node will retrain the link. This timer is examined in state MAC3.

11.5.2.1 Timer operations

Two operations are defined for timers: Start_Timer() and Timer_Expired(). The initial condition for all timers is: “expired.”

Start_Timer (Timer_name, Timer value). The timer is started by calling the function and when the timer expires, an appropriate time-out event is generated. The time-out event is generated only once and is ignored by the MAC unless that event is currently of significance.

Timer_Expired (Timer_name). This function allows the MAC to query the status of the named timer. It will return true if the timer has expired; otherwise, it will return false. A timer remains in the expired state until it is restarted.

11.5.2.2 Timer values

The important time periods are summarized in Table 11-1 and are described below. They are defined as a range of possible values to simplify the implementation.

Table 11-1—MAC timer values

Timer value	Used by	Range of values
MAC_Receive_Retrain_Duration	MAC_Timer	1–2 ms
Request_Window_Duration	MAC_Timer	1.33 µs (40 cycles of TxClk)
Retraining_Delay_Duration	MAC_Retraining_Delay_Timer	1–2 s
Training_Final_Idle_or_Incoming_Duration	MAC_Train_Response_Timer	200–400 µs
Training_Frame_Out_to_In_Duration	MAC_Train_Response_Timer	200–400 µs
Training_Up_Duration	MAC_Timer	3–9 µs
TxDisable_Duration	MAC_Timer	200–400 µs

MAC_Receive_Retrain_Duration. This value specifies the maximum time from when the MAC receives Incoming to when the MAC receives either Idle_Down or the last octet of an incoming frame. If neither is received before this timer expires, then an error has occurred.

Request_Window_Duration. This value specifies the minimum period during which a MAC can send a request. This period is required to prevent the situation in which a MAC is continually receiving frames and thus never has the opportunity to send a request before the next frame starts to arrive. The value for Request_Window_Duration is chosen so that the RxEn deassertion time at the MII for a packet followed immediately by Incoming is 41–45 cycles of TxClk.

Retraining_Delay_Duration. This value specifies the minimum period between two retraining attempts. This prevents the MAC from continuously retraining.

Training_Final_Idle_or_Incoming_Duration. This value specifies the maximum period from when the MAC sends Idle_Up, having just successfully exchanged the required number of training frames, to when the MAC expects to receive either Incoming or Idle_Down. If this period is exceeded, the MAC assumes either that the repeater never received the Idle_Up notification, or that the repeater will not accept that training has completed, and that the link must be retrained.

Training_Frame_Out_to_In_Duration. This value specifies the maximum period from when the MAC sends a training frame to the repeater to when the MAC expects to receive a response frame. If this period is exceeded, the MAC assumes that the repeater never received the frame and thus the MAC may retransmit the frame.

Training_Up_Duration. This value specifies the period that the MAC can send Training_Up before it must go silent. The MAC will send a short burst of Training_Up and then go silent for a longer time (specified by TxDisable_Duration). It will repeat this cycle until it observes Training_Down from the repeater.

TxDisable_Duration. This value specifies the period during which the MAC will disable the PMD transmitters to ensure silence between bursts of Training_Up. This is only used when the MAC is attempting to initiate training.

11.5.3 Constants

accessControlSize = 1;

addressSize = 6;

configSize = 2;

crcSize = 4;

dataSize = **...; {Maximum size of LLC data field—see 10.2 and 10.3}**

default_Access_Control_Field = B'00010000'; **{ Default AC field for ISO/IEC 8802-5 frames }**

format_802_3 = B'00'; **{ FF bits in allowed configuration field of training frame }**

format_802_5 = B'10'; **{ FF bits in allowed configuration field of training frame }**

frameControlSize = 1;

lengthSize = 2;

maxTrainingFrames = 48;

maxFrameSize = ...; **{ Maximum size of frame—see 10.2 and 10.3 }**

minFrameSize = ...; **{64 for ISO/IEC 8802-3 frames, 18 for ISO/IEC 8802-5 frames}**

min_802_3_MSDU = 46;

nullMACAddress = ...; **{ The MAC address of X'00 00 00 00 00 00' }**

Number_of_Training_Pairs_Needed = 24; **{ The number of pairs of training frames that must be exchanged between the MAC and the repeater in succession and without error in order for the link to be considered acceptable }**

routingInformationSize = 30;

trainingDataSizeMin = 594;

trainingDataSizeMax = 675;

trainingFrameSizeMin = (2 * addressSize) + (2 * configSize) + trainingDataSizeMin + crcSize;

trainingFrameSizeMax = (2 * addressSize) + (2 * configSize) + trainingDataSizeMax + crcSize;

11.5.4 Types

Bit = (0,1);

Octet = (0,1,...,255);

AccessControlValue = array [1...accessControlSize] of Octet;

AddressValue = array [1...addressSize] of Octet;

AllowedConfigValue = record

```
    Repeater_bit : Bit;
    Promiscuous_bits : array [1...2] of Bit;
    Format_bits : array [1...2] of Bit;
    Unused_bits : array [1...5] of Bit;
    Access_not_allowed_bit : Bit;
    Configuration_bit : Bit;
    Duplicate_address_bit : Bit;
    Version_bits : array [1...3] of Bit;
    End; {AllowedConfigValue}
CRCValue = array [1...crcSize] of Octet;
DataValue = array [1...dataSize] of Octet;
EventSortValue = ( MA_UNITDATA.request,
                   MAC_STATUS_CHANGE,
                   PMI_TRANSMIT.indication,
                   PMI_UNITDATA.indication,
                   RCS_IS_GRANT,
                   RCS_IS_IDLE_DOWN,
                   RCS_IS_ILLEGAL,
                   RCS_IS_INCOMING,
                   RCS_IS_LINK_WARNING,
                   RCS_IS_MODE_TRANSITION,
                   RCS_IS_TRAINING_DOWN,
                   RETRAIN_LINK_EVENT,
                   TIMER_EXPIRES );
FrameControlValue = array [1...frameControlSize] of Octet;
FrameStatusValue = ( valid,
                     error,
                     IPM,
                     nullAddressedFrame,
                     oversizeFrame);
FrameTypeValue = ( Training_Frame,
                   frame_type_802_3,
                   frame_type_802_5 );
LengthValue = array [1...lengthSize] of Octet;
LLCPriorityValue = ...; {Depending on the frame type that is used, this can have the fol-
                        lowing values: ISO/IEC 8802-3 frame format: (normal, high) ISO/IEC
                        8802-5 frame format: (0...7)}
MACPriorityValue = ( NORMAL,
                     HIGH );
PriorityWaitingValue = (NONE,
                        NORMAL,
                        HIGH );
RCSValue = ( Mode_Transition,
             Idle_Down,
             Incoming,
             Training_Down,
             Data_on_Link,
             Link_Warning );
RequestedConfigValue = array [1...configSize] of Octet;
RoutingInformationValue = array [1...routingInformationSize] of Octet;
```

```
StartMACFrameValue = ( valid,
                       invalid );

StateMachineValue = ( MAC1_REQUEST_TRAINING,
                      MAC2_REMOVE_CONTROL_SIGNAL,
                      MAC3_IDLE_OR_REQUEST,
                      MAC4_WAIT_FOR_START_MAC_FRAME,
                      MAC5_READ_FRAME,
                      MAC6_TRANSMIT_FRAME,
                      MAC7_WAIT_FOR_PMI_TRANSMIT_INDICATION,
                      MAC8_ACQUIRE_CONTROL_SIGNAL,
                      MAC9_RETRAIN_DELAY );

StatusValue = ( opened,
                closed,
                opening,
                openFailure,
                linkFailure );

TCSValue = ( TxDisable,
             Idle_Up,
             Request_Normal,
             Request_High,
             Training_Up );

TimerValue = ( MAC_Timer,
               MAC_Retraining_Delay_Timer,
               MAC_Train_Response_Timer );

TrainingDataValue = array [1...trainingDataSizeMax] of Octet;

ViewPoint = ( fields,
              octets );

Frame = record { MAC Frame format }
        Case view : ViewPoint of

            fields : (
                Case frameType: FrameTypeValue of

                    Training_Frame : (
                        Destination_Address_Field : AddressValue;
                        Source_Address_Field : AddressValue;
                        Requested_Config_Field : RequestedConfigValue;
                        Allowed_Config_Field : AllowedConfigValue;
                        Data_Field : TrainingDataValue;
                        Frame_Check_Sequence_Field : CRCValue; )

                    frame_type_802_3 : (
                        Destination_Address_Field : AddressValue;
                        Source_Address_Field : AddressValue;
                        Length_Field : LengthValue;
                        Data_and_Pad_Fields : DataValue;
                        Frame_Check_Sequence_Field : CRCValue; )

                    frame_type_802_5 : (
                        Access_Control_Field : AccessControlValue;
                        Frame_Control_Field : FrameControlValue;
                        Destination_Address_Field : AddressValue;
                        Source_Address_Field : AddressValue;
                        Routing_Information_Field : RoutingInformationValue;
                        LLC_Information_Field : DataValue;
                        Frame_Check_Sequence_Field : CRCValue; )
                )

            octets : (Contents : array [1...maxFrameSize] of Octet)

        End; {Frame}
```

11.5.5 Variables

DO_NOT_WAIT_FOR_LINK_WARNING : Boolean; When this variable is set to true, the MAC will go to MAC1_REQUEST_TRAINING rather than to MAC9_RETRAIN_DELAY when a RE-TRAIN_LINK_
EVENT occurs. This is used when training fails because of a configuration problem and not because the link itself is faulty. The MAC immediately restarts training to prevent management entities from falsely reporting a faulty link.

EVENT_SORT : Event_SortValue; This indicates the current event being considered by the MAC.

FRAME_LENGTH : LengthValue; This indicates the size (in octets) of the frame being received or transmitted.

FRAME_STATUS : FrameStatusValue; Indicates the validity or error status of the received frame.

FRAME_TYPE : FrameTypeValue; This indicates the type of frame the network is using. It is set to Training_Frame during training and then set to either frame_type_802_3 or frame_type_802_5 frames.

HIGHEST_PRIORITY_WAITING : PriorityWaitingValue; This indicates the priority of the highest priority LLC data that is waiting to be transmitted. It is updated by the autonomous process that handles the MA_UNITDATA.request from the LLC and is also updated after a frame has been sent, in the procedure Update_highest_priority() called from state MAC6_TRANSMIT_FRAME.

INCOMING_FRAME : Frame; The frame being received.

JUST_RCVD : Boolean; Set to true after the MAC has received a frame.

JUST_SENT : Boolean; Set to true after the MAC has sent a training frame.

LLC_PRIORITY : LLCPriorityValue; Indicates the priority value requested by the sending LLC.

MAC_PRIORITY : MACPriorityValue; Indicates the priority level (normal or high) of the transmitted packet.

MAC_STATE : StateMachineValue; Indicates the active state of the MAC state machine.

MAC_STATUS : StatusValue; The variable that is continuously examined by the autonomous process Monitor MAC_STATUS. When the value is “opened,” the MAC meta-state is “active.” When this variable is any other value, the MAC is either in training or waiting to be trained.

OUTGOING_FRAME : Frame; The frame being sent.

RCS : RCSValue; The Receiver Control State value reflects the last value of receiver_control_state to be set by a PMI_CONTROL.indication primitive.

RETRAIN_LINK : Boolean; When set to true this indicates that the link should be retrained. This variable is examined by Autonomous Process 2.

SEEN_MODE_TRANSITION : Boolean; Used in MAC7_WAIT_FOR_PMI_TRANSMIT_INDICA-TION to indicate that an RCS value of MODE_TRANSITION has been seen. MODE_TRANSITION must be seen before the MAC can look for a new RCS value.

SOURCE_MAC_ADDRESS : AddressValue; This is the source MAC address used in training. It may be set to the null address for link test only if a unique MAC address has not yet been assigned.

START_MAC_FRAME : StartMACFrameValue; Used to indicate the validity of the ssd at the beginning of a packet.

TCS : TCSValue; The Transmit Control State value is updated whenever the MAC issues a PMI_ CONTROL.request primitive. It is examined in state MAC3_IDLE_OR_REQUEST.

TIMER : TimerValue; This identifies the timer that has expired. If multiple timers expire then it will have the value of each timer in turn until the event is processed (or discarded).

TRAINING_PASSED : Boolean; This flag is set to indicate that the MAC has decided that training has been successful. The MAC makes this decision, not the repeater. The MAC indicates success by sending Idle_Up instead of Training_Up, but cannot enter the “opened” state until the repeater responds with Idle_Down.

TRANSMISSION_COMPLETE : Boolean; Used in state MAC7_WAIT_FOR_PMI_TRANSMIT_ INDICATION to indicate that the packet has been transmitted.

VALID_LENGTH : Boolean; Indicates that the received LLC data size has a valid number of octets. This is only used for ISO/IEC 8802-3 format frames. It is set by the function RemovePad().

11.5.6 Procedures

The procedures used by the MAC state machine are listed in alphabetical order by procedure name.

PROCEDURE Check_if_receive_error().

This procedure is invoked from MAC4_WAIT_FOR_START_MAC_FRAME. The MAC expected to see a frame but before the frame arrived, the MAC either received Idle_Down or the MAC_Timer expired (frame lost).

a) If the MAC_STATUS is opening (training), then this represents a training error.
b) If the MAC_STATUS is opened, then it is only an error if the timer has also expired.

In both cases, the RxEn is deasserted and the state machine updates its current TCS value.

This procedure is called by MAC4 (in two places).

```
Begin
    If (MAC_STATUS = opening) Then
        Train_Error();
    Else {MAC_STATUS is opened.}
```

```
        If (Timer_Expired(MAC_Timer) = True) Then
                    {The MAC_Timer was set to MAC_Receive_Retrain_Duration in state
                        MAC3_IDLE_OR_REQUEST.}
            RETRAIN_LINK := True;

    PMI_RECEIVE_ENABLE.request(deassert); {When the Receiver is deasserted, this will
            cause the PMD to return an RCS of Mode_Transition so the MAC state ma-
            chine must go to reacquire the control signal.}

    Send_Request_Or_Idle();
End
```

PROCEDURE goto_REQUEST_TRAINING().

This procedure calls the procedure Initialize_MAC() to initialize all variables to a known value, and then sets the initial transmit control code in preparation for entering MAC1. The procedure itself is called by MAC3, MAC9, and the procedure Power_On().

```
Begin
    Initialize_MAC();

    PMI_CONTROL.request(Training_Up);
    Start_Timer( MAC_Timer, Training_Up_Duration );
    MAC_STATE := MAC1_REQUEST_TRAINING;
End
```

PROCEDURE Initialize_MAC().

This procedure sets critical variables to a known value. It is called by the procedure goto_Request_ Training().

```
Begin
    FRAME_TYPE := Training_Frame;
    JUST_SENT := False;
    JUST_RCVD := False;
    TRAINING_FRAMES_SENT := 0;
    SUCCESSFUL_PACKET_PAIRS_COUNTER := 0;
    TRAINING_PASSED := False;
    DO_NOT_WAIT_FOR_LINK_WARNING := False
    RETRAIN_LINK := False;
    HIGHEST_PRIORITY_WAITING := NONE;
End
```

PROCEDURE Power_On().

The MAC state machine executes the procedure Power_On() when it first starts or after it is reset.

```
Begin
    MAC_STATUS := linkFailure;
    goto_REQUEST_TRAINING();
End
```

PROCEDURE Process_Received_MAC_Frame(INCOMING_FRAME : Frame,
FRAME_STATUS : FrameStatusValue,
FRAME_LENGTH : LengthValue,
MAC_PRIORITY : MACPriorityValue);

```
DA : AddressValue;
```

```
SA : AddressValue;
ROUTING_INFORMATION: RoutingInformationValue;
MSDU : DataValue;
LLC_PRIORITY : LLCPriorityValue;
```

This procedure takes the appropriate actions based on whether the frame has been received without error and whether training has been completed. FRAME_STATUS may be: valid, IPM, or error (bit corruption). The procedure is called by MAC4 and MAC5 (in two places).

```
Begin { First, consider the case where the MAC is still training.}
If (TRAINING_PASSED = False) Then
  Begin
    With INCOMING_FRAME Do
    Begin
      view := fields;
      frameType := Training_Frame;
      If ((FRAME_STATUS = valid) And
        (JUST_SENT = True) And
        (Frame_Check_Sequence_Field = CRC32(INCOMING_FRAME))
        ) Then
        Train_Success_Frame_Pair(Allowed_Config_Field);
      Else
        Train_Error();

      DA := Destination_Address_Field;
      FRAME_STATUS := nullAddressedFrame;
      UpdateLmeMacRxCounters(FRAME_LENGTH, DA, MAC_PRIORITY, FRAME_STATUS);
           {See 13.2.5.1.3.}
    End
  End

  Else { MAC is Active so check according to frame type. }
  Begin
    With INCOMING_FRAME Do
    Begin
      view := fields;
      frameType := FRAME_TYPE;
      DA := Destination_Address_Field;
      SA := Source_Address_Field;

      If (RecognizeAddress(Destination_Address_Field) = True) Then
      Begin {Disassemble the frame}

        { The order of the following if-then-else is important because nullAddressedFrame
          has precedence over an oversizeFrame. }

        view := octets;
        If ( ( Contents[1] = X'00' ) And ( Contents[2] = X'00' ) And ( Contents[3] = X'00' ) And
            ( Contents[4] = X'00' ) And ( Contents[5] = X'00' ) And ( Contents[6] = X'00' ) ) Then
          FRAME_STATUS := nullAddressedFrame;
        Else
          If (DA = nullMACAddress) Then
            FRAME_STATUS := nullAddressedFrame;
        Else
          If (FRAME_LENGTH > maxFrameSize) Then
            FRAME_STATUS := oversizeFrame;
        Else
          If (FRAME_LENGTH < minFrameSize) Then
            FRAME_STATUS := error;

        view := fields;

        If (FRAME_STATUS = valid) Then
        Begin
          If (Frame_Check_Sequence_Field <> CRC32(INCOMING_FRAME)) Then
            FRAME_STATUS := error;
          Else
          Begin
            Case (FRAME_TYPE) Of

              frame_type_802_3:
              Begin
                MSDU := RemovePad(Length_Field, Data_and_Pad_Fields);
                { The variable VALID_LENGTH is set by the function RemovePad(). }
                If (VALID_LENGTH = True) Then
                Begin
                  LLC_PRIORITY := MAC_PRIORITY;
                  MA_UNITDATA.indication (DA, SA, MSDU, LLC_PRIORITY);
                End
                Else
                  FRAME_STATUS := error;
              End

              frame_type_802_5:
              Begin
                ROUTING_INFORMATION := Routing_Information_Field;
                MSDU := LLC_Information_Field;
                LLC_PRIORITY := { Extract Frame priority from Frame_Control_Field; }
                MA_UNITDATA.indication (DA, SA, ROUTING_INFORMATION, MSDU,
                   LLC_PRIORITY);
              End
            End {Case}
          End
        End
        UpdateLmeMacRxCounters(FRAME_LENGTH, DA, MAC_PRIORITY,
         FRAME_STATUS);      {See 13.2.5.1.3.}
      End
    End
  End
End
```

PROCEDURE Send_Frame (OUTGOING_FRAME:Frame, MAC_PRIORITY : MACPriority-Value).

This procedure sends the OUTGOING_FRAME at priority = MAC_PRIORITY. It is called by MAC6.

```
i : LengthValue;
FRAME_LENGTH : LengthValue;
```

```
Begin
  FRAME_LENGTH := Length(OUTGOING_FRAME);

  With OUTGOING_FRAME Do
  Begin
    view := octets;

    PMI_UNITDATA.request( MAC_PRIORITY, Contents[1], begin );

    for (i := 2; i<FRAME_LENGTH; i := i+1)
    Begin
      PMI_UNITDATA.request( MAC_PRIORITY, Contents[i], more );
    End

    {Now send the last octet.}
    PMI_UNITDATA.request( MAC_PRIORITY, Contents[FRAME_LENGTH], end );
  End

  With OUTGOING_FRAME Do
  Begin
    view := fields;
    UpdateLmeMacTxCounters(FRAME_LENGTH, Destination_Address_Field,
    MAC_PRIORITY); {See 13.2.5.1.3.}
  End
End
```

PROCEDURE Send_Request_Or_Idle().

This procedure sets TCS to Training_Up, Request_Normal, Request_High, or Idle_Up, according to the current state. It is called by MAC3, MAC4 (in two places), MAC5 (in two places), MAC6, MAC7 (in two places), MAC8 (in two places), and the procedure Check_if_receive_error().

```
Begin
  If (TRAINING_PASSED = False) Then
    If (MAC_STATUS <> opening) Then {Haven't started training}
      PMI_CONTROL.request(Training_Up);
    Else {Started training}
    If (JUST_SENT = True) Then
      PMI_CONTROL.request(Training_Up);
    Else
      PMI_CONTROL.request(Request_Normal);
  Else
    Case (HIGHEST_PRIORITY_WAITING) Of
      NONE :   PMI_CONTROL.request(Idle_Up);
      NORMAL : PMI_CONTROL.request(Request_Normal);
      HIGH :   PMI_CONTROL.request(Request_High);
    End {Case}
End
```

PROCEDURE Train_Error().

This procedure detects that an error occurred during the training sequence and resets the necessary counters. If insufficient attempts remain to complete training, set the RETRAIN_LINK variable to true to force retraining to occur. The procedure is called by MAC3, MAC4, and the procedures Check_if_receive_error() and Process_Received_MAC_Frame().

```
Begin
  SUCCESSFUL_PACKET_PAIRS_COUNTER := 0;
  JUST_SENT := False;

   { If insufficient attempts remain then retrain the link.}
  If ((maxTrainingFrames - TRAINING_FRAMES_SENT) < Number_of_Training_Pairs_Needed)
     Then
        Begin
            MAC_STATUS := linkFailure;
            DO_NOT_WAIT_FOR_LINK_WARNING := True;
            RETRAIN_LINK := True;
            Start_Timer(MAC_Retraining_Delay_Timer, Retraining_Delay_Duration);
        End
    Else PMI_CONTROL.request(Request_Normal);
End
```

PROCEDURE Train_Success_Frame_Pair (ALLOWED_CONFIGURATION_FIELD: AllowedConfigValue).

The MAC has successfully exchanged a training frame pair with the repeater. This procedure updates the SUCCESSFUL_PACKET_PAIRS_COUNTER. The configuration field allowed will be checked only after the correct number of frame pairs has been exchanged. This procedure is called by the procedure Process_ Received_MAC_Frame().

```
Begin
    JUST_SENT := False;
    SUCCESSFUL_PACKET_PAIRS_COUNTER :=
        SUCCESSFUL_PACKET_PAIRS_COUNTER + 1;
  If (SUCCESSFUL_PACKET_PAIRS_COUNTER = Number_of_Training_Pairs_Needed) Then
  Begin
    { At this point the physical link is considered acceptable.}

    If (Check_Training_Configuration(ALLOWED_CONFIGURATION_FIELD) = True) Then
    Begin
      TRAINING_PASSED := True;
      PMI_CONTROL.request(Idle_Up);
      Start_Timer(MAC_Train_Response_Timer, Training_Final_Idle_or_Incoming_Duration);
      Case (ALLOWED_CONFIGURATION_FIELD.Format_bits) Of
        format_802_3 : FRAME_TYPE := frame_type_802_3;
        format_802_5 : FRAME_TYPE := frame_type_802_5;
        default : { error—illegal value } ;
      End {Case}

    End
    Else { Configuration not acceptable, so must retrain.}
    Begin
      MAC_STATUS := openFailure;
      SUCCESSFUL_PACKET_PAIRS_COUNTER := 0;
      JUST_SENT := False;
      RETRAIN_LINK := True;
      DO_NOT_WAIT_FOR_LINK_WARNING := True;
      Start_Timer(MAC_Retraining_Delay_Timer, Retraining_Delay_Duration);
    End
  End
  Else { MAC has not yet exchanged enough correct frame pairs.}
    PMI_CONTROL.request(Request_Normal);
End
```

PROCEDURE Update_highest_priority().

This procedure scans all waiting LLC_UNITDATA.requests to determine the highest priority level. It is called by MAC6.

```
Begin
  HIGHEST_PRIORITY_WAITING := {Highest priority frame that is waiting to be
    transmitted}
End
```

11.5.7 Functions

The functions used by the MAC state machine are listed in alphabetical order, by function name.

FUNCTION Build_Frame() : Frame; LOCAL_FRAME : Frame;

This function returns a frame of the appropriate type, according to the variable FRAME_TYPE. Build_Frame() is called by MAC6.

a) A training frame can be constructed directly.

b) If FRAME_TYPE indicates an ISO/IEC 8802-3 format frame or an ISO/IEC 8802-5 format frame, then these must be constructed from the parameters passed by the LLC in the MA_UNITDATA.request primitive associated with the highest priority frame that is waiting to be transmitted. The parameters of MA_UNITDATA.request are:

    1) DA : AddressValue
    2) SA : AddressValue
    3) ROUTING_INFORMATION: RoutingInformationValue
    4) MSDU : DataValue
    5) LLC_PRIORITY : LLCPriorityValue

```
Begin
  With LOCAL_FRAME Do
  Begin
    view := fields;
    frameType := FRAME_TYPE;
    Case (FRAME_TYPE) Of

      Training_Frame : { See 10.6. }
         Begin  { Set the destination address to zero. }
            Destination_Address_Field := nullMACAddress;

                { The source address is the Individual Address of this MAC.}
            Source_Address_Field := SOURCE_MAC_ADDRESS;

                { Set the training configuration requested. }
            Requested_Config_Field := ...; { See 10.6.2. }
            Allowed_Config_Field := X'00 00';

                { The remainder of the training frame should be set as specified in 10.6. }
            Data_Field := ...;

                { Generate CRC over the frame. }
            Frame_Check_Sequence_Field := CRC32(LOCAL_FRAME);
        End

      frame_type_802_3 : { See 10.2. }
         Begin
            Destination_Address_Field  := DA;
            Source_Address_Field       := SA;
            Length_Field               := Length(MSDU);
            Data_and_Pad_Fields := ComputePad(MSDU);
            Frame_Check_Sequence_Field := CRC32(LOCAL_FRAME);
         End

      frame_type_802_5 :   { See 10.3. }
        Begin
            Access_Control_Field := defaultAccessControlField;
            Frame_Control_Field := ...; { Set according to the rules in 10.3.2.
                                          Includes the priority of the frame. }
            Destination_Address_Field := DA;
            Source_Address_Field := SA; {Also set the RII if an RI field is present.}
            Routing_Information_Field := ROUTING_INFORMATION;
            LLC_Information_Field := MSDU;
            Frame_Check_Sequence_Field := CRC32(LOCAL_FRAME);
        End

    End {Case}
    End
  Build_Frame := LOCAL_FRAME;
End
```

FUNCTION Check_Training_Configuration (ALLOWED_CONFIGURATION_FIELD : AllowedConfigValue) : Boolean;

This function determines whether the bits in the allowed configuration field of the training frame response are compatible with the requested configuration. It is called by the procedure Train_Success_Frame_Pair().

```
Begin
    { Check that the configuration information in the response training frame is compatible with what was requested. See 10.6 for details of the training frame format. Function
    returns true if the configuration was accepted by the repeater. }

  With ALLOWED_CONFIGURATION_FIELD Do
    Check_Training_Configuration :=
      ( (Access_not_allowed_bit = 0) And (Configuration_bit = 0) )
End
```

FUNCTION ComputePad(MSDU): DataValue;

If the MSDU is less than min_802_3_MSDU octets, then it must be padded to a length of min_802_3_MSDU octets. Compute_Pad() is called by the function Build_Frame().

```
Begin
  If (Length(MSDU) < min_802_3_MSDU) Then
    ComputePad := { Append pad to MSDU such that Length(MSDU + Pad) is
                    min_802_3_MSDU.}
End
```

FUNCTION CRC32(MAC_Frame : Frame): CRCValue;

This function returns the CRC32 calculated over the MAC frame. The calculation covers different parts of the MAC frame according to frame type.

a) The ISO/IEC 8802-3-format CRC32 coverage is described in 10.2.4.
b) The ISO/IEC 8802-5 format CRC32 coverage is described in 10.3.6.
c) The ISO/IEC 8802-12 training-format CRC32 coverage is described in 10.6.5.

CRC32() is called by the procedure Process_Received_MAC_Frame() and the function Build_Frame().

```
Begin
   CRC32 := { The 32-bit CRC }
End
```

FUNCTION RecognizeAddress(Address : AddressValue) : Boolean;

This function checks the Destination Address of the received frame. It is called by the procedure Process_ Received_MAC_Frame().

```
Begin

  RecognizeAddress := { Returns true for the set of individual, broadcast, and group
                        addresses corresponding to this end node. For management
                        purposes this function also returns true for an address that is the
                        null MAC address when the end node is in training. }
End
```

FUNCTION RemovePad(Length : LengthValue; Data_and_Pad : DataValue) : DataValue;

This function compares the value of the Length_Field to the number of octets in the Data_and_Pad_Fields. It is called by the procedure Process_Received_MAC_Frame().

```
Begin
  VALID_LENGTH := { Check that the value represented by Length matches the
                    received LLC data size in Data_and_Pad. }
  If (VALID_LENGTH = True) Then
    RemovePad := { Truncate Data_and_Pad to the value given in Length. }
  Else
    RemovePad := Data_and_Pad;
End
```

11.6 MAC state machine state definitions

The state machine definitions are listed in order of their state number.

11.6.1 MAC1 : MAC1_REQUEST_TRAINING

While waiting to start training, the MAC alternates between sending short bursts of Training_Up and longer bursts of no energy (this pattern limits emissions if a link is not properly terminated). In this state the MAC is sending Training_Up. The MAC exits this state if:

a) The MAC_Timer expires, indicating that the MAC should stop sending Training_Up and go to state MAC2_REMOVE_CONTROL_SIGNAL.
b) The MAC receives Training_Down from the repeater AND the MAC_Retraining_Delay_Timer has expired. This indicates that the MAC can start to train the link, so it goes to the state MAC3_IDLE_OR_REQUEST and waits for permission to send a frame.

See Figure 11-9.

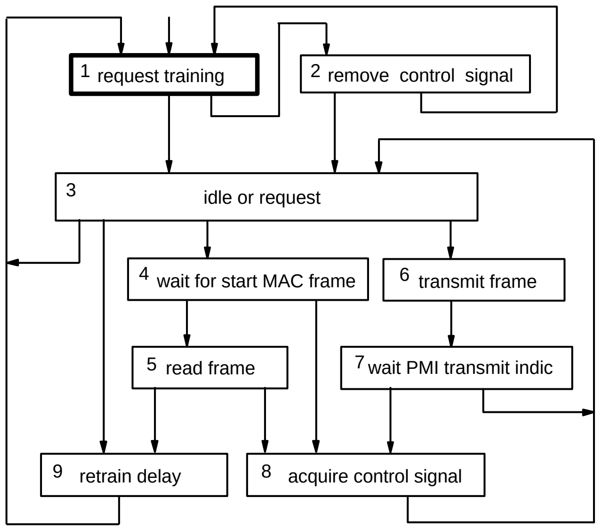


Figure 11-9—The relationship of MAC1 to the state machine

```
Case (EVENT_SORT) Of

  TIMER_EXPIRES :
  Begin
    If (TIMER = MAC_Timer) Then
    { MAC_Timer was set to Training_Up_Duration in either state
      MAC2_REMOVE_CONTROL_SIGNAL or PROCEDURE goto_REQUEST_TRAINING(). }
    Begin
      PMI_CONTROL.request(TxDisable);
      Start_Timer( MAC_Timer, TxDisable_Duration );
      MAC_STATE := MAC2_REMOVE_CONTROL_SIGNAL;
    End
  End

  RCS_IS_TRAINING_DOWN :
  Begin
```

```
    If ((Timer_Expired(MAC_Retraining_Delay_Timer) = True) And (MAC_STATUS <> closed)
      ) Then
    { MAC_Retraining_Delay_Timer was set to Retraining_Delay_Duration in PROCEDURE
      Train_Error(), in PROCEDURE Train_Success_Frame_Pair(), or in state
       MAC3_IDLE_OR_REQUEST. }
    Begin
      MAC_STATUS := opening;
      PMI_CONTROL.request(Request_Normal);
      Start_Timer( MAC_Timer, Request_Window_Duration );
      MAC_STATE := MAC3_IDLE_OR_REQUEST;
    End
  End
End {Case}
```

11.6.2 MAC2 : MAC2_REMOVE_CONTROL_SIGNAL

While waiting to start training, the MAC alternates between sending short bursts of Training_Up and longer bursts of no energy. In this state the MAC is sending no energy. The MAC exits this state if:

a) The MAC_Timer expires, indicating that the MAC should start sending another burst of Training_Up and go to state MAC1_REQUEST_TRAINING.

b) The MAC receives Training_Down from the repeater AND the MAC_Retraining_Delay_Timer has expired. This indicates that the MAC can start to train the link, so it goes to the state MAC3_IDLE_OR_REQUEST and waits for permission to send a frame.

See Figure 11-10.

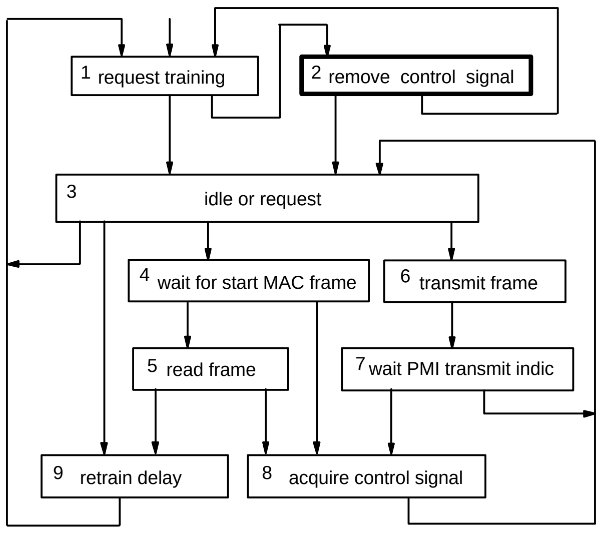


Figure 11-10—The relationship of MAC2 to the state machine

```
Case (EVENT_SORT) Of

  TIMER_EXPIRES, MAC_STATUS_CHANGE :
  Begin
    If ((TIMER = MAC_Timer) And (MAC_STATUS <> closed)) Then
    { MAC_Timer was set to TxDisable_Duration in state MAC1_REQUEST_TRAINING. }
    Begin
```

```
      PMI_CONTROL.request(Training_Up);
      Start_Timer( MAC_Timer, Training_Up_Duration );
      MAC_STATE := MAC1_REQUEST_TRAINING;
    End
  End

  RCS_IS_TRAINING_DOWN :
  Begin
    If (( Timer_Expired(MAC_Retraining_Delay_Timer) = True) And
      (MAC_STATUS <> closed) ) Then
    { MAC_Retraining_Delay_Timer was set to Retraining_Delay_Duration in PROCEDURE
      Train_Error(), in PROCEDURE Train_Success_Frame_Pair(), or in state
      MAC3_IDLE_OR_REQUEST. }
    Begin
      MAC_STATUS := opening;
      PMI_CONTROL.request(Request_Normal);
      Start_Timer( MAC_Timer, Request_Window_Duration );
      MAC_STATE := MAC3_IDLE_OR_REQUEST;
    End
  End
End {Case}
```

11.6.3 MAC3 : MAC3_IDLE_OR_REQUEST

This is the state in which the MAC waits between sending or receiving frames. Important events in this state are:

a) The MAC is granted permission to send a frame AND the MAC is requesting to send a frame. In this case, the MAC goes to state MAC6_TRANSMIT_FRAME.

b) The Timer MAC_Train_Response_Timer expires AND the MAC is still in the training process. This indicates that the repeater did not receive the last frame, so the MAC should retransmit the training frame. The training error is handled by the procedure Train_Error().

c) The MAC receives Incoming. In this case the MAC then goes to state MAC4_WAIT_FOR_START_MAC_FRAME to wait for the first octet of the expected frame (or for Idle_Down). If the MAC has just received a frame then it will also wait until the MAC_Timer has expired to ensure that it has issued its own Request for the period specified by the Request_Window_Duration.

d) The MAC receives either Incoming or Idle_Down AND the MAC has just successfully finished the training process. This indicates that the repeater is treating this link as opened.

e) An error has occurred. In this case the MAC will initiate training after some period of time.

f) The MAC receives an MA_UNITDATA.request from the LLC. The MAC may need to update the control signal that it is currently sending to the repeater.

See Figure 11-11.

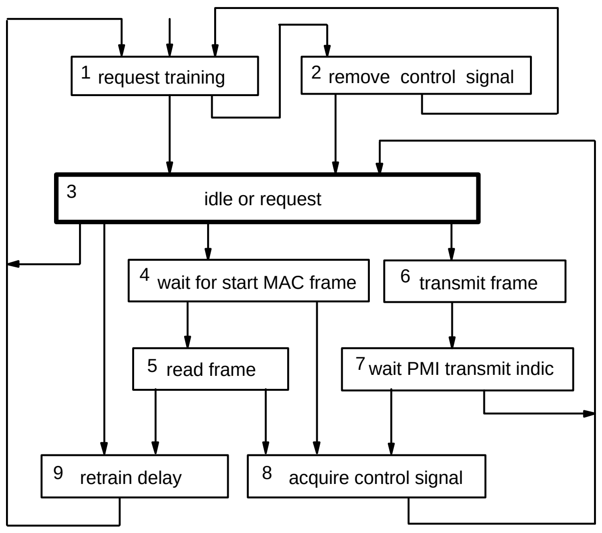


Figure 11-11—The relationship of MAC3 to the state machine

```
Case (EVENT_SORT) Of

  { The MAC is given permission to send a frame. }

  RCS_IS_GRANT :
  Begin

    If ( (TCS = Request_Normal) Or (TCS = Request_High) ) Then
    Begin
      JUST_RCVD := False;
      MAC_STATE := MAC6_TRANSMIT_FRAME;
    End
  End

    {The next case handles two situations
    a)  The first is when the end node has sent a Training Frame but it has not received a
        response frame before the MAC_Train_Response_Timer has expired OR if the end
        node has just passed training, has sent Idle_Up, and is waiting to see either
        Idle_Down or Incoming.
    b)  The second is the common case where Incoming is received. In this case, if the
        link has just received a frame, it must wait for the request window period before
        enabling the receiver. The request window is timed by the MAC_Timer and so the
        state machine must check for Incoming when the MAC_Timer expires.}

  TIMER_EXPIRES, RCS_IS_INCOMING :
  Begin

    If (TIMER = MAC_Train_Response_Timer) Then
        { The MAC_Train_Response_Timer was set to either
         Training_Frame_Out_to_In_Duration in MAC6_TRANSMIT_FRAME
         or Training_Final_Idle_or_Incoming_Duration in
         PROCEDURE Train_Success_Frame_Pair(). }
    Begin
      If ((MAC_STATUS = opening) And (JUST_SENT = True)) Then
        Train_Error();
    End

    Else
    Begin
      If ( RCS = Incoming ) Then
      Begin
        { If MAC has passed training but is not opened and sees Incoming, then this is as
          good as seeing Idle_Down—i.e., the MAC is now opened. }
        If ( ( MAC_STATUS = opening) And ( TRAINING_PASSED = True) ) Then
        Begin
          MAC_STATUS := opened;
          Start_Timer(MAC_Retraining_Delay_Timer, Retraining_Delay_Duration);
        End

        { Check to see if Request_Window is relevant and, if so, whether it has elapsed. }

        If ( (JUST_RCVD = False) Or (Timer_Expired(MAC_Timer) = True) ) Then

        { The MAC_Timer was set to Request_Window_Duration in
             state MAC1_REQUEST_TRAINING,
             state MAC2_REMOVE_CONTROL_SIGNAL,
             state MAC4_WAIT_FOR_START_MAC_FRAME, or
             state MAC5_READ_FRAME (two places). }

        Begin
          JUST_RCVD := False;
          PMI_RECEIVE_ENABLE.request(assert);
          Start_Timer( MAC_Timer, MAC_Receive_Retrain_Duration );
          MAC_STATE := MAC4_WAIT_FOR_START_MAC_FRAME;
        End
      End
    End
  End
```

{ The remaining cases cope with state changes and error conditions. }
{ If the MAC has completed training, then if it observes Idle_Down from the repeater, then the MAC can become Active. }

```
  RCS_IS_IDLE_DOWN :
  Begin
    If ( ( MAC_STATUS = opening) And ( TRAINING_PASSED = True ) ) Then
    Begin
      MAC_STATUS := opened;
      Start_Timer(MAC_Retraining_Delay_Timer, Retraining_Delay_Duration);
    End
  End
```

{ If the MAC sees:
- **a) Training_Down and is no longer training,**
- **b) Link error, or**
- **c) RETRAIN_LINK go true,**

```
then this is an error and the link shall be retrained.}

RCS_IS_TRAINING_DOWN :
  Begin
    If ( MAC_STATUS = opened) Then
    Begin
      MAC_STATUS := linkFailure;
      PMI_CONTROL.request(TxDisable);
      MAC_STATE := MAC9_RETRAIN_DELAY;
    End
  End

  RCS_IS_LINK_WARNING, RCS_IS_ILLEGAL :
  Begin
    PMI_CONTROL.request(TxDisable);
    MAC_STATUS := linkFailure;
    MAC_STATE := MAC9_RETRAIN_DELAY;
  End

  RETRAIN_LINK_EVENT :
  Begin
    If (DO_NOT_WAIT_FOR_LINK_WARNING = True) Then
      goto_REQUEST_TRAINING(); {goto_REQUEST_TRAINING provides the state
         transition path to MAC1_REQUEST_TRAINING.}
    Else
    Begin
      PMI_CONTROL.request(TxDisable);
      MAC_STATE := MAC9_RETRAIN_DELAY;
    End
  End

  MA_UNITDATA.request : Send_Request_Or_Idle();

End {Case}
```

11.6.4 MAC4 : MAC4_WAIT_FOR_START_MAC_FRAME

The MAC has received Incoming and now expects to receive either:

a) PMI_UNITDATA.indication—In this case the end node will expect to receive a frame.

b) IDLE_DOWN—If the MAC is opened (active mode) and the current frame is not for this MAC, the MAC will receive Idle_Down. The MAC will then turn off the receiver and go to state MAC8_ACQUIRE_CONTROL_SIGNAL.

If the MAC_Timer expires, then an error may have occurred. This is checked in the procedure Check_if_receive_error(), and then the MAC goes to state MAC8_ACQUIRE_CONTROL_ SIGNAL.

See Figure 11-12.

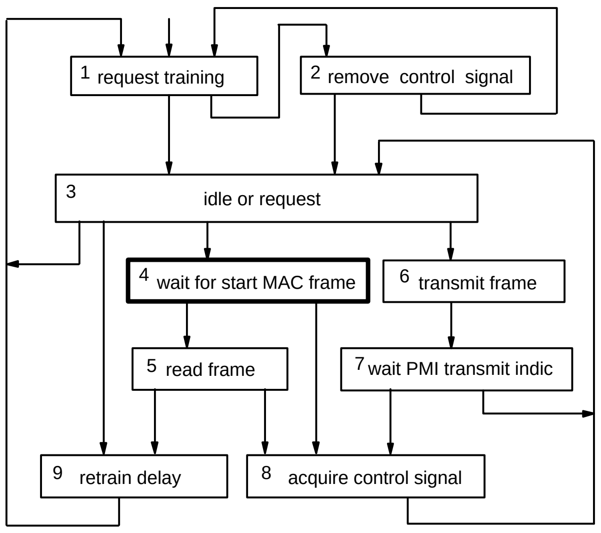


Figure 11-12—The relationship of MAC4 to the state machine

```
Case (EVENT_SORT) Of

  PMI_UNITDATA.indication( DATA_OCTET,
         MAC_PRIORITY,
         START_MAC_FRAME,
         MAC_FRAME,
         ERROR_STATUS) :

  Begin
    FRAME_LENGTH := 1; { 1 octet received so far }

    { The first case to consider is when the value of MAC_FRAME is “end,” indicating that
    the frame was not received correctly.}

    If (MAC_FRAME = end) Then
    Begin
      Process_Received_MAC_Frame(INCOMING_FRAME, error, FRAME_LENGTH,
    MAC_PRIORITY);
      Start_Timer( MAC_Timer, Request_Window_Duration );
      Send_Request_Or_Idle();
      MAC_STATE := MAC8_ACQUIRE_CONTROL_SIGNAL;
    End

    { The second case is when frame data has been received.}

    Else
    Begin
      If (START_MAC_FRAME = valid) Then
        FRAME_STATUS := valid;
      Else
        FRAME_STATUS := error;

      With INCOMING_FRAME Do
      Begin
```

```
        view := octets;
        Contents[1] := DATA_OCTET;
      End
      JUST_RCVD := True;

      **{ Even if START_MAC_FRAME was invalid, the MAC must still read the frame in
order
    to detect the End of frame.}**

      MAC_STATE := MAC5_READ_FRAME;
    End
  End

  RCS_IS_TRAINING_DOWN:
  Begin
    If (MAC_STATUS = opening) Then
    Begin
      If (JUST_SENT = True) Then
        Train_Error();
      Else
      Begin
        PMI_RECEIVE_ENABLE.request(deassert);
        Send_Request_Or_Idle();
      End
      MAC_STATE := MAC8_ACQUIRE_CONTROL_SIGNAL;
    End
  End

  RCS_IS_IDLE_DOWN :
  Begin
    Check_if_receive_error();
    MAC_STATE := MAC8_ACQUIRE_CONTROL_SIGNAL;
  End

  TIMER_EXPIRES :
  Begin
    If (TIMER = MAC_Timer) Then
    { The MAC_Timer was set to MAC_Receive_Retrain_Duration
      in state MAC3_IDLE_OR_REQUEST. }
    Begin
      Check_if_receive_error();
      MAC_STATE := MAC8_ACQUIRE_CONTROL_SIGNAL;
    End
  End
End {Case}
```

11.6.5 MAC5 : MAC5_READ_FRAME

At this point the MAC has successfully received the first octet. The MAC continues to receive octets until:

a) The last octet is passed up, in which case the frame is checked and the MAC goes to state MAC8_ACQUIRE_CONTROL_SIGNAL.

b) The MAC_Timer expires (in which case the MAC deasserts the receiver and retrains the link).[23]

c) The PMI indicates that it is now receiving a control signal rather than data, in which case an error has occurred. The MAC then indicates an error and goes to state MAC8_ACQUIRE_CONTROL_SIGNAL.

See Figure 11-13.

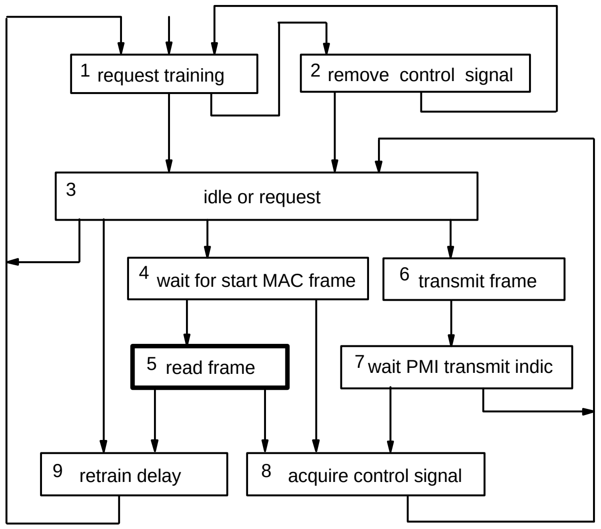


Figure 11-13—The relationship of MAC5 to the state machine

```
Case (EVENT_SORT) Of

PMI_UNITDATA.indication( DATA_OCTET,
         MAC_PRIORITY,
         START_MAC_FRAME,
         MAC_FRAME,
         ERROR_STATUS) :

  Begin
    With INCOMING_FRAME Do
    Begin
      view := octets;
      FRAME_LENGTH := FRAME_LENGTH + 1;
      Contents[FRAME_LENGTH] := DATA_OCTET;
    End

    If (MAC_FRAME = End) Then
    Begin

      If (FRAME_STATUS = valid) Then
        FRAME_STATUS := ERROR_STATUS;
      Process_Received_MAC_Frame(INCOMING_FRAME, FRAME_STATUS,
        FRAME_LENGTH, MAC_PRIORITY);
      Send_Request_Or_Idle();
```

[23]A time-out could occur if the frame never arrived, if the frame started too late, or if the frame was much too long.

```
      Start_Timer( MAC_Timer, Request_Window_Duration );
      MAC_STATE := MAC8_ACQUIRE_CONTROL_SIGNAL;
    End
  End

  TIMER_EXPIRES :
  Begin
    If (TIMER = MAC_Timer) Then
    { The MAC_Timer was set to MAC_Receive_Retrain_Duration
      in state MAC3_IDLE_OR_REQUEST. }

    Begin
      PMI_RECEIVE_ENABLE.request(deassert);
      PMI_CONTROL.request(TxDisable);
      MAC_STATUS := linkFailure;
      MAC_STATE := MAC9_RETRAIN_DELAY;
    End
  End

  RCS_IS_TRAINING_DOWN, RCS_IS_LINK_WARNING, RCS_IS_ILLEGAL,
    RCS_IS_IDLE_DOWN:
  Begin
    Process_Received_MAC_Frame(INCOMING_FRAME, error, FRAME_LENGTH,
       MAC_PRIORITY);
    Send_Request_Or_Idle();
    Start_Timer( MAC_Timer, Request_Window_Duration );
    MAC_STATE := MAC8_ACQUIRE_CONTROL_SIGNAL;
  End
End {Case}
```

11.6.6 MAC6 : MAC6_TRANSMIT_FRAME

The MAC has permission to send a frame. It constructs the frame according to the mode it is in (training or opened) and then transmits it. The MAC then goes to the state MAC7_WAIT_FOR_PMI_ TRANSMIT_INDICATION to wait for a signal from the PMI to indicate that the frame has been transmitted. See Figure 11-14.

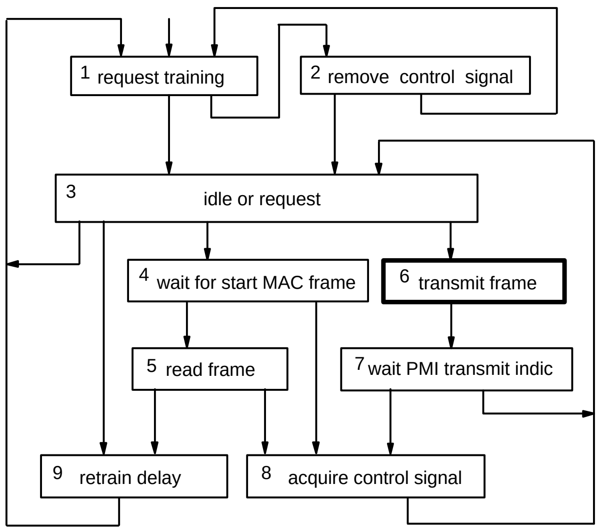


Figure 11-14—The relationship of MAC6 to the state machine

```
OUTGOING_FRAME := Build_Frame();
  Case (FRAME_TYPE) Of
    Training_Frame:
    Begin
      MAC_PRIORITY := NORMAL;
    End
    frame_type_802_3, frame_type_802_5:
    Begin
      MAC_PRIORITY := HIGHEST_PRIORITY_WAITING;
    End
  End {Case}

  Send_Frame(OUTGOING_FRAME, MAC_PRIORITY);

  { Update HIGHEST_PRIORITY_WAITING to reflect any outstanding frames. }

  Update_highest_priority();

  { Now update training counters, if necessary. }

  If (MAC_STATUS = opening) Then
  Begin
    JUST_SENT := True;
    TRAINING_FRAMES_SENT := TRAINING_FRAMES_SENT + 1;
    Start_Timer(MAC_Train_Response_Timer, Training_Frame_Out_to_In_Duration);
  End

  Send_Request_Or_Idle();
   { Clear SEEN_MODE_TRANSITION and TRANSMISSION_COMPLETE before
    going to MAC7_WAIT_FOR_PMI_TRANSMIT_INDICATION }

  SEEN_MODE_TRANSITION := False;
  TRANSMISSION_COMPLETE := False;
   MAC_STATE := MAC7_WAIT_FOR_PMI_TRANSMIT_INDICATION;
```

11.6.7 MAC7 : MAC7_WAIT_FOR_PMI_TRANSMIT_INDICATION

The MAC has passed the last octet of a frame to the PMI for transmission. It waits in this state until either:

a) The PMI indicates that the frame has been transmitted, in which case it goes to MAC8_ ACQUIRE_CONTROL_SIGNAL, or

b) The PMI indicates a possible fault with the link, in which case the MAC goes to state MAC3_ IDLE_OR_REQUEST to process the fault.

See Figure 11-15.

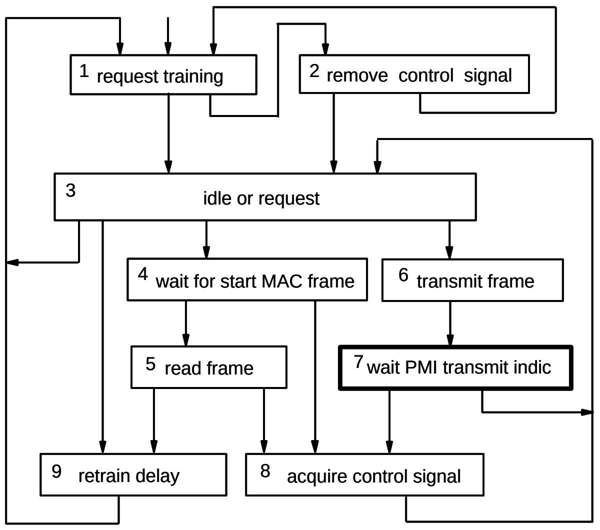


Figure 11-15—The relationship of MAC7 to the state machine

```
Case (EVENT_SORT) Of

PMI_TRANSMIT.indication(transmit_complete) :
  Begin
    If (transmit_complete = End) Then
    Begin
      TRANSMISSION_COMPLETE := True;
      If (SEEN_MODE_TRANSITION = True) Then
      Begin
        Send_Request_Or_Idle();
        MAC_STATE := MAC8_ACQUIRE_CONTROL_SIGNAL;
      End
    End
  End

 RCS_IS_MODE_TRANSITION :
  Begin
    SEEN_MODE_TRANSITION := True;
    If (TRANSMISSION_COMPLETE = True) Then
    Begin
      Send_Request_Or_Idle();
      MAC_STATE := MAC8_ACQUIRE_CONTROL_SIGNAL;
    End
  End

RCS_IS_LINK_WARNING, RCS_IS_ILLEGAL :
  Begin
    MAC_STATE := MAC3_IDLE_OR_REQUEST;
  End
End {Case}
```

11.6.8 MAC8 : MAC8_ACQUIRE_CONTROL_SIGNAL

The MAC enters this state after it has just transmitted or received a frame. The objective is to remain in this state until a valid receive control signal has been recovered. Grant is not treated as a valid code unless the MAC has just received a frame because Grant will always be asserted following a transmission for certain PMDs.

The MAC should ignore the control signals Mode_Transition and Data_on_Link. See Figure 11-16.

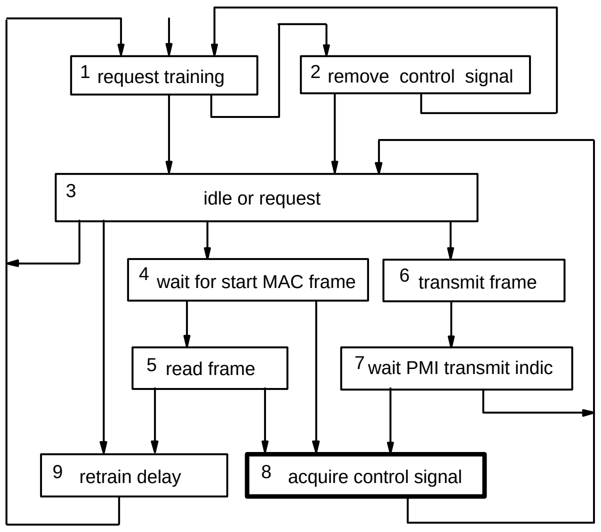


Figure 11-16—The relationship of MAC8 to the state machine

```
Case (EVENT_SORT) Of

  RCS_IS_TRAINING_DOWN, RCS_IS_INCOMING,
  RCS_IS_IDLE_DOWN :
  Begin
    Send_Request_Or_Idle();
    MAC_STATE := MAC3_IDLE_OR_REQUEST;
  End

  RCS_IS_GRANT :
  Begin
    If (JUST_RCVD = True) Then
    Begin
      Send_Request_Or_Idle();
      MAC_STATE := MAC3_IDLE_OR_REQUEST;
    End
  End

  RCS_IS_LINK_WARNING, RCS_IS_ILLEGAL :
  Begin
    MAC_STATE := MAC3_IDLE_OR_REQUEST;
  End
End {Case}
```

11.6.9 MAC9 : MAC9_RETRAIN_DELAY

State MAC9_RETRAIN_DELAY is entered only if DO_NOT_WAIT_FOR_LINK_WARNING is false. If DO_NOT_WAIT_FOR_LINK_WARNING is true when a RETRAIN_LINK_EVENT occurs, the MAC will go directly to MAC1_REQUEST_TRAINING.

The MAC state machine only stays in state MAC9_RETRAIN_DELAY until it observes an RCS value of Link_Warning. It then goes to MAC1_REQUEST_TRAINING. See Figure 11-17.

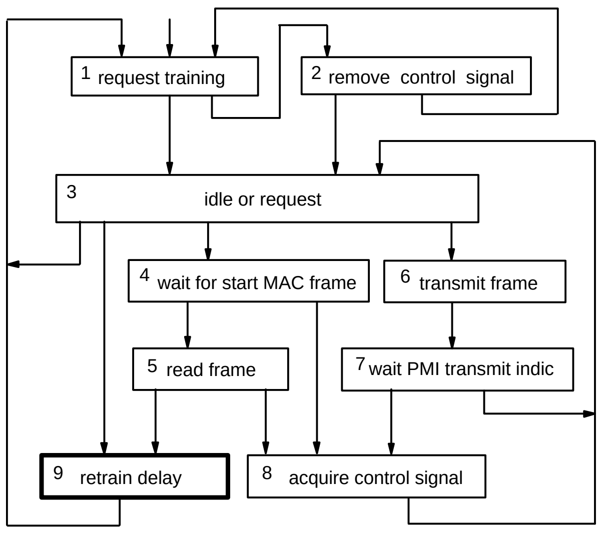


Figure 11-17—The relationship of MAC9 to the state machine

```
Case (EVENT_SORT) Of

  MAC_STATUS_CHANGE, RCS_IS_LINK_WARNING :
  Begin
    If ( (MAC_STATUS <> closed) And
        (RCS = Link_Warning) ) Then
    Begin
      goto_REQUEST_TRAINING();
    End
  End
End {Case}
```

12. Repeater Medium Access Control (RMAC) protocol

12.1 Scope

The RMAC is the primary control layer in the repeater (see Figure 12-1). This clause describes the packet monitoring and network control functions of the RMAC sublayer, assuming the logical model of a separate PMI/PMD pair for each repeater port. The descriptions are provided as state diagrams, supplemented by prose annotations, with state definitions in pseudo code. All are normative. Should differences be perceived in interpretation, pseudo code definitions shall take precedence. No particular physical implementation is implied.

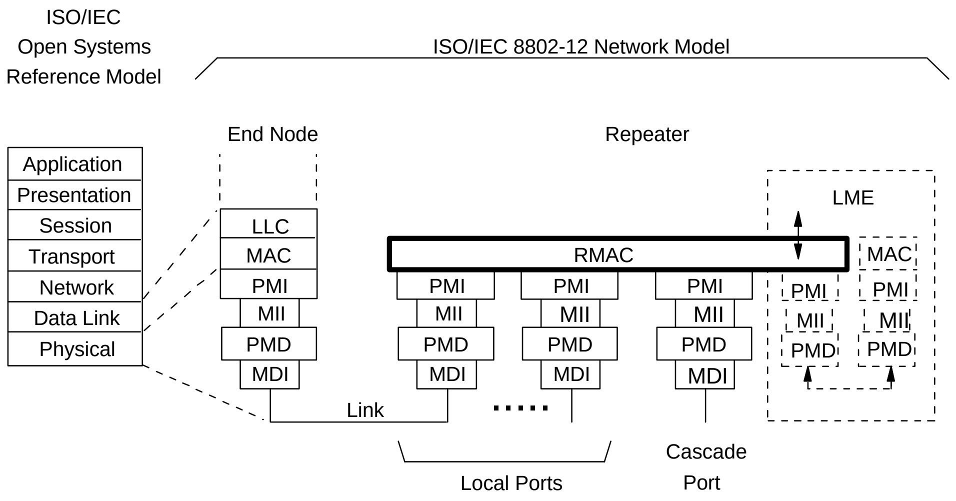


Figure 12-1—Relationship to the LAN model

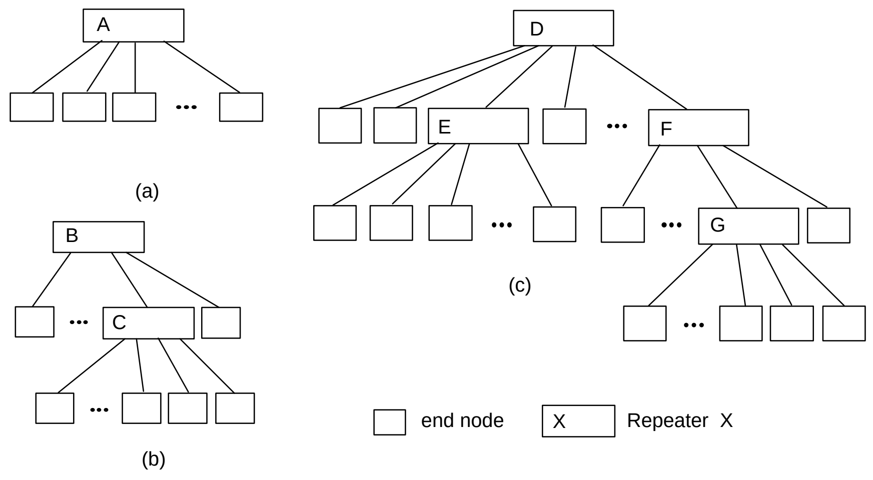
12.1.1 Network topology restrictions

Repeaters may be equipped with any number of local ports for connection to end nodes and lower repeaters, plus one optional cascade port with either one or two uplinks for connection to higher-level repeaters. Each local port may be equipped with only one downlink. Repeaters may be configured to support from one to four RMAC operational capability levels, according to the repeater's potential locations in a network as shown by the examples in Figure 12-2.

a) Repeaters that support use in single repeater networks with local port connections to end nodes only [repeater A in Figure 12-2a)]. This configuration requires local ports only and is the mandatory minimum RMAC capability level.
b) Repeaters that support use as the top repeater in a cascade [repeater B in Figure 12-2b), repeater D in Figure 12-2c)]. This configuration requires local ports and an RMAC with optional support for connecting lower-level repeaters in cascade.
c) Repeaters that support use as the lowest level repeater in a cascade [repeater C in Figure 12-2b); repeaters E and G in Figure 12-2c)]. This configuration requires an RMAC with the optional cascade port and support for cascaded operations on the uplink(s).
d) Repeaters that support use as an intermediate repeater in a cascade [repeater F in Figure 12-2c)]. This configuration requires an RMAC with the optional cascade port and support for cascaded operations on both uplinks and downlinks.

Indication of which network structures the repeater is able to support is required for link initiation. The method used to store and indicate RMAC support capabilities is an implementation issue and is beyond the scope of this standard.

The type of link is defined by the optional PMI_CONFIGURATION.indication primitive generated by the PMI associated with each port. The RMAC has been defined to be insensitive to any differences in PMD/link configuration, except where 4-UTP links to end nodes are installed in bundled cable. Group-addressed packets received from an end node on a bundled link must be transmitted in store-and-forward mode to prevent the possibility of excessive crosstalk in the bundled cable. The repeater connected to an end node that is sending a group-addressed packet in this circumstance will delay transmission of that packet until it has been fully received.



Note—Examples of networks with redundant links are provided in Annex E

Figure 12-2—Example network structures

The store-and-forward requirement does not apply to packets with individual addresses sent from end nodes connected by links installed in bundled cable. It also does not apply to subsequent transmissions by repeaters at other locations in the network, since repeater-to-repeater links are not allowed in bundled cable. Those repeaters may retransmit the packet in the normal-overlapped mode. Frame timing considerations for both normal-overlapped and store-and-forward transmission are detailed in 12.9.7.2.

Indication of bundled cable links is required for proper operation of the RMAC. The manner in which this indication is made is an implementation issue and is beyond the scope of this standard.

12.2 RMAC packet monitoring and network control overview

The RMAC sublayer provides the primary control over network access and packet sequencing. It accepts transmission requests and arbitrates their transmission order through the application of a priority-based round-robin port-selection algorithm where:

a) Transmission sequences follow network port order rather than time-of-request arrival.
b) No sending station is allowed to follow itself if other requests of equal or higher priority are pending at the repeater.
c) High-priority requests are serviced (in port order) first.

d) Packet promotion timers ensure that normal requests are not blocked by abnormally high and prolonged high-priority traffic levels.

Once port selection has been made, the RMAC sends Grant to the lower entity connected to the selected port and instructs the physical sublayers associated with that port to switch to the receive mode.

On reception of the packet:

- The destination address is decoded and compared to the port address list as the packet is being received. Incoming is sent to all active ports (except the selected port, where the packet is being received) to alert the connected entities to prepare to receive an incoming packet.
- Transmission of the frame is normally begun before the entire frame has been received, unless the incoming packet is being received over a 4-UTP link in bundled cable and the destination address of the packet is a group address.
- The incoming frame is checked for FCS violations and error_status. Errors detected in the received packet are indicated by IPMs in the repeated (retransmitted) packets.

The RMAC will interrupt its normal-priority round-robin port-selection cycle to service (in port order) any high-priority requests received. If the network contains more than one repeater, control of port selection is rotated among the repeaters, resulting in a distributed round-robin port-selection cycle across the entire network.

12.2.1 Port meta-states

The cascade port (if implemented) and each local port on the repeater shall be treated as separate entities, with separate traffic requirements and separate restrictions. The RMAC initializes the cascade port and each local port at repeater power-up.

12.2.1.1 Cascade port/uplink operational modes for repeaters with a single uplink

The cascade port in repeaters equipped with a single uplink can be viewed as being in either of the two major meta-states indicated by the dotted lines in Figure 12-3.

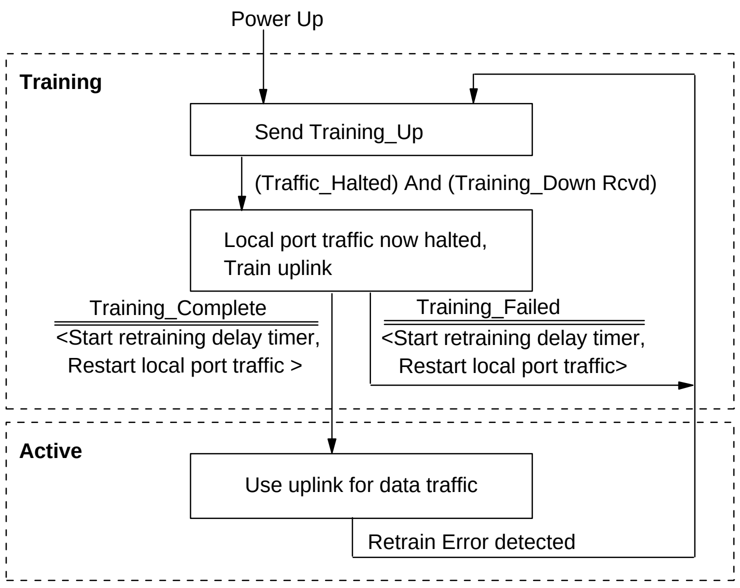

Power Up
Training
Send Training_Up
(Traffic_Halted) And (Training_Down Rcvd)
Local port traffic now halted, Train uplink
Training_Complete
<Start retraining delay timer, Restart local port traffic >
Training_Failed
<Start retraining delay timer, Restart local port traffic>
Active
Use uplink for data traffic
Retrain Error detected

Figure 12-3—Cascade-port/uplink meta-state diagram for single-uplink repeaters

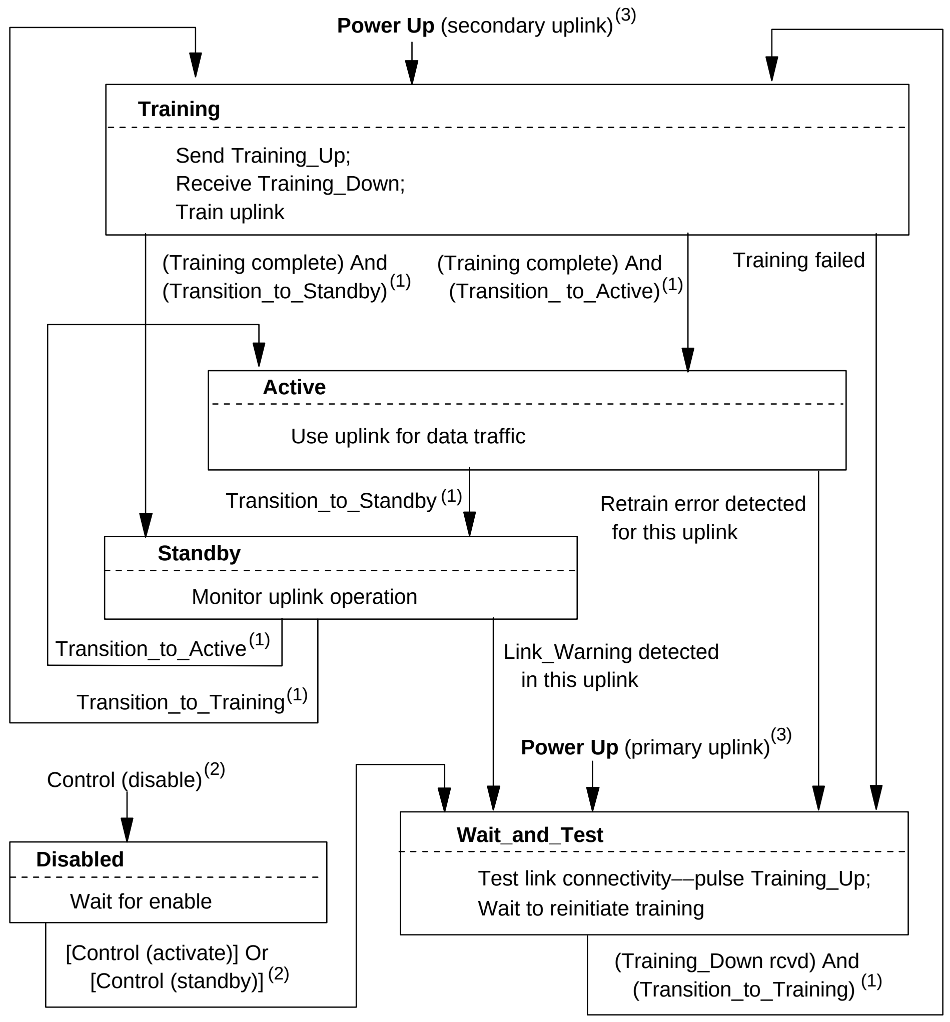
a) **Training.** Uplink training is initiated immediately upon power-up. If training fails, the RMAC waits for a short period and reattempts training. The wait/reattempt training cycle continues until training is successful or network management disables the uplink. The RMAC transitions to the Active meta-state when training is complete. The repeater may neither send nor receive data traffic while uplink training is in progress. Cascade port/uplink training is defined in 12.12.

b) **Active.** The repeater is able to use the uplink to communicate with the connected higher-level repeater. Active mode operations are defined in 12.5 through 12.11.

12.2.1.2 Cascade port/uplink operational modes for redundant-uplink repeaters

If a repeater's cascade port is configured with a redundant uplink, only one uplink may be active at any time.

The recommended meta-state diagram for each uplink in a redundant-uplink cascade port is shown in Figure 12-4. The meta-states are similar to those in single-uplink cascade ports, except that training is not immediately reinitiated after a training or uplink failure, a standby mode has been added so that either uplink can act as an immediately-available alternate uplink, and a specific meta-state is shown for when the uplink has been disabled by network management.


Power Up (secondary uplink)(3)
Training
Send Training_Up;
Receive Training_Down;
Train uplink
(Training complete) And (Transition_to_Standby)(1)
(Training complete) And (Transition_ to_Active)(1)
Training failed
Active
Use uplink for data traffic
Transition_to_Standby (1)
Retrain error detected for this uplink
Standby
Monitor uplink operation
Transition_to_Active(1)
Transition_to_Training(1)
Link_Warning detected in this uplink
Power Up (primary uplink)(3)
Control (disable)(2)
Disabled
Wait for enable
Wait_and_Test
Test link connectivity—pulse Training_Up;
Wait to reinitiate training
[Control (activate)] Or [Control (standby)] (2)
(Training_Down rcvd) And (Transition_to_Training) (1)

NOTES

1—Transitions between meta-states defined as Transition_to_"x" may be the result of either an internal RMAC decision (e.g., after a failure in the other uplink, training was for the secondary uplink, etc.) or the receipt of a network management **acLinkAdministrativeControl (x)** action.

2—Transitions defined as Control (x) are **acLinkAdministrativeControl (x)** actions initiated by network management. A Control (disable) will cause a direct transition from any other meta-state to Disable.

3—The initial meta-state after power-up depends on whether the uplink is primary or secondary. The primary uplink waits to be trained until the secondary uplink either is trained and is in Standby or has failed training and is in Wait_and_Test. The primary uplink will then begin its training sequence.

4—For additional information on state transitions, see 12.2.1.2.1.

Figure 12-4—Cascade-port/uplink meta-state diagram for redundant-uplink repeaters

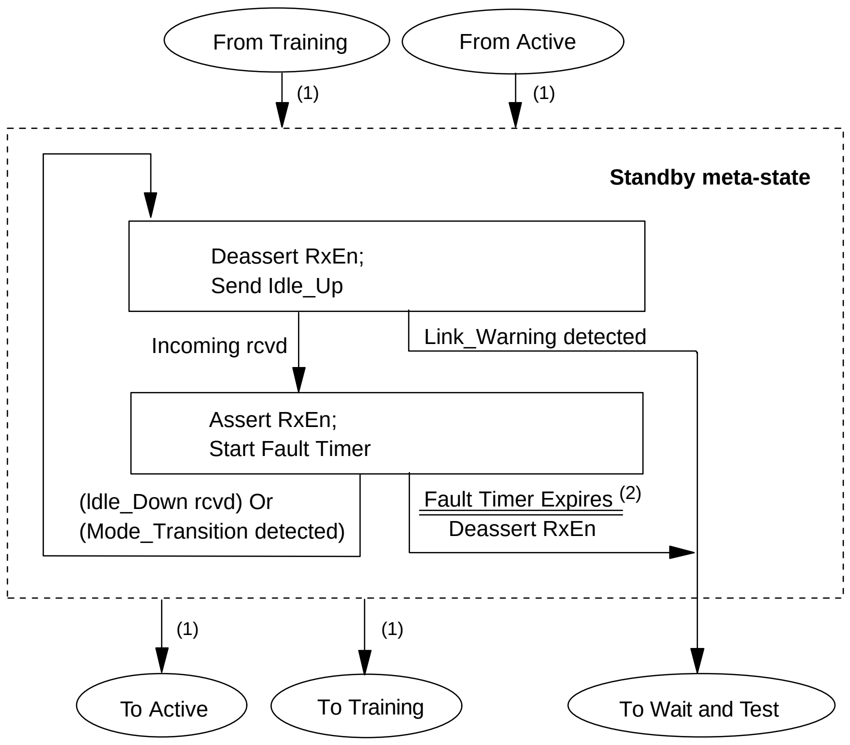
a) **Training.** This meta-state is similar to the training meta-state of a single uplink repeater except that retraining is not automatically reinstituted if training fails. If the repeater is equipped with redundant-link capability and training fails, the uplink is transitioned to Wait_and_Test. Then:

   1) If the repeater's uplinks are to be trained with an assigned non-null address, the secondary uplink should train with the D bit in the requested configuration field of the training frame set to one and should transition to Standby when training is complete. The primary uplink should train with the D bit set to zero and should transition to Active when training has passed (see 10.6.3.7).

2) If training fails, the RMAC will transition the uplink to Wait_and_Test and then an attempt to activate the other uplink will be made.

b) **Active.** The repeater is able to use the uplink to communicate with the connected higher-level repeater. This meta-state is the same as the active meta-state in single-uplink repeaters.

c) **Standby.** This meta-state provides a high level of assurance that the non-active uplink will be fully operational should the active link fail. The uplink has been trained and the RMAC is monitoring link operation by responding to incoming control signals. Incoming data traffic from the link is ignored and no outgoing data traffic is sent over the link. Figure 12-5 shows the recommended state machine for this mode.



NOTES

1—See Figure 12-4 for events and conditions causing transitions into and out of the Standby meta-state. Entry may be into, and exit may be from, either internal state.

2—If the link is inadvertently disconnected while receiving Incoming, Mode_Transition may not be detected. In this case, RxEn should be deasserted after 1–2 ms and the link should then be transitioned to Wait_and_Test.

Figure 12-5—Redundant-uplink standby-mode internal state machine

d) **Wait_and_Test.** The uplink is waiting to be retrained after link failure. Uplink connectivity is tested by periodically sending a short burst of Training_Up followed by a period during which the associated uplink transmitters are disabled. Reception of Training_Down indicates that connectivity exists. Transition from this meta-state to Training can be initiated by either the RMAC or network management after link connectivity has been established.

e) **Disabled.** This meta-state is a wait state. Receipt of a network-management Control (disable) will cause the RMAC to disable the uplink's transmitters. The link will then wait in this meta-state until receipt of a network-management Control (activate or standby).

12.2.1.2.1 Redundant-uplink operational transitions

Except where noted, transitions between meta-states should occur only after any ongoing data packet reception or transmission is complete. Transitions within and between the individual uplinks should be synchronized as follows:

a) If an uplink is in the Active meta-state and a retrain error has been detected but the link cannot be immediately retrained, that uplink should be transitioned to Wait_and_Test. The other uplink should then be activated (if possible).
b) If an uplink is in the Standby meta-state and a Link_Warning is detected, that uplink should be immediately transitioned to Wait_and_Test.
c) If a network management acLinkAdministrativeControl (activate) is received for a designated uplink and:
    1) The other uplink is currently active, it should be transitioned out of the Active meta-state before the designated uplink is activated.[24]
    2) The other uplink is in either the Standby or the Wait_and_Test meta-state, the designated uplink can be activated immediately.
    3) The other uplink is currently in the Training meta-state, the designated uplink should not be activated until the other link has completed its training cycle and has been transitioned to the appropriate meta-state (to Standby if training passes, to Wait_and_Test if training fails).
d) If a network management acLinkAdministrativeControl (standby) is received and:
    1) The designated uplink is currently active, it should be transitioned to the Standby meta-state. The other uplink can then be activated.
    2) The designated uplink is in either the Wait_and_Test or Disabled meta-state, it will need to be retrained before being transitioned to Standby. If the other uplink is currently active, that link may need to be interrupted (transitioned to Standby) long enough for training to take place in the designated uplink (after which, the other uplink can be reactivated).
e) If a network management acLinkAdministrativeControl (disable) is received, the designated uplink will transition to the Disabled meta-state. The disabled uplink will remain in this meta-state until receipt of an acLinkAdministrativeControl (standby or activate).

The physical definition of the Wait_and_Test, the Standby, and the Disabled meta-states, as well as the manner in which connectivity interruption is indicated, are implementation issues and are beyond the scope of this standard.

12.2.1.3 Local port operational modes

The local-port meta-state diagram is shown in Figure 12-6. Each local port is set to inactive upon power-up since the RMAC has no current knowledge of whether or not a lower entity is connected and powered up. Training is always initiated by the connected lower entity. Local ports wait in the Inactive meta-state until Training_Up has been received. No data traffic is allowed through ports in the Inactive or Training meta-state.

[24] The activation process may require the indicated uplink to transition through intermediate meta-states (e.g., if the uplink is in Wait_and_Test, retraining would be required before the uplink could be transitioned to Active).

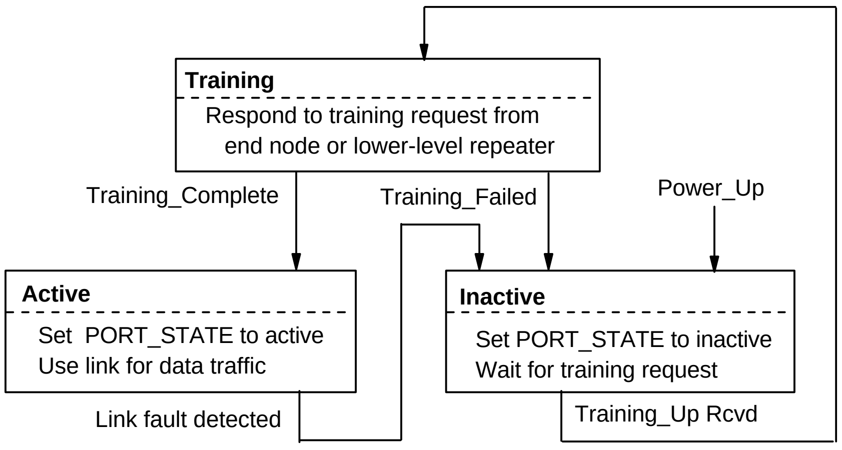

Training
Respond to training request from end node or lower-level repeater
Training_Complete
Training_Failed
Power_Up
Active
Set PORT_STATE to active
Use link for data traffic
Inactive
Set PORT_STATE to inactive
Wait for training request
Link fault detected
Training_Up Rcvd

Figure 12-6—Local-port meta-state diagram

Part of the local port training process involves determining the type of lower entity (repeater or end node) and negotiating the type of traffic it will receive—only packets addressed specifically to the lower entity (privacy mode), or all packets sent over the network (promiscuous mode). Factors affecting local port training are defined in 12.9.8.

Once training is successful, the local port transitions to the Active meta-state and may send and receive data traffic. The port remains in the Active meta-state until it detects a link fault (possibly caused by power being turned off in the end node) or network management disables the link. In either case, the port is returned to the Inactive meta-state.

12.2.2 Link control

Arbitration and control of data traffic over individual links is accomplished through exchange of control-request and control-indication primitives as shown in Figure 12-7.

a) Outgoing control requests are specified by the PMI_CONTROL.request Transmit_Control_State parameter values. These values are passed through the PMI as Transmit Control State (TCS) codes, which are ultimately converted to control signals by the PMD for transmission over the link. (See 14.3.1.)

b) Incoming control signals are extracted from the link by the PMD and converted to Receiver Control State (RCS) and Grant State (GS) codes. They are passed through the PMI and are ultimately indicated as PMI_CONTROL.indication receiver_control_state and grant_state parameter values. (See 14.3.2.)

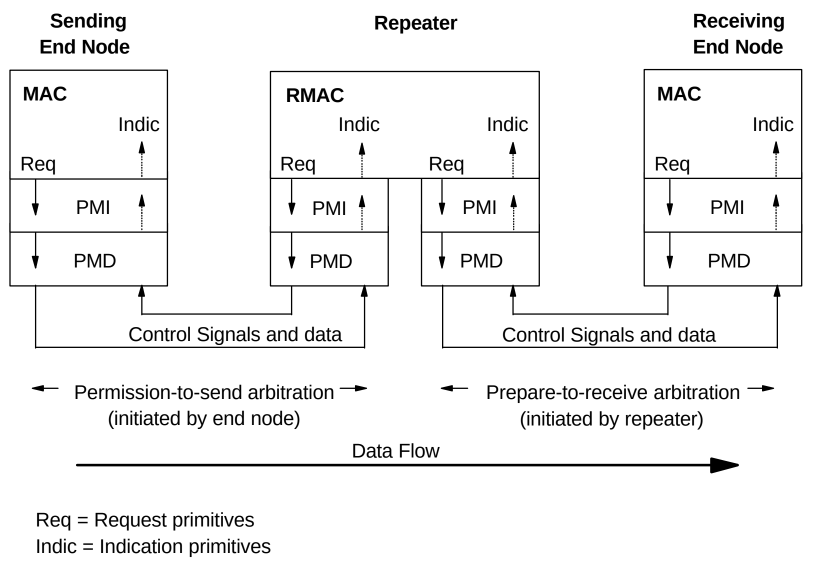


Figure 12-7—Control handshake arbitration

Link arbitration is always initiated by the sending entity, regardless of whether it is an end node requesting permission to send the initial packet or a repeater about to transmit a packet.

12.2.3 RMAC active operational modes

Any repeater(n) can be described as being in one of the three operational modes shown in Figure 12-8 during active operation.

a) Controlling the network and selecting from connected end nodes: Repeater-E is polling the end nodes directly connected to it and selecting end nodes that are currently requesting to send frames. An end node is selected by sending Grant from repeater-E to that end node. Packet sequencing and control of interpacket timing is at repeater-E. This mode is illustrated in Figures 12-9 through 12-11.

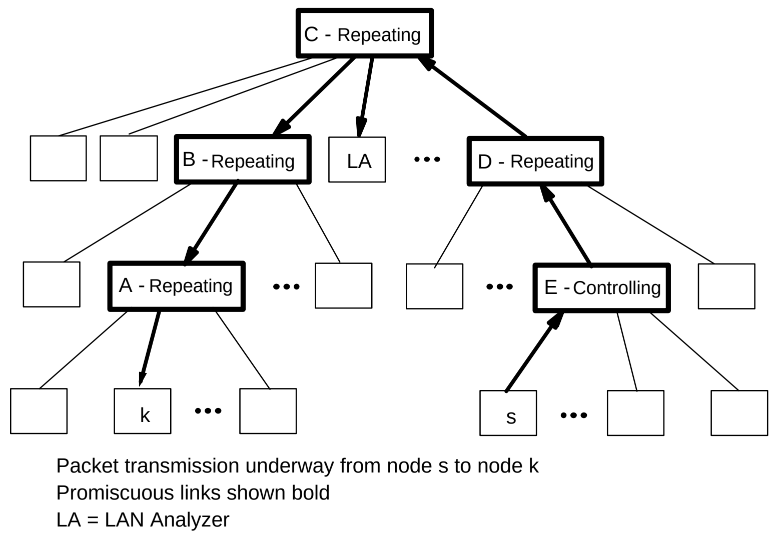
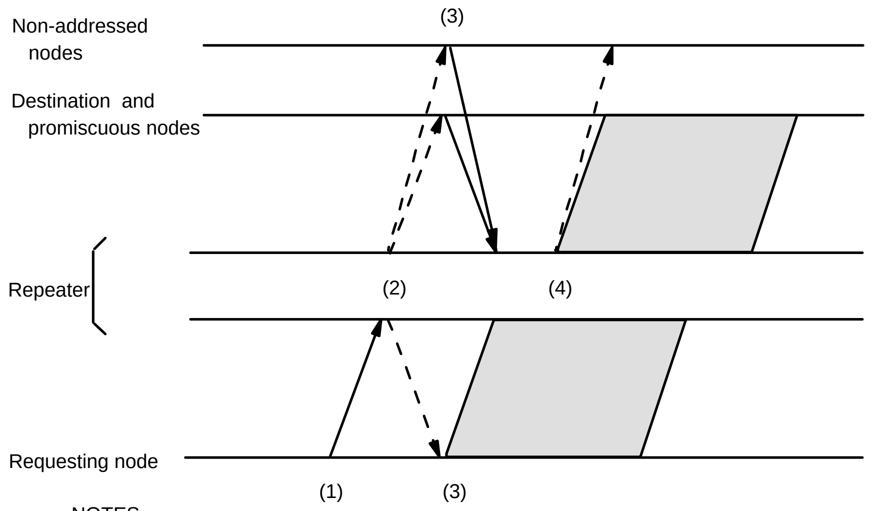


Figure 12-8—Repeater active mode examples



NOTES

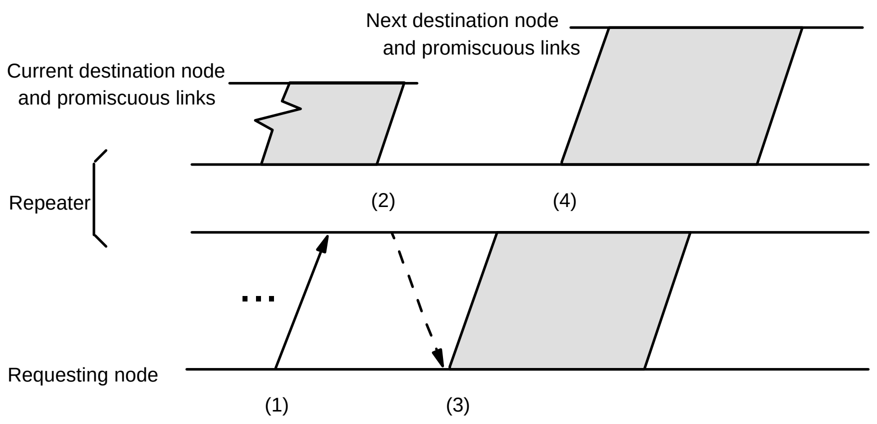
1––Requesting end node sends a request for permission to send a packet.

2––Repeater recognizes the request and sends Grant to the requesting end node; Incoming to all other connected nodes.

3––Requesting end node begins packet transmission as soon as it receives Grant; other nodes clear their links and prepare to receive a packet upon receiving Incoming.

4––After decoding the destination address, the repeater begins transmitting the packet to the destination node(s) and sends Idle to non-addressed nodes.

Figure 12-9—Selecting connected end nodes—Normal overlapped mode, starting from network idle


Next destination node and promiscuous links
Current destination node and promiscuous links
Repeater
Requesting node
(2)
(4)
...
(1)
(3)

NOTES

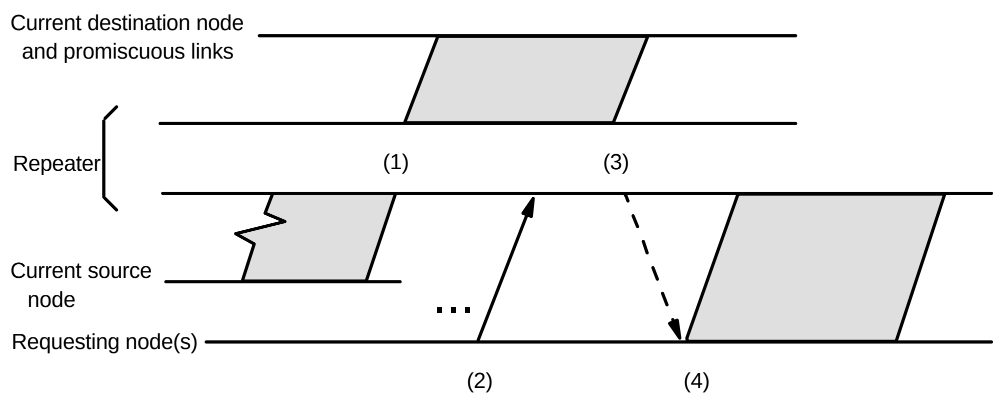
1––End node(s) continue(s) requesting permission to send a packet.

2––Repeater finishes transmitting the current packet, selects the next requesting end node in the round-robin cycle, and then sends Grant to the requesting end node; Incoming to all other connected nodes.

3––Selected end node begins packet transmission as soon as it receives Grant; other nodes clear their links and prepare to receive a packet upon receiving Incoming.

4––After decoding the destination address, the repeater begins transmitting the packet to the destination node(s) and sends Idle to non-addressed nodes.

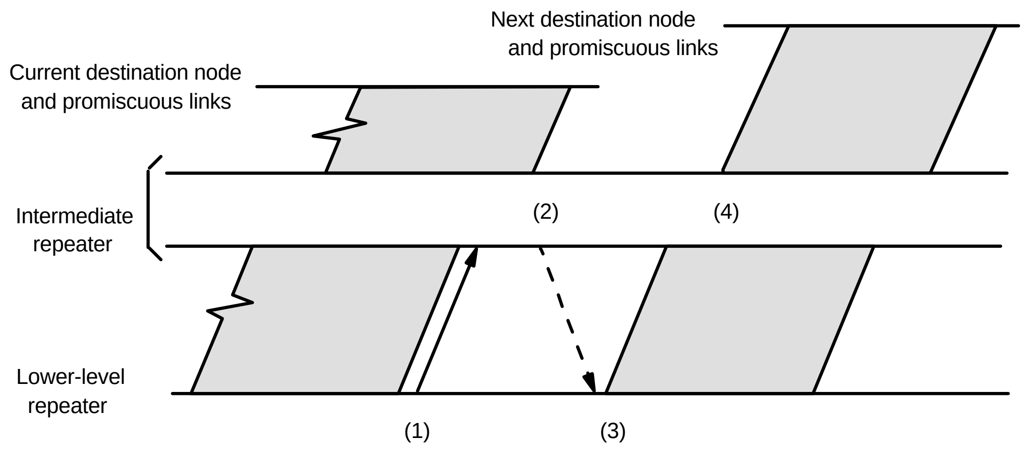
Figure 12-10—Selecting connected end nodes—Normal overlapped mode, network not idle



NOTES
1—Repeater waits until it has detected the end of incoming packet before beginning transmission to the destination node(s).
2—Requesting node(s) continues requesting permission to send a packet.
3—Repeater finishes transmitting the current packet and selects the next requesting end node in the round-robin cycle.
4—Selected end node begins packet transmission as soon as it receives Grant.

Figure 12-11—Selecting connected end nodes—Store and forward mode

b) Repeating packets from below: Repeater-D has selected the lower-level repeater-E, and is receiving packets sent up from that lower repeater. The packet sequencing and control of interpacket is at repeater-E. This is illustrated in Figure 12-2.

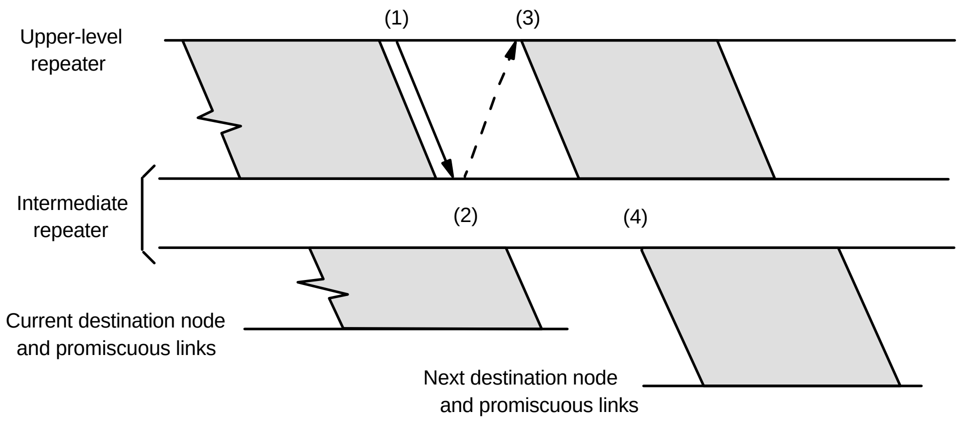

Next destination node and promiscuous links
Current destination node and promiscuous links
Intermediate repeater
(2)
(4)
Lower-level repeater
(1)
(3)

NOTES
1—Lower-level repeater finishes transmitting the current packet and requests permission to continue sending.
2—Intermediate repeater finishes transmitting the current packet and continues to grant the lower repeater permission to send.
3—Lower-level repeater begins next packet transmission as soon as it receives Grant.
4—Intermediate repeater begins transmitting the packet to the destination node(s) after it has decoded the destination address.

Figure 12-12—Repeating frame from a lower-level repeater

Repeating packets from above: Repeater-B is connected through its cascade port to the root repeater (repeater-C), which is sending packets down the link. If repeater-B senses a Request on a local port, it cannot immediately select that port. Repeater-B has to send a Request up to repeater-C and wait until repeater-C has granted permission to send before it can, in turn, select the local port.

Packet sequencing and control of interpacket timing during the time repeater-B is repeating packets from above is at repeater-E. This is illustrated in Figures 12-13 and 12-14.


(1)
(3)
Upper-level repeater
Intermediate repeater
(2)
(4)
Current destination node and promiscuous links
Next destination node and promiscuous links

NOTES

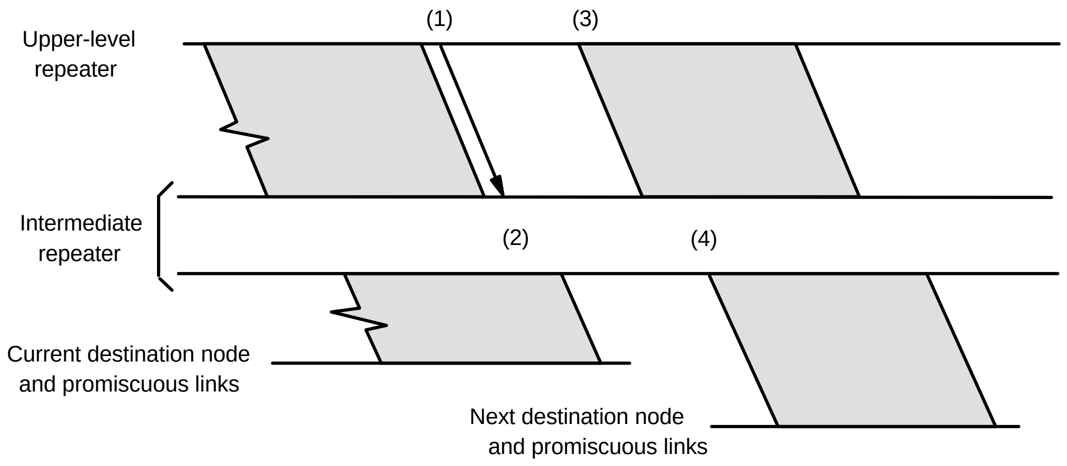
1—Upper-level repeater finishes transmitting the current packet and sends Incoming to indicate that another packet will be sent.

2—Intermediate repeater finishes transmitting the current packet and indicates the existance of local requests by sending a short burst of Request_High or Request_Normal before preparing to receive the next packet.

3—Upper-level repeater begins the next packet transmission.

4—Intermediate repeater begins transmitting the packet to the destination node(s) after it has decoded the destination address.

Figure 12-13—Repeating frames from higher-level repeater (while receiving local requests)


(1)
(3)
Upper-level repeater
Intermediate repeater
(2)
(4)
Current destination node and promiscuous links
Next destination node and promiscuous links

NOTES
1—Upper-level repeater finishes transmitting the current packet and sends Incoming to indicate that another packet will be sent.
2—Intermediate repeater finishes transmitting the current packet and prepares to receive the next packet.
3—Upper-level repeater begins the next packet transmission.
4—Intermediate repeater begins transmitting the packet to the destination node(s) after it has decoded the destination address.

Figure 12-14—Repeating frames from higher-level repeater (no local requests)

12.3 Control state definitions

Transmit Control States provide a means for the RMAC to request, via the PMI, that a particular control action be taken or that a specified control signal[25] be sent by the port's PMD sublayer. Receiver Control States provide a means for the port's PMD to indicate that a particular link condition exists or that the indicated control signal has been received. A special grant state indicates that network control has been passed to the RMAC and permission has been granted for a round-robin port select cycle.

Specific control state definitions for the RMAC depend on whether the particular port is a cascade port or a local port, and if it is a local port, whether the link is connected to an end node or to a lower repeater.

12.3.1 Transmit Control States

The TCS for any one port may be the same as, or different from, the TCS in other ports. New TCS's may be initiated at any time for any port, including while a frame is being transferred into, or out of, the RMAC frame buffer. However, the PMD may or may not act on the requested TCS until the current packet transmission or reception is complete.

TCS definitions for the RMAC are:

Idle_Up, Idle_Down. The control states used to indicate that the sending entity has released its control of the connected link.

[25]Control signal is used as a generic term to indicate a signal that is to be transmitted (e.g., as a Transmit Control State) or a signal that has been received (e.g., as a Receiver Control State or a Grant State).

— Idle_Up sent through the cascade port during the time that the RMAC has control of the network indicates that control is being passed to the connected upper repeater.
— Idle_Up sent through the cascade port during the time that the RMAC does not have control of the network indicates that the RMAC has no immediate traffic for the network.
— Idle_Down sent through a local port indicates that no immediate traffic is currently destined for the lower entity connected to that port.

Incoming. The control state used by the RMAC to alert connected lower entities that a packet may be incoming. Incoming may be sent through local ports only.

Grant. The control state used to grant a lower entity permission to send. Grant may be sent through local ports only.

— End nodes are granted permission to send one packet only.
— Lower repeaters are granted permission to send multiple packets and to control the network for one complete round-robin cycle of the requested priority in that repeater. If one or more of the granted repeater's local ports are also connected to a lower repeater (the granted repeater is an intermediate level repeater in a cascade in this case), then the granted repeater may grant control of the network to its connected lower repeater(s) as part of its round-robin cycle.

Enable_High_Only. The control state used by the RMAC to inform a connected lower repeater that a high-priority request has been received, and that the RMAC wishes to regain control of the network. Enable_High_Only will be sent regardless of whether the lower, controlling repeater is currently processing normal or high-priority requests. Enable_High_Only may only be sent through local ports connected to lower repeaters.

Request_High, Request_Normal. The control states used to inform a connected upper repeater that the RMAC wishes to send a packet. Requests are sent through the cascade port only.

— Request_High indicates that the repeater currently has at least one request to send a high-priority packet.
— Request_Normal indicates that the repeater currently has at least one request to send a normal-priority packet and that no high-priority requests are pending.

Return. The control state used by the RMAC to indicate that its interrupted, normal-priority, round-robin cycle is not complete, that further normal-priority requests in the cycle are still pending, and that the RMAC is requesting return of control after the high-priority requests have been satisfied at the upper repeater. Return is always a secondary control signal, and may be sent through the cascade port only.

Training_Up, Training_Down. The control state used to indicate that link training is desired or is in progress.

— Training_Up sent through the cascade port indicates that training of the uplink is requested or is not complete.
— Training_Down sent through a local port indicates that training of that link may begin or continue.

TxDisable. The control state that allows the RMAC to request that the PMD's transmitters be disabled.

12.3.2 Receiver Control States

The RCS's indicate either the current or previous control state for the port, depending on whether the grant state is one or zero (see 12.3.3). The RCS definitions for the RMAC are:

Data_on_Link. The control state used by the PMD to indicate that data is currently being extracted from the link.

Idle_Up, Idle_Down. The control state used to indicate that the sending entity has released its control of the connected link.

— Idle_Up received at a local port connected to an end node indicates that the end node has no immediate traffic for the network.
— Idle_Up received at a local port connected to a lower repeater indicates either that the lower repeater has no immediate traffic for the network or that control of the network is being passed up.
— Idle_Down received at the cascade port indicates that the upper repeater has no immediate traffic for the network.

Incoming. The control state used by an upper repeater to alert the RMAC that a packet may be incoming. Incoming is defined for reception at cascade ports only.

Link_Warning. The Receiver Control State used by the PMD to indicate a condition that may require the link to be retrained.

Mode_Transition. The control state indicating that the PMD has detected a signaling transition between incoming data and control.

Enable_High_Only. The control state used to indicate that a high-priority request has been received by a higher-level repeater, and that the upper repeater wishes to regain control of the network. Enable_ High_Only is only defined for reception at cascade ports.

Request_High, Request_Normal. The control states indicating that a connected lower entity wishes to send a packet. Requests are only defined for reception through the following local ports:

— Request_High indicates that a request to send a high-priority packet is pending.
— Request_Normal indicates that a request to send a normal-priority packet is pending.

Return. The control state indicating that a connected lower repeater's interrupted, normal-priority, round-robin cycle is not complete, that there are more normal-priority requests in the cycle, and that return of control is requested after the high-priority requests have been satisfied. Return is always a secondary control signal and is only defined for reception through a local port. Return is indicated by a grant_state parameter value of one.

Training_Up, Training_Down. The control state that is used to indicate that link training is desired or may continue.

— Training_Up received at a local port indicates that training of the port is either requested or is not complete.
— Training_Down received at a cascade port indicates that uplink training may begin or continue.

12.3.3 Grant state coding

The grant_state parameter value of one in the PMI_CONTROL.indication primitive at the cascade port indicates that permission has been, or continues to be, granted to the RMAC to control the network for one complete round-robin cycle When a lower repeater is passing up control, a grant_state of one is defined as the control state Return. Interpretation of a grant_state value of one is defined in Table 12-1.

Table 12-1—RMAC state machine interpretations of a grant_state value of one

Interpreting entity	Last control state transmitted by interpreting entity	State machine interpretation
Controlling Repeater (cascade port)	Req_H or Req_N	Continuing Grant
Noncontrolling Repeater (cascade port)	Req_H or Req_N	Grant to initiate one round-robin cycle
Controlling Repeater (local port with connected lower repeater)	Enable_High_Only	Return control to complete interrupted normal-priority round-robin cycle

12.3.4 Internal control

Requests for control actions and indications of conditions that are internal to the repeater and necessary for proper operation are defined by the following PMI primitives:

a) PMI_TRANSMIT.indication allows the PMI to inform the RMAC when packet transmission is complete[26] (see 8.6).
b) PMI_RECEIVE_ENABLE.request defines when the physical sublayers are to switch to the data reception mode (see 8.8).

12.3.5 PMD/Link configuration coding

The PMD/Link configuration for each port is provided by a PMI_CONFIGURATION.indication primitive indicating the type of PMD/Link combination that is installed. (See 8.7 and 15.3.11.)

12.4 Overview of the RMAC state machines

The logical structure of the RMAC is a central set of state machines supported by a separate Port Process Module (PPM) at each PMI interface as shown in Figure 12-15. The PPM is described in 12.11.

[26]Because start of stream and end of stream sequences are generated by the PMI rather than the MAC, the PMI_ TRANSMIT.indication (transmit_complete) occurs a number of clock cycles after transfer of octets from the MAC to the PMI is complete.

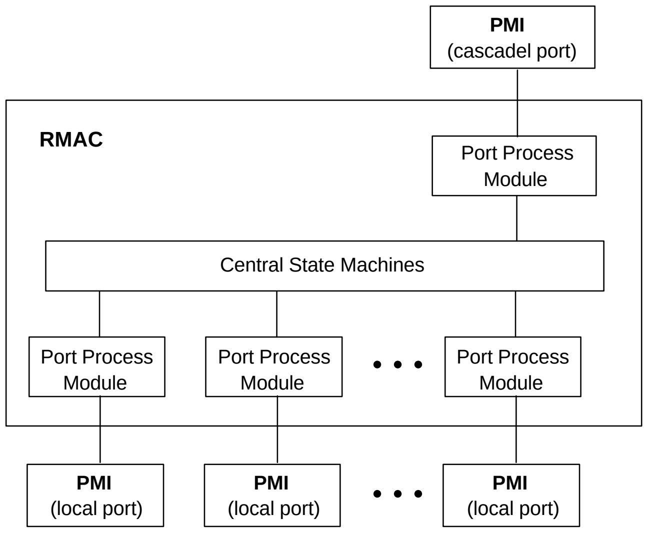


Figure 12-15—RMAC structural overview

Figure 12-16 shows an expanded view of the central state machines in the RMAC and typical interconnections with a PPM.

a) The Receive State Machine (RSM) provides master control over the packet sequencing process. It controls the round-robin port-selection cycle and coordinates the activities of the other state machines. The RSM is described in 12.7.

b) The Train Uplink Machine (TULM) is an adjunct to the RSM that pertains to the cascade port only. It effectively takes the place of the RSM during training of the uplink. It is inactive at all other times. The TULM is described in 12.12.

c) The Transmit State Machine (TSM) controls the sequence of transmit-related operations before, during, and after packet transmission. It sets the appropriate TCS's for each port and combines with the BCALL to identify when packet transmission can begin. The TSM is described in 12.8.

d) The Buffer Control and Address Lookup Logic (BCALL) state machine controls the transfer of octets into and out of the frame buffer, extracts the destination address from incoming frames, determines the ports out of which the retransmitted frames will be sent, and ensures that oversize frames do not overrun the buffer. The BCALL is described in 12.9.

e) The Fault Condition Process (FCP) state machine is in the data path to the RMAC buffer in the BCALL state machine, and shields the RSM or TULM from having to know if an expected frame either fails to appear on time or is received with a corrupted start of stream sequence. Data path output from the FCP is either data octets from the PMI or octets of all zeros from the void frame generator. The FCP is described in 12.10.

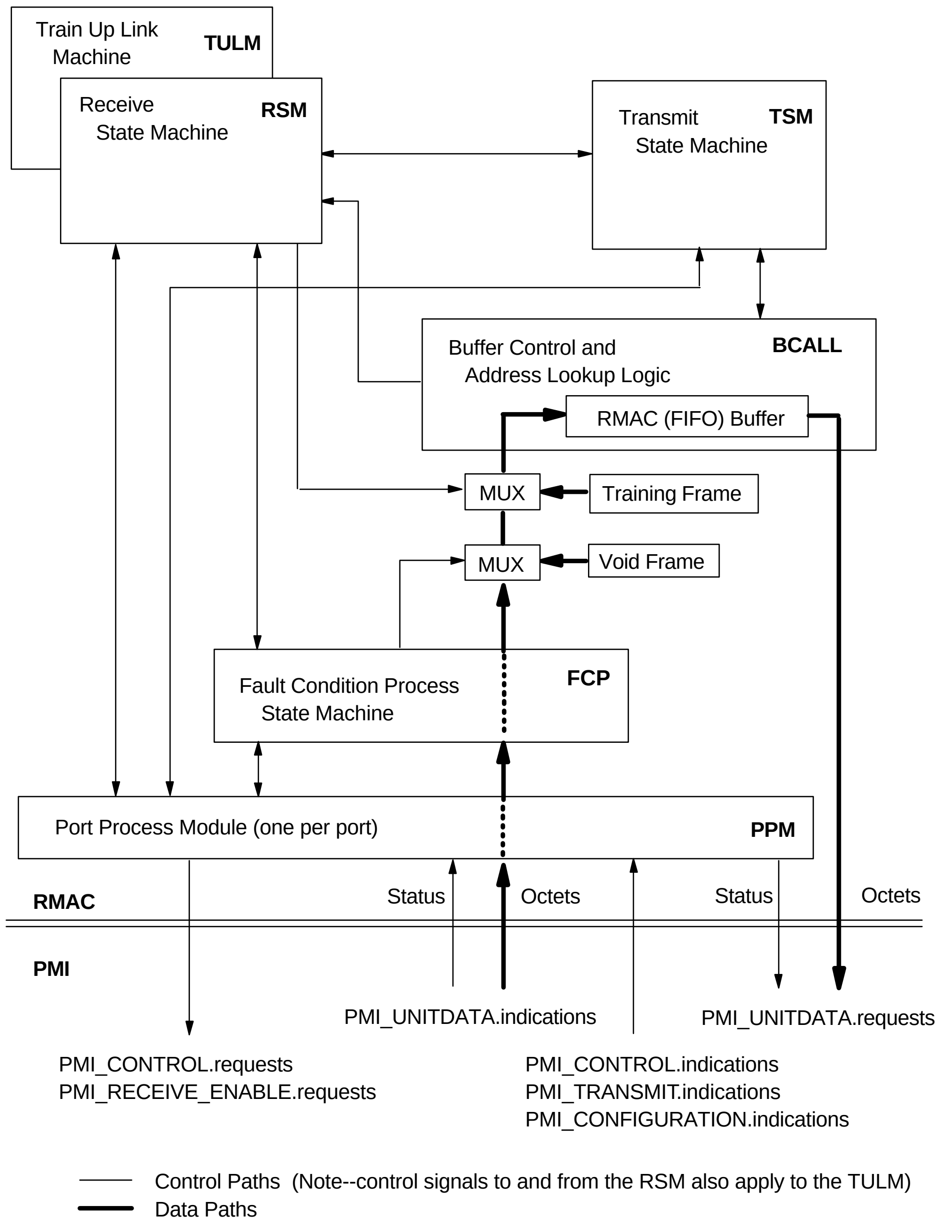


Figure 12-16—RMAC state machine interconnections

12.5 State machine constants, variables, and types

The pseudo code for the RMAC state machines contains a number of constants and variables. Those that are used by multiple machines (global constants and variables) are listed in this subclause. Constants and variables that are used exclusively within the code for one machine (or one function or procedure) are listed in the subclause defining that routine.

12.5.1 Global constants

General constants

```
ZERO = 0;
```

NUMBER_OF_PORTS = {Depends on size of repeater, arbitrary value: } 100;
CASCADE_PORT = ZERO;

Timer values

A number of timers are used by the various state machines in the RMAC. They are defined by the T_TIMER type definitions of 12.5.2.1, are initiated by either the procedure SET_FAULT_TIMER or the procedure SET_TIMER described in 12.6.2, and may be set to any value within the ranges listed in Table 12-2.

Table 12-2—RMAC timer values

Timer name	Used in state machine	Set to value
IDLE_DURATION	RSM	60–70 BT
IDLE_FILTER	RSM	60–68 BT
IN_RX6_TOO_LONG	RSM	84–90 BT
PASS_UP_CONTROL_IDLE	RSM	72 BT
PASS_UP_TO_IDLE_DOWN_TIMEOUT	RSM	200–400 µs
REQ_WINDOW	RSM	40 BT
REQUESTING_TOO_LONG	RSM	1–2 s
SEND_IDLE_BURST	RSM	24 BT
DELTA_IPG_WINDOW	TSM	12–16 BT
IPG_WINDOW	TSM	190–194 BT
SENT_INCOMING_LONG_ENOUGH	TSM	190–194 BT
FRAME_EXPECTED_TO_SMF_IN	FCP	450 µs
LONG_RECEIVE_INITIATE_TRAINING For 8802-3 frames For 8802-5 frames	FCP	 123–124 µs 363–364 µs
VOID_FRAME	FCP	550–600 octets
TR_FINAL_IDLE_OR_INCOMING	TULM	200–400 µs
TR_INTERVAL_TO_RESTART_TRAINING	TULM	1–2 s
TR_REQ_UP_TO_FRAME_RECEIVED	TULM	200–400 ms
TRAINING_PULSE_SENDING	TULM	3–9 µs
TRAINING_PULSE_TXDISABLE	TULM	200–400 µs
SEC_CTRL_SIG_DECODE_DELAY	BCALL	84–100 BT
PACKET_PROMOTION (one timer is required for each port)	PPM	200–300 ms

REQ_WINDOW is chosen so that the RxEn deassertion time at the MII for a packet followed by "Incoming" is 41–45 cycles of TxClk.

PASS_UP_CONTROL_IDLE is chosen so that at the MII there are 73–77 cycles of TxClk between the deassertion of TxEn and sending the TCS code for secondary control signal.

SEND_IDLE_BURST is chosen so that at the MII there are 25–29 cycles of TxClk between the deassertion of TxEn and sending the TCS code for grant in the case where the RMAC is ready to grant the port immediately after a transmission to the port.

The IPG_WINDOW and DELTA_IPG_WINDOW timers pace the transmission of packets into the network. This guarantees that the TxEn deassertion period for back-to-back packets at the repeater sourcing end node packets into the network is a minimum of 202 TxClk cycles. This spacing provides a window for end nodes to send up their requests between packets.

12.5.2 Types

12.5.2.1 Global types

```
T_PORT = ZERO..NUMBER_OF_PORTS;

T_LOCAL_PORT = 1..NUMBER_OF_PORTS;

T_LINK_ID_VALUE = 1...2;

T_LINK_STATUS_VALUE = (  ACTIVE,
                         DISABLED,
                         STANDBY,
                         TRAINING,
                         WAIT_AND_TEST );

T_EVENT_SORT = (TIMER_EXPIRES,              { From timer to all state machines }

            { Signals from TULM to RSM }
        S_TULM_TO_RSM_HALT_TRAFFIC,
        S_TULM_TO_RSM_RESTART_TRAFFIC,
        S_TSM_TO_RSM_EMF_XMITTED,

            { Signals to TULM only: }
        S_TO_TULM_RETRAIN_UPLINK,          { from RSM, FCP }
        S_RSM_TO_TULM_TRAFFIC_HALTED,              { from RSM }
        TRAINING_FRAME_INTO_RMAC_BUFFER_COMPLETE,   { from BCALL }

            { Signals from FCP to RSM and TULM: }
        S_FCP_SMF_RECEIVED_THRU_LOCAL_PORT,
        S_FCP_SMF_RECEIVED_THRU_CASCADE_PORT,
        S_FCP_EMF_RECEIVED_THRU_LOCAL_PORT,     {to RSM only}
        S_FCP_EMF_RECEIVED_THRU_CASCADE_PORT,

            { Signal from PPM to RSM, TULM and FCP: }
        CASCADE_PORT_CONTROL_SIGNAL_DECODED,
            { Signal from PPM to RSM and FCP: }
        LOCAL_PORT_CONTROL_SIGNAL_DECODED

            { Signals to FCP: }
        S_TO_FCP_EXPECT_FRAME,                  { from RSM, TULM }
        S_TO_FCP_STOP_EXPECTING_FRAME,          { from RSM, TULM }
        S_SMF_RECEIVED_THRU_PORT,               { from PPM }
        S_BAD_SMF_RECEIVED_THRU_PORT,           { from PPM }
        S_EMF_RECEIVED_THRU_PORT,               { from PPM }
        S_FRAME_ENDED_ABNORMAL_END,             { from PPM }

            { Signals to TSM: }
        S_TO_TSM_GOTO_TX_INCOMING,              { from RSM, TULM }
        S_TO_TSM_GOTO_TX_IDLE,                  { from RSM, TULM }
        S_BCALL_TO_TSM_CAN_START_TRANSMIT,      { from BCALL }
        S_BCALL_TO_TSM_EMF_TRANSMITTED,         { from BCALL }

            { Signals from TSM to BCALL: }
        S_TSM_TO_BCALL_START_TRANSMIT;)

T_TIMER =   (   { RSM timers: }
        IDLE_DURATION,
        IDLE_FILTER,
        IN_RX6_TOO_LONG,
        PASS_UP_CONTROL_IDLE,
        PASS_UP_TO_IDLE_DOWN_TIMEOUT,
        REQ_WINDOW,
        REQUESTING_TOO_LONG,
        SEND_IDLE_BURST,

            { TSM timers: }
        DELTA_IPG_WINDOW,
        IPG_WINDOW,
        SENT_INCOMING_LONG_ENOUGH,

            { FCP timers: }
        FRAME_EXPECTED_TO_SMF_IN,
        LONG_RECEIVE_INITIATE_TRAINING,
        VOID_FRAME,

            { TULM timers: }
        TR_FINAL_IDLE_OR_INCOMING,
        TR_INTERVAL_TO_RESTART_TRAINING,
        TR_REQ_UP_TO_FRAME_RECEIVED,
        TRAINING_PULSE_SENDING,
        TRAINING_PULSE_TXDISABLE,

            { BCALL timer: }
        SEC_CTRL_SIG_DECODE_DELAY

            { PPM timer, one instance per Local Port: }
        PACKET_PROMOTION);
```

{ The comments in the following type declaration define the correspondence between the control signal names defined in Clause 14 and the identifiers used in the pseudo code. }

```
T_CONTROL_STATE = (

            { Transmit only: }
        TXDISABLE,                  { TxDisable }

            { Receive only: }
        DATA_ON_LINK,     { Data_on_Link }
        LINK_WARNING,     { Link_Warning }
        MODE_TRANSITION,        { Mode_Transition }

      { Tx on local ports, Rx on cascade port: }
        GRANT,      { Grant }
        TRAINING_DOWN,          { Training_Down }
        ENABLE_HIGH_ONLY,       { Enable_High_Only }
        INCOMING,   { Incoming }
        IDLE_D,     { Idle_Down }

            { Tx on cascade port, Rx on local ports: }
        RETURN,     { Return }
        TRAINING_UP,      { Training_Up }
        REQ_H,      { Request_High }
        REQ_N,      { Request_Normal }
        IDLE_U,     { Idle_Up }
        );

T_RXSTATE =   (RX1_IDLE,
        RX2_WAIT_FOR_FRAME_FROM_ABOVE,
        RX3_WAIT_FOR_SELECT,
        RX4_RECEIVING_FRAME_FROM_ABOVE,
        RX6_EXPECT_IDLE_OR_REQ,
        RX7_EXPECT_FRAME_FROM_BELOW,
        RX8_RECEIVING_FRAME_FROM_BELOW,
        RX9_PASS_UP_GRANT,
        RX10_WAIT_FRAME_FROM_LOCAL_PORT_REPEATED,
        RX11_SEND_SECONDARY_CONTROL_SIGNAL);

T_TXSTATE =   (TX1_SEND_FRAME,
        TX2_SEND_IDLE,
        TX3_SEND_INCOMING);

T_FCPSTATE =  (FCPSTATE1_QUIESCENT,
        FCPSTATE2_EXPECT_SMF,
        FCPSTATE3_START_VOID_FRAME,
        FCPSTATE4_TRANSFERRING_DATA_TO_RSM);

T_TULMSTATE =  (TULM1_UP_LINK_OPERATING_OK,
        TULM2_WAIT_FOR_TRAINING_DOWN,
        TULM3_WAIT_RESTART_TRAINING,
        TULM4_WAIT_FOR_GRANT,
        TULM5_PUSH_TRAINING_FRAME_INTO_RMAC_BUF,
        TULM6_WAIT_FOR_INCOMING,
        TULM7_WAIT_FOR_FRAME,
        TULM8_RECEIVING_FRAME,
        TULM9_WAIT_IDLE_OR_INCOMING);

T_ENTITY_AT_FAR_END_OF_LINK = (
        END_NODE,
        LOWER_REPEATER);

T_PORT_STATE =  (ENABLED,
        DISABLED);

T_PORT_META_STATE = (
        ACTIVE,
        INACTIVE,
        TRAINING);

T_PRIORITY =    (HIGH_PRIORITY,
        NORMAL_PRIORITY,
        NONE    { NONE indicates no requests pending }
        );

T_LOWER_PORTS_TRAFFIC = (
        OPERATING,
        HALTED);

T_EMF_INDICATION = (
        INVALID_PACKET_MARKER,
        PACKET_OK);

T_FCP_DATA_OUTPUT = (
        OCTETS_FROM_PMI,
        NIL,
        ZERO_OCTETS);

T_RMAC_BUF_DATA_INPUT = (
        OCTETS_FROM_FCP,
        STOP_INPUT,
        TRAINING_FRAME_OCTETS);
```

12.5.2.2 Types Used Only in RSM

```
T_CSTATE =      (C_HIGH,
            C_NORMAL,
            C_HIGH_WAS_NORMAL_UNFINISHED);

T_DECISION =    (SELECT_PORT,
            NO_REQUESTS_RECEIVED,
            PASS_UP_CONTROL,
            PASS_UP_CONTROL_SECONDARY_CTL_SIG_RETURN);
```

12.5.2.3 Types Used Only in PPM

```
T_PMI_ERROR_STATUS = (
            VALID,
            IPM,
            EMF_ERROR);

T_PMI_BME =     (BEGIN,
            MORE,
            END);

T_PORT_DATA_STATE = (
            PDS_NONE,
            PDS_EXPECT_SMF,
            PDS_RECEIVE_DATA);

T_OCTET = 0..255;

T_MAC_ADDRESS = Array [1..6] Of T_OCTET;
```

12.5.3 Global variables

12.5.3.1 Variables used for configuration and management

ENTITY_AT_FAR_END_OF_LINK: Array [1..NUMBER_OF_PORTS] Of T_ENTITY_AT_FAR_ END_OF_LINK; Indicates whether the entity at the far end of the link is a lower repeater or an end node. It is initialized at the start of training as END_NODE and set to LOWER_REPEATER when training as a repeater is successful. If the port meta-state transitions to INACTIVE or TRAINING, then this variable reverts to END_NODE.

PORT_STATE: Array [Zero...NUMBER_OF_PORTS] Of T_PORT_STATE; Indicates whether the Port has been set (by network management) as "Enabled" or "Disabled."

PRIORITY_ENABLE: Boolean; Indicates whether high-priority requests are enabled (treated as high-priority requests) or disabled (treated as normal-priority requests).

12.5.3.2 Variables that are training related

CASCADE_PORT_META_STATE: T_PORT_META_STATE; Indicates whether port is Active, Inactive, or Training.

CONFIG_OK: Boolean; When True, indicates to TULM that a cascade port training sequence has completed, with acceptable configuration information.

LINK_ID : T_LINK_ID_VALUE; Identifies the specific link being interrogated or controlled.

LINK_STATUS : T_LINK_STATUS_VALUE; Indicates the meta-state of the link.

LOCAL_PORT_META_STATE: Array [1..NUMBER_OF_PORTS] Of T_PORT_META_STATE; Indicates whether port is Active, Inactive, or Training.

LOCAL_PORT_PACKETS_OK_SEQUENCE: Array [1..NUMBER_OF_PORTS] Of Integer; Keeps a count of the consecutive good packets during the training of a Local Port.

VALID_CONFIG: Boolean; When True, indicates to PPM that a local port training sequence has completed, with valid configuration information (see 12.9.8).

{ BCALL related, see 12.9.2: }

ADDRESS_MATCH: Array [ZERO..NUMBER_OF_PORTS] Of Boolean; ADDRESS_MATCH[PORT_X] indicates whether the current frame will be sent out of the port denoted by PORT_X.

EMF_INDICATION: T_EMF_INDICATION; See 12.9.2.

ERROR_INDICATION: T_EMF_INDICATION; See 12.9.2.

FCP_DATA_OUTPUT: T_FCP_DATA_OUTPUT; See 12.9.2.

RMAC_BUF_DATA_INPUT: T_RMAC_BUF_DATA_INPUT; Indicates whether octets are going into the RMAC buffer, and the source of the octets (from FCP or internally generated training frame).

12.5.3.3 Other variables used by two or more processes

FLAG_FCP_TO_PPM_FRAME_ENDED: Boolean; Indicates to the PPM's PROCESS_DATA procedure that frame reception has been aborted by FCP because the frame is too long.

LOWER_PORTS_TRAFFIC: T_LOWER_PORTS_TRAFFIC; Set by TULM to inform the RSM when Local Port Traffic (traffic involving local ports but not the cascade port) should be stopped or started.

RMAC_ENABLE_HIGH_ONLY_RECEIVED_REG: Boolean; Indicates that the repeater may process high-priority requests only.

RMAC_REQ_REG: Array [1..NUMBER_OF_PORTS] Of T_PRIORITY; Indicates whether each port is or has been requesting at high priority or normal priority or not at all. This is updated by PPM when a new control state (REQ_H, REQ_N or IDLE_U) is set in the Receiver Control State register by a Physical Layer.

SELECTED_PORT: T_PORT; The port that is currently, or was last, selected. Can be the Cascade Port.

SELECTED_PORT_PRIORITY: T_PRIORITY; Indicates the priority level of the selected port.

SILENT_WHEN_GRANTED: Boolean; Indicates that the selected port was receiving silence when granted.

12.5.3.4 Variables that indicate the current states of the RMAC machines

FCPSTATE: T_FCPSTATE; Indicates the state of the fault condition process state machine.

RXSTATE: T_RXSTATE; Indicates the state of the Receive State Machine.

TULMSTATE: T_TULMSTATE; Indicates the state of the Train Uplink State Machine.

TXSTATE: T_TXSTATE; Indicates the state of the Transmit State Machine.

12.5.4 Parameters associated with interprocess signals

CASCADE_PORT_CONTROL_SIGNAL_IN: T_CONTROL_STATE; This parameter is associated with the CASCADE_PORT_CONTROL_STATE_DECODED signal, and indicates the control signal that has been received.

EVENT_SORT: T_EVENT_SORT; This is the basic parameter indicating that an event has occurred.

LOCAL_PORT_CONTROL_SIGNAL_IN: T_CONTROL_STATE; This parameter is associated with the LOCAL_PORT_CONTROL_STATE_DECODED signal. It indicates the control signal that has been received.

PORT: T_LOCAL_PORT; This parameter is associated with the LOCAL_PORT_CONTROL_ STATE_DECODED signal. It indicates the port on which the signal has occurred.

TIMER: T_TIMER; This parameter is associated with the TIMER_EXPIRES event. It indicates which timer has expired.

12.6 Functions and procedures

The RMAC state machines make use of several functions and procedures that are common to several sections of pseudo code. (The notation for functions and procedures is defined in 5.3.3.)

12.6.1 Basic functions

Function DECREMENT_NORMAL_PRIORITY_POINTER_FROM_PORT (PORT_X : T_ LOCAL_PORT) : T_PRIORITY**;** Used by the RSM: If the normal-priority pointer points to PORT_X, that pointer is decremented so that PORT_X will be selected first when the normal-priority cycle continues, if there is a request pending.

Function HIGHEST_PRIORITY_REQUEST_AT_OTHER_PORTS (PORT_X : T_LOCAL_ PORT): T_PRIORITY**;** Used by the RSM: This function returns the highest priority of those requests on all other local ports except PORT_X.

Function HIGHEST_PRIORITY_REQUEST_RECEIVING (): T_PRIORITY; Used by the RSM: This function examines the RMAC_REQ_REG, and returns the highest priority of the requests received.

Function MORE_REQUESTS_IN_CYCLE (C_PRIORITY : T_PRIORITY): Boolean; This function indicates if there are any more requests remaining in the round-robin cycle of the xxx_PRIORITY_ POINTER. Takes the parameter HIGH_PRIORITY (more high-priority requests in the high-priority cycle) or NORMAL_PRIORITY (more normal requests).

Function SELECTED_PORT_GRANT_STATE_IS_ZERO (): Boolean; Called by FCP to determine whether a grant state value of zero has been received on the selected port (the information is held by the PPM).

12.6.2 Basic procedures

Procedure ASSERT_RECEIVE_ENABLE (IN_PORT : T_PORT); Used by the FCP: Generates a primitive to the PMI which causes RxEn to be asserted at the port indicated by IN_PORT.

Procedure CANCEL_FAULT_TIMER (TIMER_NAME : T_TIMER); Used by all state machines: Cancels the named fault timer.

Procedure CANCEL_TIMER (TIMER_NAME : T_TIMER); Used by all state machines: Cancels the named timer.

Procedure CHANGE_CASCADE_PORT_CONTROL_SIGNAL (C_SIGNAL : T_CONTROL_ STATE); Changes the control signal being output on the cascade port. This procedure is fully defined in 12.11.4.

Procedure CHANGE_LOCAL_PORT_CONTROL_SIGNAL (C_SIGNAL : T_CONTROL_STATE, IN_PORT : T_LOCAL_PORT); Changes the control signal being output on the local port IN_PORT. This procedure is fully defined in 12.11.3.

Procedure DEASSERT_RECEIVE_ENABLE (IN_PORT : T_PORT); Used by the FCP: Generates a primitive to the PMI, which causes RxEn to be deasserted on the port indicated by IN_PORT.

Procedure SEND_IDLE_TO_LOCAL_PORTS(); Used by both the RSM and TSM: Changes the control signal being output on local ports to IDLE_D.

Procedure SEND_INCOMING_TO_LOCAL_PORTS(); Used by the TSM: Changes the control signal being output on local ports to INCOMING, with the exception of the local port (if any) indicated by SELECTED_PORT.

Procedure SET_FAULT_TIMER (TIMER_NAME : T_TIMER); Used by all state machines: Sets (initiates) a fault timer. The parameter is the timer name.

Procedure SET_TIMER (TIMER_NAME : T_TIMER); Used by all state machines: Sets (initiates) a timer. The parameter is the timer name.

Signal generating procedures

The following procedures implement the sending of signals between the processes, where no additional parameters are associated with the signals:

a) **Procedure SIGNAL_TO_BCALL (SIGNAL : T_EVENT_SORT);**
b) **Procedure SIGNAL_TO_FCP (SIGNAL : T_EVENT_SORT);**
c) **Procedure SIGNAL_TO_RSM (SIGNAL : T_EVENT_SORT);**
d) **Procedure SIGNAL_TO_RSM_TULM (SIGNAL : T_EVENT_SORT);**
e) **Procedure SIGNAL_TO_TSM (SIGNAL : T_EVENT_SORT);**
f) **Procedure SIGNAL_TO_TULM (SIGNAL : T_EVENT_SORT);**

The following procedures implement the sending of signals from the PPM to other processes, where additional parameters are associated indicating a control-signal name and, in one case, a local port.

g) **Procedure SIGNAL_CS_TO_FCP (SIGNAL : T_EVENT_SORT, C_SIGNAL : T_CONTROL_ STATE);**
h) **Procedure SIGNAL_CS_TO_RSM_TULM (SIGNAL : T_EVENT_SORT, C_SIGNAL : T_ CONTROL_STATE);**
i) **Procedure SIGNAL_PORT_CS_TO_RSM (SIGNAL : T_EVENT_SORT, C_SIGNAL : T_ CONTROL_STATE, IN_PORT : T_LOCAL_PORT);**

12.6.3 Function definitions

All of the following functions are used only in the RSM.

12.6.3.1 Function CASCADE_PORT_ACTIVE

Indicates when the cascade port meta-state is Active. This function is called by the function DECISION.

```
Function CASCADE_PORT_ACTIVE () : Boolean;

Begin
 If (CASCADE_PORT_META_STATE = ACTIVE)
  Then CASCADE_PORT_ACTIVE := True
  Else CASCADE_PORT_ACTIVE := False
End;
```

12.6.3.2 Function DECISION

12.6.3.2.1 General usage

The function DECISION is called only from the procedure DECIDE_AND_ACT. It returns one of the values of NO_REQUESTS_RECEIVED, SELECT_PORT, PASS_UP_CONTROL, or PASS_UP_ CONTROL_SECONDARY_CTL_SIG_RETURN. The function first calls HIGHEST_PRIORITY_ REQUEST_RECEIVING to determine the highest priority request that is being received at the local ports.

a) If no requests are being received, the HIGH_PRIORITY_POINTER and NORMAL_ PRIORITY_POINTER are both reset to the end of their cycles. DECISION then returns NO_REQUESTS_RECEIVED.
b) If one or more requests are being received at local ports, DECISION will either cause the repeater to remain in control and select a local port from which to repeat the next frame, or will cause the repeater to pass up control. When control is to be passed up, the function will also indicate whether the secondary control signal is to be RETURN.

The information used in making the decision is:

— Whether or not there is an active cascade port,
— The value of the RMAC_ENABLE_HIGH_ONLY_RECEIVED_REG flag,
— The value of CSTATE (the current round-robin cycle state), and
— The high-priority and normal-priority requests present.

Note that the variable CSTATE is set in the function SELECT_NEXT_PORT_AND_GRANT, and is interrogated only by the function DECISION.]

12.6.3.2.2 Decision to select a port

When requests are being received, DECISION returns SELECT_PORT if:

a) There is no active cascade port,
b) The RMAC_ENABLE_HIGH_ONLY_RECEIVED_REG is set to True and there are more high-priority requests in the current (high-priority) cycle, or
c) RMAC_ENABLE_HIGH_ONLY_RECEIVED_REG is set to False, and:
    1) CSTATE is C_HIGH_WAS_NORMAL_UNFINISHED and there are more high-priority - requests in the current (high-priority) cycle, or
    2) CSTATE is C_HIGH and there are high-priority requests being received, or

3) CSTATE is C_NORMAL and there are additional high-priority requests in the current cycle (if there are more normal-priority requests in the normal-priority cycle, CSTATE will be changed to C_HIGH_WAS_NORMAL_UNFINISHED), or
4) CSTATE is C_NORMAL, there are no high-priority requests being received, and there are more normal-priority requests in the current (normal-priority) cycle.

In each case, SELECT_NEXT_PORT_AND_GRANT will be called by DECIDE_AND_ACT to select the port.

12.6.3.2.3 Decision to pass up control

When there are requests being received, DECISION returns PASS_UP_CONTROL if:

a) CSTATE is C_HIGH and there are no more high-priority requests in the current high-priority cycle, or
b) CSTATE is C_NORMAL, there is an active cascade port, there are no more normal-priority requests in the current (normal-priority) cycle, and either:
    1) RMAC_ENABLE_HIGH_ONLY_RECEIVED_REG is set to True and there are no more high-priority requests in the high-priority cycle, or
    2) RMAC_ENABLE_HIGH_ONLY_RECEIVED_REG is set to False and there are no high-priority requests being received.

In the CSTATE is C_HIGH case, the HIGH_PRIORITY_POINTER is reset to the end of its cycle. In both CSTATE is C_NORMAL cases, the NORMAL_PRIORITY_POINTER is reset to the end of its cycle.

12.6.3.2.4 Decision to pass up control with secondary control signal RETURN

When there are requests being received, DECISION returns PASS_UP_CONTROL_SECONDARY_ CTL_SIG_RETURN if:

a) CSTATE is C_HIGH_WAS_NORMAL_UNFINISHED, there is an active cascade port, and there are no more high-priority requests in the current (high-priority) cycle, or
b) CSTATE is C_NORMAL, the RMAC_ENABLE_HIGH_ONLY_RECEIVED_REG is set to True, there are no more high-priority requests in the high-priority cycle, and there are more normal-priority requests in the current normal-priority cycle.

```
Function DECISION () : T_DECISION; L_PRIORITY: T_PRIORITY;

Begin
  L_PRIORITY := HIGHEST_PRIORITY_REQUEST_RECEIVING();
  If (L_PRIORITY = NONE) Then
  Begin
    DECISION := NO_REQUESTS_RECEIVED;
    HIGH_PRIORITY_POINTER := NUMBER_OF_PORTS;
    If (CSTATE = C_NORMAL) Then
      NORMAL_PRIORITY_POINTER := NUMBER_OF_PORTS;
  End
  Else  {Receiving some request(s)}
  If (RMAC_ENABLE_HIGH_ONLY_RECEIVED_REG = True) Then
  Begin
    If (MORE_REQUESTS_IN_CYCLE(HIGH_PRIORITY) = True) Then
      DECISION := SELECT_PORT
  Else
   Begin
   HIGH_PRIORITY_POINTER := NUMBER_OF_PORTS;
      Case CSTATE Of
        C_HIGH : DECISION := PASS_UP_CONTROL;
        C_HIGH_WAS_NORMAL_UNFINISHED :
          DECISION := PASS_UP_CONTROL_SECONDARY_CTL_SIG_RETURN;
        C_NORMAL :
        If (MORE_REQUESTS_IN_CYCLE(NORMAL_PRIORITY) = True) Then
          DECISION := PASS_UP_CONTROL_SECONDARY_CTL_SIG_RETURN
          Else Begin
            NORMAL_PRIORITY_POINTER := NUMBER_OF_PORTS;
            DECISION := PASS_UP_CONTROL;
          End;
      End {Case}
    End
 End
Else { ENABLE_HIGH_ONLY was not received.}
If (CASCADE_PORT_ACTIVE()= False) Then {There is no cascade port:}
  DECISION := SELECT_PORT      {Just select the next local port in the cycle.}
Else {There is a cascade port.}
Case CSTATE Of
  C_HIGH :
  If (MORE_REQUESTS_IN_CYCLE(HIGH_PRIORITY) = True) Then
    DECISION := SELECT_PORT
  Else Begin
    HIGH_PRIORITY_POINTER := NUMBER_OF_PORTS;
    DECISION := PASS_UP_CONTROL;
  End;

  C_HIGH_WAS_NORMAL_UNFINISHED :
  If (MORE_REQUESTS_IN_CYCLE(HIGH_PRIORITY) = True) Then
    DECISION := SELECT_PORT
  Else Begin
   DECISION := PASS_UP_CONTROL_SECONDARY_CTL_SIG_RETURN;
      {Because the normal-priority cycle was interrupted. }
       HIGH_PRIORITY_POINTER := NUMBER_OF_PORTS;
    End;


  C_NORMAL :
  If (L_PRIORITY = HIGH_PRIORITY) Then
    {The normal-priority cycle is being interrupted by a local high-priority request.}
    DECISION := SELECT_PORT
  Else  {L_PRIORITY must be NORMAL_PRIORITY}
  If (MORE_REQUESTS_IN_CYCLE(NORMAL_PRIORITY) = False) Then
      {There are no more normal-priority requests in this cycle.}
  Begin
    NORMAL_PRIORITY_POINTER := NUMBER_OF_PORTS;
    DECISION := PASS_UP_CONTROL {normal Pass-up control}
  End
  Else DECISION := SELECT_PORT;
End {Case}
```

```
  {PASS_UP_CONTROL_SECONDARY_CTL_SIG_RETURN indicates that RETURN should
   be sent since there are more normal-priority requests in the current cycle.}
End;
```

12.6.3.3 Function SELECT_NEXT_PORT

This function is called by the function SELECT_NEXT_PORT_AND_GRANT to decide which port to select next. If at least one request exists, then the high- or normal-priority pointer points to the selected port, depending on the priority level of the request.

Function SELECT_NEXT_PORT returns HIGH_PRIORITY if a high-priority request is pending, NORMAL_PRIORITY if a normal request is pending, and NONE if no requests are pending. OUTPORT returns the value of the port selected.

```
Function SELECT_NEXT_PORT (VAR OUT_PORT: T_LOCAL_PORT) : T_PRIORITY;

L_PORT: 1..(NUMBER_OF_PORTS + 1);
L_EXISTS_HP_REQ, L_EXISTS_NP_REQ: Boolean;

Begin
 L_EXISTS_HP_REQ := False;
 L_EXISTS_NP_REQ := False;
 { RMAC_REQ_REG reflects Requests and Secondary Control Signal received—it is
   updated in the Port Process Module, RX7, RX10.}
 L_PORT := HIGH_PRIORITY_POINTER + 1;
 Repeat
     If (L_PORT = NUMBER_OF_PORTS + 1) Then
       L_PORT := 1;
     If (((LOCAL_PORT_META_STATE[L_PORT] = ACTIVE) Or
          (LOCAL_PORT_META_STATE[L_PORT] = TRAINING))
        And (RMAC_REQ_REG[L_PORT] = HIGH_PRIORITY)) Then
     L_EXISTS_HP_REQ := True;
     L_PORT := L_PORT + 1;
 Until (L_EXISTS_HP_REQ = True) Or
     (L_PORT = HIGH_PRIORITY_POINTER + 1);
 L_PORT := L_PORT - 1;
 If (L_EXISTS_HP_REQ = True) Then
   HIGH_PRIORITY_POINTER := L_PORT
 Else
 Begin
   L_PORT := NORMAL_PRIORITY_POINTER + 1;
   Repeat
       If (L_PORT = NUMBER_OF_PORTS + 1) Then L_PORT := 1;
       If (((LOCAL_PORT_META_STATE[L_PORT] = ACTIVE) Or
            (LOCAL_PORT_META_STATE[L_PORT] = TRAINING))
          And (RMAC_REQ_REG[L_PORT] = NORMAL_PRIORITY)) Then
         L_EXISTS_NP_REQ := True;
       L_PORT := L_PORT + 1;
   Until ((L_EXISTS_NP_REQ = True) Or
       (L_PORT = NORMAL_PRIORITY_POINTER + 1));
   L_PORT := L_PORT - 1;
   If (L_EXISTS_NP_REQ = True) Then
     NORMAL_PRIORITY_POINTER := L_PORT;
```

```
  End;
  OUT_PORT := L_PORT;
  If (L_EXISTS_HP_REQ = True) Then SELECT_NEXT_PORT := HIGH_PRIORITY
  Else If ( L_EXISTS_NP_REQ = True) Then SELECT_NEXT_PORT := NORMAL_PRIORITY
  Else SELECT_NEXT_PORT := NONE;

End;
```

12.6.3.4 Function SELECT_NEXT_PORT_AND_GRANT : T_PRIORITY

This function carries out the selection of the next local port, and usually sends Grant. However, if the selected port was the destination of the previous frame, Grant is not sent immediately; a burst of IDLE_D is sent first. In this latter case, the SEND_IDLE_BURST timer is set here for the IDLE_D to be replaced by Grant.

```
Function SELECT_NEXT_PORT_AND_GRANT : T_PRIORITY.
L_PRIORITY: T_PRIORITY;
L_PORT: T_LOCAL_PORT;

Begin
  L_PRIORITY := SELECT_NEXT_PORT(L_PORT);
  {L_PORT is assigned the value of next selected port}
  If (L_PRIORITY <> NONE) Then
  Begin
    JUST_SELECTED_PORT := True;
    If (JUST_BEEN_SELECTED = True) Then
    Begin
      {This logic is executed only if this repeater has just been selected by
       an upper repeater. Hence the use of JUST_BEEN_SELECTED.}
      If (L_PRIORITY = HIGH_PRIORITY) Then CSTATE := C_HIGH
      Else CSTATE := C_NORMAL;
      JUST_BEEN_SELECTED := False;
    End
    Else
    If ((L_PRIORITY = HIGH_PRIORITY) And (CSTATE = C_NORMAL)) Then
    Begin
      If ((MORE_REQUESTS_IN_CYCLE(NORMAL_PRIORITY) = True) Or
         (PREV_NORMAL_PRIORITY_CYCLE_UNFINISHED = True)) Then
        CSTATE := C_HIGH_WAS_NORMAL_UNFINISHED
      Else Begin
        NORMAL_PRIORITY_POINTER := NUMBER_OF_PORTS;
        CSTATE := C_HIGH
        End
          {normal-priority cycle finished}
    End;
    If (CSTATE = C_NORMAL) And (MORE_REQUESTS_IN_CYCLE(NORMAL_PRIORITY) =
True)
      Then PREV_NORMAL_PRIORITY_CYCLE_UNFINISHED := True Else
         PREV_NORMAL_PRIORITY_CYCLE_UNFINISHED := False;
    SELECTED_PORT := L_PORT;
    SELECTED_PORT_PRIORITY := L_PRIORITY;
    If (ADDRESS_MATCH[L_PORT] = True) Then
    Begin
```

```
      {Selecting previous destination. Want to allow control detect
       to determine end of packet.}
      CHANGE_LOCAL_PORT_CONTROL_SIGNAL(L_PORT,IDLE_D);
      SET_TIMER(SEND_IDLE_BURST);
    End
    Else
      CHANGE_LOCAL_PORT_CONTROL_SIGNAL(L_PORT,GRANT);
    SIGNAL_TO_FCP(S_TO_FCP_EXPECT_FRAME);
    {FCP is responsible for asserting and deasserting RXENABLE}
    SIGNAL_TO_TSM(S_TO_TSM_GOTO_TX_INCOMING);
  End;
  SELECT_NEXT_PORT_AND_GRANT := L_PRIORITY;
End;
```

12.6.3.5 Function SEND_HIGHEST_PRIORITY_REQUEST_OR_IDLE

This function is called by the RSM to determine the highest priority Request, if any, at a local port and reflects either that level of Request or Idle_Up to the Cascade port. It returns the highest priority level (high, normal, or none) as the function value.

```
Function SEND_HIGHEST_PRIORITY_REQUEST_OR_IDLE
L_PRIORITY: T_PRIORITY;

Begin
  L_PRIORITY := HIGHEST_PRIORITY_REQUEST_RECEIVING();
  Case L_PRIORITY Of
    HIGH_PRIORITY :
      CHANGE_CASCADE_PORT_CONTROL_SIGNAL(REQ_H);
    NORMAL_PRIORITY:
      CHANGE_CASCADE_PORT_CONTROL_SIGNAL(REQ_N);
    NONE :
      CHANGE_CASCADE_PORT_CONTROL_SIGNAL(IDLE_U);
  End; {Case}
  SEND_HIGHEST_PRIORITY_REQUEST_OR_IDLE := L_PRIORITY;
End;
```

12.6.4 Procedure definitions

12.6.4.1 Procedure DECIDE_AND_ACT

12.6.4.1.1 General usage

The procedure DECIDE_AND_ACT is normally called from RX10 on completion of the transmission of a frame being repeated from below. DECIDE_AND_ACT is called in order to decide which of three subsequent actions is appropriate for the repeater, and to perform the RSM state transitions and associated processing. The possible actions for the repeater are:

a) To remain in control and select the next local port (state RX7),
b) To pass up control to a higher-level repeater (state RX9), or
c) To remain in control in the idle state (state RX1) if there is no active higher-level repeater.

In case b), DECIDE_AND_ACT also indicates whether or not the secondary control signal sent through the cascade port is to be set to RETURN, following the IDLE_U that is sent to pass up control. In usual operation, the choice of a), b) or c) is made by calling DECISION (see 12.6.4.1.4). However, there are two exception conditions, resulting in actions b) or c), that are checked first (see 12.6.4.1.2 and 12.6.4.1.3).

12.6.4.1.2 Conflict of control

If FLAG_CONTROL_ACTUALLY_ABOVE is set to True, the repeater has detected that control of the network has temporarily ceased to be at a single point. The repeater had assumed that it, or a lower-level repeater, had control, since it has just been repeating a frame from below. However, IDLE_D or INCOMING has been received on the cascade port, indicating that a higher-level repeater considers that it has control of the network. The repeater passes up control to the higher level by sending IDLE_U on the cascade port, and the RSM goes to state RX9.

12.6.4.1.3 Uplink training requested

If the value of LOWER_PORTS_TRAFFIC is HALTED, this indicates that the TULM wants to halt traffic through the local ports so that training of the uplink can begin. The RSM halts the lower-port traffic when the HIGH_PRIORITY_POINTER and NORMAL_PRIORITY_POINTER both indicate that the respective round-robin cycles are completed, and signals the TULM with S_RSM_TO_TULM_TRAFFIC_HALTED. The RSM goes to the idle state RX1 until the TULM signals that the cascade port training attempt has been completed.

12.6.4.1.4 Usual operation

If neither of the exception conditions in 12.6.4.1.2 and 12.6.4.1.3 applies, the function DECISION is called, and it returns one of four values which determines the subsequent action:

a) NO_REQUESTS_RECEIVED: If there is an active higher-level repeater, the repeater passes up control by sending IDLE_U on the Cascade Port and goes to state RX9; otherwise, it goes to the idle state, RX1.

b) SELECT_PORT: The function SELECT_NEXT_PORT_AND_GRANT is called to select the port and initiate the repeat-from-below sequence; the requested priority is signaled on the cascade port if that port is active, and the RSM goes to state RX7.

c) PASS_UP_CONTROL or PASS_UP_CONTROL_SECONDARY_CTL_SIG_RETURN: The repeater passes up control by sending IDLE_U on the cascade port, and goes to state RX9; the flag SECONDARY_CONTROL_SIGNAL_IS_RETURN is set according to the value returned by the function DECISION (see 12.6.3.2.3 and 12.6.3.2.4). The timer PASS_UP_CONTROL_IDLE is started, which determines how long IDLE_U is sent before initiating the secondary control signal. The fault timer PASS_UP_TO_IDLE_DOWN_TIMEOUT is also started: this expires if IDLE_D is not received in reply to IDLE_U, and is handled as an error condition, in RX11.

```
Procedure DECIDE_AND_ACT ();

L_DECISION: T_DECISION;
L_PRIORITY: T_PRIORITY;

Begin
  PASS_UP_ATTEMPTS := 1; {To count number of attempts of pass up handshake.}
  If ((FLAG_CONTROL_ACTUALLY_ABOVE = True) And
     (CASCADE_PORT_ACTIVE()= True)) Then
  Begin {FLAG_CONTROL_ACTUALLY_ABOVE has been set to True. Assumed control had
```

```
     been passed down, but repeater above thinks IT has control. Pass up control now.}
  CHANGE_CASCADE_PORT_CONTROL_SIGNAL(IDLE_U);
  SET_TIMER(PASS_UP_CONTROL_IDLE);
  SET_FAULT_TIMER(PASS_UP_TO_IDLE_DOWN_TIMEOUT);
  RXSTATE := RX9_PASS_UP_GRANT;
End

Else If ((LOWER_PORTS_TRAFFIC = HALTED) And
        (HIGH_PRIORITY_POINTER = NUMBER_OF_PORTS) And
        (NORMAL_PRIORITY_POINTER = NUMBER_OF_PORTS)) Then
Begin {Have been asked by TULM to halt activity via
     LOWER_PORTS_TRAFFIC being HALTED. Now can halt network.}
  SIGNAL_TO_TULM(S_RSM_TO_TULM_TRAFFIC_HALTED);
  {To let TULM know network activity halted.}
  RXSTATE := RX1_IDLE;
End
Else
Begin {Usual situation}
  L_DECISION := DECISION();
  Case L_DECISION Of
    NO_REQUESTS_RECEIVED :
    If (CASCADE_PORT_ACTIVE()= True) Then
    Begin
      CHANGE_CASCADE_PORT_CONTROL_SIGNAL(IDLE_U);
      SET_TIMER(PASS_UP_CONTROL_IDLE);
      SET_FAULT_TIMER(PASS_UP_TO_IDLE_DOWN_TIMEOUT);
      RXSTATE := RX9_PASS_UP_GRANT;
    End
    Else RXSTATE := RX1_IDLE;

    SELECT_PORT :
    Begin
      L_PRIORITY := SELECT_NEXT_PORT_AND_GRANT();
      {Within SELECT_NEXT_PORT_AND_GRANT, signal to TSM to send INCOMING}
      If (CASCADE_PORT_ACTIVE()= True) Then
      Begin
        If (L_PRIORITY = HIGH_PRIORITY) Then
          CHANGE_CASCADE_PORT_CONTROL_SIGNAL(REQ_H)
        Else If (L_PRIORITY = NORMAL_PRIORITY) Then
          CHANGE_CASCADE_PORT_CONTROL_SIGNAL(REQ_N);
      End;
      RXSTATE := RX7_EXPECT_FRAME_FROM_BELOW;
    End;

    PASS_UP_CONTROL, PASS_UP_CONTROL_SECONDARY_CTL_SIG_RETURN :
    Begin {Pass up control}
      SELECTED_PORT := CASCADE_PORT;
      If (L_DECISION = PASS_UP_CONTROL) Then
        SECONDARY_CONTROL_SIGNAL_IS_RETURN := False;
      Else If (L_DECISION = PASS_UP_CONTROL_SECONDARY_CTL_SIG_RETURN) Then
        SECONDARY_CONTROL_SIGNAL_IS_RETURN := True;
      CHANGE_CASCADE_PORT_CONTROL_SIGNAL(IDLE_U);
      SET_TIMER(PASS_UP_CONTROL_IDLE);
```

```
        SET_FAULT_TIMER(PASS_UP_TO_IDLE_DOWN_TIMEOUT);
        RXSTATE := RX9_PASS_UP_GRANT;
      End;
    End {Case}
  End {Usual situation}
End;
```

12.6.4.2 Procedure DECIDE_WHETHER_TO_SEND_ENABLE_HIGH_ONLY

This procedure is called by the RSM to decide whether or not ENABLE_HIGH_ONLY should be sent to the lower repeater. Execution is null if the SELECTED_PORT is an end node.

```
Procedure DECIDE_WHETHER_TO_SEND_ENABLE_HIGH_ONLY ();

Begin
  If (ENTITY_AT_FAR_END_OF_LINK[SELECTED_PORT] = LOWER_REPEATER) Then
  Begin
    If (((RMAC_ENABLE_HIGH_ONLY_RECEIVED_REG = True) And
          (CASCADE_PORT_ACTIVE())) Or
          (HIGHEST_PRIORITY_REQUEST_AT_OTHER_PORTS(SELECTED_PORT) =
                HIGH_PRIORITY))   Then
          CHANGE_LOCAL_PORT_CONTROL_SIGNAL(SELECTED_PORT,
              ENABLE_HIGH_ONLY)
    Else CHANGE_LOCAL_PORT_CONTROL_SIGNAL(SELECTED_PORT, GRANT)
  End

  {NOTE—in Half Duplex systems, ENABLE_HIGH_ONLY is taken away from downlink when
              Receive Enable is pulled—this happens in the FCP.}
End;
```

12.6.4.3 Procedure INITIALIZE_RMAC

This procedure initializes the RMAC at power on.

```
Procedure INITIALIZE_RMAC ();

C_PORT : T_LOCAL_PORT;

Begin
  If (PORT_STATE[CASCADE_PORT] = ENABLED) Then
  Begin
    CASCADE_PORT_META_STATE := TRAINING;
    CHANGE_CASCADE_PORT_CONTROL_SIGNAL(TRAINING_UP);
    TOTAL_TRAINING_PACKETS_DONE := 0; {TULM Counter}
    SUCCESSFUL_TRAINING_PACKETS_IN_RUN := 0;
    SET_TIMER(TRAINING_PULSE_SENDING);
  End
  Else CASCADE_PORT_META_STATE := INACTIVE;
  TULMSTATE := TULM2_WAIT_FOR_TRAINING_DOWN;
  RXSTATE := RX1_IDLE;
  TXSTATE := TX2_SEND_IDLE;
  FLAG_GO_TO_TX_INCOMING := True;
  CSTATE := C_NORMAL;
  FCPSTATE := FCPSTATE1_QUIESCENT;
  HIGH_PRIORITY_POINTER := NUMBER_OF_PORTS;
  NORMAL_PRIORITY_POINTER := NUMBER_OF_PORTS;
  FLAG_CONTROL_ACTUALLY_ABOVE := False;
  FLAG_SIGNAL_RECEIVED_EXPECT_FRAME := False;
  SECONDARY_CONTROL_SIGNAL_IS_RETURN := False;
  FLAG_REQ_WINDOW_EXPIRED := True;
  FLAG_IPG_WINDOW_EXPIRED := True;
  RMAC_ENABLE_HIGH_ONLY_RECEIVED_REG := False;
  LOWER_PORTS_TRAFFIC := OPERATING;
  FCP_DATA_OUTPUT := NIL;
  RMAC_BUF_DATA_INPUT := STOP_INPUT;
  FLAG_GRANT_STATE_ZERO := False;
  PREV_NORMAL_PRIORITY_CYCLE_UNFINISHED := False;
  SILENT_WHEN_GRANTED := False;
  RECEIVING_IDLE:= False;
  For C_PORT := 1 To NUMBER_OF_PORTS Do
  Begin
    RMAC_REQ_REG[C_PORT] := NONE;
    LOCAL_PORT_META_STATE[C_PORT] := INACTIVE;
    LOCAL_PORT_PACKETS_OK_SEQUENCE[C_PORT] := 0;
  End;
End;
```

12.6.4.4 Procedure INVOKE_TRAINING

This procedure is called by the RSM and FCP to force retraining of a downlink if a link fault has been detected at a local port. The lower entity will detect the disabled link and initiate training.

```
Procedure INVOKE_TRAINING (IN_PORT: T_LOCAL_PORT);

Begin
  LOCAL_PORT_META_STATE[IN_PORT] := INACTIVE;
  CHANGE_LOCAL_PORT_CONTROL_SIGNAL(IN_PORT, TXDISABLE);
End;
```

12.6.4.5 Procedure INVOKE_CASCADE_PORT_TRAINING

This procedure is called by the RSM and FCP to initiate training of the uplink and to force retraining if a link fault has been detected at the cascade port.

```
Procedure INVOKE_CASCADE_PORT_TRAINING ();.

Begin
  CASCADE_PORT_META_STATE := TRAINING;
  SIGNAL_TO_TULM(S_TO_TULM_RETRAIN_UPLINK);
End;
```

12.7 Receive State Machine (RSM)

The RSM provides master control over the packet sequencing process. It controls the round-robin port-selection cycle and coordinates the activities of the other state machines. It communicates with the TSM, TULM, BCALL, FCP, and each PPM. The relationship of the RSM to the RMAC state machines is shown in Figure 12-17.

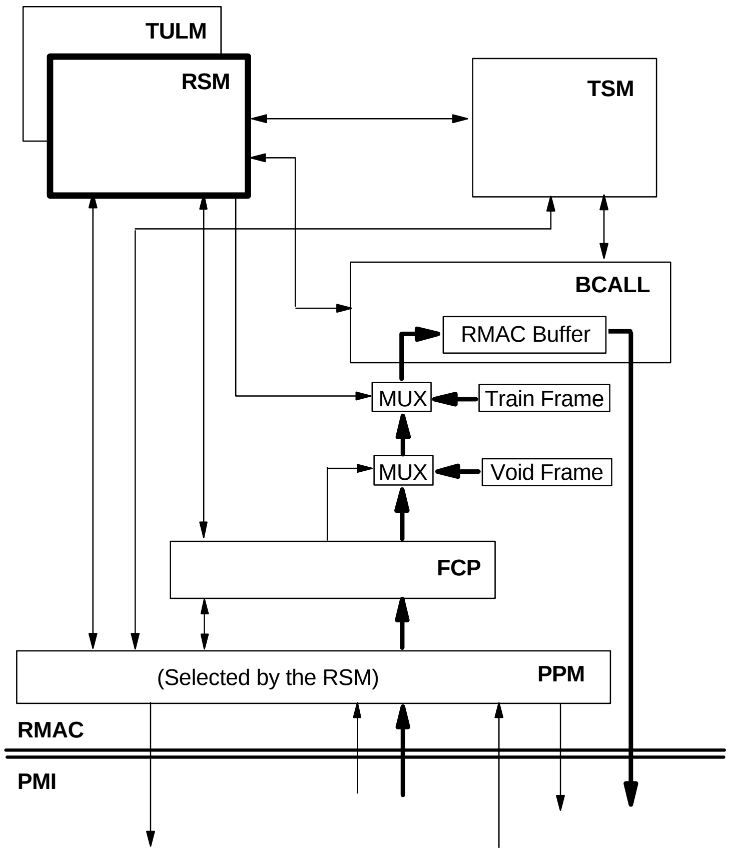


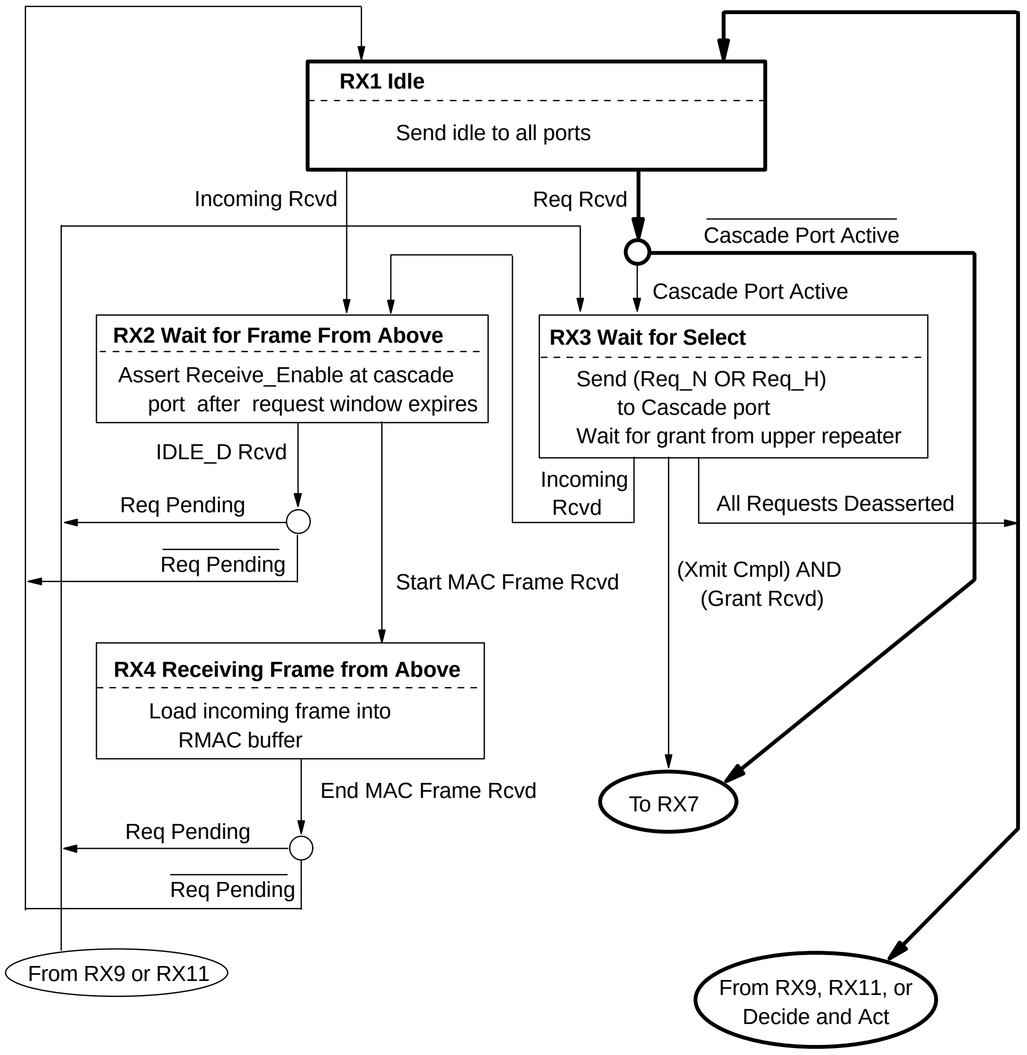
Figure 12-17—Relationship of the RSM to the RMAC state machines

The RSM is diagrammed in Figures 12-18 through 12-20. It has ten defined states, RX1 through RX4, and RX6 through RX11. RX5 was merged into RX3 during development.

a) The states and transitions shown in bold (RX 1, 7, 8, and 10) are the states that are required to support an RMAC in a single repeater network.

b) In addition to the minimum capability, State 6 and its associated transitions to States 7 and 10 are required to support connection of a lower repeater to a local port.

c) In addition to the minimum capability, States 2, 3, 4, 9, and 11 are required to support connection to an upper repeater through the cascade port.


RX1 Idle
Send idle to all ports
Incoming Rcvd
Req Rcvd
Cascade Port Active
Cascade Port Active
RX2 Wait for Frame From Above
Assert Receive_Enable at cascade port after request window expires
RX3 Wait for Select
Send (Req_N OR Req_H) to Cascade port
Wait for grant from upper repeater
IDLE_D Rcvd
Incoming Rcvd
All Requests Deasserted
Req Pending
Req Pending
Start MAC Frame Rcvd
(Xmit Cmpl) AND (Grant Rcvd)
RX4 Receiving Frame from Above
Load incoming frame into RMAC buffer
End MAC Frame Rcvd
To RX7
Req Pending
Req Pending
From RX9 or RX11
From RX9, RX11, or Decide and Act

Note—There is no RX5 in the RSM.

Figure 12-18—RMAC Receive State Machine—States 1–4

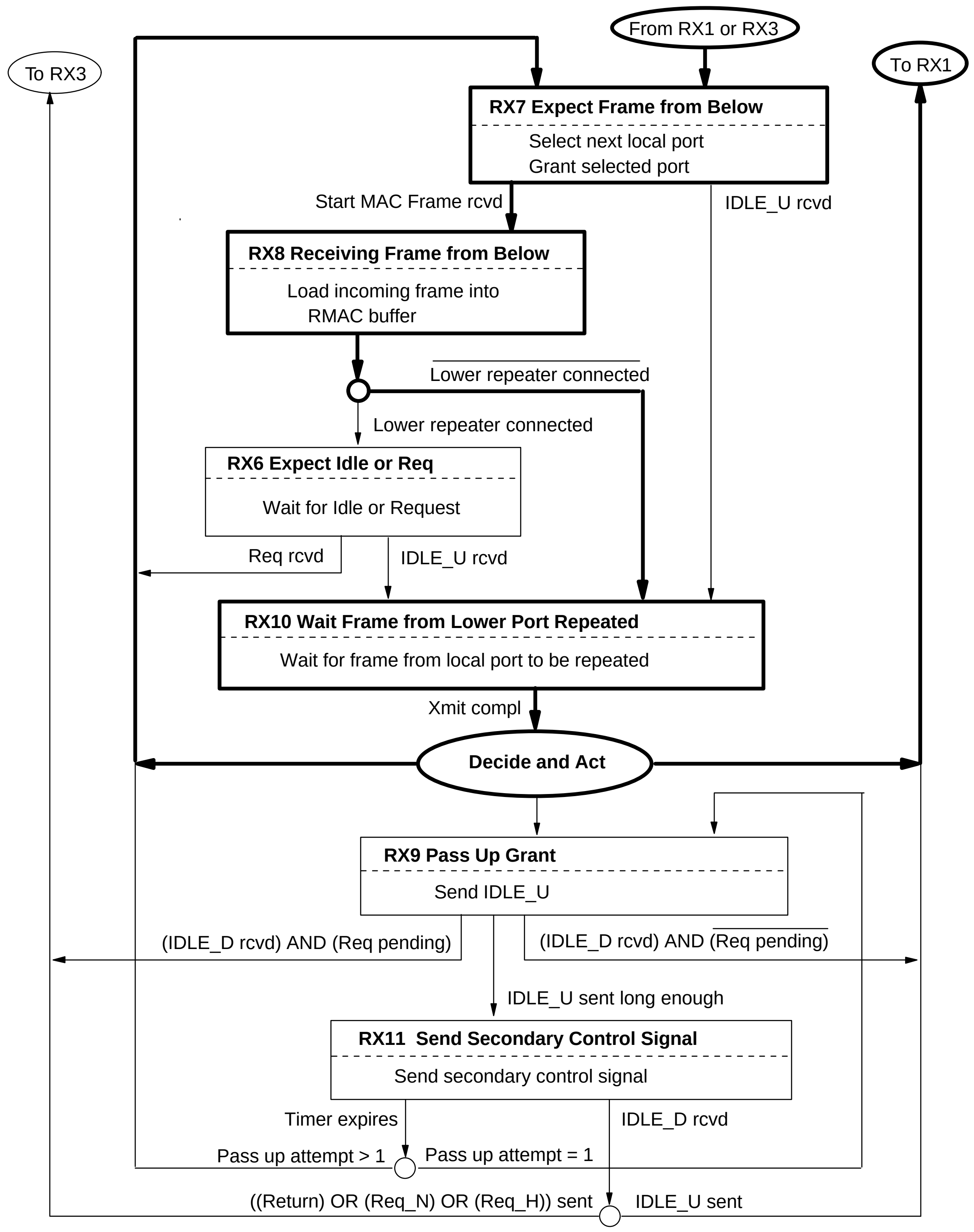
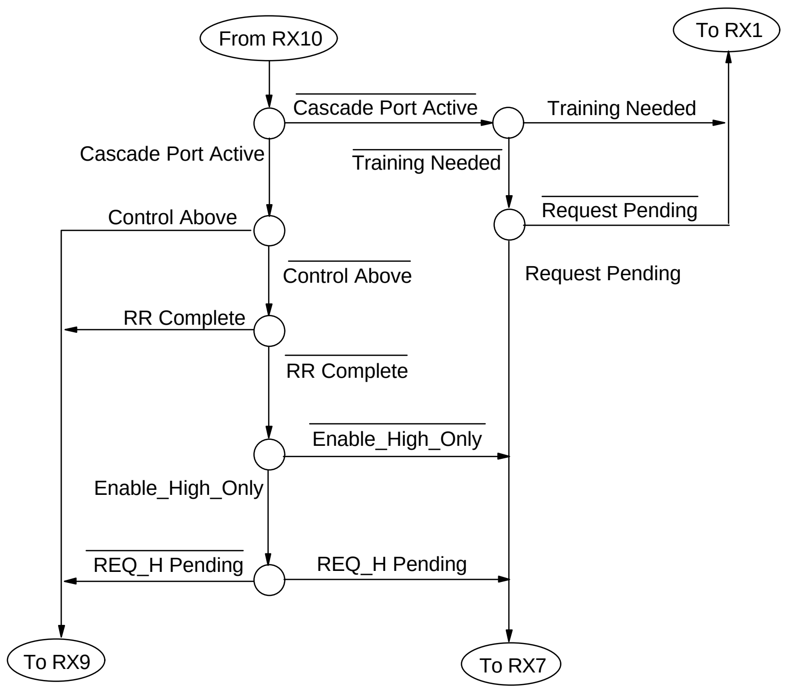


Figure 12-19—MAC Receive State Machine—States 6–11



RR Complete = Round Robin Complete

Figure 12-20—Decide_and_Act decision diagram

12.7.1 RSM local variables

The following variables are general to the RSM.

CAME_FROM_RX7 : Boolean; True indicates to RX10 code that the previous state was RX7, and False indicates to RX10 code that the previous state was not RX7.

FLAG_REQ_WINDOW_EXPIRED : Boolean; Flag set when the Request window has ended: i.e., when the RMAC has sent REQ_H or REQ_N through the cascade port for long enough to be recognized.

FLAG_CONTROL_ACTUALLY_ABOVE : Boolean; Flag set when a repeater is acting as if it has control, or control is below, but it discovers that the upper repeater thinks it has control.

The following variables are used by RX9, RX11, and RX3 when passing up control:

PREV_NORMAL_PRIORITY_CYCLE_UNFINISHED : Boolean; Indicates that normal priority was not completed.

SECONDARY_CONTROL_SIGNAL_IS_RETURN : Boolean; Set True / False in DECIDE_AND_ ACT to tell RX9 whether to send RETURN as the Secondary Control Signal on exit to RX11, to be held subsequently in RX3. Cleared (False) when the repeater is reselected following RETURN, or on other exits from RX3.

PASS_UP_ATTEMPTS : Integer; Used in RX11 to determine whether this is the first time the RSM has attempted to pass up control, or whether this is a subsequent attempt due to a previous attempt.

The following variables are for use when selecting a Local Port:

HIGH_PRIORITY_POINTER : T_LOCAL_PORT; High-priority round-robin pointer.

NORMAL_PRIORITY_POINTER : T_LOCAL_PORT; Normal-priority round-robin pointer.

CSTATE : T_CSTATE; Cycle-state, used in DECISION, to maintain normal-priority round-robin fairness.

BEEN_SELECTED : Boolean; Used in RX3 to indicate whether a repeater has been selected yet by the upper repeater.

JUST_BEEN_SELECTED : Boolean; True indicates that a repeater has just been selected by the repeater above it, and that control has just been passed down.

JUST_SELECTED_PORT : Boolean; True indicates that the repeater has just selected a port. Becomes False for the second and subsequent frames coming up from the selected port.

RECEIVING_IDLE : Boolean; Flag to indicate when receiving Idle.

12.7.2 RX1 : RX1_IDLE

The RSM can be in state RX1 in the following three circumstances:

a) If the Cascade Port is training, and the traffic through the Local Ports is suspended, then the variable LOWER_PORTS_TRAFFIC has been set to HALTED, and RX1 effectively acts as the RSM quiescent state while the TULM is training the Uplink.

b) If the Lower Ports traffic is operating (LOWER_PORTS_TRAFFIC = OPERATING), and the Cascade Port is not ACTIVE (CASCADE_PORT_ ACTIVE is False), then the repeater is effectively the root repeater, and being in RX1 means that there is no traffic on the network below this repeater.

c) If the Lower Ports traffic is operating, and the Cascade port is ACTIVE, then this means that there are no Requests being received at the Local Ports (no lower entities are requiring to transmit), and that the network control is effectively above the repeater (i.e., frames may be sent down by the repeater above, preceded by the INCOMING Control Signal).

If the repeater receives a Request (REQ_H or REQ_N), the LOWER_PORTS_TRAFFIC is OPERATING, and there is no repeater above (CASCADE_PORT_ACTIVE is False), then the RSM is able to acknowledge the requester by calling the function SELECT_NEXT_PORT_AND_GRANT, and transitioning to RX7. If the Request is received, LOWER_PORTS_TRAFFIC is OPERATING, but there is a repeater above, then the repeater sends up a Request reflecting the priority level of the request being received. RSM then transitions to RX3.

If INCOMING is received at the Cascade Port and LOWER_PORTS_TRAFFIC is OPERATING, then subject to the Request Window, S_TO_FCP_EXPECT_FRAME is sent to the FCP. (The FCP is responsible for asserting Receive Enable.) The signal S_TO_TSM_GOTO_TX_INCOMING is sent to the TSM, and the RSM then transitions to RX2.

If RSM is quiescent, and the TULM is now training the Uplink, LOWER_PORTS_TRAFFIC is HALTED. The RSM may receive S_TULM_TO_RSM_RESTART_TRAFFIC from the TSM. To restart the traffic:

- — If there are no Requests at Local Ports, the RSM waits in RX1.
- — If TULM training failed and there are local requests pending, the repeater selects the next lower entity.
- — If INCOMING is received at the cascade port, the RSM transitions to RX3.

```
Case EVENT_SORT Of
  LOCAL_PORT_CONTROL_SIGNAL_DECODED :
  Begin
    Case LOCAL_PORT_CONTROL_SIGNAL_IN Of
    REQ_H, REQ_N :
      If (LOWER_PORTS_TRAFFIC = OPERATING) Then
      Begin
        If (CASCADE_PORT_ACTIVE() = False) Then
        Begin {No repeater above. Select port immediately.}
          L_PRIORITY := SELECT_NEXT_PORT_AND_GRANT();
          RXSTATE := RX7_EXPECT_FRAME_FROM_BELOW;
        End
        Else Begin {Repeater above connected to cascade port. Send Request up.}
          CHANGE_CASCADE_PORT_CONTROL_SIGNAL(LOCAL_PORT_CONTROL_SIGNAL_IN);
          BEEN_SELECTED := False;
          SET_FAULT_TIMER(REQUESTING_TOO_LONG);
          RXSTATE := RX3_WAIT_FOR_SELECT;
        End
      End;
    End {Case}
  End;

  TIMER_EXPIRES:
    If (TIMER = REQ_WINDOW) Then
      FLAG_REQ_WINDOW_EXPIRED := True;

  CASCADE_PORT_CONTROL_SIGNAL_DECODED :
  If (CASCADE_PORT_CONTROL_SIGNAL_IN = INCOMING) Then
      {Incoming received. Prepare to receive frame through cascade port.}
    If (LOWER_PORTS_TRAFFIC = OPERATING) Then
    Begin
      SELECTED_PORT := CASCADE_PORT;
      {Pull receive enable, but subject to Request Window expiring.}
      If (FLAG_REQ_WINDOW_EXPIRED = True) Then
        SIGNAL_TO_FCP(S_TO_FCP_EXPECT_FRAME);
      SIGNAL_TO_TSM(S_TO_TSM_GOTO_TX_INCOMING);
      RXSTATE := RX2_WAIT_FOR_FRAME_FROM_ABOVE;
    End;

  S_TULM_TO_RSM_RESTART_TRAFFIC :
  Begin
    L_PRIORITY := HIGHEST_PRIORITY_REQUEST_RECEIVING();
    If (L_PRIORITY <> NONE) Then
    Begin
```

```
      If (CASCADE_PORT_ACTIVE() = False) Then
      Begin {No repeater above. Select port immediately.}
        L_PRIORITY := SELECT_NEXT_PORT_AND_GRANT();
        RXSTATE := RX7_EXPECT_FRAME_FROM_BELOW;
      End
      Else Begin {Repeater above connected to cascade port. Send Request up.}
        If (L_PRIORITY = HIGH_PRIORITY) Then
          CHANGE_CASCADE_PORT_CONTROL_SIGNAL(REQ_H)
        Else CHANGE_CASCADE_PORT_CONTROL_SIGNAL(REQ_N);
        BEEN_SELECTED := False;
        SET_FAULT_TIMER(REQUESTING_TOO_LONG);
        RXSTATE := RX3_WAIT_FOR_SELECT;
      End
    End {Else if L_PRIORITY is NONE just wait in RX1.}
  End;

  S_TULM_TO_RSM_HALT_TRAFFIC:
  SIGNAL_TO_TULM(S_RSM_TO_TULM_TRAFFIC_HALTED);
End;{Case}
```

12.7.3 RX2 : RX2_WAIT_FOR_FRAME_FROM_ABOVE

In RX2, the RSM has received INCOMING through the cascade port, and is expecting to receive a frame through the cascade port. Entry to RX2 is from either RX1 or RX3, on receiving INCOMING (see Figure 12-21). Exit from RX2 is normally to RX4, on receiving the start of a frame at the cascade port. Exit from RX2 can also be to RX1 or RX3, as described in c) below.

a) When REQ_WINDOW expires, the RSM sends S_TO_FCP_EXPECT_FRAME to the FCP. (The request Window is the minimum interval between the end of receiving one frame and the asserting of Receive Enable for a subsequent frame; this is the time during which the repeater can send a request through the cascade port. Receive Enable is asserted in FCP.)

b) If the S_FCP_SMF_RECEIVED_THRU_CASCADE_PORT is received, the RMAC_BUF_ DATA_INPUT control register is set so that the RMAC buffer receives its input octets from the FCP. The RSM goes to state RX4.

c) If the INCOMING signal on the cascade port is replaced by IDLE_D (indicating that the repeater above is not after all going to send the expected frame), the RSM sends S_TO_FCP_STOP_ EXPECTING_FRAME to FCP. If there are requests at the local ports, the RSM transitions to RX3; otherwise, it transitions to RX1.

d) If the Uplink is dual simplex, then the RSM could be requesting while in RX2. If a change in the request status occurs at the local ports, the function SEND_HIGHEST_PRIORITY_REQUEST_ OR_IDLE is called, to propagate the appropriate request or IDLE_U control signal through the cascade port.

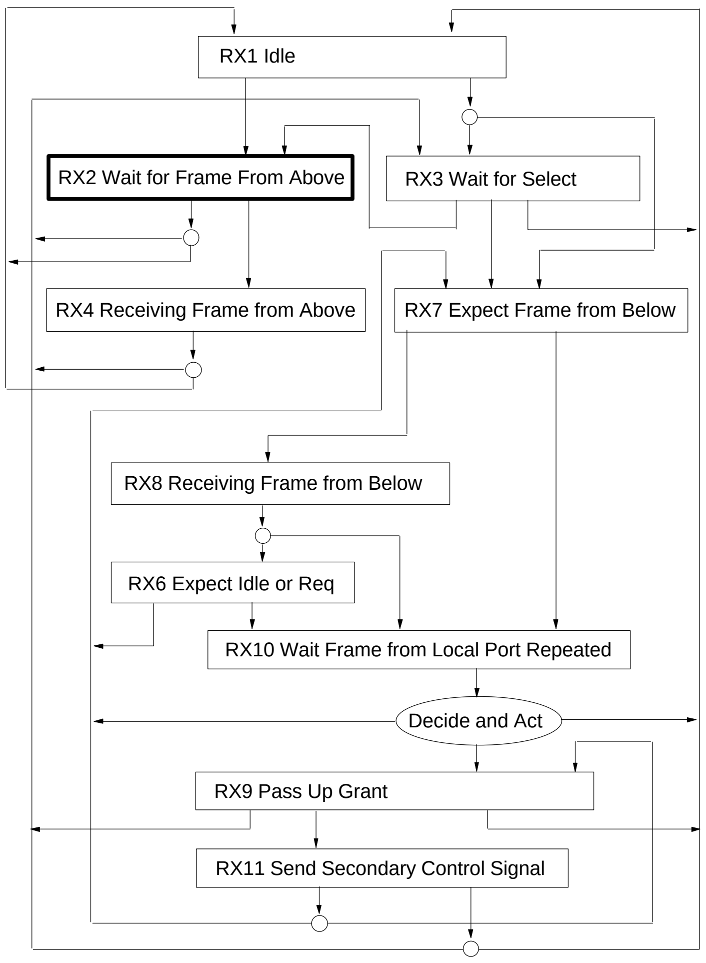

RX1 Idle
RX2 Wait for Frame From Above
RX3 Wait for Select
RX4 Receiving Frame from Above
RX7 Expect Frame from Below
RX8 Receiving Frame from Below
RX6 Expect Idle or Req
RX10 Wait Frame from Local Port Repeated
Decide and Act
RX9 Pass Up Grant
RX11 Send Secondary Control Signal

Figure 12-21—Relationship of RX2 within the state machine

```
Case EVENT_SORT Of
 TIMER_EXPIRES :
 If (TIMER = REQ_WINDOW) Then
   Begin
     FLAG_REQ_WINDOW_EXPIRED := True;
     SIGNAL_TO_FCP(S_TO_FCP_EXPECT_FRAME);
   End

 S_FCP_SMF_RECEIVED_THRU_CASCADE_PORT :
 Begin
       RMAC_BUF_DATA_INPUT := OCTETS_FROM_FCP;
       RXSTATE := RX4_RECEIVING_FRAME_FROM_ABOVE;
 End;

 CASCADE_PORT_CONTROL_SIGNAL_DECODED :
   If (CASCADE_PORT_CONTROL_SIGNAL_IN = IDLE_D) Then
   Begin
   {Repeater above had indicated that a frame was coming down but has
    changed its mind, and sent IDLE_D instead of the expected frame.}
     SIGNAL_TO_FCP(S_TO_FCP_STOP_EXPECTING_FRAME);
     SIGNAL_TO_TSM(S_TO_TSM_GOTO_TX_IDLE);
```

```
      Case SEND_HIGHEST_PRIORITY_REQUEST_OR_IDLE() Of
        HIGH_PRIORITY, NORMAL_PRIORITY :
        Begin
          BEEN_SELECTED := False;
          SET_FAULT_TIMER(REQUESTING_TOO_LONG);
          RXSTATE := RX3_WAIT_FOR_SELECT;
        End;

        NONE : RXSTATE := RX1_IDLE;
      End {Case}
    End;

  LOCAL_PORT_CONTROL_SIGNAL_DECODED :
  {Update Request level out of cascade port, as change in local port requests received
     may have occurred.}
    L_PRIORITY := SEND_HIGHEST_PRIORITY_REQUEST_OR_IDLE();

End;{Case}
```

12.7.4 RX3 : RX3_WAIT_FOR_SELECT

In RX3, the repeater is sending a request through the cascade port to the repeater above, as a result of a request(s) received on one or more local ports. Entry to RX3 is from RX1, RX2, RX4, RX9, or RX11 (see Figure 12-22). Exit is normally to RX7 when the request is granted by the upper repeater, or to RX2 if the upper repeater remains in control and sends “Incoming.” Exit from RX3 to RX1 can also occur if the requests on the local ports are all canceled and replaced by IDLE_U.

a) If there is a change in the requests being received at the local ports, the function SEND_ HIGHEST_PRIORITY_REQUEST_OR_IDLE is called, which causes the appropriate level of request (or IDLE_U) to be sent through the cascade port. If the function returns NONE (i.e., no requests are being received), the RSM transitions to RX1.

b) If GRANT is received through the cascade port, the repeater has been selected by the repeater above. A frame still may be propagating through the RMAC buffer, and in this case, a lower entity cannot be selected immediately. The only immediate action on receiving GRANT is to set the flag BEEN_SELECTED to True. After any event in RX3, if BEEN_SELECTED is True, and if either TXSTATE is not equal to TX1 (i.e., a frame is not currently being transmitted out of the RMAC buffer) or the event is SIGNAL_FROM_TSM_TO_RSM_EMF_TRANSMITTED (indicating that the frame being repeated has now been fully transmitted), the next lower entity is selected by calling the function SELECT_NEXT_PORT_AND_GRANT. The priority level of the control signal out of the cascade port is made to reflect the priority level of the selected lower entity, and the RSM transitions to RX7. (Exceptionally, it is possible for there to be no request remaining at this time; in this case, the transition is to RX1, with IDLE_U sent at the cascade port.)

c) If INCOMING is received through the cascade port, the RSM selects the cascade port, sends S_EXPECT_FRAME to the FCP and S_TO_TSM_GOTO_TX_INCOMING to the TSM, and then transitions to RX2 to await the start of the frame from above.

d) The timer REQUESTING_TOO_LONG detects if the RSM has been waiting too long in RX3 (waiting to be selected by the upper repeater). If this timer expires, the Uplink is retrained, and the RSM resumes the traffic on the local ports by calling SELECT_NEXT_PORT_AND_GRANT, with a transition to RX7 or RX1.

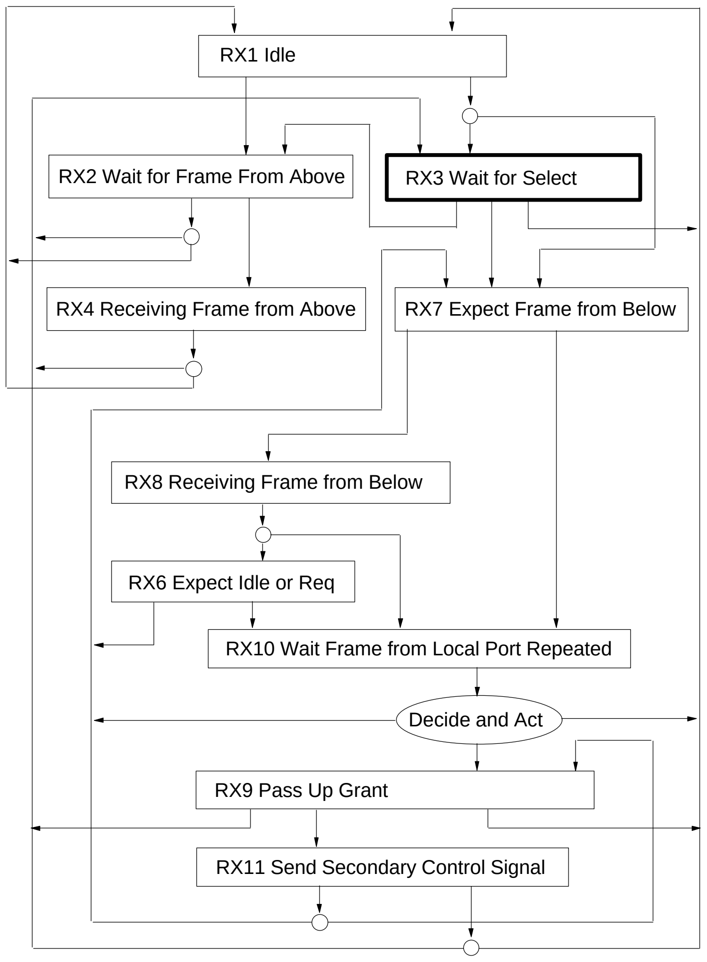

RX1 Idle
RX2 Wait for Frame From Above
RX3 Wait for Select
RX4 Receiving Frame from Above
RX7 Expect Frame from Below
RX8 Receiving Frame from Below
RX6 Expect Idle or Req
RX10 Wait Frame from Local Port Repeated
Decide and Act
RX9 Pass Up Grant
RX11 Send Secondary Control Signal

Figure 12-22—Relationship of RX3 within the state machine

```
Begin
  Case EVENT_SORT Of
    LOCAL_PORT_CONTROL_SIGNAL_DECODED :
    {New situation at port. Ignored if sending RETURN; otherwise,
      SEND_HIGHEST_PRIORITY_REQUEST_OR_IDLE is called to re-evaluate the situation
and
      send the appropriate control signal; if no requests are being received, go to RX1.}
    If (SECONDARY_CONTROL_SIGNAL_IS_RETURN = False) Then
      If (SEND_HIGHEST_PRIORITY_REQUEST_OR_IDLE() = NONE) Then
      Begin
        CANCEL_FAULT_TIMER(REQUESTING_TOO_LONG);
        RXSTATE := RX1_IDLE;
      End;

    CASCADE_PORT_CONTROL_SIGNAL_DECODED :
    Case CASCADE_PORT_CONTROL_SIGNAL_IN Of
      GRANT :
      Begin
      {Have been selected by repeater above. May have to wait while last frame
       is completely transmitted by TSM before selecting a lower entity.}
        BEEN_SELECTED := True;
        FLAG_CONTROL_ACTUALLY_ABOVE := False;

        **{Flags used when control sent to entities below, in RX7, etc.}**
      End;

      INCOMING :
      Begin
        **{Incoming received; prepare to receive frame.}**
        SECONDARY_CONTROL_SIGNAL_IS_RETURN := False;
        If (FLAG_REQ_WINDOW_EXPIRED = True) Then
          SIGNAL_TO_FCP(S_TO_FCP_EXPECT_FRAME);
        SIGNAL_TO_TSM(S_TO_TSM_GOTO_TX_INCOMING);
        SELECTED_PORT := CASCADE_PORT;
        CANCEL_FAULT_TIMER(REQUESTING_TOO_LONG);
        RXSTATE := RX2_WAIT_FOR_FRAME_FROM_ABOVE;
      End;
    End; **{Case}**

    TIMER_EXPIRES :
    Case TIMER Of
      REQUESTING_TOO_LONG :
      **{Have been requesting to upper repeater for too long.}**
      Begin
        INVOKE_CASCADE_PORT_TRAINING();
        FLAG_CONTROL_ACTUALLY_ABOVE := False;
        SECONDARY_CONTROL_SIGNAL_IS_RETURN := False;
        L_PRIORITY := SELECT_NEXT_PORT_AND_GRANT();
        FLAG_REQ_WINDOW_EXPIRED := True;
        If (L_PRIORITY = NONE) Then RXSTATE := RX1_IDLE
        Else RXSTATE := RX7_EXPECT_FRAME_FROM_BELOW;
      End;

      REQ_WINDOW: FLAG_REQ_WINDOW_EXPIRED := True;
    End; **{Case}**
  End; **{Case}**

  If (((EVENT_SORT = S_TSM_TO_RSM_EMF_XMITTED) Or
        (TXSTATE <> TX1_SEND_FRAME))  And
           (BEEN_SELECTED = True) ) Then
  Begin
    JUST_BEEN_SELECTED := True;
    SECONDARY_CONTROL_SIGNAL_IS_RETURN := False;
    L_PRIORITY := SELECT_NEXT_PORT_AND_GRANT();
    If (L_PRIORITY <> NONE) Then
    Begin
      **{Now, have been selected by upper repeater, and TSM is not (any longer)
       sending out last frame.}**
      RMAC_ENABLE_HIGH_ONLY_RECEIVED_REG := False;
      If ((L_PRIORITY = HIGH_PRIORITY) And
           (CASCADE_PORT_OUT_REG <> REQ_H)) Then
        CHANGE_CASCADE_PORT_CONTROL_SIGNAL(REQ_H)
      Else If ((L_PRIORITY = NORMAL_PRIORITY) And
             (CASCADE_PORT_OUT_REG <> REQ_N)) Then
        CHANGE_CASCADE_PORT_CONTROL_SIGNAL(REQ_N);
       {Must tell upper repeater if selecting high-priority or normal-priority frame, if this has
        changed.}
     CANCEL_FAULT_TIMER(REQUESTING_TOO_LONG);
     FLAG_REQ_WINDOW_EXPIRED := True;
     RXSTATE := RX7_EXPECT_FRAME_FROM_BELOW;
    End
    Else
    Begin {Unusual circumstance. Entity previously requesting has now stopped requesting.}
     CHANGE_CASCADE_PORT_CONTROL_SIGNAL(IDLE_U);
     CANCEL_FAULT_TIMER(REQUESTING_TOO_LONG);
     RXSTATE := RX1_IDLE;
    End
   End
  End;
```

12.7.5 RX4 : RX4_RECEIVING_FRAME_FROM_ABOVE

When in RX4, the RSM is receiving a frame through the cascade port. Entry to RX4 is always from RX2, on receipt of the start of frame. Exit from RX4 is to RX1 or RX3, on receipt of the end of frame (see Figure 12-23).

a) If the signal SIGNAL_FROM_FCP_EMF_RECEIVED_THRU_CASCADE_PORT is received, indicating that the End of MAC frame has arrived, the RSM calls SEND_HIGHEST_PRIORITY_REQUEST_OR_IDLE to update the transmit control signal for the cascade port. If the function returns a value of NONE (no Requests being received), then the RSM transitions to RX1; otherwise, it transitions to RX3. The REQ_WINDOW timer is started, to time the request window during which the updated request (or IDLE_U) is sent on the cascade port.

b) When a new control signal is received at a local port while the RSM is receiving the frame through the cascade port, the RSM calls SEND_HIGHEST_PRIORITY_REQUEST_OR_IDLE to update the transmit control signal for the cascade port. If the link from the cascade port is dual simplex, then the request (or IDLE_U) will be sent through the cascade port while the frame is being received.

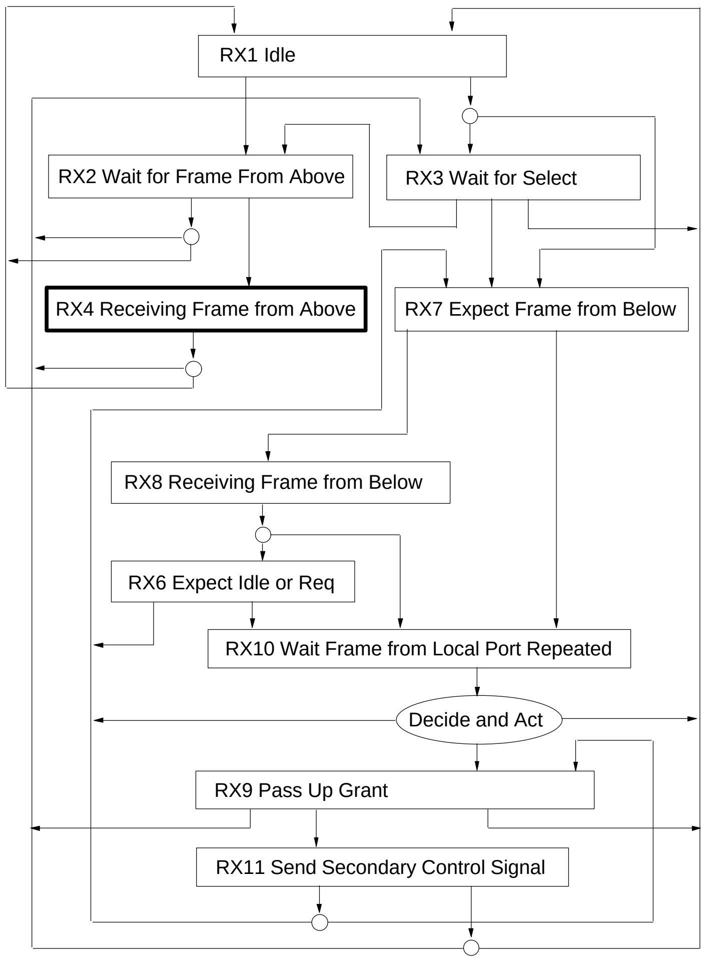

RX1 Idle
RX2 Wait for Frame From Above
RX3 Wait for Select
RX4 Receiving Frame from Above
RX7 Expect Frame from Below
RX8 Receiving Frame from Below
RX6 Expect Idle or Req
RX10 Wait Frame from Local Port Repeated
Decide and Act
RX9 Pass Up Grant
RX11 Send Secondary Control Signal

Figure 12-23—Relationship of RX4 within the state machine

```
Case EVENT_SORT Of
 S_FCP_EMF_RECEIVED_THRU_CASCADE_PORT :
 Begin
   RMAC_BUF_DATA_INPUT := STOP_INPUT;
   {Have finished receiving frame from upper repeater; decide level of request to send up,
     if any.}
   If (SEND_HIGHEST_PRIORITY_REQUEST_OR_IDLE() = NONE) Then
     RXSTATE := RX1_IDLE
   Else
   Begin
     BEEN_SELECTED := False;
     SET_FAULT_TIMER(REQUESTING_TOO_LONG);
     RXSTATE := RX3_WAIT_FOR_SELECT;
   End;
   SET_TIMER(REQ_WINDOW);
   FLAG_REQ_WINDOW_EXPIRED := False;
 End;

 LOCAL_PORT_CONTROL_SIGNAL_DECODED :
   L_PRIORITY := SEND_HIGHEST_PRIORITY_REQUEST_OR_IDLE();
 {Particularly important for dual simplex.}
```

End;**{Case}**

12.7.6 RX6 : RX6_EXPECT_IDLE_OR_REQ

When the RSM is in RX6, it has already selected a lower repeater, and received one or more frames from it (each in state RX8). On receiving the end of a frame in RX8, the RSM transitions to RX6 and waits for the lower repeater to send a Request (indicating that it has a further packet to send) or IDLE_U (indicating that it is passing up control). RX6 is entered only from RX8. Exit is to RX7 if there is another request from the lower repeater, and to RX10 if the lower repeater passes up control (see Figure 12-24).

a) On receipt of IDLE_U at the selected port, the RSM changes the control signal to the lower repeater (from GRANT or ENABLE_HIGH_ONLY) to IDLE_D and transitions to RX10 to wait for the frame to be completely transmitted before deciding on its next action.
b) If REQ_H or REQ_N is received from the lower repeater, the RSM prepares to repeat the new frame from below:
    1) If REQ_H is received and SELECTED_PORT_PRIORITY is NORMAL_PRIORITY, SELECTED_PORT_PRIORITY is updated to HIGH_PRIORITY.
    2) The control signal sent on the cascade port is made to be a request reflecting the priority level of the request received.
    3) The signal S_TO_FCP_EXPECT_FRAME is sent to the FCP, and the signal S_FROM_RSM_TO_TSM_GOTO_TX_INCOMING is sent to the TSM.
    4) The RSM transitions to RX7.
c) If a change in the request status of any of the other local ports is received, the RSM reappraises whether it should be sending ENABLE_HIGH_ONLY or GRANT to the lower repeater.
d) If IDLE_D or INCOMING is received through the cascade port, this indicates a conflict of control in the network: the repeater at the far end of the Uplink considers that it has control, although this RSM is expecting a frame from a lower entity. The situation is resolved by propagating IDLE_D through all local ports, thus relinquishing control to the higher-level repeater. FLAG_CONTROL_ACTUALLY_ABOVE is set to True, to indicate the conflict to the DECIDE_AND_ACT procedure (see 12.6.4.1.2).
e) If TRAINING_DOWN is received through the cascade port, then training on the Uplink is initiated.
f) If ENABLE_HIGH_ONLY or GRANT is received, the RSM reappraises whether it should send ENABLE_HIGH_ONLY through the selected port to the lower repeater, which has control.

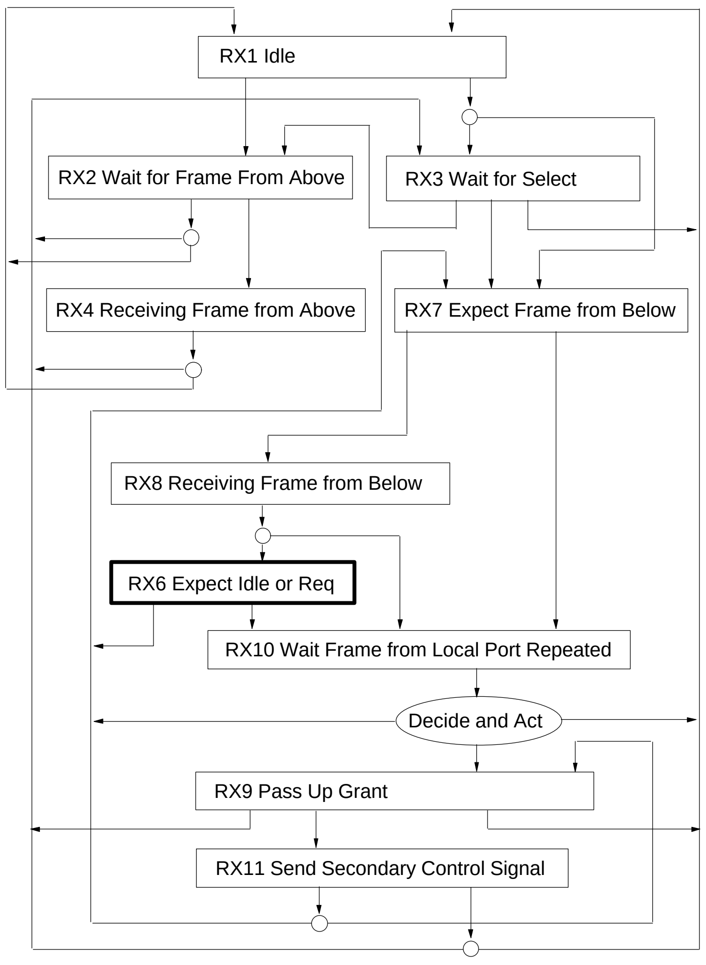

RX1 Idle
RX2 Wait for Frame From Above
RX3 Wait for Select
RX4 Receiving Frame from Above
RX7 Expect Frame from Below
RX8 Receiving Frame from Below
RX6 Expect Idle or Req
RX10 Wait Frame from Local Port Repeated
Decide and Act
RX9 Pass Up Grant
RX11 Send Secondary Control Signal

Figure 12-24—Relationship of RX6 within the state machine

```
Case EVENT_SORT Of
  LOCAL_PORT_CONTROL_SIGNAL_DECODED :
  If (PORT = SELECTED_PORT) Then
  Begin
    Case LOCAL_PORT_CONTROL_SIGNAL_IN Of
      IDLE_U :
      Begin
        {Was repeating packets from below; control is now being passed up by lower
repeater.}
        CHANGE_LOCAL_PORT_CONTROL_SIGNAL(PORT, IDLE_D);
        CAME_FROM_RX7 := False;
        CANCEL_FAULT_TIMER(IN_RX6_TOO_LONG)
        RXSTATE := RX10_WAIT_FRAME_FROM_LOCAL_PORT_REPEATED;
      End;

      REQ_H, REQ_N :
        {Must continue repeating frames from lower repeater in control.}
      Begin
        If ((LOCAL_PORT_CONTROL_SIGNAL_IN = REQ_H) And
            (SELECTED_PORT_PRIORITY = NORMAL_PRIORITY)) Then
          SELECTED_PORT_PRIORITY := HIGH_PRIORITY;
        CHANGE_CASCADE_PORT_CONTROL_SIGNAL
(LOCAL_PORT_CONTROL_SIGNAL_IN);
        SIGNAL_TO_FCP(S_TO_FCP_EXPECT_FRAME);
        SIGNAL_TO_TSM(S_TO_TSM_GOTO_TX_INCOMING);
        CANCEL_FAULT_TIMER(IN_RX6_TOO_LONG);
        RXSTATE := RX7_EXPECT_FRAME_FROM_BELOW; {Another frame coming.}
      End;
    End {Case}
  End {SELECTED_PORT}

Else
  Case LOCAL_PORT_CONTROL_SIGNAL_IN Of
    REQ_H, REQ_N, IDLE_U :
    {If there is a change at one of the other ports, update ENABLE_HIGH_ONLY.}
      DECIDE_WHETHER_TO_SEND_ENABLE_HIGH_ONLY();
  End;{Case}

  CASCADE_PORT_CONTROL_SIGNAL_DECODED :
  Case CASCADE_PORT_CONTROL_SIGNAL_IN Of
    IDLE_D, INCOMING :
    {Repeater believes control is with it or at a repeater below; however,
     repeater above believes IT has control. Need to pass control up.}
    Begin
      SEND_IDLE_TO_LOCAL_PORTS();
      FLAG_CONTROL_ACTUALLY_ABOVE := True;
    End;

    TRAINING_DOWN : INVOKE_CASCADE_PORT_TRAINING();
ENABLE_HIGH_ONLY, GRANT :
      {If ENABLE_HIGH_ONLY replaces GRANT or vice versa, reappraise whether
       GRANT or ENABLE_HIGH_ONLY should be sent to the lower repeater in control.}
      DECIDE_WHETHER_TO_SEND_ENABLE_HIGH_ONLY();
End; {Case}

TIMER_EXPIRES :
    If (TIMER = IN_RX6_TOO_LONG) Then
    Begin
        INVOKE_TRAINING(SELECTED_PORT);
        CAME_FROM_RX7 := False;
        RXSTATE := RX10_WAIT_FRAME_FROM_LOCAL_PORT_REPEATED;
    End;
End;{Case}
```

12.7.7 RX7 : RX7_EXPECT_FRAME_FROM_BELOW

In RX7, the repeater is expecting a frame on the selected port, from an end node or a lower repeater (see Figure 12-25). Initial entry to RX7 is from RX1, RX3, RX11, or DECIDE_AND_ACT; in each case, the port has been newly selected by SELECT_NEXT_PORT_AND_GRANT.

When the selected port is connected to a lower repeater, entry can also be from RX6; in this case, one or more frames have already been received on the selected port.

Exit from RX7 is normally to RX8, on receipt of the start of the expected frame. Exit can also be to RX10, as described in c) and f) below.

a) On the event S_FCP_SMF_RECEIVED_THRU_LOCAL_PORT, the RMAC_BUF_DATA_INPUT control register is set so that the RMAC buffer will receive its input octets from the FCP. The RSM transitions to RX8.

b) When the repeater has just selected a lower repeater at normal priority (SELECTED_PORT_PRIORITY = NORMAL_PRIORITY), it can receive a change in the control signal to Request_High before the expected frame arrives. In this case, the repeater knows that there is a normal-priority request somewhere below the lower repeater, which has been obscured by the high-priority request. The NORMAL_PRIORITY_POINTER is decremented so that when the normal-priority cycle resumes, it resumes where the normal-priority request previously occurred.

c) Occasionally, an end node or lower repeater can cease requesting without receiving the grant, and change to sending IDLE_U to indicate that it no longer wishes to send a frame. If this happens, on receiving IDLE_U, the RSM signals S_TO_TSM_GOTO_TX_IDLE to the TSM, and S_TO_FCP_
STOP_EXPECTING_FRAME to the FCP. The RSM then waits for IDLE_DURATION so that the IDLE_D being initiated by the TSM is sent for long enough to be decoded by the lower entities. After the wait, the RSM transitions to RX10.

d) If a new control signal is decoded at a local port other than the selected port, the RSM reappraises whether to send ENABLE_HIGH_ONLY out of the selected port.

e) If the timer SEND_IDLE_BURST (set in SELECT_NEXT_PORT_AND_GRANT) expires, the RSM sends GRANT out of the selected port.

f) If IDLE_D or INCOMING is received through the cascade port, there is a conflict of control in the network. The repeater at the far end of the Uplink considers that it has control, although this RSM is expecting a frame from a lower entity. The situation is resolved by propagating IDLE_D through all local ports, and FLAG_CONTROL_ACTUALLY_ABOVE is set to True to indicate the existence of the conflict to the DECIDE_AND_ACT procedure when it is called on exit from RX10.

g) If TRAINING_DOWN is received through the cascade port, training of the Uplink is initiated.

h) If ENABLE_HIGH_ONLY or GRANT is received through the cascade port, the RSM reappraises whether it should send ENABLE_HIGH_ONLY through the selected port.

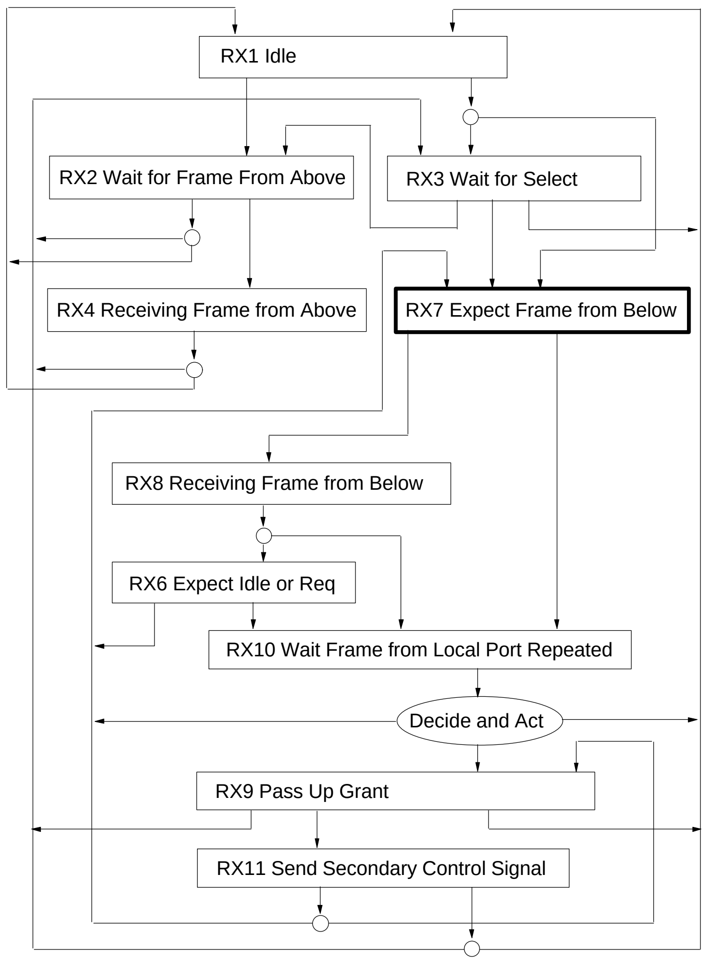

RX1 Idle
RX2 Wait for Frame From Above
RX3 Wait for Select
RX4 Receiving Frame from Above
RX7 Expect Frame from Below
RX8 Receiving Frame from Below
RX6 Expect Idle or Req
RX10 Wait Frame from Local Port Repeated
Decide and Act
RX9 Pass Up Grant
RX11 Send Secondary Control Signal

Figure 12-25—Relationship of RX7 within the state machine

```
Case EVENT_SORT Of
  S_FCP_SMF_RECEIVED_THRU_LOCAL_PORT :
  Begin
   {In RX7, waiting for a frame to arrive through selected port. It now comes.}
    RMAC_BUF_DATA_INPUT := OCTETS_FROM_FCP;
    RXSTATE := RX8_RECEIVING_FRAME_FROM_BELOW;
  End;

  LOCAL_PORT_CONTROL_SIGNAL_DECODED :
  If (PORT = SELECTED_PORT) Then
  Begin
    If ((LOCAL_PORT_CONTROL_SIGNAL_IN = REQ_H) And
        (SELECTED_PORT_PRIORITY = NORMAL_PRIORITY) And
        (JUST_SELECTED_PORT = True)) Then
   Begin

      DECREMENT_NORMAL_PRIORITY_POINTER_FROM_PORT(PORT);
      RECEIVING_IDLE := False;
      If CSTATE = C_NORMAL Then CSTATE := C_HIGH_WAS_NORMAL_UNFINISHED;
   End
Else If (LOCAL_PORT_CONTROL_SIGNAL_IN = IDLE_U) Then
```

```
  Begin
        {The lower entity no longer needs to transmit a frame. Grant has been passed
        down, but receives IDLE_U instead of a frame at the local port, indicating that
        the lower entity has changed its mind. Stop sending INCOMING to potential
        destinations, and send IDLE_D instead. However, wait for a short time so that next
        selected lower entity can decode IDLE_D, before the switch to GRANT occurs.}

   If (SILENT_WHEN_GRANTED = True) Then
     Begin
       SET_TIMER(IDLE_FILTER);
       RECEIVING_IDLE := True;
     End
    Else
     Begin
       SIGNAL_TO_TSM(S_TO_TSM_GOTO_TX_IDLE);
       SIGNAL_TO_FCP(S_TO_FCP_STOP_EXPECTING_FRAME);
       CAME_FROM_RX7 := True;
       If (TXSTATE <> TX1_SEND_FRAME) Then
       SET_TIMER(IDLE_DURATION);
       RXSTATE := RX10_WAIT_FRAME_FROM_LOCAL_PORT_REPEATED;
    End
  End
  Else RECEIVING_IDLE := False
End
Else {PORT <> SELECTED_PORT}
   {New situation at a local port. Update ENABLE_HIGH_ONLY situation.}
Case LOCAL_PORT_CONTROL_SIGNAL_IN Of
  REQ_H, REQ_N, IDLE_U :
   {If level of Request priority at port changes, reappraise whether GRANT or
    ENABLE_HIGH_ONLY should be sent to lower repeater in control.}
   If (JUST_SELECTED_PORT = False) Then
   DECIDE_WHETHER_TO_SEND_ENABLE_HIGH_ONLY();
End;{Case}

TIMER_EXPIRES :
      {Sending Idle burst because selected port was a destination of last packet.}
If (TIMER = SEND_IDLE_BURST) Then
  CHANGE_LOCAL_PORT_CONTROL_SIGNAL(SELECTED_PORT, GRANT)
  Else If ((TIMER = IDLE_FILTER) And (RECEIVING_IDLE = True)) Then
  Begin
   SIGNAL_TO_TSM(S_TO_TSM_GOTO_TX_IDLE);
   SIGNAL_TO_FCP(S_TO_FCP_STOP_EXPECTING_FRAME);
   CAME_FROM_RX7 := True;
   If (TXSTATE <> TX1_SEND_FRAME) Then
     SET_TIMER(IDLE_DURATION);
   RXSTATE := RX10_WAIT_FRAME_FROM_LOCAL_PORT_REPEATED;
  End;

CASCADE_PORT_CONTROL_SIGNAL_DECODED :
Case CASCADE_PORT_CONTROL_SIGNAL_IN Of
  IDLE_D, INCOMING :
  {Repeater believes control is with it or at a repeater below; however, the repeater above
  believes IT has control. Need to pass up control.}
  Begin
```

```
      SEND_IDLE_TO_LOCAL_PORTS();
      FLAG_CONTROL_ACTUALLY_ABOVE := True;
    End;

    TRAINING_DOWN :
      INVOKE_CASCADE_PORT_TRAINING();

ENABLE_HIGH_ONLY, GRANT :
       {If ENABLE_HIGH_ONLY replaces GRANT or vice versa, reappraise whether GRANT or
        ENABLE_HIGH_ONLY should be sent to lower repeater in control.}
      DECIDE_WHETHER_TO_SEND_ENABLE_HIGH_ONLY();
  End;{Case}
End;{Case}
```

12.7.8 RX8 : RX8_RECEIVING_FRAME_FROM_BELOW

When in RX8, the RSM is receiving a frame through the selected local port. Entry to RX8 is always from RX7, upon receipt of the start of the frame. Exit from RX8 may be to either RX6 or RX10, on receipt of the end of frame (see Figure 12-26).

a) When the selected port is training and receipt of the end of the frame is signaled by the FCP, INCOMING is sent to the lower entity, to inform it of the imminent arrival of the response Training frame. The RSM transitions to RX10.
b) When the selected port is not training and receipt of the end of the frame is signaled by FCP, then:
    1) If the lower entity is a repeater, the RSM evaluates whether ENABLE_HIGH_ONLY should be sent through the selected port, and transitions to RX6, or
    2) If the lower entity is an end node, the RSM transitions to RX10.
    3) In both cases, the RMAC_BUF_DATA_INPUT control register is changed to STOP_INPUT, so that no further octets are pushed into the RMAC buffer.
c) If IDLE_D or INCOMING is received through the cascade port, there is a conflict of control in the network. The repeater at the far end of the Uplink considers that it has control, although the local repeater is receiving a frame from a lower entity. The situation is resolved by propagating IDLE_D through all local ports, and FLAG_CONTROL_ACTUALLY_ABOVE is set to True to indicate the existence of the conflict to the DECIDE_AND_ACT procedure when it is called on exit from RX10.
d) If TRAINING_DOWN is received through the cascade port, training of the Uplink is initiated.
e) If ENABLE_HIGH_ONLY or GRANT is received through the cascade port, or REQ_H, REQ_N, or IDLE_U is received through a local port, the RSM reappraises whether it should send ENABLE_HIGH_ONLY through the selected port.

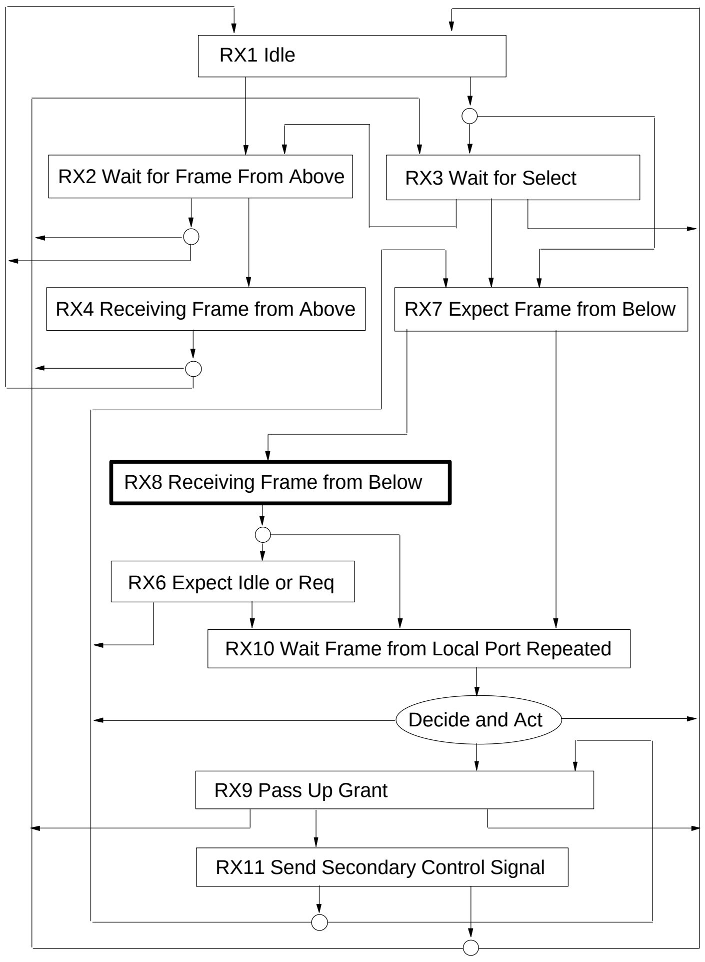

RX1 Idle
RX2 Wait for Frame From Above
RX3 Wait for Select
RX4 Receiving Frame from Above
RX7 Expect Frame from Below
RX8 Receiving Frame from Below
RX6 Expect Idle or Req
RX10 Wait Frame from Local Port Repeated
Decide and Act
RX9 Pass Up Grant
RX11 Send Secondary Control Signal

Figure 12-26—Relationship of RX8 within the state machine

```
Case EVENT_SORT Of
  S_FCP_EMF_RECEIVED_THRU_LOCAL_PORT :
  Begin
    If (PORT = SELECTED_PORT) Then
    Begin
      JUST_SELECTED_PORT := False;
      If (LOCAL_PORT_META_STATE[PORT] = TRAINING) Then
      Begin {Code required for training.}

        {Send INCOMING, in preparation for sending return training packet to lower entity.}
        CHANGE_LOCAL_PORT_CONTROL_SIGNAL(PORT, INCOMING);
        If (EMF_INDICATION = PACKET_OK) Then
        Begin
          If (LOCAL_PORT_PACKETS_OK_SEQUENCE[PORT] < 24) Then
            LOCAL_PORT_PACKETS_OK_SEQUENCE[PORT] :=
                 LOCAL_PORT_PACKETS_OK_SEQUENCE[PORT] + 1;
        End
        Else LOCAL_PORT_PACKETS_OK_SEQUENCE[PORT] := 0;
        CAME_FROM_RX7 := False;
        RXSTATE := RX10_WAIT_FRAME_FROM_LOCAL_PORT_REPEATED;
      End
      Else
      Begin {Received on link not in training.}
        If (ENTITY_AT_FAR_END_OF_LINK[PORT] = LOWER_REPEATER) Then
        Begin
          {End of MAC frame received from lower repeater. If Request is then received, the
            lower repeater will continue in control; if IDLE_U is received, control is being sent
up
            to this repeater. Wait in RX6 for Request or Idle.}
          DECIDE_WHETHER_TO_SEND_ENABLE_HIGH_ONLY();
          SET_FAULT_TIMER(IN_RX6_TOO_LONG);
          RXSTATE := RX6_EXPECT_IDLE_OR_REQ;
        End
        Else
        Begin
          CAME_FROM_RX7:=False;
          RXSTATE := RX10_WAIT_FRAME_FROM_LOCAL_PORT_REPEATED;
        End;
      End; {Else, i.e., not training frame.}
      RMAC_BUF_DATA_INPUT := STOP_INPUT;
    End
  End;

  CASCADE_PORT_CONTROL_SIGNAL_DECODED :
  Case CASCADE_PORT_CONTROL_SIGNAL_IN Of
    IDLE_D, INCOMING :
    {Repeater believes control is with it or at a repeater below; however, repeater above
     believes IT has control. Need to pass up control.}
    Begin
      SEND_IDLE_TO_LOCAL_PORTS();
      FLAG_CONTROL_ACTUALLY_ABOVE := True;
    End;

    TRAINING_DOWN :
      INVOKE_CASCADE_PORT_TRAINING();

    ENABLE_HIGH_ONLY, GRANT :
       {If ENABLE_HIGH_ONLY replaces GRANT or GRANT replaces ENABLE_HIGH_ONLY,
          reappraise whether GRANT or ENABLE_HIGH_ONLY should be sent to lower
          repeater in control.}
      DECIDE_WHETHER_TO_SEND_ENABLE_HIGH_ONLY();
  End; {Case}

  LOCAL_PORT_CONTROL_SIGNAL_DECODED :
Case LOCAL_PORT_CONTROL_SIGNAL_IN Of
    REQ_H, REQ_N, IDLE_U :
       DECIDE_WHETHER_TO_SEND_ENABLE_HIGH_ONLY();
  End; {Case}

End;{Case}
```

12.7.9 RX9 : RX9_PASS_UP_GRANT

In RX9, the RSM is passing up control to the repeater above by sending IDLE_U through the cascade port. Entry to RX9 is normally from DECIDE_AND_ACT, which is called from RX10. RX9 can also be re-entered from RX11, if the first attempt to pass up control fails. Exit from RX9 may be to RX1, RX3, or RX11 (see Figure 12-27).

When the timer PASS_UP_CONTROL_IDLE expires, the IDLE_U has been sent long enough for the upper repeater to decode it. The RSM goes to state RX11 to await receipt of IDLE_D in confirmation. Upon receipt of IDLE_D, the RSM starts sending the secondary control signal out of the cascade port:

a) If the flag SECONDARY_CONTROL_SIGNAL_IS_RETURN is True, the RSM sends RETURN through the cascade port. Otherwise, the repeater sends a Request corresponding to the highest request that it is receiving at a local port, or

b) If no requests are being received, it continues sending IDLE_U (IDLE_U is effectively the secondary control signal).

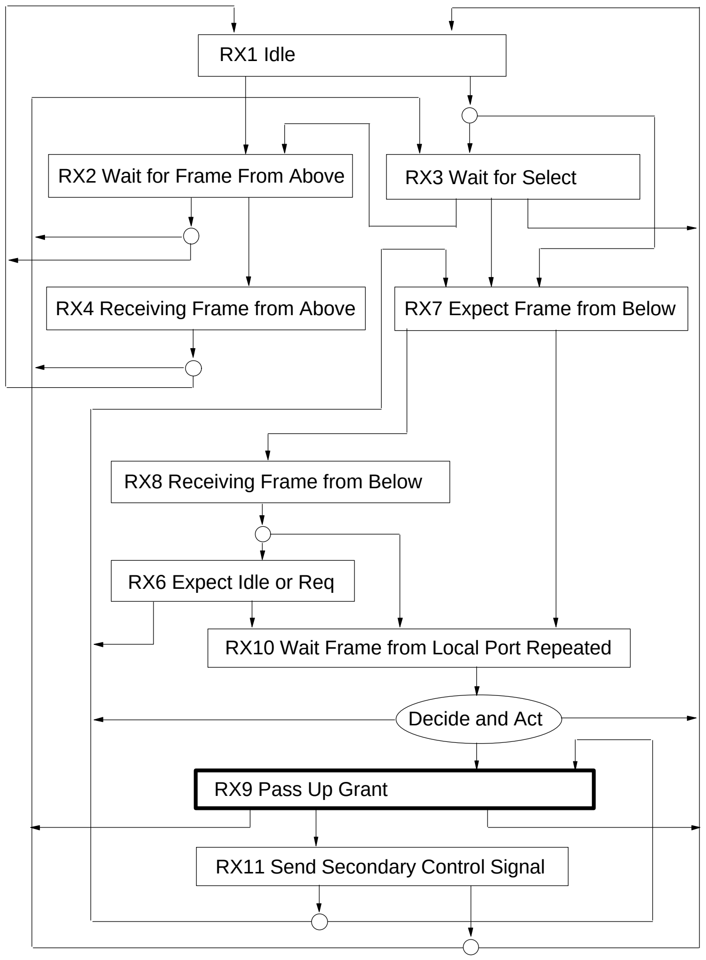

RX1 Idle
RX2 Wait for Frame From Above
RX3 Wait for Select
RX4 Receiving Frame from Above
RX7 Expect Frame from Below
RX8 Receiving Frame from Below
RX6 Expect Idle or Req
RX10 Wait Frame from Local Port Repeated
Decide and Act
RX9 Pass Up Grant
RX11 Send Secondary Control Signal

Figure 12-27—Relationship of RX9 within the state machine

Case EVENT_SORT Of

```
  TIMER_EXPIRES :
  If (TIMER = PASS_UP_CONTROL_IDLE) Then
     {Initial IDLE_U has been sent for long enough: start secondary signal.}
  Begin
    If (SECONDARY_CONTROL_SIGNAL_IS_RETURN = True) Then
      Begin
        CHANGE_CASCADE_PORT_CONTROL_SIGNAL(RETURN);
      End
    Else
      L_PRIORITY := SEND_HIGHEST_PRIORITY_REQUEST_OR_IDLE();
    RXSTATE := RX11_SEND_SECONDARY_CONTROL_SIGNAL;
  End;

IDLE_D:
    Begin
        CANCEL_TIMER(PASS_UP_CONTROL_IDLE);
        CANCEL_FAULT_TIMER(PASS_UP_TO_IDLE_DOWN_TIMEOUT);
        BEEN_SELECTED := False;
        If (SECONDARY_CONTROL_SIGNAL_IS_RETURN = True) Then
        Begin
            CHANGE_CASCADE_PORT_CONTROL_SIGNAL(RETURN);
            SET_FAULT_TIMER(REQUESTING_TOO_LONG);
            RXSTATE := RX3_WAIT_FOR_SELECT;
        End
        Else If (SEND_HIGHEST_PRIORITY_REQUEST_OR_IDLE() <> NONE) Then
        Begin
             SET_FAULT_TIMER(REQUESTING_TOO_LONG);
             RXSTATE := RX3_WAIT_FOR_SELECT;
        End
        Else
             RXSTATE := RX1_IDLE;
    End;
  End; {Case}
End; {Case}
```

12.7.10 RX10 : RX10_WAIT_FRAME_FROM_LOCAL_PORT_REPEATED

In the RX10 state, the RSM has finished receiving a frame through a local port, and is waiting until the frame has been fully repeated by the TSM and BCALL before the RSM can decide what to do next.

Entry to RX10 is normally from RX6 if the selected port is a lower repeater, and from RX8 otherwise. Entry can also be from RX7 if the selected port is a lower repeater and IDLE_UP is received. Exit from RX10 is always by calling DECIDE_AND_ACT (see Figure 12-28).

If the RSM receives RETURN at the selected port, this means that the lower repeater, which has just passed up control, has been interrupted in its normal-priority round-robin cycle, and is indicating that the interrupted normal-priority round-robin should be resumed at the lower repeater. The RSM decrements the normal-priority round-robin pointer, so that the next selected normal-priority port will be the one connected to the interrupted lower repeater.

Upon receipt of S_TSM_TO_RSM_EMF_XMITTED from the TSM, the RSM calls the procedure DECIDE_AND_ACT to decide its subsequent actions, resulting in transition to RX1, RX7, or RX9.

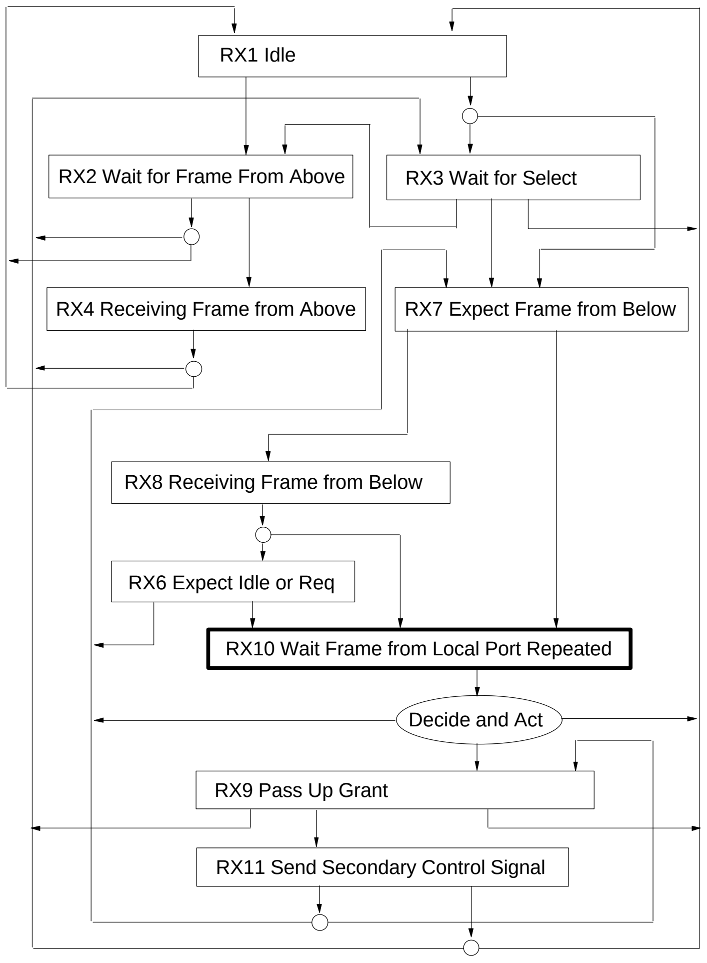

RX1 Idle
RX2 Wait for Frame From Above
RX3 Wait for Select
RX4 Receiving Frame from Above
RX7 Expect Frame from Below
RX8 Receiving Frame from Below
RX6 Expect Idle or Req
RX10 Wait Frame from Local Port Repeated
Decide and Act
RX9 Pass Up Grant
RX11 Send Secondary Control Signal

Figure 12-28—Relationship of RX10 within the state machine

```
Case EVENT_SORT Of

  LOCAL_PORT_CONTROL_SIGNAL_DECODED :
    If ((LOCAL_PORT_CONTROL_SIGNAL_IN = RETURN) And
       (PORT = SELECTED_PORT)) Then
    Begin
      DECREMENT_NORMAL_PRIORITY_POINTER_FROM_PORT(PORT);
      RMAC_REQ_REG[PORT] := NORMAL_PRIORITY;
      If (CSTATE = C_NORMAL) Then
        CSTATE := C_HIGH_WAS_NORMAL_UNFINISHED;
    End;

  S_TSM_TO_RSM_EMF_XMITTED :
      {Signal from TSM that repeater has transmitted end of packet.}
    If (CAME_FROM_RX7 = True) Then
       SET_TIMER(IDLE_DURATION)
    Else
    DECIDE_AND_ACT();

  TIMER_EXPIRES :
    If (TIMER = IDLE_DURATION) Then
       DECIDE_AND_ACT();
End;{Case}
```

12.7.11 RX11 : RX11_SEND_SECONDARY_CONTROL_SIGNAL

In RX11, the repeater has already sent IDLE_U through the cascade port to pass up control, and is now sending the Secondary Control Signal through the cascade port. It needs to receive IDLE_D from the upper repeater, as confirmation that the upper repeater has received IDLE_U. RX11 is entered only from RX9. Exit from RX11 is normally to RX1 or RX3. Following a failure to pass up control, exit can be to RX7 or RX1 (see Figure 12-29).

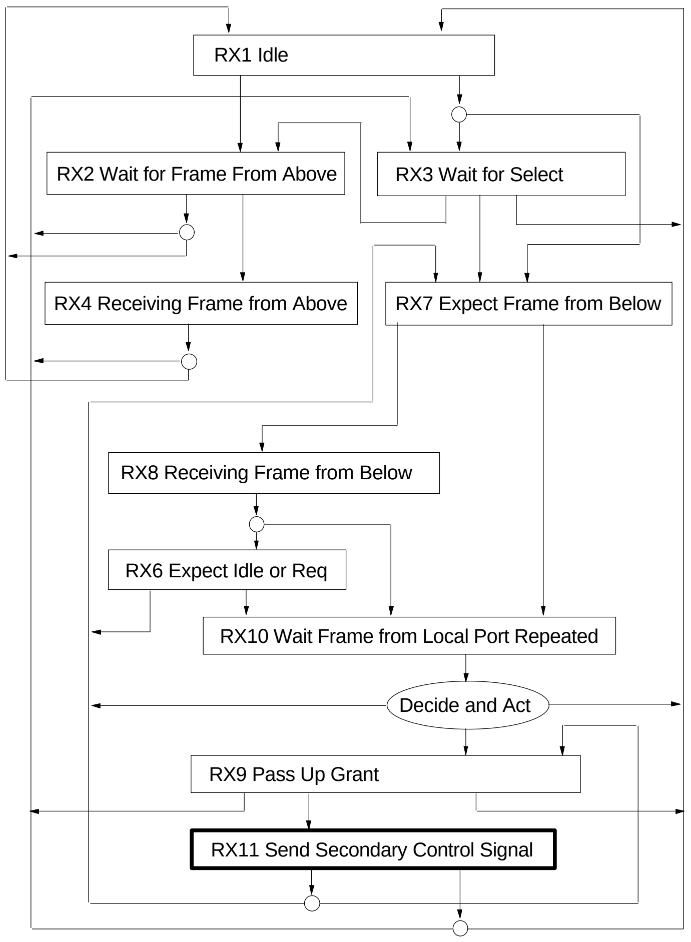

RX1 Idle
RX2 Wait for Frame From Above
RX3 Wait for Select
RX4 Receiving Frame from Above
RX7 Expect Frame from Below
RX8 Receiving Frame from Below
RX6 Expect Idle or Req
RX10 Wait Frame from Local Port Repeated
Decide and Act
RX9 Pass Up Grant
RX11 Send Secondary Control Signal

Figure 12-29—Relationship of RX11 within the state machine

a) On receipt of IDLE_D through the cascade port, if requests are being received at one or more local ports, the RSM transitions to RX3; otherwise, it transitions to RX1.

b) If the PASS_UP_TO_IDLE_DOWN_TIMEOUT timer has expired, the repeater has not received the IDLE_D reply to its IDLE_U. It then increments the counter PASS_UP_ATTEMPT.

   1) If PASS_UP_ATTEMPT = 1, the repeater tries again by sending up IDLE_U and returning to RX9.

   2) If PASS_UP_ATTEMPT > 1, it invokes cascade port training, and calls SELECT_NEXT_PORT_AND_GRANT. If this selects a local port, the RSM transitions to RX7; otherwise (there are no requests pending at any local ports), it transitions to RX1.

```
Case EVENT_SORT Of

  CASCADE_PORT_CONTROL_SIGNAL_DECODED :
  Case CASCADE_PORT_CONTROL_SIGNAL_IN Of
    IDLE_D :
       {Idle back from upper repeater; pass up control confirmed. Now send up request and
         wait for grant, if have local request(s) or just go to Idle state RX1.}
Begin
      CANCEL_FAULT_TIMER(PASS_UP_TO_IDLE_DOWN_TIMEOUT);
      BEEN_SELECTED := False;
      If (SECONDARY_CONTROL_SIGNAL_IS_RETURN = True) Then
         Begin
           SET_FAULT_TIMER(REQUESTING_TOO_LONG);
           RXSTATE := RX3_WAIT_FOR_SELECT;
         End
      Else If (SEND_HIGHEST_PRIORITY_REQUEST_OR_IDLE() <> NONE) Then
        Begin
          SET_FAULT_TIMER(REQUESTING_TOO_LONG);
          RXSTATE := RX3_WAIT_FOR_SELECT;
        End
      Else
        RXSTATE := RX1_IDLE;
    End;
End; {Case}

  TIMER_EXPIRES :
  If (TIMER = PASS_UP_TO_IDLE_DOWN_TIMEOUT) Then
    Begin
      If (PASS_UP_ATTEMPTS = 1) Then
      Begin
        PASS_UP_ATTEMPTS := PASS_UP_ATTEMPTS + 1;
        CHANGE_CASCADE_PORT_CONTROL_SIGNAL(IDLE_U);
        SET_TIMER(PASS_UP_CONTROL_IDLE);
        SET_FAULT_TIMER(PASS_UP_TO_IDLE_DOWN_TIMEOUT);
        RXSTATE := RX9_PASS_UP_GRANT;
      End
      Else
      Begin
        INVOKE_CASCADE_PORT_TRAINING();
        SECONDARY_CONTROL_SIGNAL_IS_RETURN := False;
        If (SELECT_NEXT_PORT_AND_GRANT() <> NONE) Then
          RXSTATE := RX7_EXPECT_FRAME_FROM_BELOW
        Else RXSTATE := RX1_IDLE
      End
    End;
End; {Case}
```

12.8 Transmit State Machine (TSM)

The TSM controls the sequence of transmit-related operations before, during, and after packet transmission. It sets the appropriate TCS's for each port and combines with the BCALL to identify when packet transmission can begin. The TSM communicates with the RSM, BCALL, TULM, and each PPM. The relationship of the TSM to the RMAC state machines is shown in Figure 12-30.

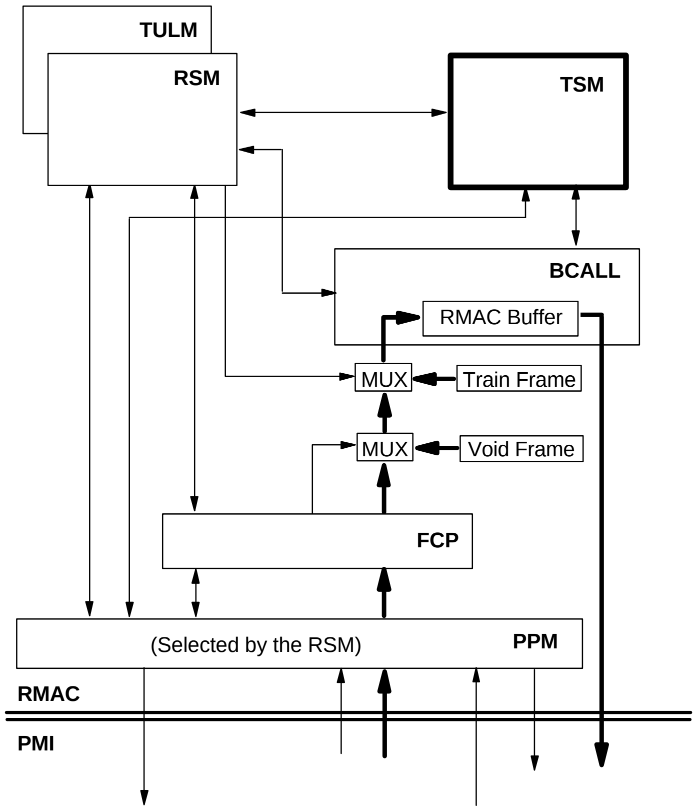


Figure 12-30—Relationship of the TSM to the RMAC state machines

12.8.1 Variables local to the TSM

FLAG_CAN_START_TRANSMIT : Boolean; Indicates that the event signal S_BCALL_TO_TSM_CAN_START_TRANSMIT has arrived.

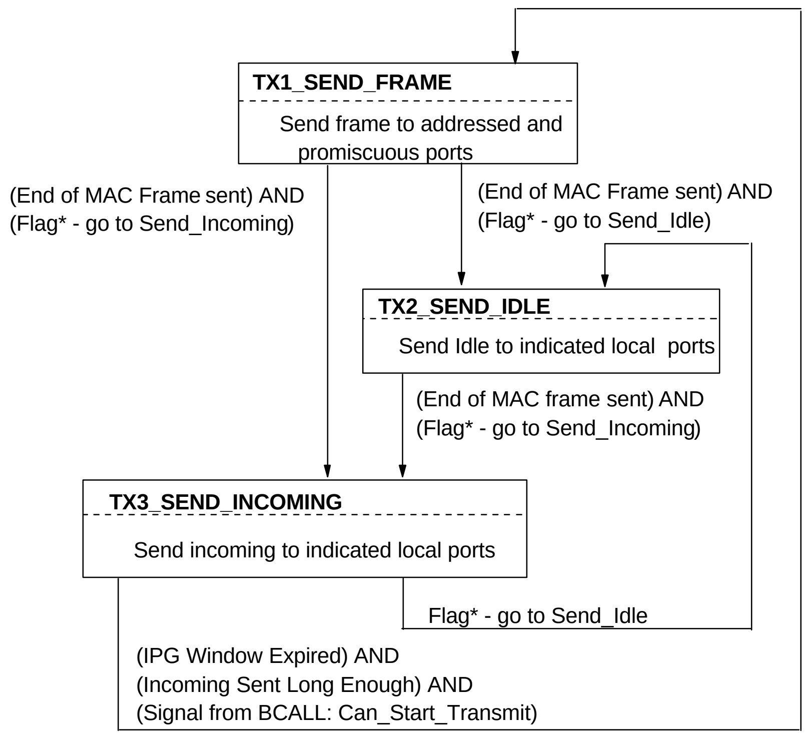
FLAG_GO_TO_TX_INCOMING : Boolean; Reset (False) on entry to TX1; set/reset in TX1 on signals ...GOTO_TX_INCOMING/...GOTO_TX_IDLE, from RSM and TULM. Used when leaving TX1, to determine whether IDLE_D or INCOMING is transmitted after the frame just sent: if the value = False, the TSM goes to TX2 (IDLE); if the value = True, TSM goes to TX3 (INCOMING).

FLAG_IPG_WINDOW_EXPIRED : Boolean; Used to indicate that the Inter Packet Gap timer has expired.

FLAG_SENT_INCOMING_LONG_ENOUGH : Boolean; Used in TSM, indicating timer has expired, which means that INCOMING has been sent for long enough to end nodes to be recognized.

12.8.2 The TSM state description

The TSM state diagram is shown in Figure 12-31.


TX1_SEND_FRAME
Send frame to addressed and promiscuous ports
(End of MAC Frame sent) AND (Flag* - go to Send_Incoming)
(End of MAC Frame sent) AND (Flag* - go to Send_Idle)
TX2_SEND_IDLE
Send Idle to indicated local ports
(End of MAC frame sent) AND (Flag* - go to Send_Incoming)
TX3_SEND_INCOMING
Send incoming to indicated local ports
Flag* - go to Send_Idle
(IPG Window Expired) AND
(Incoming Sent Long Enough) AND
(Signal from BCALL: Can_Start_Transmit)

* The referenced flag may have been set by either the RSM or the TULM.

Figure 12-31—Transmit State Machine

12.8.2.1 TX1 : TX1_SEND_FRAME

When the TSM is in TX1, a frame is being read out of the RMAC buffer, and is being sent out of the destination port(s). The BCALL module is responsible for the actual transfer of data octets from the RMAC buffer across the interface to the PMI, by generating the PMI primitives for the destination ports (see 12.9). Entry to TX1 is only from TX3, when the three conditions necessary for start of transmission are all satisfied (see 12.8.2.3). Exit from TX1 is to either TX2 or TX3, and occurs only at the end of transmitting the frame. The end of transmission is signaled by the BCALL when it transfers the last octet of the frame out of the RMAC buffer and sends S_BCALL_TO_TSM_EMF_TRANSMITTED.

The value of the flag FLAG_GO_TO_TX_INCOMING determines whether exit is to TX2 (flag set to False) or TX3 (flag set to True). The flag is always False on entry to TX1, and within TX1 it is set True / False in response to the signals S_TO_TSM_GOTO_TX_INCOMING and S_TO_TSM_GOTO_ TX_IDLE.

Further actions upon end of frame transmission are:

a) The signal S_TSM_TO_RSM_EMF_XMITTED is used to notify the RSM of the end of frame transmission. In some states, the RSM is waiting for this signal before proceeding to its next actions.

b) The IPG_WINDOW timer is started, to measure the interpacket gap that occurs between successive transmissions, and FLAG_IPG_WINDOW_EXPIRED is set to False.
c) If state TX3 is entered, the timer SENT_INCOMING_LONG_ENOUGH is started, and both FLAG_SENT_INCOMING_LONG_ENOUGH and FLAG_CAN_START_TRANSMIT are set to False.
d) IDLE_D or INCOMING is sent to all the local ports, according to whether exit is to TX2 or TX3.

Actions b) and c) initialize the three conditions that must change to enable the transition from TX3 to TX1 for sending the next frame. Both timers need to have expired, and the third flag needs to have been set to True on signal from the BCALL.

```
Case EVENT_SORT Of

  S_BCALL_TO_TSM_EMF_TRANSMITTED :
  Begin
    SET_TIMER(IPG_WINDOW);
    FLAG_IPG_WINDOW_EXPIRED := False;
    SIGNAL_TO_RSM(S_TSM_TO_RSM_EMF_XMITTED);
    {Tell RSM frame fully sent.}

    {Now send Idle or Incoming to local ports, according to whether the next
     frame is imminent.}
    If (FLAG_GO_TO_TX_INCOMING = True) Then
    Begin
      SEND_INCOMING_TO_LOCAL_PORTS();
      FLAG_CAN_START_TRANSMIT := False;
      SET_TIMER(SENT_INCOMING_LONG_ENOUGH);
      FLAG_SENT_INCOMING_LONG_ENOUGH := False;
      TXSTATE := TX3_SEND_INCOMING;
    End
    Else
    Begin
      SEND_IDLE_TO_LOCAL_PORTS();
      TXSTATE := TX2_SEND_IDLE;
    End
  End;

  S_TO_TSM_GOTO_TX_INCOMING :
    FLAG_GO_TO_TX_INCOMING := True;

  S_TO_TSM_GOTO_TX_IDLE :
    FLAG_GO_TO_TX_INCOMING := False;
End; {Case}
```

12.8.2.2 TX2 : TX2_SEND_IDLE

In state TX2, the TSM is sending IDLE_D on all active local ports. No frame is being transmitted, and the repeater is not expecting to receive a frame. Entry to TX2 is from TX1 on completion of transmitting a frame when no following frame is expected, and from TX3 when the repeater detects that an expected frame is not going to arrive (see Figure 12-32 and description of TX1 in 12.8.2.1). Exit from TX2 is only to TX3, on receipt of S_TO_TSM_GOTO_TX_INCOMING or S_TO_TSM_GOTO_TX_INCOMING. On exit to TX3, the timer SENT_INCOMING_LONG_ENOUGH is started, the flags: FLAG_SENT_

INCOMING_LONG_ENOUGH and FLAG_CAN_START_TRANSMIT are both set to False, and INCOMING is sent to all the local ports except the selected port.

The one other event of significance for TX2 is expiration of the IPG_WINDOW timer; the flag FLAG_IPG_WINDOW_EXPIRED is set to True to record the fact that an adequate interpacket gap has occurred.

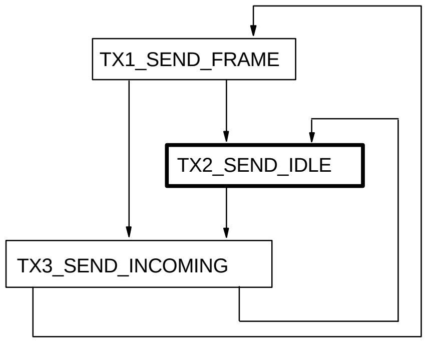


Figure 12-32—Relationship of TX2 within the state machine

```
Case EVENT_SORT Of
  TIMER_EXPIRES :
  If ((TIMER = IPG_WINDOW) Or (TIMER = DELTA_IPG_WINDOW)) Then
    FLAG_IPG_WINDOW_EXPIRED := True;

  S_TO_TSM_GOTO_TX_INCOMING :
  Begin
    SEND_INCOMING_TO_LOCAL_PORTS();
    FLAG_CAN_START_TRANSMIT := False;
    SET_TIMER(SENT_INCOMING_LONG_ENOUGH);
    FLAG_SENT_INCOMING_LONG_ENOUGH := False;
    TXSTATE := TX3_SEND_INCOMING;
  End;
End; {Case}
```

12.8.2.3 TX3 : TX3_SEND_INCOMING

In state TX3, the TSM is sending INCOMING on all the local ports other than the selected port, in preparation for transmitting a frame. Entry to TX3 is from either TX1 or TX2 (see Figure 12-33). Normal exit is to TX1, to send the frame; however, exit to TX2 can also occur, when the repeater detects that the expected frame is not going to arrive (S_TO_TSM_GOTO_TX_IDLE is received). For the normal exit to TX1 to occur, three conditions must be satisfied, as indicated by the three corresponding flags being set to True. An adequate interpacket gap window must have occurred (FLAG_IPG_WINDOW_EXPIRED); INCOMING must have been sent for long enough that all local-port links have been cleared (FLAG_SENT_INCOMING_LONG_ENOUGH); and the BCALL must have notified the TSM that transmission can start (FLAG_CAN_START_TRANSMIT). The events associated with the three conditions simply set the flags appropriately, but in each case all three flags are then inspected to see if the transition to TX1 can now occur. If all are set to True:

a) The signal S_TSM_TO_BCALL_START_TRANSMIT is sent, to indicate that the BCALL can start transmitting the frame.
b) The TSM changes the outgoing control signal to IDLE_D, for all of the local ports where ADDRESS_ MATCH has been set to False by the BCALL.

c) The TSM then transitions to TX1. Handling of the interpacket gap involves use of a secondary timer, DELTA_IPG_WINDOW, when the source of the frame is an end node connected directly to the selected port. In this case, the DELTA_IPG_WINDOW timer is used to lengthen the interpacket gap slightly. The rationale for this is that, if this lengthening did not take place at the first repeater in the path of the packet from the source end node, then clock differences between the first and subsequent repeaters could cause the incoming IPG Window on packets to a subsequent repeater to be shorter than the IPG Window which that repeater maintains on outgoing packets. This could cause overflow problems on a long stream of packets.

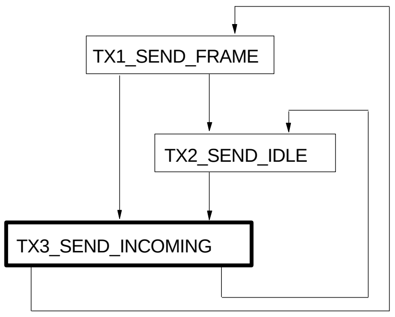


Figure 12-33—Relationship of TX3 within the state machine

```
Begin
  Case EVENT_SORT Of
    TIMER_EXPIRES :
    Case TIMER Of
      IPG_WINDOW :
      If (SELECTED_PORT = CASCADE_PORT) Then
        FLAG_IPG_WINDOW_EXPIRED := True
      Else
      If (ENTITY_AT_FAR_END_OF_LINK[SELECTED_PORT] = LOWER_REPEATER) Then
        FLAG_IPG_WINDOW_EXPIRED := True
      Else SET_TIMER(DELTA_IPG_WINDOW);
      {The short timer DELTA_IPG_WINDOW is used to have the IPG Window of the repeater
            sourcing the packet (the repeater connected to the sending end node) implement a
            slightly longer IPG window than the other repeaters. This prevents problems with
            repeaters of slightly different clock frequencies.}

      DELTA_IPG_WINDOW :
         FLAG_IPG_WINDOW_EXPIRED := True;

      SENT_INCOMING_LONG_ENOUGH :
         FLAG_SENT_INCOMING_LONG_ENOUGH := True;
         {Half-duplex destinations have been receiving INCOMING long enough for link to be
         clear at repeater on all pairs, even for longest half-duplex link.}
    End; {Case}

    S_BCALL_TO_TSM_CAN_START_TRANSMIT : FLAG_CAN_START_TRANSMIT := True;

    S_TO_TSM_GOTO_TX_IDLE :
    Begin
      SEND_IDLE_TO_LOCAL_PORTS();
      CANCEL_TIMER(SENT_INCOMING_LONG_ENOUGH);
      TXSTATE := TX2_SEND_IDLE;
    End;
  End; {Case}

  If ((FLAG_IPG_WINDOW_EXPIRED = True) And
     (FLAG_CAN_START_TRANSMIT = True) And
     (FLAG_SENT_INCOMING_LONG_ENOUGH = True)) Then
  Begin
    SIGNAL_TO_BCALL(S_TSM_TO_BCALL_START_TRANSMIT);
    For C_PORT := 1 To NUMBER_OF_PORTS Do
    Begin
      If (C_PORT = SELECTED_PORT) Then
      Begin
        If (ENTITY_AT_FAR_END_OF_LINK[C_PORT] = END_NODE) Then
          CHANGE_LOCAL_PORT_CONTROL_SIGNAL(C_PORT, IDLE_D);
      End
      Else Begin
        If (ADDRESS_MATCH[C_PORT] = False) Then
          CHANGE_LOCAL_PORT_CONTROL_SIGNAL(C_PORT, IDLE_D);
      End
    End;
    FLAG_GO_TO_TX_INCOMING := False;
    TXSTATE := TX1_SEND_FRAME;
  End
End;
```

12.9 The Buffer Control and Address Lookup Logic (BCALL) state machine

The BCALL state machine has the primary responsibility for control of the flow of octets into and out of the RMAC frame buffer. The RMAC buffer is a FIFO buffer that inputs octets received at the currently selected port (or, in some cases, octets generated by the repeater itself), and then outputs these octets in the same sequence for transmission through the appropriate port(s). The RMAC buffer is the only destination for incoming and internally generated frames, and is the only source for transmitted frames.

The BCALL is a conceptual state machine that interfaces with the other state machines in the RMAC. Event indications and shared control variables synchronize the operation of the BCALL with respect to the start and end of MAC frames, and control the flow of frames to the appropriate output ports. Figure 12-34 shows the BCALL’s relationship to other state machines in the RMAC and to a typical PPM. The BCALL shall behave as specified in the remainder of this subclause.

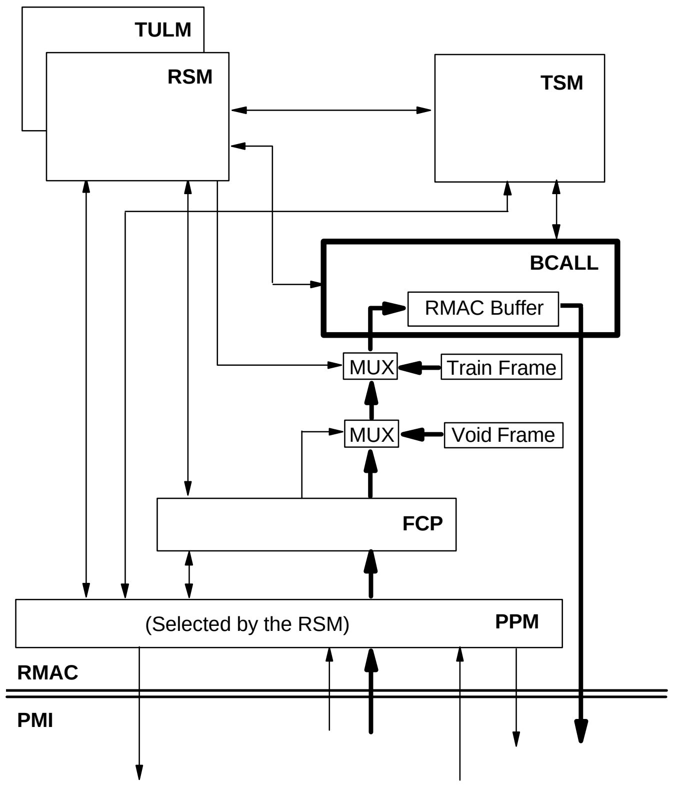


Figure 12-34—Relationship of BCALL in the RMAC state machine structure

12.9.1 BCALL functions

The functions of the BCALL are:

a) Connection of the RMAC buffer to the appropriate input sources, as directed by other RMAC state machines.
b) Monitoring of received-frame length, to enforce the maximum size of the frame being input to the RMAC buffer.
c) Identification of the type of each received frame (normal MAC frame, void frame, or training frame).
d) Extraction of destination addresses from each received frame.
e) Invocation of checking the requested configuration in training frames received at ports that are training.
f) Generation of allowed configuration field information in training frames for transmission through local ports that are training.
g) Selection of the set of ports through which each frame is to be transmitted.
h) Notification to the TSM of when a frame is ready to be sent by the BCALL.
i) Transmission of frames through the appropriate ports after receipt of notification from the TSM that transmission can now begin.
j) Indication of end-of-frame status information for each transmitted frame.
k) Notification of when the last octet of the frame has been transferred out of the RMAC buffer.

12.9.2 BCALL control registers

The BCALL uses two registers to control the source of the octet stream being written into the RMAC buffer, two status registers (RMAC_BUF_DATA_INPUT and FCP_DATA_OUTPUT), and an array of single bit ADDRESS_MATCH registers to indicate through which ports the outgoing frame is being sent.

RMAC_BUF_DATA_INPUT: T_RMAC_BUF_DATA_INPUT; The RMAC_BUF_DATA_INPUT register is an input source control that is set by either the RSM or the TULM. It controls the upper octet multiplexer in the data path to the RMAC buffer. The RMAC_BUF_DATA_INPUT value can be:

— STOP_INPUT
— OCTETS_FROM_FCP
— TRAINING_FRAME_OCTETS

FCP_DATA_OUTPUT: T_FCP_DATA_OUTPUT The FCP_DATA_OUTPUT register is also an input source control that is set by the FCP. It controls the lower multiplexer in the data path to the RMAC buffer. The FCP_DATA_OUTPUT value can be:

— OCTETS_FROM_PMI
— ZERO_OCTETS
— NIL

EMF_INDICATION: T_EMF_INDICATION; The EMF_INDICATION register is used by the BCALL and the PPM. It is set by the FCP. The EMF_INDICATION value can be:

— PACKET_OK
— INVALID_PACKET_MARKER

Before each frame is received, the FCP sets the value of EMF_INDICATION to PACKET_OK. After the frame is fully received, if there has been any error detected by FCP itself, or by the PMI, the PPM, or the CRC check, the FCP sets the value of EMF_INDICATION to INVALID_PACKET_MARKER.

ERROR_INDICATION: T_EMF_INDICATION; The ERROR_INDICATION register is set and used locally by BCALL as specified in 12.9.6 and 12.9.7. The ERROR_INDICATION value can be:

— PACKET_OK,
— INVALID_PACKET_MARKER

ADDRESS_MATCH[PORT_X]: Array [ZERO..NUMBER_OF_PORTS] Of Boolean; The ADDRESS_MATCH register is a shared array of one-bit registers, one per port, including the cascade port. BCALL sets each element of the ADDRESS_MATCH array for each frame to be transmitted. ADDRESS_MATCH element values can be:

— True, if the frame is to be transmitted through the corresponding port
— False, if the frame is not to be transmitted through the corresponding port

12.9.3 Event signaling

Event signaling between the BCALL and the TSM provides operational synchronization of these two state machines during frame transmission. Event signaling between the BCALL and the TULM provides operational synchronization of these two state machines during training.

TRAINING_FRAME_INTO_RMAC_BUFFER_COMPLETE. The BCALL uses the signal TRAINING_FRAME_INTO_RMAC_BUFFER_COMPLETE to inform the TULM that the last octet of the frame has been transferred into the RMAC buffer.

S_TSM_TO_BCALL_START_TRANSMIT. Receipt of S_TSM_TO_BCALL_START_TRANSMIT indicates that the BCALL can start transferring octets out of the RMAC buffer for transmission through the appropriate ports.

S_BCALL_TO_TSM_CAN_START_TRANSMIT. The BCALL uses the signal S_BCALL_TO_TSM_ CAN_START_TRANSMIT to inform the TSM that the destination address has been extracted, the ADDRESS_MATCH registers have been set, any necessary delays have been satisfied, and, from the BCALL's standpoint, the frame is ready to be sent. For ports that are in training, S_BCALL_TO_TSM_CAN_START_TRANSMIT will be sent a period equal to SENT_INCOMING_ LONG_ENOUGH after the signal S_FCP_EMF_RECEIVED_THROUGH_LOCAL_PORT.

S_BCALL_TO_TSM_EMF_TRANSMITTED. The BCALL uses the signal S_BCALL_TO_TSM_ EMF_ TRANSMITTED to inform the TSM that the last octet of the frame has been transferred out of the RMAC buffer.

12.9.4 Input to the RMAC buffer

Input to the RMAC buffer is controlled by the two data path multiplexers shown in Figure 12-35.

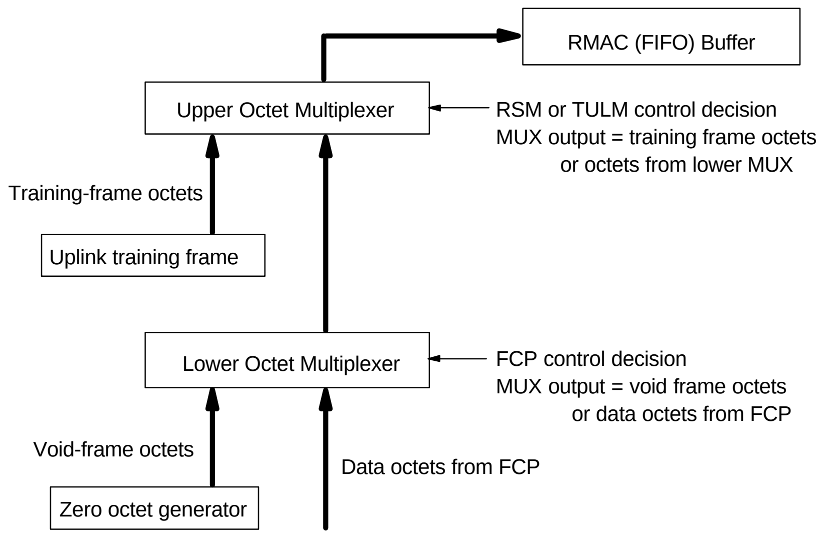


Figure 12-35—Data path multiplexers

The upper multiplexer is controlled by the RSM and the TULM; the lower multiplexer is controlled by the FCP. Together, the two multiplexers allow the RMAC buffer input to be a received data frame, an uplink training frame, a void frame, or a concatenation of void frame zeros followed by incoming frame data:

a) Data octets received at the currently selected port, via the PPM, as data-octet parameter values in PMI_UNITDATA_indication primitives. On receipt of a PMI_UNITDATA.indication primitive with a mac_frame parameter value of "begin" and a start_mac_frame_status parameter value of "valid," the input control variables will have been set to:

```
FCP_DATA_OUTPUT := OCTETS_FROM_PMI
RMAC_BUF_DATA_INPUT := OCTETS_FROM_FCP
```

Values of the input control variables are set in the FCP and the RSM pseudo code and are not changed until a PMI_UNITDATA.indication primitive with a mac_frame parameter value of “end” is received or the LONG_RECEIVE_INITIATE_TRAINING timer has expired. At that time, the input control variables will have been set to:

```
FCP_DATA_OUTPUT := NIL
RMAC_BUF_DATA_INPUT := STOP_INPUT
```

b) Void frame octets, for a void frame being constructed in place of an expected frame that has not yet been received at the selected port. When the FCP timer FRAME_EXPECTED_TO_SMF_IN_TIMEOUT has expired, the input control variables will have been set to:

```
FCP_DATA_OUTPUT := ZERO_OCTETS
RMAC_BUF_DATA_INPUT := OCTETS_FROM_FCP
```

The void frame octets shall be generated at a rate corresponding to the expected octet rate for RMAC buffer input.

1) After a period of VOID_FRAME, the input control variables will have been set to:

```
FCP_DATA_OUTPUT := NIL
RMAC_BUF_DATA_INPUT := STOP_INPUT
```

2) If a PMI_UNITDATA.indication primitive with a mac_frame parameter value of “begin” is received before a period of VOID_FRAME has occurred, the lower multiplexer is switched to incoming data. The input control variables for this case will have been set to:

```
FCP_DATA_OUTPUT := OCTETS_FROM_PMI
RMAC_BUF_DATA_INPUT := OCTETS_FROM_FCP
```

They are left unchanged until a PMI_UNITDATA.indication primitive with a mac_frame parameter value of “end” has been received, or until the LONG_RECEIVE_INITIATE_TRAINING timer expires, whichever occurs first. At that time, the input control variables will have been set to:

```
FCP_DATA_OUTPUT := NIL
RMAC_BUF_DATA_INPUT := STOP_INPUT
```

c) Void frame octets, for a void frame being constructed in place of an erroneous frame received at the selected port. On receipt of a PMI_UNITDATA.indication primitive with a mac_frame parameter value of “begin” and a start_mac_frame_status parameter value of “invalid,” the input control variables will have been set to :

```
FCP_DATA_OUTPUT := ZERO_OCTETS
RMAC_BUF_DATA_INPUT := OCTETS_FROM_FCP
```

The void frame octets are generated at a rate corresponding to the expected octet rate for RMAC buffer input such that each zero octet takes the place of one octet received in a PMI_UNITDATA_indication. The input control variables are left unchanged until a PMI_UNITDATA.indication primitive with a mac_frame parameter value of “end” has been received, or until the LONG_RECEIVE_INITIATE_TRAINING timer expires, whichever occurs first. At that time, the input control variables will have been set to:

```
FCP_DATA_OUTPUT := NIL
RMAC_BUF_DATA_INPUT := STOP_INPUT
```

d) Locally generated training-frame octets for training the uplink. The input control variables for this case are:

```
FCP_DATA_OUTPUT := NIL
RMAC_BUF_DATA_INPUT := TRAINING_FRAME_OCTETS
```

Values of the input control variables are not to be changed until the entire training frame has been transferred into the RMAC buffer. At that time, the event TRAINING_FRAME_INTO_ RMAC_BUFFER_COMPLETE is signaled to the TULM, and the input control variables will have been set to:

```
FCP_DATA_OUTPUT := NIL
RMAC_BUF_DATA_INPUT := STOP_INPUT
```

12.9.5 Address extraction

The BCALL extracts the destination address from each received frame. It then uses the extracted address as input to the ADDRESS_MATCH function and for the destination address parameter in the layer management procedure calls made by the PROCESS_DATA procedure in 12.11.5.1.1. The BCALL sets the destination address to the null address when in ISO/IEC 8802-5 framing mode and the AC and FC fields are zero.

12.9.6 Length monitoring

The BCALL is responsible for monitoring the length of each frame written into the RMAC buffer and ensuring that it does not exceed the maximum frame size for the current framing type.

The ERROR_INDICATION register is initialized to PACKET_OK at the start of each received frame. If an input source presents an octet that would cause the current frame to exceed the Maximum Frame Size, the octet is not written into the RMAC buffer, and the ERROR_INDICATION register is set to INVALID_
PACKET_MARKER. Subsequent excess octets are ignored.

12.9.7 Control of frame transmission

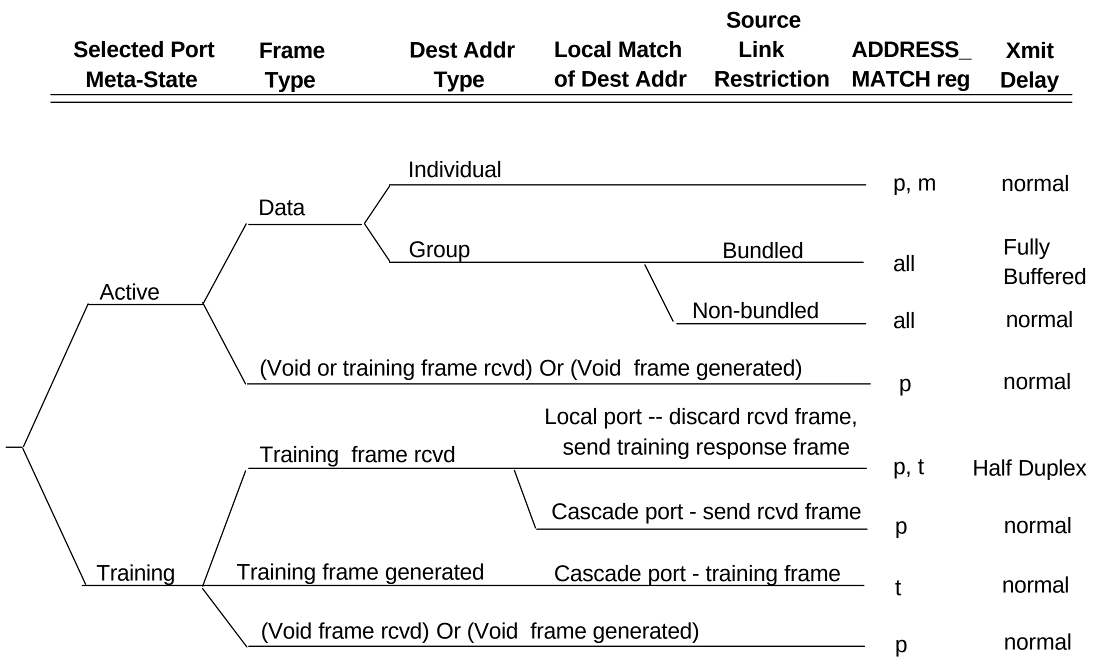
As a frame is being written into the RMAC buffer, the BCALL uses the input source, the type and destination address of the frame, the repeater's stored address information for directly attached end nodes, any link restrictions, and the meta-states of the individual ports to determine:

a) The set of ports through which the frame is to be transmitted.
b) When it is permissible for transmission of the frame to begin.

When these determinations are complete, the BCALL sets the ADDRESS_MATCH registers and then signals the event BCALL_TO_TSM_CAN_START_TRANSMIT to TSM. The actual start of transfer is a TSM function. It is indicated by the return handshake signal TSM_TO_BCALL_START_TRANSMIT.

The factors pertaining to the port selection and timing for each frame transmission are contained in the decision tree shown in Figure 12-36. Definitions of the various factors are given in 12.9.7.1 and 12.9.7.2.

BCALL is allowed to grow errored frames that are below the minimum packet length to ensure that the address lookups and other timing requirements are met. The final length shall not be over 18 bytes for a frame that was originally under 18 bytes.



all =	All ports except the selected port
m =	Destination address matched an address on the RMAC port address list except the selected port
p =	All promiscuous ports, except the selected port
t =	Port being trained (local port selected by RSM, cascade port selected by TULM)
bundled =	Source link in bundled cable
selected port =	Port selected by the RSM (port through which frame was rcvd)
normal =	Normal delay for overlapped transmission—see Figure 12-37.

ADDRESS_MATCH is:
a) Always true for m and p
b) Never true for an inactive port
c) True only for training frames in a training port (and only to return the training response frame)

BCALL_TO_TSM_CAN_START_TRANSMIT is sent only after:
a) All decision tree logic has been satisfied,
b) All ADDRESS_MATCH registers have been set, and
c) Any extra transmit delay has expired:
  1) Fully buffered for store and forward transmission—see Figure 12-38.
  2) Half duplex, training a local port—see Figure 12-39.

Figure 12-36—Frame transmission decision tree

12.9.7.1 Address match considerations

All ADDRESS_MATCH registers are set to false before each frame is received. They are set to true for each port meeting the defined criteria:

Port. The port type is defined by whether the selected port is a local port or the cascade port. The address class for each port is defined by the PP bits in the allowed_configuration field of the last response training frame returned before the port transitions to the active state.

- Local. Active local ports may be configured to receive:
  - Packets with an exact address match.
  - All packets (promiscuous mode).
- Cascade. Active cascade ports will always receive all packets (promiscuous mode).

Port meta-state. The meta-state of the port determines whether the port can receive traffic, and the type of frames that can be sent and received through it. The meta-state of each local port is evaluated each time Incoming is sent:

- **Local ports:**
  - **Active.** Active ports can receive all frame types and are considered for possible address match before frame transmission. The active state is evaluated at the time Incoming is sent out for this packet. A port that is not active at the time Incoming is first sent out and when the address lookup occurs is not to be sent the packet.
  - **Inactive.** Inactive ports cannot send or receive any traffic and are removed from address match determination.
  - **Training.** A local port in training may receive only training frames generated by the lower entity connected to that port, and may transmit only a training frame that is in response to the training frame just received through that port.
- **Cascade port:** The meta-states of uplinks connected to a cascade port are considered on a link-by-link basis:
  - **Active.** An active uplink may send and receive all frame types.
  - **Disabled.** A disabled uplink has its associated transmitters disabled and is not able to access the network in any fashion. It will remain in this meta-state until network management re-enables the uplink. (This meta-state is only implemented in repeaters that are equipped with redundant uplinks.)
  - **Standby.** A standby uplink responds to incoming control signals but may not send outgoing data frames and ignores all incoming data frames. (This meta-state is only implemented in repeaters that are equipped with redundant uplinks.)
  - **Training.** An uplink in training may only send its own RMAC-generated training frames and will only receive responses to those training frames. (The higher-level repeater will ensure that no other traffic is sent to a lower-level uplink in training.)
  - **Wait_and_Test.** An uplink in the Wait_and_Test meta-state may neither send nor receive data traffic and is removed from address match consideration. (This meta-state is only implemented in repeaters that are equipped with redundant uplinks.)

Frame type. Indication of the frame type is given by the values of the input control variables, by the port meta-state, and by the destination address of the frame:

— Data frames. Frames received on active ports will normally be data frames and will always contain a non-null destination address.

— Training frames. Training frames will always contain a null destination address. They:

- Can be generated by the RMAC for training the uplink at the cascade port. This frame is sent to the cascade port only.
- Will be received at the cascade port while it is in training. This is the uplink training response frame and it is sent to all promiscuous local ports.
- Will be received at local ports in training. The training response frame is sent to the selected port and all active promiscuous ports.
- Can be received at the cascade port (if it is active) or at any active local port connected to a repeater. Training frames received at active ports are treated in a similar way to void frames and are sent to all promiscuous ports except the selected port.

— Void frames. Void frames can be:

- Received at any active port connected to a repeater.
- Generated by the RMAC to replace an erroneous or late incoming frame [see 12.9.4, items b) and c)].
- Received at a port in training.

Void frames are sent to all promiscuous ports except the selected port.

Destination address type. Destination addresses can be individual or group. Destination address field formats are defined in 10.2.1.1 for ISO/IEC 8802-3 frames, and in 10.3.3.1 for ISO/IEC 8802-5 frames.

12.9.7.2 Frame transmission timing considerations

Normally, frame transmission can be overlapped with frame reception. Figure 12-37 shows a time-space diagram of normal packet transmission. For frames that are not stored and forwarded, the transmission of a frame shall be initiated such that the first bit of the ssd at the transmitting MII occurs within 4.5 µs of the receipt of the first bit of the ssd at the receiving MII unless the IPG spacing requires the frame to be delayed. For frames that are stored and forwarded, the transmission of a frame shall be initiated such that the first bit of the ssd at the transmitting MII occurs within 4.5 µs of the receipt of the last bit of esd at the receiving MII.

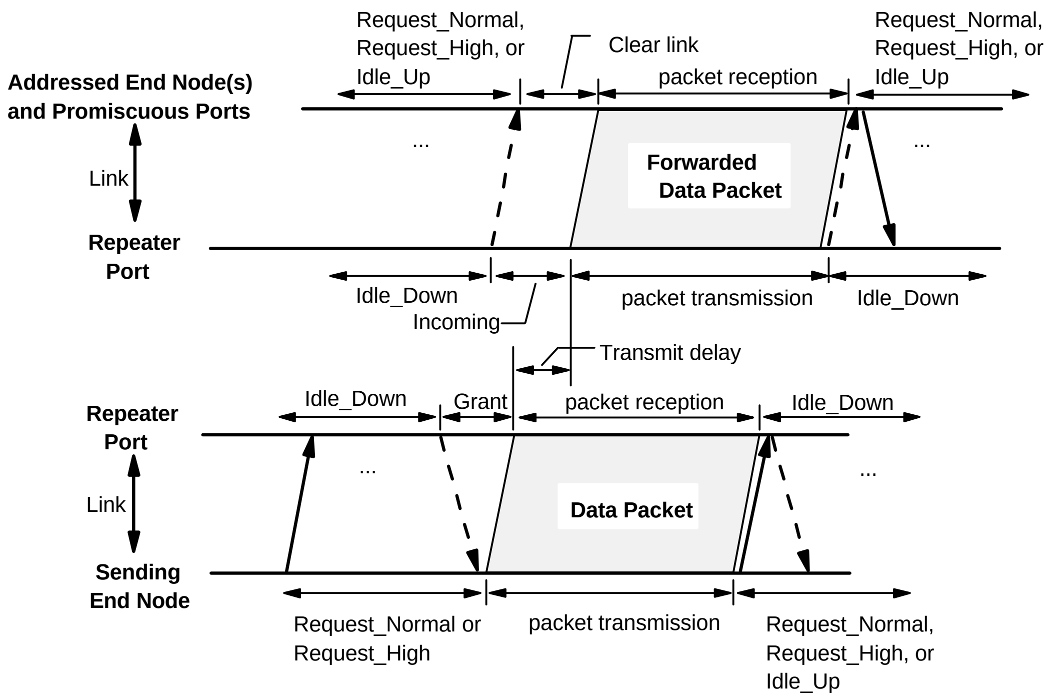


Figure 12-37—Normal overlapped frame transmission

There are two conditions where delays are required:

a) 4-UTP link in bundled cable. If the frame is received at a port that is configured with a 4-UTP link installed in bundled cable and the destination address is a group address, transmission cannot start until the frame has been completely received and stored in the RMAC buffer. This delay is shown in Figure 12-38.

b) Local port being trained. Transmission cannot start until after a half-duplex turnaround delay from the end of the received training frame, because the training response frame must be returned to the sending end node. Time must be provided for Incoming to propagate down to the end node and for the control response from the end node to propagate back up to the repeater. This delay is shown in Figure 12-39.

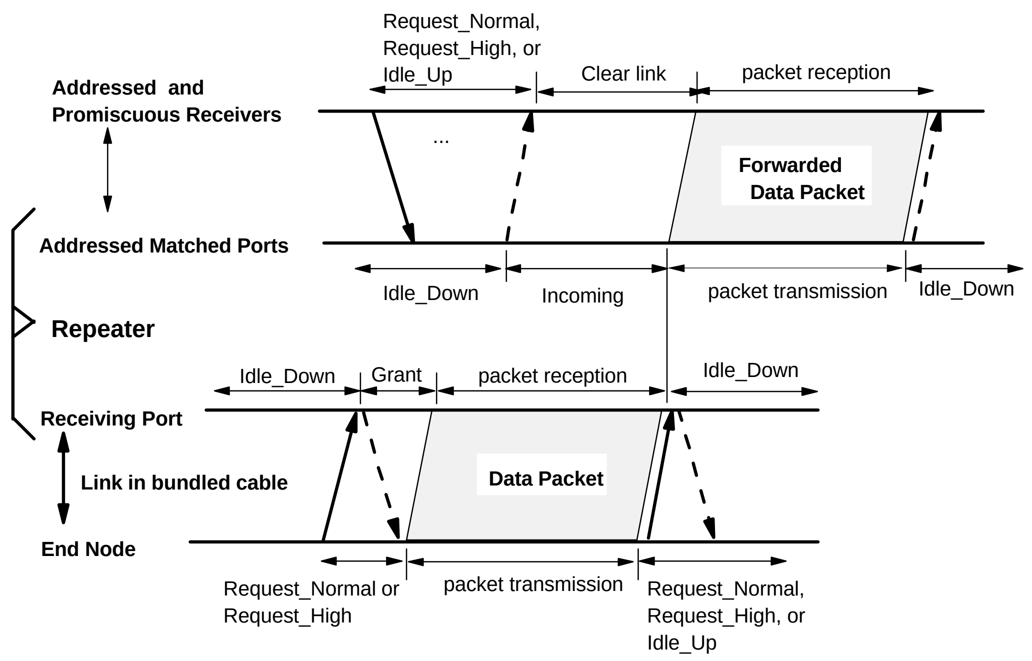
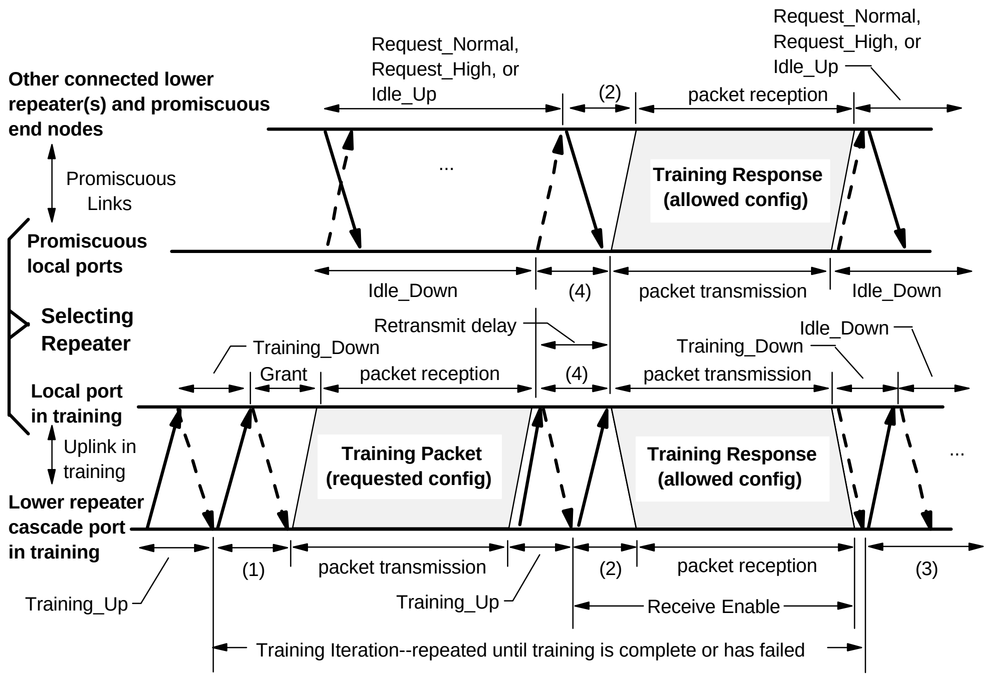


Figure 12-38—Store and forward frame transmission delay


Other connected lower repeater(s) and promiscuous end nodes
Request_Normal, Request_High, or Idle_Up
(2)
packet reception
Request_Normal, Request_High, or Idle_Up
Promiscuous Links
...
Training Response (allowed config)
Promiscuous local ports
Idle_Down
(4)
packet transmission
Idle_Down
Selecting Repeater
Retransmit delay
Training_Down
Idle_Down
Training_Down
Grant
packet reception
(4)
packet transmission
Local port in training
Uplink in training
Training Packet (requested config)
Training Response (allowed config)
...
Lower repeater cascade port in training
(1)
packet transmission
(2)
packet reception
(3)
Training_Up
Training_Up
Receive Enable
Training Iteration--repeated until training is complete or has failed

NOTES

1—Request_High

2—The control signal sent after receipt of incoming depends on the PMD/Link configuration:

a) In half-duplex links, the link is cleared of outgoing signals to prepare for incoming data.

b) In dual-simplex links, Training_Up continues to be sent during the entire period RxEn is asserted.

3—Idle_Up if training is complete, Training_Up if training has failed.

4—Incoming

Figure 12-39—Local port training frame transmission delay

12.9.7.3 Frame transmission

The BCALL starts frame transmission for uplink training frames when the frame is generated by the TULM. For the uplink training response frame and all other frames, the BCALL starts frame transmission, from the RMAC Buffer, upon receiving the event TSM_TO_BCALL_START_TRANSMIT (signaled by the TSM).

At each port for which the corresponding Address_Mask element is True:

a) A PMI_UNITDATA.request is issued, with its mac_frame parameter set to "begin" unless this was the only octet in the RMAC buffer, with its data_octet parameter set to the value of the first octet read out of the RMAC buffer, and with its priority parameter set to "high" or "normal" according to the priority parameter of the first PMI_UNITDATA.indication for the received frame or, for a generated void frame, to the selected priority.

b) Each subsequent octet is read out of the RMAC buffer, and a corresponding PMI_UNITDATA.request is issued, with its mac_frame parameter set to "more," except for the last octet.

c) The PMI_UNITDATA.request corresponding to the last octet read out of the RMAC buffer has its mac_frame parameter set to "end," and its error_indication parameter set to "valid" if the values of both EMF_INDICATION and ERROR_INDICATION are PACKET_OK; otherwise, it is set to "IPM."

d) Upon receipt of a PMI_TRANSMIT.indication primitive with the transmit_complete parameter value of "end," the BCALL signals the event S_BCALL_TO_TSM_EMF_TRANSMITTED to the TSM. If there is no Address_Mask element set to True, the event S_BCALL_TO_TSM_EMF_TRANSMITTED shall be signaled a period of SEC_CTRL_SIG_DECODE_DELAY after occurrence of the PMI_UNITDATA.indication primitive with the mac_frame parameter value of "end." This ensures that the RSM waits for sufficient time to decode a secondary control signal.

12.9.8 Local port training mode

Local port training is always initiated by the connected lower entity and follows the packet handling sequence diagrammed in Figure 12-39.

a) RMAC will transition the local port to the training mode upon receipt of a PMI_CONTROL.indication (Training_Up) and will indicate permission to proceed with training by generating a PMI_CONTROL.request (Training_Down).

b) Training of local ports connected to end nodes will be performed as part of the normal-priority round-robin cycle. Local ports connected to lower repeaters will be trained as part of a high-priority round-robin cycle.

c) Training response frames will also be forwarded to all connected repeaters and promiscuous ports.

The configuration of the lower entity is established as part of the training process. The lower entity uses the requested_configuration field of the training frame to indicate whether it is a repeater or an end node, whether it wishes to be a promiscuous receiver, and the format it wishes to use (see 10.6.2).

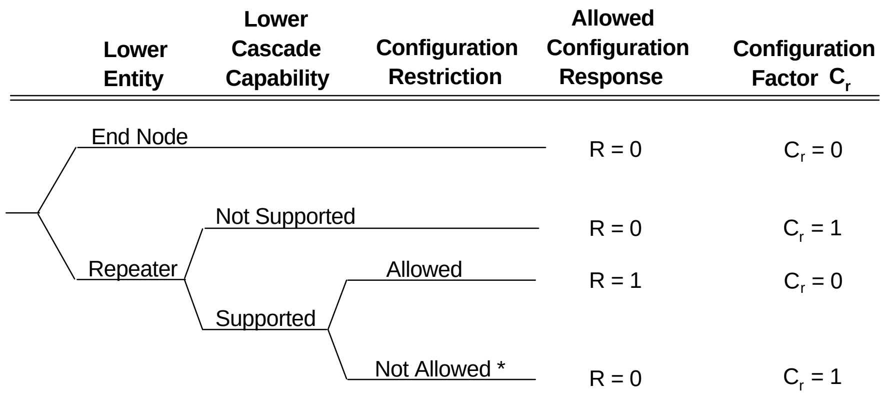
The C bit in the allowed configuration field of the training response frame (see 10.6.3) will indicate whether the lower entity's requested configuration matches the RMAC basic capabilities. The C bit shall be set according to the Boolean expression:

$$C := C_r + C_{pp} + C_{ff} + C_d + C_{na} \qquad (5)$$

where C_r, C_{pp}, C_{ff}, and C_d are the values defined in the configuration factors columns of the decision trees in Figures 12-40 through 12-43, and where:

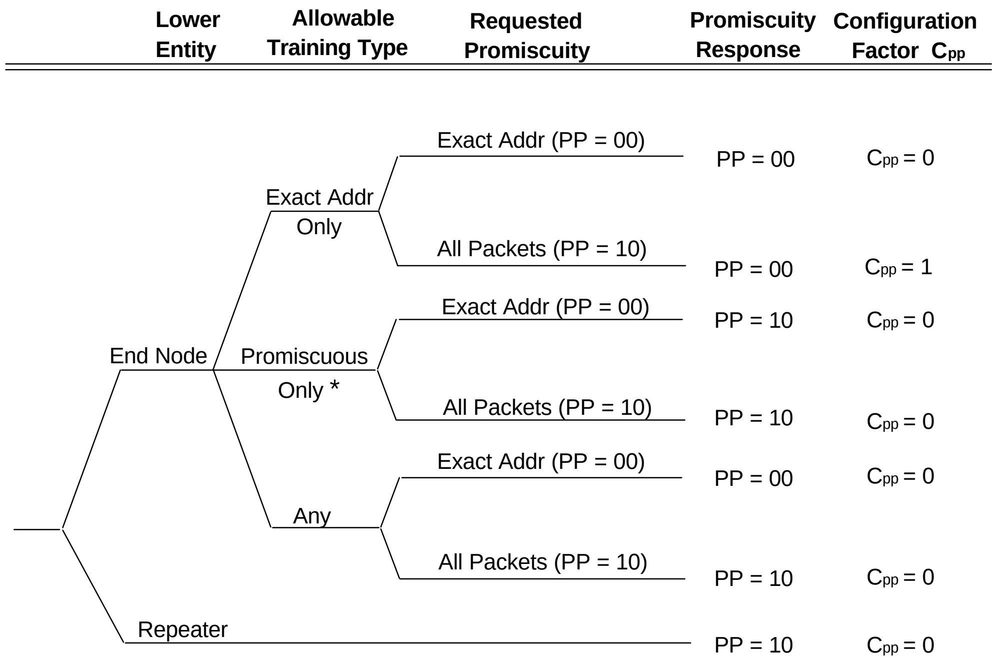
$C_{na} := 1$ if the source address of the training frame is the null address and the training initiator is an end node.

$C_{na} := 0$, otherwise.



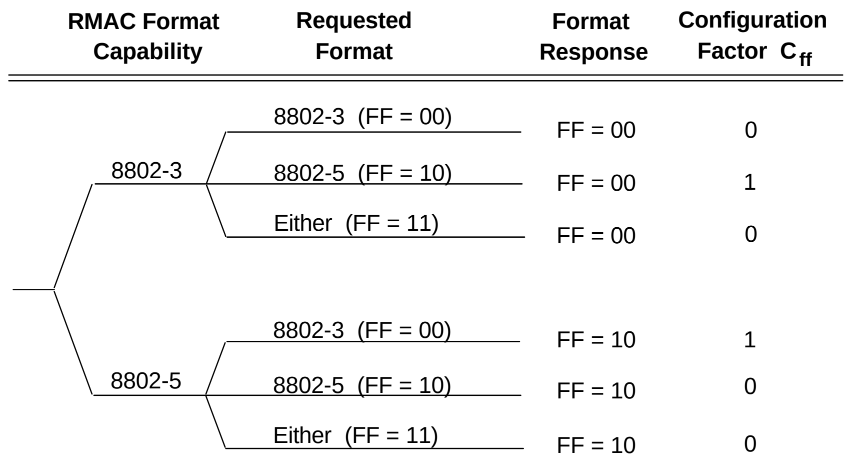
* Lower repeaters may be supported but not allowed because the network administrator has denied lower repeater access to that port using **aAllowableTrainingType**.

Figure 12-40—Requested lower entity decision tree (R bit)



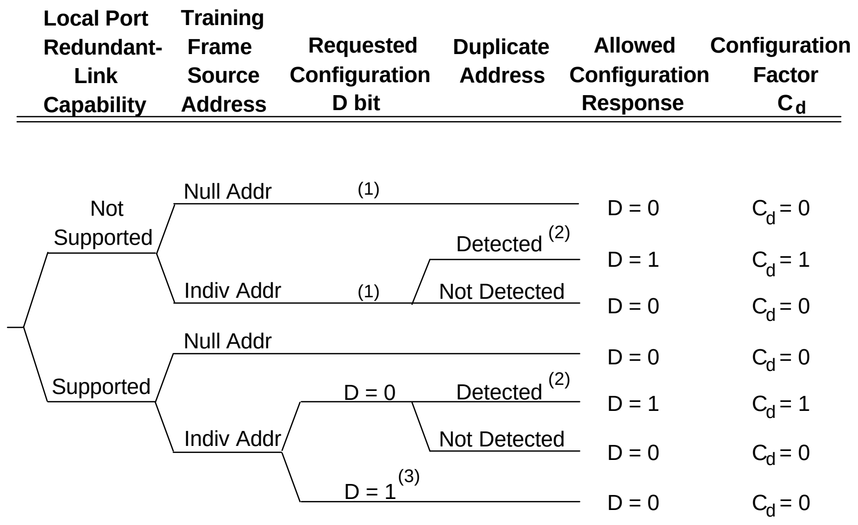
* This behavior is used when **aAllowableTrainingType** is "Any." but **aSupportedTrainingCapability** indicates that the hardware supports promiscuous mode only.

Figure 12-41—Requested promiscuity decision tree (PP bits)



NOTE—If the requested format is the reserved code 01, the format response will indicate the RMAC format capability and C_{ff} := 1

Figure 12-42—Requested format decision tree (FF bits)



NOTES

1— An RMAC without redundant-link capability for local-port training will ignore the D bit in the requested configuration field of the training frame. All downlinks will be treated as though they were nonredundant links.

2— A duplicate address will be detected:

a) If the source address of the received training frame is equal to any non-null address on the RMAC's port address list and duplicate address checking is implemented and enabled.

b) If the network management attribute **aCentralMgmtDetectedDupAddr** is TRUE.

3— An RMAC with redundant-link capability for local-port training will recognize a D bit with a value of one in the requested configuration field as indicating that the link is being trained as a redundant link. Duplicate address checking, if implemented, will be disabled for that local port (see 10.6.2.7).

Figure 12-43 Duplicate-address/redundant-link decision tree (D bit)

The lower entity will be allowed to change its requested configuration during the training process (e.g., from ISO/IEC 8802-3 format to ISO/IEC 8802-5, or from promiscuous reception to exact address match only). However, the lower entity will not be allowed to join the network if the value of C is one. VALID_CONFIG will be set to False to advise the PPM of the continued restriction.

The N bit in the configuration response field has been provided to allow the RMAC to refuse training for reasons other than configuration mismatch (such as security restrictions). Conditions for setting the N bit are an implementation issue and are beyond the scope of this standard. The N bit shall be set to zero unless specific uses have been defined and included in a particular implementation.

Training will continue until receipt of a PMI_CONTROL.indication (Idle_Up) indicating that the lower entity has received 24 consecutive error-free training responses. If all compatibility mismatches have been resolved and if there is no other reason the lower entity should be denied access to the network (C and N will both be zero), VALID_CONFIG will be set to True, and the PPM will generate a PMI_CONTROL.indication (Idle_Down) acknowledging that training is successful and that the lower entity has been allowed to join the network.

When the PPM determines that training on a local port has been successful, it will change the LOCAL_PORT_META_STATE from TRAINING to ACTIVE. Then, on sensing the TRAINING to ACTIVE transition, the BCALL loads the (PORT_NUM, Address) pair into the Port Address List. PORT_NUM is the number of the port, and Address is the Source Address field of the training frame. If the Port later transitions to INACTIVE, the entry (PORT_NUM, Address) in the Port Address List is deleted.

12.10 Fault Control Process (FCP) state machine

The FCP is in the data path between the PPM of the currently selected port (which may be the cascade port) and the input to the RMAC buffer in the BCALL. The FCP shields the RSM and the TULM from knowing about expected frames failing to turn up because of error. The FCP also sends signals to, and receives signals from, the RSM or the TULM (if the cascade port is in training). The relationship of the FCP to other RMAC state machines is shown in Figure 12-44 and the FCP state machine is shown in Figure 12-45. In any state except FCPSTATE1, a signal S_TO_FCP_STOP_EXPECTING_FRAME from the RSM or the TULM will cause the FCP to transition to its quiescent state, FCPSTATE1.

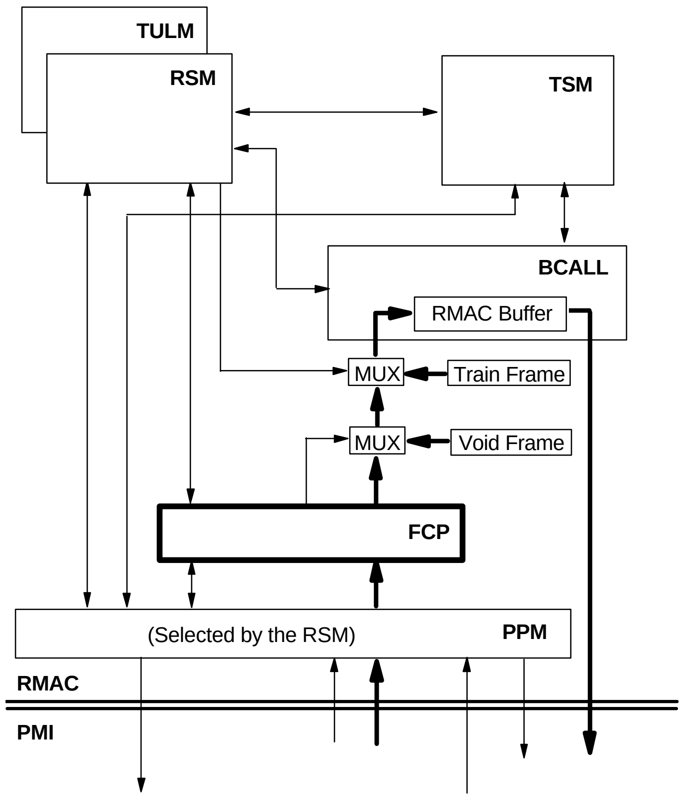


Figure 12-44—Relationship of the FCP to other RMAC state machines

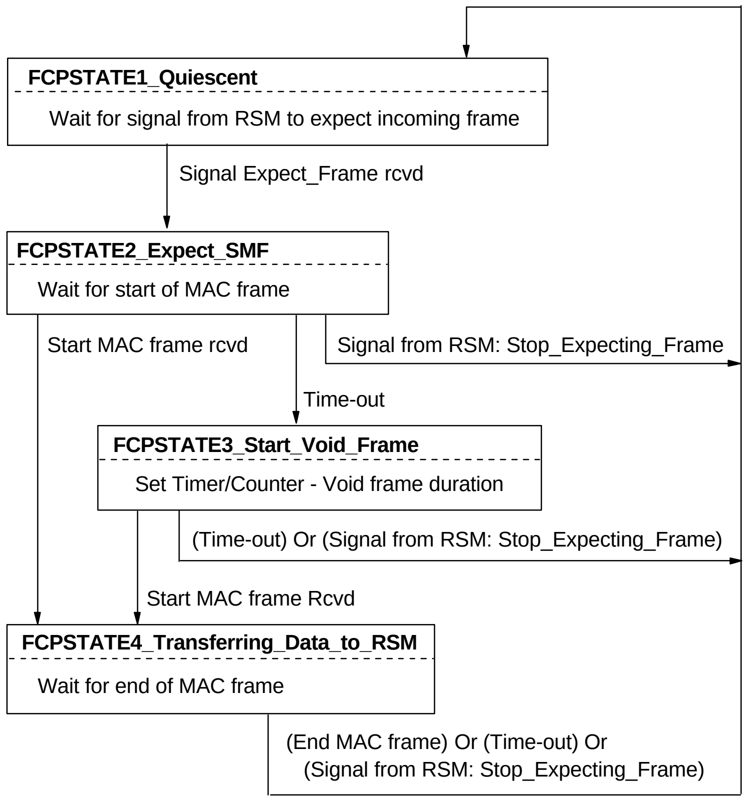

FCPSTATE1_Quiescent
Wait for signal from RSM to expect incoming frame
Signal Expect_Frame rcvd
FCPSTATE2_Expect_SMF
Wait for start of MAC frame
Start MAC frame rcvd
Signal from RSM: Stop_Expecting_Frame
Time-out
FCPSTATE3_Start_Void_Frame
Set Timer/Counter - Void frame duration
(Time-out) Or (Signal from RSM: Stop_Expecting_Frame)
Start MAC frame Rcvd
FCPSTATE4_Transferring_Data_to_RSM
Wait for end of MAC frame
(End MAC frame) Or (Time-out) Or
(Signal from RSM: Stop_Expecting_Frame)

Figure 12-45—The FCP state machine

The RSM sets SELECTED_PORT to the port where the frame is expected. SELECTED_PORT indicates either a local port number or the cascade port.

The BCALL is responsible for ensuring that not more than the maximum number of octets is pushed into the RMAC buffer. There is also a timer in the FCP of a slightly longer duration, LONG_RECEIVE_INITIATE_TRAINING, which is used to detect over-length frames that have not even been terminated by the arrival of control signals. If the LONG_RECEIVE_INITIATE_TRAINING timer expires, the link is retrained.

FCP local variables:

FLAG_GRANT_STATE_ZERO : Boolean; Indicates that the current GRANT_STATE is “zero.”

FLAG_SIGNAL_RECEIVED_EXPECT_FRAME : Boolean; Indicates that the signal EXPECT_FRAME from the RSM or the TULM has been received by the FCP.

FCP Procedure

```
PROCEDURE NO_SMF_ARRIVED(IN_PORT : T_PORT);

  Begin
   {Expecting frame too long, without SMF being received.}
```

```
    EMF_INDICATION := INVALID_PACKET_MARKER;
    FCP_DATA_OUTPUT := ZERO_OCTETS;
    {Tell RSM / TULM about start of frame—in this case, the void frame.}

    If (IN_PORT = CASCADE_PORT) Then
      SIGNAL_TO_RSM_TULM (S_FCP_SMF_RECEIVED_THRU_CASCADE_PORT)
    Else
      SIGNAL_TO_RSM_TULM (S_FCP_SMF_RECEIVED_THRU_LOCAL_PORT);
    SET_FAULT_TIMER(LONG_RECEIVE_INITIATE_TRAINING);
    SET_TIMER(VOID_FRAME);
    {Stop sending void frame when EMF primitive arrives or on long receive.}

    FCPSTATE := FCPSTATE3_START_VOID_FRAME;
  End;
```

12.10.1 FCPSTATE1_QUIESCENT

The FCP waits in state FCPSTATE1 when there is no network activity. When the RSM or the TULM realizes that a frame is about to arrive, the signal EXPECT_FRAME is sent to the FCP.

Upon receiving this signal, the FCP checks that GRANT_STATE is 0 and if necessary, waits for this to occur. It then asserts Receive Enable, and goes to FCPSTATE2.

```
Begin
  Case EVENT_SORT Of
    S_TO_FCP_EXPECT_FRAME :
    {Informed by RSM or TULM to expect some sort of frame.}
    Begin
        {Next few lines of code are so that GRANT_STATE is 0 when RECEIVE_ENABLE is
         pulled. This ensures that control signals are being received, and avoids
         RECEIVE_ENABLE being pulled too soon and the possibility of false preamble
detect.}
      FLAG_SIGNAL_RECEIVED_EXPECT_FRAME := True;
      If (SELECTED_PORT_GRANT_STATE_IS_ZERO()) Then
        Begin
          SILENT_WHEN_GRANTED := False
          FLAG_GRANT_STATE_ZERO := True
        End
      Else  Begin
        SILENT_WHEN_GRANTED := True
        FLAG_GRANT_STATE_ZERO := False;
        End
      SET_FAULT_TIMER(FRAME_EXPECTED_TO_SMF_IN);
    End;

    CASCADE_PORT_CONTROL_SIGNAL_DECODED,
    LOCAL_PORT_CONTROL_SIGNAL_DECODED :
    If (SELECTED_PORT_GRANT_STATE_IS_ZERO()) Then
      FLAG_GRANT_STATE_ZERO := True;

   TIMER_EXPIRES :
    If (TIMER = FRAME_EXPECTED_TO_SMF_IN) Then
    Begin
```

```
       FLAG_FCP_TO_PPM_FRAME_ENDED := False;
       NO_SMF_ARRIVED(PORT);
     End;
  End; **{Case}**

If ((FLAG_SIGNAL_RECEIVED_EXPECT_FRAME = True) And
      (FLAG_GRANT_STATE_ZERO = True)) Then
  Begin
    ASSERT_RECEIVE_ENABLE(SELECTED_PORT);
    EMF_INDICATION := PACKET_OK;
    {EMF Indication on packet sent out by TSM. Initially assume packet is OK. Change this if
     necessary later on.}
    FLAG_FCP_TO_PPM_FRAME_ENDED := False;
    FCPSTATE := FCPSTATE2_EXPECT_SMF;
  End
End;
```

12.10.2 FCPSTATE2_EXPECT_SMF

In FCPSTATE2, the repeater is receiving Preamble and is waiting for the start of MAC frame (see Figure 12-46).

a) If the start_mac_frame_status is "valid," the FCP sets the lower data-path multiplexer to transfer octets from the FCP.

b) If the start_mac_frame_status is "invalid," the FCP sets the lower data-path multiplexer to transfer void frame octets.

In both cases, the FCP transitions to FCPSTATE4. If no SMF arrives, the FCP starts generating a void frame, and transitions to FCPSTATE3.

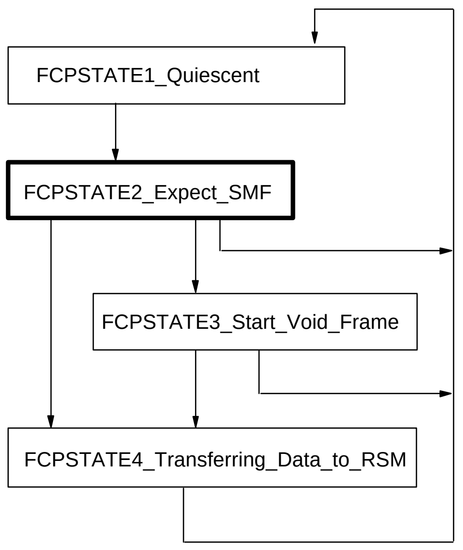

FCPSTATE1_Quiescent
FCPSTATE2_Expect_SMF
FCPSTATE3_Start_Void_Frame
FCPSTATE4_Transferring_Data_to_RSM

Figure 12-46—FCPSTATE2 relationship within the state machine

```
Case EVENT_SORT Of
  SMF_RECEIVED_THRU_PORT, BAD_SMF_RECEIVED_THRU_PORT :
  Begin {Fault-free case. Expecting Start of MAC Frame, and it arrives.}
    If (EVENT_SORT = BAD_SMF_RECEIVED_THRU_PORT) Then
    Begin
      EMF_INDICATION := INVALID_PACKET_MARKER;
      FCP_DATA_OUTPUT := ZERO_OCTETS;
    End Else
      FCP_DATA_OUTPUT := OCTETS_FROM_PMI;
    If (SELECTED_PORT = CASCADE_PORT) Then
      SIGNAL_TO_RSM_TULM (S_FCP_SMF_RECEIVED_THRU_CASCADE_PORT)
    Else SIGNAL_TO_RSM_TULM
           (S_FCP_SMF_RECEIVED_THRU_LOCAL_PORT);
    {Tell RSM / TULM start of frame has arrived.}
    CANCEL_FAULT_TIMER(FRAME_EXPECTED_TO_SMF_IN);
    SET_FAULT_TIMER(LONG_RECEIVE_INITIATE_TRAINING);
    FCPSTATE := FCPSTATE4_TRANSFERRING_DATA_TO_RSM;
  End;

  TIMER_EXPIRES :
  If (TIMER = FRAME_EXPECTED_TO_SMF_IN) Then
     NO_SMF_ARRIVED(PORT);

  S_TO_FCP_STOP_EXPECTING_FRAME :
  Begin
    {This signal is sent if the RSM is expecting a frame and then receives a control signal
     (e.g., if a lower requester is selected but then transitions from Request to IDLE_U).}
    DEASSERT_RECEIVE_ENABLE(SELECTED_PORT);
    CANCEL_FAULT_TIMER(FRAME_EXPECTED_TO_SMF_IN);
    FLAG_SIGNAL_RECEIVED_EXPECT_FRAME := False;
    FCPSTATE := FCPSTATE1_QUIESCENT;
  End;
End; {Case}
```

12.10.3 FCPSTATE3_START_VOID_FRAME

When the FCP enters FCPSTATE3, it starts the LONG_RECEIVE_INITIATE_TRAINING timer and waits until the timer expires or a start of MAC frame arrives (see Figure 12-47).

a) If a start of MAC frame arrives before the timer has expired, the lower data-path multiplexer is changed to transfer octets from the FCP, the timer is canceled, and the FCP transitions to FCPSTATE4 to wait for the end of MAC frame. The frame is extended to the duration of the actual frame.

b) If the timer has expired, the void frame is terminated as if it were an ordinary frame and the FCP transitions to FCPSTATE1.

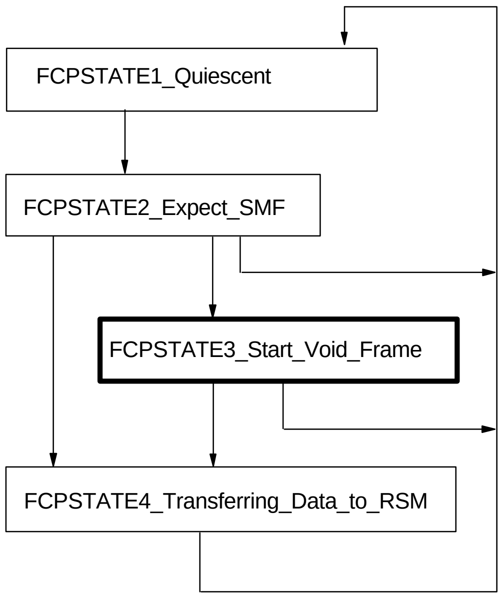

FCPSTATE1_Quiescent
FCPSTATE2_Expect_SMF
FCPSTATE3_Start_Void_Frame
FCPSTATE4_Transferring_Data_to_RSM

Figure 12-47—FCPSTATE3 relationship to the state machine

```
Case EVENT_SORT Of
  SMF_RECEIVED_THRU_PORT, BAD_SMF_RECEIVED_THRU_PORT :
  Begin {Sending void frame, but lower repeater may have started void frame at
         later time.}
      CANCEL_TIMER(VOID_FRAME);
      {FCP_DATA_OUTPUT := OCTETS_FROM_PMI; optional here.}
      FCPSTATE := FCPSTATE4_TRANSFERRING_DATA_TO_RSM;
  End;

  TIMER_EXPIRES :
  If (TIMER = VOID_FRAME) Then
  Begin          {Have finished sending Void Frame.}
    DEASSERT_ RECEIVE_ENABLE(SELECTED_PORT);
    If (SELECTED_PORT = CASCADE_PORT) Then
    Begin
      SIGNAL_TO_RSM_TULM (S_FCP_EMF_RECEIVED_THRU_CASCADE_PORT);
      INVOKE_CASCADE_PORT_TRAINING();
    End
    Else Begin
      SIGNAL_TO_RSM_TULM (S_FCP_EMF_RECEIVED_THRU_LOCAL_PORT);
      INVOKE_TRAINING(SELECTED_PORT);
    End;
    FCP_DATA_OUTPUT := NIL; {Stop pushing void frame octets to RSM/TULM.}
    CANCEL_FAULT_TIMER(LONG_RECEIVE_INITIATE_TRAINING);
    {Have started timer, need to cancel. EMF_INDICATION already set to
      INVALID_PACKET_MARKER.}
    FLAG_SIGNAL_RECEIVED_EXPECT_FRAME := False;
    FCPSTATE := FCPSTATE1_QUIESCENT;
  End;

  S_TO_FCP_STOP_EXPECTING_FRAME :
  Begin
    DEASSERT_RECEIVE_ENABLE(SELECTED_PORT);
    FCP_DATA_OUTPUT := NIL; {Stop pushing anything to RSM/TULM.}
    CANCEL_FAULT_TIMER(LONG_RECEIVE_INITIATE_TRAINING);
    FLAG_SIGNAL_RECEIVED_EXPECT_FRAME := False;
    FCPSTATE := FCPSTATE1_QUIESCENT;
  End;
End; {Case}
```

12.10.4 FCPSTATE4_TRANSFERRING_DATA_TO_RSM

In FCPSTATE4, the lower data-path multiplexer is transferring octets from the PMI (through the FCP) to the BCALL (see Figure 12-48).

a) When the end of MAC frame arrives, the FCP stops transferring octets, and goes to FCPSTATE1.
b) If the LONG_RECEIVE_INITIATE_TRAINING timer expires, the output from the FCP stops, and training is initiated. FCP goes to FCPSTATE1.

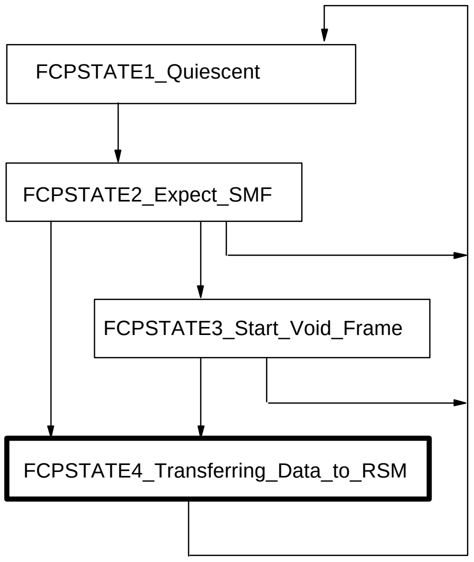

FCPSTATE1_Quiescent
FCPSTATE2_Expect_SMF
FCPSTATE3_Start_Void_Frame
FCPSTATE4_Transferring_Data_to_RSM

Figure 12-48—FCPSTATE4 relationship to the state machine

```
Case EVENT_SORT Of
  EMF_RECEIVED_THRU_PORT, FRAME_ENDED_ABNORMAL_END :
  Begin
    FCP_DATA_OUTPUT := NIL; {Stop pushing anything to RSM/TULM.}
    DEASSERT_RECEIVE_ENABLE(SELECTED_PORT);
    If (SELECTED_PORT = CASCADE_PORT) Then
      SIGNAL_TO_RSM_TULM (S_FCP_EMF_RECEIVED_THRU_CASCADE_PORT)
    Else
      SIGNAL_TO_RSM_TULM (S_FCP_EMF_RECEIVED_THRU_LOCAL_PORT);
    CANCEL_FAULT_TIMER(LONG_RECEIVE_INITIATE_TRAINING);
    If (EVENT_SORT = FRAME_ENDED_ABNORMAL_END) Then
      EMF_INDICATION := INVALID_PACKET_MARKER;
    FLAG_SIGNAL_RECEIVED_EXPECT_FRAME := False;
    FCPSTATE := FCPSTATE1_QUIESCENT;
  End;
  TIMER_EXPIRES :
  If (TIMER = LONG_RECEIVE_INITIATE_TRAINING) Then
  Begin
    EMF_INDICATION := INVALID_PACKET_MARKER;
    FLAG_FCP_TO_PPM_FRAME_ENDED := True;
    DEASSERT_RECEIVE_ENABLE(SELECTED_PORT);
    If (SELECTED_PORT = CASCADE_PORT) Then
    Begin
      SIGNAL_TO_RSM_TULM (S_FCP_EMF_RECEIVED_THRU_CASCADE_PORT);
      INVOKE_CASCADE_PORT_TRAINING();
    End
    Else Begin
      SIGNAL_TO_RSM_TULM (S_FCP_EMF_RECEIVED_THRU_LOCAL_PORT);
      INVOKE_TRAINING(SELECTED_PORT);
    End;
    FCP_DATA_OUTPUT := NIL; {Stop pushing anything to RSM/TULM.}
    FLAG_SIGNAL_RECEIVED_EXPECT_FRAME := False;
    FCPSTATE := FCPSTATE1_QUIESCENT;
  End;

  S_TO_FCP_STOP_EXPECTING_FRAME :
  Begin
    DEASSERT_RECEIVE_ENABLE(SELECTED_PORT);
    FCP_DATA_OUTPUT := NIL; {Stop pushing anything to RSM/TULM.}
    CANCEL_FAULT_TIMER(LONG_RECEIVE_INITIATE_TRAINING);
    FLAG_SIGNAL_RECEIVED_EXPECT_FRAME := False;
    FCPSTATE := FCPSTATE1_QUIESCENT;
  End;
End; {Case}
```

12.11 Port Process Module (PPM)

The PPM is the only RMAC functional entity that needs to be replicated. It is the primary interface between the PMI and the RSM, TSM, TULM, and FCP. One PPM is required for the cascade port and one for each local port. The definition of the PPM for the cascade port differs slightly from the PPM definition for local ports. The relationship of the PPM to the other RMAC state machines is shown in Figure 12-49.

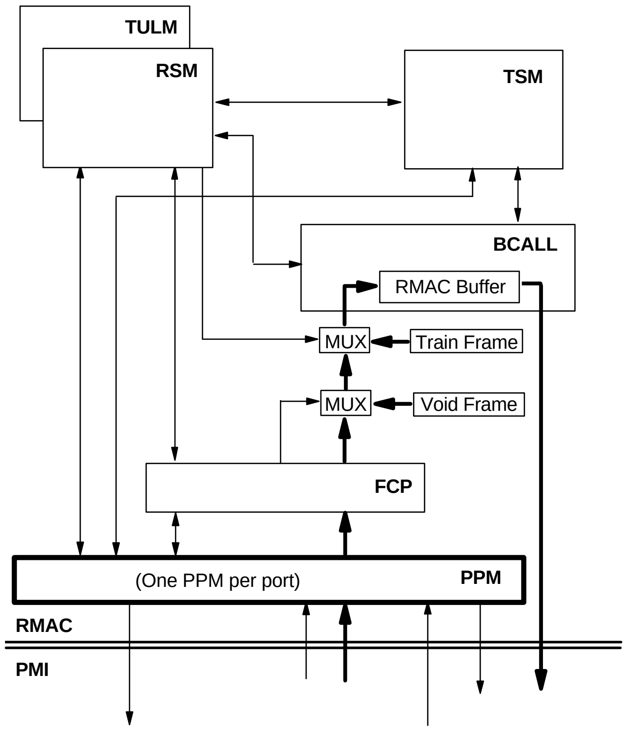


Figure 12-49—Relationship of PPM to other RMAC state machines

The major functions of the PPM are:

— Generating PMI_CONTROL.request primitives that define the desired transmit control state in response to procedure calls by the RSM, TSM, or TULM.
  - Procedure CHANGE_LOCAL_PORT_CONTROL_SIGNAL generates the primitives for local ports. The desired control state is held in the local port PPM register LOCAL_PORT_CONTROL_SIGNAL_OUT_REG.
  - Procedure CHANGE_CASCADE_PORT_CONTROL_SIGNAL generates the primitives for cascade ports. The desired control state is held in the cascade port PPM register CASCADE_PORT_CONTROL_SIGNAL_OUT_REG.

— Receiving PMI_CONTROL.indication primitives defining the receiver control states and grant state for the port and informing the RSM, TSM, and TULM. These control signals are identified by the parameters RECEIVE_CONTROL_STATE and GRANT_STATE.
  - If GRANT_STATE is one, GRANT is being received, and RECEIVE_CONTROL_STATE contains the control state value that was received prior to grant.
  - If GRANT_STATE is zero, RECEIVE_CONTROL_STATE contains the current control state value.

— Receiving PMI_UNITDATA.indication primitives from the PMI defining the value of incoming data octets and their associated status parameters.

The PPM also provides other critical functionalities that depend on the meta-state of the port, any restrictions imposed by the LME, and/or the time that a normal-priority request has been pending. For local ports:

— If the meta-state is INACTIVE, the transmit control state will be set to TxDisable:

  - If the LME has set the PORT_STATE to DISABLED, the meta-state will remain INACTIVE.
  - If PORT_STATE is ENABLED and TRAINING_UP is received, the PPM will return TRAINING_DOWN to the lower entity, and will change the meta-state to TRAINING.
- If the meta-state is TRAINING, the transmit control state:
  - INCOMING will be sent only if it is to precede the training frame response packet; otherwise, it will be converted to TRAINING_DOWN.
  - IDLE_D will be converted to TRAINING_DOWN if training is not complete.
  - IDLE_D will be sent only if the criteria for successful training of the link has been met, and IDLE_U has been received from the lower entity (indicating that the lower entity also considers training successfully completed). The meta-state will then be changed to ACTIVE.
  - Detection of a retraining error will cause the meta-state to become INACTIVE and the transmit control state to be set to TxDisable (which, in turn, will force the lower entity to reinitiate training).
- If the meta-state is ACTIVE, the PPM will process data traffic.
  - The RMAC_REQ_REG is maintained by the local port PPMs to indicate the level of request currently being received at the local port (high, normal, or none).
  - Detection of a retraining error will cause the meta-state to become INACTIVE and the transmit control state to be set to TxDisable (and will force the lower entity to initiate training).
  - If TRAINING_UP is received, then the meta-state will be changed to TRAINING.
- Local Port PPMs contain a PRIORITY_ENABLE bit. If this is set to False, the PPM will treat a REQUEST_HIGH from an end node as a REQUEST_NORMAL (high-priority requests are disabled and treated as normal-priority requests).
- Local port PPMs contain a packet priority promotion timer that is set when the PPM first detects a new normal-priority request (or a new high-priority request if PRIORITY_ENABLE is False). If the request has been pending for a period PACKET_PROMOTION, then PP_ON is set to True (by an autonomous event outside of the main “Forever” loop.) When PP_ON becomes True, the RMAC will treat this request as high priority.

For the cascade port, the PPM is similar but does not include PRIORITY_ENABLE, Packet Promotion, or link training (training of the uplink is the responsibility of the Train Uplink Module). The RMAC_ENABLE_HIGH_ONLY_RECEIVED_REG is maintained by the cascade port PPM to indicate whether ENABLE_HIGH_ONLY is being received.

The PPM also sends signals to the RSM, TSM, and TULM to inform them of changes in the control signals being received.

12.11.1 PPM local variables

One set of the following variables exists for each port (local ports and the cascade port) on the repeater.

GRANT_STATE : 0...1; Indicates the current grant state.

OLD_GRANT_STATE : 0...1; Indicates the previous grant state.

RECEIVER_CONTROL_STATE : T_CONTROL_STATE; Indicates the current receiver control state.

OLD_RECEIVER_CONTROL_STATE : T_CONTROL_STATE; Indicates the previous receiver control state.

PORT_DATA_STATE : T_PORT_DATA_STATE; Used by the “something changed” routine to indicate whether control signals are currently being received by the PMD.

The following is present only for the cascade port:

CASCADE_PORT_CONTROL_SIGNAL_OUT_REG : T_CONTROL_STATE; Indicates the control signal that is to be sent through the cascade port.

The following are present only for local ports:

LOCAL_PORT_CONTROL_SIGNAL_OUT_REG : T_CONTROL_STATE; Indicates the control signal that is to be sent through the identified local port.

PORT_X : T_LOCAL_PORT; Indicates the port number of the particular PPM.

PP_TIMER_STARTED : Boolean; Indicates that the packet-promotion timer has been started.

PP_ON : Boolean; Indicates that the packet-promotion timer has expired.

12.11.2 Local functions and procedures

The following functions and procedures are used only by PPM.

Procedure UpdateLmeRmacRxCounters; This is an LME procedure. See 13.2.4.1.3.

Function IS_CRC_Correct() : Boolean; This function checks the incoming frame to determine if it contains a CRC error.

Procedure Initialize_CRC(); This procedure initializes the CRC check variable prior to the receipt of an incoming frame.

Procedure PMI_CONTROL_indication (VAR GRANT : 0..1, VAR RCS : T_CONTROL_STATE); This procedure interrogates the grant state and receiver control state parameter values contained in the PMI_CONTROL.indication primitive generated each clock cycle by the PMI.

Procedure PMI_CONTROL_request (TCS : T_CONTROL_STATE); This procedure generates a PMI_CONTROL.request primitive each cycle of the TxClk.

Procedure Push_to_RMB_Input (Data : T_OCTET); This procedure transfers either data octets or zero octets to the FCP during frame reception.

12.11.3 RMAC transmits control signal—Local port

Each local port PPM has a register, LOCAL_PORT_CONTROL_SIGNAL_OUT_REG, that is updated by the RSM and TSM calls to the procedure CHANGE_LOCAL_PORT_CONTROL_SIGNAL.

```
PROCEDURE CHANGE_LOCAL_PORT_CONTROL_SIGNAL
        (IN_PORT : T_LOCAL_PORT, IN_CONTROL_SIGNAL : T_CONTROL_STATE);

Begin
  LOCAL_PORT_CONTROL_SIGNAL_OUT_REG[IN_PORT] := IN_CONTROL_SIGNAL;
    {LOCAL_PORT_CONTROL_SIGNAL_OUT_REG refers to the port # IN_PORT.}
  If ((IN_CONTROL_SIGNAL = GRANT) And (RECEIVER_CONTROL_STATE=REQ_N) Or
    (PRIORITY_ENABLE=FALSE)) Then
  Begin {Have to initialize packet promotion flags.}
    PP_TIMER_STARTED := False;
    PP_ON := False;
    CANCEL_TIMER(PACKET_PROMOTION);
  End
End;

Forever Do {Every cycle of TxClk send a PMI_CONTROL.request primitive to the PMI.}
Begin
  Wait_Until(CLOCK_TICK);  {CLOCK_TICK implies the next cycle of TxClk.}
  Case LOCAL_PORT_META_STATE Of
    INACTIVE : {If port INACTIVE send silence}
      PMI_CONTROL_request(TXDISABLE);

    TRAINING :
    Case LOCAL_PORT_CONTROL_SIGNAL_OUT_REG Of
      IDLE_D, TRAINING_DOWN :
        PMI_CONTROL_request(TRAINING_DOWN);
        {IDLE_DOWN is changed to TRAINING_DOWN. The switch here avoids the
         necessity of multiple references to IDLE_D and TRAINING_D in the
         RSM. Putting the switch here therefore simplifies the RSM.}

      INCOMING :
        {If a training packet has just been received, it is necessary to be prepared to send
         INCOMING to sender. In other situations, there is no need to send INCOMING out of
         port in training.}
      If (SELECTED_PORT = PORT_X) Then
        PMI_CONTROL_request(INCOMING)
      Else PMI_CONTROL_request(TRAINING_DOWN);
      GRANT :
        PMI_CONTROL_request(GRANT);
    End; {Case}

    ACTIVE :
      PMI_CONTROL_request(LOCAL_PORT_CONTROL_SIGNAL_OUT_REG);
  End; {Case}

  If (PORT_STATE[PORT_X] = DISABLED) Then
    LOCAL_PORT_META_STATE[PORT_X]  := INACTIVE;
End;
```

12.11.4 RMAC transmits control signal—Cascade port

The cascade port is similar to the local port case defined in 12.11.3. The CASCADE_PORT_CONTROL_SIGNAL_OUT_REG register is updated by the RSM or the TSM by calling the procedure.

```
PROCEDURE CHANGE_CASCADE_PORT_CONTROL_SIGNAL
        (IN_CONTROL_SIGNAL : T_CONTROL_STATE);
Begin
  CASCADE_PORT_CONTROL_SIGNAL_OUT_REG := IN_CONTROL_SIGNAL;
End;
```

```
Forever Do
Begin
  Wait_Until(CLOCK_TICK);
  Case CASCADE_PORT_META_STATE Of
    INACTIVE :
      PMI_CONTROL_request(TXDISABLE);  {If port INACTIVE, send silence.}
    ACTIVE, TRAINING :
      PMI_CONTROL_request(CASCADE_PORT_CONTROL_SIGNAL_OUT_REG);
  End; {Case}

  If ((PORT_STATE = ENABLED) And (CASCADE_PORT_META_STATE = INACTIVE)) Then
  Begin {Port state has just been switched to ENABLE (probably by Network
           Management). Get TULM and RSM machines into right states.}
    CHANGE_CASCADE_PORT_CONTROL_SIGNAL(TRAINING_UP);
    TOTAL_TRAINING_PACKETS_DONE := 0; {TULM Counter}
    SUCCESSFUL_TRAINING_PACKETS_IN_RUN := 0;
    SET_TIMER(TRAINING_PULSE_SENDING);
    CASCADE_PORT_META_STATE := TRAINING;
    TULMSTATE := TULM2_WAIT_FOR_TRAINING_DOWN;
  End;

  If (PORT_STATE = DISABLED) Then
    CASCADE_PORT_META_STATE := INACTIVE;

  {Note that the distinction between CASCADE_PORT_META_STATE being ACTIVE and
   TRAINING on the cascade port is so that Network Management can observe the
   status of the variable.}
End;
```

12.11.5 RMAC receives control signal or MAC-frame data—Local port

Each local port PPM handles incoming control signals and MAC-frame data as described in this subclause. The main routines are a procedure, PROCESS_DATA, which handles receiving the MAC frame octets, and “forever” loops for each local port and for the cascade port.

At Port number PORT_X, the logic is:

```
Case EVENT_SORT Of
   {A simple autonomous process for timer.}
TIMER_EXPIRES :
  If TIMER = PACKET_PROMOTION Then PP_ON := True;
End; {Case}
```

The PPM for the selected port enters data-state processing when it detects the receiver control state value changing to DATA_ON_LINK. The module prepares to accept data octets transferred from the PMI by PMI_UNITDATA.indication primitives.

There are four possible sequences of PMI_UNITDATA.indication primitives for a single expected received frame:

a) The normal case, which applies to all successfully received frames and to many frames received with error conditions, is a sequence of two or more data octets received from PMI. The mac_frame parameter value of “begin” in the first PMI_UNITDATA.indication primitive in the sequence indicates the start of the MAC frame. The start_mac_frame_status and priority parameters apply to the frame as a whole. The PMI marks the end of frame if it detects esd or IPM, or on occurrence of the Mode Transition value of the Receiver Control State by setting the mac_frame parameter value to "end." The mac_frame value in intermediate primitives in the sequence is "more."

b) The received sequence consists of a single data octet only (when the PMI has detected ssd, one data octet, then esd, IPM, or Mode Transition). In this case, the PMI will generate a PMI_UNITDATA.indication primitive with the mac_frame parameter value of "begin," and the error_status parameter value of "error."

c) The sequence of received octets is excessively long, without end-of-frame being detected by PMI. In this case, FCP truncates the frame, upon expiration of the Long_Receive_Timeout, and disables receiving on the port. The FCP also sets the EMF_INDICATION register to INVALID_PACKET_MARKER.

d) No PMI_UNITDATA indication primitives occur at all, when the PMI does not decode any data octets before esd, IPM, or Mode Transition. The FCP uses the timer for void-frame generation to decide when to disable receiving and set EMF_INDICATION to INVALID_PACKET_MARKER.

In all cases except d), the PPM uses the start-of-frame indication associated with the first data octet. Unless case b) applies, the PPM changes state from PDS_EXPECT_SMF to PDS_RECEIVE_DATA.

In cases a) and b), the PPM exits from data-state to PDS_NONE on processing the octet marked as end of frame. The end-of-frame processing needs to be present for state PDS_EXPECT_SMF to deal with case b), as well as in state PDS_RECEIVE_DATA for the normal case a). In both cases, the PPM signals the end of frame to FCP, having set the CRC_OK indicator according to the CRC value that has been calculated as each data octet has been transferred from the PPM to the FCP.

In cases c) and d), the PPM does not receive an end-of-frame indication from PMI. To detect the end of frame in these cases, the PPM relies upon FCP. It continually monitors the state of EMF_INDICATION, and exits from data-state processing if it detects a change from the PACKET_OK value that FCP sets when preparing to receive a new frame.

The procedure PROCESS_DATA(IN_PORT : T_PORT) processes data octets. It is called within the "forever" loops. On exiting from data-state processing, the procedure updates the management counters for received frames unless no data has been received, [case c)]. The destinationAddress parameter is the value that the BCALL has extracted from the frame. The status parameter takes the value of "valid," "IPM," or dataError, according to the end_mac_frame_status and CRC values, etc.

The "forever" loops monitor the receiver_control_state for the value of data_on_link to indicate a change from the control state to data state. This "something changed" processing causes the Port_Data_State variable to take the values of "PDS_NONE," "PDS_EXPECT_SMF," or "PDS_RECEIVE_DATA." PDS_NONE indicates the control state.

12.11.5.1 Data state processing

If the link is in the data state, determine if the next data octet can be read. The Boolean function Get_PMI_UNITDATA_indication returns False if no new PMI_UNITDATA_indication has been signaled by PMI; otherwise, it returns True, and the parameters returned are those of the PMI primitive. This allows for:

a) An initial delay while preamble is being received (between the DATA_ON_LINK being signaled and the occurrence of the first data octet + SMF), and

b) The difference in rates between TxClk and the arrival data octets (5 octets per 12 clock ticks).

12.11.5.1.1 Procedure PROCESS_DATA

Only one set of the following variables is needed, for use by successive calls of the PROCESS_DATA procedure, since at any given time only the selected port is concerned with receiving data. Apart from CRC_OK, they are used only to record information to be passed to the management entity via calls of UpdateLmeRmacRxCounters.

CRC_OK : Boolean; Indicates that the CRC check has been successful.
octetCount : Integer; Indicates the number of octets in the received frame.
destinationAddress : T_MAC_ADDRESS; Indicates the destination address of the received frame.
priority : T_PRIORITY; Indicates the priority of the received frame.
status : T_PMI_ERROR_STATUS; Indicates the error status of the received frame.

```
Procedure PROCESS_DATA(IN_PORT : T_PORT);

L_Data_Octet : T_OCTET; { This is the data octet being received from the PMI. }
L_Priority : T_PRIORITY; { Indicates the priority of the incoming frame. }
L_SMF_Valid : Boolean; { Indicates that the start-of-frame sequence was valid. }
L_BME : T_PMI_BME; { Indicates whether this octet is the first octet (begin), whether it is an
        intermediate octet (more), or whether it is the last octet (end) in the frame being
        received from the PMI. }
L_EMF_Status : T_PMI_ERROR_STATUS; { Indicates the end-of-frame error status. }

Begin
 If (Port_Data_State <> PDS_NONE) Then
 Begin
  If (FLAG_FCP_TO_PPM_FRAME_ENDED = True) Then
  Begin           { Frame aborted by FCP, e.g., too long.}
   If (Port_Data_State = PDS_RECEIVE_DATA) Then
    UpdateLmeRmacRxCounters
     (IN_PORT, octetCount, destinationAddress, priority, status);
   Port_Data_State := PDS_NONE;
  End
  Else
  If (Get_PMI_UNITDATA_indication
    (L_Data_Octet, L_Priority, L_SMF_Valid, L_BME, L_EMF_Status)) Then
  Case Port_Data_State Of

   PDS_EXPECT_SMF:
   Begin
    If (L_Priority = HIGH_PRIORITY) Then
      priority := high;
    Else If ((PP_ON = TRUE) And (ENTITY_AT_FAR_END_OF_LINK[IN_PORT] =
       END_NODE))
      Then
      priority := promoted_to_hi;
    Else
     priority := normal;

    octetCount := 1;
    destinationAddress := NullAddress;
    status := valid;
    Initialize_CRC();
```

```
        If (L_SMF_Valid = True) Then
        Begin
          SIGNAL_TO_FCP (SMF_RECEIVED_THRU_PORT);
          Push_to_RMB_Input (L_Data_Octet);      **{ and CRC calculation }**
        End
        Else Begin
          priority := normal;
          SIGNAL_TO_FCP (BAD_SMF_RECEIVED_THRU_PORT);
          Push_to_RMB_Input (ZERO);
        End;
        **{ Now see if this is a single-octet frame (unlikely but possible). }**
        If (L_BME = END) Then
        **{ L_BME is the mac_frame parameter, value BEGIN / MORE / END. }**
        Begin
          CRC_OK := False;
          status := dataError;
          SIGNAL_TO_FCP (FRAME_ENDED_ABNORMAL_END);
          UpdateLmeRmacRxCounters
            (IN_PORT, 1, NullAddress, priority, status);
          Port_Data_State := PDS_NONE;
        End
        Else
          Port_Data_State := PDS_RECEIVE_DATA;
      End;

      PDS_RECEIVE_DATA:
      Begin
        Push_to_RMB_Input (L_Data_Octet);      **{ and CRC calculation }**
        octetCount := octetCount + 1;
        If (L_BME = END) Then
            **{L_BME is the mac_frame parameter, value BEGIN / MORE / END. }**
        Begin
          CRC_OK := Is_CRC_Correct();
          If ((L_EMF_Status = IPM) Or (L_EMF_Status = EMF_ERROR)
             Or (CRC_OK = False)) Then
          Begin
            If (L_EMF_Status = VALID) Then
               status := dataError
            Else
               status := L_EMF_Status;
            SIGNAL_TO_FCP (FRAME_ENDED_ABNORMAL_END);
          End
          Else
            SIGNAL_TO_FCP (EMF_RECEIVED_THRU_PORT);
          UpdateLmeRmacRxCounters
            (IN_PORT, octetCount, destinationAddress, priority, status);
          Port_Data_State := PDS_NONE;
        End
      End;
    End **{Case, and also ends If (Get_PMI_UNITDATA_indication).}**
  End
End;**{Procedure}**
```

12.11.5.1.2 Local port forever loop

```
Forever Do
Begin
  Wait_Until(PMI_CONTROL_indication(GRANT_STATE,RECEIVER_CONTROL_STATE));
  {This occurs every clock tick, i.e., it is effectively invoked by the PMD every clock tick.}
  If ((GRANT_STATE <> OLD_GRANT_STATE) Or
     (RECEIVER_CONTROL_STATE <> OLD_RECEIVER_CONTROL_STATE)) Then
  Begin {Something Changed}
    If (((RECEIVER_CONTROL_STATE = REQ_N) Or
        ((RECEIVER_CONTROL_STATE = REQ_H) And (PRIORITY_ENABLE = False)))
        {If RCS is REQ_N or is REQ_H with REQ_H's considered as REQ_N because
         high priority not enabled.}
        And (PP_TIMER_STARTED = False)) Then {Avoid continually starting timer.}
    Begin
      SET_TIMER(PACKET_PROMOTION);
      PP_TIMER_STARTED := True;
    End;

    If (RECEIVER_CONTROL_STATE = LINK_WARNING) Then
      LOCAL_PORT_META_STATE[PORT_X] := INACTIVE;

    {If LINK_WARNING, then set LOCAL_PORT_META_STATE to INACTIVE.
     This causes silence to go to entity at far end, which then retrains.}

    {First some preliminary protocol handling.}

    If (GRANT_STATE = 0) Then
    Begin {Update RMAC_REQ_REG, which is used by RSM.}
      Case RECEIVER_CONTROL_STATE Of
        REQ_H : If ((PRIORITY_ENABLE = True) Or (PP_ON = True) Or
               (ENTITY_AT_FAR_END_OF_LINK[PORT_X] = LOWER_REPEATER)) Then
                 RMAC_REQ_REG[PORT_X] := HIGH_PRIORITY
                Else RMAC_REQ_REG[PORT_X] := NORMAL_PRIORITY;

        REQ_N : If ((PP_ON = True) And (ENTITY_AT_FAR_END_OF_LINK[PORT_X] =
                END_NODE) )  Then
                 RMAC_REQ_REG[PORT_X] := HIGH_PRIORITY
                Else RMAC_REQ_REG[PORT_X] := NORMAL_PRIORITY;

        IDLE_U, DATA_ON_LINK, MODE_TRANSITION, TRAINING_UP :
                RMAC_REQ_REG[PORT_X] := NONE;
      End; {Case}

      Case LOCAL_PORT_META_STATE Of
        ACTIVE :
        If (RECEIVER_CONTROL_STATE = TRAINING_UP) Then
        Begin
          LOCAL_PORT_META_STATE[PORT_X]  := TRAINING;
          LOCAL_PORT_CONTROL_SIGNAL_OUT_REG := TRAINING_DOWN;
          LOCAL_PORT_PACKETS_OK_SEQUENCE[PORT_X] := 0;
        End;
        TRAINING :
        If (RECEIVER_CONTROL_STATE = IDLE_U) Then
          Begin {Have been in Training; receive IDLE_U which indicates
                 lower entity regards training as finished successfully.}
            If ((LOCAL_PORT_PACKETS_OK_SEQUENCE[PORT_X]  >= 24) And
                (VALID_CONFIG = True)) Then
              {i.e., on this packet training initiator has sent configuration
               which does not cause it to fail.}
             LOCAL_PORT_META_STATE[PORT_X] := ACTIVE
              {This will cause IDLE_D or INCOMING to be sent to lower entity.}
            Else LOCAL_PORT_META_STATE[PORT_X]  := INACTIVE;
              {Wait for lower entity to initiate training with TRAINING_UP.}
          End;

        INACTIVE :
        If ((RECEIVER_CONTROL_STATE = TRAINING_UP) And
           (PORT_STATE <> DISABLED)) Then
              {PORT_STATE typically set by Network Management.
               Stay as INACTIVE if port is disabled.}
        Begin {TRAINING_UP, i.e., request to train received.}
          LOCAL_PORT_META_STATE[PORT_X] := TRAINING;
          LOCAL_PORT_CONTROL_SIGNAL_OUT_REG := TRAINING_DOWN;
          LOCAL_PORT_PACKETS_OK_SEQUENCE[PORT_X] := 0;
        End;
      End {Case}
    End;  {preliminaries}

    If (LOCAL_PORT_META_STATE[PORT_X]  <> INACTIVE) Then
    Begin  {If INACTIVE, don't want control signals going to RSM, FCP.}
      If (GRANT_STATE = 1) Then
      Begin
        If (RECEIVER_CONTROL_STATE <> LINK_WARNING) Then
        Begin {At a local port, GRANT_STATE = 1 implies RETURN received. }
          SIGNAL_PORT_CS_TO_RSM(LOCAL_PORT_CONTROL_SIGNAL_DECODED,
                            RETURN, PORT_X);
          If (SELECTED_PORT = PORT_X) Then
            SIGNAL_CS_TO_FCP(LOCAL_PORT_CONTROL_SIGNAL_DECODED, RETURN);
        End
        Else       {LINK_WARNING overrides GRANT.}
        Begin
          SIGNAL_PORT_CS_TO_RSM(LOCAL_PORT_CONTROL_SIGNAL_DECODED,
                            LINK_WARNING, PORT_X);
          If (SELECTED_PORT = PORT_X) Then
            SIGNAL_CS_TO_FCP(LOCAL_PORT_CONTROL_SIGNAL_DECODED,
                    LINK_WARNING);
        End
      End
      Else
      Begin {GRANT_STATE = 0}
        If (RECEIVER_CONTROL_STATE = TRAINING_UP) Then
        Begin
          {The RSM treats TRAINING_UP exactly as if it was IDLE_U in the general handling of
           frames. RSM aware of port in training from LOCAL_PORT_META_STATE. Similar to
           the previous switch, it simplifies the RSM by avoiding dealing with it separately.}
```

```
        SIGNAL_PORT_CS_TO_RSM(LOCAL_PORT_CONTROL_SIGNAL_DECODED,
                      IDLE_U, PORT_X);
        If (SELECTED_PORT = PORT_X) Then
          SIGNAL_CS_TO_FCP(LOCAL_PORT_CONTROL_SIGNAL_DECODED, IDLE_U);
      End
      Else
      If ((((RECEIVER_CONTROL_STATE = REQ_H) And (PRIORITY_ENABLE = False))
            Or (RECEIVER_CONTROL_STATE = REQ_N))
               {If the RCS is REQ_N, or REQ_H being regarded at all times
                as REQ_N because all REQ_H's are converted to REQ_N.}
          And (ENTITY_AT_FAR_END_OF_LINK[PORT_X] = END_NODE)
          And (PP_ON = True) {Packet Promotion is taking place}) Then
      Begin
        SIGNAL_PORT_CS_TO_RSM(LOCAL_PORT_CONTROL_SIGNAL_DECODED,
                      REQ_H, PORT_X);
        If (SELECTED_PORT = PORT_X) Then
          SIGNAL_CS_TO_FCP(LOCAL_PORT_CONTROL_SIGNAL_DECODED, REQ_H);
      End
      Else
      If ((RECEIVER_CONTROL_STATE = REQ_H) And (PRIORITY_ENABLE = False) And
     (ENTITY_AT_FAR_END_OF_LINK[PORT_X] = END_NODE) )
      Then
        {REQ_N or REQ_H regarded as REQ_N because PRIORITY_ENABLE = false.}
      Begin
        SIGNAL_PORT_CS_TO_RSM(LOCAL_PORT_CONTROL_SIGNAL_DECODED,
          REQ_N, [PORT_X]);
        If (SELECTED_PORT = PORT_X) Then
          SIGNAL_CS_TO_FCP(LOCAL_PORT_CONTROL_SIGNAL_DECODED, REQ_N);
      End Else
      Begin
        SIGNAL_PORT_CS_TO_RSM(LOCAL_PORT_CONTROL_SIGNAL_DECODED,
                      RECEIVER_CONTROL_STATE, PORT_X);
        If (SELECTED_PORT = PORT_X) Then
          SIGNAL_CS_TO_FCP
            (LOCAL_PORT_CONTROL_SIGNAL_DECODED, RECEIVER_CONTROL_STATE);
        {LINK_WARNING also signaled to RSM with this mechanism.}
      End
    End {GRANT_STATE = 0}
  End; {If <> INACTIVE}

  If ((RECEIVER_CONTROL_STATE = DATA_ON_LINK) And
      (PORT_X = SELECTED_PORT)) Then
    Port_Data_State := PDS_EXPECT_SMF;

 End;{Something changed.}

 OLD_GRANT_STATE := GRANT_STATE;
 OLD_RECEIVER_CONTROL_STATE := RECEIVER_CONTROL_STATE;

 PROCESS_DATA(PORT_X); {Procedure previously defined in 12.11.5.1.1.}
End; {Forever Do}
```

12.11.5.1.3 Cascade port forever loop

```
Forever Do
Begin
  Wait_Until(PMI_CONTROL_indication(GRANT_STATE,RECEIVER_CONTROL_STATE));
  {This occurs every clock tick, i.e., it is effectively invoked by the PMD every clock tick.}
  If ((GRANT_STATE <> OLD_GRANT_STATE) Or
     (RECEIVER_CONTROL_STATE <> OLD_RECEIVER_CONTROL_STATE)) Then
  Begin {Something Changed.}
    If (CASCADE_PORT_META_STATE <> INACTIVE) Then
    Begin
      If (GRANT_STATE = 1) Then
      Begin
        If (RECEIVER_CONTROL_STATE <> LINK_WARNING) Then
        Begin
          SIGNAL_CS_TO_RSM_TULM(CASCADE_PORT_CONTROL_SIGNAL_DECODED,
               GRANT);
          If (SELECTED_PORT = CASCADE_PORT) Then
            SIGNAL_CS_TO_FCP(CASCADE_PORT_CONTROL_SIGNAL_DECODED, GRANT);
        End
        Else {LINK_WARNING overrides GRANT.}
        Begin
          SIGNAL_CS_TO_RSM_TULM
            (CASCADE_PORT_CONTROL_SIGNAL_DECODED, LINK_WARNING);
          If (SELECTED_PORT = CASCADE_PORT) Then
            SIGNAL_CS_TO_FCP(CASCADE_PORT_CONTROL_SIGNAL_DECODED,
                LINK_WARNING);
        End
      End
      Else {GRANT_STATE = 0}
      Begin
        SIGNAL_CS_TO_RSM_TULM
          (CASCADE_PORT_CONTROL_SIGNAL_DECODED, RECEIVER_CONTROL_STATE);
        If (SELECTED_PORT = CASCADE_PORT) Then
          SIGNAL_CS_TO_FCP
            (CASCADE_PORT_CONTROL_SIGNAL_DECODED, RECEIVER_CONTROL_STATE);
        Case RECEIVER_CONTROL_STATE Of
          ENABLE_HIGH_ONLY : RMAC_ENABLE_HIGH_ONLY_RECEIVED_REG := True;
          IDLE_D, INCOMING, TRAINING_DOWN, LINK_WARNING:
             RMAC_ENABLE_HIGH_ONLY_RECEIVED_REG := False;
        End {Case}
      End
    End;

  If ((RECEIVER_CONTROL_STATE = DATA_ON_LINK) And
        (SELECTED_PORT = CASCADE_PORT)) Then
      Port_Data_State := PDS_EXPECT_SMF;

  End; { End of "something changed" processing. }

  OLD_GRANT_STATE := GRANT_STATE;
  OLD_RECEIVER_CONTROL_STATE := RECEIVER_CONTROL_STATE;

  PROCESS_DATA(CASCADE_PORT);
```

End; **{Forever loop}**

12.12 The Train Uplink Machine (TULM)

The TULM is responsible for carrying out the training of the uplink at the cascade port. While the cascade port meta-state is TRAINING, the TULM controls the cascade-port packet sequencing and the input to the RMAC buffer in a manner similar to the RSM when the cascade port meta-state is ACTIVE. The TULM is not involved with any local port activity. Its relationship to other RMAC state machines is shown in Figure 12-50.

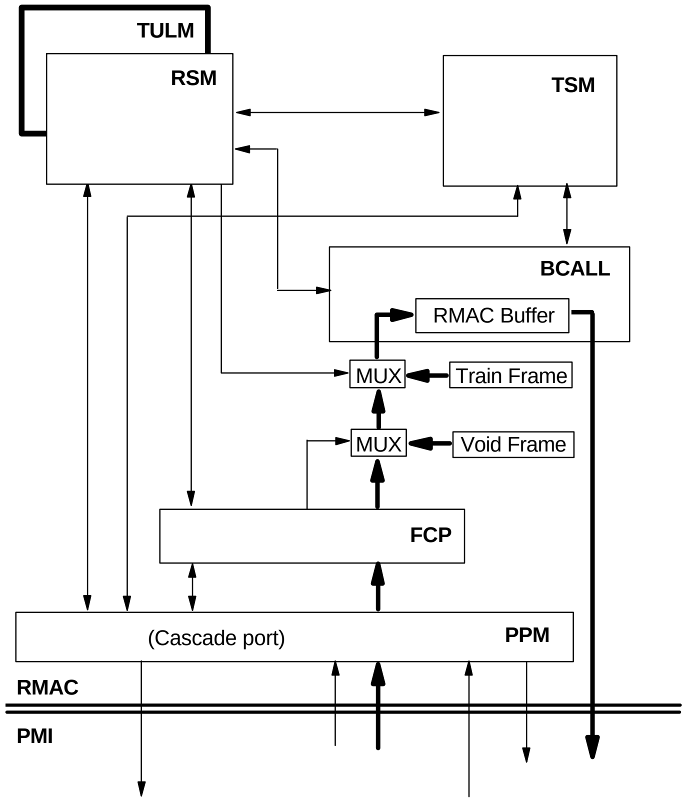


Figure 12-50—Relationship of TULM to the other RMAC state machines

The training uplink machine is shown in Figure 12-51. The states shown bolded (TULM1 through TULM3) are all wait states that are not actively involved with the training process. Local port data traffic is allowed to continue while the TULM is in any of the three wait states.

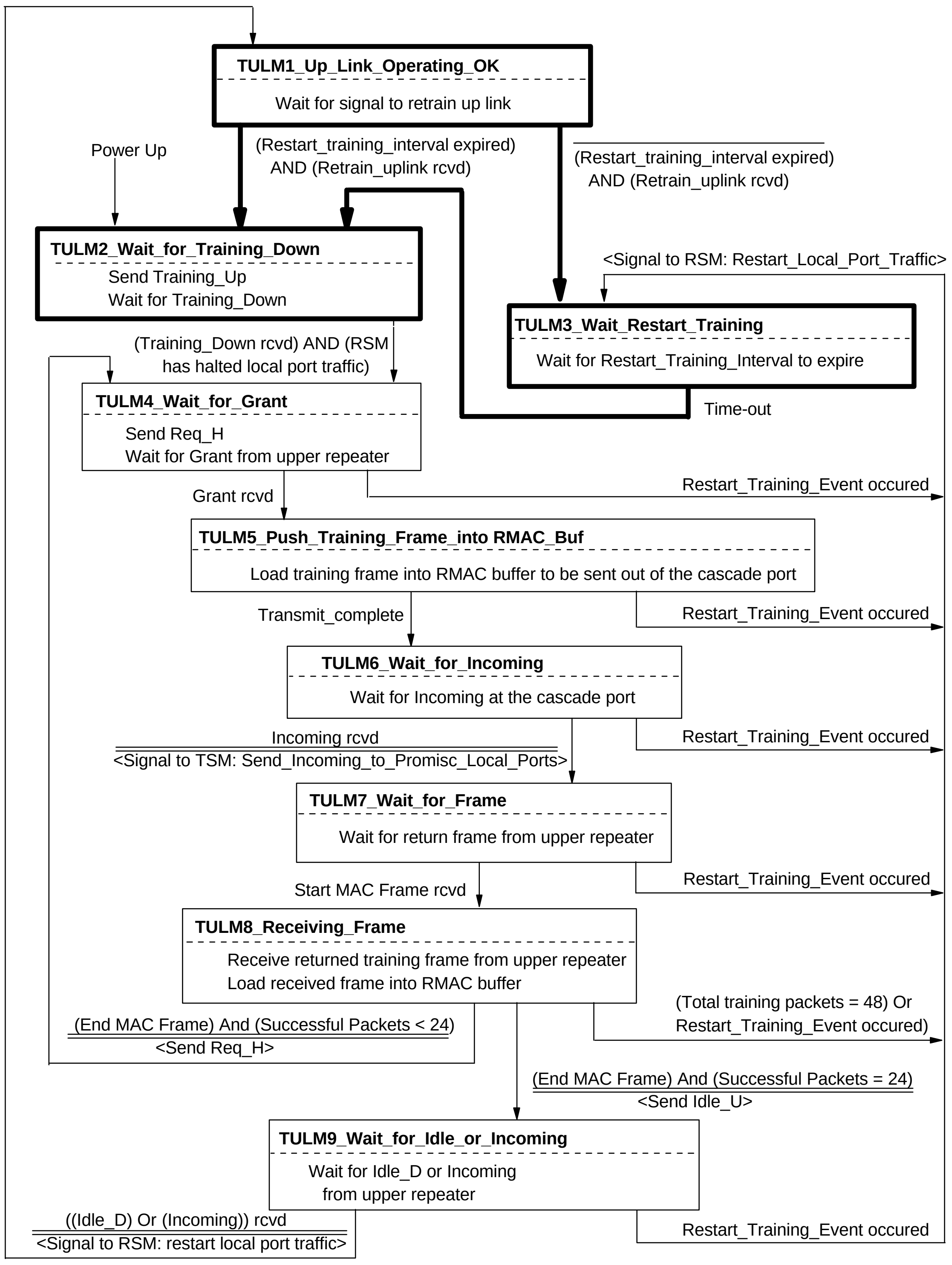

TULM1_Up_Link_Operating_OK
Wait for signal to retrain up link
Power Up
(Restart_training_interval expired) AND (Retrain_uplink rcvd)
(Restart_training_interval expired) AND (Retrain_uplink rcvd)
TULM2_Wait_for_Training_Down
Send Training_Up
Wait for Training_Down
<Signal to RSM: Restart_Local_Port_Traffic>
TULM3_Wait_Restart_Training
Wait for Restart_Training_Interval to expire
(Training_Down rcvd) AND (RSM has halted local port traffic)
Time-out
TULM4_Wait_for_Grant
Send Req_H
Wait for Grant from upper repeater
Grant rcvd
Restart_Training_Event occured
TULM5_Push_Training_Frame_into RMAC_Buf
Load training frame into RMAC buffer to be sent out of the cascade port
Transmit_complete
Restart_Training_Event occured
TULM6_Wait_for_Incoming
Wait for Incoming at the cascade port
Incoming rcvd
<Signal to TSM: Send_Incoming_to_Promisc_Local_Ports>
Restart_Training_Event occured
TULM7_Wait_for_Frame
Wait for return frame from upper repeater
Start MAC Frame rcvd
Restart_Training_Event occured
TULM8_Receiving_Frame
Receive returned training frame from upper repeater
Load received frame into RMAC buffer
(Total training packets = 48) Or Restart_Training_Event occured)
(End MAC Frame) And (Successful Packets < 24)
<Send Req_H>
(End MAC Frame) And (Successful Packets = 24)
<Send Idle_U>
TULM9_Wait_for_Idle_or_Incoming
Wait for Idle_D or Incoming from upper repeater
((Idle_D) Or (Incoming)) rcvd
<Signal to RSM: restart local port traffic>
Restart_Training_Event occured

Figure 12-51—TULM state diagram

The TULM will be quiescent (in TULM1_UP_LINK_OPERATING_OK) while the RSM is handling cascade-port data traffic. When the cascade port needs to be trained, the TULM is actively operating, and the RSM is waiting in its quiescent state (RX1_IDLE). Signals between the TULM and the RSM control which machine is active (see Figure 12-52).

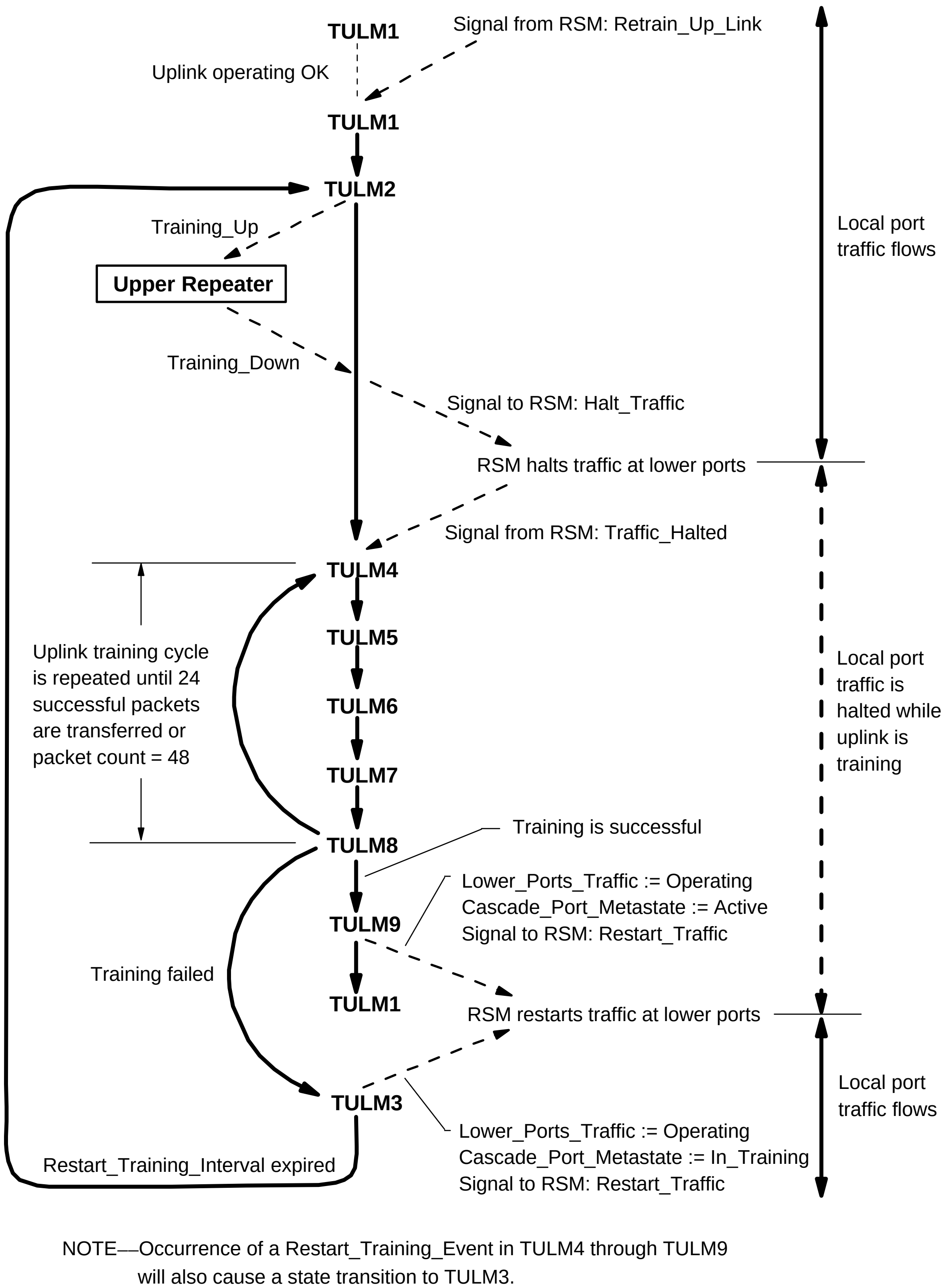

TULM1
Signal from RSM: Retrain_Up_Link
Uplink operating OK
TULM1
TULM2
Training_Up
Upper Repeater
Training_Down
Signal to RSM: Halt_Traffic
RSM halts traffic at lower ports
Signal from RSM: Traffic_Halted
Local port traffic flows
TULM4
TULM5
TULM6
TULM7
TULM8
Uplink training cycle is repeated until 24 successful packets are transferred or packet count = 48
Local port traffic is halted while uplink is training
Training is successful
Lower_Ports_Traffic := Operating
Cascade_Port_Metastate := Active
Signal to RSM: Restart_Traffic
TULM9
Training failed
TULM1
RSM restarts traffic at lower ports
TULM3
Local port traffic flows
Lower_Ports_Traffic := Operating
Cascade_Port_Metastate := In_Training
Signal to RSM: Restart_Traffic
Restart_Training_Interval expired
NOTE—Occurrence of a Restart_Training_Event in TULM4 through TULM9 will also cause a state transition to TULM3.

Figure 12-52—TULM/RSM interaction sequence

RETRAIN_UPLINK is the signal sent by the RSM to the TULM when it is required to retrain the uplink at the cascade port. The RSM does not immediately halt the local traffic but waits until the variable LOWER_PORTS_TRAFFIC is set to HALTED.

a) On receiving RETRAIN_UPLINK, the TULM sends TRAINING_UP on the uplink.

b) On receiving TRAINING_DOWN, the TULM sets LOWER_PORTS_TRAFFIC to HALTED, which tells the RSM to halt the local traffic.
c) The RSM halts the local traffic in an orderly way, and then signals to the TULM with the signal TRAFFIC_HALTED.
d) On receiving TRAFFIC_HALTED, the TULM sends REQUEST_HIGH to initiate the sending of the sequence of training packets.

TULM local variables:

FLAG_INTERVAL_TO_RESTART_TRAINING_EXPIRED : Boolean; Flag is used to ensure that a new training attempt does not start until at least a period of TR_INTERVAL_TO_RESTART_TRAINING after a successful completion of training.

FLAG_TRAINING_DOWN_RECEIVED : Boolean; Flag is set when TRAINING_DOWN is received.

SUCCESSFUL_TRAINING_PACKETS_IN_RUN : Integer; Counts the number of successive training packet send-return pairs where both send and receive are error-free.

TOTAL_TRAINING_PACKETS_DONE : Integer; Counts the total number of training packets sent in the current training sequence.

TULM local procedure

Procedure RESTART_TRAINING()

This procedure is called by a number of the states in the TULM when retraining of the uplink is indicated.

```
Begin
  SET_TIMER(TR_INTERVAL_TO_RESTART_TRAINING);
  SET_TIMER(TRAINING_PULSE_TXDISABLE);
  CHANGE_CASCADE_PORT_CONTROL_SIGNAL(TXDISABLE);
  LOWER_PORTS_TRAFFIC := OPERATING;
  SIGNAL_TO_RSM(S_TULM_TO_RSM_RESTART_TRAFFIC);
  TULMSTATE := TULM3_WAIT_RESTART_TRAINING;
End;
```

12.12.1 TULM1_UP_LINK_OPERATING_OK

This is the TULM quiescent state. Its relationship to the TULM state machine is shown in Figure 12-53. The Uplink is assumed to be operating okay. The TULM waits in TULM1 while the RSM is operating. If the TULM receives the signal RETRAIN_LINK, it starts sending TRAINING_UP out of the cascade port, and transitions to TULM2.

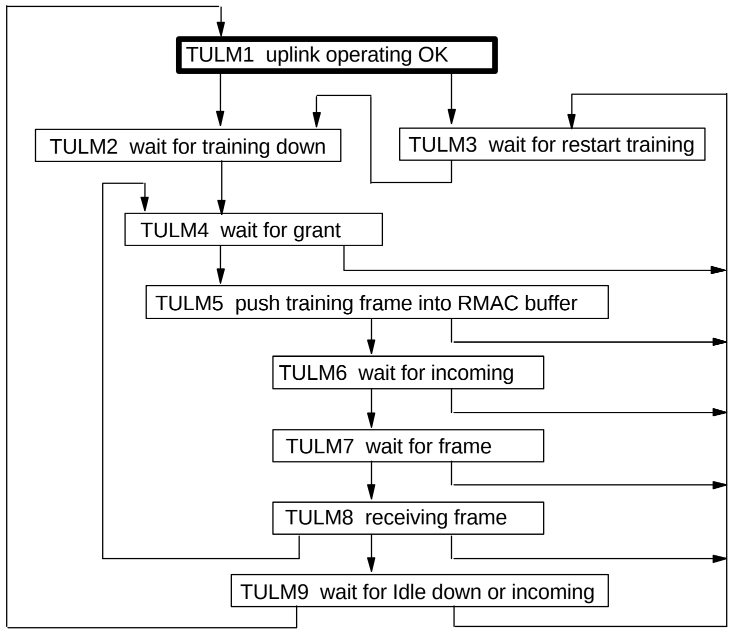

TULM1 uplink operating OK
TULM2 wait for training down
TULM3 wait for restart training
TULM4 wait for grant
TULM5 push training frame into RMAC buffer
TULM6 wait for incoming
TULM7 wait for frame
TULM8 receiving frame
TULM9 wait for Idle down or incoming

Figure 12-53—Relationship of TULM1 in the state machine

```
Case EVENT_SORT Of
  S_TO_TULM_RETRAIN_UPLINK :
  Begin
    CHANGE_CASCADE_PORT_CONTROL_SIGNAL(TRAINING_UP);
    TOTAL_TRAINING_PACKETS_DONE := 0;
    SUCCESSFUL_TRAINING_PACKETS_IN_RUN := 0;
        {At this point TRAINING_UP is sent out of the cascade port, but lower port traffic
         continues. CASCADE_PORT_META_STATE has already been set to TRAINING within
         INVOKE_CASCADE_PORT_TRAINING, so traffic at lower ports continues but with
         cascade port not part of the configuration.}
    SET_TIMER(TRAINING_PULSE_SENDING);
    If (FLAG_INTERVAL_TO_RESTART_TRAINING_EXPIRED = True) Then
      TULMSTATE := TULM2_WAIT_FOR_TRAINING_DOWN
    Else
      TULMSTATE := TULM3_WAIT_RESTART_TRAINING;
  End;

  TIMER_EXPIRES :
  If (TIMER = TR_INTERVAL_TO_RESTART_TRAINING) Then
    FLAG_INTERVAL_TO_RESTART_TRAINING_EXPIRED := True;
End; {Case}
```

12.12.2 TULM2_WAIT_FOR_TRAINING_DOWN

Training has been requested and the TULM is waiting to receive TRAINING_DOWN. While in TULM2, TRAINING_UP is pulsed (alternated with TXDISABLE), by means of the TRAINING_PULSE_ SENDING and TRAINING_PULSE_TXDISABLE timers.

If TRAINING_DOWN is received, LOWER_PORTS_TRAFFIC is set to HALTED. When the signal TRAFFIC_HALTED comes back from the RSM, the TULM sends up REQ_H and transitions to TULM4 (see Figure 12-54).

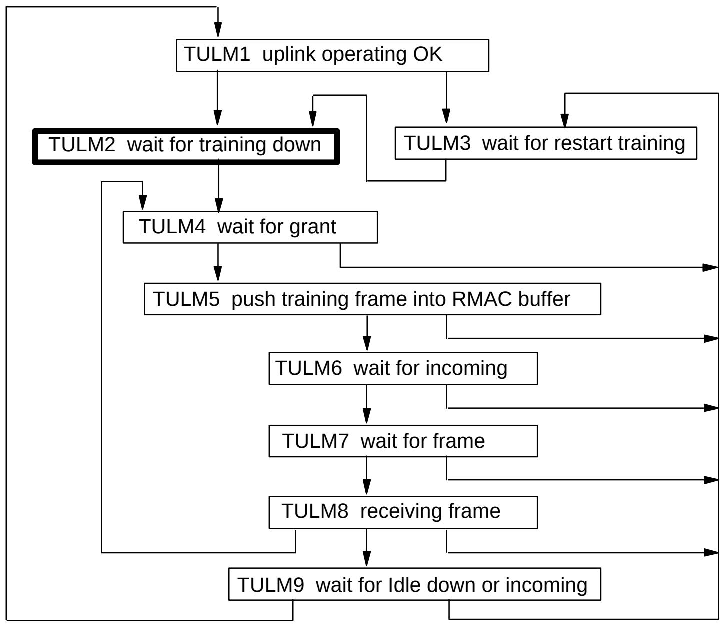

TULM1 uplink operating OK
TULM2 wait for training down
TULM3 wait for restart training
TULM4 wait for grant
TULM5 push training frame into RMAC buffer
TULM6 wait for incoming
TULM7 wait for frame
TULM8 receiving frame
TULM9 wait for Idle down or incoming

Figure 12-54—Relationship of TULM2 in the state machine

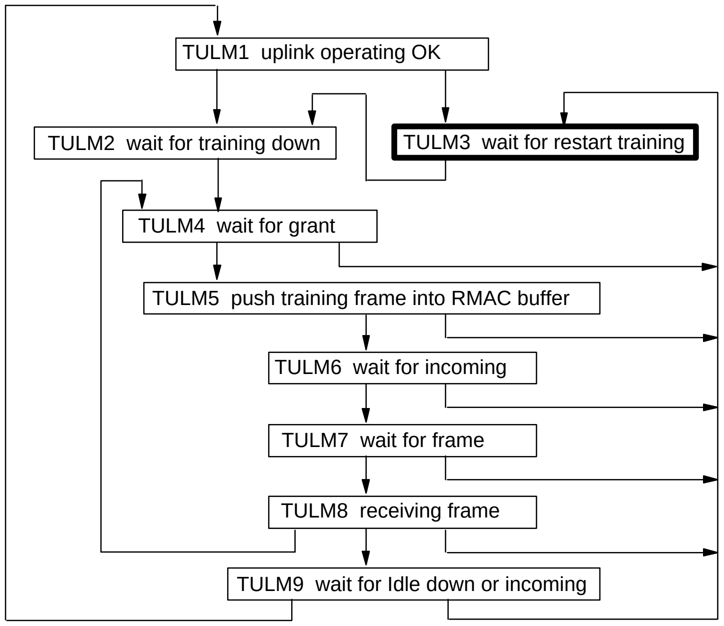
```
Case EVENT_SORT Of
  CASCADE_PORT_CONTROL_SIGNAL_DECODED :
  If (CASCADE_PORT_CONTROL_SIGNAL_IN = TRAINING_DOWN) Then
  Begin {TRAINING_D has been received; send up a high-priority Request.}
    FLAG_TRAINING_DOWN_RECEIVED := True; {Used in TULM4.}
    {Have received TRAINING_D from upper hub. Now local traffic needs to be halted.}
    SIGNAL_TO_RSM(S_TULM_TO_RSM_HALT_TRAFFIC);
    LOWER_PORTS_TRAFFIC := HALTED;
    {The change to HALTED acts as a signal to the RSM to halt local traffic.}
  End;

S_RSM_TO_TULM_TRAFFIC_HALTED : {Informed by RSM that traffic halted.}
  Begin
    CHANGE_CASCADE_PORT_CONTROL_SIGNAL(REQ_H);
    SET_FAULT_TIMER(TR_REQ_UP_TO_FRAME_RECEIVED);
    CANCEL_TIMER(TRAINING_PULSE_SENDING);
    CANCEL_TIMER(TRAINING_PULSE_TXDISABLE);
    TULMSTATE := TULM4_WAIT_FOR_GRANT;
  End;

  TIMER_EXPIRES :
  Case TIMER Of
    {These next timers implement the pulsing of sending TRAINING_UP.}
    TRAINING_PULSE_SENDING :
    Begin
      CHANGE_CASCADE_PORT_CONTROL_SIGNAL(TXDISABLE);
      SET_TIMER(TRAINING_PULSE_TXDISABLE);
    End;
    TRAINING_PULSE_TXDISABLE :
    Begin
      CHANGE_CASCADE_PORT_CONTROL_SIGNAL(TRAINING_UP);
      SET_TIMER(TRAINING_PULSE_SENDING);
    End;
  End; {Case}
End; {Case}
```

12.12.3 TULM3_WAIT_RESTART_TRAINING

Uplink training has failed. The TULM waits in TULM3 to restart training. While waiting to restart training, the RSM operates with local port traffic. The TULM pulses up TRAINING_UP, but does not attempt to initiate training. After remaining in TULM3 for TR_INTERVAL_TO_RESTART_TRAINING, the TULM transitions to TULM2 (see Figure 12-55).



TULM1 uplink operating OK
TULM2 wait for training down
TULM3 wait for restart training
TULM4 wait for grant
TULM5 push training frame into RMAC buffer
TULM6 wait for incoming
TULM7 wait for frame
TULM8 receiving frame
TULM9 wait for Idle down or incoming

Figure 12-55—Relationship of TULM3 in the state machine

```
{Local-port traffic will be handled by the RSM while the TULM is in this state.}
Case EVENT_SORT Of
  TIMER_EXPIRES :
  Case TIMER Of
    TR_INTERVAL_TO_RESTART_TRAINING :
    Begin
      TOTAL_TRAINING_PACKETS_DONE := 0;
      SUCCESSFUL_TRAINING_PACKETS_IN_RUN := 0;
      TULMSTATE := TULM2_WAIT_FOR_TRAINING_DOWN;
    End;
    {Next 2 timers implement on and off.}

    TRAINING_PULSE_SENDING :
    Begin
      CHANGE_CASCADE_PORT_CONTROL_SIGNAL(TXDISABLE);
      SET_TIMER(TRAINING_PULSE_TXDISABLE);
    End;

    TRAINING_PULSE_TXDISABLE :
    Begin
```

```
      CHANGE_CASCADE_PORT_CONTROL_SIGNAL(TRAINING_UP);
      SET_TIMER(TRAINING_PULSE_SENDING);
    End;
  End; {Case}
End; {Case}
```

12.12.4 TULM4_WAIT_FOR_GRANT

In TULM4, the TULM is waiting for GRANT from the upper repeater. If the GRANT arrives, the TULM transitions to TULM5, where it will push a training frame into the RMAC buffer. On transitioning into TULM4, a single timer TR_REQ_UP_TO_FRAME was set. This timer is used to detect if TULM stays too long in any of the states TULM4, TULM6, or TULM7. If this timer expires while in TULM4, there is a transition to TULM3, where the TULM waits to restart training. The local port traffic is restarted when transitioning to TULM3. A LINK_WARNING also results in the transition to the wait state, TULM3 (see Figure 12-56).

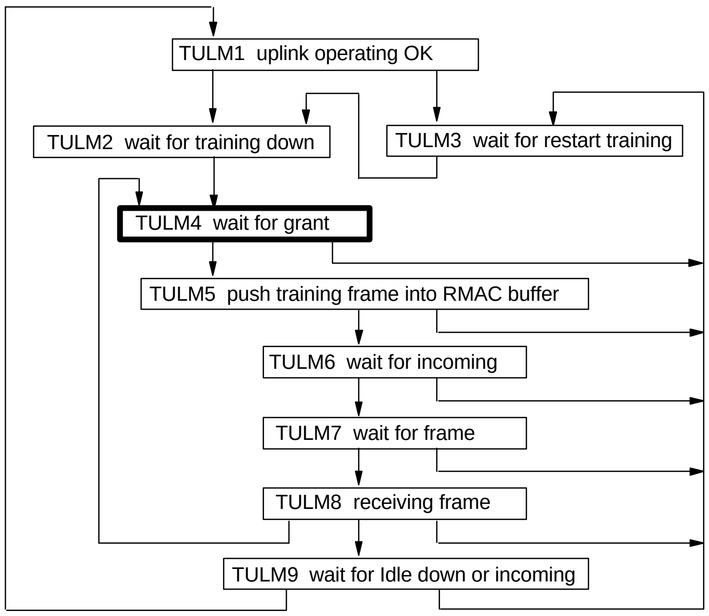

TULM1 uplink operating OK
TULM2 wait for training down
TULM3 wait for restart training
TULM4 wait for grant
TULM5 push training frame into RMAC buffer
TULM6 wait for incoming
TULM7 wait for frame
TULM8 receiving frame
TULM9 wait for Idle down or incoming

Figure 12-56—Relationship of TULM4 in the state machine

```
Case EVENT_SORT Of
  CASCADE_PORT_CONTROL_SIGNAL_DECODED :
  Case CASCADE_PORT_CONTROL_SIGNAL_IN Of
    TRAINING_DOWN :
      FLAG_TRAINING_DOWN_RECEIVED := True;

    GRANT :  {Expected Grant arrives. Start pushing Training frame into RMAC buffer.}
    If (FLAG_TRAINING_DOWN_RECEIVED = True) Then
    Begin {Expect to have received Training Down before Grant arrives.}
      RMAC_BUF_DATA_INPUT := TRAINING_FRAME_OCTETS;
      TOTAL_TRAINING_PACKETS_DONE := TOTAL_TRAINING_PACKETS_DONE + 1;
      TULMSTATE := TULM5_PUSH_TRAINING_FRAME_INTO_RMAC_BUF;
    End;

    LINK_WARNING : RESTART_TRAINING();
  End; {Case}

  TIMER_EXPIRES : {Time-out on Grant arriving; restart training after silence.}
  If (TIMER = TR_REQ_UP_TO_FRAME_RECEIVED) Then
    RESTART_TRAINING();
End; {Case}
```

12.12.5 TULM5_PUSH_TRAINING_FRAME_INTO_RMAC_BUF

In TULM5, the TULM is pushing training frame octets into the RMAC buffer, in much the same way that the RSM pushes ordinary frames into the RMAC buffer. The training frame will be sent out of the cascade port. When the training frame has been completely pushed into the RMAC buffer, the TULM transitions into TULM6 (see Figure 12-57).

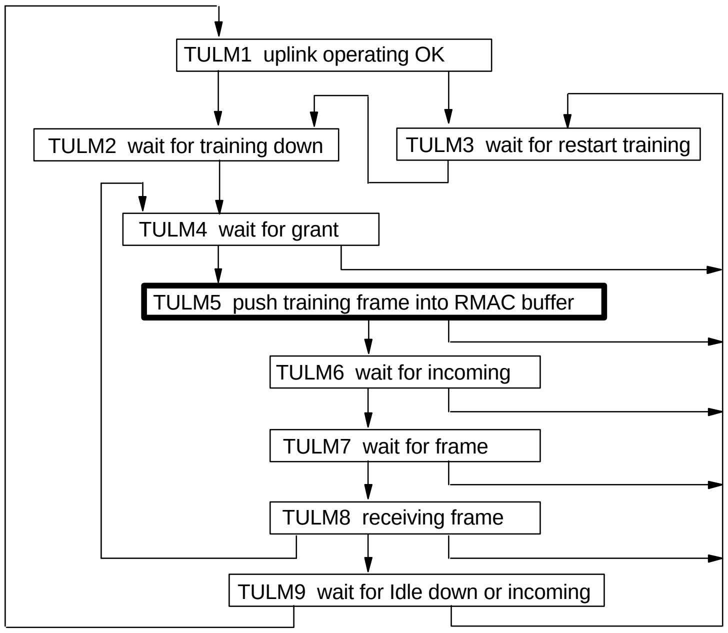

TULM1 uplink operating OK
TULM2 wait for training down
TULM3 wait for restart training
TULM4 wait for grant
TULM5 push training frame into RMAC buffer
TULM6 wait for incoming
TULM7 wait for frame
TULM8 receiving frame
TULM9 wait for Idle down or incoming

Figure 12-57—Relationship of TULM5 in the state machine

TRAINING_FRAME_INTO_RMAC_BUFFER_COMPLETE is an event indicated by BCALL when generation of the training frame has been completed. It is also possible that the timer TR_REQ_UP_TO_FRAME_RECEIVED could expire when in TULM5, which would cause the RMAC to restart training of the uplink.

```
Case EVENT_SORT Of
  TRAINING_FRAME_INTO_RMAC_BUFFER_COMPLETE :
  Begin
    RMAC_BUF_DATA_INPUT := STOP_INPUT;
    CHANGE_CASCADE_PORT_CONTROL_SIGNAL(TRAINING_UP);
    TULMSTATE := TULM6_WAIT_FOR_INCOMING;
  End;
TIMER_EXPIRES :
  If (TIMER = TR_REQ_UP_TO_FRAME_RECEIVED) Then
  Begin
```

```
    RMAC_BUF_DATA_INPUT := STOP_INPUT;
    RESTART_TRAINING();
  End;
End; {Case}
```

12.12.6 TULM6_WAIT_FOR_INCOMING

The training frame has been sent up. In TULM6, the TULM is waiting for INCOMING to be sent down from the repeater above, prior to the arrival of the return training frame. When INCOMING arrives, the TULM signals to the FCP to expect the frame (EXPECT_FRAME). The FCP will then pull Receive_ Enable. If the timer REQ_UP_TO_FRAME_RECEIVED expires, or a LINK_WARNING state is received, then the TULM transitions to the wait state, TULM3, and the lower port traffic is allowed to operate again (see Figure 12-58).

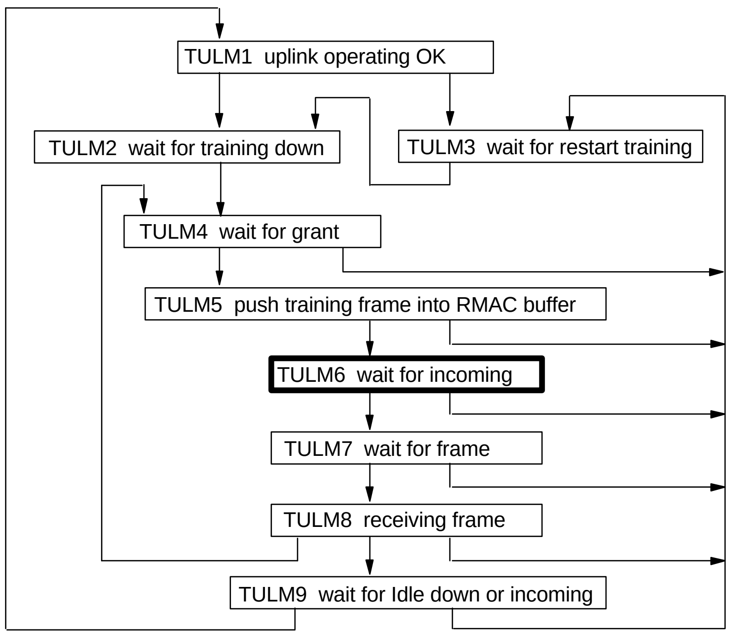

TULM1 uplink operating OK
TULM2 wait for training down
TULM3 wait for restart training
TULM4 wait for grant
TULM5 push training frame into RMAC buffer
TULM6 wait for incoming
TULM7 wait for frame
TULM8 receiving frame
TULM9 wait for Idle down or incoming

Figure 12-58—Relationship of TULM6 in the state machine

```
Case EVENT_SORT Of
  CASCADE_PORT_CONTROL_SIGNAL_DECODED :
  Case CASCADE_PORT_CONTROL_SIGNAL_IN Of
    INCOMING :
    Begin
      SELECTED_PORT := CASCADE_PORT;  {FCP uses SELECTED_PORT.}
      SIGNAL_TO_FCP(S_TO_FCP_EXPECT_FRAME);
      SIGNAL_TO_TSM(S_TO_TSM_GOTO_TX_INCOMING);
      {Need to send Incoming to potential destinations of training frame.}
      TULMSTATE := TULM7_WAIT_FOR_FRAME;
    End;

    LINK_WARNING : {If Link Fault when waiting for Incoming.}
      RESTART_TRAINING();
  End; {Case}
  TIMER_EXPIRES :
  If (TIMER = TR_REQ_UP_TO_FRAME_RECEIVED) Then
    RESTART_TRAINING();
End; {Case}
```

12.12.7 TULM7_WAIT_FOR_FRAME

In TULM7, INCOMING has arrived, and the TULM is waiting for the return training frame to arrive. When the return training frame starts to arrive, the TULM transitions to TULM8. If the timer REQ_UP_TO_FRAME_RECEIVED expires, or a LINK_WARNING state is received, the TULM again transitions back to the wait state, TULM3 (see Figure 12-59).

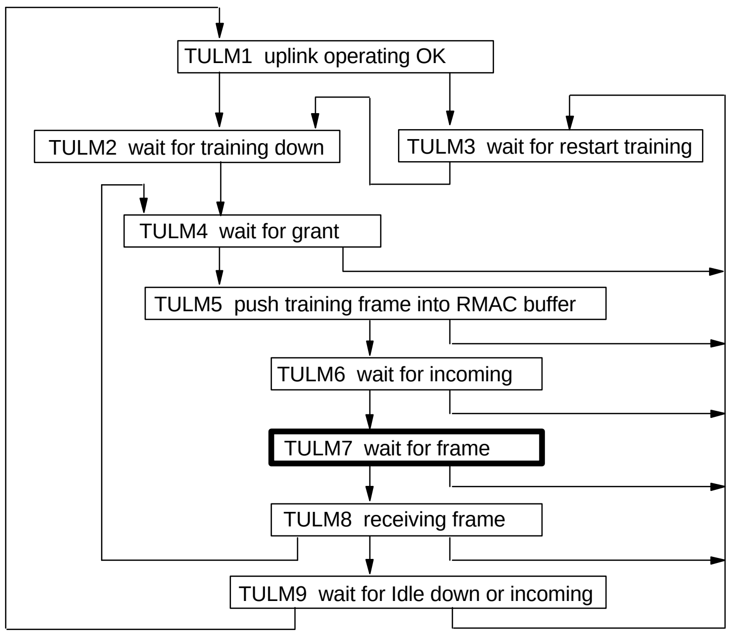

TULM1 uplink operating OK
TULM2 wait for training down
TULM3 wait for restart training
TULM4 wait for grant
TULM5 push training frame into RMAC buffer
TULM6 wait for incoming
TULM7 wait for frame
TULM8 receiving frame
TULM9 wait for Idle down or incoming

Figure 12-59—Relationship of TULM7 in the state machine

```
Case EVENT_SORT Of
  S_FCP_SMF_RECEIVED_THRU_CASCADE_PORT :
  Begin
    RMAC_BUF_DATA_INPUT := OCTETS_FROM_FCP;
    CANCEL_FAULT_TIMER(TR_REQ_UP_TO_FRAME_RECEIVED);
    TULMSTATE := TULM8_RECEIVING_FRAME;
  End;

  TIMER_EXPIRES :
  If (TIMER = TR_REQ_UP_TO_FRAME_RECEIVED) Then
  Begin
    SIGNAL_TO_FCP(S_TO_FCP_STOP_EXPECTING_FRAME);
    SIGNAL_TO_TSM(S_TO_TSM_GOTO_TX_IDLE);
    RESTART_TRAINING();
  End;

  CASCADE_PORT_CONTROL_SIGNAL_DECODED :
  If (CASCADE_PORT_CONTROL_SIGNAL_IN = LINK_WARNING) Then
    RESTART_TRAINING();
```

```
End; {Case}
```

12.12.8 TULM8_RECEIVING_FRAME

In TULM8, the return frame from the upper repeater is being received. If the packet is good, and its EMF_
INDICATION is not INVALID_PACKET_MARKER, then the good-packets counter, SUCCESSFUL_ TRAINING_PACKETS_IN_RUN, is incremented.

If the packet is bad, or the EMF_INDICATION is INVALID_PACKET_MARKER, then the counter is reset to zero.

If there have been 24 successive outward-return training packet pairs without error, then the training is considered a success, IDLE_U is sent out of the cascade port, and the TULM transitions to TULM9.

If the total number of training packet pairs reaches 48 without 24 successive good pairs, then the training is terminated, and the TULM goes to TULM3. If neither of these conditions holds, the TULM initiates the next packet pair by sending Req_H, and transitioning to TULM4 to wait for the GRANT from the upper repeater (see Figure 12-60).

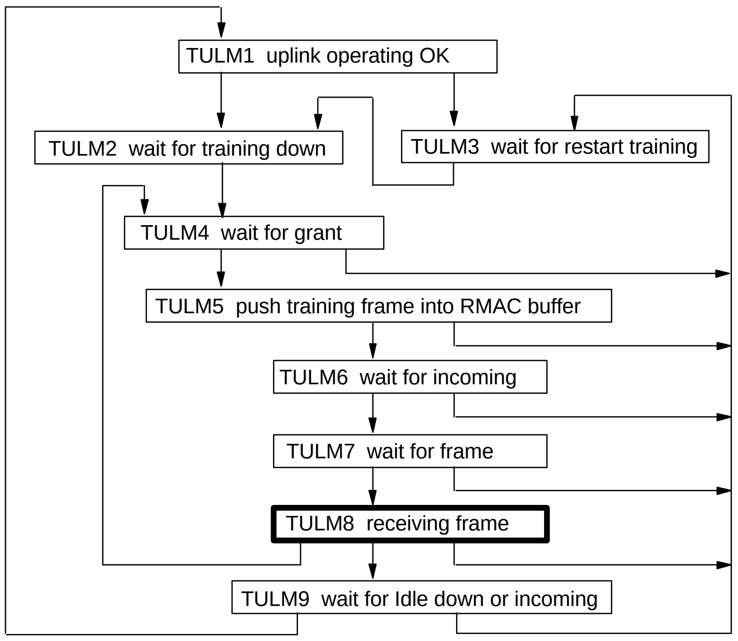

TULM1 uplink operating OK
TULM2 wait for training down
TULM3 wait for restart training
TULM4 wait for grant
TULM5 push training frame into RMAC buffer
TULM6 wait for incoming
TULM7 wait for frame
TULM8 receiving frame
TULM9 wait for Idle down or incoming

Figure 12-60—Relationship of TULM8 in the state machine

```
Case EVENT_SORT Of
  S_FCP_EMF_RECEIVED_THRU_CASCADE_PORT :
  Begin
RMAC_BUF_DATA_INPUT := STOP_INPUT;
    {Now update counters, followed by branching decision.}
    If (EMF_INDICATION <> INVALID_PACKET_MARKER) Then
      SUCCESSFUL_TRAINING_PACKETS_IN_RUN :=
          SUCCESSFUL_TRAINING_PACKETS_IN_RUN + 1
    Else SUCCESSFUL_TRAINING_PACKETS_IN_RUN := 0;
    If ((SUCCESSFUL_TRAINING_PACKETS_IN_RUN = 24) And
      (CONFIG_OK = True)) Then
    Begin
      SET_FAULT_TIMER(TR_FINAL_IDLE_OR_INCOMING);
      CHANGE_CASCADE_PORT_CONTROL_SIGNAL(IDLE_U);
      TULMSTATE := TULM9_WAIT_IDLE_OR_INCOMING;
    End
    Else
    If (((SUCCESSFUL_TRAINING_PACKETS_IN_RUN = 0) And
          (TOTAL_TRAINING_PACKETS_DONE > 24))
        Or (TOTAL_TRAINING_PACKETS_DONE = 48)
        Or ((SUCCESSFUL_TRAINING_PACKETS_IN_RUN = 24) And
            (CONFIG_OK = False)))
    Then RESTART_TRAINING()
    Else
    Begin {Send up Request to start on next training packet.}
      CHANGE_CASCADE_PORT_CONTROL_SIGNAL(REQ_H);
      SET_FAULT_TIMER(TR_REQ_UP_TO_FRAME_RECEIVED);
      FLAG_TRAINING_DOWN_RECEIVED := False;
      {To check get TRAINING_DOWN before next GRANT.}
      TULMSTATE := TULM4_WAIT_FOR_GRANT;
    End
  End;
End; {Case}
```

12.12.9 TULM9_WAIT_IDLE_OR_INCOMING

In TULM9, the TULM is sending up IDLE_U to the repeater above. The repeater above interprets this as meaning that the lower repeater considers that training has been satisfactorily completed.

The upper repeater will send down either IDLE_D or INCOMING, and this forms a handshake that allows the lower repeater to know that the upper repeater also considers training to be satisfactorily completed. The lower repeater signals to RSM RESTART_TRAFFIC and transitions to the quiescent state, TULM1 (see Figure 12-61).

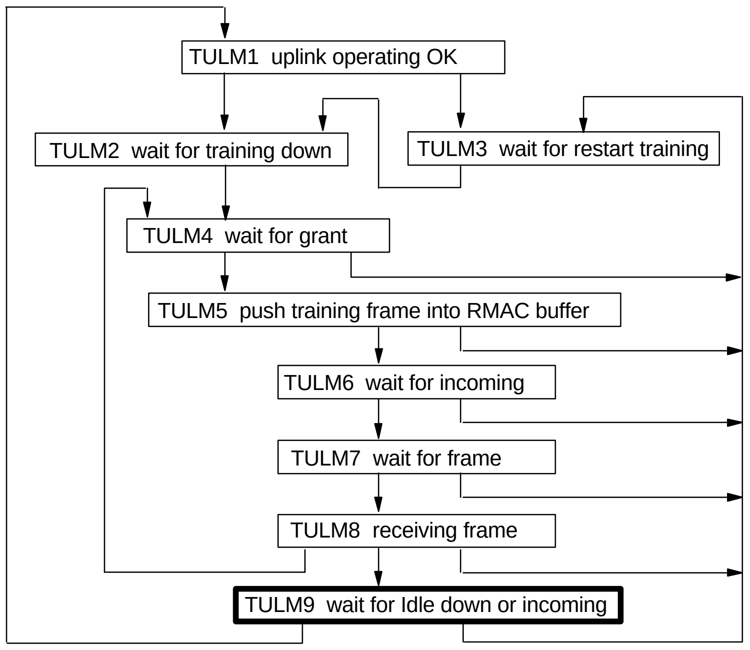

TULM1 uplink operating OK
TULM2 wait for training down
TULM3 wait for restart training
TULM4 wait for grant
TULM5 push training frame into RMAC buffer
TULM6 wait for incoming
TULM7 wait for frame
TULM8 receiving frame
TULM9 wait for Idle down or incoming

Figure 12-61—Relationship of TULM9 in the state machine

```
Case EVENT_SORT Of
  CASCADE_PORT_CONTROL_SIGNAL_DECODED :
  Case CASCADE_PORT_CONTROL_SIGNAL_IN Of
    IDLE_D, INCOMING :
    Begin {Confirmation from upper repeater that training is finished.  Back to TULM1.}
      LOWER_PORTS_TRAFFIC := OPERATING;
      SIGNAL_TO_RSM(S_TULM_TO_RSM_RESTART_TRAFFIC);
      CASCADE_PORT_META_STATE := ACTIVE;
      FLAG_INTERVAL_TO_RESTART_TRAINING_EXPIRED := False;
      SET_TIMER(TR_INTERVAL_TO_RESTART_TRAINING);
      TULMSTATE := TULM1_UP_LINK_OPERATING_OK;
    End;
  End; {Case}

  TIMER_EXPIRES :
  If (TIMER = TR_FINAL_IDLE_OR_INCOMING) Then
    RESTART_TRAINING();  {IDLE_D from upper repeater hasn't arrived; wait to restart
       training.}
End; {Case}
```

13. Layer management functions and services

13.1 Introduction

13.1.1 Purpose and scope

The layer management specification has been developed in accordance with the OSI management architecture as specified in ISO/IEC 7498-4:1989, and the specific requirements of IEEE Std 802.1F-1993.

This subclause defines network management functions and objects for both repeaters and end nodes. Network management is an optional feature in a demand-priority network, and may be included in the implementation of either repeaters or end nodes, or both. The relationship of the Layer Management Entity (LME) to the network model is shown in Figure 13-1.

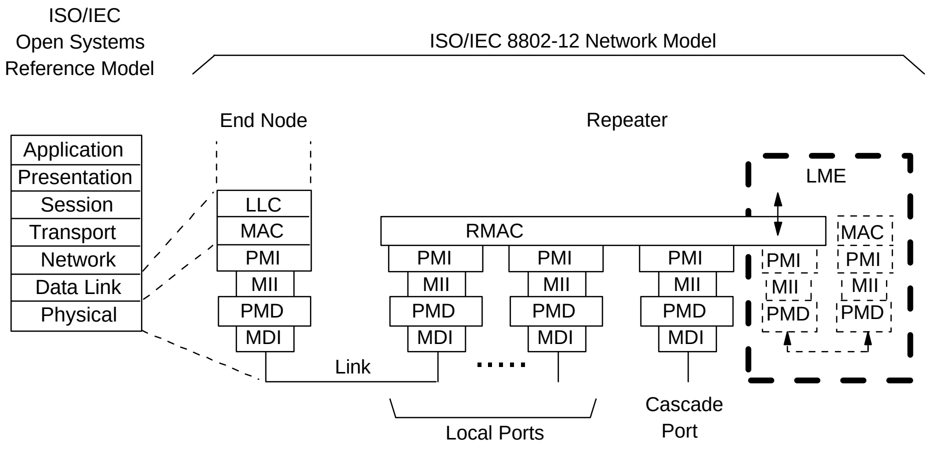


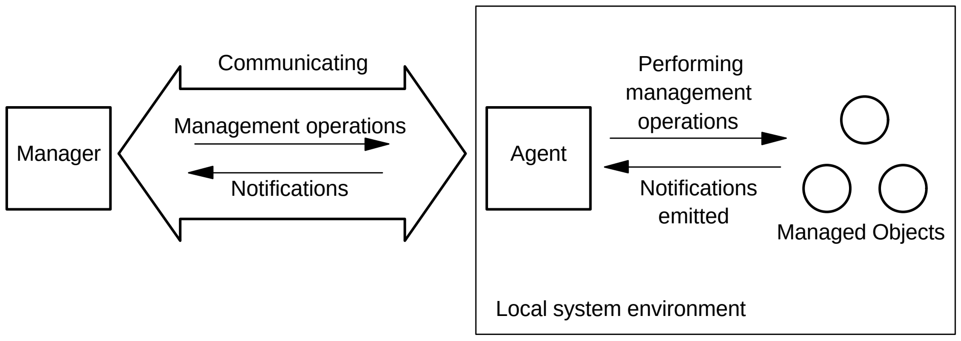
Figure 13-1—Relationship of the LME to the network model

The purpose of this subclause is to establish which management functions are considered to be mandatory and which are considered to be optional, should network management be included. Bandwidth allocation is beyond the scope of this standard.

13.1.2 Layer management model

This standard describes management of repeaters in terms of a general model of management of resources within the open systems environment. The model is described in ISO/IEC 10040:1992. A brief summary of the model is included here.

Management is viewed as a distributed application modeled as a set of interacting management processes. These processes are executed by systems within the open environment. A managing system executes a managing process that invokes management operations. A managed system executes a process that is receptive to these management operations and provides an interface to the resources to be managed. The agent is the interface between the managed objects and the manager. A managed object is the abstraction of a resource that represents its properties as seen by (and for the purpose of) management. Managed objects respond to a defined set of management operations. Managed objects are also capable of emitting a defined set of notifications. This interaction of processes is shown in Figure 13-2.



NOTE—This figure is based on Figure 1 of ISO/IEC 10040:1992. In the event of any conflict, the depiction in ISO/IEC 10040 takes precedence

Figure 13-2—Interaction between manager, agent, and objects

A managed object is a management view of a resource. The resource may be a logical construct, function, physical device, or anything subject to management. Managed objects are defined in terms of four types of elements:

a) *Attributes*. Data-like properties (as seen by management) of a managed object.
b) *Actions*. Operations that a managing process may perform on an object or its attributes.
c) *Notifications*. Unsolicited reports of events that may be generated by an object.
d) *Behaviour*. The way in which managed objects, attributes, and actions interact with the actual resources they model and with each other.

The above items are defined in Annex C of this standard in terms of the template requirements of ISO/IEC 10165-4:1992.

Some of the functions and resources within a repeater or end node are appropriate targets for management. They have been identified by specifying managed objects that provide a management view of the functions or resources. Within this general model, repeaters and end nodes are viewed as managed devices. They perform functions as defined by the applicable standard for such devices. Managed objects providing a view of those functions and resources appropriate to the management of repeaters and end nodes are specified. The purpose of this clause is to define the object classes associated with repeaters and end nodes in terms of their attributes, operations, notifications, and behaviour.

13.1.2.1 Managed object classes

13.1.2.1.1 Repeater-based managed object classes

a) **repeater.** This is the topmost managed object class of that portion of the containment tree shown in Figure 113. All other managed objects and their attributes defined in this subclause are contained within the repeater managed object.
b) **repeaterMonitor.** This is a managed object class called out by IEEE Std 802.1F-1993.
c) **resourceTypeID.** This is a managed object class called out by IEEE Std 802.1F-1993.
d) **group.** The group managed object class is a view of a collection of ports.

e) **port.** The port managed object class provides a view of the functional link between the repeater and the device at the other end of the cable. The attributes associated with a port deal with the monitoring of traffic being handled by the repeater for the port and control of the operation of the port. The port enable/disable function as reported by aPortAdministrativeState is preserved across events involving loss of power.

f) **link.** The link managed object class provides a view of the link and its current status.

13.1.2.1.2 End node based managed object classes

a) **EndNode.** This is the managed object class for all end node attributes, actions, and notifications defined in this standard.

b) **resourceTypeID.** This is a managed object class called out by IEEE Std 802.1F-1993.

13.1.2.2 Containment

A containment relationship is a structuring relationship for managed objects in which the existence of a managed object is dependent on the existence of a containing managed object. The contained managed object is said to be the subordinate managed object and the containing managed object is said to be the superior managed object. The containment relationship is used for naming managed objects. The local containment relationships among repeater object classes are depicted in Figure 13-3. The local containment relationships among end node object classes are depicted in Figure 13-4. These figures also show the names, naming attributes, and data attributes of the object classes as well as whether a particular containment relationship is one-to-one or one-to-many. For further requirements on this topic, see IEEE Std 802.1F-1993.

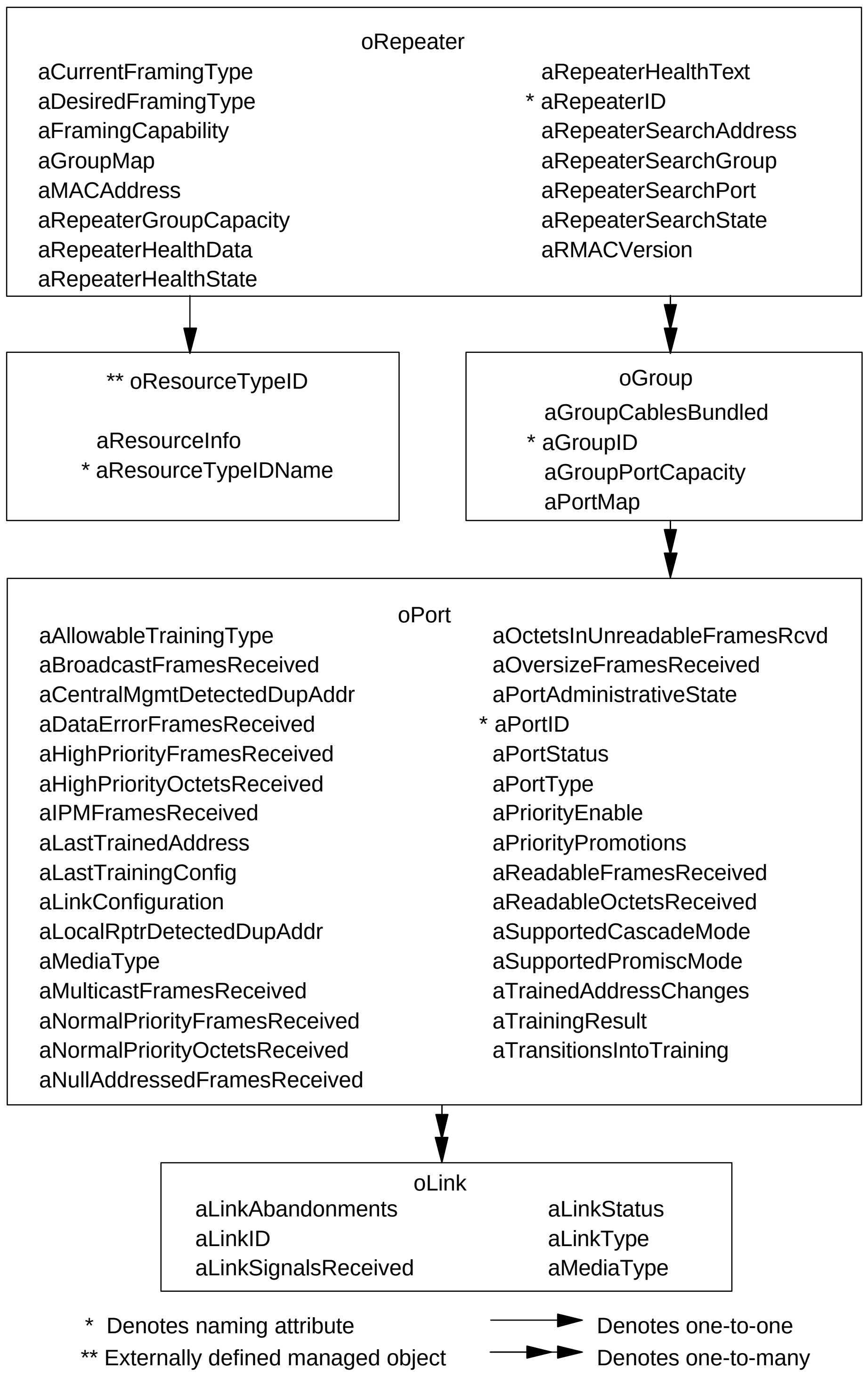


Figure 13-3—Repeater-entity-relationships diagram

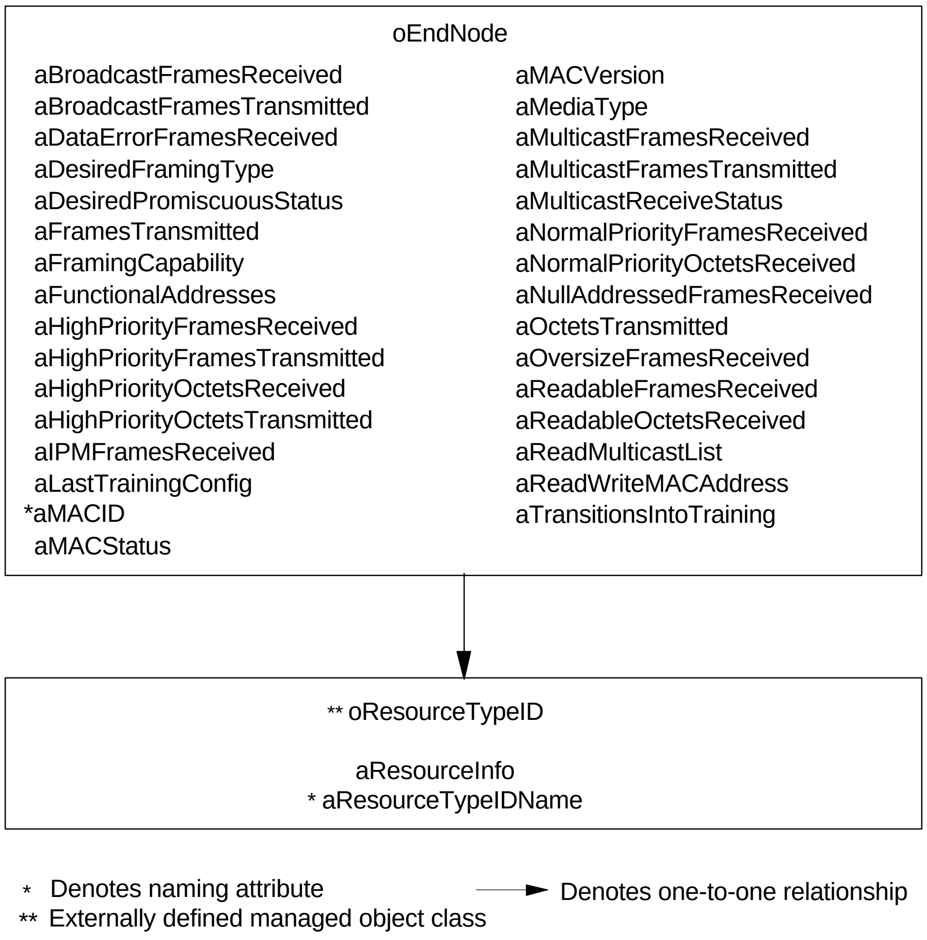


Figure 13-4—End node entity relationships diagram

13.1.2.3 Naming

The name of an individual managed object is hierarchically defined within a managed system. For example, a port might be identified as “repeater 3, group 1, port 13,” i.e., port 13 of group 01 of a repeater with repeaterID 3 within the managed system. This is represented in the relationship of the naming attributes in Figure 13-3.

13.1.3 Packages

This standard makes use of the concept of “packages” as defined in ISO/IEC 10165-4:1992 as a means of grouping behaviours, attributes, actions, and notifications within a managed object class definition. Within a standard, “capabilities” are defined, each of which corresponds to a set of packages, which are components of a number of managed object class definitions and which share the same condition for presence. The “Basic Capability” consists of the mandatory packages. The “Basic Counter Capability,” “Expanded Capability,” and “Link-Monitoring Capability” consist of optional packages. The repeater capabilities are described in 13.1.3.1. The end-node capabilities are described in 13.1.3.2. Annex C contains the GDMO templates that formally define the various packages.

13.1.3.1 Repeater based objects

13.1.3.1.1 Repeater managed object class

		Basic Capability (Mandatory)	Basic Counter Capability (Optional)	Expanded Capability (Optional)	Link-Monitoring Capability (Optional)
aCurrentFramingType	ATTRIBUTE GET	x			
aDesiredFramingType	ATTRIBUTE GET-REPLACE			x	
aFramingCapability	ATTRIBUTE GET			x	
aGroupMap	ATTRIBUTE GET	x			
aMACAddress	ATTRIBUTE GET	x			
aRepeaterGroupCapacity	ATTRIBUTE GET	x			
aRepeaterHealthData	ATTRIBUTE GET	x			
aRepeaterHealthState	ATTRIBUTE GET	x			
aRepeaterHealthText	ATTRIBUTE GET	x			
aRepeaterID	ATTRIBUTE GET	x			
aRepeaterSearchAddress	ATTRIBUTE GET			x	
aRepeaterSearchGroup	ATTRIBUTE GET			x	
aRepeaterSearchPort	ATTRIBUTE GET			x	
aRepeaterSearchState	ATTRIBUTE GET			x	
aRMACVersion	ATTRIBUTE GET	x			
acExecuteNonDisruptiveSelfTest	ACTION	x			
acRepeaterSearchAddress	ACTION			x	
acResetRepeater	ACTION	x			
nGroupMapChange	NOTIFICATION	x			
nRepeaterHealth	NOTIFICATION	x			
nRepeaterReset	NOTIFICATION	x			

13.1.3.1.2 Group managed object class

aGroupCablesBundled	ATTRIBUTE GET-REPLACE	x			
aGroupID	ATTRIBUTE GET	x			
aGroupPortCapacity	ATTRIBUTE GET	x			
aPortMap	ATTRIBUTE GET	x			
nPortMapChange	NOTIFICATION	x			

13.1.3.1.3 Port managed object class

aAllowableTrainingType	ATTRIBUTE GET-REPLACE			x	
aBroadcastFramesReceived	ATTRIBUTE GET			x	
aCentralMgmtDetectedDupAddr	ATTRIBUTE GET-REPLACE			x	
aDataErrorFramesReceived	ATTRIBUTE GET		x		
aHighPriorityFramesReceived	ATTRIBUTE GET		x		
aHighPriorityOctetsReceived	ATTRIBUTE GET			x	
aIPMFramesReceived	ATTRIBUTE GET			x	
aLastTrainedAddress	ATTRIBUTE GET		x		
aLastTrainingConfig	ATTRIBUTE GET			x	
aLinkConfiguration	ATTRIBUTE GET				x

		Basic Capability (Mandatory)	Basic Counter Capability (Optional)	Expanded Capability (Optional)	Link-Monitoring Capability (Optional)
aLocalRptrDetectedDupAddr	ATTRIBUTE GET			x	
aMediaType	ATTRIBUTE GET	x			
aMulticastFramesReceived	ATTRIBUTE GET			x	
aNormalPriorityFramesReceived	ATTRIBUTE GET			x	
aNormalPriorityOctetsReceived	ATTRIBUTE GET			x	
aNullAddressedFramesReceived	ATTRIBUTE GET			x	
aOctetsInUnreadableFramesRcvd	ATTRIBUTE GET			x	
aOversizeFramesReceived	ATTRIBUTE GET			x	
aPortAdministrativeState	ATTRIBUTE GET	x			
aPortID	ATTRIBUTE GET	x			
aPortStatus	ATTRIBUTE GET	x			
aPortType	ATTRIBUTE GET	x			
aPriorityEnable	ATTRIBUTE GET-REPLACE			x	
aPriorityPromotions	ATTRIBUTE GET		x		
aReadableFramesReceived	ATTRIBUTE GET		x		
aReadableOctetsReceived	ATTRIBUTE GET			x	
aSupportedCascadeMode	ATTRIBUTE GET	x			
aSupportedPromiscMode	ATTRIBUTE GET	x			
aTrainedAddressChanges	ATTRIBUTE GET			x	
aTrainingResult	ATTRIBUTE GET	x			
aTransitionsIntoTraining	ATTRIBUTE GET		x		
acPortAdministrativeControl	ACTION	x			

13.1.3.1.4 Link managed object class

		Basic Capability (Mandatory)	Basic Counter Capability (Optional)	Expanded Capability (Optional)	Link-Monitoring Capability (Optional)
aLinkAbandonments	ATTRIBUTE GET				x
aLinkID	ATTRIBUTE GET				x
aLinkSignalsReceived	ATTRIBUTE GET				x
aLinkStatus	ATTRIBUTE GET				x
aLinkType	ATTRIBUTE GET				x
aMediaType	ATTRIBUTE GET				x
acLinkAdministrativeControl	ACTION				x
nRedundantLinkFailure	NOTIFICATION				x

13.1.3.2 End node managed object class

		Basic Capability (Mandatory)	Basic Counter Capability (Optional)	Expanded Capability (Optional)
aBroadcastFramesReceived	ATTRIBUTE GET			x
aBroadcastFramesTransmitted	ATTRIBUTE GET			x
aDataErrorFramesReceived	ATTRIBUTE GET		x	
aDesiredFramingType	ATTRIBUTE GET-REPLACE			x

		Basic Capability (Mandatory)	Basic Counter Capability (Optional)	Expanded Capability (Optional)
aDesiredPromiscuousStatus	ATTRIBUTE GET-REPLACE			x
aFramesTransmitted	ATTRIBUTE GET		x	
aFramingCapability	ATTRIBUTE GET			x
aFunctionalAddresses	ATTRIBUTE GET-REPLACE	x		
aHighPriorityFramesReceived	ATTRIBUTE GET		x	
aHighPriorityFramesTransmitted	ATTRIBUTE GET		x	
aHighPriorityOctetsReceived	ATTRIBUTE GET			x
aHighPriorityOctetsTransmitted	ATTRIBUTE GET			x
aIPMFramesReceived	ATTRIBUTE GET			x
aLastTrainingConfig	ATTRIBUTE GET	x		
aMACID	ATTRIBUTE GET	x		
aMACStatus	ATTRIBUTE GET	x		
aMACVersion	ATTRIBUTE GET	x		
aMediaType	ATTRIBUTE GET	x		
aMulticastFramesReceived	ATTRIBUTE GET			x
aMulticastFramesTransmitted	ATTRIBUTE GET			x
aMulticastReceiveStatus	ATTRIBUTE GET-REPLACE			x
aNormalPriorityFramesReceived	ATTRIBUTE GET			x
aNormalPriorityOctetsReceived	ATTRIBUTE GET			x
aNullAddressedFramesReceived	ATTRIBUTE GET			x
aOctetsTransmitted	ATTRIBUTE GET			x
aOversizeFramesReceived	ATTRIBUTE GET			x
aReadableFramesReceived	ATTRIBUTE GET		x	
aReadableOctetsReceived	ATTRIBUTE GET			x
aReadMulticastList	ATTRIBUTE GET	x		
aReadWriteMACAddress	ATTRIBUTE GET-REPLACE			x
aTransitionsIntoTraining	ATTRIBUTE GET		x	
acAddGroupAddress	ACTION	x		
acClose	ACTION	x		
acDeleteGroupAddress	ACTION	x		
acExecuteSelftest	ACTION	x		
acInitializeMAC	ACTION	x		
acOpen	ACTION	x		

13.1.4 Conformance requirements

For a managed repeater to be conformant to this clause, it shall fully implement the Repeater Basic Capability. For a managed repeater to be conformant to an optional capability, it shall implement that entire capability. Repeater capabilities are summarized in 13.1.3.1.

For a managed end node to be conformant to this clause, it shall fully implement the End Node Basic Capability. For a managed end node to be conformant to an optional capability, it shall implement that entire capability. End node capabilities are summarized in 13.1.3.2.

13.2 Management facilities

13.2.1 Introduction

This subclause describes the network management capabilities of repeaters and end nodes. A large number of the management attributes, actions, and notifications are modeled after those defined in ISO/IEC 8802-3:1996. Although the attributes, actions, and notifications may have similar behaviour to those ISO/IEC 8802-3, they are registered under an ISO/IEC 8802-12 managed object class.

13.2.2 Counter operation

To prevent rapid counter rollover, the lengths of all octet counters shall be 64 bits. All other counters shall be 32 bits long. Octet counters count octets starting with the first octet of the DA field in ISO/IEC 8802-3 framing mode and with the first octet of the AC field in ISO/IEC 8802-5 mode, through the last octet of the FCS. The preamble, ssd, and esd are not included in octet counters.

If multiple status values can apply to the same packet, for packet counting purposes the order of counter precedence shall be: aNullAddressedFramesReceived, aOversizeFramesReceived, aIPMFramesReceived, aDataErrorFramesReceived, and aReadableFramesReceived. Only one of these counters will be incremented per packet.

Except for aLinkAbandonments, counters associated with the cascade port in repeaters equipped with redundant uplinks shall refer to the total traffic for the cascade port rather than for the individual uplinks.

13.2.3 Address formats

All MAC addresses shall be stored and reported in IEEE canonical form [i.e., the individual/group bit in the Least Significant Bit (LSB) position].

13.2.4 Repeater management facilities

This subclause defines the management of demand-priority repeaters by defining associated managed objects. This management encompasses two distinct aspects of repeater management.

The first aspect provides the means to monitor and control the functions of a repeater. These functions include, but are not limited to, identifying a repeater, testing and initializing a repeater, and enabling/disabling a port.

The second aspect provides the means to monitor traffic to or from attached segments. This is done by gathering statistics on packets that enter or exit a repeater and maintaining those statistics on a per-port basis.

13.2.4.1 Repeater functions to support management

To implement the layer management capabilities, this subclause refers to variables and states described in the RMAC protocol of this standard. At the end of each frame received, the RMAC shall invoke the UpdateLmeRmacRxCounters procedure to update the layer management packet receive counters. Other layer management attributes are handled as specified in the attribute behaviour text.

The per-port counters will actually be accessed using indexes of repeater, group, and port. To improve readability, only the port number is shown here.

13.2.4.1.1 Common constants and types

The following are the common constants and types required for the layer management procedures:

```
const
    maxPorts = 1024;
    addressSize = 6;
    nullAddress = array [ 1 .. addressSize ] of Octet value 0;
    maxOctetCount8023 = 1518;
    maxOctetCount8025 = 4520;

type
    Octet = (0 .. 255 );
    Counter32 = ( 0 .. (2**32-1) );
    Counter64 = ( 0 .. (2**64-1) );
    Counter32Array = array [ 1..maxPorts ] of Counter32;
    Counter64Array = array [ 1..maxPorts ] of Counter64;

    AddressValue = array [ 1 .. addressSize ] of Octet;
    PortType = (1..maxPorts );
    PktStatusType = ( valid, IPM, dataError );
    RxPriorityType = ( normal, high, promoted_to_hi );
```

13.2.4.1.2 Variables

Following are the variables used by the repeater layer management procedures:

```
var
    aBroadcastFramesReceived: Counter32Array;
    aDataErrorFramesReceived: Counter32Array;
    aHighPriorityFramesReceived: Counter32Array;
    aHighPriorityOctetsReceived: Counter64Array;
    aIPMFramesReceived: Counter32Array;
    aLinkAbandonments: Counter32Array;
    aMulticastFramesReceived: Counter32Array;
    aNormalPriorityFramesReceived: Counter32Array;
    aNormalPriorityOctetsReceived: Counter64Array;
    aNullAddressedFramesReceived: Counter32Array;
    aOctetsInUnreadableFramesRcvd: Counter64Array;
    aOversizeFramesReceived: Counter32Array;
    aPriorityPromotions: Counter32Array;
    aReadableFramesReceived: Counter32Array;
    aReadableOctetsReceived: Counter64Array;
```

13.2.4.1.3 Procedures

```
Procedure UpdateLmeRmacRxCounters
                              ( port                   : PortType;
                                octetCount                     : integer;
                                destinationAddress     : AddressValue;
                                priority                       : RxPriorityType;
                                status                 : PktStatusType );
```

```
Begin
   Case priority of

     normal :
        Begin
          Increment ( aNormalPriorityFramesReceived [ port ] ) ;
          aNormalPriorityOctetsReceived [ port ] := aNormalPriorityOctetsReceived [ port ]
                                                   + octetCount;
        End

     high :
        Begin
          Increment ( aHighPriorityFramesReceived [ port ]  ) ;
          aHighPriorityOctetsReceived [ port ] :=          aHighPriorityOctetsReceived [ port ]
                                                 + octetCount;
        End

     promoted_to_hi :
        Begin
          Increment ( aPriorityPromotions [ port ]  ) ;
          Increment ( aNormalPriorityFramesReceived [ port ] ) ;
          aNormalPriorityOctetsReceived [ port ] := aNormalPriorityOctetsReceived [ port ]
                                                  + octetCount;
        End

   End;  {Case priority}

   If  ( ( destinationAddress = nullAddress ) And ( octetCount >= 6 ) )
      Begin
         Increment ( aNullAddressedFramesReceived [ port ] );
         aOctetsInUnreadableFramesRcvd [ port ]  := aOctetsInUnreadableFramesRcvd [port]
                                                       + octetCount;
      End
   Else

   If ( (aCurrentFramingType = FrameType8023) And (octetCount > maxOctetCount8023) Or
      (aCurrentFramingType = FrameType8025) And (octetCount >maxOctetCount8025) )Then
      Begin
         Increment ( aOversizeFramesReceived [ port ] );
         aOctetsInUnreadableFramesRcvd [ port ]  := aOctetsInUnreadableFramesRcvd [port]
                                                       + octetCount;
      End
   Else
   Begin
      Case status of

        valid:
           Begin
             Increment ( aReadableFramesReceived [ port ] );
             aReadableOctetsReceived [ port ]  := aReadableOctetsReceived [ port ]
                                                + octetCount;
             If ( destinationAddress is a group address ) Then
                If ( destinationAddress = broadcastAddress ) Then
                   Increment ( aBroadcastFramesReceived [ port ] );
```

```
               Else
                  Increment ( aMulticastFramesReceived [ port ] );
            End

         IPM :
            Begin
              Increment ( aIPMFramesReceived [ port ] );
              aOctetsInUnreadableFramesRcvd [ port ]  =
                                        aOctetsInUnreadableFramesRcvd[port]+ octetCount;
            End

         dataError:
            Begin
              Increment ( aDataErrorFramesReceived [ port ] );
              aOctetsInUnreadableFramesRcvd [ port ]  =
                                 aOctetsInUnreadableFramesRcvd [port]+ octetCount;
            End
       End; {Case PktStatus }
    End  { Else }
End  { Procedure UpdateLmeRmacRxCounters }
```

13.2.4.2 Repeater managed object class

13.2.4.2.1 Repeater attributes

aCurrentFramingType

ATTRIBUTE

APPROPRIATE SYNTAX: ENUMERATED VALUE LIST:

frameType8023	- ISO/IEC 8802-3 framing
frameType8025	- ISO/IEC 8802-5 framing

BEHAVIOUR DEFINED AS:

This attribute is the type of framing currently in use by the repeater. It is the framing type that will be used in the requested configuration field of the training frame sent to an upper-level repeater connected to the cascade port.

aDesiredFramingType

ATTRIBUTE

APPROPRIATE SYNTAX: ENUMERATED VALUE LIST:

frameType8023	- ISO/IEC 8802-3 framing
frameType8025	- ISO/IEC 8802-5 framing

BEHAVIOUR DEFINED AS:

This attribute value has no relevance to the current operating state of the repeater. It is set by the network manager and is the framing type that will be used by the repeater the next time it is reset. The value of this attribute shall be preserved across repeater resets and power failures.

aFramingCapability
ATTRIBUTE
APPROPRIATE SYNTAX: ENUMERATED VALUE LIST:

frameType8023	- ISO/IEC 8802-3 framing
frameType8025	- ISO/IEC 8802-5 framing
frameTypeEither	- Either ISO/IEC 8802-3 or ISO/IEC 8802-5 framing

BEHAVIOUR DEFINED AS:

This attribute indicates the type of framing the repeater is capable of supporting.

aGroupMap
ATTRIBUTE
APPROPRIATE SYNTAX: BIT STRING
BEHAVIOUR DEFINED AS:

A string of bits that reflects the current configuration of units that are viewed by group managed objects. The length of the bit string is “aRepeaterGroupCapacity” bits. The first bit relates to group 1. A “1” in the bit string indicates the presence of the group, a “0” represents absence of the group.

aMACAddress
ATTRIBUTE
APPROPRIATE SYNTAX: MACAddress
BEHAVIOUR DEFINED AS:

This is the address used by the repeater when it initiates training on the uplink port. Repeaters are allowed to train with an assigned MAC address or a null (all zeros) MAC address.

aRepeaterGroupCapacity
ATTRIBUTE
APPROPRIATE SYNTAX: INTEGER
BEHAVIOUR DEFINED AS:

The aRepeaterGroupCapacity attribute is the number of groups that can be contained within the repeater. Within each managed repeater, the groups are uniquely numbered in the range from 1 to aRepeaterGroupCapacity.

Some groups may not be present in a given repeater instance, in which case the actual number of groups present is less than aRepeaterGroupCapacity. The number of groups present is never greater than aRepeaterGroupCapacity.

aRepeaterHealthData
ATTRIBUTE
APPROPRIATE SYNTAX: OCTET STRING, 0–255
BEHAVIOUR DEFINED AS:

The aRepeaterHealthData attribute is a block of data octets that provides information relevant to the operational state of the repeater. The encoding of this data block is vendor dependent. Repeater vendors may use this mechanism to provide detailed failure information or instructions for problem resolution.

aRepeaterHealthState
ATTRIBUTE
APPROPRIATE SYNTAX:
An ENUMERATED VALUE LIST:

other	- Undefined or unknown
ok	- No known failures
repeaterFailure	- Known to have a repeater-related failure
groupFailure	- Known to have a group-related failure
portFailure	- Known to have a port-related failure
generalFailure	- Has a failure condition of unspecified type

BEHAVIOUR DEFINED AS:

The aRepeaterHealthState attribute indicates the operational state of the repeater. The aRepeaterHealthData and aRepeaterHealthText attributes may be consulted for more specific information about the state of the repeater's health. In case of multiple kinds of failures (e.g., repeater failure and port failure), the value of this attribute shall reflect the highest priority in the following order:

repeater failure
group failure
port failure
general failure

aRepeaterHealthText
ATTRIBUTE
APPROPRIATE SYNTAX: A PrintableString, 255 characters max
BEHAVIOUR DEFINED AS:

The aRepeaterHealthText attribute is a text string that provides information relevant to the operational state of the repeater. Repeater vendors may use this mechanism to provide detailed failure information or instructions for problem resolution. The contents are vendor specific.

aRepeaterID
ATTRIBUTE
APPROPRIATE SYNTAX: INTEGER
BEHAVIOUR DEFINED AS:

The value of aRepeaterID is assigned so as to uniquely identify a repeater among the subordinate managed objects of the system. (The system is defined in ISO/IEC 10165-2:1992.)

aRepeaterSearchAddress
ATTRIBUTE
APPROPRIATE SYNTAX: MACAddress
BEHAVIOUR DEFINED AS:

When an error-free frame is received with a source address field that matches aRepeaterSearchAddress, the repeater will update aRepeaterSearchPort and aRepeaterSearchGroup to reflect the port and group from which the frame was received. This capability is used to draw a topologically correct map of a network that includes cascaded repeaters.

aRepeaterSearchGroup
ATTRIBUTE
APPROPRIATE SYNTAX: INTEGER
BEHAVIOUR DEFINED AS:

The group from which an error-free frame from aRepeaterSearchAddress has been received. The value of aRepeaterSearchGroup is undefined when aRepeaterSearchState is “None” or “Multiple.”

aRepeaterSearchPort
ATTRIBUTE
APPROPRIATE SYNTAX: INTEGER
BEHAVIOUR DEFINED AS:

The port from which an error-free frame from aRepeaterSearchAddress has been received. The value of aRepeaterSearchPort is undefined when aRepeaterSearchState is “None” or “Multiple.”

aRepeaterSearchState
ATTRIBUTE
APPROPRIATE SYNTAX: ENUMERATED VALUE LIST:

none -	No error-free frames have been received with a Source Address that matches aRepeaterSearchAddress. aRepeaterSearchState is initialized to this value whenever acRepeaterSearchAddress is set.
single -	All error-free frames with a Source Address matching aRepeaterSearchAddress have been received from the same port.
multiple-	Error-free frames with a Source Address matching aRepeaterSearchAddress have been received from more than one port.

BEHAVIOUR DEFINED AS:

This attribute indicates the current state of the search for aRepeaterSearchAddress and may take on one of the three values shown above.

aRMACVersion
ATTRIBUTE
APPROPRIATE SYNTAX: BIT STRING
BEHAVIOUR DEFINED AS:

This repeater-wide read-only attribute is the highest version (the vvv bits) supported by the repeater during training.

13.2.4.2.2 Repeater actions

acExecuteNonDisruptiveSelfTest
ACTION
APPROPRIATE SYNTAX: None required
BEHAVIOUR DEFINED AS:

The repeater performs a vendor-specific, nondisruptive self-test that has the following characteristics:

a) The components are not specified.
b) The test does not change the state of the repeater or management information about the repeater.

c) The test does not inject packets onto any segment.
d) The test does not prevent the transfer of any packets.
e) Completion of the test causes an nRepeaterHealth notification to be sent.

acRepeaterSearchAddress
ACTION
APPROPRIATE SYNTAX: MACAddress
BEHAVIOUR DEFINED AS:

Provides a means to alter aRepeaterSearchAddress, initialize aRepeaterSearchState, and initiate a search for the specified MAC address. When an error-free frame is received with a Source Address that matches aRepeaterSearchAddress, the repeater will update aRepeaterSearchState, aRepeaterSearchGroup, and aRepeaterSearchPort to reflect the current status of the search, and the group and port from which the frame was received. This capability is used to draw a topologically correct map of a network that includes cascaded repeaters.

acResetRepeater
ACTION
APPROPRIATE SYNTAX: None required
BEHAVIOUR DEFINED AS:

This action causes the repeater to transition to its initial state as specified in Clause 12. The repeater performs a disruptive self-test that has the following characteristics:

a) The components are not specified.
b) The test resets the repeater without affecting configurable management information. Counters may only be affected if the reset causes a change to aCurrentFramingType.
c) Packets received during the test may or may not be transferred.
d) The test may only interfere with management functions if the reset causes a change to aCurrentFramingType.

This causes an nRepeaterReset notification to be sent.

13.2.4.2.3 Repeater notifications

nGroupMapChange
NOTIFICATION
APPROPRIATE SYNTAX: BIT STRING
BEHAVIOUR DEFINED AS:

This notification is sent when a change occurs in the group structure of a repeater. This occurs only when a group is logically removed from or added to a repeater. The nGroupMapChange notification is not sent when powering up a repeater. The value of the notification is the updated value of the aGroupMap attribute.

nRepeaterHealth
NOTIFICATION
APPROPRIATE SYNTAX: A SEQUENCE of 3 data types.

The first is mandatory, and the following two are optional. The first is the value of the attribute aRepeaterHealthState. The second is the value of the attribute aRepeaterHealthText. The third is the value of the attribute aRepeaterHealthData.

BEHAVIOUR DEFINED AS:

This notification conveys information related to the operational state of the repeater. See the aRepeaterHealthState, aRepeaterHealthText, and aRepeaterHealthData attributes for descriptions of the information that is sent.

The nRepeaterHealth notification is sent only when the health state of the repeater changes. The nRepeaterHealth notification shall contain aRepeaterHealthState. aRepeaterHealthData and aRepeaterHealthText may or may not be included. The nRepeaterHealth notification is not sent as a result of powering up a repeater.

nRepeaterReset
NOTIFICATION
APPROPRIATE SYNTAX: A SEQUENCE of 3 data types.

The first is mandatory, and the following two are optional. The first is the value of the attribute aRepeaterHealthState. The second is the value of†the attribute aRepeaterHealthText. The third is the value of the attribute aRepeaterHealthData.

BEHAVIOUR DEFINED AS:

This notification conveys information related to the operational state of the repeater. The nRepeaterReset notification is sent when the repeater is reset as the result of a power-on condition or upon completion of the acResetRepeater action. The nRepeaterReset notification shall contain aRepeaterHealthState. aRepeaterHealthData and aRepeaterHealthText may or may not be included.

13.2.4.3 ResourceTypeID managed object class

Implementation of this managed object in accordance with the definition contained in IEEE Std 802.1F-1993 is a conformance requirement of this standard. A single instance of the Resource Type ID managed object exists within the repeater managed object class. The managed object itself is contained in IEEE Std 802.1F-1993; therefore, only the name binding appears in this standard.

13.2.4.4 Group managed object class

13.2.4.4.1 Group attributes

aGroupCablesBundled
ATTRIBUTE
APPROPRIATE SYNTAX: ENUMERATED VALUE LIST:

someCablesBundled	- One or more cables in this group may be in a bundle
noCablesBundled	- No cables in this group are in a bundle

BEHAVIOUR DEFINED AS:

This flag is used to select either bundled or unbundled cabling. When this flag is someCablesBundled and the port is not promiscuous or cascaded, frames received from ports in this group and destined to go out through multiple ports in this group will be buffered completely before being retransmitted out ports in this group. When this flag is noCablesBundled, or the port is promiscuous or cascaded, these frames will be retransmitted out ports in this group as the frame is being received.

Note that the value of “someCablesBundled” will work in the vast majority of all installations, regardless of whether or not any cables are physically in a bundle, since promiscuous and cascaded ports automatically avoid the store and forward.

The main situation in which “noCablesBundled” is beneficial is when there is a large amount of multicast traffic and the cables are not in a bundle. The value of this attribute shall be preserved across repeater resets and power failures.

aGroupID
ATTRIBUTE
APPROPRIATE SYNTAX: INTEGER
BEHAVIOUR DEFINED AS:

An integer value unique within the repeater. This value is never greater than aRepeaterGroupCapacity.

aGroupPortCapacity
ATTRIBUTE
APPROPRIATE SYNTAX: INTEGER
BEHAVIOUR DEFINED AS:

The aGroupPortCapacity is the number of ports contained within the group. Valid range is 1–1024. Within each group, the ports are uniquely numbered in the range from 1 to aGroupPortCapacity. Some ports may not be present in a given group instance, in which case the actual number of ports present is less than aGroupPortCapacity. The number of ports present is never greater than aGroupPortCapacity.

aPortMap
ATTRIBUTE
APPROPRIATE SYNTAX: BIT STRING
BEHAVIOUR DEFINED AS:

A string of bits that reflects the current configuration of port managed objects within this group. The length of the bit string is “aGroupPortCapacity” bits. The first bit relates to port 1. A “1” in the bit string indicates presence of the port, a “0” represents absence of the port.

13.2.4.4.2 Group actions

There are no group actions currently defined.

13.2.4.4.3 Group notifications

nPortMapChange
NOTIFICATION
APPROPRIATE SYNTAX: BIT STRING
BEHAVIOUR DEFINED AS:

This notification is sent when a change occurs in the port structure of a group. This occurs only when a port is logically removed from or added to a group. The nPortMapChange notification is not sent when powering up a repeater. The value of the notification is the updated value of the aPortMap attribute.

13.2.4.5 Port managed object class

13.2.4.5.1 Port attributes

aAllowableTrainingType
ATTRIBUTE
APPROPRIATE SYNTAX: ENUMERATED VALUE LIST:

allowEndNodesOnly	- Only nonpromiscuous end nodes permitted
allowPromiscuousEndNodes	- Promiscuous or nonpromiscuous end nodes permitted
allowEndNodesOrRepeaters	- Repeaters or nonpromiscuous end nodes permitted
allowAnything	- Repeaters or promiscuous or nonpromiscuous end nodes permitted

BEHAVIOUR DEFINED AS:

This security attribute is set by the network manager to configure what type of device is permitted to connect to the port. The values of “allowEndNodesOnly” and “allowPromiscEndNodes” may not be used for the cascade port. The LME shall reject a set of aAllowableTrainingType if it includes no capabilities that are supported by the repeater, as indicated by aSupportedPromiscMode and aSupportedCascadeMode. The value of this attribute shall be preserved across repeater resets and power failures.

aBroadcastFramesReceived
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of good broadcast frames received. These frames are also counted by aReadableFramesReceived. This counter is updated as specified in 13.2.4.1.3.

aCentralMgmtDetectedDupAddr
ATTRIBUTE
APPROPRIATE SYNTAX: BOOLEAN
BEHAVIOUR DEFINED AS:

This attribute can be set by the central network management station when it detects that there is a duplicate MAC address. This attribute is OR’d with aLocalRptrDetectedDupAddr to form the value of the “D” bit in the Allowed Configuration field of training response frames on this port. The purpose of this attribute is to provide a means for network management software to inform an end node that it is using a duplicate station address. Setting this object does not affect the current state of the link; the end node will not be informed of the duplicate address until it retrains for some reason. Note that regardless of its station address, the end node will not be able to train successfully until the network manager has cleared this bit. Although this variable exists for the cascade port, it does not perform any function since this repeater is the initiator of training on the cascade port.

aDataErrorFramesReceived
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of frames received with any of the following errors: bad FCS (with no IPM), PMI errors (excluding frames with an IPM error as the only PMI error), or undersize (with no IPM). Does not include frames counted by aIPMFramesReceived, aOversizeFramesReceived, or aNullAddressedFramesReceived. This counter is updated as specified in 13.2.4.1.3.

aHighPriorityFramesReceived
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of high-priority frames received. Includes both good and bad high-priority frames, as well as high-priority training frames. Does not include normal-priority frames that were priority promoted. This counter is updated as specified in 13.2.4.1.3.

aHighPriorityOctetsReceived
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of octets in aHighPriorityFramesReceived. This counter is updated as specified in 13.2.4.1.3.

aIPMFramesReceived
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of frames with an IPM and no PMI errors received. A repeater will write an IPM to the end of a frame containing errors as it is forwarded through the repeater to the other ports. This counter indicates problems with remote cable segments, as opposed to problems with cables directly attached to this repeater. This counter is updated as specified in 13.2.4.1.3.

aLastTrainedAddress
ATTRIBUTE
APPROPRIATE SYNTAX: MACAddress
BEHAVIOUR DEFINED AS:

The MAC address of the last node that succeeded in training on this port. A cascaded repeater may train using the null address. If no nodes have succeeded in training on this port since the most recent management system reset, this value will be the null address. For the cascade port, this value shall be the null address.

aLastTrainingConfig
ATTRIBUTE
APPROPRIATE SYNTAX: OCTET STRING
BEHAVIOUR DEFINED AS:

This 16-bit field contains the most recent training configuration requested in an error-free training frame sent by the training initiator connected to the port. For the cascade port, this is the responder’s configuration field from the most recent error-free training response frame received in response to training initiated by the repeater. The training configuration field formats are described in 10.6.

aLinkConfiguration
ATTRIBUTE
APPROPRIATE SYNTAX: ENUMERATED VALUE LIST:

single	- this port is equipped with a single link
redundant	- this port is equipped with redundant links

BEHAVIOUR DEFINED AS:

aLinkConfiguration reflects the current configuration of link managed objects within this port.

aLocalRptrDetectedDupAddr
ATTRIBUTE
APPROPRIATE SYNTAX: BOOLEAN
BEHAVIOUR DEFINED AS:

For local ports, this is a read-only bit that may be set by the repeater when an error-free training frame is received on this port with a source MAC address that matches aLastTrainedAddress of another port that is active. This bit is cleared when an error-free training frame is received with a non-null source MAC address that does not match aLastTrainedAddress of another port that is active.

For the cascade port, this bit shall take on the value of the “D” bit in the most recently received error-free training response frame.

aMediaType
ATTRIBUTE
APPROPRIATE SYNTAX: ENUMERATED VALUE LIST:

other	-	undefined
unknown	-	true state not known
pmdMissing	-	PMD device not attached
utp4	-	4-pair unshielded twisted pair
stp2	-	2-pair shielded twisted pair
fibre	-	802.12 fibre-optic cabling

BEHAVIOUR DEFINED AS:

aMediaType indicates the type of media currently in use. If aMediaType refers to a cascade port with redundant uplinks, aMediaType will reflect the media type of the uplink that is currently active. If neither uplink is active, aMediaType will reflect the media type of the primary uplink. aMediaType may be “unknown” if the implementation is not capable of identifying the PMD media type, or whether or not the PMD is even present. An implementation is capable of returning a value of “pmdMissing” only if it possesses a means to distinguish between a missing PMD and a PMD that is not driving the PMD Config[3:0] pins.

aMulticastFramesReceived
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of good multicast frames received. These frames are also counted by aReadableFramesReceived, but not by aBroadcastFramesReceived. This counter is updated as specified in 13.2.4.1.3.

aNormalPriorityFramesReceived
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of normal-priority frames received. Includes both good and bad normal-priority frames, as well as normal-priority training frames. Also includes normal-priority frames that were priority promoted. This counter is updated as specified in 13.2.4.1.3.

aNormalPriorityOctetsReceived
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of octets in aNormalPriorityFramesReceived. This counter is updated as specified in 13.2.4.1.3.

aNullAddressedFramesReceived
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of frames received with a destination MAC address of all zeros. Both void and training frames are included in this counter. This counter is updated as specified in 13.2.4.1.3.

aOctetsInUnreadableFramesRcvd
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of octets in frames counted by aIPMFramesReceived, aOversizeFramesReceived, aNullAddressedFramesReceived, or aDataErrorFramesReceived. This counter can be combined with aReadableOctetsReceived to calculate network utilization. This counter is updated as specified in 13.2.4.1.3.

aOversizeFramesReceived
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of oversize frames received. The frame size that causes this counter to increment is dependent on aCurrentFramingType. When aCurrentFramingType is frameType8025, the size is 4521 octets or larger. For frameType8023, the size is 1519 octets or larger. This counter is updated as specified in 13.2.4.1.3.

aPortAdministrativeState
ATTRIBUTE
APPROPRIATE SYNTAX: ENUMERATED VALUE LIST:

disabled
enabled

BEHAVIOUR DEFINED AS:

Port enable/disable function. Enabling a disabled port will cause training to be initiated. A disabled port neither transmits nor receives. The port shall be explicitly enabled to restore operation. The acPortAdministrativeControl action provides this ability. The port enable/disable function as reported by this attribute is preserved across repeater reset, including loss of power.

aPortID

ATTRIBUTE
APPROPRIATE SYNTAX: INTEGER
BEHAVIOUR DEFINED AS:

A value unique in the group. It is assumed that ports are partitioned into groups that also have IDs. This value can never be greater than aGroupPortCapacity.

aPortStatus

ATTRIBUTE
APPROPRIATE SYNTAX: ENUMERATED VALUE LIST:

active	- Training completed successfully
inactive	- Port is silent
training	- Training in progress

BEHAVIOUR DEFINED AS:

The status of the port, as specified by PORT_META_STATE in the port process module of Clause 12.

aPortType

ATTRIBUTE
APPROPRIATE SYNTAX: ENUMERATED VALUE LIST:

cascadeInternal	-	Port is an uplink with physical connections that are not externally visible, such as a connection to an internal backplane in a chassis.
cascadeExternal	-	Port is an uplink with physical connections that are externally visible.
localInternal	-	Port is a downlink or local port, with connections that are not externally visible, such as a connection to an internal LME agent.
localExternal	-	Port is a downlink or local port, with externally visible connections.

BEHAVIOUR DEFINED AS:

Describes the type of port. “Internal” is used to identify ports that place traffic into the repeater, but do not have any external connections.

aPriorityEnable

ATTRIBUTE
APPROPRIATE SYNTAX: BOOLEAN
BEHAVIOUR DEFINED AS:

A flag used to determine whether the repeater will service high-priority requests from end nodes on the port as high priority or normal priority. When FALSE, high-priority requests from an end node on this port will be serviced as normal-priority. The value of this attribute shall be preserved across repeater resets and power failures. This attribute has no effect on the cascade port.

aPriorityPromotions

ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of times the priority promotion timer has expired on this port and a normal-priority packet received was priority promoted. This counter is updated as specified in 13.2.4.1.3.

aReadableFramesReceived
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of good frames received. A good frame is defined as a frame that is not counted by any of the following error counters: aIPMFramesReceived, aOversizeFramesReceived, aNullAddressedFramesReceived, or aDataErrorFramesReceived. This counter is updated as specified in 13.2.4.1.3.

aReadableOctetsReceived
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of octets in frames counted by aReadableFramesReceived. This counter is updated as specified in 13.2.4.1.3.

aSupportedCascadeMode
ATTRIBUTE
APPROPRIATE SYNTAX: ENUMERATED VALUE LIST:

hwSupportsEndNodesOnly	- Local port supports end nodes only
hwSupportsEndNodesOrRptrs	- Local port supports cascaded repeaters or end nodes
cascadePort	- Used for the cascade port

BEHAVIOUR DEFINED AS:

This per-port read-only attribute describes whether the hardware is capable of supporting cascaded repeaters, or end nodes, or both. “cascadePort” shall be returned for the cascade port.

aSupportedPromiscMode
ATTRIBUTE
APPROPRIATE SYNTAX: ENUMERATED VALUE LIST:

hwSupportsSingleModeOnly
hwSupportsSingleOrPromiscMode
hwSupportsPromiscModeOnly

BEHAVIOUR DEFINED AS:

This per-port read-only attribute describes whether the hardware is capable of supporting promiscuous mode or single address mode (i.e., repeater filters unicasts not addressed to station on port), or both. “hwSupportsPromiscModeOnly” shall be returned for the cascade port.

aRMACVersion
ATTRIBUTE
APPROPRIATE SYNTAX: BIT STRING
BEHAVIOUR DEFINED AS:

This repeater-wide read-only attribute is the highest version (the “vvv” bits) supported by the repeater during training.

aTrainedAddressChanges
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of changes to aLastTrainedAddress.

aTrainingResult
ATTRIBUTE
APPROPRIATE SYNTAX: BIT STRING
BEHAVIOUR DEFINED AS:

This 18-bit field is used by layer management to obtain the result of training. It contains two bits that indicate if error-free training frames have been received, and it also contains the 16 bits of the most recent error-free training response frame on the port.

16 bits of the allowed configuration field of the training response frame (See Figure 10-17)	LinkGood	Valid

LinkGood	-	Indicates the link hardware is OK. Set to "1" if 24 consecutive error-free training packets have been received. Cleared when a training packet with errors is received, and when aPortStatus transitions to the "inactive" or "training" states.
Valid	-	Set when at least one error-free training frame has been received. Indicates if the training response frame information in aLastTrainingConfig and aTrainingResult is valid. This bit is cleared when aPortStatus transitions to the "inactive" or "training" states.

If the port is in training, layer management can examine this register to see if any training packets have been passed successfully. If there have been any good training packets, the Valid bit will be set and layer management can examine the 16 training response bits to see if there is a duplicate address, or a configuration or security problem.

NOTE—On a repeater's local port, this repeater generates the allowed configuration bits for the training response frame, while on the cascade port, the higher-level repeater originated the allowed configuration bits.

aTransitionsIntoTraining
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of times aPortStatus has transitioned into the "training" state from another state.

13.2.4.5.2 Port actions

acPortAdministrativeControl
ACTION
APPROPRIATE SYNTAX: Same as aPortAdministrativeState
BEHAVIOUR DEFINED AS:

Port enable/disable control. This action provides a means to alter aPortAdministrativeState. Enabling a disabled port will cause training to be reinitiated. Setting this to "disabled" will assert an acLinkAdministrativeControl (disable) to links connected to this port. Setting this to "enabled" will assert an acLinkAdministrativeControl (activate) to the primary link and an acLinkAdministrativeControl (standby) to the secondary link, if present.

13.2.4.6 Link managed object class

The link managed object class exists for all ports. When implemented, cascade ports shall have one or two links. Local ports shall have one.

13.2.4.6.1 Link attributes

aLinkAbandonments
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of number of times the link was abandoned because of a detected link failure or a retraining error in an active link at the cascade port. Implementations may elect not to abandon the link if the link is immediately able to be successfully retrained. Do not increment the counter if an active link is disabled or transitioned to Standby because of an acLinkAdministrativeControl action. The value of this counter is always zero for local ports.

aLinkID
ATTRIBUTE
APPROPRIATE SYNTAX: INTEGER
BEHAVIOUR DEFINED AS:

It is an integer value unique within the port. This value may be 1 or 2 for links involved in a redundant pair. Otherwise, it shall be 1.

aLinkSignalsReceived
ATTRIBUTE
APPROPRIATE SYNTAX: BOOLEAN
BEHAVIOUR DEFINED AS:

Indicates whether the link is receiving control signals appropriate to the current meta-state of the link. This attribute is set to:

a) TRUE on a local port if the link meta-state is Active or Training. TRUE on a cascade port if the link meta-state is Active or Standby (if applicable) or if Training_Down is being received in response to pulsing Training_Up.

b) FALSE if the link is in the Disabled meta-state, or if the link is for the cascade port and Training_Down is not being received in response to Training_Up. This attribute is also FALSE for local ports with an aPortStatus value of Inactive.

aLinkStatus
ATTRIBUTE
APPROPRIATE SYNTAX: ENUMERATED VALUE LIST for the following ports:

	cascade port		local port	
	1-uplink	2-uplinks		
active	x	x	x	- the link is active
disabled	x	x	x	- the link has been disabled
inactive		x		- the link is inactive
standby		x		- the link is in standby mode
training	x	x	x	- the link is in training
waitAndTest		x		- the link has failed and is waiting to be retrained

BEHAVIOUR DEFINED AS:

This attribute is defined for the various links connected to the repeater. It reflects the current status of the link. An uplink in a 2-uplink repeater is considered to be enabled if its meta-state is not disabled. The enable/disable function as reported by this attribute for 2-uplink repeaters is preserved across repeater reset, including loss of power.

aLinkType
ATTRIBUTE
APPROPRIATE SYNTAX: ENUMERATED VALUE LIST:

primary	-	this is the primary link
secondary	-	this is the secondary (backup) link

BEHAVIOUR DEFINED AS:

This designates which link is preferred and which is the backup link in network topology connections. This attribute is always “primary” except in a cascade port equipped with redundant uplinks.

aMediaType
ATTRIBUTE
APPROPRIATE SYNTAX: ENUMERATED VALUE LIST:

other	-	undefined
unknown	-	true state not known
pmdMissing	-	PMD device not attached
utp4	-	4-pair unshielded twisted pair
stp2	-	2-pair shielded twisted pair
fibre	-	ISO/IEC 8802-12 fibre-optic cabling

BEHAVIOUR DEFINED AS:

This attribute is defined for all links connected to the repeater. aMediaType indicates the type of media currently in use. The value of aMediaType may be “unknown” if the implementation is not capable of identifying the PMD media type, or whether or not the PMD is even present. An implementation is capable of returning a value of ‘pmdMissing’ only if it possesses a means to distinguish between a missing PMD and a PMD which is not driving the PMD Config[3:0] pins.

13.2.4.6.2 Link actions

acLinkAdministrativeControl

ACTION

APPROPRIATE SYNTAX: ENUMERATED VALUE LIST:

activate	- assert Control (activate)
standby	- assert Control (standby)
disable	- assert Control (disable)

BEHAVIOUR DEFINED AS:

This action provides a control capability for all types of repeater links. It will cause a change in the link meta-state. If acLinkAdministrativeControl (disable) is asserted, it will stop any receive and transmit operations over the link. Assertion of acLinkAdministrativeControl (activate) for any link or acLinkAdministrativeControl (standby) for redundant-link cascade ports is required to reenter training. (See acPortAdministrativeControl for a description of the relationship between acLinkAdministrativeControl and acPortAdministrativeControl.)

13.2.4.6.3 Link notifications

nRedundantLinkFailure

NOTIFICATION

APPROPRIATE SYNTAX: Attribute value, same as aLinkStatus

BEHAVIOUR DEFINED AS:

This notification is sent on the occurrence of a transition from Active or Standby to Wait_and_Test. The nRedundantLinkFailure notification is not sent when powering up a repeater. The value of the notification is the updated value of the aLinkStatus attribute. Implementations may elect not to send the notification if the link is immediately able to be successfully retrained.

13.2.5 End node management facilities

This subclause defines the management of demand-priority end nodes by defining associated managed objects. This management encompasses two distinct aspects of management.

The first aspect provides the means to monitor and control the functions of the end node MAC. These functions include, but are not limited to, identifying the end node MAC, testing and initializing the end node MAC, and initiating training on the end node MAC.

The second aspect provides the means to monitor traffic to or from the end node MAC. This is done by gathering statistics on packets that enter or exit the end node MAC.

13.2.5.1 End node functions to support management

To implement the layer management capabilities, this subclause refers to variables and states described in the MAC protocol of this standard. At the end of each frame received, the MAC shall invoke the UpdateLmeMacRxCounters procedure to update the layer management packet receive counters. The MAC shall invoke the UpdateLmeMacTxCounters procedure to update the packet transmit counters. Other layer management attributes are handled as specified in the attribute behaviour text.

13.2.5.1.1 Common constants and types

The following are the common constants and types required for the end node MAC layer management procedures:

```
const
        addressSize = 6;
        nullAddress = array [ 1 .. addressSize ] of Octet value 0;
        maxOctetCount8023 = 1518;
        maxOctetCount8025 = 4520;
type
        Octet = (0 .. 255 );
        Counter32 = ( 0 .. (2**32-1) );
        Counter64 = ( 0 .. (2**64-1) );
        AddressValue = array [ 1 .. addressSize ] of Octet;
        MacPktStatusType = ( nullAddressedFrame, valid, oversizeFrame, IPM, error );
        PriorityType = ( normal, high );
```

13.2.5.1.2 Variables

Following are the variables used by the end node MAC layer management procedures:

```
var
        aBroadcastFramesReceived: Counter32;
        aBroadcastFramesTransmitted: Counter32;
        aDataErrorFramesReceived: Counter32;
        aFramesTransmitted: Counter32;
        aHighPriorityFramesReceived: Counter32;
        aHighPriorityFramesTransmitted: Counter32;
        aHighPriorityOctetsReceived: Counter64;
        aHighPriorityOctetsTransmitted: Counter64;
        aIPMFramesReceived: Counter32;
        aMulticastFramesReceived: Counter32;
        aMulticastFramesTransmitted: Counter32;
        aNormalPriorityFramesReceived: Counter32;
        aNormalPriorityOctetsReceived: Counter64;
        aNullAddressedFramesReceived: Counter32;
        aOctetsTransmitted: Counter64;
        aOversizeFramesReceived: Counter32;
        aReadableFramesReceived: Counter32;
        aReadableOctetsReceived: Counter64;
```

13.2.5.1.3 Procedures

```
Procedure UpdateLmeMacRxCounters.
        (octetCount             : integer;
        destinationAddress      : AddressValue;
        priority                : PriorityType;
        status                  : MacPktStatusType );
Begin
    Case priority of

        normal :
          Begin
```

```
              Increment ( aNormalPriorityFramesReceived) ;
              aNormalPriorityOctetsReceived := aNormalPriorityOctetsReceived + octetCount;
           End

         high :
           Begin
              Increment ( aHighPriorityFramesReceived) ;
              aHighPriorityOctetsReceived := aHighPriorityOctetsReceived + octetCount;
           End
     End;  **{Case priority }**

     If  ( ( status =  nullAddressedFrame ) And ( octetCount >= 6 ) )
       Begin
         Increment ( aNullAddressedFramesReceived );
       End
    Else **{Increment the oversize counter if the frame is larger than the maximum allowed
        size for the current framing type.}**
       If ( (octetCount > maxOctetCount8025) Or
              ( (octetCount > maxOctetCount8023) And
              ( (aMACStatus <> open) Or
              ( (aMACStatus = open) And
              ( (aLastTrainingConfig[1] Div 8) Mod 4 = 0) ) )
              )
         ) Then
        Begin
         Increment ( aOversizeFramesReceived);
          End
     Else
        Begin
         Case status of

         valid:
              Begin
                   Increment ( aReadableFramesReceived);
                   aReadableOctetsReceived := aReadableOctetsReceived + octetCount;
                   If ( destinationAddress is a group address ) Then
                        If ( destinationAddress = broadcastAddress ) Then
                             Increment ( aBroadcastFramesReceived );
                        Else
                             Increment ( aMulticastFramesReceived );
               End

         IPM :
              Begin
                   Increment ( aIPMFramesReceived );
              End

         error:
              Begin
                   Increment ( aDataErrorFramesReceived );
              End
         End; **{Case PktStatus }**
     End  **{ Else }**
End  **{ Procedure UpdateLmeMacRxCounters }**
```

```
Procedure UpdateLmeMacTxCounters.
                                (octetCount            : integer;
                                 destinationAddress    : AddressValue;
                                 priority              : TxPriorityType );
Begin
    If ( priority = high ) Then
        Begin
            Increment ( aHighPriorityFramesTransmitted ) ;
            aHighPriorityOctetsTransmitted := aHighPriorityOctetsTransmitted + octetCount;
        End;
    Increment ( aFramesTransmitted) ;
    aOctetsTransmitted := aOctetsTransmitted + octetCount ;
    If ( destinationAddress is a group address) Then
        If ( destinationAddress = broadcastAddress ) Then
            Increment ( aBroadcastFramesTransmitted);
        Else
            Increment ( aMulticastFramesTransmitted );

End { Procedure UpdateLmeMacTxCounters }
```

13.2.5.2 End node managed object class

13.2.5.2.1 End node attributes

aBroadcastFramesReceived
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of good broadcast frames received. These frames are also counted by aReadableFramesReceived. This counter is updated as specified in 13.2.5.1.3.

aBroadcastFramesTransmitted
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

The count of frames transmitted to the broadcast destination address. Frames transmitted to multicast addresses are not broadcast frames and are excluded. This counter is updated as specified in 13.2.5.1.3.

aDataErrorFramesReceived
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter

BEHAVIOUR DEFINED AS:

Count of frames received with any of the following errors: bad FCS (with no IPM), PMI errors (excluding frames with an IPM as the only PMI error), undersize, bad start of frame delimiter, or bad end of packet marker. Does not include frames counted by aIPMFramesReceived, aNullAddressedFramesReceived, or aOversizeFramesReceived. This counter is updated as specified in 13.2.5.1.3.

aDesiredFramingType
ATTRIBUTE

APPROPRIATE SYNTAX: ENUMERATED VALUE LIST:

frameType8023	- ISO/IEC 8802-3 framing is desired
frameType8025	- ISO/IEC 8802-5 framing is desired
frameTypeEither	- Either ISO/IEC 8802-3 or ISO/IEC 8802-5 framing is acceptable

BEHAVIOUR DEFINED AS:

The type of framing that will be used by the end node during the next end node MAC initialization or acOpen action.

aDesiredPromiscuousStatus
ATTRIBUTE
APPROPRIATE SYNTAX: BOOLEAN
BEHAVIOUR DEFINED AS:

This flag is used to select promiscuous mode in the ‘PP’ bits of the training configuration field on the next training packet issued by the end node. A value of TRUE will cause promiscuous mode to be requested during the next acInitializeMAC or acOpen. Otherwise, privacy (single address only) mode will be requested. Whether or not promiscuous mode is actually granted by the repeater can be verified by examining the state of the ‘PP’ bits in aLastTrainingConfig.

When in promiscuous mode, the MAC RecognizeAddress function will always return true and the frame counters in this subclause will apply to all frames, regardless of their destination address.

aFramesTransmitted
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of all frames transmitted by this MAC. This counter is updated as specified in 13.2.5.1.3.

aFramingCapability
ATTRIBUTE
APPROPRIATE SYNTAX: ENUMERATED VALUE LIST:

frameType8023	- ISO/IEC 8802-3 framing is supported
frameType8025	- ISO/IEC 8802-5 framing is supported
frameTypeEither	- Both ISO/IEC 8802-3 and ISO/IEC 8802-5 framing are supported

BEHAVIOUR DEFINED AS:

This attribute indicates the type of framing the end node is capable of supporting.

aFunctionalAddresses
ATTRIBUTE
APPROPRIATE SYNTAX: OCTET STRING
BEHAVIOUR DEFINED AS:

This is the Token Ring Functional Address mask. Functional addresses are only used in ISO/IEC 8802-5 mode. This object exists to enhance compatibility with existing Token Ring software.

aHighPriorityFramesReceived
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of high-priority frames received. Includes both good and bad high-priority frames, as well as high-priority training frames. Does not include normal-priority frames that were priority promoted. This counter is updated as specified in 13.2.5.1.3.

aHighPriorityFramesTransmitted
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of high-priority frames that were transmitted successfully. This counter is updated as specified in 13.2.5.1.3.

aHighPriorityOctetsReceived
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of octets in aHighPriorityFramesReceived. This counter is updated as specified in 13.2.5.1.3.

aHighPriorityOctetsTransmitted
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of octets in aHighPriorityFramesTransmitted. This counter is updated as specified in 13.2.5.1.3.

aIPMFramesReceived
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of frames with an IPM and no PMI errors received. A repeater will write an IPM to the end of a frame containing errors as it is forwarded through the repeater to the other ports. This counter is updated as specified in 13.2.5.1.3.

aLastTrainingConfig
ATTRIBUTE
APPROPRIATE SYNTAX: OCTET STRING
BEHAVIOUR DEFINED AS:

This 16-bit field contains the allowed configuration field from the most recent error-free training response frame received in response to training initiated by the end node. The format of this field is shown in Figure 10-17. Note that aMACStatus can be examined to see if any error-free training frames have been received.

aMACID
ATTRIBUTE
APPROPRIATE SYNTAX: INTEGER
BEHAVIOUR DEFINED AS:

The value of aMACID is assigned so as to uniquely identify a MAC among the subordinate managed objects of the system. (The system is defined in ISO/IEC 10165-2:1992.)

aMACStatus

ATTRIBUTE

APPROPRIATE SYNTAX: ENUMERATED VALUE LIST:

opened	-	Training has completed successfully
closed	-	MAC has been disabled by an acClose action
opening	-	MAC is in training (Training_Down has been received from repeater)
openFailure	-	Passed 24 error-free packets, but there is a problem, noted in the training configuration bits (aLastTrainingConfig)
linkFailure	-	Training_Down not received, or could not pass 24 error-free packets

BEHAVIOUR DEFINED AS:

This attribute is the same variable as “MAC_STATUS” in Clause 11. Whenever network management issues an acClose action, the MAC will assert TxDisable, and aMACStatus will be “closed.”

When an acOpen or acInitializeMAC is performed, the MAC will send Training_Up to the repeater and initially go to the “linkFailure” state. When the repeater sends back Training_Down, aMACStatus will change to the “opening” state and training packets will be transferred.

After all of the training packets have been passed, aMACStatus will change to: “linkFailure” if 24 consecutive error-free packets were not passed, “opened” if 24 consecutive error-free packets were passed and the training configuration bits were OK, or “openFailure” if there were 24 consecutive error-free packets, but there was a problem with the training configuration bits.

When in the “openFailure” state, the aLastTrainingConfig attribute will contain the configuration bits from the last training response packet that can be examined to determine the exact reason for the training configuration failure.

If training did not succeed (aMACStatus = “linkFailure” or “openFailure”), the entire process will be restarted after the MAC_Retraining_Delay_Timer expires (see 11.5.2.2).

aMACVersion

ATTRIBUTE

APPROPRIATE SYNTAX: BIT STRING

BEHAVIOUR DEFINED AS:

A read-only 3-bit field that reports the version (the “vvv” bits) that is the highest version number supported by the MAC.

aMediaType

ATTRIBUTE

APPROPRIATE SYNTAX: ENUMERATED VALUE LIST:

other	-	undefined
unknown	-	true state not known
pmdMissing	-	PMD device not attached
utp4	-	4-pair unshielded twisted pair
stp2	-	2-pair shielded twisted pair
fibre	-	ISO/IEC 8802-12 fibre-optic cabling

BEHAVIOUR DEFINED AS:

aMediaType indicates the type of media currently in use. aMediaType may be “unknown” if the implementation is not capable of identifying the PMD media type, or whether or not the PMD is even present.

aMulticastFramesReceived
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of good multicast frames received. These frames are also counted by aReadableFramesReceived, but not by aBroadcastFramesReceived. This counter is updated as specified in 13.2.5.1.3.

aMulticastFramesTransmitted
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

The count of frames transmitted to a group destination address other than the broadcast address. This counter is updated as specified in 13.2.5.1.3.

aMulticastReceiveStatus
ATTRIBUTE
APPROPRIATE SYNTAX: BOOLEAN
BEHAVIOUR DEFINED AS:

TRUE if multicast receive is enabled, and FALSE otherwise. Setting this to TRUE provides a means to cause the MAC sublayer to return to the normal operation of multicast frame reception. Setting this to FALSE will cause the MAC Recognize_Address function to return false for multicast frames (unless in promiscuous mode), which will inhibit the reception of further multicast frames by the MAC sublayer.

aNormalPriorityFramesReceived
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of normal-priority frames received. Includes both good and bad normal-priority frames, as well as normal-priority training frames. Also includes normal-priority frames that were priority promoted. This counter is updated as specified in 13.2.5.1.3.

aNormalPriorityOctetsReceived
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of octets in aNormalPriorityFramesReceived. This counter is updated as specified in 13.2.5.1.3.

aNullAddressedFramesReceived
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of frames received with a destination MAC address of all zeros. Both void and training frames are included in this counter. This counter is updated as specified in 13.2.5.1.3.

aOctetsTransmitted
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of octets in aFramesTransmitted. This counter is updated as specified in 13.2.5.1.3.

aOversizeFramesReceived
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of oversize frames received. The frame size that causes this counter to increment is dependent on the current framing mode. This counter is updated as specified in 13.2.5.1.3.

aReadableFramesReceived
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of good frames received. A good frame is defined as a non-null-addressed frame that is not counted by any of the following error counters: aIPMFramesReceived, aOversizeFramesReceived, aNullAddressedFramesReceived, or aDataErrorFramesReceived. This counter is updated as specified in 13.2.5.1.3.

aReadableOctetsReceived
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of octets in frames counted by aReadableFramesReceived. This counter is updated as specified in 13.2.5.1.3.

aReadMulticastList
ATTRIBUTE
APPROPRIATE SYNTAX: SEQUENCE OF MACAddress
BEHAVIOUR DEFINED AS:

Returns the current multicast address list.

aReadWriteMACAddress
ATTRIBUTE
APPROPRIATE SYNTAX: MACAddress
BEHAVIOUR DEFINED AS:

The MAC address of the end node.

aTransitionsIntoTraining
ATTRIBUTE
APPROPRIATE SYNTAX: Generalized nonresettable counter
BEHAVIOUR DEFINED AS:

Count of times the end node has entered the training state. This counter is incremented when aMACStatus transitions to “linkFailure” from any state other than “opening” or “openFailure.”

13.2.5.2.2 EndNode actions

acAddGroupAddress
ACTION
APPROPRIATE SYNTAX: MAC Address
BEHAVIOUR DEFINED AS:

This action will add the specified group address to the multicast address list used by the RecognizeAddress MAC function.

acClose
ACTION
APPROPRIATE SYNTAX: None required
BEHAVIOUR DEFINED AS:

This action will stop transmit and receive actions. It sets the MAC_STATUS variable in Clause 11 to “closed.” An acOpen action is required to reenter training.

acDeleteGroupAddress
ACTION
APPROPRIATE SYNTAX: MAC Address
BEHAVIOUR DEFINED AS:

This action will delete the specified group address from the multicast address list used by the RecognizeAddress MAC function. No action will be performed if the given address is not in the multicast address list.

acExecuteSelftest
ACTION
APPROPRIATE SYNTAX: None required
BEHAVIOUR DEFINED AS:

Execute a self-test and report the results (success or failure). The actual mechanism employed to carry out the self-test is not defined in this standard.

acInitializeMAC
ACTION
APPROPRIATE SYNTAX: None required
BEHAVIOUR DEFINED AS:

This action will call the Initialize_MAC procedure in the MAC pseudo code. The MAC will then initiate training.

acOpen
ACTION
APPROPRIATE SYNTAX: None required
BEHAVIOUR DEFINED AS:

If the current value of aMACStatus is “closed,” this action will initiate training; otherwise, no action will be taken.

13.2.5.2.3 EndNode notifications

No EndNode notifications are currently defined.

13.2.5.3 ResourceTypeID managed object class

Implementation of this managed object in accordance with the definition contained in IEEE Std 802.1F-1993 is a conformance requirement of this standard. A single instance of the Resource Type ID managed object exists within the repeater managed object class. The managed object itself is contained in IEEE Std 802.1F-1993; therefore, only the name binding appears in this standard.

14. Physical Medium Independent (PMI) specifications

14.1 Scope

This clause specifies the functions of the PMI sublayer, based on the model of one PMI per PMD (see Figure 14-1). The actual method used to accomplish the requirements of this clause is an implementation decision; no particular structure is implied.

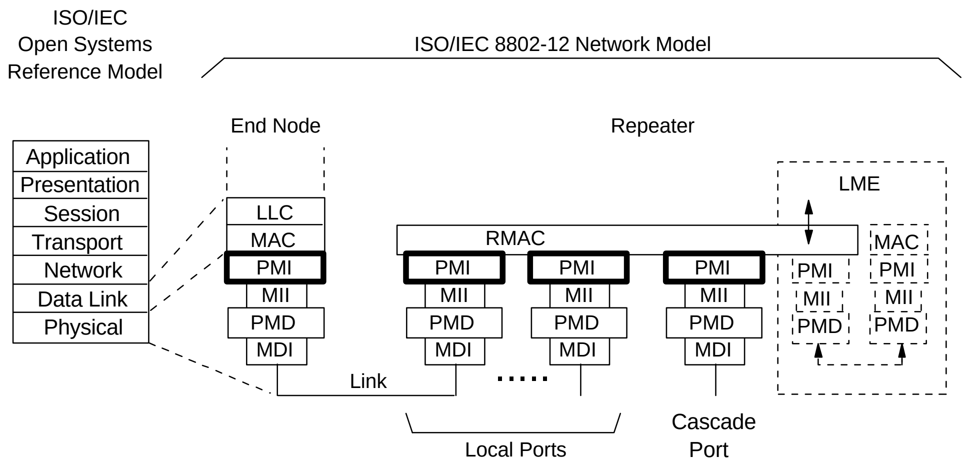


Figure 14-1—Relationship to the LAN model

14.2 Overview

The PMI is the portion of the Physical Layer that is independent of the link medium. It has the following characteristics:

a) Supports logical interfaces to the MAC or the RMAC and to the PMD sublayers shown in Figure 14-2.
b) Supports control signaling between the MAC or the RMAC, and the PMD by providing PMI control functions.
c) Supports Packet Data Transmission by providing the TData[3:0] Channel Stream Structure at the MII using PMI transmit functions.
d) Supports Packet Data Reception on RData[3:0] using PMI receive functions.
e) Generates and provides a clocking function (TxClk) for data transmission and accepts a clocking function (RxClk) during data reception.

The PMI must support the two logical interfaces shown in Figure 14-2. The interfaces between an end node's MAC and its PMI and between a repeater's RMAC and its PMIs are defined by the service primitives of Clause 8. The interface between the PMI and the PMD, the MII, is defined by Clause 15. The PMI responds to request primitives generated by the MAC or RMAC sublayer, and to indication primitives generated by the PMD sublayer.

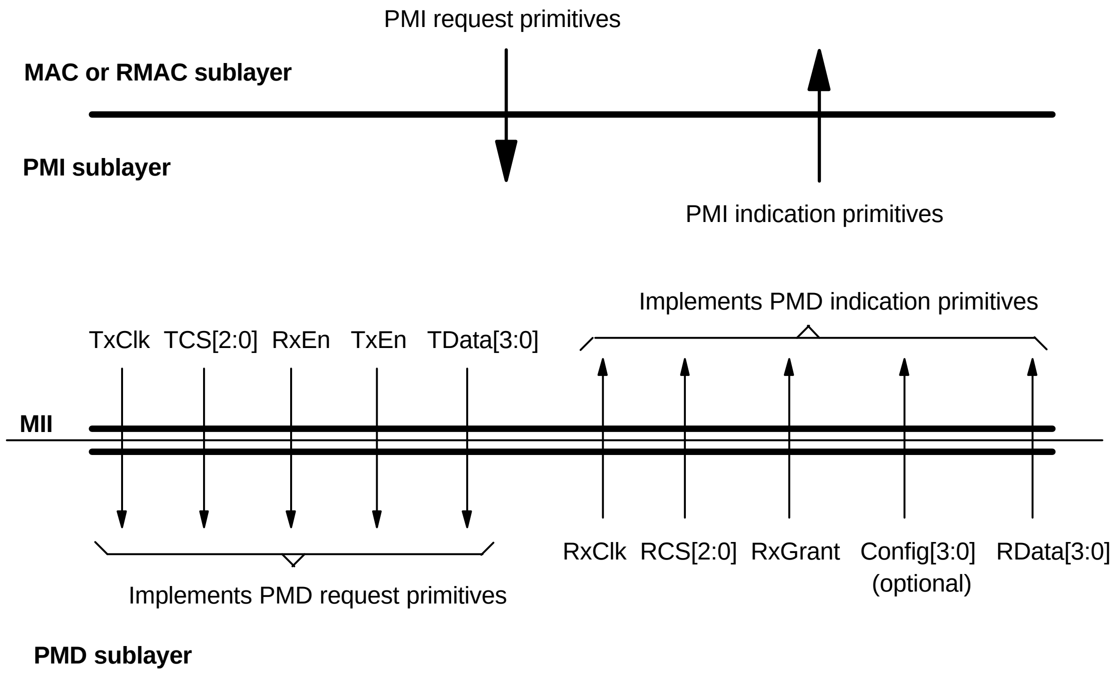


Figure 14-2—PMI sublayer interfaces

14.2.1 Control functions overview

The PMI shall pass the Transmit Control State from the MAC or the RMAC to the PMD. In addition, the PMI shall pass the Receive Control State and RxGrant from the PMD to the MAC or the RMAC. These control functions are defined in 14.3.

14.2.2 Data functions overview

The PMI enables or disables TxEn at the MII circuits in response to PMI_UNITDATA.request primitives generated by the MAC or the RMAC. The PMI also enables RxEn at the MII in response to PMI_RECEIVE_ENABLE.request primitives from the MAC or the RMAC. Normally, the PMI will deassert RxEn because it has detected the end of a packet or a transition from RCS = 101 to RCS = 000. However, the MAC or the RMAC may override the PMIs control of RxEn by issuing a PMI_RECEIVE_ ENABLE.request primitive in certain circumstances (e.g., when a packet length is longer than the MAC or RMAC's maximum length packet counter).

14.2.2.1 Data transmit overview

In the data transmit mode, the MAC or the RMAC issues a PMI_UNITDATA.request primitive to the PMI indicating that a frame is ready to be sent. The PMI transfers octets from the MAC or the RMAC and delivers serial bit streams to the MII. It notifies the MAC or the RMAC when the transmission is complete. Figure 14-3 provides a block diagram of the PMI transmission functions.

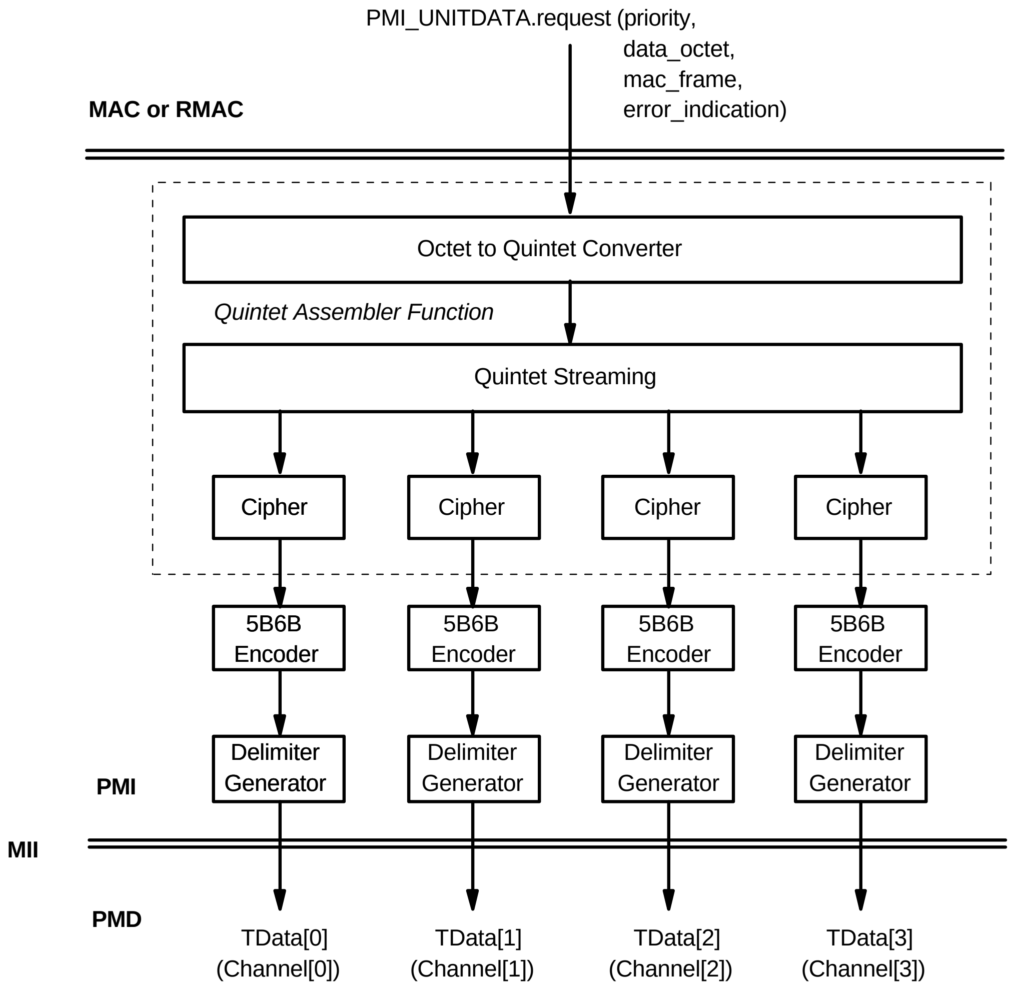


Figure 14-3—Data transmit overview

Data octets from the MAC or RMAC frame are first converted into data quintets in the Octet-to-Quintet converter and are then:

a) Separated into four separate data streams by rotating the assignment of successive data quintets to the next Channel[i], where i cycles through the values [0, 1, 2, 3].
b) Passed through a stream cipher to form ciphered quintets.
c) Encoded into a sextet in the 5B6B encoder.

The Delimiter Generator generates the appropriate start of stream and end of stream sequences that are added to identify both the beginning and the end of the encoded data stream. When required, fill patterns are also appended to the end of stream delimiters.

The PMI ensures that each output stream is the same length and that it is synchronized so that the first bit of each stream reaches the MII simultaneously. An example of the resulting bit stream is shown in Figure 14-4. Definitions of the transmit functions are given in 14.4.2. The format of each data stream is given in 14.2.3.

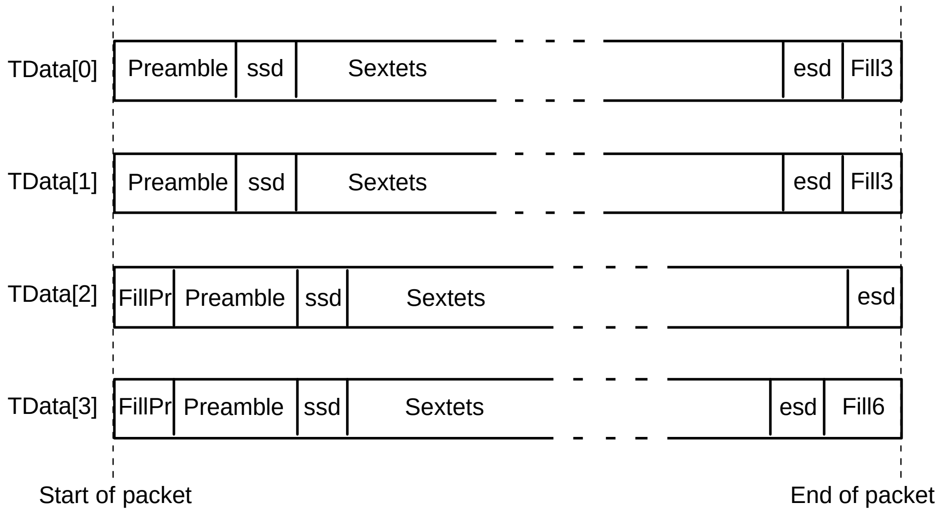


Figure 14-4—MII, TData[3:0] stream format example

14.2.2.2 Data receive overview

The packet reception is the reverse of the transmission process as shown in Figure 14-5. Definitions of the receive functions are given in 14.4.4.

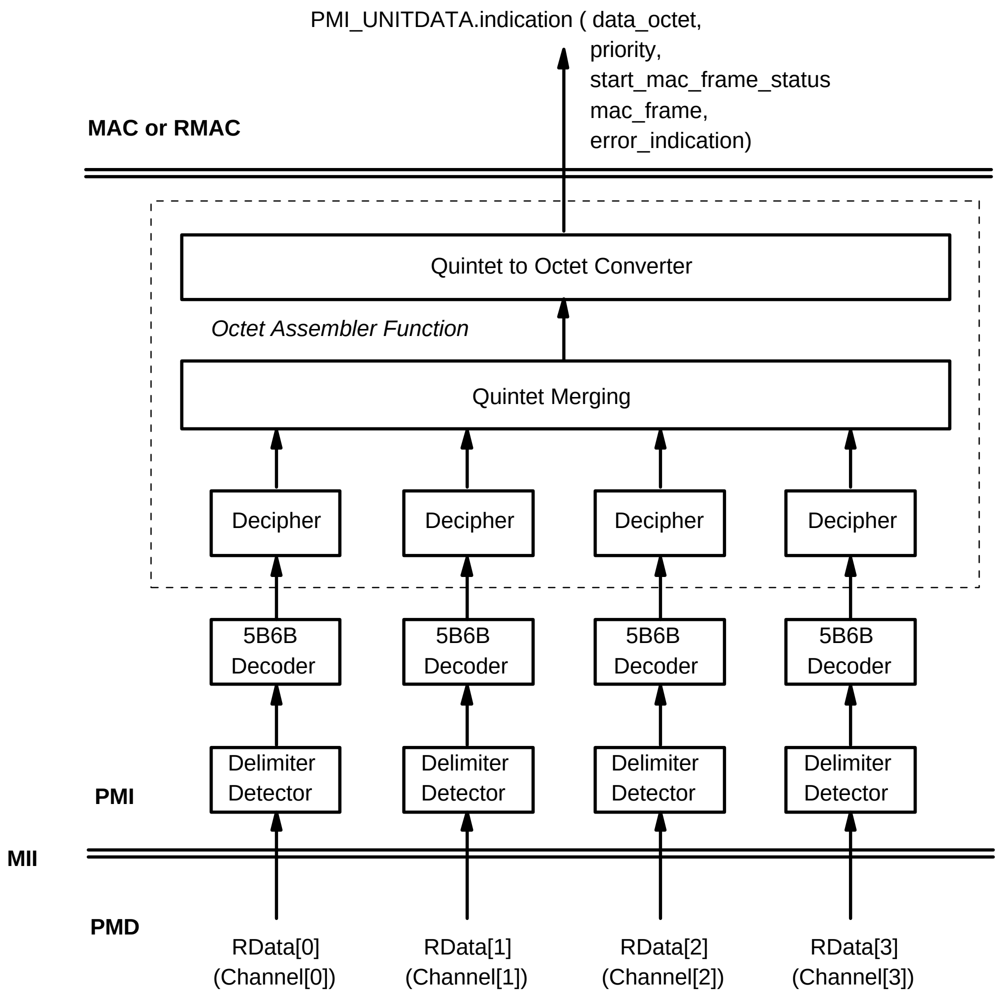


Figure 14-5—Data receive functions overview

14.3 Control functions

14.3.1 TCS transfer function

The PMI shall map the transmit_control_state parameter to a binary value that is sent on the TCS[2:0] circuits for every PMI_CONTROL.request received from the MAC or the RMAC. Mapping of PMI_ CONTROL.request parameters to TCS codes is defined in Table 14-1. The PMI shall set TCS[2:0] within 4 TxClk cycles of the PMI_CONTROL.request being issued by the MAC or the RMAC.

Table 14-1—Transmit control names ⇒ Transmit Control State (TCS) encoding

End node transmit control name	Repeater (Cascade port) transmit control name	Repeater (Local port) transmit control name	Transmit Control State (TCS) code
TxDisable	TxDisable	TxDisable	000
Idle_Up	Idle_Up	Idle_Down	001
Request_Normal	Request_Normal	Incoming	010
Request_High	Request_High	Enable_High_Only[a]	011
Training_Up	Training_Up	Training_Down	100
(reserved)	(reserved)	(reserved)	101
(reserved)	(reserved)	(reserved)	110
(invalid)	Return[b]	Grant	111

[a] Valid if the link is connected to a lower-level repeater, invalid if the link is connected to an end node.
[b] Return is always a secondary control state.

14.3.2 RCS transfer function

The MII RCS[2:0] circuits shall be sampled every cycle of TxClk by the PMI. The PMI shall map this sampled binary value to the value of the receiver_control_state parameter according to Table 14-2 and shall then generate a PMI_CONTROL.indication to the MAC or RMAC within four TxClk cycles of the RCS[2:0] circuits being updated by the PMD.

Table 14-2—Receiver Control State (RCS) code ⇒ Receiver control name decoding

Receiver Control State (RCS) Code	End node control name	Repeater (Cascade port) control name	Repeater (Local port) control name
000	Mode Transition[a]	Mode Transition[a]	Mode Transition[a]
001	Idle_Down	Idle_Down	Idle_Up
010	Incoming	Incoming	Request_Normal
011[b]	—	Enable_High_Only	Request_High
100	Training_Down	Training_Down	Training_Up
101	Data on link	Data on link	Data on link
110	(reserved)	(reserved)	(reserved)
111	Link Warning [c]	Link Warning	Link Warning

[a] RCS code value of “000” indicates that the PMD has not yet determined which control signals are present on the link. The PMD initially sets this value when either TxEn or RxEn is deasserted.
[b] Invalid if received by an end node.
[c] Link warning conditions are defined in the clauses for each type PMD/Link.

14.3.3 RxGrant transfer function

The RxGrant circuit of the MII shall be sampled every cycle of TxClk by the PMI. The PMI shall indicate the grant_state to the MAC or the RMAC by generating a PMI_CONTROL.indication with a grant_state parameter equal to the binary value of RxGrant within 4 TxClk cycles of the RxGrant circuit being updated by the PMD.

14.3.4 TxEn control function

The PMI shall assert the MII TxEn circuit within 3 TxClk cycles after a PMI_UNITDATA.request with a mac_frame parameter value of “begin” is generated by the MAC or the RMAC. The PMI shall deassert the MII TxEn circuit on the TxClk edge following the transfer of the last bit of packet data.

14.3.5 RxEn control function

The PMI shall assert or deassert the MII RxEn circuit within 3 TxClk cycles after a PMI_RECEIVE_ENABLE.request with a receive_state parameter of “assert” or “deassert,” respectively being generated by the MAC or the RMAC. The PMI shall deassert RxEn within 5 cycles of TxClk of RCS[2:0] transitioning from 101 to 000 at the MII.

14.3.6 Configuration function

The PMI shall sample the MII Config[3:0] circuits, if implemented, to determine the binary value of the configuration code. The PMI shall indicate the configuration code to the MAC or the RMAC by generating PMI_CONFIGURATION.indication primitives with the value of the configuration_indication parameter equal to the binary value of Config[3:0].

14.4 Data functions

14.4.1 Clocking functions

14.4.1.1 TxClk function

TxClk shall be provided by the PMI. The frequency of TxClk shall be 30 MHz. The frequency of TxClk shall not deviate from 30 MHz by more than 100 ppm. The duty cycle of TxClk shall be 40–60%. TxClk shall be the reference clock source of the PMI. It shall be continuously available to the PMI and at the MII TxClk circuit.

14.4.1.2 RxClk

During packet reception, RxClk is recovered by the PMD and is made available to the PMI by the PMD on the RxClk circuit of the MII.

14.4.2 Transmit functions

14.4.2.1 Quintet assembler function

The order of both the MAC frame serial bit stream and the MAC frame serial octet stream are uniquely defined by the MAC frame format. Octets are transferred to the Quintet Assembler in the exact order of the MAC frame octet stream.

The PMI shall start forming quintets upon reception of a PMI_UNITDATA.request with a mac_frame parameter equal to “begin.” The PMI shall form quintets using the bit stream defined by the value of the PMI_UNITDATA.request parameter data_octet. The leftmost bit of each quintet shall be the bit that occurs soonest in the MAC frame serial bit stream. The subsequent bits of a quintet shall maintain the bit ordering of the originating octet bit stream. The last data_octet shall be identified by the PMI as the value of the data_octet parameter associated with the PMI_UNITDATA.request having the mac_frame parameter “end.”

The octet stream from the MAC or the RMAC consists of an integer number of octets and may give rise to an incomplete final data quintet. If the final bit of the last octet of the octet stream does not complete a final quintet, up to four extension bits with arbitrary binary values shall be used to complete the final quintet.

14.4.2.1.1 Quintet streaming

PMI quintets are assigned to the four Channels[i] in a cyclic manner. The first complete quintet formed from the octet having a mac_frame value equal to “begin” shall be the first quintet of the PMI quintet stream. The first quintet of the PMI quintet stream shall be the first Channel[0] quintet, the second quintet of the PMI stream shall be the first Channel[1] quintet, the third quintet of the PMI stream shall be the first Channel[2] quintet, and the fourth quintet of the PMI stream shall be the first Channel[3] quintet. This process shall be continued until the last quintet has been assigned to a channel (see Figure 14-6).

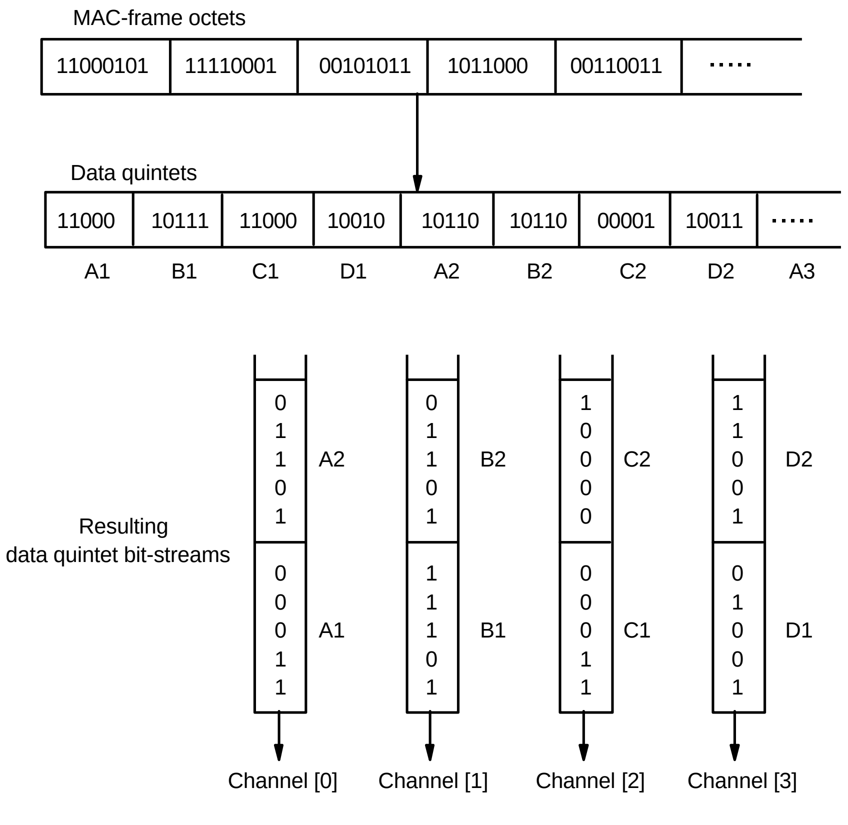


Figure 14-6—MAC frame octet to quintet conversion

14.4.2.1.2 Channel[i] quintet ciphering

Each Channel[i] quintet stream shall be ciphered separately. The resulting ciphered serial bit stream shall be identical to the bit stream produced by ciphering the Channel[i] serial bit stream with the cipher generating polynomial:

$$G(x) = x^{11} + x^9 + 1 \qquad (6)$$

The seed values of the ciphers shall be as defined in Table 14-3 where the leftmost bit of each cipher seed is the bit corresponding to x^{10} and the rightmost is the bit corresponding to x^0.

Table 14-3—Cipher seeds

Channel	Seed x^{10} .... x^0
0	10000000000
1	01011111100
2	11111100000
3	01011111000

The ciphering bit that is XORed with the data stream is the coefficient of x^4 as shown in Figure 14-7. Ciphering shall only be applied to the Channel[i] quintet bit streams. Preamble, start of stream delimiters, end of stream delimiters, and fill patterns defined in 14.4.2.3 shall not be ciphered. The seeds for each channel shall be initialized before the first quintet of each packet transmission is ciphered and shall not be reset until the last quintet of each packet transmission has been ciphered. The first bit of the ciphered bit stream is the first bit of the data bit stream XORed with bit 4 of the seed. This ciphering is also referred to as frame synchronous scrambling.

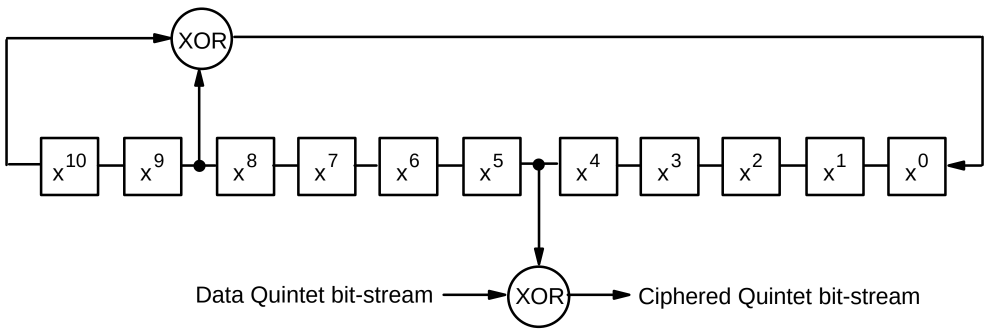


Figure 14-7—Stream cipher schematic

14.4.2.2 Encoder function

The PMI shall have the capability to encode the data to be transmitted on each channel. A bimodal 5B/6B block coding scheme shall be used to map ciphered quintets to sextets. In the first mode (mode 2), the coding scheme uses either sextets of weight 2 (i.e., the sextet contains two 1's and four 0's) or sextets of weight 3 (i.e., the sextet contains three 1's and three 0's). In the second mode (mode 4), the coding scheme uses either sextets of weight 3 or sextets of weight 4 (i.e., the sextet contains four 1's and two 0's). The coding scheme alternates between the two modes such that the total number of 1's in the coded data stream never differs from the total number of 0's by more than two after an integer number of sextets has been transmitted.

Ciphered quintets shall be received on each channel from the Quintet Assembler function, and shall be used to determine the sextets to be passed to the Delimiter Generator function on the same channel, according to Table 14-4.

For each input quintet, Table14-4 lists both the output sextet and the encoder mode that shall be used to encode the next quintet. The mode 2 columns shall be used for the encoding of the first quintet of each Channel[i] stream. The leftmost bit of a tx_quintet occurs soonest in the serial data bit stream on the Channel[i]. The leftmost bit of the corresponding output sextet shall be the first bit of the sextet to be transmitted on TData[i] at the MII circuits (see Figure 14-8).

The Encoder function shall map the ciphered quintets to sextets from the mode 2 columns of Table 14-4 until any weight 2 sextet is used on that channel. Sextets from the mode 4 columns shall then be used until any weight 4 sextet is chosen. The Encoder function shall then revert to using the mode 2 columns and shall continue to alternate between the modes whenever a weight 2 or weight 4 sextet is used on that channel.

The alternation between mode 2 and mode 4 shall occur independently on each of the four channels. The order of sextets output on each channel shall be identical to the order of their corresponding input quintets.

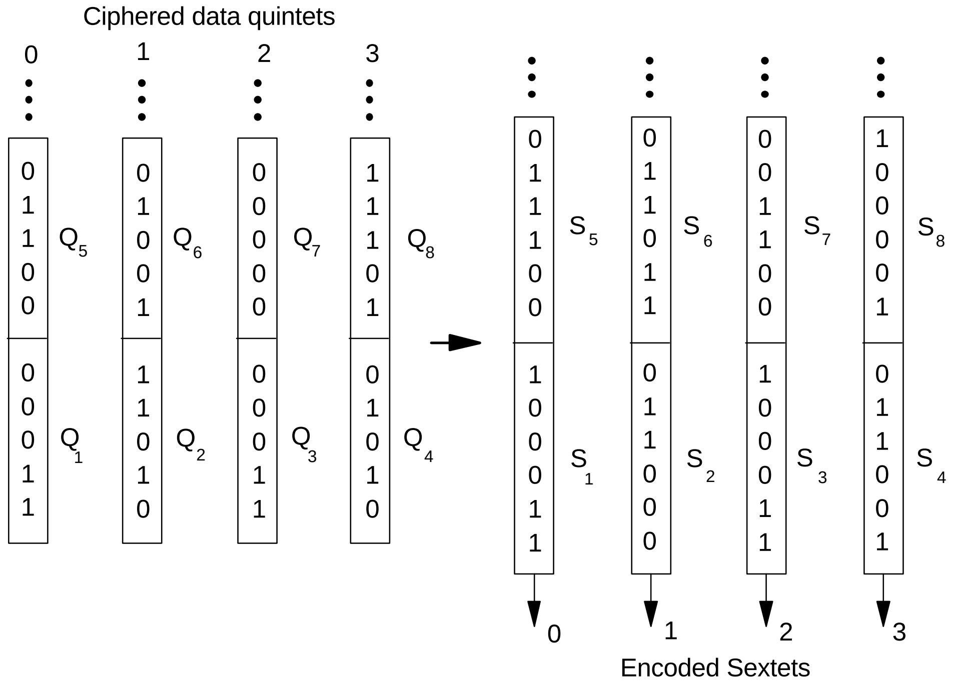


Figure 14-8—Example quintet to sextet conversion showing order of transmission

Table 14-4—State table for the encoder function (with both the output sextet and next encoder mode listed)

	MODE 2		MODE 4	
Input quintet (tx_quintet)	**Output sextet (Mode_2_code)**	**Next mode (Mode_2_next)**	**Output sextet (Mode_4_code)**	**Next mode (Mode_4_next)**
00000	001100	4	110011	2
00001	101100	2	101100	4
00010	100010	4	101110	2
00011	001101	2	001101	4
00100	001010	4	110101	2
00101	010101	2	010101	4
00110	001110	2	001110	4
00111	001011	2	001011	4
01000	000111	2	000111	4
01001	100011	2	100011	4
01010	100110	2	100110	4
01011	000110	4	111001	2
01100	101000	4	010111	2
01101	011010	2	011010	4
01110	100100	4	011011	2
01111	101001	2	101001	4
10000	000101	4	111010	2
10001	100101	2	100101	4
10010	001001	4	110110	2
10011	010110	2	010110	4
10100	111000	2	111000	4
10101	011000	4	100111	2
10110	011001	2	011001	4
10111	100001	4	011110	2
11000	110001	2	110001	4
11001	101010	2	101010	4
11010	010100	4	101011	2
11011	110100	2	110100	4
11100	011100	2	011100	4
11101	010011	2	010011	4
11110	010010	4	101101	2
11111	110010	2	110010	4

14.4.2.3 Delimiter Generator function

The Delimiter Generator function is the final function of the channel and its output shall be to the corresponding MII TData circuits. The PMI shall be capable of generating preamble patterns, start of stream delimiters, end of stream delimiters, and fill bits. Along with data, these shall be transmitted on each of the MII circuits TData[3:0] at a rate of 30 million code bits/s, synchronous with TxClk.

14.4.2.3.1 Start fill pattern

A start fill pattern shall be transmitted simultaneously on each of TData[3:2] whenever the PMI_UNITDATA.request parameter mac_frame has the value of “begin.” The start fill pattern shall be:

FillPr = 101

FillPr shall be transmitted in bit order, starting with the leftmost bit.

14.4.2.3.2 Preamble patterns

A preamble pattern shall be transmitted on each of TData[1:0] simultaneous with the start fill patterns on TData[3:2] whenever the mac_frame parameter of a PMI_UNITDATA.request has the value of “begin.” A preamble pattern shall be transmitted simultaneously on each of TData[3:2] whenever the Delimiter Generator function has completed generation of FillPr on these circuits. The preamble pattern shall be:

preamble = 010101 010101 010101 010101 010101 010101 010101 010101

The preamble pattern shall be transmitted in bit order starting with the leftmost bit.

14.4.2.3.3 Start of stream delimiters

A start of stream delimiter (ssd) shall be transmitted on each of TData[3:0] immediately following the preamble pattern. For a normal-priority packet, the ssd shall be:

ssdn = 111100 000011

The <ssdn> shall be transmitted in bit order, starting with the leftmost bit, immediately following the preamble pattern, whenever the priority parameter of a PMI_UNITDATA.request has the value of “normal.”

For a high-priority packet, the ssd transmitted shall be:

ssdh = 100000 111110

The <ssdh> shall be transmitted in bit order, starting with the leftmost bit, immediately following the preamble pattern, whenever the priority parameter of a PMI_UNITDATA.request has the value of “high.”

14.4.2.3.4 Data transmission

Sextets for each channel are generated by the Encoder function and passed to the Delimiter Generator function. The first sextet received from the Encoder function for each Channel[i] shall be transmitted immediately following the ssd on TData[i]. Subsequent sextets shall be transmitted sequentially and without interruption, maintaining the order in which sextets are received from the Encoder function, until the last sextet has been transmitted. The sextet transmission rate shall be 5 million sextets per second on each of TData[3:0] and the resulting signaling rate shall be 30 million code bits/s on each of TData[3:0].

14.4.2.3.5 End of stream delimiters

An end of stream delimiter (esd) shall be transmitted on each of TData[3:0] immediately following the last sextet on that channel. All end of stream delimiters shall be transmitted in bit order starting with the leftmost bit.

The esd transmitted on each of TData[3:0] depends upon the mode of the Encoder function after the last sextet has been passed to the Delimiter Generator function, and on the value of the error_indication parameter of the PMI_UNITDATA.request primitive. If the error_indication parameter has the value of "IPM," then the esd transmitted on all of TData[3:0] shall be the IPM pattern:

IPM = 110000 011111 110000

The first sextet of IPM shall be 110000, the second shall be 011111, and the third shall be 110000.

If the error_indication parameter has the value of "valid," and when encoding the last quintet on the channel, next mode = 2, then the esd transmitted on Channel[i] (i=0,1,2 or 3) shall be the <esd2> pattern:

esd2 = 111111 000011 000001

The first sextet of <esd2> shall be 111111, the second shall be 000011, and the third shall be 000001.

If the error_indication parameter has the value of "valid," and when encoding the last quintet on the channel, next mode = 4, then the esd transmitted on Channel[i] (i=0,1,2 or 3) shall be the <esd4> pattern:

esd4 = 000000 111100 111110

The first sextet of <esd4> shall be 000000, the second shall be 111100, and the third shall be 111110.

When error_indication = IPM, the IPM esd is transmitted on all four TData[3:0] circuits. However, when error_indication = valid, the esd is chosen from <esd2> or <esd4> independently on each of TData[3:0].

14.4.2.3.6 End fill patterns

If, on completing transmission of the final sextet on TData[0], the final sextet on TData[2] is still being transmitted, a 3-bit fill pattern shall be appended to the esd on each TData[1:0] and, if the transmission of the final sextet on TData[3] has also been completed, a 6-bit fill pattern shall be appended to the esd on TData[3].

Otherwise, a 3-bit fill pattern shall be appended to the esd on each TData[3:2] and, if the transmission of the final sextet on TData[1] has been completed before the final sextet on TData[2], a 6-bit fill pattern shall be appended to the esd on TData[1].

If the esd2 end of stream delimiter has been transmitted on TData[i], then the end fill pattern used shall be:

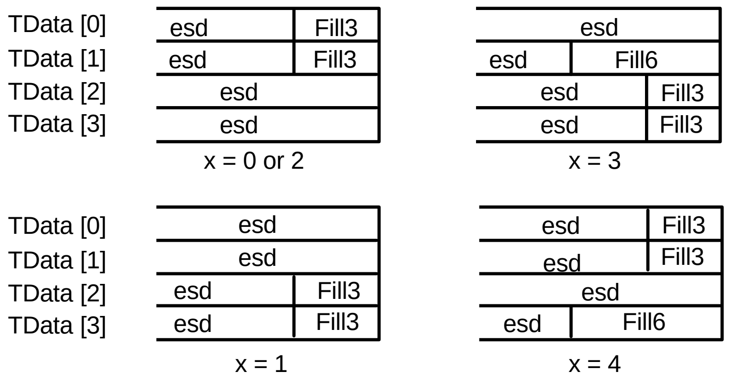
Fill6_2	=	100011	if a 6-bit end fill pattern is required
Fill3_2	=	100	if a 3-bit end fill pattern is required

If the esd4 or IPM end of stream delimiter has been transmitted on TData[i], then the end fill pattern used shall be:

Fill6_4	=	011100	if a 6-bit end fill pattern is required
Fill3_4	=	011	if a 3-bit end fill pattern is required

The end fill patterns shall be transmitted in bit order starting with the leftmost bit. End fill patterns are chosen independently on each of Tdata[3:0].

The particular combination of end fill patterns required on TData[3:0] is a function of the MAC frame length. The four combinations that are possible are illustrated in Figure 14-9.



x = the number (modulo 5) of octets in the MAC frame.
esd = <esd2>, <esd4>, or IPM as required.

Figure 14-9—Possible TData[3:0] end combinations

14.4.3 TData[3:0] frame format at the MII

The physical frame structure at the TData[3:0] circuits of the MII shall be:

TData[0]:	<preamble><ssd><sextets><esd><fill (as necessary)>
TData[1]:	<preamble><ssd><sextets><esd><fill (as necessary)>
TData[2]:	<3-bit filler><preamble><ssd><sextets><esd><fill (as necessary)>
TData[3]:	<3-bit filler><preamble><ssd><sextets><esd><fill (as necessary)>

During packet transmission, the Delimiter Generator function produces a stream of preamble pattern, an ssd, sextets, and an esd on each of TData[3:0]. In addition, this stream includes fill patterns that are added to the preamble pattern on TData[3:2] and, when required, appended to the esds on TData[3:0]. The specification of the Delimiter Generator function ensures that the streams on each of TData[3:0] begin and end simultaneously for each packet transmission.

The TData[3:0] stream format is shown in Figure 14-4 for the transmission of a 64-octet MAC frame. Twenty-six sextets are transmitted on each of TData[2:0], and 25 sextets are transmitted on TData[3]. A 6-bit end fill pattern (Fill6_2 or Fill6_4) is appended to the esd on TData[3] to equalize the length of the streams on TData[2] and TData[3]. A 3-bit end fill pattern (Fill3_2 or Fill3_4) is appended to the esd on each of TData[1:0] to equalize the length of the streams on TData[3:2]. In this way, all four streams are guaranteed have an equal number of bits.

14.4.4 Receive functions

14.4.4.1 Delimiter detector function

The PMI shall be capable of detecting preamble patterns, start of stream delimiters, and end of stream delimiters within the streams received on MII circuits RData[3:0]. The Delimiter Detector function shall detect these patterns and also extract data from the streams on RData[3:0] while the PMI_RECEIVE_ENABLE.request parameter receive_enable_state has the value of assert. Data shall be received on RData[3:0] synchronous with RxClk.

14.4.4.1.1 Preamble and ssd detection

The PMI shall detect the occurrence of a repeating pattern of 01 whenever the PMI_RECEIVE_ENABLE.request parameter receive_enable_state has the value of "assert" and RxClk is active. When the first deviation from this pattern during each packet reception is detected on RData[i], the PMI shall attempt to detect an ssd on RData[i]. The PMI shall examine 16 consecutive bits starting with the four bits received immediately prior to the first deviation in the pattern. A valid ssd indicating a normal-priority packet shall be when the pattern:

```
0101 111100  000011
```

is received on RData[i] leftmost bit first. If this pattern is detected on all of RData[3:0] within 8 cycles of RxClk of the pattern being detected on either of Rdata[1:0], and within 5 cycles of RxClk of the pattern being detected on either of Rdata[3:2], then the PMI shall set the PMI_UNITDATA.indication parameter start_mac_frame_status to "valid" when the first octet of packet data is passed to the MAC or the RMAC. The priority parameter shall be set to "normal" for all octets passed to the MAC or the RMAC during the current packet reception.

A valid ssd indicating a high-priority packet shall be when the pattern:

```
0101 100000  111110
```

is received on RData[i] leftmost bit first. If this pattern is detected on all of RData[3:0] within 8 cycles of RxClk of the pattern being detected on either of Rdata[1:0], and within 5 cycles of RxClk of the pattern being detected on either of Rdata[3:2], then the PMI shall set the PMI_UNITDATA.indication parameter start_mac_frame_status to "valid" when the first octet of packet data is passed to the MAC or the RMAC. The priority parameter shall be set to "high" for all octets passed to the MAC or the RMAC during the current packet reception.

If an invalid ssd combination is detected on RData[3:0], then the Delimiter Detector function shall continue with data extraction and the PMI shall set the PMI_UNITDATA.indication parameter start_mac_frame_status to the value of "invalid" when passing the first octet of the packet to the MAC or the RMAC and shall set the error_status parameter to the value of "error" when passing the last octet to the MAC or the RMAC. The priority parameter shall be set to "normal." An invalid ssd combination shall be detected when:

a) An invalid start of stream delimiter is detected on one or more of RData[3:0].
b) Valid start of stream delimiters are detected on RData[3:0] but these are not identical (i.e., a mixture of normal and high-priority).
c) Valid start of stream delimiters are not detected on all of RData[3:0] prior to the completion of 8 cycles of RxClk after the first start of stream delimiter has been detected on any RData[1:0].
d) Valid start of stream delimiters are not detected on all of RData[3:0] prior to the completion of 5 cycles of RxClk after the first start of stream delimiter has been detected on any Rdata[3:2].

14.4.4.1.2 Data extraction

Sextets shall be extracted from the stream following the ssd on RData[i]. The first bit of data extracted shall be the bit immediately following the 16 bit pattern used for ssd detection. This bit shall become the leftmost bit of the sextet passed by the Delimiter Detector function to the Decoder function on Channel[i]. Subsequent sextets shall be passed to the Decoder function until an esd is detected. The final sextet extracted from the stream prior to an esd being detected on RData[i] shall be the last sextet passed to the Decoder function.

14.4.4.1.3 End of stream delimiter detection

During data extraction, the PMI shall examine the stream on each of RData[3:0] and attempt to detect an esd. An esd shall be detected on RData[i] whenever either of the patterns:

esd2 = 111111 000011 000001
esd4 = 000000 111100 111110

is received on RData[i] leftmost bit first immediately following an integer number of sextets. An esd shall also be detected on RData[i] whenever the pattern:

IPM = 110000 011111 110000

is received on Rdata[i] leftmost bit first immediately following an integer number of sextets. If the IPM pattern is detected on all of RData[3:0], and if the error_status has not been given the value of “error,” then the PMI shall set the PMI_UNITDATA.indication, error_status parameter to “IPM” when passing the last octet of the packet to the MAC or the RMAC.

The PMI shall end packet reception, cease to extract sextets from RData[3:0], and set the PMI_UNITDATA.indication parameter error_status to “error” when passing the last octet of data to the MAC or the RMAC if either of the following hold:

a) An IPM pattern is detected on one but not all of Rdata[3:0].
b) The value of RCS[2:0] changes from 101 to 000.

14.4.4.2 Decoder function

The PMI shall have the capability to decode sextets received on each RData channel during packet reception. Sextets shall be received from the Delimiter Detector function and shall be used by the Decoder function to determine the quintet to be passed to the Octet Assembler function, according to Table 14-5. The PMI shall decode the first sextet of each RData[i], (i = 0, 1, 2, 3), stream using the mode 2 columns of Table 14-5. The PMI shall continue decoding sextets using the mode 2 columns of Table 14-5 until a valid weight 2 sextet has been received and decoded. For subsequent sextets, the PMI shall use the mode 4 columns until a valid weight 4 sextet has been received and decoded. The PMI shall then revert to using the mode 2 columns and continue to alternate between modes whenever a valid weight 2 or valid weight 4 sextet has been received.

For each received sextet, Table 14-5 lists both the output quintet and the received sextet weight. The leftmost bit of an rx_Sextet is received first on the Channel[i]. The leftmost bit of the corresponding output quintet shall be the bit of the quintet that occurs soonest in the Channel[i] serial bit stream.

The alternation between mode 2 and mode 4 shall occur independently on each of the four channels.

If an invalid sextet is received during reception of a packet, the PMI shall give the error_status parameter of the PMI_UNITDATA.indication primitive the value of “error” when passing the last octet of the received packet to the MAC or the RMAC. A sextet is invalid if it is not included in the mode 2 or mode 4 columns of Table 14-5. A sextet is also invalid if it has weight 2 and is received while the Decoder function is in mode 4, or if it has weight 4 and is received while the Decoder function is in mode 2. If either:

a) The Decoder function is in mode 2 after the last sextet has been received on Channel[i], and the Delimiter Detector function detects an esd4 pattern on Rdata[i], or

b) The Decoder function is in mode 4 after the last sextet has been received on Channel[i], and the Delimiter Detector function detects an esd2 pattern on RData[i],

the PMI shall give the error_status parameter the value of “error” when passing the last octet of the received packet to the MAC or the RMAC.

Table 14-5—Decoder table for the Decoder function

	Mode 2		**Mode 4**	
Received sextet rx_Sextet	**Output quintet (Mode_2_decode)**	**Received sextet weight (Mode_2_status)**	**Output quintet (Mode_4_decode)**	**Received sextet weight (Mode_4_status)**
000000	*	error	*	error
000001	*	error	*	error
000010	*	error	*	error
000011	*	error	*	error
000100	*	error	*	error
000101	10000	2	*	error
000110	01011	2	*	error
000111	01000	3	01000	3
001000	*	error	*	error
001001	10010	2	*	error
001010	00100	2	*	error
001011	00111	3	00111	3
001100	00000	2	*	error
001101	00011	3	00011	3
001110	00110	3	00110	3
001111	*	error	*	error
010000	*	error	*	error
010001	*	error	*	error
010010	11110	2	*	error
010011	11101	3	11101	3
010100	11010	2	*	error
010101	00101	3	00101	3
010110	10011	3	10011	3
010111	*	error	01100	4
011000	10101	2	*	error
011001	10110	3	10110	3
011010	01101	3	01101	3
011011	*	error	01110	4
011100	11100	3	11100	3
011101	*	error	*	error
011110	*	error	10111	4
011111	*	error	*	error

* Indicates that an arbitrary quintet value is decoded.

Table 14-5—Decoder table for the Decoder function *(continued)*

	Mode 2		Mode 4	
Received sextet rx_Sextet	**Output quintet (Mode_2_decode)**	**Received sextet weight (Mode_2_status)**	**Output quintet (Mode_4_decode)**	**Received sextet weight (Mode_4_status)**
100000	*	error	*	error
100001	10111	2	*	error
100010	00010	2	*	error
100011	01001	3	01001	3
100100	01110	2	*	error
100101	10001	3	10001	3
100110	01010	3	01010	3
100111	*	error	10101	4
101000	01100	2	*	error
101001	01111	3	01111	3
101010	11001	3	11001	3
101011	*	error	11010	4
101100	00001	3	00001	3
101101	*	error	11110	4
101110	*	error	00010	4
101111	*	error	*	error
110000	*	error	*	error
110001	11000	3	11000	3
110010	11111	3	11111	3
110011	*	error	00000	4
110100	11011	3	11011	3
110101	*	error	00100	4
110110	*	error	10010	4
110111	*	error	*	error
111000	10100	3	10100	3
111001	*	error	01011	4
111010	*	error	10000	4
111011	*	error	*	error
111100	*	error	*	error
111101	*	error	*	error
111110	*	error	*	error
111111	*	error	*	error

* Indicates that an arbitrary quintet value is decoded.

14.4.4.3 Octet assembler function

The octet assembler function deciphers four streams of ciphered quintets from the 5B6B decoders, merges the four streams into one composite PMI data quintet stream, and then converts the merged data quintets to octets for delivery to the MAC or the RMAC.

14.4.4.3.1 Deciphering function

Each RData[i] stream shall be deciphered using the identical stream cipher with the same initialization and operating conditions described in 14.4.2.1.2.

14.4.4.3.2 Quintet merging function

The reverse of the process described in 14.4.2.1.1 shall be used to merge PMI RData[i] quintet streams into a single PMI quintet stream. That is, the first PMI quintet shall be the first RData[0] quintet, the second PMI quintet shall be the first RData[1] quintet, the third PMI quintet shall be the first RData[2] quintet, and the fourth PMI quintet shall be the first RData[3] quintet. This cyclic process shall be continued until the last quintet has been merged into the single PMI quintet stream.

The number of quintets received on each channel may not necessarily be equal. However, a valid packet should obey the following rules simultaneously:

a) $N[3] \leq N[2] \leq N[0]$
b) $N[0] - 1 \leq N[3]$

where N[i] is the total number of quintets received on the Channel[i].

If either of these rules is violated at the end of a packet reception, the PMI shall set the error_status parameter to the value of “error” when passing the last octet to the MAC or the RMAC.

NOTE—Conditions a) and b) above simply ensure that the end of stream pattern is one of the valid patterns shown in Figure 14-9.

14.4.4.3.3 Quintet to octet converter

The quintet to octet converter shall form octets using the bits of the PMI data quintet stream. The leftmost bit of each octet formed shall be the bit, from that octet, that occurs soonest in the MAC or the RMAC frame serial bit stream. The subsequent bits of that octet shall maintain the bit ordering of the originating quintet bit stream. The PMI shall indicate the value of each octet it forms by a PMI_ UNITDATA.indication having a data_octet value equal to the octet value.

The PMI shall set the priority parameter of each PMI_UNITDATA.indication primitive in accordance with 14.4.4.1.1 during reception of a packet.

The start_mac_frame_status parameter of the first PMI_UNITDATA.indication primitive generated by the PMI, during reception of a packet, shall be set to either “valid” or “invalid” by the PMI in accordance with 14.4.4.1.1. The PMI shall set the mac_frame parameter to the value of “begin” for the first assembled octet, “end” for the last assembled octet, and “more” for intervening octets.

The PMI shall set the error_status of the last PMI_UNITDATA.indication primitive of a received packet in accordance with 14.4.4.2 and 14.4.4.1.

The PMI quintet stream, consisting of an integer number of quintets, may give rise to an incomplete final octet. The bits of such an incomplete octet shall be regarded as extension bits. The PMI shall count the number of extension bits received and shall discard them. If there are more than four extension bits, the PMI shall set the PMI_UNITDATA.indication, error_status parameter to the value of “error” when passing the last octet of the received packet to the MAC or the RMAC.

14.5 State diagrams

State diagrams are included to clarify the PMI specifications. The model used is that of a PMI Master state machine that controls the PMI Quintet Assembler functions, Delimiter Generator function, Delimiter Detector function, and Octet Assembler functions. An overview of the interactions between the state machines and the functions of this clause is shown in Figure 14-10.

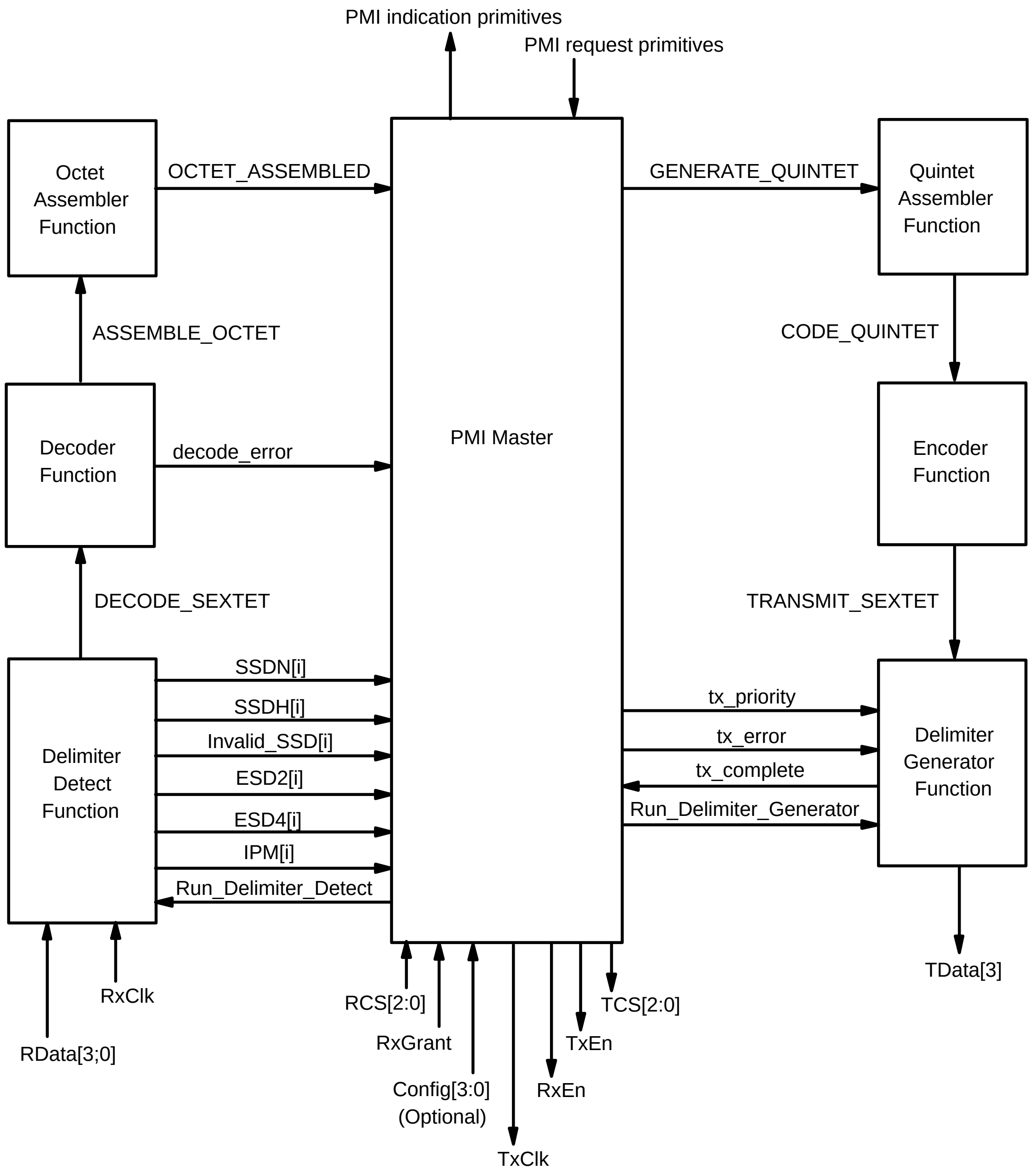


Figure 14-10—PMI state diagram and functions overview

The PMI Master also accepts input from the Data functions and control information from the PMD or the MAC or the RMAC. The PMI Master generates PMI_Control.indication and PMI_UNITDATA.indication primitives in response to MII inputs from the PMD. It converts PMI_CONTROL.request primitives received from the MAC or the RMAC into the appropriate MII signals to control the PMD.

PMI_UNITDATA.request primitives are converted to TData[3:0] signals by the PMI Data Transmit functions under the control of the PMI Master.

14.5.1 State variable definitions

Variables are used in the state diagrams to indicate the status of PMI inputs and signals, to control PMI operation, and to pass state information between functions.

In the variable definitions, the name of the variable is followed by a brief definition of the variable and a list of values the variable may take. For those variables for which a particular value has been identified as the default, the variable has the default value when no active state contains a term assigning a different value. For example, the variable TxEn has the value of “true” whenever the PMI is in a state that enables TxEn = true. Otherwise, this variable has the default value of “false.”

For those variables for which no default value has been defined, the variable shall maintain its current value until explicitly changed by an operation.

The PMI service primitives are used as variables in the PMI state diagrams. These primitives and their parameters are defined in Clause 8. The following variables are also used in the PMI state diagrams.

AnyESD. Indicates that an esd has been detected on at least one channel.

Values:

true when:	(ESD2[0]= true) or (ESD2[1]= true) or (ESD2[2]= true) or (ESD2[3]= (ESD4[0]= true) or (ESD4[1]= true) or (ESD4[2]= true) or (ESD4[3]= true).
false:	Otherwise.

AnySSD. Indicates that an ssd has been detected on at least one channel.

Values:

true when:	(SSDH[0]= true) or (SSDH[1]= true) or (SSDH[2]= true) or (SSDH[3]= true) or (SSDN[0]=true) or (SSDN[1]= true) or (SSDN[2]= true) or (SSDN[3]= true).
false:	Otherwise.

ASSEMBLE_OCTET(rx_quintet, rx_quintet_status). Controls the operation of the Octet Assembler function. It is generated by the Decoder function.

Parameters:

rx_quintet:	The quintet of data to be assembled into the octet stream.
rx_quintet_status:	The status of the current quintet. Set to: “begin” for the first quintet of data on each channel, “end” for the last quintet, and “more” for intermediate quintets.

CODE_QUINTET(tx_quintet, tx_quintet_status). Controls the operation of the Encoder function. It is generated by the Quintet Assembler function.

Parameters:

tx_quintet:	The quintet of data to be coded. Values: Binary 00000 through 11111.
tx_quintet_status:	The status of the current quintet. Set to: “begin” for the first quintet of data to be coded on each channel, “end” for the last quintet, and “more” for intermediate quintets.

codetable(row, column). Indicates an entry in the code table (Table 14-4).

Parameters:

row:	A quintet indexing a row of entries in the codetable.
column:	A pointer to one of the following columns of the codetable:

Mode_2_code -	The sextet to be used in the Mode 2 coding state.
Mode_2_next -	The next mode to be used while in the Mode 2 coding state.
Mode_4_code -	The sextet to be used in the Mode 4 coding state.
Mode_4_next -	The next mode to be used while in the Mode 4 coding state.

decode_error. Indicates that a code violation has occurred in the received data.

Values:

true when:	(decode_error[0]= true) or (decode_error[1]= true) or (decode_error[2]= true) or (decode_error[3]= true).
false:	Otherwise.

decode_error[i]. Indicates that the Decoder function has detected a code violation in the received data on Channel[i] (i=0,1,2 or 3).

Values:

true:	A decoding error has occurred on Channel[i].
false:	A decoding error has not occurred on Channel[i].

DECODE_SEXTET(rx_Sextet, rx_Sextet_status). Controls the operation of the Decoder function. It is generated by the Delimiter Detector function.

Parameters:

rx_Sextet:	The sextet of coded data to be decoded.
rx_Sextet_status:	The status of the current sextet. Set to: “begin” for the first sextet of data on each channel, “end” for the last sextet, and “more” for intermediate sextets.

decode_status. Indicates the status of the sextet currently being decoded by the Decoder function.

Values:

2:	The current sextet is weight 2 and obeys alternation rule.
3:	The current sextet is weight 3.
4:	The current sextet is weight 4 and obeys alternation rule.
error:	The current sextet is either invalid or violates coding rules.

decodetable(row, column). Indicates an entry in the decode table (Table14-5).

Parameters:

row:	A sextet indexing a row of entries in the decode table.
column:	A pointer to one of the following columns of the decode table:

Mode_2_decode -	The quintet to be used in the Mode 2 decoding state.
Mode_2_status -	The status of the current sextet while in Mode 2 decoding state.
Mode_4_decode -	The quintet to be used in the Mode 4 decoding state.
Mode_4_status -	The status of the current sextet while in Mode 4 decoding state.

ESD. A composite variable indicting that a valid esd has been received on all four channels.

Values:

true when:	((ESD2[0]= true) or (ESD4[0]= true)) and ((ESD2[1]= true) or (ESD4[1]= true)) and ((ESD2[2]= true) or (ESD4[2]= true)) and ((ESD2[3]= true) or (ESD4[3]= true)).
false:	Otherwise.

ESD2[i]. Indicates that the Delimiter Detector function has detected an esd <esd2> on Channel[i] (i=0,1,2 or 3).

Values:

true:	An <esd2> has been detected during the current packet reception.
false:	An <esd2> has not been detected during the current packet reception.

ESD4[i]. Indicates that the Delimiter Detector function has detected an esd <esd4> on Channel[i] (i=0,1,2 or 3).

Values:

true:	An <esd4> has been detected during the current packet reception.
false:	An <esd4> has not been detected during the current packet reception.

esdskew. Count of the number of clock cycles during which a valid esd has been detected on one but not all channels of received Data.

Values: Non-negative integers.

GENERATE_QUINTET(tx_octet, tx_octet_status). Controls the operation of the Quintet Assembler function.

Parameters:

tx_octet:	The octet to be input to the quintet assembler.
tx_octet_status:	The status of the current octet. Set to: “begin” for the first octet of data, “end” for the last octet, and “more” for intermediate octets.

Invalid_SSD. A composite variable indicting that an invalid start of stream delimiter or an invalid combination of start of stream delimiters has been detected.

Values:

```
true when:  ((SSDH[0]= true or SSDH[1]= true or SSDH[2]= true or SSDH[3]= true)
               and (SSDN[0]= true or SSDN[1]= true or SSDN[2]= true or SSDN[3]= true))
            or (Invalid_SSD[0]= true) or (Invalid_SSD[1]= true) or (Invalid_SSD[2]= true)
            or (Invalid_SSD[3]= true) or (skew > 8)
            or ((SSDN[2] or SSDN[3] or SSDH[2] or SSDN[3]) and (skew > 5)).
false:      Otherwise.
```

Invalid_SSD[i]. Indicates that the Delimiter Detector function has detected an invalid ssd on Channel[i] (I=0, 1, 2, or 3).

Values:

true:	An invalid ssd has been detected during the current packet reception.
false:	An invalid ssd has not been detected during the current packet reception.

IPM. A composite variable indicting that an IPM has been received on all four channels.

Values:

true when:	(IPM[0]= true) and (IPM[1]= true) and (IPM[2]= true) and (IPM[3]= true).
false:	Otherwise.

IPM[i]. Indicates that the Delimiter Detector function has detected an IPM on Channel[i] (i=0,1,2 or 3).

Values:

true:	An <IPM> esd has been detected during current packet reception.
false:	An <IPM> esd has not been detected during current packet reception.

next_mode. Indicates the set of sextets the Encoder function should use to code the next quintet.

Values:

2: The next sextet shall be chosen from the Mode 2 set.
4: The next sextet shall be chosen from the Mode 4 set.

OCTET_ASSEMBLED(rx_octet, rx_octet_status). Reports the results of the Octet Assembler function.

Parameters:

rx_octet:	The octet output from the octet assembler.
rx_octet_status:	The status of the current octet. Set to: “begin” for the first octet of data that has been assembled, “end” for the last octet, and “more” for intermediate octets. If a valid octet cannot be assembled, the status shall be set to “error” (see 14.4.4.3).

RCS. Indicates the status of the MII circuits RCS[2:0].

Values: binary 000 through 111.

Run_Delimiter_Detector. Controls operation of the Delimiter Detector function.

Values:

true:	Run the Delimiter Detector function.
false:	Idle the Delimiter Detector function (default).

Run_Delimiter_Generator. Controls operation of the Delimiter Generator function.

true:	Run the Delimiter Generator function.
false:	Idle the Delimiter Generator function (default).

RxClk_edge. Indicates positive-going edges occurring on the RxClk circuit.

Values:

true:	A leading edge has occurred.
false:	A leading edge has not occurred.

RxEn. Controls the status of the MII RxEn circuit.

Values:

true:	RxEn is enabled.
false:	RxEn is disabled (default).

Rx_End. Indicates that packet reception has been terminated.

Values:

true when:	(PMI_RECEIVE_ENABLE.request(receive_enable_state = deassert) or (RCS= 000).
false:	Otherwise.

rx_error. Indicates the error status of the frame being received.

Values:

true:	An invalid ssd, code error, or weight error has been detected.
false:	An invalid ssd, code error, or weight error has not been detected.

Rx_Priority. Indicates the priority of the received packet.

Values:

normal:	Normal-priority packet.
high:	High-priority packet.

skew. Count of the number of clock cycles during which a valid ssd has been detected on one but not all channels of received Data.

Values: non-negative integers.

SSDH. Indicates that a high-priority packet is being received.

Values:

true when:	(SSDH[0]= true) and (SSDH[1]= true) and (SSDH[2]= true) and (SSDH[3]= true).
false:	Otherwise.

SSDH[i]. Indicates that the Delimiter Detector function has detected a high-priority ssd on Channel[i] (i=0,1,2 or 3).

Values:

true:	A high-priority ssd has been detected during the current packet reception.
false:	A high-priority ssd has not been detected during the current packet reception.

SSDN. Indicates that a normal-priority packet is being received.

Values:

true when:	(SSDN[0]= true) and (SSDN[1]= true) and (SSDN[2]= true) and (SSDN[3]= true).
false:	Otherwise.

SSDN[i]. Indicates that the Delimiter Detector function has detected a normal-priority ssd on Channel[i] (i=0,1,2 or 3).

Values:

true:	A high-priority ssd has been detected during the current packet reception.
false:	A high-priority ssd has not been detected during the current packet reception.

TRANSMIT_SEXTET(tx_Sextet, tx_Sextet_status). Transfers coded data from the Encoder function to the Delimiter Generator function. It is generated by the Encoder function.

Parameters:

tx_Sextet:	The sextet of coded data to be transmitted.
tx_Sextet_status:	The status of the current sextet. Set to: “begin” for the first sextet of data on each channel, “end_2” or “end_4” for the last sextet, and “more” for intermediate sextets.

transmit_complete. Indicates the value to be sent to the PMI in the PMI_TRANSMIT.indication primitive.

Values:

more:	Indicates that the current packet transmission is still in progress.
end:	Indicates that packet transmission has been completed. A PMI_TRANSMIT.indication with the value of “end” shall be sent on the next cycle of TxClk after the PMI has deasserted “TxEn”.

tx_complete. Indicates the results of the Delimiter Generator function.

Values:

true:	The Delimiter Generator function has completed passing <esd> or <IPM> plus any needed <Fill> to the MII circuits TData[3:0] during the current packet transmission.
false:	The Delimiter Generator function has not completed passing <esd> or <IPM> plus any needed <Fill> to the MII circuits TData[3:0] during the current packet transmission.

TxEn. Controls the status of the MII TxEn circuit.

Values:

true:	TxEn is enabled.
false:	TxEn is disabled (default).

tx_error. Indicates to the Delimiter Generator function the error status of the frame being transmitted. Set by the PMI_UNITDATA.request primitive.

Values:

valid:	The received packet was error-free.
IPM:	The received packet contained either an error or an <IPM>.

tx_priority. Indicates to the Delimiter Generator function the priority of the frame being transmitted. Set by the PMI_UNITDATA.request primitive.

Values:

normal:	Normal-priority packet.
high:	High-priority packet.

14.5.2 PMI state diagrams

State diagrams are given for the PMI Master (Figures 14-11 and 14-12), the Encoder function (Figure 14-13) and the Decoder function (Figure 14-14).

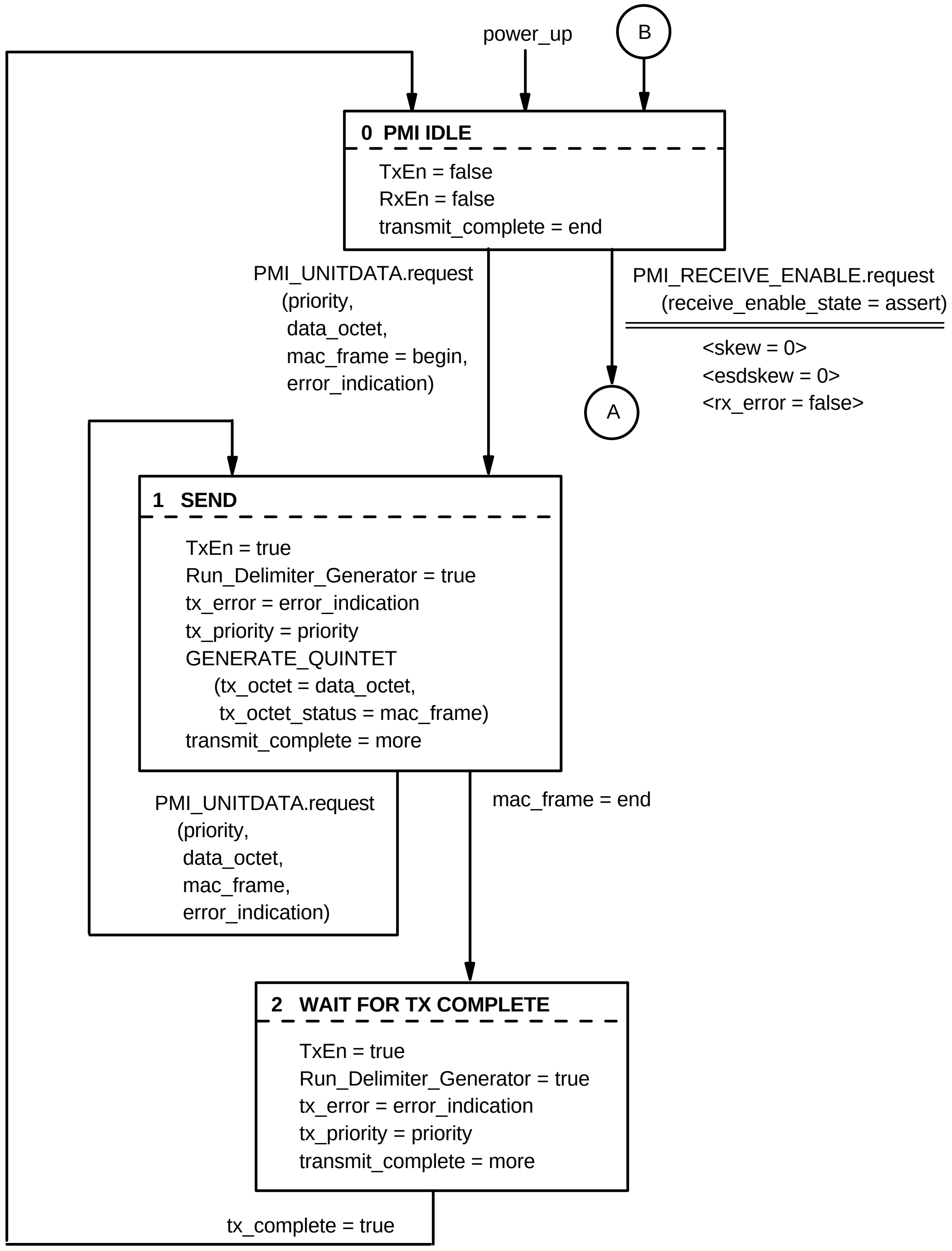

power_up
B
0 PMI IDLE
TxEn = false
RxEn = false
transmit_complete = end
PMI_UNITDATA.request (priority, data_octet, mac_frame = begin, error_indication)
PMI_RECEIVE_ENABLE.request (receive_enable_state = assert)
<skew = 0>
<esdskew = 0>
<rx_error = false>
A
1 SEND
TxEn = true
Run_Delimiter_Generator = true
tx_error = error_indication
tx_priority = priority
GENERATE_QUINTET (tx_octet = data_octet, tx_octet_status = mac_frame)
transmit_complete = more
PMI_UNITDATA.request (priority, data_octet, mac_frame, error_indication)
mac_frame = end
2 WAIT FOR TX COMPLETE
TxEn = true
Run_Delimiter_Generator = true
tx_error = error_indication
tx_priority = priority
transmit_complete = more
tx_complete = true

Figure 14-11—PMI master, part 1

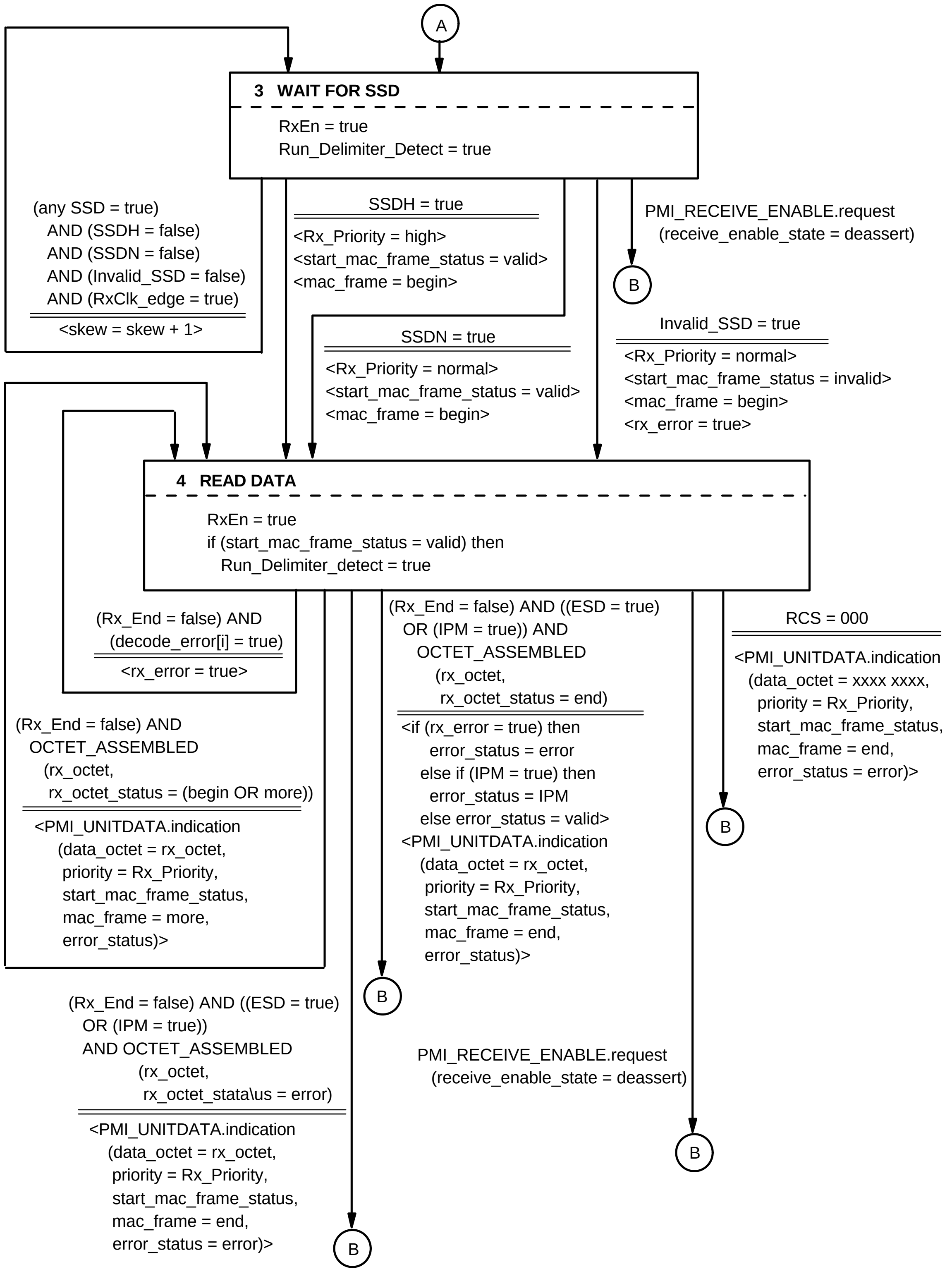


Figure 14-12—PMI master, part 2

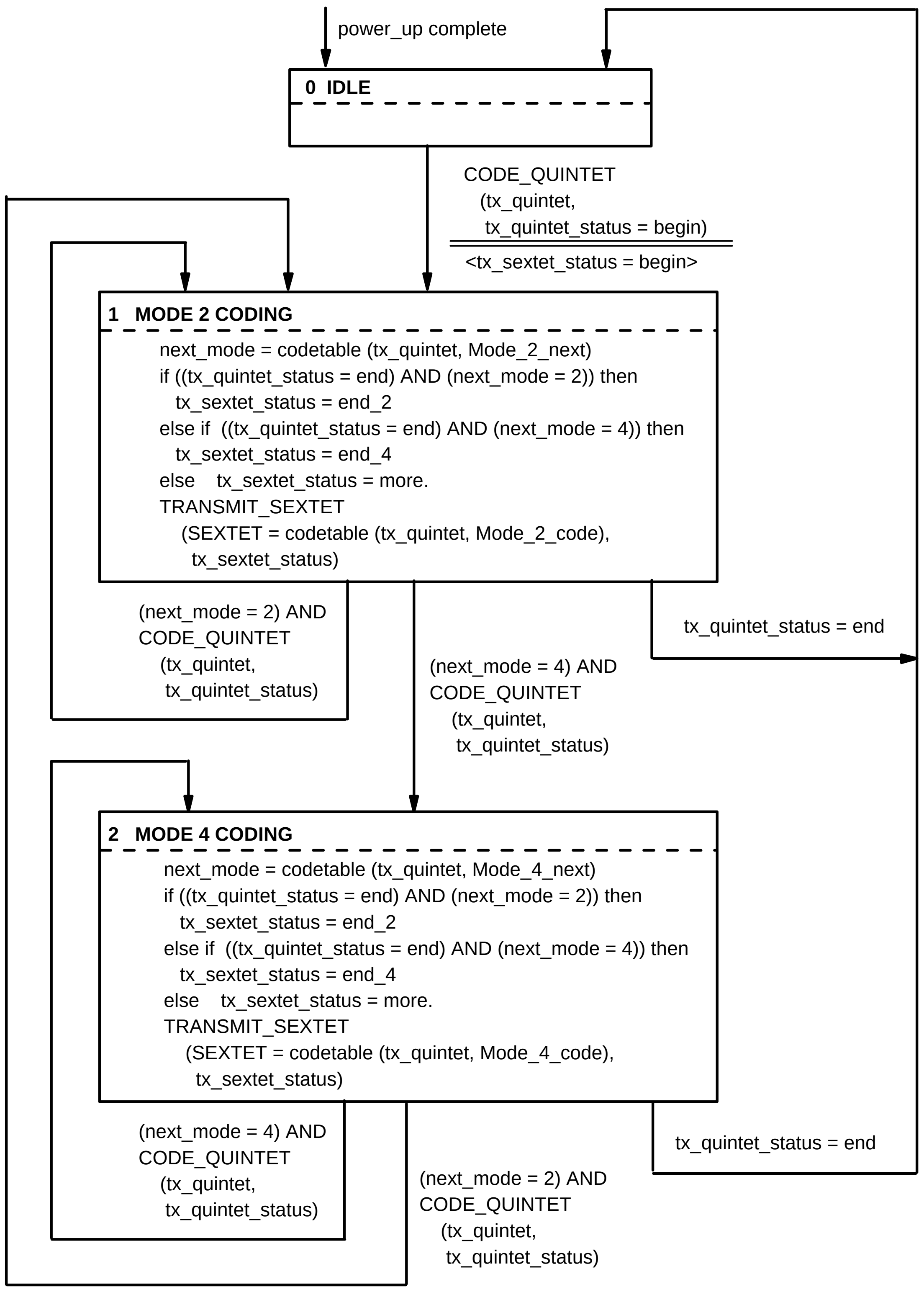

power_up complete
0 IDLE
CODE_QUINTET (tx_quintet, tx_quintet_status = begin)
<tx_sextet_status = begin>
1 MODE 2 CODING
next_mode = codetable (tx_quintet, Mode_2_next)
if ((tx_quintet_status = end) AND (next_mode = 2)) then
tx_sextet_status = end_2
else if ((tx_quintet_status = end) AND (next_mode = 4)) then
tx_sextet_status = end_4
else tx_sextet_status = more.
TRANSMIT_SEXTET
(SEXTET = codetable (tx_quintet, Mode_2_code),
tx_sextet_status)
(next_mode = 2) AND CODE_QUINTET (tx_quintet, tx_quintet_status)
(next_mode = 4) AND CODE_QUINTET (tx_quintet, tx_quintet_status)
tx_quintet_status = end
2 MODE 4 CODING
next_mode = codetable (tx_quintet, Mode_4_next)
if ((tx_quintet_status = end) AND (next_mode = 2)) then
tx_sextet_status = end_2
else if ((tx_quintet_status = end) AND (next_mode = 4)) then
tx_sextet_status = end_4
else tx_sextet_status = more.
TRANSMIT_SEXTET
(SEXTET = codetable (tx_quintet, Mode_4_code),
tx_sextet_status)
(next_mode = 4) AND CODE_QUINTET (tx_quintet, tx_quintet_status)
(next_mode = 2) AND CODE_QUINTET (tx_quintet, tx_quintet_status)
tx_quintet_status = end

Figure 14-13—Encoder state diagram

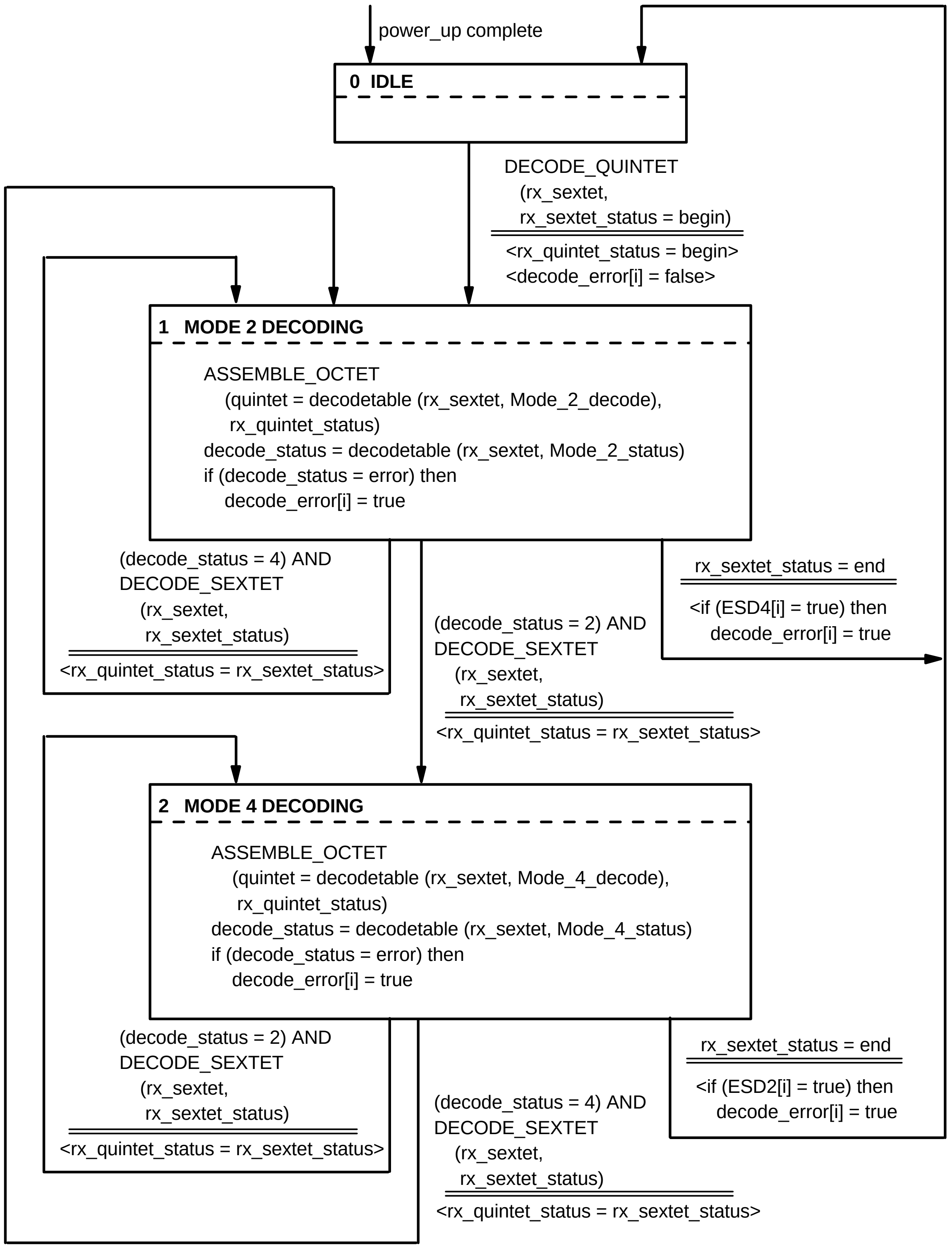

power_up complete
0 IDLE
DECODE_QUINTET
(rx_sextet,
rx_sextet_status = begin)
<rx_quintet_status = begin>
<decode_error[i] = false>
1 MODE 2 DECODING
ASSEMBLE_OCTET
(quintet = decodetable (rx_sextet, Mode_2_decode),
rx_quintet_status)
decode_status = decodetable (rx_sextet, Mode_2_status)
if (decode_status = error) then
decode_error[i] = true
(decode_status = 4) AND
DECODE_SEXTET
(rx_sextet,
rx_sextet_status)
<rx_quintet_status = rx_sextet_status>
(decode_status = 2) AND
DECODE_SEXTET
(rx_sextet,
rx_sextet_status)
<rx_quintet_status = rx_sextet_status>
rx_sextet_status = end
<if (ESD4[i] = true) then
decode_error[i] = true
2 MODE 4 DECODING
ASSEMBLE_OCTET
(quintet = decodetable (rx_sextet, Mode_4_decode),
rx_quintet_status)
decode_status = decodetable (rx_sextet, Mode_4_status)
if (decode_status = error) then
decode_error[i] = true
(decode_status = 2) AND
DECODE_SEXTET
(rx_sextet,
rx_sextet_status)
<rx_quintet_status = rx_sextet_status>
(decode_status = 4) AND
DECODE_SEXTET
(rx_sextet,
rx_sextet_status)
<rx_quintet_status = rx_sextet_status>
rx_sextet_status = end
<if (ESD2[i] = true) then
decode_error[i] = true

Figure 14-14—Decoder state diagram

15. Medium Independent Interface (MII) specifications

15.1 Scope

This clause defines the functional characteristics of the MII and, additionally, provides electrical specifications for an optional, physically exposed MII. The MII is not required to adhere to any particular mechanical specification. The relationship of the MII to the LAN model is shown in Figure 15-1.

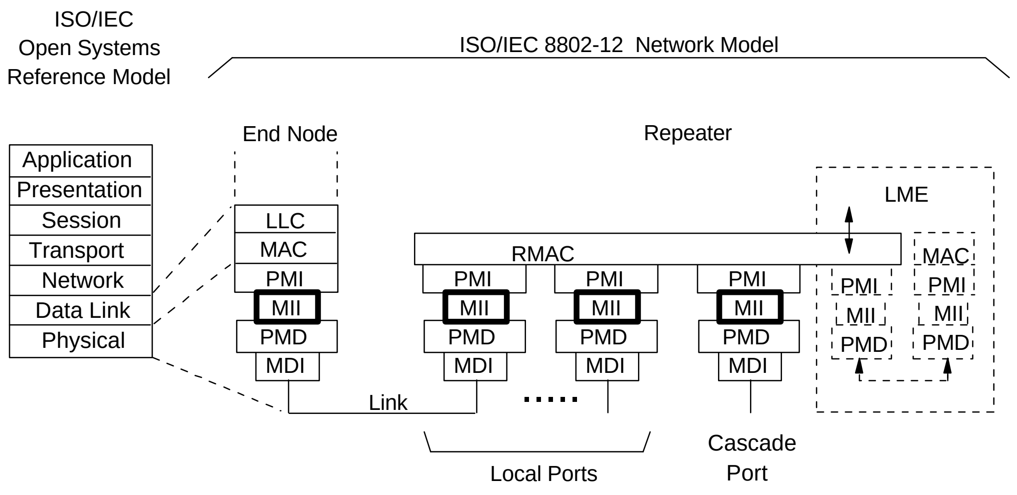


Figure 15-1—Relationship of the MII to the LAN model

15.2 MII functional characteristics

This subclause defines the functional characteristics of the MII. This subclause applies to all implementations of the PMI and PMD sublayers regardless of whether or not the MII interface is physically exposed.

15.2.1 RxGrant

The RxGrant interface transfers RxGrant from the PMD to the PMI (see 14.3.3).

15.2.2 Transmit Clock (TxClk)

A transmit clock is supplied to the MII by the PMI (see 14.4.1).

15.2.3 Transmit Control State (TCS[2:0])

The transmit control state interface transfers the transmit control state from the PMI to the PMD (see 14.3.1).

15.2.4 Transmit Enable (TxEn)

Transmit enable is supplied by the PMI to the MII to enable the transfer of data from the PMI to the PMD through the MII (see 14.3.4).

15.2.5 Transmit Data (TData[3:0])

The transmit data interface is used to transfer the four channels of data from the PMI to the PMD when TxEn is asserted.

15.2.6 Receive Control State (RCS[2:0])

The receive control state interface transfers the receive control state from the PMD to the PMI (see 14.3.2).

15.2.7 Receive Enable (RxEn)

Receive enable is supplied by the PMI to the MII to enable the transfer of data from the PMD to the PMI through the MII (see 14.3.5).

15.2.8 Receive Clock (RxClk)

Receive clock is provided to the MII by the PMD during packet reception. This clock provides for the transfer of received data from the PMD to the PMI through the MII (see 14.4.1.2).

15.2.9 Receive Data (RData[3:0])

The receive data interface is used to transfer the four channels of data from the PMD to the PMI when RxEn is asserted.

15.2.10 PMD configuration [Optional]

The PMD configuration interface may be used by the PMD to indicate one of sixteen PMD types to the PMI through the MII.

15.3 Electrical characteristics of an optional physically exposed MII

This subclause defines the electrical requirements of an optional physically exposed MII between the PMI and the PMD sublayers. A physically exposed MII:

a) Allows interchangeability of PMDs for easy link reconfiguration (e.g., from 4-UTP to fibre)
b) Makes it possible for the same repeater to support multiple link media simultaneously

The interface shall contain power, data, and control circuits as listed in Table 15-1. The interface may also contain the optional PMD configuration signals as defined in 15.3.11.

Table 15-1—MII circuits

Circuit	Name	PMI to PMD	PMD to PMI	Remarks
Vdd	Voltage Plus	x		DC power
Gnd	Ground	x		Power return
TxClk	Transmit Clock	x		Control
TCS[2:0]	Transmit Control State	x		Encoded control
TxEn	Transmit Enable	x		Control
TData[3:0]	Data Out	x		4 serial streams of Encoded Data
RxEn	Receive Enable	x		Control
RxGrant	RxGrant		x	Control
RCS[2:0]	Receiver Control State		x	Encoded control
RxClk	Receive Clock		x	Control
RData[3:0]	Data In		x	4 serial streams of Encoded Data

Timing delay between the PMI and MII shall be limited to less than 1 ns. Timing delay is defined as the time difference in a signal from when it is generated and when it is received as measured at the input low voltage (Vil) on a rising edge and input high voltage (Vih) on a falling edge.

Additional control circuits, or mapping of signals other than Demand Priority on to the MII, are not prohibited, but are beyond the scope of this standard.

15.3.1 DC power (Vdd and Gnd)

Power supplied by the PMI, at the MII, shall be 5 V ± 5%. Following power-on, the surge current drawn by the PMD shall be such that:

$$i_p \times T_W \leq 2\times10^{-3} \text{ Ampere-seconds} \quad (7)$$

where

i_p is the peak surge current

T_W is the time during which the current exceeds the larger of 0.75 A or 0.5 i_p

After 100 ms following power-on, the current drawn by the PMD shall not exceed 0.75 A when powered by the MII. The PMD shall be capable of powering up into a normal operating state when supplied by a 0.75 A current limited source.

15.3.2 RxGrant

This circuit is asserted high (Voh) by the PMD when the PMD is decoding grant from the MDI input circuits. While the PMD is not decoding grant, the RxGrant circuit shall be deasserted (Vol) (see 15.3.12). This circuit shall be synchronized with the rising edge of TxClk.

15.3.3 Transmit Clock (TxClk)

A 30 MHz transmit clock is supplied to the MII by the PMI (see 14.4.1). Peak-to-peak jitter of TxClk shall be less than 1.0 ns when measured on the rising edge at the 2.5 V point.

15.3.4 Transmit Control State (TCS[2:0])

The 3-bit encoded value of TCS[2:0] indicates which control signal is to be transmitted by the PMD. TCS[2:0] shall be synchronized with the rising edge of TxClk.

15.3.5 Transmit Enable (TxEn)

This circuit shall be asserted low (Vol) to enable the transfer of output data from the PMI to the PMD. TxEn shall be driven high (Voh) while deasserted (see 15.3.12). TxEn shall be synchronized with the rising edge of TxClk. See Figure 15-2 for TData[3:0] timing relative to TxEn.

15.3.6 Transmit Data (TData[3:0])

These circuits are used to transfer the four channels of serial output data from the PMI to the PMD when TxEn is asserted. TData[3:0] shall be synchronized with the rising edge of TxClk. Figure 15-2 shows TData[3:0] timing relative to TxEn and TxClk.

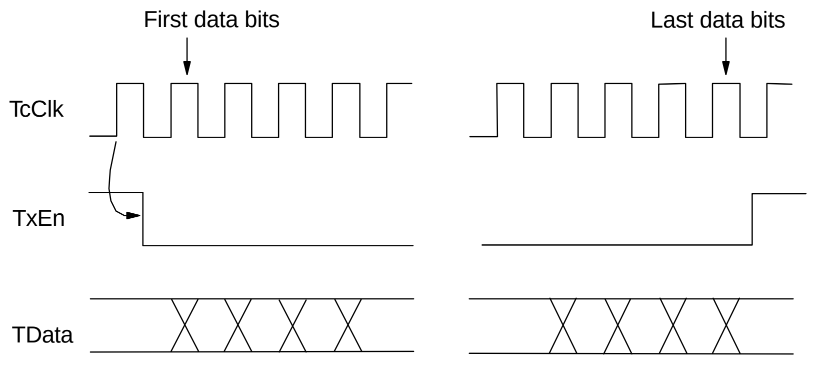


Figure 15-2—Transmit data timing

15.3.7 Receive Control State (RCS[2:0])

These circuits are used to transfer the state of the Receiver Control State from the PMD to the PMI. RCS[2:0] shall be synchronized with the rising edge of TxClk.

15.3.8 Receive Enable (RxEn)

This circuit shall be asserted low (Vol) by the PMI to enable the transfer of input data from the PMD to the PMI. RxEn shall be driven high (Voh) while deasserted (see 15.3.13). RxEn shall be synchronized with the rising edge of TxClk.

15.3.9 Receive Clock (RxClk)

The receive clock is a nominal 30 MHz clock that is recovered from the incoming data stream(s) by the PMD. The output circuit that drives the RxClk signal shall be placed in a high-impedance state while RxEn is deasserted. The PMD shall drive RxClk low (Vol) within 15 ns of the first rising edge of TxClk after RxEn assertion, and RxClk shall be held low until the clock is stable. When stable, the first rising edge of RxClk shall be used to input the first data bits of RData[3:0]. RxClk shall have a duty cycle within the range 40–60% and cycle-to-cycle jitter shall be < ±2 ns. RxClk shall be driven to high impedance within 15 ns of the first rising edge of TxClk after RxEn deassertion.

15.3.10 Receive Data (RData[3:0])

These circuits are used to transfer the four channels of serial input data from the PMD to the PMI while RxEn is asserted. RData[3:0] shall be synchronized with the rising edge of RxClk. While RxEn is deasserted, RData[3:0] shall be driven to high impedance. RData[3:0] shall be driven low (Vol) within 15 ns of the first rising edge of TxClk after RxEn assertion. RData[3:0] shall be valid with the first rising edge of RxClk. RData[3:0] shall be driven to high impedance within 15 ns of the first rising edge of TxClk after RxEn deassertion. Figure 15-3 shows RData[3:0] and RxEn timing relative to RxClk.

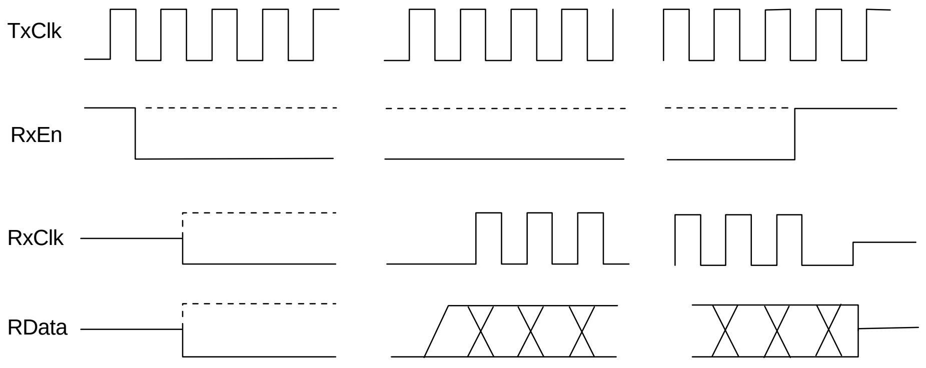


Figure 15-3—Receive data timing

15.3.11 PMD Config[3:0] [Optional]

These circuits allow the PMD to indicate its configuration to the PMI. See Table 15-2 for configuration codes. When implemented, these signals shall be pulled up with 4.6 kΩ ±10% resistors to Vdd.

Table 15-2—PMD/Link configuration codes

Configuration code Config[3:0]	PMD/Link configuration	Configuration code Config[3:0]	PMD/Link configuration
0000	**	1000	**
0001	4-UTP	1001	Fibre
0010	**	1010	**
0011	**	1011	2-STP
0100	**	1100	**
0101	**	1101	**
0110	**	1110	**
0111	**	1111	PMD missing or unknown

** Reserved for future standardization

15.3.12 Output drivers

The requirements of the MII output drivers, whether sourced from the PMD or the PMI, shall be met while driving a test load with resistance of 2 KΩ and capacitance between 10 pF and 50 pF. See Figure 15-4 for the test load circuit.

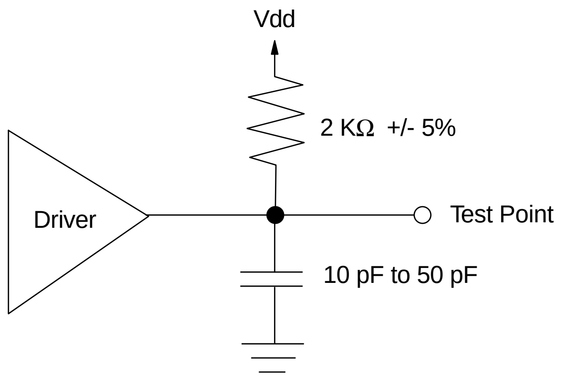
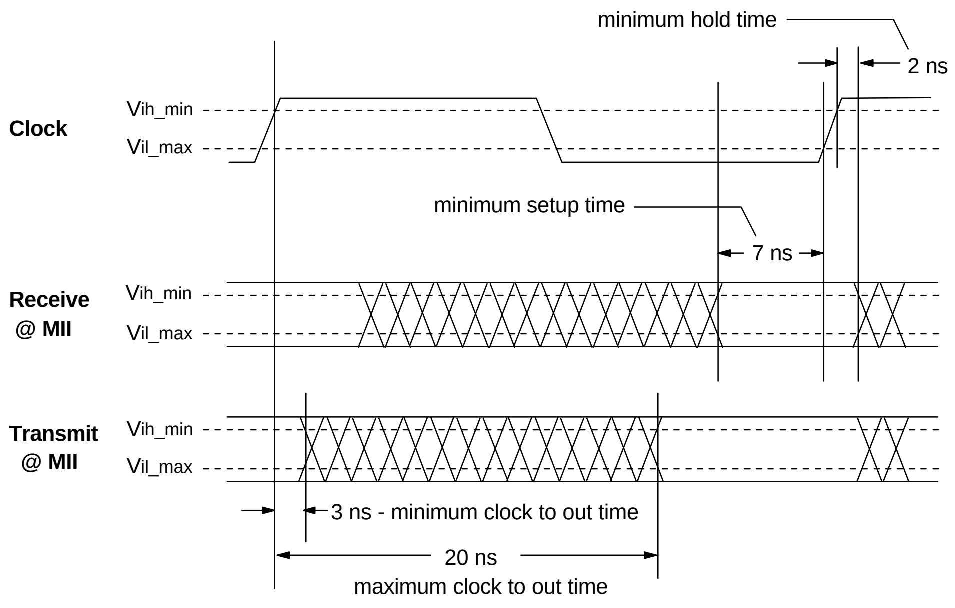


Figure 15-4—MII output driver test load

Circuits with outputs driving a “high” level (Voh) shall have a voltage range of (Vdd – 0.5 V) to (Vdd + 0.3 V). Circuits with outputs driving a “low” level (Vol), shall have a voltage range of – 0.3 V to + 0.5 V. A logic 1 shall be defined as a “high” level. A logic 0 shall be defined as a “low level.” Circuits in the high-impedance state shall have a leakage current of less than 10 µA.

All output waveforms shall have a rise and fall time between 1 ns and 6 ns. This shall be measured between the 20–80% points of the waveform.

Circuits that are synchronized with TxClk or RxClk shall be driven with a “clock to out” time between 3 ns and 20 ns. These times shall be measured between the 50% point of the waveforms. See Figure 15-5.



NOTES

1—Delay (as measured at Vil_max on a rising edge) of any signal, when measured at the PMI and the MII, must be less than 1 ns.

2—Clock refers to either TxClk or RxClk, depending on the signal being described.

3—Vih_min = Vdd – 1.0 V

4—Vil_max = 1.0 V

Figure 15-5—Generic transmit clock-to-out and receive setup and hold times

15.3.13 Input receivers

Inputs shall decode the proper state when the received signal has crossed the input voltage threshold for a minimum set-up time of 7 ns and a minimum hold time of 2 ns. Input capacitance shall be less than 6 pf. Input voltage thresholds shall be Vih ≥ Vdd – 1.0 V and Vil ≤ 1.0 V. Input current shall be less than 1.5 mA.

16. 4-UTP Physical Medium Dependent (PMD) sublayer, Medium Dependent Interface (MDI), and link specifications

16.1 Scope

This clause defines the functional characteristics of the PMD sublayer and provides electrical and mechanical specifications for the MDI and for the 4-UTP (four-pair 100 Ω balanced cable, category 3 or better) link implementation. Annex F provides information on the changes that would be required to this clause to allow the optional use of 4-UTP balanced cabling with a nominal characteristic impedance of 120 Ω.

The relationship of the PMD specification to the LAN model is shown in Figure 16-1. No particular implementation of the PMD is implied. General environmental and safety specifications are given in Annex B.

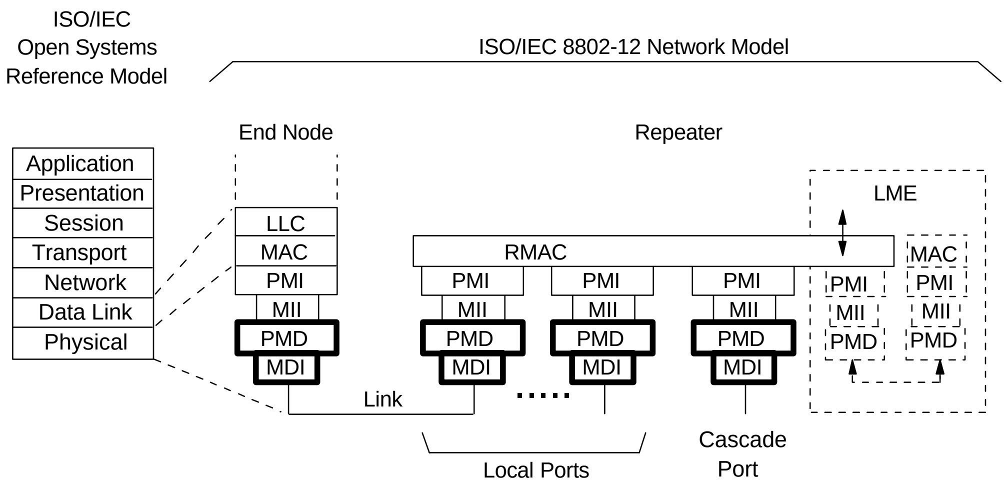


Figure 16-1—PMD and MDI relationship to the LAN model

16.2 Overview

The 4-UTP PMD layer:

a) Enables coupling of the PMI sublayer and MII to four-pair 100 Ω balanced cable link segments as defined in 16.9. (See Figure 16-2.)

b) Supports data packet traffic at a signaling rate of 120 MBd.

c) Allows link segment lengths from 0 m to 100 m or more between an end node and a repeater, and between repeaters.

d) Supports network configurations using the demand-priority access method with baseband signaling on four channels.

16.2.1 Signal conditioning and control functions

The PMD is the medium dependent connection between the MII and the network link segment. It:

a) Generates and transmits outgoing control signals.

b) Transmits outgoing data.

c) Recognizes incoming control signals and differentiates them from incoming data.
d) Recognizes data packet preambles and recovers the receive clock (RxClk) from the incoming data streams.
e) Recognizes and recovers incoming data streams.

16.2.2 PMD interfaces

The PMD has two defined interfaces: one to the PMI sublayer, and one to the link segment. The PMD may be implemented as a separate unit, with a physically exposed MII to the PMI, or it may be logically integrated with the PMI. The interface to the link segment (MDI) shall always be physically exposed.

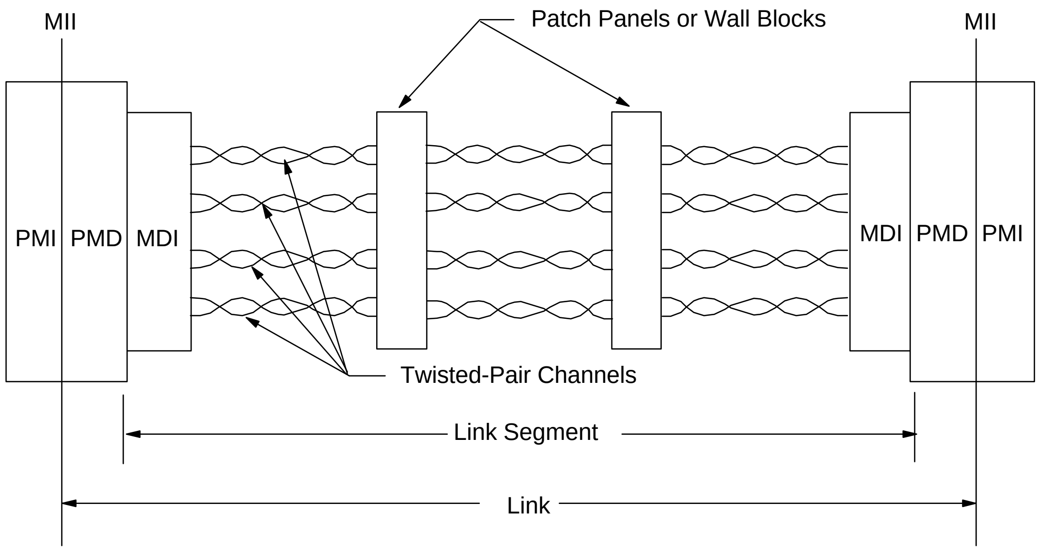


Figure 16-2—4-UTP link

16.2.3 Twisted-pair media

The medium for 4-UTP links is 100 Ω balanced cable meeting the performance requirements defined in 16.9. The medium will typically be up to 100 m of four-pair cable meeting the specifications of ISO/IEC 11801:1995 for 100 Ω balanced cable. Lengths longer than 100 m are permitted providing the link segment meets the performance requirements defined in 16.9.

Bundled cable (25-pair binder-groups) meeting the performance requirements defined in 16.9 is allowed in end-node-to-repeater links, but this requires repeaters with privacy mode and store-and-forward capability to prevent excessive crosstalk during reception and retransmission of packets with group addressing. Bundled cable shall not be used in cascade links between upper- and lower-level repeaters, between repeaters and bridges, or between repeaters and promiscuous end nodes.

16.2.4 PMD/Link configuration

When implemented, the PMD/Link configuration circuit Config[3:0] shall be set to 0001 (see 15.3.11).

16.3 PMD to PMI interface

The PMD to PMI interface shall comply with the requirements of Clause 15.

16.4 PMD/Medium interface circuits

The PMD sublayer and the MDI interface reference four differential pair circuits, which shall map to the MDI connector pins according to 16.10. These circuits provide data and control input from the media and output to the media.

16.4.1 Circuit TPIO:0

Twisted-pair input/output circuit zero (TPIO:0) shall be a differential pair consisting of TPIO:0+ and TPIO:0–.

16.4.2 Circuit TPIO:1

Twisted-pair input/output circuit one (TPIO:1) shall be a differential pair consisting of TPIO:1+ and TPIO:1–.

16.4.3 Circuit TPIO:2

Twisted-pair input/output circuit two (TPIO:2) shall be a differential pair consisting of TPIO:2+ and TPIO:2–.

16.4.4 Circuit TPIO:3

Twisted-pair input/output circuit three (TPIO:3) shall be a differential pair consisting of TPIO:3+ and TPIO:3–.

16.5 PMD transmit functions

The PMD transmit functions include the transmission of both data and control signals. Data transmission requires the encoding and transmission of data present on the MII circuits TData[3:0]. Control signal transmission also requires generation of the control signals prior to encoding and transmission.

16.5.1 Data and control signal encoding

Non-Return-to-Zero (NRZ) encoding shall be used for the transmission of data and control signals. A value of 1 shall be indicated by a positive differential voltage, and a value of 0 shall be indicated by a negative differential voltage. A positive differential voltage exists when TPIO:x+ is more positive than TPIO:x-. A negative differential voltage exists when TPIO:x- is more positive than TPIO:x+.

16.5.2 Data signaling

Data on the MII circuits TData[3:0] shall be NRZ encoded by the PMD prior to being transmitted simultaneously at 30 MBd on each of the four link channels, resulting in a total signaling rate of 120 MBd. Data on circuits TData[3:0] shall be transmitted synchronized to TxClk on circuits TPIO[3:0] according to Table 16-1 and the requirements of 16.8.2, whenever TxEn is asserted and control signaling has been completed according to 16.5.4. The steady-state delay between data being present on TData[3:0] and the same data being transmitted on TPIO[3:0] shall be no less than 2 BT and no more than 8 BT.

Table 16-1—Mapping of TData[3:0] circuits to TPIO[3:0] circuits

MII circuit	Circuit for data transmission
TData[0]	TPIO:3
TData[1]	TPIO:2
TData[2]	TPIO:1
TData[3]	TPIO:0

16.5.3 Control signaling

A combination of control signals transmitted on circuits TPIO:3 and TPIO:2 is used to communicate the control state over the link between a repeater and an end node or between cascaded repeaters. Transmit control signal combinations shall be controlled by the state of the TCS[2:0] circuits at the MII. Control signals shall be transmitted synchronized to TxClk.

16.5.3.1 Control signal generation

The 4-UTP PMD shall be capable of generating the following two control signals and transmitting them on both TPIO:3 and TPIO:2:

a) Control signal 1 (CS1) shall be a continuous repetition of two alternating patterns of 16 bits, <HCS1a> and <HCS1b>. <HCS1a> is the pattern 1111111100000000; <HCS1b> is the pattern 0000000011111111. These patterns shall be NRZ encoded and transmitted at 30 MBd. The fundamental frequency of CS1 is 0.9375 MHz.

b) Control signal 2 (CS2) shall be a continuous repetition of two alternating patterns of 8 bits, <HCS2a> and <HCS2b>. <HCS2a> is the pattern 11110000; <HCS2b> is the pattern 00001111. These patterns shall be NRZ encoded and transmitted at 30 MBd. The fundamental frequency of CS2 is 1.875 MHz.

NOTE—“Half Control Signal” (HCS) is a generic term used to refer to any of the patterns <HCS1a>, <HCS1b>, <HCS2a>, or <HCS2b>.

In addition to generating these signals, the PMD shall disable the transmitters on the appropriate TPIO[3:0] circuits whenever a condition of “silence” is required (see 16.8.2.1).

Whenever RxEn and TxEn are both deasserted, the PMD shall transmit either control signals or “silence” on TPIO[3:2], and maintain “silence” on TPIO[1:0]. Table 16-2 shows the relationship between TCS[2:0] and the state of TPIO[3:2]. (See 14.3.1 and Table 14-1 for the meaning of these control signals.)

Table 16-2—Relationship between Transmit Control States and control signals

TCS[2:0]	TPIO:2 control signal	TPIO:3 control signal
000	disable*	disable*
001	CS1	CS1
010	CS1	CS2
011	CS2	CS1
100	CS2	CS2
101	reserved	reserved
110	reserved	reserved
111	disable*	disable*

*disable PMD transmitters

16.5.3.2 Control signal transitions

After “silence,” CS1 shall begin with an <HCS1a> pattern, and CS2 shall begin with an <HCS2a> pattern.

Transitions between CS1 and CS2 shall be accomplished such that any <HCSxa> (x = 1 or 2) pattern is always followed by an <HCSyb> (y = 1 or 2) pattern and any <HCSxb> (x = 1 or 2) pattern is always followed by an <HCSya> (y = 1 or 2) pattern. Transitions between CS1 and CS2 shall be accomplished such that an equal number of 1’s and 0’s has been transmitted on both TPIO:3 and TPIO:2 when the transition occurs. This requires that an integer number of HCS patterns shall be transmitted on both TPIO:3 and TPIO:2 before the transition occurs.

Transitions between CS1 or CS2 and “silence” shall be accomplished such that an equal number of 1’s and 0’s has been transmitted on both TPIO:3 and TPIO:2 when the transition occurs. This requires that an integer number of HCS patterns shall be transmitted on both TPIO:3 and TPIO:2 before the start of “silence.” In addition, “silence” shall begin simultaneously on TPIO:3 and TPIO:2. Examples of these transitions are shown in Figure 16-3.

While transmitting “silence,” the first bit of a control signal required by any valid value of TCS[2:0] shall be transmitted on TPIO[3:2] within no less than 2 BT and no more than 8 BT of that value of TCS[2:0] being set. While transmitting a control signal, the first bit of the next control signal required by any valid value of TCS[2:0] shall be transmitted on TPIO[3:2] within 20 BT of that value of TCS[2:0] being set. Similarly, “silence” shall occur on TPIO[3:2] within 20 BT of TCS[2:0] being set to 111 or 000.

16.5.4 Transitions between data and control signaling

On assertion of TxEn, the PMD shall continue to transmit control signals until an equal number of 1’s and 0’s has been transmitted on both TPIO:2 and TPIO:3. This requires that an integer number of HCS patterns, as defined in 16.5.3.1, has been transmitted on both TPIO:2 and TPIO:3. The PMD shall then begin to transmit data on circuits TPIO[3:0] as specified in 16.5.2. The transition from control signals to data shall occur simultaneously on TPIO:2 and TPIO:3 within 20 BT of TxEn being asserted. No more than 20 bits of preamble presented on each of TData[3:0] shall be discarded from data transmitted on each of TPIO[3:0]. Examples of the transition between control signals and data transmission are shown in Figure 14-4.

When TxEn is asserted, the PMD shall complete transmission of the current control signals and then transmit no more control signals until TxEn is deasserted. However, the value of TCS[2:0] may be updated by the PMI during this time. The transition from data signaling to control signaling shall occur immediately after the last bit of packet data is transmitted. Packet data transmission shall be completed and no additional data bits shall be added before switching to the appropriate CS pattern or disabling PMD transmitters.

On assertion of RxEn, the PMD shall continue to transmit control signals until an equal number of 1's and 0's has been transmitted on both TPIO:3 and TPIO:2. This requires that an integer number of HCS patterns, as defined in 16.5.3.1, has been transmitted on both TPIO:3 and TPIO:2. The PMD shall then maintain "silence" on circuits TPIO[3:0] irrespective of the value of TCS[2:0]. The transition from control signals to "silence" shall occur simultaneously on TPIO:3 and TPIO:2 within 20 BT of RxEn being asserted. When RxEn is deasserted, the PMD shall resume control signaling on TPIO[3:2] within no less than 2 BT and no more than 8 BT.

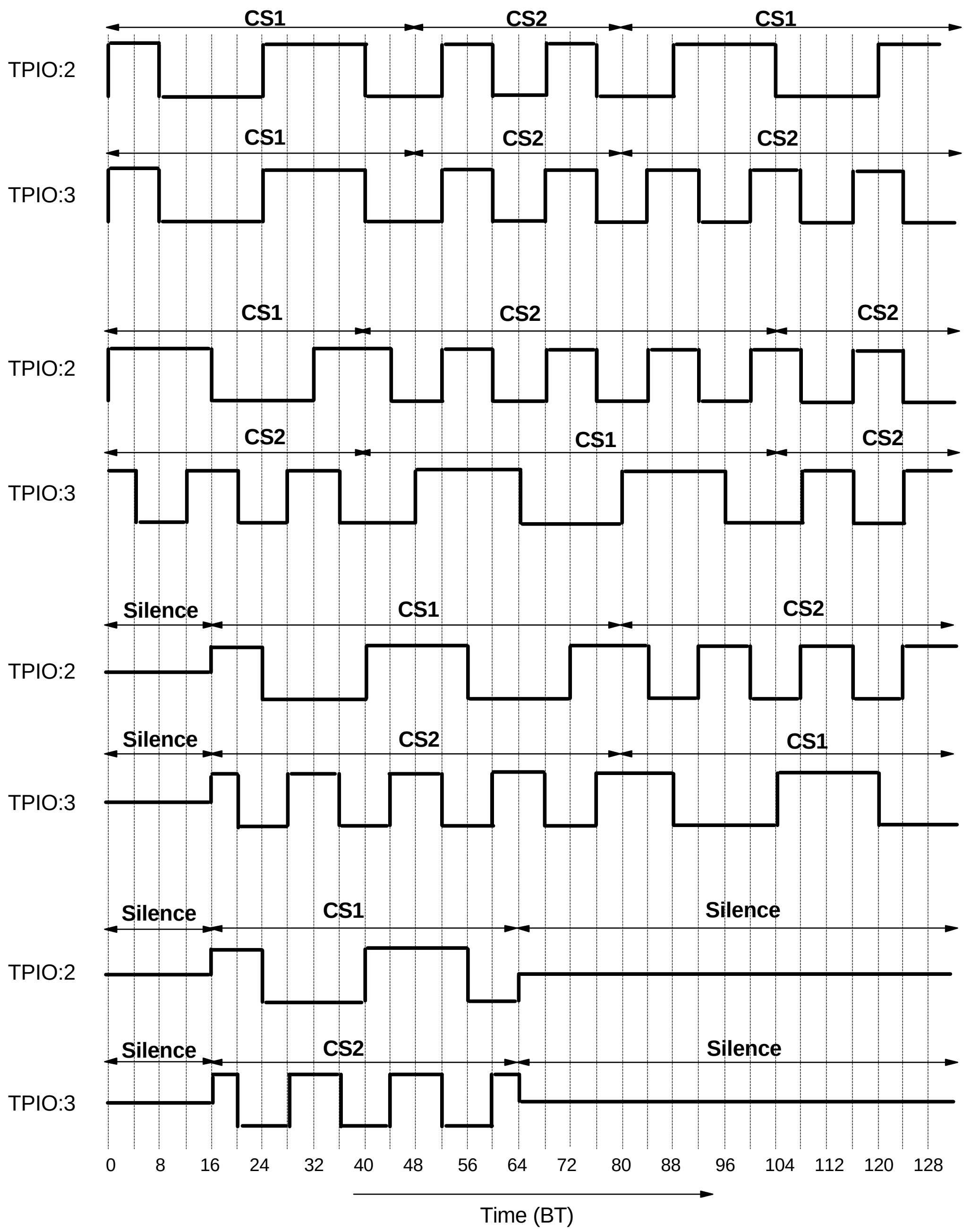


Figure 16-3—Transitions between control signals

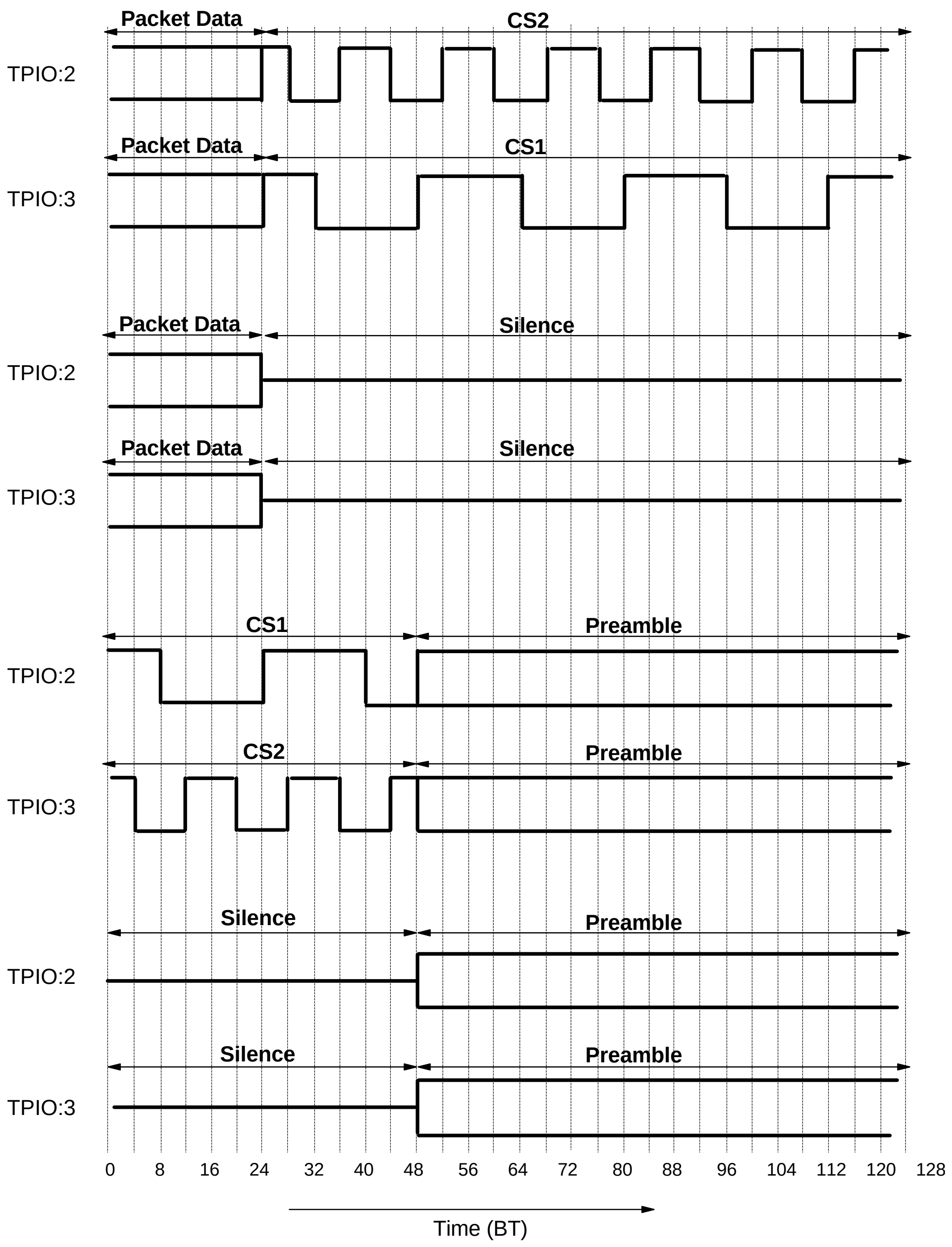


Figure 16-4—Control signal to data transition examples

16.6 Receive function requirements

Receive functions of the PMD include control signal recovery and decoding, control state reporting, preamble detection, clock recovery, and data recovery. The PMD shall receive the link control signals on TPIO:0 and TPIO:1 and from these signals determine the receiver control state. The PMD shall recover data from signals on the TPIO[3:0] circuits of the MDI and send them to the MII RData[3:0] circuits. PMD receive functions are controlled according to 16.6.6.

16.6.1 Control signal recovery and decoding

Control signals shall be recovered from signals on TPIO:1 and TPIO:0, unless “silence” is detected on TPIO:1 or TPIO:0 (see 16.8.3.3). The PMD shall determine the receiver control state from the received control signals according to 16.6.6, and indicate the Receiver Control State on the MII RCS[2:0] circuits. The PMD shall indicate the receiver control state within 48 BT of the first bit of any control signals being present on TPIO[1:0] either after a period of “silence” or immediately after RxEn has been deasserted. The PMD shall detect a change from CS1 within 20 BT of the first bit of CS2 being present on either TPIO:0 or TPIO:1. The PMD shall detect a change from CS2 within 32 BT of the first bit of CS1 being present on either TPIO:0 or TPIO:1. The PMD shall meet these timing requirements when valid, error-free control signals are received.

The PMD shall not detect CS1 on TPIO:0 or TPIO:1 unless a run of between 14 and 18 consecutive equal bits has been received at least once. Similarly, the PMD shall not detect CS2 on TPIO:0 or TPIO:1 unless a run of between 6 and 10 consecutive equal bits has been received at least once. In addition, the PMD shall tolerate at least 3 errored bits per 320 consecutive bits during a control signal without indicating an incorrect control state on circuits RCS[2:0].

16.6.2 Signal monitor function

While RxEn is asserted and RCS is currently set to 101, indicating that data signals have been detected, the PMD shall detect nondata signals on TPIO:0. Nondata signals shall be detected when either CS1 or CS2 is present on TPIO:0. The PMD shall set RCS[2:0] to 000 within 25 BT of the first bit of nondata signal being received on TPIO:0. The PMD shall tolerate at least 2 errored bits during a packet without detecting nondata signals.

16.6.3 Clock recovery

After RxEn is asserted and preamble has been detected, the PMD shall recover the clock of the received data. RxClk shall be deasserted until the clock has been recovered. The PMD shall continue to send the recovered clock to RxClk until RxEn is deasserted.

16.6.4 Data recovery

While RxEn is asserted and the PMD is sending the recovered clock to the RxClk circuit, data signals shall be recovered from circuits TPIO[3:0] and sent to the MII circuits RData[3:0] synchronized to RxClk, according to Table 16-3 and the requirements of 16.8.3.3. All data received on TPIO[3:0] shall be sent to RData[3:0], apart from a maximum of 24 bits of preamble on each of TPIO[3:0], which the PMD may discard during clock recovery. After the PMD has recovered clock, the steady-state delay between data being received on TPIO[3:0] and being sent to circuits RData[3:0] shall be less than 10 BT. Furthermore, the PMD shall add no more than 2 BT of differential delay while sending the data received on TPIO[3:0] to RData[3:0].

Table 16-3—Mapping of TPIO[3:0] circuits to RData[3:0] circuits

Circuit from which data is recovered	MII circuit
TPIO:0	RData0
TPIO:1	RData1
TPIO:2	RData2
TPIO:3	RData3

16.6.5 Grant detection

Subject to the requirements of 16.8.3.3 and 16.6.6, the PMD shall assert the MII RxGrant circuit no more than 6 BT after “silence” exists simultaneously on both TPIO:0 and TPIO:1. The RxGrant circuit shall be deasserted no more than 6.5 BT after “silence” does not exist on TPIO:0 and/or TPIO:1.

16.6.6 Receiver Control State (RCS) description

The RCS shall be indicated to the PMI on MII circuits RCS[2:0] and RxGrant. The RCS is controlled by the control signals recovered from circuits TPIO[1:0], and the MII circuits (TxEn and RxEn.) Upon power-up, the PMD shall set circuits RCS[2:0] to 000 and deassert the RxGrant circuit. While neither RxEn nor TxEn are asserted, the PMD shall indicate the RCS on RCS[2:0] and RxGrant. During this time, if Grant is detected or if valid control signals cannot be recovered on TPIO[1:0], the PMD shall maintain the existing value of RCS[2:0] for a period up to “fault_time.” If Grant is detected for a period longer than “fault_time,” or if valid control signals cannot be recovered on TPIO[1:0] for a period longer than “fault_time,” then RCS[2:0] shall be set to 111. RCS[2:0] shall then be maintained at 111 until a valid combination of control signals has been detected on TPIO[1:0]. The period “fault_time” shall be between 1 and 2 ms. Table 16-4 shows the relationship between the control signals on TPIO[1:0] and RCS[2:0] states.

While RxEn is asserted, RxGrant shall be deasserted, but the PMD shall continue to indicate the RCS on RCS[2:0] until preamble is detected on TPIO[3:0]. RCS[2:0] shall then be set to 101. If, subsequent to preamble being detected, RXEN is deasserted or nondata signals are detected on TPIO[3:0], RCS shall be set to 000.

RCS[2:0] shall be set to 111 on the:

a) Second of any two consecutive transitions of TxEn from deassert to assert when Grant has not been detected at the time the TxEn transition occurs, or
b) Third of any three consecutive transitions of TxEn from deassert to assert between which no valid control signal combinations (CS1, CS2) have been detected on TPIO[1:0].

Otherwise, the PMD shall maintain the existing value of RCS[2:0] and deassert RxGrant while TxEn is asserted. Whenever TxEn is deasserted, RCS[2:0] shall initially be set to 000. The PMD shall not attempt to recover control signals on TPIO[1:0] for a period “blind_receiver_time” after TxEn is deasserted and shall also deassert RxGrant during this period. The value of “blind_receiver_time” shall be 8 BT.

Table 16-4—Relationship between TPIO[1:0] and RCS[2:0]

TPIO:0 control signal	TPIO:1 control signal	RCS[2:0]
Mode transition*	Mode transition*	000
CS1	CS1	001
CS2	CS1	010
CS1	CS2	011
CS2	CS2	100
Data on link*	Data on link*	101
Reserved	Reserved	110
Link Warning*	Link Warning*	111

*See Figure 16-8 for detailed information on these codes.

16.7 PMD state diagrams

The state diagrams shown in Figures 16-5 through 16-8 depict the operation of the PMD functions relative to the status of the MII and MDI circuits. The state diagrams for the transmit functions on channels 2 and 3 are identical and a single channel is depicted in Figures 16-5 and 16-6. The state diagrams for the transmit functions on channels 0 and 1 are identical and a single channel is depicted in Figure 16-7. The notation used in the state diagrams follows the conventions of Clause 5. The variables and timers used in the state diagrams are defined in 16.7.1 and 16.7.2.

16.7.1 State diagram variables

Variables are used in the state diagrams to: indicate the status of PMD inputs and outputs, control PMD operation, and pass state information between functions.

In the variable definitions, the name of the variable is followed by a brief definition of the variable and a list of values the variable may take. For those variables that are state diagram outputs, one value will be identified as the default. The variable has the default value when no active state contains a term assigning a different value.

For example, the variable “RCS” has the value of “000” whenever the Receive Function is in a state that asserts “RCS = 000”. Otherwise, this variable has the default value of “update.”

The following variables are used in the PMD state diagrams:

Bal2n3. Indicates the balance between 1’s and 0’s transmitted on TPIO:2 and TPIO:3.

Values:	true;	during control signaling, an integer number of HCS patterns has been transmitted simultaneously on TPIO:2 and TPIO:3
	false;	non-integer number of HCS patterns has been transmitted on TPIO:2 and/or TPIO:3.

BusyLink. Count of the number of successive occurrences of energy on the link at the time when TxEn is asserted.

Values: non-negative integers.

clock. Indicates the results of the Clock Recovery function.

Values:	true;	PMD has recovered a valid clock.
	false;	PMD has not recovered a valid clock.

data_sent. Indicates that the last bit of a packet has been transmitted on TPIO[3:0].

Values:	true:	the last bit of the current packet has been transmitted.
	false:	the last bit of the current packet has not been transmitted.

Grant. Controls the value of the MII circuit RxGrant.

Values:	update;	Grant Detection function sets RxGrant to reflect current received signals (default).
	true;	Grant Detection function asserts RxGrant regardless of current received signals.
	false;	Grant Detection function deasserts RxGrant regardless of current received signals.

RCS. Controls the value of the MII circuits RCS[2:0].

Values:	update;	Control Signal Recovery and Decoding function updates RCS[2:0] whenever valid control signal combinations are decoded. If valid control signal combinations are not decoded, the last value of RCS is maintained (default).
	last_known;	Control Signal Recovery and Decoding function maintains the most recent value of RCS[2:0].
	binary 000 - 111;	Control Signal Recovery and Decoding function sets RCS[2:0] to binary value.

RData. Controls the status of the MII circuits RData[3:0].

Values:	inactive;	RData[3:0] are high impedance, as defined in Clause 15 (default).
	TPIO;	data from TPIO[3:0] are decoded and placed on RData[3:0] by the Data Recovery function.
	deassert;	RData[3:0] are deasserted.

RxClk. Controls the status of the MII circuit RxClk.

Values:	inactive;	RxClk is high impedance, as defined in Clause 15 (default).
	active;	the clock recovered from data on TPIO[3:0] is placed on RxClk by the Clock Recovery function.
	deassert;	RxClk is deasserted.

RxEn. Indicates the status of the signal received by the PMD on the RxEn circuit.

Values:	true;	RxEn is asserted.
	false;	RxEn is deasserted.

send. Indicates the particular control signal to be transmitted on either TPIO:2 or TPIO:3.

Values:	CS1 or CS2;	the value of “send” is determined by the entry for TPIO:2 or TPIO:3 in Table 16-2.
	disable;	TCS=111 or TCS=000 (default).

signal_type. Indicates the type of signal the PMD has detected at its inputs.

Values:	preamble;	preamble has been detected.
	not_data;	nondata signals have been detected (see 16.6.2).

other; neither preamble nor nondata patterns have been detected.

SilentGap. Count of the number of successive occurrences of no valid control signal combinations detected on TPIO[1:0] during gap between transmitted packets.

Values: non-negative integers.

TCS. Indicates the status of the MII circuits TCS[2:0].

Values: binary 000 through 111.

TPIO[i]. Controls the status of the MDI circuit TPIO:i (I=0, 1, 2, or 3).

Values:	disable;	no energy is transmitted, but signals may be received, on TPIO:i (default).
	Tdata[z];	data on MII circuit TData[z] is transmitted on TPIO:i by the Data Signaling function (z = 0, 1, 2, or 3).
	HCS1a;	an <HCS1a> pattern is transmitted on TPIO[i].
	HCS1b;	an <HCS1b> pattern is transmitted on TPIO[i].
	HCS2a;	an <HCS2a> pattern is transmitted on TPIO[i].
	HCS2b;	an <HCS2b> pattern is transmitted on TPIO[i].

TxDisable. Indicates when the PMI has requested the transmitters on TPIO[3:0] to be disabled.

Values:	true;	TCS[2:0] has the value of 000 or 111.
	false;	TCS[2:0] does not have the value of 000 and does not have the value of 111.

TxEn. Indicates the status of the signal received by the PMD on the TxEn circuit.

Values:	true;	TxEn is asserted.
	false;	TxEn is deasserted.

valid_CS. Indicates that a valid combination of control signals on TPIO[0:1] has been detected by the Control Recovery function.

Values:	true;	a valid control signal combination has been detected. Grant is not a valid control signal in this context.
	False;	a valid control signal combination has not been detected. Grant is not a valid control signal in this context.

16.7.2 State diagram timers

A timer is reset and starts counting upon entering a state where “start x_timer” is asserted. Time “x_time” after the timer has been started, “x_timer_done” is asserted and remains asserted until the timer is reset. At all other times, “x_timer_not_done” is asserted.

When entering a state in which “start x_timer” is asserted, the timer is reset and restarted even if the entered state is the same as the exited state. For example, when in the Control Decode state of the Receive Functions state diagram, the “fault_timer” is reset each time the term “valid_CS” is satisfied.

blind_receiver_timer. Timer for delay between end of transmitted packet by the Data Signaling function and resumption of control signal decoding by the Control Signal Recovery and Decoding function (see 16.6.6).

fault_timer. Timer for delay between last valid control signal combination detected and RCS being set to indicate a Link Warning. Valid control signals are CS1 and CS2. Grant is not a valid control signal in this context (see 16.6.6).

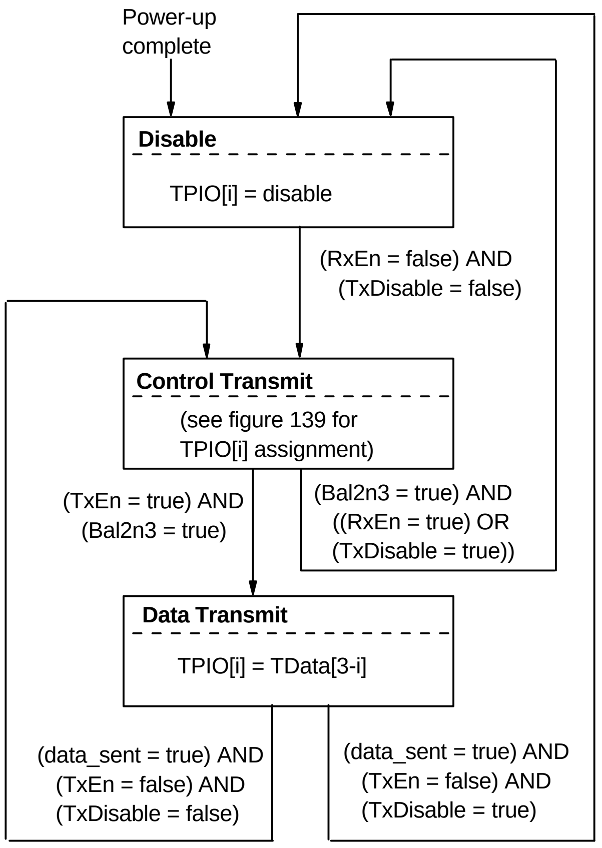

Power-up complete
Disable
TPIO[i] = disable
(RxEn = false) AND (TxDisable = false)
Control Transmit
(see figure 139 for TPIO[i] assignment)
(TxEn = true) AND (Bal2n3 = true)
(Bal2n3 = true) AND ((RxEn = true) OR (TxDisable = true))
Data Transmit
TPIO[i] = TData[3-i]
(data_sent = true) AND (TxEn = false) AND (TxDisable = false)
(data_sent = true) AND (TxEn = false) AND (TxDisable = true)

Figure 16-5—State diagram for transmit functions on TPIO[3:2]

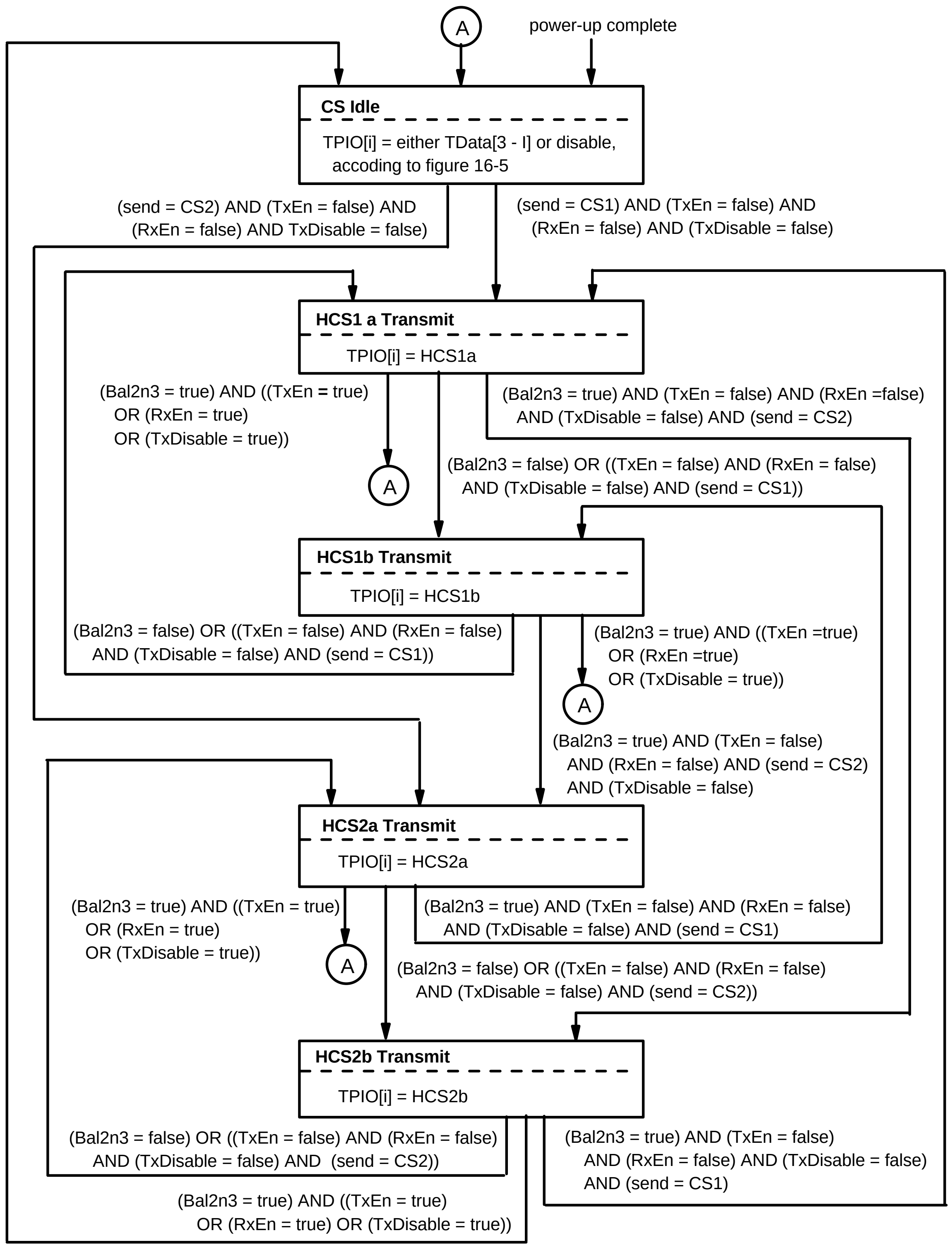

A
power-up complete
CS Idle
TPIO[i] = either TData[3 - I] or disable, accoding to figure 16-5
(send = CS2) AND (TxEn = false) AND (RxEn = false) AND TxDisable = false)
(send = CS1) AND (TxEn = false) AND (RxEn = false) AND (TxDisable = false)
HCS1 a Transmit
TPIO[i] = HCS1a
(Bal2n3 = true) AND ((TxEn = true) OR (RxEn = true) OR (TxDisable = true))
A
(Bal2n3 = true) AND (TxEn = false) AND (RxEn =false) AND (TxDisable = false) AND (send = CS2)
(Bal2n3 = false) OR ((TxEn = false) AND (RxEn = false) AND (TxDisable = false) AND (send = CS1))
HCS1b Transmit
TPIO[i] = HCS1b
(Bal2n3 = false) OR ((TxEn = false) AND (RxEn = false) AND (TxDisable = false) AND (send = CS1))
(Bal2n3 = true) AND ((TxEn =true) OR (RxEn =true) OR (TxDisable = true))
A
(Bal2n3 = true) AND (TxEn = false) AND (RxEn = false) AND (send = CS2) AND (TxDisable = false)
HCS2a Transmit
TPIO[i] = HCS2a
(Bal2n3 = true) AND ((TxEn = true) OR (RxEn = true) OR (TxDisable = true))
A
(Bal2n3 = true) AND (TxEn = false) AND (RxEn = false) AND (TxDisable = false) AND (send = CS1)
(Bal2n3 = false) OR ((TxEn = false) AND (RxEn = false) AND (TxDisable = false) AND (send = CS2))
HCS2b Transmit
TPIO[i] = HCS2b
(Bal2n3 = false) OR ((TxEn = false) AND (RxEn = false) AND (TxDisable = false) AND (send = CS2))
(Bal2n3 = true) AND (TxEn = false) AND (RxEn = false) AND (TxDisable = false) AND (send = CS1)
(Bal2n3 = true) AND ((TxEn = true) OR (RxEn = true) OR (TxDisable = true))

Figure 16-6—State diagram for control signaling on TPIO[3:2]

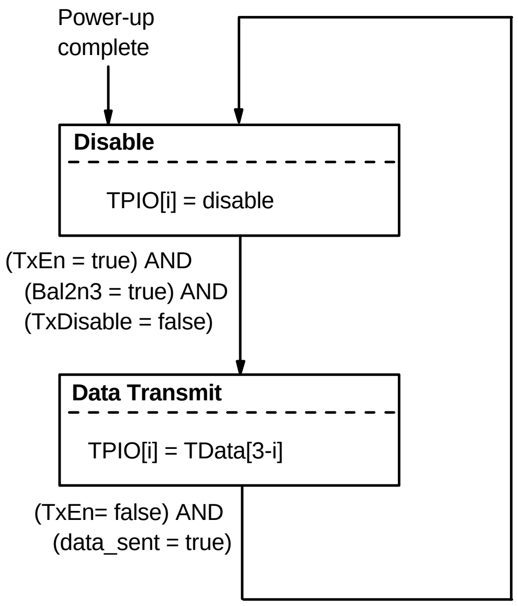


Figure 16-7—State diagram for transmit functions on TPIO[1:0]

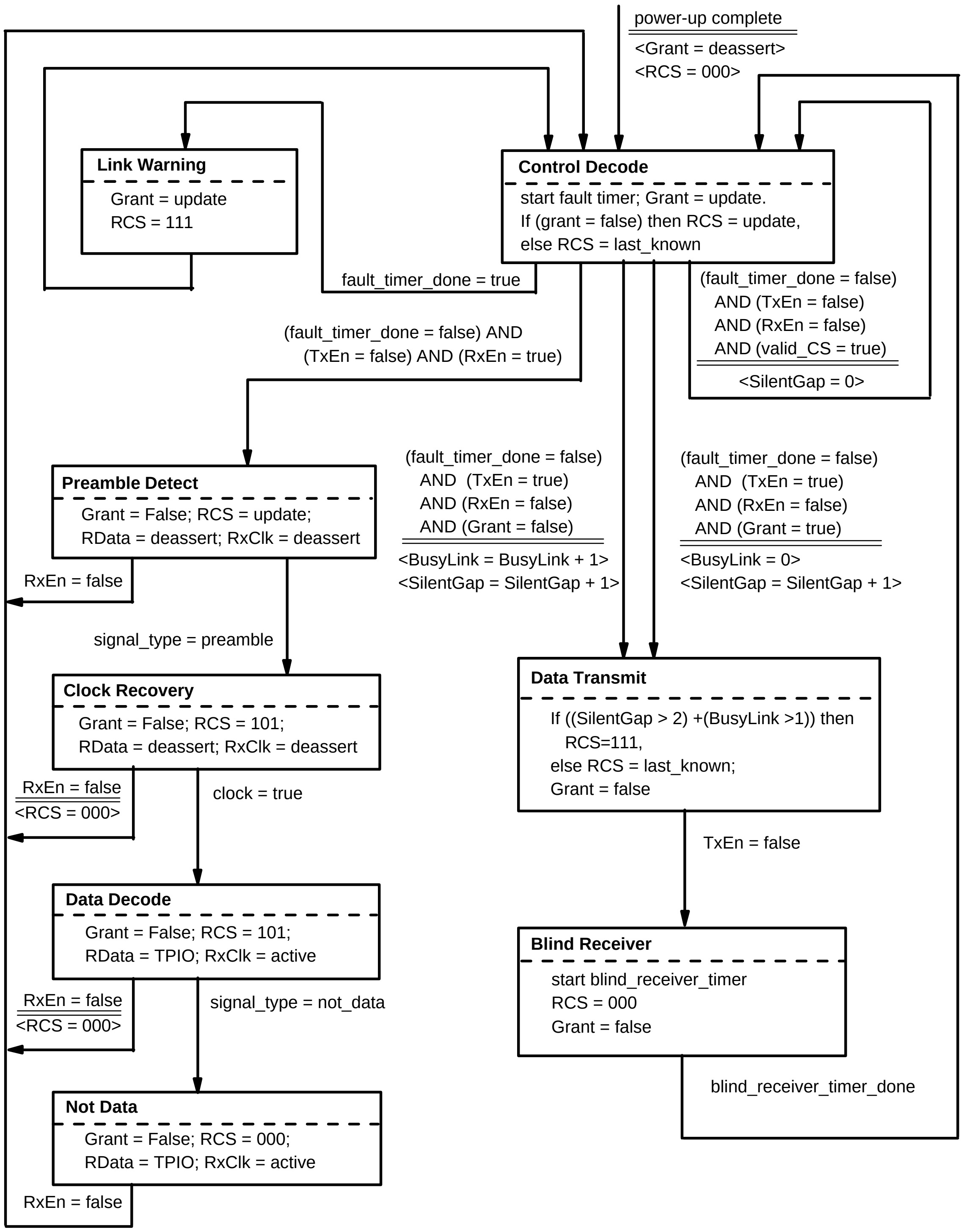

power-up complete
<Grant = deassert>
<RCS = 000>
Link Warning
Grant = update
RCS = 111
Control Decode
start fault timer; Grant = update.
If (grant = false) then RCS = update,
else RCS = last_known
fault_timer_done = true
(fault_timer_done = false) AND (TxEn = false) AND (RxEn = true)
(fault_timer_done = false) AND (TxEn = false) AND (RxEn = false) AND (valid_CS = true)
<SilentGap = 0>
Preamble Detect
Grant = False; RCS = update;
RData = deassert; RxClk = deassert
(fault_timer_done = false) AND (TxEn = true) AND (RxEn = false) AND (Grant = false)
<BusyLink = BusyLink + 1>
<SilentGap = SilentGap + 1>
(fault_timer_done = false) AND (TxEn = true) AND (RxEn = false) AND (Grant = true)
<BusyLink = 0>
<SilentGap = SilentGap + 1>
RxEn = false
signal_type = preamble
Clock Recovery
Grant = False; RCS = 101;
RData = deassert; RxClk = deassert
Data Transmit
If ((SilentGap > 2) +(BusyLink >1)) then
RCS=111,
else RCS = last_known;
Grant = false
RxEn = false
<RCS = 000>
clock = true
TxEn = false
Data Decode
Grant = False; RCS = 101;
RData = TPIO; RxClk = active
Blind Receiver
start blind_receiver_timer
RCS = 000
Grant = false
RxEn = false
<RCS = 000>
signal_type = not_data
blind_receiver_timer_done
Not Data
Grant = False; RCS = 000;
RData = TPIO; RxClk = active
RxEn = false

Figure 16-8—State diagram for receive functions

16.8 PMD electrical specifications

This subclause defines the electrical characteristics of the PMD at the MDI interface. Although the electrical characteristics are specified for a single channel of the twisted-pair link, all four channels of the link shall conform to this specification simultaneously.

16.8.1 Isolation requirement

The PMD shall provide isolation between the end node PMI sublayer circuits and the MDI. This electrical separation shall withstand at least one of the following electrical strength tests:

a) 1500 Vrms at 50–60 Hz for 60 s, applied as specified in Section 5.3.2 of IEC 60950:1991.
b) 2250 Vdc for 60 s, applied as specified in Section 5.3.2 of IEC 60950:1991.
c) A sequence of ten 2400 V impulses of alternating polarity, applied at intervals of not less than 1 s. The shape of the impulses shall be 1.2/50 µs (1.2 µs virtual front time, 50 µs virtual time of half value), as defined in IEC 60950:1991.

There shall be no insulation breakdown, as defined in Section 5.3.2 of IEC 60950:1991, during the test. The resistance after the test shall be at least 2 MΩ, measured at 500 Vdc.

16.8.2 Transmitter specifications

The PMD shall provide the transmit functions specified in 16.5 in accordance with the electrical specifications of this subclause.

Unless otherwise specified, the requirements of this subclause shall be met by each of the four PMD transmitters. Where a test load is not specified, the transmitters shall meet the requirements of 16.8.2 when connected to a 100 Ω (± 0.1%) resistive load.

16.8.2.1 Differential output voltage

Some of the text and figures of this subclause describe the differential voltage in terms of magnitudes. The requirements of this subclause apply to negative as well as positive pulses.

The steady-state differential voltage, V_{out}, on the TPIO[3:0] circuits when terminated with the test load at the MDI connector shall be between 2.2 V and 2.8 V. This shall be measured midway between zero-crossing transitions on the following six steady-state-patterns: CS1, CS2, and a repeating sequence of n “ones” followed by n “zeros” for n = 6, 5, 4, 3. Figure 16-9 shows the timing for this measurement for the CS1 signal. In addition, the steady-state differential voltage on TPIO[3:0], measured midway between zero-crossing transitions, shall be between 2.2 V and 3.5 V for repeating patterns of n “ones” followed by n “zeros” for n = 2 and n = 1.

When transmitting the repetitive CS1 signal, the transmitter’s output waveform shall not enter the shaded areas shown in Figure 16-9. The rise and fall times for this waveform shall be 20 ± 2 ns, and the duty cycle shall be 50% ± 2%. The rise and fall times shall be measured as the time taken for the signal to transition between the normalized amplitudes of 0.8 and –0.8 shown in Figure 16-9.

NOTE—Figure 16-9 illustrates the allowable droop and undershoot/overshoot.

While the TPIO[3:0] circuits are transmitting a repetitive pattern of 1111100000, the rms power at each frequency shall be less than that shown in Table 16-5. The power shall be measured in a 10 kHz bandwidth.

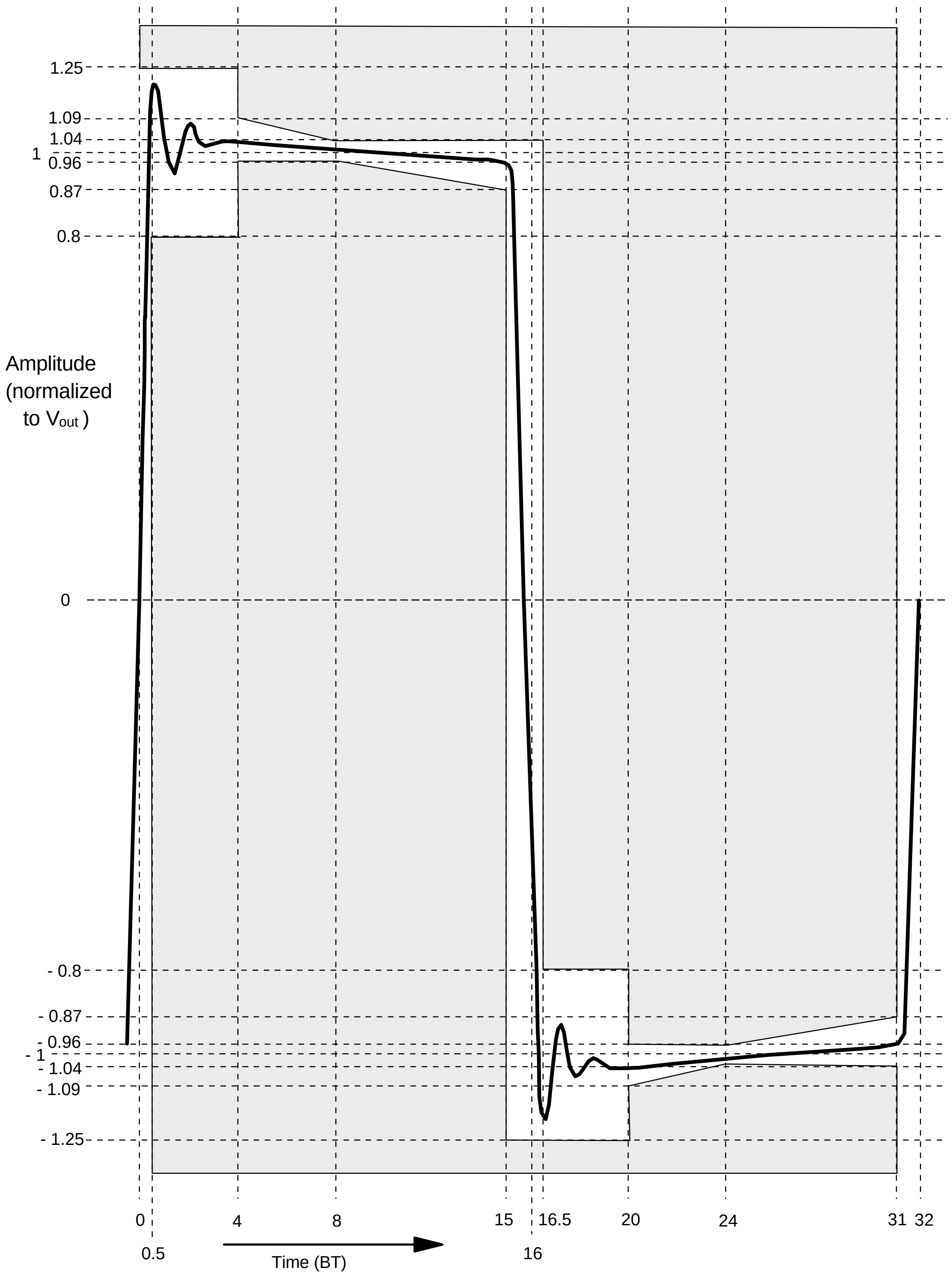


Figure 16-9—Voltage template for transmitter output waveform

Table 16-5—Transmitter frequency rolloff

Frequency (MHz)	Power (dB normalized to power at 15 MHz)
15	0
21	–5
27	–16
33	–25

NOTE—The specification of transmitter output waveform is not intended to ensure compliance with regulations concerning RF emissions. The implementor should consider any applicable local, national, or international regulations.

When a PMD TPIO transmitter is disabled, the amplitude of any signal on the TPIO circuit shall be less than 50 mV peak to peak.

16.8.2.2 Transmitter output jitter

For all valid sequences of packet data applied to TData[3:0], the resulting zero-crossing jitter at the MDI shall be less than 3 ns peak to peak. Jitter shall be measured with TxClk as reference.

16.8.2.3 Transmitter differential output impedance

When the PMD is powered, the differential output impedance as measured on the TPIO[3:0] circuits shall be such that any reflection, due to differential signals incident on a TPIO[3:0] circuit from its corresponding link segment channel, shall be at least 15 dB below the incident level. This shall be true for all frequencies between 1 MHz and 15 MHz and for all valid transmitter operating conditions.

16.8.2.4 Transmitter impedance balance

The common-mode to differential-mode impedance balance of the TPIO[3:0] circuits shall be greater than:

$$29 - 17 \log_{10}\left(\frac{f}{10}\right) \text{ dB} \tag{8}$$

where f is the frequency in megahertz over the frequency range 1–30 MHz.

This balance is defined as:

$$\text{Impedance balance} = 20 \log_{10}\left(\frac{E_{cm}}{E_{diff}}\right) \tag{9}$$

where E_{cm} is an externally applied sine wave voltage as shown in Figure 16-10
E_{diff} is measured as shown in Figure 16-10.

NOTE—The balance of the test equipment (such as the matching of the 147 Ω resistors) used to measure the transmitter impedance balance should exceed that required of the transmitter by 20 dB.

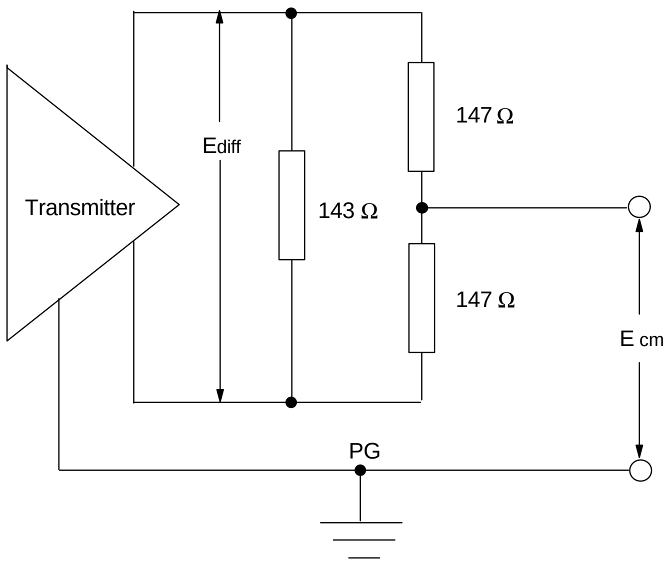


Figure 16-10—Transmitter impedance balance test circuit

16.8.2.5 Transmitter fault tolerance

Transmitters, when either idle or non-idle, shall remain fully functional after the application of short circuits across any of the TPIO[3:0] circuits for an indefinite period of time. The magnitude of any resulting short circuit current shall be less than 300 mA. The TPIO[3:0] circuits shall resume normal operation after such faults.

Transmitters, when either idle, non-idle, or not powered, shall remain fully functional after the application of a 1000 V common-mode impulse applied at E_{cm} of either polarity, as indicated in Figure 16-11. The shape of the impulse shall be 0.3/50 µs (300 ns virtual front time, 50 µs virtual time of half value), as defined in IEC 60060-1:1989

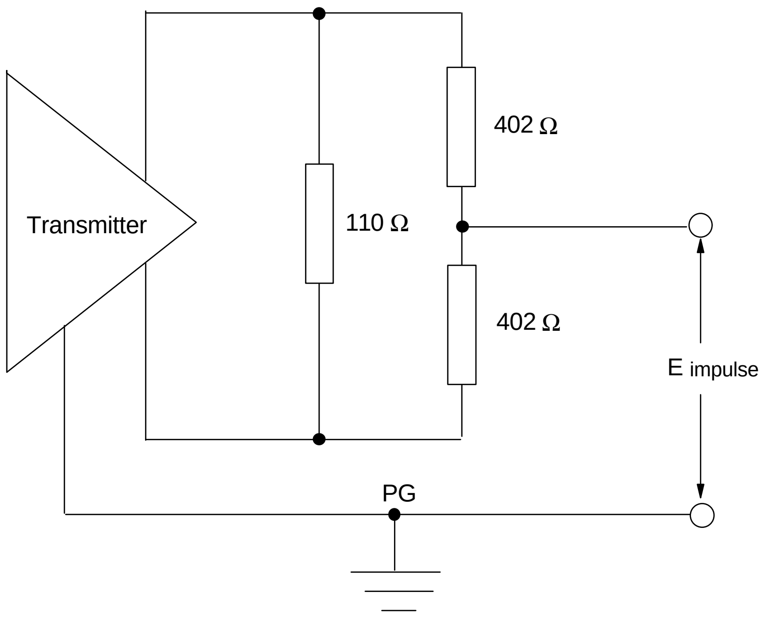


Figure 16-11—Common-mode impulse test

16.8.3 Receiver specifications

The PMD shall provide the receive functions specified in 16.6 in accordance with the electrical specifications of this subclause.

16.8.3.1 Receiver performance testing

When specified, receiver performance shall be tested using the signals output from a test link. The test link is intended to represent a 4-UTP link channel. The test link shall comprise a transmitter compliant with the requirements of this clause, coupled to a twisted-pair channel with frequency dependent loss as specified in Table 16-6. The scaling factor, k, in Table 16-6 represents varying cable length. Worst-case testing is normally performed when k = 1. However, receiver performance shall be compliant with the requirements of 16.8.3 for all values of k between 0.01 and 1.

Table 16-6—Test link frequency dependent loss

Frequency (MHz)	Loss (dB)
0.256	$k \times 1.3 \pm 0.5$
0.512	$k \times 1.8 \pm 0.5$
0.772	$k \times 2.2 \pm 0.5$
1	$k \times 2.6 \pm 0.5$
4	$k \times 5.6 \pm 0.5$
10	$k \times 9.8 \pm 0.5$
15	$k \times 12.5 \pm 0.5$
16	$k \times 13.1 \pm 0.5$

NOTE—The frequency dependent loss required may be obtained by choosing an appropriate length of Category 3 100 Ω balanced cable.

The test link shall also include frequency independent loss such that when a preamble signal is transmitted, the peak amplitude measured at the output of the test link is less than $1.3 \times 10^{-(k \times 12.5 + 1.5)/20} \times 2.2$ V. This amplitude shall be measured with the test link terminated in a 100 ± 0.1 Ω resistive load.

16.8.3.2 Receiver noise immunity

In addition to valid data or control signals, noise due to internal cable crosstalk or external sources may also be present at the input to the receiver. When required, the immunity of the receiver to these noise sources shall be tested by deliberately adding noise to the test link specified in 16.8.3.1. The circuit shown in Figure 16-12 shall be used to add differential mode sinusoidal noise signals with frequency and amplitude as specified in Table 16-7. The circuit shown in Figure 16-13 shall be used to add common mode sinusoidal noise signals with frequency and amplitude as specified in Table 16-7.

NOTE—It is not a requirement that both differential and common mode noise signals are added to the test link simultaneously.

Table 16-7—Noise signal amplitudes

Frequency (MHz)	Differential mode noise peak amplitude (V_{n_diff})	Common mode noise peak amplitude (V_{n_comm})
1	0.33	25
5.1	0.33	16.4
13.1	0.29	6.4
15.1	0.30	6.0
19.1	0.33	5.3
21	0.38	5.6

16.8.3.3 Receiver differential input signals

When presented with control signals simultaneously on both TPIO:1 and TPIO:0 output from the test link specified in 16.8.3.1, the receiver shall recover and correctly decode the control signals according to the requirements of 16.6.1. The receiver shall also recover and correctly decode the control signals when noise is added to the test link as specified in 16.8.3.2. This requirement shall be met for all values of differential delay permitted by 16.9.1.3 between control signals on TPIO:0 and TPIO:1.

A condition of “silence” shall be detected for all signals (data, control, or noise) which when measured at the output of the following filter would have a magnitude less than 330 mV for longer than 4 BT. Silence shall not be detected for any signals which when measured at the output of the following filter would have a magnitude greater than 630 mV for longer than 1 BT. See Figure 16-14. The filter shall have the character-istics of a fifth order Butterworth low-pass filter with a 3 dB cutoff at 20 MHz.

The receiver shall recover all valid packet data on TPIO[3:0] output from the test link specified in 16.8.3.1 and pass this data to the MII circuits RData[3:0] with a BER of less than 1 error in 10^8 bits. This BER shall also be achieved when noise is added to the test link as specified in 16.8.3.2.

16.8.3.4 Receiver differential input impedance

The differential input impedance shall be such that any reflection due to differential signals incident upon the TPIO[3:0] circuits from a twisted pair having any impedance within the range specified in 16.9.1.2 shall be at least 15 dB below the incident over the frequency range of 1–15 MHz.

NOTE—The requirements of 16.8.2.1 and 16.8.3.3 may also constrain the allowable input impedance of a receiver.

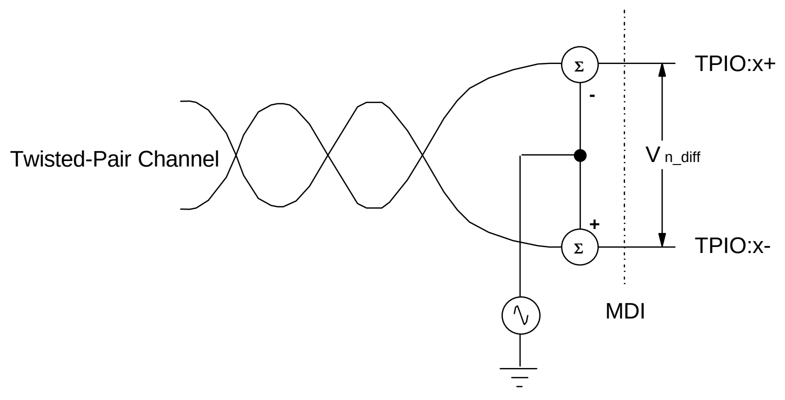


Figure 16-12—Differential-mode noise test circuit

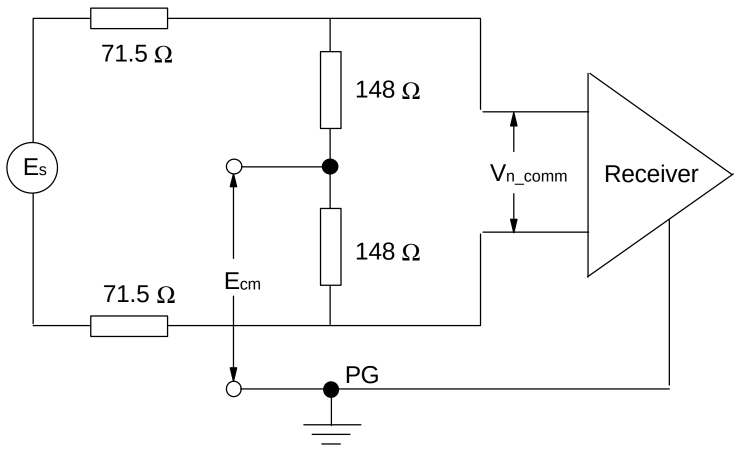


Figure 16-13—Common-mode noise test circuit

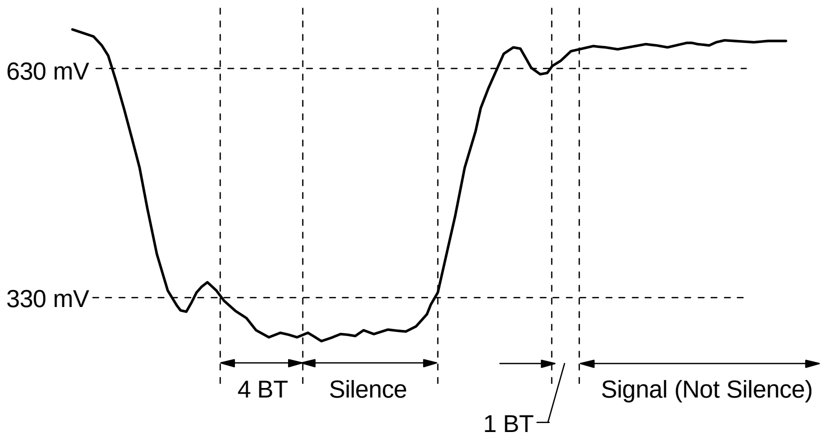


Figure 16-14—Silence thresholds

16.8.3.5 Receiver impedance balance

The common-mode to differential-mode impedance balance of the TPIO[3:0] circuits shall be:

$$\text{cm} - \text{diff imped bal} > 29 - 17\log_{10}\left(\frac{f}{10}\right)\ \text{dB} \tag{10}$$

where cm – diff imped bal is the common-mode to differential-mode impedance balance
f is the frequency in megahertz over the frequency range 1–30 MHz.

This balance is defined as:

$$\text{Impedance balance} = 20\log_{10}\left(\frac{E_{cm}}{E_{diff}}\right) \tag{11}$$

where E_{cm} is an externally applied sine wave voltage as shown in Figure 16-15
E_{diff} is measured as shown in Figure 16-15.

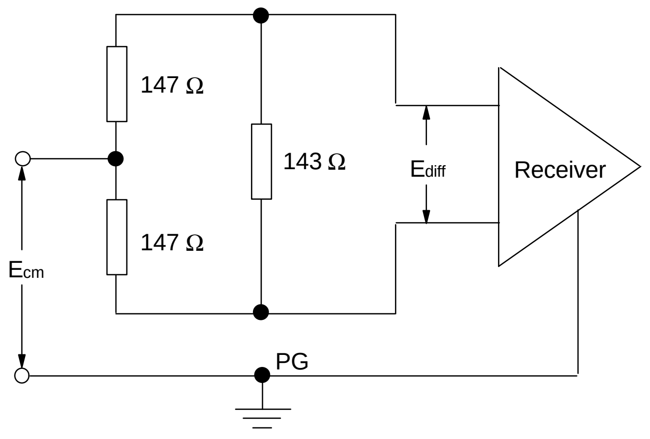


Figure 16-15—Receiver impedance balance test circuit

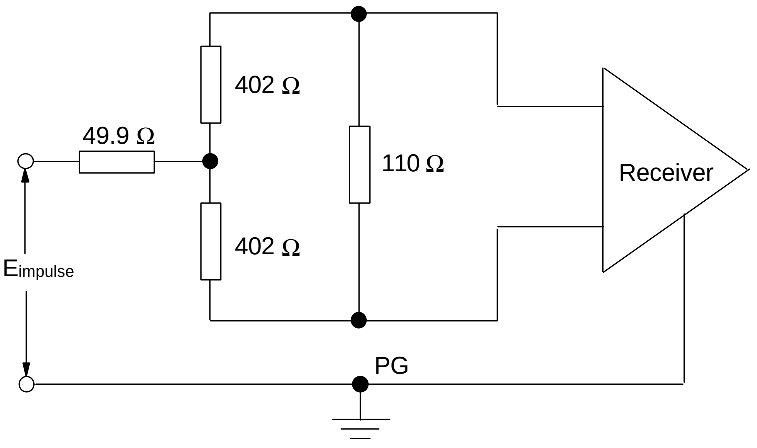


Figure 16-16—Receiver common-mode impulse test

NOTE—The balance of the test equipment (such as the matching of the 147 Ω resistors) used to measure the receiver impedance balance should exceed that required of the transmitter by 20 dB.

16.8.3.6 Receiver fault tolerance

The receiver shall tolerate the application of short circuits between the leads of the TPIO[3:0] circuits for an indefinite period of time without damage and shall resume normal operation after such faults are removed. Receivers shall withstand without damage a 1000 V common-mode impulse of either polarity ($E_{impulse}$ as shown in Figure 16-16). The shape of the impulse shall be 0.3/50 µs (300 ns virtual front time, 50 µs virtual time of half value), as defined in IEC 60060-1:1989.

16.9 Characteristics of the twisted-pair link segment

4-UTP PMD link segments consist of the link medium, two end-connectors, and connection devices such as patch panels and wall plates, as shown in Figure 16-2.

The medium for the 4-UTP PMD is 4-pair balanced cable. When the store-and-forward function is implemented in the repeater, repeater-to-end node link segments may use four pairs from a 25-pair bundle. Otherwise, 4-pair cable shall be used, and all media specifications and installation requirements shall comply with ISO/IEC 11801:1995 requirements for 100 Ω balanced Category 3 cable or 100 Ω balanced class-C links in addition to the provisions of this clause.

16.9.1 Transmission parameters

Each channel of a link segment shall have the following characteristics. (These characteristics are generally met by twisted-pair links compliant with ISO/IEC 8802-3 Type 10BASE-T specifications.) Except where specifically noted, each channel of a link segment shall be tested with source and load impedances of 100 Ω.

16.9.1.1 Insertion loss

The insertion loss of a channel shall be no greater than 14 dB at all frequencies between 100 kHz and 15 MHz. This insertion loss value includes attenuation of the twisted pairs, connector losses, and reflection losses due to impedance mismatches. This specification shall be met when the channel is terminated in source and load impedances that satisfy 16.8.2.3 and 16.8.3.4.

NOTE—Multipair polyvinyl chloride (PVC) insulated 0.5 mm [24 AWG] cable typically exhibits an attenuation at 15 MHz of between 12 dB and 13 dB per 100 m at 20 °C. The loss of PVC insulated cable exhibits significant temperature dependence, and therefore it may be necessary to use a less temperature-dependent cable at temperatures greater than 40 °C.

16.9.1.2 Differential characteristic impedance

The magnitude of the input impedance of each channel of a link segment, terminated with 100 Ω, averaged over the 1–15 MHz frequency band, shall be between 85 Ω and 115 Ω.

16.9.1.3 Delay

The maximum propagation delay of each channel of a twisted-pair link segment shall be 5.7 ns/m. The maximum differential delay between any two channels of a link segment shall be 67 ns. The total one-way propagation delay of each channel of a 100 Ω balanced cable link segment shall be less than 1.2 µs.

16.9.2 Differential Near-End Crosstalk (NEXT) loss (25-pair bundles)

Near-end crosstalk may occur between repeater ports during simultaneous reception and transmission of individually addressed packets when 4-UTP link segments are formed from subsets of a 25-pair bundle. To avoid excessive crosstalk between link segments, the NEXT loss is specified for twisted pairs in a 25-pair bundle that are used for demand-priority link segments. NEXT loss shall be measured with the far ends of both the disturbed and disturbing pairs terminated with 100 Ω and the near end of the disturbed pair terminated with 100 Ω. The source of the disturbing signal shall have an impedance of 100 Ω. The NEXT loss between any two twisted pairs in a 25-pair bundle or binder group used for demand-priority link segments shall be at least:

$$\text{pair-pair NEXT loss} = 30 - 15\log_{10}\left(\frac{f}{10}\right)\quad \text{dB} \tag{12}$$

where f is the frequency in megahertz over the frequency range 1–15 MHz.

16.9.3 Noise environment

The noise environment generally consists of crosstalk from other demand-priority channels, or externally induced impulse noise (typically from other office and building equipment). The noise level present on the link segments shall not cause the error rate to exceed the objective specified in 16.8.3.3.

16.9.3.1 Impulse noise

The average rate of occurrence of impulses greater than 264 mV shall be $\leq$ 0.2 impulses per second, as measured at the output of a 5-pole Butterworth low-pass filter with a 3 dB cutoff at 20 MHz. Following the start of any particular impulse that is counted, any additional impulse shall be ignored for a period of 1 μs. The link segment shall be terminated at the far end with 100 Ω.

NOTE—Typically, the impulse noise occurrence rate changes inversely by one decade for each 5–9 dB change in the threshold voltage. If a count rate of N counts/s is measured on a specific twisted pair and filter at the specified voltage threshold, the media noise margin is:

$$\text{media noise margin} = \text{approx } 7 \log_{10}\left(\frac{0.2}{N}\right) \text{ dB} \tag{13}$$

Impulse noise may be a burst phenomenon and should be measured over an extended period of time.

16.9.3.2 Crosstalk noise (25-pair bundles)

The crosstalk noise on each channel of a link segment formed from twisted pairs in a 25-pair bundle shall be measured while random packet data is transmitted on all four channels of any one other link segment at the near end of the same 25-pair bundle, with the maximum transmit level as specified in 16.8.2.1. In addition, random packet data shall be transmitted on the three remaining channels of the disturbed link segment, contradirectional with the disturbing link segment. The transmit level of the three far end disturbers shall be such that when a preamble signal is transmitted, the peak amplitude measured at the output of each of the three channels is less than $1.3 \times 10^{-(k \text{ X } 12.5 + 1.5)/20} \times 2.2$ V. The peak crosstalk noise on the disturbed channel at the MDI shall be ≤ 330 mV when measured at the output of a fifth-order Butterworth low-pass filter with a 3 dB cutoff at 20 MHz.

16.10 MDI specification

This clause defines the electrical and mechanical specifications for the MDI connectors used for the 4-UTP system.

16.10.1 MDI connectors

Eight-pin modular connectors, meeting the requirements of Section 3 and Figures 1 through 5 of ISO/IEC 8877:1992, shall be used as the mechanical interface to the twisted-pair link segment. Figure 16-17a) shows the modular connector that shall be used on the end node and the repeater MDI interface.

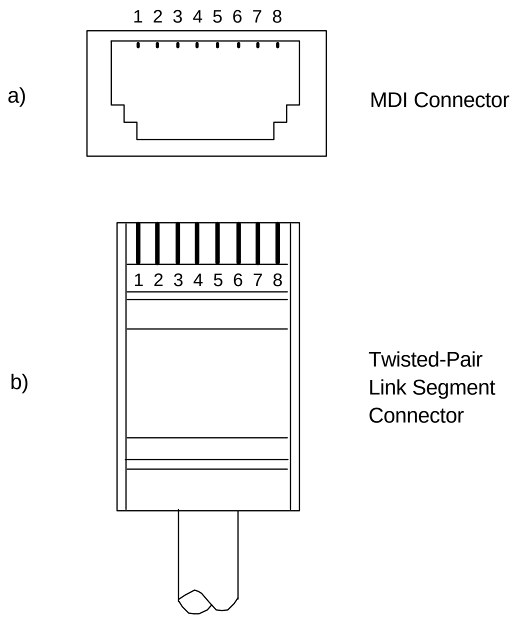


Figure 16-17—MDI and twisted-pair link segment connectors

16.10.2 MDI connector pin assignments

Table 16-8 shows the assignment of circuits to each MDI connector contact. Note that a crossover is implemented in the case of a repeater downlink. This crossover facilitates transmission of control signals on circuits TPIO:3 and TPIO:2, and reception of control signals on circuits TPIO:0 and TPIO:1, respectively, by PMDs at both ends of a link simultaneously. (No internal crossover function is required in a repeater cascade port as it functions similar to an end node and is always connected to an upper-level repeater local port.) Figure 16-18 illustrates the crossover function in the repeater local ports.

Table 16-8—MDI connector pin assignments

MDI connector contact	MDI circuit end node and repeater uplink	MDI circuit repeater downlink
1	TPIO:0+	TPIO:3+
2	TPIO:0–	TPIO:3–
3	TPIO:1+	TPIO:2+
4	TPIO:2+	TPIO:1+
5	TPIO:2–	TPIO:1–
6	TPIO:1–	TPIO:2–
7	TPIO:3+	TPIO:0+
8	TPIO:3–	TPIO:0–

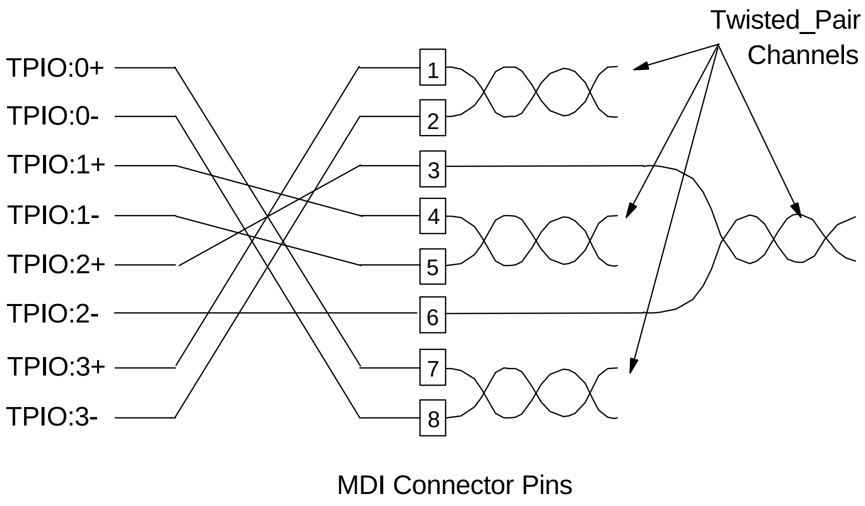


Figure 16-18—Repeater local port (downlink) internal crossover function

16.10.3 Link segment connectors

Figure 16-17b) shows the modular connector that shall be used on the twisted-pair link segment. When the medium is four-pair UTP cable, connector configurations and pin assignments shall comply with ISO/IEC 11801:1995, Clause 8. See Figure 16-19. When the medium is a 4-pair cable or part of a 25-pair bundle, the link segment connector pins shall be assigned to twisted pairs such that no crossover occurs in the link segment.

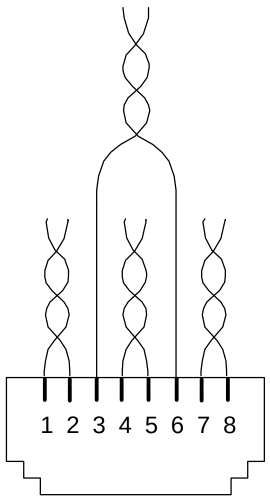


Figure 16-19—Link segment modular connector

17. Dual simplex STP Physical Medium Dependent (PMD) sublayer, Medium Dependent Interface (MDI), and link specifications

17.1 Scope

This clause defines the functional characteristics of the PMD sublayer and provides electrical and mechanical specifications for the MDI and media for 150 Ω shielded balanced cable link implementation. The relationship of the PMD specification to the LAN model is shown in Figure 17-1. No particular implementation of the PMD is implied. General environmental and safety specifications are given in Annex B.

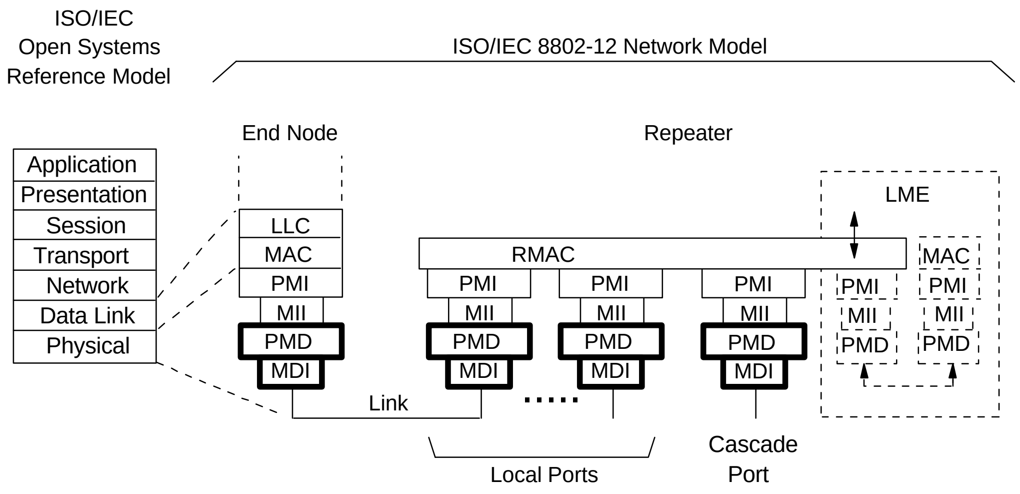


Figure 17-1—PMD and MDI relationship to the LAN model

17.2 Overview

The STP PMD layer has the following general characteristics:

a) Enables coupling of the PMI sublayer and the MII to two-pair 150 Ω shielded balanced cable link segments as defined in 17.9. (See Figure 17-2.)

b) Supports data packet traffic at a signaling rate of 120 MBd.

c) Allows link segment lengths of 0–100 m between an end node and a repeater, and between repeaters.

d) Supports network configurations using the demand-priority access method with baseband signaling on two channels.

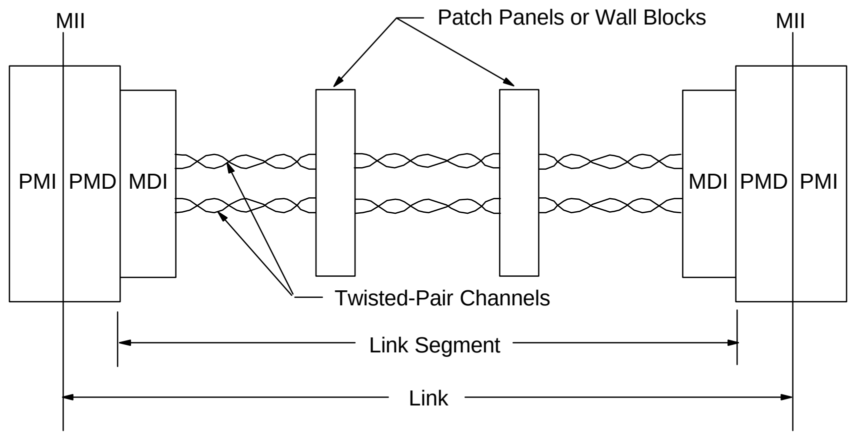


Figure 17-2—STP link

17.2.1 Signal conditioning and control functions

The PMD sublayer provides transmit and receive functions to control the link status and to facilitate packet flow between the end node or repeater and the twisted-pair link segment. The PMD is the medium dependent connection between the MII and the network link segment. It:

a) Generates and transmits outgoing control signals.
b) Transmits outgoing data.
c) Recognizes incoming control signals and differentiates them from incoming data.
d) Recognizes data packet preambles and recovers the receive clock (RxClk) from the incoming data streams.
e) Recognizes and recovers incoming data streams.
f) Multiplexes and demultiplexes packet data.

17.2.2 PMD interfaces

The PMD has two defined interfaces: one to the PMI sublayer, and one to the link segment. The PMD may be implemented as a separate unit, with a physically exposed MII to the PMI, or it may be logically integrated with the PMI. The interface to the link segment (MDI) shall always be physically exposed.

17.2.3 Twisted-pair medium

The STP PMD/MDI is intended for operation over links up to 100 m of 150 Ω balanced cable and connecting hardware as specified in ISO/IEC 11801:1995.

17.2.4 PMD/Link configuration

When implemented, the PMD/Link configuration circuit Config[3:0] shall be set to 1011 (see 15.3.11).

17.3 PMD to PMI interface

The PMD to PMI interface shall comply with the logical signal and timing requirements defined for the MII in Clause 15.

17.4 Functional description of interchange circuits

The PMD to MII circuits are described in Clause 15. The PMD sublayer and the MDI interface both reference two differential pair circuits.

17.4.1 Circuit Tx+/-

Tx+/- is an output differential pair consisting of Tx+ and Tx-.

17.4.2 Circuit Rx+/-

Rx+/- is an input differential pair consisting of Rx+ and Rx-.

17.5 PMD transmit functions

The PMD transmit functions include the transmission of both data and control signals. Data transmission requires the multiplexing, encoding, and transmission of data present on the MII circuits TData[3:0]. Control signal transmission also requires generation of the control signals prior to encoding and transmission.

17.5.1 Multiplexing physical frames

The PMD shall multiplex the data on the MII circuits TData[3:0] prior to transmission on a link segment as a single serialized data stream. The PMD shall multiplex TData[3:0] so that the physical frame structure at the MDI is:

<mpreamble><mssd>< S_1><S_2><S_3>........<S_n>< mesd><F>

where

a) <mpreamble> shall be a stream of alternating 0’s and 1’s, terminating in a one, of length from 192 bits to 192 bits minus the maximum number of preamble bits that may be discarded by the PMD (see 17.5.5).

b) <mssd> shall be the multiplexed start of stream delimiters on the MII circuits TData[3:0].

   1) For a normal-priority packet <mssd> shall be:

      111100 111100 111100 111100 000011 000011 000011 000011

   2) For a high-priority packet <mssd> shall be:

      100000 100000 100000 100000 111110 111110 111110 111110

   The leftmost bit of <mssd> as written above shall be transmitted first at the MDI.

c) <S_m> is the mth code sextet to be transmitted (see 14.4 and Figure 14-8). Complete code sextets on the MII circuits TData[3:0] shall be multiplexed in a cyclic manner to form a serialized bit stream. <S_1> shall be the first code sextet on TData[0], <S_2> shall be the first code sextet on TData[1], <S_3> shall be the first code sextet on TData[2], <S_4> shall be the first code sextet on TData[3], <S_5> shall be the second code sextet on Tdata[0], and so on. This cyclic multiplexing of complete code sextets shall continue until the last code sextet has been multiplexed.

   Code sextets shall be transmitted serially in the bit order received on the MII circuits TData[3:0].

d) <mesd> shall be the sextet multiplexed end of stream delimiters received on the MII circuits TData[3:0]. Four <mesd> sequences are possible due to the four possible TData[3:0] end combinations (see Figure 14-9).

Let x be the number of octets modulo 5 in the MAC frame.

1) If x = 0 or 2, <mesd> shall be:
<esd$_{01}$><esd$_{11}$>< esd$_{21}$ ><esd$_{31}$><esd$_{02}$><esd$_{12}$><esd$_{22}$><esd$_{32}$><esd$_{03}$><esd$_{13}$>
<esd$_{23}$><esd$_{33}$>

2) If x = 1, <mesd> shall be:
<esd$_{21}$><esd$_{31}$><esd$_{01}$><esd$_{11}$><esd$_{22}$><esd$_{32}$><esd$_{02}$><esd$_{12}$><esd$_{23}$><esd$_{33}$>
<esd$_{03}$><esd$_{13}$>

3) If x = 3, <mesd> shall be:
<esd$_{11}$><esd$_{21}$><esd$_{31}$><esd$_{01}$><esd$_{12}$><esd$_{22}$><esd$_{32}$><esd$_{02}$><esd$_{13}$><esd$_{23}$>
<esd$_{33}$><esd$_{03}$>

4) If x = 4, <mesd> shall be:
<esd$_{31}$><esd$_{01}$><esd$_{11}$><esd$_{21}$><esd$_{32}$><esd$_{02}$><esd$_{12}$><esd$_{22}$><esd$_{33}$><esd$_{03}$>
<esd$_{13}$><esd$_{23}$>

<esd$_{jk}$> shall be the kth sextet of the esd associated with the channel j (see 14.4.2.3.5). Although the four cases of <mesd> patterns are shown with reference to x, the number of octets modulo 5 in the MAC frame, the PMI does not need to count octets, quintets, or sextets (see 14.4.2.3.5, 14.4.2.3.6, and Figure 14-9).

e) <F> are end fill bits as required. The value of F does not affect the receiver operation. The actual value chosen is an implementation issue and is beyond the scope of this standard.

17.5.2 Data and control signal encoding

NRZ encoding shall be used for the transmission of data and control signals across the link medium. A value of 1 shall be indicated by a positive differential voltage (Tx+ is more positive than Tx-) and a value of 0 shall be indicated by a negative differential voltage.

17.5.3 Data signaling function

Data on the MII circuits TData[3:0] is multiplexed according to 17.5.1 and transmitted serially on the Tx+/- circuits, resulting in a signaling rate of 120 MBd on the link medium. The steady-state delay between data being present on TData[3:0] and the same data being transmitted on Tx+/- shall be no less than 8 BT and no more than 12 BT.

17.5.4 Control signals

Control signals transmitted on the STP medium are used to communicate the link state between a repeater and an end node or between cascaded repeaters. Control signals shall be controlled by the state of the TCS[2:0] circuits of the MII. Control signals shall be transmitted on the Tx+/- differential circuit.

17.5.4.1 Control signal generation

The PMD shall be capable of generating the following five control signals and transmitting them on Tx+/-:

a) Control signal 1 (CS1) is a continuous repetition of two alternating patterns, <HCS1a> and <HCS1b>.

1) <HCS1a> is the pattern:

1111 1111 1111 1111 0000 0000 0000 0000.

2) <HCS1b> is the pattern:

0000 0000 0000 0000 1111 1111 1111 1111.

These patterns shall be NRZ encoded and transmitted at 120 MBd. The fundamental frequency of CS1 is 1.875 MHz.

b) Control signal 2 (CS2) is a continuous repetition of two alternating patterns, <HCS2a> and <HCS2b>.

1) <HCS2a> is the pattern:

1111 1111 1111 1100 0000 0000 0000.

2) <HCS2b> is the pattern:

0000 0000 0000 0001 1111 1111 1111 11.

These patterns shall be NRZ encoded and transmitted at 120 MBd. The fundamental frequency of CS2 is 2.069 MHz.

c) Control signal 3 (CS3) is a continuous repetition of two alternating patterns, <HCS3a> and <HCS3b>.

1) <HCS3a> is the pattern:

1111 1111 1111 1000 0000 0000 00.

2) <HCS3b> is the pattern:

0000 0000 0000 0111 1111 1111 11.

These patterns shall be NRZ encoded and transmitted at 120 MBd. The fundamental frequency of CS3 is 2.308 MHz.

d) Control signal 4 (CS4) is a continuous repetition of two alternating patterns, <HCS4a> and <HCS4b>.

1) 1) <HCS4a> is the pattern:

1111 1111 1110 0000 0000 00.

2) <HCS4b> is the pattern:

0000 0000 0000 1111 1111 1111.

These patterns shall be NRZ encoded and transmitted at 120 MBd. The fundamental frequency of CS4 is 2.609 MHz.

e) Control signal 5 (CS5) is a continuous repetition of two alternating patterns, <HCS5a> and <HCS5b>.

1) <HCS5a> is the pattern:

1111 1111 1100 0000 0000.

2) <HCS5b> is the pattern:

0000 0000 0011 1111 1111.

These patterns shall be NRZ encoded and transmitted at 120 MBd. The fundamental frequency of CS5 is 3.000 MHz.

NOTE—Half Control Signal (HCS) is a generic term used to refer to any of the patterns <HCSxa>, <HCSxb> (x=1, 2, 3, 4 or 5).

Whenever TxEn is deasserted, the PMD shall transmit a control signal on Tx+/- or shall disable Tx+/-. Table 17-1 shows the relationship between TCS[2:0] and the state of Tx+/-.

Table 17-1—Relationship between transmit control states and control signals

TCS[2:0]	Tx+/- control signal
000	disable*
001	CS3
010	CS2
011	CS4
100	CS1
101	reserved
110	reserved
111	CS5

*disable PMD transmitter

17.5.4.2 Control signal transitions

At the start of control signaling, the first control signal transmitted shall begin with a <HCSxa> pattern (x=1, 2, 3, 4 or 5).

Transitions between any two different control signals shall be accomplished such that any <HCSxa> pattern (x=1, 2, 3, 4 or 5) is always followed by an <HCSya> pattern ($y = 1, 2, 3, 4$ or 5, $y \neq x$) , or any <HCSxb> pattern (x=1, 2, 3, 4 or 5) is always followed by an <HCSyb> pattern ($y = 1, 2, 3, 4$ or 5, $y \neq x$). Transitions between any two control signals shall be accomplished such that an equal number of 1's and 0's has been transmitted on Tx+/- when the transition occurs. This requires that an integer number of HCS patterns shall be transmitted on Tx+/- before the transition.

NOTE—The first bit of a new control signal is never equal to the last bit of the previous control signal.

When the PMD is required to disable the transmitter on Tx+/-, this shall be accomplished such that an equal number of 1's and 0's has been transmitted on Tx+/- before the transmitter is disabled. This requires that an integer number of HCS patterns shall be transmitted on Tx+/- before the transmitter is disabled.

While transmitting a control signal, the first bit of the next control signal required by any valid value of TCS[2:0] shall be transmitted on Tx+/- within 20 BT of that value of TCS[2:0] being set. The PMD transmitter shall be disabled within 20 BT of TCS[2:0] being set to 000.

17.5.5 Transitions between data and control signaling

On assertion of TxEn, the PMD shall continue to transmit the current control signal until an equal number of 1's and 0's has been transmitted on Tx+/-. This requires that an integer number of HCS patterns, as defined in 17.5.4.1, has been transmitted on Tx+/-. The PMD shall then begin to transmit data on Tx+/-, as specified in 17.5.3. The transition from control signal to data shall occur within 20 BT of TxEn being asserted. No more than 40 bits of preamble shall be discarded from the serialized data stream transmitted on Tx+/-.

When TxEn is asserted, the PMD shall complete transmission of the current control signals and then transmit no more control signals until TxEn is deasserted. However, the value of TCS[2:0] may be updated by the PMI during this time. The transition from data signaling to control signaling shall occur immediately after the last bit of packet data is transmitted. Packet data transmission shall be completed and no additional data bits shall be added before switching to the appropriate HCS pattern or disabling PMD transmitters. The first HCS pattern transmitted after the completion of data transmission shall be an <HCSxa> pattern (x=1, 2, 3, 4 or 5).

17.5.6 Transmit Control State (TCS) description

The TCS shall be controlled by the TCS[2:0] and TxEn circuits at the MII. The dual simplex PMD either transmits data or CSx on the Tx+/- circuits. When TxEn is asserted, the PMD shall transmit data on the Tx+/- circuits.

17.6 Receive function requirements

Receive functions of the PMD include control signal recovery and decoding, preamble detect, clock recovery, and data recovery. The PMD shall receive the link control signals on Rx+/- and from these control signals determine the receiver control state. The PMD shall recover packet data from signals on the Rx+/- circuits of the MDI and send them to the MII RData[3:0] circuit. PMD receive functions are controlled according to 17.6.7.

17.6.1 Control signal recovery and decoding function

The PMD shall have the capability to recover control signals from the signal on Rx+/-, subject to the requirements of 17.8.3.1. The PMD shall determine the receiver control state from the received control signals according to 17.6.7, and indicate the receiver control state on the MII RCS[2:0] circuits within 24 BT of any change in the control signals present on Rx+/-. The PMD shall tolerate at least 2 errored bits per 320 consecutive bits during a control signal without indicating an incorrect control state on circuits RCS[2:0].

17.6.2 Signal monitor function

While RxEn is asserted and the RCS is currently set to 101, indicating that data signals have been detected, the PMD shall detect nondata signals on Rx+/-. Nondata signals shall be detected when any valid control signal is present on Rx+/-. The PMD shall set RCS[2:0] to 000 within 25 BT of the first bit of nondata signal being received on Rx+/-. The PMD shall tolerate at least 2 errored bits during a packet without detecting nondata signals.

17.6.3 Clock recovery function

The PMD shall recover the symbol clock from the data received on Rx+/-. The PMD shall detect preamble and send the recovered clock divided down in frequency by a factor of 4 to the MII RxClk circuit. The PMD shall continue to send the clock to RxClk until RxEn is deasserted.

17.6.4 Data recovery function

Serial data signals shall be recovered from circuit Rx+/- and shall be sent to the MII circuits RData[3:0], following demultiplexing according to 17.6.5. The steady-state delay between data being present on Rx+/- and the same data being sent to RData[3:0] shall be no less than 8 BT and no more than 12 BT. The differential delay between the first bit of any two of <ssd$_{k1}$> (where k = 0, 1, 2 or 3) being sent to the appropriate RData[k] circuits shall be no more than 8 BT.

17.6.5 Demultiplexing of physical frame

The PMD shall demultiplex each received packet such that the physical frame structure at the MII shall be:

Channel k:

$$<\text{preamble}><\text{ssd}_{k1}><\text{ssd}_{k2}><S_{k+1}><S_{k+5}><S_{k+9}>..........<\text{esd}_{k1}><\text{esd}_{k2}><\text{esd}_{k3}><x>$$

where k = 0 to 3, <preamble> is at least three consecutive occurrences of the pattern 01, and <x> are end fill bits as required.

The occurrence of <ssd$_{01}$> after <preamble > shall be used by the PMD to guarantee recovery of sextet word boundaries for demultiplexing of the physical frame into the four channels of Rdata[3:0].

17.6.6 Grant detection function

The PMD shall assert the MII RxGrant circuit no more than 15 BT after the first bit of CS5 is present on Rx+/-. The RxGrant circuit shall be deasserted no more than 22 BT after CS5 is no longer present on Rx+/-

17.6.7 Receiver Control State (RCS) description

The RCS shall be indicated to the PMI on MII circuits RCS[2:0] and RxGrant. The RCS is controlled by the control signals recovered from Rx+/-, and the MII circuit RxEn. On power-up, the PMD shall set circuits RCS[2:0] to 000 and deassert the RxGrant circuit. While RxEn is not asserted, the PMD shall indicate the RCS on RCS[2:0] and RxGrant. During this time, if CS5 is detected or if no valid control signal can be recovered on Rx+/-, the PMD shall maintain the existing value of RCS[2:0]. If valid control signals cannot be recovered on Rx+/- for a period longer than “fault_time,” then RCS[2:0] shall be set to 111. RCS[2:0] shall then be maintained at 111 until a valid control signal has been detected on Rx+/-. The period “fault_time,”shall be between 1 and 2 ms. Table 17-2 shows the relationship between the control signal on Rx+/- and RCS[2:0] states.

While RxEn is asserted, RxGrant shall be deasserted, but the PMD shall continue to indicate the RCS on RCS[2:0] until preamble is detected on Rx+/-, when RCS[2:0] shall then be set to 101. If, subsequent to preamble being detected, RxEn is deasserted, or the PMD detects nondata signals on Rx+/- before RxEn is deasserted, RCS[2:0] shall be set to 000.

Whenever TxEn is deasserted, the RCS shall be set to 000 for one cycle of TxClk or until a valid control signal (CS1, CS2, CS3, or CS4) has been detected on RX+/-, whichever is longer.

Table 17-2—Relationship between Rx+/- and RCS[2:0]

Rx+/- control signal	RCS[2:0]
Mode Transition*	000
CS3	001
CS2	010
CS4	011
CS1	100
Data on link*	101
reserved	110
Link Warning*	111

*See Figure 17-4 for detailed information on these codes.

17.7 PMD state diagrams

The state diagrams depict the operation of the PMD functions relative to the status of the MII and MDI circuits. The state diagram for the transmit functions is depicted in Figure 17-3, the state diagram for the receive functions is depicted in Figure 17-4, and the state diagram for the control functions is depicted in Figure 17-5. The notation used in the state diagrams follows the conventions of Clause 5. The variables and timers used in the state diagrams are defined in 17.7.1 and 17.7.2.

17.7.1 State diagram variables

Variables are used in the state diagrams to: indicate the status of PMD inputs and outputs, control PMD operation, and pass state information between functions.

In the variable definitions, the name of the variable is followed by a brief definition of the variable and a list of values the variable may take. For those variables that are state diagram outputs, one value will be identified as the default. The variable has the default value when no active state contains a term assigning a different value.

For example, the variable “RCS” has the value of “000” whenever the Receive Function is in a state that asserts “RCS=000.” Otherwise, this variable has the default value of “update.”

The following variables are used in the PMD state diagrams.

Bal. Indicates the balance between 1’s and 0’s transmitted on Tx+/-.

Values:

true;	during control signaling, an equal number of 1’s and 0’s has been transmitted on Tx+/
false;	during control signaling, an equal number of 1’s and 0’s has not been transmitted on Tx+/-.

clock. Indicates the results of the Clock Recovery function.

Values:

true;	PMD has recovered a valid clock.
false;	PMD has not recovered a valid clock.

current. Indicates the particular control signal (if any) being transmitted on Tx+/-.

Values:

CS1, CS2, CS3, CS4, or CS5;	set at start of each HCS transmission, using “send”
disable;	TCS = 000 (default).

data_sent. Indicates that the last bit of a packet has been transmitted on TX+/-.

Values:

true:	the last bit of the current packet has been transmitted.
False:	the last bit of the current packet has not been transmitted.

Grant. Controls the value of the MII circuit RxGrant.

Values:

update;	Grant Detection function sets RxGrant to reflect current received signals (default).
true;	Grant Detection function asserts RxGrant regardless of current received signals.
false;	Grant Detection function deasserts RxGrant regardless of current received signals.

RCS. Controls the value of the MII circuits RCS[2:0].

Values:

update;	Control Recovery and Decoding function updates RCS[2:0] whenever a valid control signal is decoded. If a valid control signal is not decoded, the last value of RCS is maintained (default).
last_known;	Control Recovery and Decoding function maintains the most recent value of RCS[2:0].
binary 000–111;	Control Recovery function sets RCS[2:0] to binary value.

RData. Controls the status of the MII circuits Rdata[3:0].

Values:

inactive;	RData[3:0] are high impedance, as defined in 15.3.10 (default).
Rx+/-;	data from Rx+/- are demultiplexed, decoded, and placed on RData[3:0] by the Data Recovery function.
deassert;	RData[3:0] are deasserted.

RxClk. Controls the status of the MII circuit RxClk.

Values:

inactive;	RxClk is high impedance, as defined in 15.3.9 (default).
active;	the clock recovered from data on Rx+/- is placed on RxClk by the Clock Recovery function.
deassert;	RxClk is deasserted.

RxEn. Indicates the status of the signal received by the PMD on the RxEn circuit.

Values:

true;	RxEn is asserted.
false;	RxEn is deasserted.

send. Indicates the particular control signal to be transmitted on Tx+/-.

Values:

CS1, CS2, CS3, CS4, or CS5;	the value of “send” is determined by Table 17-1.
disable;	TCS = 000 (default).

signal_type. Indicates the type of signal the PMD has detected at its inputs.

Values:

preamble;	preamble has been detected.
not_data;	nondata signals have been detected (see 17.6.2).
other;	neither preamble nor nondata patterns have been detected.

TCS. Indicates the status of the MII circuits TCS[2:0].

Values: binary 000–111.

TxDisable. Indicates when the PMI has requested the transmitter on Tx+/- to be disabled.

Values:

true; TCS[2:0] has the value of 000.
false; TCS[2:0] does not have the value of 000.

TxEn. Indicates the status of the signal received by the PMD on the TxEn circuit.

Values:

true; TxEn is asserted.
false; TxEn is deasserted.

Tx+/-. Controls the status of the MDI circuit Tx+/-.

Values:

disable;	no energy is transmitted on Tx+/- (default).
CSx;	control signals are transmitted on Tx+/- by the Control Signaling Function.
Data;	data on MII circuit TData[3:0] is multiplexed and transmitted serially on Tx+/- by the Data Signaling function.

valid_CS. Indicates that a valid control signal on Rx+/- has been detected by the Control Recovery and Decoding function.

Values:

true; a valid control signal (CS1, CS2, CS3, CS4 or CS5) has been detected.
false; a valid control signal has not been detected.

17.7.2 State diagram timers

A timer is reset and starts counting upon entering a state where “start x_timer” is asserted. Time “x” after the timer has been started, “x_timer_done” is asserted and remains asserted until the timer is reset. At all other times, “x_timer_not_done” is asserted.

When entering a state in which “start x_timer” is asserted, the timer is reset and restarted even if the entered state is the same as the exited state. For example, when in the Control Decode state of the Receive Functions state diagram, the “fault_timer” is reset each time the term “valid_CS” is satisfied.

fault_timer. Timer for delay between last valid control signal combination detected and RCS being set to indicate a Link Warning. Valid control signals are CS1, CS2, CS3, CS4, and CS5.

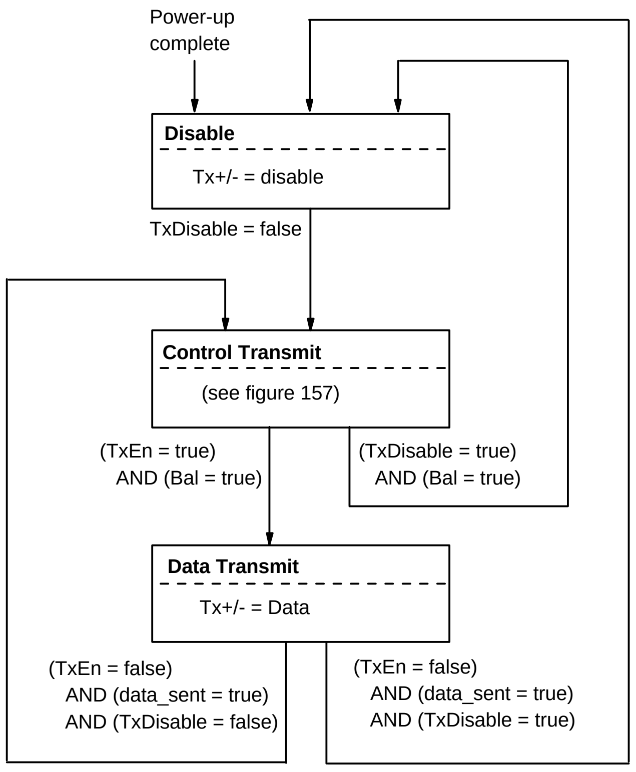

Power-up complete
Disable
Tx+/- = disable
TxDisable = false
Control Transmit
(see figure 157)
(TxEn = true) AND (Bal = true)
(TxDisable = true) AND (Bal = true)
Data Transmit
Tx+/- = Data
(TxEn = false) AND (data_sent = true) AND (TxDisable = false)
(TxEn = false) AND (data_sent = true) AND (TxDisable = true)

Figure 17-3—State diagram for transmit functions

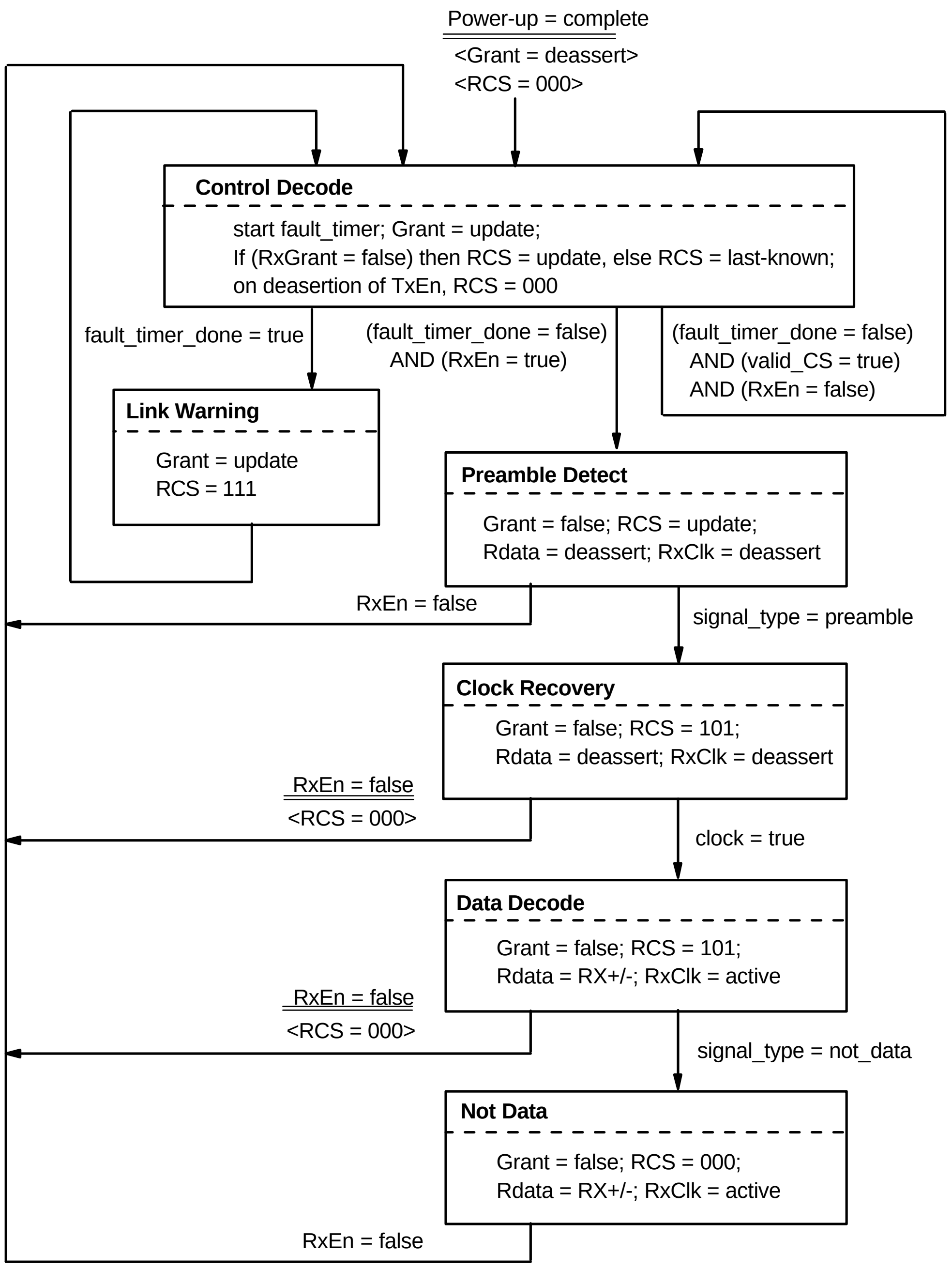

Power-up = complete
<Grant = deassert>
<RCS = 000>
Control Decode
start fault_timer; Grant = update;
If (RxGrant = false) then RCS = update, else RCS = last-known;
on deasertion of TxEn, RCS = 000
fault_timer_done = true
(fault_timer_done = false) AND (RxEn = true)
(fault_timer_done = false) AND (valid_CS = true) AND (RxEn = false)
Link Warning
Grant = update
RCS = 111
Preamble Detect
Grant = false; RCS = update;
Rdata = deassert; RxClk = deassert
RxEn = false
signal_type = preamble
Clock Recovery
Grant = false; RCS = 101;
Rdata = deassert; RxClk = deassert
RxEn = false
<RCS = 000>
clock = true
Data Decode
Grant = false; RCS = 101;
Rdata = RX+/-; RxClk = active
RxEn = false
<RCS = 000>
signal_type = not_data
Not Data
Grant = false; RCS = 000;
Rdata = RX+/-; RxClk = active
RxEn = false

Figure 17-4—State diagram for receive functions

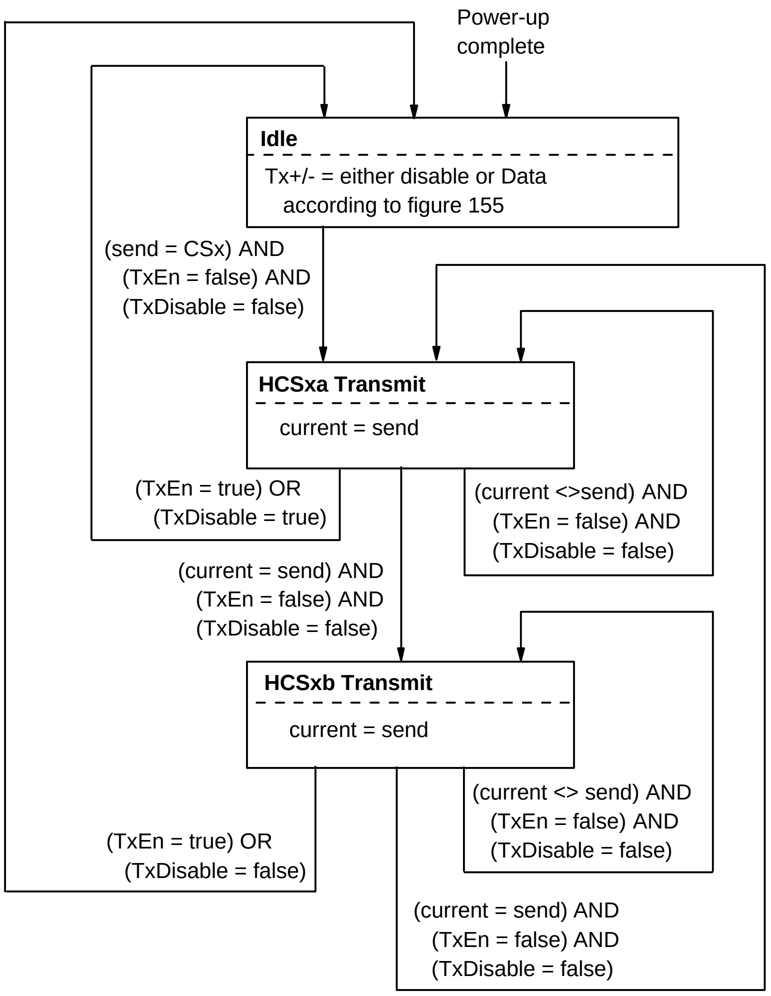

Power-up complete
Idle
Tx+/- = either disable or Data according to figure 155
(send = CSx) AND (TxEn = false) AND (TxDisable = false)
HCSxa Transmit
current = send
(TxEn = true) OR (TxDisable = true)
(current <>send) AND (TxEn = false) AND (TxDisable = false)
(current = send) AND (TxEn = false) AND (TxDisable = false)
HCSxb Transmit
current = send
(current <> send) AND (TxEn = false) AND (TxDisable = false)
(TxEn = true) OR (TxDisable = false)
(current = send) AND (TxEn = false) AND (TxDisable = false)

Figure 17-5—State diagram for control signals

17.8 PMD electrical specifications

This subclause defines the electrical characteristics of the PMD at the MDI interface. Although this subclause specifies the electrical characteristics for a single channel of the twisted-pair link, both channels of the link shall conform to this specification simultaneously. The link shall satisfy the electromagnetic compatibility requirements of Annex B.

17.8.1 Isolation requirement

The PMD shall provide isolation between the end node PMI layer circuits and the MDI. This electrical separation shall withstand at least one of the following electrical strength tests:

a) 1500 Vrms at 50–60 Hz for 60 s, applied as specified in Section 5.3.2 of IEC 60950:1991.

b) 2250 Vdc for 60 s, applied as specified in Section 5.3.2 of IEC 60950:1991.

c) A sequence of ten 2400 V impulses of alternating polarity, applied at intervals of not less than 1 s. The shape of the impulses shall be 1.2/50 µs (1.2 µs virtual front time, 50 µs virtual time of half value), as defined in IEC 60060–1:1989.

There shall be no insulation breakdown, as defined in Section 5.3.2 of IEC 60950:1991, during the test. The resistance after the test shall be at least 2 MΩ, measured at 500 Vdc.

17.8.2 Transmitter specifications

The PMD shall provide the Transmit functions specified in 17.5 in accordance with the electrical specifications of this clause.

Where a load is not specified, the transmitters shall meet the requirements of this clause when connected to a 150 Ω (± 0.1%) resistive load.

17.8.2.1 Differential output voltage

Some of the text and figures of this clause describe the differential voltage in terms of magnitudes. The requirements of this clause apply to negative as well as positive pulses.

The steady-state peak-to-peak differential voltage on the Tx+/- circuit when terminated with a 150 Ω (± 0.1%) resistive load shall be between 0.45 V and 0.56 V. This shall be measured midway between zero crossing transitions on the following eight steady state patterns: CS1, CS5, and a repeating sequence of n ones followed by n zeros for n = 6,5,4,3,2,1. Figure 17-6 shows the timing for this measurement for the CS1 signal.

Figure 17-6 shows the boundaries the output waveform shall remain within for the repetitive CS1 signal.

NOTE—Figure 17-6 illustrates the allowable droop and undershoot/overshoot and how to make rise and fall time measurements.

For all valid sequences of packet data applied to Tx+/-, the resulting zero crossing jitter at the MDI shall be less than 1.8 ns peak to peak. Jitter shall be measured with TxClk as reference. The duty cycle shall be 50 ± 2%.

In addition, the Tx+/- output characteristics shall comply with Table 17-3.

When the PMD transmitter is disabled, the amplitude of the Tx+/- output signal shall not exceed 50 mV peak to peak.

17.8.2.2 Transmitter differential output impedance

When the PMD is powered, the differential output impedance as measured on the Tx+/- circuit shall be such that any reflection, due to differential signals incident upon the Tx+/- circuit from a link segment having any impedance within the range of 150 ± 15 Ω, shall be at least 12 dB below the incident level, over the frequency range of 1–100 MHz.

17.8.2.3 Transmitter fault tolerance

Transmitters, when either idle or non-idle, shall withstand without damage the application of short circuits across any of the Tx+/- or Rx+/- circuits for an indefinite period of time. The magnitude of any resulting short circuit current shall be less than 300 mA. The Tx+/- and Rx+/- circuits shall resume normal operation after such faults.

Transmitters, when either idle or non-idle, or not powered, shall withstand without damage the application of a 1000 V common-mode impulse applied at $E_{impulse}$ of either polarity, as indicated in Figure 17-7. The shape of the impulse shall be 0.3/50 µs (300 ns virtual front time, 50 µs virtual time of half value), as defined in IEC 60060–1:1989.

17.8.3 Receiver specifications

The PMD shall provide the receive functions specified in 17.6 in accordance with the electrical specifications of this subclause.

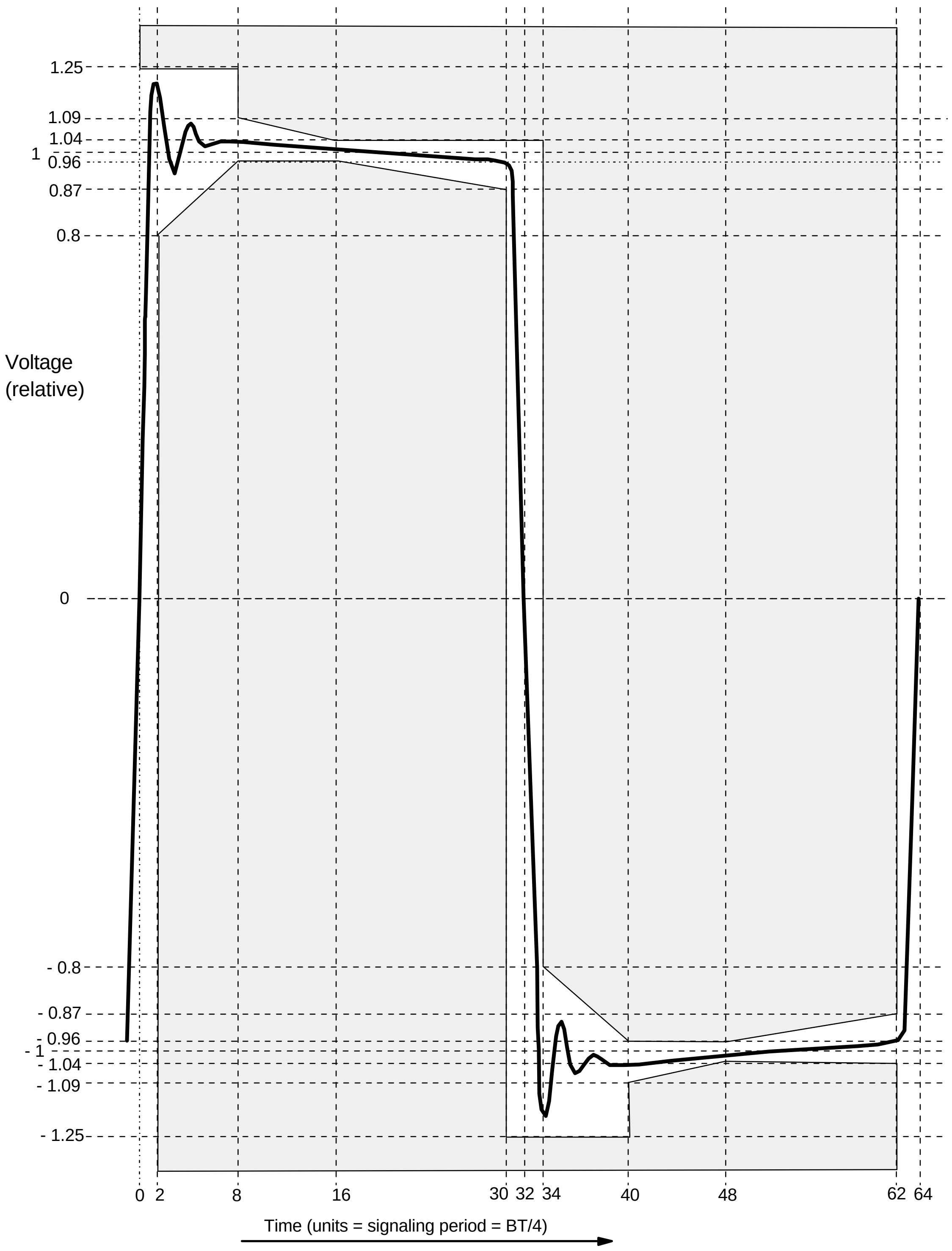


Figure 17-6—Voltage template for transmitter output waveform

Table 17-3—x+/- output characteristics

Characteristic	Minimum	Maximum	Units
Peak-to-peak differential signal level	0.45	0.56	V
Risetime (10–90%)	2	6	ns
Falltime (10–90%)	2	6	ns

NOTE—The specification of the transmitter output waveform is not intended to ensure compliance with regulations concerning RF emissions. The implementor should consider any applicable local, national, or international regulations.

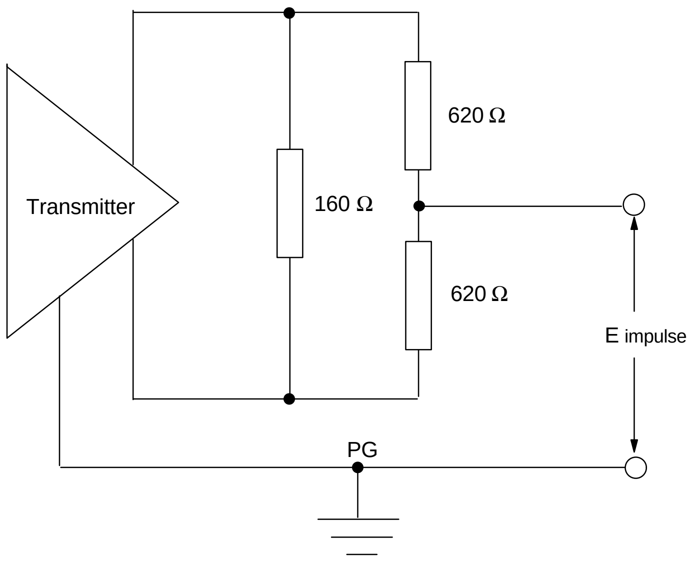


Figure 17-7—Transmitter common-mode impulse test

17.8.3.1 Receiver differential input signals

When presented with valid data or control signals on circuits Rx+/-, the PMD shall either send the data to the RData[3:0] circuits, with a BER no greater than 10^{-8}, or recover the control signals in accordance with 17.6. The PMD shall attempt to recover control signals whenever the magnitude of the signal on Rx+/- exceeds 200 mV for longer than 1 BT. The PMD shall not attempt to recover control signals when the magnitude of the signal on Rx+/- is less than 100 mV for longer than 4 BT.

17.8.3.2 Receiver equalization

The received signal shall be equalized such that the link BER is less than 10^{-8}.

17.8.3.3 Receiver differential input impedance

The differential input impedance shall be such that any reflection, due to differential signals incident upon the Rx+/- circuit from a link segment having any impedance within the range of 150 ± 15 Ω, shall be at least 12 dB below the incident over the frequency range of 1–100 MHz.

17.8.3.4 Receiver fault tolerance

The receiver shall tolerate the application of short circuits between the leads of the Rx+/- circuit for an indefinite period of time without damage and shall resume normal operation after such faults are removed. Receivers shall withstand without damage a 1000 V common-mode impulse of either polarity ($E_{impulse}$ as shown in Figure 17-8). The shape of the impulse shall be 0.3/50 µs (300 ns virtual front time, 50 µs virtual time of half value), as defined in IEC 60060–1:1989.

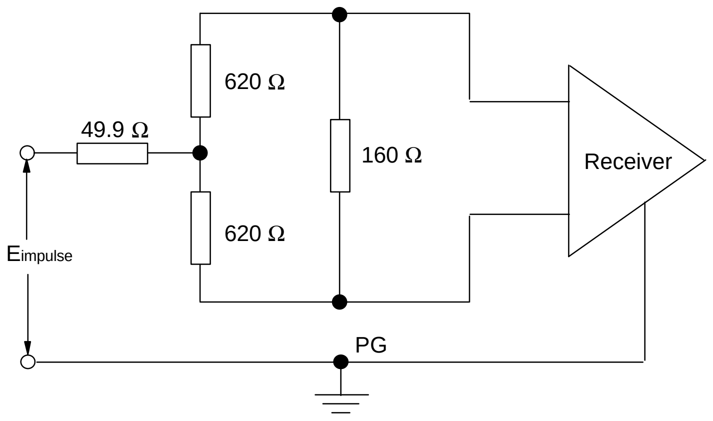


Figure 17-8—Receiver common-mode impulse test

17.9 Characteristics of the 150 Ω balanced cable link segment

STP (150 Ω balanced cable) PMD link segments consist of the link medium, two end-connectors, and connection devices such as patch panels and wall plates, as shown in Figure 17-2.

The link consists of up to 100 m of 150 Ω balanced cable, connecting hardware, cross-connect jumpers, and patch cords that shall meet or exceed the specifications of ISO/IEC 11801:1995 subclauses 7.2 and 8.3.

17.10 MDI specification

This subclause defines the electrical and mechanical specifications for the preferred MDI connectors used for the STP PMD.

17.10.1 MDI connectors

The choice of MDI STP PMD connector is optional except that if a 9-pin D shell connector is chosen, the connector and pin assignments shall comply with 17.10.2.

17.10.2 Optional STP PMD MDI 9-pin D-shell connector

If a 9-pin D shell connector is chosen (as shown in Figure 17-9), it shall be a receptacle and comply with the intermateability geometry requirements of ISO 4902:1989.

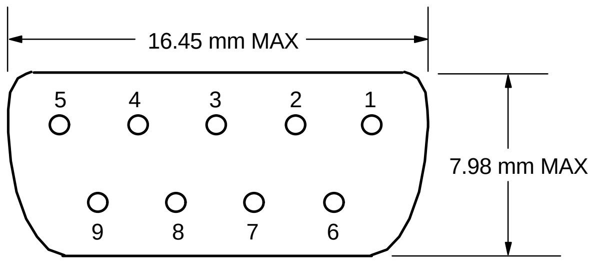


Figure 17-9—MDI receptacle

The assignment of circuits to each MDI contact shall comply with Table 17-4. The shell shall be grounded.

Note that a crossover is implemented in the case of a repeater local port. No internal crossover is required in a repeater cascade port as it functions similar to an end node and is always connected to an upper-level repeater local port.

Table 17-4—MDI connector pin assignments

D Connector pin	Cable wire	MDI signal (End node or repeater cascade port)	MDI signal (Repeater local port)
1	R	Rx+	Tx+
6	G	Rx–	Tx–
9	O	Tx–	Rx–
5	B	Tx+	Rx+

18. Dual simplex fibre optic Physical Medium Dependent (PMD) sublayer, Medium Dependent Interface (MDI), and link specifications for PMD 800 nm and PMD 1300 nm

18.1 Scope

This clause defines the functional characteristics of the PMD sublayer and provides optical and mechanical specifications for the MDI and media for two Light Emitting Diode (LED) fibre-optic link implementations. The relationship of the PMD specification to the LAN model is shown in Figure 18-1. No particular implementation of the PMD is implied.

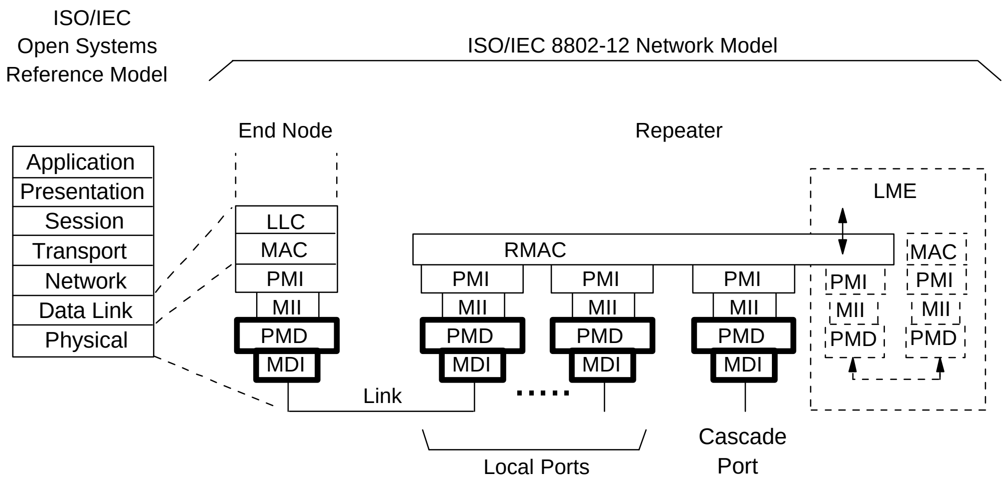


Figure 18-1—PMD and MDI relationship to the LAN model

18.2 Overview

The LED fibre-optic PMD layer has the following general characteristics:

a) Enables coupling of the PMI sublayer and MII to fibre-optic link segments (see Figure 18-2).

b) Supports data packet traffic at a signaling rate of 120 MBd.

c) Allows link segments between an end node and a repeater, or between connected repeaters in the following lengths:

   1) From 1–500 m if 800 nm transceivers are used.
   2) From 1–2 km if 1300 nm transceivers are used.

d) Supports network configurations using the demand-priority access method with baseband signaling on two optical fibres.

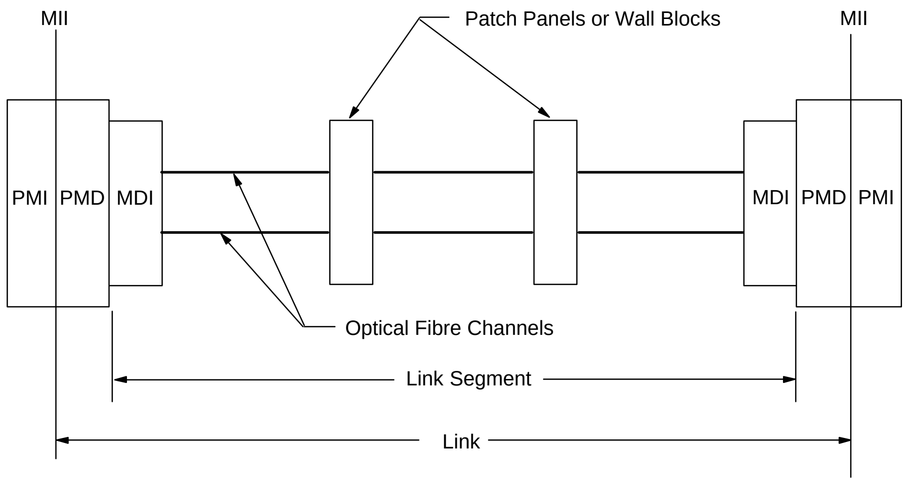


Figure 18-2—Fibre-optic link

18.2.1Signal conditioning and control functions

The PMD sublayer provides transmit and receive functions to control the link status and to facilitate data packet flow between the end node or repeater and the fibre-optic link segment. The PMD is the medium dependent connection between the MII and the network link segment. It:

a) Generates and transmits outgoing control signals.
b) Transmits outgoing data.
c) Recognizes incoming control signals and differentiates them from incoming data.
d) Recognizes packet preambles and recovers the receive clock (RxClk) from the incoming data streams.
e) Recognizes and recovers incoming data streams.
f) Multiplexes and demultiplexes packet data.

18.2.2PMD interfaces

The PMD has two defined interfaces: one to the PMI sublayer (MII), and one to the fibre-optic link segment (MDI). The PMD may be implemented as a separate unit, with a physically exposed MII to the PMI, or it may be logically integrated with the PMI. The interface to the link segment shall always be physically exposed.

18.2.3Fibre-optic medium

The normal medium for fibre-optic links is fibre-optic cable meeting the specifications of ISO/IEC 11801:1995 for 62.5/125, multimode, graded-index, optical fibre (see 18.10).

18.2.4PMD/Link configuration

When implemented, the PMD/Link configuration circuit Config[3:0] shall be set to 1001 (see 15.3.11).

18.3PMD to PMI interface

The PMD to PMI interface shall comply with the logical signal and timing requirements defined for the MII in Clause 15.

18.4 Functional description of interchange circuits

The PMD to MII circuits are described in Clause 15. The PMD sublayer and the MDI interface both reference an Active Optical Output (AO) and an Active Optical Input (AI).

18.5 Transmit functions

The PMD transmit functions include the transmission of both data and control signals.

a) Data transmission requires the encoding and transmission of data present on the MII circuits TData[3:0].

b) Control signal transmission requires generation of the control signals prior to encoding and transmission.

18.5.1 Multiplexing physical frames

The PMD shall multiplex the data on the MII circuits TData[3:0] prior to transmission on a link segment as a single serialized data stream. The PMD shall multiplex TData[3:0] so that the physical frame structure at the MDI is:

<mpreamble><mssd>< S_1><S_2><S_3>........<S_n>< mesd><F>

where

a) <mpreamble> shall be a stream of alternating 0’s and 1’s, terminating in a 1, of length from 192 bits to 192 bits minus the maximum number of preamble bits that may be discarded by the PMD (see 18.5.5).

b) <mssd> shall be the multiplexed start of stream delimiters on the MII circuits TData[3:0].

   1) For a normal-priority packet, <mssd> shall be:

      111100 111100 111100 111100 000011 000011 000011 000011

   2) For a high-priority packet, <mssd> shall be:

      100000 100000 100000 100000 111110 111110 111110 111110

   The leftmost bit of <mssd> as defined above shall be transmitted first at the MDI.

c) <S_m> is the m^{th} code sextet to be transmitted (see 14.4 and Figure 14-8). Complete code sextets on the MII circuits TData[3:0] shall be multiplexed in a cyclic manner to form a serialized bit stream. <S_1> shall be the first code sextet on TData[0], <S_2> shall be the first code sextet on TData[1], <S_3> shall be the first code sextet on TData[2], <S_4> shall be the first code sextet on TData[3], <S_5> shall be the second code sextet on Tdata[0], and so on. This cyclic multiplexing of complete code sextets shall continue until the last code sextet has been multiplexed.

   Code sextets shall be transmitted serially in the bit order received on the MII circuits TData[3:0].

d) <mesd> shall be the sextet multiplexed end of stream delimiters received on the MII circuits TData[3:0]. Four <mesd> sequences are possible due to the four possible TData[3:0] end combinations (see Figure 14-9).

   Let x be the number of octets modulo 5 in the MAC frame.

   1) If x = 0 or 2, <mesd> shall be:

      <esd_{01}><esd_{11}><esd_{21}><esd_{31}><esd_{02}><esd_{12}><esd_{22}><esd_{32}><esd_{03}><esd_{13}> <esd_{23}><esd_{33}>

   2) If x = 1, <mesd> shall be:

      <esd_{21}><esd_{31}><esd_{01}><esd_{11}><esd_{22}><esd_{32}><esd_{02}><esd_{12}><esd_{23}><esd_{33}> <esd_{03}><esd_{13}>

3) If x = 3, <mesd> shall be:

<esd$_{11}$><esd$_{21}$><esd$_{31}$><esd$_{01}$><esd$_{12}$><esd$_{22}$><esd$_{32}$><esd$_{02}$><esd$_{13}$><esd$_{23}$><esd$_{33}$><esd$_{03}$>

4) If x = 4, <mesd> shall be:

<esd$_{31}$><esd$_{01}$><esd$_{11}$><esd$_{21}$><esd$_{32}$><esd$_{02}$><esd$_{12}$><esd$_{22}$><esd$_{33}$><esd$_{03}$><esd$_{13}$><esd$_{23}$>

<esd$_{jk}$> shall be the kth sextet of the esd associated with channel j (see 14.4.2.3.5). Although the four cases of <mesd> patterns are shown with reference to x, the number of octets modulo 5 in the MAC frame, the PMI does not need to count octets, quintets, or sextets (see 14.4.2.3.5, 14.4.2.3.6, and Figure 14-9).

e) <F> are end fill bits as required. The value of F does not affect the receiver operation. The actual value chosen is an implementation issue and is beyond the scope of this standard.

18.5.2 Data and control signal encoding

NRZ encoding is used for the transmission of data and control signals across the optical fibre medium. A value of 1 is transmitted as a high optical signal level, and a value of 0 is transmitted as a low optical signal level.

18.5.3 Data signaling function

Data on the MII circuits TData[3:0] is multiplexed according to 18.5.1 and transmitted serially on the fibre-optic medium, resulting in a total signaling rate of 120 MBd. Data on circuits TData[3:0] shall be transmitted from the AO of the MDI according to the requirements of 18.8, whenever the MII circuit TxEn is asserted.

The steady-state delay between data being present on TData[3:0] and the same data being transmitted on AO shall be no less than 8 BT and no more than 12 BT.

18.5.4 Control signaling function

Control signals transmitted on the fibre-optic medium are used to communicate the link state between a repeater and an end node or between cascaded repeaters. Control signals are controlled by the state of the TCS[2:0] circuits at the MII. Control signals shall be transmitted from the AO using a symbol clock generated by multiplication in frequency of the clock on circuit TxClk by a factor of four.

18.5.4.1 Control signal generation

The PMD shall be capable of generating the following five control signals and transmitting them on AO:

a) Control signal 1 (CS1) is a continuous repetition of two alternating patterns, <HCS1a> and <HCS1b>.

1) <HCS1a> is the pattern:

1111 1111 1111 1111 0000 0000 0000 0000.

2) <HCS1b> is the pattern:

0000 0000 0000 0000 1111 1111 1111 1111.

These patterns shall be NRZ encoded and transmitted at 120 MBd. The fundamental frequency of CS1 is 1.875 MHz.

b) Control signal 2 (CS2) is a continuous repetition of two alternating patterns, <HCS2a> and <HCS2b>.

   1) <HCS2a> is the pattern:

      1111 1111 1111 1100 0000 0000 0000.

   2) <HCS2b> is the pattern:

      0000 0000 0000 0001 1111 1111 1111 11.

   These patterns shall be NRZ encoded and transmitted at 120 MBd. The fundamental frequency of CS2 is 2.069 MHz.

c) Control signal 3 (CS3) is a continuous repetition of two alternating patterns, <HCS3a> and <HCS3b>.

   1) <HCS3a> is the pattern:

      111 1111 1111 1000 0000 0000 00.

   2) <HCS3b> is the pattern:

      0000 0000 0000 0111 1111 1111 11.

   These patterns shall be NRZ encoded and transmitted at 120 MBd. The fundamental frequency of CS3 is 2.308 MHz.

d) Control signal 4 (CS4) is a continuous repetition of two alternating patterns, <HCS4a> and <HCS4b>.

   1) <HCS4a> is the pattern:

      1111 1111 1110 0000 0000 00.

   2) <HCS4b> is the pattern:

      0000 0000 0000 1111 1111 1111.

   These patterns shall be NRZ encoded and transmitted at 120 MBd. The fundamental frequency of CS4 is 2.609 MHz.

e) Control signal 5 (CS5) is a continuous repetition of two alternating patterns, <HCS5a> and <HCS5b>.

   1) <HCS5a> is the pattern:

      1111 1111 1100 0000 0000.

   2) <HCS5b> is the pattern:

      0000 0000 0011 1111 1111.

   These patterns shall be NRZ encoded and transmitted at 120 MBd. The fundamental frequency of CS5 is 3.000 MHz.

NOTE—Half Control Signal (HCS) is a generic term used to refer to any of the patterns <HCSxa> or <HCSxb> (where x=1, 2, 3, 4 or 5).

Whenever TxEn is deasserted, the PMD shall transmit a control signal on AO or shall disable AO. Table 18-1 shows the relationship between TCS[2:0] and the state of AO.

Table 18-1—Relationship between Transmit Control States and control signals

TCS[2:0]	AO control signal
000	disable*
001	CS3
010	CS2
011	CS4
100	CS1
101	reserved
110	reserved
111	CS5

* See 18.5.7.

18.5.4.2 Control signal transitions

At the start of control signaling, the first control signal transmitted shall begin with an <HCSxa> pattern (x = 1, 2, 3, 4 or 5).

Transitions between any two different control signals shall be accomplished such that any <HCSxa> pattern (x = 1, 2, 3, 4 or 5) is always followed by an <HCSya> pattern (y = 1, 2, 3, 4 or 5, y ≠ x) , or any <HCSxb> pattern (x = 1, 2, 3, 4 or 5) is always followed by an <HCSyb> pattern (y = 1, 2, 3, 4 or 5, y ≠ x). Transitions between any two control signals shall be accomplished such that an equal number of 1’s and 0’s has been transmitted on AO when the transition occurs. This requires that an integer number of HCS patterns shall be transmitted on AO before the transition.

NOTE—The first bit of a new control signal is never equal to the last bit of the previous control signal.

When the PMD is required to disable the transmitter on AO, this shall be accomplished such that an equal number of 1’s and 0’s has been transmitted on AO before the transmitter is disabled. This requires that an integer number of HCS patterns shall be transmitted on AO before the transmitter is disabled.

While transmitting a control signal, the first bit of the next control signal required by any valid value of TCS[2:0] shall be transmitted on AO within 12 BT of that value of TCS[2:0] being set. The PMD transmitter shall be disabled within 20 BT of TCS[2:0] being set to 000.

18.5.5 Transitions between data and control signaling

On assertion of TxEn, the PMD shall continue to transmit the current control signal until an equal number of 1’s and 0’s has been transmitted on AO. This requires that an integer number of HCS patterns, as defined in 18.5.4.1, has been transmitted on AO. The PMD shall then begin to transmit data on AO as specified in 18.5.3. The transition from control signal to data shall occur within 20 BT of TxEn being asserted. No more than 40 bits of preamble shall be discarded from the serialized data stream transmitted on AO.

When TxEn is asserted, the PMD shall complete transmission of the current control signals and then transmit no more controls signals until TxEn is deasserted. However, the value of TCS[2:0] may be updated by the PMI during this time. The transition from data signaling to control signaling shall occur immediately after the last data bit is transmitted. Data transmission shall be completed and no additional data bits shall be added before switching to the appropriate HCS pattern or disabling the PMD transmitter. The first HCS pattern transmitted after the completion of data transmission shall be an <HCSxa> pattern (x = 1, 2, 3, 4, or 5).

18.5.6 Transmit Control State (TCS) description

The transmit control state is controlled by the TCS [2:0] and TxEn circuits at the MII. The dual simplex PMD either transmits data or control signals from the AO.

When TxEn is enabled, the PMD shall transmit data on AO.

18.5.7 Disable

Disable shall be a continuous repetition of 17 ones followed by 17 zeros. This pattern shall be NRZ encoded and transmitted at 120 MBd. Disable is equivalent to disabling the transmitters in PMD’s supporting copper links and is not included in the set of defined control signals CS1, CS2, CS3, CS4, and CS5. See 18.5.4.1.

18.6 Receive function requirements

Receive functions of the PMD include control signal recovery and decoding, preamble detect, clock recovery, and data recovery. The PMD shall receive data and the link control signals from the AI. The PMD shall recover packet data from the optical signals on the AI of the MDI and send them to the MII RData[3:0] circuits.

18.6.1 Control signal recovery and decoding

The PMD shall have the capability to recover control signals from the signal on AI. The PMD shall determine the RCS from the received control signals according to 18.6.8, and indicate the RCS on the MII RCS[2:0] circuits within 24 BT of any change in the control signals present on AI. The PMD shall tolerate at least 2 errored bits per 320 consecutive bits during a control signal without indicating an incorrect line state on circuits RCS[2:0].

18.6.2 Signal detect function

The PMD shall have a Signal Detect Function to indicate the presence of an optical signal with sufficient quality to allow the recovery of valid receiver control states. The PMD shall attempt to recover control signals or data from the AI when a condition of “high_light” (see 18.8.2.5 and 18.9.2.5) exists. The PMD shall not attempt to recover control signals or data from the AI when a condition of “low_light” (see 18.8.2.6 and 18.9.2.6) exists.

18.6.3 Signal monitor function

While RxEn is asserted and RCS is currently set to 101, indicating that data signals have been detected, the PMD shall detect nondata signals on AI. Nondata signals shall be detected when any valid control signal or disable is present on AI. The PMD shall set RCS[2:0] to 000 within 25 BT of the first bit of nondata signal being received on AI. The PMD shall tolerate at least 2 errored bits during a packet reception without detecting nondata signals.

18.6.4 Clock recovery function

The PMD shall recover the symbol clock from the data received at the AI. The PMD shall detect preamble and send the recovered clock divided down in frequency by a factor of four to the MII RxClk circuit. The PMD shall continue to send the divided down clock to RxClk while RxEn is asserted.

18.6.5 Data recovery function

Serial data signals shall be recovered from the optical signal at the AI, demultiplexed according to 18.6.6, and sent to the MII circuits RData[3:0]. The steady-state delay between data being present on AI and the same data being set to RData[3:0] shall be no less than 8 BT and no more than 12 BT. The differential delay between the first bit of any two of <ssd$_{k1}$> (where k = 0, 1, 2 or 3) being sent to the appropriate RData[k] circuits shall be no more than 8 BT.

18.6.6 Demultiplexing of physical frame

The PMD shall demultiplex each received packet such that the physical frame structure at the MII shall be:

Channel k: <preamble><ssd$_{k1}$><ssd$_{k2}$><S$_{k+1}$><S$_{k+5}$><S$_{k+9}$>....<esd$_{k1}$><esd$_{k2}$><esd$_{k3}$><x>

where k = 0 to 3, <preamble> shall be at least three consecutive occurrences of the sequence 01, and <x> are filling bits as required.

The occurrence of <ssd$_{01}$> after <preamble> shall be used by the PMD to guarantee recovery of sextet word boundaries for demultiplexing of the physical frame into the four channels of RData[3:0].

18.6.7 Grant detection function

The PMD shall assert the MII RxGrant circuit no more than 15 BT after the first bit of CS5 is present on AI. The RxGrant circuit shall be deasserted no more than 22 BT after CS5 is no longer present on AI.

18.6.8 Receiver Control State (RCS) description

The RCS shall be indicated to the PMI on MII circuits RCS[2:0] and RxGrant. The RCS is controlled by the control signals recovered from AI, and the MII circuit RxEn. Upon power-up, the PMD shall set circuits RCS[2:0] to 000 and deassert the RxGrant circuit. While RxEn is not asserted, the PMD shall indicate the RCS on RCS[2:0] and RxGrant. During this time, if CS5 is detected or if no valid control signal can be recovered on AI, the PMD shall maintain the existing value of RCS[2:0]. If valid control signals cannot be recovered on AI for a period longer than “fault_time,” then RCS[2:0] shall be set to 111. RCS[2:0] shall then be maintained at 111 until a valid control signal has been detected on AI. The period “fault_time” shall be between 1 ms and 2 ms. Table 18-2 shows the relationship between the control signal on AI and RCS[2:0] states.

Table 18-2—Relationship between AI control signal and RCS[2:0]

AI control signal	RCS[2:0]
Mode Transition*	000
CS3	001
CS2	010
CS4	011
CS1	100
Data on link*	101
reserved	110
Link Warning*	111

*See Figure 18-4 for detailed information on these codes.

While RxEn is asserted, RxGrant shall be deasserted, but the PMD shall continue to indicate the RCS on RCS[2:0] until preamble is detected on AI, when RCS[2:0] shall then be set to 101. If subsequent to preamble being detected, RxEn is deasserted, or the PMD detects nondata signals on AI before RxEn is deasserted, RCS[2:0] shall be set to 000.

Whenever TxEn is deasserted, RCS shall be set to 000 for one cycle of TxClk or until a valid control signal (CS1, CS2, CS3, or CS4) has been detected on AI, whichever is longer.

18.7PMD state diagrams

The state diagrams depict the operation of the PMD functions relative to the status of the MII and MDI circuits. The state diagram for the transmit functions is depicted in Figure 18-3. The state diagram for the receive functions is depicted in Figure 18-4 and the state diagram for the control signal function is shown in Figure 18-5. The notation used in the state diagrams follows the conventions of Clause 5. The variables and timers used in the state diagrams are defined in 18.7.1 and 18.7.2.

18.7.1State diagram variables

Variables are used in the state diagrams to: indicate the status of PMD inputs and outputs, control PMD operation, and pass state information between functions.

In the variable definitions, the name of the variable is followed by a brief definition of the variable and a list of values the variable may take. For those variables that are state diagram outputs, one value will be identified as the default. The variable has the default value when no active state contains a term assigning a different value.

For example, the variable “RCS” has the value of “000” whenever the Receive Function is in a state that asserts “RCS=000”. Otherwise, this variable has the default value of “update.”

The following variables are used in the PMD state diagrams.

AO. Controls the status of the MDI active optical output: AO.

Values:	disable;	a continuous repetition of 17 ones followed by 17 zeros.
	CSX;	control signals that are transmitted on AO by the Control Signaling Function.
	Data;	data on MII circuit TData[3:0] that is multiplexed and transmitted serially on AO by the Data Signaling Function.

Bal. Indicates the balance between 1’s and 0’s transmitted on AO.

Values:	true;	during control signaling, an equal number of 1’s and 0’s has been transmitted on AO.
	false;	during control signaling, an equal number of 1’s and 0’s has not been transmitted on AO.

clock. Indicates the results of the Clock Recovery function.

Values:	false;	PMD has not recovered a valid clock.
	true;	PMD has recovered a valid clock.

current. Indicates the particular control signal (if any) being transmitted on AO.

Values: CS1, CS2, CS3, CS4, or CS5; set at start of each HCS transmission, by “send”
disable; TCS=000 (default).

data_sent. indicates that the last bit of a packet has been transmitted on AO.

Values:	true:	the last bit of the current packet has been transmitted.
	false:	the last bit of the current packet has not been transmitted.

Grant. Controls the value of the MII circuit RxGrant.

Values:	update;	Grant Detection function sets RxGrant to reflect current received signals (default).
	true;	Grant Detection function asserts RxGrant regardless of current received signals.
	false;	Grant Detection function deasserts RxGrant regardless of current received signals.

RCS. Controls the value of the MII circuits RCS[2:0].

Values:	update;	Control Recovery function updates RCS[2:0] whenever a valid signal is decoded. If a valid control signal is not decoded, the last value of RCS is maintained (default)
	last_known;	Control Recovery function maintains the most recent value of RCS[2:0].
	binary 000–111:	Control Recovery function sets RCS[2:0] to binary value.

RData. Controls the status of the MII circuits Rdata[3:0].

Values:	inactive;	RData[3:0] are high impedance, as defined in 15.3.10 (default).
	AI;	Data from AI are decoded, demultiplexed, and placed on RData[3:0] by the Data Recovery function.
	deassert;	RData[3:0] are deasserted.

RxClk. Controls the status of the MII circuit RxClk.

Values:	inactive;	RxClk is high impedance, as defined in 15.3.9 (default).
	active;	The clock recovered from data on AI is placed on RxClk by the Clock Recovery function.
	deassert;	RxClk is deasserted.

RxEn. Indicates the status of the signal received by the PMD on the RxEn circuit.

Values:	false;	RxEn is deasserted.
	true;	RxEn is asserted.

send. Indicates the particular control signal to be transmitted on AO.

Values:	CS1, CS2, CS3, CS4, or CS5;	The value of “send” is determined by Table 18-1.
	disable;	TCS=000 (default).

signal_type. Indicates the type of signal the PMD has detected at its inputs.

Values:	preamble;	Preamble has been detected.
	not_data;	Nondata signals have been detected (see 18.6.3).
	other;	Neither preamble nor nondata patterns have been detected.

TCS. Indicates the status of the MII circuits TCS[2:0].

Values: binary 000–111.

TxDisable. Indicates when the PMI has requested the transmitter on AO to be disabled.

Values:	true;	TCS[2:0] has a value of 000.
	false;	TCS[2:0] does not have a value of 000.

TxEn. Indicates the status of the signal received by the PMD on the TxEn circuit.

Values:	true;	TxEn is asserted.
	false;	TxEn is deasserted.

valid_CS. Indicates that a valid control signal on AI has been detected by the Control Recovery function.

Values:	true;	A valid control signal (CS1, CS2, CS3, CS4, or CS5) has been detected.
	false;	A valid control signal has not been detected.

18.7.2 State diagram timers

A timer is reset and starts counting upon entering a state where “start x_timer” is asserted. Time “x” after the timer has been started, “x_timer_done” is asserted and remains asserted until the timer is reset. At all other times, “x_timer_not_done” is asserted.

When entering a state in which “start x_timer” is asserted, the timer is reset and restarted even if the entered state is the same as the exited state. For example, when in the Control Decode state of the Receive Functions state diagram, the “fault_timer” is reset each time the term “valid_CS” is satisfied.

fault_timer. Timer for delay between last valid control signal combination detected and RCS being set to indicate a Link Warning. Valid control signals are CS1, CS2, CS3, CS4, and CS5.

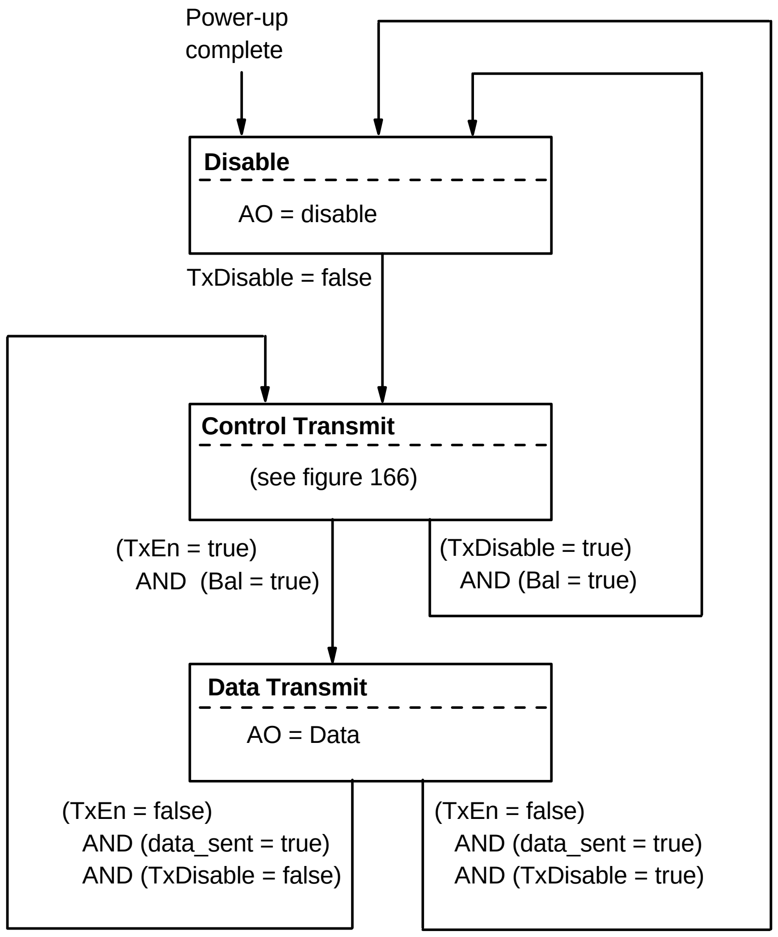

Power-up complete
Disable
AO = disable
TxDisable = false
Control Transmit
(see figure 166)
(TxEn = true) AND (Bal = true)
(TxDisable = true) AND (Bal = true)
Data Transmit
AO = Data
(TxEn = false) AND (data_sent = true) AND (TxDisable = false)
(TxEn = false) AND (data_sent = true) AND (TxDisable = true)

Figure 18-3—State diagram for transmit functions

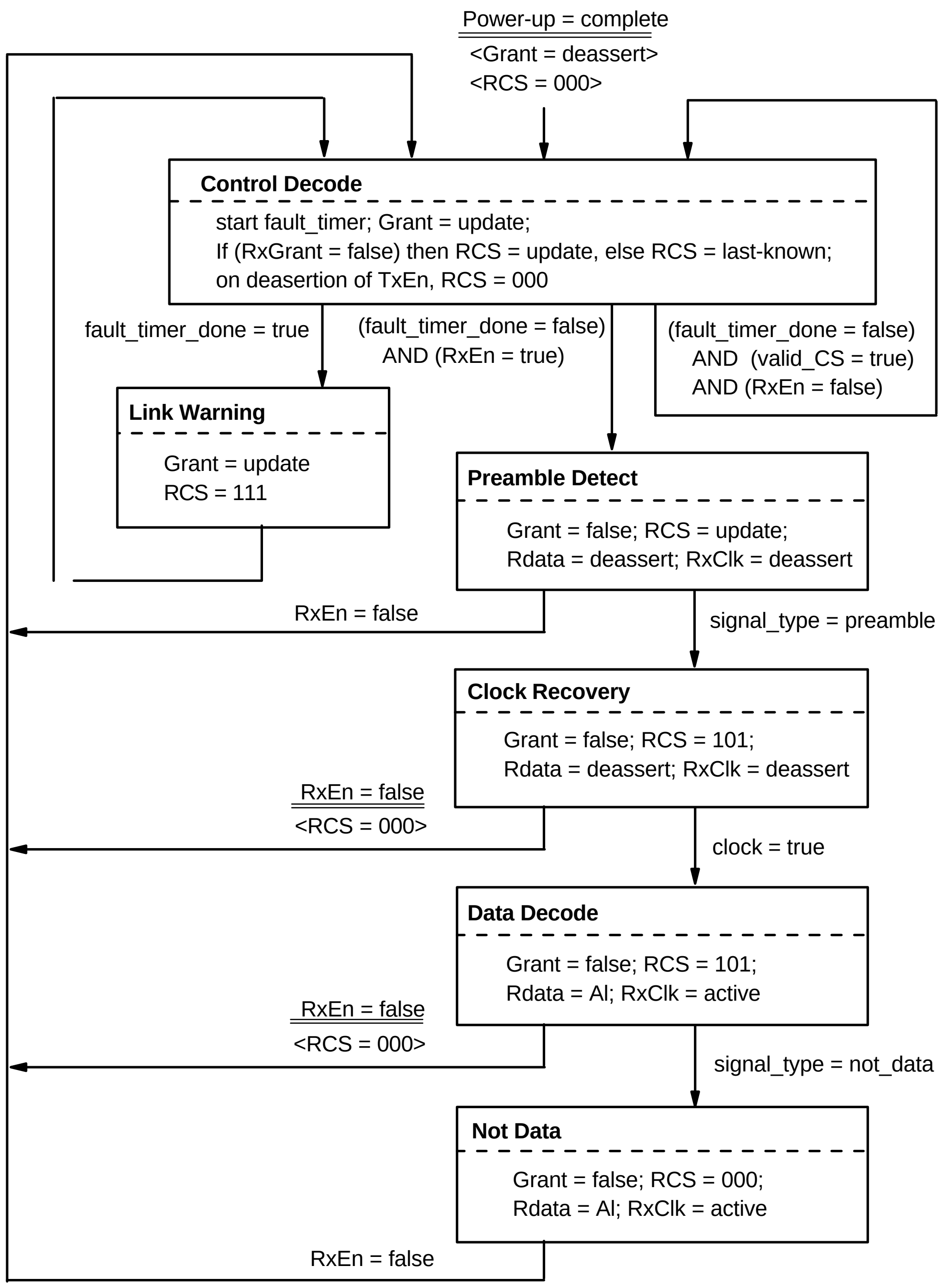

Power-up = complete
<Grant = deassert>
<RCS = 000>
Control Decode
start fault_timer; Grant = update;
If (RxGrant = false) then RCS = update, else RCS = last-known;
on deasertion of TxEn, RCS = 000
fault_timer_done = true
(fault_timer_done = false) AND (RxEn = true)
(fault_timer_done = false) AND (valid_CS = true) AND (RxEn = false)
Link Warning
Grant = update
RCS = 111
Preamble Detect
Grant = false; RCS = update;
Rdata = deassert; RxClk = deassert
RxEn = false
signal_type = preamble
Clock Recovery
Grant = false; RCS = 101;
Rdata = deassert; RxClk = deassert
RxEn = false
<RCS = 000>
clock = true
Data Decode
Grant = false; RCS = 101;
Rdata = AI; RxClk = active
RxEn = false
<RCS = 000>
signal_type = not_data
Not Data
Grant = false; RCS = 000;
Rdata = AI; RxClk = active
RxEn = false

Figure 18-4—State diagram for receive functions

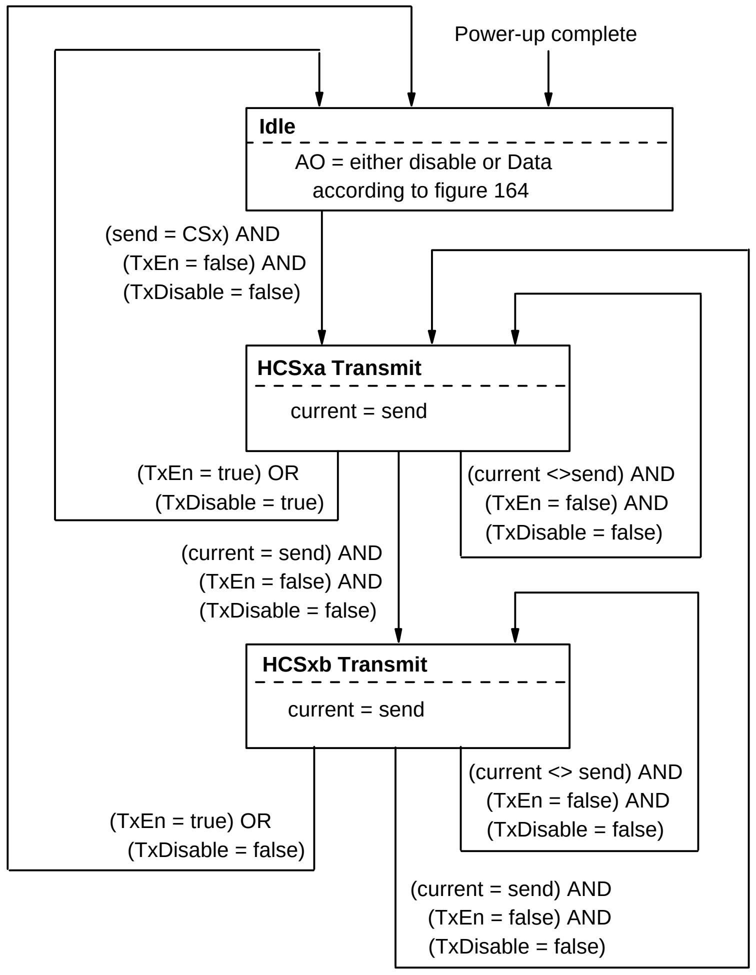

Power-up complete
Idle
AO = either disable or Data according to figure 164
(send = CSx) AND (TxEn = false) AND (TxDisable = false)
HCSxa Transmit
current = send
(TxEn = true) OR (TxDisable = true)
(current <>send) AND (TxEn = false) AND (TxDisable = false)
(current = send) AND (TxEn = false) AND (TxDisable = false)
HCSxb Transmit
current = send
(current <> send) AND (TxEn = false) AND (TxDisable = false)
(TxEn = true) OR (TxDisable = false)
(current = send) AND (TxEn = false) AND (TxDisable = false)

Figure 18-5—State diagram for control signal transmission

18.8 PMD 800 nm optical characteristics

The optical parameters at the MDI, AO, and AI for PMDs equipped with 800 nm transceivers are summarized in Table 18-3. Optical measurements shall be made with the MDI terminated in the optical connector specified in 18.11 and the optical fibre specified in 18.10. The fibre length shall be sufficient to attenuate cladding modes.

NOTE—Fibres currently available typically require 1–5 m to strip the cladding modes.

18.8.1 Transmit specifications

The PMD shall provide the transmit functions specified in 18.5 in accordance with the optical specifications of 18.8.

18.8.1.1 Center wavelength

The center wavelength of the optical source emission shall be as specified in Table 18-3. For LEDs, the center wavelength is defined as the average of the two wavelengths measured at the half amplitude points of the power spectrum.

18.8.1.2 Spectral width

The full width at half maximum (FWHM) of the optical source output spectrum at the AO shall be in the range specified in Table 18-3.

18.8.1.3 Average transmit optical power

The average optical power at the AO shall be in the range specified in Table 18-3.

18.8.1.4 Average optical power of logic zero

The average optical power of logic zero at the AO shall be as specified in Table 18-3.

18.8.1.5 Optical transmit pulse rise and fall times

The time interval for the optical pulse to rise (transition) from 10% to 90% and to fall (transition) from 90% to 10% of the pulse amplitude at the AO shall be as specified in Table 18-3.

18.8.1.6 Optical transmit pulse overshoot

The maximum optical overshoot observed at the AO shall be as specified in Table 18-3.

18.8.1.7 Optical transmit contributed random jitter

The optical transmit random jitter measured at the AO shall be as specified in Table 18-3.

18.8.1.8 Optical transmit contributed systematic jitter

The optical transmit contributed systematic jitter measured at the AO shall be as specified in Table 18-3.

18.8.1.9 Transmit test waveform

The CS1 signal shall be used to measure the transmit average optical power, the average power at logic zero, the rise and fall times, and the overshoot at the AO.

18.8.2 Receive specifications

18.8.2.1 Average receive optical power

The average optical power received at the AI shall be in the range specified in Table 18-3.

18.8.2.2 Optical receive pulse rise and fall times

The time interval for the optical pulse to rise (transition) from 10% to 90% and to fall (transition) from 90% to 10% of the pulse amplitude at the Al shall be as specified in Table 18-3.

18.8.2.3 Optical receive contributed systematic jitter

The optical receive systematic jitter measured at the MDI receiver shall be as specified in Table 18-3.

18.8.2.4 Active optical input signals

When presented with NRZ encoded data signals at the AI that are consistent with the parameters defined by Table 18-3, the PMD shall send the data to the RData[3:0] circuits with a BER no greater than 1 error per 10^8 bits.

Table 18-3—800 nm MDI optical parameters for 500 m links

Active output specification

Parameter	Minimum	Maximum
Center wavelength (nm)	800	900
Spectral width (FWHM) (nm)		100
Average optical power (dBm)	–20	–12
Average optical power at logic zero (dBm)		–45
Rise/fall time, 10–90% (ns)		4.5
Overshoot (%)		25
Contributed random jitter, peak to peak (ns)		0.69
Contributed systematic jitter, peak to peak (ns)		1.7

Active input specification

Parameter	Minimum	Maximum
Average optical power (dBm)	–27.5	–12
Rise/fall time, 10–90% (ns)		8.0
Contributed systematic jitter, peak to peak (ns)		1.2

18.8.2.5High light

A condition of “high_light” shall be detected within 100 µs of valid ISO/IEC 8802-12 optical signals having an average power between P_m and (P_m + 4 dB) appearing at the AI. P_m is the relevant minimum allowed input average optical power for the PMD at the AI as defined in Table 18-3. (The input optical power is averaged over a period of 1 µs or more.)

The condition of “high_light” shall remain asserted while the optical power remains above P_m (see Figure 18-6).

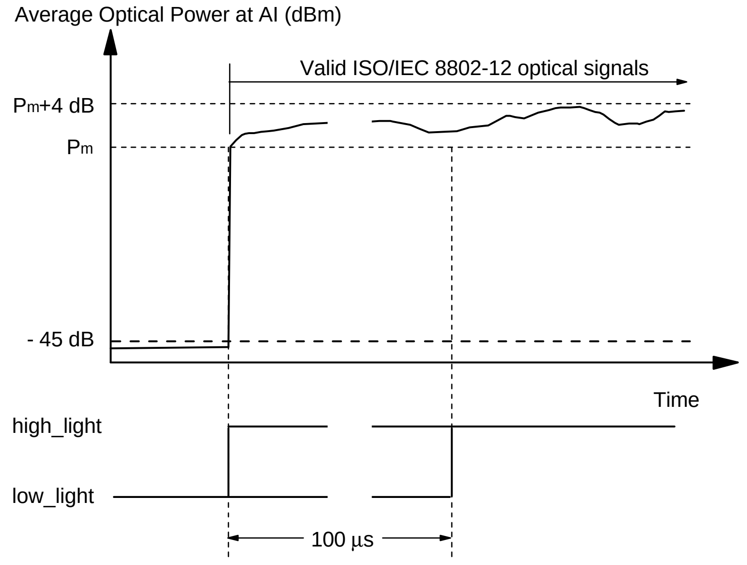

Average Optical Power at AI (dBm)
Valid ISO/IEC 8802-12 optical signals
Pm+4 dB
Pm
- 45 dB
Time
high_light
low_light
100 µs

Figure 18-6—“high_light” condition detect and timing

18.8.2.6 Low light

The PMD shall not detect a condition of “low_light” while valid ISO/IEC 8802-12 optical signals with an input average optical power greater than P_m are being received at the AI.

The PMD shall detect a condition of “low_light” within 350 µs after valid ISO/IEC 8802-12 optical signals cease to be present at the AI and the input average optical power, at the AI, has fallen monitonically from a level between (P_m + 4 dB) and P_m to a level below –45 dBm and remained below –45 dBm (see
Figure 18-7). (The input optical power is averaged over a period of 1 µs or more.)

NOTE—The definitions of “high_light” and “low_light” are consistent with Annex G, [B1].

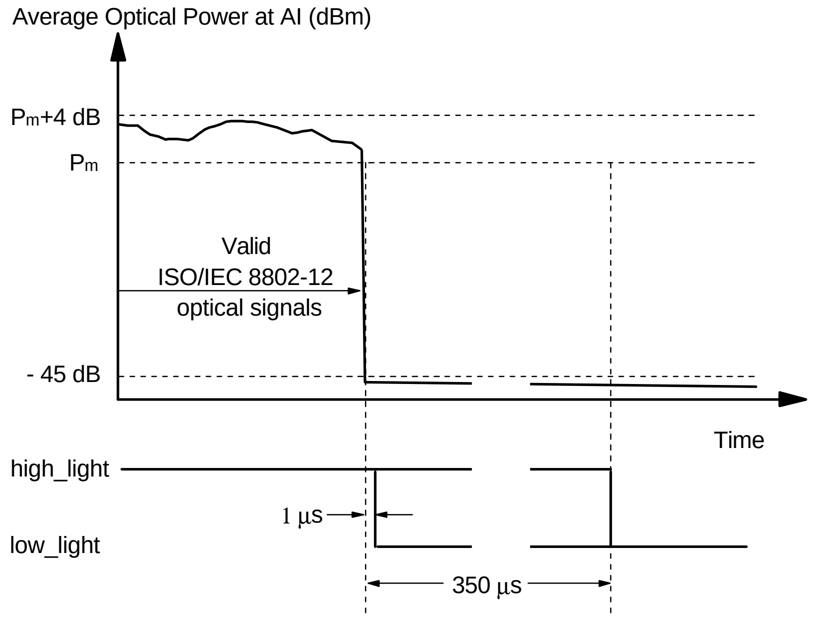


Figure 18-7—“low_light” condition detect and timing

18.9 PMD 1300 nm optical characteristics

The optical parameters at the AO and AI of the MDI for PMDs equipped with 1300 nm transceivers are summarized in Table 18-4. Optical measurements shall be made with the MDI terminated in the optical connector specified in 18.11 and the optical fibre specified in 18.10. The fibre length shall be sufficient to attenuate cladding modes.

NOTE—Fibres currently available typically require 1–5 m to strip the cladding modes.

18.9.1 Transmit specifications

The PMD shall provide the transmit functions specified in 18.5 in accordance with the optical specifications of this subclause.

18.9.1.1 Center wavelength

The center wavelength of the optical source emission shall be as specified in Table 18-4. For LEDs, the center wavelength is defined as the average of the two wavelengths measured at the half amplitude points of the power spectrum.

18.9.1.2 Spectral width

The FWHM of the optical source output spectrum at the AO shall be in the range specified in Table 18-4.

18.9.1.3 Average transmit optical power

The average optical power at the AO shall be in the range specified in Table 18-4.

18.9.1.4 Average optical power of logic zero

The average optical power of logic zero at the AO shall be as specified in Table 18-4.

18.9.1.5 Optical transmit pulse rise and fall times

The time interval for the optical pulse to rise (transition) from 10% to 90% and to fall (transition) from 90% to 10% of the pulse amplitude at the AO shall be as specified in Table 18-4.

18.9.1.6 Optical transmit pulse overshoot

The maximum optical overshoot observed at the AO shall be as specified in Table 18-4.

18.9.1.7 Optical transmit contributed random jitter

The optical transmit random jitter measured at the AO shall be as specified in Table 18-4.

18.9.1.8 Optical transmit contributed systematic jitter

The optical transmit contributed systematic jitter measured at the AO shall be as specified in Table 18-4.

18.9.1.9 Transmit test waveform

The CS1 signal shall be used to measure the transmit average optical power, the average power at logic zero, the rise and fall times, and the overshoot at the AO.

Table 18-4—1300 nm MDI optical parameters for 2 km links

Active output specification

Parameter	Minimum	Maximum
Center wavelength (nm)	1270	1380
Spectral width (FWHM) (nm)		200
Average optical power (dBm)	–22	–14
Average optical power at Logic Zero (dBm)		–45
Rise/Fall Time, 10–90% (ns)		3
Overshoot (%)		25
Contributed random jitter, peak to peak (ns)		0.69
Contributed systematic jitter, peak to peak (ns)		1.2

Active input specification

Parameter	Minimum	Maximum
Average optical power (dBm)	–29	–14
Rise/fall time, 10–90% (ns)		6.0
Contributed systematic jitter, peak to peak (ns)		1.4

18.9.2 Receive specifications

18.9.2.1 Average receive optical power

The average optical power received at the AI shall be in the range specified in Table 18-4.

18.9.2.2 Optical receive pulse rise and fall times

The time interval for the optical pulse to rise (transition) from 10% to 90% and to fall (transition) from 90% to 10% of the pulse amplitude at the Al shall be as specified in Table 18-4.

18.9.2.3 Optical receive contributed systematic jitter

The optical receive systematic jitter measured at the MDI receiver shall be as specified in Table 18-4.

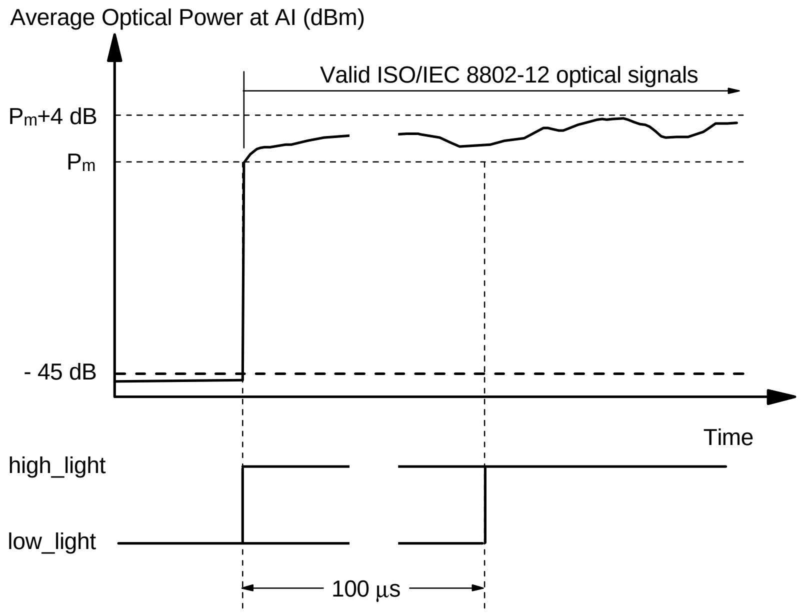
18.9.2.4 Active optical input signals

When presented with NRZ encoded data signals at the AI that are consistent with the parameters defined by Table 18-4, the PMD shall send the data to the RData[3:0] circuits with a BER no greater than 1 error per 10^8 bits.

18.9.2.5High light

A condition of “high_light” shall be detected within 100 µs of valid ISO/IEC 8802-12 optical signals having an average power between P_m and (P_m + 4 dB) appearing at the AI. P_m is the relevant minimum allowed input average optical power for the PMD at the AI as defined in Table 18-4. (The input optical power is averaged over a period of 1 µs or more.)

The condition of “high_light” shall remain asserted while the optical power remains above Pm (see Figure 18-8).



Average Optical Power at AI (dBm)
Valid ISO/IEC 8802-12 optical signals
Pm+4 dB
Pm
- 45 dB
Time
high_light
low_light
100 µs

Figure 18-8—“high_light” condition detect and timing

18.9.2.6Low light

The PMD shall not detect a condition of “low_light” while valid ISO/IEC 8802-12 optical signals with an input average optical power greater than P_m are being received at the AI.

The PMD shall detect a condition of “low_light” within 350 µs after valid ISO/IEC 8802-12 optical signals cease to be present at the AI and the input average optical power, at the AI, has fallen monitonically from a level between (P_m + 4 dB) and P_m to a level below –45 dBm and remained below –45 dBm (see
Figure 18-9). (The input optical power is averaged over a period of 1 µs or more.)

NOTE—The definitions of “high_light” and “low_light” are consistent with Annex G, [B1].

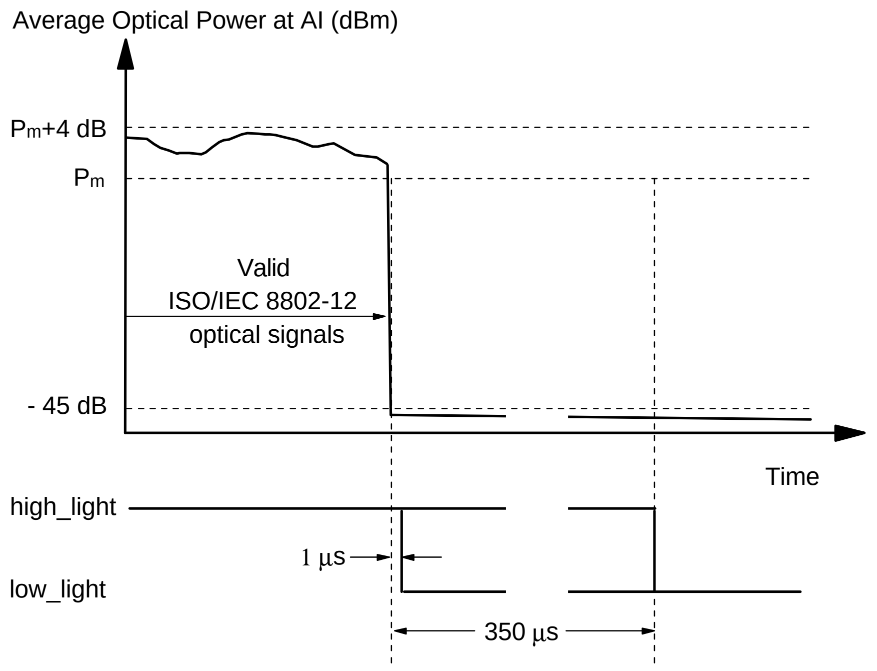


Figure 18-9—"low_light" condition detect and timing

18.10 Characteristics of the fibre-optic medium

The fibre-optic transmission medium consists of the link medium, two end-connectors, and connection devices such as patch panels and wall plates, as shown in Figure 18-2. The fibre-optic medium spans from one MDI to another MDI.

The normal medium for fibre-optic links connected to PMD 800 nm is up to 500 m of fibre-optic cable meeting the specifications of ISO/IEC 11801:1995 for 62.5/125, multimode, graded-index, optical fibre. Other types of multimode fibre are permitted provided the optical specifications of 18.8 are met.

The normal medium for fibre-optic links connected to PMD 1300 nm is up to 2 km of fibre-optic cable meeting the specifications of ISO/IEC 11801:1995 for 62.5/125, multimode, graded-index, optical fibre. Other types of multimode fibre are permitted provided the optical specifications of 18.9 are met.

18.11 MDI specification

This subclause defines the optical and mechanical specifications for the MDI connectors used for the system.

18.11.1 MDI connectors

The MDI connector shall conform to the industry standard duplex SC connector documented in ISO/IEC 11801:1995 and IEC 60874-1:1993 as it relates to intermateability. The MDI, AO, and AI are polarized as shown in Figure 18-10.

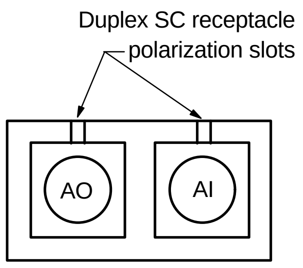


Figure 18-10—MDI duplex SC receptacle

18.12 Environmental specifications

18.12.1 Safety

General environmental and safety specifications are given in Annex B.

18.12.1.1 Isolation requirement

Electrical isolation shall be provided between MDIs attached to the fibre-optic cable. There shall be no conducting path between the optical medium connector plug and any conducting element within the fibre-optic cable. This electrical separation shall withstand at least one of the following electrical strength tests:

a) 1500 Vrms at 50–60 Hz for 60 s, applied as specified in Section 5.3.2 of IEC 60950:1991.

b) 2250 Vdc for 60 s, applied as specified in Section 5.3.2 of IEC 60950:1991.

c) A sequence of ten 2400 V impulses of alternating polarity, applied at intervals of not less than 1 s. The shape of the impulses shall be 1.2/50 μs (1.2 μs virtual front time, 50 μs virtual time of half value), as defined in IEC 60060–1:1989.

There shall be no insulation breakdown, as defined in Section 5.3.2 of IEC 60950:1991, during the test. The resistance after the test shall be at least 2 MΩ, measured at 500 Vdc.

19. 2-TP Physical Medium Dependent (PMD) sublayer, Medium Dependent Interface (MDI), and link specifications

The specification of a PMD to support links of two-pair 100 Ω balanced cable is a matter for future study.

Annex A[28]

(normative)

Protocol Implementation Conformance Statement (PICS) proforma

A.1 Introduction

The supplier of a protocol implementation that is claimed to conform to ISO/IEC 8802-12: 1998 shall complete the following Protocol Implementation Conformance Statement (PICS) proforma.

A completed PICS proforma is the PICS for the implementation in question. The PICS is a statement of which capabilities and options of the protocol have been implemented. The PICS can have a number of uses, including use

a) By the protocol implementor, as a checklist to reduce the risk of failure to conform to the standard through oversight

b) By the supplier and acquirer, or potential acquirer, of the implementation, as a detailed indication of the capabilities of the implementation, stated relative to the common basis for understanding provided by the standard PICS proforma

c) By the user, or potential user, of the implementation, as a basis for initially checking the possibility of interworking with another implementation (note that, while interworking can never be guaranteed, failure to interwork can often be predicted from incompatible PICS proformas

d) By a protocol tester, as the basis for selecting appropriate tests against which to assess the claim for conformance of the implementation

A.2 Abbreviations and special symbols

A.2.1 Status symbols

M	mandatory
O	optional
O.<n>	optional, but support of at least one of the group of options labeled by the same numeral <*n*> is required
pred:	conditional symbol, including predicate identification (see A.3.4)

A.2.2 General abbreviations

N/A	not applicable
PICS	Protocol Implementation Conformance Statement

[28]*Copyright release for PICS proformas:* Users of this standard may freely reproduce the PICS proforma in this annex so that it can be used for its intended purpose and may further publish the completed PICS.

A.3 Instructions for completing the PICS proforma

A.3.1 General structure of the PICS proforma

The first part of the PICS proforma, Implementation Identification and Protocol Summary, is to be completed as indicated with the information necessary to identify fully both the supplier and the implementation.

The main part of the PICS proforma is a fixed-format questionnaire, divided into subclauses, each containing a number of individual items. Answers to the questionnaire items are to be provided in the rightmost column, either by simply marking an answer to indicate a restricted choice (usually Yes or No) or by entering a value or a set or range of values. (Note that there are some items where two or more choices from a set of possible answers can apply. All relevant choices are to be marked, in these cases.)

Each item is identified by an item reference in the first column. The second column contains the question to be answered. The third column contains the reference or references to the material that specifies the item in the main body of ISO/IEC 8802-12: 1998. The remaining columns record the status of the item, i.e., whether support is mandatory, optional, or conditional, and provide the space for the answers (see also A.3.4). Marking an item as supported is to be interpreted as a statement that all the relevant requirements of the subclauses and normative annexes, cited in the References column for the item, are met by the implementation.

A supplier may also provide, or be required to provide, further information, categorized as either Additional Information or Exception Information. When present, each kind of further information is to be provided in a further subclause of items labeled A<*i*> or X<*i*>, respectively, for crossreferencing purposes, where <*i*> is any unambiguous identification for the item (e.g., simply a numeral). There are no other restrictions on its format and presentation.

The PICS proforma for an end-node consists of Clauses A.4, A.5, and A.7. The PICS proforma for a repeater consists of Clauses A.4, A.6, and A.7. In each case, multiple copies of A.7 are used when there is more than one Physical Layer implementation present. The PICS proforma for a PMD consists of Clause A.7.

A completed PICS proforma, including any Additional Information and Exception Information, is the Protocol Implementation Conformation Statement for the implementation in question.

NOTE—Where an implementation is capable of being configured in more than one way, a single PICS may be able to describe all such configurations. However, the supplier has the choice of providing more than one PICS, each covering some subset of the implementation's configuration capabilities, if this makes for easier and clearer presentation of the information.

A.3.2 Additional Information

Items of Additional Information allow a supplier to provide further information intended to assist in the interpretation of the PICS. It is not intended or expected that a large quantity of information will be supplied, and a PICS can be considered complete without any such information. Examples of Additional Information might be an outline of the ways in which an (single) implementation can be set up to operate in a variety of environments and configurations, or information about aspects of the implementation that are outside the scope of this standard but that have a bearing upon the answers to some items.

References to items of Additional Information may be entered next to any answer in the questionnaire, and may be included in items of Exception Information.

A.3.3 Exception Information

It may happen occasionally that a supplier will wish to answer an item with mandatory status (after any conditions have been applied) in a way that conflicts with the indicated requirement. No pre-printed answer will be found in the Support column for this. Instead, the supplier shall write the missing answer into the Support column, together with an X<*i*> reference to an item of Exception Information, and shall provide the appropriate rationale in the Exception Information item itself.

An implementation for which an Exception Information item is required in this way does not conform to ISO/IEC 8802-12: 1998.

NOTE—A possible reason for the situation described above is that a defect in ISO/IEC 8802-12: 1998 has been reported, a correction for which is expected to change the requirement not met by the implementation.

A.3.4 Conditional status

A.3.4.1 Conditional items

The PICS proforma contains a number of conditional items. These are items for which both the applicability of the item itself, and its status if it does apply, mandatory or optional, are dependent upon whether or not certain other items are supported.

Where a group of items is subject to the same condition for applicability, a separate preliminary question about the condition appears at the head of the group, with an instruction to skip to a later point in the questionnaire if the “Not Applicable” answer is selected. Otherwise, individual conditional items are indicated by a conditional symbol in the Status column.

A conditional symbol is of the form “**<pred>:** <S>”, where “**<pred>**” is a predicate as described in A.3.4.2 below, and “<S>” is one of the status symbols M or O.

If the value of the predicate is true (see A.3.4.2), the conditional item is applicable, and its status is given by S: the support column is to be completed in the usual way. Otherwise, the conditional item is not relevant and the Not Applicable (N/A) answer is to be marked.

A.3.4.2 Predicates

A predicate is one of the following:

a) An item-reference for an item in the PICS proforma: the value of the predicate is true if the item is marked as supported, and is false otherwise.

b) A predicate name, for a predicate defined as a Boolean expression constructed by combining item-references using the Boolean operator OR: the value of the predicate is true if one or more of the items is marked as supported.

Each item whose reference is used in a predicate or predicate definition, or in a preliminary question for grouped conditional items, is indicated by an asterisk in the Item column.

A.4 PICS Proforma—ISO/IEC 8802-12: 1998: Identification

A.4.1 Implementation identification

Supplier	
Contact point for queries about the PICS	
Implementation Name(s) and Version(s)	
Other information necessary for full identification e.g., name(s) and version(s) of machines and/or operating system(s), system names	

NOTES

1—Only the first three items are required for all implementations. Other information may be completed as appropriate in meeting the requirement for full identification.

2—The terms Name and Version should be interpreted appropriately to correspond with a supplier's terminology (e.g., Type, Series, Model).

A.4.2 Protocol summary, ISO/IEC 8802-12: 1998

Identification of Protocol Standard	ISO/IEC 8802-12: 1998
Identification of Amendments and Corrigenda to this PICS proforma that have been completed as part of this PICS	Amd. : Corr. : Amd. : Corr. :
Have any Exception items been required? (See A.3.3: the answer Yes means that the implementation does not conform to ISO/IEC 8802-12: 1998.)	Yes ☐ No ☐

Date of statement (dd/mm/yy)	

A.5 End-node PICS proforma

A.5.1 End-node: Major capabilities

Item	Protocol feature	References	Status	Support
	What frame formats are supported:			
EF3	—8802-3 format MAC frames?	10.2	O.1	Yes ☐ No ☐
*EF5	—8802-5 format MAC frames?	10.3	O.1	Yes ☐ No ☐
EFtr	—8802-12 training frames?	10.6	M	Yes ☐
EFv	—8802-12 void frames (reception only)?	10.7	M	Yes ☐
	What link configuration is supported?	11.3		
ELs	—Single links		M	Yes ☐ No ☐
*ELr	—Either single or redundant links		O	Yes ☐ No ☐
	Are the following Management capabilities supported:	13, 13.1.3.2, 13.2.5, E.2		
*EMG	—Basic Capability?		O	Yes ☐ No ☐
EMGc	—Basic Counter Capability?		**EMG:** O	N/A ☐ Yes ☐ No ☐
EMGx	—Expanded Capability?		**EMG:** O	N/A ☐ Yes ☐ No ☐
	What Physical Layer implementations are supported:			
E4	—4-UTP?	16	O.2	Yes ☐ No ☐
E2	—dual simplex STP?	17	O.2	Yes ☐ No ☐
EFO	—dual simplex fibre optic?	18	O.2	Yes ☐ No ☐
EMII	Is an exposed MII supported?	15.3	O.2	Yes ☐ No ☐

A.5.2 Support for MAC functions

Item	MAC function	References	Status	Support
	Are the following MAC functions supported:	11, 11.2, 11.3, 11.4, 11.5, 11.6		
EMtn	—Transmission of normal-priority frames?	11.4.1.1, 11.6	M	Yes ☐
EMth	—Transmission of high-priority frames?	11.4.1.1, 11.6	O	Yes ☐ No ☐
EMrn	—Reception of normal-priority frames?	11.4.1.4, 11.6	M	Yes ☐
EMrh	—Reception of high-priority frames?	11.4.1.4, 11.6	M	Yes ☐
EMtr	—Training?	11.3.1, 11.4.1.2, 11.6	M	Yes ☐
EMrl	—Setting the requested configuration D bit for redundant-link training and responding to the D bit response in the allowed configuration field?	10.6.2.5, 10.6.3.7, 11.3	**Elr:** O	Yes ☐ No ☐
EMrd	—Retrain Delay Mode?	11.3.3, 11.6	M	Yes ☐

A.5.3 Support for MAC addressing functions

Item	Addressing function	References	Status	Support
	Are the following addressing functions supported:			
EAua	Is the end node provided with a universal address?	10.4.1	M	Yes ☐
EAla	Will the end node accept a locally administered address?	10.4.1, 10.4.2	O	Yes ☐ No ☐
EAsp	Use of a specific individual MAC address as Destination Address in received frames and as Source Address in transmitted frames?	10.2.1.1, 10.3.3.1, 10.2.1.2, 10.3.3.2	M	Yes ☐
EAtx	Transmission of frames with other individual MAC address values as Source Address?	10.2.1.2, 10.3.3.2	O	Yes ☐ No ☐
EApm	Reception of frames in promiscuous mode?	10.2.1.1, 10.3.3.1, 10.6.2.2	O	Yes ☐ No ☐
EAtg	Transmission of frames with group MAC addresses as Destination Address?	10.2.1.2, 10.3.3.2	M	Yes ☐
EArg	Reception of frames with group MAC addresses as Destination Address?	10.2.1.1, 10.3.3.1	M	Yes ☐
EAng	State the maximum number of group MAC addresses supported for reception:			Number: ________
	If 8802-5 frame format is supported:			
EAtri	Can the end node transmit frames with the RII bit set to one?	10.3.4	**EF5:** O	N/A ☐ Yes ☐ No ☐
EArri	Does the end node accept received frames with the RII bit set to one?	10.3.4	**EF5:** M	N/A ☐ Yes ☐
EAfa	Are Functional Addresses supported in received frames?	10.3.3.1, Figure 10-10	**EF5:** M	N/A ☐ Yes ☐

A.5.4 MAC timer support

Item	MAC timers	References	Status	(Range)	Support
	Are the following timers supported with values in the specified ranges:	11.5.2.2			
ETrqw	Request Window Duration?		M	40 BT	Yes ☐
ETrrt	MAC Receive Retrain Duration?		M	1–2 ms	Yes ☐
ETrtd	Retraining Delay Duration?		M	1–2 s	Yes ☐
ETtfi	Training Final Idle or Incoming Duration?		M	200–400 µs	Yes ☐
ETtoi	Training Frame Out to In Duration?		M	200–400 µs	Yes ☐
ETtru	Training Up Duration?		M	3–9 µs	Yes ☐
ETtds	TxDisable Duration?		M	200–400 µs	Yes ☐

A.6 Repeater PICS proforma

A.6.1 Repeater: Major capabilities

Item	Protocol feature	References	Status	Support
	What RMAC capability levels are supported:	12.1.1		
RCsr	—Single-repeater network with local port connections to end nodes?	12.7.1, 12.7.7, 12.7.8, 12.7.10	M	Yes ☐
RCtr	—Operation as top repeater in a cascade?	12.7.1, 12.7.6 thru 12.7.8, 12.7.10	O	Yes ☐ No ☐
*RCbr	—Operation as lowest-level repeater in a cascade?	12.7.1 thru 12.7.5, 12.7.7 thru 12.7.11	O	Yes ☐ No ☐
*RCmr	—Operation as an intermediate-level repeater in a cascade?	12.7.1 thru 12.7.11	O	Yes ☐ No ☐
*RCru	—Redundant uplink support for the cascade port?	12.2.1.2, 12.9.7.1,	O	Yes ☐ No ☐
*RCrd	—Redundant downlink support for local ports?	12.9.7.1, 12.9.8	O	Yes ☐ No ☐
	What frame formats are supported:			
RF3	—8802-3 format MAC frames?	10.2	O.1	Yes ☐ No ☐
*RF5	—8802-5 format MAC frames?	10.3	O.1	Yes ☐ No ☐
RFtr	—8802-12 training frames?	10.6	M	Yes ☐
RFv	—8802-12 void frames?	10.7	M	Yes ☐
	Are the following Management capabilities supported:	13, 13.1.3 thru 13.2.4, C.1		
*RMG	—Basic capability?		O	Yes ☐ No ☐
RMGc	—Basic counter capability?		**RMG:** O	N/A ☐ Yes ☐ No ☐
RMGx	—Expanded capability?		**RMG:** O	N/A ☐ Yes ☐ No ☐
RMGl	—link-monitoring capability?		**RMG:** O	N/A ☐ Yes ☐ No ☐

Item	**Protocol feature** (continued)	**References**	**Status**	**Support**
	Are the following training capabilities supported:			
RTdc	—Duplicate-address checking?	10.6.3.7	O	Yes ☐ No ☐
RTnb	—Setting of the “access not allowed” bit?	10.6.3.5	O	Yes ☐ No ☐
RTpp	—Use of private protocol information in training frames?	10.6.4	O	Yes ☐ No ☐
RTul	—Setting a D bit request for the cascade port?	10.6.2.5, 10.6.3.7	O	Yes ☐ No ☐
RTdl	—Recognizing a D bit request from a local port?	10.6.2.5, 10.6.3.7	O	Yes ☐ No ☐
	What Physical Layer implementations are supported:			
E4	—4-UTP?	16	O.2	Yes ☐ No ☐
E2	—Dual simplex STP?	17	O.2	Yes ☐ No ☐
EFO	—Dual simplex fibre optic?	18	O.2	Yes ☐ No ☐
EMII	Is an exposed MII supported?	15.3	O.2	Yes ☐ No ☐

Predicate definition: CP = (RCbr) Or (Rcmr)

A.6.2 Support for RMAC functions and addressing functions

Item	RMAC function	References	Status	Support
	Are the following RMAC functions supported:			
RMrn	—Repeating normal-priority frames?		M	Yes ☐
RMrh	—Repeating high-priority frames?		M	Yes ☐
RMpp	—Packet promotion?		M	Yes ☐
RMvg	—Void-frame generation?		M	Yes ☐
RMrv	—Repeating void frames?		M	Yes ☐
RMtu	—Training the uplink?	12.12	**CP:** M	N/A ☐ Yes ☐
RMtd	—Training downlinks?	12.11.1, 12.11.3	M	Yes ☐
RMru	—Cascade port training for redundant uplinks?	12.2.1.2	**RCru:** O	Yes ☐ No ☐
RMrd	—Local port training for redundant downlinks?	12.9.8	**RCrd:** O	Yes ☐ No ☐
RMsf	—Store-and-forward operation?		M	Yes ☐
	Are end-nodes supported using:			
RAsp	—Specific individual MAC addresses?	12.9.7	O.3	Yes ☐ No ☐
RApm	—Promiscuous mode?	12.9.7	O.3	Yes ☐ No ☐
RAri	If 8802-5 format MAC frames are supported, are frames with the RII bit set repeated?	10.3.4	**RF5:** O	N/A ☐ Yes ☐ No ☐

A.6.3 RMAC timer support

Item	RMAC timers	References	Status	(Value/Range)	Support
	Are the following timers supported with values in the specified ranges:	12.5.1			
Rtid	IDLE_DURATION		M	60–70 BT	Yes ☐
Rtif	IDLE_FILTER		M	60–68 BT	Yes ☐
RTrxl	IN_RX_TOO_LONG		M	84–90 BT	Yes ☐
RTib	SEND_IDLE_BURST		M	24 BT	Yes ☐
RTdw	DELTA_IPG_WINDOW		M	12–16 BT	Yes ☐
RTip	IPG_WINDOW		M	190–194 BT	Yes ☐
RTsi	SENT_INCOMING_LONG_ ENOUGH		M	190–194 BT	Yes ☐
RTfx	FRAME_EXPECTED_TO_SMF_IN		M	450 µs	Yes ☐
RTlr	LONG_RECEIVE_INITIATE_ TRAINING		M		
	For 8802-3 frames			123–124 µs	Yes ☐
	For 8802-5 frames			363–364 µs	Yes ☐
RTvf	VOID_FRAME		M	550–600 octets	Yes ☐
	PACKET_PROMOTION		M	200–300 ms	Yes ☐
	If a cascade port is not supported, predicate CP, mark N/A and continue at A.7:			N/A ☐	
RTpu	PASS_UP_CONTROL_IDLE		M	72 BT	Yes ☐
RTid	PASS_UP_TO_IDLE_DOWN_ TIMEOUT		M	200–400 µs	Yes ☐
RTrw	REQ_WINDOW		M	40 BT	Yes ☐
Rtrl	REQUESTING_TOO_LONG		M	1–2 s	Yes ☐
RTdd	SEC_CTRL_SIG_DECODE_ DELAY		M	84–100 BT	Yes ☐
RTtf	TR_FINAL_IDLE_OR_INCOMING		M	200–400 µs	Yes ☐
RTrt	TR_INTERVAL_TO_RESTART_ TRAINING		M	1–2 s	Yes ☐
RTtx	TR_REQ_UP_TO_FRAME_ RECEIVED		M	200–400 ms	Yes ☐
RTtp	TRAINING_PULSE_SENDING		M	3–9 µs	Yes ☐
RTts	TRAINING_PULSE_TXDISABLE		M	200–400 µs	Yes ☐

A.7 Physical Layer items

A.7.1 PMI and MII

One copy of the following table shall be completed for each different PMD implementation in the end node or repeater. Where there is more than one such PMD, identify unambiguously the associated PMD table:

Identification of PMD with which this table is associated: _____________

Item	Protocol feature	References	Status	Support
	Are the following PMI functions supported:			
Ptcs	—TCS transfer	14.3.1	M	Yes ☐
Prcs	—RCS transfer	14.3.2	M	Yes ☐
Prg	—RxGrant transfer	14.3.3	M	Yes ☐
Pte	—TxEn control	14.3.4	M	Yes ☐
Pre	—RxEn control, including deassertion on end of received frame	14.3.5	M	Yes ☐
Pcfg	—Configuration	14.3.6	O	Yes ☐ No ☐
Ptcg	TxClk generation	14.4.1.1	M	Yes ☐
	Data transmission:	14.4.2, 14.4.3		
Ptqa	—Quintet assembly	14.4.2.1	M	Yes ☐
Ptqs	—Quintet streaming	14.4.2.1.1	M	Yes ☐
Ptqc	—Channel quintet ciphering	14.4.2.1.2	M	Yes ☐
Ptqe	—Quintet to sextet encoding	14.4.2.2	M	Yes ☐
Ptsf	—Generation of start fill, preamble and ssd	14.4.2.3.1, 14.4.2.3.2, 14.4.2.3.3	M	Yes ☐
Ptef	—Generation of esd and end fill	14.4.2.3.5, 14.4.2.3.6	M	Yes ☐
	Data reception:			
Prsf	—Detection of preamble and ssd	14.4.4.1.1	M	Yes ☐
Prqd	—Sextet to quintet decoding	14.4.4.2	M	Yes ☐
Prqx	—Quintet-stream deciphering	14.4.4.3.1	M	Yes ☐
Prqm	—Quintet merging	14.4.4.3.2	M	Yes ☐
Proa	—Octet assembly	14.4.4.3	M	Yes ☐
Pref	—Detection of esd	14.4.4.1.3	M	Yes ☐
Pred	—Error detection	14.4.4.1.1, 14.4.4.1.3, 14.4.4.2, 14.4.4.3.2, 14.5	M	Yes ☐
Pmii	Is an exposed MII supported?	15.3	O	Yes ☐ No ☐

A.7.2 PMD/MDI for 4-UTP links

One copy of the following table shall be completed for each different 4-UTP implementation in the end node or repeater. Where there is more than one PMD implementation in the end node or repeater, indicate here the identification used to associate this PMD with its corresponding copy of the PMI table:

Identification of this PMD, for association with PMI table: ________________

Item	PMD/MDI feature, 4-UTP	References	Status	Support
M4mii	Is an exposed MII supported?	15.3	O.1	Yes ☐ No ☐
M4er	Is the PMD part of an end node or repeater?		O.1	Yes ☐ No ☐
M4cfc	Is the PMD/Link configuration circuit supported?	16.2.4	O	Yes ☐ No ☐
	Are the PMD transmit functions supported:	16.5, 16.7, Figures 16-5, 16-6, 16-7		
M4tec	—Data and control signal encoding?	16.5.1	M	Yes ☐
M4tds	—Data signaling?	16.5.2	M	Yes ☐
M4tcs	—Control signaling?	16.5.3, 16.5.3.1, 16.5.3.2	M	Yes ☐
M4tdc	—Transition between data and control signaling?	16.5.4	M	Yes ☐
	Are the PMD receive functions supported:	16.6, 16.7, Figure 16-8		
M4rrd	—Control signal recovery and decoding?	16.6.1	M	Yes ☐
M4rsm	—Signal monitoring?	16.6.2	M	Yes ☐
M4rcr	—Clock recovery?	16.6.3	M	Yes ☐
M4rdr	—Data recovery?	16.6.4	M	Yes ☐
M4rgd	—Grant detection?	16.6.5	M	Yes ☐
M4rsi	—Receiver Control State indication?	16.6.6	M	Yes ☐
M4isl	Is the isolation requirement met?	16.8.1	M	Yes ☐
	Are the following transmitter specifications met:	16.8.2,		
M4dov	—Transmitter differential output voltage?	16.8.2.1	M	Yes ☐
M4toj	—Transmitter output jitter?	16.8.2.2	M	Yes ☐
M4doi	—Transmitter differential output impedance?	16.8.2.3	M	Yes ☐
M4tib	—Transmitter impedance balance?	16.8.2.4	M	Yes ☐
M4tft	—Transmitter fault tolerance?	16.8.2.5	M	Yes ☐

Item	PMD/MDI feature, 4-UTP	References	Status	Support
	Are the following receiver specifications met:	16.8.3,		
M4rp	—Receiver performance?	16.8.3.1	M	Yes ☐
M4ni	—Receiver noise immunity?	16.8.3.2	M	Yes ☐
M4dis	—Receiver differential input signals?	16.8.3.3	M	Yes ☐
M4dii	—Receiver differential input impedance?	16.8.3.4	M	Yes ☐
M4rib	—Receiver impedance balance?	16.8.3.5	M	Yes ☐
M4rft	—Receiver fault tolerance?	16.8.3.6	M	Yes ☐
M4con	Is the specified MDI connector used?	16.10.1	M	Yes ☐
M4cpa	Are the specified MDI connector pin assignments used?	16.10.2	M	Yes ☐

A.7.3 PMD/MDI for dual simplex STP links

One copy of the following table shall be completed for each different dual simplex STP implementation in the end node or repeater. Where there is more than one PMD implementation in the end node or repeater, indicate here the identification used to associate this PMD with its corresponding copy of the PMI table:

Identification of this PMD, for association with PMI table: ________________

Item	PMD/MDI feature, dual simplex STP	References	Status	Support
M2mii	Is an exposed MII supported?	15.3	O.1	Yes ☐ No ☐
M2er	Is the PMD part of an end node or repeater?		O.1	Yes ☐ No ☐
M2cfc	Is the PMD/Link configuration circuit supported?	17.2.4	O	Yes ☐ No ☐
	Are the PMD transmit functions supported:	17.5, 17.7, Figures 17-3, 17-5		
M2tmx	—Multiplexing physical frames?	17.5.1	M	Yes ☐
M2tec	—Data and control signal encoding?	17.5.2	M	Yes ☐
M2tds	—Data signaling?	17.5.3	M	Yes ☐
M2tcs	—Control signaling?	17.5.4, 17.5.4.1, 17.5.4.2	M	Yes ☐
M2tdc	—Transition between data and control signaling?	17.5.5	M	Yes ☐
	Are the PMD receive functions supported:	17.6, 17.7, Figure 17-4		
M2rrd	—Control signal recovery and decoding?	17.6.1	M	Yes ☐
M2rsm	—Signal monitoring?	17.6.2	M	Yes ☐
M2rcr	—Clock recovery?	17.6.3	M	Yes ☐
M2rdr	—Data recovery?	17.6.4	M	Yes ☐
M2rdm	—Demultiplexing of physical frames?	17.6.5	M	Yes ☐
M2rgd	—Grant detection?	17.6.6	M	Yes ☐
M2rsi	—Receiver Control State indication?	17.6.7	M	Yes ☐
M2isl	Is the isolation requirement met?	17.8.1	M	Yes ☐
	Are the following transmitter specifications met:	17.8.2,		
M2dov	—Transmitter differential output voltage?	17.8.2.1	M	Yes ☐
M2doi	—Transmitter differential output impedance?	17.8.2.2	M	Yes ☐
M2tft	—Transmitter fault tolerance?	17.8.2.3	M	Yes ☐
	Are the following receiver specifications met:	17.8.3,		
M2dis	—Receiver differential input signals?	17.8.3.1	M	Yes ☐
M2req	—Receiver equalization?	17.8.3.2	M	Yes ☐
M2rdi	—Receiver differential impedance?	17.8.3.3	M	Yes ☐
M2rft	—Receiver fault tolerance?	17.8.3.4	M	Yes ☐
M2con	Is a 9-pin D shell MDI connector used?	17.10.1	O	Yes ☐ No ☐
M2ccw	If a 9-pin D shell connector is used, is it compliant with ISO 4902:1989 and wired according to table 23?	17.10.1, 17.10.2	**M2con:** M	N/A ☐ Yes ☐

A.7.4 PMD/MDI for dual simplex fibre optic links

One copy of the following table shall be completed for each different dual simplex fibre optic implementation in the end node or repeater. Where there is more than one PMD implementation in the end node or repeater, indicate here the identification used to associate this PMD with its corresponding copy of the PMI table:

Identification of this PMD, for association with PMI table: ___________________

Item	PMD/MDI feature, dual simplex fibre optic	References	Status	Support
MFmii	Is an exposed MII supported?	15.3	O.1	Yes ☐ No ☐
MFer	Is the PMD part of an end node or repeater?		O.1	Yes ☐ No ☐
MFcfc	Is the PMD/Link configuration circuit supported?	18.2.4	O	Yes ☐ No ☐
	Are the PMD transmit functions supported:	18.5, 18.7, Figures 18-3, 18-5		
MFtmx	—Multiplexing physical frames?	18.5.1	M	Yes ☐
MFtec	—Data and control signal encoding?	18.5.2	M	Yes ☐
MFtds	—Data signaling?	18.5.3	M	Yes ☐
MFtcs	—Control signaling?	18.5.4, 18.5.4.1, 18.5.4.2	M	Yes ☐
MFtdc	—Transition between data and control signaling?	18.5.5	M	Yes ☐
MFtxd	—Disable?	18.5.7	M	Yes ☐
	Are the PMD receive functions supported:	18.6, 18.7, Figure 18-4		
MFrrd	—Control signal recovery and decoding?	18.6.1	M	Yes ☐
MFrsd	—Signal detection?	18.6.2	M	Yes ☐
MFrsm	—Signal monitoring?	18.6.3	M	Yes ☐
MFrcr	—Clock recovery?	18.6.4	M	Yes ☐
MFrdr	—Data recovery?	18.6.5	M	Yes ☐
MFrdm	—Demultiplexing of physical frames?	18.6.6	M	Yes ☐
MFrgd	—Grant detection?	18.6.7	M	Yes ☐
MFrsi	—Receiver Control State indication?	18.6.8	M	Yes ☐
MFisl	Is the isolation requirement met?	18.12.1.1	M	Yes ☐

Item	PMD/MDI feature, dual simplex fibre optic (continued)	References	Status	Support
	If the supported transmitter wavelength is 800 nm, are the following transmitter specifications met:	18.8.1,		
MF8cwl	—Center wavelength?	18.8.1.1	M	Yes ☐
MF8spw	—Spectral width?	18.8.1.2	M	Yes ☐
MF8top	—Average transmit optical power?	18.8.1.3	M	Yes ☐
MF8apz	—Average optical power of logic zero?	18.8.1.4	M	Yes ☐
MF8trf	—Optical transmit pulse rise and fall times?	18.8.1.5	M	Yes ☐
MF8tpo	—Optical transmit pulse overshoot?	18.8.1.6	M	Yes ☐
MF8trj	—Optical transmit contributed random jitter?	18.8.1.7	M	Yes ☐
MF8tsj	—Optical transmit contributed systematic jitter?	18.8.1.8	M	Yes ☐
	Are the following receiver specifications met:	18.8.2,		
MF8arp	—Average receive optical power?	18.8.2.1	M	Yes ☐
MF8rrf	—Optical receive pulse rise and fall times?	18.8.2.2	M	Yes ☐
MF8rsj	—Optical receive contributed systematic jitter?	18.8.2.3	M	Yes ☐
MF8ois	—Active optical input signals?	18.8.2.4	M	Yes ☐
MF8hld	—High light detection?	18.8.2.5	M	Yes ☐
MF8lld	—Low light detection?	18.8.2.6	M	Yes ☐
	If the supported transmitter wavelength is 1300 nm, are the following transmitter specifications met:	18.9.1,		
MF13cwl	—Center wavelength?	18.9.1.1	M	Yes ☐
MF13spw	—Spectral width?	18.9.1.2	M	Yes ☐
MF13top	—Average transmit optical power?	18.9.1.3	M	Yes ☐
MF13apz	—Average optical power of logic zero?	18.9.1.4	M	Yes ☐
MF13trf	—Optical transmit pulse rise and fall times?	18.9.1.5	M	Yes ☐
MF13tpo	—Optical transmit pulse overshoot?	18.9.1.6	M	Yes ☐
MF13trj	—Optical transmit contributed random jitter?	18.9.1.7	M	Yes ☐
MF13tsj	—Optical transmit contributed systematic jitter?	18.9.1.8	M	Yes ☐
	Are the following receiver specifications met:	18.9.2,		
MF13arp	—Average receive optical power?	18.9.2.1	M	Yes ☐
MF13rrf	—Optical receive pulse rise and fall times?	18.9.2.2	M	Yes ☐
MF13rsj	—Optical receive contributed systematic jitter?	18.9.2.3	M	Yes ☐
MF13ois	—Active optical input signals?	18.9.2.4	M	Yes ☐
MF13hld	—High light detection?	18.9.2.5	M	Yes ☐
MF13lld	—Low light detection?	18.9.2.6	M	Yes ☐
MFcon	Is the specified MDI connector used?	18.11.1	M	Yes ☐

Annex B

(normative)

General environmental and safety specifications

B.1 General safety

All equipment meeting this standard shall conform to IEC 60950:1991.

B.2 Network safety

This clause sets forth a number of recommendations and guidelines related to safety concerns; the list is neither complete nor does it address all possible safety issues. The designer is urged to consult the relevant local, national, and international safety regulations to ensure compliance with the appropriate requirements.

LAN cable systems described in this clause are subject to at least four direct electrical safety hazards during their installation and use. These hazards are:

a) Direct contact between LAN components and power, lighting, or communication circuits.
b) Static charge buildup on LAN cables and components.
c) High-energy transients coupled onto the LAN cable system.
d) Voltage potential differences between safety grounds to which various LAN components are connected.

Such electrical safety hazards must be avoided or appropriately protected against for proper network installation and performance. In addition to provisions for proper handling of these conditions in an operational system, special measures must be taken to ensure that the intended safety features are not negated during installation of a new network or during modification or maintenance of an existing network. Isolation requirements for 4-UTP networks and STP networks are defined in 16.8.1 and 17.8.1, respectively.

B.2.1 Installation

Sound installation practice, as defined by applicable local codes and regulations, shall be followed in every instance in which such practice is applicable.

B.2.2 Grounding

There are significant safety considerations associated with bonding cable shields and other metallic parts to ground that need to be taken into account. Cable shields and connector shells shall be earthed in compliance with all local regulations and safety practices.

B.2.3 Installation and maintenance guidelines

During installation and maintenance of the cable plant, care shall be taken to ensure that uninsulated network cable conductors do not make electrical contact with unintended conductors or ground.

B.2.4 Telephony voltages

The use of building wiring brings with it the possibility of wiring errors that may connect telephony voltages to demand-priority equipment. Other than voice signal (which is low voltage), the primary voltages that may be encountered are the “battery” and ringing voltages. Although there is no universal standard, the following maximums generally apply.

Battery voltage to a telephone line is generally 56 Vdc applied to the line through a balanced 400 Ω source impedance.

Ringing voltage is a composite signal consisting of an ac component and a dc component. The ac component is up to 175 V peak at 20–60 Hz with a 100 Ω source resistance. The dc component is 56 Vdc with a 300–600 Ω source resistance. Large reactive transients can occur at the start and end of each ring interval.

Although demand-priority equipment is not required to survive such wiring hazards without damage, application of any of the above voltages shall not result in any safety hazard.

NOTE—Wiring errors may impose telephony voltages differentially across demand-priority transmitters and receivers. Because the termination resistance likely to be present across a receiver’s input is of substantially lower impedance than an off-hook telephone instrument, receivers will generally appear to the telephone system as off-hook telephones. Therefore, full-ring voltages will be applied for only short periods. Transmitters that are coupled using transformers will similarly appear like off-hook telephones (though perhaps more slowly) due to the low resistance of the transformer coil.

B.3 Environment

B.3.1 Electromagnetic compatibility

The PMD shall comply with all applicable local, national, and international regulations (such as CISPR Publication 22:1993) concerning electromagnetic emissions and susceptibility to electromagnetic interference.

B.3.2 Temperature and humidity

The PMD is expected to operate over a reasonable range of environmental conditions related to temperature, humidity, and physical handling (such as shock and vibration). The specific requirements and values for these parameters are considered to be beyond the scope of this International Standard.

Annex C

(normative)

GDMO Specifications for Demand-Priority Managed Objects

C.1 Repeater GDMO specification

This clause formally defines the Repeater Management objects using the templates specified in ISO/IEC 10165-4:1992. This specification uses common definitions and procedures from IEEE Std 802.1F-1993.

The protocol encodings for CMIP, and therefore ISO/IEC 15802-2:1995 can be derived from these definitions directly. The template defined in ISO/IEC 10165-4:1992 specifies precisely the syntax used in this standard to define the operation, objects, and attributes. The application of a GDMO template compiler against these definitions will produce the proper protocol encodings.

Each attribute definition in this clause references directly by means of the WITH ATTRIBUTE SYNTAX construct or indirectly by means of the DERIVED FROM construct an ASN.1 type or subtype that defines the attribute's type and range. Those ASN.1 types and subtypes defined exclusively for Demand Priority Management appear in a single ASN.1 module in C.4.

C.1.1 Repeater managed object class

C.1.1.1 Repeater, formal definition

```
oRepeater                  MANAGED OBJECT CLASS
    DERIVED FROM           "ITU-T Rec. X.721 (1992) | ISO/IEC 10165-2:1992":top;
    CHARACTERIZED BY

        pRepeaterBasic     PACKAGE
            ATTRIBUTES     aCurrentFramingType         GET,
                           aGroupMap                   GET,
                           aMACAddress                 GET,
                           aRepeaterGroupCapacity      GET,
                           aRepeaterHealthData         GET,
                           aRepeaterHealthState        GET,
                           aRepeaterHealthText         GET,
                           aRepeaterID                 GET,
                           aRMACVersion                GET;
            ACTIONS        acExecuteNonDisruptiveSelfTest,
                           acResetRepeater
                           ;
            NOTIFICATIONS  nGroupMapChange,
                           nRepeaterHealth,
                           nRepeaterReset;
        ;
    ;
    CONDITIONAL PACKAGES

        pRepeaterExpanded  PACKAGE
            ATTRIBUTES     aDesiredFramingType         GET-REPLACE,
                           aFramingCapability          GET,
                           aRepeaterSearchAddress      GET,
                           aRepeaterSearchGroup        GET,
                              aRepeaterSearchPort          GET,
                              aRepeaterSearchState         GET;
                ACTIONS       acRepeaterSearchAddress;
                REGISTERED AS {iso(1) member-body(2) us(840) 802dot12(10037)
                              repeaterMgt(8) package(4) repeaterExpandedPkg(2)};
                PRESENT IF    !The Expanded Capability is implemented.!;
        REGISTERED AS         {iso(1) member-body(2) us(840) 802dot12(10037)
                              repeaterMgt(8) managedObjectClass(3)
                              repeaterObjectClass(1)};
```

```
nbRepeaterName                NAME BINDING
        SUBORDINATE OBJECT CLASS
                              repeater;
        NAMED BY SUPERIOR OBJECT CLASS
                              "ISO/IEC 10165-2":system AND SUBCLASSES;
        WITH ATTRIBUTE        aRepeaterID;
        REGISTERED AS         {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                          nameBinding(6) repeaterName(1)};
```

```
nbRepeaterMonitor             NAME BINDING
        SUBORDINATE OBJECT CLASS
                              "IEEE802.1F":oEWMAMetricMonitor;
        NAMED BY SUPERIOR OBJECT CLASS
                              "ISO/IEC 10165-2":system AND SUBCLASSES;
        WITH ATTRIBUTE        aScannerId;
                CREATE        WITH-AUTOMATIC-INSTANCE-NAMING;
                DELETE        ONLY-IF-NO-CONTAINED-OBJECTS;
        REGISTERED AS         {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                          nameBinding(6) repeaterMonitor(2)};
```

C.1.1.2 Repeater attributes

```
aCurrentFramingType           ATTRIBUTE
        WITH ATTRIBUTE SYNTAX IEEE802dot12-MgmtAttributeModule.OneFramingType;
        MATCHES FOR           EQUALITY;
        BEHAVIOUR             bCurrentFramingType;
        REGISTERED AS         {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                          attribute(7) currentFramingType(3)};

        bCurrentFramingType   BEHAVIOUR
        DEFINED AS         !See "aCurrentFramingType BEHAVIOUR DEFINED AS" in
                           3.2.4.2.1!;
```

```
aDesiredFramingType           ATTRIBUTE
        WITH ATTRIBUTE SYNTAX IEEE802dot12-MgmtAttributeModule.OneFramingType;
        MATCHES FOR           EQUALITY;
        BEHAVIOUR             bDesiredFramingType;
        REGISTERED AS         {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                          attribute(7) desiredFramingType(4)};

        bDesiredFramingType   BEHAVIOUR
        DEFINED AS         !See "aDesiredFramingType BEHAVIOUR DEFINED AS" in
                           13.2.4.2.1!;
```

```
aFramingCapability            ATTRIBUTE
      WITH ATTRIBUTE SYNTAX   IEEE802dot12-MgmtAttributeModule.FramingType;
      MATCHES FOR             EQUALITY;
      BEHAVIOUR               bFramingCapability;
      REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                        attribute(7) framingCapability(5)};

      bFramingCapability      BEHAVIOUR
      DEFINED AS        !See "aFramingCapability BEHAVIOUR DEFINED AS" in
                        13.2.4.2.1!;

aGroupMap                     ATTRIBUTE
      WITH ATTRIBUTE SYNTAX   IEEE802dot12-MgmtAttributeModule.BitString;
      MATCHES FOR             EQUALITY;
      BEHAVIOUR               bGroupMap;
      REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                        attribute(7) groupMap(7)};

      bGroupMap               BEHAVIOUR
      DEFINED AS        !See "aGroupMap BEHAVIOUR DEFINED AS" in 13.2.4.2.1!;

aMACAddress                   ATTRIBUTE
      WITH ATTRIBUTE SYNTAX   IEEE802CommonDefinitions.MACAddress;
      MATCHES FOR             EQUALITY;
      BEHAVIOUR               bMACAddress;
      REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                        attribute(7) macAddress(2)};
      bMACAddress             BEHAVIOUR
      DEFINED AS        !See "aMACAddress BEHAVIOUR DEFINED AS" in 13.2.4.2.1!;

aRepeaterGroupCapacity        ATTRIBUTE
      WITH ATTRIBUTE SYNTAX   IEEE802dot12-MgmtAttributeModule.OneOfName;
      MATCHES FOR             EQUALITY, ORDERING;
      BEHAVIOUR               bRepeaterGroupCapacity;
      REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                        attribute(7) repeaterGroupCapacity(6)};

      bRepeaterGroupCapacity  BEHAVIOUR
      DEFINED AS        !See "aRepeaterGroupCapacity BEHAVIOUR DEFINED AS"
                        in 13.2.4.2.1.!;

aRepeaterHealthData           ATTRIBUTE
      WITH ATTRIBUTE SYNTAX   IEEE802dot12-MgmtAttributeModule.RepeaterHealthData;
      MATCHES FOR             EQUALITY;
      BEHAVIOUR               bRepeaterHealthData;
      REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                        attribute(7) repeaterHealthData(10)};

      bRepeaterHealthData     BEHAVIOUR
      DEFINED AS        !See "aRepeaterHealthData BEHAVIOUR DEFINED AS" in
                        13.2.4.2.1.!;
```

```
aRepeaterHealthState          ATTRIBUTE
      WITH ATTRIBUTE SYNTAX   IEEE802dot12-MgmtAttributeModule.RepeaterHealthState;
      MATCHES FOR             EQUALITY;
      BEHAVIOUR               bRepeaterHealthState;
      REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                        attribute(7) repeaterHealthState(8)};

      bRepeaterHealthState    BEHAVIOUR
      DEFINED AS        !See "aRepeaterHealthState BEHAVIOUR DEFINED AS" in
                        13.2.4.2.1.!;

aRepeaterHealthText           ATTRIBUTE
      WITH ATTRIBUTE SYNTAX   IEEE802dot12-MgmtAttributeModule.RepeaterHealthText;
      MATCHES FOR             EQUALITY;
      BEHAVIOUR               bRepeaterHealthText;
      REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                        attribute(7) repeaterHealthText(9)};

      bRepeaterHealthText     BEHAVIOUR
      DEFINED AS        !See "aRepeaterHealthText BEHAVIOUR DEFINED AS" in
                        13.2.4.2.1.!;

aRepeaterID                   ATTRIBUTE
      WITH ATTRIBUTE SYNTAX   IEEE802dot12-MgmtAttributeModule.OneOfName;
      MATCHES FOR             EQUALITY;
      BEHAVIOUR               bRepeaterID;
      REGISTERED AS     {     iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                        attribute(7) repeaterID(1)};

      bRepeaterID             BEHAVIOUR
      DEFINED AS        !See "aRepeaterID BEHAVIOUR DEFINED AS" in
                        13.2.4.2.1.!;

aRepeaterSearchAddress        ATTRIBUTE
      WITH ATTRIBUTE SYNTAX   IEEE802CommonDefinitions.MACAddress;
      MATCHES FOR             EQUALITY;
      BEHAVIOUR               bRepeaterSearchAddress;
      REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                        attribute(7) repeaterSearchAddress(11)};

      bRepeaterSearchAddress  BEHAVIOUR
      DEFINED AS        !See "aRepeaterSearchAddress BEHAVIOUR DEFINED AS"
                              in 13.2.4.2.1.!;

aRepeaterSearchGroup          ATTRIBUTE
      WITH ATTRIBUTE SYNTAX   IEEE802dot12-MgmtAttributeModule.OneOfName;
      MATCHES FOR             EQUALITY, ORDERING;
      BEHAVIOUR               bRepeaterSearchGroup;
      REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                        attribute(7) repeaterSearchGroup(13)};

      bRepeaterSearchGroup    BEHAVIOUR
      DEFINED AS        !See "aRepeaterSearchGroup BEHAVIOUR DEFINED AS"
                        in 13.2.4.2.1.!;
```

```
aRepeaterSearchPort              ATTRIBUTE
      WITH ATTRIBUTE SYNTAX     IEEE802dot12-MgmtAttributeModule.OneOfName;
      MATCHES FOR               EQUALITY, ORDERING;
      BEHAVIOUR                 bRepeaterSearchPort;
      REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                          attribute(7) repeaterSearchPort(14)};

      bRepeaterSearchPort       BEHAVIOUR
      DEFINED AS           !See “aRepeaterSearchPort BEHAVIOUR DEFINED AS”
                           in 13.2.4.2.1.!;

aRepeaterSearchState             ATTRIBUTE
      WITH ATTRIBUTE SYNTAX     IEEE802dot12-MgmtAttributeModule.SearchState;
      MATCHES FOR               EQUALITY;
      BEHAVIOUR                 bRepeaterSearchState;
      REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                          attribute(7) repeaterSearchState(12)};

      bRepeaterSearchState      BEHAVIOUR
      DEFINED AS           !See “aRepeaterSearchState BEHAVIOUR DEFINED AS”
                           in 13.2.4.2.1.!;

aRMACVersion                     ATTRIBUTE
      WITH ATTRIBUTE SYNTAX     IEEE802dot12-MgmtAttributeModule.
                                VersionBitString;
      BEHAVIOUR                 bRMACVersion;
      REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037)
                           repeaterMgt(8) attribute(7) rmacVersion(57) };

      bRMACVersion              BEHAVIOUR
      DEFINED AS           !See "aRMACVersion BEHAVIOUR DEFINED AS"
                           in 13.2.4.2.1.!;
```

C.1.1.3 Repeater actions

```
acExecuteNonDisruptiveSelfTest  ACTION
      BEHAVIOUR                 bExecuteNonDisruptiveSelfTest;
      REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                          action(9) executeNonDisruptiveSelfTestAction(2)};

      bExecuteNonDisruptiveSelfTest    BEHAVIOUR
      DEFINED AS           !See “acExecuteNonDisruptiveSelftest BEHAVIOUR DEFINED
                           AS” in 13.2.4.2.2.!;

acRepeaterSearchAddress          ACTION
      BEHAVIOUR                 bAcRepeaterSearchAddress;
      REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                          action(9) repeaterSearchAddressAction(3)};

      bAcRepeaterSearchAddress         BEHAVIOUR
      DEFINED AS           !See “acRepeaterSearchAddress BEHAVIOUR DEFINED AS” in
                           13.2.4.2.2.!;
```

```
acResetRepeater                 ACTION
      BEHAVIOUR                 bResetRepeater;
      MODE                      CONFIRMED;
      REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                          action(9) resetRepeater(1)};

      bResetRepeater            BEHAVIOUR
      DEFINED AS            !See "acResetRepeater BEHAVIOUR DEFINED AS" in
                                13.2.4.2.2.!;
```

C.1.1.4 Repeater notifications

```
nGroupMapChange                 NOTIFICATION
      BEHAVIOUR                 bGroupMapChange;
      WITH INFORMATION SYNTAX         IEEE802dot12-MgmtAttributeModule.BitString;
      REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                          notification(10) groupMapChange(3)};

      bGroupMapChange           BEHAVIOUR
      DEFINED AS            !See "nGroupMapChange BEHAVIOUR DEFINED AS" in
                            13.2.4.2.3.!;

nRepeaterHealth                 NOTIFICATION
      BEHAVIOUR                 bRepeaterHealth;
      WITH INFORMATION SYNTAX         IEEE802dot12-
                                      MgmtAttributeModule.RepeaterHealthInfo
      AND ATTRIBUTE IDS         fnRepeaterHealthState aRepeaterHealthState,
                                fnRepeaterHealthText aRepeaterHealthText,
                                fnRepeaterHealthData aRepeaterHealthData;
      REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                          notification(10) repeaterHealth(1)};

      bRepeaterHealth           BEHAVIOUR
      DEFINED AS            !See "nRepeaterHealth BEHAVIOUR DEFINED AS" in
                            13.2.4.2.3.!;

nRepeaterReset                  NOTIFICATION
      BEHAVIOUR                 bRepeaterReset;
      WITH INFORMATION SYNTAX         IEEE802dot12-
                                      MgmtAttributeModule.RepeaterHealthInfo
      AND ATTRIBUTE IDS               fnRepeaterHealthState aRepeaterHealthState,
                                      fnRepeaterHealthText aRepeaterHealthText,
                                      fnRepeaterHealthData aRepeaterHealthData;
      REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                          notification(10) repeaterReset(2)};

      bRepeaterReset            BEHAVIOUR
      DEFINED AS            !See "nRepeaterReset BEHAVIOUR DEFINED AS" in
                            13.2.4.2.3.!;
```

C.1.2 ResourceTypeID managed object class

C.1.2.1 ResourceTypeID, formal definition

Implementation of this managed object in accordance with the definition contained in IEEE Std 802.1F-1993 is a conformance requirement of this standard.

NOTE—A single instance of the Resource Type ID managed object exists within the Repeater managed object class. The managed object itself is contained in IEEE Std. 802.1F-1993; therefore, only the name binding appears in this standard**.**

```
nbResourceTypeId                NAME BINDING
          SUBORDINATE OBJECT CLASS
                                "IEEE802.1F":oResourceTypeID;
          NAMED BY SUPERIOR OBJECT CLASS
                                oRepeater AND SUBCLASSES;
          WITH ATTRIBUTE        "IEEE802.1F":aResourceTypeIDName;
          REGISTERED AS         {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                            nameBinding(6) resourceTypeID(3)};
```

C.1.3 Group managed object class

C.1.3.1 Group, formal definition

```
oGroup                          MANAGED OBJECT CLASS
          DERIVED FROM          "ITU-T Rec. X.721 (1992) | ISO/IEC 10165-2:1992":top;
          CHARACTERIZED BY
                    pGroupBasicControl  PACKAGE
                              ATTRIBUTES  aGroupCablesBundled   GET-REPLACE,
                                          aGroupID              GET,
                                          aGroupPortCapacity    GET,
                                          aPortMap              GET;
                              NOTIFICATIONS nPortMapChange;
                    ;
          ;

          REGISTERED AS         {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                            managedObjectClass(3) groupObjectClass(2)};

nbGroupName                     NAME BINDING
          SUBORDINATE OBJECT CLASS
                                oGroup;
          NAMED BY SUPERIOR OBJECT CLASS
                                oRepeater AND SUBCLASSES;
          WITH ATTRIBUTE        aGroupID;
          REGISTERED AS         {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                            nameBinding(6) groupName(4)};
```

C.1.3.2 Group attributes

```
aGroupCablesBundled              ATTRIBUTE
      WITH ATTRIBUTE SYNTAX   IEEE802dot12-MgmtAttributeModule.CableBundling;
      MATCHES FOR             EQUALITY;
      BEHAVIOUR               bGroupCablesBundled;
      REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                        attribute(7) groupCablesBundled(52)};

      bGroupCablesBundled         BEHAVIOUR
      DEFINED AS            !See "aGroupCablesBundled BEHAVIOUR DEFINED AS" in
                            13.2.4.4.1.!;

aGroupID                      ATTRIBUTE
      WITH ATTRIBUTE SYNTAX   IEEE802dot12-MgmtAttributeModule.OneOfName;
      MATCHES FOR             EQUALITY;
      BEHAVIOUR               bGroupID;
      REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                        attribute(7) groupID(15)};

      bGroupID                BEHAVIOUR
      DEFINED AS            !See "aGroupID BEHAVIOUR DEFINED AS" in 13.2.4.4.1.!;

aGroupPortCapacity            ATTRIBUTE
      WITH ATTRIBUTE SYNTAX   IEEE802dot12-MgmtAttributeModule.OneOfName;
      MATCHES FOR             EQUALITY, ORDERING;
      BEHAVIOUR               bGroupPortCapacity;
      REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                        attribute(7) groupPortCapacity(16)};

      bGroupPortCapacity      BEHAVIOUR
      DEFINED AS            !See "aGroupPortCapacity BEHAVIOUR DEFINED AS" in
                            13.2.4.4.1.!;

aPortMap                      ATTRIBUTE
      WITH ATTRIBUTE SYNTAX   IEEE802dot12-MgmtAttributeModule.BitString;
      MATCHES FOR             EQUALITY;
      BEHAVIOUR               bPortMap;
      REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                        attribute(7) portMap(17)};

      bPortMap                BEHAVIOUR
      DEFINED AS            !See "aPortMap BEHAVIOUR DEFINED AS" in 13.2.4.4.1.!;
```

C.1.3.3 Group notifications

```
nPortMapChange                NOTIFICATION
      BEHAVIOUR               bPortMapChange;
      WITH INFORMATION SYNTAX
                              IEEE802dot12-MgmtAttributeModule.BitString;
      REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                        notification(10) portMapChange(46)};

      bPortMapChange          BEHAVIOUR
      DEFINED AS        !See "nPortMapChange BEHAVIOUR DEFINED AS" in
                              13.2.4.4.3.!;
```

C.1.4 Port managed object class

C.1.4.1 Port, formal definition

```
oPort                     MANAGED OBJECT CLASS
    DERIVED FROM          "ITU-T Rec. X.721 (1992) | ISO/IEC 10165-2:1992":top;
    CHARACTERIZED BY

        pPortBasicPkg     PACKAGE
            ATTRIBUTES    aPortID                          GET,
                          aPortAdministrativeState         GET,
                          aPortType                        GET,
                          aPortStatus                      GET,
                          aSupportedCascadeMode            GET,
                          aSupportedPromiscMode            GET,
                          aTrainingResult                  GET,
                          aMediaType                       GET;
            ACTIONS       acPortAdministrativeControl;
        ;
    ;
    CONDITIONAL PACKAGES

        pPortBasicCtrPkg  PACKAGE
            ATTRIBUTES    aDataErrorFramesReceived         GET,
                          aHighPriorityFramesReceived      GET,
                          aLastTrainedAddress              GET,
                          aPriorityPromotions              GET,
                          aReadableFramesReceived          GET,
                          aTransitionsIntoTraining               GET;
            REGISTERED AS {iso(1) member-body(2) us(840) 802dot12(10037)
                      repeaterMgt(8) package(4) portBasicCtrPkg(6)};
            PRESENT IF    !The Basic Counter Capability is implemented.!;

        pPortExpandedPkg  PACKAGE
            ATTRIBUTES    aAllowableTrainingType           GET-REPLACE,
                          aBroadcastFramesReceived         GET,
                          aCentralMgmtDetectedDupAddr      GET-REPLACE,
                          aHighPriorityOctetsReceived      GET,
                          aIPMFramesReceived               GET,
                          aLastTrainingConfig              GET,
                          aLocalRptrDetectedDupAddr       GET,
                          aMulticastFramesReceived        GET,
                          aNormalPriorityFramesReceived   GET,
                          aNormalPriorityOctetsReceived   GET,
                          aNullAddressedFramesReceived    GET,
                          aOctetsInUnreadableFramesRcvd   GET,
                          aOversizeFramesReceived         GET,
                          aPriorityEnable                 GET-REPLACE,
                          aReadableOctetsReceived         GET,
                          aTrainedAddressChanges          GET;
            REGISTERED AS       {iso(1) member-body(2) us(840) 802dot12(10037)
                      repeaterMgt(8) package(4) portExpandedPkg(5)};
            PRESENT IF          !The Expanded Capability is implemented.!;

        pLinkMonitoringPkg      PACKAGE
            ATTRIBUTES          aLinkConfiguration              GET;
            REGISTERED AS       {iso(1) member-body(2) us(840) 802dot12(10037)
                      repeaterMgt(8) package(4) linkMonitoringPkg(7)};

            REGISTERED AS       {iso(1) member-body(2) us(840) 802dot12(10037)
                      repeaterMgt(8) managedObjectClass(3)
                  portObjectClass(3)};
            PRESENT IF          !The Link Monitoring Capability is implemented.!;

nbPortName    NAME BINDING
    SUBORDINATE OBJECT CLASS
                          oPort;
    NAMED BY SUPERIOR OBJECT CLASS
                          oGroup AND SUBCLASSES;
    WITH ATTRIBUTE              aPortID;
    REGISTERED AS               {iso(1) member-body(2) us(840) 802dot12(10037)
                          repeaterMgt(8) nameBinding(6) portName(11)};
```

C.1.4.2 Port attributes

```
aAllowableTrainingType          ATTRIBUTE
      WITH ATTRIBUTE SYNTAX     IEEE802dot12-MgmtAttributeModule.
                                AllowableTrainingValues;
      MATCHES FOR               EQUALITY;
      BEHAVIOUR                 bAllowableTrainingType;
      REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                          attribute(7) allowableTrainingType(38)};

      bAllowableTrainingType    BEHAVIOUR
      DEFINED AS              !See "aAllowableTrainingType BEHAVIOUR
                              DEFINED AS" in 13.2.4.5.1.!;

aBroadcastFramesReceived        ATTRIBUTE
      DERIVED FROM              aCounter32;
      BEHAVIOUR                 bBroadcastFramesReceived;
      REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                          attribute(7) broadcastFramesReceived(29)};
```

```
        bBroadcastFramesReceived BEHAVIOUR
        DEFINED AS              !See “aBroadcastFramesReceived BEHAVIOUR
                                        DEFINED AS” in 13.2.4.5.1.!;

aCentralMgmtDetectedDupAddr     ATTRIBUTE
        WITH ATTRIBUTE SYNTAX   IEEE802dot12-MgmtAttributeModule.Boolean;
        MATCHES FOR             EQUALITY;
        BEHAVIOUR               bCentralMgmtDetectedDupAddr;
        REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                            attribute(7) centralMgmtDetectedDupAddr(53)};

        bCentralMgmtDetectedDupAddr     BEHAVIOUR
        DEFINED AS              !See “aCentralMgmtDetectedDupAddr BEHAVIOUR
                                DEFINED AS” in 13.2.4.5.1.!;

aDataErrorFramesReceived        ATTRIBUTE
        DERIVED FROM            aCounter32;
        BEHAVIOUR               bDataErrorFramesReceived;
        REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                            attribute(7) dataErrorFramesReceived(31)};

        bDataErrorFramesReceived BEHAVIOUR
        DEFINED AS              !See “aDataErrorFramesReceived BEHAVIOUR
                                DEFINED AS” in 13.2.4.5.1.!;

aHighPriorityFramesReceived     ATTRIBUTE
        DERIVED FROM            aCounter32;
        BEHAVIOUR               bHighPriorityFramesReceived;
        REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                            attribute(7) highPriorityFramesReceived(25)};

        bHighPriorityFramesReceived BEHAVIOUR
        DEFINED AS              !See “aHighPriorityFramesReceived  BEHAVIOUR
                                DEFINED AS” in 13.2.4.5.1.!;

aHighPriorityOctetsReceived     ATTRIBUTE
        DERIVED FROM            aCounter64;
        BEHAVIOUR               bHighPriorityOctetsReceived;
        REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                            attribute(7) highPriorityOctetsReceived(26)};

        bHighPriorityOctetsReceived BEHAVIOUR
        DEFINED AS              !See “aHighPriorityOctetsReceived BEHAVIOUR DEFINED
                                AS” in 13.2.4.5.1.!;

aIPMFramesReceived              ATTRIBUTE
        DERIVED FROM            aCounter32;
        BEHAVIOUR               bIPMFramesReceived;
        REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                            attribute(7) ipmFramesReceived(32)};

        bIPMFramesReceived      BEHAVIOUR
        DEFINED AS              !See “aIPMFramesReceived BEHAVIOUR
                                        DEFINED AS” in 13.2.4.5.1.!;
```

```
aLastTrainedAddress             ATTRIBUTE
      WITH ATTRIBUTE SYNTAX     IEEE802CommonDefinitions.MACAddress;
      MATCHES FOR               EQUALITY;
      BEHAVIOUR                 bLastTrainedAddress;
      REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                          attribute(7) lastTrainedAddress(43)};

      bLastTrainedAddress       BEHAVIOUR
      DEFINED AS          !See "aLastTrainedAddress BEHAVIOUR
                          DEFINED AS" in 13.2.4.5.1.!;

aLastTrainingConfig             ATTRIBUTE
      WITH ATTRIBUTE SYNTAX     IEEE802dot12-MgmtAttributeModule.
                                TrainingConfig;
      MATCHES FOR               EQUALITY;
      BEHAVIOUR                 bLastTrainingConfig;
      REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                          attribute(7) lastTrainingConfig(39)};

      bLastTrainingConfig       BEHAVIOUR
      DEFINED AS          !See "aLastTrainingConfig BEHAVIOUR
                          DEFINED AS" in 13.2.4.5.1.!;

aLinkConfiguration              ATTRIBUTE
      WITH ATTRIBUTE SYNTAX     IEEE802dot12-MgmtAttributeModule.LinkConfiguration;
      MATCHES FOR         EQUALITY;
      BEHAVIOUR                 bLinkConfiguration;
      REGISTERED AS       {iso(1) member-body(2) us(840) 802dot12(10037)
                  repeaterMgt(8) attribute(7) linkConfiguration(58)};

      bLinkConfiguration        BEHAVIOUR
      DEFINED AS          !See "aLinkConfiguration BEHAVIOUR
                  DEFINED AS" in 13.2.4.5.1.!;

aLocalRptrDetectedDupAddr       ATTRIBUTE
      WITH ATTRIBUTE SYNTAX     IEEE802dot12-MgmtAttributeModule.Boolean;
      MATCHES FOR               EQUALITY;
      BEHAVIOUR                 bLocalRptrDetectedDupAddr;
      REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                          attribute(7) localRptrDetectedDupAddr(54)};

      bLocalRptrDetectedDupAddr       BEHAVIOUR
      DEFINED AS          !See "aLocalRptrDetectedDupAddr BEHAVIOUR
                                DEFINED AS" in 13.2.4.5.1.!;

aMediaType                      ATTRIBUTE
      WITH ATTRIBUTE SYNTAX     IEEE802dot12-MgmtAttributeModule.MediaType;
      MATCHES FOR               EQUALITY;
      BEHAVIOUR                 bMediaType;
      REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                          attribute(7) mediaType(49)};
```

```
      bMediaType                    BEHAVIOUR
      DEFINED AS              !See “aMediaType BEHAVIOUR
                              DEFINED AS” in 13.2.4.5.1.!;

aMulticastFramesReceived            ATTRIBUTE
      DERIVED FROM                  aCounter32;
      BEHAVIOUR                     bMulticastFramesReceived;
      REGISTERED AS                 {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                              attribute(7) multicastFramesReceived(30)};

      bMulticastFramesReceived      BEHAVIOUR
      DEFINED AS              !See “aMulticastFramesReceived BEHAVIOUR
                              DEFINED AS” in 13.2.4.5.1.!;

aNormalPriorityFramesReceived       ATTRIBUTE
      DERIVED FROM                  aCounter32;
      BEHAVIOUR                     bNormalPriorityFramesReceived;
      REGISTERED AS                 {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                              attribute(7) normalPriorityFramesReceived(27)};

      bNormalPriorityFramesReceived BEHAVIOUR
      DEFINED AS              !See “aNormalPriorityFramesReceived BEHAVIOUR
                              DEFINED AS” in 13.2.4.5.1.!;

aNormalPriorityOctetsReceived       ATTRIBUTE
      DERIVED FROM                  aCounter64;
      BEHAVIOUR                     bNormalPriorityOctetsReceived;
      REGISTERED AS                 {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                              attribute(7) normalPriorityOctetsReceived(28)};

      bNormalPriorityOctetsReceived BEHAVIOUR
      DEFINED AS              !See “aNormalPriorityOctetsReceived BEHAVIOUR
                              DEFINED AS” in 13.2.4.5.1.!;

aNullAddressedFramesReceived        ATTRIBUTE
      DERIVED FROM                  aCounter32;
      BEHAVIOUR                     bNullAddressedFramesReceived;
      REGISTERED AS                 {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                              attribute(7) nullAddressedFramesReceived(37)};

      bNullAddressedFramesReceived  BEHAVIOUR
      DEFINED AS              !See “aNullAddressedFramesReceived BEHAVIOUR
                              DEFINED AS” in 13.2.4.5.1.!;

aOctetsInUnreadableFramesRcvd ATTRIBUTE
      DERIVED FROM                  aCounter64;
      BEHAVIOUR                     bOctetsInUnreadableFramesRcvd;
      REGISTERED AS                 {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                              attribute(7) octetsInUnreadableFramesRcvd(24)};

      bOctetsInUnreadableFramesRcvd BEHAVIOUR
      DEFINED AS              !See “aOctetsInUnreadableFramesRcvd  BEHAVIOUR
                                    DEFINED AS” in 13.2.4.5.1.!;
```

```
aOversizeFramesReceived         ATTRIBUTE
        DERIVED FROM            aCounter32;
        BEHAVIOUR               bOversizeFramesReceived;
        REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                            attribute(7) oversizeFramesReceived(33)};

        bOversizeFramesReceived BEHAVIOUR
        DEFINED AS          !See "aOversizeFramesReceived BEHAVIOUR
                            DEFINED AS" in 13.2.4.5.1.!;

aPortAdministrativeState        ATTRIBUTE
        WITH ATTRIBUTE SYNTAX   IEEE802dot12-MgmtAttributeModule.
                                PortAdministrativeState;
        MATCHES FOR             EQUALITY;
        BEHAVIOUR               bPortAdministrativeState;
        REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                            attribute(7) portAdministrativeState(19)};

        bPortAdministrativeState    BEHAVIOUR
        DEFINED AS          !See "aPortAdministrativeState BEHAVIOUR DEFINED AS" in
                            13.2.4.5.1.!;

aPortID                         ATTRIBUTE
        WITH ATTRIBUTE SYNTAX   IEEE802dot12-MgmtAttributeModule.OneOfName;
        MATCHES FOR             EQUALITY;
        BEHAVIOUR               bPortID;
        REGISTERED AS       {iso(1) member-body(2) us(840) 802dot12(10037) repeaterMgt(8)
                            attribute(7) portID(18)};

        bPortID                 BEHAVIOUR
        DEFINED AS          !See "aPortID BEHAVIOUR DEFINED AS" in 13.2.4.5.1.!;

aPortStatus                     ATTRIBUTE
        WITH ATTRIBUTE SYNTAX   IEEE802dot12-MgmtAttributeModule.PortStatus;
        MATCHES FOR             EQUALITY;
        BEHAVIOUR               bPortStatus;
        REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                            attribute(7) portStatus(21)};

        bPortStatus             BEHAVIOUR
        DEFINED AS          !See "aPortStatus  BEHAVIOUR DEFINED AS" in
                            13.2.4.5.1.!;

aPortType                       ATTRIBUTE
        WITH ATTRIBUTE SYNTAX   IEEE802dot12-MgmtAttributeModule.PortType;
        MATCHES FOR             EQUALITY;
        BEHAVIOUR               bPortType;
        REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                            attribute(7) portType(20)};

        bPortType               BEHAVIOUR
        DEFINED AS          !See "aPortType BEHAVIOUR DEFINED AS" in
                            13.2.4.5.1.!;
```

```
aPriorityEnable              ATTRIBUTE
      WITH ATTRIBUTE SYNTAX  IEEE802dot12-MgmtAttributeModule.Boolean;
      MATCHES FOR            EQUALITY;
      BEHAVIOUR              bPriorityEnable;
      REGISTERED AS          {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                       attribute(7) priorityEnable(40)};

      bPriorityEnable        BEHAVIOUR
      DEFINED AS       !See "aPriorityEnable BEHAVIOUR
                       DEFINED AS" in 13.2.4.5.1.!;

aPriorityPromotions          ATTRIBUTE
      DERIVED FROM           aCounter32;
      BEHAVIOUR              bPriorityPromotions;
      REGISTERED AS          {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                       attribute(7) priorityPromotions(41)};

      bPriorityPromotions    BEHAVIOUR
      DEFINED AS       !See "aPriorityPromotions BEHAVIOUR
                       DEFINED AS" in 13.2.4.5.1.!;

aReadableFramesReceived      ATTRIBUTE
      DERIVED FROM           aCounter32;
      BEHAVIOUR              bReadableFramesReceived;
      REGISTERED AS          {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                       attribute(7) readableFramesReceived(22)};

      bReadableFramesReceived  BEHAVIOUR
      DEFINED AS       !See "aReadableFramesReceived BEHAVIOUR DEFINED
                       AS" in 13.2.4.5.1.!;

aReadableOctetsReceived      ATTRIBUTE
      DERIVED FROM           aCounter64;
      BEHAVIOUR              bReadableOctetsReceived;
      REGISTERED AS          {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                       attribute(7) readableOctetsReceived(23)};

      bReadableOctetsReceived  BEHAVIOUR
      DEFINED AS       !See "aReadableOctetsReceived BEHAVIOUR DEFINED AS" in
                       13.2.4.5.1.!;

aSupportedCascadeMode        ATTRIBUTE
      WITH ATTRIBUTE SYNTAX IEEE802dot12-MgmtAttributeModule.
                             SupportedCascadeValues;
      BEHAVIOUR              bSupportedCascadeMode;
      REGISTERED AS                {iso(1) member-body(2) us(840) 802dot12(10037)
                       repeaterMgt(8) attribute(7) supportedCascadeMode(55)};

      bSupportedCascadeMode  BEHAVIOUR
      DEFINED AS       !See "aSupportedCascadeMode BEHAVIOUR DEFINED AS"
                       in 13.2.4.5.1.!;
```

```
aSupportedPromiscMode          ATTRIBUTE
     WITH ATTRIBUTE SYNTAX     IEEE802dot12-MgmtAttributeModule.
                               SupportedPromiscValues;
     BEHAVIOUR                 bSupportedPromiscMode;
     REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037)
                          repeaterMgt(8) attribute(7) supportedPromiscMode(56) };

     bSupportedPromiscMode     BEHAVIOUR
     DEFINED AS           !See "aSupportedPromiscMode BEHAVIOUR DEFINED AS"
                          in 13.2.4.5.1.!;

aTrainedAddressChanges         ATTRIBUTE
     DERIVED FROM              aCounter32;
     BEHAVIOUR                 bTrainedAddressChanges;
     REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                        attribute(7) trainedAddressChanges(42)};

     bTrainedAddressChanges    BEHAVIOUR
     DEFINED AS           !See "aTrainedAddressChanges BEHAVIOUR
                          DEFINED AS" in 13.2.4.5.1.!;

aTrainingResult                ATTRIBUTE
     WITH ATTRIBUTE SYNTAX     IEEE802dot12-MgmtAttributeModule.
                               TrainingResultString;
     MATCHES FOR               EQUALITY;
     BEHAVIOUR                 bTrainingResult;
     REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                        attribute(7) trainingResult(51)};

     bTrainingResult           BEHAVIOUR
     DEFINED AS           !See "aTrainingResult BEHAVIOUR
                          DEFINED AS" in 13.2.4.5.1.!;

aTransitionsIntoTraining       ATTRIBUTE
     DERIVED FROM              aCounter32;
     BEHAVIOUR                 bTransitionsIntoTraining;
     REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                        attribute(7) transitionsIntoTraining(34)};

     bTransitionsIntoTraining  BEHAVIOUR
     DEFINED AS           !See "aTransitionsIntoTraining BEHAVIOUR
                          DEFINED AS" in 13.2.4.5.1.!;
```

C.1.4.3 Port actions

```
acPortAdministrativeControl         ACTION
     BEHAVIOUR                 bPortAdministrativeControl;
     WITH INFORMATION SYNTAX
                               IEEE802dot12-
                               MgmtAttributeModule.PortAdministrativeState;
     REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037) re-
peaterMgt(8)                        action(9) portAdministrativeControl(4)};
```

```
      bPortAdministrativeControl  BEHAVIOUR
      DEFINED AS              !See “acPortAdministrativeControl BEHAVIOUR DEFINED AS” in
                              13.2.4.5.2.!;
```

1.1.1 C.1.5 Link managed object class

1.1.1.1 C.1.5.1 Link, formal definition

```
oLink                       MANAGED OBJECT CLASS
    DERIVED FROM            “ITU-T Rec. X.721 (1992) | ISO/IEC 10165-2:1992”:top;
    CHARACTERIZED BY


    pLinkMonitoringPkg      PACKAGE
        ATTRIBUTES          aLinkConfiguration              GET,
                            aLinkAbandonments                   GET,
                            aLinkID                             GET,
                            aLinkStatus                         GET,
                            aLinkSignalsReceived                GET,
                            aLinkType                           GET,
                            aMediaType                          GET;
        ACTIONS             acLinkAdministrativeControl;
        NOTIFICATIONS       nRedundantLinkFailure;
        ;
    ;

    REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037)
                    repeaterMgt(8) managedObjectClass(3) linkObjectClass(4)};

nbLinkName    NAME BINDING
    SUBORDINATE OBJECT CLASS
                            oLink;
    NAMED BY SUPERIOR OBJECT CLASS
                            oPort AND SUBCLASSES;
    WITH ATTRIBUTE          aLinkID;
    REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037)
                    repeaterMgt(8) nameBinding(6) linkName(12)};
```

1.1.1.2 C.1.5.2 Link attributes

```
aLinkAbandonments               ATTRIBUTE
    DERIVED FROM                aCounter32;
    BEHAVIOUR                   bLinkAbandonments;
    REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037)
                    repeaterMgt(8) attribute(7) linkAbandonments (59)};

    bLinkAbandonments           BEHAVIOUR
    DEFINED AS              !See “aLinkAbandonments BEHAVIOUR
                            DEFINED AS” in 13.2.4.6.1.!;

aLinkID                         ATTRIBUTE
    WITH ATTRIBUTE SYNTAX       IEEE802dot12-MgmtAttributeModule.LinkID;
    MATCHES FOR                 EQUALITY;
    BEHAVIOUR                   bLinkID;
    REGISTERED AS            {iso(1) member-body(2) us(840) 802dot12(10037)
                     repeaterMgt(8) attribute(7) linkID(60)};

    bLinkID                       BEHAVIOUR
    DEFINED AS               !See “aLinkID BEHAVIOUR DEFINED AS” in 13.2.4.6.1.!;

aLinkSignalsReceived              ATTRIBUTE
    WITH ATTRIBUTE SYNTAX         IEEE802dot12-MgmtAttributeModule.Boolean;
    MATCHES FOR                   EQUALITY;
    BEHAVIOUR                     bLinkSignalsReceived;
    REGISTERED AS            {iso(1) member-body(2) us(840) 802dot12(10037)
                     repeaterMgt(8) attribute(7) linkSignalsReceived(61)};

    bLinkSignalsReceived          BEHAVIOUR
    DEFINED AS               !See “aLinkSignalsReceived BEHAVIOUR
                             DEFINED AS” in 13.2.4.6.1.!;

aLinkStatus                       ATTRIBUTE
    WITH ATTRIBUTE SYNTAX         IEEE802dot12-MgmtAttributeModule.LinkStatus;
    MATCHES FOR                   EQUALITY;
    BEHAVIOUR                     bLinkStatus;
    REGISTERED AS            {iso(1) member-body(2) us(840) 802dot12(10037)
                     repeaterMgt(8) attribute(7) linkStatus(62)};

    bLinkStatus                   BEHAVIOUR
    DEFINED AS               !See “aLinkStatusBEHAVIOUR DEFINED AS” in 13.2.4.6.1.!;

aLinkType                         ATTRIBUTE
    WITH ATTRIBUTE SYNTAX         IEEE802dot12-MgmtAttributeModule.LinkType;
    MATCHES FOR                   EQUALITY;
    BEHAVIOUR                     bLinkType;
    REGISTERED AS            {iso(1) member-body(2) us(840) 802dot12(10037)
                     repeaterMgt(8) attribute(7) linkType(63)};

    bLinkType                            BEHAVIOUR
    DEFINED AS               !See “aLinkType BEHAVIOUR DEFINED AS” in 13.2.4.6.1.!;

aMediaType                        ATTRIBUTE
    WITH ATTRIBUTE SYNTAX         IEEE802dot12-MgmtAttributeModule.MediaType;
    MATCHES FOR                   EQUALITY;
    BEHAVIOUR                     bMediaType;
    REGISTERED AS            {iso(1) member-body(2) us(840) 802dot12(10037)
                     repeaterMgt(8) attribute(7) mediaType(64)};

    bMediaType                    BEHAVIOUR
    DEFINED AS               !See “aMediaType BEHAVIOUR DEFINED AS” in 13.2.4.6.1.!;
```

1.1.1.3 C.1.5.3 Link actions

```
acLinkAdministrativeControl       ACTION
    BEHAVIOUR                     bLinkAdministrativeControl;
    WITH INFORMATION SYNTAX
                             IEEE802dot12-MgmtAttributeModule.LinkAdministrativeControl;
    REGISTERED AS            {iso(1) member-body(2) us(840) 802dot12(10037)
                     repeaterMgt(8) action(9) linkAdministrativeControl(5)};
    bLinkAdministrativeControl    BEHAVIOUR
    DEFINED AS                   !See "acLinkAdministrativeControl BEHAVIOUR
                                 DEFINED AS" in 13.2.4.6.2.!;
```

1.1.1.4 C.1.5.4 Link notifications

```
nRedundantLinkFailure              NOTIFICATION
    BEHAVIOUR                      bRedundantLinkFailure;
    WITH INFORMATION SYNTAX IEEE802dot12-MgmtAttributeModule.LinkStatus
    AND ATTRIBUTE IDS              fnLinkStatusInfo aLinkStatus;
    REGISTERED AS              {iso(1) member-body(2) us(840) 802dot12(10037)
                        repeaterMgt(8) notification(10) redundantLinkFailure(4)};

    bRedundantLinkFailure          BEHAVIOUR
    DEFINED AS                 !See "nRedundantLinkFailure BEHAVIOUR
                        DEFINED AS" in 13.2.4.6.3.!;
```

C.2 EndNode GDMO specification

C.2.1 EndNode MAC managed object class

C.2.1.1 EndNode MAC formal definition

```
oEndNode                                MANAGED OBJECT CLASS
    DERIVED FROM               "ITU-T Rec. X.721 (1992) | ISO/IEC 10165-2:1992":top;
    CHARACTERIZED BY

        pEndNodeBasicPkg            PACKAGE
            ATTRIBUTES        aFunctionalAddresses            GET-REPLACE,
                              aLastTrainingConfig             GET,
                              aMACID                          GET,
                              aMACStatus                      GET,
                              aMACVersion                     GET,
                              aMediaType                           GET,
                              aReadMulticastList              GET;
            ACTIONS           acAddGroupAddress,
                              acClose,
                              acDeleteGroupAddress,
                              acExecuteSelftest,
                              acInitializeMAC,
                              acOpen;
        ;
    ;
    CONDITIONAL PACKAGES

        pEndNodeBasicCtrPkg         PACKAGE
           ATTRIBUTES         aDataErrorFramesReceived        GET,
                              aFramesTransmitted              GET,
                              aHighPriorityFramesReceived     GET,
                              aHighPriorityFramesTransmitted  GET,
                              aReadableFramesReceived         GET,
                              aTransitionsIntoTraining        GET;
```

```
                REGISTERED AS  {iso(1) member-body(2) us(840) 802dot12(10037)
                                        dteMgt(1) package(4) dteBasicCtrPkg(6)};
                PRESENT IF!The EndNode Basic Counter Capability is implemented.!;

        pEndNodeExpandedPkg     PACKAGE
                ATTRIBUTES      aBroadcastFramesReceived         GET,
                                aBroadcastFramesTransmitted      GET,
                                aDesiredFramingType              GET-REPLACE,
                                aDesiredPromiscuousStatus        GET-REPLACE,
                                aFramingCapability               GET,
                                aHighPriorityOctetsReceived      GET,
                                aHighPriorityOctetsTransmitted   GET,
                                aIPMFramesReceived               GET,
                                aMulticastFramesReceived         GET,
                                aMulticastFramesTransmitted      GET,
                                aMulticastReceiveStatus                  GET-REPLACE,
                                aNormalPriorityFramesReceived    GET,
                                aNormalPriorityOctetsReceived    GET,
                                aNullAddressedFramesReceived     GET,
                                aOctetsTransmitted               GET,
                                aOversizeFramesReceived          GET,
                                aReadableOctetsReceived          GET,
                                aReadWriteMACAddress             GET-REPLACE;

                REGISTERED AS   {iso(1) member-body(2) us(840) 802dot12(10037)
                                        dteMgt(1) package(4) dteExpandedPkg(5)};
                PRESENT IF      !The Expanded Capability is implemented.!;
        REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                                managedObjectClass(3) dteObjectClass(3)};

nbDteName                       NAME BINDING
        SUBORDINATE OBJECT CLASS
                                oEndNode;
        NAMED BY SUPERIOR OBJECT CLASS
                                "ISO/IEC 10165-2":system;
        WITH ATTRIBUTE          aMACID;
        REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                                nameBinding(6) dteName(11)};

nbMACMonitor                    NAME BINDING
        SUBORDINATE OBJECT CLASS
                                "IEEE802.1F":ewmaMetricMonitor;
        NAMED BY SUPERIOR OBJECT CLASS
                                "ISO/IEC 10165-2":system;
        WITH ATTRIBUTE          aScannerId;
                CREATE          WITH-AUTOMATIC-INSTANCE-NAMING;
                DELETE          ONLY-IF-NO-CONTAINED-OBJECTS;
        REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                                nameBinding(6) macMonitor(110)};
```

C.2.1.2 EndNode MAC attributes

```
aBroadcastFramesReceived        ATTRIBUTE
      DERIVED FROM              aCounter32;
      BEHAVIOUR                 bBroadcastFramesReceived;
      REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                                attribute(7) broadcastFramesReceived(11)};

      bBroadcastFramesReceived        BEHAVIOUR
      DEFINED AS          !See "aBroadcastFramesReceived BEHAVIOUR
                                DEFINED AS" in 13.2.5.2.1.!;

aBroadcastFramesTransmitted     ATTRIBUTE
      DERIVED FROM              aCounter32;
      BEHAVIOUR                 bBroadcastFramesTransmitted;
      REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                                attribute(7) broadcastFramesTransmitted(28)};

      bBroadcastFramesTransmitted     BEHAVIOUR
      DEFINED AS          !See "aBroadcastFramesTransmitted BEHAVIOUR
                          DEFINED AS" in 13.2.5.2.1.!;

aDataErrorFramesReceived        ATTRIBUTE
      DERIVED FROM              aCounter32;
      BEHAVIOUR                 bDataErrorFramesReceived;
      REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                                attribute(7) dataErrorFramesReceived(18)};

      bDataErrorFramesReceived        BEHAVIOUR
      DEFINED AS          !See "aDataErrorFramesReceived BEHAVIOUR
                          DEFINED AS" in 13.2.5.2.1.!;

aDesiredFramingType             ATTRIBUTE
      WITH ATTRIBUTE SYNTAX     IEEE802dot12-MgmtAttributeModule.
                                FramingType;
      MATCHES FOR               EQUALITY;
      BEHAVIOUR                 bDesiredFramingType;
      REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                                attribute(7) desiredFramingType(2)};

      bDesiredFramingType       BEHAVIOUR
      DEFINED AS          !See "aDesiredFramingType BEHAVIOUR DEFINED AS" in
                          13.2.5.2.1.!;

aDesiredPromiscuousStatus             ATTRIBUTE
      WITH ATTRIBUTE SYNTAX     IEEE802dot12-MgmtAttributeModule.Boolean;
      MATCHES FOR               EQUALITY;
      BEHAVIOUR                 bDesiredPromiscuousStatus;
      REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                                attribute(7) desiredPromiscuousStatus(25)};

      bDesiredPromiscuousStatus       BEHAVIOUR
      DEFINED AS          !See "aDesiredPromiscuousStatus BEHAVIOUR
                          DEFINED AS" in 13.2.5.2.1.!;
```

```
aFramesTransmitted              ATTRIBUTE
    DERIVED FROM                aCounter32;
    BEHAVIOUR                   bFramesTransmitted;
    REGISTERED AS               {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                                attribute(7) framesTransmitted(21)};

    bFramesTransmitted          BEHAVIOUR
    DEFINED AS              !See "aFramesTransmitted BEHAVIOUR
                            DEFINED AS" in 13.2.5.2.1.!;

aFramingCapability              ATTRIBUTE
    WITH ATTRIBUTE SYNTAX       IEEE802dot12-MgmtAttributeModule.
                                FramingType;
    MATCHES FOR                 EQUALITY;
    BEHAVIOUR                   bFramingCapability;
    REGISTERED AS               {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                                attribute(7) framingCapability(3)};

    bFramingCapability          BEHAVIOUR
    DEFINED AS              !See "aFramingCapability BEHAVIOUR DEFINED AS" in
                            13.2.5.2.1.!;

aFunctionalAddresses            ATTRIBUTE
    WITH ATTRIBUTE SYNTAX       IEEE802dot12-MgmtAttributeModule.FunctionalAddresses;
    MATCHES FOR                 EQUALITY;
    BEHAVIOUR                   bFunctionalAddresses;
    REGISTERED AS               {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                                attribute(7) functionalAddresses(32)};

    bFunctionalAddresses        BEHAVIOUR
    DEFINED AS              !See "aFunctionalAddresses BEHAVIOUR
                            DEFINED AS" in 13.2.5.2.1.!;

aHighPriorityFramesReceived     ATTRIBUTE
    DERIVED FROM                aCounter32;
    BEHAVIOUR                   bHighPriorityFramesReceived;
    REGISTERED AS               {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                                attribute(7) highPriorityFramesReceived(7)};

    bHighPriorityFramesReceived     BEHAVIOUR
    DEFINED AS              !See "aHighPriorityFramesReceived  BEHAVIOUR
                            DEFINED AS" in 13.2.5.2.1.!;

aHighPriorityFramesTransmitted  ATTRIBUTE
    DERIVED FROM                aCounter32;
    BEHAVIOUR                   bHighPriorityFramesTransmitted;
    REGISTERED AS               {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                                attribute(7) highPriorityFramesTransmitted(23)};

    bHighPriorityFramesTransmitted  BEHAVIOUR
    DEFINED AS              !See "aHighPriorityFramesTransmitted BEHAVIOUR
                            DEFINED AS" in 13.2.5.2.1.!;
```

```
aHighPriorityOctetsReceived        ATTRIBUTE
        DERIVED FROM               aCounter64;
        BEHAVIOUR                  bHighPriorityOctetsReceived;
        REGISTERED AS              {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                                   attribute(7) highPriorityOctetsReceived(8)};

        bHighPriorityOctetsReceived        BEHAVIOUR
        DEFINED AS             !See "aHighPriorityOctetsReceived BEHAVIOUR DEFINED
                               AS" in 13.2.5.2.1.!;

aHighPriorityOctetsTransmitted     ATTRIBUTE
        DERIVED FROM               aCounter64;
        BEHAVIOUR                  bHighPriorityOctetsTransmitted;
        REGISTERED AS              {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                                   attribute(7) highPriorityOctetsTransmitted(24)};

        bHighPriorityOctetsTransmitted     BEHAVIOUR
        DEFINED AS             !See "aHighPriorityOctetsTransmitted BEHAVIOUR
                               DEFINED AS" in 13.2.5.2.1.!;

aIPMFramesReceived                 ATTRIBUTE
        DERIVED FROM               aCounter32;
        BEHAVIOUR                  bIPMFramesReceived;
        REGISTERED AS              {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                                   attribute(7) ipmFramesReceived(16)};

        bIPMFramesReceived         BEHAVIOUR
        DEFINED AS             !See "aIPMFramesReceived BEHAVIOUR
                                   DEFINED AS" in 13.2.5.2.1.!;

aLastTrainingConfig                ATTRIBUTE
        WITH ATTRIBUTE SYNTAX      IEEE802dot12-MgmtAttributeModule.
                                   TrainingConfig;
        MATCHES FOR                EQUALITY;
        BEHAVIOUR                  bLastTrainingConfig;
        REGISTERED AS              {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                                   attribute(7) lastTrainingConfig(15)};

        bLastTrainingConfig        BEHAVIOUR
        DEFINED AS             !See "aLastTrainingConfig BEHAVIOUR
                               DEFINED AS" in 13.2.5.2.1.!;

aMACID                             ATTRIBUTE
        WITH ATTRIBUTE SYNTAX      IEEE802dot12-MgmtAttributeModule.OneOfName;
        MATCHES FOR                EQUALITY;
        BEHAVIOUR                  bMACID;
        REGISTERED AS          {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                               attribute(7) macID(1)};

        bMACID                     BEHAVIOUR
        DEFINED AS             !See "aMACID BEHAVIOUR DEFINED AS" in 13.2.5.2.1.!;

aMACStatus                         ATTRIBUTE
        WITH ATTRIBUTE SYNTAX      IEEE802dot12-MgmtAttributeModule.MACStatus;
```

```
    MATCHES FOR             EQUALITY;
    BEHAVIOUR               bMACStatus;
    REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                            attribute(7) macStatus(33)};

    bMACStatus              BEHAVIOUR
    DEFINED AS          !See “aMACStatus  BEHAVIOUR DEFINED AS” in
                        13.2.5.2.1.!;

aMACVersion                 ATTRIBUTE
    WITH ATTRIBUTE SYNTAX   IEEE802dot12-MgmtAttributeModule.
                            VersionBitString;
    MATCHES FOR             EQUALITY, ORDERING;
    BEHAVIOUR               bMACVersion;
    REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                            attribute(7) macVersion(35)};

    bMACVersion             BEHAVIOUR
    DEFINED AS          !See “aMACVersion  BEHAVIOUR DEFINED AS” in
                        13.2.5.2.1.!;

aMediaType                  ATTRIBUTE
    WITH ATTRIBUTE SYNTAX   IEEE802dot12-MgmtAttributeModule.MediaType;
    MATCHES FOR             EQUALITY;
    BEHAVIOUR               bMediaType;
    REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                            attribute(7) mediaType(31)};

    bMediaType              BEHAVIOUR
    DEFINED AS          !See “aMediaType BEHAVIOUR DEFINED AS” in 13.2.5.2.1.!;

aMulticastFramesReceived    ATTRIBUTE
    DERIVED FROM            aCounter32;
    BEHAVIOUR               bMulticastFramesReceived;
    REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                            attribute(7) multicastFramesReceived(12)};

    bMulticastFramesReceived  BEHAVIOUR
    DEFINED AS          !See “aMulticastFramesReceived BEHAVIOUR
                        DEFINED AS” in 13.2.5.2.1.!;

aMulticastFramesTransmitted ATTRIBUTE
    DERIVED FROM            aCounter32;
    BEHAVIOUR               bMulticastFramesTransmitted;
    REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                            attribute(7) multicastFramesTransmitted(27)};

    bMulticastFramesTransmitted     BEHAVIOUR
    DEFINED AS          !See “aMulticastFramesTransmitted BEHAVIOUR
                        DEFINED AS” in 13.2.5.2.1.!;

aMulticastReceiveStatus     ATTRIBUTE
    WITH ATTRIBUTE SYNTAX   IEEE802dot12-MgmtAttributeModule.Boolean;
    MATCHES FOR             EQUALITY;
```

```
        BEHAVIOUR                bMulticastReceiveStatus;
        REGISTERED AS            {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                                 attribute(7) multicastReceiveStatus(29)};

        bMulticastReceiveStatus  BEHAVIOUR
        DEFINED AS             !See "aMulticastReceiveStatus BEHAVIOUR
                               DEFINED AS" in 13.2.5.2.1.!;

aNormalPriorityFramesReceived    ATTRIBUTE
        DERIVED FROM             aCounter32;
        BEHAVIOUR                bNormalPriorityFramesReceived;
        REGISTERED AS            {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                                 attribute(7) normalPriorityFramesReceived(9)};

        bNormalPriorityFramesReceived   BEHAVIOUR
        DEFINED AS             !See "aNormalPriorityFramesReceived BEHAVIOUR
                               DEFINED AS" in 13.2.5.2.1.!;

aNormalPriorityOctetsReceived    ATTRIBUTE
        DERIVED FROM             aCounter64;
        BEHAVIOUR                bNormalPriorityOctetsReceived;
        REGISTERED AS            {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                                 attribute(7) normalPriorityOctetsReceived(10)};

        bNormalPriorityOctetsReceived   BEHAVIOUR
        DEFINED AS             !See "aNormalPriorityOctetsReceived BEHAVIOUR
                               DEFINED AS" in 13.2.5.2.1.!;

aNullAddressedFramesReceived     ATTRIBUTE
        DERIVED FROM             aCounter32;
        BEHAVIOUR                bNullAddressedFramesReceived;
        REGISTERED AS            {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                                 attribute(7) nullAddressedFramesReceived(34)};

        bNullAddressedFramesReceived    BEHAVIOUR
        DEFINED AS             !See "aNullAddressedFramesReceived BEHAVIOUR
                                        DEFINED AS" in 13.2.5.2.1.!;

aOctetsTransmitted               ATTRIBUTE
        DERIVED FROM             aCounter64;
        BEHAVIOUR                bOctetsTransmitted;
        REGISTERED AS            {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                                 attribute(7) octetsTransmitted(22)};

        bOctetsTransmitted       BEHAVIOUR
        DEFINED AS             !See "aOctetsTransmitted BEHAVIOUR
                               DEFINED AS" in 13.2.5.2.1.!;

aOversizeFramesReceived          ATTRIBUTE
        DERIVED FROM             aCounter32;
        BEHAVIOUR                bOversizeFramesReceived;
        REGISTERED AS            {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                                 attribute(7) oversizeFramesReceived(17)};
```

```
      bOversizeFramesReceived   BEHAVIOUR
      DEFINED AS              !See "aOversizeFramesReceived BEHAVIOUR
                              DEFINED AS" in 13.2.5.2.1.!;

aReadableFramesReceived         ATTRIBUTE
      DERIVED FROM              aCounter32;
      BEHAVIOUR                 bReadableFramesReceived;
      REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                                attribute(7) readableFramesReceived(4)};

      bReadableFramesReceived   BEHAVIOUR
      DEFINED AS              !See "aReadableFramesReceived BEHAVIOUR DEFINED
                              AS" in 13.2.5.2.1.!;

aReadableOctetsReceived         ATTRIBUTE
      DERIVED FROM              aCounter64;
      BEHAVIOUR                 bReadableOctetsReceived;
      REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                                attribute(7) readableOctetsReceived(5)};

      bReadableOctetsReceived   BEHAVIOUR
      DEFINED AS              !See "aReadableOctetsReceived BEHAVIOUR DEFINED AS" in
                              13.2.5.2.1.!;

aReadMulticastList              ATTRIBUTE
      WITH ATTRIBUTE SYNTAX     IEEE802dot12-MgmtAttributeModule.MulticastAddressList;
      BEHAVIOUR                 bReadMulticastList;
      REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                                attribute(7) readMulticastList(26)};

      bReadMulticastList        BEHAVIOUR
      DEFINED AS              !See "aReadMulticastList BEHAVIOUR
                              DEFINED AS" in 13.2.5.2.1.!;

aReadWriteMACAddress            ATTRIBUTE
      WITH ATTRIBUTE SYNTAX     IEEE802CommonDefinitions.MACAddress;
      MATCHES FOR               EQUALITY;
      BEHAVIOUR                 bReadWriteMACAddress;
      REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                                attribute(7) readWriteMACAddress(30)};

      bReadWriteMACAddress      BEHAVIOUR
      DEFINED AS              !See "aReadWriteMACAddress BEHAVIOUR
                                DEFINED AS" in 13.2.5.2.1.!;

aTransitionsIntoTraining        ATTRIBUTE
      DERIVED FROM              aCounter32;
      BEHAVIOUR                 bTransitionsIntoTraining;
      REGISTERED AS             {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                                attribute(7) transitionsIntoTraining(19)};

      bTransitionsIntoTraining  BEHAVIOUR
      DEFINED AS              !See "aTransitionsIntoTraining BEHAVIOUR
                              DEFINED AS" in 13.2.5.2.1.!;
```

C.2.1.3 EndNode actions

```
acAddGroupAddress              ACTION
      BEHAVIOUR                bAddGroupAddress;
      MODE                     CONFIRMED;
      WITH INFORMATION SYNTAX
                               IEEE802CommonDefinitions.MACAddress;
      REGISTERED AS            {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                               action(9) addGroupAddress(2)};

      bAddGroupAddress         BEHAVIOUR
      DEFINED AS            !See "acAddGroupAddress BEHAVIOUR DEFINED AS" in
                            13.2.5.2.2.!;

acClose                        ACTION
      BEHAVIOUR                bClose;
      MODE                     CONFIRMED;
      REGISTERED AS            {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                               action(9) close(6)};

      bClose                   BEHAVIOUR
      DEFINED AS            !See "acClose BEHAVIOUR DEFINED AS" in 13.2.5.2.2.!;

acDeleteGroupAddress           ACTION
      BEHAVIOUR                bDeleteGroupAddress;
      MODE                     CONFIRMED;
      WITH INFORMATION SYNTAX
                               IEEE802CommonDefinitions.MACAddress;
      REGISTERED AS            {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                               action(9) deleteGroupAddress(3)};

      bDeleteGroupAddress      BEHAVIOUR
      DEFINED AS            !See "acDeleteGroupAddress BEHAVIOUR DEFINED AS" in
                               13.2.5.2.2.!;

acExecuteSelftest              ACTION
      BEHAVIOUR                bExecuteSelftest;
      MODE                     CONFIRMED;
      REGISTERED AS            {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                               action(9) executeSelftest(4)};

      bExecuteSelftest         BEHAVIOUR
                              DEFINED AS            !See "acExecuteSelftest BEHAVIOUR
                              DEFINED AS" in 13.2.5.2.2.!;

acInitializeMAC                ACTION
      BEHAVIOUR                bInitializeMAC;
      MODE                     CONFIRMED;
      REGISTERED AS            {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                               action(9) initializeMAC(1)};

      bInitializeMAC           BEHAVIOUR
      DEFINED AS            !See "acInitializeMAC BEHAVIOUR DEFINED AS" in 13.2.5.2.2.!;
```

```
acOpen                      ACTION
    BEHAVIOUR               bOpen;
    MODE                    CONFIRMED;
    REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                            action(9) open(5)};

    bOpen                   BEHAVIOUR
    DEFINED AS          !See "acOpen BEHAVIOUR DEFINED AS" in 13.2.5.2.2.!;
```

C.2.2 ResourceTypeID managed object class

C.2.2.1 ResourceTypeID, formal definition

Implementation of this managed object in accordance with the definition contained in IEEE Std 802.1F-1993 is a conformance requirement of this standard.

NOTE—A single instance of the Resource Type ID managed object exists within the Repeater managed object class. The managed object itself is contained in IEEE Std 802.1F-1993; therefore, only the name binding appears in this standard.

```
nbResourceTypeId            NAME BINDING
    SUBORDINATE OBJECT CLASS
                            "IEEE802.1F":oResourceTypeID;
    NAMED BY SUPERIOR OBJECT CLASS
                            oEndNode;
    WITH ATTRIBUTE          "IEEE802.1F":aResourceTypeIDName;
    REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) dteMgt(1)
                            nameBinding(6) resourceTypeID(9)};
```

C.3 Common attributes template

```
aCounter32                  ATTRIBUTE
                        DERIVED FROM                 "ISO/IEC 10165-2": gen-
                        ericWrappingCounter;
    BEHAVIOUR               bCounter32;
    REGISTERED AS           {iso(1) member-body(2) us(840) 802dot12(10037) global(0)
                            attribute(7) counter32(1)};

    bCounter32              BEHAVIOUR
    DEFINED AS          !The internal event that is counted is specified by each calling
                        attribute The maximum value is 2^32–1.  This is a wraparound
                        counter. The counter that this is derived from initializes to zero.
                        Initialization to zero is not a requirement of this standard. This
                        standard only supports a GET operation of this counter.!;
```

```
aCounter64                    ATTRIBUTE
                           DERIVED FROM              "ISO/IEC 10165-2": gen-
                           ericWrappingCounter;
    BEHAVIOUR                  bCounter64;
    REGISTERED AS              {iso(1) member-body(2) us(840) 802dot12(10037) global(0)
                               attribute(7) counter64(2)};

    bCounter64                 BEHAVIOUR
    DEFINED AS             !The internal event that is counted is specified by each calling
                           attribute.  The maximum value is 2^64–1.  This is a wraparound
                           counter. The counter that this is derived from initializes to zero.
                           Initialization to zero is not a requirement of this standard. This
                           standard only supports a GET operation of this counter.!;
```

C.4 ASN.1 module for demand priority managed objects.

This ASN.1 module defines the ASN.1 types and subtypes that are referred to immediately after the WITH ATTRIBUTE SYNTAX construct in this standard's uses of the attribute template defined in ISO/IEC 10165-4:1992.

```
IEEE802dot12-MgmtAttributeModule
                        {iso(1) member-body(2) us(840) 802dot12(10037) global(0)
                        asn1Module(2) commonDefinitions(0) version1(1)} DEFINITIONS
                        IMPLICIT TAGS::= BEGIN

EXPORTS  --everything
IMPORTS  --implicitly imports ISO 8824
    MACAddress          FROM IEEE802CommonDefinitions
                        {iso(1) member-body(2) us(840) ieee802dot1partF(10011)
                         asn1Module(2) commonDefinitions(0) version1(0)};

AllowableTrainingValues ::= ENUMERATED {
    allowEndNodesOnly              (1),
    allowPromiscuousEndNodes       (2),
    allowEndNodesOrRepeaters       (3),
    allowAnything                  (4)
    }

BitString ::= BIT STRING (SIZE (1..1024))

Boolean ::= BOOLEAN

CableBundling ::= ENUMERATED {
    someCablesBundled      (1),
    noCablesBundled        (2)
    }

FramingType ::= ENUMERATED {
    frameType8023    (1),
    frameType8025    (2),
    frameTypeEither  (3)
    }

FunctionalAddresses ::= OCTET STRING(SIZE(4))
```

```
LinkAdministrativeControlType ::= ENUMERATED {
            activate              (1),
            standby               (2),
            disable               (3)
            }

LinkID ::= ENUMERATED {
            firstLink             (1),
            secondLink            (2)
            }

LinkStatus ::= ENUMERATED {
            active                (1),
            inactive              (2),
            training              (3),
            disabled              (4),
            standby               (5),
            waitAndTest           (6)
            }

LinkType ::= ENUMERATED {
            primary               (1),
            secondary             (2)
            }

MACStatus ::= ENUMERATED {
         opened              (1),
         closed              (2),
         opening             (3),
         openFailure         (4),
         linkFailure         (5)
         }
MediaType ::= ENUMERATED {
         other               (1),
         unknown             (2),
         pmdMissing          (3),
         utp4                (4),
         stp2                (5),
         fibre               (6)
         }

MulticastAddressList::= SEQUENCE OF MACAddress

OneFramingType ::= ENUMERATED {
         frameType8023    (1),
         frameType8025    (2)
         }

OneOfName ::= INTEGER (1..1024)

PortAdministrativeState::= ENUMERATED {
         disabled      (1),
         enabled       (2)
         }
```

```
PortStatus ::= ENUMERATED {
        active          (1),
        inactive        (2),
        training        (3)
        }

PortType ::= ENUMERATED {
        cascadeExternal   (1),
        cascadeInternal   (2),
        localExternal     (3),
        localInternal             (4)
        }

RepeaterHealthData ::= OCTET STRING (SIZE (0..255))

RepeaterHealthInfo::= SEQUENCE {
        repeaterHealthState     [1]    RepeaterHealthState,
        repeaterHealthText      [2]    RepeaterHealthText  OPTIONAL,
        repeaterHealthData      [3]    RepeaterHealthData  OPTIONAL
        }

RepeaterHealthState::= ENUMERATED {
        other               (1),    --undefined or unknown
        ok                  (2),    --no known failures
        repeaterFailure     (3),    --known to have a repeater-related failure
        groupFailure        (4),    --known to have a group-related failure
        portFailure         (5),    --known to have a port-related failure
        generalFailure      (6)     --has a failure condition, unspecified type
        }

RepeaterHealthText ::= PrintableString (SIZE (0..255))

SearchState ::= ENUMERATED {
        none                (1),
        single              (2),
        multiple            (3)
        }

SupportedCascadeValues ::= ENUMERATED {
        hwSupportsEndNodesOnly      (1),
        hwSupportsEndNodesOrRptrs (2),
        cascadePort                 (3)
        }

SupportedPromiscValues ::= ENUMERATED {
        hwSupportsSingleModeOnly            (1),
        hwSupportsSingleOrPromiscMode    (2),
        hwSupportsPromiscModeOnly                   (3)
        }

TrainingConfig ::= OCTET STRING (SIZE (2))

TrainingResultString ::= BIT STRING (SIZE (18))
```

```
VersionBitString ::= BIT STRING (SIZE (3))

END
```

Annex D

(normative)

Allocation of object identifier values

D.1 Introduction

This annex contains a summary of all object identifier values that have been allocated by this standard. Currently, all object identifier values are used for layer management purposes.

Each table shows allocations related to a specific category of information object. The heading of the table identifies the category of information object, and shows the invariant part of the object identifier value allocated to the entries in the table. The Arc column shows the value allocated to the arc subsequent to the invariant part, which completes the object identifier allocated. The Purpose column contains a text description of the information object, and, in the case of current allocations, a reference to the location of the definition of the information object in the standard. The Status column shows the status of the allocated values, using the following convention:

R	*Reserved*. The object identifier is reserved for future use by this standard.
C	*Current*. The object identifier value has been allocated to an information object that is defined within the current version of the standard.
D	*Deprecated*. The object identifier has been allocated to an information object that was defined in a previous version of the standard, and whose use is now deprecated.

D.2 ISO/IEC 8802-12 global object identifiers

By convention, the first letter of the object identifier names is lowercase. The objects in the following tables track those defined in Clause 13 and specified in Annex C.

Allocations for global ASN.1 module identifiers

Invariant part of object identifier value =

{iso(1) member-body(2) us(840) ieee802dot12(10037) global(0) asn1Module(2)}

Arc	Purpose	Status
commonDefinitions(1)	Identifier for ISO/IEC 8802-12 common ASN.1 definitions [C.4]	C

Allocations for global attribute identifiers

Invariant part of object identifier value =

{iso(1) member-body(2) us(840) ieee802dot12(10037) global(0) attribute(7)}

Arc	Purpose	Status
aCounter32(1)	General-purpose 32-bit counter attribute [C.3]	C
aCounter64(2)	General-purpose 64-bit counter attribute [C.3]	C

D.3 Repeater object identifiers

Allocations for repeater object class identifiers

Invariant part of object identifier value =

{iso(1) member-body(2) us(840) ieee802dot12(10037) repeaterMgt(8) managedObjectClass(3)}

Arc	Purpose	Status
repeaterObjectClass(1)	Repeater managed object class name [C.1.11]	C
groupObjectClass(2)	Group managed object class name [C.1.3]	C
portObjectClass(3)	Port managed object class name C.1.4]	C
linkObjectClass(4)	Link managed object class name [C.1.5]	C

Allocations for repeater package identifiers

Invariant part of object identifier value =

{iso(1) member-body(2) us(840) ieee802dot12(10037) repeaterMgt(8) package(4)}

Arc	Purpose	Status
repeaterExpandedPkg(2)	Package for repeater expanded capability [C.1.1.1]	C
portBasicCtrPkg(6)	Package for port basic counters [C.1.4.1]	C
portExpandedPkg(5)	Package for port expanded capability [C.1.4.1]	C
linkMonitoringPkg(7)	Package for link monitoring capability [C.1.5.1]	C

Allocations for repeater name binding identifiers

Invariant part of object identifier value =

{iso(1) member-body(2) us(840) ieee802dot12(10037) repeaterMgt(8) nameBinding(6)}

Arc	Purpose	Status
repeaterName(1)	Repeater name binding [C.1.1.1]	C
repeaterMonitor(2)	Name binding for IEEE Std 802.1F: oEWAMetricMonitor [C.1.1.1]	C
resourceTypeID(3)	Name binding for IEEE Std 802.1F: oResourceTypeID [C.1.2.1]	C
groupName(4)	Group name binding [C.1.3.1]	C
portName(11)	Port name binding [C.1.4.1]	C
linkName(12)	Link name binding [C.1.5.1]	C

Allocations for repeater attribute identifiers

Invariant part of object identifier value =

{iso(1) member-body(2) us(840) ieee802dot12(10037) repeaterMgt(8) attribute(7)}

Arc	Purpose	Status
currentFramingType(3)	Current framing mode attribute [C.1.1.2]	C
desiredFramingType(4)	Desired framing mode attribute [C.1.1.2]	C
framingCapability(5)	Framing capability attribute [C.1.1.2]	C
groupMap(7)	Repeater group map attribute [C.1.1.2]	C
macAddress(2)	Repeater MAC address attribute [C.1.1.2]	C
repeaterGroupCapacity(6)	Repeater group capacity attribute [C.1.1.2]	C
repeaterHealthData(10)	Repeater health data attribute [C.1.1.2]	C
repeaterHealthState(8)	Repeater health state attribute [C.1.1.2]	C
repeaterHealthText(9)	Repeater health text attribute [C.1.1.2]	C
repeaterID(1)	Repeater ID attribute [C.1.1.2]	C
repeaterSearchAddress(11)	Repeater search address attribute [C.1.1.2]	C
repeaterSearchGroup(13)	Repeater search group attribute [C.1.1.2]	C
repeaterSearchPort(14)	Repeater search port attribute [C.1.1.2]	C
repeaterSearchState(12)	Repeater search state attribute [C.1.1.2]	C
rmacVersion(57)	Repeater version attribute [C.1.1.2]	C
groupCablesBundled(52)	Cable bundling attribute [C.1.3.2]	C
groupID(15)	Group ID attribute [C.1.3.2]	C
groupPortCapacity(16)	Group port capacity attribute [C.1.3.2]	C
portMap(17)	Group port map attribute [C.1.3.2]	C
allowableTrainingType(38)	Port allowable training configuration attribute [C.1.4.2]	C
broadcastFrames Received(29)	Port broadcast frames received attribute [C.1.4.2]	C
centralMgmtDetected DupAddr(53)	Port mgmt detected duplicate address attribute [C.1.4.2]	C
dataErrorFrames Received(31)	Port data error frames received attribute [C.1.4.2]	C
highPriorityFrames Received(25)	Port high-priority frames received attribute [C.1.4.2]	C
highPriorityOctets Received(26)	Port high-priority octets received attribute [C.1.4.2]	C
ipmFramesReceived(32)	Port IPM frames received attribute [C.1.4.2]	C

Allocations for repeater attribute identifiers (continued)

Invariant part of object identifier value =

{iso(1) member-body(2) us(840) ieee802dot12(10037) repeaterMgt(8) attribute(7)}

lastTrainedAddress(43)	Port last trained address attribute [C.1.4.2]	C
lastTrainingConfig(39)	Port last training configuration attribute [C.1.4.2]	C
localRptrDetected DupAddr(54)	Port rptr detected duplicate address attribute [C.1.4.2]	C
mediaType(49)	Port media type attribute [C.1.4.2]	C
multicastFramesReceived(30)	Port multicast frames received attribute [C.1.4.2]	C
normalPriorityFrames Received(27)	Port normal-priority frames received attribute [C.1.4.2]	C
normalPriorityOctets Received(28)	Port normal-priority octets received attribute [C.1.4.2]	C
nullAddressedFrames Received(37)	Port null addressed frames received attribute [C.1.4.2]	C
octetsInUnreadableFrames Rcvd(24)	Port octets in unreadable frames received attribute [C.1.4.2]	C
oversizeFramesReceived(33)	Port oversize frames received attribute [C.1.4.2]	C
portAdministrativeState(19)	Port administrative state attribute [C.1.4.2]	C
portID(18)	Port ID attribute [C.1.4.2]	C
portStatus(21)	Port status attribute [C.1.4.2]	C
portType (20)	Port type attribute [C.1.4.2]	C
priorityEnable(40)	Port high-priority enable attribute [C.1.4.2]	C
priorityPromotions(41)	Port priority promotions attribute [C.1.4.2]	C
readableFramesReceived(22)	Port readable frames received attribute [C.1.4.2]	C
readableOctetsReceived(23)	Port readable octets received attribute [C.1.4.2]	C
supportedCascadeMode(55)	Port supported cascade capability attribute [C.1.4.2]	C
supportedPromiscMode(56)	Port supported promiscuous capability attribute [C.1.4.2]	C
trainedAddressChanges(42)	Port trained address changes attribute [C.1.4.2]	C
trainingResult(51)	Port training result attribute [C.1.4.2]	C
transitionsIntoTraining(34)	Port transitions into training attribute [C.1.4.2]	C
linkConfiguration(58)	Link configuration attribute [C1.4.2]	C
linkAbandonments(59)	Link abandonment attribute [C.1.5.2]	C
linkID(60)	Link identification attribute [C.1.5.2]	C
linkSignalsReceived(61)	Link signals attribute [C.1.5.2]	C
linkStatus(62)	Link status attribute [C.1.5.2]	C
linkType(63)	Link type attribute [C.1.5.2]	C
mediaType(64)	Link media type attribute[C.1.5.2]	C

Allocations for repeater action types

Invariant part of object identifier value =

{iso(1) member-body(2) us(840) ieee802dot12(10037) repeaterMgt(8) action(9)}

Arc	Purpose	Status
executeNonDisruptiveSelftest Action(2)	Repeater nondisruptive selftest action [C.1.1.3]	C
repeaterSearchAddress Action(3)	Repeater search address action [C.1.1.3]	C
resetRepeater(1)	Repeater reset action [C.1.1.3]	C
portAdministrativeControl(4)	Port enable/disable action [C.1.4.3]	C
linkAdministrativeControl(5)	Link enable/disable/control action [C.1.5.3]	C

Allocations for repeater notification types

Invariant part of object identifier value =

{iso(1) member-body(2) us(840) ieee802dot12(10037) repeaterMgt(8) notification(10)}

Arc	Purpose	Status
groupMapChange(3)	Repeater group map change notification [C.1.1.4]	C
repeaterHealth(1)	Repeater health notification [C.1.1.4]	C
repeaterReset(2)	Repeater reset notification [C.1.1.4]	C
portMapChange(46)	Group port map change notification [C.1.3.3]	C
redundantLinkFailure(4)	Redundant-link failure notification [C.1.5.4]	C

D.4 End node object identifiers

Allocations for end node object class identifiers

Invariant part of object identifier value =

{iso(1) member-body(2) us(840) ieee802dot12(10037) dteMgt(1) managedObjectClass(3)}

Arc	Purpose	Status
dteObjectClass(3)	End Node managed object class name [C.2.1]	C

Allocations for end node package identifiers

Invariant part of object identifier value =

{iso(1) member-body(2) us(840) ieee802dot12(10037) dteMgt(1) package(4)}

Arc	Purpose	Status
dteBasicCtrPkg(6)	Package for End Node basic counters [C.2.1.1]	C
dteExpandedPkg(5)	Package for End Node expanded capability [C.2.1.1]	C

Allocations for end node name binding identifiers

Invariant part of object identifier value =

{iso(1) member-body(2) us(840) ieee802dot12(10037) dteMgt(1) nameBinding(6)}

Arc	Purpose	Status
dteName(11)	End Node name binding [C.2.1.1]	C
macMonitor(110)	Name binding for IEEE Std 802.1F: oEWAMetricMonitor [C.2.1.1]	C
resourceTypeID(9)	Name binding for IEEE Std 802.1F: oResourceTypeID [C.2.2.1]	C

Allocations for end node attribute identifiers

Invariant part of object identifier value =

{iso(1) member-body(2) us(840) ieee802dot12(10037) dteMgt(1) attribute(7)}

Arc	Purpose	Status
broadcastFrames Received(11)	End Node broadcast frames received attribute [C.2.1.2]	C
broadcastFrames Transmitted(28)	End Node broadcast frames transmitted attribute [C.2.1.2]	C
dataErrorFrames Received(18)	End Node data error frames received attribute [C.2.1.2]	C
desiredFramingType(2)	Desired framing mode attribute [C.2.1.2]	C
desiredPromiscuousStatus(25)	Desired promiscuous mode attribute [C.2.1.2]	C
framesTransmitted(21)	Frames transmitted attribute [C.2.1.2]	C
framingCapability(3)	Framing capability attribute [C.2.1.2]	C
functionalAddresses(32)	Functional address mask attribute [C.2.1.2]	C
highPriorityFrames Received(7)	High-priority frames received attribute [C.2.1.2]	C
highPriorityFrames Transmitted(23)	High-priority frames transmitted attribute [C.2.1.2]	C
highPriorityOctets Received(8)	High-priority octets receieved attribute [C.2.1.2]	C
highPriorityOctets Transmitted(24)	High-priority octets transmitted attribute [C.2.1.2]	C
ipmFramesReceived(16)	IPM frames received attribute [C.2.1.2]	C

Allocations for end node attribute identifiers (continued)

Invariant part of object identifier value =

{iso(1) member-body(2) us(840) ieee802dot12(10037) dteMgt(1) attribute(7)}

lastTrainingConfig(15)	Last training configuration attribute [C.2.1.2]	C
macID(1)	MAC ID attribute [C.2.1.2]	C
macStatus(33)	MAC status attribute [C.2.1.2]	C
macVersion(35)	Highest MAC version attribute [C.2.1.2]	C
mediaType(31)	Media type attribute [C.2.1.2]	C
multicastFramesReceived(12)	Multicast frames received attribute [C.2.1.2]	C
multicastFrames Transmitted(27)	Multicast frames transmitted attribute [C.2.1.2]	C
multicastReceiveStatus(29)	Multicast receive enable attribute [C.2.1.2]	C
normalPriorityFrames Received(9)	Normal-priority frames received attribute [C.2.1.2]	C
normalPriorityOctets Received(10)	Normal-priority octets received attribute [C.2.1.2]	C
nullAddressedFrames Received(34)	Null addressed frames received attribute [C.2.1.2]	C
octetsTransmitted(22)	Octets transmitted attribute [C.2.1.2]	C
oversizeFramesReceived(17)	Oversize frames received attribute [C.2.1.2]	C
readableFramesReceived(4)	Readable frames received attribute [C.2.1.2]	C
readableOctetsReceived(5)	Readable octets received attribute [C.2.1.2]	C
readMulticastList(26)	Multicast address list attribute [C.2.1.2]	C
readWriteMACAddress(30)	MAC Address attribute [C.2.1.2]	C
transitionsIntoTraining(19)	Transitions into training attribute [C.2.1.2]	C

Allocations for end node action types

Invariant part of object identifier value =

{iso(1) member-body(2) us(840) ieee802dot12(10037) dteMgt(1) action(9)}

Arc	**Purpose**	**Status**
addGroupAddress(2)	Multicast address add action [C.2.1.3]	C
close(6)	MAC disable action [C.2.1.3]	C
deleteGroupAddress(3)	Multicast address remove action [C.2.1.3]	C
executeSelftest(4)	Selftest action [C.2.1.3]	C
initializeMAC(1)	Initialization action [C.2.1.3]	C
open(5)	MAC enable action [C.2.1.3]	C

Annex E

(normative)

Network topology rules

E.1 Allowed topologies

All topologies that are allowed (in terms of distances and numbers of repeaters) in Clause 13 of ISO/IEC 8802-3:1996 for collision domains of 10BASE-T and 10BASE-FL segments are allowed in demand priority access domains.

E.2 Maximum topologies

Maximum topologies are determined by delay from the root repeater, subject to length restrictions imposed by the various link media. When there is one repeater between an end node and the root repeater, the maximum distance between the end node and the root repeater is 4 km. Each additional intermediate repeater reduces the distance by 1 km. This results in a maximum topology with five levels of repeaters having a radius of 1 km as shown in Figure E.1.

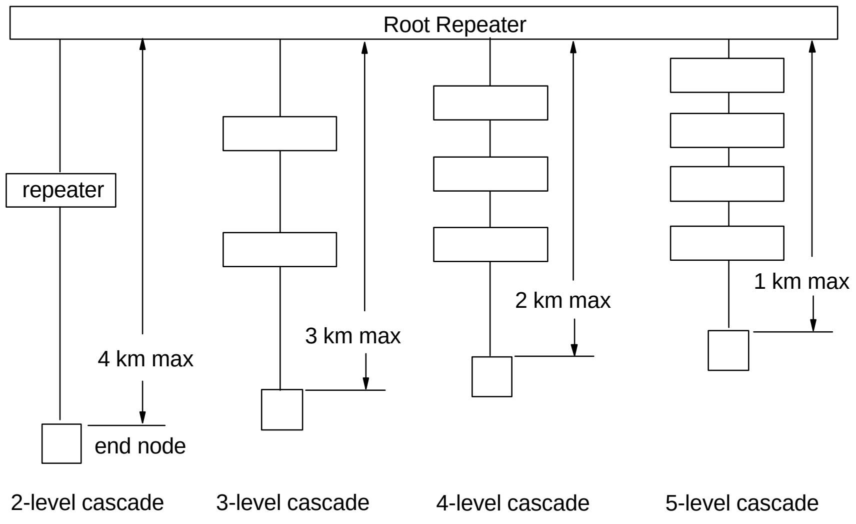


Figure E.1—Maximum root-repeater-to-last-end-node distances for alternative cascade configurations

Protocol timing limits the 4-UTP links to a maximum cable length of 200 m. Cable attenuation, noise environment, and other specifications can further limit the maximum 4-UTP link length. See 16.9 for the required characteristics.

The cable length of STP links can be up to 100 m. Cable lengths of fibre optic links can be up to 500 m with 800 nm transceivers and 2 km with 1300 nm transceivers.

E.3 4-UTP bundled cable restrictions

Bundled cable (cable containing multiple 4-pair groups) may only be used for repeater-to-end-node links where the end node is configured to receive only unicast packets specifically addressed to it (privacy mode).

a) Repeater-to-repeater or repeater-to-end-node links that are configured in promiscuous mode (to receive all traffic) shall not be implemented with bundled cable.
b) Repeater-to-end-node links from multiple repeaters shall not share the same bundle.
c) Redundant links from the same end node shall not share the same bundle.

E.4 Network topology rules and recommendations for repeaters with redundant links

The purpose of redundant links is to maintain network connectivity for all of a repeater's lower entities in the event of a failure in either of the repeater's uplinks or in a connected higher-level repeater.

All redundant links shall be connected within the same arbitration domain and shall be connected in a manner that will not result in a network loop for any combination of active links:

a) One Level-2 repeater should be designated as the alternate-root repeater.
b) The alternate-root repeater should have both uplinks connected to local ports on the root repeater.
c) All non-alternate-root, Level-2 repeaters should have one uplink (designated as the primary uplink) connected to a local port on the root repeater and the other uplink (designated as the secondary uplink) connected to a local port on the alternate-root repeater.
d) All Level-3 and lower repeaters should have their two uplinks connected to local ports on different repeaters at the next higher level.

The topology example shown in Figure E.2 maintains network connectivity in both of the following cases: a single uplink failure from any non-root repeater, or the failure of any one connected higher-level repeater.

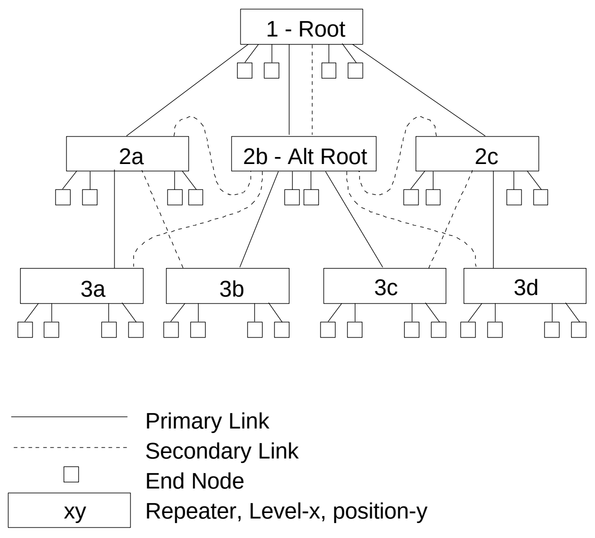


Figure E.2—An example network topology with redundant-link repeaters

It should be noted that failure of the (primary) uplink between a Level-2, non-alternate-root repeater and the root repeater can cause an extension of the root-repeater-to-lowest-end-node distance (see Figure E.3). Caution is necessary to ensure that the longest path distance between the root repeater and the lowest-level end nodes, including any added levels due a the Level-2 secondary link connection, does not exceed the maximum topology length defined in E.2.

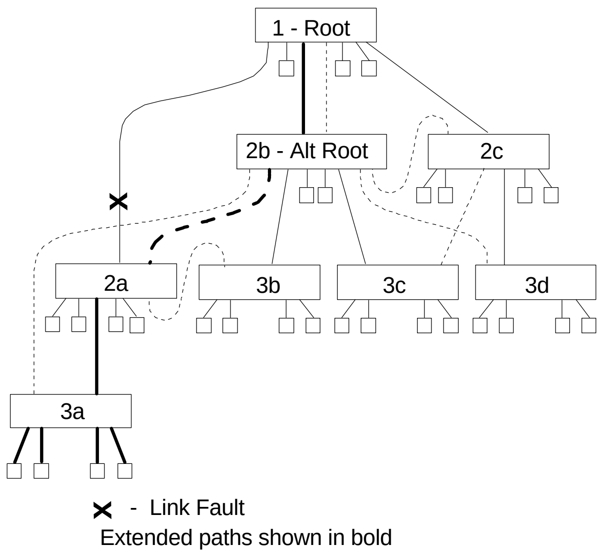


Figure E.3—Extended root-repeater-to-lowest-end-node distance due to link failure

The primary uplink from a Level-2, non-alternate-root repeater should be the active link whenever that link is operational.

E.4.1 Recommendations for interconnecting non-redundant-link repeaters with redundant-link repeaters

Non-redundant-link repeaters may be interconnected with redundant-link repeaters in the same fault-tolerant network. However, a repeater without redundant-link capability will lose network connectivity should its uplink or the connected higher-level repeater fail. To minimize connectivity loss, non-redundant-link repeaters should be:

a) Used as the root repeater. [29]
b) Connected at the lowest repeater level in a cascade.

Consider repeater 2a in Figure E.4. Failure of the root repeater or of repeater 2a's uplink will result in loss of connectivity, not only for repeater 2a's end nodes, but also for all lower repeaters and their end nodes in the cascade.

[29] Root repeaters do not require a cascade port or any uplinks.

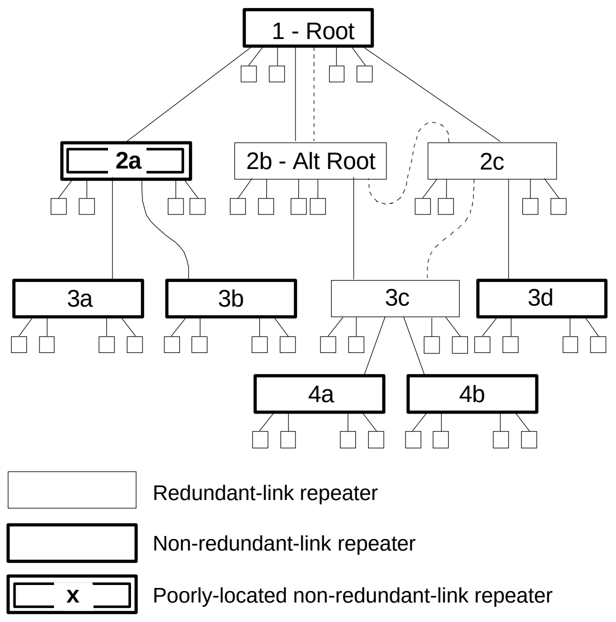

1 - Root
2a
2b - Alt Root
2c
3a
3b
3c
3d
4a
4b
Redundant-link repeater
Non-redundant-link repeater
x
Poorly-located non-redundant-link repeater

Figure E.4—Example network with both redundant-link and non-redundant-link repeaters

E.5 Network topology recommendations for end nodes with redundant links

End nodes that are equipped with multiple MACs may be connected to repeaters at any level in the network. However, to ensure maximum recoverability of network connectivity from single-link or single-repeater failures, alternate links from the same end node should be connected so that their alternate network paths do not contain any instances where both paths utilize the same link or the same repeater. Where possible, redundant end-node links should be connected to local ports on redundant-link repeaters.

The network of Figure E.5 provides examples of both properly and improperly connected redundant-link end nodes:

a) The alternate connections of end nodes x, y, and z have no common redundant path elements and will recover network connectivity in case of a single-link or single-repeater failure anywhere in the network.
b) The alternate connections for end nodes r, s, and t each have common path elements and may not recover connectivity in case of a repeater or link failure:
    1) End node r will lose most of its network connectivity if repeater 2c fails.
    2) End node s will lose most of its network connectivity if repeater 3b's uplink fails and all of its connectivity if repeater 3b itself fails.
    3) End node t will lose most of its network connectivity if repeater 2b fails.

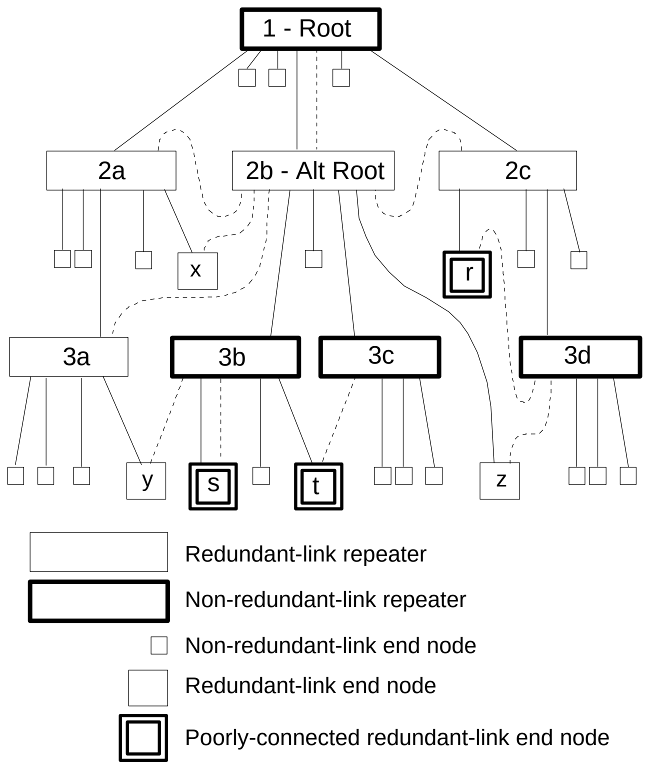

1 - Root
2a
2b - Alt Root
2c
x
r
3a
3b
3c
3d
y
s
t
z
Redundant-link repeater
Non-redundant-link repeater
Non-redundant-link end node
Redundant-link end node
Poorly-connected redundant-link end node

Figure E.5—Example connections for redundant-link end nodes

Annex F

(informative)

Use of cabling systems with a nominal characteristic impedance of 120 Ω for 4-UTP links

Clause 16 specifies the use of 100 Ω 4-UTP link segments. This annex defines the conditions for the use of cabling with a nominal characteristic impedance of 120 Ω by end nodes conforming to Clause 16.

The use of cables with a characteristic impedance outside the range specified in 16.9.1.2 will generally increase the mismatching effects in the link components, inducing additional jitter in the received signal.

F.1 Use of 120 Ω link segments

The use of a homogeneous link segment having a characteristic impedance of 120 Ω ± 15 Ω over the frequency band 1–16 MHz may add up to 0.55 ns of additional jitter to the signal at the input of the receiver (worst-case short-length link segment).

Consequently, in order to keep the overall jitter budget at the same value as for a 100 Ω link segment when using a 120 Ω link segment, the pair-to-pair NEXT loss value specified in 16.9.2 is increased by 7 dB. Equation 12 is replaced by:

$$\text{pair-to pair NEXT loss} = 37 - 15 \log_{10}\left(\frac{f}{10}\right) \text{ dB} \tag{F.1}$$

where f is the frequency in megahertz over the frequency range 1–15 MHz.

This requirement may be met with 4-pair 120 Ω cables meeting or exceeding the Category 4 specification in ISO/IEC 11801:1995.

F.2 Use of 100 Ω cables in conjunction with 120 Ω premise cabling

The use of 100 Ω cables in conjunction with 120 Ω premise cabling may add up to 0.82 ns (instead of 0.55 ns) of jitter to the signal at the input of the receivers. This can be tolerated provided that the NEXT loss values specified in 16.9.2 are increased by 13 dB (instead of 7 dB), such that:

$$\text{pair-to-pair NEXT loss} = 43 - 15 \log_{10}\left(\frac{f}{10}\right) \text{ dB} \tag{F.2}$$

where f is the frequency in megahertz over the frequency range 1–15 MHz.

This requirement may be met with 120 Ω cables meeting or exceeding the Category 5 specification in ISO/IEC 11801:1995.

NOTE—The use of 100 Ω cables at any intermediate cross-connects on 120 Ω links as well as the use of cables with a characteristic impedance of 120 Ω ± 15 Ω in conjunction with 100 Ω ± 15 Ω premise cabling is not allowed since it would result in worst-case jitter greater than that allowed in this standard.

Annex G

(informative)

Bibliography

[B1] ANSI X3T9.5 LCF-PMD Rev 1.3 (1 September 1992) FDDI Low Cost Fibre Physical Layer Medium Dependent (LCF-PMD).

[B2] EIA/TIA Bulletin TSB-36 Technical Systems Bulletin—Additional Cable Specifications for Unshielded Twisted Pair Cables.

[B3] EIA/TIA-568 Commercial Building Telecommunications Wiring Standard.

[B4] FCC Docket 20780-1980 (part 15) Technical Standards for computing Equipment. Amendment of Part 15 to redefine and clarify the rules governing restricted radiation devices and low-power communications devices.[29]

[B5] Nussbaumer, Henri. “Computer Communication Systems, vol. 1: Data Circuits, Error Detection, Data Links,” John Wiley and Sons, 1990.

[29]FCC publications are available from the Federal Communications Commission, Washington, DC, 20402, USA.