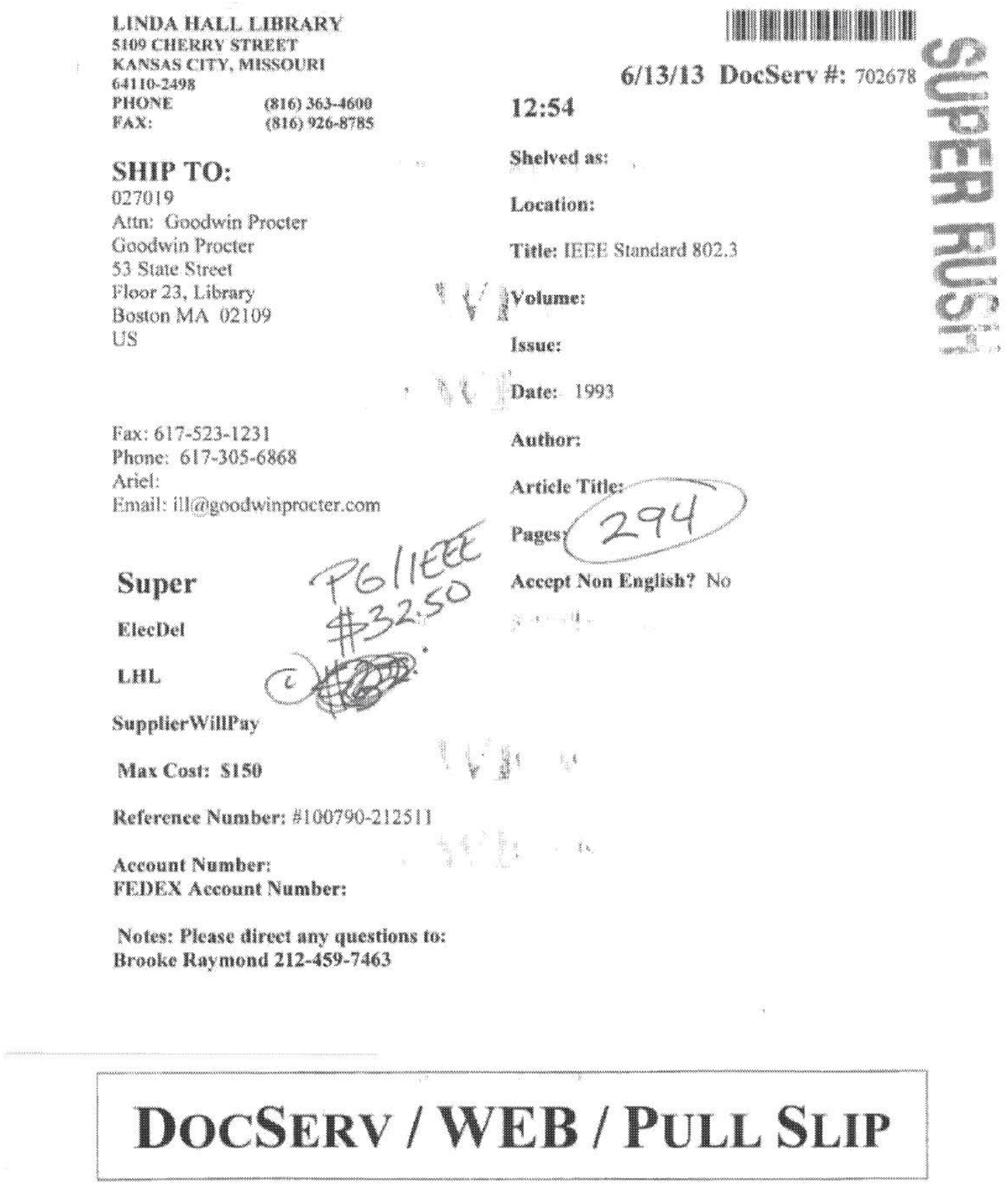


LINDA HALL LIBRARY
5109 CHERRY STREET
KANSAS CITY, MISSOURI
64110-2498
PHONE (816) 363-4600
FAX: (816) 926-8785

6/13/13 DocServ #: 702678

12:54

SHIP TO:
027019
Attn: Goodwin Procter
Goodwin Procter
53 State Street
Floor 23, Library
Boston MA 02109
US

Fax: 617-523-1231
Phone: 617-305-6868
Ariel:
Email: ill@goodwinprocter.com

Shelved as:

Location:

Title: IEEE Standard 802.3

Volume:

Issue:

Date: 1993

Author:

Article Title:

Pages: 294

Accept Non English? No

Super

ElecDel

LHL

SupplierWillPay

PG IEEE
$32.50

Max Cost: $150

Reference Number: #100790-212511

Account Number:
FEDEX Account Number:

Notes: Please direct any questions to:
Brooke Raymond 212-459-7463

DOCSERV / WEB / PULL SLIP

4021

INTERNATIONAL
STANDARD

IEEE

~~ISO/IEC~~
~~8802-3~~

ANSI/IEEE
Std 802.3

Fourth edition
1993-07-08

For supplements
See:
IEEE 802.3u
1995
IEEE 802.3j-
1993
IEEE 802.3p 1993

Information technology — Local and metropolitan area networks —

Part 3:
Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications

Technologie de l'information — Réseaux locaux et métropolitains —

Partie 3: Accès multiple par surveillance du signal et détection de collision et spécifications pour la couche physique

Reference number
ISO/IEC 8802-3:1993 (E)
ANSI/IEEE
Std 802.3, 1993 Edition

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

ISBN 1-55937-324-5

International Standard ISO/IEC 8802-3 : 1993
ANSI/IEEE Std 802.3, 1993 Edition

(This edition contains ANSI/IEEE Std 802.3-1985,
ANSI/IEEE Std 802.3c-1985, ANSI/IEEE Std 802.3d-1987,
ANSI/IEEE Std 802.3b-1985, ANSI/IEEE Std 802.3e-1987,
ANSI/IEEE Std 802.3h-1990, ANSI/IEEE Std 802.3i-1990, and
corrections resulting from Maintenance Ballot #1)

Information technology— Local and metropolitan area networks—

Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications

Sponsor

Technical Committee on Computer Communications
of the
IEEE Computer Society

Abstract: This Local and Metropolitan Area Network standard, ISO/IEC 8802-3 : 1993 [ANSI/IEEE Std 802.3, 1993 Edition], specifies the media access control characteristics for the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) access method. It also specifies the media, Medium Attachment Unit (MAU) and physical layer repeater unit for 10 Mb/s baseband and broadband systems, and it provides a 1 Mb/s baseband implementation. Specifications for MAU types 10BASE5, 10BASE2, FOIRL (fiber optic inter-repeater link), 10BROAD36, 1BASE5, and 10BASE-T are included. System considerations for multisegment 10 Mb/s baseband networks are provided. Layer and sublayer interface specifications are aligned to the ISO Open Systems Interconnection Basic Reference Model and 8802 models. The 8802-3 internal model is defined and used.

Keywords: data processing, information interchange, local area networks, mode of data transmission, network interconnection, models

Adopted as an International Standard by the
International Organization for Standardization
and by the
International Electrotechnical Commission

Published by
The Institute of Electrical and Electronics Engineers, Inc.

American National Standard

International Standard ISO/IEC 8802-3 : 1993

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and nongovernmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75% of the national bodies casting a vote.

In 1985, IEEE Standard 802.3-1985 was adopted by ISO Technical Committee 97, *Information processing systems*, as draft International Standard ISO/DIS 8802-3. Following the procedures described above, the Standard was subsequently approved by ISO and published as ISO 8802-3 : 1989, incorporating ISO 8802-3/DAD 1 which had resulted from the adoption by ISO in 1987 of ANSI/IEEE Std 802.3a.

A further revision was subsequently approved by ISO/IEC JTC 1 in 1990, incorporating ISO/IEC 8802-3/Amendments 2 and 5.

A third edition, published in 1992, incorporated ISO/IEC 8802-3/Amendments 3 and 4.

This fourth edition cancels and replaces ISO/IEC 8802-3 : 1992 and incorporates ISO/IEC 8802-3/Amendment 6, *Maintenance Ballot*; Amendment 7, *Layer management*; and Amendment 9, *System considerations for multisegment 10 Mb/s baseband networks and Twisted-pair medium attachment unit (MAU) and baseband medium, type 10BASE-T.* These amendments were approved in 1992.

For the purpose of assigning organizationally unique identifiers, the Institute of Electrical and Electronics Engineers, Inc., USA, has been designated by the ISO Council as the Registration Authority. Communications on this subject should be addressed to

Registration Authority for ISO/IEC 8802-3
c/o The Institute of Electrical and Electronics Engineers, Inc.
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
USA

During the preparation of this International Standard, information was gathered on patents upon which application of this standard might depend. Relevant patents were identified as belonging to Xerox Corporation. However, ISO and IEC cannot give authoritative or comprehensive information about evidence, validity or scope of patent and like rights. The patent-holder has stated that licenses will be granted under reasonable terms and conditions and communications on this subject should be addressed to

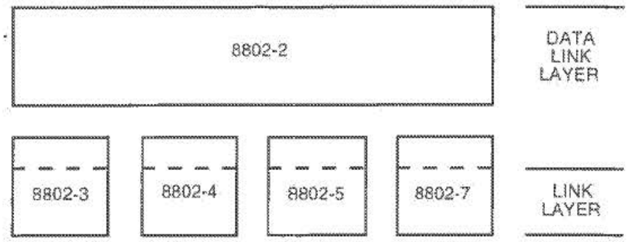
Xerox Corporation
P.O. Box 1600
Stamford, CT 06904
USA

ISO IEC

International Organization for Standardization/International Electrotechnical Commission
Case postale 56 • CH-1211 Genève 20 • Switzerland

Foreword to International Standard ISO/IEC 8802-3 : 1993

This standard is part of a family of standards for Local and Metropolitan Area Networks. The relationship between this standard and the other members of the family is shown below. (The numbers in the figure refer to ISO standard numbers.)



This family of standards deals with the Physical and Data Link layers as defined by the ISO Open Systems Interconnection Basic Reference Model (ISO 7498 : 1984). The access standards define four 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 these technologies are as follows:

(1) ISO/IEC 8802-3 [ANSI/IEEE Std 802.3, 1993 Edition], a bus utilizing CSMA/CD as the access method,
(2) ISO/IEC 8802-4 [ANSI/IEEE Std 802.4-1990], a bus utilizing token passing as the access method,
(3) ISO/IEC 8802-5 [ANSI/IEEE Std 802.5-1992], a ring utilizing token passing as the access method,
(4) ISO 8802-7, a ring utilizing slotted ring as the access method.

ISO 8802-2 [ANSI/IEEE Std 802.2-1989], *Logical Link Control protocol*, is used in conjunction with the medium access standards.

ISO/IEC 10038 [ANSI/IEEE Std 802.1D, 1993 Edition], *Media access control (MAC) bridges*, specifies an architecture and protocol for the interconnection of IEEE 802 LANs below the MAC service boundary.

The reader of this document is urged to become familiar with the complete family of standards.

The main body of this standard serves for both the ISO/IEC 8802-3 and ANSI/IEEE Std 802.3 standards. ISO/IEC and IEEE each have unique foreword sections. The Annex applies to the IEEE standard only. The Appendixes serve as useful reference material to both standards.

IEEE Standards documents are developed within the Technical Committees of the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE which have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least once every five years for revision or reaffirmation. When a document is more than five years old, and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason IEEE and the members of its technical committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

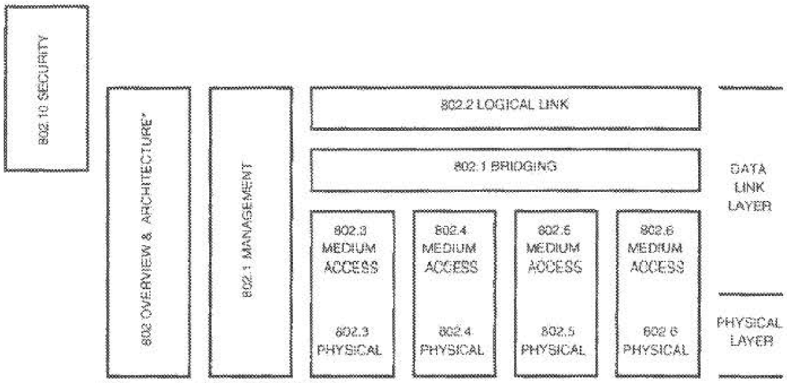
Secretary, IEEE Standards Board
345 East 47th Street
New York, NY 10017
USA

IEEE Standards documents are adopted by the Institute of Electrical and Electronics Engineers without regard to whether their adoption may involve patents on articles, materials, or processes. Such adoptions does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the standards documents.

Foreword to ANSI/IEEE Std 802.3, 1993 Edition

(This Foreword is not a part of this International Standard or of ANSI/IEEE 802.3, 1993 Edition.)

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 Basic Reference Model (ISO 7498 : 1984). The access standards define several 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 these technologies are as follows:

- IEEE Std 802[†]: Overview and Architecture. This standard provides an overview to the family of IEEE 802 standards. This document forms part of the 802.1 scope of work.
- IEEE Std 802.1B: LAN/MAN Management. Defines an Open System Interconnection (OSI) management-compatible architecture, and services and protocol elements for use in a LAN/MAN environment for performing remote management.
- ISO/IEC 10038 : 1993 [ANSI/IEEE Std 802.1D] MAC Bridging. Specifies an architecture and protocol for the interconnection of IEEE 802 LANs below the MAC service boundary.
- IEEE Std 802.1E: 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.
- ISO 8802-2 [ANSI/IEEE Std 802.2]: Logical Link Control
- ISO/IEC 8802-3 [ANSI/IEEE Std 802.3]: CSMA/CD Access Method and Physical Layer Specifications

[†] The 802 Architecture and Overview Specification, originally known as IEEE Std 802.1A, has been renumbered as IEEE Std 802. This has been done to accommodate recognition of the base standard in a family of standards. References to IEEE Std 802.1A should be considered as references to IEEE Std 802.

- ISO/IEC 8802-4 [ANSI/IEEE Std 802.4]: Token Bus Access Method and Physical Layer Specifications
- ISO/IEC 8802-5 [ANSI/IEEE Std 802.5]: Token Ring Access Method and Physical Layer Specifications
- IEEE Std 802.6: Metropolitan Area Network Access Method and Physical Layer Specifications
- IEEE Std 802.10: Interoperable Local Area Network Security, *Currently Contains* Secure Data Exchange (SDE)

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

- IEEE Std 802.7: IEEE Recommended Practice for Broadband Local Area Networks

The reader of this document is urged to become familiar with the complete family of standards.

Conformance Test Methodology

Another standards series, identified by the number 1802, has been established to identify the conformance test methodology documents for the 802 family of standards. This makes the correspondence between the various 802 standards and their applicable conformance test requirements readily apparent. Thus the conformance test documents for 802.3 are numbered 1802.3, the conformance test documents for 802.5 will be 1802.5, and so on. Similarly, ISO will use 18802 to number conformance test standards for 8802 standards.

ISO/IEC 8802-3 : 1993 (ANSI/IEEE Std 802.3, 1993 Edition)

This edition of the standard defines 10 Mb/s baseband and broadband implementations and a 1 Mb/s baseband implementation of the Physical Layer using the CSMA/CD access method. It is anticipated that future editions of the standard may provide additional implementations of the physical layer to support different needs (for example, media, and data rates).

This standard contains state-of-the-art material. The area covered by this standard is undergoing evolution. Revisions are anticipated to this standard within the next few years to clarify existing material, to correct possible errors, and to incorporate new related material.

Readers wishing to know the state of revisions should contact

Secretary
IEEE Standards Board
Institute of Electrical and Electronics Engineers, Inc
PO Box 1331, 445 Hoes Lane
Piscataway, NJ 08855-1331
USA

The IEEE 802.3 Working Group acknowledges and appreciates that many concepts embodied in this standard are based largely upon the CSMA/CD access method earlier described in *The Ethernet* specification as written jointly by individuals from Xerox Corporation, Digital Equipment Corporation, and Intel Corporation. Appreciation is also expressed to Robert M. Metcalfe and David R. Boggs for their pioneering work in establishing the original concepts.

Participants

When the IEEE 802.3 Working Group approved the original standard (ANSI/IEEE Std 802.3-1985) in 1983, it had the following membership:

Donald C. Loughry, *Chair*

Phil L. Arst
Robert F. Bridge
Charles Brill
G. J. Clancy
John Davidson
Ralph DeMent
Hank (H. N.) Dorris
Judith Estrin
Richard Fabbri
Ingrid Fromm
Milton C. Harper
Bryan Hoover
George D. Jelatis
Harold W. Katz

Donald E. Kotas
William P. Lidinsky
Laurie Lindsey
William D. Livingston
Andy Luque
Daniel Maltbie
Jerry McDowell
C. Kenneth Miller
Robert L. Morrell
Wendell Nakamine
W. P. Neblett
James Nelson
Thomas L. Phinney
David Potter

Robert S. Printis
Gary S. Robinson
Robert Rosenthal
Gary Stephens
Daniel P. Stokesberry
Ken. F. Sumner
Daniel Sze
Victor J. Tarassov
P. E. Wainwright
Lyle Weiman
Hugh E. White
Choa-Ping Wu
Nick Zades
Mo R. Zonoun

Additional individuals who contributed actively in the development of the original standard (ANSI/IEEE Std 802.3-1985) throughout its elaboration were

Juan Bulnes
Ron Crane
Dane Elliot
Alan Flatman
Maris Graube
Guy Harkins

Dean Lindsay
Then. T. Liu
Robert Moles
Tony Lauck
Joseph St. Amand
Richard Seifert
Nathan Tobol

Mark Townsend
Roger Van Brunt
Bo Vicklund
Chris Wargo
Richard Williams
Ron Yara

The ECMA TC24 Committee on Communication Protocols also provided helpful input in the development of this standard.

When the IEEE 802.3 Working Group approved ANSI/IEEE Std 802.3a-1988 (Section 10) in November 1984, it had the following membership:

Donald C. Loughry, *Chair*
Alan Flatman, *Chair, Type 10BASE2 Task Force*

Menachem Abraham
R. V. Balakrishnan
William Belknap
Charles Brill
Juan Bulnes
Stephen Cooper
Ronald Crane
John Davidson
Mark Devon
Phil Edholm
Gregory Ennis
Judy Estrin
Richard Fransen
Ingrid Fromm
Robert Galin
Rich Graham

Guy Harkins
Greg Hopkins
Joe Kennedy
Hiroshi Kobayashi
Tony Lauck
William Livingston
Hugh Logan
Leland Long
Andy Luque
Daniel Maltbie
Steven Moustakas
Wendell Nakamine
Lloyd Oliver
Aidan Paul
David Potter
Eugene Reilly

Joseph Rickert
Gary Robinson
Robert Rosenthal
Joseph St. Amand
Walter Schreuer
Stephen Soto
Gary Spencer
Robert Summers
Pat Thaler
Geoff Thompson
Wendell Turner
David White
Lawrence White
Rich Williams
Ronald Yara
Mo Zonoun

The following persons were on the balloting committee that approved ANSI/IEEE Std 802.3-1985 for submission to the IEEE Standards Board:

W. Adams
R. Appleby
G. Arnold
Y. Baeg
E. Beauregard
J. Becker
E. Bergaimini
Boorstyn
A. Carrato
G. Carson
S. Chakradarti
S. Chandra
F. Chang
C. Chao
C. Chen
P. Chen
K. Chon
R. Chow
G. Clinque
I. Cotton
D. Cox
R. DeJardins
D. Dickel
C. Eldridge
P. Enslow
J. Fendirch
M. Figuerea
D. Fisher
J. Fletcher
W. Franta
R. Gagliano
D. Gan
M. Graube
M. Greene
R. Gustin
K. Harbaugh
G. Harkins
R. Harrington
H. Heilborn
L. Heselton
D. Hislop
C. Hobbs
S. Hollander
P. Hutton
P. Induiago
T. Ishida
J. Jelemenshy
O. Kahn
S. Kak
K. Katzeff
C. Kessler
D. Kirschen
R. Kolm
T. Kuki
R. Kunkel
W. Lai
V. Lasker
N. Lau
R. Laughlin
F. Lim
T. Liu
J. Loo
K. Loughner
D. Loughry
T. Louhenkilibi
D. Manchester
M. Marco
D. Matters
D. McInode
D. Michels
L. Moraes
D. Morriss
J. Murayama
R. Nelson
D. Ofsevit
C. Ostereicher
M. Papa
S. Peter
D. Phuoc
T. Phinney
G. Power
A. Reddi
M. Repko
F. Restivo
L. Rich
D. Rine
R. Rosenthal
P. Ruosadri
S. Samoylenko
B. Sashi
A. Sauer
N. Schneidewind
O. Serlin
D. Shepard
D. Sloyer
H. Solomon
G. Stephens
C. Stillebroer
K. Sumner
E. Sykas
A. Tantawi
D. Tether
J. Tourret
K. Tu
D. Umbaugh
J. Vorhies
A. Weissberger
W. Wenker
T. Wicklund
T. Wolf
F. Wolff
R. Youg

The following persons were on the balloting committee that approved ANSI/IEEE Std 802.3a-1988 (Section 10) for submission to the IEEE Standards Board:

Marshall Abrams
John Adams
William B. Adams
S. R. Ahuja
Kit Athul
William Ayen
Yong-Myung Baeg
Wesley A. Ballenger, Jr.
Eduardo W. Bergamini
Henk F. Bokey
Betty Brannick
George S. Carson
Po Chen
L. Y. Cheung
Kilnam Chon
T. Ricky Chow
David Cohen
Alice F. Conrad
Ira W. Cotton
Robert S. Crowder
Michel Diaz
Mitchell G. Duncan
Philip H. Enslow, Jr.
Judith Estrin
John W. Fendrich
Harvey A. Freeman
Patrick Gonia
Ambuj Goyal
Michael D. Graebner
Maris Graube
Nobuhiro Hamasia
Joseph L. Hammond

Keith W. Harbaugh
S. M. Harris
J. Scott Haugdahl
Sharon Healy
C. W. Hobbs
Jim P. Hong
Paul L. Hutton
Richard Iliff
George D. Jelatis
Guy Juanole
Siegel L. Junker
Karl H. Kellermayr
Mladen Kezunovic
Samuel Kho
David Kollm
Sastri L. Kota
Hirayr M. Kudyan
Takahiko Kuki
Lee LaBarre
Wai-Sum Lai
Valerie Lasker
Lanse M. Leach
Edward Y. S. Lee
Stephen E. Levin
F. C. Lim
Don C. Loughry
Joseph F. P. Luhukay
Wo-Shun Luk
Marco Marsan
Joseph Massi
Darrell B. McIndoe
Patrick S. McIntosh

Maron Meli
David S. Millman
Aditya N. Mishra
Richard J. Moff
David E. Morgan
Mike Morganti
Kinji Mori
D. J. Morris
H. T. Mouftah
Dale A. Murray
Ruth Nelson
J. Duane Northcutt
Charles Oestereicher
David Ofsevit
Young Oh
George Parowski
Thomas L. Phinney
Nikitas Pimopoulos
David Potter
John Potvosk
Gary S. Robinson
Marya Repko
Robert Rosenthal
Gian Paolo Rossi
David J. Rypka
S. I. Samoylenko
Norman F. Schneidewind
Oscar Sepulveda
Omri Serlin
D. Sheppard
R. M. Simmons
David W. Sloyer

When the IEEE Standards Board approved ANSI/IEEE Std 802.3-1988 on June 9, 1988, and ANSI/IEEE Std 802.3a-1988 (Section 10) on October 20, 1988, it had the following membership:

Donald C. Fleckenstein, *Chair* **Marco Migliaro,** *Vice Chair*

Andrew G. Salem, *Secretary*

Arthur A. Blaisdell
Fletcher J. Buckley
James M. Daly
Stephen R. Dillon
Eugene P. Fogarty
Jay Forster*
Thomas L. Hannan
Kenneth D. Hendrix
Theodore W. Hissey, Jr.

John W. Horch
Jack M. Kinn
Frank D. Kirschner
Frank C. Kitzantides
Joseph L. Koepfinger*
Irving Kolodny
Edward Lohse
John E. May, Jr.
Lawrence V. McCall

L. Bruce McClung
Donald T. Michael*
Richard E. Mosher
L. John Rankine
Gary S. Robinson
Frank L. Rose
Helen M. Wood
Karl H. Zaininger
Donald W. Zipse

*Member emeritus

ANSI/IEEE Std 802.3-1988 and ANSI/IEEE Std 802.3a-1988 were approved by the American National Standards Institute on January 12, 1989.

When the IEEE 802.3 Working Group approved ANSI/IEEE Std 802.3c-1985 (9.1–9.8) in July 1985, it had the following membership:

Donald C. Loughry, *Chair*
Geoffrey O. Thompson, *Chair, Repeater Task Force*

Menachem Abraham
Keith Albright
R. V. Balakrishnan
William Belknap
Richard Bennett
Charles Brill
Juan Bulnes
Stephen Cooper
Paul Eastman
Phil Edholm
Gregory Ennis
Alan Flatman
Richard Fransen
Ingrid Fromm
Robert Galin
Sharad Gandhi
Rich Graham
Richard Gumpertz

Hacene Hariti
Guy Harkins
Fred Huang
Stephen Janshego
Donald Johnson
Kwi-Yung Jung
Paul Kellam
Joe Kennedy
Hiroshi Kobayashi
Lee LaBarre
Tony Lauck
John Laynor
William Livingston
Terry Lockyer
James Lucas
Andy Luque
Daniel Maltbie
Steven Moustakas
Lloyd Oliver

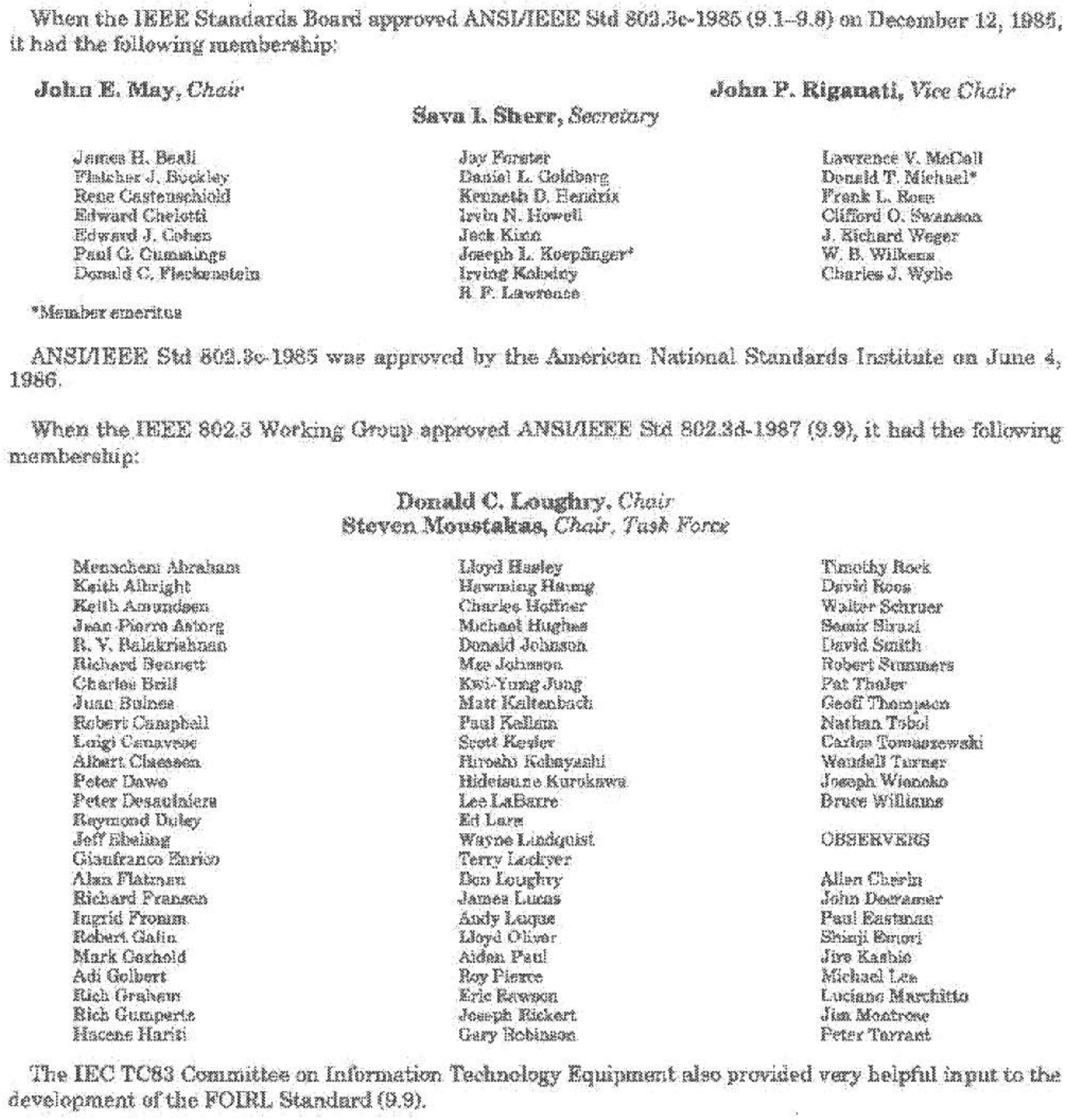
Aidan Paul
David Potter
Eric Rawson
Joseph Rickert
Gary Robinson
Timothy Rock
David Roos
Robert Rosenthal
Joseph St. Amand
Walter Schreuer
Semir Sirazi
David Smith
Stephen Soto
Robert Summers
Pat Thaler
Wendell Turner
Marc Warshaw
Ronald Yara

The following persons were on the balloting committee that approved ANSI/IEEE Std 802.3c-1985 (9.1–9.8) for submission to the IEEE Standards Board:

Marshall Abrams
John Adams
William B. Adams
S. R. Ahuja
P. D. Amer
Kit Athul
William Ayen
Yong-Myung Baeg
Wesley A. Ballenger, Jr.
Edwardo W. Bergamini
H. F. Boley
Paul W. Campbell, Jr.
George S. Carson
Po Chen
L. Y. Cheung
Kilnam Chon
T. Ricky Chow
W. F. Chow
David Cohen
Allen F. Conrad
Robert S. Crowder
Michel Diaz
Philip H. Enslow, Jr.
Judith Estrin
John W. Fendrich
Harvey A. Freeman
R. J. Gagliano
Patrick Gonia
Ambuj Goyal
Michael D. Graebner
Maris Graube
Nobushiro Hamada
Joseph L. Hammond
S. M. Harris
J. Scott Haugdahl
C. W. Hobbs
Jim P. Hong
Paul L. Hutton

Richard Iliff
George D. Jelatis
E.D. Jensen
Guy Juanole
Karl H. Kellermayr
Mladen Kozunovic
Samuel Kho
David Kollm
Sastri L. Kota
Hirayr M. Kudyan
Takahiko Kuki
Lee LaBarre
Wai-Sum Lai
Lanse M. Leach
Stephen E. Levin
F. C. Lim
William Livingston
Don C. Loughry
Joseph F. P. Luhukay
Meli Marco
Marco Marsan
Joseph Massi
Darrell B. McIndoe
Patrick S. McIntosh
David S. Millman
Aditya N. Mishra
David E. Morgan
Mike Morganti
Kinji Mori
D. J. Morris
H. T. Mouftah
Dale A. Murray
Ruth Nelson
J. Duane Northcutt
Charles Oestereicher
Young Oh
George Parowski
Thomas L. Phinney
David Potter

John Potvcek
Gary S. Robinson
Marya Repko
Robert Rosenthal
Gian Paolo Rossi
David J. Rypka
S. I. Samoylenko
Norman F. Schneidewind
Oscar Sepulveda
Omri Serlin
D. Sheppard
R. M. Simmons
L. Sintonen
David W. Sloyer
Stephen Soto
Fred Strauss
Bart W. Stuck
Tatsuya Suda
Efstathios D. Sykas
Daniel T. W. Sze
Ahmed N. Tantaui
Mario Tokoro
H. C. Torng
Donald F. Towsley
Wei-Tek Tsai
M. Tsuchiya
Richard Tung
Stanko Turk
L. David Umbaugh
James Vorhies
Pearl S. C. Wang
Don Weir
Alan J. Weissberger
William J. Wenker
Earl J. Whitaker
Michael Willett
Tsong-Ho Wu
Oren Yuen



When the IEEE Standards Board approved ANSI/IEEE Std 802.3c-1985 (9.1–9.8) on December 12, 1985, it had the following membership:

John E. May, *Chair* **John P. Riganati**, *Vice Chair*

Sava I. Sherr, *Secretary*

James H. Beall
Fletcher J. Buckley
Rene Castenschiold
Edward Chelotti
Edward J. Cohen
Paul G. Cummings
Donald C. Fleckenstein
Jay Forster
Daniel L. Goldberg
Kenneth D. Hendrix
Irvin N. Howell
Jack Kinn
Joseph L. Koepfinger*
Irving Kolodny
R. F. Lawrence
Lawrence V. McCall
Donald T. Michael*
Frank L. Rose
Clifford O. Swanson
J. Richard Weger
W. B. Wilkens
Charles J. Wylie

*Member emeritus

ANSI/IEEE Std 802.3c-1985 was approved by the American National Standards Institute on June 4, 1986.

When the IEEE 802.3 Working Group approved ANSI/IEEE Std 802.3d-1987 (9.9), it had the following membership:

Donald C. Loughry, *Chair*
Steven Moustakas, *Chair, Task Force*

Menachem Abraham
Keith Albright
Keith Amundsen
Jean-Pierre Astorg
R. V. Balakrishnan
Richard Bennett
Charles Brill
Juan Bulnes
Robert Campbell
Luigi Canavese
Albert Claessen
Peter Dawe
Peter Desaulniers
Raymond Duley
Jeff Ebeling
Gianfranco Enrico
Alan Flatman
Richard Fransen
Ingrid Fromm
Robert Galin
Mark Gerhold
Adi Golbert
Rich Graham
Rich Gumpertz
Hacene Hariti
Lloyd Hasley
Hawming Haung
Charles Hoffner
Michael Hughes
Donald Johnson
Mac Johnson
Kwi-Yung Jung
Matt Kaltenbach
Paul Kellam
Scott Kesler
Hiroshi Kobayashi
Hidetsune Kurokawa
Lee LaBarre
Ed Lare
Wayne Lindquist
Terry Lockyer
Don Loughry
James Lucas
Andy Luque
Lloyd Oliver
Aiden Paul
Roy Pierce
Eric Rawson
Joseph Rickert
Gary Robinson
Timothy Rock
David Roos
Walter Schruer
Samir Sirazi
David Smith
Robert Summers
Pat Thaler
Geoff Thompson
Nathan Tobol
Carlos Tomaszewski
Wendell Turner
Joseph Wiencko
Bruce Williams

OBSERVERS

Allan Cherin
John Deeramer
Paul Eastman
Shinji Emori
Jiro Kashio
Michael Lee
Luciano Marchitto
Jim Montrose
Peter Tarrant

The IEC TC83 Committee on Information Technology Equipment also provided very helpful input to the development of the FOIRL Standard (9.9).

The following persons were on the balloting committee that approved ANSI/IEEE Std 802.3d-1987 (9.9) for submission to the IEEE Standards Board:

William B. Adams
S. R. Ahuja
Kit Athul
William Ayen
Eduardo W. Bergamini
Paul W. Campbell, Jr.
George S. Carson
Po Chen
L. Y. Cheung
Kilnam Chon
W. F. Chow
Michael Coden
A. F. Conrad
Robert S. Crowder
Michel Diaz
N. I. Dimopoulos
M. G. Duncan
Philip H. Enslow, Jr.
Judith Estrin
John W. Fendrich
Harvey A. Freeman
Patrick S. Gonia
R. L. Gordon
A. Goyal
M. D. Graebner
Maris Graube
Joseph L. Hammond
Stephen Harris
J. Scott Haugdahl
C. W. Hobbs
Paul Hutton
Richard Illif
E. D. Jenson
Guy Juanole
Karl H. Kellermayr
M. Kezunovic
Samuel Kno
S. E. Kille
David Kollm
Takahiko Kuki
Lee LaBarre
Wai-Sum Lai
Lanse M. Leach
Edward Y. Lee
R. C. Lightburn
F. C. Lim
William D. Livingston
Don C. Loughry
Joseph F. P. Luhukay
Wo-Shun Luk
Marco Ajmone Marsan
Joseph Massi
Marco Meli
Darrel B. McIndoe
P. S. McIntosh
David S. Millman
Aditya N. Mishra
David E. Morgan
Mike Morganti
Kanji Mori
David Morris
H. H. T. Mouftah
Dale N. Murray
R. R. Nelson
J. D. Northcut
Charles Oestereicher
Young Oh
George Parowski
Thomas L. Phinney
J. M. Potucek
Marya Repko
Gary S. Robinson
Robert Rosenthal
Gian Paolo Rossi
David J. Rypka
S. I. Samaylenko
Norman F. Schneidewind
Omri Serlin
D. Sheppard
Ron Simmons
J. B. Sinclair
L. Sintonen
Tom Stack
Carel M. Stillebroer
Fred Strauss
Tatsuya Suda
P. Sugar
Efstathios D. Sykas
Daniel T. W. Sze
Ahmed N. Tantawi
H. C. Torng
D. F. Towsley
Wei-Tek Tsai
Stanko Turk
L. David Umbaugh
J. T. Vorhies
Pearl S. C. Wang
Don Weir
Alan J. Weissburger
W. J. Wenker
Earl J. Whitaker
Bryan Whittle
Michael Willett
David C. Wood
Tsong-Hu Wu
Oren Yuen

When the IEEE Standards Board approved ANSI/IEEE Std 802.3d-1987 (9.9) on December 12, 1985, it had the following membership:

Donald C. Fleckenstein, *Chair*

Marco Migliaro, *Vice Chair*

Andrew G. Salem, *Secretary*

James H. Beall
Dennis Bodson
Marshall L. Cain
James M. Daly
Stephen R. Dillon
Eugene P. Fogarty
Jay Forster
Kenneth D. Hendrix
Irvin N. Howell
Leslie R. Kerr
Jack Kinn
Irving Kolodny
Joseph L. Koepfinger*
Edward Lohse
John May
Lawrence V. McCall
L. Bruce McClung
Donald T. Michael*
L. John Rankine
John P. Riganati
Gary S. Robinson
Frank L. Rose
Robert E. Rountree
William R. Tackaberry
William B. Wilkens
Helen M. Wood

*Member emeritus

ANSI/IEEE Std 802.3d-1987 was approved by the American National Standards Institute on February 9, 1989.

When the IEEE 802.3 Working Group approved ANSI/IEEE Std 802.3b-1985 (Section 11), it had the following membership:

Donald C. Loughry, *Chair*
Menachem Abraham, *Chair, Type 10BROAD36 Task Force*

Keith Albright
R. V. Balakrishnan
William Belknap
Richard Bennett
Charles Brill
Juan Bulnes
Stephen Cooper
Ronald Crane
John Davidson
Mark Devon
Paul Eastman
Phil Edholm
Gregory Ennis
Judy Estrin
Alan Flatman
Richard Fraquer
Ingrid Fromm
Robert Galin
Sharad Gandhi
Rich Graham
Richard Gumpertz
Hacene Harizi
Guy Harkins
Gregory Hopkins

Fred Hwang
Stephen Janshego
Donald Johnson
Kwi-Yung Jung
Paul Kellam
Joe Kennedy
Hiroshi Kobayashi
Lee LaBarre
Ed Lare
Tony Lauck
John Laynor
William Livingston
Terry Lockyer
Hugh Logan
Leland Long
James Lucas
Andy Luque
Daniel Maltbie
Joseph Mazor
Steven Moustakas
Narayan Murthy
Wendell Nakamine
Lloyd Oliver

Aidan Paul
David Potter
Eric Rawson
Eugene Reilly
Joseph Rickert
Anthony Rizzolo
Gary Robinson
Timothy Rock
David Ross
Robert Rosenthal
Joseph St. Amand
Walter Schreuer
Semir Sirazi
David Smith
Stephen Soto
Gary Spencer
Robert Summers
Pat Thaler
Geoff Thompson
Nathan Tobol
Wendell Turner
Marc Warshaw
David White
Mo Zonoun

The following persons were on the balloting committee that approved ANSI/IEEE Std 802.3b-1985 (Section 11) for submission to the IEEE Standards Board:

Marshall Abrams
John Adams
William B. Adams
S. R. Ahuja
Kit Athul
William Ayen
Yong-Myung Baeg
Wesley A. Ballenger, Jr.
Edwardo W. Bergamini
Henk F. Boley
George S. Carson
Po Chen
L. Y. Cheung
Kilnam Chon
T. Ricky Chow
David Cohen
Allen F. Conrad
Ira W. Cotton
Robert S. Crowder
Michel Diaz
Mitchell G. Duncan
Philip H. Enslow, Jr.
Judith Estrin
John W. Fendrich
Harvey A. Freeman
Patrick Gonia
Ambuj Goyal
Michael D. Grachner
Maris Graube
Nobuhiro Hamada
Joseph L. Hammond
Keith W. Harbaugh
S. M. Harris
J. Scott Haugdahl
Sharon Healy
C. W. Hobbs
Jim P. Hong
Paul L. Hutton
Richard Iliff
George D. Jelatis

E. Douglas Jensen
Guy Juanole
Siegel L. Junker
Karl H. Kellermayr
Mladen Kezunovic
Samuel Kho
David Kollm
Sastri L. Kota
Hirayr M. Kudyan
Takahiko Kuki
Lee LaBarre
Wai-Sum Lai
Valerie Lasker
Lanse M. Leach
Edward Y. S. Lee
Stephen R. Lewis
F. C. Lim
Donald C. Loughry
Joseph F. P. Luhukay
Wo-Shun Luk
Marco Marsan
Joseph Massi
Darrell B. McIndoe
Patrick S. McIntosh
Marco Meli
David S. Millman
Aditya N. Mishra
Richard J. Moff
David E. Morgan
Mike Morganti
Kinji Mori
D. J. Morris
H. T. Mouftah
Dale A. Murray
Ruth Nelson
J. Duane Northcutt
Charles Oestereicher
David Ofsevit
Young Oh

George Parowski
Thomas L. Phinney
Nikitas Pinopoulos
David Potter
John Potvock
Gary S. Robinson
Marya Repko
Robert Rosenthal
Gian Paolo Rossi
David J. Rypka
S. I. Samoylenko
Norman F. Schneidewind
Oscar Sepulveda
Omri Serlin
D. Sheppard
R. M. Simmons
David W. Sloyer
Stephen Soto
Tom Stack
Carel M. Stillebroer
Fred Strauss
Bart W. Stuck
Tatsuya Suda
Peter Sugar
Efstathios D. Sykas
Daniel T. W. Sze
Ahmed N. Tantawi
Mario Tokoro
H. C. Torng
Donald F. Townley
Wei-Tek Tsai
M. Tsuchiya
Richard Tung
Stanko Turk
L. David Umbaugh
James Vorhies
Pearl S. C. Wang
Don Weir
Alan J. Weissberger
William J. Wenker

Earl J. Whitaker
Bryan S. Whittle
Michael Willett
Donald Wittman
George B. Wood
Tsong-Ho Wu

When the IEEE Standards Board approved ANSI/IEEE Std 802.3b-1985 (Section 11) on September 19, 1985, it had the following membership:

John E. May, *Chair*
John P. Riganati, *Vice Chair*
Sava I. Sherr, *Secretary*

James H. Beall
Fletcher J. Buckley
Rene Castenschiold
Edward Chelotti
Edward J. Cohen
Paul G. Cummings
Donald C. Fleckenstein
Jay Forster
Daniel L. Goldberg
Kenneth D. Hendrix
Irvin N. Howell
Jack Kinn
Joseph L. Koepfinger*
Irving Kolodny
R. F. Lawrence
Lawrence V. McCall
Donald T. Michael*
Frank L. Rose
Clifford O. Swanson
J. Richard Weger
W. B. Wilkens
Charles J. Wylie

*Member emeritus

ANSI/IEEE Std 802.3b-1985 was approved by the American National Standards Institute on February 28, 1986.

When the IEEE 802.3 Working Group approved ANSI/IEEE Std 802.3e-1987 (Section 12) in November 1986, it had the following membership:

Donald C. Loughry, *Chair*
Robert Galin, *Chair, Type 1BASE5 Task Force*

Menachem Abraham
Keith Albright
Keith Amundsen
Jean-Pierre Astorg
R. V. Balakrishnan
Ian Barker
Charles Brill
Juan Bulnes
Robert Campbell
Luigi Canavese
Albert Claessen
Michael Coden
Bill Cronin
Peter Dawe
Peter Desaulniers
Raymond Duley
Jeff Ebeling
Gianfranco Enrico
Alan Flatman
Richard Franzen
Mark Gerhold
Adi Gelbert
Rich Graham
Richard Gumpertz
Hacene Hariti
Lloyd Hasley
Haw Ming Haung
Charles Heffner
Michael Hughes
Donald Johnson
Mise Johnson
Kwi-Yung Jung
Matt Kaltenbach
Paul Kellam
Scott Kesler
Hiroshi Kobayashi
Hidetsune Kurokawa
Michael Lee
Lee LaBarre
Terry Lockyer
James Lucas
Andy Luque
Luciano Marchitto
Steven Moustakas
Lloyd Oliver
Roy Pierce
Bill Poston
Eric Rawson
Joseph Rickert
Gary Robinson
Timothy Rock
David Ross
Ed Sakaguchi
Walter Schreuer
Samir Sirazi
David Smith
Robert Summers
Peter Tarrant
Mark Taylor
Pat Thaler
Geoff Thompson
Nathan Tobol
Carlos Tomaszewski
Jayshree Ullal
Joseph Wiencko
Bruce Williams

The following persons were on the balloting committee that approved ANSI/IEEE Std 802.3e-1987 (Section 12) for submission to the IEEE Standards Board:

Marshall D. Abrams
William B. Adams
S. R. Ahuja
P. D. Amer
Kit Athul
William Ayen
Eduardo W. Bergamini
H. F. Boley
Paul W. Campbell, Jr.
George S. Carson
Po Chen
L. Y. Cheung
Kilnam Chon
W. F. Chow
Michael Coden
A. F. Conrad
Ira Cotton
D. E. Crosby
Robert S. Crowder
Michel Diaz
N. I. Dimopoulos
M. G. Duncan
P. M. Elliot
Philip H. Enslow, Jr.
Judith Estrin
John W. Fendrich
G. A. Foggiato
Harvey A. Freeman
Robert J. Gagliano
T. F. Gannon III
Patrick S. Gonia
R. L. Gordon
A. Goyal
M. D. Graebner
Maris Graube
Joseph L. Hammond
Stephen Harris
J. Scott Haugdahl
C. W. Hobbs
Paul Hutton

Richard Iliff
E. D. Jensen
Guy Juanole
S. L. Junker
Karl H. Kellermayr
M. Kezunovic
Samuel Kho
S. E. Kille
David Kollm
Takahiko Kuki
Lee LaBarre
Wai-Sum Lai
Lanse M. Leach
Edward Y. Lee
S. E. Levin
R. C. Lightburn
F. C. Lim
William D. Livingston
Don C. Loughry
Joseph F. P. Luhukay
Wo-Shun Luk
Marco Ajmone Marsan
Joseph Massi
Marco Meli
Darrel B. McIndoe
P. S. McIntosh
David S. Millman
Aditya N. Mishra
David E. Morgan
Mike Morganti
Kanji Mori
David Morris
H. H. T. Mouftah
Dale N. Murray
R. R. Nelson
J. D. Northcutt
Charles Oestereicher
Young Oh
George Parowski
Thomas L. Phinney
David Potter

J. M. Potocek
Marya Repko
Gary S. Robinson
Robert Rosenthal
Gian Paolo Rossi
David J. Rypka
S. I. Samoylenko
Norman F. Schneidewind
Omri Serlin
D. Sheppard
Ron Simmons
J. B. Sinclair
L. Sintonen
Stephen H. Soto
Tom Stack
Carel M. Stillebroer
Fred Strauss
Bart W. Stuck
Tatsuya Suda
P. Sagar
Efstathios D. Sykas
Daniel T. W. Sze
Ahmed N. Tantawi
H. C. Torng
D. F. Towsley
Wei-Tek Tsai
Masahiro Tsuchiya
Stanko Turk
L. David Umbaugh
J. T. Vorhies
Pearl S. C. Wang
Don Weir
Alan J. Weissberger
W. J. Wenker
Earl J. Whitaker
Bryan Whittle
Michael Willett
David C. Wood
Tsong-Ho Wu
Oren Yuen

When the IEEE Standards Board approved ANSI/IEEE Std 802.3e-1987 (Section 12) on June 11, 1987, it had the following membership:

Donald C. Fleckenstein, *Chair*
Marco W. Migliaro, *Vice Chair*
Andrew G. Salem, *Secretary*

James H. Beall
Dennis Bodson
Marshall L. Cain
James M. Daly
Stephen R. Dillon
Eugene P. Fogarty
Jay Forster
Kenneth D. Hendrix
Irvin N. Howell

Leslie R. Kerr
Jack Kinn
Irving Kolodny
Joseph L. Koepfinger*
Edward Lohse
John May
Lawrence V. McCall
L. Bruce McClung
Donald T. Michael*

L. John Rankine
John P. Riganati
Gary S. Robinson
Frank L. Rose
Robert E. Rountree
Sava I. Sherr*
William R. Tackaberry
William B. Wilkens
Helen M. Wood

*Member emeritus

ANSI/IEEE Std 802.3e-1987 was approved by the American National Standards Institute on December 15, 1987.

When the IEEE 802.3 Working Group approved ANSI/IEEE Std 802.3h-1990 (Section 5), it had the following membership:

Donald C. Loughry, *Chair*
Andy J. Luque, *Chair, Layer Management Task Force*

Menachem Abraham
John R. Agee
Richard Anderson
Ekkehard Antz
Keith Amundsen
Susie Armstrong
R. V. Balakrishnan
Mark Bohrer
Richard Brand
Thomas Butler
Luca Cafiero
Robert R. Campbell
Luigi Canavese
Jacques Christ
Michael Coden
Robert Conte
Bill Cronin
Peter Cross
John DeCramer
Ian Crayford
Nabil Damouny
Sanjay Dhawan
Raymond S. Duley
Paul Eastman
Richard Ely
Gianframco Enrico
Norman Erbacher
Steve Evitts
Alan V. Flatman
Ingrid Fromm
Mel Gable
Bob Galin
Mark Gerhold
Rich Graham
Andreas Gulle
Richard Gumpertz
Clive Hallatt
Kevin Hamilton
Benny Hanigal
Lloyd Hasley
W. B. Hatfield
Stephen Haughey
Carl G. Hayssen
Ariel Hendel
Chip Hicks
William Hingston
Charles Hoffner
Ernie Jensen
Clarence Joh
Dieter W. Junkers
Donald C. Johnson
Mize Johnson
Scott Kesler
Bob Kilgore
Yongbum Kim
Bill Kind
John Kincaid
Tadayoshi Kitayama
Paul Kopera
David Kung
Michael Lee
Richard Lena
Yoseph Linde
Wayne Lindquist
T. D. Lockyer
James A. Lucas
Ian Lyon
Kenneth MacLeod
Luciano Marchitto
Charles Marsh
Bob Matthys
Steven Moustakas
Narayan Murthy
Darcy Nelson
Bob Norton
Mike O'Connor
Chris Oliver
Lloyd Oliver
Kazuyuki Ozawa
Keith Onodera
Tony Peatfield
Peter Rautenberg
Bill Reysen
Gary Robinson
Steven Robinson
Moni Samaan
Fred Sammartino
Stan Sassower
F. Sarles
Ronald Schmidt
Tom Schmitt
Frederick Scholl
Ron Shani
Semir Sirazi
Joseph Skorupa
David A. Smith
Bob Smith
Steve Smith
Robert Snyder
Graham Starkins
David E. Stein
Peter Tarrant
Mark Taylor
Patricia Thaler
Douglas Thompson
Geoffrey O. Thompson
Nathan Tobol
Carlos Tomaszewski
Herbert Uhl
Steven Ulrich
John Visser
William Wager
Joseph A. Wiencko, Jr.
Bruce Williams
Richard Williams
Roger Wilmarth
Mike Wincn
Mark Wingrove
Nobushige Yokota

The following persons were on the balloting committee that approved ANSI/IEEE Std 802.3h-1990 for submission to the IEEE Standards Board:

William Adams
Kit Athul
William E. Ayen
Ali Bahrololoomi
George S. Carson
Chih-Tsai M. Chen
Michael H. Coden
R. A. Conser
R. S. Crowder
Andrew Davidson
Luis F. M. De Moraes
N. I. Dimopoulos
Mitchell Duncan
John E. Emrich
John W. Fendrich
Harold C. Folts
Harvey Freeman
Ingrid Fromm
D. G. Gan
Patrick Gonia
Julio Gonzalez Sanz
Michael Graebner
Maris Graube
Joseph L. Hammond
Stephen Harris
J. Scott Haugdahl
C.W.L. Hobbs
Chris Hsieh
Richard J. Iliff
Raj Jain
M. Kezunovic
Samuel Kho
Tom Kurihara
Lee Labarre
Anthony B. Lake
Mike Lawler
Jaiyong Lee
F. C. Lim
Randolph S. Little
William Livingston
Joseph Leo
Donald C. Loughry
Andy J. Luque
Kelly C. McDonald
Darrell B. McIndoe
Richard H. Miller
David S. Millman
Adirya Mishra
John E. Montague
M. A. F. Morganti
Kinji Mori
D. J. Morris
M. T. Mouftah
Arne A. Nilsson
Charles Oestereicher
Young Oh
Thomas L. Phinney
Rafat Pirzada
Udo Pooch
Robert S. Printis
Marya S. Repko
John P. Riganati
Gary S. Robinson
N. F. Schneidewind
Manfred H. Seifert
D. A. Sheppard

Glen Sherwood
R. M. Simmons
Leo Sintonen
Harry P. Solomon
Robert K. Southard
John Spragins
C. M. Stillebroer

Frank J. Strauss
E. D. Sykes
A. N. Tantawi
Nathan Tobol
Wei-Tek Tsai
David L. Umbaugh
T. A. Varetoni

James Vorhies
Don Weir
A. F. Wheeler
Earl J. Whitaker
D. C. Wood
George B. Wright
Oren Yuen

When the IEEE Standards Board approved ANSI/IEEE Std 802.3h-1990 on September 28, 1990, it had the following membership:

Marco W. Migliaro, *Chairman* **James M. Daly,** *Vice Chairman*

Andrew G. Salem, *Secretary*

Dennis Bodson
Paul L. Borrill
Fletcher J. Buckley
Allen L. Clapp
Stephen R. Dillon
Donald C. Fleckenstein
Jay Forster*
Thomas L. Hannan

Kenneth D. Hendrix
John W. Horch
Joseph L. Koepfinger*
Irving Kolodny
Michael A. Lawler
Donald J. Loughry
John E. May, Jr.

Lawrence V. McCall
L. Bruce McClung
Donald T. Michael*
Stig Nilsson
Roy T. Oishi
Gary S. Robinson
Terrance R. Whittemore
Donald W. Zipse

*Member Emeritus

ANSI/IEEE Std 802.3h-1990 was approved by the American National Standards Institute on March 11, 1991.

When the IEEE 802.3 Working Group approved ANSI/IEEE Std 802.3i-1990 (Sections 13 and 14), it had the following membership:

Donald C. Loughry, *Chair*[*]

Patricia Thaler, *Chair, Type 10BASE-T Task Force*[†]

Menachem Abraham
Luc Adriaenssens
John E. Agee
Keith Amundsen
Richard Anderson
Stephen J. Anderson
Ekkehard Antz
Susie Armstrong
K. V. Balakrishnan
Roberto Bertaldi
Dave Bethune
Mark Bohrer
Richard Brand
Thomas Butler
Luca Cafiero
Robert R. Campbell
Luigi Canavese
Michael Coden
Kevin Cone
Robert Conte
Neil Coote
Ian Crayford
Bill Cronin
Peter Cram
Joe Curcio
Nabil Damouny
Mark Darby
John DeCramer
Tazio M. Denicolo
Sanjay Dawan

Paul (Skip) Ely
Richard Ely
Norman Erbacher
Steve Evitts
Eldon Feist
Alan V. Flatman
Ingrid Fromm
Mel Gable
Robert Galin
Mark Gerhold
Andreas Gulla
Richard Gumpertz
Clive Hallatt
Benny Hanigal
W. B. Hatfield
Stephen Haughey
Carl G. Hayssen
Ernie Jensen
Clarence Joh
Donald C. Johnson
Max Johnson
Imre Juhasz
Dieter W. Junkers
Joel S. Kalman
Rainer Kaps
Bob Kilgore
Yongbum Kim
John Kincaid
Bill Kind
Tadayoshi Kitayama

Steven Keller
Paul Kopera
Leonid Kozhavey
Ted Kummert
David Kung
Michael Lehar
Michael Lee
Richard Lefkowitz
Richard Lena
Yoseph Linde
T. D. Lockyer
Andy J. Luque
Kenneth MacLeod
Luciano Marchitto
Christine Marsh
Steven Moustakas
Narayan Murthy
Darcy Nelson
Bob Norton
Mike O'Connor
Chris Oliver
Lloyd Oliver
Keith Onodera
Kazuyuki Ozawa
Charles Palazzo
Tony Penfield
Peter Rautenberg
Bill Reynen
Gary Robinson
Steven Robinson

[*] **Patricia Thaler,** *Current Chair*

[†] **Richard Anderson,** *Current Chair*

Paul F. Russo
Moni Samman
P. Sarles
Stan Sassower
Ronald Schmidt
Tom Schmitt
Frederick Scholl
Ron Shani
Joseph Skorupa
David A. Smith

Bob Smith
Steve Smith
Robert Snyder
Graham Starkins
David E. Stein
Peter Tarrant
Mark Taylor
Douglas Thompson
Geoffrey Thompson
Nathan Tobol

Carlos Tomaszewski
Herbert Uhl
John Visser
William Wager
Joseph Wiencko, Jr.
Richard Williams
Roger Wilmarth
Mike Wincn
Mark Wingrove
Nobuhige Yokota

The following persons were on the balloting committee for ANSI/IEEE Std 802.3i-1990:

Bandula W. Abeysundara
William B. Adams
Don Aelmore
Hasan S. Alkhatib
Jonathan Allan
Sule Arslander
Kit Athul
Michael Atkinson
William E. Ayen
Yong Myung Baeg
Subhash Bhatia
Asa O. Bishop
Alan L. Bridges
Richard Cassi
Mehmet U. Caglayan
Anthony L. Carrato
George S. Carson
Brian J. Casey
George C. Chachis
Chih-Tsai Chen
Gerald W. Cichanowski
Michael H. Coden
Keith Collins
Rodney A. Conner
Robert Crowder
Jose A. Cueto
F. Deravi
Ashwani K. Dhawan
Siyi Terry Dong
Mitchell G. Duncan
Andrew M. Dunn
Sourav Dutta
Ted Duik
Hans Eklund
John E. Emrich
Richard G. Estock
Changxin Fan
John W. Fendrich
John N. Ferguson
Samuel Fineberg
Ernest L. Fogle
Harold C. Folts
Sandra J. Forney
Harvey A. Freeman
Ingrid Fromm
Eithan Froumine
Robert Gagliano
Isaac Ghansah
Patrick Gonia
Michael D. Graebner
Maris Graube
Abraham Grund
Crag Guarnieri
Sandor V. Halasz
Joseph L. Hammond
Clark M. Hay
Lee A. Hollaar
Marsha D. Hopwood
Anne B. Horton
Genesio L. Hubacher
Wing Huen

Bob Jacobsen
Raj Jain
Gerrit K. Janssen
Jack R. Johnson
Reijo Juvonen
Richard H. Karpinski
Julian Kateley
Gary C. Kessler
Samuel Kho
Jens Kolind
Vijaya Konangi
Peter Kornerup
Jon Kramp
Stephen B. Kruger
Thomas M. Kurihara
Anthony B. Lake
Lak Ming Lam
Glen Langdon
Mike Lawler
Lanse M. Leach
John E. Lecky
Jai-Yong Lee
Michael E. Lee
Lewis E. Leinenweber
Kin Fun Lo
F. C. Lim
Ping Liu
Randolph S. Little
William D. Livingston
Mauro Lolli
Wayne M. Loucks
Donald Loughry
Nam C. Low
Andy J. Luque
Carl R. Manson
Eduardo G. Marmol
Gerald M. Masson
Richard McBride
Kelly C. McDonald
William McDonald
Darrell B. McIndoe
Richard H. Miller
David S. Millman
C. B. M. Mishra
Wen Hsien Lim Moh
John E. Montague
Kinji Mori
Gerald Moseley
H. H. T. Mouftah
K. R. S. Murthy
Charles E. Neblock
Ruth Nelson
Arne A. Nilsson
Donal O'Mahony
Frederic Oakland
Charles Oestereicher
Attila Orgit
Richard J. Paroline
Thomas E. Phillips
Art J. Pina
Rafat Pirzada
Udo W. Pooch

Hardy J. Pottinger
Andris Putnins
Thad L. D. Regulinski
Francisco J. Retivo
John R. Rignanti
Saber Risk
Philip T. Robinson
Gary S. Robinson
Robert Rosenthal
Daniel Rosich
Floyd E. Ross
Victor Rozentouler
Chiaki Sagawa
Mark S. Sanders
Ravi Sankar
Julio Gonzalez Sanz
Ambatipudi Sastry
Vidyadhar S. Savant
Manoj Kunar Saxena
Lorne Schachter
Norman Schneidewind
Jeffrey R. Schwab
A. D. Sheppard
Glen Sherwood
William T. Smith
I. A. Soceanu
Robert K. Southard
Charles Spurgeon
Michael Stephansen
Fred J. Strauss
Efstathios D. Sykas
Roy S. Syler
Gregory M. Sylvain
Daniel Sze
Nhi P. Ta
Hassan Tabaie
Hao Tang
Ahmed N. Tantawi
Steven R. Taylor
James N. Thomas
Geoffrey O. Thompson
Nathal Tobol
Robert Tripi
L. David Umbaugh
Thomas A. Varetoni
James T. Vorhies
Barry Vornbrock
Clarence M. Weaver
Donald F. Weir
Alan J. Weissberger
Raymond Wenig
William J. Wenker
Earl J. Whitaker
Thomas P. Wiggen
Michael Willett
Paul A. Willis
George B. Wright
Jen-Kun Yang
Oren Yuen
William H. Yundt
Zhao Wei

When the IEEE Standards Board approved ANSI/IEEE Std 802.3i-1990 on September 28, 1990, it had the following membership:

Marco W. Migliaro, *Chairman* **James M. Daly**, *Vice Chairman*

Andrew G. Salem, *Secretary*

Dennis Bodson
Paul L. Borrill
Fletcher J. Buckley
Allen L. Clapp
Stephen R. Dillon
Donald C. Fleckenstein
Jay Forster*
Thomas L. Hannan

Kenneth D. Hendrix
John W. Horch
Joseph L. Koepfinger*
Irving Kolodny
Michael A. Lawler
Donald J. Loughry
John E. May, Jr.

Lawrence V. McCall
L. Bruce McClung
Donald T. Michael*
Stig Nilsson
Roy T. Oishi
Gary S. Robinson
Terrance R. Whittemore
Donald W. Zipse

*Member Emeritus

ANSI/IEEE Std 802.3i-1990 was approved by the American National Standards Institute on March 11, 1991.

Contents

SECTION PAGE

SECTION PAGE

SECTION PAG

SECTION PAGE

SECTION PAGE

FIGURES

FIGURES PAGE

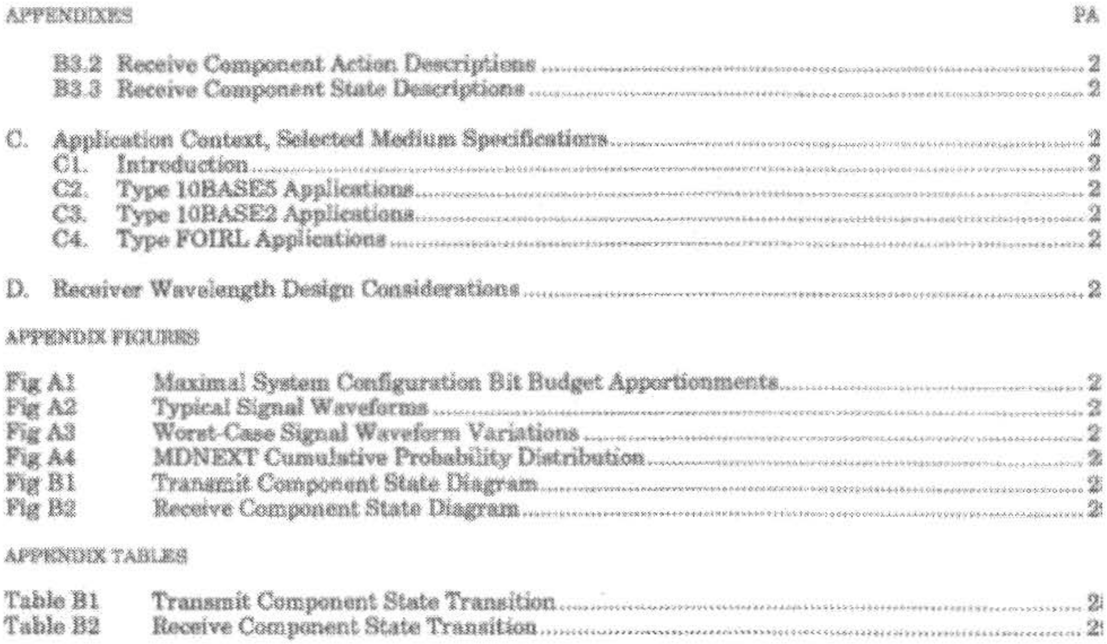
FIGURES PA

FIGURES PAGE

TABLES

ANNEX

APPENDIXES



APPENDIXES PA

APPENDIX FIGURES

APPENDIX TABLES

Information technology—Local and metropolitan area networks—

Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications

1. Introduction

1.1 Overview

1.1.1 Basic Concepts. The Carrier Sense Multiple Access with Collision Detection (CSMA/CD) media access method is the means by which two or more stations share a common transmission medium. To transmit, a station waits (defers) for a quiet period on the medium (that is, no other station is transmitting) and then sends the intended message in bit-serial form. If, after initiating a transmission, the message collides with that of another station, then each transmitting station intentionally sends a few additional bytes to ensure propagation of the collision throughout the system. The station remains silent for a random amount of time (backoff) before attempting to transmit again. Each aspect of this access method process is specified in detail in subsequent sections of this standard.

This is a comprehensive standard for Local Area Networks employing CSMA/CD as the access method. This standard is intended to encompass several media types and techniques for signal rates of from 1 Mb/s to 20 Mb/s. This edition of the standard provides the necessary specifications for 10 Mb/s baseband and broadband systems, a 1 Mb/s baseband system, and a Repeater Unit.

1.1.2 Architectural Perspectives. There are two important ways to view local area network design corresponding to

(1) *Architecture.* Emphasizing the logical divisions of the system and how they fit together.
(2) *Implementation.* Emphasizing actual components, their packaging and interconnection.

This standard is organized along architectural lines, emphasizing the large-scale separation of the system into two parts: the Media Access Control (MAC) sublayer of the Data Link Layer, and the Physical Layer. These layers are intended to correspond closely to the lowest layers of the ISO Model for Open Systems Interconnection (see Fig 1-1). See ISO 7498:1984 [10].[1] The Logical Link Control (LLC) sublayer and MAC sublayer together encompass the functions intended for the Data Link Layer as defined in the OSI model.

1.1.2.1 An architectural organization of the standard has two main advantages:

(1) *Clarity.* A clean overall division of the design along architectural lines makes the standard clearer.
(2) *Flexibility.* Segregation of medium-dependent aspects in the Physical Layer allows the LLC and MAC sublayers to apply to a family of transmission media.

Partitioning the Data Link Layer allows various media access methods within the family of Local Area Network standards.

[1] The numbers in brackets correspond to those of the references listed in 1.3; when preceded by A, they correspond to those listed in the Annex.

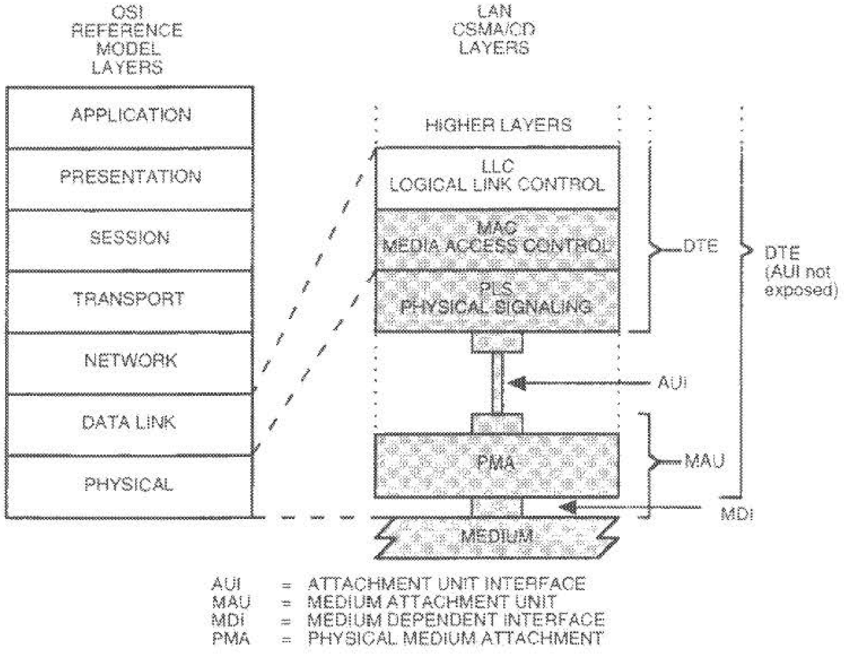


Fig 1-1
LAN Standard Relationship to the ISO Open Systems Interconnection (OSI) Reference Model

The architectural model is based on a set of interfaces that may be different from those emphasized implementations. One critical aspect of the design, however, shall be addressed largely in terms of tl implementation interfaces: compatibility.

1.1.2.2 Two important compatibility interfaces are defined within what is architecturally the Physic Layer.

(1) *Medium-Dependent Interface (MDI).* To communicate in a compatible manner, all stations sh adhere rigidly to the exact specification of physical media signals defined in Section 8 (and beyon in this standard, and to the procedures that define correct behavior of a station. The medium-ind pendent aspects of the LLC sublayer and the MAC sublayer should not be taken as detracting fro this point; communication by way of the ISO 8802-3 [IEEE 802.3] Local Area Network requires co plete compatibility at the Physical Medium interface (that is, the coaxial cable interface).

(2) *Attachment Unit Interface (AUI).* It is anticipated that most DTEs will be located some distan from their connection to the coaxial cable. A small amount of circuitry will exist in the Mediu Attachment Unit (MAU) directly adjacent to the coaxial cable, while the majority of the hardwa and all of the software will be placed within the DTE. The AUI is defined as a second compatibili interface. While conformance with this interface is not strictly necessary to ensure communicatio it is highly recommended, since it allows maximum flexibility in intermixing MAUs and DTEs. Tl AUI may be optional or not specified for some implementations of this standard that are expected be connected directly to the medium and so do not use a separate MAU or its interconnecting Al cable. The PLS and PMA are then part of a single unit, and no explicit AUI specification is require

1.1.3 Layer Interfaces. In the architectural model used here, the layers interact by way of well defin interfaces, providing services as specified in Sections 2 and 6. In general, the interface requirements are follows.

(1) The interface between the MAC sublayer and the LLC sublayer includes facilities for transmitting and receiving frames, and provides per-operation status information for use by higher-layer error recovery procedures.

(2) The interface between the MAC sublayer and the Physical Layer includes signals for framing (carrier sense, transmit initiation) and contention resolution (collision detect), facilities for passing a pair of serial bit streams (transmit, receive) between the two layers, and a wait function for timing.

These interfaces are described more precisely in 4.3. Additional interfaces are necessary to allow higher level network management facilities to interact with these layers to perform operation, maintenance, and planning functions. Network management functions will be discussed in Section 5.

1.1.4 Application Areas. The applications environment for the Local Area Network is intended to be commercial and light industrial. Use of CSMA/CD LANs in home or heavy industrial environments, while not precluded, is not considered within the scope of this standard.

1.2 Notation

1.2.1 State Diagram Conventions. The operation of a protocol can be described by subdividing the protocol into a number of interrelated functions. The operation of the functions can be described by state diagrams. Each diagram represents the domain of a function and consists of a group of connected, mutually exclusive states. Only one state of a function is active at any given time (see Fig 1-2).

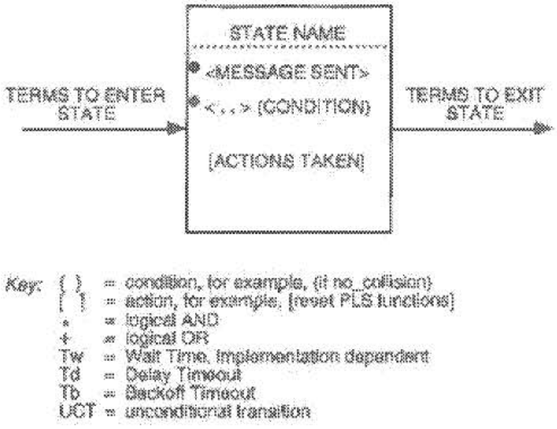

STATE NAME
<MESSAGE SENT>
<' . . > (CONDITION)
[ACTIONS TAKEN]
TERMS TO ENTER STATE
TERMS TO EXIT STATE
Key: { } = condition, for example, (if no_collision)
[ ] = action, for example, [reset PLS functions]
* = logical AND
+ = logical OR
Tw = Wait Time, implementation dependent
Td = Delay Timeout
Tb = Backoff Timeout
UCT = unconditional transition

Fig 1-2
State Diagram Notation Example

Each state that the function can assume is represented by a rectangle. These are divided into two parts by a horizontal line. In the upper part the state is identified by a name in capital letters. The lower part contains the name of any ON signal that is generated by the function. Actions are described by short phrases and enclosed in brackets.

All permissible transitions between the states of a function are represented graphically by arrows between them. A transition that is global in nature (for example, an exit condition from all states to the IDLE or RESET state) is indicated by an open arrow. Labels on transitions are qualifiers that must be fulfilled before the transition will be taken. The label UCT designates an unconditional transition. Qualifiers described by short phrases are enclosed in parentheses.

State transitions and sending and receiving of messages occur instantaneously. When a state is entered and the condition to leave that state is not immediately fulfilled, the state executes continuously, sending the messages and executing the actions contained in the state in a continuous manner.

Some devices described in this standard (e.g., repeaters) are allowed to have two or more ports. Sta diagrams capable of describing the operation of devices with an unspecified number of ports, required qua ifier notation that allows testing for conditions at multiple ports. The notation used is a term that includ a description in parentheses of which ports must meet the term for the qualifier to be satisfied (e.g., AN and ALL). It is also necessary to provide for term-assignment statements that assign a name to a port th satisfies a qualifier. The following convention is used to describe a term-assignment statement that is ass ciated with a transition:

(1) The character ":" (colon) is a delimiter used to denote that a term assignment statement follows.
(2) The character "⇐" (left arrow) denotes assignment of the value following the arrow to the term pr ceding the arrow.

The state diagrams contain the authoritative statement of the functions they depict; when apparent co flicts between descriptive text and state diagrams arise, the state diagrams are to take precedence. Th does not override, however, any explicit description in the text that has no parallel in the state diagrams

The models presented by state diagrams are intended as the primary specifications of the functions to l provided. It is important to distinguish, however, between a model and a real implementation. The mode are optimized for simplicity and clarity of presentation, while any realistic implementation may pla heavier emphasis on efficiency and suitability to a particular implementation technology. It is the fun tional behavior of any unit that must match the standard, not its internal structure. The internal details the model are useful only to the extent that they specify the external behavior clearly and precisely.

1.2.2 Service Specification Method and Notation. The service of a layer or sublayer is the set capabilities that it offers to a user in the next higher (sub)layer. Abstract services are specified here l describing the service primitives and parameters that characterize each service. This definition of servi is independent of any particular implementation (see Fig 1-3).

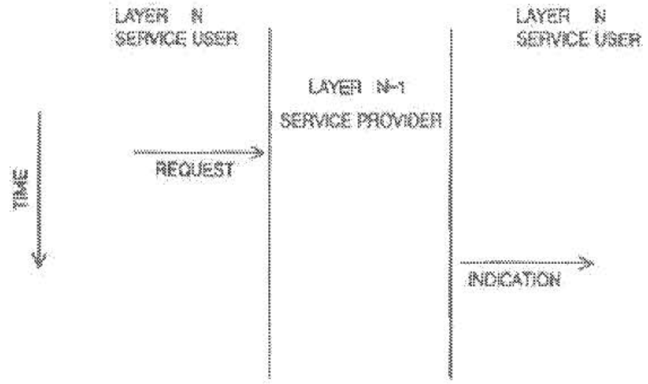


Fig 1-3
Service Primitive Notation

Specific implementations may also include provisions for interface interactions that have no direct en to-end effects. Examples of such local interactions include interface flow control, status requests and ind cations, error notifications, and layer management. Specific implementation details are omitted from th service specification both because they will differ from implementation to implementation and becau they do not impact the peer-to-peer protocols.

1.2.2.1 Classification of Service Primitives. Primitives are of two generic types:

(1) REQUEST. The request primitive is passed from layer N to layer N-1 to request that a service l initiated.

(2) INDICATION. The indication primitive is passed from layer N-1 to layer N to indicate an internal layer N-1 event that is significant to layer N. This event may be logically related to a remote service request, or may be caused by an event internal to layer N-1.

The service primitives are an abstraction of the functional specification and the user-layer interaction. The abstract definition does not contain local detail of the user/provider interaction. For instance, it does not indicate the local mechanism that allows a user to indicate that it is awaiting an incoming call. Each primitive has a set of zero or more parameters, representing data elements that shall be passed to qualify the functions invoked by the primitive. Parameters indicate information available in a user/provider interaction; in any particular interface, some parameters may be explicitly stated (even though not explicitly defined in the primitive) or implicitly associated with the service access point. Similarly, in any particular protocol specification, functions corresponding to a service primitive may be explicitly defined or implicitly available.

1.2.3 Physical Layer and Media Notation. Users of this standard need to reference which particular implementation is being used or identified. Therefore, a means of identifying each implementation is given by a simple, three-field, type notation that is explicitly stated at the beginning of each relevant section. In general, the Physical Layer type is specified by these fields:

<data rate in Mb/s> <medium type> <maximum segment length (× 100 m)>

For example, the standard contains a 10 Mb/s baseband specification identified as "TYPE 10BASE5," meaning a 10 Mb/s baseband medium whose maximum segment length is 500 m. Each successive Physical Layer specification will state its own unique TYPE identifier along similar lines.

1.2.4 Physical Layer Message Notation. Messages generated within the Physical Layer, either within or between PLS and the MAU (that is, PMA circuitry), are designated by an italic type to designate either form of physical or logical message used to execute the physical layer signaling process (for example, *input_idle* or *mau_available*).

1.3 References

[1] CISPR Publication 22 (1985), Limits and Methods of Measurement of Radio Interference Characteristics of Information Technology Equipment.[2]

[2] IEC Publication 96-1, Radio-frequency cables, Part 1: General requirements and measuring methods.[3]

[3] IEC Publication 96-1A, 1st Supplement to Radio-frequency cables, Part 1: Appendix Section 5.4, Terminated triaxial test method for transfer impedance up to 100 MHz.

[4] IEC Publication 169-8 and -16, Radio-frequency connectors, Part 8: Radio-frequency coaxial connectors with inner diameter of outer conductor 6.5 mm (0.256 in) with bayonet lock—Characteristic impedance 50 ohms (Type BNC); Part 16: Radio-frequency coaxial connectors with inner diameter of outer conductor 7 mm (0.276 in) with screw coupling—Characteristic impedance 50 ohms (75 ohms) (Type N).

[5] IEC Publication 380, Safety of electrically energized office machines.

[6] IEC Publication 435, Safety of data processing equipment.

[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.

[7] IEC Publication 807-2, Rectangular connectors for frequencies below
3 MHz, Part 2: Detail specification for a range of connectors with round contacts—Fixed solder contact types.

[8] IEC Publication 950, Safety of Information Technology Equipment, Including Electrical Business Equipment.

[9] ISO 2382-9 : 1984, Data processing—Vocabulary—Part 09: Data communication.[4]

[10] ISO 7498 : 1984, Information processing systems–open systems interconnection–Basic reference model.

[11] IEC Publication 60, High-voltage test techniques.

[12] IEC Publication 68, Environmental testing.

[13] IEC Publication 793-1, Optical fibres, Part 1: Generic specification.

[14] IEC Publication 793-2, Optical fibres, Part 2: Product specifications.[5]

[15] IEC Publication 794-1, Optical fibre cables, Part 1: Generic specification.

[16] IEC Publication 794-2, Optical fibres cables, Part 2: Product specifications.

[17] IEC Publication 825, Radiation safety of laser products, equipment classification, requirements, and user's guide.

[18] IEC Publication 874-1, Connectors for optical fibres and cables, Part 1: Generic specification.

[19] IEC Publication 874-2, Connectors for optical fibres and cables, Part 2: Sectional specification for fibre optic connector type F-SMA.

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

[21] ISO 8877 : 1987, Information processing systems—Interface connector and contact assignments for ISDN basic access interface located at reference points S and T.

Local and national standards such as those supported by ANSI, EIA, IEEE, MIL, NFPA, and UL are not a formal part of the ISO/IEC 8802-3 standard. Reference to such local or national standards may be useful resource material and are identified by a bracketed number beginning with the letter A and located in Annex A.

1.4 Definitions. The definitions used in this standard are consistent with ISO 2382-9:1984 [9]. A more specific Part 25, pertaining to LAN systems, is in development.

[4] ISO 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.

[5] Subsection 9.9 is to be read with the understanding that the following changes to IEC Publication 793-2 [14] have been requested
(1) Correction of the numerical aperture tolerance in Table III to ± 0.015.
(2) Addition of another bandwidth category, of ≥ 150 MHz referred to 1 km, for the type A1b fiber in Table III.

2. MAC Service Specification

2.1 Scope and Field of Application. This section specifies the services provided by the Media Access Control (MAC) sublayer to the Logical Link Control (LLC) sublayer for the ISO [IEEE] Local Area Network standard (see Fig 2-1). The services are described in an abstract way and do not imply any particular implementation, or any exposed interface. There is not necessarily a one-to-one correspondence between the primitives and the formal procedures and interfaces described in 4.2 and 4.3.

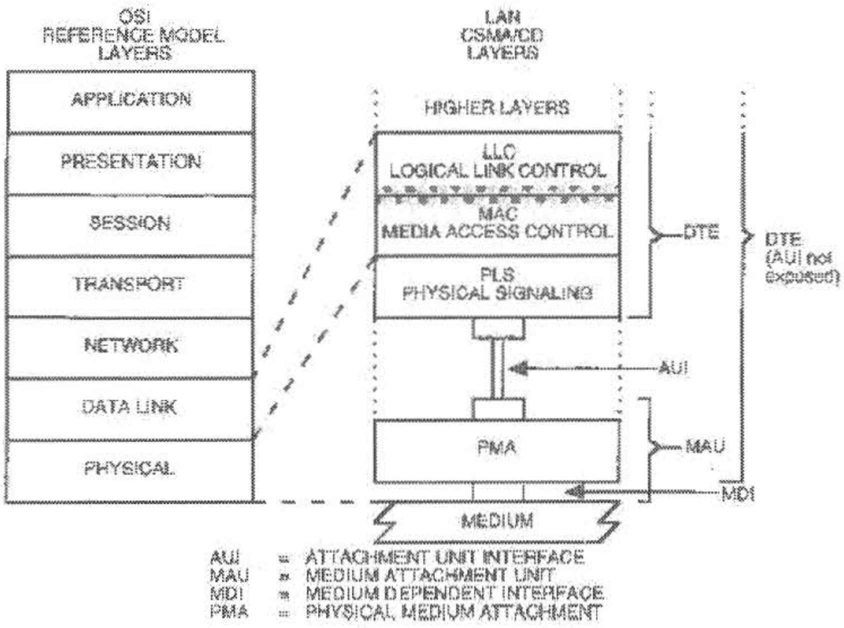


Fig 2-1
Service Specification Relation to the LAN Model

2.2 Overview of the Service

2.2.1 General Description of Services Provided by the Layer. The services provided by the MAC sublayer allow the local LLC sublayer entity to exchange LLC data units with peer LLC sublayer entities. Optional support may be provided for resetting the MAC sublayer entity to a known state.

2.2.2 Model Used for the Service Specification. The model used in this service specification is identical to that used in 1.2.

2.2.3 Overview of Interactions

MA_DATA.request
MA_DATA.indication

2.2.4 Basic Services and Options. The MA_DATA.request and MA_DATA.indication service primitives described in this section are considered mandatory.

2.3 Detailed Service Specification

2.3.1 MA_DATA.request

2.3.1.1 Function. This primitive defines the transfer of data from a local LLC sublayer entity to a single peer LLC entity or multiple peer LLC entities in the case of group addresses.

2.3.1.2 Semantics of the Service Primitive. The semantics of the primitive are as follows:

```
MA_DATA.request     (
                    destination_address,
                    m_sdu,
                    service_class
                    )
```

The destination_address parameter may specify either an individual or a group MAC entity address. It must contain sufficient information to create the DA field that is appended to the frame by the local MAC sublayer entity and any physical information. The m_sdu parameter specifies the MAC service data unit to be transmitted by the MAC sublayer entity. There is sufficient information associated with m_sdu for the MAC sublayer entity to determine the length of the data unit. The service_class parameter indicates a quality of service requested by LLC or higher layer (see 2.3.1.5).

2.3.1.3 When Generated. This primitive is generated by the LLC sublayer entity whenever data shall be transferred to a peer LLC entity or entities. This can be in response to a request from higher protocol layers or from data generated internally to the LLC sublayer, such as required by Type 2 service.

2.3.1.4 Effect of Receipt. The receipt of this primitive will cause the MAC entity to append all MAC specific fields, including DA, SA, and any fields that are unique to the particular media access method, and pass the properly formed frame to the lower protocol layers for transfer to the peer MAC sublayer entity or entities.

2.3.1.5 Additional Comments. The CSMA/CD MAC protocol provides a single quality of service regardless of the service_class requested.

2.3.2 MA_DATA.indication

2.3.2.1 Function. This primitive defines the transfer of data from the MAC sublayer entity to the LLC sublayer entity or entities in the case of group addresses.

2.3.2.2 Semantics of the Service Primitive. The semantics of the primitive are as follows:

```
MA_DATA.indication  (
                    destination_address,
                    source_address,
                    m_sdu,
                    reception_status
                    )
```

The destination_address parameter may be either an individual or a group address as specified by the DA field of the incoming frame. The source_address parameter is an individual address as specified by the SA field of the incoming frame. The m_sdu parameter specifies the MAC service data unit as received by the local MAC entity. The reception_status parameter is used to pass status information to the peer LLC sublayer entity.

2.3.2.3 When Generated. The MA_DATA.indication is passed from the MAC sublayer entity to the LLC sublayer entity or entities to indicate the arrival of a frame to the local MAC sublayer entity. Such frames are reported only if they are validly formed, received without error, and their destination address designates the local MAC entity.

2.3.2.4 Effect of Receipt. The effect of receipt of this primitive by the LLC sublayer is unspecified.

2.3.2.5 Additional Comments. If the local MAC sublayer entity is designated by the destination_address parameter of an MA_DATA.request, the indication primitive will also be invoked by the MAC entity to the local LLC entity. This full duplex characteristic of the MAC sublayer may be due to unique functionality within the MAC sublayer or full duplex characteristics of the lower layers (for example, all frames transmitted to the broadcast address will invoke MA_DATA.indication at all stations in the network including the station that generated the request).

3. Media Access Control Frame Structure

3.1 Overview. This section defines in detail the frame structure for data communication systems using local area network MAC procedures. It defines the relative positions of the various components of the MAC frame. It defines the method for representing station addresses. It defines a partition of the address space into individual (single station) and group (multicast or multistation) addresses, and into user administered and globally administered addresses.

3.1.1 MAC Frame Format. Figure 3-1 shows the eight fields of a frame: the preamble, Start Frame Delimiter (SFD), the addresses of the frame's source and destination, a length field to indicate the length of the following field containing the LLC data to be transmitted, a field that contains padding if required, and the frame check sequence field containing a cyclic redundancy check value to detect errors in received frames. Of these eight fields, all are of fixed size except the LLC data and PAD fields, which may contain any integer number of octets between the minimum and maximum values determined by the specific implementation of the CSMA/CD Media Access mechanism. See 4.4 for particular implementations.

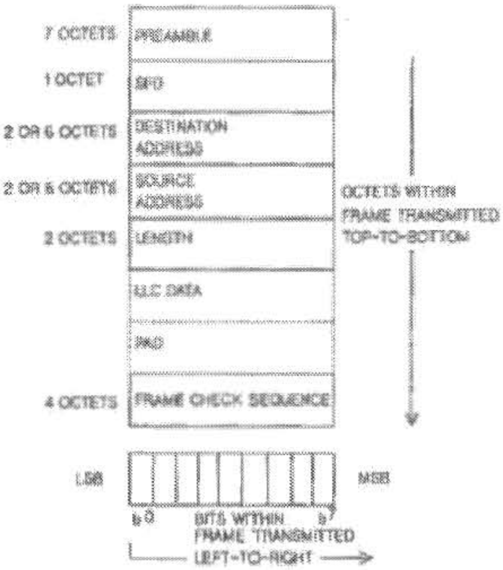


Fig 3-1
MAC Frame Format

The minimum and maximum frame size limits in 4.4 refer to that portion of the frame from the destination address field through the frame check sequence field, inclusive.

Relative to Fig 3-1, the octets of a frame are transmitted from top to bottom, and the bits of each octet are transmitted from left to right.

3.2 Elements of the MAC Frame

3.2.1 Preamble Field. The preamble field is a 7-octet field that is used to allow the PLS circuitry to reach its steady-state synchronization with the received frame timing (see 4.2.5).

3.2.2 Start Frame Delimiter (SFD) Field. The SFD field is the sequence 10101011. It immediately f lows the preamble pattern and indicates the start of a frame.

3.2.3 Address Fields. Each MAC frame shall contain two address fields: the Destination Address fi and the Source Address field, in that order. The Destination Address field shall specify the destinati addressee(s) for which the frame is intended. The Source Address field shall identify the station from whi the frame was initiated. The representation of each address field shall be as follows (see Fig 3-2):

(1) Each address field shall contain either 16 bits or 48 bits. However, at any given time, the Source a Destination Address size shall be the same for all stations on a particular local area network.
(2) The support of 16 or 48 bit address length for Source and Destination Address shall be left to t manufacturer as an implementation decision. There is no requirement that manufacturers supp both sizes.
(3) The first bit (LSB) shall be used in the Destination Address field as an address type designation l to identify the Destination Address either as an individual or as a group address. In the Sour Address field, the first bit is reserved and set to 0. If this bit is 0, it shall indicate that the addre field contains an individual address. If this bit is 1, it shall indicate that the address field contain group address that identifies none, one or more, or all of the stations connected to the local ar network.
(4) For 48 bit addresses, the second bit shall be used to distinguish between locally or globally admin tored addresses. For globally administered (or U, universal) addresses, the bit is set to 0. If address is to be assigned locally, this bit shall be set to 1. Note that for the broadcast address, tl bit is also a 1.
(5) Each octet of each address field shall be transmitted least significant bit first.

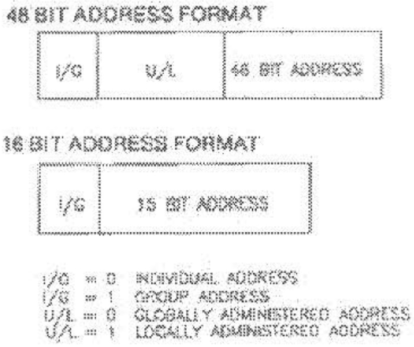


Fig 3-2
Address Field Format

3.2.3.1 Address Designation. A MAC sublayer address is of one of two types:

(1) *Individual Address.* The address associated with a particular station on the network.
(2) *Group Address.* A multidestination address, associated with one or more stations on a given n work. There are two kinds of multicast address:
    (a) *Multicast-Group Address.* An address associated by higher-level convention with a group of l ically related stations.
    (b) *Broadcast Address.* A distinguished, predefined multicast address that always denotes the of all stations on a given local area network.

All 1's in the Destination Address field (for 16 or 48 bit address size LANs) shall be predefined to be t Broadcast address. This group shall be predefined for each communication medium to consist of all static actively connected to that medium; it shall be used to broadcast to all the active stations on that medium. All stations shall be able to recognize the Broadcast address. It is not necessary that a station be capable of generating the Broadcast address.

The address space shall also be partitioned into locally administered and globally administered addresses. The nature of a body and the procedures by which it administers these global (U) addresses is beyond the scope of this standard.[6]

3.2.4 Destination Address Field. The Destination Address field specifies the station(s) for which the frame is intended. It may be an individual or multicast (including broadcast) address.

3.2.5 Source Address Field. The Source Address field specifies the station sending the frame. The Source Address field is not interpreted by the CSMA/CD MAC sublayer.

3.2.6 Length Field. The length field is a 2-octet field whose value[7] indicates the number of LLC data octets in the data field. If the value is less than the minimum required for proper operation of the protocol, a PAD field (a sequence of octets) will be added at the end of the data field but prior to the FCS field, specified below. The procedure that determines the size of the pad field is specified in 4.2.8. The length field is transmitted and received with the high order octet first.

3.2.7 Data and PAD Fields. The data field contains a sequence of n octets. Full data transparency is provided in the sense that any arbitrary sequence of octet values may appear in the data field up to a maximum number specified by the implementation of this standard that is used. A minimum frame size is required for correct CSMA/CD protocol operation and is specified by the particular implementation of the standard. If necessary, the data field is extended by appending extra bits (that is, a pad) in units of octets after the LLC data field but prior to calculating and appending the FCS. The size of the pad, if any, is determined by the size of the data field supplied by LLC and the minimum frame size and address size parameters of the particular implementation. The maximum size of the data field is determined by the maximum frame size and address size parameters of the particular implementation.

The length of PAD field required for LLC data that is n octets long is max (0, minFrameSize – (8 × n + 2 × addressSize + 48)) bits. The maximum possible size of the LLC data field is maxFrameSize – (2 × addressSize + 48)/8 octets. See 4.4 for a discussion of implementation parameters; see 4.2.3.3 for a discussion of the minFrameSize.

3.2.8 Frame Check Sequence Field. A cyclic redundancy check (CRC) is used by the transmit and receive algorithms to generate a CRC value for the FCS field. The frame check sequence (FCS) field contains a 4-octet (32-bit) cyclic redundancy check (CRC) value. This value is computed as a function of the contents of the source address, destination address, length, LLC data and pad (that is, all fields except the preamble, SFD, and FCS). The encoding is defined by the following generating polynomial.

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

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

(1) The first 32 bits of the frame are complemented.

(2) 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 Destination Address field corresponds to the $x^{(n-1)}$ term and the last bit of the data field corresponds to the x^0 term.)

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

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

(5) The bit sequence is complemented and the result is the CRC.

[6] For information on how to use MAC addresses, see IEEE Std 802-1990, Overview and Architecture. To apply for an Organizationally Unique Identifier for building a MAC address, contact the Registration Authority, IEEE Standards Department, P.O. Box 1331, 445 Hoes Lane, Piscataway, NJ 08855-1331, USA; (908) 562-3813; fax (908) 562-1571.

[7] Packets with a length field value greater than those specified in 4.4.2 may be ignored, discarded, or used in a private manner. The use of such packets is beyond the scope of this standard.

The 32 bits of the CRC value are placed in the frame check sequence field so that the x^{31} term is the le most bit of the first octet, and the x^0 term is the right most bit of the last octet. (The bits of the CRC a thus transmitted in the order $x^{31}, x^{30}, \ldots, x^1, x^0$.) See reference [A18].

3.3 Order of Bit Transmission. Each octet of the MAC frame, with the exception of the FCS, transmitted low-order bit first.

3.4 Invalid MAC Frame. An invalid MAC frame shall be defined as one that meets at least one of t following conditions:

(1) The frame length is inconsistent with the length field.
(2) It is not an integral number of octets in length.
(3) The bits of the incoming frame (exclusive of the FCS field itself) do not generate a CRC value iden cal to the one received.

The contents of invalid MAC frames shall not be passed to LLC. The occurrence of invalid MAC fram may be communicated to network management.

4. Media Access Control

4.1 Functional Model of the Media Access Control Method

4.1.1 Overview. The architectural model described in Section 1 is used in this section to provide a functional description of the Local Area Network CSMA/CD MAC sublayer.

The MAC sublayer defines a medium-independent facility, built on the medium-dependent physical facility provided by the Physical Layer, and under the access-layer-independent local area network LLC sublayer. It is applicable to a general class of local area broadcast media suitable for use with the media access discipline known as Carrier Sense Multiple Access with Collision Detection (CSMA/CD).

The LLC sublayer and the MAC sublayer together are intended to have the same function as that described in the OSI model for the Data Link Layer alone. In a broadcast network, the notion of a data link between two network entities does not correspond directly to a distinct physical connection. Nevertheless, the partitioning of functions presented in this standard requires two main functions generally associated with a data link control procedure to be performed in the MAC sublayer. They are as follows:

(1) Data encapsulation (transmit and receive)
  (a) Framing (frame boundary delimitation, frame synchronization)
  (b) Addressing (handling of source and destination addresses)
  (c) Error detection (detection of physical medium transmission errors)
(2) Media Access Management
  (a) Medium allocation (collision avoidance)
  (b) Contention resolution (collision handling)

The remainder of this section provides a functional model of the CSMA/CD MAC method.

4.1.2 CSMA/CD Operation. This section provides an overview of frame transmission and reception in terms of the functional model of the architecture. This overview is descriptive, rather than definitional; the formal specifications of the operations described here are given in 4.2 and 4.3. Specific implementations for CSMA/CD mechanisms that meet this standard are given in 4.4. Figure 4-1 provides the architectural model described functionally in the sections that follow.

The Physical Layer Signaling (PLS) component of the Physical Layer provides an interface to the MAC sublayer for the serial transmission of bits onto the physical media. For completeness, in the operational description that follows some of these functions are included as descriptive material. The concise specification of these functions is given in 4.2 for the MAC functions and in Section 7 for PLS.

Transmit frame operations are independent from the receive frame operations. A transmitted frame addressed to the originating station will be received and passed to the LLC sublayer at that station. This characteristic of the MAC sublayer may be implemented by functionality within the MAC sublayer or full duplex characteristics of portions of the lower layers.

4.1.2.1 Normal Operation

4.1.2.1.1 Transmission Without Contention. When a LLC sublayer requests the transmission of a frame, the Transmit Data Encapsulation component of the CSMA/CD MAC sublayer constructs the frame from the LLC-supplied data. It appends a preamble and a start of frame delimiter to the beginning of the frame. Using information passed by the LLC sublayer, the CSMA/CD MAC sublayer also appends a PAD at the end of the MAC information field of sufficient length to ensure that the transmitted frame length satisfies a minimum frame size requirement (see 4.2.3.3). It also appends destination and source addresses, a length count field, and a frame check sequence to provide for error detection. The frame is then handed to the Transmit Media Access Management component in the MAC sublayer for transmission.

Transmit Media Access Management then attempts to avoid contention with other traffic on the medium by monitoring the carrier sense signal provided by the Physical Layer Signaling (PLS) component and deferring to passing traffic. When the medium is clear, frame transmission is initiated (after a brief inter-

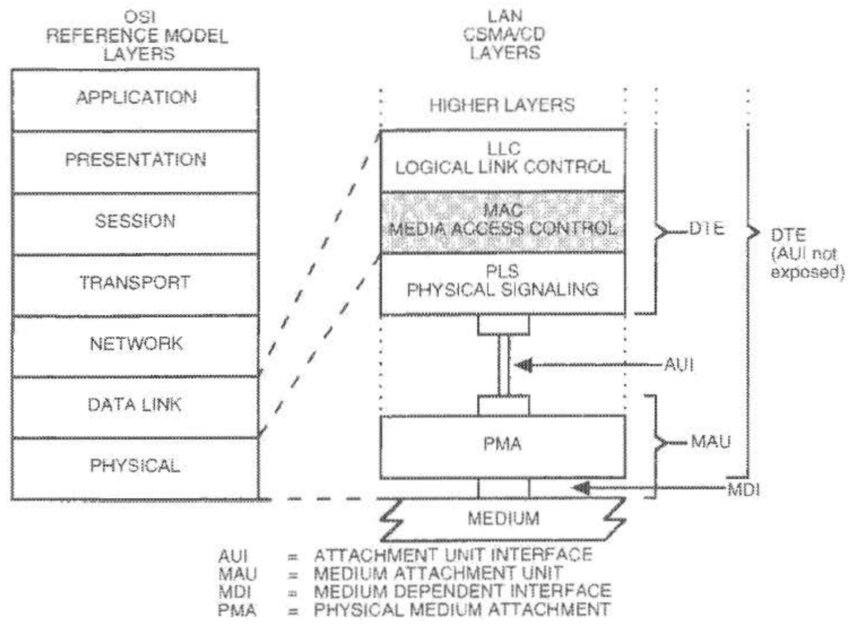


Fig 4-1
MAC Sublayer Partitioning, Relationship to the ISO Open Systems Interconnection (OSI) Reference Model

frame delay to provide recovery time for other CSMA/CD MAC sublayers and for the physical mediur The MAC sublayer then provides a serial stream of bits to the PLS interface for transmission.

The PLS performs the task of actually generating the electrical signals on the medium that represent t bits of the frame. Simultaneously, it monitors the medium and generates the collision detect signal, whic in the contention-free case under discussion, remains off for the duration of the frame. A functional descri tion of the Physical Layer is given in Sections 7 and beyond.

When transmission has completed without contention, the CSMA/CD MAC sublayer so informs the LI sublayer using the LLC to MAC interface and awaits the next request for frame transmission.

4.1.2.1.2 Reception Without Contention. At each receiving station, the arrival of a frame is fi detected by the PLS, which responds by synchronizing with the incoming preamble, and by turning on t carrier sense signal. As the encoded bits arrive from the medium, they are decoded and translated ba into binary data. The PLS passes subsequent bits up to the MAC sublayer, where the leading bits are d carded, up to and including the end of the preamble and Start Frame Delimiter.

Meanwhile, the Receive Media Access Management component of the MAC sublayer, having observ carrier sense, has been waiting for the incoming bits to be delivered. Receive Media Access Manageme collects bits from the PLS as long as the carrier sense signal remains on. When the carrier sense signal removed, the frame is truncated to an octet boundary, if necessary, and passed to Receive Data Decapsul tion for processing.

Receive Data Decapsulation checks the frame's Destination Address field to decide whether the frar should be received by this station. If so, it passes the Destination Address (DA), the Source Address (S/ and the LLC data unit (LLCDU) to the LLC sublayer along with an appropriate status code indicati reception_complete or reception_too_long. It also checks for invalid MAC frames by inspecting the frar check sequence to detect any damage to the frame enroute, and by checking for proper octet-bounda alignment of the end of the frame. Frames with a valid FCS may also be checked for proper octet bounda alignment.

4.1.2.2 Access Interference and Recovery. If multiple stations attempt to transmit at the sar time, it is possible for them to interfere with each other's transmissions, in spite of their attempts to avc this by deferring. When transmissions from two stations overlap, the resulting contention is called a collision. A given station can experience a collision during the initial part of its transmission (the collision window) before its transmitted signal has had time to propagate to all stations on the CSMA/CD medium. Once the collision window has passed, a transmitting station is said to have acquired the medium; subsequent collisions are avoided since all other (properly functioning) stations can be assumed to have noticed the signal (by way of carrier sense) and to be deferring to it. The time to acquire the medium is thus based on the round-trip propagation time of the physical layer whose elements include the PLS, PMA, and physical medium.

In the event of a collision, the transmitting station's Physical Layer initially notices the interference on the medium and then turns on the collision detect signal. This is noticed in turn by the Transmit Media Access Management component of the MAC sublayer, and collision handling begins. First, Transmit Media Access Management enforces the collision by transmitting a bit sequence called jam. In 4.4 an implementation that uses this enforcement procedure is provided. This ensures that the duration of the collision is sufficient to be noticed by the other transmitting station(s) involved in the collision. After the jam is sent, Transmit Media Access Management terminates the transmission and schedules another transmission attempt after a randomly selected time interval. Retransmission is attempted again in the face of repeated collisions. Since repeated collisions indicate a busy medium, however, Transmit Media Access Management attempts to adjust to the medium load by backing off (voluntarily delaying its own retransmissions to reduce its load on the medium). This is accomplished by expanding the interval from which the random retransmission time is selected on each successive transmit attempt. Eventually, either the transmission succeeds, or the attempt is abandoned on the assumption that the medium has failed or has become overloaded.

At the receiving end, the bits resulting from a collision are received and decoded by the PLS just as are the bits of a valid frame. Fragmentary frames received during collisions are distinguished from valid transmissions by the MAC sublayer's Receive Media Access Management component.

4.1.3 Relationships to LLC Sublayer and Physical Layer. The CSMA/CD MAC sublayer provides services to the LLC sublayer required for the transmission and reception of frames. Access to these services is specified in 4.3. The CSMA/CD MAC sublayer makes a best effort to acquire the medium and transfer a serial stream of bits to the PLS. Although certain errors are reported to the LLC, error recovery is not provided by MAC. Error recovery may be provided by the LLC or higher (sub)layers.

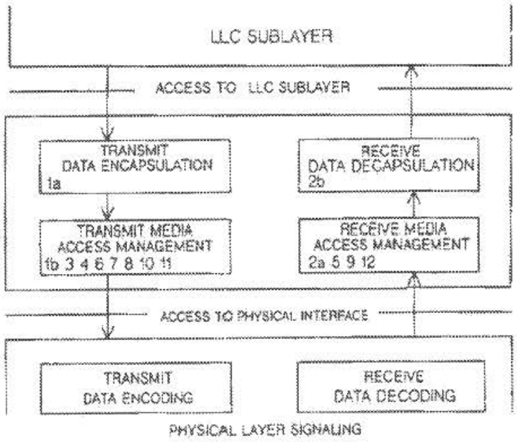
4.1.4 CSMA/CD Access Method Functional Capabilities. The following summary of the functional capabilities of the CSMA/CD MAC sublayer is intended as a quick reference guide to the capabilities of the standard, as depicted in Fig 4-2:

(1) For Frame Transmission
  (a) Accepts data from the LLC sublayer and constructs a frame
  (b) Presents a bit-serial data stream to the physical layer for transmission on the medium

  NOTE: Assumes data passed from the LLC sublayer are octet multiples.

(2) For Frame Reception
  (a) Receives a bit-serial data stream from the physical layer
  (b) Presents to the LLC sublayer frames that are either broadcast frames or directly addressed to the local station
  (c) Discards or passes to Network Management all frames not addressed to the receiving station
(3) Defers transmission of a bit-serial stream whenever the physical medium is busy
(4) Appends proper FCS value to outgoing frames and verifies full octet boundary alignment
(5) Checks incoming frames for transmission errors by way of FCS and verifies octet boundary alignment
(6) Delays transmission of frame bit stream for specified interframe gap period
(7) Halts transmission when collision is detected
(8) Schedules retransmission after a collision until a specified retry limit is reached
(9) Enforces collision to ensure propagation throughout network by sending jam message
(10) Discards received transmissions that are less than a minimum length
(11) Appends preamble, Start Frame Delimiter, DA, SA, length count, and FCS to all frames, and inserts pad field for frames whose LLC data length is less than a minimum value

(12) Removes preamble, Start Frame Delimiter, DA, SA, length count, FCS and pad field (if necessary) from received frames


LLC SUBLAYER
ACCESS TO LLC SUBLAYER
TRANSMIT DATA ENCAPSULATION
1a
RECEIVE DATA DECAPSULATION
2b
TRANSMIT MEDIA ACCESS MANAGEMENT
1b 3 4 6 7 8 10 11
RECEIVE MEDIA ACCESS MANAGEMENT
2a 5 9 12
ACCESS TO PHYSICAL INTERFACE
TRANSMIT DATA ENCODING
RECEIVE DATA DECODING
PHYSICAL LAYER SIGNALING

NOTE: Numbers refer to functions listed in 4.1.4.

Fig 4-2
CSMA/CD Media Access Control Functions

4.2 CSMA/CD Media Access Control Method (MAC): Precise Specification

4.2.1 Introduction. A precise algorithmic definition is given in this section, providing procedural model for the CSMA/CD MAC process with a program in the computer language Pascal. See references [A2] and [A17] for resource material. Note whenever there is any apparent ambiguity concerning the definition of some aspect of the CSMA/CD MAC method, it is the Pascal procedural specification in 4.2.7 through 4.2.10 which should be consulted for the definitive statement. Sections 4.2.2 through 4.2.6 provide, in prose, a description of the access mechanism with the formal terminology to be used in the remaining subsections.

4.2.2 Overview of the Procedural Model. The functions of the CSMA/CD MAC method are presented below, modeled as a program written in the computer language Pascal. This procedural model is intended as the primary specification of the functions to be provided in any CSMA/CD MAC sublayer implementation. It is important to distinguish, however, between the model and a real implementation. The model is optimized for simplicity and clarity of presentation, while any realistic implementation shall place heavier emphasis on such constraints as efficiency and suitability to a particular implementation technology or computer architecture. In this context, several important properties of the procedural model shall be considered.

4.2.2.1 Ground Rules for the Procedural Model

(1) First, it shall be emphasized that *the description of the MAC sublayer in a computer language is in no way intended to imply that procedures shall be implemented as a program executed by a computer.* The implementation may consist of any appropriate technology including hardware, firmware, software, or any combination.

(2) Similarly, it shall be emphasized that it is the behavior of any MAC sublayer implementations that shall match the standard, not their internal structure. The internal details of the procedural model are useful only to the extent that they help specify that behavior clearly and precisely.

(3) The handling of incoming and outgoing frames is rather stylized in the procedural model, in the sense that frames are handled as single entities by most of the MAC sublayer and are only serial-

fixed for presentation to the Physical Layer. In reality, many implementations will instead handle frames serially on a bit, octet or word basis. This approach has not been reflected in the procedural model, since this only complicates the description of the functions without changing them in any way.

(4) The model consists of algorithms designed to be executed by a number of concurrent processes; these algorithms collectively implement the CSMA/CD procedure. The timing dependencies introduced by the need for concurrent activity are resolved in two ways:

- (a) *Processes Versus External Events.* It is assumed that the algorithms are executed "very fast" relative to external events, in the sense that a process never falls behind in its work and fails to respond to an external event in a timely manner. For example, when a frame is to be received, it is assumed that the Media Access procedure ReceiveFrame is always called well before the frame in question has started to arrive.
- (b) *Processes Versus Processes.* Among processes, no assumptions are made about relative speeds of execution. This means that each interaction between two processes shall be structured to work correctly independant of their respective speeds. Note, however, that the timing of interactions among processes is often, in part, an indirect reflection of the timing of external events, in which case appropriate timing assumptions may still be made.

It is intended that the concurrency in the model reflect the parallelism intrinsic to the task of implementing the LLC and MAC procedures, although the actual parallel structure of the implementations is likely to vary.

4.2.2.2 Use of Pascal in the Procedural Model. Several observations need to be made regarding the method with which Pascal is used for the model. Some of these observations are as follows:

(1) Some limitations of the language have been circumvented to simplify the specification:

- (a) The elements of the program (variables and procedures, for example) are presented in logical groupings, in top-down order. Certain Pascal ordering restrictions have thus been circumvented to improve readability.
- (b) The *process* and *cycle* constructs of Concurrent Pascal, a Pascal derivative, have been introduced to indicate the sites of autonomous concurrent activity. As used here, a process is simply a parameterless procedure that begins execution at "the beginning of time" rather than being invoked by a procedure call. A cycle statement represents the main body of a process and is executed repeatedly forever.
- (c) The lack of variable array bounds in the language has been circumvented by treating frames as if they are always of a single fixed size (which is never actually specified). The size of a frame depends on the size of its data field, hence the value of the "pseudo-constant" frameSize should be thought of as varying in the long-term, even though it is fixed for any given frame.
- (d) The use of a variant record to represent a frame (as fields and as bits) follows the spirit but not the letter of the Pascal Report, since it allows the underlying representation to be viewed as two different data types.

(2) The model makes no use of any explicit interprocess synchronization primitives. Instead, all interprocess interaction is done by way of carefully stylized manipulation of shared variables. For example, some variables are set by only one process and inspected by another process in such a manner that the net result is independent of their execution speeds. While such techniques are not generally suitable for the construction of large concurrent programs, they simplify the model and more nearly resemble the methods appropriate to the most likely implementation technologies (microcode, hardware state-machines, etc.)

4.2.2.3 Organization of the Procedural Model. The procedural model used here is based on five cooperating concurrent processes. Three are actually defined in the MAC sublayer. The remaining two processes are provided by the clients of the MAC sublayer (which may include the LLC sublayer) and utilize the interface operations provided by the MAC sublayer. The five processes are thus:

(1) Frame Transmitter Process
(2) Frame Receiver Process
(3) Bit Transmitter Process

(4) Bit Receiver Process
(5) Deference Process

This organization of the model is illustrated in Fig 4-3 and reflects the fact that the communication ‹ entire frames is initiated by the client of the MAC sublayer, while the timing of collision backoff and ‹ individual bit transfers is based on interactions between the MAC sublayer and the Physical-Laye dependent bit time.

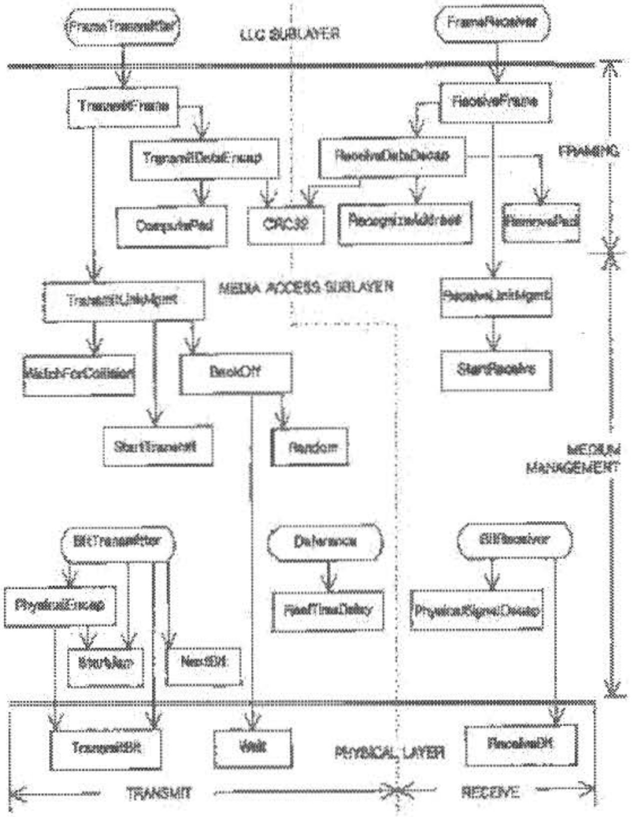
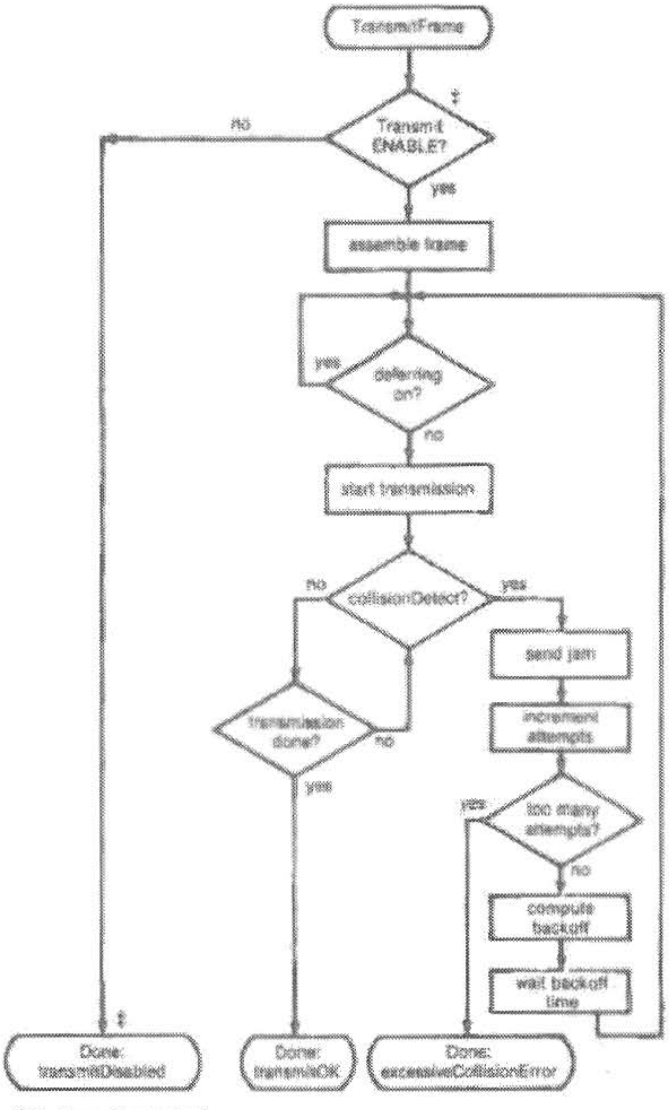


Fig 4-3
Relationship Among CSMA/CD Procedures

Figure 4-3 depicts the static structure of the procedural model, showing how the various processes an procedures interact by invoking each other. Figures 4-4 and 4-5 summarize the dynamic behavior of th model during transmission and reception, focusing on the steps that shall be performed, rather than th procedural structure that performs them. The usage of the shared state variables is not depicted in the fi ures, but is described in the comments and prose in the following sections.

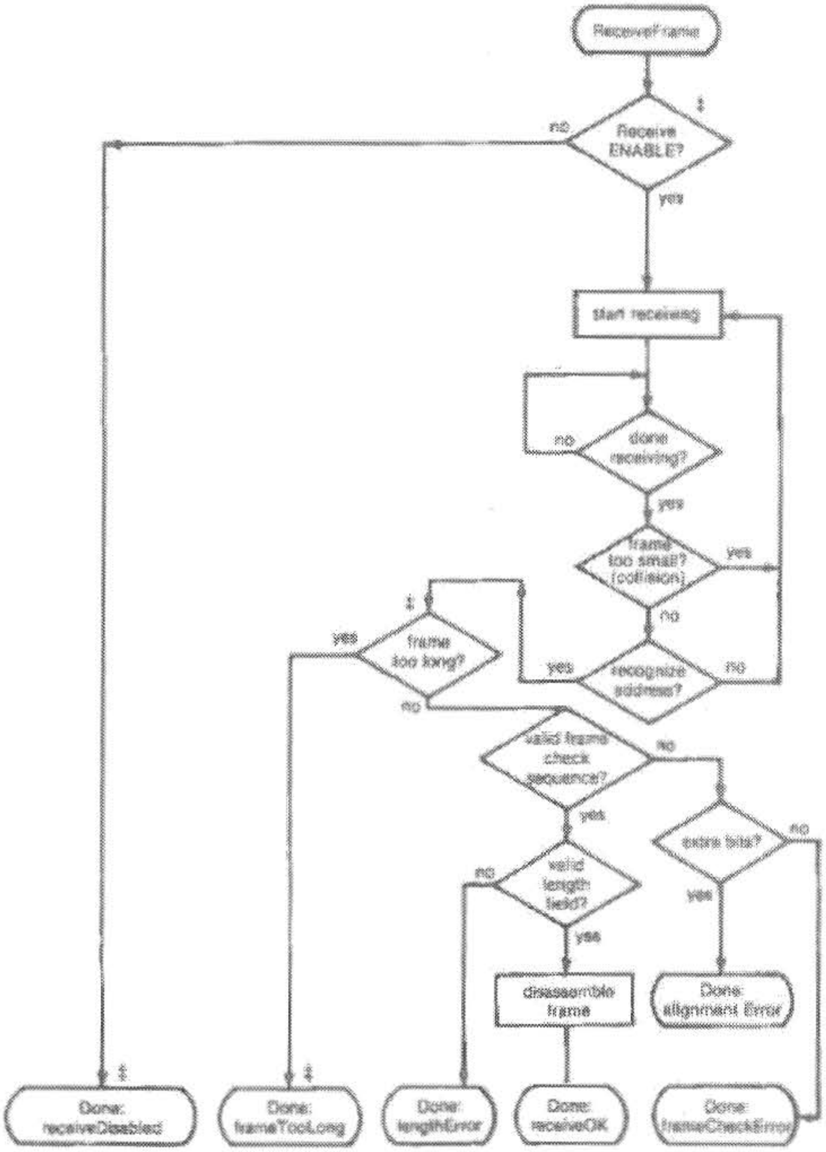
4.2.2.4 Layer Management Extensions to Procedural Model. In order to incorporate networ management functions, this Procedural Model has been expanded beyond that in ISO/IEC 8802-3 : 199 Network management functions have been incorporated in two ways. First, 4.2.7–4.2.10, 4.3.2, and Fig 4- have been modified and expanded to provide management services. Second, Layer Management procedure have been added as 5.2.4. Note that Pascal variables are shared between Sections 4 and 5. Within the Pa cal descriptions provided in Section 4, a "‡" in the left margin indicates a line that has been added to su port management services. These lines are only required if Layer Management is being implemente



‡ For Layer Management

(a) TransmitFrame

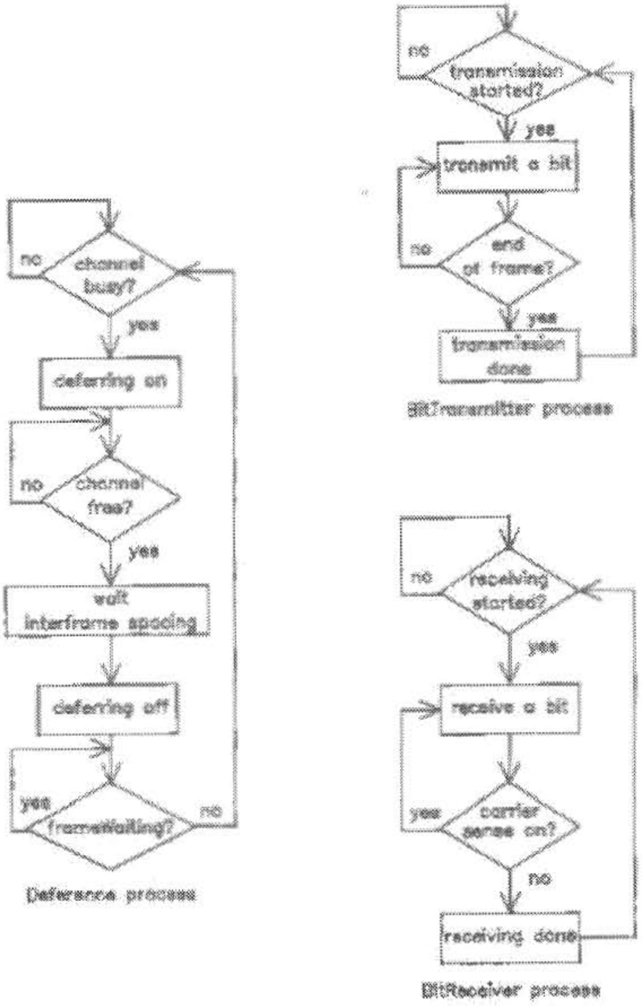
Fig 4-4
Control Flow Summary



‡ For Layer Management

(b) ReceiveFrame

Fig 4-4
Control Flow Summary



Deference process

BitTransmitter process

BitReceiver process

Fig 4-6
Control Flow: MAC Sublayer

These changes do not affect any aspect of the MAC behavior as observed at the LLC-MAC and MAC-PLS interfaces of ISO/IEC 8802-3 : 1990.

The Pascal procedural specification shall be consulted for the definitive statement when there is any apparent ambiguity concerning the definition of some aspect of the CSMA/CD MAC access method.

4.2.3 Frame Transmission Model. Frame transmission includes data encapsulation and Media Access management aspects:

(1) Transmit Data Encapsulation includes the assembly of the outgoing frame (from the values provided by the LLC sublayer) and frame check sequence generation.
(2) Transmit Media Access Management includes carrier deference, interframe spacing, collision detection and enforcement, and collision backoff and retransmission.

4.2.3.1 Transmit Data Encapsulation

4.2.3.1.1 Frame Assembly. The fields of the CSMA/CD MAC frame are set to the values provided by the LLC sublayer as arguments to the TransmitFrame operation (see 4.3) with the exception of the padding necessary to enforce the minimum framesize and the frame check sequence that is set to the CRC value generated by the MAC sublayer.

4.2.3.1.2 Frame Check Sequence Generation. The CRC value defined in 3.8 is generated and inserted in the frame check sequence field, following the fields supplied by the LLC sublayer.

4.2.3.2 Transmit Media Access Management

4.2.3.2.1 Carrier Deference. Even when it has nothing to transmit, the CSMA/CD MAC sublayer monitors the physical medium for traffic by watching the carrierSense signal provided by the PLS. Whenever the medium is busy, the CSMA/CD MAC sublayer defers to the passing frame by delaying any pending transmission of its own. After the last bit of the passing frame (that is, when carrierSense changes from true to false), the CSMA/CD MAC sublayer continues to defer for a proper interFrameSpacing (see 4.2.3.2.2).

If, at the end of the interFrameSpacing, a frame is waiting to be transmitted, transmission is initiated independent of the value of carrierSense. When transmission has completed (or immediately, if there was nothing to transmit) the CSMA/CD MAC sublayer resumes its original monitoring of carrierSense.

When a frame is submitted by the LLC sublayer for transmission, the transmission is initiated as soon as possible, but in conformance with the rules of deference stated above.

NOTE: It is possible for the PLS carrier sense indication to fail to be asserted briefly during a collision on the media. If the Deference process simply times the interFrame gap based on this indication it is possible for a short interFrame gap to be generated, leading to a potential reception failure of a subsequent frame. To enhance system robustness the following optional measures, as specified in 4.2.8, are recommended when interFrame SpacingPart1 is other than zero:
(1) Upon completing a transmission, start timing the interpacket gap as soon as transmitting and carrierSense are both false.
(2) When timing an interFrame gap following reception, reset the interFrame gap timing if carrierSense becomes true during the first 2/3 of the interFrame gap timing interval. During the final 1/3 of the interval the timer shall not be reset to ensure fair access to the medium. An initial period shorter than 2/3 of the interval is permissible including zero.

4.2.3.2.2 Interframe Spacing. As defined in 4.2.3.2.1, the rules for deferring to passing frames ensure a minimum interframe spacing of interFrameSpacing seconds. This is intended to provide interframe recovery time for other CSMA/CD sublayers and for the physical medium.

Note that interFrameSpacing is the minimum value of the interframe spacing. If necessary for implementation reasons, a transmitting sublayer may use a larger value with a resulting decrease in its throughput. The larger value is determined by the parameters of the implementation, see 4.4.

4.2.3.2.3 Collision Handling. Once a CSMA/CD sublayer has finished deferring and has started transmission, it is still possible for it to experience contention for the medium. Collisions can occur until acquisition of the network has been accomplished through the deference of all other stations' CSMA/CD sublayers.

The dynamics of collision handling are largely determined by a single parameter called the slot time. This single parameter describes three important aspects of collision handling:

(1) It is an upper bound on the acquisition time of the medium.
(2) It is an upper bound on the length of a frame fragment generated by a collision.
(3) It is the scheduling quantum for retransmission.

To fulfill all three functions, the slot time shall be larger than the sum of the Physical Layer round-trip propagation time and the Media Access Layer maximum jam time. The slot time is determined by the parameters of the implementation, see 4.4.

4.2.3.2.4 Collision Detection and Enforcement. Collisions are detected by monitoring the collisionDetect signal provided by the Physical Layer. When a collision is detected during a frame transmission, the transmission is not terminated immediately. Instead, the transmission continues until additional bits specified by jamSize have been transmitted (counting from the time collisionDetect went on). This collision enforcement or jam guarantees that the duration of the collision is sufficient to ensure its detection by all transmitting stations on the network. The content of the jam is unspecified; it may be any fixed or variable pattern convenient to the Media Access implementation, however, the implementation shall not be intentionally designed to be the 32-bit CRC value corresponding to the (partial) frame transmitted prior to the jam.

4.2.3.2.5 Collision Backoff and Retransmission. When a transmission attempt has terminated due to a collision, it is retried by the transmitting CSMA/CD sublayer until either it is successful or a maximum number of attempts (attemptLimit) have been made and all have terminated due to collisions. Note that all attempts to transmit a given frame are completed before any subsequent outgoing frames are transmitted. The scheduling of the retransmissions is determined by a controlled randomization process called "truncated binary exponential backoff." At the end of enforcing a collision (jamming), the CSMA/CD sublayer delays before attempting to retransmit the frame. The delay is an integer multiple of slotTime. The number of slot times to delay before the nth retransmission attempt is chosen as a uniformly distributed random integer r in the range:

$0 \le r < 2^k$

where

$k = \min (n, 10)$

If all attemptLimit attempts fail, this event is reported as an error. Algorithms used to generate the integer r should be designed to minimize the correlation between the numbers generated by any two stations at any given time.

Note that the values given above define the most aggressive behavior that a station may exhibit in attempting to retransmit after a collision. In the course of implementing the retransmission scheduling procedure, a station may introduce extra delays that will degrade its own throughput, but in no case may a station's retransmission scheduling result in a lower average delay between retransmission attempts than the procedure defined above.

4.2.3.3 Minimum Frame Size. The CSMA/CD Media Access mechanism requires that a minimum frame length of minFrameSize bits be transmitted. If frameSize is less than minFrameSize, then the CSMA/CD MAC sublayer shall append extra bits in units of octets, after the end of the LLC data field but prior to calculating, and appending, the FCS. The number of extra bits shall be sufficient to ensure that the frame, from the DA field through the FCS field inclusive, is at least minFrameSize bits. The content of the pad is unspecified.

4.2.4 Frame Reception Model. CSMA/CD MAC sublayer frame reception includes both data decapsulation and Media Access management aspects:

(1) Receive Data Decapsulation comprises address recognition, frame check sequence validation, and frame disassembly to pass the fields of the received frame to the LLC sublayer.
(2) Receive Media Access Management comprises recognition of collision fragments from incoming frames and truncation of frames to octet boundaries.

4.2.4.1 Receive Data Decapsulation

4.2.4.1.1 Address Recognition. The CSMA/CD MAC sublayer is capable of recognizing individual and group addresses.

(1) *Individual Addresses.* The CSMA/CD MAC sublayer recognizes and accepts any frame whose DA field contains the individual address of the station.
(2) *Group Addresses.* The CSMA/CD MAC sublayer recognizes and accepts any frame whose DA field contains the Broadcast address.

The CSMA/CD MAC sublayer is capable of activating some number of group addresses as specified by higher layers. The CSMA/CD MAC sublayer recognizes and accepts any frame whose Destination Address field contains an active group address. An active group address may be deactivated.

4.2.4.1.2 Frame Check Sequence Validation. FCS validation is essentially identical to FCS generation. If the bits of the incoming frame (exclusive of the FCS field itself) do not generate a CRC value identical to the one received, an error has occurred and the frame is identified as invalid.

4.2.4.1.3 Frame Disassembly. Upon recognition of the Start Frame Delimiter at the end of the preamble sequence, the CSMA/CD MAC sublayer accepts the frame. If there are no errors, the frame is disassembled and the fields are passed to the LLC sublayer by way of the output parameters of the ReceiveFrame operation.

4.2.4.2 Receive Media Access Management

4.2.4.2.1 Framing. The CSMA/CD sublayer recognizes the boundaries of an incoming frame by monitoring the carrierSense signal provided by the PLS. There are two possible length errors that can occur, that indicate ill-framed data: the frame may be too long, or its length may not be an integer number of octets.

(1) *Maximum Frame Size.* The receiving CSMA/CD sublayer is not required to enforce the frame size limit, but it is allowed to truncate frames longer than maxFrameSize octets and report this event as an (implementation-dependent) error.
(2) *Integer Number of Octets in Frame.* Since the format of a valid frame specifies an integer number of octets, only a collision or an error can produce a frame with a length that is not an integer multiple of 8 bits. Complete frames (that is, not rejected as collision fragments; see 4.2.4.2.2) that do not contain an integer number of octets are truncated to the nearest octet boundary. If frame check sequence validation detects an error in such a frame, the status code alignmentError is reported.

4.2.4.2.2 Collision Filtering. The smallest valid frame shall be at least one slotTime in length. This determines the minFrameSize. Any frame containing less than minFrameSize bits is presumed to be a fragment resulting from a collision. Since occasional collisions are a normal part of the Media Access management procedure, the discarding of such a fragment is not reported as an error to the LLC sublayer.

4.2.5 Preamble Generation. In a LAN implementation, most of the Physical Layer components are allowed to provide valid output some number of bit times after being presented valid input signals. Thus it is necessary for a preamble to be sent before the start of data, to allow the PLS circuitry to reach its steady-state. Upon request by TransmitLinkMgmt to transmit the first bit of a new frame, PhysicalSignalEncap shall first transmit the preamble, a bit sequence used for physical medium stabilization and synchronization, followed by the Start Frame Delimiter. If, while transmitting the preamble, the PLS asserts the collision detect signal, any remaining preamble bits shall be sent. The preamble pattern is:

10101010 10101010 10101010 10101010 10101010 10101010 10101010

The bits are transmitted in order, from left to right. The nature of the pattern is such that, for Manchester encoding, it appears as a periodic waveform on the medium that enables bit synchronization. It should be noted that the preamble ends with a "0."

4.2.6 Start Frame Sequence. The PLS recognizes the presence of activity on the medium through the carrier sense signal. This is the first indication that the frame reception process should begin. Upon reception of the sequence 10101011 immediately following a latter part of the preamble pattern, PhysicalSignalDecap shall begin passing successive bits to ReceiveLinkMgmt for passing to the LLC sublayer.

4.2.7 Global Declarations. This section provides detailed formal specifications for the CSMA/CD MAC sublayer. It is a specification of generic features and parameters to be used in systems implementing this media access method. Subsection 4.4 provides values for these sets of parameters for recommended implementations of this media access mechanism.

4.2.7.1 Common Constants and Types. The following declarations of constants and types are used by the frame transmission and reception sections of each CSMA/CD sublayer:

```
const
    addressSize = ... ; {16 or 48 bits in compliance with 3.2.3}
    lengthSize = 16; {in bits}
    LLCdataSize = ...; {LLC Data, see 4.2.2.2, (1)(c)}
    padSize = ...; {in bits, = max (0, minFrameSize – (2 × addressSize + lengthSize + LLCdataSize +
                    crcSize))}.
    dataSize = ...; {= LLCdataSize + padSize}
    crcSize = 32; {32 bit CRC = 4 octets}
    frameSize = ...; {= 2 × addressSize + lengthSize + dataSize + crcSize, see 4.2.2.2(1)}
    minFrameSize = ... ; {in bits, implementation-dependent, see 4.4}
    slotTime = ... ; {unit of time for collision handling, implementation-dependent, see 4.4}
    preambleSize = ... ; {in bits, physical-medium-dependent}
    sfdSize = 8; {8 bit start frame delimiter}
    headerSize = ...; {sum of preambleSize and sfdSize}
type
    Bit = 0..1;
    AddressValue = array [1..addressSize] of Bit;
    LengthValue = array [1..lengthSize] of Bit;
    DataValue = array [1..dataSize] of Bit;
    CRCValue = array [1..crcSize] of Bit;
    PreambleValue = array [1..preambleSize] of Bit;
    SfdValue = array [1..sfdSize] of Bit;
    ViewPoint = (fields, bits); {Two ways to view the contents of a frame}
    HeaderViewPoint = (headerFields, headerBits);
    Frame = record {Format of Media Access frame}
        case view: ViewPoint of
            fields: (
                destinationField: AddressValue;
                sourceField: AddressValue;
                lengthField: LengthValue;
                dataField: DataValue;
                fcsField: CRCValue);
            bits: (contents: array [1..frameSize] of Bit)
        end; {Frame}
    Header = record {Format of preamble and start frame delimiter}
        case headerView : HeaderViewPoint of
            headerFields : (
                preamble : PreambleValue;
                sfd : SfdValue);
            headerBits : (
                headerContents : array [1..headerSize] of Bit)
        end; {defines header for MAC frame}
```

4.2.7.2 Transmit State Variables. The following items are specific to frame transmission. (See also 4.4.)

```
const
    interFrameSpacing = ... ; {minimum time between frames}
    interFrameSpacingPart1= ...;{duration of first portion of interFrame timing. In range 0 up to 2/3
                    interFrameSpacing}
    interFrameSpacingPart2= ...;{duration of remainder of interFrame timing. Equal to
                    interFrameSpacing – interFrameSpacingPart1}
    attemptLimit = ... ; {Max number of times to attempt transmission}
    backOffLimit = ... ; {Limit on number of times to back off}
    jamSize = ... ; {in bits: the value depends upon medium and collision detect implementation}

var
    outgoingFrame: Frame; {The frame to be transmitted}
    outgoingHeader: Header;
    currentTransmitBit, lastTransmitBit: 1..frameSize;
    {Positions of current and last outgoing bits in outgoingFrame}
    lastHeaderBit: 1..headerSize;
    deferring: Boolean; {Implies any pending transmission must wait for the medium to clear}
    frameWaiting: Boolean; {Indicates that outgoingFrame is deferring}
    attempts: 0..attemptLimit; {Number of transmission attempts on outgoingFrame}
    newCollision: Boolean; {Indicates that a collision has occurred but has not yet been jammed}
    transmitSucceeding: Boolean; {Running indicator of whether transmission is succeeding}
```

4.2.7.3 Receive State Variables. The following items are specific to frame reception. (See also 4.4.)

```
var
    incomingFrame: Frame; {The frame being received}
    currentReceiveBit: 1..frameSize; {Position of current bit in incomingFrame}
    receiving: Boolean; {Indicates that a frame reception is in progress}
    excessBits: 0..7; {Count of excess trailing bits beyond octet boundary}
    receiveSucceeding: Boolean; {Running indicator of whether reception is succeeding}
    validLength: Boolean; {Indicator of whether received frame has a length error}
    exceedsMaxLength: Boolean; {Indicator of whether received frame has a length longer than the
                    maximum permitted length}
```

4.2.7.4 Summary of Interlayer Interfaces

(1) The interface to the LLC sublayer, defined in 4.3.2, is summarized below:

```
  type
‡     TransmitStatus = (transmitDisabled, transmitOK, excessiveCollisionError);
                          {Result of TransmitFrame operation}
‡     ReceiveStatus = (receiveDisabled, receiveOK, frameTooLong, frameCheckError, lengthError,
                        alignmentError); {Result of ReceiveFrame operation}

  function TransmitFrame (
      destinationParam: AddressValue;
      sourceParam: AddressValue;
      lengthParam: LengthValue;
      dataParam: DataValue): TransmitStatus; {Transmits one frame}

  function ReceiveFrame (
      var destinationParam: AddressValue;
      var sourceParam: AddressValue;
      var lengthParam: LengthValue;
```

```
        var dataParam: DataValue): ReceiveStatus; {Receives one frame}
```

(2) The interface to the Physical Layer, defined in 4.3.3, is summarized below:

```
var
    carrierSense: Boolean; {Indicates incoming bits}
    transmitting: Boolean; {Indicates outgoing bits}
    wasTransmitting: Boolean; {Indicates transmission in progress or just completed}
    collisionDetect: Boolean; {Indicates medium contention}
procedure TransmitBit (bitParam: Bit); {Transmits one bit}
function ReceiveBit: Bit; {Receives one bit}
procedure Wait (bitTimes: integer); {Waits for indicated number of bit-times}
```

4.2.7.5 State Variable Initialization. The procedure Initialize must be run when the MAC sublayer begins operation, before any of the processes begin execution. Initialize sets certain crucial shared state variables to their initial values. (All other global variables are appropriately reinitialized before each use.) Initialize then waits for the medium to be idle, and starts operation of the various processes.

If Layer Management is implemented, the Initialize procedure shall only be called as the result of the initializeMAC action (5.2.2.2.1).

```
procedure Initialize;
begin
    frameWaiting := false;
    deferring := false;
    newCollision := false;
    transmitting := false; {In interface to Physical Layer; see below}
    receiving := false;
    while carrierSense do nothing;
    {Start execution of all processes}
end; {Initialize}
```

4.2.8 Frame Transmission. The algorithms in this section define MAC sublayer frame transmission. The function TransmitFrame implements the frame transmission operation provided to the LLC sublayer:

```
function TransmitFrame (
    destinationParam: AddressValue;
    sourceParam: AddressValue;
    lengthParam: LengthValue;
    dataParam: DataValue): TransmitStatus;
procedure TransmitDataEncap; ... {nested procedure; see body below}
begin
    if transmitEnabled then
    begin
        TransmitDataEncap;
        TransmitFrame := TransmitLinkMgmt
    end
    else TransmitFrame := transmitDisabled
end; {TransmitFrame}
```

If transmission is enabled, TransmitFrame calls the internal procedure TransmitDataEncap to construct the frame. Next, TransmitLinkMgmt is called to perform the actual transmission. The TransmitStatus returned indicates the success or failure of the transmission attempt.

TransmitDataEncap builds the frame and places the 32-bit CRC in the frame check sequence field:

```
procedure TransmitDataEncap;
begin
    with outgoingFrame do
```

```
        begin {assemble frame}
            view := fields;
            destinationField := destinationParam;
            sourceField := sourceParam;
            lengthField := lengthParam;
            dataField := ComputePad (lengthparam, dataParam);
            fcsField := CRC32(outgoingFrame);
            view := bits
        end {assemble frame}
        with outgoingHeader do
        begin
            headerView := headerFields;
            preamble := ...; {* '1010...10,' LSB to MSB*}
            sfd := ...; {* '10101011,' LSB to MSB*}
            headerView := headerBits
        end
    end; {TransmitDataEncap}
```

ComputePad appends an array of arbitrary bits to the LLCdataField to pad the frame to the minimum frame size.

```
    function ComputePad(
        var lengthParam:LengthValue
        var dataParam:DataValue) :DataValue;
    begin
        ComputePad := {Append an array of size padSize of arbitrary bits to the LLCdataField}
    end;{ComputePadParam}
```

TransmitLinkMgmt attempts to transmit the frame, deferring first to any passing traffic. If a collision occurs, transmission is terminated properly and retransmission is scheduled following a suitable backoff interval:

```
    function TransmitLinkMgmt: TransmitStatus;
    begin
        attempts := 0; transmitSucceeding := false;
        lateCollisionCount := 0;
        deferred := false; {initialize}
        excessDefer := false;
        while(attempts < attemptLimit) and (not transmitSucceeding)do
        begin {loop}
            if attempts > 0 then BackOff;
            frameWaiting := true;
            lateCollisionError := false;
‡           while deferring do {defer to passing frame, if any}
            begin
‡               nothing;
                deferred := true;
            end;
            frameWaiting := false;
            StartTransmit;
            while transmitting do WatchForCollision;
            if lateCollisionError then lateCollisionCount := lateCollisionCount + 1;
            attempts := attempts+1
        end; {loop}
        if transmitSucceeding then TransmitLinkMgmt := transmitOK
        else TransmitLinkMgmt := excessiveCollisionError;
        LayerMgmtTransmitCounters; {update transmit and transmit error counters in 5.2.4.2}
        end; {TransmitLinkMgmt}
```

Each time a frame transmission attempt is initiated, StartTransmit is called to alert the BitTransmitter process that bit transmission should begin:

```
procedure StartTransmit;
begin
    currentTransmitBit := 1;
    lastTransmitBit := frameSize;
    transmitSucceeding := true;
    transmitting := true;
    lastHeaderBit := headerSize
end; {StartTransmit}
```

Once frame transmission has been initiated, TransmitLinkMgmt monitors the medium for contention by repeatedly calling WatchForCollision:

```
procedure WatchForCollision;
begin
    if transmitSucceeding and collisionDetect then
    begin
        if currentTransmitBit > (minFrameSize – headerSize) then
            lateCollisionError := true;
        newCollision := true;
        transmitSucceeding := false
    end
end; {WatchForCollision}
```

WatchForCollision, upon detecting a collision, updates *newCollision* to ensure proper jamming by the BitTransmitter process. The current transmit bit number is checked to see if this is a late collision. If the collision occurs later than a collision window of 512 bit times into the packet, it is considered as evidence of a late collision. The point at which the collision is received is determined by the network media propagation time and the delay time through a station and, as such, is implementation-dependent (see 4.1.2.2). An implementation may optionally elect to end retransmission attempts after a late collision is detected.

After transmission of the jam has been completed, if TransmitLinkMgmt determines that another attempt should be made, BackOff is called to schedule the next attempt to retransmit the frame.

```
var maxBackOff: 2..1024; {Working variable of BackOff}
procedure BackOff;
begin
    if attempts = 1 then maxBackOff := 2
    else if attempts ≤ backOffLimit
    then maxBackOff := maxBackOff × 2;
    Wait(slotTime × Random(0, maxBackOff))
end; {BackOff}

function Random (low, high: integer): integer;
begin
    Random := ...{uniformly distributed random integer r such that low ≤ r < high}
end; {Random}
```

BackOff performs the truncated binary exponential backoff computation and then waits for the selected multiple of the slot time.

The Deference process *runs* asynchronously to continuously compute the proper value for the variable deferring.

```
process Deference;
    begin
        cycle{main loop}
```

```
          while not carrierSense do nothing; {watch for carrier to appear}
          deferring := true; {delay start of new transmissions}
          wasTransmitting := transmitting;
          while carrierSense or transmitting then
             wasTransmitting := wasTransmitting or transmitting;
          if wasTransmitting do
             begin
                  StartRealTimeDelay; {time out first part interframe gap}
                  while RealTimeDelay(interFrameSpacingPart1) do nothing
             end
          else
             begin
                  StartRealTimeDelay;
                  repeat
                     while carrierSense do StartRealTimeDelay
                  until not RealTimeDelay(interFrameSpacingPart1)
             end;
          StartRealTimeDelay; {time out second part interframe gap}
          while RealTimeDelay(interFrameSpacingPart2) do nothing;
          deferring := false; {allow new transmissions to proceed}
          while frameWaiting do nothing; {allow waiting transmission if any}
       end {main loop}
    end; {Deference}

    procedure StartRealTimeDelay
       begin
          {reset the realtime timer and start it timing}
       end; {StartRealTimeDelay}

    function RealTimeDelay (usec:real): Boolean;
       begin
          {return the value true if the specified number of microseconds have
          not elapsed since the most recent invocation of StartRealTimeDelay,
          otherwise return the value false}
       end; {RealTimeDelay}
```

The BitTransmitter process runs asynchronously, transmitting bits at a rate determined by the Physical Layer's TransmitBit operation:

```
    process BitTransmitter;
    begin
       cycle {outer loop}
          if transmitting then
          begin {inner loop}
             PhysicalSignalEncap; {Send preamble and start of frame delimiter}
             while transmitting do
             begin
                TransmitBit(outgoingFrame[currentTransmitBit]); {send next bit to Physical Layer}
                if newCollision then StartJam else NextBit
             end;
          end; {inner loop}
       end; {outer loop}
    end; {BitTransmitter}
    procedure PhysicalSignalEncap;
    begin
       while currentTransmitBit ≤ lastHeaderBit do
       begin
```

```
        TransmitBit(outgoingHeader[currentTransmitBit]); {transmit header one bit at a time}
        currentTransmitBit := currentTransmitBit + 1;
    end
    if newCollision then StartJam else
    currentTransmitBit := 1
end; {PhysicalSignalEncap}

procedure NextBit;
begin
    currentTransmitBit := currentTransmitBit + 1;
    transmitting := (currentTransmitBit ≤ lastTransmitBit)
end; {NextBit}

procedure StartJam;
begin
    currentTransmitBit := 1;
    lastTransmitBit := jamSize;
    newCollision := false
end; {StartJam}
```

BitTransmitter, upon detecting a new collision, immediately enforces it by calling startJam to initiate the transmission of the jam. The jam should contain a sufficient number of bits of arbitrary data so that it is assured that both communicating stations detect the collision. (StartJam uses the first set of bits of the frame up to jamSize, merely to simplify this program.)

4.2.9 Frame Reception. The algorithms in this section define CSMA/CD Media Access sublayer frame reception.

The procedure ReceiveFrame implements the frame reception operation provided to the LLC sublayer:

```
function ReceiveFrame (
        var destinationParam: AddressValue;
        var sourceParam: AddressValue;
        var lengthParam: LengthValue;
        var dataParam: DataValue): ReceiveStatus;
    function ReceiveDataDecap: ReceiveStatus; ... {nested function; see body below}
begin
    if receiveEnabled then
    repeat
        ReceiveLinkMgmt;
        ReceiveFrame := ReceiveDataDecap;
    until receiveSucceeding
    else
        ReceiveFrame := receiveDisabled
; {ReceiveFrame}
```

If enabled, ReceiveFrame calls ReceiveLinkMgmt to receive the next valid frame, and then calls the internal procedure ReceiveDataDecap to return the frame's fields to the LLC sublayer if the frame's address indicates that it should do so. The returned ReceiveStatus indicates the presence or absence of detected transmission errors in the frame.

```
    function ReceiveDataDecap: ReceiveStatus;
‡       var status: ReceiveStatus; {holds receive status information}
    begin
‡       with incomingFrame do
‡       begin
‡           view := fields;
            receiveSucceeding := RecognizeAddress (incomingFrame, destinationField);
```

```
              receiveSucceeding := LayerMgmtRecognizeAddress (destinationField);
  ‡           if receiveSucceeding then
              begin {disassemble frame}
                 destinationParam := destinationField;
                 sourceParam := sourceField;
                 lengthParam := lengthField;
                 dataParam := RemovePad (lengthField, dataField);
                 exceedsMaxLength := ...; {check to determine if receive frame size exceeds the maximum
                                           permitted frame size (maxFrameSize)}
                 if exceedsMaxLength then status := frameTooLong;
                 else
                 if fcsField = CRC32 (incomingFrame) then
                 begin
  ‡                 if validLength then status := receiveOK
  ‡                 else status := lengthError
                 end
                 else
                 begin
  ‡                 if excessBits = 0 then status := frameCheckError
  ‡                 else status := alignmentError;
                 end;
                 LayerMgmt:ReceiveCounters(status);
                    {update receive and receive error counters in 5.2.4.3}
                 view := bits
              end {disassemble frame}
  ‡        end {with incomingFrame}
  ‡        ReceiveDataDecap := status;
        end; {ReceiveDataDecap}

        function RecognizeAddress (address: AddressValue): Boolean;
        begin
           RecognizeAddress := ... {Returns true for the set of physical, broadcast, and multicast-group
                                    addresses corresponding to this station}
        end ;{RecognizeAddress}

        function RemovePad(
              var lengthParam:LengthValue
              var dataParam:DataValue):DataValue;
        begin
              validLength := {Check to determine if value represented by lengthParam matches received
                              LLCdataSize};
              if validLength then
                 RemovePad := {truncate the dataParam (when present) to value represented by lengthParam
                               (in octets) and return the result}
                 else
                 RemovePad := dataParam
        end; {RemovePad}
```

ReceiveLinkMgmt attempts repeatedly to receive the bits of a frame, discarding any fragments from collisions by comparing them to the minimum valid frame size:

```
        procedure ReceiveLinkMgmt;
        begin
           repeat
              StartReceive;
              while receiving do nothing; {wait for frame to finish arriving}
              excessBits := frameSize mod 8;
```

```
            frameSize := frameSize – excessBits; {truncate to octet boundary}
            receiveSucceeding := (frameSize ≥ minFrameSize); {reject collision fragments}
        until receiveSucceeding
    end; {ReceiveLinkMgmt}

    procedure StartReceive;
    begin
        currentReceiveBit := 1;
        receiving := true
    end; {StartReceive}
```

The BitReceiver process runs asynchronously, receiving bits from the medium at the rate determined by the Physical Layer's ReceiveBit operation:

```
    process BitReceiver;
        var b: Bit;
    begin
        cycle {outer loop}
            while receiving do
            begin {inner loop}
            if currentReceiveBit = 1 then
                PhysicalSignalDecap; {Strip off the preamble and start frame delimiter}
                b := ReceiveBit; {Get next bit from physical Media Access}
                if carrierSense then
                begin{append bit to frame}
                    incomingFrame[currentReceiveBit] := b;
                    currentReceiveBit := currentReceiveBit + 1
                end; {append bit to frame}
                receiving := carrierSense
            end {inner loop}
            frameSize := currentReceiveBit – 1
        end {outer loop}
    end; {BitReceiver}

    procedure PhysicalSignalDecap;
    begin
        {Receive one bit at a time from physical medium until a valid sfd is detected, discard bits, and
         return}
    end; {PhysicalSignalDecap}
```

4.2.10 Common Procedures. The function CRC32 is used by both the transmit and receive algorithms to generate a 32 bit CRC value:

```
    function CRC32 (f: Frame): CRCValue;
    begin
        CRC32 := {The 32-bit CRC }
    end; {CRC32}
```

Purely to enhance readability, the following procedure is also defined:

```
    procedure nothing; begin end;
```

The idle state of a process (that is, while waiting for some event) is cast as repeated calls on this procedure.

4.3 Interfaces to/from Adjacent Layers

4.3.1 Overview. The purpose of this section is to provide precise definitions of the interfaces between the architectural layers defined in Section 1 in compliance with the Media Access Service Specification given in Section 2. In addition, the services required from the physical medium are defined.

The notation used here is the Pascal language, in keeping with the procedural nature of the precise MAC sublayer specification (see 4.2). Each interface is described as a set of procedures or shared variables, or both, that collectively provide the only valid interactions between layers. The accompanying text describes the meaning of each procedure or variable and points out any implicit interactions among them.

Note that the description of the interfaces in Pascal is a notational technique, and in no way implies that they can or should be implemented in software. This point is discussed more fully in 4.2, that provides complete Pascal declarations for the data types used in the remainder of this section. Note also that the "synchronous" (one frame at a time) nature of the frame transmission and reception operations is a property of the architectural interface between the LLC and MAC sublayers, and need not be reflected in the implementation interface between a station and its sublayer.

4.3.2 Services Provided by the MAC Sublayer. The services provided to the LLC sublayer by the MAC sublayer are transmission and reception of LLC frames. The interface through which the LLC sublayer uses the facilities of the MAC sublayer therefore consists of a pair of functions.

Functions:
TransmitFrame
ReceiveFrame

Each of these functions has the components of a LLC frame as its parameters (input or output), and returns a status code as its result. Note that the service_class defined in 2.3.1 is ignored by CSMA/CD MAC.

The LLC sublayer transmits a frame by invoking TransmitFrame:

```
function TransmitFrame (
        destinationParam: AddressValue;
        sourceParam: AddressValue;
        lengthParam: LengthValue;
        dataParam: DataValue): TransmitStatus;
```

The TransmitFrame operation is synchronous. Its duration is the entire attempt to transmit the frame; when the operation completes, transmission has either succeeded or failed, as indicated by the resulting status code:

```
    type TransmitStatus = (transmitOK, excessiveCollisionError);
‡   type TransmitStatus = (transmitDisabled, transmitOK, excessiveCollisionError);
```

The transmitDisabled status code indicates that the transmitter is not enabled. Successful transmission is indicated by the status code transmittOK; the code excessiveCollisionError indicates that the transmission attempt was aborted due to the excessive collisions, because of heavy traffic or a network failure.

The LLC sublayer accepts incoming frames by invoking ReceiveFrame:

```
function ReceiveFrame (
        var destinationParam: AddressValue;
        var sourceParam: AddressValue;
        var length Param: LengthValue;
        var dataParam: DataValue): ReceiveStatus;
```

The ReceiveFrame operation is synchronous. The operation does not complete until a frame has been received. The fields of the frame are delivered via the output parameters with a status code:

```
type ReceiveStatus = (receiveOK, lengthError, frameCheckError, alignmentError);
```

‡ *type* Receive Status = (receiveDisabled, receive OK, frameTooLong, frameCheck Error, length Error, alignmentError);

The receiveDisabled status indicates that the receiver is not enabled. Successful reception is indicated by the status code receiveOK. The frameTooLong error indicates that a frame was received whose frameSize was beyond the maximum allowable frame size. The code frameCheckError indicates that the frame received was damaged by a transmission error. The lengthError indicates the lengthParam value was inconsistent with the frameSize of the received frame. The code alignmentError indicates that the frame received was damaged, and that in addition, its length was not an integer number of octets.

4.3.3 Services Required from the Physical Layer. The interface through which the CSMA/CD MAC sublayer uses the facilities of the Physical Layer consists of a function, a pair of procedures and three Boolean variables.

Function:	*Procedures:*	*Variables:*
ReceiveBit	TransmitBit	collisionDetect
	Wait	carrierSense
		transmitting

During transmission, the contents of an outgoing frame are passed from the MAC sublayer to the Physical Layer by way of repeated use of the TransmitBit operation:

procedure TransmitBit (bitParam: Bit);

Each invocation of TransmitBit passes one new bit of the outgoing frame to the Physical Layer. The TransmitBit operation is synchronous. The duration of the operation is the entire transmission of the bit. The operation completes, when the Physical Layer is ready to accept the next bit and it transfers control to the MAC sublayer.

The overall event of data being transmitted is signaled to the Physical Layer by way of the variable transmitting:

var transmitting: Boolean;

Before sending the first bit of a frame, the MAC sublayer sets transmitting to true, to inform the Physical Media Access that a stream of bits will be presented via the TransmitBit operation. After the last bit of the frame has been presented, the MAC sublayer sets transmitting to false to indicate the end of the frame.

The presence of a collision in the physical medium is signaled to the MAC sublayer by the variable collisionDetect:

var collisionDetect: Boolean;

The collisionDetect signal remains true during the duration of the collision.

NOTE: Since an entire collision may occur during preamble generation, the MAC sublayer shall handle this possibility by monitoring collisionDetect concurrently with its transmission of outgoing bits. See 4.2 for details.

The collisionDetect signal is generated only during transmission and is never true at any other time; in particular, it cannot be used during frame reception to detect collisions between overlapping transmissions from two or more other stations.

During reception, the contents of an incoming frame are retrieved from the Physical Layer by the MAC sublayer via repeated use of the ReceiveBit operation:

function ReceiveBit: Bit;

Each invocation of ReceiveBit retrieves one new bit of the incoming frame from the Physical Layer. The ReceiveBit operation is synchronous. Its duration is the entire reception of a single bit. Upon receiving a bit, the MAC sublayer shall immediately request the next bit until all bits of the frame have been received. (See 4.2 for details.)

The overall event of data being received is signaled to the MAC sublayer by the variable carrierSense:

var carrierSense: Boolean;

When the Physical Layer sets carrierSense to true, the MAC sublayer shall immediately begin retrieving the incoming bits by the ReceiveBit operation. When carrierSense subsequently becomes false, the MAC sublayer can begin processing the received bits as a completed frame. Note that the true/false transitions of carrierSense are not defined to be precisely synchronized with the beginning and end of the frame, but may precede the beginning and lag the end, respectively. If an invocation of ReceiveBit is pending when carrierSense becomes false, ReceiveBit returns an undefined value, which should be discarded by the MAC sublayer. (See 4.2 for details.)

The MAC sublayer shall also monitor the value of carrierSense to defer its own transmissions when the medium is busy.

The Physical Layer also provides the procedure Wait:

procedure Wait (bitTimes: integer);

This procedure waits for the specified number of bit times. This allows the MAC sublayer to measure time intervals in units of the (physical-medium-dependent) bit time.

Another important property of the Physical Layer, which is an implicit part of the interface presented to the MAC sublayer, is the round-trip propagation time of the physical medium. Its value represents the maximum time required for a signal to propagate from one end of the network to the other, and for a collision to propagate back. The round-trip propagation time is primarily (but not entirely) a function of the physical size of the network. The round-trip propagation time of the Physical Layer is defined in 4.4 for a selection of physical media.

4.4 Specific Implementations

4.4.1 Compatibility Overview. To provide total compatibility at all levels of the standard, it is required that each network component implementing the CSMA/CD MAC sublayer procedure adheres rigidly to these specifications. The information provided in 4.4.2.1 below provides design parameters for a specific implementation of this access method. Variations from these values result in a system implementation that violates the standard.

4.4.2 Allowable Implementations

4.4.2.1 Parameterized Values. The following table identifies the parameter values that shall be used in the 10 Mb/s implementation of a CSMA/CD MAC procedure. The primary assumptions are that the physical medium is a baseband coaxial cable with properties given in the Physical Layer section(s) of this standard.

Parameters	Values
slotTime	512 bit times
interFrameGap	9.6 μs
attemptLimit	16
backoffLimit	10
jamSize	32 bits
maxFrameSize	1518 octets
minFrameSize	512 bits (64 octets)
addressSize	48 bits

WARNING: Any deviation from the above plans specified for a 10 Mb/s system may affect proper operation of the LAN.

See also DTE Deference Delay in 12.9.2.

4.4.2.2 Parameterized Values. The following parameter values shall be used for 1BASE5 implementations:

Parameters	Values
slotTime	512 bit times
interFrameGap	96 μs
attemptLimit	16
backoffLimit	10
jamSize	32 bits
maxFrameSize	1518 octets
minFrameSize	512 bits (64 octets)
addressSize	48 bits

See also DTE Deference Delay in 12.9.2.

WARNING: Any deviation from the above specified values may affect proper operation of the network.

5. Layer Management

5.1 Introduction. This section provides the Layer Management specification for networks based on the CSMA/CD access method. It defines facilities comprised of a set of statistics and actions needed to provide Layer Management services. The information in this chapter should be used in conjunction with the Procedural Model defined in 4.2.7–4.2.10. The Procedural Model provides a formal description of the relationship between the CSMA/CD Layer Entities and the Layer Management facilities.

This Layer Management specification has been developed in accordance with the OSI management architecture as specified in the ISO Management Framework document, ISO/IEC 7498-4:1989 [20]. It is independent of any particular management application or management protocol.

The management facilities defined in this standard may be accessed both locally and remotely. Thus, the Layer Management specification provides facilities that can be accessed from within a station or can be accessed remotely by means of a peer management protocol operating between application entities.

In CSMA/CD no peer management facilities are necessary for initiating or terminating normal protocol operations or for handling abnormal protocol conditions. The monitoring of these activities is done by the carrier sense and collision detection mechanisms. Since these activities are necessary for normal operation of the protocol, they are not considered to be a function of Layer Management and are therefore not discussed in this section.

At this time, this standard does not include management facilities that address the unique features of repeaters or of 10BROAD36 broadband MAUs.

5.1.1 Systems Management Overview. Within the ISO Open Systems Interconnection (OSI) architecture, the need to handle the special problems of initializing, terminating, and monitoring on-going activities and assisting in their harmonious operations, as well as handling abnormal conditions, is recognized. These needs are collectively addressed by the systems management component of the OSI architecture.

The systems management component may conceptually be subdivided into a System Management Application Entity (SMAE) and Layer Management Entities (LMEs). In addition, a Management Protocol is required for the exchange of information between systems on a network. This Layer Management standard is independent of any particular Management Protocol.

The SMAE is concerned with the management of resources and their status across all layers of the OSI architecture. The System Management Application facilities have been grouped into five entities: Configuration, Fault, Performance, Security, and Accounting.

Configuration and Name Management is concerned with the initialization, normal operation, and close-down of communication facilities. It is also concerned with the naming of these resources and their interrelationship as part of a communication system. Fault Management is concerned with detection, isolation, and correction of abnormal operations. Performance Management is concerned with evaluating the behavior and the effectiveness of the communication activities. Security Management is concerned with monitoring the integrity and controlling access to the communication facilities. Accounting Management is concerned with enabling charges to be established and cost to be assigned and providing information on tariffs for the use of communication resources.

This Layer Management standard, in conjunction with the Layer Management standards of other layers, provides the means for the SMAE to perform its various functions. Layer Management collects information needed by the SMAE from the MAC and Physical Layers. It also provides a means for the SMAE to exercise control over those layers. This Layer Management standard is independent of any specific SMAE.

The SMAE has a conceptual interface to an LME concerned with the actual monitoring and control of a specific layer. The LME interfaces directly only with the SMAE, to whom it provides Layer Management facilities.

Strictly, only those management activities that imply actual exchanges of information between peer entities are pertinent to OSI architecture. Therefore, only the protocols needed to conduct such exchanges are candidates for standardization. As a practical matter, however, the specification of the Layer Management facilities provided across the conceptual Layer Management Interface (LMI) between the LME and SMAE is required. Standardization of these facilities will make practical the use of higher layer protocols for the control and maintenance of LANs.

There are two interfaces that relate the various management entities. These are as follows:

(1) The conceptual Layer Management Interface between the SMAE and LME.

(2) The normal layer service interface for peer-to-peer communication.

The relationship between the various management entities and the layer entities according to the ISO Model is shown in Fig 5-1.

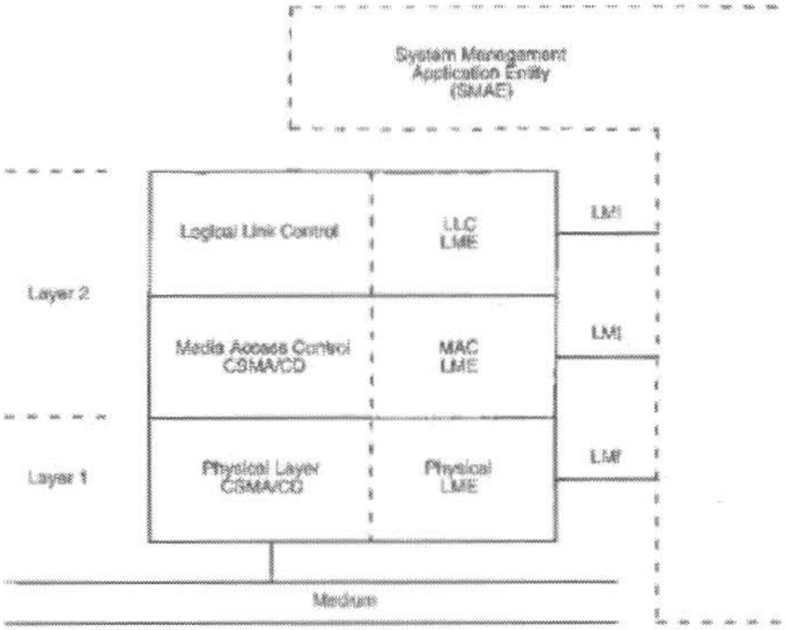


Fig 5-1
Relationship Between the Various Management Entities and Layer Entities According to the ISO Open Systems Interconnection (OSI) Reference Model

The conceptual LMI between the SMAE and the LME will be described in this standard in terms of the Layer Management facilities provided. It is particularly important that these facilities be defined because they may be indirectly requested on behalf of a remote SMAE. The use of this specification by other management mechanisms is not precluded.

5.1.2 Layer Management Model. The Layer Management facilities provided by the CSMA/CD MAC and Physical Layer LMEs, using the conceptual LMI, enable the SMAE to manipulate management counters and initiate actions within the layers. The LMI provides a means to monitor and control the facilities of the LMEs.

The CSMA/CD MAC/Physical Layer LMEs, in order to support the above facilities, offer a set of statistics and actions that constitute the conceptual LMI. The client of the LME (i.e., the SMAE) is thus able to read these statistics and to execute actions.

It is by executing these actions that the SMAE can cause certain desired effects on the MAC or Physical Layer Entities. The precise semantics of the relationship between the CSMA/CD Layer Entities and the Layer Management facilities are defined in 4.2.7–4.2.10 and in 5.2.4.

5.2 Management Facilities

5.2.1 Introduction. This section of the standard defines the Layer Management facilities for the IEEE 802.3 CSMA/CD MAC and Physical Layers. The intent of this standard is to furnish a management specification that can be used by the wide variety of different devices that may be attached to a network specified by ISO/IEC 8802-3. Thus, a comprehensive list of management facilities is provided.

The improper use of some of the facilities described in this section may cause serious disruption of the network. It should be noted that access to these facilities can only be obtained by means of the SMAE. To avoid duplication by each LME, and in accordance with ISO management architecture, any necessary security provisions should be provided by the SMAE. This can be in the form of specific SMAE security features or in the form of security features provided by the peer-to-peer communication facilities used by the SMAE

The statistics and actions are categorized into the three classifications defined as follows:

Mandatory—Shall be implemented.
Recommended—Should be implemented if possible.
Optional—May be implemented.

All counters defined in this specification are wraparound counters. Wraparound counters are those that automatically go from their maximum value (or final value) to zero and continue to operate. These unsigned counters do not provide for any explicit means to return them to their minimum (zero), i.e., reset. Because of their nature, wraparound counters should be read frequently enough to avoid loss of information.

5.2.2 MAC Sublayer Management Facilities. This section of the standard defines the Layer Management facilities specific to the MAC sublayer.

5.2.2.1 MAC Statistics. The statistics defined in this section are implemented by means of counters.

In the following definitions, the term "Read only" specifies that the object cannot be written to by the client of the LME.

Frame fragments are not included in any of the statistics in this section unless otherwise stated.

The Layer Management Model in 5.2.4 and the Pascal Procedural Model in 4.2.7–4.2.10 defines the semantics of these statistics in terms of the behavior of the MAC sublayer.

5.2.2.1.1 MAC Transmit Statistics Descriptions

(1) Number of framesTransmittedOK: Mandatory, Read only, 32 bit counter.
This contains a count of frames that are successfully transmitted. This counter is incremented when the TransmitStatus is reported as transmitOK. The update occurs in the LayerMgmtTransmitCounters procedure (5.2.4.2).

(2) Number of singleCollisionFrames: Mandatory, Read only, 32 bit counter.
This contains a count of frames that are involved in a single collision and are subsequently transmitted successfully. This counter is incremented when the result of a transmission is reported as transmitOK and the attempt value is 2. The update occurs in the LayerMgmtTransmitCounters procedure (5.2.4.2).

(3) Number of multipleCollisionFrames: Mandatory, Read only, 32 bit counter.
This contains a count of frames that are involved in more than one collision and are subsequently transmitted successfully. This counter is incremented when the TransmitStatus is reported as transmitOK and the value of the attempts variable is greater than 2 and less than or equal to attemptLimit. The update occurs in the LayerMgmtTransmitCounters procedure (5.2.4.2).

(4) Number of collisionFrames: Recommended, Read only, Array {1..attemptLimit – 1} of 32 bit counters.
This array provides a histogram of collision activity. The indices of this array
(1 to attemptLimit — 1) denote the number of collisions experienced in transmitting a frame. Each element of this array contains a counter that denotes the number of frames that have experienced a specific number of collisions. When the TransmitStatus is reported as transmitOK and the value of the attempts variable equals n, then collisionFrames[n–1] counter is incremented. The elements of this array are incremented in the LayerMgmtTransmitCounters procedure (5.2.4.2).

(5) Number of octetsTransmittedOK: Recommended, Read only, 32 bit counter.
This contains a count of data and padding octets of frames that are successfully transmitted. This counter is incremented when the TransmitStatus is reported as transmitOK. The update occurs in the LayerMgmtTransmitCounters procedure (5.2.4.2).

(6) Number of Frames with deferredTransmissions: Recommended, Read only, 32 bit counter.
This contains a count of frames whose transmission was delayed on its first attempt because the medium was busy. This counter is incremented when the Boolean variable deferred has been asserted by the TransmitLinkMgmt function (4.2.8). Frames involved in any collisions are not counted. The update occurs in the LayerMgmtTransmitCounters procedure (5.2.4.2).

(7) Number of multicastFramesTransmittedOK: Optional, Read only, 32 bit counter.
This contains a count of frames that are successfully transmitted, as indicated by the status value transmitOK, to a group destination address other than broadcast. The update occurs in the LayerMgmtTransmitCounters procedure (5.2.4.2).

(8) Number of broadcastFramesTransmittedOK: Optional, Read only, 32 bit counter.
This contains a count of the frames that were successfully transmitted as indicated by the TransmitStatus transmitOK, to the broadcast address. Frames transmitted to multicast addresses are not broadcast frames and are excluded. The update occurs in the LayerMgmtTransmitCounters procedure (5.2.4.2).

5.2.2.1.2 MAC Transmit Error Statistics Descriptions. This section defines the MAC sublayer transmission related error statistics.

(1) Number of lateCollision: Recommended Read only, 32 bit counter.
This contains a count of the times that a collision has been detected later than 512 bit times into the transmitted packet. A late collision is counted twice, i.e., both as a collision and as a lateCollision. This counter is incremented when the lateCollisionCount variable is nonzero. The update is incremented in the LayerMgmtTransmitCounters procedure (5.2.4.2).

(2) Number of frames aborted due to excessiveCollision: Recommended, Read only, 32 bit counter.
This contains a count of the frames that due to excessive collisions are not transmitted successfully. This counter is incremented when the value of the attempts variable equals attemptLimit during a transmission. The update occurs in the LayerMgmtTransmitCounters procedure (5.2.4.2).

(3) Number of frames lost due to internalMACTransmitError: Recommended, Read only, 32 bit counter.
This contains a count of frames that would otherwise be transmitted by the station, but could not be sent due to an internal MAC sublayer transmit error. If this counter is incremented, then none of the other counters in this section are incremented. The exact meaning and mechanism for incrementing this counter is implementation-dependent.

(4) Number of carrierSenseErrors: Recommended, Read only, 32 bit counter.
This contains a count of times that the carrierSense variable was not asserted or was deasserted during the transmission of a frame without collision (see 7.2.4.6). This counter is incremented when the carrierSenseFailure flag is true at the end of transmission. The update occurs in the LayerMgmtTransmitCounters procedure (5.2.4.2).

(5) Number of frames with excessiveDeferral: Optional, Read only, 32 bit counter.
This contains a count of frames that were deferred for an excessive period of time. This counter may only be incremented once per LLC transmission. This counter is incremented when the excessDefer flag is set. The update occurs in the LayerMgmtTransmitCounters procedure (5.2.4.2).

5.2.2.1.3 MAC Receive Statistics Descriptions

(1) Number of framesReceivedOK: Mandatory, Read only, 32 bit counter.
This contains a count of frames that are successfully received (receiveOK). This does not include frames received with frame-too-long, FCS, length or alignment errors, or frames lost due to internal MAC sublayer error. This counter is incremented when the ReceiveStatus is reported as receiveOK. The update occurs in the LayerMgmtReceiveCounters procedure (5.2.4.3).

(2) Number of octetsReceivedOK: Recommended, Read only, 32 bit counter.
This contains a count of data and padding octets in frames that are successfully received. This does not include octets in frames received with frame-too-long, FCS, length or alignment errors, or frames lost due to internal MAC sublayer error. This counter is incremented when the result of a

reception is reported as a receiveOK status. The update occurs in the LayerMgmtReceiveCounters procedure (5.2.4.3).

(3) Number of multicastFramesReceivedOK: Optional, Read only, 32 bit counter.
This contains a count of frames that are successfully received and are directed to an active non-broadcast group address. This does not include frames received with frame-too-long, FCS, length or alignment errors, or frames lost due to internal MAC sublayer error. This counter is incremented as indicated by the receiveOK status, and the value in the destinationField. The update occurs in the LayerMgmtReceiveCounters procedure (5.2.4.3).

(4) Number of broadcastFramesReceivedOK: Optional, Read only, 32 bit counter.
This contains a count of frames that are successfully received and are directed to the broadcast group address. This does not include frames received with frame-too-long, FCS, length or alignment errors, or frames lost due to internal MAC sublayer error. This counter is incremented as indicated by the receiveOK status, and the value in the destinationField. The update occurs in the LayerMgmtReceiveCounters procedure (5.2.4.3).

5.2.2.1.4 MAC Receive Error Statistics Descriptions. This section defines the MAC sublayer reception related error statistics. Note that a hierarchical order has been established such that when multiple error statuses can be associated with one frame, only one status is returned to the LLC. This hierarchy in descending order is as follows:

frameTooLong
alignmentError
frameCheckError
lengthError

The following counters are primarily incremented based on the status returned to the LLC, and therefore the hierarchical order of the counters is determined by the order of the status.

(1) Number of frames received with frameCheckSequenceErrors: Mandatory, Read only, 32 bit counter.
This contains a count of frames that are an integral number of octets in length and do not pass the FCS check. This counter is incremented when the ReceiveStatus is reported as frameCheckError. The update occurs in the LayerMgmtReceiveCounters procedure (5.2.4.3).

(2) Number of frames received with alignmentErrors: Mandatory, Read only, 32 bit counter.
This contains a count of frames that are not an integral number of octets in length and do not pass the FCS check. This counter is incremented when the ReceiveStatus is reported as alignmentError. The update occurs in the LayerMgmtReceiveCounters procedure (5.2.4.3).

(3) Number of frames lost due to internalMACReceiveError: Recommended, Read only, 32 bit counter.
This contains a count of frames that would otherwise be received by the station, but could not be accepted due to an internal MAC sublayer receive error. If this counter is incremented, then none of the other counters in this section are incremented. The exact meaning and mechanism for incrementing this counter is implementation-dependent.

(4) Number of frames received with inRangeLengthErrors: Optional, Read only, 32 bit counter.
This contains a count of frames with a length field value between the minimum unpadded LLC data size and the maximum allowed LLC data size, inclusive, that does not match the number of LLC data octets received. The counter also contains frames with a length field value less than the minimum unpadded LLC data size. The update occurs in the LayerMgmtReceiveCounters procedure (5.2.4.3).

(5) Number of frames received with outOfRangeLengthField: Optional, Read only, 32 bit counter.
This contains a count of frames with a length field value greater than the maximum allowed LLC data size. The update occurs in the LayerMgmtReceiveCounters procedure (5.2.4.3).

(6) Number of frames received with frameTooLongErrors: Optional, Read only, 32 bit counter.
This contains a count of frames that are received and exceed the maximum permitted frame size. This counter is incremented when the status of a frame reception is frameTooLong. The update occurs in the LayerMgmtReceiveCounters procedure (5.2.4.3).

5.2.2.2 MAC Actions. This subsection defines the actions offered by the MAC sublayer to the LME client.

These actions enable the LME client to influence the behavior of the MAC sublayer, i.e., to execute "actions" on the MAC sublayer for management purposes. Many of the following actions enable or disable some function; if either the enable or disable action is implemented, the corresponding disable or enable action must also be implemented. If the enable/disable action is supported, then its corresponding read action must also be supported.

In implementing any of the following actions, receptions and transmissions that are in progress are completed before the action takes effect.

The security considerations related to the following actions should be properly addressed by the SMAE.

The items in parenthesis in the descriptions are the procedures that are affected by these actions.

5.2.2.2.1 MAC Action Definitions

(1) initializeMAC: Mandatory
Call the Initialize procedure (4.2.7.5). This action also results in the initialization of the PLS.

(2) enablePromiscuousReceive: Recommended
Cause the LayerMgmtRecognizeAddress function to accept frames regardless of their destination address (LayerMgmtRecognizeAddress function).
Frames without errors received solely because this action is set are counted as frames received correctly; frames received in this mode that do contain errors update the appropriate error counters.

(3) disablePromiscuousReceive: Recommended
Cause the MAC sublayer to return to the normal operation of carrying out address recognition procedures for station, broadcast, and multicast group addresses (LayerMgmtRecognizeAddress function).

(4) readPromiscuousStatus: Recommended
Return true if promiscuous mode enabled, and false otherwise (LayerMgmtRecognizeAddress function).

(5) addGroupAddress: Recommended
Add the supplied multicast group address to the address recognition filter (RecognizedAddress function).

(6) deleteGroupAddress: Recommended
Delete the supplied multicast group address from the address recognition filter (RecognizeAddress function).

(7) readMulticastAddressList: Recommended
Return the current multicast address list.

(8) enableMacSublayer: Optional
Cause the MAC sublayer to enter the normal operational state at idle. The PLS is reset by this operation (see 7.2.2.2.1). This is accomplished by setting receiveEnabled and transmitEnabled to true.

(9) disableMacSublayer: Optional
Cause the MAC sublayer to end all transmit and receive operations, leaving it in a disabled state. This is accomplished by setting receiveEnabled and transmitEnabled to false.

(10) readMACEnableStatus: Optional
Return true if MAC sublayer is enabled, and false if disabled. This is accomplished by checking the values of the receiveEnabled and transmitEnabled variables.

(11) enableTransmit: Optional
Enable MAC sublayer frame transmission (TransmitFrame function). This is accomplished by setting transmitEnabled to true.

(12) disableTransmit: Optional
Inhibit the transmission of further frames by the MAC sublayer (TransmitFrame function). This is accomplished by setting transmitEnabled to false.

(13) readTransmitEnableStatus: Optional
Return true if transmission is enabled and false otherwise. This is accomplished by checking the value of the transmitEnabled variable.

(14) enableMulticastReceive: Optional
Cause the MAC sublayer to return to the normal operation of multicast frame reception.

(15) disableMulticastReceive: Optional
Inhibit the reception of further multicast frames by the MAC sublayer.

(16) readMulticastReceiveStatus: Optional
Return true if multicast receive is enabled, and false otherwise.

(17) modifyMACAddress: Optional
Change the MAC station address to the one supplied (RecognizeAddress function). Note that the supplied station address shall not have the group bit set and shall not be the null address.

(18) readMACAddress: Optional
Read the current MAC station address.

(19) executeSelftest: Optional
Execute a self test and report the results (success or failure). The mechanism employed to carry out the self test is not defined in this standard.

5.2.3 Physical Layer Management Facilities. This section of the standard defines the Layer Management facilities for the Physical Layer.

5.2.3.1 Physical Statistics. The statistics defined in this section are implemented by means of counters.

In the following definition, the term "Read only" specifies that the object cannot be written by the client of the LME.

Note that the carrierSenseFailed statistic is a statistic relating to the physical layer, but is listed and maintained in the MAC sublayer for ease of implementation.

5.2.3.1.1 Physical Statistics Descriptions

(1) Number of SQETestErrors: Recommended, Read only, 32[8] bit counter.
This contains a count of times that the SQE_TEST_ERROR was received. The SQE_TEST_ERROR is set in accordance with the rules for verification of the SQE detection mechanism in the PLS Carrier Sense Function (see 7.2.4.6).

5.2.4 Layer Management Model. The following model provides the descriptions for Layer Management facilities.

5.2.4.1 Common Constants and Types. The following are the common constants and types required for the Layer Management procedures:

```
const
   maxFrameSize = ...; {in octets, implementation-dependent, see 4.4}
   maxDeferTime = ...; {2 × (maxFrameSize × 8), in bits, error timer limit for maxDeferTime}
   maxLarge = 4294967295; {maximum value (2^32 – 1) of wraparound 32 bit counter}
   max64 = xxxxxxxxx; {maximum value (2^64 – 1) of wraparound 64 bit counter}
   oneBitTime = 1; {the period it takes to transmit one bit}
type
   CounterLarge = 0..maxLarge--See footnote.;
```

5.2.4.2 Transmit Variables and Procedures. The following items are specific to frame transmission:

```
var
   excessDefer: Boolean; {set in process DeferTest}
   carrierSenseFailure: Boolean; {set in process CarrierSenseTest}
   transmitEnabled: Boolean; {set by MAC action}
   lateCollisionError: Boolean; {set in Section 4 procedure WatchForCollision}
   deferred: Boolean; {set in Section 4 function TransmitLinkMgmt}
   carrierSenseTestDone: Boolean; {set in process CarrierSenseTest}
```

[8] 32 bit counter size specification is not a part of this ISO/IEC standard. Resolution of 32 vs. 64 bit counter size will be addressed during the further work required to develop this section into a specification sufficient for ISO/IEC interoperability requirements.

```
    lateCollisionCount: 0..attemptLimit – 1; {count of late collision that is used in Section 4
                                              TransmitLinkMgmt}

    {MAC transmit counters}
    framesTransmittedOK: CounterLarge; {mandatory}
    singleCollisionFrames: CounterLarge; {mandatory}
    multipleCollisionFrames: CounterLarge; {mandatory}
    collisionFrames: array [1..attemptLimit – 1] of CounterLarge; {recommended}
    octetsTransmittedOK: CounterLarge; {recommended}
    deferredTransmissions: CounterLarge; {recommended}
    multicastFramesTransmittedOK: CounterLarge; {optional}
    broadcastFramesTransmittedOK: CounterLarge; {optional}
    {MAC transmit error counters}
    lateCollision: CounterLarge; {recommended}
    excessiveCollision: CounterLarge; {recommended}
    carrierSenseErrors: CounterLarge; {optional}
    excessiveDeferral: CounterLarge; {optional}
```

Procedure LayerMgmtTransmitCounters is invoked from the TransmitLinkMgmt function in 4.2.8 to update the transmit and transmit error counters.

```
procedure LayerMgmtTransmitCounters;
begin
  while not carrierSenseTestDone do nothing;
  if transmitSucceeding then
  begin
    IncLargeCounter(framesTransmittedOK);
    SumLarge(octetsTransmittedOK, dataSize/8); {dataSize (in bits) is defined in 4.2.7.1}
    if destinationField = ... {check to see if to a multicast destination}
      then IncLargeCounter(multicastFramesTransmittedOK);
    if destinationField = ... {check to see if to a broadcast destination}
      then IncLargeCounter(broadcastFramesTransmittedOK);

    if attempts > 1 then
    begin {transmission delayed by collision}
      if attempts = 2 then
        IncLargeCounter(singleCollisionFrames) {delay by 1 collision}
      else {attempts > 2, delayed by multiple collisions}
        IncLargeCounter(multipleCollisionFrames)
      IncLargeCounter(collisionFrames[attempts – 1]);
    end; {delay by collision}
  end; {transmitSucceeding}

  if deferred and (attempts = 1) then
    IncLargeCounter(deferredTransmissions);
  if lateCollisionCount > 0 then {test if late collision detected}
    SumLarge(lateCollision, lateCollisionCount);
  if attempts = attemptLimit and not transmitSucceeding then
    IncLargeCounter(excessiveCollision);
  if carrierSenseFailure then
    IncLargeCounter(carrierSenseErrors);
  if excessDefer then
    IncrementLargeCounter(excessiveDeferral);
end; {LayerMgmtTransmitCounters}
```

The DeferTest process sets the excessDefer flag if a transmission attempt has been deferred for a period of time longer than maxDeferTime.

```
process DeferTest;
   var deferBitTimer: 0..maxDeferTime;
begin
   cycle
   begin
      deferCount := 0;
      while frameWaiting and not excessDefer do
      begin
         Wait(oneBitTime); {see 4.3.3}
         if deferBitTimer = maxDeferTime then
            excessDefer := true
         else
            deferBitTimer := deferBitTimer + 1;
      end; {while}
      while transmitting do nothing;
   end; {cycle}
end; {DeferTest}
```

The CarrierSenseTest process sets the carrierSpenseFailure flag if carrier sense disappears while transmitting or if it never appears during an entire transmission.

```
process CarrierSenseTest;
   var
      carrierSeen: Boolean; {Running indicator of whether or not carrierSense has been true at any
                                 time during the current transmission}
      collisionSeen: Boolean; {Running indicator of whether or not the collisionDetect asserted any
                                   time during the entire transmission}
begin
   cycle {main loop}
      while not transmitting do nothing; {wait for start of transmission}
      carrierSenseFailure := false;
      carrierSeen := false;
      collisionSeen := false;
      carrierSenseTestDone := false;
      while transmitting do
      begin {inner loop}
         if carrierSense then
            carrierSeen := true;
         else
            if carrierSeen then {carrierSense disappeared before end of transmission}
                  carrierSenseFailure := true;
         if collisionDetect then
            collisionSeen := true;
      end; {inner loop}
      if not carrierSeen then
         carrierSenseFailure := true {carrier sense never appeared}
      else
         if collisionSeen then
            carrierSenseFailure := false;
      carrierSenseTestDone := true;
   end; {main loop}
end; {CarrierSenseTest}
```

5.2.4.3 Receive Variables and Procedures. The following items are specific to frame reception:

```
var
   receiveEnabled: Boolean; {set by MAC action}
```

```
{MAC receive counters}
framesReceivedOK: CounterLarge; {mandatory}
octetsReceivedOK: CounterLarge; {recommended}

{MAC receive error counters}
frameCheckSequenceErrors: CounterLarge; {mandatory}
alignmentErrors: CounterLarge; {mandatory}
inRangeLengthErrors: CounterLarge; {optional}
outOfRangeLengthField: CounterLarge; {optional}
frameTooLongErrors: CounterLarge; {optional}

{MAC receive address counters}
multicastFramesReceivedOK: CounterLarge; {optional}
broadcastFramesReceivedOK: CounterLarge; {optional}
```

Procedure LayerMgmtReceiveCounters is called by ReceiveLinkMgmt in 4.2.9 and increments the appropriate receive counters.

```
procedure LayerMgmtReceiveCounters (status: ReceiveStatus);
begin
  case status of
    receiveDisabled:
    begin
      nothing;
    end {receiveDisabled}
    receiveOK:
    begin
      IncLargeCounter(framesReceivedOK);
      SumLarge(octetsReceivedOK, dataSize/8); {dataSize (in bits) is defined in 4.2.7.1}
      if destinationField = ... {check to see if to a multicast destination}
        then IncLargeCounter(multicastFramesReceivedOK);
      if destinationField = ... {check to see if to a broadcast destination}
        then IncLargeCounter(broadcastFramesReceivedOK);
    end; {receiveOK}
    frameTooLong:
    begin
      IncLargeCounter(frameTooLongErrors);
    end; {frameTooLong}
    frameCheckError:
    begin
      IncLargeCounter(frameCheckSequenceErrors);
    end; {frameCheckError}
    alignmentError:
    begin
      IncLargeCounter(alignmentErrors);
    end; {alignmentError}
    lengthError:
    begin
      if {length field value is between the minimum unpadded LLCDataSize and maximum allowed
      LLCDataSize inclusive, and does not match the number of LLC data octets received} or
      {length field value is less than the minimum allowed unpadded LLC data size and the number
      of LLC data octets received is greater than the minimum unpadded LLCDataSize} then
      IncLargeCounter(inRangeLengthError);
    end; {lengthError}
  end; {case status}
  if {length field value is greater than the maximum allowed LLCDataSize} then
  IncLargeCounter(outOfRangeLengthField);
```

```
    end; {LayerMgmtReceiveCounters}
```

Function LayerMgmtRecognizeAddress checks if reception of certain addressing types has been enabled. Note that in Pascal, assignment to a function causes the function to return immediately.

```
function LayerMgmtRecognizeAddress(address: AddressValue): Boolean;
begin
   if {promiscuous receive enabled} then
      LayerMgmtRecognizeAddress := true;
   if address = ... {MAC station address} then
      LayerMgmtRecognizeAddress := true;
   if address = ... {broadcast address} then
      LayerMgmtRecognizeAddress := true;
   if address = ... {one of the addresses on the multicast list and multicast reception is enabled} then
      LayerMgmtRecognizeAddress := true;
   LayerMgmtRecognizeAddress := false;
end; {LayerMgmtRecognizeAddress}
```

5.2.4.4 Common Procedures. Procedure LayerMgmtInitialize initializes all the variables and constants required to implement Layer Management.

```
procedure LayerMgmtInitialize;
begin
   {initialize flags for enabling/disabling transmission and reception}
   receiveEnabled := true;
   transmitEnabled := true;

   {initialize transmit flags for DeferTest and CarrierSenseTest}
   deferred := false;
   lateCollisionError := false;
   excessDefer := false;
   carrierSenseFailure := false;
   carrierSenseTestDone := false;

   {Initialize all MAC sublayer management counters to zero}

end; {LayerMgmtInitialize}
```

Procedure IncLargeCounter increments a 32 bit wraparound counter.

```
procedure IncLargeCounter (var counter: CounterLarge);
begin
   {increment the 32 bit counter}
end; {IncLargeCounter}
```

Procedure SumLarge adds a value to a 32 bit wraparound counter.

```
procedure SumLarge (
   var counter: CounterLarge;
   var offset: Integer);
begin
   {add offset to the 32 bit counter}
end; {SumLarge}
```

6. PLS Service Specifications

6.1 Scope and Field of Application. This section specifies the services provided by the Physical Signaling (PLS) sublayer to the MAC sublayer for the CSMA/CD section of the Local Area Network Standard, Fig 6-1. The services are described in an abstract way and do not imply any particular implementation.

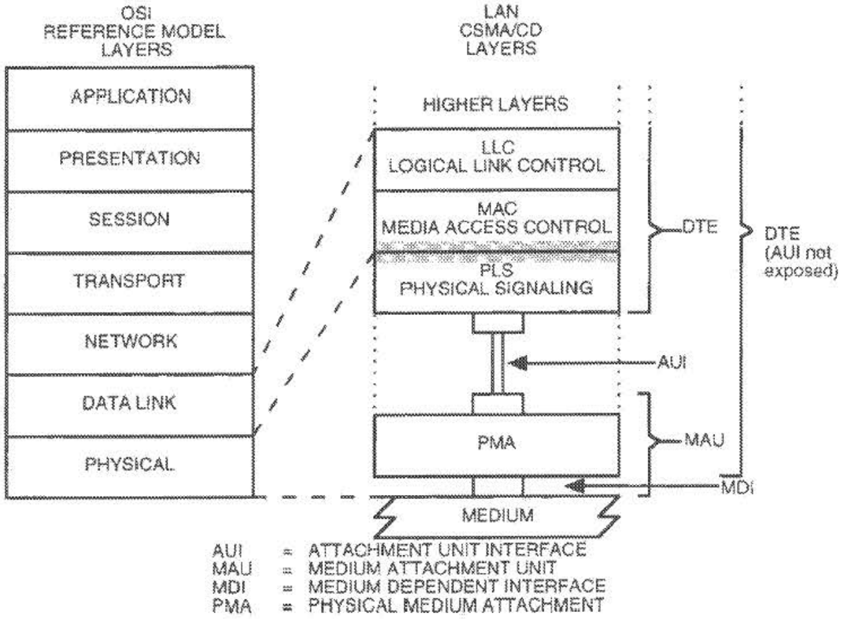


Fig 6-1
Service Specification Relationship to the IEEE 802.3 CSMA/CD LAN Model

6.2 Overview of the Service

6.2.1 General Description of Services Provided by the Layer. The services provided by the PLS sublayer allow the local MAC sublayer entity to exchange data bits (PLS data_units) with peer MAC sublayer entities.

6.2.2 Model Used for the Service Specification. The model used in this service specification is identical to that used in 1.2.2.1.

6.2.3 Overview of Interactions. The primitives associated with the MAC sublayer to PLS sublayer interface fall into two basic categories:

(1) Service primitives that support MAC peer-to-peer interactions
(2) Service primitives that have local significance and support sublayer-to-sublayer interactions

The following primitives are grouped into these two categories:

(1) Peer-to-Peer
PLS_DATA.request
PLS_DATA.indication

(2) Sublayer-to-Sublayer
PLS_CARRIER.indication
PLS_SIGNAL.indication

The PLS_DATA primitives support the transfer of data from a single MAC sublayer entity to all other peer MAC sublayer entities contained within the same local area network defined by the broadcast medium.

NOTE: This also means that all bits transferred from a given MAC sublayer entity will in turn be received by the entity itself.

The PLS_CARRIER and the PLS_SIGNAL primitives provide information needed by the local MAC sublayer entity to perform the media access functions.

6.2.4 Basic Services and Options. All of the service primitives described in this section are considered mandatory.

6.3 Detailed Service Specification

6.3.1 Peer-to-Peer Service Primitives

6.3.1.1 PLS_DATA.request

6.3.1.1.1 Function. This primitive defines the transfer of data from the MAC sublayer to the local PLS entity.

6.3.1.1.2 Semantics of the Service Primitive. The primitive shall provide the following parameters:

PLS_DATA.request (OUTPUT_UNIT)

The OUTPUT_UNIT parameter can take on one of three values: ONE, ZERO, or DATA_COMPLETE and represent a single data bit. The DATA_COMPLETE value signifies that the Media Access Control sublayer has no more data to output.

6.3.1.1.3 When Generated. This primitive is generated by the MAC sublayer to request the transmission of a single data bit on the physical medium or to stop transmission.

6.3.1.1.4 Effect of Receipt. The receipt of this primitive will cause the PLS entity to encode and transmit either a single data bit or to cease transmission.

6.3.1.2 PLS_DATA.indicate

6.3.1.2.1 Function. This primitive defines the transfer of data from the PLS sublayer to the MAC sublayer.

6.3.1.2.2 Semantics of the Service Primitive. The semantics of the primitive are as follows:

PLS_DATA.indicate (INPUT_UNIT)

The INPUT_UNIT parameter can take one of two values each representing a single bit: ONE or ZERO.

6.3.1.2.3 When Generated. The PLS_DATA.indicate is generated to all MAC sublayer entities in the network after a PLS_DATA.request is issued.

NOTE: An indicate is also presented to the MAC entity that issued the request.

6.3.1.2.4 Effect of Receipt. The effect of receipt of this primitive by the MAC sublayer is unspecified.

6.3.2 Sublayer-to-Sublayer Service Primitives

6.3.2.1 PLS_CARRIER.indicate

6.3.2.1.1 Function. This primitive transfers the status of the activity on the physical medium from the PLS sublayer to the MAC sublayer.

6.3.2.1.2 Semantics of the Service Primitive. The semantics of the primitive are as follows:

PLS_CARRIER.indicate (CARRIER_STATUS)

The CARRIER_STATUS parameter can take one of two values: CARRIER_ON or CARRIER_OFF. The CARRIER_ON value indicates that the DTE Physical Layer had received an *input* message or a *signal_quality_error* message from the MAU. The CARRIER_OFF value indicates that the DTE Physical Layer had received an *input_idle* message and is not receiving an SQE *signal_quality_error* message from the MAU.

6.3.2.1.3 When Generated. The PLS_CARRIER.indicate service primitive is generated whenever CARRIER_STATUS makes a transition from CARRIER_ON to CARRIER_OFF or vice versa.

6.3.2.1.4 Effect of Receipt. The effect of receipt of this primitive by the MAC sublayer is unspecified.

6.3.2.2 PLS_SIGNAL.indicate

6.3.2.2.1 Function. This primitive transfers the status of the Physical Layer signal quality from the PLS sublayer to the MAC sublayer.

6.3.2.2.2 Semantics of the Service Primitive. The semantics of the service primitive are as follows:

PLS_SIGNAL.indicate (SIGNAL_STATUS)

The SIGNAL_STATUS parameter can take one of two values: SIGNAL_ERROR or NO_SIGNAL_ERROR. The SIGNAL_ERROR value indicates to the MAC sublayer that the PLS has received a signal_quality_error message from the MAU. The NO_SIGNAL_ERROR value indicates that the PLS has ceased to receive signal_quality_error messages from the MAU.

6.3.2.2.3 When Generated. The PLS_SIGNAL.indicate service primitive is generated whenever SIGNAL_ STATUS makes a transition from SIGNAL_ERROR to NO_SIGNAL_ERROR or vice versa.

6.3.2.2.4 Effect of Receipt. The effect of receipt of this primitive by the MAC sublayer is unspecified.

7. Physical Signaling (PLS) and Attachment Unit Interface (AUI) Specifications

7.1 Scope. This section defines the logical, electrical, and mechanical characteristics for the PLS and AUI between Data Terminal Equipment and Medium Attachment Units used in CSMA/CD local area networks. The relationship of this specification to the entire ISO [IEEE] Local Area Network standards is shown in Fig 7-1. The purpose of this interface is to provide an interconnection that is simple and inexpensive and that permits the development of simple and inexpensive MAUs.

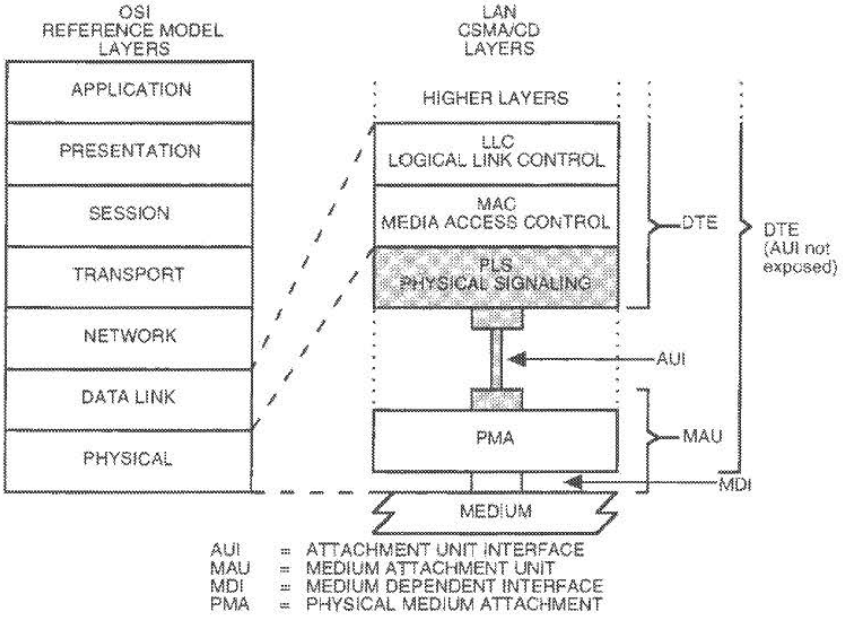


Fig 7-1
Physical Layer Partitioning, Relationship to the ISO Open Systems Interconnection (OSI) Reference Model

This interface has the following characteristics:

(1) Capable of supporting one or more of the specified data rates
(2) Capable of driving up to 50 m (164 ft) of cable
(3) Permits the DTE to test the AUI, AUI cable, MAU, and the medium itself
(4) Supports MAUs for baseband coax, broadband coax, and baseband fiber

7.1.1 Definitions

Attachment Unit Interface (AU Interface) (AUI). In a local area network, the interface between the medium attachment unit and the data terminal equipment within a data station.

NOTE: The AUI carries encoded control and data signals between the DTE's PLS sublayer and the MAU's PMA sublayer and provides for duplex data transmission.

BR. The rate of data throughput (bit rate) on the medium in bits per second.

bit time. The duration of one bit symbol (1/BR).

circuit. The physical medium on which signals are carried across the AUI. The data and control circuits consist of an A circuit and a B circuit forming a balanced transmission system so that the signal carried on the B circuit is the inverse of the signal carried on the A circuit.

Clocked Data One (CD1). A Manchester encoded data "1." A CD1 is encoded as a LO for the first half of the bit-cell and a HI for the second half of the bit-cell.

Clocked Data Zero (CD0). A Manchester encoded data "0." A CD0 is encoded as a HI for the first half of the bit-cell and a LO for the second half of the bit-cell.

Control Signal One (CS1). An encoded control signal used on the Control In and Control Out circuits. A CS1 is encoded as a signal at half the bit rate (BR/2).

Control Signal Zero (CS0). An encoded control signal used on the Control In and Control Out circuits. A CS0 is encoded as a signal at the bit rate (BR).

idle (IDL). A signal condition where no transition occurs on the transmission line is used to define the end of a frame and ceases to exist after the next LO to HI transition on the AUI circuits. An IDL always begins with a HI signal level. A driver is required to send the IDL signal for at least 2 bit times and a receiver is required to detect IDL within 1.6 bit times. See 7.3 for additional details.

7.1.2 Summary of Major Concepts

(1) Each direction of data transfer is serviced with two (making a total of four) balanced circuits: "Data" and "Control."
(2) The Data and Control circuits are independently self-clocked, thereby, eliminating the need for separate timing circuits. This is accomplished with encoding of all signals. The Control circuit signaling rate is nominally (but not of necessity exactly) equal to the Data circuit signaling rate.
(3) The Data circuits are used only for data transfer. No control signals associated with the interface are passed on these circuits. Likewise, the Control circuits are used only for control message transfer. No data signals associated with the interface are passed on these circuits.

7.1.3 Application. This standard applies to the interface used to interconnect Data Terminal Equipment (DTE) to a MAU that is not integrated as a physical part of the DTE. This interface is used to

(1) Provide the DTE with media independence for baseband coax, broadband coax, and baseband fiber media so that identical PLS, MAC and LLC may be used with any of these media.
(2) Provide for the separation by cable of up to 50 m (164 ft) the DTE and the MAU.

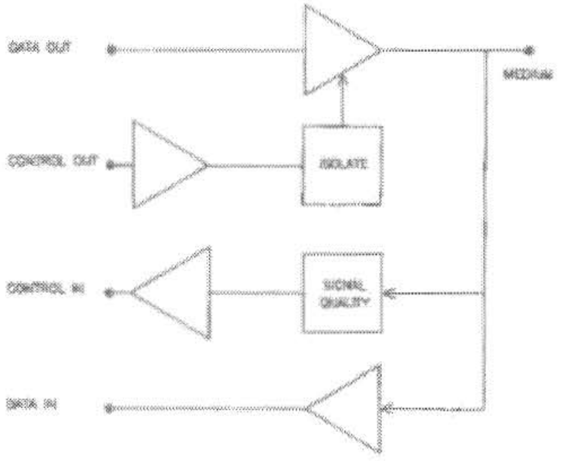
7.1.4 Modes of Operation. The AUI can operate in two different modes. All interfaces shall support the normal mode. The monitor mode is optional.

When the interface is being operated in the *normal* mode, the AUI is logically connected to the MDI. The DTE is required to follow the media access algorithms, which provide a single access procedure compatible with all local area network media, to send data over the AUI. The MAU always sends back to the DTE whatever data the MAU receives on the MDI.

When the interface is in the optional *monitor* mode, the MAUs transmitter is logically isolated from the medium. The MAU, in this mode, functions as an observer on the medium. Both the input function and the signal quality error function are operational (see the MAU state diagrams for specific details).

7.1.5 Allocation of Function. The allocation of functions in the AUI is such that the majority of the functionality required by the interface can be provided by the DTE, leaving the MAU as simple as possible. This division of functions is based upon the recognition of the fact that since, in many cases, the MAU may be located in an inaccessible location adjacent to the physical medium, service of the MAU may often be difficult and expensive.

7.2 Functional Specification. The AUI is designed to make the differences among the various media as transparent as possible to the DTE. The selection of logical control signals and the functional procedures are all designed to this end. Figure 7-2 is a reference model, a generalized MAU as seen by the DTE through the AUI.



NOTE: The AUI (comprised of DO, DI, CO, CI circuits) is not exposed when the MAU is, optionally, part of the DTE.

Fig 7-2
Generalized MAU Model

Many of the terms used in this section are specific to the interface between this sublayer and the MAC sublayer. These terms are defined in the Service Specification for the PLS sublayer.

7.2.1 PLS–PMA (DTE–MAU) Interface Protocol. The DTE and MAU communicate by means of a simple protocol across the AUI.

7.2.1.1 PLS to PMA Messages. The following messages can be sent by PLS sublayer entities in the DTE to PMA sublayer entities in the MAU:

Message	Meaning
output	Output information
output_idle	No data to be output
normal	Cease to isolate the MAU
(Optional)	
isolate	Isolate MAU
mau_request	Request that the MAU be made available

7.2.1.1.1 *output* Message. The PLS sublayer sends an output message to the PMA sublayer when the PLS sublayer receives an OUTPUT_UNIT from the MAC sublayer.

The physical realization of the *output* message is a CD0 or a CD1 sent by the DTE to the MAU on the Data Out circuit. The DTE sends a CD0 if the OUTPUT_UNIT is a ZERO or a CD1 if the OUTPUT_UNIT is a ONE. This message is time coded—that is, once this message has been sent, the function is not completed over the AUI until one bit time later. The *output* message cannot be sent again until the bit cell being sent as a result of sending the previous *output* message is complete.

7.2.1.1.2 *output_idle* Message. The PLS sublayer sends an *output_idle* message to the PMA sublayer at all times when the MAC sublayer is not in the process of transferring output data across the MAC to PLS interface. The *output_idle* message is no longer sent (and the first OUTPUT_UNIT is sent using the *output* message) as soon after the arrival of the first OUTPUT_UNIT as the MAU can be made available for data output. The *output_idle* message is again sent to the MAU when the DATA_COMPLETE is received from the MAC sublayer. The dotailed usage of the *output_idle* message is shown in Fig 7-5.

The physical realization of the *output_idle* message is IDL sent by the DTE to the MAU on the Data Out circuit.

7.2.1.1.3 *normal* Message. The PLS sublayer sends a *normal* message to the PMA sublayer after it receives the PLS *start* message from the PLS Reset and Identify Function. The *normal* message is also sent after receipt of RESET_MONITOR_MODE from the management entity. The *normal* message is sent continuously by the PLS sublayer to the MAU, unless the PLS Output Function roquires that the *mau_request* message be sent to permit data output. If *mau_request* is sent during data output, the sending of *normal* will be resumed when the PLS Output Function returns to the IDLE state. The *normal* signal is reset by the SET_MONITOR_MODE (this reset function is described more fully by Fig 7-4).

7.2.1.1.4 *isolate* Message (Optional). The PLS sublayer sends an *isolate* message to the PMA (in the MAU) whenever the PLS sublayer receives SET_MONITOR_MODE from the management entity. In response to the *isolate* message, the MAU causes the means employed to impress data on the physical medium to be positively prevented from affecting the medium. Since signaling and isolation techniques differ from medium to medium, the manner in which this positive isolation of the transmitting means is accomplished is specified in the appropriate MAU section. However, the intent of this positive isolation of the transmitter is to ensure that the MAU will not interfere with the physical medium in such a way as to affect transmissions of other stations even in the event that the means normally employed to prevent the transmitter from affecting the medium have failed to do so. The specification of positive isolation is not to be construed to preclude use of either active or passive devices to accomplish this function.

The physical realization of the *isolate* message is a CS0 *signal* sent by the DTE to the MAU over the Control Out circuit.

7.2.1.1.5 *mau_request* Message (Optional). The PLS sublayer sense the mau_request message to the PMA sublayer if the PMA sublayer is sending the *mau_not_available* message and the MAC sublayer has sent the first OUTPUT_UNIT of a new transmission. The PLS sublayer continues to send the *mau_request* message to the MAU until the MAC sublayer sends the DATA_COMPLETE roquest to the PLS sublayer across the MAC to PLS interface. See Figs 7-3, 7-5, and 7-9 for details.

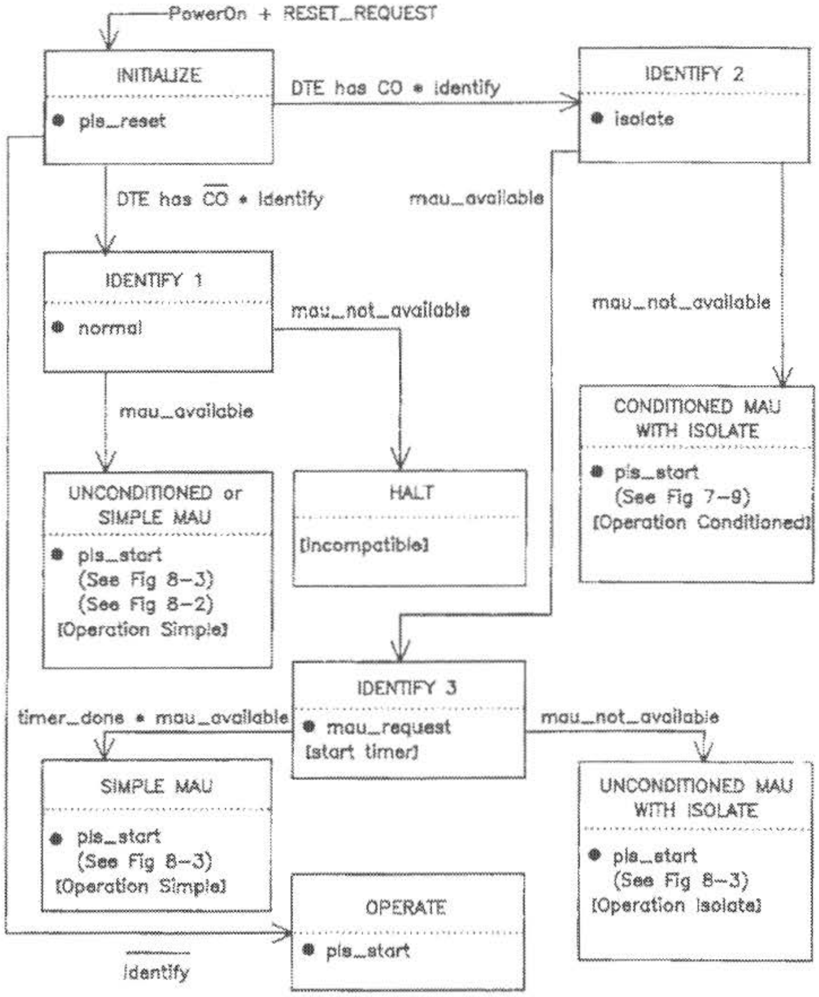
In addition, the *mau_request* message is used by the Reset and Identify Function in the IDENTIFY 3 state to determine whether the MAU has the Isolate Function.

The physical realization of *mau_request* is a CS1 sent by the DTE to the MAU on the Control Out circuit.

The physical realization of the *normal* message is the IDL signal sent by the DTE to the MAU on the Control Out circuit. In the absence of the CO circuit, MAUs implementing the Isolate Function shall act as if the *normal* message is present. The CO circuit components may be absent from the DTE, AUI, or MAU.

7.2.1.2 PMA to PLS Interface. The following messages can be sent by the Physical Medium Attachment sublayer entities in the MAU to the PLS sublayer entities in the DTE:

Message	Meaning
input	Input information
input_idle	No input information
signal_quality_error	Error detected by MAU
mau_available	MAU is available for output
(Optional)	
mau_not_available	MAU is not available for output



NOTES: (1) All states may be omitted except INITIALIZE and OPERATE

(2) "Identify" means DTE can recognize uniquely all CI messages and the entire function has been implemented

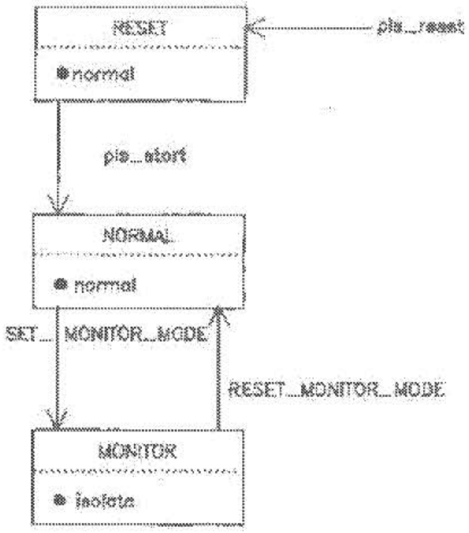
(3) "Identify" with bar means DTE fails to recognize *mau_not_available* or has a partial implementation of the function

Fig 7-3
PLS Reset and Identify Function

7.2.1.2.1 *input* Message. The PMA sublayer sends an *input* message to the PLS sublayer when the MAU has received a bit from the medium and is prepared to transfer this bit to the DTE. The actual mapping of the signals on the medium to the type of *input* message to be sent to the DTE is contained in the specifications for each specific MAU type. In general, when the *signal_quality_error* message is being sent by the MAU, the symmetry specifications for circuit DI are not guaranteed to be met.

The physical realization of the *input* message consists of CD0 or CD1 waveforms. If the *signal_quality_error* message is being sent from the MAU, the input waveform is unpredictable.

NOTE: This signal is not necessarily retimed by the MAU. Consult the appropriate MAU specification for timing and jitter.



NOTE: Monitor State is optional.

Fig 7-4
PLS Mode Function

7.2.1.2.2 *input_idle* Message. The PMA sublayer sends an *input_idle* message to the PLS sublayer when the MAU does not have data to send to the DTE.

The physical realization of the *input_idle* message is an IDL sent by the MAU to the DTE on the Data In circuit.

7.2.1.2.3 *signal_quality_error* Message. The PMA sublayer sends a *signal_quality_error* message to the PLS sublayer in response to any of three possible conditions. These conditions are improper signals on the medium, collision on the medium, and reception of the *output_idle* message. They are described in the following numbered paragraphs. The physical realization of the *signal_quality_error* message is a CS0 sent by the MAU to the DTE on the Control In circuit.

NOTE: The MAU is required to assert the *signal_quality_error* message at the appropriate times whenever the MAU is powered, and not just when the DTE is requesting data output. See Figs 7-9, 8-2, and 8-3 for details.

(1) Improper Signals on the Medium. The MAU may send the *signal_quality_error* message at any time due to improper signals on the medium. The exact nature of these improper signals are medium-dependent. Typically, this condition might be caused by a malfunctioning MAU (for example, repeater or head-end) connected to the medium or by a break or short in the medium. See the appropriate MAU specification for specific conditions that may cause improper signals on a given medium.

(2) Collision. Collision occurs when more than one MAU is transmitting on the medium. The local MAU shall send the *signal_quality_error* message in every instance when it is possible for it to ascertain that more than one MAU is transmitting on the medium. The MAU shall make the best determination possible. The MAU shall not send the *signal_quality_error* message when it is unable to determine conclusively that more than one MAU is transmitting.

(3) *signal_quality_error* Message Test. The MAU sends the *signal_ quality_error* message at the completion of the Output Function. See Fig 7-9 and Section 8 for a more complete description of this test.

7.2.1.2.4 *mau_available* Message. The PMA sublayer sends the *mau_available* message to the PLS sublayer when the MAU is available for output. The *mau_available* message is always sent by a MAU that is always prepared to output data except when it is required to signal the *signal_quality_error*

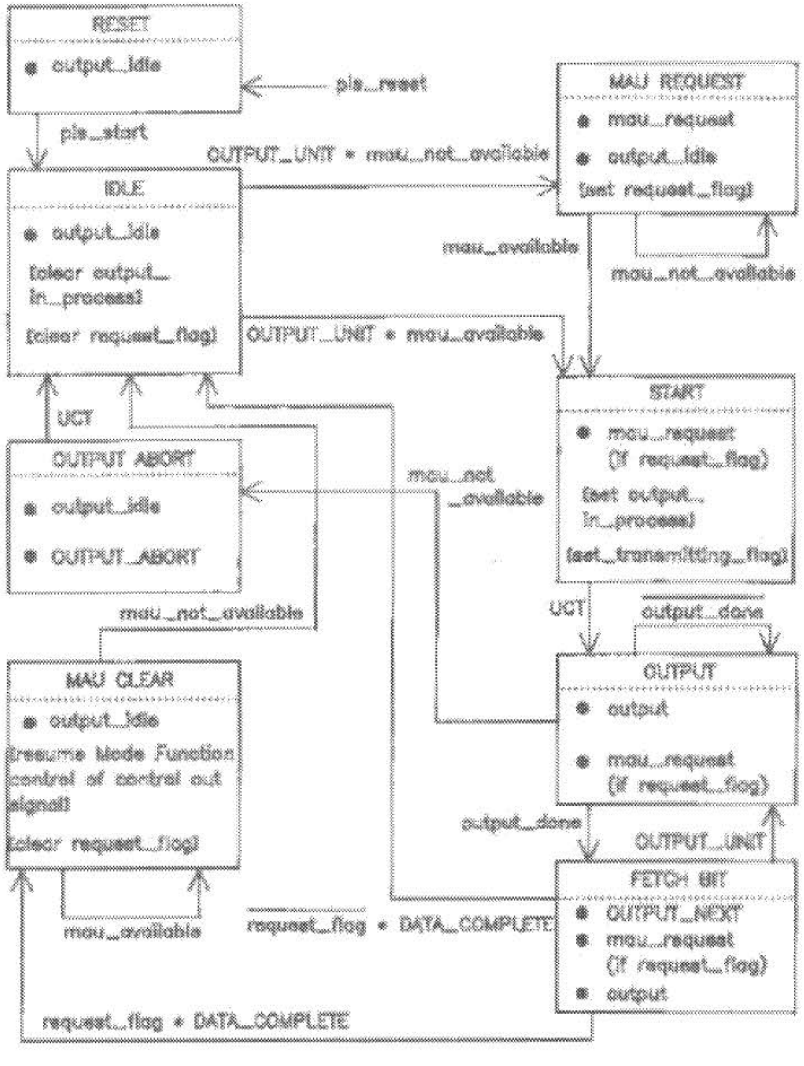


Fig 7-5
PLS Output Function

message. Such a MAU does not require *mau_request* to prepare itself for data output. See Figs 7-3, 7-6, and 7-9 for details.

The physical realization of the *mau_available* message is an IDL sent by the MAU to the DTE on the Control In circuit.

7.2.1.2.5 *mau_not_available* Message (OPTIONAL). The PMA sublayer sends a *mau_not_available* message to the PLS sublayer when the MAU is not available for output. Figure 7-5 shows the relationship of *mau_not_available* to the Output Function.

The *mau_not_available* message is also used by a MAU that contains the Isolate Function and does not need to be conditioned for output to signal the presence of the Isolate Function during the PLS Reset Function (see Fig 7-3 and 8-3).

The physical realization of the *mau_not_available* message is a CS1 sent by the MAU to the DTE on the Control In circuit.

7.2.2 PLS Interface to MAC and Management Entities. The PLS sublayer interfaces described here are for reference only. This section specifies the services sent between the MAC sublayer and the PLS sublayer.

7.2.2.1 PLS–MAC Interface. The following messages can be sent between PLS sublayer entities and MAC sublayer entities:

Message	Meaning
OUTPUT_UNIT	Data sent to the MAU
OUTPUT_STATUS	Response to OUTPUT_UNIT
INPUT_UNIT	Data received from the MAU
CARRIER_STATUS	Indication of input activity
SIGNAL_STATUS	Indication of error/no error condition

7.2.2.1.1 OUTPUT_UNIT. The MAC sublayer sends the PLS sublayer an OUTPUT_UNIT every time the MAC sublayer has a bit to send. Once the MAC sublayer has sent an OUTPUT_UNIT to the PLS sublayer, it may not send another OUTPUT_UNIT until it has received an OUTPUT_STATUS message from the PLS sublayer. The OUTPUT_UNIT is a ONE if the MAC sublayer wants the PLS sublayer to send a CD1 to the PMA sublayer, a ZERO if a CD0 is desired, or a DATA_COMPLETE if an IDL is desired.

7.2.2.1.2 OUTPUT_STATUS. The PLS sublayer sends the MAC sublayer OUTPUT_STATUS in response to every OUTPUT_UNIT received by the PLS sublayer. OUTPUT_STATUS sent is an OUTPUT_NEXT if the PLS sublayer is ready to accept the next OUTPUT_UNIT from the MAC sublayer, or an OUTPUT_ABORT if the PLS sublayer was not able to process the previous OUTPUT_UNIT. (The purpose of OUTPUT_STATUS is to synchronize the MAC sublayer data output with the data rate of the physical medium.)

7.2.2.1.3 INPUT_UNIT. The PLS Sublayer sends the MAC sublayer an INPUT_UNIT every time the PLS receives an *input* message from the PMA sublayer. The INPUT_UNIT is a ONE if the PLS sublayer receives a CD1 from the PMA sublayer, a ZERO if the PLS sublayer receives a CD0 from the PMA sublayer.

7.2.2.1.4 CARRIER_STATUS. The PLS sublayer sends the MAC sublayer CARRIER_STATUS whenever the PLS sublayer detects a change in carrier status. The PLS sublayer sends CARRIER_ON when it receives an *input* or *signal_quality_error* message from the PMA and the previous CARRIER_STATUS that the PLS sublayer sent to the MAC sublayer was CARRIER_OFF. The PLS sublayer sends CARRIER_OFF when it receives an *input_idle* from the PMA sublayer, no *signal_quality_error* (either *mau_available* or *mau_not_available*) message and the previous CARRIER_STATUS that the PLS sublayer sent to the MAC sublayer was CARRIER_ON.

7.2.2.1.5 SIGNAL_STATUS. The PLS sublayer sends the MAC sublayer SIGNAL_STATUS whenever the PLS sublayer detects a change in the signal quality (as reported by the PMA). The PLS sublayer sends SIGNAL_ERROR when it receives a *signal_quality_error* message from the PMA sublayer and the previous SIGNAL_STATUS the PLS sublayer sent was NO_SIGNAL_ERROR. The PLS sublayer sends NO_SIGNAL_ERROR when it receives no *signal_quality_error* (either *mau_available* or *mau_not_available*) message from the PMA sublayer and the previous CARRIER_STATUS that the PLS sent to the MAC sublayer was SIGNAL_ERROR.

7.2.2.2 PLS-Management Entity Interface. The following messages may be sent between the PLS sublayer entities and intralayer or higher layer management entities:

Message	Meaning
RESET_REQUEST	Reset PLS to initial "Power On" state
RESET RESPONSE	Provides operational information
MODE_CONTROL	Control operation
SQE_TEST	Signal Quality Error test results

7.2.2.2.1 RESET_REQUEST. The management entity sends the PLS sublayer RESET_REQUEST when the PLS sublayer needs to be reset to a known state. Upon receipt of RESET_REQUEST, the PLS sublayer resets all internal logic and restarts all functions. See Fig 7-3 for details.

7.2.2.2.2 RESET_RESPONSE. The PLS sublayer sends the management entity RESET_R SPONSE upon completion of the Reset and Identify Function (see Fig 7-3 and 7.2.4.1) whether invoked due to power on or due to a RESET_REQUEST. Which RESET_RESPONSE was sent is determined by the Reset and Identify Function. A RESET_RESPONSE of OPERATION SIMPLE, OPERATION ISOLATE, or OPERATION CONDITIONED is sent if the MAU is compatible with the DTE and the MAU is simple (no isolate) or if the DTE does not support Isolate even if Isolate is supported by the MAU, supports Isolate but does not require conditioning, or supports Isolate and does require conditioning to output. A RESET_RESPONSE of INCOMPATIBLE is sent if the MAU is not compatible with the DTE (that is, the MAU requires conditioning but the DTE does not support conditioning).

7.2.2.2.3 MODE_CONTROL. The management entity sends MODE_CONTROL to the PLS sublayer to control PLS functions. MODE_CONTROL capabilities are as follows:

Message	Meaning
ACTIVATE PHYSICAL	Supply power on circuit VP
DEACTIVATE PHYSICAL	Remove power from circuit VP
SET_MONITOR_MODE	Send Isolate to MAU
RESET_MONITOR_MODE	Send Normal to MAU

7.2.2.2.4 SQE_TEST. The PLS sublayer sends SQE_TEST to the management entity at the conclusion of each *signal_quality_error* test (see Output Function, 7.2.4.3). The PLS sublayer sends SQE_TEST_ERROR if the *signal_quality_error* test fails or SQE_TEST_OK if the *signal_quality_error* test passes.

7.2.3 Frame Structure. Frames transmitted on the AUI shall have the following structure:

<silence><preamble><sfd><data><etd><silence>

The frame elements shall have the following characteristics:

Element	Characteristics
<silence>	= no transitions
<preamble>	= alternating (CD1) and (CD0) | 56 bit times (ending in CD0)
<sfd>	= (CD1)(CD0)(CD1)(CD0)(CD1)(CD0)(CD1)(CD1)
<data>	= 8 × N
<etd>	= IDL

7.2.3.1 Silence. The <silence> delimiter provides an observation window for an unspecified period of time during which no transitions occur on the AUI. The minimum length of this period is specified by the access procedure.

7.2.3.2 Preamble. The <preamble> delimiter begins a frame transmission and provides a signal for receiver synchronization. The signal shall be an alternating pattern of (CD1) and (CD0). This pattern shall be transmitted on the Data Out circuit by the DTE to the MAU for a minimum of 56 bit times at the beginning of each frame. The last bit of the preamble (that is, the final bit of preamble before the start of frame delimiter) shall be a CD0.

The DTE is required to supply at least 56 bits of preamble in order to satisfy system requirements. System components consume preamble bits in order to perform their functions. The number of preamble bits sourced ensures an adequate number of bits are provided to each system component to correctly implement its function.

7.2.3.3 Start of Frame Delimiter (SFD). The <sfd> indicates the start of a frame, and follows the preamble. The <sfd> element of a frame shall be

(CD1)(CD0)(CD1)(CD0)(CD1)(CD0)(CD1)(CD1)

7.2.3.4 Data. The <data> in a transmission shall be in multiples of eight (8) encoded data bits (CD0s and CD1s).

7.2.3.5 End of Transmission Delimiter. The <etd> delimiter indicates the end of a transmission and serves to turn off the transmitter. The signal shall be an IDL.

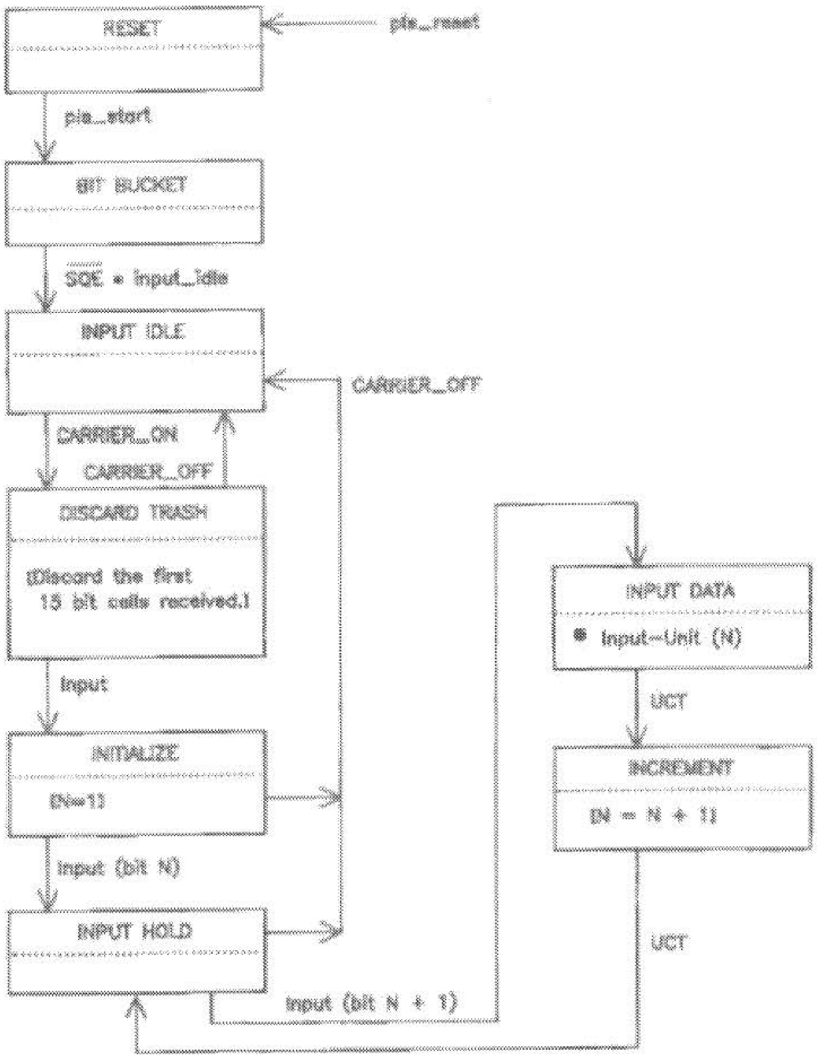
7.2.4 PLS Functions. The PLS sublayer functions consist of a Reset and Identify Function and five simultaneous and asynchronous functions. These functions are Output, Input, Mode, Error Sense, and Carrier Sense. All of the five functions are started immediately following the completion of the Reset and Identify Function. These functions are depicted in the state diagrams shown in Figs 7-3 through 7-8, using notation described in 1.2.1.

7.2.4.1 Reset and Identify Function. The Reset and Identify Function is executed any time either of two conditions occur. These two conditions are "power on" and the receipt of RESET_REQUEST from the management entity. The Reset and Identify Function initializes all PLS functions, and (optionally) determines the capability of the MAU attached to the AUI. Figure 7-3 is the state diagram of the Reset and Identify Function. The Identify portion of the function is optional.

7.2.4.2 Mode Function. The MAU functions in two modes: normal and monitor. The monitor mode is optional. The state diagram of Fig 7-4 depicts the operation of the Mode Function. When the MAU is operating in the normal mode, it functions as a direct connection between the DTE and the medium. Data sent from the DTE are impressed onto the medium by the MAU and all data appearing on the medium are sent to the DTE by the MAU. When the MAU is operating in the monitor mode, data appearing on the medium is sent to the DTE by the MAU as during the normal mode. *signal_quality_error* is also asserted on the AUI as during operation in the normal mode. However, in the monitor mode, the means employed to impress data on the physical medium is positively prevented from affecting the medium. Since signaling and isolation techniques differ from medium to medium, the manner in which this positive isolation of the transmitting means is accomplished is specified in the appropriate MAU document. However, the intent of this positive isolation of the transmitter is to ensure that the MAU will not interfere with the physical medium in such a way as to affect transmission of other stations even in the event of failure of the normal transmitter disabling control paths within the transmitting mechanism of the MAU.

The monitor mode is intended to permit a network station to determine if it is the source of interference observed on the medium.

NOTE: The monitor mode is intended to be used only by Network Management for fault isolation and network operation verification. It is intended that the *isolate* message provide direct control over the mode function so that these tasks can be performed. IMPROPER USE OF THE ISOLATE FUNCTION CAN CAUSE ERRONEOUS FRAMES. Section 5, Layer Management, provides details on the proper use of this function.

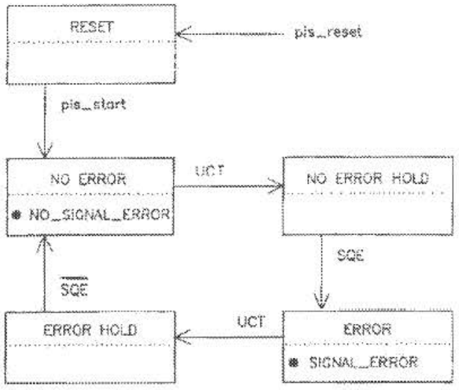


NOTE: UCT= unconditional transition

Fig 7-6
PLS Input Function

7.2.4.3 Output Function. The PLS sublayer Output Function transparently performs the tasks of conditioning the MAU for output and data transfer from the MAC sublayer to the MAU. The state diagram of Fig 7-5 depicts the Output Function operation.

At the conclusion of the Output Function, if a collision has not occurred, a test is performed to verify operation of the signal quality detection mechanism in the MAU and to verify the ability of the AUI to pass the *signal_quality_error* message to the PLS sublayer. The operation of this test in the DTE is shown in Fig 7-8.



NOTE: UCT = unconditional transition

Fig 7-7
PLS Error Sense Function

7.2.4.4 Input Function. The PLS sublayer Input Function transparently performs the task of data transfer from the MAU to the MAC sublayer. The state diagram of Fig 7-6 depicts the Input Function operation.

7.2.4.5 Error Sense Function. The PLS sublayer Error Sense Function performs the task of sending SIGNAL_STATUS to the MAC sublayer whenever there is a change in the signal quality information received from the MAU. The state diagram of Fig 7-7 depicts the Error Sense Function operation.

7.2.4.6 Carrier Sense Function. The PLS sublayer Carrier Sense Function performs the task of sending CARRIER_STATUS to the MAC sublayer every time there is a change in CARRIER_STATUS. The state diagram of Fig 7-8 depicts the Carrier Sense Function operation.

Verification of the *signal_quality_error* detection mechanism occurs in the following manner (in the absence of a fault on the medium).

(1) At the conclusion of the output function, the DTE opens a time window during which it expects to see the *signal_quality_error* signal asserted on the Control In circuit. The time window begins when CARRIER_STATUS becomes CARRIER_OFF. If execution of the Output Function does not cause CARRIER_ON to occur, no SQE test occurs in the DTE. The duration of the window shall be at least 4.0 μs but no more than 8.0 μs. During the time window (depicted as carrier_inhibit_timer, Fig 7-8) the Carrier Sense Function is inhibited.

(2) The MAU, upon waiting Tw (wait time) after the conclusion of output, activates as much of the signal quality error detecting mechanism as is possible without placing signals on the medium, thus sending the *signal_quality_error* message across the AUI for 10 ± 5 bit times (10/BR ± 5/BR seconds).

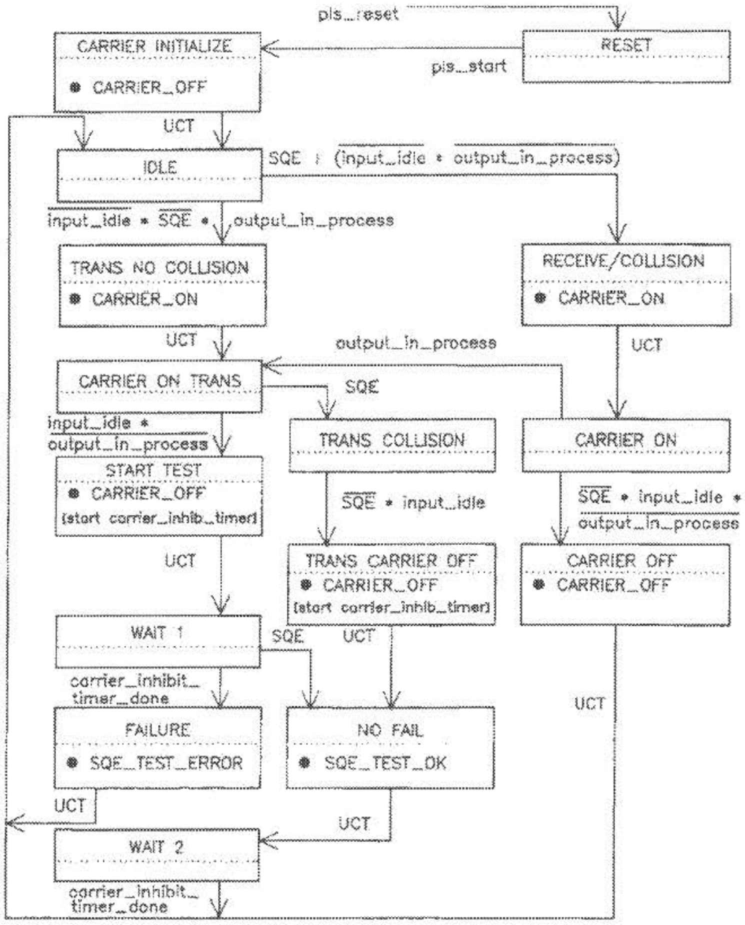
(3) The DTE interprets the reception of the *signal_quality_error* message from the MAU as indication that the *signal_quality_error* detecting mechanism is operational and the *signal_quality_error* message may be both sent by the MAU and received by the DTE.

NOTES: (1) The occurrence of multiple (overlapping) transmitters on the medium during the time that the test window is open, as specified above, will satisfy the test and will verify proper operation of the signal quality error detecting mechanism and sending and receiving of the appropriate physical error message.

(2) If *signal_quality_error* exists at the DTE before CARRIER_OFF occurs, then the Collision Presence test sequence within the PLS as described in 7.2.4.3 above shall be aborted as shown in Fig 7-8.

7.3 Signal Characteristics

7.3.1 Signal Encoding. Two different signal encoding mechanisms may be used by the AUI. One of the mechanisms is used to encode data, the other to encode control.

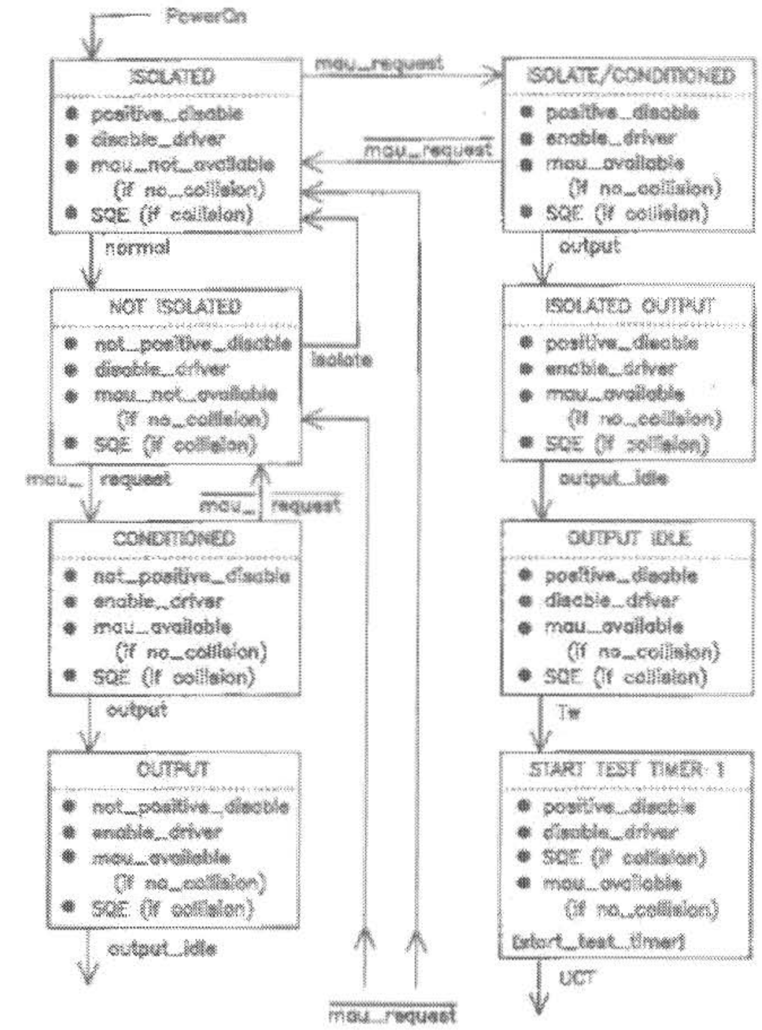


NOTE: UCT = unconditional transition
SQE = *signal_quality_error*

Fig 7-8
PLS Carrier Sense Function

7.3.1.1 Data Encoding. Manchester encoding is used for the transmission of data across the AUI. Manchester encoding is a binary signaling mechanism that combines data and clock into "bit-symbols." Each bit-symbol is split into two halves with the second half containing the binary inverse of the first half; a transition always occurs in the middle of each bit-symbol. During the first half of the bit-symbol, the encoded signal is the logical complement of the bit value being encoded. During the second half of the bit-symbol, the encoded signal is the uncomplemented value of the bit being encoded. Thus, a CD0 is encoded as a bit-symbol in which the first half is HI and the second half is LO. A CD1 is encoded as a bit-symbol in which the first half is LO and the second half is HI. Examples of Manchester waveforms are shown in Fig 7-10.

The line condition IDL is also used as an encoded signal. An IDL always starts with a HI signal level. Since IDL always starts with a HI signal, an additional transition will be added to the data stream if the last bit sent was a zero. This transition cannot be confused with clocked data (CD0 or CD1) since the transition will occur at the start of a bit cell. There will be no transition in the middle of the bit cell. The IDL



(a)

NOTE: See Figs 8-2 and 8-3 for simple and isolate type MAUs.

Fig 7-6
Interface Function for MAU with Conditioning

condition, as sent by a driver, shall be maintained for a minimum of 2 bit times. The IDL condition shall be detected within 1.6 bit times at the receiving device.

(1) System jitter considerations make detection of IDL (etd, end transmission delimiter) earlier than 1.3 bit times impractical. The specific implementation of the phase-locked loop or equivalent clock recovery mechanism determines the lower bound on the actual IDL detection time. Adequate margin between lower bound and 1.6 bit times should be considered.

(2) Recovery of timing implicit in the data is easily accomplished at the receiving side of the interface because of the wealth of binary transitions guaranteed to be in the encoded waveform, independent