American National Standard

for Information Technology –

Fibre Channel Arbitrated Loop (FC-AL-2)

BSR NCITS 332

ANSI American National Standards Institute
11 West 42nd Street
New York, New York
10036

American National Standard
for Information Technology –

Fibre Channel Arbitrated Loop (FC-AL-2)

Secretariat

Information Technology Industry Council

Approved (not yet approved)

American National Standards Institute, Inc.

Abstract

This standard defines functional requirements for an interoperable Arbitrated Loop topology to support the Fibre Channel standard.

American National Standard

Approval of an American National Standard requires review by ANSI that the requirements for due process, consensus, and other criteria for approval have been met by the standards developer.

Consensus is established when, in the judgement of the ANSI Board of Standards Review, substantial agreement has been reached by directly and materially affected interests. Substantial agreement means much more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered, and that a concerted effort be made towards their resolution.

The use of American National Standards is completely voluntary; their existence does not in any respect preclude anyone, whether he has approved the standards or not, from manufacturing, marketing, purchasing, or using products, processes, or procedures not conforming to the standards.

The American National Standards Institute does not develop standards and will in no circumstances give an interpretation of any American National Standard. Moreover, no person shall have the right or authority to issue an interpretation of an American National Standard in the name of the American National Standards Institute. Requests for interpretations should be addressed to the secretariat or sponsor whose name appears on the title page of this standard.

CAUTION NOTICE: This American National Standard may be revised or withdrawn at any time. The procedures of the American National Standards Institute require that action be taken periodically to reaffirm, revise, or withdraw this standard. Purchasers of American National Standards may receive current information on all standards by calling or writing the American National Standards Institute.

CAUTION: The developers of this standard have requested that holders of patents that may be required for the implementation of the standard disclose such patents to the publisher. However, neither the developers nor the publisher have undertaken a patent search in order to identify which, if any, patents may apply to this standard. As of the date of publication of this standard and following calls for the identification of patents that may be required for the implementation of the standard, no such claims have been made. No further patent search is conducted by the developer or publisher in respect to any standard it processes. No representation is made or implied that licenses are not required to avoid infringement in the use of this standard.

Published by

American National Standards Institute, Inc.
11 West 42nd Street, New York, NY 10036

Printed in the United States of America

Contents

Foreword (This foreword is not part of BSR NCITS 332.)

This standard defines functional requirements for an inter-operable Arbitrated Loop topology for Fibre Channel.

This standard was prepared by Task Group T11 (formerly X3T9.3) of the Accredited Standards Committee X3 during 1993. The standard process started in 1989. This document includes annexes that are informative and are not considered part of the standard.

Requests for interpretation, suggestions for improvements or addenda, or defect reports are welcome. They should be sent to the NCITS Secretariat, Information Technology Industry Council, 1250 Eye Street, NW, Suite 200, Washington, DC 20005-3922.

This standard was processed and approved for submittal to ANSI by the National Committee for Information Technology Standards (NCITS). Committee approval of the standard does not necessarily imply that all committee members voted for approval. At the time it approved this standard, NCITS had the following members:

Technical Committee T11 on Lower Level Interfaces, which developed and reviewed this standard, had the following members:

Kumar Malavalli, Chair
Edward L. Grivna, Vice-Chair
Neil Wanamaker, Secretary

David Baldwin
Paul Boulay
Joe Breher
Scott Carlson
James Coomes
Robert Cornelius
Roger Cummings
Zane Daggett
Scott Darnell
Jan V. Dedek
Don Deel
David Deming
Mark DeWilde
Schelto Van Doorn
Mike Dudek
Peter Dunlap
Mike Fitzpatrick
David Ford
Michael S. Foster
Kenneth J. Fredericks
Edward M. Frymoyer
Edward A. Gardner
Chuck Grant
Michael E. Griffin
Virginia F. Haydu
David F. Hepner
Michael Hoard
Albert F. Kelley
Robert W. Kembel
Bret Ketchum
Ronald J. Kleckowski
Dale LaFollette
Paul A. Levin
Tom Lindsay
William Lynn
William R. Martin
Gregory McSorley
Vince Melendy
Dennis P. Moore
Francis Mottini
Chris Mulvey
James Myers
Hari Naidu
J. Michael Nauman
James Nelson
Tom Palkert
Elwood Parsons
Robert K. Pedersen
Curtis A. Ridgeway
Elizabeth G. Rodriguez
Roger Ronald
Earl E. Rydell
Colin L. Schaffer
John Scheible
Pak Seto
Robert N. Snively
Jeffrey Stai
Gary R. Stephens
Arlan Stone
Rich Taborek
Fred Van Roessel
Matt Wakeley
Gary Warden
Jeffrey L. Williams
John Williams
Michael Wingard
Danny Ybarra
Leonard Young
Jeff Young
Carl Zeitler
Dal Allan (Alt.)
Rick Allison (Alt.)
Greg Alvey (Alt.)
Ravi Anantharaman (Alt.)
Charles Binford (Alt.)
Daniel Brown (Alt.)
Craig Carlson (Alt.)
Edward Chang (Alt.)
Terry Cobb (Alt.)
Bill Collette (Alt.)
Jeff Connell (Alt.)
Dave Cravens (Alt.)
Jerry D'Alessandro (Alt.)
Robert Dahlgren (Alt.)
Mike Dorsett (Alt.)
Steve Finch (Alt.)
Dave Ford (Alt.)
Ren Franse (Alt.)
Matt Gaffney (Alt.)
Dave Gampell (Alt.)
Michael Gerwig (Alt.)
Joe Golio (Alt.)
Thom Hall (Alt.)
Bill Ham (Alt.)
Daniel Heim (Alt.)
Scott Hilliker (Alt.)
Lee Hu (Alt.)
David E. Instone (Alt.)
James R. Johns (Alt.)
Skip Jones (Alt.)
Larry Jones (Alt.)
Gregory Kapraun (Alt.)
Michael J. Karg (Alt.)
Julie Ann Kembel (Alt.)
Allen N. Kramer (Alt.)
Bill Kuypers (Alt.)
Michael Lamatsch (Alt.)
Larry Lamers (Alt.)
Alan Langerman (Alt.)
Edwin S. Lee III (Alt.)
Mark Lippitt (Alt.)
Bill Mable (Alt.)
Paul Manka (Alt.)
Roland Marot (Alt.)
Bob Mayer (Alt.)
Jim McGillis (Alt.)
Brian McKean (Alt.)
Stephan Meyer (Alt.)
Gene Milligan (Alt.)
Mike Morandi (Alt.)
Eli Moyle (Alt.)
Jay H. Neer (Alt.)
Chris Nieves (Alt.)
Charles Nogales (Alt.)
John J. Nutter (Alt.)
Michael O'Donnell (Alt.)
Robert Pearson (Alt.)
George Penokie (Alt.)
David Peterson (Alt.)
Craig Prunty (Alt.)
Michael Pugh (Alt.)
Said Rahman (Alt.)
Bart Raudebaugh (Alt.)
Ron Reynolds (Alt.)
Wayne Rickard (Alt.)
Chris Simoneaux (Alt.)
Brian R. Smith (Alt.)
Bernard Warnakula Sooriya (Alt.)
Steven E. Swanson (Alt.)
Jacqueline Sylvia (Alt.)
Tad Szostak (Alt.)
Jonathan Thatcher (Alt.)
Lloyd E. Thorsbakken (Alt.)
Luis Torres (Alt.)
Kevin White (Alt.)
Lynn Whitfield (Alt.)
Steven Wilson (Alt.)
Paula Zoller (Alt.)

Introduction

This American National Standard specifies an enhancement to the signaling protocol of the Fibre Channel Physical and Signaling Interface (FC-PH), ANSI X3.230, to support communication among two or more Ports without using the Fabric topology. The following diagram shows the relationship of this document to other parts of Fibre Channel. The roadmap is intended to show the general relationship of documents to one another, not a hierarchy, protocol stack, system architecture; it does not show the complete set of Fibre Channel documents.

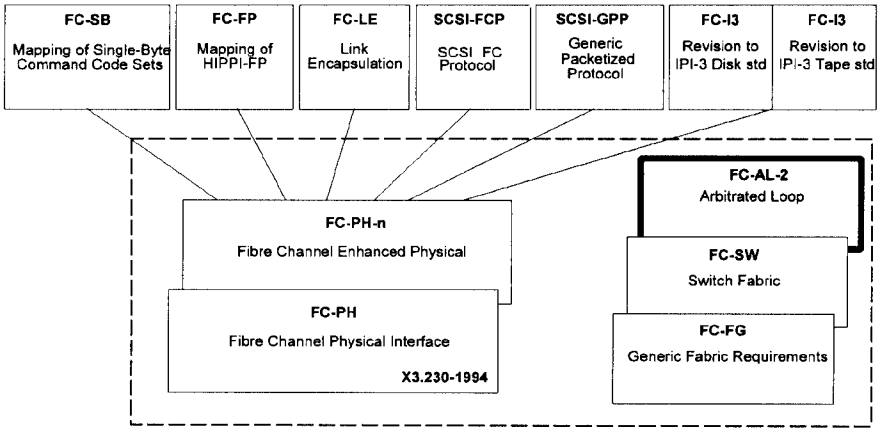

Figure 0 - Fibre Channel roadmap

FC-AL features enhanced Ports, called L_Ports, which arbitrate to access an Arbitrated Loop. Once an L_Port wins arbitration, a second L_Port may be opened to complete a single point-to-point circuit (i.e., communication path between two L_Ports). When the two connected L_Ports release control of the Arbitrated Loop, another point-to-point circuit may be established. An L_Port may have the ability to discover its environment and work properly, without outside intervention, with an F_Port, an N_Port, or with other L_Ports.

There is no change to the framing protocol of ANSI X3, FC-PH-x, however, modification to the Port hardware is required to transmit, receive, and interpret the new Arbitrated Loop Primitive Signals and Sequences. The clauses in this document are organized as follows:

Clause 1 describes the scope.

Clause 2 lists the normative references.

Clause 3 provides descriptions and conventions.

Clause 4 provides an overview and general description of FC-AL.

Clause 5 describes the Arbitrated Loop Physical Address.

Clause 6 describes the FC-AL Ordered Sets.

Clause 7 describes the Primitive Signals and Sequences.

Clause 8 describes the operation of an L_Port including the state machine.

Clause 9 provides a table representation of the FC-AL states.

Clause 10 describes the L_Port initialization procedure.

for Information Technology –

Fibre Channel – Arbitrated Loop Topology (FC-AL-2)

1 Scope

This American National Standard for FC-AL specifies signaling interface enhancements for ANSI X3, FC-PH-x to allow L_Ports to operate with an Arbitrated Loop topology. This standard defines L_Ports that retain the functionality of Ports as specified in ANSI X3, FC-PH-x. The Arbitrated Loop topology attaches multiple communicating points in a Loop without requiring switches.

The Arbitrated Loop topology is a distributed topology where each L_Port includes the minimum necessary function to establish a Loop circuit. A single FL_Port connected to an Arbitrated Loop allows multiple NL_Ports to attach to a Fabric.

When an L_Port is operating on a Loop with at least one other L_Port, the L_Port uses the protocol extensions to ANSI X3, FC-PH-x that are specified in this standard.

When an L_Port is connected with an N_Port or an F_Port, the L_Port communicates using the protocol defined in ANSI X3, FC-PH-x.[1]

Each L_Port may use a self-discovering procedure to find the correct operating mode without the need for external controls.

[1] In order to interoperate with an N_Port or an F_Port, the L_Port must have implemented the OLD-PORT state.

2 Normative references

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

2.1 Approved references

ANSI X3.272-1996, *Information Technology – Fibre Channel – Arbitrated Loop (FC-AL)**

ANSI X3.230-1994, *Information Technology – Fibre Channel – Physical and Signaling Interface (FC-PH)**

ANSI X3.297-1997, *Information Technology – Fibre Channel – Physical and Signaling Interface (FC-PH-2)**

ANSI X3.303-1998, *Information Technology – Fibre Channel – Physical and Signaling Interface (FC-PH-3)**

ANSI X3.289-1996, *Information Technology – Fibre Channel – Fabric Generic Requirements (FC-FG)**

ANSI NCITS 321-1998, *Information Technology – Fibre Channel – Switch Topologies and Switch Control (FC-SW)**

ANSI X3/TR-18-1997, *Information Technology – Fibre Channel – Private Loop Direct Attach (FC-PLDA)**

ANSI NCITS TR-20-1998, *Information Technology – Fibre Channel – Fabric Loop (FC-FLA)**

2.2 References under development

At the time of publication, the following referenced standards were still under development. For information on the current status of the documents, or regarding availability, contact the relevant standards body or other organization as indicated.

NCITS Project 1305-D, *Information Technology – Fibre Channel – Switch Topologies and Switch Control (FC-SW-2)*

NCITS Project 1315-DT, *Information Technology – Fibre Channel – Tape (FC-TAPE)*

* For electronic copies of some standards, visit ANSI's Electronic Standards Store (ESS) at www.ansi.org. For printed versions of all standards listed here, contact Global Engineering Documents, 15 Inverness Way East, Englewood, CO 80112-5704, (800) 854-7179.

3 Definitions and conventions

3.1 Definitions

For the purpose of this standard, the definitions in clause 3 of ANSI X3, FC-PH-x and the following definitions apply. Definitions in this clause take precedence over any definitions in ANSI X3, FC-PH-x.

3.1.1 Arbitrated Loop: A Fibre Channel topology where Ports use arbitration to gain access to the Loop.

3.1.2 Arbitrated Loop Physical Address (AL_PA): A unique one-byte valid value as established in 5.1.

3.1.3 Arbitrated Loop Destination Address (AL_PD): The Arbitrated Loop Physical Address of the L_Port on the Loop that should receive the Primitive Signal or Primitive Sequence. For example, the AL_PD is the y value of the OPNyx or OPNyy Primitive Signal.

3.1.4 Arbitrated Loop Source Address (AL_PS): The Arbitrated Loop Physical Address of the L_Port on the Loop that transmitted the Primitive Signal or Primitive Sequence. For example, the AL_PS is the x value of the OPNyx Primitive Signal.

3.1.5 close: A procedure used by an L_Port to terminate a Loop circuit.

3.1.6 current Fill Word: The Fill Word currently selected by the LPSM to be transmitted when needed. The initial value is the Idle Primitive Signal (see 8.4).

3.1.7 Dynamic Half-Duplex: A procedure initiated by the L_Port in the OPEN state to change a full-duplex transfer to a half-duplex transfer. The resulting half-duplex transfer is from the L_Port in the OPENED state to the L_Port in the OPEN state (see 7.5 and annex C).

3.1.8 fairness window: the period during which a fair L_Port can arbitrate and win access to the Loop only once (see 4.3).

3.1.9 Fill Word: A Transmission Word which is an Idle or an ARByx Primitive Signal. These words are transmitted between frames, Primitive Signals, and Primitive Sequences to keep a fibre active (see ANSI X3, FC-PH-x, clause 17).

3.1.10 FL_Port: An F_Port (i.e., Fabric Port) which contains the Loop Port State Machine defined by this document.

3.1.11 F/NL_Port: An NL_Port that detects OPN(00,x) and provides Fibre Channel services in the absence of an FL_Port.

3.1.12 full-duplex: Communication model 2 referred to as *duplex* in ANSI X3, FC-PH-x. Both L_Ports are allowed to transmit and receive Data frames.

3.1.13 half-duplex: Communication model 1 in ANSI X3, FC-PH-x. Only one L_Port is allowed to transmit Data frames.

3.1.14 Hub: a device for interconnecting L_Ports.

3.1.15 Loop: The Arbitrated Loop described in this document.

3.1.16 Loop circuit: A bidirectional path that allows communication between two L_Ports on the same Loop.

3.1.17 Loop Failure: Loss of word synchronization for greater than R_T_TOV; or loss of signal (see ANSI X3, FC-PH-x, 16.4.2).

3.1.18 L_Port: Either an FL_Port or an NL_Port as defined in ANSI X3, FC-PH-x, 3.1.

3.1.19 NL_Port: An N_Port (i.e., Node Port) which contains the Loop Port State Machine defined by this document. Without the qualifier "Public" or "Private," an NL_Port is assumed to be a Public NL_Port.

3.1.20 non-L_Port: A Port that does not support the Loop functions defined in this standard (see *Port* in ANSI X3, FC-PH-x).

3.1.21 Non-Participating mode: The operational mode of an L_Port which does not have an AL_PA, but is enabled into the Loop (see 8.1.4).

3.1.22 Non-Participating Bypassed mode: The operational mode of an L_Port which does not have an AL_PA and is bypassed from the Loop (see 8.1.4).

3.1.23 open: A procedure used by an L_Port to establish a Loop circuit.

3.1.24 Participating mode: The operational mode of an L_Port which has an AL_PA and is enabled into the Loop (see 8.1.4).

3.1.25 Participating Bypassed mode: The operational mode of an L_Port which has an AL_PA, but is bypassed from the Loop (see 8.1.4).

3.1.26 Port_Name: A unique 64-bit identifier as defined in the LOGI or ACC frame (see ANSI X3, FC-PH-x, 23.6.4).

3.1.27 Primitive Sequence: Three identical consecutive Ordered Sets before the function conveyed by the Primitive Sequence is performed (see ANSI X3, FC-PH-x, 16.4).

3.1.28 Private Loop: A Loop that does not include a Participating FL_Port (see figure 1 and annex J).

3.1.29 Private NL_Port: An NL_Port that does not attempt a Fabric Login and does not transmit OPN(00,x) (see figure 1 and 5.2).

3.1.30 Public Loop: A Loop that includes a Participating FL_Port and may contain both Public and Private NL_Ports (see figure 1 and annex J).

13.1.31 Public NL_Port: An NL_Port that attempts a Fabric Login (see figure 1 and 5.2).

3.1.32 replicate frame: A Class 3 frame which may be received and processed by one or more NL_Ports while being forwarded (see 7.3).

3.1.33 transfer: A procedure used by an L_Port to close an existing Loop circuit in order to establish a new Loop circuit without relinquishing control of the Loop.

3.1.34 trusted AL_PA: an AL_PA which is assumed to be valid and unique through a vendor-specified means (e.g., a Hard Address).

3.2 Editorial conventions

In FC-AL, many conditions, mechanisms, sequences, events or similar terms are printed with the first letter of each word in upper case and the rest lower case (e.g., Loop). States are defined in all upper case letters. Any lower case words not defined in 3.1 have the normal technical English meaning.

In case of conflicts between text, tables, and figures, the following precedence shall be used: text, tables, figures. State diagrams have precedence as stated in the appropriate clauses.

The word, *shall*, when used in this standard, states a mandatory rule or requirement. The word, *may*, when used in this standard, states an optional rule. The word, *should*, when used in this standard denotes flexibility of choice with a strongly preferred alternative (equivalent to the phrase 'is recommended').

The words, *recognize, recognizes, or recognized*, when used in this standard, indicates that an L_Port has detected a Primitive Signal or Primitive Sequence.

Each individual entry that appears within parentheses of FC-AL Ordered Sets (e.g., ARByx and OPNy) represents the hexadecimal value of an AL_PA or special flags (e.g., hex 'F7', hex 'F8', and hex 'FF').

All history variables, when used in this standard, are assumed to be set to zero (0) at power-on time. Any optional history variable that is not implemented tests as zero (0) in all tests of that variable.

The ISO convention of numbering is used; i.e., the thousands and higher multiples are separated by a space and a comma is used instead of the decimal point (e.g., 1 062,5 Mbits/sec).

Whenever ANSI X3, FC-PH-x is referenced, all ANSI X3, FC-PH documents referenced in clause 2 are implied.

3.3 Abbreviations, acronyms, and other special words

ACCESS **Access** history variable – a two-valued variable (i.e., *0/1*) to indicate access fairness history (see 8.1.1).

AL_PA **A**rbitrated **L**oop **P**hysical **A**ddress (see 5.1)

AL_PD **A**rbitrated **L**oop **D**estination Physical Address (e.g., the y value in OPNyx and OPNyy)

AL_PS **A**rbitrated **L**oop **S**ource Physical Address (e.g., the x value in OPNyx)

AL_TIME **A**rbitrated **L**oop **time**out value (see 8.2.2)

ARB_PEND **Arb**itration **PEND**ing history variable – a two-valued variable (i.e., 0/1) to help an L_Port remember that it has originated one or more ARB(AL_PA) Primitive Signals (see 8.1.1).

ARB_WON **Arb**itration **Won** history variable – a two-valued variable (i.e., *0/1*) to remember whether the L_Port won arbitration (see 8.1.1).

ARB(AL_PA) **Arb**itrate Primitive Signal – an ARByx in which y = x = AL_PA of the L_Port (see 7.1.1)

ARB(val) **Arb**itrate Primitive Signal – an ARByx in which y = x = val ('val' represents a one byte hexadecimal value).

ARByx **Arb**itrate Primitive Signal – any Ordered Set that begins with K28.5, D20.4 ('y' and 'x' may be used in adjacent text to denote the value of characters 3 and 4 within the Ordered Set) (see 7.1).

ARBf_SENT **Arb**itrate hex 'FF' **Sent** history variable – a two-valued variable (i.e., *0/1*) that indicates that the L_Port has requested REQ(arbitrate (FF)) and the LPSM has modified the current Fill Word to ARB(FF) (see 8.1.7 and 8.4.3).

Available_BB_Credit **Available** **B**uffer-to-**B**uffer **Credit** (see 8.3.4)

BB_Credit **B**uffer-to-**B**uffer **Credit** as established during Login (see 8.3.4 and ANSI X3, FC-PH-x, 26.5.2)

BYPASS **BYPASS** history variable – a two-valued variable (i.e., *0/1*) which indicates whether an L_Port is bypassed (see 8.1.4).

CLS **CL**o**S**e Primitive Signal (see 7.4)

CFW **C**urrent **F**ill **W**ord (i.e., Idle or ARByx) (see 3.1.10 and 7.1)

DHD **D**ynamic **H**alf-**D**uplex Primitive Signal (see 7.5)

DHD_RCV **D**ynamic **H**alf-**D**uplex **ReCeiV**ed history variable – a two valued variable (i.e., *0/1*) to indicate that the L_Port in the OPENED state has detected and supports DHD (see 8.1.5).

DUPLEX **DUPLEX** history variable – a two-valued variable (i.e., *0/1*) to indicate whether the L_Port is allowed to originate Data frames (see 8.1.2).

EE_Credit **E**nd-to-**E**nd **Credit** (see ANSI X3, FC-PH-x, 26.4.4)

ERR_INIT **ERR**or **INIT**ialization history variable – a two-valued variable (i.e., *0/1*) to indicate that the L_Port has attempted initialization which failed (see 8.1.6).

LIFA **L**oop **I**nitialization **F**abric **A**ssigned – Loop Initialization Sequence (see 10.5)

LIHA **L**oop **I**nitialization **H**ard **A**ssigned – Loop Initialization Sequence (see 10.5)

LILP **L**oop **I**nitialization **L**oop **P**osition – Loop Initialization Sequence (see 10.5)

LIM **L**oop **I**nitialization **M**aster – the L_Port which is responsible for initializing the Loop (see clause 10)

LIP **L**oop **I**nitialization **P**rimitive Sequence – any of the LIP Primitive Sequences (see 7.8)

LIPfx **L**oop **I**nitialization **P**rimitive Sequence – perform a vendor unique reset of all (except AL_PA = x) L_Ports (f = hex 'FF') (see 7.8.5)

LIPA **L**oop **I**nitialization **P**reviously **A**cquired – Loop Initialization Sequence (see 10.5)

LIPyx **L**oop **I**nitialization **P**rimitive Sequence – perform a vendor unique reset of an L_Port at AL_PA = y (see 7.8.5)

LIRP **L**oop **I**nitialization **R**eport **P**osition – Loop Initialization Sequence (see 10.5)

LISA **L**oop **I**nitialization **S**oft **A**ssigned – Loop Initialization Sequence (see 10.5)

LISM **L**oop **I**nitialization **S**elect **M**aster – Loop Initialization Sequence (see 10.5)

LI_FL **L**oop **I**nitialization **FL**ag – Loop Initialization flag (see 10.5)

LI_ID **L**oop **I**nitialization **ID**entifier – Loop Initialization identifier (see 10.5)

LP_TOV **LooP** **T**ime**O**ut **V**alue (see 8.2.3)

LPB **L**oop **P**ort **B**ypass Primitive Sequence – either LPByx or LPBfx (f = hex 'FF') (see 7.7.1 and 7.7.2)

LPBfx **L**oop **P**ort **B**ypass Primitive Sequence – used to bypass all (except AL_PA = x) L_Ports (f = hex 'FF') (see 7.7.2)

LPByx **L**oop **P**ort **B**ypass Primitive Sequence – used to bypass an L_Port at y = AL_PA (see 7.7.1)

LPE **L**oop **P**ort **E**nable Primitive Sequence – either LPEyx or LPEfx (see 7.7.3 and 7.7.4)

LPEfx **L**oop **P**ort **E**nable Primitive Sequence – used to enable all bypassed L_Ports (f = hex 'FF') (see 7.7.4)

LPEyx **L**oop **P**ort **E**nable Primitive Sequence – used to enable a bypassed L_Port at y = AL_PA (see 7.7.3)

LPSM	**L**oop **P**ort **S**tate **M**achine (see 8.4)
MRKtx	**Mark** Primitive Signal (see 7.6)
MK_TP	**Mark Type** – used to identify the type of Mark Primitive Signal (see 7.6)
(Non F8) LIP	Any Loop Initialization Primitive Sequence as defined in 7.8, where character 3 is not equal to hex 'F8'.
OPNr	**Open** Replicate Primitive Signal – either OPNyr or OPNfr (see 7.3)
OPNfr	**Open** Primitive Signal – broadcast replicate(see 7.3.2)
OPNy	**Open** Primitive Signal – either OPNyx or OPNyy (see 7.2)
OPNyr	**Open** Primitive Signal – selective replicate(see 7.3.1)
OPNyx	**Open** Primitive Signal – full-duplex (see 7.2.1)
OPNyy	**Open** Primitive Signal – half-duplex (see 7.2.2)
PARTICIPATE	**PARTICIPATE** history variable – a two-valued variable (i.e., *0/1*) that indicates whether an L_Port has an AL_PA and is participating on the Loop (see 8.1.4).
REPEAT	A symbol whose value is derived from BYPASS and PARTICIPATE, which indicates that an L_Port merely repeats received Transmission Words (see 8.1.4).
REPLICATE	**Replicate** history variable – a two valued variable (i.e., *0/1*) to indicate if an L_Port has transmitted OPNr while in the OPEN state or an NL_Port has received OPNr while in the MONITORING or ARBITRATING states (see 8.1.3).
SOFiL	**S**tart_**o**f_**F**rame Primitive Signal (K28.5 D21.5 D22.2 D22.2) used during Loop Initialization (see 10.5).
XMIT_2_IDLES	**X**mit **2** Idles history variable – a two-valued variable (i.e., *0/1*) that indicates whether the L_Port needs to transmit two (2) Idles (see 8.1.1).

3.4 Symbols

Logic symbols are represented in state tables and diagrams as follows:

— the logical 'or' is represented as '|';

— the logical 'and' is represented as '&';

— the logical negation is represented as '~' (tilda);

— the 'less-than' is represented as '<';

— the 'greater-than' is represented as '>';

— comparisons are represented as '=' (equal) and '<>' (not equal);

— setting a variable is done using the colon equal operator, ':='; and,

— the concatenation symbol is represented as '||'.

4 Structure and concepts

This clause provides an overview of the structure, concepts, and mechanisms that allow two or more L_Ports to communicate without using a Fabric topology. Readers unfamiliar with FC-AL should read or scan clauses 1 and 4 before attempting to master the detailed material in clauses 5 through 10.

4.1 Overview

FC-AL is a serial data channel, structured for low-cost connectivity, that provides a logical bidirectional, point-to-point service between two L_Ports. Each L_Port represents a communication point. The additional functions, that are added to allow an N_Port or F_Port to operate on a Loop, permit the L_Ports to form a simple, blocking, non-meshed, switching environment.

— Blocking refers to the number of circuits that can be concurrently active. Only one pair of L_Ports may communicate at one time although there may be up to 127 Participating L_Ports attached on one Loop. All other communication must wait (i.e., is blocked).

— Non-meshed refers to the attribute of a Loop where there is exactly one path on each Loop between L_Ports. Non-meshed implies that any single fibre problem may stop all activity on the Loop. Meshed, in this context, means that there may be alternate paths available between L_Ports.

— Switching refers to the Loop circuitry added to each L_Port compared to a non-L_Port. The circuitry acts as a two-port switch where information received on the inbound fibre of the L_Port is directed to either the local FC-2 function or placed on the outbound fibre for another L_Port to process.

The Loop supports a maximum of one point-to-point circuit at a time. When two L_Ports are communicating, the Loop topology supports simultaneous, symmetrical, bidirectional flow between the two L_Ports. All other L_Ports are monitoring or arbitrating for access to the Loop.

The Loop supports all Classes of Service as specified in ANSI X3, FC-PH-x, 4.9. Individual L_Ports may choose to implement, at the FC-2 level, only a subset of the Classes of Service which are available. Such an implementation does not affect the operation of the Loop protocol.

The Loop guarantees in-order delivery of frames in all Classes of Service when the source and destination are on the same Loop. Frames transmitted from an NL_Port to an FL_Port are received at the FL_Port in the transmitted order; frames transmitted from an FL_Port are received at the NL_Port in the transmitted order. Out-of-order frames may be received at a destination NL_Port, but that out-of-order characteristic is not caused by the FL_Port or the Loop.

Unlike the Fabric topology where a circuit is established only for a dedicated connection or virtual circuit, a Loop circuit must be established between two L_Ports on the Loop before the FC-PH framing protocol may be used. The two L_Ports may use the framing protocol and any Class of Service appropriate for their implementations and for the FC-4 protocol being used. Other L_Ports on the same Loop may form their own Loop circuit after the current Loop circuit is closed.

4.2 General description

In a Fabric topology, one or more Fabric Element(s) is required to connect more than two Ports together (see ANSI X3.289, FC-FG, for the minimum requirements of a Fabric).

The Loop topology reduces the number of transceivers required to interconnect L_Ports to one transceiver per L_Port. Up to 127 L_Ports may be in the Participating mode on one Loop.

The different topologies which are defined in ANSI X3, FC-PH-x have certain pertinent distinguishing characteristics.

— The **point-to-point topology** is non-blocking. Each N_Port may transmit frames to the other at any time within the limits of the implemented protocols of the N_Ports.

  The number of transceivers needed to completely interconnect *n* N_Ports using multiple links may be calculated using the formula: $t = n (n - 1)$ where *t* and *n* represent the number of transceivers and N_Ports, respectively. For example, to connect six (6) N_Ports requires 30 transceivers.

  Also, if a link in a point-to-point topology fails, communication between that pair of Ports stops. Communication between other point-to-point connected Ports continues.

— The **Fabric topology** may be configured to be non-blocking between any two N_Ports. It is commonly acknowledged that most data processing-type Nodes cannot sustain high-speed data transfer for long periods of time to all peripheral devices (although there may be some exceptions). A Fabric offers a way to take advantage of these natural pauses in communication, allowing fewer interconnects. The available bandwidth is shared between the N_Ports, but this sharing adds contention and therefore a management function is required.

  One advantage for the Fabric topology is that when there is at least one free F_Port in the Fabric, a new N_Port can be added to the free F_Port without disrupting the remaining N_Ports. The new N_Port has the potential to communicate with all other N_Ports in the Fabric. However, adding an N_Port does not guarantee that the new N_Port can communicate with any of the currently attached N_Ports (see ANSI X3.289, FC-FG).

  Because a Fabric topology may permit multiple paths between any two F_Ports in the Fabric (i.e., the meshing capability of the Fabric topology), a Fabric topology may be more robust. For a Node with only one N_Port, there is always a single point of failure at the link to the Fabric Element.

— The **Loop topology** functions are the result of reducing the Fabric topology to its simplest form. There is exactly one link bandwidth to share among all L_Ports. This makes the Loop the ultimate blocking topology, yet it retains considerable connectivity. There can be only one active Loop circuit at a time, independent of the number of L_Ports on a Loop. New L_Ports can be added at any time, although only a maximum of 127 may be participating. Should any link in a Loop fail, communication between all L_Ports stops on that Loop.

  Fabric management is reduced to a minimum with the remaining functions distributed in each L_Port on the Loop. This eliminates the central management function of a Fabric and at least one-half of the transceivers compared to a Fabric topology. Once communication is established between two L_Ports, the normal ANSI X3, FC-PH-x protocol is used for all operations.

Figure 1 shows two independent Loop configurations each with multiple L_Ports connected. Each line in the figure between L_Ports represents a single fibre. The configuration in figure 1(a) shows two Loops: one includes two NL_Ports (i.e., point-to-point) and the other includes six NL_Ports (i.e., Private Loops). The configuration in figure 1(b) shows a Loop which includes one FL_Port (i.e., a Public Loop) and five NL_Ports (either Public or Private NL_Ports). (See annex J.)

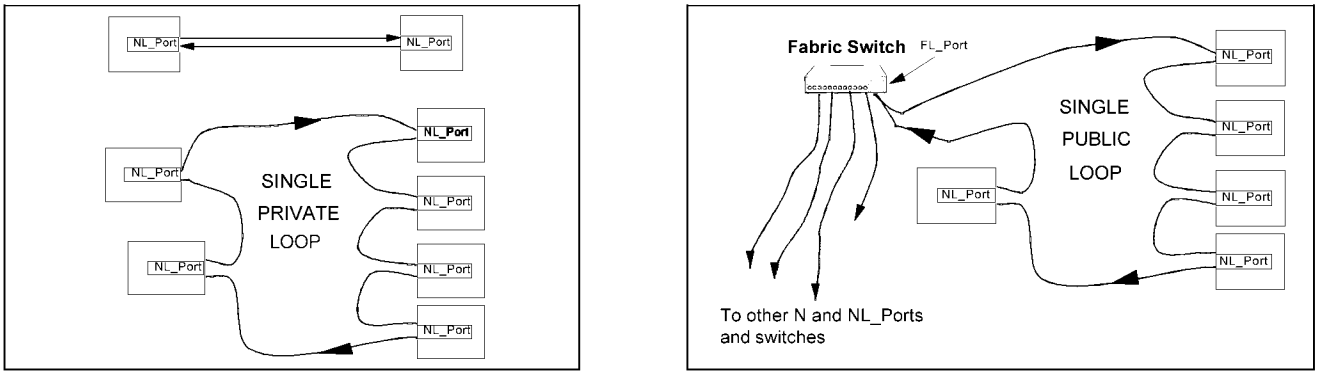

Figure 1 — Examples of the Loop topology

The Loop topology and the Fabric topology together provide a compromise between connectivity and performance. A number of Loops may be connected through a Fabric. For example, four sixteen-port Loops (one FL_Port and fifteen NL_Ports) may be connected through a four-port Fabric to achieve a connectivity of sixty L_Ports with better performance than if all sixty NL_Ports were on one Loop.

4.3 Access fairness algorithm

The protocol for the Loop permits each L_Port to continuously arbitrate to access the Loop. A priority is assigned to each Participating L_Port based on the Arbitrated Loop Physical Address (AL_PA). As with other prioritized protocols, this could lead to situations where the lower priority L_Ports cannot gain access to the Loop. The access fairness algorithm sets up an access window in which all L_Ports are given an opportunity to arbitrate and win access to the Loop. When all L_Ports have had an opportunity to access the Loop once, a new access window is started. An L_Port may arbitrate again and eventually win access to the Loop in the new access window. Not every L_Port is required to access the Loop in any one access window.

When an L_Port which uses the access fairness algorithm has arbitrated for and won access to the Loop, the L_Port shall not arbitrate again until at least two (2) Idles have been transmitted by the L_Port. The access window is defined as the time period between when the no L_Ports are arbitrating until all L_Ports requesting to arbitrate have won arbitration. An access window may vary in size depending on the number of arbitrating L_Ports. A special arbitration Primitive Signal (i.e., ARB(F0)) is used as the Fill Word during this interval to prevent an early reset of the access window. The details of the access fairness algorithm are contained in the Loop state machine (see 8.4).

The access fairness algorithm does not limit the time that an L_Port controls the Loop once it wins arbitration, just as ANSI X3, FC-PH-x does not limit the time for a Class 1 connection. However, if access is denied longer than LP_TOV, the access window is reset and an L_Port may begin arbitrating.

All L_Ports shall implement the access fairness algorithm, FL_Ports or NL_Ports are not required to use the access fairness algorithm nor use it consistently. For example, if one L_Port requires more Loop accesses than the other L_Ports, that L_Port may choose to be unfair. Although, the standard encourages all L_Ports to use the access fairness algorithm, the decision when to be fair or unfair is beyond the scope of this standard (see annex D).

4.3.1 Access fairness for NL_Ports

To provide equal access to the Loop for all NL_Ports, it is recommended that each NL_Port use the access fairness algorithm. When an NL_Port is using the access fairness algorithm, it is called a *fair* NL_Port.

When a fair NL_Port has access to the Loop and detects that another L_Port is arbitrating, the fair NL_Port should close the Loop at the earliest possible time and arbitrate again in the next access window.

4.3.2 Access unfairness for NL_Ports

The configuration of some Loops may require that certain NL_Ports have more access to the Loop than just once per access window. Examples of these NL_Ports include, but are not limited to, a subsystem controller or a file server.

An NL_Port may be initialized (or may temporarily choose) not to use the access fairness algorithm. When an NL_Port is not using the access fairness algorithm, it is called an *unfair* NL_Port. The decision whether to use the access fairness algorithm is beyond the scope of this standard. (See annex D.)

When an unfair NL_Port has arbitrated for and won access to the Loop and does not detect that another L_Port is arbitrating, that NL_Port may keep the existing Loop circuit open indefinitely or the L_Port may use the transfer procedure to open another L_Port on the Loop.

When an unfair NL_Port controls the Loop and detects that another L_Port is arbitrating, the unfair NL_Port may close the Loop, keep the existing Loop circuit open, or it may use the transfer procedure to open another L_Port on the Loop.

4.3.3 Access unfairness for FL_Ports

A Participating FL_Port is always the highest priority L_Port on the Loop based on its AL_PA. An FL_Port is encouraged to use the access fairness algorithm, but it may choose to be unfair since the majority of its traffic is with the rest of the Fabric. If the FL_Port were required to use the fairness algorithm at all times, it would be more likely to fill buffers in the Fabric, causing non-Loop communications to be affected.

When an FL_Port controls the Loop and detects that an NL_Port is arbitrating, the FL_Port may close the Loop, keep the existing Loop circuit open, or it may use the transfer procedure to open another NL_Port on the Loop.

4.4 Relationship to ANSI X3, FC-PH-x

If a Port uses FC-AL, it extends the FC-2 and FC-1 functions of ANSI X3, FC-PH-x. Figure 2 shows logically where the Loop (FC-AL) function is located. This functional level does not have a formally defined interface to the other levels.

Figure 2 — FC-PH with Arbitrated Loop addition

When two L_Ports are communicating, the L_Ports may use all of the functions specified in ANSI X3, FC-PH-x. The following list is a clause-by-clause analysis of the differences between N_Ports or F_Ports and NL_Ports or FL_Ports, respectively. The Loop:

— supports communication models 1 and 2 identified in ANSI X3, FC-PH-x, 4.6, but it does not support model 3. Any two L_Ports may operate in half-duplex mode during one Loop circuit. The direction of the half-duplex mode may be changed by establishing a new Loop circuit in the opposite direction;

— adds new error detection or recovery protocols in 8.3 in addition to those identified in ANSI X3, FC-PH-x, 4.14, and related clauses;

— places no limit on the use of any one type of transmitter (although they shall all be of the same data rate) for the cable plant of a Loop. Some requirements (e.g., Open Fibre Control) may prevent interoperability when mixed on a single Loop. (See ANSI X3, FC-PH-x, clauses 5 through 10);

— specifies that all NL_Ports and the optional FL_Port on a Loop shall use the same data rate. (See ANSI X3, FC-PH-x, clause 5);

— the ANSI X3, FC-PH-x buffer-to-buffer flow control is not used for L_Ports that are monitoring the Loop. (See 8.3.4);

— expands the number of Ordered Sets beyond those specified in ANSI X3, FC-PH-x, clause 11. (See clause 6);

— expands the number of Primitive Signals and Sequences beyond those specified in ANSI X3, FC-PH-x, clause 16. (See clause 7);

— extends the Ordered Sets that may be deleted to include Idle, ARByx, and all Primitive Sequences. (See 8.3.2);

— specifies the Primitive Signals that may be inserted on a Loop between frames for clock skew management. (See 8.3.2);

— modifies an F_Port behavior to allow clock skew management by L_Ports on a Loop. An FL_Port in the OPEN, OPENED, or RECEIVED CLOSE state shall originate at least six (6) Primitive Signals between Class 2 or Class 3 frames. In a Class 1 connection, the clock skew needs to be managed between the two NL_Ports (i.e., from one end of the circuit to the other end);

— defines a local physical address and native address identifier assignment algorithms when an FL_Port is not present on a Loop. (See clause 10);

— requires a minimum payload size of 132 bytes for Loop Initialization. (See 10.5);

— permits an L_Port to manage a separate BB_Credit for each L_Port on the Loop or the L_Port may choose to use a single value for BB_Credit. The single value shall be between zero (0) and the minimum value for all L_Ports;

— requires that the L_Port set the "Alternate BB_Credit Management" bit to *1* in the N_Port Common Service Parameters during Login. (See ANSI X3, FC-PH-x, 23.6.3 and 26.5);

— permits a Loop circuit to be terminated when Available_BB_Credit is unbalanced;

— requires that each L_Port is capable of mapping the S_ID in each frame it receives to the AL_PA of the L_Port that transmitted this frame;

— requires that the destination of a connect request (SOFc1) sent through an FL_Port is to a Port not on the Loop; the FL_Port is not able to open another NL_Port on the same Loop (this would require three open L_Ports);

— requires Loop Initialization Sequences to be used during the initialization procedure; and,

— allows an NL_Port (in the absence of an FL_Port) to act as an F/NL_Port. The F/NL_Port shall provide the Fabric Login service associated with well-known address identifier hex 'FFFFFE'. The F/NL_Port may also provide services associated with other well-known address identifiers.

When a Loop circuit has been established between two L_Ports (i.e., an FL_Port to an NL_Port or an NL_Port to an NL_Port) (see ANSI X3, FC-PH-x, clause 26 for flow control), FC-2 uses:

— the point-to-point topology model, when both communicating NL_Ports are on the same Loop; or,

— the Fabric topology model, when one communicating Port is outside the Loop.

5 Addressing

5.1 Arbitrated Loop Physical Address (AL_PA)

Each L_Port (if it chooses to participate on the Loop, see 8.1.4) shall be assigned a local Arbitrated Loop Physical Address (AL_PA). The AL_PA establishes the priority of an arbitrating L_Port (i.e., the lower the AL_PA, the higher the priority).

Each L_Port shall use an AL_PA value that results in neutral disparity. (See ANSI X3, FC-PH-x, clause 11). The algorithm described below or in table 1 provides a means for the L_Port to select an AL_PA.

The AL_PA shall be a valid data character as specified in ANSI X3, FC-PH-x, clause 11 that does not change the current running disparity of a Transmission Word. The algorithm below is dependent on the FC-1 naming convention for an information byte in ANSI X3, FC-PH-x, 11.1 and table 26, identified as Dxx.y. The xx portion of the FC-1 naming convention is based on bits identified as E, D, C, B, and A in ANSI X3, FC-PH-x, 11.1, in that order. The y portion of the FC-1 naming convention is based on bits identified as H, G, and F in ANSI X3, FC-PH-x, 11.1, in that order. A decimal value is assigned to each bit combination with the range of 0 to 31 for xx and 0 to 7 for y, respectively. The entire range for valid data characters using the FC-1 naming convention is D00.0 through D31.7.

Disparity for a valid data character is calculated as follows:

— arrange an information byte in the manner prescribed for the naming convention in ANSI X3, FC-PH-x, 11.1, to obtain the Dxx.y data byte name;

— if the xx portion of a valid data character is (in decimal) 0, 1, 2, 4, 8, 15, 16, 23, 24, 27, 29, 30, or 31, set HI to 1. If the xx portion is (in decimal) 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 17, 18, 19, 20, 21, 22, 25, 26, or 28, set HI to 0;

— if the y portion of a valid data character is (in decimal) 0, 4, or 7, set LO to 1. If the y portion is (in decimal) 1, 2, 3, 5, or 6, set LO to 0; and,

— compute the XOR function for HI and LO (i.e., XOR(HI,LO)).

If the computed value of the XOR function is 0, the value is disparity neutral and is a valid AL_PA. If the computed value of the XOR function is 1, the value is disparity biased and the FC-2 byte is not a valid AL_PA.

Table 1 identifies with an asterisk (*) each 8B/10B character that has neutral disparity ordered by the Dxx.y naming convention. The right-most column shows the FC-2 byte notation values for each row with neutral disparity in table 1.

Table 1 — 8B/10B characters with neutral disparity

D xx . y	y								Hex value
xx	0	1	2	3	4	5	6	7	
00	*				*			*	00, 80, E0
01	*				*			*	01, 81, E1
02	*				*			*	02, 82, E2
03		*	*	*		*	*		23, 43, 63, A3, C3
04	*				*			*	04, 84, E4
05		*	*	*		*	*		25, 45, 65, A5, C5
06		*	*	*		*	*		26, 46, 66, A6, C6
07		*	*	*		*	*		27, 47, 67, A7, C7
08	*				*			*	08, 88, E8
09		*	*	*		*	*		29, 49, 69, A9, C9
10		*	*	*		*	*		2A, 4A, 6A, AA, CA
11		*	*	*		*	*		2B, 4B, 6B, AB, CB
12		*	*	*		*	*		2C, 4C, 6C, AC, CC
13		*	*	*		*	*		2D, 4D, 6D, AD, CD
14		*	*	*		*	*		2E, 4E, 6E, AE, CE
15	*				*			*	0F, 8F, EF
16	*				*			*	10, 90, F0
17		*	*	*		*	*		31, 51, 71, B1, D1
18		*	*	*		*	*		32, 52, 72, B2, D2
19		*	*	*		*	*		33, 53, 73, B3, D3
20		*	*	*		*	*		34, 54, 74, B4, D4
21		*	*	*		*	*		35, 55, 75, B5, D5
22		*	*	*		*	*		36, 56, 76, B6, D6
23	*				*			*	17, 97, F7
24	*				*			*	18, 98, F8
25		*	*	*		*	*		39, 59, 79, B9, D9
26		*	*	*		*	*		3A, 5A, 7A, BA, DA
27	*				*			*	1B, 9B, FB
28		*	*	*		*	*		3C, 5C, 7C, BC, DC
29	*				*			*	1D, 9D, FD
30	*				*			*	1E, 9E, FE
31	*				*			*	1F, 9F, FF
TOTAL 134	13	19	19	19	13	19	19	13	
Legend: * — character with neutral disparity									

5.1.1 Valid AL_PAs

The following valid AL_PAs are assigned to the first 127 neutral disparity values from table 1:

hex '00': (1) AL_PA for FL_Port or alias AL_PA of F/NL_Port

Highest priority.

AL_PA hex '00' shall be assigned to the FL_Port in the Participating mode. The maximum number of FL_Ports in the Participating mode on a single Loop shall not exceed one. Additional FL_Ports may be present, but they shall be in the Non-Participating or Non-Participating Bypassed mode.

If there is no Participating FL_Port on the Loop, a Participating NL_Port may accept this value as an alias AL_PA for its LPSM, but not as its only AL_PA.

hex '01' through hex 'EF': (126) AL_PA for NL_Ports

Descending priority is assigned as AL_PA values increase in the range from hex '01' through hex 'EF'. All valid values in this range are lower in priority than hex '00'.

Each Participating NL_Port shall be assigned one valid AL_PA in this range. The maximum number of Participating NL_Ports on a single Loop shall not exceed 126.

5.1.2 Special AL_PAs and flags

The following special AL_PAs and flags are assigned to the last 7 neutral disparity values from table 1. These values replace an AL_PA to provide special functions:

hex 'F0': (1) Value used in ARB for fairness and during Loop Initialization. Value hex 'F0' is the next lower priority outside the range hex '00' through hex 'EF'.

hex 'F1' through hex 'F6': (0) Reserved

hex 'F7': (1) Value used in LIP to indicate that the L_Port is initializing and also a value for an L_Port which does not have an AL_PA.

hex 'F8': (1) Value used in LIP to indicate a Loop Failure has been detected at the receiver of the L_Port.

hex 'F9' through hex 'FE': (3) Reserved

hex 'FF': (1) Value used to address all L_Ports (except the originating L_Port) in OPNfr, LPBfx, LPEfx, and LIPfx and as a special ARB(FF) Fill Word.

5.2 Native address identifier

A native address identifier shall be assigned to each Participating NL_Port (up to the maximum of 126 NL_Ports) with the following characteristics:

— the low-order byte (bits 7-0) of the native address identifier is the AL_PA of the L_Port. The AL_PA shall be unique on a Loop, shall be in the range of hex '01' through hex 'EF', and shall be valid according to table 1.

— all Private NL_Ports shall have the upper two bytes of their native address identifier (bits 23-8) equal to hex '0000'.

— all Public NL_Ports shall have the upper two bytes of their native address identifier (bits 23-8) equal to the upper two bytes of the native address identifier of the FL_Port (hex 'XXXX00'). The FL_Port shall acquire this value (which shall not equal hex '000000') from its Fabric Element. The upper two bytes shall be unique for each Loop to allow multiple Loops to attach to the same Fabric.

  All Public NL_Port shall process PLOGI, LOGO, and ABTS frames with a D_ID of hex '0000'||AL_PA or hex 'XXXX'||AL_PA.

  If an FL_Port is not present, these upper two bytes shall be set to zero (hex '0000').

— a native address identifier may be assigned to another NL_Port if a Participating NL_Port enters the Non-Participating mode.

6 FC-AL Ordered Sets

Table 2 specifies the Ordered Sets that shall be detected and may be originated by L_Ports on the Loop as additional Primitive Signals (see ANSI X3, FC-PH-x, 11.4). Table 3 specifies the Ordered Sets that shall be recognized and may be originated by L_Ports on the Loop as additional Primitive Sequences (see clause 7).

Table 2 — Primitive Signals

Primitive Signal		Beginning RD	Ordered Set 1	2	3	4
Arbitrate	ARByx	Negative	K28.5	D20.4	y	x
Arbitrate	ARB(val)	Negative	K28.5	D20.4	val	val
Close	CLS	Negative	K28.5	D5.4	D21.5	D21.5
Dynamic Half-Duplex	DHD	Negative	K28.5	D10.4	D21.5	D21.5
Mark	MRKtx	Negative	K28.5	D31.2	MK_TP	AL_PS
Open full-duplex	OPNyx	Negative	K28.5	D17.4	AL_PD	AL_PS
Open half-duplex	OPNyy	Negative	K28.5	D17.4	AL_PD	AL_PD
Open selective replicate	OPNyr	Negative	K28.5	D17.4	AL_PD	D31.7
Open broadcast replicate	OPNfr	Negative	K28.5	D17.4	D31.7	D31.7

Characters 3 and 4 of Ordered Sets ARByx, ARB(val), MRKtx, OPNyx, OPNyy, OPNyr, and OPNfr shall contain one of the neutral disparity values from table 1, whenever such an Ordered Set is originated (see 7.1 to 7.6).

Table 3 — Primitive Sequences

Primitive Sequence		Beginning RD	Ordered Set 1	2	3	4
Loop Initialization--F7,F7	LIP(F7,F7)	Negative	K28.5	D21.0	D23.7	D23.7
Loop Initialization--F8,F7	LIP(F8,F7)	Negative	K28.5	D21.0	D24.7	D23.7
Loop Initialization--F7,x	LIP(F7,x)	Negative	K28.5	D21.0	D23.7	AL_PS
Loop Initialization--F8,x	LIP(F8,x)	Negative	K28.5	D21.0	D24.7	AL_PS
Loop Initialization--reset	LIPyx	Negative	K28.5	D21.0	AL_PD	AL_PS
Loop Initialization--reset all	LIPfx	Negative	K28.5	D21.0	D31.7	AL_PS
Loop Initialization--reserved	LIPba	Negative	K28.5	D21.0	b	a
Loop Port Bypass	LPByx	Negative	K28.5	D9.0	AL_PD	AL_PS
Loop Port Bypass all	LPBfx	Negative	K28.5	D9.0	D31.7	AL_PS
Loop Port Enable	LPEyx	Negative	K28.5	D5.0	AL_PD	AL_PS
Loop Port Enable all	LPEfx	Negative	K28.5	D5.0	D31.7	AL_PS

AL_PD and AL_PS characters of the above Ordered Sets shall contain one of the neutral disparity values from table 1, whenever such an Ordered Set is originated (see 7.7 to 7.8).
The b and a characters of the above Ordered Sets shall contain one of the neutral disparity values from table 1, whenever such an Ordered Set is originated (see 7.7 to 7.8).

7 FC-AL Primitive Signals and Sequences

The Arbitrate Primitive Signal (ARByx) may be transmitted in place of an Idle and therefore becomes a Fill Word which may be removed for clock skew management. The Mark Primitive Signal (MRKtx) may also be transmitted in place of a Fill Word, but it shall not be removed for clock skew management. All Primitive Signals (except MRKtx and Fill Words) defined in this standard shall follow the ANSI X3, FC-PH-x rule for transmitting R_RDYs (i.e., two (2) Fill Words shall precede and follow these Primitive Signals with at least six (6) Primitive Signals between frames). (See ANSI X3, FC-PH-x, 16.3.2 and clause 6 for a specification of the following Ordered Sets.)

Except as specifically described, the LPSM shall fully decode (test the value of all bits in all four characters of) received Ordered Sets to detect reception of a Primitive Signal or Primitive Sequence. Any received Ordered Set that does not fully match one defined in this standard or in ANSI X3, FC-PH-x shall be treated as an 'Other Ordered Set.'

7.1 Arbitrate Primitive Signals (ARByx)

A received Ordered Set shall be detected as an Arbitrate Primitive Signal (ARByx) by detecting that its first two characters (fully decoded) are equal to the value shown in table 2, regardless of the value of characters 3 and 4 (y and x). L_Ports shall only originate an Arbitrate Primitive Signal (ARByx) where y = x. All Arbitrate Primitive Signals shall be treated as Fill Words for clock skew management. An Arbitrate Primitive Signal is further classified by examining the values of y and x. The values of y and x used, shall be the data bytes resulting from 8B/10B decoding.

If the values of y and x are equal, a received Arbitrate Primitive Signal shall be detected as an ARB(val), where 'val' is the value of y or x. If the values of y and x are different, a received Arbitrate Primitive Signal should not be recognized as an ARB(val).[2]

7.1.1 ARB(AL_PA)

ARB(AL_PA) is an ARB(val) which is transmitted on a Loop by a Participating L_Port to request access to the Loop. Each ARB(AL_PA) shall contain the AL_PA of the L_Port making the request (REQ(arbitrate own AL_PA)).

7.1.2 ARB(F0)

ARB(F0) is an ARB(val) which is transmitted on a Loop to manage the access fairness algorithm. Since this is a low-priority ARByx, any arbitrating L_Port shall replace the ARB(F0) with its ARB(AL_PA). ARB(F0) is also used while selecting a temporary Loop Initialization Master during Loop Initialization.

7.1.3 ARB(FF)

ARB(FF) is an ARB(val) which may be originated by any L_Port in the MONITORING state when the access fairness window has been reset and no Fill Words other than ARB(FF) or Idle are received. Since this is the lowest-priority ARByx, any arbitrating L_Port may replace the ARB(FF) with its ARB(AL_PA). ARB(FF) has fewer bit transitions than the Idle. ARB(FF) may replace Idle to reduce the Electromagnetic Interference (EMI) which may be caused by the Idle.

7.2 Open Primitive Signals (OPNy)

A received Ordered Set shall be detected as an Open Primitive Signal (OPNy) by detecting that its first two characters (fully decoded) are equal to the value shown in table 2 and that its fourth character is not equal to hex 'FF'. If characters 3 and 4 are not equal, the received OPNy shall be detected as an Open full-duplex (OPNyx); if characters 3 and 4 are equal, the received OPNy shall be detected as an Open half-duplex (OPNyy). The values used for comparison shall be the data bytes resulting from 8B/10B decoding.

---

[2]Some L_Ports may not implement this recommendation, and may detect a received ARByx as an ARB(val) when y <> x. With such L_Ports it is vendor specific whether y or x is used as the value of 'val'. However, the Primitive Signals ARB(F0), ARB(FF) and ARB(AL_PA) (where AL_PA is the AL_PA of the receiving L_Port) should not be detected except when y = x (i.e., by checking all four characters of the Primitive Signal). L_Ports should only originate Arbitrate Primitive Signals (ARByx) where y = x, regardless of whether they test this on received Ordered Sets. These recommendations may be required by future standards.

An originating L_Port may determine the AL_PD (y value) of OPNy by checking the D_ID of the frame. If the left-most two bytes of the D_ID are the same as the left-most two bytes of the native address identifier of the originating L_Port, the left-most two bytes of

the D_ID are hex '0000', or the originating L_Port is a Private NL_Port, then the AL_PD shall be the right most byte of the D_ID. Otherwise, the AL_PD shall be hex '00' (the FL_Port); the D_ID is addressed to the Fabric or to a Port not on the same Loop.

7.2.1 Open full-duplex (OPNyx)

Open full-duplex (OPNyx) is transmitted on a Loop by a Participating L_Port to indicate that it is ready for Data and Link_Control frame transmission and reception (i.e., full-duplex) (see ANSI X3, FC-PH-x, 4.6, model 2). The OPNyx shall contain the AL_PD (destination = y value) of the L_Port to be opened and the AL_PS (source = x value) of the L_Port which transmitted OPNyx.

OPNyx that is received by a Participating L_Port (where y = AL_PA of the L_Port) in the MONITORING or ARBITRATING states indicates that another Participating L_Port desires to communicate in full-duplex mode with the L_Port that received OPNyx. The opened L_Port may transmit Data frames.

7.2.2 Open half-duplex (OPNyy)

Open half-duplex (OPNyy) is transmitted on a Loop by a Participating L_Port to indicate that it is ready for Data and Link_Control frame transmission and Link_Control frame reception (i.e., half-duplex) (see ANSI X3, FC-PH-x, 4.6, model 1). The OPNyy shall contain the AL_PD (destination) y value) of the L_Port to be opened.

OPNyy that is received by a Participating L_Port (where y = AL_PA of the L_Port) in the OPEN state indicates that the L_Port opened itself. OPNyy that is received by an L_Port (where y = AL_PA of the L_Port) in the MONITORING, ARBITRATING states indicates that another Participating L_Port desires to communicate in half-duplex mode with the L_Port that received OPNyy. In half-duplex mode, the opened L_Port shall not transmit Data frames.

7.3 Open Replicate Primitive Signals (OPNr)

A received Ordered Set shall be detected as an Open Replicate Primitive Signal (OPNr) by detecting that its first two characters (fully decoded) are equal to the value shown in table 2 and that its fourth character is equal to hex 'FF'. If character 3 is not equal to hex 'FF', the received OPNr shall be detected as an Open selective replicate (OPNyr); if character 3 is equal to hex 'FF', the received OPNr shall be detected as an Open broadcast replicate (OPNfr). The values used for comparison shall be the data bytes resulting from 8B/10B decoding.

Open Replicate (OPNr) is transmitted on a Loop by a Participating L_Port which desires to communicate with a group of NL_Ports on the same Loop. The requesting L_Port has won arbitration and is in the OPEN state. Transmitted frames shall be Class 3, although no buffer-to-buffer flow control (R_RDY) is used. If R_RDYs are transmitted by the L_Port in the OPEN state, they shall not be passed to FC-2. Frame reception is not guaranteed at each designated NL_Port (i.e., D_ID of the frame header may not be recognized by FC-2 or receive buffers may not be available). To avoid overflowing buffers and to assure that all designated NL_Ports can receive each replicate frame, the requesting L_Port should limit the number and size of frames that it transmits. The L_Port in the OPEN state shall discard all received frames.

NOTE — Although an FL_Port does not replicate frames through the Fabric, an FL_Port may transmit OPNr to communicate with multiple NL_Ports.

When a Participating L_Port is in the MONITORING or ARBITRATING state and detects OPNr (where the AL_PD is either hex 'FF' or the AL_PA of the NL_Port), it shall set REPLICATE to TRUE(1). While REPLICATE is TRUE(1), each received Transmission Word (except for normal Fill Word processing - updating the CFW appropriately) shall be retransmitted to the next L_Port on the Loop and shall also be provided to the FC-2 of the NL_Port for further processing; FL_Ports shall not propagate any frame through the Fabric.

NOTE — Restricting the FL_Port prevents duplicate frames from being delivered to an NL_Port on the same Loop as the originator of the OPNr from a broadcast or multicast server in the Fabric.

When CLS is received, all L_Ports with REPLICATE set to TRUE(1), shall set REPLICATE to FALSE(0).

7.3.1 Open selective replicate (OPNyr)

Open selective replicate (OPNyr where y = AL_PD and r = hex 'FF') is transmitted on a Loop by a Participating L_Port which desires to communicate with a subset of NL_Ports on the Loop. The requesting L_Port shall transmit OPNyr (where y is a member of the subset) to each NL_Port in the subset group. OPNyr may be transmitted to group members in any order. (See annex L.)

NOTE — The following sequence of events is a valid example and shows some of the versatility of using OPNyr.

Arbitrate and win
Transmit OPN(17,FF), transmit frame (17 processes)
Transmit OPN(23,FF), transmit frame (17 and 23 process)
Transmit OPN(76,FF), transmit frame (17, 23, and 76 process)
CLS

7.3.2 Open broadcast replicate (OPNfr)

Open broadcast replicate (OPNfr where f and r = hex 'FF') is transmitted on a Loop by a Participating L_Port which desires to communicate with all Participating NL_Ports on the Loop.

7.4 Close Primitive Signal (CLS)

Close (CLS) is transmitted on a Loop by an L_Port in the OPEN, OPENED, or RECEIVED CLOSE state. Once an L_Port has transmitted CLS, the L_Port shall not transmit frames or R_RDYs in the current Loop circuit. CLS indicates that the transmitting L_Port is prepared to or has ended the current Loop circuit (see 8.4). CLS is also transmitted to indicate that the INITIALIZATION process has completed (see 10.5.4).

7.5 Dynamic Half-Duplex Primitive Signal (DHD)

Dynamic Half-Duplex (DHD) is transmitted on a Loop by the L_Port in the OPEN state to indicate to the L_Port in the OPENED state that the L_Port in the OPEN state has no more Data frames to transmit. DHD shall only be requested by the L_Port in the OPEN state if both L_Ports in the current Loop circuit have indicated support of DHD via Login and if the L_Port in the OPEN state had transmitted OPNyx (full-duplex open). The DHD supported Login bit is found in FC-PH-3 (see ANSI X3.303-199x, FC-PH-3, 23.6.2.3). DHD may allow L_Ports to make more efficient use of the established Loop circuit (see annex C) by allowing an L_Port which is in the OPENED state to transmit all Data frames, even though the L_Port in the OPEN state has finished its data transfer.

NOTE — DHD adds minimal delay to the closing process (i.e., the next arbitrating L_Port will win at nearly the same time whether DHD is used or not.

Transmitting DHD only affects Data frames (i.e., Link_Control frames and R_RDYs may still be transmitted) just as in the definition of OPNyy (half-duplex open) (see 7.2.2). The recipient of DHD shall transmit CLS when it has finished its transmissions.

NOTE — DHD does not prohibit either L_Port from transmitting the first CLS. However, the L_Port in the OPEN state, if it had transmitted DHD, would normally wait for the L_Port in the OPENED state to transmit the first CLS.

7.6 Mark Primitive Signal (MRKtx)

A received Ordered Set shall be detected as a Mark Primitive Signal (MRKtx) by detecting that its first two characters (fully decoded) are equal to the value shown in table 2.

Mark (MRKtx) is transmitted on a Loop by a master control point to inform other Nodes of a certain action (e.g., synchronization; see annex H). The L_Port shall request to transmit MRKtx at the appropriate time (REQ(mark as tx)) and the LPSM shall attempt to transmit one MRKtx for this request by transmitting MRKtx instead of the next Fill Word. Since MRKtx shall only replace a Fill Word, it is possible that the mark may not be sent in the desired time. The REQ(mark as tx) may be withdrawn before the MRKtx can be transmitted (i.e., no MRKtx is transmitted).

NOTE — In order to avoid any delay when transmitting MRKtx, Fill Words are not required to precede or follow the MRKtx (i.e., Fill Words are not inserted before or after MRKtx).

The Mark Type (MK_TP) is expressed in character 3; the AL_PA of the originator of the MRKtx is in character 4 (x value). MK_TP is vendor unique and the interpretation and use is beyond the scope of this standard. The value(s) shall be assigned from the neutral disparity characters in table 1.

When MRKtx is received by the originator (i.e., x = AL_PS) and REPEAT is FALSE(0), the MRKtx shall be replaced with the CFW. All other L_Ports which are in the MONITORING, ARBITRATING, XMITTED CLOSE, or TRANSFER state or with REPEAT is TRUE(1) shall retransmit the received MRKtx.

NOTE — Since not all states retransmit MRKtx, in order to guarantee that all L_Ports receive MRKtx, the originator should be in the OPEN state and no other L_Ports in the OPENED state (i.e., all other L_Ports are either in the MONITORING or ARBITRATING state).

7.7 Loop Port Bypass/Enable Primitive Sequences

If an L_Port receives three consecutive identical Ordered Sets whose first two characters (fully decoded) are equal to the values shown in table 3, then the L_Port shall recognize a Loop Port Bypass Primitive Sequence or a Loop Port Enable Primitive Sequence. If character 3 is not equal to hex 'FF', a normal Loop Port Bypass or Loop Port Enable Primitive Sequence shall be recognized; if character 3 is equal to hex 'FF', a Loop Port Bypass all or Loop Port Enable all Primitive Sequence shall be recognized.

The Loop Port Bypass and Loop Port Enable Primitive Sequences are used to control access of an L_Port to the Loop as well as to control the optional Port Bypass Circuit. The Port Bypass Circuit may be used to physically bypass an L_Port, however, the L_Port is also logically bypassed (i.e., the L_Port shall not originate Transmission Words on the Loop). (See 8.1.4 and annex I.)

When an L_Port is in either Participating Bypassed or Non-Participating Bypassed mode, the L_Port shall not originate Transmission words (except for clock skew). The L_Port shall only monitor the Loop (as in the Non-Participating mode). If a Participating Bypassed L_Port recognizes LIP, it shall relinquish its AL_PA and enter the Non-Participating mode.

Although LPB or LPE may be transmitted in a number of states, not all states retransmit LPB or LPE. To guarantee that L_Ports receive LPB or LPE, the originator shall be in the OPEN state and all other L_Ports shall be in the MONITORING or ARBITRATING state; or all L_Ports shall be in the MONITORING, NORMAL-INITIALIZE, LOOP-FAIL-INITIALIZE, LOOP-FAIL-ERR_INIT, LOOP-FAIL-ERR_INIT-2, OPEN-INIT-SELECT-MASTER, or SLAVE-WAIT-FOR-MASTER state. To ensure that all L_Ports are in the MONITORING, NORMAL-INITIALIZE, LOOP-FAIL-INITIALIZE, LOOP-FAIL-ERR_INIT, LOOP-FAIL-ERR_INIT-2, OPEN-INIT-SELECT-MASTER, or SLAVE-WAIT-FOR-MASTER state, the originator of LPB or LPE shall enter NORMAL-INITIALIZE, LOOP-FAIL-INITIALIZE, or OPEN-INIT-START and not forward any LISMs from other L_Ports or any LIPs until it has completed its management of bypassed L_Ports. After the originator of LPB or LPE has completed its management of bypassed L_Ports it shall enter the NORMAL-INITIALIZE state.

Since an L_Port cannot guarantee that other L_Ports will not begin sending LIPs, it cannot be guaranteed that an L_Port which has an AL_PA and is bypassed will not lose its AL_PA while it is bypassed due to another L_Port transmitting LIPs. Due to this, LPB or LPE operation should be used with caution except with L_Ports that have trusted AL_PAs as described in 3.1.34.

7.7.1 Loop Port Bypass (LPByx)

Loop Port Bypass (LPByx) is transmitted on a Loop to bypass an L_Port and to activate the optional Port Bypass Circuit. The originator of the LPByx (as identified by AL_PS in character 4 — x value) may be a diagnostic manager or an operating L_Port that has determined that a "defective" L_Port (identified by AL_PD in character 3 — y value) exists on the Loop. (See annex I.)

When LPByx is recognized, where y = AL_PA of the L_Port, the L_Port shall set BYPASS to TRUE(1). LPByx may be used to diagnose the optional Port Bypass Circuit, for error recovery, or for any reason to cause an L_Port to be bypassed.

Each L_Port in the MONITORING, ARBITRATING, NORMAL-INITIALIZE, LOOP-FAIL-INITIALIZE, LOOP-FAIL-ERR_INIT, LOOP-FAIL-ERR_INIT-2, OPEN-INIT-SELECT-MASTER, or SLAVE-WAIT-FOR-MASTER state shall retransmit the received LPByx. When LPByx is received by the originator (i.e., x = AL_PA of the L_Port) and REPEAT is FALSE(0), the LPByx shall be replaced with the CFW.

Once an L_Port is bypassed and the optional Port Bypass Circuit has been activated, the L_Port shall only monitor the Loop for LPEyx (where y = AL_PA of the L_Port), LPEfx, or LIP. LIP is only used as a signal to relinquish its AL_PA; i.e., upon receipt of LIP, if BYPASS is TRUE(1), then the LPSM shall set PARTICIPATE to FALSE(0) and shall not go to the OPEN-INIT-START state.

7.7.2 Loop Port Bypass all (LPBfx)

Loop Port Bypass all (LPBfx where f = hex 'FF') is transmitted on a Loop to bypass all L_Ports and activate the optional Port Bypass Circuit(s) of all L_Ports (except the L_Port at x) . The originator of the LPBfx is identified by the AL_PS in character 4 (x value). LPBfx may be used to verify that an operating Loop is possible. It may also be useful to bypass a Non-Participating L_Port (i.e., the L_Port does not have an AL_PA). (See annex I.)

When LPBfx is recognized, all L_Ports on the Loop (participating or non-participating) except the L_Port at x, shall set BYPASS to TRUE(1).

NOTE — If multiple L_Ports are simultaneously transmitting LPBfx, all L_Ports will be bypassed. An L_Port which transmitted LPBfx and which was bypassed by another LPBfx (where x <> AL_PA of the L_Port), may at a later time attempt to deactivate the optional Port Bypass Circuit and participate on the Loop. The L_Port, which is attempting Loop recovery with LPBfx, may have a faulty transmitter and therefore, can by this means be bypassed by another L_Port.

Each L_Port in the MONITORING, ARBITRATING, NORMAL-INITIALIZE, LOOP-FAIL-ERR_INIT, LOOP-FAIL-ERR_INIT-2, OPEN-INIT-SELECT-MASTER, or SLAVE-WAIT-FOR-MASTER state shall retransmit the received LPBfx. When LPBfx is received by the originator (i.e., x = AL_PA of the L_Port) and REPEAT is FALSE(0), the LPBfx shall be replaced with the CFW.

7.7.3 Loop Port Enable (LPEyx)

Loop Port Enable (LPEyx) is transmitted on a Loop to enable an L_Port that had been previously bypassed and to deactivate the optional Port Bypass Circuit without an intervening LIP being received. The destination L_Port is identified by the AL_PD in character 3 (y value). The originator of the LPEyx is identified by the AL_PS in character 4 (x value). (See annex I.)

When LPEyx is recognized, where y = AL_PA of the L_Port, the L_Port shall set BYPASS to FALSE(0).

Each L_Port in the MONITORING, ARBITRATING, NORMAL-INITIALIZE, LOOP-FAIL-ERR_INIT, LOOP-FAIL-ERR_INIT-2, OPEN-INIT-SELECT-MASTER, or SLAVE-WAIT-FOR-MASTER state shall retransmit the received LPEyx. When LPEyx is received by the originator (i.e., x = AL_PA of the L_Port) and REPEAT is FALSE(0), the LPEyx shall be replaced with the CFW.

7.7.4 Loop Port Enable all (LPEfx)

Loop Port Enable all (LPEfx where f = hex 'FF') is transmitted on a Loop to deactivate all Port Bypass Circuits and enable all L_Ports into the Loop. The originator of the LPEfx is identified by the AL_PS in character 4 (x value). When an L_Port has been bypassed, it may have lost its AL_PA (e.g., the L_Port is required to relinquish its AL_PA upon recognizing LIP). Therefore, LPEfx allows these L_Ports (which no longer have an AL_PA) to be enabled on the Loop. (See annex I.)

When LPEfx is recognized, the L_Port shall set BYPASS to FALSE(0).

Each L_Port in the MONITORING, ARBITRATING, NORMAL-INITIALIZE, LOOP-FAIL-ERR_INIT, LOOP-FAIL-ERR_INIT-2, OPEN-INIT-SELECT-MASTER, or SLAVE-WAIT-FOR-MASTER state shall retransmit the received LPEfx. When LPEfx is received by the originator (i.e., x = AL_PA of the L_Port) and REPEAT is FALSE(0), the LPEfx shall be replaced with the CFW.

7.8 Loop Initialization Primitive Sequences (LIP)

If an L_Port receives three consecutive identical Ordered Sets whose first two characters (fully decoded) are equal to the values shown in table 3, then the L_Port shall recognize a Loop Initialization Primitive Sequence. If character 3 is equal to hex 'F7', then a normal LIP (LIP(F7)) shall be recognized; if character 3 is equal to hex 'F8', then a Loop Failure LIP (LIP(F8)) shall be recognized; if character 3 is equal to the AL_PA of the L_Port, then a reset LIP (LIPyx) shall be recognized; and, if character 3 is equal to hex 'FF', then a reset all LIP (LIPfx) shall be recognized.

Loop Initialization (LIP) is a Primitive Sequence used by an L_Port to detect if it is part of a Loop or to recover from certain Loop errors (see 8.4.3, item 21 and item 23 and clause 10).

The LIP contains information on why the LIP was transmitted in the right-most two characters (characters 3 and 4). Other L_Ports may make decisions based on this information (e.g., inform an operator of a Loop Failure).

7.8.1 Loop Initialization — no valid AL_PA

Loop Initialization (LIP(F7,F7)) is used by the originating L_Port to acquire an AL_PA.

7.8.2 Loop Initialization — Loop Failure; no valid AL_PA

Loop Initialization (LIP(F8,F7)) is used by the originating L_Port to indicate that a Loop Failure has been detected at its receiver (hex 'F8'); hex 'F7' is used to indicate that the L_Port does not have a valid AL_PA.

7.8.3 Loop Initialization — valid AL_PA

Loop Initialization (LIP(F7,AL_PS)) is used by the originating L_Port (identified by AL_PS) to reinitialize the Loop. The L_Port may have noticed a performance degradation (e.g., it has been arbitrating longer than it deemed reasonable) and is trying to restore the Loop into a known state.

7.8.4 Loop Initialization — Loop Failure; valid AL_PA

Loop Initialization (LIP(F8,AL_PS)) is used by the originating L_Port (identified by AL_PS) to indicate that a Loop Failure has been detected at its receiver.

7.8.5 Loop Initialization — reset L_Port

Loop Initialization (LIP(AL_PD,AL_PS)) is used by the originating L_Port (identified by AL_PS) to reset the NL_Port (identified by AL_PD). All L_Ports shall treat this LIP as specified in 7.8.3, however, the NL_Port at AL_PD shall also perform a vendor specific reset. If AL_PD = hex 'FF', a vendor specific reset shall be performed by all L_Ports (including those without an AL_PA, but not the one at AL_PS).

7.8.6 Loop Initialization — reserved

Loop Initialization (LIPba) is reserved for future use. The 'b' and 'a' characters of the LIPba Ordered Set shall contain one of the neutral disparity values from table 1, and the LIPba shall not be a member of any LIP defined in 7.8.1 through 7.8.5. A LIPba shall be treated just like any other (Non F8) LIP (see 10.5).

8 L_Port operation

To simplify L_Port design and minimize Transmission Word propagation delay, the following rules apply:

— all routing decisions by the LPSM (except during the Loop INITIALIZATION process) shall be made based on the AL_PA in the Primitive Signals (i.e., during normal operation, no LPSM routing decisions are made based on frame content);

— logging errors that are detected when retransmitting Transmission Words is optional; and,

NOTE — While an L_Port is not required to log errors encountered while retransmitting Transmission Words, fault isolation and error analysis may be enhanced by doing so.

— Transmission Words are not routed to the FC-2 of the NL_Port in the ARBITRATING and MONITORING states unless REPLICATE is set to TRUE(1) (see 7.3).

The maximum delay of a Transmission Word through an L_Port in the MONITORING or ARBITRATING state shall not exceed six (6) Transmission Word periods except when in clock skew management deletion pending state (see annex A).

The following steps provide an example for how an L_Port transfers one or more ANSI X3, FC-PH-x frames on a Loop:

(1) The L_Port requests the LPSM to obtain access to the Loop.

(2) The LPSM enters the ARBITRATING state and transmits its ARB(AL_PA) in place of the appropriate received Fill Word (see 7.1) until a matching ARB(AL_PA) is received. When the matching ARB(AL_PA) is received, the L_Port opens the Loop (i.e., stops retransmitting received Transmission Words).

(3) The LPSM transmits OPNy to establish a point-to-point Loop circuit on the Loop with another L_Port. OPNy may be followed by ANSI X3, FC-PH-x frame(s). The number of frames that can immediately be transmitted is based on BB_Credit (see 8.3.4).

(4) Either L_Port (of the Loop circuit) may transmit CLS when the L_Port desires to close the Loop circuit. When an L_Port receives CLS, it completes transmitting its frame(s), retransmits the CLS, and closes its end of the Loop circuit. When the CLS returns to the L_Port which originated the CLS, this L_Port closes its end of the Loop circuit.

NOTE — Since either open L_Port may transmit CLS, an L_Port must be prepared to handle CLS simultaneously with or on the next Transmission Word after entering the XMITTED CLOSE state.

8.1 History variables

8.1.1 Access fairness history

The access fairness algorithm requires four memory elements that shall be maintained and used by each L_Port (see 8.4 for management requirements of each memory element):

a) **ACCESS** — the value of this variable is used by an L_Port to determine the status of the fairness window (i.e., whether the L_Port may arbitrate for access to the Loop). If ACCESS is FALSE(0), then an L_Port which is using the fairness algorithm, shall not arbitrate for access to the Loop; if ACCESS is TRUE(1), then an L_Port may arbitrate for access to the Loop and the L_Port may use the TRANSFER state to open a Loop circuit with another L_Port.

b) **ARB_WON** — the value of this variable is used by an L_Port to indicate that this L_Port has won arbitration. If ARB_WON is FALSE(0), then the L_Port did not win arbitration; if ARB_WON is TRUE(1), then the L_Port won arbitration.

c) **ARB_PEND** — the value of this variable is used by an L_Port which has been opened while arbitrating to remember that it has transmitted one or more ARB(AL_PA) Primitive Signals. If ARB_PEND is FALSE(0), then the L_Port was not arbitrating; if ARB_PEND is TRUE(1), the L_Port was arbitrating.

NOTE — This history variable forces the L_Port to finish arbitrating even if the L_Port no longer desires access to the Loop to assure that the fairness window is reset.

d) **XMIT_2_IDLES** — the value of this variable is used by an L_Port to remember that after receiving ARB(F0), two (2) Idles shall be transmitted when Idle is received. The current Fill Word when set to Idle shall not be modified until XMIT_2_IDLES is FALSE(0).

NOTE — Since the Idle is used to reset the fairness window, by transmitting two Idles, the probability of at least one Idle traversing the Loop is increased. If only one Idle is transmitted, it could be removed by another L_Port if that L_Port needs to delete a Fill Word for clock skew.

8.1.2 Duplex mode history

The OPEN, OPENED and RECEIVED CLOSE state requires one memory element, called DUPLEX, to determine whether the L_Port is allowed to originate Data frames. If DUPLEX is FALSE(0), the Loop circuit is operating in half-duplex mode; if DUPLEX is TRUE(1), the Loop circuit is operating in full-duplex mode (see 8.4 for management requirements of DUPLEX).

8.1.3 Replicate mode history

The MONITORING and ARBITRATING states require one memory element, called REPLICATE, to remember if OPNr had been received. If REPLICATE is FALSE(0), the states operate normally; if REPLICATE is TRUE(1), all received Transmission Words (except for normal Fill Word processing - updating the CFW appropriately) shall be retransmitted and also shall be provided to the FC-2 of the NL_Port for further processing. (See 7.3 and 8.4.3, item 13 and item 14.)

The OPEN state requires REPLICATE to remember that OPNr was transmitted in the ARBITRATION WON or TRANSFER state. If REPLICATE is FALSE(0), the state operates normally; if REPLICATE is TRUE(1), the L_Port may originate additional OPNr's; the L_Port shall not use BB_Credit management. (See 7.3 and 8.4.3, item 15 and item 16.)

8.1.4 Operational mode history

An L_Port uses two memory elements to record the L_Port's operational mode:

a) **PARTICIPATE** — set TRUE(1) if the L_Port has an AL_PA, set FALSE(0) if the L_Port does not have an AL_PA.

b) **BYPASS** — set TRUE(1) if the L_Port activates its optional Port Bypass Circuit, set FALSE(0) if the L_Port deactivates its optional Port Bypass Circuit.

REPEAT is a symbol that is defined to simplify the LPSM description. REPEAT is TRUE(1), if PARTICIPATE is FALSE(0) or BYPASS is TRUE(1) or both. REPEAT is FALSE(0), if PARTICIPATE is TRUE(1) and BYPASS is FALSE(0). When REPEAT is TRUE(1), the LPSM repeats most incoming transmission words (except for normal Fill Word processing—updating the CFW appropriately) without responding to them. When REPEAT is FALSE(0), the LPSM actively participates on the Loop.

The combined values of the PARTICIPATE and BYPASS variables record the four L_Port operational modes:

1) **Participating** (PARTICIPATE = 1, BYPASS = 0) — the L_Port has an AL_PA and is enabled into the Loop. The L_Port may use the Loop and respond to all requests directed to it. This is the normal operational mode in which most Loop access occurs. In this mode, REPEAT is FALSE(0).

2) **Non-Participating** (PARTICIPATE = 0, BYPASS = 0) — the L_Port does not have an AL_PA, but is enabled into the Loop. The L_Port repeats transmission words (except for normal Fill Word processing—updating the CFW appropriately) and only responds to a limited number of requests such as Loop Initialization. If the L_Port wishes to obtain an AL_PA and participate in the Loop, the L_Port may initiate Loop Initialization; it shall attempt to obtain an AL_PA if Loop Initialization occurs. In this mode, REPEAT is TRUE(1).

3) **Participating Bypassed** (PARTICIPATE = 1, BYPASS = 1) — the L_Port has an AL_PA, but is bypassed (i.e., not enabled) from the Loop. The L_Port activates its optional Port Bypass Circuit if one is present. The L_Port also repeats transmission words (except for normal Fill Word processing—updating the CFW appropriately) in case no Port Bypass Circuit is present. The L_Port shall respond to an LPEyx directed to its AL_PA. In this mode, REPEAT is TRUE(1).

4) **Non-Participating Bypassed** (PARTICIPATE = 0, BYPASS = 1) — the L_Port does not have an AL_PA and is bypassed (i.e., not enabled) from the Loop. The L_Port activates its optional Port Bypass Circuit if one is present. The L_Port also repeats transmission words (except for normal Fill Word processing—updating the CFW appropriately) in case no Port Bypass Circuit is present. The L_Port does not respond to any Primitive Signal or Primitive Sequences directed to a specific AL_PA. In this mode, REPEAT is TRUE(1).

8.1.5 DHD received history

The OPENED state requires one memory element if DHD is supported, called DHD_RCV. This variable is set to TRUE(1) if DHD is received. The variable is checked when the L_Port in the OPENED state has completed all transmissions to the L_Port in the OPEN state. If DHD_RCV is FALSE(0), then the L_Port may continue to wait to receive CLS (normal operation) or it may transmit CLS. If DHD_RCV is TRUE(1), then the L_Port shall transmit CLS. (See annex C.)

8.1.6 Error Initialization history

The LOOP-FAIL-INITIALIZE state may use one memory element, called ERR_INIT. The variable is checked by the L_Port in the LOOP-FAIL-INITIALIZE state to determine whether Loop Initialization (see clause 10) should be continued or delayed (to avoid initializing in 10.5 when there is a low probability that it will complete). If LIP(F8) is received and ERR_INIT is FALSE(0), Loop Initialization shall be attempted; if ERR_INIT is TRUE(1), then Loop Initialization shall be delayed. (See 8.4, item 13 and item 21 and 10.5.4.)

8.1.7 ARB(FF) history

The MONITORING state uses a memory element, called ARBf_SENT, to indicate that the L_Port has requested the LPSM to modify its CFW to ARB(FF) from Idles. If ARBf_SENT is FALSE(0), the current Fill Word is managed normally; if ARBf_SENT is TRUE(1), then the current Fill Word shall remain ARB(FF) until an ARB(AL_PA) is received or ARB(F0) (if REPEAT is TRUE(1)) is received. (See 8.4.)

8.2 Timeouts

8.2.1 FC-PH timeout values

Timeout values (e.g., R_T_TOV) and related timeout procedures in ANSI X3, FC-PH-x, 29.2, shall be used as appropriate.

8.2.2 Arbitrated Loop timeout value

The Arbitrated Loop timeout value (AL_TIME) is 15 ms, which represents two times the worst case round-trip latency for a very large Loop. AL_TIME is based on twice the sum of the following values:

— 134 times an L_Port internal latency of six (6) Transmission Word periods at 1,062 5 Gbits/sec of the L_Port and

— 134 times 10 km, the cable latency (5 ns/meter).

AL_TIME is primarily used during the INITIALIZATION process to control events that require a Loop round-trip latency to complete. The sequencing of these events must be coordinated between multiple L_Ports, which requires that all L_Ports use AL_TIME consistently. During the INITIALIZATION process, L_Ports shall measure AL_TIME with a tolerance of -0%, +20% (i.e., an AL_TIME timeout shall expire from a minimum of 15ms up to a maximum of 18ms).

NOTE — It is conceivable that the maximum round-trip delay of a Loop configuration is greater than the AL_TIME. However, determining interoperability when using a different AL_TIME value is outside the scope of this standard.

8.2.3 Loop timeout value

The Loop timeout value (LP_TOV) is 2 seconds. LP_TOV is used to keep a Loop from deteriorating due to protocol errors or lost Ordered Sets. For example, LP_TOV is used to reset the fairness window (see 4.3) and during the INITIALIZATION process to time start-up events (see 10.5.4).

8.3 Operational characteristics

8.3.1 Transmission Word processing

8.3.1.1 Power-on Transmission Words

At power-on, the L_Port shall turn off its transmitter until it is ready to participate in Loop Initialization.

8.3.1.2 Invalid Transmission Words and Transmission Characters

An L_Port shall make substitutions for invalid received Transmission Words and Transmission Characters (see 8.4) as follows:

— in the MONITORING or ARBITRATING states:

  – if an invalid Transmission Word is detected, the L_Port shall substitute the CFW for that Transmission Word.

  – if an invalid Beginning Running Disparity is detected on an Ordered Set, the L_Port shall substitute the CFW.

— in any other state the L_Port shall follow the rules defined in ANSI X3, FC-PH-x, 24.3.5, and clause 29.

8.3.2 Clock skew management

When an L_Port implements receive and transmit clocks with different reference sources, a buffer is required between the receiver and transmitter logic to manage the clock frequency and phase differences (see annex G for clock design options).When a buffer is required, the L_Port shall implement the buffer as defined in annex A. To prevent buffer over-run or under-run, the L_Port shall use the clock skew management rules defined in annex A to control the level of data.

When processing Transmission Words between frames, any ARByx shall be treated the same as Idle. Fill Words or any Ordered Set defined for use as a Primitive Sequence shall be treated equally. (See clause 7; ANSI X3, FC-PH-x, clause 17; and, annex A and G.)

8.3.3 Error detection and recovery

Each state in 8.4 contains the procedures for handling failures. State transitions are considered to take place instantaneously and no error detection takes place during a state transition. Any failure or subsequent state request that occurs during a state transition shall be detected in the subsequent state.

Following recovery from a failure, the L_Port shall comply with the provisions for Sequence integrity, error detection, and Sequence recovery specified in ANSI X3, FC-PH-x, 24.3.5 and clause 29.

8.3.4 BB_Credit and Available_BB_Credit

BB_Credit and Available_BB_Credit are used when transmitting a SOFc1, a Class 2, or a Class 3 frame. Before Login, the "Alternate BB_Credit Management" bit (see ANSI X3, FC-PH-x, 23.6.3 and 26.5) and BB_Credit shall be set to *0* and one (1) in the OLD-PORT state and to *1* and zero (0) in the OPEN-INIT-START state, respectively (see 8.4.3 item 21 and item 23, and 10.5.4). During Login, BB_Credit shall be set to a value that represents the number of receive buffers that the L_Port shall guarantee to have available when a Loop circuit is established.

When on a Loop, L_Ports have unique characteristics (unlike point-to-point or Fabric-attached N_Ports):

— Loop circuits are dynamic;

— if not properly managed, an L_Port may have frames in the receive buffers from the previous Loop circuit when a new Loop circuit is established; even a BB_Credit equal to one (1) may overrun the receive buffers;

— using BB_Credit equal to zero (0) requires a turn-around delay and impedes performance at the beginning of each Loop circuit; and,

— balancing BB_Credit at the end of a Loop circuit may impede performance.

"Alternate BB_Credit Management" is used to achieve the best performance while addressing these unique Loop characteristics. To avoid a turn-around delay at the beginning of a Loop circuit, L_Ports may take advantage of the BB_Credit which is established during Login. Although balancing BB_Credit is not required (receive buffers may be emptied after the Loop circuit is closed), the BB_Credit value represents the number of receive buffers that an L_Port is assumed to have available when the next Loop circuit is established. Therefore, an L_Port shall not enter the MONITORING state until the number of available receive buffers is at least equal to the largest BB_Credit value which the L_Port disseminated during Login.

BB_Credit in the following discussion is identified as *open* BB_Credit (i.e., the BB_Credit of the L_Port which transmits the OPNy) and *opened* BB_Credit (i.e., the BB_Credit of the L_Port which receives the OPNy) (see annex F).

A positive opened BB_Credit allows the L_Port to follow OPNy with frames, without waiting for an R_RDY (i.e., there is no round-trip delay).

NOTE — "Alternate BB_Credit Management" is written from the view of the L_Port which transmits the OPNy. This L_Port knows the opened BB_Credit and its open BB_Credit, but it has no knowledge of what the opened L_Port will use for the open BB_Credit. The L_Port which receives the OPNy may choose to use the open BB_Credit, or immediately use Available_BB_Credit.

8.3.4.1 BB_Credit management per Loop circuit

For each Loop circuit, BB_Credit for the L_Ports in the OPEN and OPENED state is any value less than or equal to the BB_Credit which the other L_Port in the Loop circuit advertised during Login. L_Ports shall not have more than 255 outstanding R_RDYs during any Loop circuit.

NOTE — If the L_Port in the OPEN state is using an opened BB_Credit of zero (0), a Loop turn-around delay is required (i.e., an R_RDY must be received) before the L_Port is allowed to transmit the first frame.

The L_Port which transmits OPNy shall obey the following rules for transmitting R_RDYs:

NOTE — Since a minimum of six (6) Fill Words are required between the OPNy and the first frame, the L_Port may transmit one R_RDY instead of one Fill Word without any performance penalty. The number of R_RDYs which the L_Port transmits before the first frame is a balance between delaying the transmission of the first frame and delaying receiving frames.

— if the open BB_Credit equals zero (0), the L_Port shall transmit one R_RDY for each currently available receive buffer.

— if the open BB_Credit is greater than zero (0), the L_Port shall transmit one R_RDY for each BB_Credit which this L_Port advertised plus one R_RDY for each additional available receive buffer.

— If CLS is received before all R_RDYs have been transmitted, the remaining R_RDYs are not required to be transmitted in the Loop circuit.

The L_Port may transmit the number of frames specified by the opened BB_Credit before receiving an R_RDY. The L_Port shall discard one received R_RDY for each of these frames sent. When the number of discarded R_RDYs equals the opened BB_Credit, the L_Port shall use Available_BB_Credit management.

The L_Port which receives OPNy shall obey the following rules for transmitting R_RDYs:

NOTE — The number of R_RDYs which the L_Port transmits before the first frame is a balance between delaying the transmission of the first frame and delaying receiving frames.

— if the opened BB_Credit equals zero (0), the L_Port shall transmit one R_RDY for each currently available receive buffer.

— if the opened BB_Credit equals zero (0), the L_Port may transmit CLS if there are no available receive buffers.

— if the opened BB_Credit is greater than zero (0), the L_Port shall transmit one R_RDY for each BB_Credit which this L_Port advertised plus one R_RDY for each additional available receive buffer.

— If CLS is received before all R_RDYs have been transmitted, the remaining R_RDYs are not required to be transmitted in the Loop circuit.

The L_Port shall initialize the open BB_Credit to zero (0). If the L_Port can determine the open BB_Credit, it may transmit the number of frames specified by the open BB_Credit. If the L_Port transmitted frames based on the open BB_Credit, it shall discard one received R_RDY for each of these frames sent. When the number of discarded R_RDYs equals the open BB_Credit, the L_Port shall use Available_BB_Credit management.

8.3.4.2 Available_BB_Credit management per Loop circuit

Once the L_Port has discarded the same number of R_RDYs as BB_Credit, the L_Port shall use Available_BB_Credit for transmitting additional frames.

Available_BB_Credit is one of the following values:

— zero (0) – the initial value until one or more R_RDYs have been received; or.

— the number of R_RDYs received less the number of frames transmitted.

The L_Port may transmit the number of frames specified by Available_BB_Credit. For each frame sent, Available_BB_Credit is decremented by one (1); for each R_RDY received, Available_BB_Credit is incremented by one (1). As long as Available_BB_Credit greater than zero, the L_Port may transmit frames during this Loop circuit.

8.4 Loop Port State Machine (LPSM)

The Loop Port State Machine (LPSM) shall be used to define the behavior of the L_Ports when they require access to and use of a Loop. The following subclauses specify the state names, state diagram, and item references for the LPSM.

8.4.1 State names

The state names and numbers used in the LPSM, along with a brief description, are given below. Reference items for each state are considered part of each state. The reference item numbers are identified in the L_Port state machine diagram in 8.4.2. The reference item text follows the state machine diagram in 8.4.3.

MONITORING (0): The LPSM is transmitting received Transmission Words and, if it is in the Participating mode, monitoring the Loop for certain Ordered Sets (e.g., OPNy and OPNr). This is the default state of any L_Port.

ARBITRATING (1): The LPSM is arbitrating for control of the Loop.

ARBITRATION WON (2): The LPSM has received a matching ARB(AL_PA) (i.e., AL_PA = AL_PA of this L_Port) while arbitrating.

OPEN (3): The LPSM has transmitted OPNy while in the ARBITRATION WON state. Normal FC-2 protocol follows.

OPENED (4): The LPSM has received a matching OPNy (i.e., y = AL_PA of this L_Port) while in the MONITORING or ARBITRATING state. Normal FC-2 protocol follows.

XMITTED CLOSE (5): The LPSM has transmitted CLS and intends to relinquish control of the Loop.

RECEIVED CLOSE (6): The LPSM has received CLS.

TRANSFER (7): The LPSM, while in the OPEN state, has transmitted CLS and requires the Loop to communicate with another L_Port.

INITIALIZATION process (8): The LPSM is initializing or re-initializing.[3]

OLD-PORT (A): The LPSM has determined that it wants to operate in a point-to-point mode utilizing ANSI X3, FC-PH-x protocol without FC-AL.

[3]The INITIALIZATION process encompasses the INITIALIZING and OPEN-INIT states that were defined in the first publication of this standard.

8.4.2 State diagram

The state diagram is shown in figure 3. The numbered reference items for states and state transitions in 8.4.3 are normative parts of the LPSM definition. If the details were in the state diagrams, the diagrams would be difficult to read and interpret.

States are identified with a single letter or digit followed by a single colon character (e.g., 6:). Transitions identified as "(Xn):", where *n* is a single digit or letter, represent valid transitions from multiple states to the ending state, *n*, caused by an event outside the steady state operation of the LPSM. A transition identified as "(mn):", where *m* and *n* are single digits or letters, represents a transition from state *m* to state *n*. Each transition and state is accompanied by detailed specifications and requirements identified by the numbered reference item.

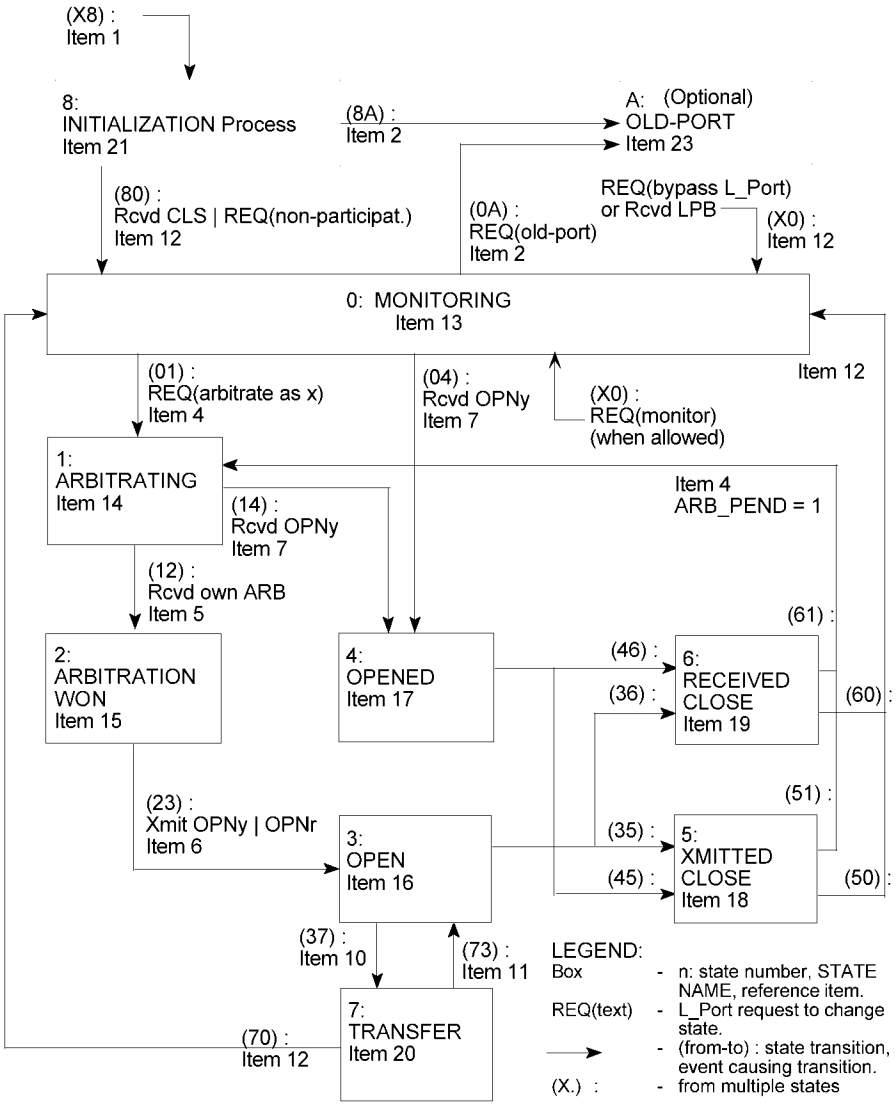

Figure 3 — State Diagram

8.4.3 Reference items

For detailed information about a state or state transition, refer to the item number in the list below.

For conditions that are not explicitly listed in this section as causing state changes to occur, the LPSM shall remain in the current state.

1 **Transition (X8):** This transition[4] shall be made at power-on of an L_Port, after detecting a failure (see clause 10 and ANSI X3, FC-PH-x, clause 23), when a LIP is recognized, or from any state when the L_Port requests it (see 10.5.3).

All fibre-type dependent operations shall be complete before making this transition (e.g., Open Fibre Control) (see ANSI X3, FC-PH-x, clauses 5 to 10).

2 **Transition (0A:; (8A):** The LPSM shall make the transition to the OLD-PORT state (if supported) (see item 13, item 23, and 10.5.4).

3 **Transition (01):** The LPSM shall make the transition to the ARBITRATING state (see item 13 and item 14).

4 **Transitions (51):, (61):** The LPSM shall make the transition to the ARBITRATING state (see item 14, item 18, and item 19).

5 **Transition (12):** The LPSM shall make the transition to the ARBITRATION WON state (see item 14 and item 15).

6 **Transition (23):** The LPSM shall make the transition to the OPEN state (see item 15 and item 16).

7 **Transitions (04):, (14):** The LPSM shall make the transition to the OPENED state (see item 13, item 14, and item 17).

8 **Transitions (35):, (45):** The LPSM shall make the transition to the XMITTED CLOSE state (see item 15, item 16, item 17 and item18).

9 **Transitions (36):, (46):** The LPSM shall make the transition to the RECEIVED CLOSE state (see item 16, item 17, and item 19).

10 **Transition (37):** The LPSM shall make the transition to the TRANSFER state (see item 16 and item 20).

11 **Transition (73):** The LPSM shall make the transition to the OPEN state (see item 16 and item 20).

12 **Transitions (X0):, (50):, (60):, (70):** The LPSM shall make the transition to the MONITORING state (see item 13, item 15, item 18, and item 20).

[4]Some implementations may choose to allow an L_Port to exit the INITIALIZATION process without completing initialization. These L_Ports 'initialize' outside the scope of this standard.

13 **State 0 (MONITORING) actions (**table 4 and the following text describe the MONITORING state): The LPSM shall set ERR_INIT to FALSE(0), DUPLEX to FALSE(0), ARB_WON to FALSE(0), ARB_PEND to FALSE(0), ARBf_SENT to FALSE(0), and REPLICATE to FALSE(0). The LPSM shall retransmit all received Transmission Words unless specifically stated otherwise.

If PARTICIPATE is FALSE(0), the L_Port does not have an AL_PA. Therefore, the tests for val = AL_PA, x = AL_PA, or y = AL_PA are ignored and the entries for val <> AL_PA, x<> AL_PA, or y <> AL_PA shall be followed.

If Idle is received, the CFW shall be modified as follows:

— if REPEAT is FALSE(0) and:

  – if ARBf_SENT is FALSE(0), the CFW shall be set to Idle and ACCESS shall be set to TRUE(1) or

  – if ARBf_SENT is TRUE(1), the CFW shall be changed to ARB(FF).

— if REPEAT is TRUE(1), the CFW shall be set to Idle.

If ARByx is received, the CFW shall be modified as follows:

— if ARByx = ARB(F0) and the CFW is Idle or ARB(FF):

  – if REPEAT is FALSE(0), XMIT_2_IDLES shall be set to TRUE(1) and the CFW shall not be changed or

  – if REPEAT is TRUE(1), the CFW shall be changed to ARB(F0).

  NOTE — If an L_Port is in the MONITORING state while the Loop is in the INITIALIZATION process as described in clause 10, then PARTICIPATE must remain FALSE(0) or BYPASS must remain TRUE(1) until ARB(F0) is received by every L_Port. ARB(F0) during the INITIALIZATION process indicates that a LIM has been selected

— if ARByx = ARB(F0) and the CFW is neither Idle nor ARB(FF), the CFW shall be set to ARB(F0), ARBf_SENT shall be set to FALSE(0), and XMIT_2_IDLES shall be set to TRUE(1);

— if ARByx = ARB(FF), the CFW shall be modified as follows:

  – if REPEAT is FALSE(0):

    – if the CFW is not Idle or XMIT_2_IDLES is FALSE(0), the CFW shall be changed to ARB(FF) or

    – if the CFW is Idle and XMIT_2_IDLES is TRUE(1), the CFW shall not be changed.

  – if REPEAT is TRUE(1), the CFW shall be changed to ARB(FF).

— if ARByx <> ARB(val) (i.e., y <> x), the CFW should not be changed;[5] or,

[5]Some L_Ports may set the CFW to the received ARByx, but this practice is not recommended.

— if ARByx = ARB(val):

  – if REPEAT is FALSE(0):

    – if val <> AL_PA of the L_Port and:

      - if the CFW is not Idle or XMIT_2_IDLES is FALSE(0), the CFW shall be set to ARB(val) and ARBf_SENT shall be set to FALSE(0) or

      - if the CFW is Idle and XMIT_2_IDLES is TRUE(1), the CFW shall not be changed.

    – if val = AL_PA of the L_Port, the CFW shall be set to Idle, ARBf_SENT shall be set to FALSE(0).

  – if REPEAT is TRUE(1), the CFW shall be changed to ARB(val).

If a Fill Word is to be transmitted, the CFW shall be used. If the CFW is Idle and XMIT_2_IDLES is TRUE(1), XMIT_2_IDLES shall be set to FALSE(0) after two Idles are transmitted.

If REPEAT is FALSE(0):

a) if REPLICATE is TRUE(1), the LPSM shall receive (i.e., present to the FC-2 of the NL_Port for further processing) and retransmit all Transmission Words (except for normal Fill Word processing – updating the CFW appropriately);

b) if OPNfr is received, the LPSM of the NL_Port shall set REPLICATE to TRUE(1) and shall retransmit the received OPNfr;

c) if OPNyr is received, where y = AL_PA of the NL_Port, the LPSM shall set REPLICATE to TRUE(1) and shall retransmit the received OPNyr;

d) if OPNy is received, where y = AL_PA of the L_Port, the LPSM shall make the transition to the OPENED state (see item 7 and item 17);

e) if any other OPNy is received, it shall be retransmitted;

f) if MRKtx is received and:

  – if x = AL_PA of the L_Port, the LPSM shall transmit the CFW; the MRKtx is discarded;

  – if the MK_TP and AL_PS match the expected values, the action identified by MK_TP shall be performed; or,

  – if x <> AL_PA of the L_Port, the received MRKtx shall be retransmitted.

g) if CLS is received while REPLICATE is TRUE(1), the LPSM shall set REPLICATE to FALSE(0) and retransmit the received CLS;

h) if the L_Port requests arbitration (REQ(arb own AL_PA)) and ACCESS is TRUE(1), the LPSM shall make the transition to the ARBITRATING state (see item 3 and item 14);

i) if LP_TOV has elapsed since the L_Port began requesting arbitration (REQ(arb own AL_PA)), ACCESS may be set to TRUE(1); or,

j) if the L_Port requests to transmit a MRKtx (REQ(mark as tx)), the LPSM shall transmit one MRKtx at the next Fill Word (see clause 7), unless REQ(mark as tx) is removed before MRKtx is transmitted.

If LIP is recognized:

— if BYPASS is FALSE(0), the LPSM shall make the transition to the OPEN-INIT-START state (see item 1, item 21, and clause 10); or,

— if BYPASS is TRUE(1), the L_Port shall relinquish its AL_PA, shall set PARTICIPATE to FALSE(0), and shall remain in the MONITORING state.

If LPByx (y = AL_PA of the L_Port) or LPBfx is recognized or the L_Port requests to be bypassed (REQ(bypass L_Port)), the LPSM shall set BYPASS to TRUE(1); shall set REPLICATE to FALSE(0); shall set ARBf_SENT to FALSE(0); and, shall retransmit the LPB, if one was received.

If LPEyx (y = AL_PA of the L_Port) or LPEfx is recognized or the L_Port requests to be enabled (REQ(enable L_Port)), the LPSM shall set BYPASS to FALSE(0) and shall retransmit the LPE, if one was received.

The LPSM shall retransmit all other received Transmission Words on the Loop (see 8.3.1).

Invalid Transmission Character substitution shall be performed as specified in 8.3.1.

If the L_Port requests to transmit ARB(FF) (REQ(arbitrate (FF))) the LPSM shall set ARBf_SENT to TRUE(1) after six (6) Idles have been forwarded.

If the L_Port requests initialization (REQ(initialize)), the LPSM shall make the transition to the NORMAL-INITIALIZE state (see item 1, item 21, and clause 10).

If the L_Port requests to use the point-to-point protocol as defined in FC-PH-x (REQ(old-port)), the LPSM shall make the transition to the OLD-PORT-REQ state (see item 2, item 23, and 10.5.4.3).

If the LPSM detects a Loop Failure on its inbound fibre and:

— if REPEAT is FALSE(0), the LPSM shall make the transition to the INITIALIZATION process (see item 21 and 10.5.4).

— if REPEAT is TRUE(1), the LPSM shall transmit LIP(F8) until the Loop recovers from the failure. The L_Port shall remain in the MONITORING state.

If the L_Port requests not to participate on the Loop (REQ(nonparticipat.)), it shall relinquish its AL_PA; set PARTICIPATE to FALSE(0); and, shall set ARBf_SENT to FALSE(0). The LPSM may transmit LIPs (with the right-most two characters equal to hex 'F7F7') to invoke the Loop Initialization procedure and allow another L_Port to acquire the relinquished AL_PA. If LIPs are transmitted, the L_Port shall transmit at least 12 LIPs. The LIPs are only transmitted once for each REQ(nonparticipat.) to allow this request to be active until the L_Port requests to participate (REQ(participating)).

NOTE — The L_Port may arbitrate for the Loop (using ARB(AL_PA) in order to quiesce any ongoing activity before transmitting LIP.

If the L_Port requests to participate on the Loop (REQ(participating)), the LPSM shall make the transition to the NORMAL-INITIALIZE state (see item 1, item 21, and clause 10).

NOTE — The L_Port may arbitrate for the Loop (using ARB(val) where val is a trusted AL_PA) in order to quiesce any ongoing activity before transmitting LIP.

14 **State 1 (ARBITRATING) actions** (table 5 and the following text describe the ARBITRATING state): The LPSM shall set DUPLEX to FALSE(0) and ARB_WON to FALSE(0). The LPSM shall retransmit all received Transmission Words unless specifically stated otherwise. The LPSM shall transmit an ARB(AL_PA) (where AL_PA is the AL_PA of the L_Port) when either an Idle or a lower priority ARB(AL_PA), ARB(F0), or ARB(FF) is received. Once the LPSM has transmitted its own ARB(AL_PA), it shall set ARB_PEND to TRUE(1) and shall not transmit a lower-priority ARB(AL_PA).

If Idle is received and:

— if XMIT_2_IDLES is FALSE(0), the CFW shall be set to ARB(AL_PA) (where AL_PA is the AL_PA of the L_Port) or

— if XMIT_2_IDLES is TRUE(1), the CFW shall be set to Idle.

If ARByx is received, where ARByx <> ARB(val) or ARByx = ARB(val) and val does not equal the AL_PA of the L_Port, the CFW shall be modified as follows:

— if XMIT_2_IDLES is FALSE(0) or the CFW is not Idle and:

  – if ARByx = ARB(val) where val < AL_PA of the L_Port, the CFW shall be changed to the received ARB(val);

  – if ARByx = ARB(val) where val = hex 'F0', the CFW shall be changed to ARB(AL_PA) (where AL_PA is the AL_PA of the L_Port) and XMIT_2_IDLES shall be set to TRUE(1);

  – if ARByx = ARB(val) where val > AL_PA of the L_Port, the CFW shall be changed to ARB(AL_PA) (where AL_PA is the AL_PA of the L_Port); or,

  – if ARByx <> ARB(val), the CFW should not be changed.[6]

— if XMIT_2_IDLES is TRUE(1) and the CFW is Idle, the CFW shall not be changed.

If a Fill Word is to be transmitted, the CFW shall be used. If the CFW is Idle and XMIT_2_IDLES is TRUE(1), XMIT_2_IDLES shall be set to FALSE(0) after two Idles are transmitted.

If ARB(val) is received, where val = AL_PA of the L_Port, the LPSM shall make the transition to the ARBITRATION WON state (see item 5 and item 15).

If REPLICATE is TRUE(1), the LPSM shall receive (i.e., present to the FC-2 of the NL_Port for further processing) and retransmit all Transmission Words (except for normal Fill Word processing - updating the CFW appropriately);

NOTE — To avoid a "broadcast storm", an FL_Port does not propagate any Transmission Words into the Fabric.

If OPNfr is received, the LPSM of the NL_Port shall set REPLICATE to TRUE(1) and shall retransmit the received OPNfr.

If OPNyr is received, where y = AL_PA of the NL_Port, the LPSM shall set REPLICATE to TRUE(1) and shall retransmit the received OPNyr.

If OPNy is received, where y = AL_PA of the L_Port, the LPSM shall make the transition to the OPENED state (see item 7 and item 17).

If any other OPNy or OPNr is received, it shall be retransmitted.

If CLS is received and REPLICATE is TRUE(1), REPLICATE shall be set to FALSE(0). The CLS shall be retransmitted.

[6]Some L_Ports may set the CFW to the received ARByx, this practice is not recommended.

If MRKtx is received and:

— if x = AL_PA of the L_Port, the LPSM shall transmit the CFW; the MRKtx is discarded;

— if the MK_TP and AL_PS match the expected values, the action identified by MK_TP shall be performed; or,

— if x <> AL_PA of the L_Port, the received MRKtx shall be retransmitted.

If LIP is recognized, the LPSM shall make the transition to the OPEN-INIT-START state (see item 1, item 21, and clause 10).

If LPByx (y = AL_PA of the L_Port) or LPBfx is recognized, the LPSM shall set BYPASS to TRUE(1) and shall make the transition to the MONITORING state (see item 12 and item 13).

Invalid Transmission Word substitution shall be performed as specified in 8.3.1; any other received Transmission Words shall be retransmitted on the Loop.

If the LPSM detects a Loop Failure on its inbound fibre or the L_Port requests initialization (REQ(initialize)), the LPSM shall make the transition to the INITIALIZATION process (see item 1, item 21, and clause 10).

If the L_Port requests to transmit a MRKtx (REQ(mark as tx)), the LPSM shall transmit one MRKtx at the next Fill Word (see clause 7), unless REQ(mark as tx) is removed before MRKtx is transmitted.

15 **State 2 (ARBITRATION WON)[7] actions** (table 6 and the following text describe the ARBITRATION WON state): This is a transition state during which Transmission Words are not received.

The ARB(val) which was received in the ARBITRATING state, is replaced with the appropriate OPN as follows:

— if the L_Port requires access to the Loop (REQ(open yx), REQ(open yy), REQ(open fr), or REQ(open yr)), the LPSM shall transmit OPNy or the requested OPNr and shall make the transition to the OPEN state (see item 6 and item 16). If OPNr is transmitted, REPLICATE shall be set to TRUE(1).

— if the L_Port does not need access to the Loop (REQ(close)), the LPSM shall transmit OPNyy (where y = AL_PA of the L_Port) and shall make the transition to the OPEN state (see item 6 and item 16).

NOTE — The L_Port transitions through the OPEN state in order to properly manage the fairness window.

[7]The ARBITRATION WON state is a documentation artifact and may not be defined in some implementations. In FC-AL (ANSI X3.272:1996), a decision was made in this state whether to open the Loop or not; in this standard, the only decision is whether to open this L_Port (and to close the Loop without sacrificing the fairness window), to open another L_Port (normal operation), or to initialize.

16 **State 3 (OPEN) actions** (table 7 and the following text describe the OPEN state): To identify this as the L_Port that won arbitration, the LPSM shall set ARB_WON to TRUE(1), ARB_PEND to FALSE(0), DUPLEX to TRUE(1), and the CFW to ARB(F0). If the L_Port is using the access fairness algorithm, ACCESS shall be set to FALSE(0); if the L_Port is not using the access fairness algorithm, ACCESS shall be set to TRUE(1). The LPSM shall transmit at least six (6) Ordered Sets (i.e., CFWs and R_RDYs) before transmitting any frames or CLS (see 8.3.4.1).

NOTE — The six (6) Ordered Sets maintain the ANSI X3, FC-PH-x spacing before SOF; R_RDYs allow the OPENED L_Port to transmit frame(s) without using BB_Credit. One R_RDY does not take any extra bandwidth.

The L_Port shall process, and shall not retransmit subsequent Transmission Words received on its inbound fibre. The L_Port shall transmit Primitive Signals, Primitive Sequences, or frames as specified in ANSI X3, FC-PH-x (see 8.3.4).

If Idle is received, the CFW shall be set to Idle and ACCESS shall be set to TRUE(1).

If ARB(F0) is received, the CFW shall be set to Idle and XMIT_2_IDLES shall be set to TRUE(1).

NOTE — Receiving ARB(F0) indicates that no other L_Port is now arbitrating (i.e., no L_Port changed ARB(F0) to ARB(val)).

If a Fill Word is to be transmitted, the CFW shall be used.

NOTE — Since ARB_WON is TRUE(1) in this state, XMIT_2_IDLES will not be set to FALSE(0).

If CLS is received, the LPSM shall make the transition to the RECEIVED CLOSE state (see item 9 and item 19).

If MRKtx is received, where the MK_TP and AL_PS match the expected values, the action identified by MK_TP shall be performed. The received MRKtx shall not be retransmitted.

If REPLICATE is TRUE(1) and the L_Port requests a broadcast replicate (REQ(open fr) or another selective replicate REQ(open yr)), the LPSM shall transmit OPN(fr) or one OPN(yr) for each request at the next appropriate Fill Word, respectively.

If ACCESS is TRUE(1) and the L_Port requests a transfer (REQ(transfer)), the LPSM shall transmit CLS instead of the appropriate Fill Word and then shall make the transition to the TRANSFER state (see item 10 and item 20). If ACCESS is FALSE(0), the request to transfer is treated as a request to close. If a Class 1 connection exists, the L_Port shall remove the Class 1 connection before transmitting CLS; only the L_Port which received EOFdt shall transmit the first CLS.

The LPSM may begin to close the Loop (REQ(close)) or REQ(send DHD) by transmitting CLS or DHD instead of the next appropriate Fill Word. If CLS is transmitted, the LPSM shall make the transition to the XMITTED CLOSE state or the TRANSFER state. If DHD is transmitted, the LPSM shall remain in the OPEN state, shall set DUPLEX to FALSE(0) and shall not transmit Data frames. If a Class 1 connection exists, the L_Port shall remove the Class 1 connection before transmitting CLS; only the L_Port which received EOFdt shall transmit CLS (see item 8 and item 18 or item 20).

NOTE — Reasons for transmitting CLS or DHD include, but are not limited to:

- ARB(val) was detected to indicate that another L_Port is arbitrating (the OPEN L_Port may close the Loop at a convenient time);
- frame transmission is required with a different L_Port;
- the L_Port has not received any credit to transmit frames before a timeout occurred (an appropriate value would be AL_TIME since this L_Port is the originator of the Loop circuit);
- there are no additional frames to transmit to the other L_Port; or,
- the L_Port is making the transition to the Non-Participating mode.

If LIP is recognized, the LPSM shall make the transition to the OPEN-INIT-START state (see item 1, item 21, and clause 10).

If LPB is recognized:

— if x = AL_PA of the L_Port, the L_Port should stop transmitting LPB; the received LPByx shall be discarded, or

— if y = AL_PA of the L_Port or hex 'FF', the LPSM shall set BYPASS to TRUE(1) and transition to the MONITORING state (see item 12 and item 13).

If REPLICATE is TRUE(1), received Transmission Words shall not be forwarded; they shall either be processed or be discarded.

If the L_Port requests another L_Port to be bypassed (REQ(bypass L_Port y) or REQ(bypass all)) or enabled (REQ(enable L_Port y) or REQ(enable all)), the LPSM shall begin to transmit LPB or LPE at the next Fill Word, until the Primitive Sequence is received.

If the LPSM detects a Loop Failure on its inbound fibre or the L_Port requests initialization (REQ(initialize)), the LPSM shall make the transition to the INITIALIZATION process (see item 1, item 21, and clause 10).

If the L_Port requests to transmit a MRKtx (REQ(mark as tx)), the LPSM shall transmit one MRKtx at the next Fill Word (see clause 7), unless REQ(mark as tx) is removed before MRKtx is transmitted.

17 **State 4 (OPENED) actions** (table 8 and the following text describe the OPENED state): The LPSM shall set ARB_WON to FALSE(0), REPLICATE to FALSE(0), DHD_RCV to FALSE(0), and shall transmit the CFW to replace the received OPNy. The L_Port shall transmit at least six (6) Ordered Sets (CFWs and R_RDYs) before transmitting any frames or CLS (see 8.3.4.1). If the opened BB_Credit is zero (0), and the L_Port has no available receive buffers, the L_Port may transmit CLS and make the transition to the XMITTED CLOSE state (see item 8 and item18). The L_Port shall process, and shall not retransmit subsequent Transmission Words received on its inbound fibre. The L_Port shall transmit Primitive Signals, Primitive Sequences, or frames as specified in ANSI X3, FC-PH-x (see 8.3.4). If OPNyx was received, DUPLEX shall be set to TRUE(1); if OPNyy was received, DUPLEX shall be set to FALSE(0) and no Data frames shall be transmitted.

If ARB_PEND is TRUE(1) and if the L_Port no longer needs access to the Loop (e.g., it was able to complete the transmission of all its frames), then the L_Port may set ARB_PEND to FALSE(0). Subsequent to setting ARB_PEND to FALSE(0), the L_Port shall change the CFW to the next received Fill Word and shall not transmit a CLS until a minimum of six (6) CFWs have been transmitted. Normal LPSM processing for CFW and XMIT_2_IDLES shall resume.

If Idle is received and:

— if ARB_PEND is FALSE(0), the CFW shall be set to Idle and ACCESS shall be set to TRUE(1) or

— if ARB_PEND is TRUE(1) and:

  – if XMIT_2_IDLES is FALSE(0), the CFW shall be changed to ARB(val) (where val = AL_PA of the L_Port) or

  – if XMIT_2_IDLES is TRUE(1), the CFW shall be changed to Idle.

If ARB(F0) is received and:

— if the CFW is not Idle or XMIT_2_IDLES is FALSE(0), XMIT_2_IDLES shall be set to TRUE(1) and:

  – if ARB_PEND is FALSE, the CFW shall be set to ARB(F0) or

  – if ARB_PEND is TRUE(1), the CFW shall be set to ARB(AL_PA) (where AL_PA is the AL_PA of the L_Port)

— if the CFW is Idle and XMIT_2_IDLES is TRUE(1), the CFW shall not be changed.

If ARByx is received and ARByx <> ARB(val), the CFW should not be changed.[8]

If ARByx is received, where ARByx = ARB(val) and either XMIT_2_IDLES is FALSE(0) or the CFW is not Idle, then:

— if ARB_PEND is FALSE(0), the CFW shall be modified as follows:

  – if ARByx = ARB(val) where val = AL_PA of the L_Port, the CFW shall be changed to Idle or

  – if ARByx = ARB(val) where val <> AL_PA of the L_Port, the CFW shall be changed to the received ARB(val).

— if ARB_PEND is TRUE(1), the CFW shall be modified as follows:

  – if ARByx = ARB(val) where val >= AL_PA of the L_Port, the CFW shall be changed to ARB(AL_PA) (where AL_PA is the AL_PA of the L_Port) or

  – if ARByx = ARB(val) where val < AL_PA of the L_Port, the CFW shall be changed to the received ARB(val).

[8]Some L_Ports may set the CFW to the received ARByx, this practice is not recommended.

If a Fill Word is to be transmitted, the CFW shall be used. If the CFW is Idle and XMIT_2_IDLES is TRUE(1), XMIT_2_IDLES shall be set to FALSE(0) after two Idles are transmitted.

If OPNr or OPNy are received, they shall be discarded.

If DHD is received and if DHD is supported, the LPSM shall set DHD_RCV to TRUE(1). Receiving DHD is an indication to this L_Port that the LPSM in the OPEN state has no more Data frames to transmit. The L_Port shall respond to DHD by transmitting frames or CLS.

If DHD_RCV is TRUE(1) and the L_Port has completed all transfers (or it had nothing to transmit when it received DHD) to the L_Port in the OPEN state, it shall REQ(close) to begin closing the Loop circuit (see annex C).

If CLS is received, the LPSM shall make the transition to the RECEIVED CLOSE state (see item 9 and item 19).

If MRKtx is received, where the MK_TP and AL_PS match the expected values, the action identified by MK_TP shall be performed. The received MRKtx shall not be retransmitted.

The LPSM may begin to close the Loop (REQ(close)) by transmitting CLS instead of the next appropriate Fill Word and then shall make the transition to the XMITTED CLOSE state. If a Class 1 connection exists, the L_Port shall remove the Class 1 connection before transmitting CLS; only the L_Port which received EOFdt shall transmit the first CLS (see item 8 and item 18).

NOTE — Before transmitting CLS and transitioning to the XMITTED CLOSE state, the L_Port should ensure that it has enough buffers for the current outstanding credit plus the maximum BB_Credit which the L_Port distributed during Login..

Reasons for transmitting CLS include, but are not limited to:

- frame transmission is required with a different L_Port;
- the L_Port was opened full-duplex and has not received any credit to transmit frames before a timeout occurred (an appropriate value would be LP_TOV since this L_Port is the responder in the Loop circuit);
- the L_Port was opened half-duplex and the L_Port has Data frames to transmit to the other L_Port; or,
- the L_Port is making the transition to the Non-Participating mode.

If LIP is recognized, the LPSM shall make the transition to the OPEN-INIT-START state (see item 1, item 21, and clause 10).

If LPByx (y = AL_PA of the L_Port) or LPBfx is recognized, the LPSM shall set BYPASS to TRUE(1), and transition to the MONITORING state (see item 12 and item 13).

If the LPSM detects a Loop Failure on its inbound fibre or the L_Port requests initialization (REQ(initialize)), the LPSM shall make the transition to the INITIALIZATION process (see item 1, item 21, and clause 10).

If the L_Port requests to transmit a MRKtx (REQ(mark as tx)), the LPSM shall transmit one MRKtx at the next Fill Word (see clause 7), unless REQ(mark as tx) is removed before MRKtx is transmitted.

18 **State 5 (XMITTED CLOSE) actions** (table 9 and the following text describe the XMITTED CLOSE state): The LPSM shall set DUPLEX to FALSE(0) and transmit only the CFW (except MRKtx). The L_Port shall process, but shall not retransmit subsequent Transmission Words received on its inbound fibre (except MRKtx).

If the L_Port does not receive CLS in less than LP_TOV, the LPSM may make the transition to the NORMAL-INITIALIZE state to transmit LIP(F7).

If Idle is received and:

— if ARB_PEND is FALSE(0), the CFW shall be set to Idle and ACCESS shall be set to TRUE(1) or

— if ARB_PEND is TRUE(1) and:

  – if XMIT_2_IDLES is FALSE(0), the CFW shall be changed to ARB(val) (where val = AL_PA of the L_Port) or

  – if XMIT_2_IDLES is TRUE(1), the CFW shall be changed to Idle.

If ARB(F0) is received and:

— if the CFW is not Idle or XMIT_2_IDLES is FALSE(0), XMIT_2_IDLES shall be set to TRUE(1) and:

  – if ARB_WON is TRUE(1), the CFW shall be set to Idle or

    NOTE — Receiving ARB(F0) indicates that no other L_Port is now arbitrating (i.e., no L_Port changed ARB(F0) to ARB(val)).

  – if ARB_WON is FALSE(0), the CFW shall be modified as follows:

    - if ARB_PEND is FALSE(0), the CFW shall be changed to ARB(F0) or

    - if ARB_PEND is TRUE(1), the CFW shall be changed to ARB(val) (where val = AL_PA of the L_Port).

— if the CFW is Idle and XMIT_2_IDLES is TRUE(1), the CFW shall not be changed.

If ARByx is received and ARByx <> ARB(val) (i.e., y <> x), the CFW should not be changed.[9]

If ARByx is received, ARB_WON is FALSE(0), ARB_PEND is TRUE(1), and either XMIT_2_IDLES is FALSE(0) or the CFW is not Idle, then the CFW shall be modified as follows:

— if ARByx = ARB(val) where val >= AL_PA of the L_Port, the CFW shall be changed to ARB(AL_PA) (where AL_PA is the AL_PA of the L_Port) or

— if ARByx = ARB(val) where val < AL_PA of the L_Port, the CFW shall be changed to the received ARB(val).

If ARByx is received, ARB_WON is FALSE(0), ARB_PEND is FALSE(0), and either XMIT_2_IDLES is FALSE(0) or the CFW is not Idle, then the CFW shall be modified as follows:

— if ARByx = ARB(val) where val = AL_PA of the L_Port, the CFW shall be changed to Idle or

— if ARByx = ARB(val) where val <> AL_PA of the L_Port, the CFW shall be changed to the received ARB(val).

If a Fill Word is to be transmitted, the CFW shall be used. If the CFW is Idle, XMIT_2_IDLES is TRUE(1) and ARB_WON is FALSE(0), XMIT_2_IDLES shall be set to FALSE(0) after two Idles are transmitted.

NOTE — When ARB_WON is TRUE(1) in this state, XMIT_2_IDLES will not be set to FALSE(0).

[9]Some L_Ports may set the CFW to the received ARByx, this practice is not recommended.

If CLS is received and:

— if ARB_PEND is FALSE(0), the LPSM shall transmit the CFW and shall make the transition to the MONITORING state (see item 12 and item13).

NOTE — If ARB_WON is TRUE(1) and if the L_Port had advertised a BB_Credit > 0, in order to avoid any over-runs, it is advisable that the number of available buffers at least equal BB_Credit before making the transition to the MONITORING state.

— if ARB_PEND is TRUE(1), the LPSM shall transmit the CFW and shall make the transition to the ARBITRATING state (see item 4 and item 13).

If MRKtx is received and:

— if x = AL_PA of the L_Port, the LPSM shall transmit the CFW; the MRKtx is discarded;

— if the MK_TP and AL_PS match the expected values, the action identified by MK_TP shall be performed; or,

— if x <> AL_PA of the L_Port, the received MRKtx shall be retransmitted.

If LIP is recognized, the LPSM shall make the transition to the OPEN-INIT-START state (see item 1, item 21, and clause 10).

If LPByx (y = AL_PA of the L_Port) or LPBfx is recognized, the LPSM shall set BYPASS to TRUE(1) and shall make the transition to the MONITORING state (see item 12 and item 13). If any other LPByx is received, it shall be replaced with the CFW.

If the LPSM detects a Loop Failure on its inbound fibre or the L_Port requests initialization (REQ(initialize)), the LPSM shall make the transition to the INITIALIZATION process (see item 1, item 21, and clause 10).

If the L_Port requests to transmit a MRKtx (REQ(mark as tx)), the LPSM shall transmit one MRKtx at the next Fill Word (see clause 7), unless REQ(mark as tx) is removed before MRKtx is transmitted.

19 **State 6 (RECEIVED CLOSE) actions** (table 10 and the following text describe the RECEIVED CLOSE state): The L_Port may continue to transmit frames until Available_BB_Credit or EE_Credit is exhausted. Any frame or R_RDY received from the other L_Port shall be discarded. The LPSM shall process and shall not retransmit subsequent Transmission Words received on its inbound fibre. The L_Port shall transmit Primitive Signals, Primitive Sequences, or frames as specified in ANSI X3, FC-PH-x. The L_Port should promptly transmit any remaining frames (if any) and then transmit CLS.

NOTE — The other L_Port participating in the Loop circuit may timeout receipt of this L_Port's CLS after LP_TOV.

When the LPSM transmits CLS (REQ(close)):

— if ARB_PEND is FALSE(0), the LPSM shall transition to the MONITORING state (see item 12 and item 13).

NOTE — Before transmitting CLS, if the L_Port had advertised a BB_Credit > 0, in order to avoid any overruns, it is advisable that the number of available buffers at least equal BB_Credit before making the transition to the MONITORING state.

— if ARB_PEND is TRUE(1), the LPSM shall transition to the ARBITRATING state (see item 4 and item 13).

If ARB_PEND is TRUE(1) and if the L_Port no longer needs access to the Loop (e.g., it was able to complete the transmission of all its frames), then the L_Port may set ARB_PEND to FALSE(0). Subsequent to setting ARB_PEND to FALSE(0), the L_Port shall change the CFW to the next received Fill Word and then transmit a minimum of six (6) CFWs before transmitting CLS.

If Idle is received and:

— if ARB_PEND is FALSE(0), the CFW shall be set to Idle and ACCESS shall be set to TRUE(1) or

— if ARB_PEND is TRUE(1) and:

  – if XMIT_2_IDLES is FALSE(0), the CFW shall be changed to ARB(val) (where val = AL_PA of the L_Port) or

  – if XMIT_2_IDLES is TRUE(1), the CFW shall be changed to Idle.

If ARB(F0) is received and:

— if the CFW is not Idle or XMIT_2_IDLES is FALSE(0), XMIT_2_IDLES shall be set to TRUE(1) and:

  – if ARB_WON is TRUE(1), the CFW shall be set to Idle or

    NOTE — Receiving ARB(F0) indicates that no other L_Port is now arbitrating (i.e., no L_Port changed ARB(F0) to ARB(val)).

  – if ARB_WON is FALSE(0), the CFW shall be modified as follows:

    - if ARB_PEND is FALSE(0), the CFW shall be changed to ARB(F0) or

    - if ARB_PEND is TRUE(1), the CFW shall be changed to ARB(val) (where val = AL_PA of the L_Port).

— if the CFW is Idle and XMIT_2_IDLES is TRUE(1), the CFW shall not be changed.

If ARByx is received and ARByx <> ARB(val) (i.e., y <> x), the CFW should not be changed.[10]

If ARByx is received, ARB_WON is FALSE(0), ARB_PEND is TRUE(1), and either XMIT_2_IDLES is FALSE(0) or the CFW is not Idle, then the CFW shall be modified as follows:

[10]Some L_Ports may set the CFW to the received ARByx, this practice is not recommended.

— if ARByx = ARB(val) where val >= AL_PA of the L_Port, the CFW shall be changed to ARB(AL_PA) (where AL_PA is the AL_PA of the L_Port) or

— if ARByx = ARB(val) where val < AL_PA of the L_Port, the CFW shall be changed to the received ARB(val).

If ARByx is received, ARB_WON is FALSE(0), ARB_PEND is FALSE(0), and either XMIT_2_IDLES is FALSE(0) or the CFW is not Idle, then the CFW shall be modified as follows:

— if ARByx = ARB(val) where val = AL_PA of the L_Port, the CFW shall be changed to Idle or

— if ARByx = ARB(val) where val <> AL_PA of the L_Port, the CFW shall be changed to the received ARB(val).

If a Fill Word is to be transmitted, the CFW shall be used. If the CFW is Idle; XMIT_2_IDLES is TRUE(1); and, ARB_WON is FALSE(0), XMIT_2_IDLES shall be set to FALSE(0) after two Idles are transmitted.

NOTE — When ARB_WON is TRUE(1) in this state, XMIT_2_IDLES will not be set to FALSE(0).

If MRKtx is received, where the MK_TP and AL_PS match the expected values, the action identified by MK_TP shall be performed. The received MRKtx shall not be retransmitted.

If LIP is recognized, the LPSM shall make the transition to the OPEN-INIT-START state (see item 1, item 21, and clause 10).

If LPByx (y = AL_PA of the L_Port) or LPBfx is recognized, the LPSM shall set BYPASS to TRUE(1) and transition to the MONITORING state (see item 12 and item 13).

If the LPSM detects a Loop Failure on its inbound fibre or the L_Port requests initialization (REQ(initialize)), the LPSM shall make the transition to the INITIALIZATION process (see item 1, item 21, and clause 10).

If the L_Port requests to transmit a MRKtx (REQ(mark as tx)), the LPSM shall transmit one MRKtx at the next Fill Word (see clause 7), unless REQ(mark as tx) is removed before MRKtx is transmitted.

20 **State 7 (TRANSFER) actions** (table 11 and the following text describe the TRANSFER state): The L_Port shall set DUPLEX to FALSE(0). The LPSM shall transmit only the CFW (except MRKtx). The L_Port shall process, but shall not retransmit subsequent Transmission Words received on its inbound fibre (except MRKtx).

If the L_Port does not receive CLS in less than LP_TOV, the LPSM may make the transition to the NORMAL-INITIALIZE state to transmit LIP(F7).

If Idle is received, the CFW shall be set to Idle and ACCESS shall be set to TRUE(1).

If ARB(F0) is received, the CFW shall be set to Idle and XMIT_2_IDLES shall be set to TRUE(1).

NOTE — Receiving ARB(F0) indicates that no other L_Port is now arbitrating (i.e., no L_Port changed ARB(F0) to ARB(val)).

If a Fill Word is to be transmitted, the CFW shall be used.

NOTE — Since ARB_WON is TRUE(1) in this state, XMIT_2_IDLES will not be set to FALSE(0).

If CLS is received and:

— the L_Port still requires access to the Loop (REQ(open yx) or REQ(open yy), the LPSM shall transmit OPNy to replace the received CLS, shall set REPLICATE to FALSE(0), and shall make the transition to the OPEN state (see item 11 and 16);

— the L_Port still requires access to the Loop (REQ(open fr) or REQ(open yr)), the LPSM shall transmit OPNr to replace the received CLS, shall set REPLICATE to TRUE(1), and shall make the transition to the OPEN state (see item 11 and 16); or,

— the L_Port no longer needs access to the Loop (REQ(monitor)), the LPSM shall make the transition to the MONITORING state (see item 12 and item 13).

NOTE — If the L_Port had advertised a BB_Credit > 0, in order to avoid any overruns, it is advisable that the number of available buffers at least equal BB_Credit before making the transition to the OPEN or MONITORING state.

If MRKtx is received and:

— if x = AL_PA of the L_Port, the LPSM shall transmit the CFW; the MRKtx is discarded;

— if the MK_TP and AL_PS match the expected values, the action identified by MK_TP shall be performed; or,

— if x <> AL_PA of the L_Port, the received MRKtx shall be retransmitted.

If LIP is recognized, the LPSM shall make the transition to the OPEN-INIT-START state (see item 1, item 21, and clause 10).

If LPByx (y = AL_PA of the L_Port) or LPBfx is recognized, the LPSM shall set BYPASS to TRUE(1) and shall make the transition to the MONITORING state (see item 12 and item 13). If any other LPByx is received, it shall be replaced by the CFW.

If the LPSM detects a Loop Failure on its inbound fibre or the L_Port requests initialization (REQ(initialize)), the LPSM shall make the transition to the INITIALIZATION process (see item 1, item 21, and clause 10).

If the L_Port requests to transmit a MRKtx (REQ(mark as tx)), the LPSM shall transmit one MRKtx at the next Fill Word (see clause 7), unless REQ(mark as tx) is removed before MRKtx is transmitted.

21 **Process 8 (INITIALIZATION process) actions** (table 12 and the following text describe the entry actions for the INITIALIZATION process, and the state diagrams and text of 10.5.4 describe the detailed actions in these states): The LPSM shall set BYPASS to FALSE(0); shall transmit the Transmissions Words as defined in 10.5.4; and, shall not retransmit received Transmission Words except LPB and LPE or as required by 10.5.4.

NOTE — The INITIALIZATION process encompasses the INITIALIZING state that was defined in the first publication of this standard.

22 **Reserved**

NOTE — The INITIALIZATION process encompasses the OPEN-INIT state that was defined in the first publication of this standard.

23 **State A (OLD-PORT) actions** (table 14 and the following text describe the entry actions to the optional OLD-PORT state; the state diagrams and text of 10.5.4 describe the detailed actions in the OLD-PORT state): The LPSM shall set the CFW to Idle; the L_Port shall process, but the LPSM shall not retransmit Transmission Words received on its inbound fibre. The L_Port shall transmit Primitive Signals, Primitive Sequences, or frames as specified in ANSI X3, FC-PH-x. Before Login[11], the "Alternate BB_Credit Management" bit shall be set to *0* and the BB_Credit shall be set to one (1).

[11]Private NL_Ports only support NL_Port Login.

9 L_Port state transition tables

The following tables provide complimentary and necessary information to the text in 8.4. The tables show the transitions from one state to the next in the LPSM that are based directly on receipt of information from the inbound fibre or from L_Port controls. All inputs affecting the LPSM are repeated in each table to provide an exhaustive list of related outputs and transitions.

The following notations, conventions, and abbreviations are used in table 4 through table 14:

— The ENTRY ACTIONS in the top block of tables 4-14 shall be completed before the LPSM shall accept any condition specified in the column labeled 'INPUT.' The column labeled 'ACTION / OUTPUT' typically represents the action taken based on the received Transmission Word and identifies the next Transmission Word which shall be transmitted on the Loop. In addition, there is minimal descriptive text as to what action should be taken (e.g., 'Receive Word' states that this Transmission Word is to be received by the L_Port).

— When XMIT_2_IDLES is TRUE(1), ARB_WON is FALSE(0), and the LPSM has transmitted two (2) Idles, the LPSM shall set XMIT_2_IDLES to FALSE(0) (see 8.1.1 and 8.4.3).

— L_Port requests typically cause Transmission Words to be transmitted asynchronously to the request. Therefore, the column labeled 'ACTION / OUTPUT' for L_Port requests may not relate to a Transmission Word.

— 'None/Inst.' indicates that no Transmission Word is transmitted in this state.

— 'when ... BB_Credit' implies that the number of receive buffers equals the advertised BB_Credit Login value of the L_Port.

— 'same word' implies that the Transmission Word which was received is retransmitted.

— 'receive word' implies that the Transmission Word is presented to FC-2 of the L_Port for further processing.

— Each Primitive Sequence entry represents the point of recognition (i.e., the third consecutive Transmission Word of the same Ordered Set has been received).

— If PARTICIPATE is FALSE(0), the L_Port does not have an AL_PA. Therefore, the entries for val = AL_PA, x = AL_PA, or y = AL_PA are ignored and the entries for val <> AL_PA, x<> AL_PA, or y <> AL_PA shall be followed.

— Requests to an L_Port which are not appropriate inputs may be ignored. Some implementations may choose to report an error status to the L_Port for these requests.

— The entry labeled "ANY OTHER O.S." addresses the first and second Ordered Set of each Primitive Sequence. (See ANSI X3, FC-PH-x, 16.4.1.)

— The entry labeled "ANY OTHER O.S." is used to process an Ordered Set which is a valid Transmission Word, but which is not specifically accounted for in the state tables. This allows new Ordered Sets to be defined without disrupting previous implementations.

— Entries "at the next Fill Word" and "at the next app. Fill Word" pertain to Fill Words at the L_Port's output.

app. appropriate
DISP Disparity
FC-2 ANSI X3, FC-PH-x FC-2 Protocol
FP Framing Protocol
f-d full-duplex
h-d half-duplex
Inst. Instantaneous (i.e., no Transmission Words are transmitted in this state)
N/A Not Applicable to this state
N/C No Change
Opt. Optionally
O.S. Ordered Set
PSeq FC-PH Primitive Sequence(s)
PSig FC-PH Primitive Signal(s)
REQd Required

Table 4 — MONITORING (State 0) transitions

ENTRY ACTIONS		
ACCESS := N/C	DUPLEX := 0	DHD_RCV := N/C
ARB_PEND := 0	REPLICATE := 0	BYPASS := N/C
ARB_WON := 0	CFW := N/C	ERR_INIT := 0
ARBf_SENT := 0		XMIT_2_IDLES := N/C

INPUT	ACTION / OUTPUT	NEXT STATE
LOSS of SYNC. < R_T_TOV	Idle or CFW[12]	MONITORING
Loop Failure		
REPEAT = 0	None/Inst.	LOOP-FAIL-INITIALIZE
REPEAT = 1	LIP(F8)	MONITORING
INVALID TRANS. WORD	CFW	MONITORING
RUNNING DISP at O.S.	CFW	MONITORING
ELASTICITY WORD REQd	CFW	MONITORING
VALID DATA WORD		
FL_Port	Same Word	MONITORING
NL_Port		
REPLICATE = 0	Same Word	MONITORING
REPLICATE = 1	Receive Word Same Word	MONITORING
VALID TRANS. WORD = O.S.		
FRAME DELIMITER		
FL_Port	Same Word	MONITORING
NL_Port		
SOFxx		
REPLICATE = 0	Same Word	MONITORING
REPLICATE = 1	Receive Word Same Word	MONITORING
EOFxx		
REPLICATE = 0	Same Word	MONITORING
REPLICATE = 1	Receive Word Same Word	MONITORING
PRIMITIVE SIGNALS		
Idle		
REPEAT = 0		
ARBf_SENT = 0	CFW := Idle ACCESS := 1 CFW	MONITORING
ARBf_SENT = 1	CFW := ARB(FF) CFW	MONITORING
REPEAT = 1	CFW := Idle CFW	MONITORING
R_RDY	Same Word	MONITORING
ARByx		
y <> x	CFW[13]	MONITORING

[12]The previous version of this standard used Idle for the Output. Implementors felt it was simpler to use the current Fill Word.

[13]Some L_Ports may set the CFW to the received ARByx, however, it is recommended that the CFW is not changed.

INPUT	ACTION / OUTPUT	NEXT STATE
ARB(FF) REPEAT = 0 CFW <> Idle | XMIT_2_IDLES = 0 CFW = Idle & XMIT_2_IDLES = 1 REPEAT = 1	 CFW := ARB(FF) CFW CFW CFW := ARB(FF) CFW	 MONITORING MONITORING MONITORING
ARB(F0) CFW = Idle | ARB(FF) REPEAT = 0 REPEAT = 1 CFW <> Idle | ARB(FF)	 XMIT_2_IDLES :=1 CFW CFW := ARB(F0) CFW CFW := ARB(F0) ARBf_SENT := 0 XMIT_2_IDLES := 1 CFW	 MONITORING MONITORING MONITORING
ARB(val) REPEAT = 0 val <> AL_PA CFW <> Idle | XMIT_2_IDLES = 0 CFW = Idle & XMIT_2_IDLES = 1 val = AL_PA REPEAT = 1	 CFW := ARB(val) ARBf_SENT := 0 CFW ARBf_SENT := 0 CFW CFW := Idle ARBf_SENT := 0 CFW CFW := ARB(val) CFW	 MONITORING MONITORING MONITORING MONITORING
OPNr (OPNfr | OPNyr) PARTICIPATE = 1 FL_Port NL_Port f = hex 'FF' REPEAT = 0 REPEAT = 1 y = AL_PA REPEAT = 0 REPEAT = 1 All other OPNr PARTICIPATE = 0	 Same Word REPLICATE := 1 Same Word Same Word REPLICATE := 1 Same Word Same Word Same Word Same Word	 MONITORING MONITORING MONITORING MONITORING MONITORING MONITORING MONITORING
OPNy REPEAT = 0 y = AL_PA y <>AL_PA REPEAT = 1	 None/Inst. Same Word Same Word	 OPENED MONITORING MONITORING

INPUT	ACTION / OUTPUT	NEXT STATE
CLS		
REPLICATE = 0	Same Word	MONITORING
REPLICATE = 1	REPLICATE := 0	
	Same Word	MONITORING
DHD	Same Word	MONITORING
MRKtx		
REPEAT = 0		
x = AL_PA	CFW	MONITORING
x <>AL_PA	Same Word	MONITORING
REPEAT = 1	Same Word	MONITORING
PRIMITIVE SEQUENCES		
LIP		
BYPASS = 0	None/Inst.	OPEN-INIT-START
BYPASS = 1	PARTICIPATE := 0	
	ARBf_SENT := 0	
	Same Word	MONITORING
LPB (LPByx | LPBfx)		
x = AL_PA		
REPEAT = 0	CFW	MONITORING
REPEAT = 1	Same Word	MONITORING
y <>AL_PA	Same Word	MONITORING
y = AL_PA	REPLICATE := 0	
	BYPASS := 1	
	ARBf_SENT := 0	
	Same Word	MONITORING
f = hex 'FF'	REPLICATE := 0	
	BYPASS := 1	
	ARBf_SENT := 0	
	Same Word	MONITORING
LPE (LPEyx | LPEfx)		
x = AL_PA		
REPEAT = 0	CFW	MONITORING
REPEAT = 1	Same Word	MONITORING
y <>AL_PA	Same Word	MONITORING
y = AL_PA	BYPASS := 0	
	Same Word	MONITORING
f = hex 'FF'	BYPASS := 0	
	Same Word	MONITORING
ANY OTHER O.S.	Same Word	MONITORING

INPUT	ACTION / OUTPUT	NEXT STATE
L_Port CONTROLS		
REQ(monitor)	None/Inst.	MONITORING
REQ(arb own AL_PA) ACCESS = 0 ACCESS = 1 REPEAT = 0 REPEAT = 1	 None/Inst. None/Inst. None/Inst.	 MONITORING ARBITRATING MONITORING
REQ(arbitrate FF) REPEAT = 0 CFW = Idle CFW <> Idle REPEAT = 1	 ARBf_SENT := 1 (when 6 Idles sent) None/Inst. None/Inst.	 MONITORING MONITORING MONITORING
REQ(open yx) f-d	None/Inst.	MONITORING
REQ(open yy) h-d	None/Inst.	MONITORING
REQ(open fr)	None/Inst.	MONITORING
REQ(open yr)	None/Inst.	MONITORING
REQ(close)	None/Inst.	MONITORING
REQ(send DHD)	None/Inst.	MONITORING
REQ(transfer)	None/Inst.	MONITORING
REQ(old-port)	None/Inst.	OLD-PORT-REQ
REQ(participating)	None/Inst.	NORMAL-INITIALIZE
REQ(nonparticipat.)	Optionally Transmit 12 LIPs PARTICIPATE := 0 ARBf_SENT := 0	 MONITORING
REQ(mark as tx) REPEAT = 0 REPEAT = 1	 MRKtx at the next Fill Word None/Inst.	 MONITORING MONITORING
REQ(bypass L_Port)	REPLICATE := 0 BYPASS := 1 ARBf_SENT := 0	 MONITORING
REQ(bypass L_Port y)	None/Inst.	MONITORING
REQ(bypass all)	None/Inst.	MONITORING
REQ(enable L_Port)	BYPASS := 0	MONITORING
REQ(enable L_Port y)	None/Inst.	MONITORING
REQ(enable all)	None/Inst.	MONITORING
REQ(initialize)	None/Inst.	NORMAL-INITIALIZE

Table 5 — ARBITRATING (State 1) transitions

ENTRY ACTIONS		
ACCESS := N/C	DUPLEX := 0	DHD_RCV := N/C
ARB_PEND := N/C	REPLICATE := N/C	BYPASS := N/C
ARB_WON := 0	CFW := N/C	ERR_INIT := N/C
ARBf_SENT := N/C		XMIT_2_IDLES := N/C

INPUT	ACTION / OUTPUT	NEXT STATE
LOSS of SYNC. < R_T_TOV	Idle or CFW	ARBITRATING
Loop Failure	None/Inst.	LOOP-FAIL-INITIALIZE
INVALID TRANS. WORD	CFW	ARBITRATING
RUNNING DISP at O.S.	CFW	ARBITRATING
ELASTICITY WORD REQd	CFW	ARBITRATING
VALID DATA WORD FL_Port NL_Port: REPLICATE = 0 REPLICATE = 1	 Same Word Same Word Receive Word Same Word	 ARBITRATING ARBITRATING ARBITRATING
VALID TRANS. WORD =O.S.		
FRAME DELIMITER		
FL_Port NL_Port: SOFxx REPLICATE = 0 REPLICATE = 1 EOFxx REPLICATE = 0 REPLICATE = 1	Same Word Same Word Receive Word Same Word Same Word Receive Word Same Word	ARBITRATING ARBITRATING ARBITRATING ARBITRATING ARBITRATING
PRIMITIVE SIGNALS		
Idle XMIT_2_IDLES = 0 XMIT_2_IDLES = 1	 CFW := ARB(AL_PA) ARB_PEND := 1 CFW CFW := Idle CFW	 ARBITRATING ARBITRATING
R_RDY	Same Word	ARBITRATING
ARByx y <> x	 CFW[14]	 ARBITRATING

[14] While some L_Ports may set the CFW to the received ARByx, it is recommended that the CFW is not changed.

INPUT	ACTION / OUTPUT	NEXT STATE
ARB(val) CFW <> IDLE | XMIT_2_IDLES = 0 val < AL_PA val = hex 'F0' val > AL_PA val = AL_PA CFW = IDLE & XMIT_2_IDLES = 1	 CFW := ARB(val) CFW CFW := ARB(AL_PA) ARB_PEND := 1 XMIT_2_IDLES := 1 CFW CFW := ARB(AL_PA) ARB_PEND := 1 CFW None/Inst. CFW	 ARBITRATING ARBITRATING ARBITRATING ARBITRATION WON ARBITRATING
OPNr (OPNfr | OPNyr) FL_Port NL_Port f = hex 'FF' y = AL_PA All other OPNr	 Same Word REPLICATE := 1 Same Word REPLICATE := 1 Same Word Same Word	 ARBITRATING ARBITRATING ARBITRATING ARBITRATING
OPNy y = AL_PA y <>AL_PA	 None/Inst. Same Word	 OPENED ARBITRATING
CLS REPLICATE = 0 REPLICATE = 1	 Same Word REPLICATE := 0 Same Word	 ARBITRATING ARBITRATING
DHD	Same Word	ARBITRATING
MRKtx x = AL_PA x <>AL_PA	 CFW Same Word	 ARBITRATING ARBITRATING
PRIMITIVE SEQUENCES		
LIP	None/Inst.	OPEN-INIT-START
LPB (LPByx | LPBfx) x = AL_PA y <>AL_PA y = AL_PA f = hex 'FF'	 CFW Same Word BYPASS := 1 None/Inst. BYPASS := 1 None/Inst.	 ARBITRATING ARBITRATING MONITORING MONITORING
LPE (LPEyx | LPEfx)	Same Word	ARBITRATING
ANY OTHER O.S.	Same Word	ARBITRATING

INPUT	ACTION / OUTPUT	NEXT STATE
L_Port CONTROLS		
REQ(monitor)	None/Inst.	ARBITRATING
REQ(arb own AL_PA)	None/Inst.	ARBITRATING
REQ(open yx) f-d	None/Inst.	ARBITRATING
REQ(open yy) h-d	None/Inst.	ARBITRATING
REQ(open fr)	None/Inst.	ARBITRATING
REQ(open yr)	None/Inst.	ARBITRATING
REQ(close)	None/Inst.	ARBITRATING
REQ(send DHD)	None/Inst.	ARBITRATING
REQ(transfer)	None/Inst.	ARBITRATING
REQ(old-port)	None/Inst.	ARBITRATING
REQ(participating)	None/Inst.	ARBITRATING
REQ(nonparticipat.)	None/Inst.	ARBITRATING
REQ(mark as tx)	MRKtx at the next Fill Word	ARBITRATING
REQ(bypass L_Port)	None/Inst.	ARBITRATING
REQ(bypass L_Port y)	None/Inst.	ARBITRATING
REQ(bypass all)	None/Inst.	ARBITRATING
REQ(enable L_Port)	None/Inst.	ARBITRATING
REQ(enable L_Port y)	None/Inst.	ARBITRATING
REQ(enable all)	None/Inst.	ARBITRATING
REQ(initialize)	None/Inst.	NORMAL-INITIALIZE

Table 6 — ARBITRATION WON (State 2) transitions

ENTRY ACTIONS		
ACCESS := N/C ARB_PEND := N/C ARB_WON := N/C ARBf_SENT := N/C	DUPLEX := N/C REPLICATE := N/C CFW := N/C	DHD_RCV := N/C BYPASS := N/C ERR_INIT := N/C XMIT_2_IDLES := N/C

INPUT	ACTION / OUTPUT	NEXT STATE
LOSS of SYNC. < R_T_TOV	N/A	N/A
Loop Failure	N/A	N/A
INVALID TRANS. WORD	N/A	N/A
RUNNING DISP at O.S.	N/A	N/A
ELASTICITY WORD REQd	N/A	N/A
VALID DATA WORD	N/A	N/A
VALID TRANS. WORD =O.S.		
FRAME DELIMITERS		
SOFxx EOFxx	N/A N/A	N/A N/A
PRIMITIVE SIGNALS		
Idle	N/A	N/A
R_RDY	N/A	N/A
ARByx	N/A	N/A
OPNr	N/A	N/A
OPNy	N/A	N/A
CLS	N/A	N/A
DHD	N/A	N/A
MRKtx	N/A	N/A
PRIMITIVE SEQUENCES		
LIP	N/A	N/A
LPB (LPByx | LPBfx)	N/A	N/A
LPE (LPEyx | LPEfx)	N/A	N/A
ANY OTHER O.S.	N/A	N/A

INPUT	ACTION / OUTPUT	NEXT STATE
L_Port CONTROLS		
REQ(monitor)	N/A	N/A
REQ(arb own AL_PA)	N/A	N/A
REQ(open yx) f-d	OPNyx	OPEN
REQ(open yy) h-d	OPNyy	OPEN
REQ(open fr)	REPLICATE:=1 OPNfr	OPEN
REQ(open yr)	REPLICATE:=1 OPNyr	OPEN
REQ(close)	y := AL_PA of L_Port OPNyy	OPEN
REQ(send DHD)	N/A	N/A
REQ(transfer)	N/A	N/A
REQ(old-port)	N/A	N/A
REQ(participating)	N/A	N/A
REQ(nonparticipat.)	N/A	N/A
REQ(mark as tx)	N/A	N/A
REQ(bypass L_Port)	N/A	N/A
REQ(bypass L_Port y)	N/A	N/A
REQ(bypass all)	N/A	N/A
REQ(enable L_Port)	N/A	N/A
REQ(enable L_Port y)	N/A	N/A
REQ(enable all)	N/A	N/A
REQ(initialize)	None/Inst.	NORMAL-INITIALIZE

Table 7 — OPEN (State 3) transitions

ENTRY ACTIONS		
ACCESS := 0 if using fairness ACCESS := 1 if not using fairness ARB_PEND := 0 ARB_WON := 1 ARBf_SENT := N/C	DUPLEX := 1 REPLICATE := N/C CFW := ARB(F0)	DHD_RCV := N/C BYPASS := N/C ERR_INIT := N/C XMIT_2_IDLES := N/C

INPUT	ACTION / OUTPUT	NEXT STATE
LOSS of SYNC. < R_T_TOV	FC-2 FP/PSig/PSeq	OPEN
Loop Failure	None/Inst.	LOOP-FAIL-INITIALIZE
INVALID TRANS. WORD	FC-2 FP/PSig/PSeq	OPEN
RUNNING DISP at O.S.	FC-2 FP/PSig/PSeq	OPEN
ELASTICITY WORD REQd	N/A	OPEN
VALID DATA WORD	FC-2 FP/PSig/PSeq	OPEN
VALID TRANS. WORD =O.S.		
FRAME DELIMITERS		
SOFxx EOFxx	FC-2 FP/PSig/PSeq FC-2 FP/PSig/PSeq	OPEN OPEN
PRIMITIVE SIGNALS		
Idle	CFW := Idle ACCESS := 1 FC-2 FP/PSig/PSeq	 OPEN
R_RDY	FC-2 FP/PSig/PSeq	OPEN
ARB(F0)	CFW := Idle XMIT_2_IDLES := 1 FC-2 FP/PSig/PSeq	 OPEN
ARByx	FC-2 FP/PSig/PSeq	OPEN
OPNr	FC-2 FP/PSig/PSeq	OPEN
OPNy	FC-2 FP/PSig/PSeq	OPEN
CLS	FC-2 FP/PSig/PSeq	RECEIVED CLOSE
DHD	FC-2 FP/PSig/PSeq	OPEN
MRKtx	FC-2 FP/PSig/PSeq	OPEN
PRIMITIVE SEQUENCES		
LIP	None/Inst.	OPEN-INIT-START
LPB(LPByx | LPBfx) x = AL_PA y <>AL_PA y = AL_PA f = hex 'FF'	 FC-2 FP/PSig/PSeq FC-2 FP/PSig/PSeq BYPASS := 1 None/Inst. BYPASS := 1 None/Inst.	 OPEN OPEN MONITORING MONITORING
LPE (LPEyx | LPEfx)	FC-2 FP/PSig/PSeq	OPEN
ANY OTHER O.S.	FC-2 FP/PSig/PSeq	OPEN

INPUT	ACTION / OUTPUT	NEXT STATE
L_Port CONTROLS		
REQ(monitor)	None/Inst.	OPEN
REQ(arb own AL_PA)	None/Inst.	OPEN
REQ(open yx) full-duplex	None/Inst.	OPEN
REQ(open yy) half-duplex	None/Inst.	OPEN
REQ(open fr) REPLICATE = 0 REPLICATE = 1	 None/Inst. OPNfr at the next app. Fill Word	 OPEN OPEN (when OPNfr sent)
REQ(open yr) REPLICATE = 0 REPLICATE = 1	 None/Inst. OPNyr at the next app. Fill Word	 OPEN OPEN (when OPNyr sent)
REQ(close)	CLS at the next app. Fill Word	XMITTED CLOSE (when CLS sent)
REQ(send DHD) DUPLEX = 0 DUPLEX = 1	 None/Inst. DUPLEX := 0 DHD at the next app. Fill Word	 OPEN OPEN (when DHD sent)
REQ(transfer) ACCESS = 0 ACCESS = 1	 CLS at the next app. Fill Word CLS at the next app. Fill Word	 XMITTED CLOSE (when CLS sent) TRANSFER (when CLS sent)
REQ(old-port)	None/Inst.	OPEN
REQ(participating)	None/Inst.	OPEN
REQ(nonparticipat.)	None/Inst.	OPEN
REQ(mark as tx)	MRKtx at the next Fill Word	OPEN
REQ(bypass L_Port)	None/Inst.	OPEN
REQ(bypass L_Port y)	LPByx at the next Fill Word	OPEN
REQ(bypass all)	LPBfx at the next Fill Word	OPEN
REQ(enable L_Port)	None/Inst.	OPEN
REQ(enable L_Port y)	LPEyx at the next Fill Word	OPEN
REQ(enable all)	LPEfx at the next Fill Word	OPEN
REQ(initialize)	None/Inst.	NORMAL-INITIALIZE

Table 8 — OPENED (State 4) transitions

ENTRY ACTIONS		
ACCESS := N/C	DUPLEX := 0 if OPNyy	DHD_RCV := 0
ARB_PEND := N/C	DUPLEX := 1 if OPNyx	BYPASS := N/C
ARB_WON := 0	REPLICATE := 0	ERR_INIT := N/C
ARBf_SENT := N/C	CFW := N/C	XMIT_2_IDLES := N/C

INPUT	ACTION / OUTPUT	NEXT STATE
LOSS of SYNC. < R_T_TOV	FC-2 FP/PSig/PSeq	OPENED
Loop Failure	None/Inst.	LOOP-FAIL-INITIALIZE
INVALID TRANS. WORD	FC-2 FP/PSig/PSeq	OPENED
RUNNING DISP at O.S.	FC-2 FP/PSig/PSeq	OPENED
ELASTICITY WORD REQd	N/A	OPENED
VALID DATA WORD	FC-2 FP/PSig/PSeq	OPENED
VALID TRANS. WORD = O.S.		
FRAME DELIMITERS		
SOFxx EOFxx	FC-2 FP/PSig/PSeq FC-2 FP/PSig/PSeq	OPENED OPENED
PRIMITIVE SIGNALS		
Idle ARB_PEND = 0 ARB_PEND = 1 XMIT_2_IDLES = 0 XMIT_2_IDLES = 1	 CFW := Idle ACCESS := 1 FC-2 FP/PSig/PSeq CFW := ARB(AL_PA) FC-2 FP/PSig/PSeq CFW := Idle FC-2 FP/PSig/PSeq	 OPENED OPENED OPENED
R_RDY	FC-2 FP/PSig/PSeq	OPENED
ARB(F0) CFW <> Idle | XMIT_2_IDLES = 0 ARB_PEND = 0 ARB_PEND = 1 CFW = Idle & XMIT_2_IDLES = 1	 CFW := ARB(F0) XMIT_2_IDLES := 1 FC-2/FP/PSig/PSeq CFW := ARB(AL_PA) XMIT_2_IDLES := 1 FC-2/FP/PSig/PSeq FC-2/FP/PSig/PSeq	 OPENED OPENED OPENED
ARByx y <> x	 FC-2 FP/PSig/PSeq[15]	 OPENED

[15]While some L_Ports may set the CFW to the received ARByx, it is recommended that the CFW is not changed.

INPUT	ACTION / OUTPUT	NEXT STATE
ARB(val) CFW <> Idle | XMIT_2_IDLES = 0 ARB_PEND = 0 val <>AL_PA val = AL_PA ARB_PEND = 1 val >=AL_PA val < AL_PA CFW = Idle & XMIT_2_IDLES = 1	 CFW := ARB(val) FC-2 FP/PSig/PSeq CFW := Idle FC-2 FP/PSig/PSeq CFW := ARB(AL_PA) FC-2 FP/PSig/PSeq CFW := ARB(val) FC-2 FP/PSig/PSeq FC-2 FP/PSig/PSeq	 OPENED OPENED OPENED OPENED OPENED
OPNr	FC-2 FP/PSig/PSeq	OPENED
OPNy	FC-2 FP/PSig/PSeq	OPENED
CLS	FC-2 FP/PSig/PSeq	RECEIVED CLOSE
DHD	DHD_RCV := 1 FC-2 FP/PSig/PSeq	 OPENED
MRKtx	FC-2 FP/PSig/PSeq	OPENED
PRIMITIVE SEQUENCES		
LIP	None/Inst.	OPEN-INIT-START
LPB (LPByx | LPBfx) y <>AL_PA y = AL_PA f = hex 'FF'	 FC-2 FP/PSig/PSeq BYPASS := 1 None/Inst. BYPASS := 1 None/Inst.	 OPENED MONITORING MONITORING
LPE (LPEyx|LPEfx)	FC-2 FP/PSig/PSeq	OPENED
ANY OTHER O.S.	FC-2 FP/PSig/PSeq	OPENED

INPUT	ACTION / OUTPUT	NEXT STATE
L_PORT CONTROLS		
REQ(monitor)	None/Inst.	OPENED
REQ(arb own AL_PA)	None/Inst.	OPENED
REQ(open yx) f-d	None/Inst.	OPENED
REQ(open yy) h-d	None/Inst.	OPENED
REQ(open fr)	None/Inst.	OPENED
REQ(open yr)	None/Inst.	OPENED
REQ(close)	CLS at the next app. Fill Word	XMITTED CLOSE (when CLS sent)
REQ(send DHD)	None/Inst.	OPENED
REQ(transfer)	None/Inst.	OPENED
REQ(old-port)	None/Inst.	OPENED
REQ(participating)	None/Inst.	OPENED
REQ(nonparticipat.)	None/Inst.	OPENED
REQ(mark as tx)	MRKtx at the next Fill Word	OPENED
REQ(bypass L_Port)	None/Inst.	OPENED
REQ(bypass L_Port y)	None/Inst.	OPENED
REQ(bypass all)	None/Inst.	OPENED
REQ(enable L_Port)	None/Inst.	OPENED
REQ(enable L_Port y)	None/Inst.	OPENED
REQ(enable all)	None/Inst.	OPENED
REQ(initialize)	None/Inst.	NORMAL-INITIALIZE

Table 9 — XMITTED CLOSE (State 5) transitions

ENTRY ACTIONS		
ACCESS := N/C	DUPLEX := 0	DHD_RCV := N/C
ARB_PEND := N/C	REPLICATE := N/C	BYPASS := N/C
ARB_WON := N/C	CFW := N/C	ERR_INIT := N/C
ARBf_SENT := N/C		XMIT_2_IDLES := N/C

INPUT	ACTION / OUTPUT	NEXT STATE
LOSS of SYNC. < R_T_TOV	Idle or CFW	XMITTED CLOSE
Loop Failure	None/Inst.	LOOP-FAIL-INITIALIZE
INVALID TRANS. WORD	CFW	XMITTED CLOSE
RUNNING DISP at O.S.	CFW	XMITTED CLOSE
ELASTICITY WORD REQd	N/A	XMITTED CLOSE
VALID DATA WORD	CFW	XMITTED CLOSE
VALID TRANS. WORD =O.S.		
FRAME DELIMITERS		
SOFxx	CFW	XMITTED CLOSE
EOFxx	CFW	XMITTED CLOSE
PRIMITIVE SIGNALS		
Idle ARB_PEND = 0 ARB_PEND = 1 XMIT_2_IDLES = 0 XMIT_2_IDLES = 1	 CFW := Idle ACCESS := 1 CFW CFW := ARB(AL_PA) CFW CFW := Idle CFW	 XMITTED CLOSE XMITTED CLOSE XMITTED CLOSE
R_RDY	CFW	XMITTED CLOSE
ARB(F0) CFW <> Idle | XMIT_2_IDLES = 0 ARB_WON = 1 ARB_WON = 0 ARB_PEND = 0 ARB_PEND = 1 CFW = Idle & XMIT_2_IDLES = 1	 CFW := Idle XMIT_2_IDLES := 1 CFW CFW := ARB(F0) XMIT_2_IDLES := 1 CFW CFW := ARB(AL_PA) XMIT_2_IDLES := 1 CFW CFW	 XMITTED CLOSE XMITTED CLOSE XMITTED CLOSE XMITTED CLOSE
ARByx y <> x	CFW[16]	XMITTED CLOSE

[16] While some L_Ports may set the CFW to the received ARByx, it is recommended that the CFW is not changed.

INPUT	ACTION / OUTPUT	NEXT STATE
ARB(val)		
ARB_WON = 1	CFW	XMITTED CLOSE
ARB_WON = 0		
CFW <> Idle |		
XMIT_2_IDLES = 0		
ARB_PEND = 0		
val <>AL_PA	CFW := ARB(val)	
	CFW	XMITTED CLOSE
val = AL_PA	CFW := Idle	
	CFW	XMITTED CLOSE
ARB_PEND = 1		
val >=AL_PA	CFW := ARB(AL_PA)	
	CFW	XMITTED CLOSE
val < AL_PA	CFW := ARB(val)	
	CFW	XMITTED CLOSE
CFW = Idle &		
XMIT_2_IDLES = 1	CFW	XMITTED CLOSE
OPNr	CFW	XMITTED CLOSE
OPNy	CFW	XMITTED CLOSE
CLS		
ARB_WON = 0		
ARB_PEND = 0	CFW	MONITORING
ARB_PEND = 1	CFW	ARBITRATING
ARB_WON = 1		
ARB_PEND = 0	CFW	MONITORING (when BB_Credit)
ARB_PEND = 1	CFW	ARBITRATING (when BB_Credit)
DHD	CFW	XMITTED CLOSE
MRKtx		
x = AL_PA	CFW	XMITTED CLOSE
x <>AL_PA	Same Word	XMITTED CLOSE
PRIMITIVE SEQUENCES		
LIP	None/Inst.	OPEN-INIT-START
LPB (LPByx | LPBfx)		
x = AL_PA	CFW	XMITTED CLOSE
y <>AL_PA	CFW	XMITTED CLOSE
y = AL_PA	BYPASS := 1	
	None/Inst.	MONITORING
f = hex 'FF'	BYPASS := 1	
	None/Inst.	MONITORING
LPE (LPEyx | LPEfx)	CFW	XMITTED CLOSE
ANY OTHER O.S.	CFW	XMITTED CLOSE

INPUT	ACTION / OUTPUT	NEXT STATE
L_Port CONTROLS		
REQ(monitor)	None/Inst.	XMITTED CLOSE
REQ(arb own AL_PA)	None/Inst.	XMITTED CLOSE
REQ(open yx) f-d	None/Inst.	XMITTED CLOSE
REQ(open yy) h-d	None/Inst.	XMITTED CLOSE
REQ(open fr)	None/Inst.	XMITTED CLOSE
REQ(open yr)	None/Inst.	XMITTED CLOSE
REQ(close)	None/Inst.	XMITTED CLOSE
REQ(send DHD)	None/Inst.	XMITTED CLOSE
REQ(transfer)	None/Inst.	XMITTED CLOSE
REQ(old-port)	None/Inst.	XMITTED CLOSE
REQ(participating)	None/Inst.	XMITTED CLOSE
REQ(nonparticipat.)	None/Inst.	XMITTED CLOSE
REQ(mark as tx)	MRKtx at the next Fill Word	XMITTED CLOSE
REQ(bypass L_Port)	None/Inst.	XMITTED CLOSE
REQ(bypass L_Port y)	None/Inst.	XMITTED CLOSE
REQ(bypass all)	None/Inst.	XMITTED CLOSE
REQ(enable L_Port)	None/Inst.	XMITTED CLOSE
REQ(enable L_Port y)	None/Inst.	XMITTED CLOSE
REQ(enable all)	None/Inst.	XMITTED CLOSE
REQ(initialize)	None/Inst.	NORMAL-INITIALIZE

Table 10 — RECEIVED CLOSE (State 6) transitions

ENTRY ACTIONS		
ACCESS := N/C	DUPLEX :=N/C	DHD_RCV := N/C
ARB_PEND := N/C	REPLICATE := N/C	BYPASS := N/C
ARB_WON := N/C	CFW := N/C	ERR_INIT := N/C
ARBf_SENT := N/C		XMIT_2_IDLES := N/C

INPUT	ACTION / OUTPUT	NEXT STATE
LOSS of SYNC. < R_T_TOV	FC-2 FP/PSig/PSeq	RECEIVED CLOSE
Loop Failure	None/Inst.	LOOP-FAIL-INITIALIZE
INVALID TRANS. WORD	FC-2 FP/PSig/PSeq	RECEIVED CLOSE
RUNNING DISP at O.S.	FC-2 FP/PSig/PSeq	RECEIVED CLOSE
ELASTICITY WORD REQd	N/A	RECEIVED CLOSE
VALID DATA WORD	FC-2 FP/PSig/PSeq	RECEIVED CLOSE
VALID TRANS. WORD =O.S.		
FRAME DELIMITERS		
SOFxx	FC-2 FP/PSig/PSeq	RECEIVED CLOSE
EOFxx	FC-2 FP/PSig/PSeq	RECEIVED CLOSE
PRIMITIVE SIGNALS		
Idle ARB_PEND = 0 ARB_PEND = 1 XMIT_2_IDLES = 0 XMIT_2_IDLES = 1	 CFW := Idle ACCESS := 1 FC-2 FP/PSig/PSeq CFW := ARB(AL_PA) FC-2 FP/PSig/PSeq CFW := Idle FC-2/FP/PSig/PSeq	 RECEIVED CLOSE RECEIVED CLOSE RECEIVED CLOSE
R_RDY	FC-2 FP/PSig/PSeq	RECEIVED CLOSE
ARB(F0 CFW <> Idle | XMIT_2_IDLES = 0 ARB_WON = 1 ARB_WON = 0 ARB_PEND = 0 ARB_PEND = 1 CFW = Idle & XMIT_2_IDLES = 1	 CFW := Idle XMIT_2_IDLES := 1 FC-2 FP/PSig/PSeq CFW := ARB(F0) XMIT_2_IDLES := 1 FC-2 FP/PSig/PSeq CFW := ARB(AL_PA) XMIT_2_IDLES := 1 FC-2 FP/PSig/PSeq FC-2/FP/PSig/PSeq	 RECEIVED CLOSE RECEIVED CLOSE RECEIVED CLOSE RECEIVED CLOSE
ARByx y <> x	 FC-2 FP/PSig/PSeq[17]	 RECEIVED CLOSE

[17] While some L_Ports may set the CFW to the received ARByx, it is recommended that the CFW is not changed.

INPUT	ACTION / OUTPUT	NEXT STATE
ARB(val) ARB_WON = 1 ARB_WON = 0 CFW <> Idle | XMIT_2_IDLES = 0 ARB_PEND = 0 val <>AL_PA val = AL_PA ARB_PEND = 1 val >=AL_PA val < AL_PA CFW = Idle & XMIT_2_IDLES = 1	 FC-2 FP/PSig/PSeq CFW := ARB(val) FC-2 FP/PSig/PSeq CFW := Idle FC-2 FP/PSig/PSeq CFW := ARB(AL_PA) FC-2 FP/PSig/PSeq CFW := ARB(val) FC-2 FP/PSig/PSeq FC-2 FP/PSig/PSeq	 RECEIVED CLOSE RECEIVED CLOSE RECEIVED CLOSE RECEIVED CLOSE RECEIVED CLOSE RECEIVED CLOSE
OPNr	FC-2 FP/PSig/PSeq	RECEIVED CLOSE
OPNy	FC-2 FP/PSig/PSeq	RECEIVED CLOSE
CLS	FC-2 FP/PSig/PSeq	RECEIVED CLOSE
DHD	FC-2 FP/PSig/PSeq	RECEIVED CLOSE
MRKtx	FC-2 FP/PSig/PSeq	RECEIVED CLOSE
PRIMITIVE SEQUENCES		
LIP	None/Inst.	OPEN-INIT-START
LPB (LPByx | LPBfx) x = AL_PA y <>AL_PA y = AL_PA f = hex 'FF'	 FC-2 FP/PSig/PSeq FC-2 FP/PSig/PSeq BYPASS := 1 None/Inst. BYPASS := 1 None/Inst.	 RECEIVED CLOSE RECEIVED CLOSE MONITORING MONITORING
LPE (LPEyx | LPEfx)	FC-2 FP/PSig/PSeq	RECEIVED CLOSE
ANY OTHER O.S.	FC-2 FP/PSig/PSeq	RECEIVED CLOSE

INPUT	ACTION / OUTPUT	NEXT STATE
L_Port CONTROLS		
REQ(monitor)	None/Inst.	RECEIVED CLOSE
REQ(arb own AL_PA)	None/Inst.	RECEIVED CLOSE
REQ(open yx) f-d	None/Inst.	RECEIVED CLOSE
REQ(open yy) h-d	None/Inst.	RECEIVED CLOSE
REQ(open fr)	None/Inst.	RECEIVED CLOSE
REQ(open yr)	None/Inst.	RECEIVED CLOSE
REQ(close) ARB_PEND = 0 ARB_PEND = 1	 When BB_Credit, CLS at the next app. Fill Word When BB_Credit, CLS at the next app. Fill Word	 MONITORING (when CLS sent) ARBITRATING (when CLS sent)
REQ(send DHD)	None/Inst.	RECEIVED CLOSE
REQ(transfer)	None/Inst.	RECEIVED CLOSE
REQ(old-port)	None/Inst.	RECEIVED CLOSE
REQ(participating)	None/Inst.	RECEIVED CLOSE
REQ(nonparticipat.)	None/Inst.	RECEIVED CLOSE
REQ(mark as tx)	MRKtx at the next Fill Word	RECEIVED CLOSE
REQ(bypass L_Port)	None/Inst.	RECEIVED CLOSE
REQ(bypass L_Port y)	None/Inst.	RECEIVED CLOSE
REQ(bypass all)	None/Inst.	RECEIVED CLOSE
REQ(enable L_Port)	None/Inst.	RECEIVED CLOSE
REQ(enable L_Port y)	None/Inst.	RECEIVED CLOSE
REQ(enable all)	None/Inst.	RECEIVED CLOSE
REQ(initialize)	None/Inst.	NORMAL-INITIALIZE

Table 11 — TRANSFER (State 7) transitions

ENTRY ACTIONS		
ACCESS := N/C	DUPLEX := 0	DHD_RCV := N/C
ARB_PEND := N/C	REPLICATE := N/C	BYPASS := N/C
ARB_WON := N/C	CFW := N/C	ERR_INIT := N/C
ARBf_SENT := N/C		XMIT_2_IDLES := N/C

INPUT	ACTION / OUTPUT	NEXT STATE
LOSS of SYNC. < R_T_TOV	Idle or CFW	TRANSFER
Loop Failure	None/Inst.	LOOP-FAIL-INITIALIZE
INVALID TRANS. WORD	CFW	TRANSFER
RUNNING DISP at O.S.	CFW	TRANSFER
ELASTICITY WORD REQd	N/A	TRANSFER
VALID DATA WORD	CFW	TRANSFER
VALID TRANS. WORD =O.S.		
FRAME DELIMITERS		
SOFxx	CFW	TRANSFER
EOFxx	CFW	TRANSFER
PRIMITIVE SIGNALS		
Idle	CFW := Idle ACCESS := 1 CFW	 TRANSFER
R_RDY	CFW	TRANSFER
ARB(F0)	CFW := Idle XMIT_2_IDLES := 1 CFW	 TRANSFER
ARByx	CFW	TRANSFER
OPNr	CFW	TRANSFER
OPNy	CFW	TRANSFER
CLS	Per applicable request under L_Port CONTROLS in this table	OPEN / MONITORING
DHD	CFW	TRANSFER
MRKtx x = AL_PA x <>AL_PA	 CFW Same Word	 TRANSFER TRANSFER
PRIMITIVE SEQUENCES		
LIP	None/Inst.	OPEN-INIT-START
LPB (LPByx | LPBfx) x = AL_PA y <>AL_PA y = AL_PA f = hex 'FF'	 CFW CFW BYPASS := 1 None/Inst. BYPASS := 1 None/Inst.	 TRANSFER TRANSFER MONITORING MONITORING
LPE (LPEyx | LPEfx)	CFW	TRANSFER

INPUT	ACTION / OUTPUT	NEXT STATE
ANY OTHER O.S.	CFW	TRANSFER
L_Port CONTROLS		
REQ(monitor)	CFW when CLS received	MONITORING (when BB_Credit)
REQ(arb own AL_PA)	None/Inst.	TRANSFER
REQ(open yx) f-d	When CLS received & BB_Credit REPLICATE := 0 OPNyx	OPEN
REQ(open yy) h-d	When CLS received & BB_Credit REPLICATE := 0 OPNyy	OPEN
REQ(open fr)	When CLS received & BB_Credit REPLICATE := 1 OPNfr	OPEN
REQ(open yr)	When CLS received & BB_Credit REPLICATE := 1 OPNyr	OPEN
REQ(close)	None/Inst.	TRANSFER
REQ(send DHD)	None/Inst.	TRANSFER
REQ(transfer)	None/Inst.	TRANSFER
REQ(old-port)	None/Inst.	TRANSFER
REQ(participating)	None/Inst.	TRANSFER
REQ(nonparticipat.)	None/Inst.	TRANSFER
REQ(mark as tx)	MRKtx at the next Fill Word	TRANSFER
REQ(bypass L_Port)	None/Inst.	TRANSFER
REQ(bypass L_Port y)	None/Inst.	TRANSFER
REQ(bypass all)	None/Inst.	TRANSFER
REQ(enable L_Port)	None/Inst.	TRANSFER
REQ(enable L_Port y)	None/Inst.	TRANSFER
REQ(enable all)	None/Inst.	TRANSFER
REQ(initialize)	None/Inst.	NORMAL-INITIALIZE

Table 12 — INITIALIZATION process (State 8) transitions

ENTRY ACTIONS		
ACCESS := 1 ARB_PEND := 0 ARB_WON := 0 ARBf_SENT := 0	DUPLEX := 0 REPLICATE := 0 CFW := Idle BB_Credit := 0	DHD_RCV := 0 BYPASS := 0 ERR_INIT := (see 10.5.4) XMIT_2_IDLES := 0 Alternate BB_Credit := 1
INPUT	**ACTION / OUTPUT**	**NEXT STATE**
Described in 10.5.4		

Table 13 — Reserved

NOTE — This table was removed and is now incorporated into the Initialization state diagrams of clause 10.

Table 14 — OLD-PORT (State A) transitions (optional)

ENTRY ACTIONS		
ACCESS := N/C ARB_PEND := N/C ARB_WON := N/C ARBf_SENT := N/C	DUPLEX := N/C REPLICATE := N/C CFW := Idle BB_Credit := 1	DHD_RCV := N/C BYPASS := (see 10.5.4) ERR_INIT := (see 10.5.4) XMIT_2_IDLES := 0 Alternate BB_Credit := 0
INPUT	**ACTION / OUTPUT**	**NEXT STATE**
Described in 10.5.4		

10 Loop Initialization procedure

Loop Initialization is a logical procedure used by an L_Port to determine its environment and to validate an AL_PA. During the procedure, the L_Port uses the LPSM and FC-2 protocol to discover its environment and react appropriately. At least a 132 byte receive buffer shall be available to receive each of the following Loop Initialization frames (see 10.5): LIFA, LIPA, LIHA, LISA, LIRP, and LILP; all other frames (i.e., LISM) may be discarded if the L_Port cannot accept the frame (e.g., buffers are full).

10.1 Loop Initialization summary

A general summary of Loop Initialization follows (see 8.4.3, item 21, table 12, and 10.5.4):

— During Loop Initialization, one L_Port shall win as Loop Initialization Master (LIM) to manage the initialization procedure. All FL_Ports shall be capable of performing this function; NL_Ports may perform this function. However, if no L_Port is selected as the LIM during the LISM sequence, then the Loop is inoperative.

— During Loop Initialization, only SOFiL shall be used to precede the Loop Initialization Sequences. R_RDYs shall not be used for flow-control and shall not be transmitted. If an R_RDY is received, it shall be discarded.

— PARTICIPATE is set FALSE(0) when ARB(F0) is received and TRUE(1) when an AL_PA is taken in the Loop Initialization Sequences LIFA, LIPA, LIHA or LISA. In FC-AL-1, PARTICIPATE was set TRUE(1) after CLOSE was received and not well defined between ARB(F0) and CLOSE being received.

— If a non-L_Port is attached point-to-point to the L_Port, the L_Port may complete the initialization procedure described in ANSI X3, FC-PH-x, the OLD-PORT state. While in the OLD-PORT state, only FC-2 specified communication shall be used between the L_Port and the non-L_Port without further use of the Loop protocol.

— If two or more L_Ports are connected in a Loop without any non-L_Ports present, one FL_Port and up to 126 NL_Ports may finish the initialization procedure in the MONITORING state and in the Participating mode. FC-2 specified communication is used as permitted by the Loop LPSM.

— If one or more non-L_Ports are connected in a Loop with one or more L_Ports, the Loop is not operational.

  Arbitrary positioning of non-L_Ports on a Loop may cause one or more L_Ports to discover that at least one upstream L_Port is an L_Port. However, the L_Ports are unable to successfully complete the remaining portions of the initialization procedure and remain in the initialization procedure.

— If more than one FL_Port or more than 126 NL_Ports are connected to a Loop, only one FL_Port and up to 126 NL_Ports may enter the Participating mode. The remaining L_Ports operate in the MONITORING state and in the Non-Participating mode.

  The initialization procedure permits a Non-Participating L_Port to attempt Loop Initialization after waiting an implementation-selected time or when a Participating L_Port voluntarily yields its AL_PA. This allows the limited number of available AL_PAs to be shared.

  NOTE — If an L_Port in the Participating mode goes to the Non-Participating mode, it may invoke the Loop Initialization procedure to allow another L_Port to use its AL_PA (see 8.4.3, item 13).

— If an FL_Port exits the initialization procedure in the Participating mode, its AL_PA shall be hex '00' and it shall accept a D_ID of hex 'FFFFFE' as specified in ANSI X3, FC-PH-x. A Public NL_Port on the Loop may form a Loop circuit with AL_PA hex '00' and shall receive normal Fabric topology responses from the FL_Port as specified in ANSI X3, FC-PH-x.

— If a Public NL_Port exits the initialization procedure in the Participating mode, it attempts Login (if required in 10.5.3, step (6)) with the well-known address hex 'FFFFFE' through AL_PA hex '00' to obtain its native address identifier (see ANSI X3, FC-PH-x, 21.4.7 and 23.3.1). The S_ID = hex '0000'||AL_PA or 'xxxx'||AL_PA where 'xxxx' is the previous Login value.

— If a Public NL_Port exits the initialization procedure in the Participating mode and detects that an FL_Port is not in the Participating mode on the Loop, it may accept the responsibility of providing Fibre Channel services (e.g., accept well-known addresses hex 'FFFFF0' to hex 'FFFFFE'). An NL_Port in this mode (known as an F/NL_Port) shall accept an alias AL_PA of hex '00' (in addition to its normal AL_PA) and detect OPN(00,x), but shall not transmit ARB(00,00).

If an FL_Port initializes later than this F/NL_Port, the NL_Port shall no longer respond to alias AL_PA hex '00'.

10.2 Loop Initialization introduction

An L_Port starts the Loop Initialization procedure by making the transition to the NORMAL-INITIALIZE state.

NOTE — Loop Initialization may be disruptive (i.e., frames may be lost if frames are being transmitted). To minimize this disruption, the Loop may be quiesced by transmitting ARB(val) (where val is a trusted AL_PA of the L_Port) to win access to the Loop prior to issuing the first LIP. If the L_Port wins arbitration, the L_Port may assume that the Loop is not being used by another L_Port and the L_Port may begin transmitting LIPs.

Reasons for entering Loop Initialization include:

— to acquire an AL_PA so that the L_Port may participate on the Loop. The AL_PA of a Participating L_Port, and the corresponding priority of an NL_Port, may change each time the initialization procedure is invoked. The priority of the FL_Port is always the same;

— provide notification of a possible configuration change; and,

— error recovery.

When BYPASS is FALSE(0), an L_Port shall enter the OPEN-INIT-START state whenever any LIP is detected. This may interrupt two communicating L_Ports, but normal FC-PH error recovery (after returning to the MONITORING state) may be used to restore any exchanges in progress.

Figure P.1 (see annex P) provides a flowchart-like view of the Loop Initialization procedure; 10.5 provides a detailed set of state diagrams for the INITIALIZATION process.

10.3 Loop Initialization timers

The INITIALIZATION process times the completion of many events using multiples of AL_TIME (see 8.2.2). The notation *n*xAL_TIME denotes a timer whose nominal expiration occurs at *n* times the value of AL_TIME.

In some cases a timer expiration triggers a specific event. In others, the timer represents the minimum or maximum time to wait for an event. These are denoted as follows:

— **Start(*n*xAL_TIME)** - the starting of the timer;

— **expire(*n*xAL_TIME)** - an event that shall occur no earlier than *n* times the value of AL_TIME and no later than *n* times the value of AL_TIME plus 20% (e.g., expire(2xAL_TIME) denotes an event that shall occur between 30ms and 36ms from Start(2xAL_TIME));

— **minimum(*n*xAL_TIME)** - an event that shall occur no earlier than *n* times the value of AL_TIME (e.g., minimum(1xAL_TIME) denotes an event that shall occur no earlier than 15ms from Start(1xAL_TIME)); and,

— **maximum(*n*xAL_TIME)** - an event that shall occur no later than *n* times the value of AL_TIME plus 20% (e.g., maximum(4xAL_TIME) denotes an event that shall occur no later than 72ms from Start(4xAL_TIME)).

Timer values persist across state transitions. A timer started in one state may reach a value that causes a condition, such as expire(timer), to be true in another state. Once a timer condition is true, that condition remains true until the timer is explicitly started again with start(timer).

10.4 Node-initiated L_Port initialization

This procedure is entered for one of the following reasons:

— to decide if a Loop is present and to acquire an AL_PA at power-on;

— at the discretion of the Node (e.g., for Loop Failures);

— whenever the AL_PA is modified during Fabric Login to the well-known address hex 'FFFFFE'; or,

— to acquire an AL_PA after a previous attempt was unsuccessful. (This would be the case if more than 126 NL_Ports or more than one FL_Port existed on the Loop.)

NOTE — Loop Initialization may be disruptive (unless the Loop is quiesced before Loop Initialization begins). For this reason initializing should be used infrequently and the time delay between retrying is recommended to be in minutes.

— to relinquish an AL_PA when going to the Non-Participating mode.

The L_Port that is attempting to initialize shall make the transition to the NORMAL-INITIALIZE state (REQ(initialize)) (see 10.5). If the L_Port recognizes LIP, it shall transfer to the OPEN-INIT-START state (see 10.5.4.6). If the LIP is not recognized within minimum(3xAL_TIME) while the received signal is valid, the LPSM may remain in the NORMAL-INITIALIZE state or if supported, shall make the transition to the OLD-PORT state.

10.5 L_Port initialization

After the L_Port has performed the entry actions for the OPEN-INIT-START state (see 10.5.4.6), the L_Port shall continue the initialization procedure as defined in 10.5.3, steps (1) to (6). This initialization procedure shall use the Loop Initialization Sequences as defined in figure 4.

10.5.1 Loop Initialization Sequences

Start_of_Frame delimiter - 4 bytes

SOFiL

Frame_Header - 24 bytes

22XXXXXX	00XXXXXX	01380000	00000000	FFFFFFFF	00000000

where 'XXXXXX' is hex '000000' for an FL_Port and hex '0000EF' for an NL_Port or F/NL_Port, or some other value specified by a future standard.[18]

Payload - 12, 20, or 132 bytes

LI_ID and LI_FL	8-byte Port_Name
	16-byte AL_PA bit map
	128-byte AL_PA position map (1-byte offset followed by up to 127 AL_PAs)

where LI_ID and LI_FL contain the following:

LI_ID (Identifiers) (16 bits)

Value (hex)	Name	Description	Payload size)
'1101'	LISM	Select Master based on 8-byte Port_Name	(12-byte)
'1102'	LIFA	Fabric Assign AL_PA bit map	(20-byte)
'1103'	LIPA	Previously Acquired AL_PA bit map	(20-byte)
'1104'	LIHA	Hard Assigned AL_PA bit map	(20-byte)
'1105'	LISA	Soft Assigned AL_PA bit map	(20-byte)
'1106'	LIRP	Report AL_PA position map	(132-byte)
'1107'	LILP	Loop AL_PA position map	(132-byte)

LI_FL (Flag) (16 bits; all 'r's are reserved—not checked, but originated as zero)

LI_ID	Flag	Mask (binary)	Meaning
LISM	-	rrrr rrrr rrrr rrrr	reserved
LIFA	-	rrrr rrrr rrrr rrrr	reserved
LIPA	-	rrrr rrrr rrrr rrrr	reserved
LIHA	-	rrrr rrrr rrrr rrrr	reserved
LISA	8	rrrr rrr1 rrrr rrrr	LIRP and LILP supported
LIRP	-	rrrr rrrr rrrr rrrr	reserved
LILP	-	rrrr rrrr rrrr rrrr	reserved

Cyclic Redundancy Check - 4 bytes

CRC

End_of_Frame delimiter - 4 bytes

EOFt

Figure 4 — Loop Initialization Sequences

[18]To allow a future NL_Port to win as LIM, the NL_Port may use a LISM frame with a D_ID of hex '000000' and S_ID of hex 'XXXXXX' (where 'XXXXXX' is to be defined, however, the right-most bit is reserved). These NL_Ports will yield to FL_Ports with an S_ID of hex '000000'; existing NL_Ports will yield to D_ID of hex '000000'.

The ANSI X3, FC-PH-x rules for valid frames apply to transmitting the Loop Initialization Sequences which are shown in figure 4. When an L_Port receives these Loop Initialization Sequences, the L_Port shall discard or not process all frames which contain the following errors (the L_Port is not required to verify the frame header):

— code violations;
— CRC errors;
— a frame that does not end in EOFt or EOFn;
— payload violations (i.e., a payload which does adhere to the payloads described in figure 4).

When forwarding Loop Initialization Sequences, the Non-Loop Initialization Master L_Ports may use the frame header defined in figure 4, or the received frame header. However, the D_ID and S_ID of the received LISM frame shall be used in either case.

The one Loop Initialization Sequence that carries an 8-byte Port_Name is:

LISM) **Select Master**: used to select a LIM.

The four Loop Initialization Sequences that carry a 16-byte AL_PA bit map are:

LIFA) **Fabric Assigned**: used to gather all Fabric Assigned AL_PAs.

LIPA) **Previously Acquired**: used to gather all Previously Acquired AL_PAs.

LIHA) **Hard Assigned**: used to gather all Hard Assigned AL_PAs (e.g., configuration switches (see annex K)).

LISA) **Soft Assigned**: used to assign any remaining bits as a Soft Assigned AL_PA.

The two Loop Initialization Sequences that carry a 128-byte AL_PA position map are:

LIRP) **Report Position:** used to collect the relative positions of all Participating L_Ports on the Loop.

LILP) **Loop Position:** used to inform all L_Ports of the relative positions of all Participating L_Ports on the Loop from the perspective of the LIM.

10.5.2 Assigned AL_PA values

All AL_PAs that are used in the Loop protocol are specified in table 1. The AL_PAs are assigned to the 16-byte AL_PA bit maps of figure 5 as shown in table 15.

Table 15 — AL_PA mapped to bit maps

AL_PA (hex)	Bit Word	Map Bit	AL_PA (hex)	Bit Word	Map Bit	AL_PA (hex)	Bit Word	Map Bit	AL_PA (hex)	Bit Word	Map Bit
--	0	31	3C	1	31	73	2	31	B3	3	31
00	0	30	43	1	30	74	2	30	B4	3	30
01	0	29	45	1	29	75	2	29	B5	3	29
02	0	28	46	1	28	76	2	28	B6	3	28
04	0	27	47	1	27	79	2	27	B9	3	27
08	0	26	49	1	26	7A	2	26	BA	3	26
0F	0	25	4A	1	25	7C	2	25	BC	3	25
10	0	24	4B	1	24	80	2	24	C3	3	24
17	0	23	4C	1	23	81	2	23	C5	3	23
18	0	22	4D	1	22	82	2	22	C6	3	22
1B	0	21	4E	1	21	84	2	21	C7	3	21
1D	0	20	51	1	20	88	2	20	C9	3	20
1E	0	19	52	1	19	8F	2	19	CA	3	19
1F	0	18	53	1	18	90	2	18	CB	3	18
23	0	17	54	1	17	97	2	17	CC	3	17
25	0	16	55	1	16	98	2	16	CD	3	16
26	0	15	56	1	15	9B	2	15	CE	3	15
27	0	14	59	1	14	9D	2	14	D1	3	14
29	0	13	5A	1	13	9E	2	13	D2	3	13
2A	0	12	5C	1	12	9F	2	12	D3	3	12
2B	0	11	63	1	11	A3	2	11	D4	3	11
2C	0	10	65	1	10	A5	2	10	D5	3	10
2D	0	9	66	1	9	A6	2	9	D6	3	9
2E	0	8	67	1	8	A7	2	8	D9	3	8
31	0	7	69	1	7	A9	2	7	DA	3	7
32	0	6	6A	1	6	AA	2	6	DC	3	6
33	0	5	6B	1	5	AB	2	5	E0	3	5
34	0	4	6C	1	4	AC	2	4	E1	3	4
35	0	3	6D	1	3	AD	2	3	E2	3	3
36	0	2	6E	1	2	AE	2	2	E4	3	2
39	0	1	71	1	1	B1	2	1	E8	3	1
3A	0	0	72	1	0	B2	2	0	EF	3	0

NOTE — '--' is reserved for the L_bit (Fabric Login required); AL_PA = '00' is reserved for the FL_Port

10.5.3 Loop Initialization steps

The following initialization steps shall be performed. From the time the LIM is selected and sends the first ARB(F0) until the CLS at the end of the INITIALIZATION process has gone around the Loop, AL_PAs are unstable and any addressed Primitive Sequence (e.g., LPByx) may not be acted upon by the desired L_Port.

1) Select initial Native Address Identifiers (D_ID and S_ID)

Each FL_Port shall choose an initial value for its Native Address Identifier of hex '000000' to be used in its LISM frame (see 10.5.1).

Each NL_Port shall choose an initial value for its Native Address Identifier of hex '0000EF' to be used in its LISM frame (see 10.5.1).

Until an L_Port has acquired an AL_PA by completing Loop Initialization (or through some other means), it cannot be uniquely distinguished with an LPByx. If an L_Port has a trusted AL_PA, it may respond to this AL_PA (e.g., as in LPB or LPE) until the AL_PA is determined not to be usable by this L_Port (see 10.5.4.1).

If an NL_Port implements the LIM function, the NL_Port shall continue at step (2); otherwise, the NL_Port shall continue at step (3).

2) Select a Loop Initialization Master (LIM)

The L_Port shall continuously transmit Loop Initialization Sequences (LI_ID='LISM') formatted as shown in figure 4. Successive Loop Initialization Sequences shall be separated by six or more Idles.

NOTE — Frames are sent continuously because they may be discarded by any L_Port that does not have a receive buffer available (flow control is not used during Loop Initialization).

When a valid Loop Initialization Sequence (LI_ID='LISM') is received, the D_ID, S_ID, and Port_Name shall be compared to the transmitted frames as follows:

a) if the received D_ID, S_ID, and Port_Name are equal to the transmitted D_ID, S_ID, and Port_Name, respectively, then the L_Port shall become the LIM. The LIM shall continue at step (4).

b) if the received D_ID is lower than the transmitted D_ID, then the received Loop Initialization Sequence is algebraically lower. The L_Port shall continue at step (3).

c) if the received D_ID is equal to the transmitted D_ID and the received S_ID is lower than the transmitted S_ID, then the received Loop Initialization Sequence is algebraically lower. The L_Port shall continue at step (3).

d) if the received D_ID is equal to the transmitted D_ID, the received S_ID is equal to the transmitted S_ID, and the received Port_Name is algebraically lower than the transmitted Port_Name, then the received Loop Initialization Sequence is algebraically lower. The L_Port shall continue at step (3).

If a LIM is not selected before LP_TOV expires, the L_Port shall make the transition to the NORMAL-INITIALIZE state to transmit LIP(F7).

3) Wait for a Loop Initialization Master

The L_Port shall repeat all frames that it receives until the L_Port receives ARB(F0). When ARB(F0) is received, PARTICIPATE shall be set to FALSE(0) and the L_Port shall continue at step (5). An L_Port shall wait a minimum of LP_TOV for ARB(F0). If LP_TOV expires before ARB(F0) is received (no LIM was selected), the L_Port shall make the transition to the NORMAL-INITIALIZE state to transmit LIP(F7).

NOTE — Frames may be originated or repeated at a faster or slower rate than they are received. Frames may be forwarded without any qualification or error checking.

4) LIM — transmit remaining Loop Initialization Sequences

a) The LIM shall transmit ARB(F0) a minimum of LP_TOV or until ARB(F0) is received. When ARB(F0) is received, PARTICIPATE shall be set to FALSE(0). If LP_TOV expires before ARB(F0) is received, the LIM shall make the transition to the NORMAL-INITIALIZE state to transmit LIP(F7).

b) The LIM shall transmit the Loop Initialization Sequences (LI_ID='LIFA', 'LIPA', 'LIHA', and 'LISA'). These Loop Initialization Sequences contain a 16-byte AL_PA bit map in the payload. Each bit represents one AL_PA (see figure 4, figure 5, and table 15).

	Bits
Word	3322 2222 2222 1111 1111 11 1098 7654 3210 9876 5432 1098 7654 3210
0	**L**000 0000 0000 0000 0000 0000 0000 0000
1	0000 0000 0000 0000 0000 0000 0000 0000
2	0000 0000 0000 0000 0000 0000 0000 0000
3	0000 0000 0000 0000 0000 0000 0000 0000
	where '**L**' is the Fabric Login Required bit (L_bit)

Figure 5 — Loop Initialization Sequence AL_PA bit map

Except for the L_bit, each bit in figure 5 represents a valid AL_PA (according to tables 1 and 15). The L_bit shall only be set by the FL_Port or F/NL_Port to indicate that a new Fabric Login is required.

The LIM shall transmit the four Loop Initialization Sequences that contain the 16-byte AL_PA bit maps as follows:

LIFA The LIM shall prime the AL_PA bit map with binary zero (*0*) and shall set to one (*1*) the bit that corresponds to its Fabric Assigned AL_PA and shall set PARTICIPATE to TRUE(1). If the LIM is an FL_Port, it shall set the bit associated with AL_PA hex '00'. The L_bit may be set if the FL_Port requires a Fabric Login. The L_bit shall be set if this is the first initialization attempt by an NL_Port that has assumed the role of an F/NL_Port.

LIPA The LIM shall prime the AL_PA bit map with the AL_PA bit map of the previously received Loop Initialization Sequence (LI_ID='LIFA'). The LIM shall check if the bit that corresponds to its Previously Acquired AL_PA is set. If it is not set to 1, the LIM shall set the bit to 1 and shall set PARTICIPATE to TRUE(1) (unless a bit was set in LIFA); if the bit is already set to 1, the LIM may attempt a Hard Assigned AL_PA.

LIHA The LIM shall prime the AL_PA bit map with the AL_PA bit map of the previously received Loop Initialization Sequence (LI_ID='LIPA'). The LIM shall check if the bit that corresponds to its Hard Assigned AL_PA is set. If it is not set to 1, the LIM shall set the bit to 1 and shall set PARTICIPATE to TRUE(1) (unless a bit was set in LIFA or LIPA); if the bit is already set to 1, the LIM may attempt a Soft Assigned AL_PA.

LISA The LIM shall prime the AL_PA bit map with the AL_PA bit map of the previously received Loop Initialization Sequence (LI_ID='LIHA'). The LIM shall set the AL_PA position map, Flag 8 in LI_FL, to one(1). The LIM may set any available bit to 1 (unless a bit was set in LIFA, LIPA, or LIHA) which corresponds to its Soft Assigned AL_PA. If a bit was available, the LIM shall adjust its AL_PA according to which bit it set, shall set PARTICIPATE to TRUE(1), and shall continue in step c. If no bits were available, the LIM shall continue in step c (the L_Port may attempt to re-initialize per 10.4 at the request of the Node).

c) When the LISA Sequence is received, the LIM shall check Flag 8 in LI_FL. If Flag 8 is set to one (1), the LIM shall transmit two additional Loop Initialization Sequences as follows:

LIRP The LIM shall set the AL_PA position map to all hex 'FF'. If the LIM has an AL_PA, the AL_PA position map shall be set to hex '01xxFFFFFF...FF' (where 'xx' is the AL_PA of the LIM). If the LIM does not have an AL_PA, the AL_PA position map that the LIM originates shall be set to hex '00FF...FF'. The left-most byte is the offset of the last AL_PA added to the map.

LILP The LIM shall transmit the AL_PA position map which was received in the previous Loop Initialization Sequence (LI_ID='LIRP'). The first byte indicates the number of participating L_Ports on the Loop. The second byte shows either the address of the LIM or the first participating L_Port after the LIM. The other bytes contain the AL_PAs of the remaining participating L_Ports on the Loop (in order and relative to each other) with the last AL_PA being adjacent to the first AL_PA in byte two. A hex 'FF' is not a valid AL_PA.

d) When the last Loop Initialization Sequence (LI_ID='LISA' or 'LILP') is returned, the LIM shall transmit CLS to place all L_Ports into the MONITORING state. When CLS is received by the LIM, the LIM shall make the transition to the MONITORING state (either in the Participating mode if it has a valid AL_PA or in the Non-Participating mode) and relinquish its LIM role. At this time, all possible AL_PA values have been assigned for the number of L_Ports and every L_Port that has a valid AL_PA shall be in Participating mode.

NOTE — If the LIM advertised BB_Credit > 0, it should assure that sufficient receive buffers are available for the next Loop circuit before transmitting CLS.

If the LIM detects an invalid or unexpected Loop Initialization Sequence, the L_Port shall make the transition to the NORMAL-INITIALIZE state to transmit LIP(F7).

The LIM shall use LP_TOV to wait for each of the above Loop Initialization Sequences and the CLS. If LP_TOV expires before each transmitted Loop Initialization Sequence or CLS is received, the LIM shall make the transition to the NORMAL-INITIALIZE state to transmit LIP(F7).

When CLS is received, the LIM shall transition to the MONITORING state and be prepared to receive an immediate OPN. The LIM shall continue at step (6).

5) Non-Loop Initialization Master L_Port — select unique AL_PA

A non-Loop Initialization Master L_Port shall retransmit any received ARB(F0) when it is prepared to receive (e.g., empty its receive buffers) and retransmit the following Loop Initialization Sequences (LI_ID='LIFA', 'LIPA', 'LIHA', 'LISA', 'LIRP', and 'LILP'), followed by CLS.

The Loop Initialization Sequences are updated as follows (see figure 4, figure 5, and table 15):

LIFA The L_Port shall check if the bit that corresponds to its Fabric Assigned AL_PA is set. If it is not set to 1, the L_Port shall set the bit to 1 and shall set PARTICIPATE to TRUE(1); if the bit is already set to 1, the L_Port may attempt setting a bit in LIPA. The L_Port shall retransmit the Loop Initialization Sequence.

LIPA The L_Port shall check if the bit that corresponds to its Previously Acquired AL_PA is set. If it is not set to 1, the L_Port shall set the bit to 1 and shall set PARTICIPATE to TRUE(1)(unless a bit was set in LIFA); if the bit is already set to 1, the L_Port may attempt setting a bit in LIHA. The L_Port shall retransmit the Loop Initialization Sequence.

LIHA The L_Port shall check if the bit that corresponds to its Hard Assigned AL_PA is set. If it is not set to 1, the L_Port shall set the bit to 1 and shall set PARTICIPATE to TRUE(1) (unless a bit was set in LIFA or LIPA); if the bit is already set to 1, the L_Port shall either attempt setting a bit in LISA or go to the Non-Participating mode. The L_Port shall retransmit the Loop Initialization Sequence.

LISA The L_Port shall set any available bit to 1 (unless a bit was set in LIFA, LIPA, or LIHA) that corresponds to its Soft Assigned AL_PA. The L_Port shall set any flags in LI_FL to zero(0) which it does not recognize (or support). Flag 8 in LI_FL (LIRP and LILP supported) shall be supported. If a bit was available, the L_Port shall adjust its AL_PA according to which bit was set and shall set PARTICIPATE to TRUE(1). The L_Port shall retransmit the Loop Initialization Sequence.

LIRP If LIRP is received, if the L_Port has an AL_PA, it shall read the left-most byte (offset), increment it by one, store the offset, and store its AL_PA into the offset position. The L_Port shall retransmit the Loop Initialization Sequence.

LILP If LILP is received, the L_Port may use the AL_PA position map to save the relative positions of all Participating L_Ports on the Loop. This information may be useful for error recovery. The first byte indicates the number of nodes participating on the Loop. The second byte shows either the address of the LIM or the node after the LIM. The other bytes contain the AL_PAs of the participating nodes on the Loop, in order relative to each other, with the last AL_PA being adjacent to the first AL_PA in the list. An AL_PA of hex 'FF' is invalid. The L_Port shall retransmit the Loop Initialization Sequence.

If the L_Port detects an invalid or unexpected Loop Initialization Sequences, the L_Port may make the transition to the NORMAL-INITIALIZE state to transmit LIP(F7).

The L_Port shall use LP_TOV to wait for each Loop Initialization Sequence and the CLS. If LP_TOV expires before each Loop Initialization Sequence or CLS is received, the L_Port shall make the transition to the NORMAL-INITIALIZE state to transmit LIP(F7).

When CLS is received, the L_Port shall retransmit CLS and make the transition to the MONITORING state (either in the Participating or the Non-Participating mode). If the L_Port is in the Participating mode, it shall continue at step (6); if the L_Port is in the Non-Participating mode, it has completed Loop Initialization (the L_Port may attempt to re-initialize at 10.4 at the request of the Node). When CLS is transmitted, the L_Port shall be prepared to receive an immediate OPN.

NOTE — If the L_Port advertised BB_Credit > 0, it should assure that sufficient receive buffers are available for the next Loop circuit before transmitting CLS.

6) Select final AL_PA and exit Loop Initialization

a) If an FL_Port is in Participating mode, it has completed the initialization procedure with an AL_PA of hex '00' and shall exit the Loop Initialization.

b) If a Private NL_Port is in Participating mode, the NL_Port has completed the initialization procedure with an AL_PA in the range of hex '01' through hex 'EF' and shall exit Loop Initialization.

c) If a Public NL_Port is in Participating mode, the NL_Port shall have acquired an AL_PA in the range of hex '01' through hex 'EF'. If one of the following occurred, the NL_Port shall implicitly logout with the Fabric:

— the NL_Port detected that the L_bit (Login required) was set to 1 in the Loop Initialization Sequence (LI_ID= 'LISA').

— the NL_Port was unable to set to 1 its Fabric Assigned AL_PA bit or its Previously Acquired AL_PA bit in the Loop Initialization Sequence (LI_ID='LIFA' or 'LIPA') (i.e., another NL_Port is using the AL_PA); or,

— the NL_Port has not previously executed a Fabric Login.

Normal responses to a Fabric Login request are:

— the transmitted OPN(00,AL_PS) is returned to the NL_Port. No L_Port on the Loop has accepted the OPNy. The NL_Port shall set its native address identifier to hex '0000XX' (where 'XX' is its AL_PA).

If the NL_Port is capable of providing Fabric services in the absence of an FL_Port (i.e., the NL_Port accepts the well-known address hex 'FFFFFE' as well as its own native address identifier), this NL_Port (known as an F/NL_Port) shall accept OPN(00,x) in addition to its own AL_PA. If this is the first time that the NL_Port is assuming the responsibility of an F/NL_Port, to ensure that all previous Login requests are reset, the F/NL_Port shall make the transition to the NORMAL-INITIALIZE state (REQ(initialize)) and set the L_bit (Login required) to 1 in the Loop Initialization Sequence (LI_ID='LIFA');

NOTE — To prevent another L_Port from winning arbitration, this F/NL_Port should not relinquish control of the Loop (i.e., not transmit CLS or make the transition to the NORMAL-INITIALIZE state) until it is prepared to receive OPN(00,AL_PS).

If the NL_Port is not capable of becoming an F/NL_Port, the NL_Port shall exit Loop Initialization.

— the NL_Port receives an Accept (ACC) Link Service Sequence. The NL_Port shall use the D_ID in the ACC Sequence as its native address identifier and bits 7-0 of the D_ID as its Fabric Assigned AL_PA. The NL_Port shall compare the Fabric Assigned AL_PA in the ACC sequence with the AL_PA acquired prior to step (5):

  – if they are equal, the NL_Port shall exit Loop Initialization or

  – if they are unequal, the NL_Port shall make the transition to the NORMAL-INITIALIZE state (REQ(initialize)) to re-initialize and acquire the Fabric Assigned AL_PA value.

10.5.4 Loop Initialization state diagram

The following text provides a detailed state diagram of the INITIALIZATION process. In clause 10, in case of conflicts between text and figures, the following precedence shall be used: figures and then text. 10.5.4 takes precedence over clause 9 table 12 and table 14, which takes precedence over 8.4.3 item 21 and item 23, which takes precedence over 10.5.3.

All state diagrams in this subclause use the style shown in figure 6.

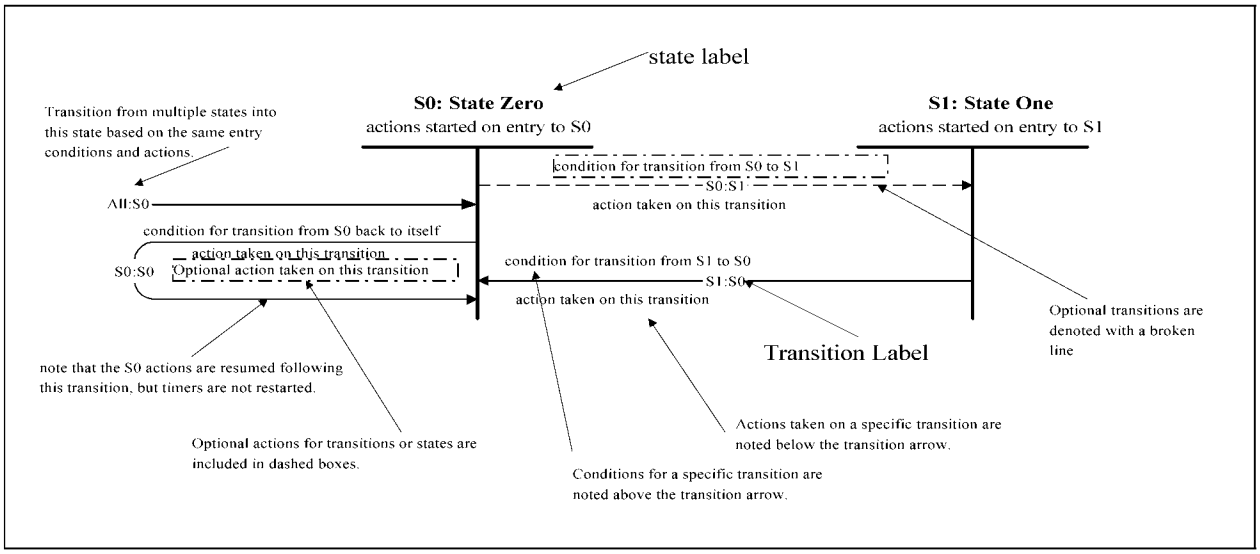

Figure 6 — Loop Initialization state diagram example

These state diagrams are represented using vertical staffs to represent states and horizontal arrows to represent state transitions.

Time elapses only within discrete states with instantaneous transitions between states. Transitions are illustrated with triggering conditions located above the transition arrow and any actions on transition located below the transition arrow. Transition actions are performed while remaining in the previous state, before entry into the new state. The state name appears above the vertical staff representing the state, immediately followed by entry actions if any.

Entry actions are executed every time a state is started. This means that a transition that points back to the same state shall repeat the actions from the beginning. All the actions started upon entry complete before any tests are made to exit the state.

Optional states are depicted with a broken box around the state. Entries to an optional state are shown as solid lines to depict that they are mandatory if the state is implemented. If there is a broken transition line into an optional state this is an optional transition even when the state is implemented.

Broken line transitions into required states, indicate optional transitions.

The following event-processing sequence is assumed:

a) Evaluate all transition conditions from the current state.

b) If a transition condition is satisfied, then:

   1) perform the associated transition actions in the current state;

   2) enter the new state; and,

   3) perform entry actions, if any for the new state.

There are four additional memory elements used in this state diagram definition:

lip_type This is an 8-bit byte that contains a hexadecimal value of the next LIP to be generated.

lip_addr This is an 8-bit byte that contains the hexadecimal value of the address field for the next LIP to be sent.

prev_addr This is an 8 bit byte that contains the previously acquired address if one exists. This is used during the address selection phase of Loop Initialization. It is set to hex 'FF' when it is not valid.

my_addr This is the 8 bit address that this L_Port uses when comparing LIP, LPB, and LPE addresses, and when originating LIPs. It is set to hex 'F7' when it is not valid.

The following state diagrams only define the states necessary to perform Loop Initialization. They do not attempt to document the normal Loop Port State Machine (LPSM) that is documented in a state diagram in clause 8 and in table format in clause 9 of this specification. These state diagrams define the operation of the INITIALIZATION process and the OLD-PORT state.

Transitions notes with the state diagrams, describe transitions that enter or leave state diagrams or need additional clarification.

10.5.4.1 Validity of AL_PA

During Loop Initialization, the AL_PA that an L_Port had previously acquired becomes unstable. This subclause defines the point where the AL_PA is valid for various uses.

Initially, an L_Port does not have an AL_PA. This means that the L_Port cannot respond to addressed LPB, LPE, or LIPyx. Additionally, this L_Port is not participating in normal Loop operation. When Loop Initialization begins, the L_Port may attempt to acquire an AL_PA during the Hard Assigned or Soft Assigned phases of Loop Initialization.

After an L_Port has completed Loop Initialization once with PARTICIPATE is TRUE(1), it has an acquired AL_PA (this value shall be stored in prev_addr and my_addr upon the transition to MONITORING state). This AL_PA also becomes a fabric assigned AL_PA if FLOGI is completed. At this point in time the L_Port may respond to addressed LPB, LPE, LIPyx, in addition to participating in normal Loop operation.

If LIP occurs, all AL_PAs must be revalidated. Because AL_PAs may change during this process, the following rules apply:

1) An L_Port may respond to addressed LPB, LPE, and LIPyx until it has forwarded an ARB(F0). The L_Port shall use the value in my_addr to validate the address in these Primitive Sequences. After an ARB(F0) has been forwarded, the AL_PA is considered unstable, my_addr is set to hex 'F7', and shall therefore only allow the L_Port to validate the address of LPB, LPE, or LIPyx to all L_Ports (i.e., y = hex 'FF') until Loop Initialization has been completed with the CLS being both received and transmitted, at which point the AL_PA acquired during this Loop Initialization is be placed in my_addr, and may be used to validate the addresses in recognized LPB, LPE, and LIPyx Primitive Sequences.

2) An L_Port may attempt to regain its current AL_PA, which is stored in prev_addr, during either the Fabric Assigned (if FLOGI has been completed) or Previously Acquired phase of Loop Initialization. This AL_PA may be used for

regaining the current AL_PA until some other L_Port claims it during Loop Initialization. At this point in time prev_addr is set to hex 'FF' to indicate that it is no longer valid. When Loop Initialization is completed, either the L_Port has a new acquired AL_PA, or is Non-Participating with no AL_PA. If the L_Port is Non-Participating, it shall not recognize any addressed Primitive Signal or Primitive Sequences.

If REQ(nonparticipat.) is asserted in the INITIALIZATION process, an immediate transition to the MONITORING state shall be made with the transition actions being: PARTICIPATE is FALSE(0); my_addr = hex 'F7'; and, prev_addr = hex 'FF'.

If REQ(bypass L_Port) is asserted or if LPBfx is recognized in the INITIALIZATION process (except in the POWER-ON-INIT (P0) and the OLD-PORT (OP0) states) and the state transitions are not explicitly identified, an immediate transition to the MONITORING state shall be made with the transition actions being: BYPASS is TRUE(1); PARTICIPATE is FALSE(0); my_addr = hex 'F7'; and, prev_addr = hex 'FF'.

In all states of the INITIALIZATION process, except P0, Loop Failure causes a transition to the F0 state as defined by All:F0. REQ(initialize) shall be removed upon entry into the NORMAL-INITIALIZE state, unless the L_Port is attempting to bypass or enable other L_Ports. REQ(old port) shall be removed upon entry into OLD-PORT state.

10.5.4.2 POWER-ON state diagram

Figure 7 shows the POWER-ON state diagram. This state diagram maintains the transmitter off until the L_Port is capable of initializing, and accomplishes the transition to the NORMAL-INITIALIZE or MONITORING state.

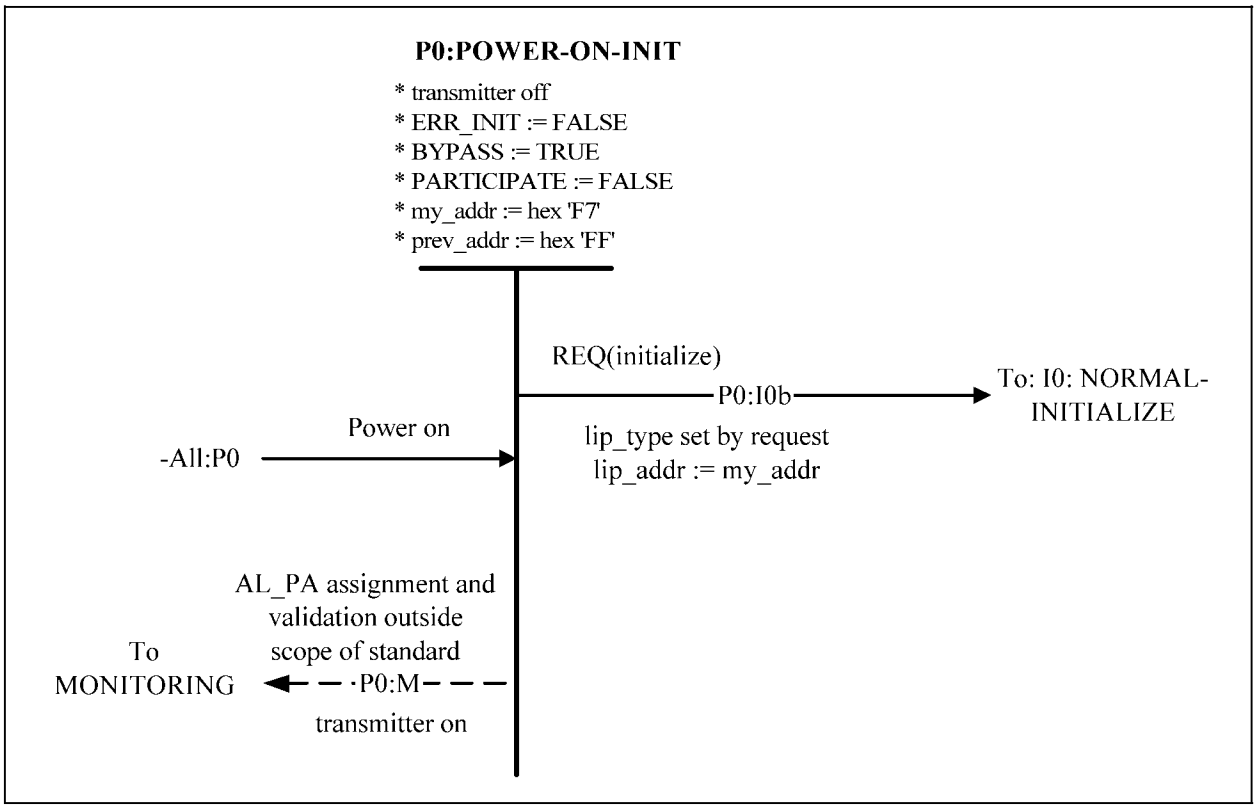

Figure 7 — POWER-ON state diagram

POWER-ON state diagram notes:

Transmitter off requires that for copper links the transmitter is either tri-stated, or driven to a constant value. For optical links, the optical output should be zero. It is not sufficient for optical links to drive a constant intermediate power level, as this may cause the receiver which has very high gain to mistakenly perceive that the remote L_Port is actually transmitting data.

Transition All:P0 This transition is taken from any state within the LPSM at power-on.

Transition P0:I0b This transition is taken to the NORMAL-INITIALIZE state when REQ(initialize) is asserted.

Transition P0:M This optional transition is taken by an L_Port that does not use Loop Initialization to acquire and verify an AL_PA. If Loop Initialization is not used to acquire and verify AL_PAs, my_addr and prev_addr shall be assigned by a method outside the scope of this standard. It is the responsibility of the implementation to assure AL_PAs are valid and that there are no conflicts. Even if the L_Port does not use Loop Initialization to acquire and verify an AL_PA, it shall participate in Loop Initializations initiated by other L_Ports on the Loop.

10.5.4.3 OLD-PORT state diagram

Figure 8 shows the OLD-PORT state diagram. In the OLD-PORT state, the LPSM is not running except to respond to recognized LIPs. The Port State Machine is operating as defined in ANSI X3, FC-PH-x.

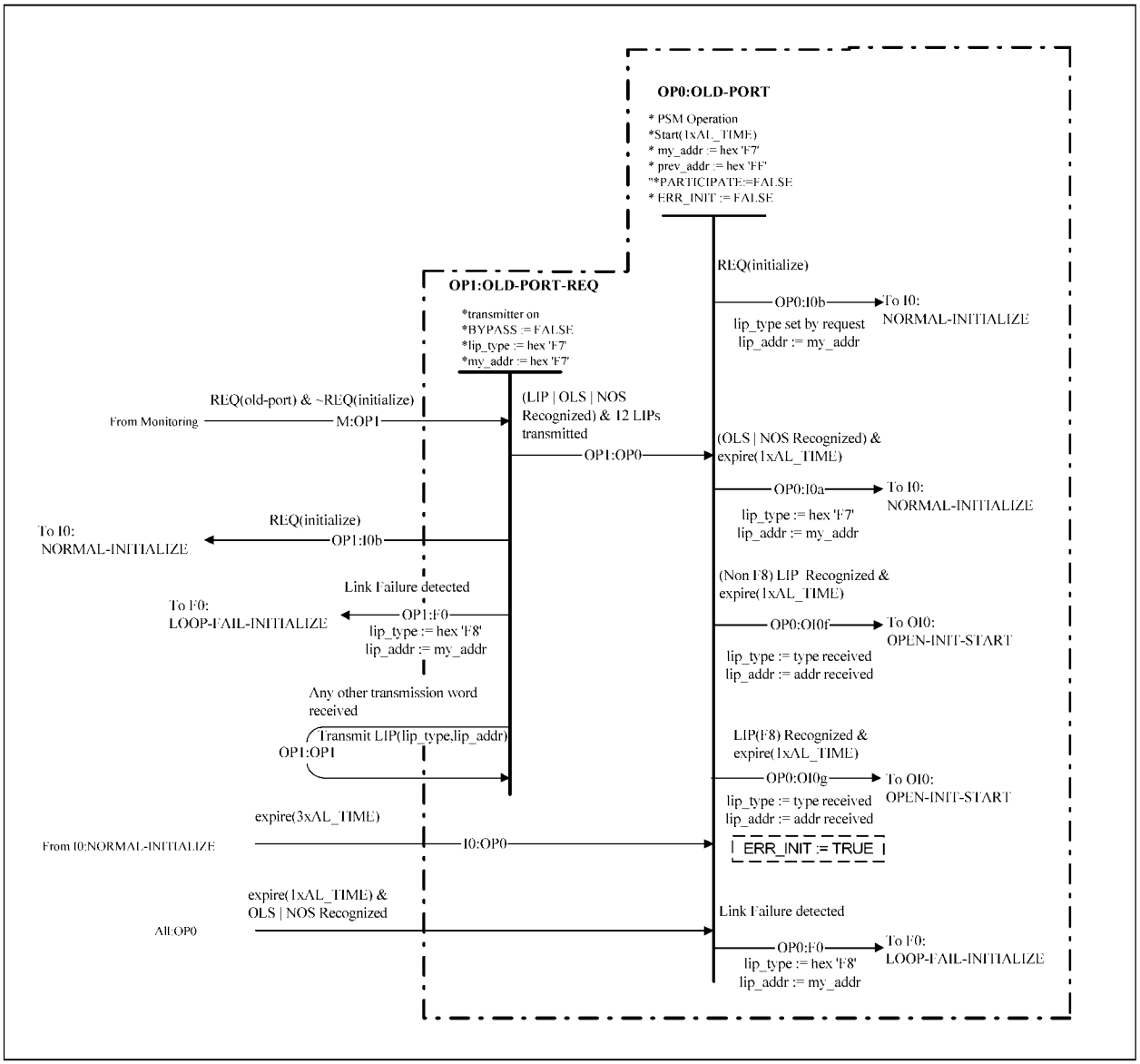

Figure 8 — OLD-PORT state diagram

OLD-PORT state diagram notes:

Transition I0:OP0 This transition is taken from the NORMAL-INITIALIZE state to the optional OLD-PORT state after minimum(3xAL_TIME).

Transition AII:OP0 This transition is taken from the OPEN-INIT-SELECT-MASTER or SLAVE-WAIT-FOR-MASTER state to the optional OLD-PORT state after minimum(1xAL_TIME) if OLS or NOS is recognized.

Transition M:OP1 This transition is taken when REQ(old-port) is asserted and REQ(initialize) is not asserted in the MONITORING state.

Transition OP0:I0b This transition shall be taken to the NORMAL-INITIALIZE state when REQ(initialize) is asserted.

Transition OP0:I0a This transition is taken to the NORMAL-INITIALIZE state when NOS or OLS are recognized, after expire(1xAL_TIME).

Transition OP0:OI0f This transition to the OPEN-INIT-START state is taken when a (Non F8) LIP is recognized after expire(1xAL_TIME).

Transition OP0:OI0g This transition to the OPEN-INIT-START state is taken when LIP(F8) is recognized after expire(1xAL_TIME).

Transition OP0:F0 This transition is taken when BYPASS is FALSE(0) and a Link Failure is detected as defined in ANSI X3, FC-PH-x.

Transition OP1:I0b This transition is taken to the NORMAL-INITIALIZE state when REQ(initialize) is asserted.

Transition OP1:OP1 On any Transmission Word other than LIP, OLS, or NOS, when REQ(initialize) is not asserted, LIP(F7,F7) shall be transmitted.

10.5.4.4 Loop Failure Initialization state diagram

Figure 9 shows the Loop Failure Initialization state diagram. This state diagram attempts to initialize the Loop when a Loop Failure has been detected.

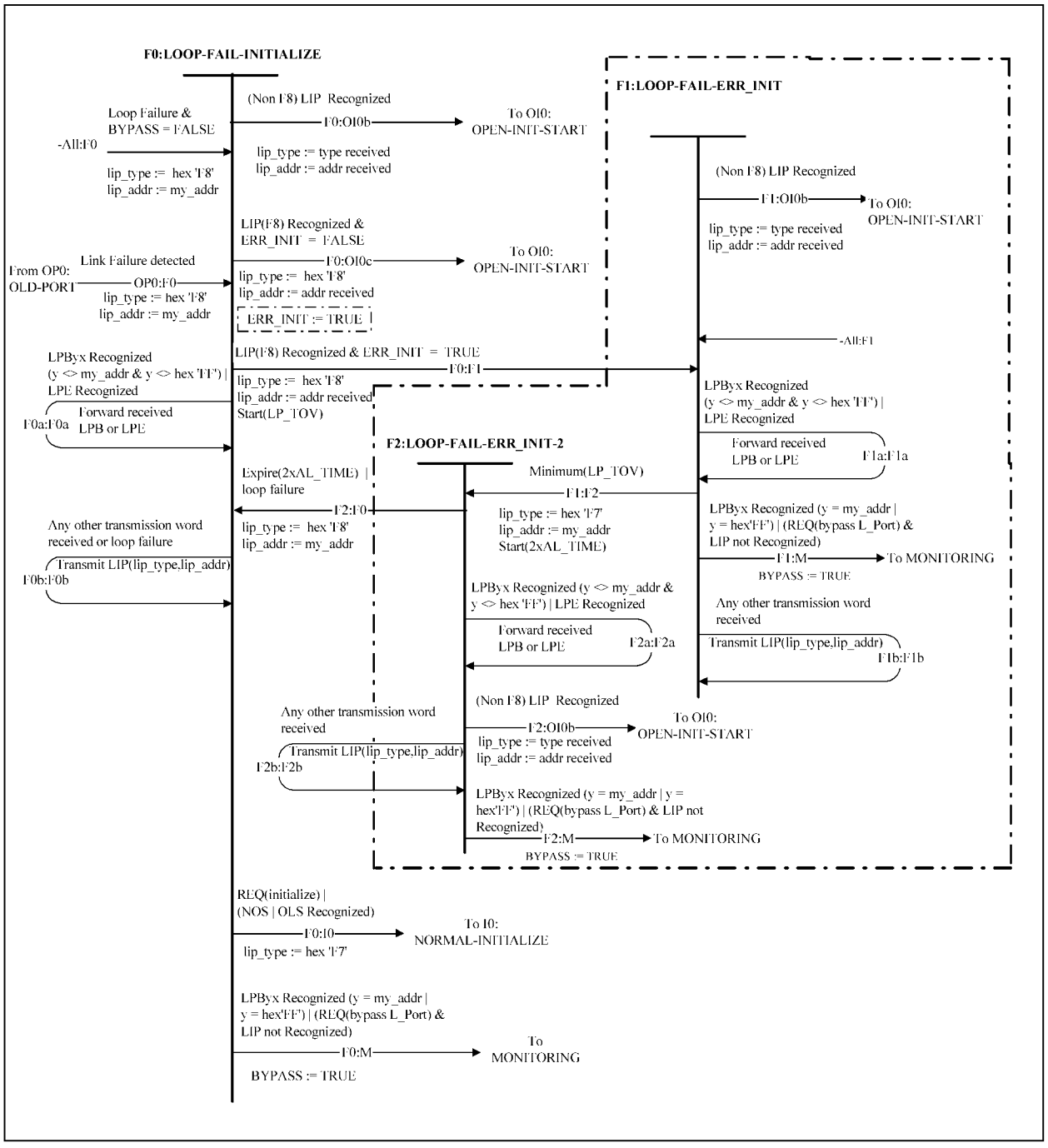

Figure 9 — Loop Fail Initialization state diagram

Loop Fail Initialization state diagram notes:

Transition All:F0 This transition is taken from any state other than P0:POWER-ON-INIT when a Loop Failure is detected and BYPASS is FALSE(0).

Transition F0:I0 This transition is taken from the LOOP-FAIL-INITIALIZE state to the NORMAL-INITIALIZE state when REQ(initialize) is asserted.

Transition All:F1 This transition is taken from the OPEN-INIT-SELECT-MASTER or the SLAVE-WAIT-FOR-MASTER state when LIP(F8) continues to be recognized after the Loop should be flushed of LIPs and ERR_INIT is TRUE(1). This indicates that there is still an error condition on the Loop.

Transition F0:OI0b This transition is taken from the LOOP-FAIL-INITIALIZE state to the OPEN-INIT-START state when a (Non F8) LIP has been recognized by this L_Port, and the remainder of the INITIALIZATION process may be attempted.

Transition F0:OI0c This transition is taken from the LOOP-FAIL-INITIALIZE state to the OPEN-INIT-START state when LIP(F8) has been recognized by this L_Port, and the remainder of the INITIALIZATION process may be attempted. On this transition, the optional ERR_INIT memory element is set TRUE(1).

Transition F0:I0 This transition is taken from the LOOP-FAIL-INITIALIZE state to the NORMAL-INITIALIZE state when NOS or OLS are recognized. This allows for waiting expire(3xAL_TIME) before transitioning to the optional OLD-PORT state.

Transition F0:M This transition is taken from the LOOP-FAIL-INITIALIZE state to the MONITORING state when LPB for this AL_PA or LPBfx is recognized. The L_Port shall remain in the MONITORING state until it is enabled on the Loop and recognizes LIP, or REQ(initialize) is asserted.

Transition F1:OI0b This transition is taken from the LOOP-FAIL-ERR_INIT state to the OPEN-INIT-START state after error initialization has been started and (Non F8) LIP is recognized.

Transition F2:OI0b This transition is taken from the LOOP-FAIL-ERR_INIT-2 state to the OPEN-INIT-START state after error initialization has been started and (Non F8) LIP is recognized.

Transition F0a:F0a When LPB that is not addressed to this L_Port or LPE is recognized it shall be forwarded, and the LPSM shall remain in the LOOP-FAIL-INITIALIZE state.

Transition F0b:F0b If any Transmission Word other than those explicitly identified in the state transitions from LOOP-FAIL-INITIALIZE is recognized, LIP(lip_type, lip_addr) shall be originated and the LPSM shall remain in the LOOP-FAIL-INITIALIZE state.

Transition F1a:F1a When LPB that is not addressed to this L_Port or LPE is recognized it shall be forwarded, and the LPSM shall remain in the LOOP-FAIL-ERR_INIT state.

Transition F1b:F1b If any Transmission Word other than those explicitly identified in the state transitions from LOOP-FAIL-ERR_INIT is recognized, a LIP(lip_type, lip_addr) shall be originated and the LPSM shall remain in the LOOP-FAIL-ERR_INIT state.

Transition F2a:F2a When LPB that is not addressed to this L_Port or LPE is recognized it shall be forwarded, and the LPSM shall remain in the LOOP-FAIL-ERR_INIT-2 state.

TransitionF2b:F2b If any Transmission Word other than those explicitly identified in the state transitions from LOOP-FAIL-ERR_INIT-2 is recognized, a LIP(lip_type, lip_addr) shall be originated and the LPSM shall remain in the LOOP-FAIL-ERR_INIT-2 state.

LOOP-FAIL-ERR_INIT state: This state is used by multi-Loop L_Ports that do not desire to use bandwidth to continually perform Loop Initialization on a Loop that has a failure while attached to another Loop that is operational. These L_Ports can use the ERR_INIT process to periodically test and see if the Loop has become operational. By setting ERR_INIT after LIP(F8) is recognized after expire(1xAL_TIME), these L_Ports can limit the time spent in the OPEN-INIT-START state. These L_Ports shall set ERR_INIT and wait in the LOOP-FAIL-ERR_INIT state until a (Non F8) LIP is recognized, or until a minimum(LP_TOV) timeout occurs. If a minimum(LP_TOV) timeout occurs, an L_Port shall transition to the LOOP-FAIL-ERR_INIT_2 state where it shall generate LIP(F7) until expire(2xAL_TIME) or a (Non F8) LIP is recognized. After expire(2xAL_TIME), the L_Port shall transition through LOOP-FAIL-INITIALIZE back to the LOOP-FAIL-ERR_INIT

state and wait another LP_TOV. This allows the L_Port to only attempt initialization every LP_TOV, limiting the amount of time spent in the INITIALIZATION process.

10.5.4.5 Normal Initialization state diagram

Figure 10 shows the normal initialization state diagram. This state diagram attempts to circulate LIP around the Loop before the address selection phase of initialization is begun.

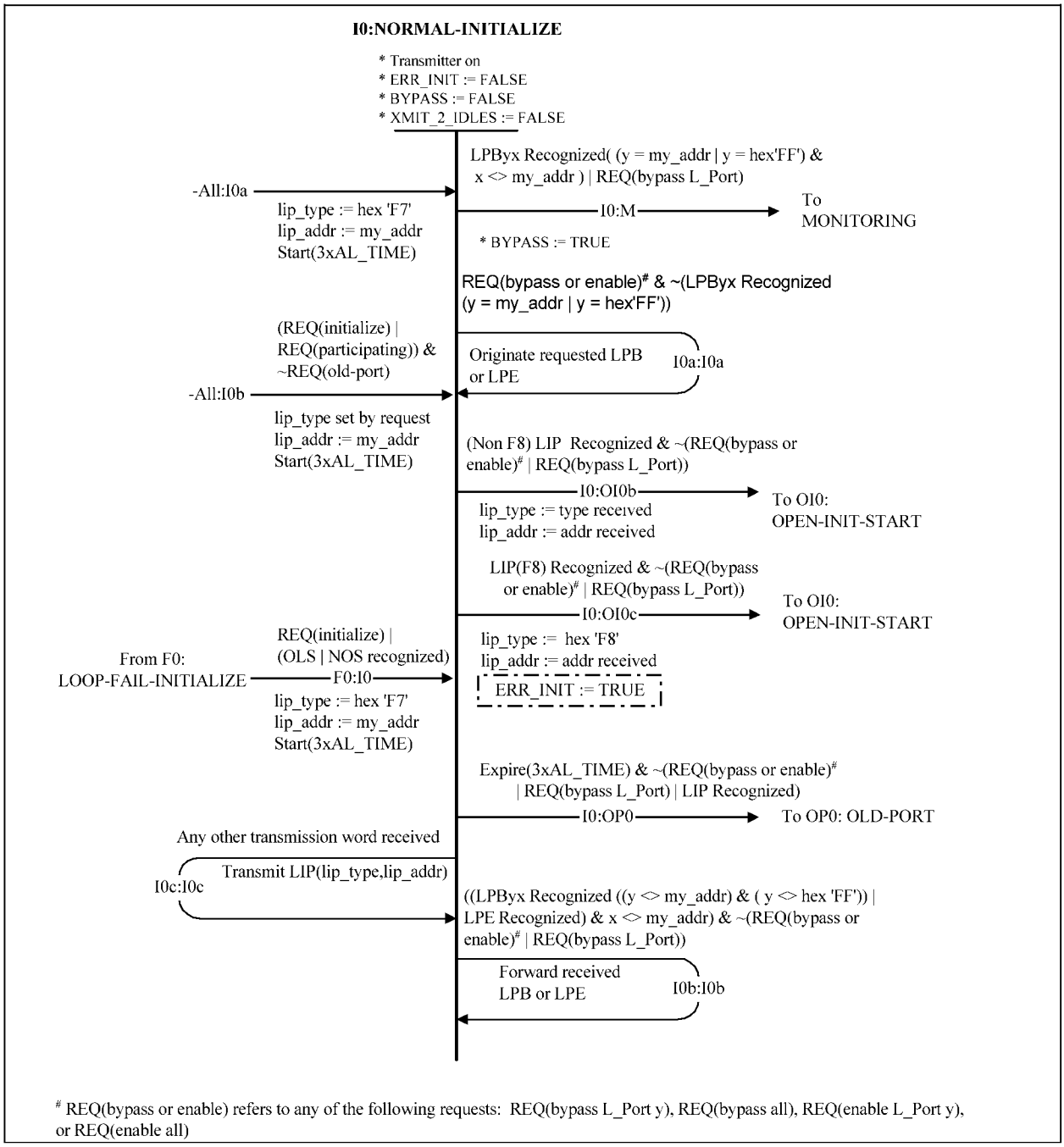

Figure 10 — Normal Initialization state diagram

Normal Initialization state diagram notes:

Unless this L_Port is attempting to manage other L_Port's Port Bypass Circuits, REQ(initialize) shall be removed upon entering the NORMAL-INITIALIZE state.

Transition AII:I0a This transition is made from subsequent states of the INITIALIZATION process when an error in initialization is detected. The errors that are detected are protocol errors or LP_TOV timeout. The transition may actually be controlled from outside the LPSM, and therefore, may take an extended time.

Transition AII:I0b This transition is taken from any state when REQ(participating) or REQ(initializing) is asserted. The type of LIP to be generated is indicated in the request.

Transition I0:OI0b This transition from the NORMAL-INITIALIZE state to the OPEN-INIT-START state is taken when (Non F8) LIP has been recognized, and the address selection phase of initialization can be started.

Transition I0:OI0c This transition is taken when LIP(F8) has been recognized, and the address selection phase of initialization can be started. The LIP that was recognized indicates that there may still be a Loop Failure. If ERR_INIT is supported, it shall be set to TRUE(1) on this transition.

Transition I0:OP0 This transition from the NORMAL-INITIALIZE state to the optional OLD-PORT state is taken when LIP has been transmitted until expire(3xAL_TIME) without LIP being recognized and none of the listed REQs being asserted.

Transition I0:M This transition from the NORMAL-INITIALIZE state to the MONITORING state is taken when LPByx (where y = AL_PA of the L_Port) or LPBfx is recognized; or when REQ(bypass L_Port) is asserted. The L_Port shall remain in the MONITORING state until it is enabled on the Loop and recognizes LIP or REQ(initialize) is asserted while REQ(old-port) is not asserted.

Transition I0a:I0a When LPB that is not addressed to this L_Port or LPE is recognized, it shall be forwarded and the LPSM shall remain in the NORMAL-INITIALIZE state if none of the specified REQs are asserted.

Transition I0b:I0b If a request to bypass or enable (REQ(bypass L_Port y), REQ(bypass all), REQ(enable L_Port y), or REQ(enable all)) is asserted, and if LPB addressed to this L_Port is not recognized, the L_Port shall originate the requested LPB or LPE and remain in the NORMAL-INITIALIZE state.

Transition I0c:I0c If any Transmission Word other than those explicitly identified in the state transitions from NORMAL-INITIALIZE is recognized while none of the specified REQs are asserted, a LIP(lip_type, lip_addr) shall be originated and the LPSM shall remain in the NORMAL-INITIALIZE state.

10.5.4.6 OPEN-INIT state diagram

Figure 11 shows the OPEN-INIT state diagram. The OPEN-INIT state diagram is where the Loop Initialization Master (LIM) is determined for this Loop Initialization.

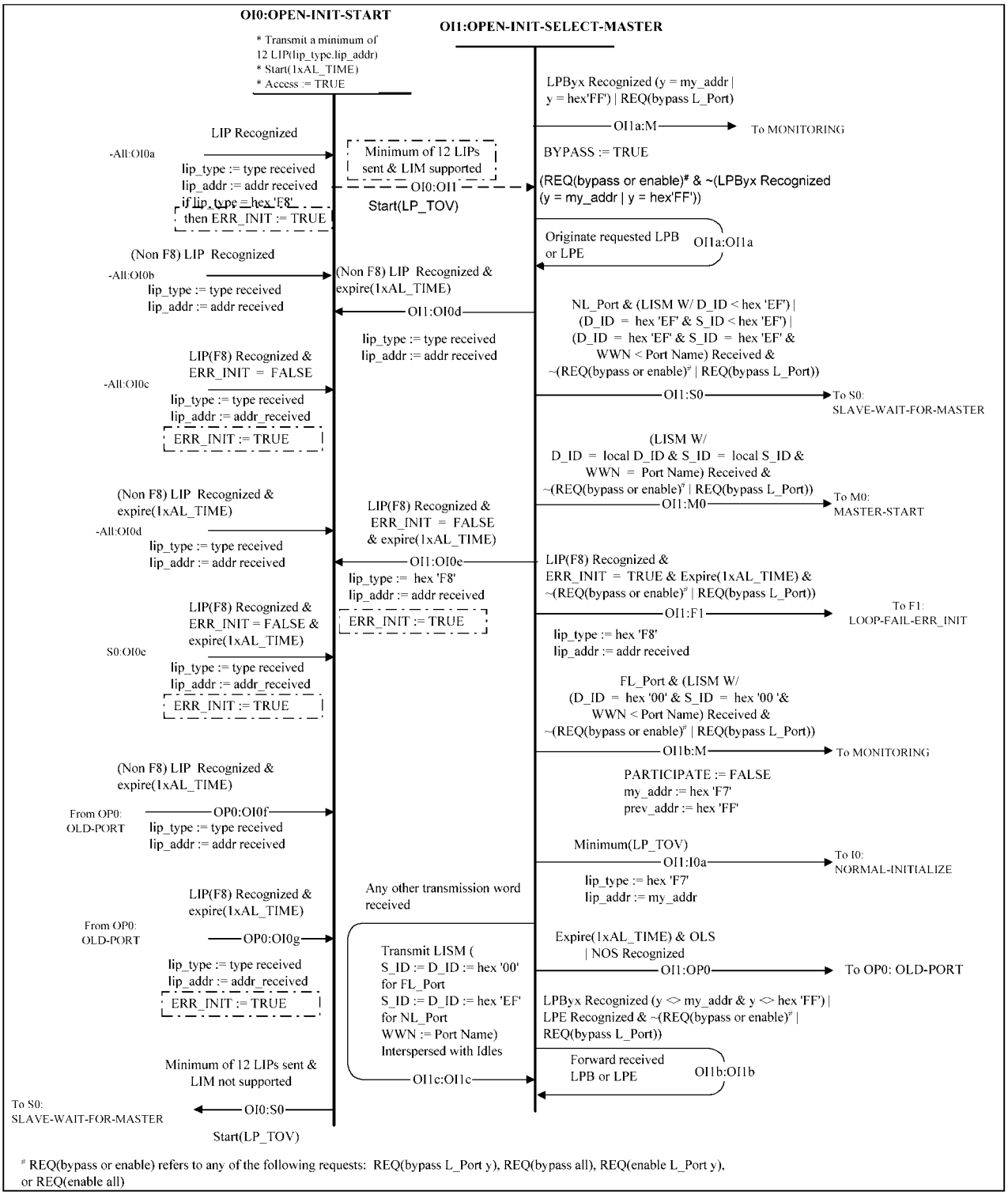

Figure 11 — OPEN-INIT state diagram

OPEN-INIT state diagram notes:

Transition AII:OI0a This transition is taken from any Loop Initialization state if LIP is recognized after the LIM has been selected or from any state outside the INITIALIZATION process if LIP is recognized.

Transition AII:OI0b This transition is the normal entrance to the OPEN-INIT-START state upon recognition of a (Non F8) LIP.

Transition AII:OI0c This transition is the entrance to the OPEN-INIT-START state upon recognition of LIP(F8), when ERR_INIT is FALSE(0), in the LOOP-FAIL-INITIALIZE and NORMAL-INITIALIZE states.

Transition AII:OI0d This is the re-entrance to the OPEN-INIT-START state upon the recognition of a (Non F8) LIP after expire(1xAL_TIME) without successfully selecting a LIM.

Transition S0:OI0e This is the re-entrance to the OPEN-INIT-START state upon the recognition of LIP(F8) when ERR_INIT is FALSE(F0) after expire(1xAL_TIME) without successfully selecting a LIM.

Transition OP0:OI0f This is the entrance to the OPEN-INIT-START state upon the recognition of a (Non F8) LIP after expire(1xAL_TIME) in OLD-PORT state.

Transition OP0:OI0g This is the entrance to the OPEN-INIT-START state upon the recognition of LIP(F8) after expire(1xAL_TIME) in OLD-PORT state.

Transition OI0:S0 This transition from the OPEN-INIT-START state to the SLAVE-WAIT-FOR-MASTER state is taken after forwarding at least 12 received LIP(lip_type,lip_addr) if the L_Port is not capable of performing the LIM functions.

Transition OI1a:M This transition from the OPEN-INIT-SELECT-MASTER state to the MONITORING state is taken when LPByx (where y = AL_PA of the L_Port) or LPBfx is recognized; or when REQ(bypass L_Port) is asserted. The L_Port shall remain in the MONITORING state until it is enabled on the Loop and recognizes LIP or REQ(initialize) is asserted while REQ(old-port) is not asserted.

Transition OI1b:M This transition from the OPEN-INIT-SELECT-MASTER state to the MONITORING state is taken when an FL_Port does not become a LIM.

Transition OI1:I0a This transition from the OPEN-INIT-SELECT-MASTER state to the NORMAL-INITIALIZE state is taken when the L_Port after minimum(LP_TOV) while trying to select a LIM. This takes the L_Port back to attempting initialization.

Transition OI1:S0 This transition is taken when this L_Port has determined that it is not the LIM.

Transition OI1:M0 This transition is taken when this L_Port has determined that it is the LIM.

Transition OI1:F1 This transition from the OPEN-INIT-SELECT-MASTER state to the LOOP-FAIL-INITIALIZE state is taken when LIP(F8) is recognized after expire(1xAL_TIME) and ERR_INIT is TRUE(1).

Transition OI1:OP0 This transition from the OPEN-INIT-SELECT-MASTER state to the optional OLD-PORT state is taken when OLS or NOS is recognized after expire(1xAL_TIME).

Transition OI1a:OI1a If a request to bypass or enable (REQ(bypass L_Port y), REQ(bypass all), REQ(enable L_Port y), or REQ(enable all)) is asserted, and if LPB addressed to this L_Port is not recognized, the L_Port shall originate the requested LPB or LPE and remain in the OPEN-INIT-SELECT-MASTER state.

Transition OI1b:OI1b When an LPB that is not addressed to this L_Port or an LPE is recognized, it shall be forwarded and the LPSM shall remain in the OPEN-INIT-SELECT-MASTER state, if none of the REQs are asserted.

Transition OI1c:OI1c If any Transmission Word other than those explicitly identified in the state transitions from the OPEN-INIT-SELECT-MASTER state is recognized while none of the specified REQs are asserted, LISM shall be originated and the LPSM shall remain in the OPEN-INIT-SELECT-MASTER state.

10.5.4.7 Slave Initialization state diagram

Figure 12 shows the Slave Initialization state diagram. This machine is where the L_Port acquires an address if it is not the LIM. The L_Port enters this state when it has determined that there is another L_Port with higher priority on the Loop or the LIM function is not supported by this L_Port.

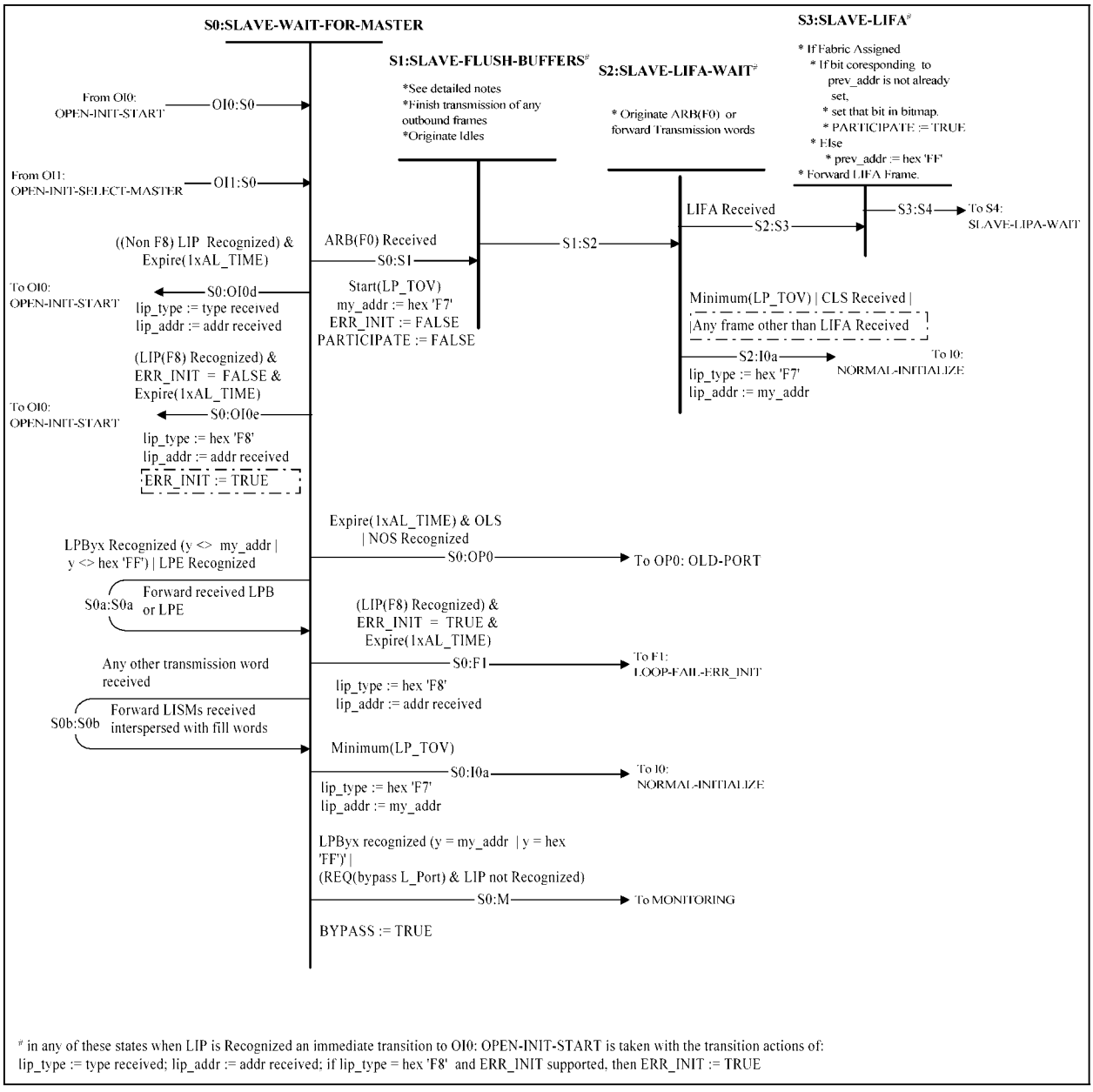

Figure 12 — Slave Initialization state diagram

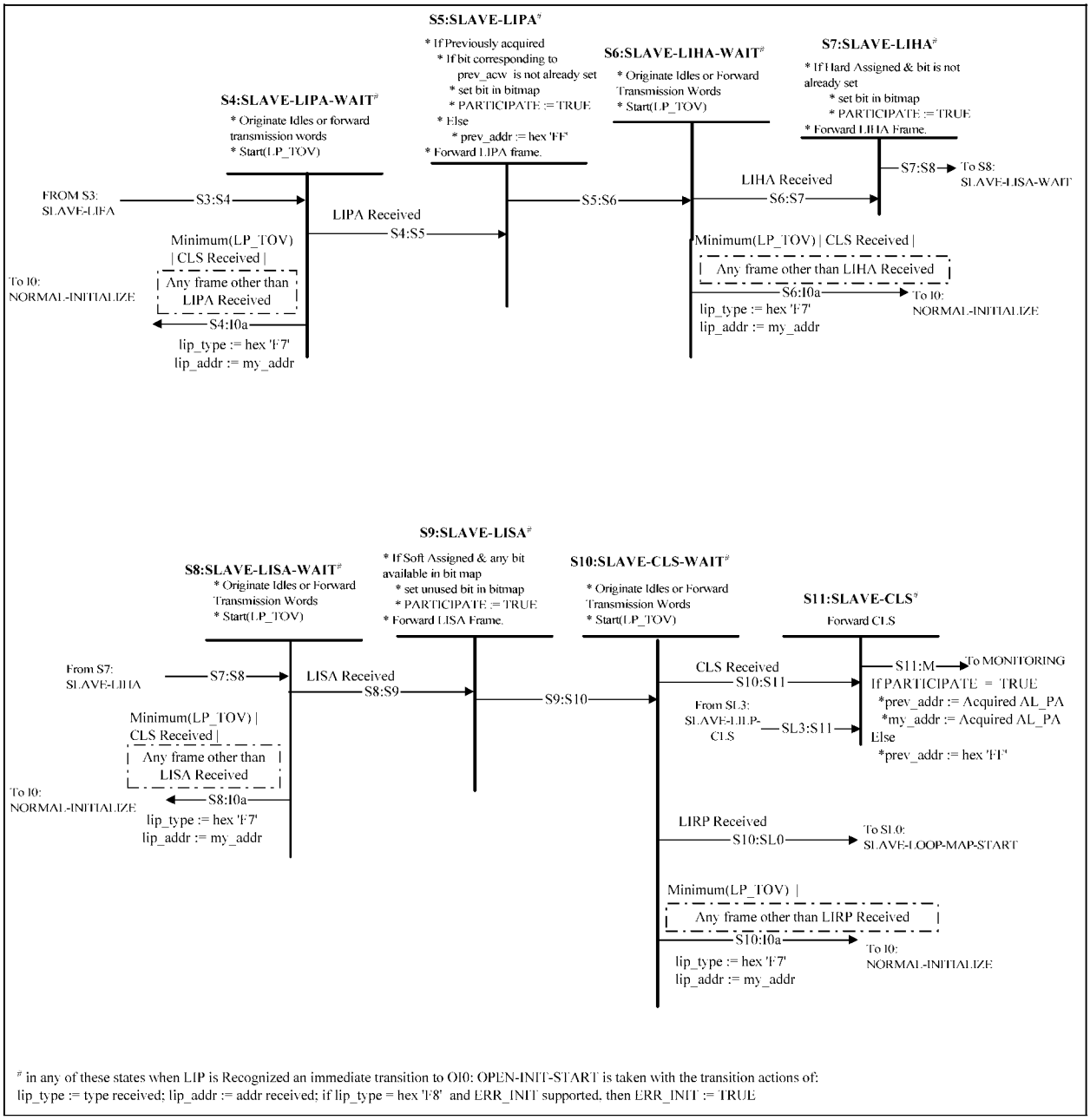

Figure 12 (concluded) — Slave Initialization state diagram

Slave Initialization state diagram notes:

State S0: SLAVE-WAIT-FOR-MASTER is used to forward received LISM frames. Implementations may directly forward frames as they are received; however, not all LISM frames will necessarily be received due to buffer limitations. Additionally, the L_Port may store the most recently received LISM frame and transmit it interspersed with Idles until another LISM is received and stored in the buffer. This may yield LISM transmission at a different rate than reception.

NOTE — Because this state is being timed by LP_TOV, it is important to update the LISM being forwarded. If every device takes more than 15 msec to forward each LISM received, and there are 128 devices on the Loop, the delay to win LIM would exceed LP_TOV.

State S1: SLAVE-FLUSH-BUFFERS is used to ensure that the L_Port is ready to receive the LIFA frame that may be received shortly after forwarding the ARB(F0) around the Loop. In this state, the L_Port should complete sending any frame that it was transmitting when ARB(F0) was received and transmit Idle until it has a buffer to receive the LIFA frame. Upon reception of LIP in this state, the transition All:OI0a shall be taken back to the OPEN-INIT-START state.

NOTE — Because this state and the next is being timed by LP_TOV, it is important that the buffers be flushed quickly, and ARB(F0) forwarded quickly. If each device takes more than 15 msec to flush buffers and originate or forward the LIFA frame, the total delay, would exceed LP_TOV.

Transition OI0:S0 This transition is the starting point for the Slave Initialization state diagram when the LIM is not supported. It is reached directly from the OPEN-INIT-START state.

Transition OI1:S0 This transition is the starting point for the Slave Initialization state diagram when the LIM is supported (but the L_Port yielded to another L_Port). It is reached when the L_Port determines in the OPEN-INIT-SELECT-MASTER state that it is not the LIM.

Transition S0:OI0d This transition is taken when a (Non F8) LIP is recognized in the SLAVE-WAIT-FOR-MASTER state after expire(1xAL_TIME). This causes the L_Port to return to the OPEN-INIT-START state.

Transition S0:OI0e This transition is taken when the L_Port recognized LIP(F8) when ERR_INIT is FALSE(0) after expire(1xAL_TIME). This transition sets the ERR_INIT flag to TRUE(1) if ERR_INIT is supported.

Transition All:OI0a This transition is taken when any LIP is recognized in the SLAVE-FLUSH-BUFFERS, SLAVE-LIFA-WAIT, SLAVE-LIFA, SLAVE-LIPA-WAIT, SLAVE-LIPA, SLAVE-LIHA-WAIT, SLAVE-LIHA, SLAVE-LISA-WAIT, SLAVE-LISA, SLAVE-CLS-WAIT, SLAVE-CLS states. This causes the L_Port to return to the OPEN-INIT-START state and set ERR_INIT to TRUE(1) if the lip_type is hex 'F8' and if ERR_INIT is supported.

Transition S0:F1 This transition to the LOOP-FAIL-ERR-INIT state is taken when LIP(F8) is recognized and ERR_INIT is TRUE(1).

Transition S0:OP0 This transition from the SLAVE-WAIT-FOR-MASTER state to the optional OLD-PORT state is taken when OLS or NOS is recognized after expire(1xAL_TIME).

Transition All:I0a This transition occurs in any of the indicated states when an error is detected in the address selection machine. The errors that are detected are protocol errors or LP_TOV timeouts. The transition may actually be controlled from outside the LPSM, and therefore, may take an extended time. These transitions go back to the NORMAL-INITIALIZE state and attempt to begin initialization again using LIP(F7).

Transition S10:SL0 This transition is made after all addresses have been assigned, and all L_Ports are capable of generating an AL_PA position map. This transition is to the SLAVE-LOOP-MAP-START state in the Slave AL_PA position map generation state diagram.

Transition S11:M This transition to the MONITORING state is the completion of Loop Initialization. At this point the L_Port is participating in the Loop if PARTICIPATE is TRUE(1), or it remains in the MONITORING state and forgets any previously acquired AL_PA as a Non-Participating L_Port if PARTICIPATE is FALSE(0).

Transition SL3:S11 This transition is from the SLAVE-LILP-CLS state in the Slave AL_PA position map state diagram after it has received CLS. It transitions here, generates CLS and finishes the INITIALIZATION process.

Transition S0a:S0a When an LPB that is not addressed to this L_Port or an LPE is recognized it shall be forwarded, and the LPSM shall remain in the SLAVE-WAIT-FOR-MASTER state.

Transition S0b:S0b If any Transmission Word other than those explicitly identified in the state transitions from SLAVE-WAIT-FOR-MASTER is recognized, received LISMs are forwarded and the LPSM shall remain in the SLAVE-WAIT-FOR-MASTER state. If no LISM frames have been received, the CFW shall be transmitted.

10.5.4.8 Slave AL_PA position map state diagram

Figure 13 shows the Slave AL_PA position map state diagram. All FC-AL-2 L_Ports must implement this; however, if there are FC-AL-1 L_Ports on the Loop this state diagram may not be used.

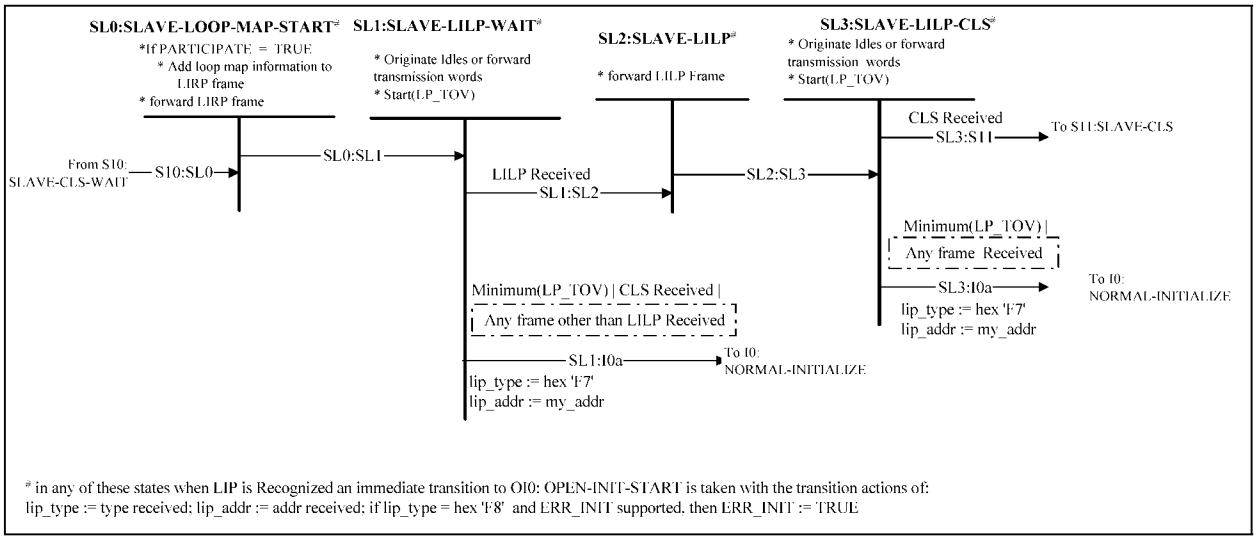

Figure 13 — Slave AL_PA position map state diagram

Slave AL_PA position map state diagram notes:

Transition S10:SL0 This is the main entry point to the Slave AL_PA position map state diagram from the Slave Initialization state diagram.

Transition All:OI0a This transition is taken when LIP is recognized in the SLAVE-LOOP-MAP-START, SLAVE-LILP-WAIT, SLAVE-LILP, SLAVE-LILP-CLS states. This causes the L_Port to return to the OPEN-INIT-START state.

Transition All:I0a This transition is taken when an error is detected in any of the indicated states of the address selection machine. The errors that are detected are protocol errors or LP_TOV timeouts. The transition may actually be controlled from outside the LPSM, and therefore, may take an extended time. These transitions go back to the NORMAL-INITIALIZE state and attempt to begin initialization again.

Transition SL3:S11 This transition is taken to S11:SLAVE-CLS, when the AL_PA position map process is completed and CLS has been received. It is the final transition before Loop Initialization is completed in the Slave Initialization state diagram.

10.5.4.9 Master Initialization state diagram

Figure 14 shows the Master Initialization state diagram. This machine is where the L_Port acquires an AL_PA if it is the LIM. The L_Port enters this state when it has received its own LISM frame back.

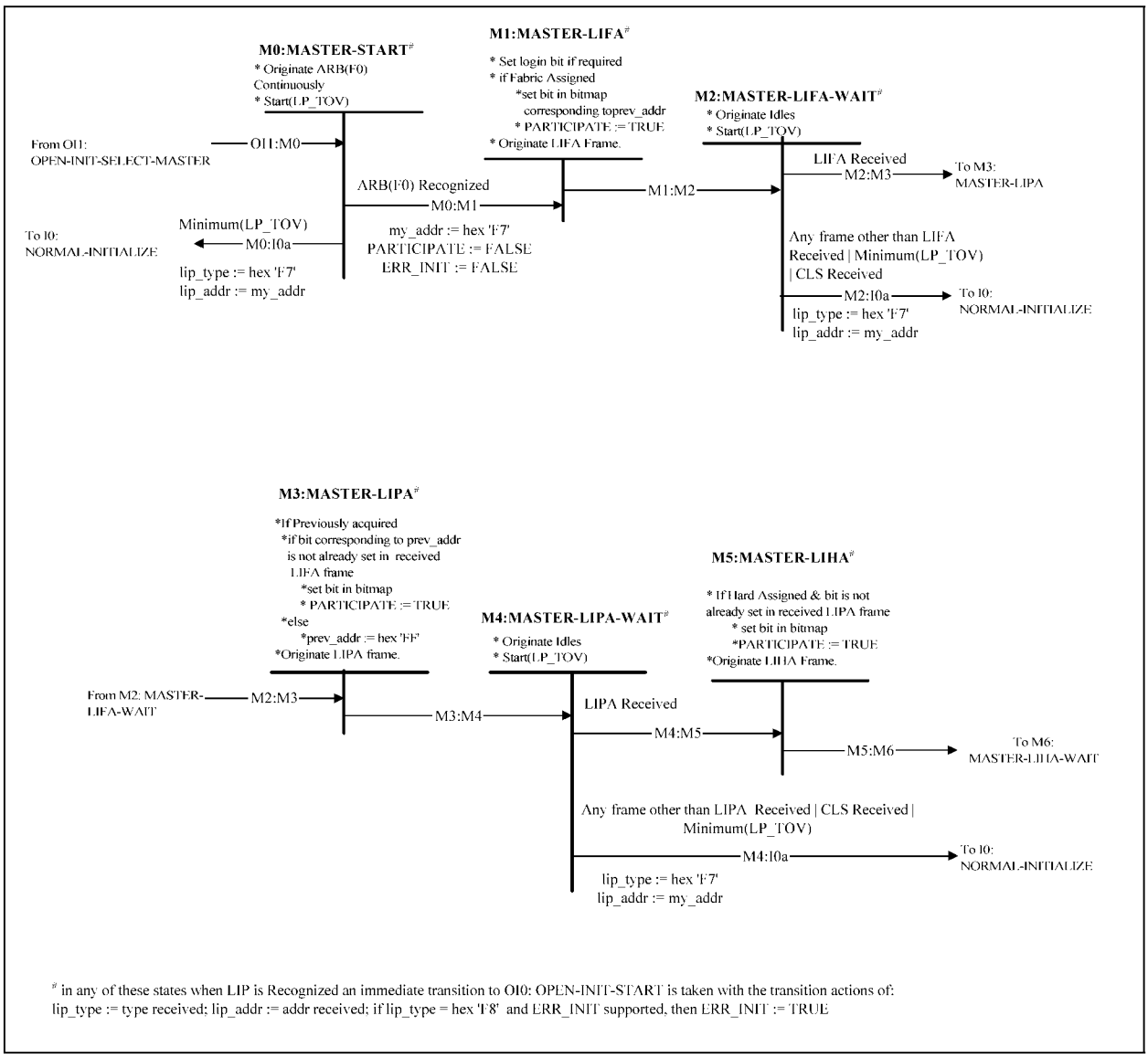

Figure 14 — Master Initialization state diagram

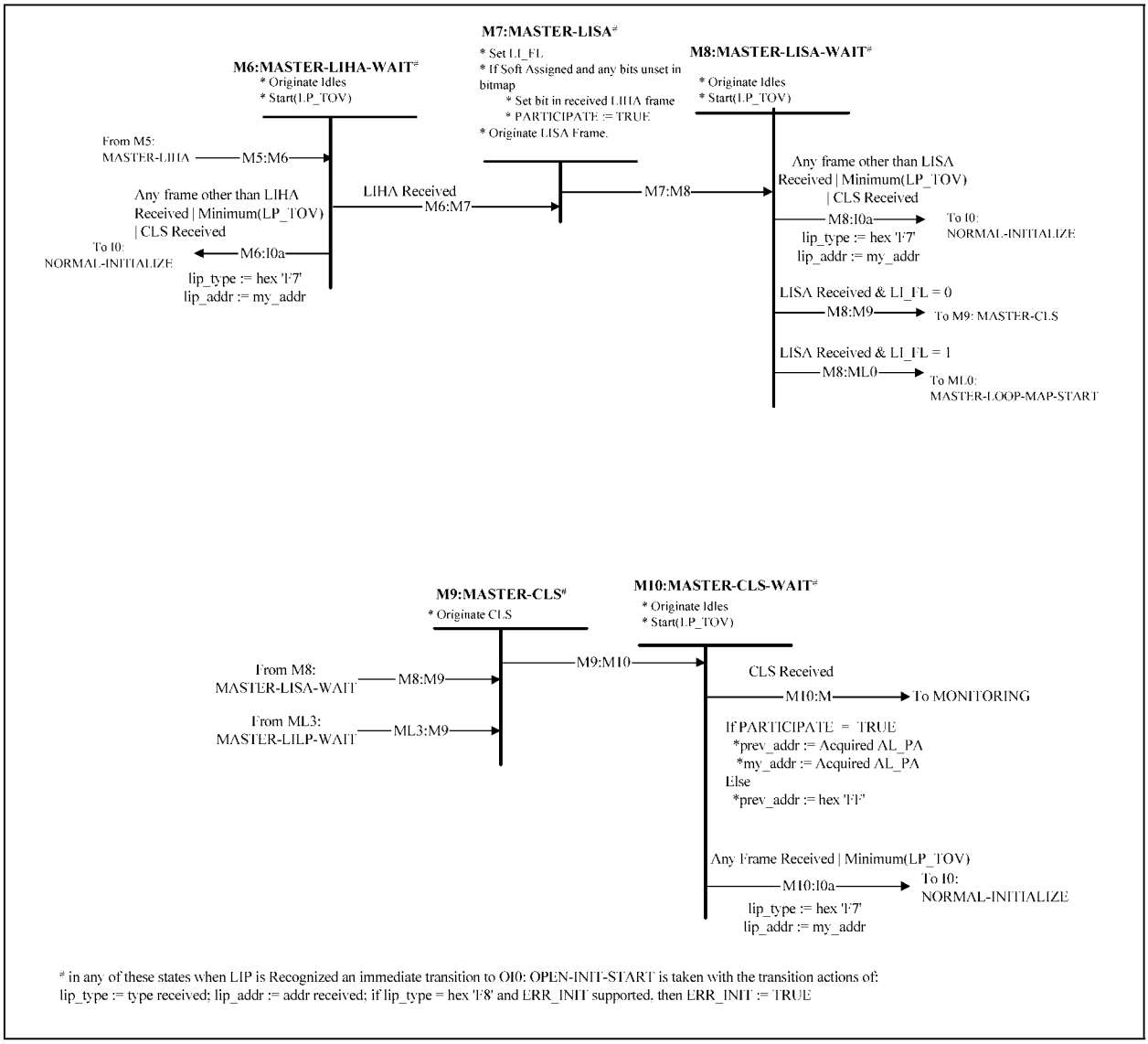

Figure 14 (concluded) — Master Initialization state diagram

Master Initialization state diagram notes:

Transition OI1:M0 This transition is the starting point for the Master Initialization state diagram. It is reached when the L_Port determines in the OPEN-INIT-SELECT-MASTER state that it is the LIM.

Transition AII:OI0a This transition is taken when LIP is recognized in the MASTER-LIFA, MASTER-LIFA-WAIT, MASTER-LIPA, MASTER-LIPA-WAIT, MASTER-LIHA, MASTER-LIHA-WAIT, MASTER-LISA, MASTER-LISA-WAIT, MASTER-CLS, MASTER-CLS-WAIT states. This causes the L_Port to return to the OPEN-INIT-START state.

Transition AII:I0a This transition is taken when an error is detected in any of the indicated states of the address selection machine. The errors that are detected are protocol errors or LP_TOV timeouts. The transition may actually be controlled from outside the LPSM, and therefore, may take an extended time. These transitions go back to the NORMAL-INITIALIZE state and attempt to begin initialization again.

Transition M8:ML0 This transition to the MASTER-LOOP-MAP-START state is taken after all addresses have been assigned, and all L_Ports are capable of generating an AL_PA position map.

Transition M10:M This transition to the MONITORING state is the completion of Loop Initialization. At this point the L_Port is participating in the Loop if PARTICIPATE is TRUE(1), or it remains in the MONITORING state and forgets any previously acquired AL_PA as a Non-Participating L_Port if PARTICIPATE is FALSE(0).

Transition ML3:M9 This transition is taken from the MASTER-LILP-WAIT state, and indicates that Loop Initialization is ready to be completed. The LIM must generate CLS to complete Loop Initialization.

10.5.4.10 Master AL_PA position map state diagram

Figure 15 shows the Master AL_PA position map state diagram. All FC-AL-2 L_Ports shall implement this; however, if there are FC-AL-1 L_Ports on the Loop this state diagram may not be used.

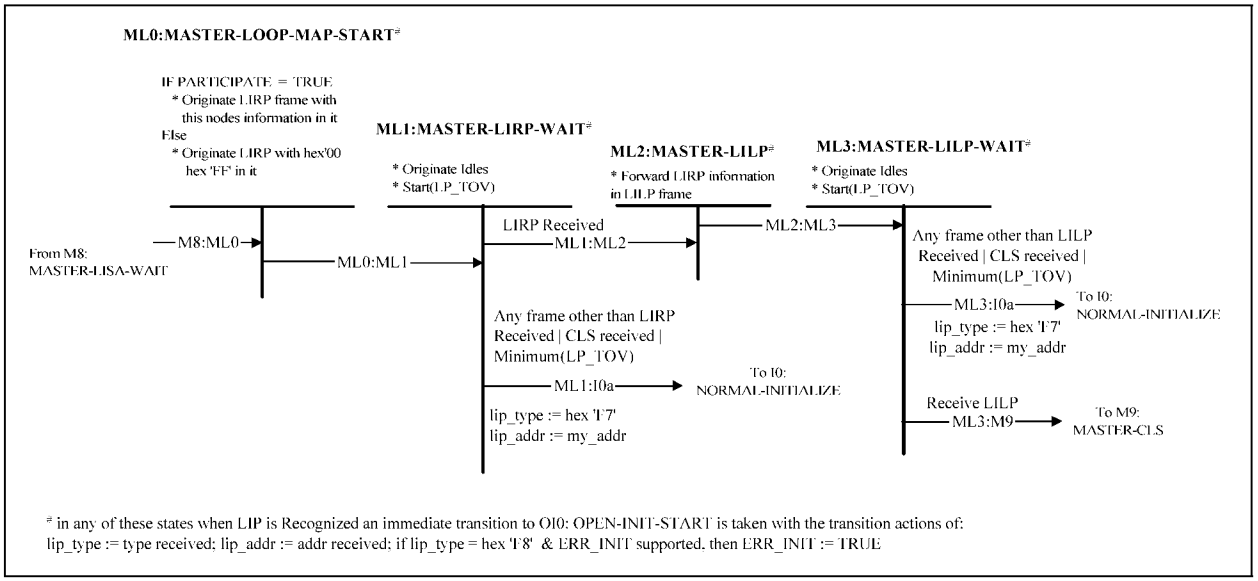

Figure 15 — Master AL_PA position map state diagram

Master AL_PA position map state diagram notes:

Transition M8:ML0 This is the main entry point to the Master AL_PA position map state diagram from the Master Initialization state diagram.

Transition All:OI0a This transition is taken when LIP is recognized in the MASTER-LOOP-MAP-START, MASTER-LIRP-WAIT, MASTER-LILP, MASTER-LILP-WAIT states. This causes the L_Port to return to the OPEN-INIT-START state.

Transition All:I0a This transition is taken when an error is detected in any of the indicated states of the address selection machine. The errors that are detected are protocol errors or LP_TOV timeouts. The transition may actually be controlled from outside the LPSM, and therefore, may take an extended time. These transitions go back to the NORMAL-INITIALIZE state and attempt to begin initialization again.

Transition ML3:M9 This transition is taken to MASTER-CLS, when the AL_PA position map process is completed.

Annex A
(normative)

L_Port Elasticity buffer management

This annex defines the L_Port elastic buffer and the clock skew management rules for inserting and deleting Fill Words. The elasticity buffer provides buffering between the receiver input and the transmitter to prevent over-run and under-run conditions at the transmitter. To prevent L_Ports from being starved of opportunities to delete and minimize buffer requirements in all L_Ports, a two priority algorithm for clock skew management delete operations is incorporated. For a description of the elasticity buffer function and example see annex G.

A.1 L_Port elasticity buffer implementation

The elasticity buffer shall be implemented as a first-in-first-out (FIFO) device with the input coming from the receiver logic and the output going to the transmitter logic. An example of the elasticity buffer is illustrated in figure A.1. Buffering required for clock resynchronization is not shown.

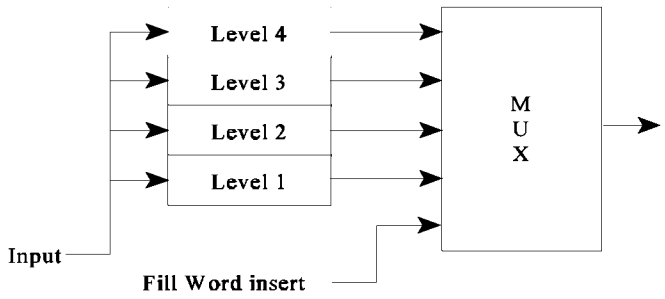

Figure A.1 — Elasticity buffer

Only valid Transmission Words are entered into the elasticity buffer. The information content of the valid Transmission Words entered into the buffer is called *valid information*. The buffer is divided into four levels. Each level represents buffering for a clock skew management state.

The buffer may be implemented either in bit, character, half word (two characters), or word wide units. The amount of valid information in the buffer is a count of the units entered into the buffer minus the units removed at the output.

NOTE — It is recommended to use either a character or half word wide buffer. Bit wide implementations require a very high logic speeds for implementation. Word wide implementations do not provide a fine enough unit of measuring the level of valid information in the buffer.

A.2 Clock skew management

Clock skew management inserts and deletes Transmission Words to control the amount of valid information in the buffer. These operations are performed outside of FC-2 frames to allow the frames to be forwarded without modification. For clock skew management, Fill Words or any Ordered Set defined for use as a Primitive Sequence shall be treated equally.

The insertion of Fill Words is required when the receive clock is slower than the transmit clock to prevent buffer under-run. Deletion of Fill Words is required when the receive clock is faster than the transmit clock to prevent buffer over-run.

A.3 Clock skew management states

Clock skew management has 4 states as shown in figure A.2: insertion pending, quiescent, low priority deletion pending, and high priority deletion pending. The current state is determined by the amount of valid information in the buffer.

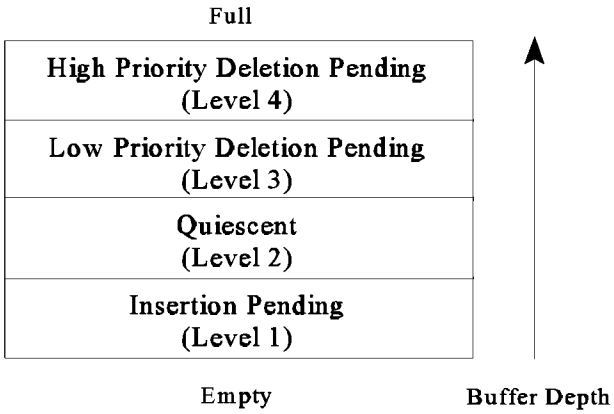

Figure A.2 — Clock skew management states

A.3.1 Insertion pending

To allow an FC-2 frame to be transmitted unmodified, at least a minimum amount of valid information must be in the buffer to ensure buffer under-run will not occur during the frame transmission.

Rule for insertion: When the amount of valid information is less than level 2, in level 1, the L_Port shall insert the current Fill Word immediately after any Fill Word.

A.3.2 Quiescent

When the amount of valid information in the buffer is greater than the level requiring an insertion and less than the level requiring a deletion, in level 2, the clock skew management algorithm is in the quiescent state. No requests to change the buffer depth are pending. This is the nominal state of the clock skew management.

A.3.3 Deletion pending

At least a minimum of free space must be in the buffer to prevent buffer over-run during frame retransmission. Transmission Words may be deleted to reduce the depth of the buffer to provide free space.

Deletion of Transmission Words is more difficult than insertion. When a deletion is required between frames, e.g., in an inter-frame gap, a minimum number of Primitive Signals shall be maintained between frames to meet ANSI X3, FC-PH-x requirements and the Idle Primitive Signals which reset the fairness window shall be propagated.

When frames are originated onto the Loop, the FC-PH requirement of six Primitive Signals between frames is followed. The inter-frame gap may be reduced to 2 Fill Words plus other Primitive Signals by clock skew management. To prevent Ports from being starved of opportunities to delete, a two priority algorithm shall be used.

A.3.3.1 Low priority deletion pending

When the amount of valid information in the buffer triggers a request to delete, the low priority rules for deletion are followed. The low priority rules protect some Fill Words in inter-frame gaps for L_Ports requiring a more critical delete after all the low priority deletes have occurred.

Low Priority Rules: When the amount of valid information reaches level 3, the L_Port shall:

— after 4 Fill Words with no intervening non-Ordered Set (data words), the L_Port deletes the next Fill Word and

NOTE — A non-Ordered Set indicates an intervening frame. By detecting non-Ordered Sets, the L_Port is not required to detect SOF and EOF frame delimiters. New frame delimiters may be defined in the future.

— if the CFW changes to Idle while a delete is pending, the L_Port shall not delete the first Idle.

NOTE — This Idle is protected to allow the LPSM to manage the fairness window as required.

After a low priority Transmission Word delete, the L_Port shall:

— enter the low priority state and wait 4 Fill Words before another delete or

— enter the quiescent state with no delete pending.

Examples:

```
EOF FW FW FW FW FW FW SOF
                  ^  ^            May delete one for low priority
EOF AR AR AR AR ID ID SOF
                     ^            May delete one for low priority
EOF AR AR AR ID ID ID SOF
                  ^  ^            May delete one for low priority
EOF FW FW RR FW FW RR FW FW SOF
                        ^  ^      May delete one for low priority

Legend:
EOF = End of Frame                      AR  = ARByx
SOF = Start of Frame                    ID  = Idle
FW  = Fill Word (ARByx or Idle)         RR  = R_RDY
```

A.3.3.2 High priority deletion pending

L_Ports that are close to over-running their buffers follow the high priority rules for deletion.

High Priority Rules: When the amount of valid information reaches level 4, the L_Port shall:

— after 2 Fill Words with no intervening non-Ordered Set (data words), the L_Ports deletes the next Fill Word and

— if the CFW changes to Idle while a delete is pending, the L_Port shall not delete the first Idle

After a high priority Transmission Word delete, the L_Port shall:

— enter the low priority state and wait 4 Fill Words before another delete or

— re-enter the high priority state and wait 2 Fill Words before another delete depending on buffer free space.

Examples:

```
EOF FW FW FW FW FW FW SOF
          ^  ^  ^  ^            May delete one for high priority

EOF AR AR AR ID ID ID SOF
          ^     ^  ^            May delete one for high priority

EOF FW FW RR FW FW RR FW FW SOF
             ^  ^     ^  ^      May delete one for high priority

EOF FW FW FW FW FW FW SOF
          ^        ^            May delete both (third and sixth) for high priority

Legend:
EOF = End of Frame                    AR  = ARByx
SOF = Start of Frame                  ID  = Idle
FW  = Fill Word (ARByx or Idle)       RR  = R_RDY
```

A.4 Buffer size

The size of the elasticity buffer allows for maximum phase and frequency mismatch between the transmitter and receiver for the length of the largest frame size and the number of frames before a deletion opportunity occurs. See annex G for a description of the worst case period between clock skew management operations. The objective is to keep the size of the elasticity buffer to a minimum. This will ensure minimum Port latency and maximize Loop performance.

The size of the buffer is the sum of space required for each clock skew management state.

The L_Port shall implement a buffer space of at least .25 word (1 character) in level 1 for the insertion pending state.

The L_Port shall implement a buffer space of at least 1 word (4 characters) in level 2 for the quiescent state.

The L_Port shall implement a buffer space of at least 1 word (4 characters) in level 3 for the low priority deletion state.

The L_Port shall implement a buffer space of at least 1 word (4 characters) in level 4 for the high priority deletion state.

Annex B
(informative)

Loop Port State Machine examples

The two examples in this annex use seven of the nine states in the LPSM and twelve of the twenty-two state transitions as defined in 8.4. Of the ten unused state transitions, four are for rare events or error handling. Therefore, much of the Loop protocol is covered in these two simple examples.

B.1 L_Port initialization example

The general, error free procedure for taking the LPSM of a Public NL_Port through Loop Initialization to the point of Fabric Login follows (see clause 8 for reference items and LPSM transitions):

— The NL_Port powers on and attempts to join the Loop;

— The NL_Port may use a trusted AL_PA (since it does not have a valid AL_PA) to instruct the LPSM to arbitrate (REQ(arb own AL_PA)) and to initialize (REQ(initialize)).

The LPSM makes transition (01);

— The LPSM, now in the ARBITRATING state, begins to replace any Idle or ARB(val) (where val is higher than the trusted AL_PA) with ARB(val) (where val is a trusted AL_PA). The LPSM monitors its inbound fibre for the ARB(val) which it transmitted. (See 8.4.3 item 14 for details);

The LPSM makes transition (12);

— The LPSM, now in the ARBITRATION WON state, detects (REQ(initialize)) (see 8.4.3 item 15 for details);.

The LPSM makes transition (28);

— The LPSM, now in the INITIALIZATION process, performs Loop Initialization as described in clause 10 and waits for CLS to indicate that the INITIALIZATION process has completed.

— The LPSM detects CLS.

The LPSM makes transition (80);

— The NL_Port, now in the MONITORING state, is ready to execute a Fabric Login if it is a Public NL_Port (see 10.5, step (5)).

The complete list of state transitions and states in the order used is: (01), 1: ARBITRATING; (12) 2: ARBITRATION WON; (28), 8: INITIALIZATION process; (80), 0: MONITORING.

B.2 N_Port Login example

After the NL_Port initialization procedure has completed (see 10.5), and the NL_Port has a native address identifier (and an AL_PA) and is in Participating mode, a general, error-free procedure for performing NL_Port Login with another NL_Port is described below. In this example, one NL_Port AL_PA is hex '26'; the other NL_Port AL_PA is hex '32'. The example assumes: there is no Participating FL_Port; the arbitrating NL_Port is using the access fairness algorithm; BB_Credit is the default value zero (0) for both NL_Ports; and, both NL_Ports start from the MONITORING state.

— NL_Port 26 arbitrates to access the Loop (REQ(arbitrate as 26)). Assume that the variable ACCESS is set to TRUE(1) (see 8.4.3, item 14 for details).

The LPSM makes transition (01);

— NL_Port 26, now in the ARBITRATING state, can arbitrate because its access window has been reset (ACCESS is TRUE(1). The LPSM begins replacing all Idles and lower priority ARB(val) with its own ARB(26,26) (see 8.4.3, item 14 for details).

The LPSM monitors for ARB(26,26) on its inbound fibre.

Assuming no higher priority NL_Port is arbitrating, ARB(26,26) is received.

The LPSM makes transition (12);

— NL_Port 26, now in the ARBITRATION WON state, must decide whether to open the Loop or not. ACCESS is set to FALSE(0). In this example, the Loop is to be opened (REQ(open 32,26)). (See 8.4.3, item 15 for details.)

The NL_Port 26 transmits OPN(32,26) to cause the other NL_Port to go to the OPENED state in full-duplex mode.

The LPSM makes transition (23);

— NL_Port 26, now in the OPEN state in full-duplex mode, follows OPN(32,26) with one or more R_RDY (one for each available receive buffer) and the CFW until a frame is transmitted. ARB_WON is set to TRUE(1). (See 8.4.3, item 16 for details.)

Concurrently, NL_Port 32, in the MONITORING state, receives OPN(32,26) and goes to the OPENED state.

The LPSM for NL_Port 32 makes transition (04);

— NL_Port 32, now in the OPENED state, transmits the CFW to replace OPN(32,26) on the Loop, followed by one or more R_RDYs (one for each available receive buffer) and the CFW until a frame is transmitted. ARB_WON is set to FALSE(0). DUPLEX is set to TRUE(1). (See 8.4.3, item 17 for details);

— NL_Port 26 is in the OPEN state and NL_Port 32 is in the OPENED state.

A Loop circuit has been established between the two NL_Ports. Since BB_Credit was zero (0), at least one R_RDY must be received before a frame may be transmitted by each NL_Port (i.e., normal ANSI X3, FC-PH-x Login protocol can now be used);

— NL_Port 26 and NL_Port 32 have previously (during L_Port initialization) determined that there is no Fabric. NL_Port 26 transmits an N_Port Login Sequence with D_ID of hex '000032' and S_ID of hex '000026'. Both NL_Ports recognize these as legitimate native address identifiers for a Loop without an FL_Port. The CFW again follows transmission of the Sequence.

The N_Port Login Sequence arrives at NL_Port 32 and is processed; an R_RDY is transmitted when the receive buffer becomes available;

— NL_Port 32 transmits an Accept response to NL_Port 26 honoring its requested native address identifier and confirming its own native address identifier.

NL_Port 26 receives the Accept Sequence and begins to close the Loop (REQ(close)) by transmitting CLS, followed by the CFW (note that an R_RDY was not required) (see 8.4.3, item 18 for details).

The LPSM for NL_Port 26 makes transition (35) to the XMITTED CLOSE state;

— The CLS is received by NL_Port 32.

The LPSM for NL_Port 32 makes transition (46);

— The LPSM for NL_Port 32, now in the RECEIVED CLOSE state, transmits CLS (REQ(close)) (see 8.4.3, item 19 for details).

The LPSM for NL_Port 32 makes transition (60);

— The LPSM for NL_Port 32, now in the MONITORING state, has completed its work for the Loop circuit. NL_Port 32 may begin arbitrating to carry out its own work;

— The LPSM for NL_Port 26, now in the XMITTED CLOSE state, monitors its inbound fibre for CLS (see 8.4.3, item 18 for details).

The CLS is received on its inbound fibre.

The LPSM for NL_Port 26 makes transition (50);

— The LPSM for NL_Port 26, now in the MONITORING state, has completed its work for the Loop circuit. If NL_Port 26 is a fair L_Port, it must now wait until an Idle is seen (i.e., ACCESS is TRUE(1)) before it can attempt to arbitrate again.

NOTE — NL_Port 26 could have used the TRANSFER state if it required further use of the Loop.

The complete list of state transitions and states in the order used are:

NL_Port 26
0: MONITORING, (01)
1: ARBITRATING, (12)
2: ARBITRATION WON, (23)
3: OPEN, (35)
5: XMITTED CLOSE, (50)
0: MONITORING

NL_Port 32
0: MONITORING, (04)
4: OPENED, (46)
6: RECEIVED CLOSE, (60)
0: MONITORING

Annex C
(informative)

Dynamic Half-Duplex

Although Fibre Channel is by nature a full-duplex link (i.e., Data frames may travel in both directions in the fibre pairs simultaneously), some L_Port implementations can only support one-directional data transfers. There are two types of L_Ports possible:

1) Full-duplex L_Ports are those that may simultaneously transmit and receive Data frames (this type of Data frame transfer is referred to *bi-directional transfer*).

2) Half-duplex L_Ports are those which can transmit or receive Data frames, but not at the same time (this type of Data frame transfer is referred to as *simplex transfer*).

This annex describes a method to minimize the number of arbitration cycles for a full-duplex L_Port by using the established Loop circuit more efficiently. The description is independent of which Class of Service is being used.

C.1 Close initiative description

Although, not required by this standard, the L_Port in the OPEN state normally transmits the first CLS to close the Loop. When a full-duplex Loop circuit exists (i.e., OPNyx was transmitted and the L_Port in the OPENED state receives CLS, it may continue to transmit frames until it has no more credit (i.e., Available_BB_Credit=0 or EE_Credit=0). Once the OPENED L_Port is no longer able to transmit any frames, it must forward CLS. This assumes that both L_Ports may have transferred Data frames in opposite directions when a full-duplex Loop circuit exists.

NOTE — An L_Port which has transmitted CLS is not allowed to transmit any frames or R_RDYs.

There are at least two cases where it may be useful to transfer the close initiative rather than transmitting CLS to allow the L_Port which holds the close initiative to transmit the first CLS.

1) Some implementation are not able to handle simultaneous transmit and receive Data frames at the node. Often, these nodes have Data frames pending for the OPEN L_Port, but because of the implementation, cannot take advantage of the bi-directional Loop circuit which exists.

2) Even if full-duplex data transfers are possible, if the OPEN L_Port transmits CLS, the OPENED L_Port can only transmit Data frames based on existing credit.

Both of these cases would require a re-arbitration to transmit the Data frames which an L_Port was unable to transmit..

To avoid this extra re-arbitration cycle, the DHD Primitive Signal is provided. Transmitting DHD allows the OPEN L_Port to continue to transmit R_RDYs and Link_Control frames (but no Data frames). The OPENED L_Port remembers that it has received DHD by setting DHD_RCV to TRUE(1). If DHD_RCV is TRUE(1), the OPENED L_Port holds the close initiative and is expected to transmit the first CLS when it has no more frames to transmit to the OPEN L_Port. The OPEN L_Port may transmit CLS at any time following transmission of DHD, although it would normally wait until it received the CLS from the OPENED L_Port.

C.2 Dynamic Half-Duplex examples

Table C.1 describes how two L_Ports (**A** in the OPEN state and **B** in the OPENED state) may make better use of a full-duplex Loop circuit by using DHD. Table C.1 shows both R_RDY and Link_Control frame flow control. If the Class of Service does not use one or the other, these would be absent from the table.

NOTE — Once the close initiative is transferred from one L_Port to the other via DHD, the OPEN L_Port is only allowed to transmit Link_Control frames (e.g., ACKs) and R_RDYs (i.e., Data frames may not be transmitted once DHD has been transmitted).

Table C.1 — Dynamic Half-Duplex

L_Port A (OPEN state)	Transmits	L_Port B (OPENED state)
Full-duplex L_Port (able to transmit and receive Data frames simultaneously) - Transmit n-1 Data frames.	OPNyx ==> R_RDYs ==> frame(s) ==>	**Full-duplex L_Port** (able to transmit and receive Data frames simultaneously)
	<== R_RDYs	- Transmit R_RDY and Link_Control frames (if any)
	<== frames	- Transmit Data frames
- Transmit last frame If Login DHD is FALSE(0), transmit CLS.	frame(n) ==> CLS ==>	- CLS received, continue transmitting frames until Available_BB_Credit=0 or EE_Credit=0).
Loop circuit is closed—next arbitrating L_Port wins Loop.	<== frame(s) <== CLS	- Transmit CLS (a new arbitration cycle is required to transmit remaining frames).
- If Login DHD is TRUE(1), transmit DHD.	DHD ==>	- DHD received, continue transmitting frame; set DHD_RCV to TRUE(1).
	<== frames	
Continue to transmit R_RDYs and Link_Control frames (if any)	R_RDYs ==> <==R_RDYs	- Transmit n-1 Data frames - Transmit R_RDYs and Link_Control frames (if any)
	<== frame(n)	
	<== CLS	- Transmit last frame
	CLS ==>	- Transmit CLS
Transmit CLS to close Loop circuit.		- Loop circuit is closed—next arbitrating L_Port wins Loop.

NOTE — Table C.1 shows frame transfers based on R_RDY flow control. For certain Classes of service (e.g., unbuffered Class 1), the R_RDYs would not be used (except on the first frame) and all flow control would be based on End-to-end-Credit.

FC-AL allows an L_Port in the OPEN state to use the TRANSFER state to make a connection to another L_Port without re-arbitrating. When DHD is transmitted by the OPEN L_Port, it normally would not transmit the first CLS. However, based on the access fairness algorithm (i.e., when to use the TRANSFER state), if ACCESS is TRUE(1), the L_Port in the OPEN state may still go to the TRANSFER state by transmitting CLS (assuming this is done before the L_Port received CLS).

Annex D
(informative)

Access unfairness

This annex describes how access unfairness might be used to improve Loop performance, and how access unfairness can be used to allow an NL_Port to reclaim ACK buffers when they become full.

D.1 Improving Loop performance

One possible use of the Loop is a serial version of a conventional single-Initiator parallel Small Computer System Interface (SCSI) bus. In this configuration, there is only one Initiator and many Target devices connected to the Loop. If all NL_Ports on the Loop, including the Initiator, follow the access fairness algorithm, then the Initiator may not be able to obtain sufficient Loop bandwidth to optimize overall performance by achieving a high level of parallelism among its Targets. A specific example is the situation where the Initiator transmits READ commands to all Targets. The Targets may take some time to locate the read data and then the Targets need to transmit the data to the Initiator.

Once a Target acquires the Loop and transmits its read data, it may be inactive unless it is given another command. If the Initiator follows the access fairness algorithm, it will wait for all Targets that have read data pending to access the Loop, before it can access the Loop and transmit a new command to the Target. To reduce the time that a Target is inactive, the Initiator may want to unfairly acquire the Loop and transmit a new command to the Target. The Target can then start locating the data for the new command.

D.2 Emptying ACK buffers

With a Loop topology, a low-cost NL_Port may need to buffer outbound ACKs in Class 2.

One example occurs if a simple First-In-First-Out (FIFO) ACK buffer is used. The ACKs destined for the currently connected NL_Port cannot be sent because they are queued behind ACKs for other NL_Ports in the ACK FIFO.

Another example occurs in a full-duplex Loop circuit. If NL_Port A transmits CLS to NL_Port B at the same time that NL_Port B transmits Data frames to NL_Port A, then NL_Port A must buffer ACKs for the Data frames that it receives after it has sent the CLS. This occurs because NL_Port A will receive Data frames after it has sent the CLS and made the transition to the XMITTED CLOSE state. NL_Port A cannot transmit frames (including ACKs) or R_RDYs in the XMITTED CLOSE state. Using DHD (instead of CLS), allows the L_Port in the OPEN state to continue to transmit ACKs for the received Data frames.

Alternatively, if the ACK buffer becomes full, NL_Port A may choose to unfairly arbitrate and acquire the Loop so that it can transmit the queued ACKs and reclaim its ACK buffer space. NL_Port A may use the TRANSFER state to speed up the process.

Annex E
(informative)

Half-duplex operation

This annex describes where half-duplex mode may be used to prevent ACK buffers from overflowing during Class 2 operation.

The operational characteristics of the Loop differ slightly from a point-to-point or from a Fabric topology. When an N_Port is directly connected to an F_Port, it can transmit an ACK frame whenever buffer-to-buffer credit is available. With the Loop, not only is Available_BB_Credit required, but the Loop must also have a circuit open with the correct L_Port before an ACK can be sent. At times, an NL_Port may need to buffer ACKs because it cannot access the Loop to transmit them.

Depending upon how ACK buffering is implemented, an NL_Port that is receiving Data frames may not be able to transmit the corresponding ACKs. If a simple FIFO is used to buffer ACKs, the ACKs for the currently connected NL_Port cannot be sent because they are queued behind ACKs for other NL_Ports in the ACK FIFO.

Another example where an NL_Port must buffer ACKs is in a full-duplex Loop circuit. If NL_Port A transmits CLS to NL_Port B at the same time that NL_Port B transmits Data frames to NL_Port A, then NL_Port A must buffer ACKs for the Data frames that it receives after it has sent the CLS. This occurs because NL_Port A will receive Data frames after it has sent the CLS and made the transition to the XMITTED CLOSE state. NL_Port A cannot transmit frames (including ACKs) or R_RDYs in the XMITTED CLOSE state.

When an N_Port is connected directly to an F_Port, if it experiences a resource shortage to buffer ACKs, it will not transmit R_RDYs to the F_Port. The F_Port cannot transmit any frames to the N_Port without BB_Credit and therefore the N_Port will not owe EE_Credit and will not have to buffer the ACKs. The N_Port may continue to transmit ACKs and once sufficient ACK buffering is available the N_Port will transmit R_RDY to enable the F_Port to transmit frames.

On a Loop, an NL_Port has two techniques that can be used to reduce or prevent the receipt of Data frames and therefore, the number of ACKs it must buffer. The first technique is similar to the Fabric example above, where the NL_Port withholds R_RDYs when a Loop circuit is opened. For example, an NL_Port can specify at Login that it has an BB_Credit of zero (0). When the NL_Port receives OPNy, it will not transmit any R_RDYs, preventing the opening NL_Port from transmitting any frames. The NL_Port will therefore not have to buffer ACKs.

A second technique, which an NL_Port can use to reduce or prevent the receipt of Data frames, is to establish a Loop circuit in half-duplex mode. When the opened NL_Port receives the OPNyy, it is not allowed to transmit Data frames, only Link_Control frames. This guarantees that the NL_Port that established the Loop circuit, will not receive Data frames and therefore will not be required to buffer ACKs.

Annex F
(informative)

BB_Credit and Available_BB_Credit management example

The following is an example implementation using BB_Credit and Available_BB_Credit which was described in 8.3.4. Assume two L_Ports, A and B, and that A arbitrated and won and plans to open B; full-duplex is used; and, both A and B have frames to transmit. L_Port A has sixteen receive buffers available and a BB_Credit Login value of two for L_Port B. L_Port B has eight receive buffers available and a BB_Credit Login value of one for L_Port A.

L_Port A:

1) looks up the opened BB_Credit Login value for B (two);

2) checks to see how many receive buffers it has available (sixteen);

3) transmits OPN(B,A), CFW, CFW, R_RDY, CFW, CFW, R_RDY, CFW, CFW, R_RDY, and two Fill Words, followed by two frames (opened BB_Credit Login value for B). Note that had the opened BB_Credit been zero (0), no frames could have been sent by A. The remaining thirteen R_RDYS are transmitted after the first frame and subsequent frames;

4) receives and counts the R_RDYs sent by B (eight). Once L_Port A has received and discarded the two R_RDYs (which are for the frames already shipped against the opened BB_Credit in 3 above), L_Port A has an Available_BB_Credit of six that it may use to transmit up to six additional frames;

5) transmits one frame for each Available_BB_Credit;

6) receives R_RDYs sent by B and increments Available_BB_Credit;

7) receives the number of frames sent by B into the receive buffer(s);

8) transmits one R_RDY for each receive buffer that has been made available

9) repeats steps 5 through 8 until all frames have been sent;

10) transmits CLS;

11) continues to receive frames from B, but transmits no R_RDYs or frames; and,

12) receives CLS and closes its end of the Loop.

L_Port B:

1) receives OPN(B,A) and opens the Loop;

2) looks up the open BB_Credit for A (one);

3) checks to see how many receive buffers it has available (eight);

4) transmits CFW, CFW, R_RDY, CFW, CFW, R_RDY, CFW, CFW, R_RDY and two Fill Words, followed by one frame (open BB_Credit Login value for A). Note that had the open BB_Credit been zero(0), no frames could have been sent by B until the first R_RDY was received from A. The remaining five R_RDYs are transmitted after the first frame. Once L_Port B has received and discarded one R_RDY (which is for the frame already shipped against the open BB_Credit) and counted the other R_RDYs from A, L_Port B has an Available_BB_Credit of fifteen that it may use to transmit up to fifteen additional frames;

5) receives and counts the R_RDYs sent by A by increasing Available_BB_Credit by one for each R_RDY. L_Port B can now transmit an additional frame for each R_RDY that it has received;

6) receives the number of frames sent by A into the receive buffer(s);

7) transmits an R_RDY for each receive buffer that has been made available;

8) repeats steps 5 through 7 until CLS is received from A; and,

9) may continue to transmit frames until Available_BB_Credit or EE_Credit is exhausted, followed by CLS. When the CLS is sent, L_Port B closes its end of the Loop.

There are several variations on the previous example.

1) Data frame transfer is only from A to B even though OPN(B,A) (full-duplex) is used. L_Port A transmits three R_RDYs followed by the number of frames represented by the opened BB_Credit Login value. In this case, B has no Data frames to transmit, but transmits one R_RDY for each available receive buffer and Link_Control frames (e.g., ACKs) to A. Note that B may limit the number of frames that A can transmit by transmitting one R_RDY for each available receive buffer, followed by CLS. Once L_Port A has received the CLS, it can then only transmit frames until its Available_BB_Credit is exhausted before it must close its end of the Loop;

2) Both open and opened BB_Credit is zero (0). In this case, neither A nor B can transmit any frames until at least one R_RDY is received. For each R_RDY received, a frame may be sent. Note that when BB_Credit is zero (0), a Loop turn-around delay is required at A before transmitting the first (and possibly the only) frame. By guaranteeing a minimum number of receive buffers (as indicated by BB_Credit), this turn-around delay may be eliminated;

3) L_Port A transmits OPN(B,A) (full-duplex), followed by at least one R_RDY, followed by two frames (the opened BB_Credit Login value for L_Port B). L_Port B does not look up the open BB_Credit for A, and transmits at least one R_RDY (one for each available receive buffer). L_Port B waits for the first R_RDY from A. When at least one R_RDY is received (i.e., Available_BB_Credit is one (1)), B can transmit one frame;

4) L_Port A transmits OPN(B,B) (half-duplex). In this case, L_Port B cannot identify A and must wait for the first frame from A to transmit any frames (note: since this is a half-duplex OPNyy, no Data frames may be transmitted by B). Once a frame is received, the S_ID in the frame header may be used to determine the AL_PA of A. B can then establish the open BB_Credit for A. If B is using a minimum Loop value for the open BB_Credit, the AL_PA of A is not required; and,

5) L_Port A transmits OPN(B,B) (half-duplex). In this case, L_Port B must wait for the first R_RDY from A. Once at least one R_RDY is received (i.e., Available_BB_Credit is one (1)), B can transmit one Link_ Control frame to A.

Annex G
(informative)

L_Port clock design options

This annex describes two approaches to clock implementations for L_Port design.

G.1 L_Port synchronous clock design

When the L_Port uses the receive clock for transmission, the design is synchronous. A buffer is not required between the receiver and the transmitter. An example is shown in figure G1. This design approach is not recommended for the L_Port design operating at FC-PH specified frequencies. Some of the jitter properties of the received clock are transferred to the output even when reconditioning or filtering is used. This transfer of jitter makes the number of L_Ports that may be connected in a Loop undetermined.

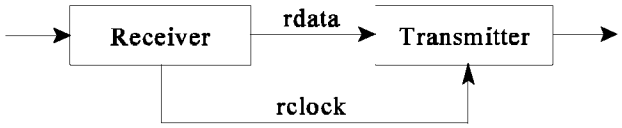

Figure G.1 — Example of a synchronous L_Port design

G.2 L_Port asynchronous clock design

When the L_Port uses a local reference clock for transmission, the design is asynchronous. An elasticity buffer is required between the receive logic and the transmitter. This buffer is necessary because of the clock frequency difference between the receiver and transmitter. The receiver is recovering its clock from the input data stream. The transmitter clock is generated from an oscillator at the L_Port. This buffer is also required by an L_Port if the receive data is resynchronized to a local clock such as the transmit clock.

The elasticity buffer expands and contracts to control the over-run and under-run conditions resulting from the clock frequency difference. The buffer control directs the insertion and removal of Fill Words outside FC-2 frames to prevent over-run and under-run conditions from occurring in FC-2 frames. This control is called *clock skew management*.

An example of an asynchronous design is shown in figure G.2. This design approach is recommended for L_Ports. The use of a stable local clock for transmission provides isolation from the receive clock jitter.

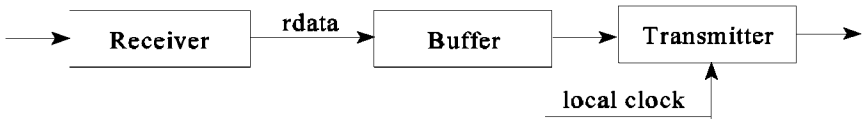

Figure G.2 — Example of an asynchronous L_Port design

G.2 Clock skew management function periodicity

The period between clock skew management operations is dependent on the possible difference between the receive and transmit clocks. ANSI X3, FC-PH-x specifies the allowed clock deviations of + or - 100 ppm (parts per million), independent of the frequency.

Assuming a worst case frequency mismatch between two connected Ports (i.e., 200 ppm), the maximum duration for a frame is based on the total frame length of 2156 characters (i.e., 2112 (data field) + 24 (FC-2 header) + 12 (SOF, EOF, and CRC) + 8 (two Fill Words)). The net elasticity needed for a maximum size frame is:

2156 Characters * 10 bits/char * 200 ppm = 4,3 bits per frame.

This means that before a frame is transmitted, the elasticity buffer must have at least 4,3 bits of data and free space to ensure against buffer under-run and over-run.

Since L_Ports are required to maintain word synchronization and clock skew management is only valid at Transmission Word boundaries, a clock skew adjustment may be required every:

40 bits / 4,3 bits per frame = 9,3 frames.

Annex H
(informative)

Mark Synchronization examples

This annex describes two examples of how the Mark (MRKtx) Primitive Signal may be used on a Loop. Since some states do not retransmit MRKtx, the only way to guarantee that the originator receives the transmitted MRKtx is for the originator to be in the OPEN state and for all other L_Ports to be in the MONITORING or ARBITRATING states.

H.1 Clock synchronization

When the type of mark (MK_TP) is clock synchronization (e.g., hex '00'), the Mark Primitive Signal may be used to synchronize clocks between a number of processors. Through configuration or implementation, one processor is assigned the task of providing a Master clock. It is the responsibility of this processor to generate enough MRKtx Primitive Signals to keep the other processors within a prespecified clock tolerance.

The processor with the Master clock transmits one MRKtx (with t = hex '00' and x = its AL_PA) instead of a Fill Word for each REQ(mark as tx). Each recipient of the MRKtx checks the AL_PA to synchronize on the correct Master clock (since the AL_PA is used to identify the originator, this allows multiple Master clock originators). If the AL_PA matches the one being used for synchronization, the receiving processor adjusts (if necessary) its clock and retransmits the MRKtx. If the MRKtx is returned to the originator, the originator replaces it with the CFW.

Because the MRKtx may not be inserted onto the Loop except during normal Fill Word transmission, it is possible that a MRKtx may not be originated or retransmitted. If the processors can accept a missing MRKtx, the MRKtx provides a low-cost method (does not require a separate clock synchronization interface) for keeping clocks synchronized. To obtain an initial clock value, the ANSI X3, FC-PH-x defined Time Server at well-known address hex 'FFFFFB' may be used. Once every processor has this initial clock value, the MRKtx may be used to maintain clock synchronization. The Time Server function may be provided by an F/NL_Port in the absence of a Fabric.

If the originator of the MRKtx receives the MRKtx, it may calculate the latency for the MRKtx to traverse the Loop. If this latency is greater than the clock synchronization tolerance, a system administrator may be informed that the MRKtx is unpredictable in the configured environment.

H.2 Disk spindle synchronization

When the type of mark (MK_TP) is disk spindle synchronization (e.g., hex '01'), the Mark Primitive Signal may be used to synchronize disk spindles between a number of disk drives. Through configuration or implementation, one disk drive is assigned the task of providing a Master clock. It is the responsibility of this disk drive to generate enough MRKtx Primitive Signals to keep the other disk drives within a prespecified spindle synchronization tolerance.

The disk drive with the Master clock transmits one MRKtx (with t = hex '01' and x = its AL_PA) instead of a Fill Word for each REQ(mark as tx). The recipient of the MRKtx checks the AL_PA to synchronize on the correct Master disk spindle (since the AL_PA is used to identify the originator, this allows multiple Master spindle synchronization originators). If the AL_PA matches the one being used for synchronization, the receiving disk drive adjusts (if necessary) its spindle motor and retransmits the MRKtx. If the MRKtx is returned to the originator, the originator replaces it with the CFW.

Because the MRKtx may not be inserted onto the Loop except during normal Fill Word transmission, it is possible that a MRKtx may not be originated or retransmitted. If the disk drives can accept a missing MRKtx, the MRKtx provides a low-cost method (does not require a separate spindle synchronization interface) for keeping disk spindles synchronized.

If the originator of the MRKtx receives the MRKtx, it may calculate the latency for the MRKtx to traverse the Loop. If this latency is greater than the disk spindle synchronization tolerance, a system administer may be informed that the MRKtx is unpredictable in the configured environment.

Annex I
(informative)

Port Bypass Circuit example and usage

This annex describes a Port Bypass Circuit which may be used to keep a Loop operating when an L_Port location is physically removed or not populated; L_Ports are powered-off; or, a failing L_Port is present. A Port Bypass Circuit provides the means to route the serial channel signal past an L_Port. Also described are the L_Port Bypass/Enable (LPB or LPE) Primitive Sequences. The main purpose of these Primitive Sequences is to physically control the Port Bypass Circuit and logically control the L_Port (i.e., the LPSM is forced into and held in the MONITORING state). LPB and LPE are Primitive Sequences which are usually transmitted for 2xAL_TIMEs or until the transmitted Primitive Sequence is received.

I.1 Port Bypass Circuit

Figure I.1 shows an example Port Bypass Circuit. The input from the previous L_Port, *n*-1, feeds a multiplexer (MUX) and the local L_Port. The other input to the multiplexer is from the local L_Port. A select signal determines whether the input from L_Port *n*-1 or the input from the local L_Port is transmitted to the next L_Port, *n*+1. The Port Bypass Circuit is an asynchronous switch (i.e., when it switches, it may cause a loss of synchronization at the next L_Port). To avoid unnecessary reinitializations and error counters to overflow, L_Ports and Port Bypass Circuits should be used which avoid counting this loss of synchronization as an error and going to the LOOP-FAIL-INITIALIZE state.

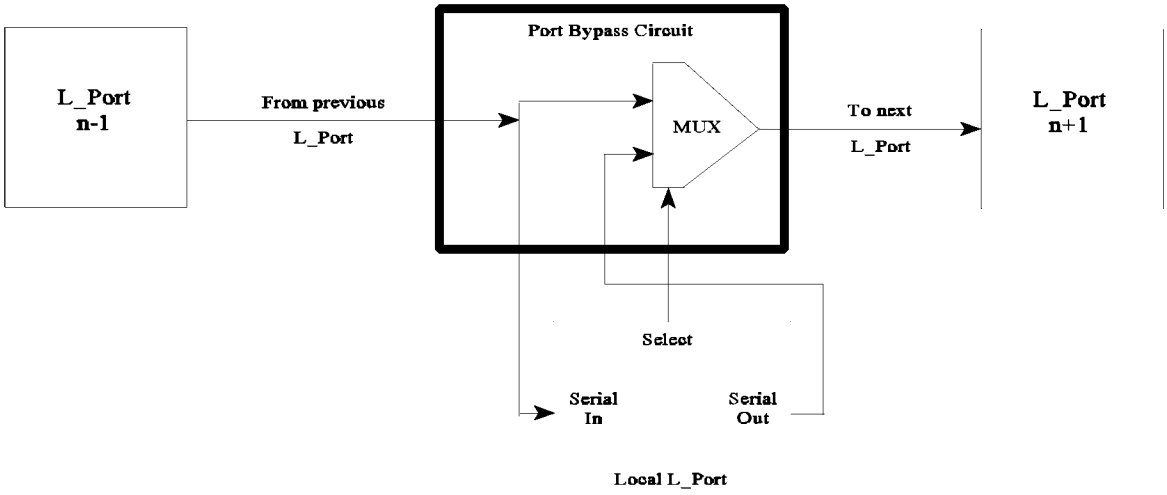

Figure I.1 — Example Port Bypass Circuit

I.1.1 Default bypass

The Port Bypass Circuit for an unpopulated location or a powered-off L_Port defaults to the Port Bypass Circuit being set (i.e., the input from *n*-1 passes through the multiplexer to *n*+1).

I.1.2 Power-on reset bypass

At power-on, an L_Port leaves the Port Bypass Circuit set and enters the MONITORING state with PARTICIPATE set to FALSE(0) (i.e., in Non-Participating mode). This allows the Loop to continue to function while the L_Port performs a self-test. When the L_Port is ready to enter the Participating mode, the L_Port deactivates its Port Bypass Circuit and enters the NORMAL-INITIALIZE state.

I.2 Using a Port Bypass Circuit

Any L_Port may accept the role of Loop manager to execute diagnostics and to recover a failing Loop. The selection criteria of a Loop manager should include the ability to report failures to an operator or system log. A "Loop manager" in the context of this discussion is an L_Port that has the ability to diagnose a Loop (i.e., uses LPB and LPE to bypass and enable other L_Ports).

I.2.1 Diagnostic Test of the Port Bypass Circuit

Loop Initialization must be completed before the Port Bypass Circuit may be tested. L_Ports must be in the Participating mode (i.e., have an AL_PA) for the test to be effective.

The Loop manager arbitrates for the Loop; transmits OPNyy (where y is the AL_PA of the Loop manager); and, transmits the L_Port Bypass Primitive Sequence (LPByx, where y = AL_PA of the L_Port under test and x is its AL_PA) until the Primitive Sequence is received. This allows any receiver, affected by an L_Port switching out of the Loop, to synchronize to the new input. The Loop manager removes all LPBs that it originated.

In order to verify that the Port Bypass Circuit is present and operating, the Loop manager may: transmit CLS and enter the TRANSFER state; when CLS is received, transmit OPNyx (where y = AL_PA of the L_Port under test). If the OPNyx is received by the Loop manager, the Port Bypass Circuit is functioning normally.

The Loop manager completes the test by transmitting the L_Port Enable Primitive Sequence (LPEyx where y is the AL_PA of the bypassed L_Port and x is its AL_PA) until the Primitive Sequence is received. The Loop manager removes all LPEs that it originated. In order to verify that the L_Port is no longer bypassed the Loop manager may: transmit CLS and enter the TRANSFER state; when CLS is received, transmit OPNyx (where y = AL_PA of the L_Port under test). If the OPNyx is not received by the Loop manager within AL_TIME, the Port Bypass Circuit has been deactivate and is functioning normally. The Loop manager may then transmit CLS and wait for the CLS to be returned. If other Port Bypass Circuits are to be tested, the Loop manager may use the TRANSFER state until all AL_PAs have been tested.

If at any time during the above test, the Loop manager recognizes a LIP, the AL_PA of the bypassed L_Port has been relinquished. The Loop manager then can only use LPEfx to enable a previously bypassed L_Port.

I.2.2 Recovery from Loop Failure

If an L_Port detects a Loop Failure, it may use the following recovery procedure. Waiting for R_T_TOV, allows recovery from transient conditions.

After the Loop Failure is detected, the L_Port may perform an optional Loop-back self test. If the Loop-back test fails, the L_Port may request to be bypassed (REQ(bypass L_Port)) to allow the Loop to recover. If the Loop-back test passes, the L_Port requests to go to the LOOP-FAIL-INITIALIZE state where it transmits a LIP(F8) (see 7.8.2 or 7.8.4) for up to 2xAL_TIMEs or until LIP is recognized.

If LIP is recognized, the L_Port goes to the OPEN-INIT-START state and transmits at least twelve of the received LIPs (see 7.8.1 or 7.8.3).

If LIP is not recognized within 2xAL_TIMEs, the L_Port transmits LPByx to bypass the failing L_Port (identified by the y value). The Loop manager may use the AL_PA position map from the most recent Loop Initialization to identify the failing L_Port (it is the L_Port adjacent to the L_Port that detects the Loop Failure). If this AL_PA position map is not available, the Loop manager may attempt all valid AL_PAs (excluding its own), bypassing all L_Ports until the failing L_Port is identified. The Loop manager may also transmit LPBfx to bypass all L_Ports (even those that do not have a valid AL_PA). The Loop manager transmits each LPByx for up to 2xAL_TIMEs or until the Primitive Sequence is recognized to allow any receiver affected by a Port switching out of the Loop to synchronize to the new input. Once the failing L_Port has been bypassed, if any other L_Ports had been bypassed, they should be reenabled with LPEyx. If the Loop Failure occurs during a time when the failing L_Port does not have a valid AL_PA, manual intervention may be required to find the failing L_Port or LPBfx may be used to bypass all L_Ports and if successful, one L_Port at a time may be reenabled with LPEyx until the failing L_Port can be identified.

I.2.3 Power-on with a failing L_Port

An L_Port that is unable to pass its self-test does not deactivate its Port Bypass Circuit. If an L_Port has an AL_PA which it previously saved in non-volatile storage, it follows the same procedure as if the Loop failed after being operational (see I.2.2).

In a Loop without valid AL_PAs, recovery of the Loop may require manual intervention (e.g., physically removing one L_Port after another until the failing L_Port is identified), unless the failing L_Port can initiate a bypass (REQ(bypass L_Port)).

I.2.4 Reconfiguring a Loop with LPB and LPE

A Loop manager may elect to use the Port Bypass Circuit to physically switch Participating L_Ports in and out of the Loop. When an L_Port is enabled on the Loop, the Loop manager should begin Loop Initialization to ensure that there are no AL_PA conflicts.

Annex J
(informative)

Public L_Ports and Private NL_Ports on a Loop

This annex describes how Public L_Ports and Private NL_Ports may be used on a Public Loop. Figure J.1 shows an example of such a configuration. Two advantages of connecting L_Ports in this fashion are: security (Private NL_Ports may not be addressed by Ports not on the Loop) and the Private NL_Ports may be lower cost.

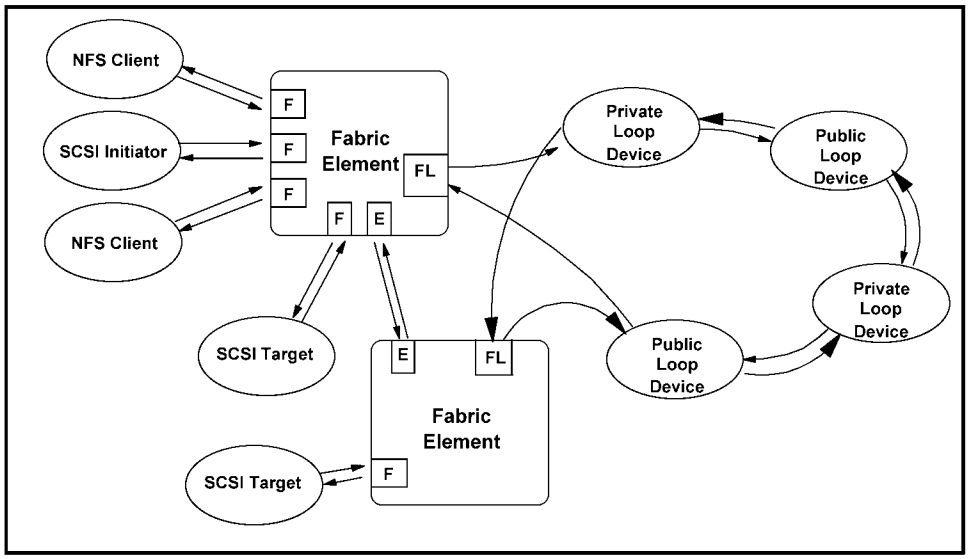

Figure J.1 — Public L_Ports and Private NL_Ports on a Loop

In this example, an NFS (Network File System) client on the left transmits an NFS command through the Fabric to the NFS server (the NFS client only knows that the NFS server has access to the requested data, but does not know the location of the data). The NFS server is also a SCSI initiator and it knows which SCSI target has the data. A SCSI command is sent by the NFS server (SCSI initiator) to any of the SCSI targets on the Loop. When the SCSI target has the requested data, it transmits the data to the SCSI initiator, which in turn transmits it via an NFS response to the NFS client.

As shown in this example, Private NL_Ports (SCSI targets) do not communicate with any Port not on the Loop, including the FL_Port. However, the Public NL_Port (SCSI target) may be addressed by both the NFS server (SCSI initiator) and the SCSI initiator on the other side of the Fabric.

Annex K
(informative)

Assigned Loop Identifier

This annex shows in table K.1 how a 7-bit Loop Identifier (e.g., a switch) may be used to represent the Hard Assigned AL_PA as used in clause 10.5. If there are no conflicts or an attached Fabric does not reassign the AL_PA, the value represented by this Assigned Loop Identifier will be the AL_PA of the L_Port. (See also table 15.)

Table K.1 — Assigned Loop Identifier

AL_PA (hex)	Switch (hex)	Setting (dec)	AL_PA (hex)	Switch (hex)	Setting (dec)	AL_PA (hex)	Switch (hex)	Setting (dec)
EF	00	0	A3	2B	43	4D	56	86
E8	01	1	9F	2C	44	4C	57	87
E4	02	2	9E	2D	45	4B	58	88
E2	03	3	9D	2E	46	4A	59	89
E1	04	4	9B	2F	47	49	5A	90
E0	05	5	98	30	48	47	5B	91
DC	06	6	97	31	49	46	5C	92
DA	07	7	90	32	50	45	5D	93
D9	08	8	8F	33	51	43	5E	94
D6	09	9	88	34	52	3C	5F	95
D5	0A	10	84	35	53	3A	60	96
D4	0B	11	82	36	54	39	61	97
D3	0C	12	81	37	55	36	62	98
D2	0D	13	80	38	56	35	63	99
D1	0E	14	7C	39	57	34	64	100
CE	0F	15	7A	3A	58	33	65	101
CD	10	16	79	3B	59	32	66	102
CC	11	17	76	3C	60	31	67	103
CB	12	18	75	3D	61	2E	68	104
CA	13	19	74	3E	62	2D	69	105
C9	14	20	73	3F	63	2C	6A	106
C7	15	21	72	40	64	2B	6B	107
C6	16	22	71	41	65	2A	6C	108
C5	17	23	6E	42	66	29	6D	109
C3	18	24	6D	43	67	27	6E	110
BC	19	25	6C	44	68	26	6F	111
BA	1A	26	6B	45	69	25	70	112
B9	1B	27	6A	46	70	23	71	113
B6	1C	28	69	47	71	1F	72	114
B5	1D	29	67	48	72	1E	73	115
B4	1E	30	66	49	73	1D	74	116
B3	1F	31	65	4A	74	1B	75	117
B2	20	32	63	4B	75	18	76	118
B1	21	33	5C	4C	76	17	77	119
AE	22	34	5A	4D	77	10	78	120
AD	23	35	59	4E	78	0F	79	121
AC	24	36	56	4F	79	08	7A	122
AB	25	37	55	50	80	04	7B	123
AA	26	38	54	51	81	02	7C	124
A9	27	39	53	52	82	01	7D	125
A7	28	40	52	53	83			
A6	29	41	51	54	84	00	7E	126
A5	2A	42	4E	55	85	--	7F	127

NOTE — The values are intentionally from lowest to highest priority. AL_PA = 00 is reserved for an FL_Port; '--' is not available.

Annex L
(informative)

Selective replicate for parallel query acceleration

This annex describes a relational database example that benefits from the selective replicate capability of FC-AL. Because queries are ad hoc, it is not known in advance which Ports will take part in a given multicast group, and the group can change with each query. This annex illustrates how OPNyr is preferable to using traditional multicast groups, which must be set up in advance. OPNyr significantly speeds up the execution of parallel queries.

The examples show a sort-merge join which is a technique employed by several database vendors. An example of a hypothetical parallel query engine is provided in clause L.2 and a specific example of a query is provided in clause L.3.

L.1 Parallel query technology

Parallel query is a technique for significantly reducing the response time of complex queries against very large databases. The basic technique divides the query among multiple CPUs, where each CPU applies the query against a partitioned, disjoint subset of the database tables selected in the query. Most parallel query algorithms focus on joins, where row pieces from one or more tables are combined on some matching attribute.

L.2 Shared disk cluster

A cluster composed of multiple hosts accessing a large shared disk pool is illustrated in figure L.1. There are *n* database servers accessing a shared database striped across all drives that are attached via one or more Loops. Every server has direct access to any partition of the database on the shared disk pool. The Loops serve a dual role in the cluster. First, they function as a high speed disk channel for SCSI traffic between servers and disk. Second, they function as a high bandwidth, low-latency port-to-port interconnect for IP (Internet Protocol) traffic between servers. As this example shows, many database blocks are passed directly between servers during the parallel query.

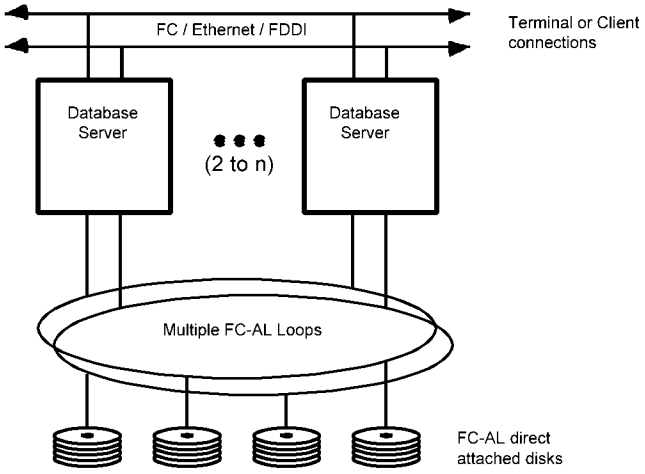

Figure L.1 — FC-AL parallel query server

L.3 Parallel query example

To illustrate a complex query typical of a direct marketing application often found in DSS (Decision Support Systems) or Data Warehousing, the following query is used as an example of how a parallel query could be executed to take advantage of selective replicate on the above configuration:

```
1       Select customer, address, num_purchases
2           From R, S
3           Where R.a = S.b
4           and num_purchases > 10
5           and (area code = 415
6             or area code = 408
7             or area code = 510)
```

This is a join of relations (database tables) R and S that satisfy the matching attribute in line 3. The matching attribute (R.a and S.b) is the customer ID. The query is intended to find all customers in the San Francisco Bay Area that have made a large number of purchases (more than 10). Relation S is the total worldwide customer population. Relation R is all purchases in California for the past three months. Relation S is many times larger than relation R, since R will only contain the subset of customers who have made purchases in the San Francisco Bay Area over the last three months. Tuples (records) from relation R are qualified by line 4, while tuples from relation S are qualified by lines 5 to 7. Applying qualifiers to a relation are known as a *projection* in database language. A projection reduces the number of tuple candidates that must be examined to determine if they match the join criteria in line 3.

Relations R and S are striped across all disk drives, D, in the cluster. Assume that there are *m* hosts available to participate in the query, where *m* <= *n*.

When the query is submitted to the cluster, a processor (e.g., B) is selected as the query coordinator. The query coordinator determines how the query will be executed and elects the other processors to participate in the parallel query. The decision as to which processor and how many processors will participate is made at run time. It is based on such factors as the load at each individual processors, the type of query being submitted, and the privilege of the user. For example, Tony CEO may be allowed to use all processors while Joe Clerk may only use 4 processors. The *m* participating processors form a multicast group that is dynamically created when the query is parsed. The query coordinator determines that relation R is too small to parallelize, since the overhead of parallelism will outweigh any speedup.

The query plan generated by the coordinator is illustrated by the following pseudo-code:

```
        CPU B (query coordinator)
            notify all participants 1 to m-1
1           multicast query plan            // includes split table
            scan R                          // read all tuples in R
            apply predicate to R -> R'
            sort R' -> R"                   // sort on joining attribute R.a
2           multicast R"
            wait for "scan S done" messages 1 to m-1
3           multicast "do join" messages" 1 to m-1
            wait for "join done" messages 1 to m-1
            done!
        CPU 1 to m-1 (query slaves)
            receive query plan
            scan S / (m-1) -> S'
                        // partitioned on tuple ID into m-1 buckets
            for each tuple
                    apply predicate
                    hash lookup in split table -> I
                            // hashed on phone # into m-1 buckets
                    if I <> me
                            transmit to CPU I
```

```
transmit "scan S done" message to B
when "do join" message received
        sort S'   -> S"     // sort on joining attribute S.a
while not at end of S"
        for each tuple in R"
                lookup R".a in S"
                        if match write to join result
transmit "join done" message to B
done!
```

The selective FC-AL Primitive Signal, OPNyr, is used by the query coordinator in lines 1 to 3 above. In line 1, the query plan must be sent to all participants. It tells each CPU which partition of relation S it should scan, and the hash function and split table values to use for each bucket. Line 2 is used to transmit the sorted relation R" to all participants. Line 3 is used to synchronize the completion of the scan phase with the start of the join phase.

This example demonstrates the utility of the selective replicate Primitive Signal for different uses. It can be employed to synchronize multiple CPUs with control messages (1 and 3). More importantly, it can be used to replicate large blocks of data between coordinating CPUs (2). In addition, the low overhead of forming constantly changing multicast groups allows efficient schedulers to determine the appropriate level of parallelism for each task at run time.

Annex M
(informative)

Controlled FC-AL configurations

This annex describes FC-AL implementations that require control of address assignments, the ability to control when and if Loop Initialization occurs, and uninterrupted processing while L_Ports are being inserted and removed from a Loop. One example of such an implementation is a disk storage subsystem. The degree of configuration control varies from implementation to implementation. These implementations are allowed by this standard, but may not interoperate with other L_Ports. The implementor of controlled configurations is responsible for functionality.

M.1 Address Control

FC-AL devices may have an interface connector for attachment to backplanes in storage cabinets. This connector provides 7 pins for delivering an address to the device. Annex K provides a mapping of these addresses: assigned Loop identifiers, to AL_PAs. The AL_PAs selected by these addresses are defined as Hard Addresses. The degree of address control required may vary between implementations.

M.1.1 Preferred Hard Addressing

In some implementations, the Hard Address may simply be a desired starting point for AL_PA determination using the Loop Initialization procedure. If there are no conflicts for the Hard Address the device obtains this AL_PA and participates. If there is a conflict for the Hard Address, the device may accept a Soft Address and participates. The host system must be capable in these implementations to dynamically determine the identity of the device at each AL_PA, since AL_PAs of the device may change with configuration changes.

M.1.2 Required Hard Addressing

In some implementations, the use of the Hard Address may be required. These implementations want the device at a Hard Address for device identification. Identification may be for maintenance or to insure devices are located in proper cooling and power distribution zones (e.g., in RAID configurations).

The cabinet may have device numbers affixed at device locations or slots. With the address fixed to a location, the device at a given AL_PA may be easily identified for removal and replacement. If the device is unable to obtain its Hard Address during Loop Initialization, the device is directed to become non-participating (i.e., it does not select a Soft Address). The device may attempt to obtain its Hard Address during the next Loop Initialization.

The direction to require hard addressing is outside the scope of this standard.

M.2 Configuration Change Control

FC-AL provides dynamic configuration changes with in-band control of Port Bypass Circuits and Loop Initialization to verify or obtain AL_PAs. Enabling an L_Port onto a Loop and Loop Initialization may be disruptive.

M.2.1 Port Bypass Circuit Control

The switching of a Port Bypass Circuit causes the receiver of the next L_Port in the Loop to resynchronize to a new serial input. During this process, FC-2 transfers or control information may be lost and error recovery may be required.

Implementations may control devices when to enable their L_Ports into the Loop. This method may use the Loop Port Enable (LPE) Primitive Sequence or provide control of the Port Bypass Circuits external to the L_Port. A controlling L_Port may win arbitration, open itself, and then switch the Port Bypass Circuits to not disrupt normal Loop transfers.

If the LPE is used, the AL_PA in the LPE may be the Hard Address, if required hard addressing is used in the configuration, or the enable all, hex 'FF'. Multiple L_Ports may be enabled if there are duplicate AL_PAs or enable all is used.

The direction to not enable the L_Port until LPE is recognized is outside the scope of this standard.

M.2.2 Loop Initialization Control

An L_Port requesting initialization may also disrupt Transmission Words between other L_Ports and cause information loss and error recovery. L_Ports are not required to request initialization if address determination and verification is by a method outside the scope of the standard.

Devices may be configured to not request initialization when their L_Ports are enabled into the Loop. The host may periodically, or at operator intervention, begin Loop Initialization. Loop Initialization is not required if the implementor disables the use of Soft Addresses and ensures there are no Hard Address conflicts.

Annex N
(Informative)

Insertion modes of Hubs

There are several different ways that Hubs insert Loop Segments into a Loop. A Loop Segment is a portion of a Loop which includes one or more L_Ports. The following are four examples and are not considered to be an exhaustive list.

1. Hubs may insert a Loop Segment when periodic K28.5 characters are received for a minimum time period. If an L_Port transmits Idle, LIP, or any other Ordered Set containing a K28.5 character during power-up, this type of Hub would switch the Loop Segment into the Loop before the L_Port(s) were ready to participate. This would make the Loop non-functional while the L_Port(s) are becoming ready.

2. Hubs may insert a Loop Segment when any valid Transmission Word is received. If an L_Port transmits Idle, LIP, any other Ordered Set containing a K28.5 character, or any other valid 8B/10B encoded character during power-up, this type of Hub would switch the Loop Segment into the Loop before the L_Port(s) were ready to participate. This would make the Loop non-functional while the L_Port(s) are becoming ready.

3. "Smart" Hubs may use more sophisticated methods to determine when to insert a Loop Segment. Since this behavior is Hub dependent it should not be used to determine the power-up operation of an L_Port.

4. Hubs may also wait for a management function to insert a Loop Segment into the Loop.

Additionally, while these are some ways that Hubs may operate for the purpose of inserting a Loop Segment into a Loop, it is not possible for an L_Port to determine which type of Hub it is attached to.

Annex O
(Informative)

L_Port power-on considerations

When looking at the power-on condition of an L_Port, there are three different cases which must be considered:

1. An L_Port connected directly in a Loop;
2. An L_Port connected in a cabinet (e.g., Just-a-Bunch-of-Disks (JBOD) or disk array); and,
3. An L_Port connected to a Hub.

An NL_Port should power on with its transmitter disabled until it is able to either begin the INITIALIZATION process or enter the MONITORING state in Non-Participating mode.

In Case 1, the Loop will not be operational until the L_Port turns on its transmitter, and enters either the NORMAL-INITIALIZE state or the MONITORING state. However, the Loop was not operational before this L_Port powered up. Waiting some additional time to get the Loop operational should not present a problem in this case.

In Case 2, there is a Port Bypass Circuit that is controlled externally or by the L_Port, which will keep this L_Port off the Loop until some external event indicates that it should be inserted into the Loop. If the Port Bypass Circuit is enabled before the L_Port powers up, and remains enabled until the L_Port turns on its transmitter and enters the NORMAL-INITIALIZE state or MONITORING state, there will be no disruption to the Loop.

In Case 3, all Hubs should be capable of bypassing an L_Port when its transmitter is off; since this is equivalent to either no L_Port connected, or an L_Port connected with its power turned off. By having the L_Port maintain its transmitter in the off condition until the L_Port is ready to enter the NORMAL-INITIALIZE or MONITORING state, a Hub can keep the L_Port bypassed until the L_Port is capable of participating in the Loop.

Annex P
(Informative)

L_Port initialization flow diagram

Figure P.1 provides a high-level flowchart-like view of the Loop Initialization procedure. The processes are represented as rectangles. The flows are represented as directed lines. The text in the diagram is brief and highly abbreviated. The major steps for clause 10 are identified in the upper right-hand corner of selected process blocks.

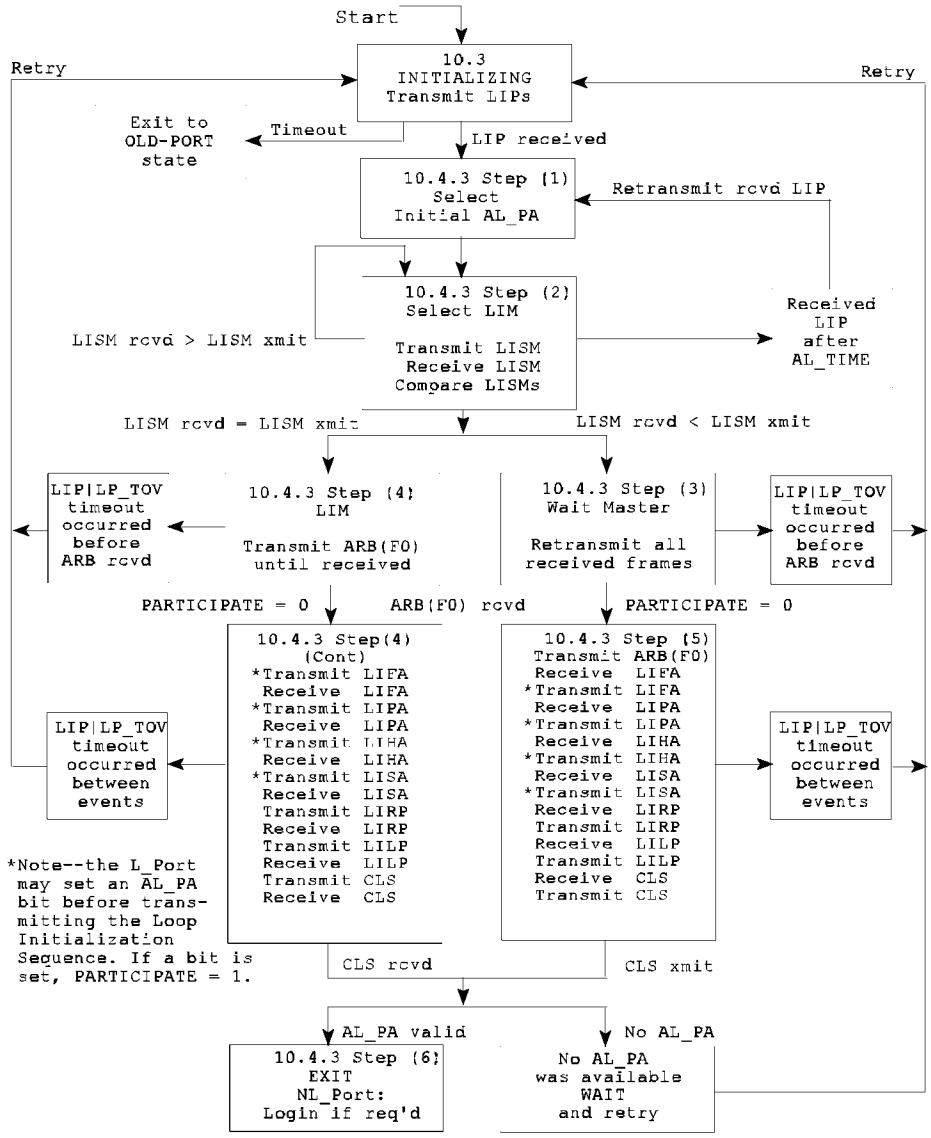

Figure P.1 — L_Port initialization flow diagram

Annex Q
(informative)

Examples of Switch Port Initialization

This annex presents some example scenarios that may occur during Switch Port Initialization. This aids in the understanding of how to use the OLD-PORT state diagram to achieve two operational switches when connected together.

NOTE — This annex is a modification and a clarification of an annex published in FC-SW.

Q.1 Example 1: two E/F/FL_Port-capable Switch Ports

In this example, two Switch Ports that are E/F/FL_Port-capable (i.e., this Port may be used as an E_Port, an F_Port, or an FL_Port) are attached to each other. Figure Q.1 illustrates this example.

Figure Q.1 — Switch Initialization example 1

According to the initialization algorithm, since both Switch Ports are E/F/FL_Port-capable, they start the process with Loop Initialization. LIPs are sent and recognized, and each Switch Port starts sending LISM frames. When Switch Port X receives LISM from Switch Port Y, it sees that its Port_Name is lower than the Port_Name in the Payload, and continues sending the same LISM.

On the other hand, when Switch Port Y receives LISM from Switch Port X, it sees that its Port_Name is higher than the Port_Name in the Payload. This causes Switch Port Y to start sending the LISM it received, with the Port_Name belonging to Switch Port X. Switch Port Y also transitions to the MONITORING state with PARTICIPATE = FALSE(0), because only one FL_Port may be Participating on a Loop.

Switch Port X receives its LISM and assumes the role of Loop Master. Switch Port X then proceeds to send all of the other Loop Initialization Sequences, and by the end of Loop Initialization, discovers that it is the only L_Port on the Loop. Because there may be a Non-Participating Switch Port on the Loop, Switch Port X knows it must attempt Link Initialization. Switch Port X begins Link Initialization by REQ(old-port). Switch Port X transitions to the OLD-PORT-REQ state and begins transmitting LIP; this causes Switch Port Y to begin Loop Initialization. Switch Port Y transmits a minimum of 12 of the received LIPs in the OPEN-INIT-START state and transitions to the OPEN-INIT-SELECT-MASTER state. When Switch Port X recognizes LIP, it transitions to the OLD-PORT state and transmits OLS for minimum(2xAL_TIME). After a maximum(1xAL_TIME), Switch Port Y recognizes ANSI X3, FC-PH-x Primitive Sequences (OLS, NOS) and transitions from the FL_Port operating mode to E/F_Port mode. The Link protocol continues to completion and a point-to-point Link is now active.

Switch Port X and Switch Port Y may now attempt to Exchange Link Parameters and establish an Inter-Switch Link.

Q.2 Example 2: two E/F/FL_Port-capable Switch Ports and one Nx_Port

In this example, two Switch Ports that are E/F/FL_Port-capable are attached to each other as in the first example, but there is also an N/NL_Port on the Loop. Figure Q.2 illustrates this example.

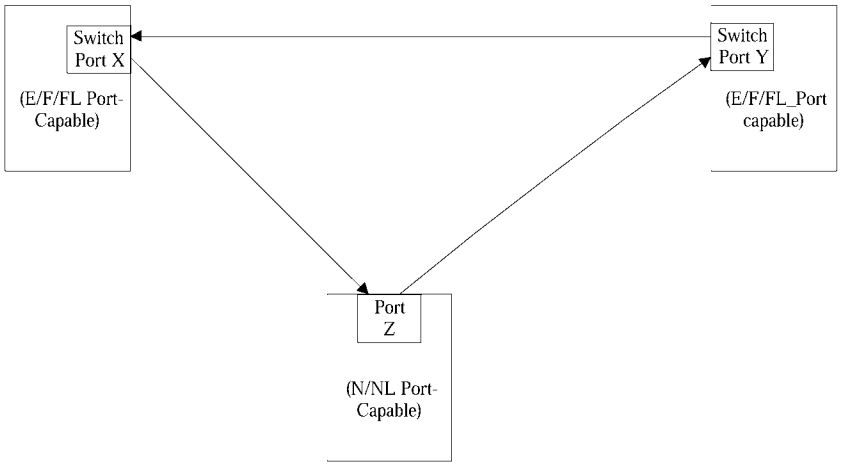

Figure Q.2 — Switch Initialization example 2

According to the initialization algorithm, since both Switch Port are E/F/FL_Port-capable and Port Z is N/NL_Port-capable, they start the process with normal Loop Initialization. LIPs are sent and recognized, and each Switch Port and the N/NL_Port start sending LISM frames. As in the first example, Switch Port X receives LISM from Switch Port Y, it sees that its Port_Name is lower than the Port_Name in the Payload, and continues sending the same LISM.

When Port Z receives the LISM from Switch Port X, Port Z finds a D_ID of zero, meaning that the originator is an FL_Port. Since an FL_Port always wins as a Loop Master, the NL_Port continues sending the received LISM from Switch Port X. When Switch Port Y receives Switch Port X's LISM from Port Z, it sees that its Port_Name is higher than the Port_Name in the Payload. This causes Switch Port Y to start sending the LISM it received, with the Port_Name belonging to Switch Port X. Switch Port Y also transitions to the MONITORING state in Non-Participating mode, because only one FL_Port may be Participating on a Loop.

Switch Port X receives its LISM and assumes the role of Loop Master. Switch Port X then proceeds to send all of the other Loop Initialization Sequences, and by the end of Loop Initialization, discovers that there is only one other L_Port on the Loop. Because that one other L_Port may be capable of point-to-point operation, Switch Port X knows it must attempt Link Initialization.

Switch Port X begins Link Initialization by asserting REQ(old-port) which begins transmitting LIP in the OLD-PORT-REQ state, and causes Port Z to begin Loop Initialization. Port Z transmits a minimum of 12 received LIPs in the OPEN-INIT-START state (which causes Switch Port Y to begin Loop Initialization) and transitions to either the OPEN-INIT-SELECT-MASTER or the SLAVE-WAIT-FOR-MASTER state. Switch Port Y transmits a minimum of 12 received LIPs in the OPEN-INIT-START state and transitions to the OPEN-INIT-SELECT-MASTER state. Switch Port X recognizes LIP, transitions to the OLD-PORT state and transmits OLS for minimum(2xAL_TIME). If after minimum(1xAL_TIME), Port Z recognizes OLS and reacts to it, it transitions to the OLD-PORT state and transmits LR in response. Switch Port Y being in the OPEN-INIT-SELECT-MASTER state does not recognize LR, and continues with the INITIALIZATION process; thereby blocking LR to Switch Port X. Switch Port X will fail Link Initialization; it should remove REQ(old-port) to allow Loop Initialization to complete. When Loop Initialization completes successfully, and Switch Port X operates as an FL_Port, and Port Z operates as an NL_Port. Switch Port Y stays Non-Participating until a system administrator comes to save it from oblivion.

Note that if Port Z had been bypassed, the process would have completed as in example 1, because the Primitive Sequences would have been ignored by Port Z. At a later time, when Port Z is enabled, Loop Initialization begins (i.e., Port Z starts sending LIP to get an AL_PA), and things sort themselves out as described in example 2. If Switch Port Y had been bypassed, then Switch Port X would have become an F_Port in a point-to-point Link with N_Port Z.

If Port Z was L_Port capable only when it went to the OPEN-INIT-START state, it would stall in either the OPEN-INIT-SELECT-MASTER or SLAVE-WAIT-FOR-MASTER state transmitting LISM or waiting for an ARB(F0). This would cause Switch Port X to fail at Link Initialization, and then go back to Loop Initialization. Again, Switch Port Y stays Non-Participating until a system administrator comes to save it from oblivion.

Q.3 Example 3: one E/F/FL_Port-capable Port and one E/F_Port-capable Port

Figure Q.3 — Switch Initialization example 3

In this example, a Switch Port that is E/F/FL_Port-capable is attached to a Switch Port that is E/F_Port-capable. Figure Q.3 illustrates this example. According to the initialization algorithm, the Switch Port that is E/F/FL_Port-capable starts the process with normal Loop Initialization. However, the Switch Port that is E/F_Port-capable starts the process with Link Initialization as defined in ANSI X3, FC-PH-x. Switch Port X sends LIP; Switch Port Y sends OLS Primitive Sequences. If Switch Port X recognizes OLS during Loop Initialization, it transitions to the OLD-PORT state after expire(2xAL_TIME) and completes Link Initialization.

Switch Port X and Switch Port Y may now attempt to Exchange Link Parameters and establish an Inter-switch Link.

Index

End of Document

Printed:
June 28, 1999 at 11:10PM