Copies of this document may be purchased from:
Global Engineering, 15 Inverness Way East,
Englewood, CO 80112-5704
Phone: (800)854-7179 or (303)792-2181 Fax: (303)792-2192

X3.272-199x
T11/Project 1133D/Rev 6.1

FIBRE CHANNEL

ARBITRATED LOOP
(FC-AL-2)

REV 6.1

NCITS working draft proposal
American National Standard
for Information Technology

February 16, 1998

SECRETARIAT: Information Technology Industry Council

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

NOTE: ***This is an internal working document of T11, a Task Group of Accredited Standards Committee NCITS. As such, this is not a completed standard. The contents are actively being modified by the T11 Task Group. This document is made available and may be reproduced for review and comment only. For current information on the status of this document contact the individuals shown below:***

POINTS OF CONTACT:

Roger Cummings (T11 Chair)
Distributed Processing Technology
140 Candace Drive
Maitland, FL 32751
(407)830-5522x348 Fax: (407)260-5366
E-Mail: cummings_roger@dpt.com

I. Dal Allan
(Fibre Channel Working Group Chair)
14426 Black Walnut Court
Saratoga, CA 95070
(408)867-6630 Fax: (408)867-2115
E-Mail: dal.allan@mcimail.com

Edward L Grivna (T11 Vice-Chair)
Cypress Semiconductor
2401 East 86th Street
Bloomington, MN 55425
(612)851-5046 Fax: (512)851-5087
E-Mail: elg@cypress.com

Horst L Truestedt (Editor)
ENDL Associates
12 Elmwood Dr NE
Pine Island, MN 55963-9740
Voice/Fax: (507)356-4701
E-Mail: hotrues@ibm.net

Changes in this document from FC-AL version 5.7

(as marked in this document)

- removed ARB(F7)
- added ARBf
- modified DHD for full-duplex operation only
- added new annex A (elasticity buffer management)
- replaced annex G (clock skew)
- removed annex C
- removed the requirement to transmit LIP(F7) during Loop recovery in the INITIALIZING state.
- corrected all "controversial" items discussed in Palm Springs, San Jose, and San Diego.
- global change to not allow an L_Port to go non-participating unless the L_Port is in the MONITORING or OPEN-INIT states.
- global change to not allow an L_Port to REQ(Bypass L_Port) unless the L_Port is in the MONITORING, INITIALIZING, or OPEN-INIT states.
- global change from LP_BYPASS to REPEAT (for repeater function) and BYPASS (for the port bypass circuit set to bypass).
- added REQ(send DHD) line to state tables.
- updated all entry conditions to the state tables.

- Note: the Ordered Set bits that need to be received have not been defined in this document.

ANSI®
dp ANSI X3.xxx-199x

draft proposed American National Standard
for Information Technology

Fibre Channel — Arbitrated Loop (FC-AL-2)

Secretariat

Information Technology Industry Council

Approved xxxx xx, 199x

American National Standards Institute, Inc

Abstract

This standard defines functional requirements for an interoperable Arbitrated Loop topology to support the Fibre Channel standard. This standard replaces X3.272-1996.

American National Standard

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

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 toward 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 on 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.

PATENT STATEMENT

The developers of this Standard have requested that holder's 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 the publisher in respect to any Standard in 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
11 W. 42nd Street, New York, New York 10036

Printed in the United States of America

Contents

Annexes and Index

Figures

Tables

Foreword (This foreword is not part of American National Standard X3.272-199x.)

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

This standard was prepared by Task Group X3T11 (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 X3 Secretariat, Computer and Business Equipment Manufacturers Association, 1250 Eye Street, NW, Suite 200, Washington, DC 20005-3922.

This standard was processed and approved for submittal to ANSI by the Accredited Standards Committee on Information Processing Systems, X3. Committee approval of the standard does not necessarily imply that all committee members voted for approval. At the time it approved this standard, the X3 Committee had the following members:

Richard Gibson, Chair
Donald C. Loughry, Vice-Chair
Joanne M. Flanagan, Secretary

NOTE: 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 publishers have undertaken a patent search in order to identify which, if any patents, may apply to this standard. No position is taken with respect to the validity of any claims or any patent rights that may have been disclosed. Details may be obtained from the publisher concerning any statement of patents and willingness to grant a license on a non-discriminatory basis and with reasonable terms and conditions to applicants desiring to obtain such a license.

Organization Represented	Name of Representative
[To be supplied]	

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, or system architecture.

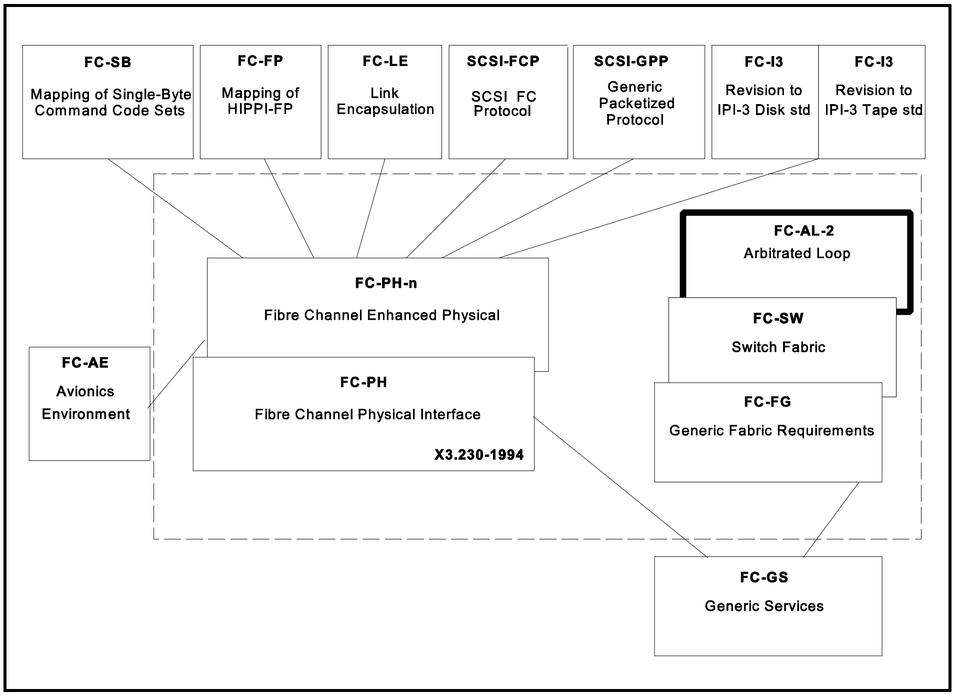

FC-SB
Mapping of Single-Byte Command Code Sets
FC-FP
Mapping of HIPPI-FP
FC-LE
Link Encapsulation
SCSI-FCP
SCSI FC Protocol
SCSI-GPP
Generic Packetized Protocol
FC-I3
Revision to IPI-3 Disk std
FC-I3
Revision to IPI-3 Tape std
FC-AL-2
Arbitrated Loop
FC-PH-n
Fibre Channel Enhanced Physical
FC-SW
Switch Fabric
FC-AE
Avionics Environment
FC-PH
Fibre Channel Physical Interface
X3.230-1994
FC-FG
Generic Fabric Requirements
FC-GS
Generic Services

Figure 0 — Fibre Channel roadmap

FC-AL features enhanced Ports, called L_Ports, that 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., communications 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 works 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 Port initialization procedure.

draft proposed American National Standard 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 hubs and 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.

Copies of the following documents can be obtained from ANSI: Approved ANSI standards, approved and draft international and regional standards (ISO, IEC, CEN/CENELEC, ITUT), and approved and draft foreign standards (including BSI, JIS, and DIN). For further information, contact ANSI Customer Service Department at 212-642-4900 (phone), 212-302-1286 (fax) or via the World Wide Web at http://www.ansi.org.

2.1 Approved references

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

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

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)*

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.

X3 Project 959-D, *Information Technology — Fibre Channel — Switch Topologies and Switch Control (FC-SW)*

X3 Project 1119-D, *Information Technology — Fibre Channel — Physical and Signaling Interface (FC-PH-3)*

| X3 Project 1162-DT, *Information Technology — Fibre Channel — Private Loop Direct Attach (FC-PLDA)*

| X3 Project 1236-DT, *Information Technology — Fibre Channel — Fabric Loop Attachment (FC-FLA)*

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. 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. 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 to change a full-duplex transfer to a half-duplex transfer 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 ARB 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 **Loop:** The Arbitrated Loop described in this document.

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

3.1.16 **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.)

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

3.1.18 **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.

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

3.1.20 **non-participating mode:** An L_Port that is not active on the Loop (i.e., it does not have a valid AL_PA). (See 8.3.1.)

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

3.1.22 participating mode: An L_Port that is active on the Loop (i.e., it does have a valid AL_PA). (See 8.3.1.)

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

3.1.24 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.)

3.1.25 Private Loop: A Loop that does not include a participating FL_Port. (See figure 1 and annex J.)

3.1.26 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.)

3.1.27 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.)

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

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

3.1.30 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.

3.1.31 trusted AL_PA: an AL_PA which is assumed to be valid 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.

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

Each individual entry that appears within parentheses behind the Primitive Signals ARBx and OPNy represents the hexadecimal value of an AL_PA.

All history variables, when used in this standard, are assumed to be set to zero (0) at power-on time.

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 (e.g., the x value in ARBx) (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 transmitted one or more ARBx 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	**Arb**itrate Primitive Signal — either ARBx or ARBf (see 7.1)
ARBx	**Arb**itrate Primitive Signal — an ARB Primitive Signal which contains an AL_PA or hex 'F0' (see 7.1.1 and 7.1.2)
ARBf	**Arb**itrate Primitive Signal — a special Fill Word to replace Idles where f = hex 'FF' (see 7.1.3)
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 has modified the current Fill Word to ARBf (see 8.1.7 and 8.4.3).
Available_BB_Credit	**Available B**uffer-to-**B**uffer **Credit** (see 8.3.5)
BB_Credit	**B**uffer-to-**B**uffer **Credit** (see 8.3.5 and ANSI X3, FC-PH-x, 26.5.2)
BYPASS	**BYPASS** history variable — a two-valued variable (i.e., *0*/*1*) that indicates the position of the optional bypass circuit (see 8.1.4.2)
CLS	**CL**o**S**e Primitive Signal (see 7.4)
CFW	**C**urrent **F**ill **W**ord (i.e., Idle or ARB) (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 **R**e**C**ei**V**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 10.4)
LIHA	**L**oop **I**nitialization **H**ard **A**ssigned — Loop Initialization Sequence (see 10.4)

LILP	**L**oop **I**nitialization **L**oop **P**osition — Loop Initialization Sequence (see 10.4)
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 — reset 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 10.4)
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 10.4)
LISA	**L**oop **I**nitialization **S**oft **A**ssigned — Loop Initialization Sequence (see 10.4)
LISM	**L**oop **I**nitialization **S**elect **M**aster — Loop Initialization Sequence (see 10.4)
Login DHD	**Login** **D**ynamic **H**alf **D**uplex — a two-valued variable (i.e., *0/1*) to indicate that the L_Port in the OPENED state has implemented the DHD function (see 7.5)
LI_FL	**L**oop **I**nitialization **FL**ag — Loop Initialization flag (see 10.4)
LI_ID	**L**oop **I**nitialization **ID**entifier — Loop Initialization identifier (see 10.4)
LP_TOV	**L**oo**P** **T**ime**O**ut **V**alue — timer used during Loop Initialization and to reset the fairness window (see 8.2.2)
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 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	**Ma**r**k** Primitive Signal (see 7.6)
MK_TP	**M**ar**k** **T**y**p**e — used to identify the type of Mark Primitive Signal (e.g., clock synchronization) (see 7.6)
OPNr	**Op**e**n** Replicate Primitive Signal — either OPNyr or OPNfr Primitive Signals (see 7.3)
OPNfr	**Op**e**n** Primitive Signal — broadcast replicate(see 7.3.2)
OPNy	**Op**e**n** Primitive Signal — either OPNyx or OPNyy Primitive Signals (see 7.2)
OPNyr	**Op**e**n** Primitive Signal — selective replicate(see 7.3.1)
OPNyx	**Op**e**n** Primitive Signal — full-duplex (see 7.2.1)
OPNyy	**Op**e**n** Primitive Signal — half-duplex (see 7.2.2)

REPEAT	**REPEAT** history variable — a two-valued variable (i.e., *0*/*1*) to indicate that an L_Port is retransmitting received Transmission Words (see 8.1.4.1)
REPLICATE	**Replicate** history variable — a two valued variable (i.e., *0*/*1*) to indicate if the NL_Port has detected OPNr while in the MONITORING, ARBITRATING, or OPEN 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 10.4)

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 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.

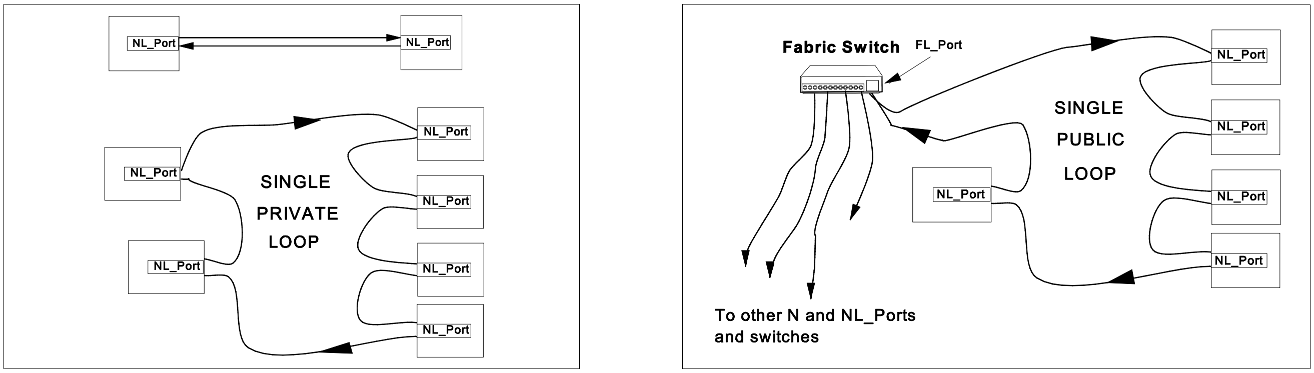
  One advantage for the Fabric topology is that when there is at least one free F_Port in the Fabric, a new N_Port can be added to the free F_Port without disrupting the remaining N_Ports. The new N_Port has the potential to communicate with all other N_Ports in the Fabric. However, adding an N_Port does 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 which include two (i.e., point-to-point) and six NL_Ports only (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.)



a) b)

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 consecutive Idles have been transmitted by the L_Port. The time between when the first L_Port wins arbitration and transmits two consecutive Idles is an access window. 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.

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.

Although it is recommended that all NL_Ports implement the fairness algorithm, neither FL_Ports nor NL_Ports are required to use the fairness algorithm at all times. For example, if one L_Port requires more Loop accesses than the other L_Ports, that L_Port may choose to be unfair. 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 L_Port has arbitrated for and won access to the Loop and does not detect that another L_Port is arbitrating, that L_Port may keep the existing Loop circuit open indefinitely or close that circuit and not relinquish control of the Loop to open another L_Port on the Loop.

When a fair NL_Port has access to the Loop and detects that another L_Port is arbitrating, the NL_Port may 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 fairness algorithm, it is called an *unfair* NL_Port. The decision whether to participate in access fairness is beyond the scope of this standard. (See annex D.)

When an unfair L_Port has arbitrated for and won access to the Loop and does not detect that another L_Port is arbitrating, that L_Port may keep the existing Loop circuit open indefinitely or close that circuit and not relinquish control of the Loop 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 at the earliest possible time. The unfair NL_Port may not relinquish control of the Loop 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 exempted from using access fairness algorithm because the majority of its traffic is with the rest of the Fabric.

When an FL_Port controls the Loop and detects that an NL_Port is arbitrating, the FL_Port may close the Loop at the earliest possible time. Because the FL_Port has the highest priority and is exempted from fairness, it will always win arbitration. Therefore, if communication is required with another NL_Port, the FL_Port may not relinquish control of the Loop to open another L_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.

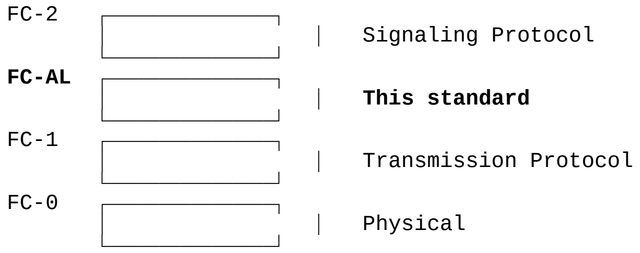


Figure 2 — FC-PH with Arbitrated Loop addition

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 8.3.5);

- 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, ARBx, and all Primitive Sequences. (See 8.3.3);

- specifies the Primitive Signals that may be inserted on a Loop between frames for clock skew management. (See 8.3.3);

- 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 10.4);

- 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 8.3.1) shall be assigned a local Arbitrated Loop Physical Address (AL_PA) during the initialization procedure. When an FL_Port is not present on a Loop, the assigned AL_PA shall be extended and used as its native address identifier. (See clause 10.) 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								
xx	0	1	2	3	4	5	6	7	Hex value
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 participating mode. The maximum number of FL_Ports in participating mode on a single Loop shall not exceed one. Additional FL_Ports may be present, but they shall be in the non-participating 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 for fairness

Value hex 'F0' is the next lower priority outside the range hex '00' through hex 'EF'. Hex 'F0' shall only be used for the access fairness algorithm and during Loop Initialization.

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 ARBf 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.

  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 originated and detected 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 originated and recognized by L_Ports on the Loop as additional Primitive Sequences. (See clause 7.)

Table 2 — Primitive Signals

Primitive Signal		Beginning RD	Ordered Set
Arbitrate (F0) fairness	(ARB)	Negative	K28.5 D20.4 D16.7 D16.7
Arbitrate as x	(ARBx)	Negative	K28.5 D20.4 AL_PS AL_PS
Arbitrate (FF)	(ARBf)	Negative	K28.5 D20.4 D31.7 D31.7
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

The Ordered Set definitions of ARBx and OPNyy require the same valid AL_PA value in characters 3 and 4. (See 7.1.1 and 7.2.2.)

The Ordered Set definitions of ARBf requires a D31.7 (hex 'FF') in both characters 3 and 4. (See 7.1.3.)

The Ordered Set definition of OPNyx require a valid AL_PA in characters 3 and 4. (See 7.2.1.)

The Ordered Set definition of OPNyr requires a valid AL_PA in character 3 and D31.7 (hex 'FF') in character 4 (see 7.3.1). The Ordered Set definition of OPNfr requires a D31.7 (hex 'FF') in both characters 3 and 4. (See 7.3.2.)

The Ordered Set definition of MRKtx requires a valid MK_TP and AL_PS in characters 3 and 4, respectively. (See 7.6.)

Table 3 — Primitive Sequences

Primitive Signal		Beginning RD	Ordered Set
Loop Initialization--F7,F7	(LIP)	Negative	K28.5 D21.0 D23.7 D23.7
Loop Initialization--F8,F7	(LIP)	Negative	K28.5 D21.0 D24.7 D23.7
Loop Initialization--F7,x	(LIP)	Negative	K28.5 D21.0 D23.7 AL_PS
Loop Initialization--F8,x	(LIP)	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 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

The Ordered Set definition of LIP includes two values (F7 and F8) to identify the reason for the LIP. AL_PS is used to identify the L_Port that initiated the LIP; AL_PD is used to identify the L_Port that should be reset (see 7.8).

NOTE — LIPfx uses D31.7 (hex 'FF') to indicate that the LIP is processed by all L_Ports (except the L_Port at AL_PD = x) (see 7.8.5).

The Ordered Set definitions of LPByx and LPEyx require a valid AL_PA in characters 3 and 4.

NOTE — LPBfx and LPEfx use D31.7 (hex 'FF') to indicate that it shall be processed by all L_Ports (except the L_Port at AL_PD = x) (See 7.7.)

7 FC-AL Primitive Signals and Sequences

The Arbitrate Primitive Signal 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 may also be transmitted in place of a Fill Word, but it shall not be removed for clock skew management. All Primitive Signals (except Fill Words) defined in this standard shall follow the FC-PH 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.)

7.1 Arbitrate Primitive Signals (ARB)

7.1.1 Arbitrate (ARBx)

Arbitrate (ARBx) is transmitted on a Loop by a participating L_Port to request access to the Loop. Each ARBx shall contain the AL_PA (x value) of the L_Port making the request.

7.1.2 Arbitrate (ARB(F0))

Arbitrate (ARB(F0)) is transmitted on a Loop to manage access fairness. Since this is a low-priority ARB, any arbitrating L_Port may replace the ARB(F0) with its ARBx. ARB(F0) is also used while selecting a temporary Loop Initialization Master during Loop Initialization.

7.1.3 Arbitrate (ARBf)

Arbitrate (ARBf) may be transmitted by an L_Port in the MONITORING state when the arbitration window has been reset and no Fill Words other than ARBf or Idle are received.

7.2 Open Primitive Signals (OPNy)

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 an 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 an L_Port in the OPEN state indicates that the L_Port opened itself. OPNyy that is received by an 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. The opened L_Port shall not transmit Data frames.

7.3 Open Replicate Primitive Signals (OPNr)

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 be ignored. 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 an 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 frame shall be retransmitted to the next L_Port on the Loop. NL_Ports shall provide all frames to FC-2 for further processing, however, the FL_Port 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).

If an L_Port wins arbitration while REPLICATE is TRUE(1) (e.g., the L_Port which originated the OPNr was removed from the Loop before transmitting CLS), the L_Port in the ARBITRATION WON state shall transmit CLS (i.e., causes all L_Ports to set REPLICATE to FALSE(0)) and shall go to the TRANSFER state. (See 8.4.3, item 15 and table 6.)

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 a participating L_Port. 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.)

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 it has no more Data frames to transmit. DHD shall only be requested by the L_Port in the OPEN state if both L_Port in the current Loop circuit via Login support the DHD feature. 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 two Transmission Words (negligible) to the closing process, but otherwise causes no round-trip delay.

Transmitting DHD only affects Data frames (i.e., Link_Control frames and R_RDYs may still be transmitted) just as in the definition of an 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, an 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)

Mark (MRKtx) is transmitted on a Loop by a master control point to synchronize other Nodes. (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 window is exceeded (i.e., REQ(mark as tx) is 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 type of synchronization (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), the MRKtx shall be replaced with the CFW. All other L_Ports which are in the MONITORING, ARBITRATING, XMITTED CLOSE, or TRANSFER state 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

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 Bypass Circuit. The Bypass Circuit may be used to physically bypass an L_Port, however, the L_Port is also logically bypassed (i.e., the L_Port cannot originate Transmission Words on the Loop). (See 8.1.4 and annex I.)

When an L_Port is bypassed, 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), but shall keep its AL_PA until it recognizes LIP. When LIP is recognized, the L_Port shall relinquish its AL_PA and it shall enter the non-participating mode.

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 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.

LPByx may be used to diagnose the optional Bypass Circuit and for error recovery. (See annex I.)

Each L_Port in the MONITORING, ARBITRATING, INITIALIZING, or OPEN-INIT state shall retransmit the received LPByx. When LPByx is received by the originator (i.e., x = AL_PA of the L_Port), the LPByx shall be replaced with the CFW.

Although LPByx may be transmitted in a number of states, not all states retransmit LPByx. To guarantee that the designated L_Port (as identified by the y value) receives LPByx, 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 OPEN-INIT state.

Once an L_Port is bypassed and the optional Bypass Circuit has been activated, the L_Port shall only monitor the Loop for a LPEyx (where y = AL_PA) or LPEfx and LIP. LIP is only used as a signal to relinquish its AL_PA; the L_Port shall not go to the OPEN-INIT 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 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).

When LPBfx is recognized, all L_Ports on the Loop except the L_Port at x (participating or non-participating), shall set REPEAT to TRUE(1) and shall activate the optional Bypass Circuit. (See annex I.)

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 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, INITIALIZING, or OPEN-INIT state shall retransmit the received LPBfx. When LPBfx is received by the originator (i.e., x = AL_PA), the LPBfx shall be replaced with the CFW.

Although LPBfx may be transmitted at any time, not all states retransmit LPBfx. To guarantee that all L_Ports receive LPBfx, 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 OPEN-INIT state.

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 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).

When LPEyx is recognized, the previously bypassed L_Port may participate on the Loop (e.g., arbitrate). LPEyx is primarily used to detect if a Bypass Circuit is present and operational and for error recovery. (See annex I.)

Each L_Port in the MONITORING, ARBITRATING, INITIALIZING, or OPEN-INIT state shall retransmit the received LPEyx. When LPEyx is received by the originator (i.e., x = AL_PA), the LPEyx shall be replaced with the CFW.

Although LPEyx may be transmitted at any time, not all states retransmit LPEyx. To guarantee that the designated L_Port (as identified by the y value) receives LPEyx, 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 OPEN-INIT state.

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 Circuit(s) that may have been previously activated and to enable all L_Ports to participate on the Loop (e.g., arbitrate). 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 a LIP). Therefore, LPEfx allows these L_Ports (which no longer have an AL_PA) to be enabled on the Loop.

When LPEfx is recognized, a previously bypassed participating L_Port may participate on the Loop; a previously non-participating L_Port may perform Loop Initialization. (See annex I.)

Each L_Port in the MONITORING, ARBITRATING, INITIALIZING, or OPEN-INIT state shall retransmit the received LPEfx. When LPEfx is received by the originator (i.e., x = AL_PA), the LPEfx shall be replaced with the CFW.

Although LPEfx may be transmitted at any time, not all states retransmit LPEfx. To guarantee that all L_Ports receive LPEfx, 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 OPEN-INIT state.

7.8 Loop Initialization Primitive Sequences (LIP)

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, items 21, 22, and 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 (except the one at AL_PS). All L_Ports (including those which do not have an AL_PA), shall treat this as a reset LIP.

8 L_Port operation

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

- all routing decisions by the LPSM (except during initialization) 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,
- frame detection shall not be supported 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.

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 ARBx in place of the appropriate received Fill Word until a matching ARBx is received. When the matching ARBx 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 8.3.5).

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

NOTE — Since either open L_Port may transmit a CLS, an L_Port must be prepared to handle a 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 three memory elements that shall be maintained and used by each L_Port:

(1) **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). (See 8.4 for management requirements of ACCESS);

(2) **ARB_WON —** the value of this variable is used by an L_Port to indicate that this L_Port has won arbitration. (See 8.4 for management requirements of ARB_WON.)

(3) **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 ARBx Primitive Signals. (See 8.4 for management requirements of ARB_PEND.)

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 an OPNr had been received. If REPLICATE is FALSE(0), the states operate normally; if REPLICATE is TRUE(1), all Transmission Words are received and retransmitted and all frames are provided to the FC-2 of the NL_Port for further processing. (See 7.3 and 8.4.3, items 13 and 14.)

The OPEN state requires REPLICATE to remember that OPNr was transmitted in the ARBITRATION WON state. If REPLICATE is FALSE(0), the state operates normally; if REPLICATE is TRUE(1), the L_Port shall absorb all received frames (i.e., not retransmit the received frames). (See 7.3 and 8.4.3, items 15 and 16.)

The ARBITRATION WON state requires REPLICATE to detect that the originator of the OPNr left the Loop circuit without transmitting a CLS. (See 7.3 and 8.4.3, item 15.)

8.1.4 L_Port repeat/bypass history

The MONITORING state requires two memory elements called REPEAT and BYPASSED.

8.1.4.1 L_Port repeat history

The REPEAT memory element is used to determine whether the L_Port may originate Transmission Words on the Loop (e.g., ARBx, OPNy, frames, etc.). When REPEAT is FALSE(0), the L_Port is allowed to originate Transmission Words; if REPEAT is TRUE(1), the L_Port is is not allowed to originate Transmission Words; the L_Port is only allowed to retransmit received Transmission Words. When REPEAT is set to TRUE(1) and a LIP is recognized, the L_Port shall relinquish its AL_PA and shall enter the non-participating mode. (See 8.3.1 and 8.4.3, item 13.)

8.1.4.2 L_Port bypass history

The BYPASS memory element is used by the L_Port to remember if the optional Port Bypass Circuit is set or reset. An L_Port is bypassed when it recognizes LPByx (where y is the AL_PA of the L_Port), LPBfx (when x <> AL_PA of the L_Port) (see 7.7.2), or at the request of the L_Port (REQ(bypass L_Port)). When an L_Port is bypassed, it shall set BYPASS and REPEAT to TRUE(1) and it shall activate the optional Port.

An L_Port is enabled when it recognizes LPEyx (where y is the AL_PA of the L_Port), LPEfx, or at the request of the L_Port (REQ(enable L_Port)). When an L_Port is enabled, it shall set BYPASS to FALSE(0) and shall deactivate the optional Bypass Circuit. When BYPASS is FALSE(0), the L_Port shall operate normally depending on the state of REPEAT (i.e., it shall be in the non-participating or participating mode). (See 8.3.1 and 8.4.3, item 13.)

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 OPEN-INIT state may use one memory element, called ERR_INIT. The variable is checked by the L_Port in the OPEN-INIT state to determine whether Loop Initialization (clause 10) should be continued or delayed (to avoid initializing in 10.4 when there is a low probability that it will complete). (See 8.4, items 13 and 22.)

8.1.7 ARBf history

The MONITORING state uses a history value called ARBf_SENT to indicate that the L_Port has modified its CFW to ARBf from Idles.

8.2 Timeouts

8.2.1 FC-PH timeout values

Timeout values and related timeout procedures in ANSI X3, FC-PH-x, 29.2, shall be used including the definition of E_D_TOV and R_A_TOV during Login, as appropriate.

8.2.2 Arbitrated Loop timeout values

8.2.2.1 Arbitrated Loop timeout

The Arbitrated Loop timeout (AL_TIME) is specified as 15 ms (-0%+20%), 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 Gbits/sec of the L_Port and

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

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

8.2.2.2 Loop timeout value

The Loop timeout value (LP_TOV) is specified to be 2 seconds. LP_TOV is used to reset the fairness window (see 4.3) and during the INITIALIZING and OPEN-INIT states to time start-up events (see 8.4.3, items 21 and 22, and 10.4).

8.3 Operational characteristics

8.3.1 Modes of operation

An L_Port is in one of two operational modes:

participating mode: An L_Port is in participating mode when it has acquired an AL_PA (i.e., through the initialization procedure or by some other means. See clause 10.) Since there is no enforceable limit to the number of L_Ports that may be physically connected to a Loop, a maximum of 127 L_Ports (1 FL_Port and 126 NL_Ports) shall be in participating mode on the same Loop at the same time.

NOTE — Some laser safety requirements may further limit the number of L_Ports that may be connected in a Loop because of propagation delays.

An L_Port that is in participating mode may voluntarily relinquish control of its AL_PA and enter the non-participating mode. This allows another L_Port to reuse that AL_PA.

non-participating mode: An L_Port is in non-participating mode when it does not have a valid AL_PA. Reasons for not having an AL_PA are: the L_Port was unable to obtain an AL_PA; the L_Port voluntarily does not participate, or the L_Port has been bypassed and has recognized a LIP. Non-participating mode is the default operational mode for an L_Port. An L_Port in non-participating mode shall not arbitrate for access to and shall not respond to any ARBx, OPNy, or OPNr received on the Loop. (See 8.4.3 MONITORING.)

NOTE — A non-participating L_Port does not have a valid AL_PA and can therefore not be individually bypassed (if a Bypass Circuit is present) by another L_Port. To prevent a non-participating L_Port from causing a Loop Failure, it is recommended that a non-participating L_Port requests a bypass (REQ(bypass L_Port)).

8.3.2 Transmission Word processing

8.3.2.1 Power-on Transmission Words

Transmission Words which are transmitted during the various phases of the power-on cycle of an L_Port are undefined. (See annex I).

8.3.2.2 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 condition is detected on an Ordered Set, the L_Port shall substitute the CFW.
- in the INITIALIZING state, the L_Port shall substitute LIP.
- in any other state the L_Port shall follow the rules defined in ANSI X3, FC-PH-x, 24.3.5, and clause 29.

8.3.3 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 ARB 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.)

8.3.4 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.

8.3.5 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 state, respectively. (See 8.4.3, and items 22 and 23). 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.

8.3.5.1 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 a 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 up to 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 up to one R_RDY for each additional available receive buffer.

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 up to 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 up to 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.

8.3.5.2 Available_BB_Credit management per Loop circuit

Once the L_Port has discarded the same number of R_RDYs as the BB_Credit value, 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 *n* R_RDYs (where *n* equals BB_Credit) 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 is positive, the L_Port may transmit frames during this Loop circuit.

8.4 Loop Port State Machine (LPSM)

A Loop Port State Machine (LPSM) is 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 ARBx (i.e., x = 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 a CLS and intends to relinquish control of the Loop.
RECEIVED CLOSE (6):	The LPSM has received a CLS.
TRANSFER (7):	The LPSM, while in the OPEN state, has transmitted CLS and requires the Loop to communicate with another L_Port.
INITIALIZING (8):	The LPSM is initializing or re-initializing.
OPEN-INIT (9):	The LPSM has recognized a LIP.
OLD-PORT (A):	The LPSM has discovered that a non-L_Port is attached and the Arbitrated Loop protocol is not required.

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.

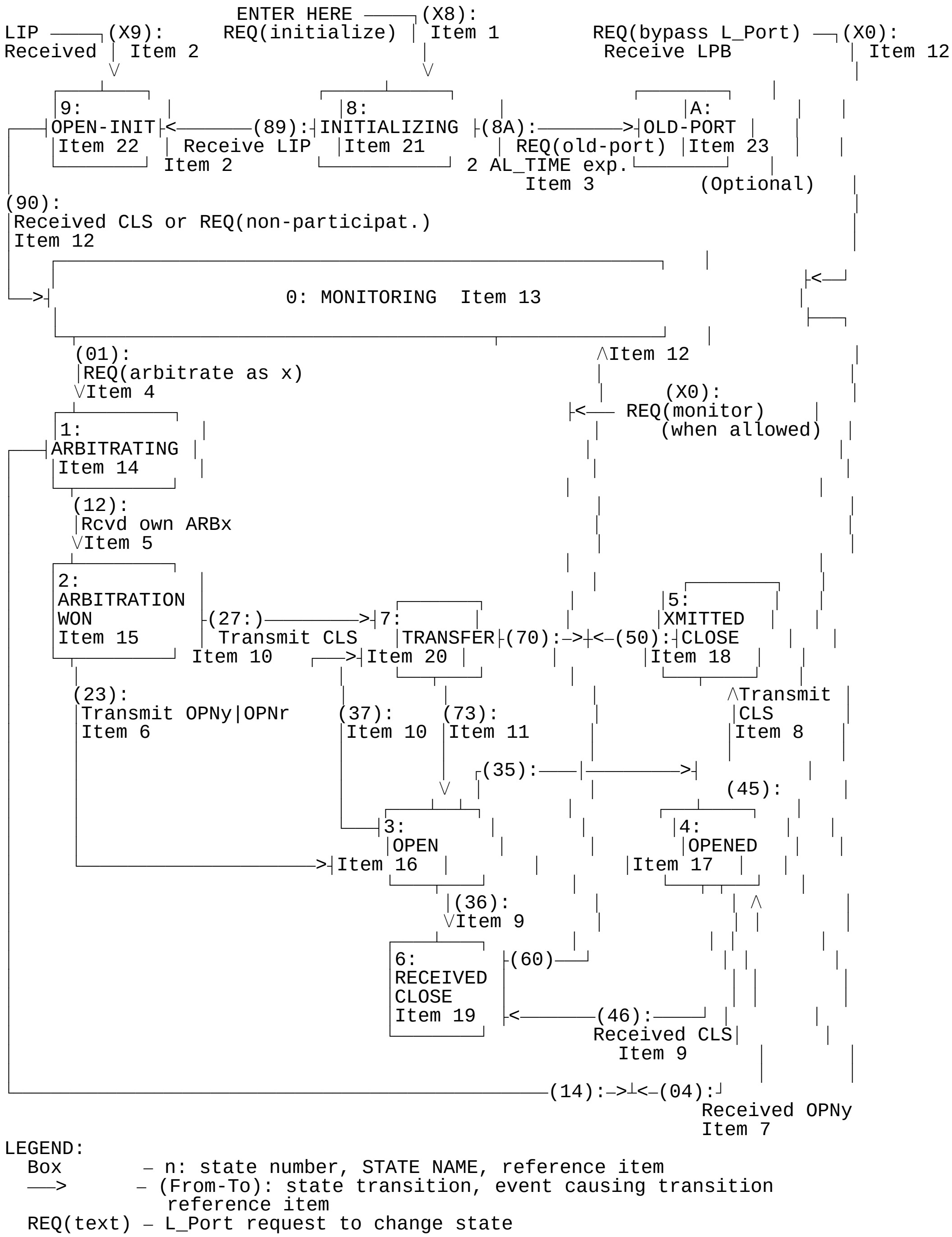


Figure 3 — State diagram

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 may[2] be made at power-on of an L_Port, after detecting a failure (see clause 10 and ANSI X3, FC-PH-x, clause 23), or from any state when the L_Port requests it. (See item 21.)

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 **Transitions (X9):, (89):** The LPSM shall make the transition to the OPEN-INIT state. (See items 13, 14, 16, 17, 18, 19, 20, 21, 22, and 23.)

3 **Transition (8A):** The LPSM shall make the transition to the OLD-PORT state (if supported). (See items 21 and 23.)

4 **Transition (01):** The LPSM shall make the transition to the ARBITRATING state. (See items 13 and 14.)

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

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

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

8 **Transitions (35):, (45):** The LPSM shall make the transition to the XMITTED CLOSE state. (See items 15, 16, 17 and 18.)

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

10 **Transition (27):, (37):** The LPSM shall make the transition to the TRANSFER state. (See items 15, 16, and 20.)

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

12 **Transitions (X0):, (50):, (60):, (70):, (90):** The LPSM shall make the transition to the MONITORING state. (See items 13, 15, 18, 20, and 22.)

---

[2] Some applications may choose to not allow an L_Port to got to the INITIALIZING state. 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), and REPLICATE to FALSE(0). The LPSM shall retransmit all received Transmission Words unless specifically stated otherwise. If ARB_PEND is TRUE(1) and REPEAT is TRUE(1), ARB_PEND shall be set to FALSE(0). If ARB_PEND is TRUE(1) and REPEAT is FALSE(0), the L_Port shall make an immediate transition (i.e., without receiving any Transmission Words) to the ARBITRATING state. (See items 4 and 14.)

NOTE – ARB_PEND is set to TRUE(1) when an L_Port has transmitted one or more ARBx Primitive Signals (where x = AL_PA of the L_Port). The L_Port may have been opened by another L_Port while arbitrating. This history variable forces the L_Port to finish arbitrating even if the L_Port no longer desires access to the Loop in order to assure that the fairness window is reset.

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

- if ARBf_SENT is FALSE(0), the CFW shall be set to Idle and when two Idles have been transmitted, ACCESS shall be set to TRUE(1). Since the Idle is used to reset the fairness window, at least two Idles shall be transmitted (to assure that a single Idle is not discarded for clock skew by another L_Port) before modifying the CFW to another value. If the L_Port requests to transmit ARBf (REQ(arbitrate (FF))) the LPSM shall change the CFW to ARBf after six (6) Idles have been forwarded and shall set ARBf_SENT to TRUE(1).

- if ARBf_SENT is TRUE(1), the CFW shall not be changed.

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

- if ARB = ARB(F0) and the CFW is Idle:
    - if REPEAT is FALSE(0), the CFW shall not be changed;
    - if REPEAT is TRUE(1), the CFW shall be changed to ARB(F0) and ACCESS shall be set to FALSE(0);
- if ARB = ARB(F0) and the CFW is not Idle, the CFW shall be set to ARB(F0) and ACCESS shall be set to FALSE(0);
- if ARB = ARBf, the CFW shall not be changed;
- if ARB = ARBx where x <> AL_PA of the L_Port, the CFW shall be set to ARBx; or,
- if ARB = ARBx where x = AL_PA of the L_Port, the CFW shall be changed to Idle.

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

If the L_Port is in participating mode with REPEAT set to FALSE(0):

- 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 Frame Delimiters and valid Data Words;

- 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 items 7 and 17);

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

- if ARB is received:
  - if ARB = ARBx where x = AL_PA of the L_Port, the LPSM shall transmit the CFW; the ARBx is discarded;
  - if ARB = ARB(F0), the LPSM shall transmit the CFW; or,
  - if ARB = ARBx where x <> AL_PA of the L_Port, the LPSM shall retransmit the ARB.
- if MRKtx is received:
  - 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, synchronization shall be performed; or,
  - if x <> AL_PA of the L_Port, the received MRKtx shall be retransmitted.
- if CLS is received while REPLICATE is TRUE(1), the LPSM shall set REPLICATE to FALSE(0) and retransmit the received CLS.
- if the L_Port requests arbitration (REQ(arbitrate as x)) and ACCESS is TRUE(1), the LPSM shall make the transition to the ARBITRATING state. (See items 4 and 14).
- if LP_TOV has elapsed since the L_Port began requesting arbitration (REQ(arbitrate as x)), ACCESS shall be set to TRUE(1) and the LPSM shall make the transition to the ARBITRATING state. (See items 4 and 14).
- if the LPSM detects a Loop Failure on its inbound fibre, the LPSM shall make the transition to the INITIALIZING state. (See items 1 and 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.

If LIP is received:

- if REPEAT is FALSE(0), the LPSM shall make the transition to the OPEN-INIT state. (See items 2 and 22); or,
- if REPEAT is TRUE(1), the L_Port shall relinquish its AL_PA (i.e., go to the non-participating mode) and 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 REPEAT and BYPASS to TRUE(1) and shall set REPLICATE to FALSE(0).

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).

The LPSM shall retransmit all other received Transmission Words on the Loop. (See 8.3.2.)

Invalid Transmission Character substitution shall be performed as specified in 8.3.2.

If the L_Port requests initialization (REQ(initialize)), the LPSM shall make the transition to the INITIALIZING state. (See items 1 and 21 and clause 10.)

If the LPSM detects a Loop Failure on its inbound fibre and BYPASS is TRUE(1), the LPSM shall transmit LIP(F8), but remain in the MONITORING state.

If the L_Port requests not to participate on the Loop (REQ(nonparticipat.)), it relinquishes its AL_PA. The LPSM may transmit 12 LIPs (with the right-most two characters equal to hex 'F7F7') to invoke the Loop Initialization procedure to allows another L_Port to acquire the relinquished AL_PA. 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)). The L_Port shall not participate further in Loop Initialization until REQ(initialize) or REQ(participating) is set.

NOTE – The L_Port may arbitrate for the Loop (using ARBx where 'x' is a trusted 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 INITIALIZING state. (See items 1 and 21 and clause 10.)

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

If Idle is received, the current Fill Word shall be set to ARBx (where x equals the AL_PA of the L_Port). To assure that a single Idle is not discarded for clock skew by another L_Port, at least two Idles shall be transmitted before changing the current Fill Word to another value.

If ARB is received and x does not equal the AL_PA of the L_Port, the CFW shall be modified as follows:

- if ARB = ARB(F0) | ARBf | ARBx where x > AL_PA, the CFW shall be changed to ARBx (where x equals the AL_PA of the L_Port);

- if ARB = ARBx, x < AL_PA, the CFW shall be changed to the received ARBx.

If a Fill Word is to be transmitted, the CFW shall be used. If the LPSM has transmitted its own ARBx, it shall set ARB_PEND to TRUE(1).

If ARBx is received and x equals the AL_PA of the L_Port, the LPSM shall make the transition to the ARBITRATION WON state. (See items 5 and 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 Frame Delimiters and valid Data Words;

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

If OPNfr is received, the LPSM 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 items 7 and 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.

If MRKtx is received:

- 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, synchronization 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 state. (See items 2 and 22.)

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

Invalid Transmission Character substitution shall be performed as specified in 8.3.2; any other received Transmission Words shall be retransmitted on the Loop. (See 8.3.2.)

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 INITIALIZING state. (See items 1 and 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)[3] actions** (table 6 and the following text describe the ARBITRATION WON state): This is a transition state during which Transmission Words are not received. If REPLICATE is TRUE(1), (e.g., the L_Port which originated the OPNr was removed from the Loop before transmitting CLS), the L_Port shall change the current Fill Word to ARB(F0), transmit CLS, and go to the TRANSFER state. (See items 10 and 20.)

The ARBx 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 items 6 and 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 items 6 and 16.)

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

---

[3] 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) or 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 fairness algorithm, ACCESS shall be set to FALSE(0); if the L_Port is not using the 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) (see 8.3.5.1).

NOTE — The six (6) Ordered Sets maintain the FC-PH 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 8.3.5.)

If the available BB_Credit of the L_Port is zero, and the L_Port does not receive an R_RDY in less than ED_TOV, the LPSM shall transmit CLS and make the transition to the XMITTED CLOSE state.

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

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

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

If MRKtx is received where the MK_TP and AL_PS match the expected values, synchronization 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 items 10 and 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 a CLS; only the L_Port which received EOFdt shall transmit 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 a CLS; only the L_Port which received EOFdt shall transmit CLS. (See items 8 and 18 or 20.)

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

- ARBx was detected to indicate that another L_Port is arbitrating (the OPEN L_Port may close the Loop at a convenient time);
- the L_Port has not received any credit to transmit frames before an E_D_TOV timeout occurred;
- there are no additional Sequences 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 state. (See items 2 and 22.)

If LPB is recognized:

- if x = AL_PA of the L_Port, the received LPByx shall be discarded or
- if y = AL_PA of the L_Port or hex 'FF', the LPSM shall end the current transmission; set REPEAT to TRUE(1); BYPASS to TRUE(1); and, make the transition to the MONITORING state. (See items 12 and 13.)

If REPLICATE is TRUE(1), all received Transmission Words (except CLS, MRKtx, and any Primitive Sequence) shall be discarded.

NOTE – This includes any frame(s) that traverses the Loop.

If the L_Port requests another L_Port to be either bypassed or enabled (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 INITIALIZING state. (See items 1 and 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 shall be set 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). 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 items 8 and 18). 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 8.3.5.) 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 this is a fair L_Port and ARB_PEND is TRUE(1), and if the x value of the OPNyx is equal to the AL_PA of the L_Port with which this L_Port wished to communicate, and the LPSM was able to transmit the desired number of frames to the other L_Port, then this L_Port shall set ACCESS to FALSE(0) and ARB_PEND to FALSE(0); to indicate that it has met its arbitration requirement for the current access window.

If the available BB_Credit of the L_Port is zero, and the L_Port does not receive an R_RDY in less than ED_TOV, the LPSM shall transmit CLS and make the transition to the XMITTED CLOSE state.

If Idle is received and ARB_PEND is FALSE(0), the CFW shall be set to Idle and ACCESS shall be set to TRUE(1). To assure that a single Idle is not discarded for clock skew by another L_Port, at least two Idles shall be transmitted before changing the CFW to another value.

If Idle is received and ARB_PEND is TRUE(1), the CFW shall be changed to ARBx (where x equals the AL_PA of the L_Port).

If ARB is received and ARB_PEND is TRUE(1), the CFW shall be modified as follows:

- if ARB = ARB(F0) | ARBf | ARBx where x > AL_PA, the CFW shall be changed to ARBx (where x equals the AL_PA of the L_Port);
- if ARB = ARBx where x = AL_PA, the CFW shall be changed to ARBx (where x equals the AL_PA of the L_Port); or,
- if ARB = ARB(x) where x < AL_PA, the CFW shall be changed to the received ARBx.

If ARB is received and ARB_PEND is FALSE(0), the CFW shall be modified as follows:

- if ARB = ARBx where x = AL_PA, the CFW shall not be changed or
- if ARB = ARBx where x <> AL_PA, the CFW shall be changed to the received ARBx.

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

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

If DHD is received, 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. (See annex C.)

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

If MRKtx is received where the MK_TP and AL_PS match the expected values, synchronization 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 a CLS; only the L_Port which received EOFdt shall transmit CLS. (See items 8 and 18.)

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

- ARBx has been detected to indicate that another L_Port is arbitrating (the OPENED L_Port may close the Loop at a convenient time);
- frame transmission is required with a different L_Port;
- there are no more Sequences to process with the other L_Port in this Loop circuit; 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 state. (See items 2 and 22.)

If LPByx (y = AL_PA of the L_Port) or LPBfx is recognized, the LPSM shall end the current transmission (e.g., frame); set REPEAT to TRUE(1); BYPASS to TRUE(1); and, make the transition to the MONITORING state. (See items 12 and 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 INITIALIZING state. (See items 1 and 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 L_Port shall continue to operate on the Loop. 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 shall make the transition to the INITIALIZING state.

If Idle is received and ARB_PEND is FALSE(0), the CFW shall be set to Idle and ACCESS shall be set to TRUE(1). To assure that a single Idle is not discarded for clock skew by another L_Port, at least two Idles shall be transmitted before changing the CFW to another value.

If Idle is received and ARB_PEND is TRUE(1), the CFW shall be changed to ARBx (where x equals the AL_PA of the L_Port).

If ARB(F0) is received and ARB_WON is TRUE(1), the CFW shall be set to Idle. Receiving ARB(F0) indicates that no other L_Port is now arbitrating (i.e., no L_Port changed ARB(F0) to ARBx).

If ARB(F0) is received and 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 ARBx (where x equals the AL_PA of the L_Port).

If ARB is received, ARB_WON is FALSE(0), and ARB_PEND is TRUE(1), the CFW shall be modified as follows:

- if ARB = ARB(F0) | ARBf | ARBx where x > AL_PA, the CFW shall be changed to ARBx (where x equals the AL_PA of the L_Port);
- if ARB = ARBx where x = AL_PA, the CFW shall be changed to ARBx (where x equals the AL_PA of the L_Port); or,
- if ARB = ARB(x) where x < AL_PA, the CFW shall be changed to the received ARBx.

If ARB is received and ARB_PEND is FALSE(0), the CFW shall be modified as follows:

- if ARB = ARBx where x = AL_PA, the CFW shall not be changed or
- if ARB = ARBx where x <> AL_PA, the CFW shall be changed to the received ARBx.

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

If CLS is received, the LPSM shall transmit the CFW and shall make the transition to the MONITORING state. (See items 12 and 13.)

NOTE – If the L_Port had advertised a Login 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 transmitting CLS if ARB_WON = 0 or making the transition to the MONITORING state.

If MRKtx is received:

- 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, synchronization 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 state. (See items 2 and 22.)

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

| If the L_Port does not receive frames or CLS before an LP_TOV timeout occurs, the L_Port shall make the transition to the INITIALIZING state to transmit LIP(F7).

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 INITIALIZING state. (See items 1 and 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 LPSM shall set REPLICATE to FALSE(0). 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 L_Port shall transmit frames or CLS before an LP_TOV occurs. 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. When the LPSM transmits CLS (REQ(close)), the LPSM shall make the transition to the MONITORING state. (See items 12 and 13.)

NOTE — Before transmitting CLS, if the L_Port had advertised a Login 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.

The L_Port shall transmit CLS within LP_TOV.

If Idle is received and ARB_PEND is FALSE(0), the CFW shall be set to Idle and ACCESS shall be set to TRUE(1). To assure that a single Idle is not discarded for clock skew by another L_Port, at least two Idles shall be transmitted before changing the CFW to another value.

If Idle is received and ARB_PEND is TRUE(1), the CFW shall be changed to ARBx (where x equals the AL_PA of the L_Port).

If ARB(F0) is received and ARB_WON is TRUE(1), the CFW shall be set to Idle. Receiving ARB(F0) indicates that no other L_Port is now arbitrating (i.e., no L_Port changed ARB(F0) to ARBx).

If ARB(F0) is received and 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 ARBx (where x equals the AL_PA of the L_Port).

If ARB is received, ARB_WON is FALSE(0), and ARB_PEND is TRUE(1), the CFW shall be modified as follows:

- if ARB = ARB(F0) | ARBf | ARBx where x > AL_PA, the CFW shall be changed to ARBx (where x equals the AL_PA of the L_Port);
- if ARB = ARBx where x = AL_PA, the CFW shall be changed to ARBx (where x equals the AL_PA of the L_Port); or,
- if ARB = ARB(x) where x < AL_PA, the CFW shall be changed to the received ARBx.

If ARB is received and ARB_PEND is FALSE(0), the CFW shall be modified as follows:

- if ARB = ARBx where x = AL_PA, the CFW shall not be changed or
- if ARB = ARBx where x <> AL_PA, the CFW shall be changed to the received ARBx.

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

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

If LIP is recognized, the LPSM shall make the transition to the OPEN-INIT state. (See items 2 and 22.)

If LPByx (y = AL_PA of the L_Port) or LPBfx is recognized, the LPSM shall end the current transmission (e.g., frame); set REPEAT to TRUE(1); BYPASS to TRUE(1); and, make the transition to the MONITORING state. (See items 12 and 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 INITIALIZING state. (See items 1 and 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 LPSM shall set REPLICATE to FALSE(0). The L_Port shall set DUPLEX to FALSE(0) and continue to operate on the Loop. 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 shall make the transition to the INITIALIZING state.

If Idle is received, the CFW shall be set to Idle and ACCESS shall be set to TRUE(1). To assure that a single Idle is not discarded for clock skew by another L_Port, at least two Idles shall be transmitted before changing the CFW to another value.

If ARB(F0) is received, the CFW shall be set to Idle. Receiving ARB(F0) indicates that no other L_Port is now arbitrating (i.e., no L_Port changed ARB(F0) to ARBx).

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

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 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 items 12 and 13.)

NOTE — If the L_Port had advertised a Login 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:

- 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, synchronization 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 state. (See items 2 and 22.)

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

If the L_Port does not receive frames or CLS before an LP_TOV timeout occurs, the L_Port shall make the transition to the INITIALIZING state to transmit LIP(F7).

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 INITIALIZING state. (See items 1 and 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 **State 8 (INITIALIZING) actions** (table 12 and the following text describe the INITIALIZING state): The LPSM shall set REPEAT to FALSE(0); shall transmit the desired type of LIP (i.e., LIP(F7), reset LIP, or LIP(F8)); and, shall not retransmit received Transmission Words except LPB and LPE. The L_Port shall continue to transmit LIP for up to two (2) maximum AL_TIMEs (see 7.8) and monitor only for LIP, LPB, and LPE (any other Ordered Sets are discarded).

NOTE — It is recommended that at least twelve (12) words of the desired LIP type be transmitted before recognizing any received Transmission words.

During normal initialization (i.e., the L_Port is transmitting a LIP(F7) or a reset LIP) the L_Port shall react as follows:

- if any LIP is recognized, the LPSM shall make the transition to the OPEN-INIT state. (See items 2 and 22).
- if LPByx (y = AL_PA of the L_Port) or LPBfx is recognized, the LPSM shall set REPEAT and BYPASS to TRUE(1) and shall make the transition to the MONITORING state. (See items 12 and 13.) Any other LPByx shall be retransmitted.
- if LPByx or LPBfx is recognized, where x = AL_PA of the L_Port, the LPB shall be discarded. Since the transmitted LPB was received, presumably, the Loop is operational. The L_Port shall reenter the INITIALIZING state and transmit LIP(F7). See items 1 and 21.)
- if LPEyx or LPEfx is recognized, the received LPE shall be retransmitted.
- if LIP, LPByx, or LPBfx has not been recognized within two (2) maximum AL_TIMEs, if the L_Port supports the OLD-PORT state, then the L_Port shall go to the OLD-PORT state. (See items 3 and 23.) If the L_Port does not support the OLD-PORT state, the L_Port shall continue in the INITIALIZING state (see items 1 and 21) and may use the procedure outlined in annex I.2.2 to bypass a failing L_Port.

During Loop recovery (i.e., the L_Port is transmitting LIP(F8), LPB, or LPE) the L_Port shall react as follows:

- if any LIP is recognized, the LPSM shall make the transition to the OPEN-INIT state. (See items 2 and 22.)
- if LPByx or LPBfx is recognized, where x = AL_PA of the L_Port, the LPB shall be discarded. Since the transmitted LPB was received, presumably, the Loop is operational. The L_Port shall reenter the INITIALIZING state and transmit LIP(F7). See items 1 and 21.)
- if LPByx (y = AL_PA of the L_Port) or LPBfx is recognized, the LPSM shall set REPEAT and BYPASS to TRUE(1) and shall make the transition to the MONITORING state. (See items 12 and 13.)
- if any other LPByx is received, it shall be retransmitted.
- if LPEyx or LPEfx is recognized, where x = AL_PA of the L_Port, the LPE shall be replaced by LIP. Since the transmitted LPE was received, presumably, the Loop is operational. The L_Port shall reenter the INITIALIZING state and transmit LIP(F7). (See items 1 and 21.)
- if any other LPE is received, it shall be retransmitted.
- if the L_Port is unsuccessful in its attempt to recover the Loop (i.e., LIP or the transmitted LPB was not received during two (2) AL_TIMEs of each bypass attempt), the Loop is not operational (i.e., recovery is outside the scope of this standard).

If the L_Port requests to be bypassed (REQ(bypass L_Port)), the LPSM shall set REPEAT and BYPASS to TRUE(1) and shall make the transition to the MONITORING state. (See items 12 and 13.)

If the L_Port requests another L_Port to be either bypassed or enabled (REQ(bypass L_Port y) or REQ(bypass all) or REQ(enable L_Port y) or REQ(enable all)), the LPSM shall transmit LPB or LPE until the Primitive Sequence is received or the request has been dropped.

22 **State 9 (OPEN-INIT) actions** (table 13 and the following text describe the OPEN-INIT state): The L_Port shall set ACCESS to TRUE(1), shall set the CFW to Idle, shall set the "Alternate BB_Credit Management" bit to *1*, shall set BB_Credit to zero (0), shall set ARB_PEND to FALSE(0), and shall transmit at least twelve (12) LIPs of the same type as the last LIP received. Depending on the last LIP received, the following action shall be taken on entry into the OPEN-INIT state:[4]

— if the received LIP is a reset LIP (i.e., LIPyx (where y = AL_PA of this L_Port) or LIP(fx)), a vendor specified L_Port reset shall be performed (see 7.8.5); the L_Port shall ignore all LIPs for AL_TIME or until the L_Port has received or transmitted ARB(F0) as described in 10.4 to allow all existing LIPs on the Loop to be removed from the Loop. The L_Port shall continue with the initialization procedure by transmitting the LISM sequence as described in 10.4 or the L_Port shall set REPEAT to TRUE(1) and continue in the MONITORING state.

— if the received LIP is a reset LIP (i.e., LIPyx (where y is not the AL_PA of this L_Port) or LIP(F7), the L_Port shall continue with the initialization procedure by transmitting the LISM sequence as described in 10.4. The L_Port shall ignore all LIPs for AL_TIME (or until the L_Port has received or transmitted ARB(F0) as described in 10.4) to allow all existing LIPs on the Loop to be removed from the Loop.

— if the received LIP is a LIP(F8) and:

- ERR_INIT is not supported by the LPSM, the L_Port shall continue with the initialization procedure by transmitting the LISM sequence as described in 10.4. The L_Port shall ignore all LIPs for AL_TIME (or until the L_Port has received or transmitted ARB(F0) as described in 10.4) to allow all existing LIPs on the Loop to be removed from the Loop.

- ERR_INIT = FALSE(0), the L_Port shall set ERR_INIT = TRUE(1). The L_Port shall continue with the initialization procedure by transmitting the LISM sequence as described in 10.4. The L_Port shall ignore all LIPs for AL_TIME (or until the L_Port has received or transmitted ARB(F0) as described in 10.4) to allow all existing LIPs on the Loop to be removed from the Loop.

- ERR_INIT = TRUE(1), the LPSM shall transmit the received LIP(F8) for up to LP_TOV. If LIP(F7) is recognized before LP_TOV expires, the LPSM shall re-enter the OPEN-INIT state; if LP_TOV expires, the LPSM shall make the transition to the INITIALIZING state to transmit LIP(F7). (See items 1 and 21.)

During the initialization steps which are defined in 10.4, only two types of L_Ports are identified in the OPEN-INIT state:

(1) **Loop Initialization Master (LIM):** This L_Port shall transmit and receive the Loop Initialization Sequences.

(2) **Non-Loop Initialization Master:** This L_Port shall receive and retransmit the Loop Initialization Sequences.

The following common events apply to all L_Ports while in the initialization procedure in 10.4:

— if CLS is received, the initialization procedure has completed. Before making the transition to the MONITORING state (see items 12 and 13), the originator of the CLS (i.e., the LIM) shall discard the CLS; all other L_Ports shall retransmit the CLS.

— if LIP is recognized (after AL_TIME or after the L_Port has received or transmitted ARB(F0)), the LPSM shall reenter the OPEN-INIT state (i.e., transmit at least twelve (12) LIPs of the same type as the last LIP received, etc.).

---

[4] The reset LIP and LIP(F8) have been redefined from FC-AL (ANSI X3.272:1996), but interoperability should be preserved with those L_Ports that go through the initialization procedure in 10.4.

- if LPB is recognized:
    - if x = AL_PA of the L_Port, the received LPB shall be discarded.
    - if y = AL_PA of the L_Port or y = hex 'FF', the LPSM shall set REPEAT to TRUE(1); BYPASS to TRUE(1); and, make the transition to the MONITORING state. (See items 12 and 13.)
    - if y <> AL_PA of the L_Port, the received LPByx shall be retransmitted.
- if LPE is recognized:
    - if x = AL_PA of the L_Port, the received LPE shall be discarded or
    - if x <> AL_PA of the L_Port, the received LPE shall be retransmitted.

If the L_Port requests to go non-participating (REQ(nonparticipat.)), the LPSM shall set REPEAT to TRUE(1); BYPASS to TRUE(1); and, make the transition to the MONITORING state. (See items 12 and 13.)

If the L_Port requests to be bypassed (REQ(bypass L_Port)), the LPSM shall set REPEAT to TRUE(1); BYPASS to TRUE(1); and, make the transition to the MONITORING state. (See items 12 and 13).

If the L_Port requests another L_Port to be either bypassed or enabled (REQ(bypass L_Port y) or REQ(bypass all) or REQ(enable L_Port y) or REQ(enable all)), the LPSM shall transmit LPB or LPE until the Primitive Sequence is received or the request has been dropped.

If the LP_TOV timer has elapsed before an L_Port has either received or transmitted ARB(F0), the L_Port shall restart the initialization procedure by going to the INITIALIZING state to transmit LIP(F7). (See items 1 and 21.)

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 INITIALIZING state. (See items 1 and 21.)

23 **State A (OLD-PORT) actions** (table 14 and the following text describe the optional 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[5], the "Alternate BB_Credit Management" bit shall be set to *0* and the BB_Credit shall be set to one (1).

The LPSM shall monitor for LIP. If LIP is recognized, the L_Port shall implicitly Logout with the other non-L_Port and shall make the transition to the OPEN-INIT state. (See items 2 and 22.)

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 INITIALIZING state. (See items 1 and 21 and clause 10.)

---

[5] Private NL_Ports may not support a Fabric 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.

NOTES

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

2 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.

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

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 '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).

L_Port requests typically cause Transmission Words to be transmitted asynchronously to the request. Therefore, the column labeled 'OUTPUT' for L_Port requests may not relate to a Transmission Word. The notation 'None/Inst.' indicates that a Transmission Word is not transmitted in this state. The notation 'when BB_Credit' implies that the number of receive buffers equals the advertised login BB_Credit of the L_Port.

The following abbreviations are used in tables 4 through 14:

app.	appropriate
DISP	Disparity
FC-2	FC-PH 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

ARBf_SENT = 0	ARB_PEND = see below	ARB_WON = 0
DUPLEX = 0	REPLICATE = 0	CFW = N/C
DHD_RCV = 0	BYPASS = N/C	REPEAT = N/C

If ARB_PEND = 1 and REPEAT = 0, go to ARBITRATING state.
If ARB_PEND = 1 and REPEAT = 1, set ARB_PEND = 0.

INPUT	**OUTPUT**	**NEXT STATE**
LOSS of SYNC. < R_T_TOV	Idle or CFW	MONITORING
Loop Failure		
BYPASS = 0	None/Inst.	INITIALIZING
BYPASS = 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.		
FL_Port	Same Word	MONITORING
NL_Port:		
FRAME DELIMITERS		
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		
ARBf_SENT = 0	CFW = Idle	
REPEAT = 0	ACCESS = 1 (when 2 Idles sent)	
	CFW	MONITORING
REPEAT = 1	CFW	MONITORING
ARBf_SENT = 1	CFW	MONITORING
R_RDY	Same Word	MONITORING
ARB(F0)		
CFW = Idle		
REPEAT = 0	CFW	MONITORING
REPEAT = 1	CFW = ARB(F0)	
	ACCESS = 0	
	CFW	MONITORING
CFW <>Idle	CFW = ARB(F0)	
	ACCESS = 0	
	CFW	MONITORING
ARBx		
Participating		
x <>AL_PA	CFW = ARBx	
	CFW	MONITORING
x = AL_PA	CFW = Idle	
	CFW	MONITORING
Non-participating	CFW = ARBx	
	CFW	MONITORING
OPNr (OPNfr|OPNyr)		
Participating		
f = hex 'FF'		
REPEAT = 0	REPLICATE = 1	
	Same Word	MONITORING
REPEAT = 1	Same Word	MONITORING
y = AL_PA		
REPEAT = 0	REPLICATE = 1	
	Same Word	MONITORING
REPEAT = 1	Same Word	MONITORING
All other OPNr	Same Word	MONITORING
Non-participating	Same Word	MONITORING

Continued on next page

Table 4 (concluded)

OPNy		
Participating		
y = AL_PA		
REPEAT = 0	None/Inst.	OPENED
REPEAT = 1	Same Word	MONITORING
y <>AL_PA	Same Word	MONITORING
Non-participating	Same Word	MONITORING
CLS		
REPLICATE = 0	Same Word	MONITORING
REPLICATE = 1	REPLICATE = 0	
	Same Word	MONITORING
DHD	Same Word	MONITORING
MRKtx		
x = AL_PA	CFW	MONITORING
x <>AL_PA	Same Word	MONITORING
PRIMITIVE SEQUENCES		
LIP		
REPEAT = 0	None/Inst.	OPEN-INIT
REPEAT = 1	Same Word	non-participating
		MONITORING
LPB (LPByx|LPBfx)		
x = AL_PA	CFW	MONITORING
y <>AL_PA	Same Word	MONITORING
y = AL_PA	REPLICATE = 0	
	REPEAT = 1	
	BYPASS = 1	
	Same Word	MONITORING
f = hex 'FF'	REPLICATE = 0	
	REPEAT = 1	
	BYPASS = 1	
	Same Word	MONITORING
LPE (LPEyx|LPEfx)		
x = AL_PA	CFW	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
L_PORT CONTROLS		
REQ(monitor)	None/Inst.	MONITORING
REQ(arbitrate as x)		
ACCESS = 0	None/Inst.	MONITORING
ACCESS = 1	None/Inst.	ARBITRATING
REQ(arbitrate FF)		
CFW = Idle	When 6 Idles sent	
	CFW = ARBf	
	ARBf_SENT = 1	MONITORING
CFW <>Idle	None/Inst.	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.	MONITORING
REQ(participating)	None/Inst.	INITIALIZING
REQ(nonparticipat.)	Opt. Transmit LIPs	
	REPEAT = 1	MONITORING
REQ(mark as tx)		
REPEAT = 0	MRKtx at the next Fill Word	MONITORING
REPEAT = 1	None/Inst.	MONITORING
REQ(bypass L_Port)	REPLICATE = 0	
	REPEAT = 1	
	BYPASS = 1	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.	INITIALIZING

Table 5 — ARBITRATING (State 1) transitions

	ENTRY ACTIONS	
ACCESS = N/C	ARB_PEND = N/C	ARB_WON = N/C
DUPLEX = 0	REPLICATE = N/C	CFW = N/C
DHD_RCV = 0	BYPASS = N/C	REPEAT = N/C

INPUT	**OUTPUT**	**NEXT STATE**
LOSS of SYNC. < R_T_TOV	Idle or CFW	ARBITRATING
Loop Failure	None/Inst.	INITIALIZING
INVALID TRANS. WORD	CFW	ARBITRATING
RUNNING DISP at O.S.	CFW	ARBITRATING
ELASTICITY WORD REQd	CFW	ARBITRATING
VALID DATA WORD		
FL_Port	Same Word	ARBITRATING
NL_Port:		
REPLICATE = 0	Same Word	ARBITRATING
REPLICATE = 1	Receive Word	
	Same Word	ARBITRATING
VALID TRANS. WORD =O.S.		
FL_Port	Same Word	ARBITRATING
NL_Port:		
FRAME DELIMITERS		
SOFxx		
REPLICATE = 0	Same Word	ARBITRATING
REPLICATE = 1	Receive Word	
	Same Word	ARBITRATING
EOFxx		
REPLICATE = 0	Same Word	ARBITRATING
REPLICATE = 1	Receive Word	
	Same Word	ARBITRATING
PRIMITIVE SIGNALS		
Idle	CFW = own ARBx	
	after xmit 2 Idles	
	ARB_PEND = 1	
	CFW	ARBITRATING
R_RDY	Same Word	ARBITRATING
ARBx		
x < AL_PA	CFW = ARBx	
	CFW	ARBITRATING
x > AL_PA	CFW = own ARBx	
	ARB_PEND = 1	
	CFW	ARBITRATING
x = AL_PA	None/Inst.	ARBITRATION WON
OPNr (OPNfr|OPNyr)		
f = hex 'FF'	REPLICATE = 1	
	Same Word	ARBITRATING
y = AL_PA	REPLICATE = 1	
	Same Word	ARBITRATING
All other OPNr	Same Word	ARBITRATING
OPNy		
y <>AL_PA	Same Word	ARBITRATING
y = AL_PA	None/Inst.	OPENED
CLS		
REPLICATE = 0	Same Word	ARBITRATING
REPLICATE = 1	REPLICATE = 0	
	Same Word	ARBITRATING
DHD	Same Word	ARBITRATING
MRKtx		
x = AL_PA	CFW	ARBITRATING
x <>AL_PA	Same Word	ARBITRATING

Continued on next page

Table 5 (concluded)

PRIMITIVE SEQUENCES		
LIP	None/Inst.	OPEN-INIT
LPB (LPByx|LPBfx)		
X = AL_PA	CFW	ARBITRATING
y <>AL_PA	Same Word	ARBITRATING

y = AL_PA	REPEAT = 1
	BYPASS = 1
	ARB_PEND = 0
	None/Inst.
		MONITORING

f = hex 'FF'	REPEAT = 1
	BYPASS = 1
	ARB_PEND = 0
	None/Inst.
		MONITORING

LPE (LPEyx|LPEfx)	Same Word	ARBITRATING
ANY OTHER O.S.	Same Word	ARBITRATING
L_PORT CONTROLS		
REQ(monitor)	None/Inst.	ARBITRATING
REQ(arbitrate as x)	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.	INITIALIZING

Table 6 — ARBITRATION WON (State 2) transitions

ENTRY ACTIONS

ACCESS = N/C	ARB_PEND = N/C	ARB_WON = N/C
DUPLEX = N/C	REPLICATE = N/C	CFW = N/C
DHD_RCV = N/C	BYPASS = N/C	REPEAT = N/C

If REPLICATE = 1, CFW = ARB(F0), transmit CLS, and go to TRANSFER state.

INPUT	**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	N/A	N/A
EOFxx	N/A	N/A
PRIMITIVE SIGNALS		
Idle	N/A	N/A
R_RDY	N/A	N/A
ARBx	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
L_PORT CONTROLS		
REQ(monitor)	N/A	N/A
REQ(arbitrate as x)	N/A	N/A
REQ(open yx) f-d	OPNyx	OPEN
REQ(open yy) h-d	OPNyy	OPEN
REQ(open fr)	OPNfr; REPLICATE=1	OPEN
REQ(open yr)	OPNyr; REPLICATE=1	OPEN
REQ(close)	OPNyy; y = AL_PA	OPEN
REQ(send DHD)	None/Inst.	OPEN
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.	INITIALIZING

Table 7 — OPEN (State 3) transitions

ENTRY ACTIONS		
ACCESS = see below	ARB_PEND = 0	ARB_WON = 1
DUPLEX = 1	REPLICATE = N/C	CFW = ARB(F0)
DHD_RCV = 0	BYPASS = N/C	REPEAT = N/C

If using fairness, ACCESS = 0
If not using fairness, ACCESS = 1
Transmit six (6) CFWs and R_RDY (see 8.3.5)

INPUT	**OUTPUT**	**NEXT STATE**
LOSS of SYNC. < R_T_TOV	FC-2 FP/PSig/PSeq	OPEN
Loop Failure	None/Inst.	INITIALIZING
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	FC-2 FP/PSig/PSeq	OPEN
EOFxx	FC-2 FP/PSig/PSeq	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 FC-2 FP/PSig/PSeq	OPEN
ARBx	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
LPB (LPByx|LPBfx)		
x = AL_PA	FC-2 FP/PSig/PSeq	OPEN
y <>AL_PA	FC-2 FP/PSig/PSeq	OPEN

y = AL_PA	REPEAT = 1
	BYPASS = 1
	None/Inst.
		MONITORING

f = hex 'FF'	REPEAT = 1
	BYPASS = 1
	None/Inst.
		MONITORING

LPE (LPEyx|LPEfx)	FC-2 FP/PSig/PSeq	OPEN
ANY OTHER O.S.	FC-2 FP/PSig/PSeq	OPEN

Continued on next page

Table 7 (concluded)

L_PORT CONTROLS		
REQ(monitor)	None/Inst.	OPEN
REQ(arbitrate as x)	None/Inst.	OPEN
REQ(open yx) f-d	None/Inst.	OPEN
REQ(open yy) h-d	None/Inst.	OPEN
REQ(open fr)		OPEN
REPLICATE = 0	None/Inst.	OPEN
REPLICATE = 1	OPNfr at the next app. Fill Word	OPEN
REQ(open yr)		
REPLICATE = 0	None/Inst.	OPEN
REPLICATE = 1	OPNyr at the next app. Fill Word	OPEN
REQ(close)	CLS at the next app. Fill Word	XMITTED CLOSE
REQ(send DHD)	DHD at the next app. Fill Word	OPEN
REQ(transfer)		
ACCESS = 0	CLS at the next app. Fill Word	XMITTED CLOSE
ACCESS = 1	CLS at the next app. Fill Word	TRANSFER
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.	INITIALIZING

Table 8 — OPENED (State 4) transitions

ENTRY ACTIONS

ACCESS = N/C	ARB_PEND = N/C	ARB_WON = 0
DUPLEX = see below	REPLICATE = N/C	CFW = N/C
DHD_RCV = 0	BYPASS = N/C	REPEAT = N/C

If OPNyx, DUPLEX = 1
If OPNyy, DUPLEX = 0
Transmit six (6) CFWs and R_RDY (see 8.3.5) before CLS.

INPUT	**OUTPUT**	**NEXT STATE**
LOSS of SYNC. < R_T_TOV	FC-2 FP/PSig/PSeq	OPENED
Loop Failure	None/Inst.	INITIALIZING
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	FC-2 FP/PSig/PSeq	OPENED
EOFxx	FC-2 FP/PSig/PSeq	OPENED
PRIMITIVE SIGNALS		
Idle		

ARB_PEND = 0	CFW = Idle
	ACCESS = 1 (when 2 Idles transmitted)
	FC-2 FP/PSig/PSeq
		OPENED

ARB_PEND = 1	CFW = own ARBx FC-2 FP/PSig/PSeq	OPENED
R_RDY	FC-2 FP/PSig/PSeq	OPENED
ARBx		
ARB_PEND = 0		
x <>AL_PA	CFW = ARBx FC-2 FP/PSig/PSeq	OPENED
x = AL_PA	FC-2 FP/PSig/PSeq	OPENED
ARB_PEND = 1		
x > AL_PA	CFW = own ARBx FC-2 FP/PSig/PSeq	OPENED
x = AL_PA	CFW = own ARBx FC-2 FP/PSig/PSeq	OPENED OPENED
x < AL_PA	CFW = ARBx FC-2 FP/PSig/PSeq	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
LPB (LPByx|LPBfx)		
y <>AL_PA	FC-2 FP/PSig/PSeq	OPENED

y = AL_PA	REPEAT = 1
	BYPASS = 1
	None/Inst.
		MONITORING

f = hex 'FF'	REPEAT = 1
	BYPASS = 1
	None/Inst.
		MONITORING

LPE (LPEyx|LPEfx)	FC-2 FP/PSig/PSeq	OPENED
ANY OTHER O.S.	FC-2 FP/PSig/PSeq	OPENED

Continued on next page

Table 8 (concluded)

L_PORT CONTROLS		
REQ(monitor)	None/Inst.	OPENED
REQ(arbitrate as x)	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
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.	INITIALIZING

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
REPEAT = N/C

INPUT	**OUTPUT**	**NEXT STATE**
LOSS of SYNC. < R_T_TOV	Idle or CFW	XMITTED CLOSE
Loop Failure	None/Inst.	INITIALIZING
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	CFW = Idle
	ACCESS = 1 (when 2 Idles transmitted)
	CFW
		XMITTED CLOSE

ARB_PEND = 1	CFW = own ARBx CFW	XMITTED CLOSE
R_RDY	CFW	XMITTED CLOSE
ARB(F0)		
ARB_WON = 1	CFW = Idle CFW	XMITTED CLOSE
ARB_WON = 0		
ARB_PEND = 0	CFW = ARB(F0) CFW	XMITTED CLOSE
ARB_PEND = 1	CFW = own ARBx CFW	XMITTED CLOSE
ARBx		
ARB_WON = 1	CFW	XMITTED CLOSE
ARB_WON = 0		
ARB_PEND = 0		
x <>AL_PA	CFW = ARBx CFW	XMITTED CLOSE
x = AL_PA	CFW	
ARB_PEND = 1		
x > AL_PA	CFW = own ARBx CFW	XMITTED CLOSE
x = AL_PA	CFW = own ARBx CFW	XMITTED CLOSE
x < AL_PA	CFW = ARBx CFW	XMITTED CLOSE
OPNr	CFW	XMITTED CLOSE
OPNy	CFW	XMITTED CLOSE
CLS	CFW	MONITORING 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
LPB (LPByx|LPBfx)		
y <>AL_PA	CFW	XMITTED CLOSE

y = AL_PA	REPEAT = 1
	BYPASS = 1
	None/Inst.
		MONITORING

f = hex 'FF'	REPEAT = 1
	BYPASS = 1
	None/Inst.
		MONITORING

LPE (LPEyx|LPEfx)	CFW	XMITTED CLOSE
ANY OTHER O.S.	CFW	XMITTED CLOSE

Continued on next page

Table 9 (concluded)

L_PORT CONTROLS		
REQ(monitor)	None/Inst.	MONITORING
REQ(arbitrate as x)	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.	INITIALIZING

Table 10 — RECEIVED CLOSE (State 6) transitions

ENTRY ACTIONS

ACCESS = N/C   ARB_PEND = N/C   ARB_WON = N/C
DUPLEX = N/C   REPLICATE = N/C   CFW = N/C
DHD_RCV = N/C   BYPASS = N/C   REPEAT = N/C

INPUT	OUTPUT	NEXT STATE
LOSS of SYNC. < R_T_TOV	FC-2 FP/PSig/PSeq	RECEIVED CLOSE
Loop Failure	None/Inst.	INITIALIZING
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	CFW = Idle
	ACCESS = 1 (when 2 Idles transmitted)
	FC-2 FP/PSig/PSeq
		RECEIVED CLOSE

ARB_PEND = 1	CFW = own ARBx FC-2 FP/PSig/PSeq	RECEIVED CLOSE
R_RDY	FC-2 FP/PSig/PSeq	RECEIVED CLOSE
ARB(F0)		
ARB_WON = 1	CFW = Idle FC-2 FP/PSig/PSeq	RECEIVED CLOSE
ARB_WON = 0		
ARB_PEND = 0	CFW = ARB(F0) FC-2 FP/PSig/PSeq	RECEIVED CLOSE
ARB_PEND = 1	FC-2 FP/PSig/PSeq	RECEIVED CLOSE
ARBx		
ARB_WON = 1	FC-2 FP/PSig/PSeq	RECEIVED CLOSE
ARB_WON = 0		
ARB_PEND = 0		
x <>AL_PA	CFW = ARBx FC-2 FP/PSig/PSeq	RECEIVED CLOSE
x = AL_PA	FC-2 FP/PSig/PSeq	RECEIVED CLOSE
ARB_PEND = 1		
x > AL_PA	CFW = own ARBx FC-2 FP/PSig/PSeq	RECEIVED CLOSE
x = AL_PA	CFW = own ARBx FC-2 FP/PSig/PSeq	RECEIVED CLOSE
x < AL_PA	CFW = ARBx FC-2 FP/PSig/PSeq	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
LPB (LPByx|LPBfx)		
y <>AL_PA	FC-2 FP/PSig/PSeq	RECEIVED CLOSE

y = AL_PA	REPEAT = 1
	BYPASS = 1
	None/Inst.
		MONITORING

f = hex 'FF'	REPEAT = 1
	BYPASS = 1
	None/Inst.
		MONITORING

LPE (LPEyx|LPEfx)	FC-2 FP/PSig/PSeq	RECEIVED CLOSE
ANY OTHER O.S.	FC-2 FP/PSig/PSeq	RECEIVED CLOSE

Continued on next page

Table 10 (concluded)

L_PORT CONTROLS		
REQ(monitor)	None/Inst.	RECEIVED CLOSE
REQ(arbitrate as x)	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)	CLS at the next app. Fill Word When BB_Credit and CLS sent	MONITORING
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.	INITIALIZING

Table 11 — TRANSFER (State 7) transitions

ENTRY ACTIONS

ACCESS = N/C
ARB_PEND = N/C
ARB_WON = N/C
DUPLEX = 0
REPLICATE = 0
CFW = N/C
DHD_RCV = N/C
BYPASS = N/C
REPEAT = N/C

INPUT	OUTPUT	NEXT STATE
LOSS of SYNC. < R_T_TOV	Idle or CFW	TRANSFER
Loop Failure	None/Inst.	INITIALIZING
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 (when 2 Idles transmitted)	
	CFW	TRANSFER
R_RDY	CFW	TRANSFER
ARB(F0)	CFW = Idle	
	CFW	TRANSFER
ARBx	CFW	TRANSFER
OPNr	CFW	TRANSFER
OPNy	CFW	TRANSFER
CLS	None/Inst.	
	When BB_Credit	OPEN/MONITORING (L_PORT CONTROLS)
DHD	CFW	TRANSFER
MRKtx		
x = AL_PA	CFW	TRANSFER
x <>AL_PA	Same Word	TRANSFER
PRIMITIVE SEQUENCES		
LIP	None/Inst.	OPEN-INIT
LPB (LPByx|LPBfx)		
X = AL_PA	CFW	TRANSFER
y <>AL_PA	CFW	TRANSFER
y = AL_PA	REPEAT = 1	
	BYPASS = 1	
	None/Inst.	MONITORING
f = hex 'FF'	REPEAT = 1	
	BYPASS = 1	
	None/Inst.	MONITORING
LPE (LPEyx|LPEfx)	CFW	TRANSFER
ANY OTHER O.S.	CFW	TRANSFER

Continued on next page

Table 11 (concluded)

L_PORT CONTROLS		
REQ(monitor)	When CLS received and BB_Credit	MONITORING
REQ(arbitrate as x)	None/Inst.	TRANSFER
REQ(open yx) f-d	When CLS received and BB_Credit OPNyx	OPEN
REQ(open yy) h-d	When CLS received and BB_Credit OPNyy	OPEN
REQ(open fr)	When CLS received and BB_Credit OPNfr; REPLICATE=1	OPEN
REQ(open yr)	When CLS received and BB_Credit OPNyr; REPLICATE=1	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.	INITIALIZING

Table 12 — INITIALIZING (State 8) transitions

ENTRY ACTIONS

ACCESS = N/C
ARB_PEND = N/C
ARB_WON = N/C
DUPLEX = N/C
REPLICATE = N/C
CFW = N/C
DHD_RCV = N/C
BYPASS = 0
REPEAT = 0
Transmit requested LIPs

INPUT	**OUTPUT**	**NEXT STATE**
LOSS of SYNC. < R_T_TOV	LIP	INITIALIZING
Loop Failure	LIP	INITIALIZING
INVALID TRANS. WORD	LIP	INITIALIZING
RUNNING DISP at O.S.	LIP	INITIALIZING
ELASTICITY WORD REQd	N/A	INITIALIZING
VALID DATA WORD	LIP	INITIALIZING
VALID TRANS. WORD =O.S.		
FRAME DELIMITERS		
SOFxx	LIP	INITIALIZING
EOFxx	LIP	INITIALIZING
PRIMITIVE SIGNALS		
Idle	LIP	INITIALIZING
R_RDY	LIP	INITIALIZING
ARBx	LIP	INITIALIZING
OPNr	LIP	INITIALIZING
OPNy	LIP	INITIALIZING
CLS	LIP	INITIALIZING
DHD	LIP	INITIALIZING
MRKtx	LIP	INITIALIZING
PRIMITIVE SEQUENCES		
LIP	None/Inst.	OPEN-INIT
LPB (LPByx|LPBfx)		
x = AL_PA	LIP	INITIALIZING
y <>AL_PA	Same Word	INITIALIZING

y = AL_PA	REPEAT = 1
	BYPASS = 1
	None/Inst.
		MONITORING

f = hex 'FF'	REPEAT = 1
	BYPASS = 1
	None/Inst.
		MONITORING

LPE (LPEyx|LPEfx)		
x = AL_PA	LIP	INITIALIZING
x <>AL_PA	Same Word	INITIALIZING
ANY OTHER O.S.	LIP	INITIALIZING
L_PORT CONTROLS		
REQ(monitor)	None/Inst.	INITIALIZING
REQ(arbitrate as x)	None/Inst.	INITIALIZING
REQ(open yx) f-d	None/Inst.	INITIALIZING
REQ(open yy) h-d	None/Inst.	INITIALIZING
REQ(open fr)	None/Inst.	INITIALIZING
REQ(open yr)	None/Inst.	INITIALIZING
REQ(close)	None/Inst.	INITIALIZING
REQ(send DHD)	None/Inst.	INITIALIZING
REQ(transfer)	None/Inst.	INITIALIZING
REQ(old-port)	None/Inst.	OLD-PORT
REQ(participating)	None/Inst.	INITIALIZING
REQ(nonparticipat.)	None/Inst.	INITIALIZING
REQ(mark as tx)	None/Inst.	INITIALIZING

REQ(bypass L_Port)	REPEAT = 1
	BYPASS = 1
		MONITORING
		MONITORING

REQ(bypass L_Port y)	LPByx	INITIALIZING
REQ(bypass all)	LPBfx	INITIALIZING
REQ(enable L_Port)	None/Inst.	INITIALIZING
REQ(enable L_Port y)	LPEyx	INITIALIZING
REQ(enable all)	LPEfx	INITIALIZING
REQ(initialize)	None/Inst.	INITIALIZING

Table 13 — OPEN-INIT (State 9) transitions

ENTRY ACTIONS

ACCESS = 1
ARB_PEND = 0
ARB_WON = 0
DUPLEX = 0
REPLICATE = 0
CFW = Idle
DHD_RCV = 0
BYPASS = 0
REPEAT = 0
ERR_INIT = see below
Alternate BB_Credit = 1
BB_Credit = 0

Transmit at least twelve (12) received LIPs.
If LIP(F8) and ERR_INIT = 0, ERR_INIT = 1, begin clause 10.
If LIP(F8) and ERR_INIT = 1, transmit LIP(F8) for LP_TOV or until LIP(F7) recognized. If LP_TOV timeout and ERR_INIT = 1, go to INITIALIZING and transmit LIP(F7). If LP_TOV timeout and ERR_INIT = 0, begin clause 10.

INPUT	OUTPUT	NEXT STATE
LOSS of SYNC. < R_T_TOV	FC-2 FP/PSig/PSeq	OPEN-INIT
Loop Failure	None/Inst.	INITIALIZING
INVALID TRANS. WORD	FC-2 FP/PSig/PSeq	OPEN-INIT
RUNNING DISP at O.S.	FC-2 FP/PSig/PSeq	OPEN-INIT
ELASTICITY WORD REQd	N/A	OPEN-INIT
VALID DATA WORD	FC-2 FP/PSig/PSeq	OPEN-INIT
VALID TRANS. WORD =O.S.		
FRAME DELIMITERS		
SOFxx	See clause 10	OPEN-INIT
EOFxx	See clause 10	OPEN-INIT
PRIMITIVE SIGNALS		
Idle	FC-2 FP/PSig/PSeq	OPEN-INIT
R_RDY	FC-2 FP/PSig/PSeq	OPEN-INIT
ARB(F0)		
(LIM)	FC-2 FP/PSig/PSeq	OPEN-INIT
(other L_Ports)	ARB(F0)	OPEN-INIT
ARBx	CFW	OPEN-INIT
OPNr	FC-2 FP/PSig/PSeq	OPEN-INIT
OPNy	FC-2 FP/PSig/PSeq	OPEN-INIT
CLS (LIM)	CFW	MONITORING
(other L_Ports)	CLS at the next app. Fill Word None/Inst.	MONITORING
DHD	FC-2 FP/PSig/PSeq	OPEN-INIT
MRKtx	FC-2 FP/PSig/PSeq	OPEN-INIT
PRIMITIVE SEQUENCES		
LIP	None/Inst.	Reenter OPEN-INIT
LPB (LPByx|LPBfx)		
x = AL_PA	FC-2 FP/PSig/PSeq	OPEN-INIT
y <>AL_PA	Same Word	OPEN-INIT

y = AL_PA	REPEAT = 1
	BYPASS = 1
	None/Inst.
		MONITORING

f = hex 'FF'	REPEAT = 1
	BYPASS = 1
	None/Inst.
		MONITORING

LPE (LPEyx|LPEfx)		
x = AL_PA	FC-2 FP/PSig/PSeq	OPEN-INIT
x <>AL_PA	Same Word	OPEN-INIT
ANY OTHER O.S.	See clause 10	OPEN-INIT

Continued on next page

Table 13 (concluded)

L_PORT CONTROLS		
REQ(monitor)	None/Inst.	OPEN-INIT
REQ(arbitrate as x)	None/Inst.	OPEN-INIT
REQ(open yx) f-d	None/Inst.	OPEN-INIT
REQ(open yy) h-d	None/Inst.	OPEN-INIT
REQ(open fr)	None/Inst.	OPEN-INIT
REQ(open yr)	None/Inst.	OPEN-INIT
REQ(close)		
non-master	None/Inst.	OPEN-INIT
master	CLS	OPEN-INIT
REQ(send DHD)	None/Inst.	OPEN-INIT
REQ(transfer)	None/Inst.	OPEN-INIT
REQ(old-port)	None/Inst.	OPEN-INIT
REQ(participating)	None/Inst.	OPEN-INIT
REQ(nonparticipat.)	REPEAT = 1	MONITORING
REQ(mark as tx)	None/Inst.	OPEN-INIT
REQ(bypass L_Port)	REPEAT = 1	
	BYPASS = 1	MONITORING
REQ(bypass L_Port y)	LPByx	OPEN-INIT
REQ(bypass all)	LPBfx	OPEN-INIT
REQ(enable L_Port)	None/Inst.	OPEN-INIT
REQ(enable L_Port y)	LPEyx	OPEN-INIT
REQ(enable all)	LPEfx	OPEN-INIT
REQ(initialize)	None/Inst.	INITIALIZING

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

ENTRY ACTIONS

ACCESS = N/C
ARB_PEND = N/C
ARB_WON = N/C
DUPLEX = N/C
REPLICATE = N/C
CFW = Idle
DHD_RCV = N/C
BYPASS = N/C
REPEAT = N/C
Alternate BB_Credit = 0
BB_Credit = 1

INPUT	OUTPUT	NEXT STATE
LOSS of SYNC. < R_T_TOV	FC-2 FP/PSig/PSeq	OLD-PORT
Loop Failure	None/Inst.	INITIALIZING
INVALID TRANS. WORD	FC-2 FP/PSig/PSeq	OLD-PORT
RUNNING DISP at O.S.	FC-2 FP/PSig/PSeq	OLD-PORT
ELASTICITY WORD REQd	N/A	OLD-PORT
VALID DATA WORD	FC-2 FP/PSig/PSeq	OLD-PORT
VALID TRANS. WORD =O.S.		
FRAME DELIMITERS		
SOFxx	FC-2 FP/PSig/PSeq	OLD-PORT
EOFxx	FC-2 FP/PSig/PSeq	OLD-PORT
PRIMITIVE SIGNALS		
Idle	FC-2 FP/PSig/PSeq	OLD-PORT
R_RDY	FC-2 FP/PSig/PSeq	OLD-PORT
ARBx	FC-2 FP/PSig/PSeq	OLD-PORT
OPNr	FC-2 FP/PSig/PSeq	OLD-PORT
OPNy	FC-2 FP/PSig/PSeq	OLD-PORT
CLS	FC-2 FP/PSig/PSeq	OLD-PORT
DHD	FC-2 FP/PSig/PSeq	OLD-PORT
MRKtx	FC-2 FP/PSig/PSeq	OLD-PORT
PRIMITIVE SEQUENCES		
NOS	OLS	OLD-PORT
OLS	LR	OLD-PORT
LR	LRR	OLD-PORT
LRR	Idle	OLD-PORT
LIP	None/Inst.	OPEN-INIT
LPB (LPByx|LPBfx)	FC-2 FP/PSig/PSeq	OLD-PORT
LPE (LPEyx|LPEfx)	FC-2 FP/PSig/PSeq	OLD-PORT
ANY OTHER O.S.	FC-2 FP/PSig/PSeq	OLD-PORT
L_PORT CONTROLS		
REQ(monitor)	None/Inst.	OLD-PORT
REQ(arbitrate as x)	None/Inst.	OLD-PORT
REQ(open yx) f-d	None/Inst.	OLD-PORT
REQ(open yy) h-d	None/Inst.	OLD-PORT
REQ(open fr)	None/Inst.	OLD-PORT
REQ(open yr)	None/Inst.	OLD-PORT
REQ(close)	None/Inst.	OLD-PORT
REQ(send DHD)	None/Inst.	OLD-PORT
REQ(transfer)	None/Inst.	OLD-PORT
REQ(old-port)	None/Inst.	OLD-PORT
REQ(participating)	None/Inst.	OLD-PORT
REQ(nonparticipat.)	None/Inst.	OLD-PORT
REQ(mark as tx)	None/Inst.	OLD-PORT
REQ(bypass L_Port)	None/Inst.	OLD-PORT
REQ(bypass L_Port y)	None/Inst.	OLD-PORT
REQ(bypass all)	None/Inst.	OLD-PORT
REQ(enable L_Port)	None/Inst.	OLD-PORT
REQ(enable L_Port y)	None/Inst.	OLD-PORT
REQ(enable all)	None/Inst.	OLD-PORT
REQ(initialize)	None/Inst.	INITIALIZING

10 Loop Initialization procedure

Loop Initialization is a logical procedure used by an L_Port to determine its environment and possibly 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 10.4): 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 22 and table 13):

— 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 (while in the OPEN-INIT state), 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.

— If a non-L_Port is attached point-to-point to the L_Port, the L_Port may complete the initialization procedure in 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 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 device 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 shall enter 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 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 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. An 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 participating mode it attempts Login (if required in 10.4.3, 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 an NL_Port exits the initialization procedure in participating mode and detects that an FL_Port is not in 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).

  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 INITIALIZING 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 ARBx (where x 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.

An L_Port shall enter the OPEN-INIT 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 6 (see the end of 10.4) provides a 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 the following subclauses are identified in the upper right-hand corner of selected process blocks.

10.3 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 INITIALIZING state (REQ(initialize)) (see 8.4.3, item 21). If the L_Port recognizes LIP, it shall transfer to and remain in the OPEN-INIT state (LIP received) until the initialization procedure has completed or unless directed otherwise by the L_Port. (See 8.4.3.) If the LIP is not recognized within two (2) AL_TIME periods (and all attempts to recover from a Loop Failure have failed), the LPSM shall make the transition to the INITIALIZING state or if supported, shall make the transition to the OLD-PORT state.

10.4 L_Port initialization

After the L_Port has performed the entry functions for the OPEN-INIT state (see 8.4.3, item 22), the L_Port shall continue the initialization procedure as defined in 10.4.3, steps (1) to (6). This initialization procedure shall use the Loop Initialization Sequences as defined in figure 4.

10.4.1 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.[6]

```
Payload                      - 12, 20, or 132 bytes
┌─────────┬──────────────────┐
│         │  8-byte Port_Name  │
│  LI_ID  ├──────────────────────────────┐
│   and   │ 16-byte AL_PA bit map            │
│  LI_FL  ├──────────────────────────────┴──────────//──────────┐
│         │ 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

The FC-PH rules for valid frames apply to the Loop Initialization Sequences which are shown in figure 4. However, the frame header shall not be used to validate the Loop Initialization Sequences. If an L_Port detects frames which contain code violations or CRC errors, the frames shall be discarded. The D_ID and S_ID shall be checked only in the LISM frame during selection of the LIM.

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

---

[6] 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'.

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.

10.4.2 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 Map Word	Bit	AL_PA (hex)	Bit Map Word	Bit	AL_PA (hex)	Bit Map Word	Bit	AL_PA (hex)	Bit Map Word	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

10.4.3 Loop Initialization steps

The following initialization steps are performed unless one of the common events identified in 8.4.3, item 22, occurs. Until the LISA frame has been processed, AL_PAs may be unstable and any Primitive Sequence (e.g., LPByx) may not be acted upon by the desired L_Port.

1) Select initial AL_PA

Each FL_Port shall choose an initial value for its AL_PA of hex '00'.

Each NL_Port shall choose an initial value for its AL_PA of hex 'EF'.

Because of these initial AL_PA values, until the L_Port establishes a new AL_PA in step (4), only FL_Ports and NL_Ports can be uniquely distinguished with an LPByx. If an L_Port has a trusted AL_PA (from a previous initialization attempt or from a hard address) 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.

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 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 initialization).

The L_Port shall transmit its own Loop Initialization Sequence (LI_ID='LISM') until a valid Loop Initialization Sequence (LI_ID='LISM') is received whose D_ID, S_ID, and Port_Name compare 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).

3) Wait for a Loop Initialization master

The L_Port shall retransmit valid received frames until the L_Port receives ARB(F0). An L_Port shall wait a minimum of LP_TOV for ARB(F0). If the timer expires before ARB(F0) is received (no LIM was selected), the L_Port shall make the transition to the INITIALIZING state to transmit LIP(F7).

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. If the timer expires before ARB(F0) is received, the LIM shall make the transition to the INITIALIZING 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 figures 4 and 5 and table 15).

	Bits
Word	3322 2222 2222 1111 1111 11 1098 7654 3210 9876 5432 1098 7654 3210
0	L000 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. If the LIM is an FL_Port, it shall set the bit associated with AL_PA hex '00'. The L_bit shall also be set if this is the first initialization attempt by an FL_Port or 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 (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 (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 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 at 10.3 at the request of the Node).

c) When the LIM receives the LISA Sequence, it 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', shall enter an offset of hex '01', followed by its AL_PA (if acquired). For example, if AL_PA = hex '05', the AL_PA position map contains hex '0105FFFFFF...FF'. If the LIM is in the non-participating mode, the AL_PA position map that the LIM originates contains hex '00FF...FF'.

LILP The LIM shall transmit the AL_PA position map of the previous Loop Initialization Sequence (LI_ID='LIRP').

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 Loop Initialization Sequences, the L_Port shall make the transition to the INITIALIZING state to transmit LIP(F7).

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

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)s and shall prepare 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 figures 4 and 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; 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 (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 (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 the 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). If a bit was available, the L_Port shall adjust its AL_PA according to which bit was set. If no bits are available, the L_Port shall go to the non-participating mode. The L_Port shall retransmit the Loop Initialization Sequence.

LIRP If LIRP is received, the L_Port 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 L_Port shall retransmit the Loop Initialization Sequence.

If the L_Port detects an invalid Loop Initialization Sequences, the L_Port shall make the transition to the INITIALIZING state to transmit LIP(F7).

The L_Port shall use the LP_TOV timer to wait for each Loop Initialization Sequence and the CLS. If the timer expires before each Loop Initialization Sequence or CLS is received, the L_Port shall make the transition to the INITIALIZING 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 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 10.3 at the request of the Node).

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., it 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 INITIALIZING 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 INITIALIZING 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 INITIALIZING state (REQ(initialize)) to re-initialize and acquire the Fabric Assigned AL_PA value.

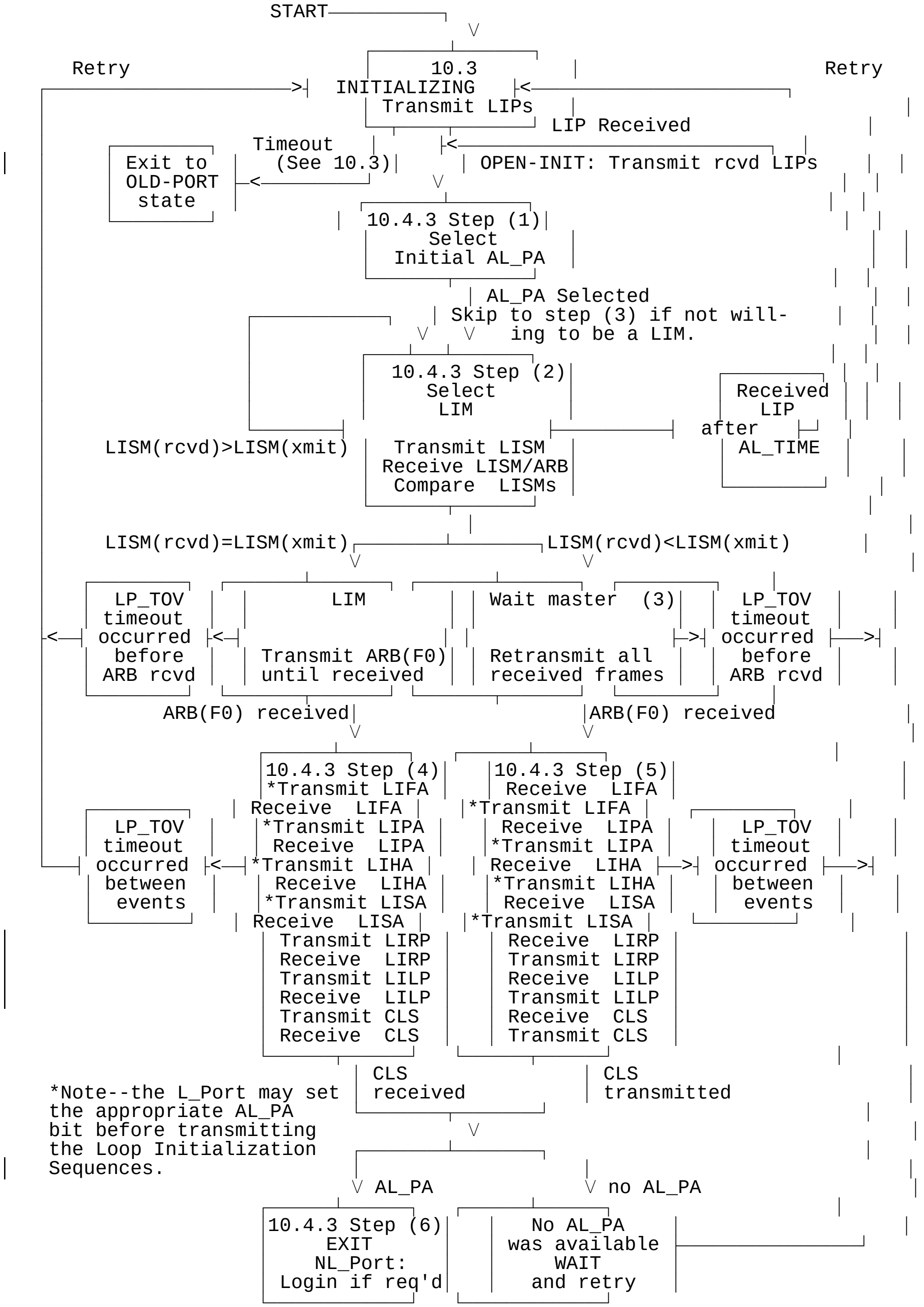

START
Retry
10.3 INITIALIZING Transmit LIPs
Retry
LIP Received
Timeout (See 10.3)
Exit to OLD-PORT state
OPEN-INIT: Transmit rcvd LIPs
10.4.3 Step (1) Select Initial AL_PA
AL_PA Selected
Skip to step (3) if not willing to be a LIM.
10.4.3 Step (2) Select LIM
Received LIP after AL_TIME
LISM(rcvd)>LISM(xmit)
Transmit LISM Receive LISM/ARB Compare LISMs
LISM(rcvd)=LISM(xmit)
LISM(rcvd)<LISM(xmit)
LP_TOV timeout occurred before ARB rcvd
LIM
Transmit ARB(F0) until received
Wait master (3)
Retransmit all received frames
LP_TOV timeout occurred before ARB rcvd
ARB(F0) received
ARB(F0) received
10.4.3 Step (4) *Transmit LIFA Receive LIFA *Transmit LIPA Receive LIPA *Transmit LIHA Receive LIHA *Transmit LISA Receive LISA Transmit LIRP Receive LIRP Transmit LILP Receive LILP Transmit CLS Receive CLS
10.4.3 Step (5) Receive LIFA *Transmit LIFA Receive LIPA *Transmit LIPA Receive LIHA *Transmit LIHA Receive LISA *Transmit LISA Receive LIRP Transmit LIRP Receive LILP Transmit LILP Receive CLS Transmit CLS
LP_TOV timeout occurred between events
LP_TOV timeout occurred between events
CLS received
CLS transmitted
*Note--the L_Port may set the appropriate AL_PA bit before transmitting the Loop Initialization Sequences.
AL_PA
no AL_PA
10.4.3 Step (6) EXIT NL_Port: Login if req'd
No AL_PA was available WAIT and retry

Figure 6 — L_Port initialization procedure

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 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. The amount of valid information in the buffer is indicated by four levels. Each level represents buffering for a clock skew management state. Buffering required for clock resynchronization is not shown.

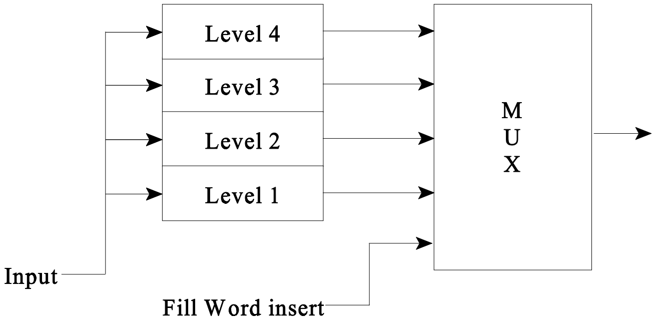


Figure A.1 — Elasticity buffer

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.

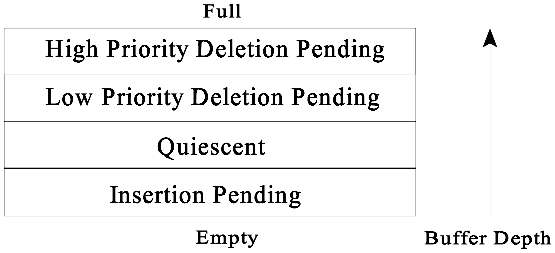


Figure A.2 — Clock skew management states

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 level 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 primitives shall be maintained between frames to meet ANSI X3, FC-PH-x requirements and the Idle primitives that reset the fairness window shall be propagated.

When frames are originated onto the Loop, the FC-PH requirement of six primitives between frames is followed. The inter-frame gap may be reduced to 2 Fill Words plus other primitives 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

L_Ports that reach the first level of buffer depth that triggers a request to delete, follow the low priority rules for deletion. 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-Order Set (data words), the L_Port deletes the next Fill Word; and,

NOTE — A non-Order Set indicates an intervening frame. By detecting non-Order 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 attempting a deletion, 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 delete, the L_Port shall enter the low priority state and wait 4 Fill Words before another delete or the quiescent state with no delete pending.

Examples:

```
EOF FW FW FW FW FW FW SOF
                ^  ^          May delete for low priority
EOF AR AR AR ID ID ID SOF
                   ^          May delete for low priority
EOF FW FW RR FW FW RR FW FW SOF
                      ^  ^    May delete for low priority

Legend:
EOF = End of Frame               |    AR  = ARB(x)
SOF = Start of Frame             |    ID  = Idle
FW  = Fill Word, ARB(x) 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-Order Set (data words), the L_Ports deletes the next Fill Word; and

- if the CFW changes to Idle while attempting a deletion, the L- Port shall not delete the first Idle.

After a high priority delete, the L_Port shall re-enter:

- the low priority state and wait 4 Fill Words before another delete; or,
- 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 for high priority

EOF AR AR AR ID ID ID SOF
    ^  ^  ^                   May delete for high priority

EOF FW FW RR FW FW RR FW FW SOF
             ^  ^     ^  ^    May delete for high priority

Legend:
EOF = End of Frame               |    AR  = ARB(x)
SOF = Start of Frame             |    ID  = Idle
FW  = Fill Word, ARB(x) 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 nine of the eleven states in the LPSM (see 8.4 for states and items) and eleven of the twenty-three state transitions. Of the twelve 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 Node 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 hard AL_PA (since it does not have a valid AL_PA) to instruct the LPSM to arbitrate (REQ(arbitrate as x)) and to initialize (REQ(initialize)).

The LPSM makes transition (01);

— The LPSM, now in the ARBITRATING state, begins to replace any Idle, ARB(F0), ARBf, or ARBx (where x is higher than the trusted hard AL_PA) with ARBx (where x is a trusted hard AL_PA). The LPSM monitors its inbound fibre for the ARBx 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 INITIALIZING state, begins to transmit LIP(F7,F7) and monitors its inbound fibre for LIP. (See 8.4.3, item 21 for details);

— The LPSM detects LIP.

The LPSM makes transition (89);

— The LPSM, now in the OPEN-INIT state, sets ACCESS to TRUE(1), transmits at least twelve (12) received LIPs, continuously transmits the Loop Initialization Sequence (LI_ID='LISM'), and monitors its inbound fibre for a Loop Initialization Sequence (LI_ID='LISM' or for ARB(F0)). (See 8.4.3, item 22 for details.)

- ARB(F0) is received (this indicates that another L_Port has been selected as the LIM).
- The L_Port sets its AL_PA bit and AL_PA value in the appropriate Loop Initialization Sequence AL_PA bit map (LI_ID='LIFA', 'LIPA', 'LIHA', 'LISA') and AL_PA position map (LI_ID='LIRP'), respectively, and retransmits them.
- The L_Port monitors it inbound fibre for CLS sent by the LIM.

— The LPSM detects CLS.

The LPSM makes transition (90);

- The NL_Port, now in the MONITORING state, is ready to execute a Fabric Login if it is a Public NL_Port. (See 10.4, step (5).)

The complete list of state transitions and states in the order used is: (01), 1: ARBITRATING; (12) 2: ARBITRATION WON; (28), 8: INITIALIZING; (89), 9: OPEN-INIT; (90), 0: MONITORING.

B.2 N_Port Login example

After the NL_Port initialization procedure has completed (see 10.4), 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 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 ARBx with its own ARB(26). (See 8.4.3, item 14 for details.)

  The LPSM monitors for ARB(26) on its inbound fibre.

  Assuming no higher priority NL_Port is arbitrating, ARB(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 a 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 – If no other L_Port had been arbitrating, 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	**NL_Port 32**
0: MONITORING, (01)	0: MONITORING, (04)
1: ARBITRATING, (12)	4: OPENED, (46)
2: ARBITRATION WON, (23)	6: RECEIVED CLOSE, (60)
3: OPEN, (35)	0: MONITORING
5: XMITTED CLOSE, (50)	
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 FC-AL, 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 a 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)	OPNyx ==> R_RDYs ==>	**Full-duplex L_Port** (able to transmit and receive Data frames simultaneously)
- Transmit n-1 Data frames.	frame(s) ==>	
	<== R_RDYs	- Transmit R_RDY and Link_Control frames (if any)
	<== frames	- Transmit Data frames
- Transmit last frame	frame(n) ==>	
If Login DHD is FALSE(0), transmit CLS.	CLS ==>	- CLS received, continue transmitting frames until Available_BB_Credit=0 or EE_Credit=0).
	<== frame(s)	
Loop is closed--next arbitrating L_Port wins Loop.	<== 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).
Continue to transmit R_RDYs and Link_Control frames (if any)	<== frames	- Transmit n-1 Data frames
	R_RDYs ==>	- Transmit R_RDYs and Link_Control frames (if any)
	<==R_RDYs	
	<== frame(n)	- Transmit last frame
Transmit CLS to close Loop.	<== CLS	- Transmit CLS
	CLS ==>	- Loop is closed--next arbitrating L_Port wins Loop.

NOTE — Table C.1 shows frame transfers based on R_RDY flow control. For dedicated Classes of service (e.g., 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 fairness rules (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 a 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 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. Another performance enhancement would be to use full-duplex connections such that when the Target connects to the Initiator to transmit the requested data, the Initiator can transmit subsequent commands to that Target.

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 a 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 a 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 8.3.5. 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 the CLS is received from A; and,

9) may continue to transmit frames until Available_BB_Credit or EE_Credit is exhausted, followed by a 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 an 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 an 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 low-order byte of the S_ID is 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 an 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, filtering, is used. This transfer of jitter makes the number of ports that may be connected in a loop undetermined.

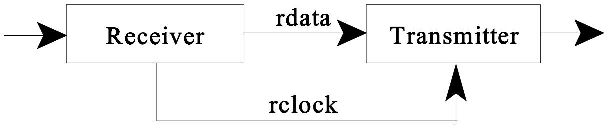


Figure G.1 — Example of a synchronous L_Port design

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.

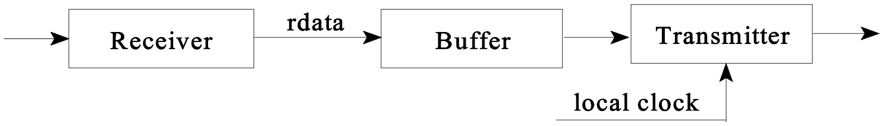


Figure G.2 — Example of an asynchronous L_Port design

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).

Assuming a worst case frequency mismatch between two connected Ports (i.e., transmitter at f+(100 ppm)*f and receiver at f- (100 ppm)*f). The maximum duration for a frame is 2156 characters: 2112 (data field) + 24 (FC-2 header) + 12 (SOF, EOF, and CRC) + 8 (two Fill Words).

Using 1.0625 GHZ as an example, the net elasticity needed for a maximum size frame is:

2156 Characters * 10 bits/char * 200 ppm * 1.0625 * 10e9 / 1.0625 * 10e9= 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 state.

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)

Bypass Circuit example and usage

This annex describes a 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 Bypass Circuit provides the means to route the serial channel signal past an L_Port. Also described are the L_Port Bypass/Enable (LPB/LPE) Primitive Sequences. The main purpose of these Primitive Sequences is to physically control the 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 AL_TIME or until the transmitted Primitive Sequence is received.

I.1 Bypass Circuit

Figure I.1 shows an example 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 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 Bypass Circuits should be used which avoid counting this loss of synchronization as an error and going to the INITIALIZING state.

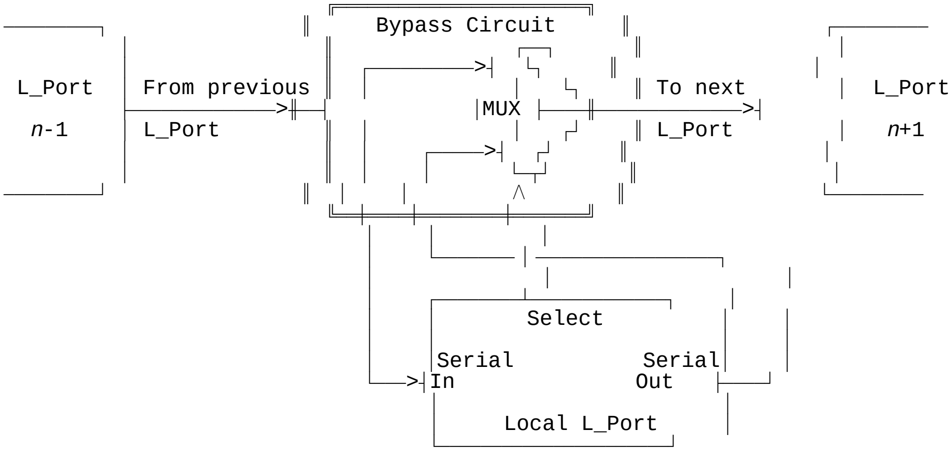


Figure I.1 — Example Bypass Circuit

I.1.1 Default bypass

The Bypass Circuit for an unpopulated location or a powered-off L_Port defaults to the 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 Bypass Circuit set and enters the MONITORING state in non-participating mode. This allows the Loop to continue to function while the Port performs a self-test. When the L_Port is ready to enter the participating mode, the L_Port deactivates it's Bypass Circuit and enters the INITIALIZING state.

I.2 Using a 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 Bypass Circuit

Loop Initialization must be completed before the 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 an OPNyy (where y is the AL_PA of the Loop manager); and, transmits the L_Port Bypass Primitive Sequence (LPByx, where y is the 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 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 is the AL_PA of the L_Port under test). If the OPNyx is received by the Loop manager, the 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 it's 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 is the AL_PA of the L_Port under test). If the OPNyx is not received by the Loop manager within AL_TIME, the 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 Bypass Circuit s 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 receives 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 at its receiver for R_T_TOV, it may use the following recovery procedure. Waiting for R_T_TOV, allows recovery from transient conditions.

After an R_T_TOV time-out, 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 INITIALIZING state where it transmits an error LIP (see 7.8.2 or 7.8.4) for up to two (2) AL_TIMEs or until LIP is received.

If LIP is received, the L_Port goes to the OPEN-INIT state and transmits at least twelve (12) of the received LIPs (see 7.8.1 or 7.8.3).

If LIP is not received within two (2) AL_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 two (2) AL_TIMEs or until the Primitive Sequence is received 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 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 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.

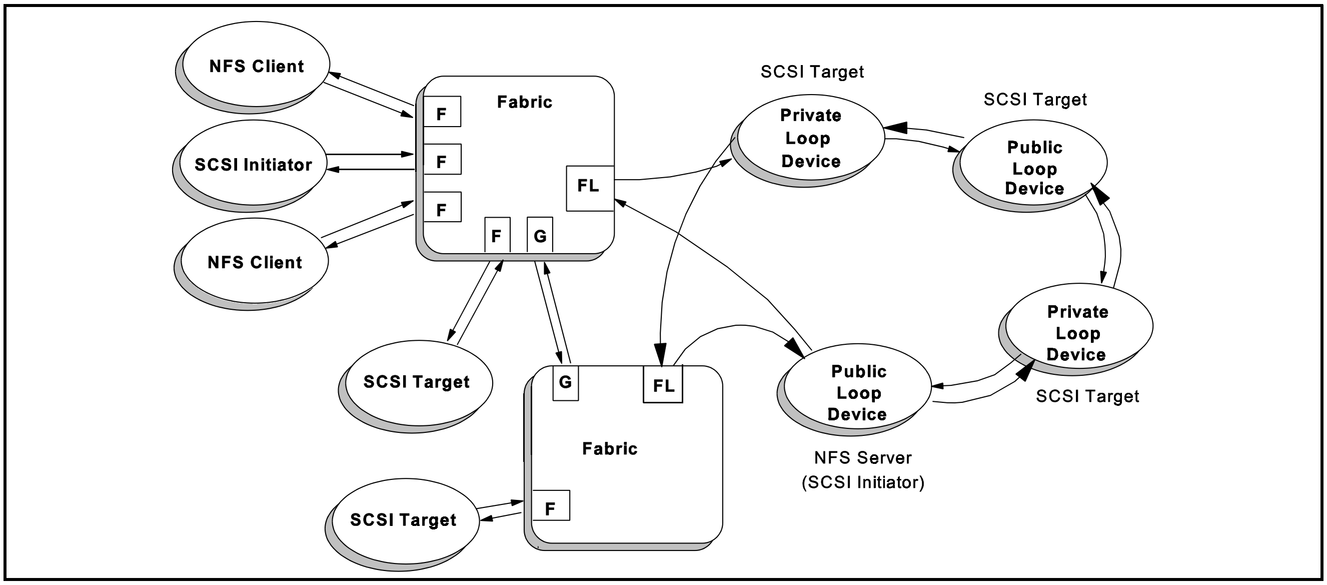


Figure J.1 — Public L_Ports and Private NL_Ports on a Loop

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 (Small Computer System Interface) 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 10.4. 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	Switch Setting		AL_PA	Switch Setting		AL_PA	Switch Setting	
(hex)	(hex)	(dec)	(hex)	(hex)	(dec)	(hex)	(hex)	(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 participate 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 FC-AL Loops. Every server has direct access to any partition of the database on the shared disk pool. The FC-AL 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 traffic between servers. As this example shows, many database blocks are passed directly between servers during the parallel query.

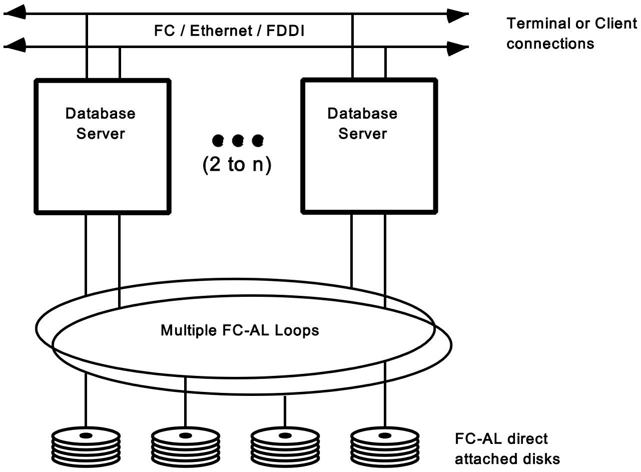


Figure L.1 — FC-AL parallel query server

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 replicate 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 shall 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.

Index

End of Document

Printed:
February 17, 1998 at 03:39PM