INTERNATIONAL STANDARD

ISO/IEC 14165-122

First edition
2005-06

Information technology –
Fibre channel –

Part 122:
Arbitrated loop-2 (FC-AL-2)

Reference number
ISO/IEC 14165-122:2005(E)

INTERNATIONAL STANDARD

ISO/IEC 14165-122

First edition
2005-06

Information technology – Fibre channel –

Part 122: Arbitrated loop-2 (FC-AL-2)

PRICE CODE XB

For price, see current catalogue

CONTENTS

INFORMATION TECHNOLOGY – FIBRE CHANNEL –

PART 122: Arbitrated loop-2 (FC-AL-2)

FOREWORD

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

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

3) All users should ensure that they have the latest edition of this publication.

4) No liability shall attach to IEC or ISO or its directors, employees, servants or agents including individual experts and members of their technical committees and IEC or ISO National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication of, use of, or reliance upon this ISO/IEC publication or any other IEC or IEC/ISO publications.

5) Attention is drawn to the normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.

6) Attention is drawn to the possibility that some of the elements of this ISO/IEC Publication may be the subject of patent rights. ISO/IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 14165-122 was prepared by subcommittee 25: Interconnection of information technology equipment, of ISO/IEC joint technical committee 1: Information technology.

INTRODUCTION

International Standard ISO/IEC 14165-122 specifies an enhancement to the signaling protocol of the Fibre Channel Physical and Signaling Interface (FC-PH), ISO/IEC 14165-251, 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. FC-PH-n refers to *n* versions of FC-PH. The roadmap is intended to show the general relationship of documents to one another, not a hierarchy, protocol stack or system architecture. It does not show the complete set of Fibre Channel documents.

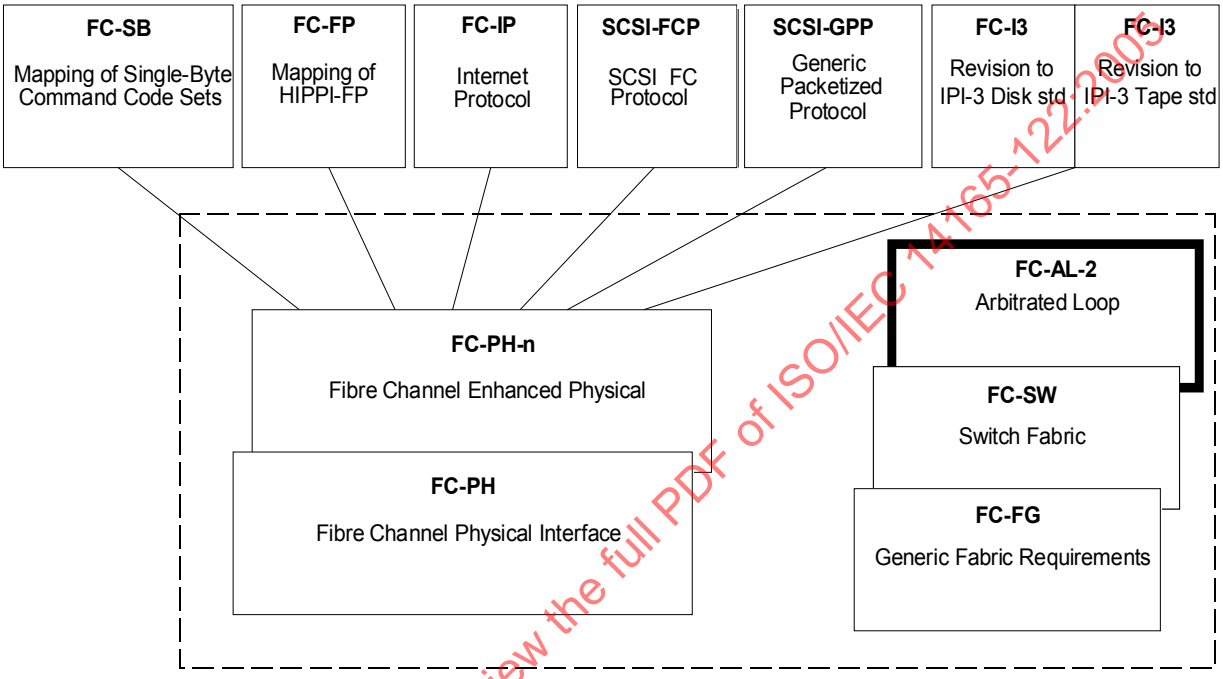


Figure 1 — Fibre channel roadmap

FC-AL features enhanced Ports, called L_Ports, which arbitrate to access an Arbitrated Loop. Once an L_Port wins arbitration, a second L_Port may be opened to complete a single point-to-point circuit (i.e., communication path between two L_Ports). When the two connected L_Ports release control of the Arbitrated Loop, another point-to-point circuit may be established. An L_Port may have the ability to discover its environment and 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 FC-PH-n, however, modification to the Port hardware is required to transmit, receive and interpret the new Arbitrated Loop Primitive Signals and Sequences.

INFORMATION TECHNOLOGY – FIBRE CHANNEL –

PART 122: Arbitrated loop-2 (FC-AL-2)

1 Scope

This part of ISO/IEC 14165 specifies signaling interface enhancements for FC-PH, 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 FC-PH. The Arbitrated Loop topology attaches multiple communicating points in a loop without requiring switches.

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

When an L_Port is operating on a Loop with at least one other L_Port, the L_Port uses the protocol extensions to FC-PH 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 FC-PH.

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

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 14165-131, *Information technology — Fibre Channel — Part 131: Switch Fabric Requirements (FC-SW)*

ISO/IEC 14165-141, *Information technology — Fibre Channel — Part 141: Fabric Generic Requirements (FC-FG)*

ISO/IEC 14165-251, *Information technology — Fibre Channel — Part 251: Framing and Signaling (FC-FS)*[1]

INCITS 230:1994 [R2004], *Fibre Channel — Physical and Signaling Interface (FC-PH) [T11]*

Amendment 1 (1996)

Amendment 2 (1999)

[1] Under consideration.

3 Definitions and conventions

3.1 Definitions

For the purposes of this International Standard, the definitions in FC-FH and the following definitions apply. Definitions in this clause take precedence over any definitions in FC-FH.

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

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

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

3.1.4
Arbitrated Loop Source Address (AL_PS)
Arbitrated Loop Physical Address of the L_Port on the Loop that transmitted the Primitive Signal or Primitive Sequence

For example, the AL_PS is the x value of the OPNyx Primitive Signal.

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

3.1.6
current Fill Word
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
procedure initiated by the L_Port in the OPEN state to change a full-duplex transfer to a half-duplex transfer

The resulting half-duplex transfer is from the L_Port in the OPENED state to the L_Port in the OPEN state (see 7.4 and annex C).

3.1.8
fairness window
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
Transmission Word which is an Idle or an ARByx Primitive Signal
These words are transmitted between frames, Primitive Signals and Primitive Sequences to keep a fibre active. For information about Primitive Signals and Primitive Sewuences, see FC-FH.

3.1.10
FL_Port
an F_Port (i.e., Fabric Port) which contains the Loop Port State Machine as defined in this standard

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 FC-PH. Both L_Ports are allowed to transmit and receive Data frames

3.1.13
half-duplex
communication model 1 in FC-PH. Only one L_Port is allowed to transmit Data frames

3.1.14
Hub
device for interconnecting L_Ports

3.1.15
Loop
the Arbitrated Loop as described in this standard

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

3.1.17
Loop Failure
loss of word synchronization for greater than R_T_TOV; or loss of signal (see FC-PH)

3.1.18
L_Port
either an FL_Port or a Public or Private NL_Port as defined in FC-PH

3.1.19
NL_Port
N_Port (i.e., Node Port) which contains the Loop Port State Machine defined by this document

Without the qualifier "Public" or "Private," an NL_Port is assumed to be a Public NL_Port.

3.1.20
non-L_Port
Port that does not support the Loop functions defined in this standard (see *Port* in FC-PH)

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

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

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

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

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

3.1.26
Port_Name
A unique 64-bit identifier as defined in the LOGI or ACC frame (see FC-PH)

3.1.27
Primitive Sequence
three identical consecutive Ordered Sets before the function conveyed by the Primitive Sequence is performed (see FC-PH)

3.1.28
Private Loop
a Loop that does not include a Participating FL_Port (see Figure 2 and annex J)

3.1.29
Private NL_Port
an NL_Port that does not attempt a Fabric Login and does not transmit OPN(00,x) (see Figure 2 and 5.2)

3.1.30
Public Loop
a Loop that includes a Participating FL_Port and may contain both Public and Private NL_Ports (see Figure 2 and annex J)

3.1.31
Public NL_Port
an NL_Port that attempts a Fabric Login (see Figure 2 and 5.2)

3.1.32
replicate frame
a Class 3 frame which may be received and processed by one or more NL_Ports while being forwarded (see 7.2)

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

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

3.2 Editorial conventions

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

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

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

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

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

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

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

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

3.3 Abbreviations, acronyms and other special words

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

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

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

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

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

ARB_PEND **Arb**itration **PEND**ing history variable – a two-valued variable (i.e., 0/1) to help an L_Port remember that it was arbitrating

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

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

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

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

ARBf_SENT **Arb**itrate hex 'FF' **Sent** history variable – a two-valued variable (i.e., *0/1*) that indicates that the L_Port has requested REQ(arbitrate (FF)) and the LPSM has modified the current Fill Word to ARB(FF) (see 8.1.6 and 8.3.3)

Available_BB_Credit **Available** **B**uffer-to-**B**uffer **Credit** (see 8.2.4)

BB_Credit **B**uffer-to-**B**uffer **Credit** as established during Login (see 8.2.4 and FC-PH)

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

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

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

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

DHD_RCV **D**ynamic **H**alf-**D**uplex **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 FC-PH)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

LPSM **L**oop **P**ort **S**tate **M**achine (see 8.4)

MRKtx **M**ar**k** Primitive Signal (see 7.6)

MK_TP **M**ar**k** **T**ype – used to identify the type of Mark Primitive Signal (see 7.6)

OPNr **Open** Replicate Primitive Signal – either OPNyr or OPNfr (see 7.2)

OPNfr **Open** Primitive Signal – broadcast replicate(see 7.2.2)

OPNy **Open** Primitive Signal – either OPNyx or OPNyy (see 7.1)

OPNyr **Open** Primitive Signal – selective replicate(see 7.2.1)

OPNyx **Open** Primitive Signal – full-duplex (see 7.1.1)

OPNyy **Open** Primitive Signal – half-duplex (see 7.1.2)

PARTICIPATE **PARTICIPATE** history variable – a two-valued variable (i.e., *0/1*) that indicates whether an L_Port has an AL_PA and is participating on the Loop (see 8.1.4)

REPEAT symbol whose value is derived from BYPASS and PARTICIPATE, which indicates that an L_Port merely repeats received Transmission Words (see 8.1.4)

REPLICATE **Replicate** history variable – a two valued variable (i.e., *0*/*1*) to indicate if an L_Port has transmitted OPNr while in the OPEN state or an NL_Port has received OPNr while in the MONITORING or ARBITRATING states (see 8.1.3)

SOFiL **S**tart_**o**f_**F**rame Primitive Signal (K28.5 D21.5 D22.2 D22.2) used during Loop Initialization (see 10.5)

XMIT_2_IDLES **X**mit **2** **I**dles history variable – a two-valued variable (i.e., 0/1) that indicates whether the L_Port needs to transmit two (2) Idles (see 8.1.1)

3.4 Symbols

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

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

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

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

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

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

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

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

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

4 Structure and concepts

4.0 Purpose

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 FC-PH. 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 shall 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) are required to connect more than two Ports together (see FC-FG, for the minimum requirements of a Fabric).

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

The different topologies which are defined in FC-PH 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 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 FC-PH protocol is used for all operations.

Figure 2 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 2(a) shows two Loops: one includes two NL_Ports (i.e., point-to-point) and the other includes six NL_Ports (i.e., Private Loops). The configuration in Figure 2(b) shows a Loop which includes one FL_Port (i.e., a Public Loop) and five NL_Ports (either Public or Private NL_Ports). (See annex J.)

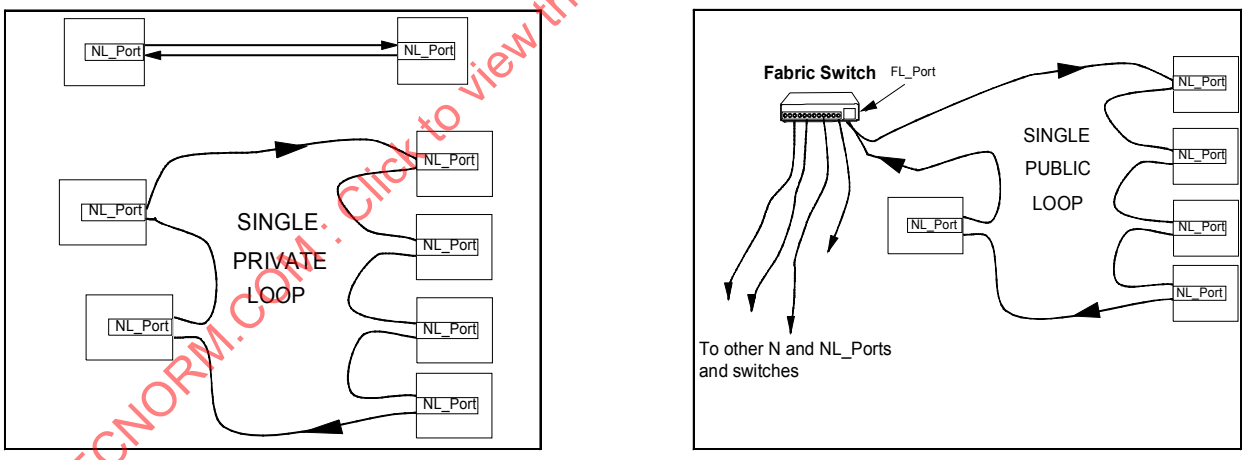


Figure 2 — Examples of the Loop topology

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

4.3.0 Access fairness overview

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

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

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

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

4.3.1 Access fairness for NL_Ports

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

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

4.3.2 Access unfairness for NL_Ports

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

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

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

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

4.3.3 Access unfairness for FL_Ports

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

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

4.4 Relationship to FC-PH

If a Port uses FC-AL, it extends the FC-2 and FC-1 functions of FC-PH. Figure 3 shows logically where the Loop (FC-AL) function is located. This functional level does not have a formally defined interface to the other levels.

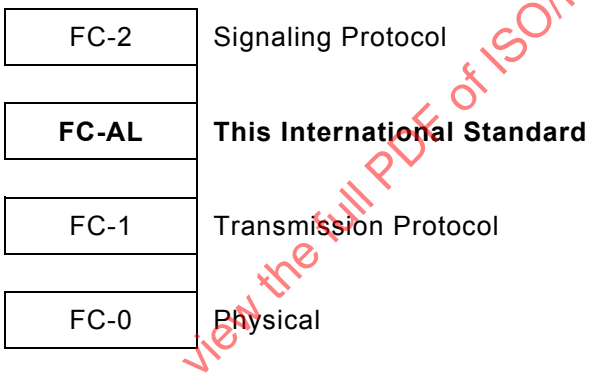


Figure 3 — FC-PH with Arbitrated Loop addition

When two L_Ports are communicating, the L_Ports may use all of the functions specified in FC-PH. 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 FC-PH, 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 8.2 in addition to those identified in FC-PH, 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 FC-PH);

— specifies that all NL_Ports and the optional FL_Port on a Loop shall use the same data rate (see FC-PH);

— the FC-PH buffer-to-buffer flow control is not used for L_Ports that are monitoring the Loop. (See 8.2.3);

— expands the number of Ordered Sets beyond those specified in FC-PH (see clause 6);

— expands the number of Primitive Signals and Sequences beyond those specified in FC-PH, (see clause 7);

— extends the Ordered Sets that may be deleted to include Idle, ARByx, and all Primitive Sequences, (See 8.2.2);

— specifies the Primitive Signals that may be inserted on a Loop between frames for clock skew management (see 8.2.2);

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

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

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

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

— requires that the L_Port set the "Alternate BB_Credit Management" bit to 1 in the N_Port Common Service Parameters during Login (see FC-PH);

— 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 FC-PH for flow control), FC-2 uses

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

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

5 Addressing

5.1 Arbitrated Loop Physical Address (AL_PA)

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

Each L_Port shall use an AL_PA value that results in neutral disparity (see FC-PH). 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 FC-PH 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 FC-PH 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 FC-PH in that order. The y portion of the FC-1 naming convention is based on bits identified as H, G, and F in FC-PH, 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 FC-PH, to obtain the Dxx.y data byte name;

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

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

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

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

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

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

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

5.1.1 Valid AL_PAs

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

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

Highest priority.

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

NOTE Adding a new FL_Port to a loop with an existing FL_Port, may cause the access to the ports on that loop to change to the new FL_Port and the fabric that it is a part of, including possible changes in address for the ports on the loop. If the fabric that the two FL_Ports are part of is not the same, this may impact access to the ports on the loop from the fabric.

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

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

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

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

5.1.2 Special AL_PAs and flags

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

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

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

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

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

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

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

5.2 Native address identifier

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

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

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

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

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

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

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

6 FC-AL Ordered Sets

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

Table 2 – Primitive Signals

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

Characters 3 and 4 of Ordered Sets ARByx, ARB(val), MRKtx, OPNyx, OPNyy, OPNyr, and OPNfr shall contain one of the neutral disparity values from Table 1, whenever such an Ordered Set is originated (see 7.1 to 7.7).

Table 3 — Primitive Sequences

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

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

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

7 FC-AL Primitive Signals and Sequences

7.0 Overview

The Arbitrate Primitive Signal (ARByx) may be transmitted in place of an Idle and therefore becomes a Fill Word which may be removed for clock skew management. The Mark Primitive Signal (MRKtx) may also be transmitted in place of a Fill Word, but it shall not be removed for clock skew management. All Primitive Signals (except MRKtx and Fill Words) defined in this standard shall follow the 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 FC-PH and clause 6 for a specification of the following Ordered Sets.)

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

7.1 Arbitrate Primitive Signals (ARByx)

7.1.0 ARByx overview

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

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

7.1.1 ARB(AL_PA)

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

7.1.2 ARB(F0)

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

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

7.1.3 ARB(FF

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

7.2 Open Primitive Signals (OPNy)

7.2.0 OPNy overview

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

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 FC-PH). The OPNyx shall contain the AL_PD (destination = y value) of the L_Port to be opened and the AL_PS (source = x value) of the L_Port which transmitted OPNyx.

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

7.2.2 Open half-duplex (OPNyy)

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

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

7.3 Open Replicate Primitive Signals (OPNr)

7.3.0 OPNr overview

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

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

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

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

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

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

7.3.1 Open selective replicate (OPNyr)

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

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

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

7.3.2 Open broadcast replicate (OPNfr)

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

7.4 Close Primitive Signal (CLS)

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

7.5 Dynamic Half-Duplex Primitive Signal (DHD)

Dynamic Half-Duplex (DHD) is transmitted on a Loop by the L_Port in the OPEN state to indicate to the L_Port in the OPENED state that the L_Port in the OPEN state has no more Data frames to transmit. DHD shall only be requested by the L_Port in the OPEN state if both L_Ports in the current Loop circuit have indicated support of DHD via Login and if the L_Port in the OPEN state had transmitted OPNyx (full-duplex open). The DHD supported Login bit is found in FC-PH-3 (see ISO/IEC 14165-251). DHD may allow L_Ports to make more efficient use of the established Loop circuit (see annex C) by allowing an L_Port which is in the OPENED state to transmit all Data frames, even though the L_Port in the OPEN state has finished its data transfer.

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

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

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

7.6 Mark Primitive Signal (MRKtx)

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

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

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

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

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

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

7.7 Loop Port Bypass/Enable Primitive Sequences

7.7.0 LPE/LPB overview

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

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

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

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

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

7.7.1 Loop Port Bypass (LPByx)

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

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

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

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

7.7.2 Loop Port Bypass all (LPBfx)

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

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

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

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

7.7.3 Loop Port Enable (LPEyx)

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

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

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

7.7.4 Loop Port Enable all (LPEfx)

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

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

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

7.8 Loop initialization primitive sequences (LIP)

7.8.0 LIP overview

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

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

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

7.8.1 Loop Initialization — no valid AL_PA

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

7.8.2 Loop Initialization — Loop Failure; no valid AL_PA

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

7.8.3 Loop Initialization — valid AL_PA

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

7.8.4 Loop Initialization — Loop Failure; valid AL_

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

7.8.5 Loop Initialization — reset L_Port

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

7.8.6 Loop Initialization — reserved

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

8 L_Port operation

8.0 Overview

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

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

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

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

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

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

The following steps provide an example for how an L_Port transfers one or more frames on a Loop.

a) The L_Port requests the LPSM to obtain access to the Loop.

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

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

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

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

8.1 History

8.1.1 Access fairness history

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

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

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

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

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

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

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

8.1.2 Duplex mode history

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

8.1.3 Replicate mode history

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

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

8.1.4 Operational mode history

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

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

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

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

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

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

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

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

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

8.1.5 DHD received history

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

8.1.6 ARB(FF) history

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

8.2 Timeouts

8.2.1 FC-PH timeout values

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

8.2.2 Arbitrated Loop timeout value

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

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

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

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

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

8.2.3 Loop timeout

The Loop timeout value (LP_TOV) is 2 s. LP_TOV is used to keep a Loop from deteriorating due to protocol errors or lost Ordered Sets. For example, LP_TOV is used to reset the fairness window (see 4.2) and during the INITIALIZATION process to time start-up events (see 10.5.4).

8.3 Operational characteristics

8.3.1 Transmission Word

8.3.1.1 Power-on Transmission Words

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

8.3.1.2 Invalid Transmission Words and Transmission Characters

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

— in the MONITORING or ARBITRATING states:

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

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

— in any other state the L_Port shall follow the rules defined in FC-PH and clause 29.

8.3.2 Clock skew management

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

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

8.3.3 Error detection and recovery

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

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

8.3.4 BB_Credit and Available_BB_Credit

8.3.4.0 BB_Credit overview

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 FC-PH) and BB_Credit shall be set to 0 and one (1) in the OLD-PORT state and to 1 and zero (0) in the OPEN-INIT-START state, respectively (see 8.4.3 item 21 and item 23, and 10.5.4). During Login, BB_Credit shall be set to a value that represents the number of receive buffers that the L_Port shall guarantee to have available when a Loop circuit is established.

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

— Loop circuits are dynamic;

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

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

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

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

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

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

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

8.3.4.1 BB_Credit management per Loop circuit

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

8.3.4.2 Available_BB_Credit management per Loop

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

Available_BB_Credit is one of the following values:

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

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

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

8.4 Loop Port State Machine (LPSM)

8.4.0 LPSM overview

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

8.4.1 State names

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

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

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

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

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

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

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

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

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

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

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

8.4.2 State diagram

The state diagram is shown in Figure 4. 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.

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

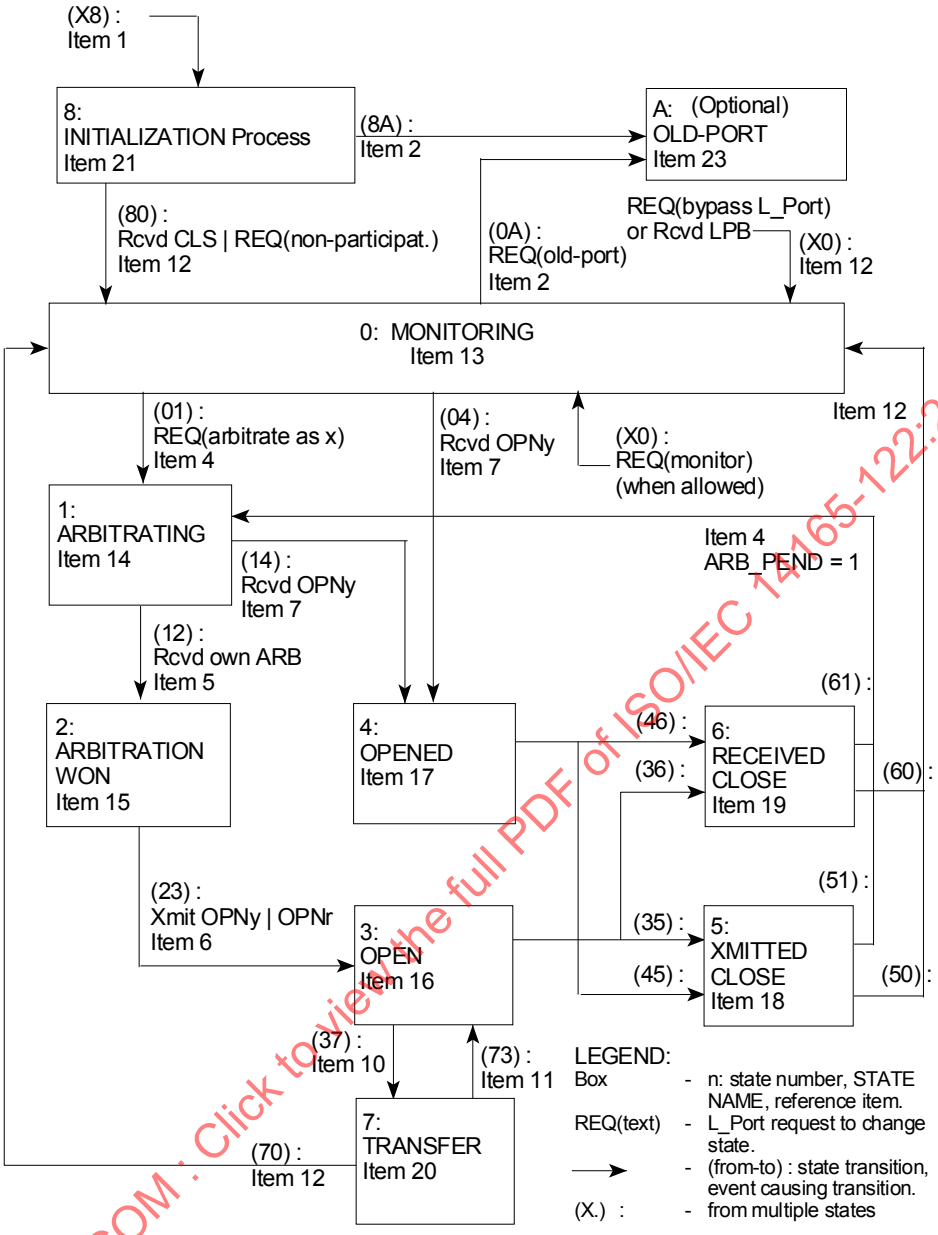


Figure 4 — State Diagram

8.4.3 Reference items

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

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

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

All fibre-type dependent operations shall be complete before making this transition (e.g., Open Fibre Control) (see FC-PH).

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

— if REPEAT is FALSE(0) and:

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

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

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

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

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

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

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

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

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

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

  – if REPEAT is FALSE(0):

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

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

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

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

— if ARByx = ARB(val):

  – if REPEAT is FALSE(0):

    – if val <> AL_PA of the L_Port and

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

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

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

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

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

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

If REPEAT is FALSE(0):

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

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

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

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

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

f) if MRKtx is received and:

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

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

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

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

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

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

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

If LIP is recognized:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

14 **State 1 (ARBITRATING) actions** (Table 5 and the following text describe the ARBITRATING state): The LPSM shall set DUPLEX to FALSE(0) and ARB_WON to FALSE(0). The LPSM shall retransmit all received Transmission Words unless specifically stated otherwise. The LPSM shall transmit an ARB(AL_PA) (where AL_PA is the AL_PA of the L_Port) when either an Idle or a lower priority ARB(AL_PA), ARB(F0), or ARB(FF) is received.

If Idle is received and:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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 and:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

NOTE The six (6) Ordered Sets maintain the FC-FS 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 FC-PH (see 8.3.4).

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

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

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

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

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

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

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

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

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

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

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

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

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

If LPB is recognized:

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

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

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

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

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

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

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

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

If Idle is received and:

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

— if ARB_PEND is TRUE(1) and:

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

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

If ARB(F0) is received and:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The LPSM may begin to close the Loop (REQ(close)) by transmitting CLS instead of the next appropriate Fill Word and then shall make the transition to the XMITTED CLOSE state.

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

If a Class 1 connection exists, the L_Port shall remove the Class 1 connection before transmitting CLS; only the L_Port which received EOFdt shall transmit the first CLS (see item 8 and item 18).

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

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

— frame transmission is required with a different L_Port;

— the L_Port was opened full-duplex and has not received any credit to transmit frames before a timeout occurred (an appropriate value would be LP_TOV since this L_Port is the responder in the Loop circuit);

— the L_Port was opened half-duplex and the L_Port has Data frames to transmit to the other L_Port; or,

— the L_Port is making the transition to the Non-Participating mode.

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

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

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

If the L_Port requests arbitration (REQ(arb own AL_PA)) and ACCESS is TRUE(1), the LPSM shall set ARB_PEND to TRUE(1).

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

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

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

If Idle is received and:

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

— if ARB_PEND is TRUE(1) and:

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

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

If ARB(F0) is received and:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

If CLS is received and:

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

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

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

If MRKtx is received and:

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

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

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

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

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

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

If the L_Port requests arbitration (REQ(arb own AL_PA)) and ACCESS is TRUE(1), the LPSM shall set ARB_PEND to TRUE(1).

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

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

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

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

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

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

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

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

If Idle is received and:

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

— if ARB_PEND is TRUE(1) and

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

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

If ARB(F0) is received and

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

If the L_Port requests arbitration (REQ(arb own AL_PA)) and ACCESS is TRUE(1), the LPSM shall set ARB_PEND to TRUE(1).If the L_Port requests to transmit a MRKtx (REQ(mark as tx)), the LPSM shall transmit one MRKtx at the next Fill Word (see clause 7), unless REQ(mark as tx) is removed before MRKtx is transmitted.

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

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

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

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

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

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

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

If CLS is received and:

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

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

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

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

If MRKtx is received and:

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

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

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

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

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

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

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

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

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

22 **Reserved**

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

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

11 Private NL_Ports only support NL_Port Login.

9 L_Port state transition tables

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

The following notations, conventions and abbreviations are used in Table 4 to Table 14.

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

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

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

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

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

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

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

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

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

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

— The entry labeled "ANY OTHER O.S." addresses the first and second Ordered Set of each Primitive Sequence (see FC-PH).

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

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

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

Table 4 — MONITORING (State 0) transitions

ENTRY ACTIONS		
ACCESS := N/C ARB_PEND := 0 ARB_WON := 0 ARBf_SENT := 0	DUPLEX := 0 REPLICATE := 0 CFW := N/C	DHD_RCV := N/C BYPASS := N/C XMIT_2_IDLES := N/C
INPUT	**ACTION / OUTPUT**	**NEXT STATE**
LOSS of SYNC. < R_T_TOV	Idle or CFW[12]	MONITORING
Loop Failure REPEAT = 0 REPEAT = 1	 None/Inst. LIP(F8)	 LOOP-FAIL-INITIALIZE MONITORING
INVALID TRANS. WORD	CFW	MONITORING
RUNNING DISP at O.S.	CFW	MONITORING
ELASTICITY WORD REQd	CFW	MONITORING
VALID DATA WORD FL_Port NL_Port REPLICATE = 0 REPLICATE = 1	 Same Word Same Word Receive Word Same Word	 MONITORING MONITORING MONITORING
VALID TRANS. WORD = O.S.		
FRAME DELIMITER		
FL_Port NL_Port SOFxx REPLICATE = 0 REPLICATE = 1 EOFxx REPLICATE = 0 REPLICATE = 1	Same Word Same Word Receive Word Same Word Same Word Receive Word Same Word	MONITORING MONITORING MONITORING MONITORING MONITORING
PRIMITIVE SIGNALS		
Idle REPEAT = 0 ARBf_SENT = 0 ARBf_SENT = 1 REPEAT = 1	 CFW := Idle ACCESS := 1 CFW CFW := ARB(FF) CFW CFW := Idle CFW	 MONITORING MONITORING MONITORING
R_RDY	Same Word	MONITORING
ARByx y <> x	 CFW[13]	 MONITORING

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

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

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

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

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

Table 5 — ARBITRATING (State 1) transitions

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

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

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

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

Table 6 — ARBITRATION WON (State 2) transitions

ENTRY ACTIONS		
ACCESS := N/C ARB_PEND := N/C ARB_WON := N/C ARBf_SENT := N/C	DUPLEX := N/C REPLICATE := N/C CFW := N/C	DHD_RCV := N/C BYPASS := N/C XMIT_2_IDLES := N/C
INPUT	**ACTION / OUTPUT**	**NEXT STATE**
LOSS of SYNC. < R_T_TOV	N/A	N/A
Loop Failure	N/A	N/A
INVALID TRANS. WORD	N/A	N/A
RUNNING DISP at O.S.	N/A	N/A
ELASTICITY WORD REQd	N/A	N/A
VALID DATA WORD	N/A	N/A
VALID TRANS. WORD =O.S.		
FRAME DELIMITERS		
SOFxx EOFxx	N/A N/A	N/A N/A
PRIMITIVE SIGNALS		
Idle	N/A	N/A
R_RDY	N/A	N/A
ARByx	N/A	N/A
OPNr	N/A	N/A
OPNy	N/A	N/A
CLS	N/A	N/A
DHD	N/A	N/A
MRKtx	N/A	N/A
PRIMITIVE SEQUENCES		
LIP	N/A	N/A
LPB (LPByx | LPBfx)	N/A	N/A
LPE (LPEyx | LPEfx)	N/A	N/A
ANY OTHER O.S.	N/A	N/A

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

Table 7 — OPEN (State 3) transitions

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

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

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

Table 8 — OPENED (State 4) transitions

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

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

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

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

Table 9 — XMITTED CLOSE (State 5) transitions

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

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

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

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

Table 10 — RECEIVED CLOSE (State 6) transitions

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

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

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

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

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

Table 11 — TRANSFER (State 7) transitions

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

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

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

Table 12 — INITIALIZATION process (State 8) transitions

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

Table 13 — Reserved

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

Table 14 — OLD-PORT (State A) transitions

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

10 Loop Initialization procedure

10.0 Loop Initialization overview

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

10.1 Loop Initialization summary

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

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

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

— PARTICIPATE is set to FALSE(0) when ARB(F0) is received and TRUE(1) when an AL_PA is chosen during the Loop Initialization Sequences LIFA, LIPA, LIHA or LISA.

NOTE In FC-AL-1, PARTICIPATE was set TRUE(1) after CLS was received and not well defined between ARB(F0) and when CLS was received.

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

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

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

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

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

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

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

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

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

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

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

10.2 Loop Initialization introduction

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

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

Reasons for entering Loop Initialization include:

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

— provide notification of a possible configuration change; and,

— error recovery.

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

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

10.3 Loop Initialization timers

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

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

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

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

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

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

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

10.4 Node-initiated L_Port initialization

This procedure is entered for one of the following reasons:

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

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

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

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

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

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

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

10.5 L_Port initialization

10.5.0 Initialization overview

After the L_Port has performed the entry actions for the OPEN-INIT-START state (see 10.5.4.6), the L_Port shall continue the initialization procedure as defined in 10.5.3, steps (1) to (6). This initialization procedure shall use the Loop Initialization Sequences as defined in Figure 5.

10.5.1 Loop Initialization Sequences

Start_of_Frame delimiter **- 4 bytes**

SOFiL

Frame_Header **- 24 bytes**

22XXXXXX | 00XXXXXX | 01380000 | 00000000 | FFFFFFFF | 00000000

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

Payload **- 12, 20, or 132 bytes**

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

where LI_ID and LI_FL contain the following:

LI_ID (Identifiers) (16 bits)

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

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

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

Cyclic Redundancy Check **- 4 bytes**

CRC

End_of_Frame delimiter **- 4 bytes**

EOFt

Figure 5 — Loop Initialization Sequences

The FC-FS rules for valid frames apply to transmitting the Loop Initialization Sequences which are shown in Figure 5. When an L_Port receives these Loop Initialization Sequences, the L_Port shall discard or not process all frames which contain the following errors (the L_Port is not required to verify the frame header):

— code violations;

— CRC errors;

— a frame that does not end in EOFt or EOFn;

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

— payload violations (i.e., a payload which does adhere to the payloads described in Figure 5).

When forwarding Loop Initialization Sequences, the Non-Loop Initialization Master L_Ports may use the frame header defined in Figure 5, or the received frame header. However, the D_ID and S_ID of the received LISM frame shall be used in either case.

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

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

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

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

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

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

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

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

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

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

10.5.2 Assigned AL_PA values

All AL_PAs that are used in the Loop protocol are specified in Table 1. The AL_PAs are assigned to the 16-byte AL_PA bit maps of Figure 6 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.

10.5.3 Loop Initialization steps

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

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

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

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

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

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

2) Select a Loop Initialization Master (LIM)

The L_Port shall continuously transmit Loop Initialization Sequences (LI_ID='LISM') formatted as shown in Figure 5. Successive Loop Initialization Sequences shall be separated by six or more Idles.

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

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

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

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

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

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

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

3) Wait for a Loop Initialization Master

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

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

4) LIM — transmit remaining Loop Initialization Sequences

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

b) The LIM shall transmit the Loop Initialization Sequences (LI_ID='LIFA', 'LIPA', 'LIHA', and 'LISA'). These Loop Initialization Sequences contain a 16-byte AL_PA bit map in the payload. Each bit represents one AL_PA (see Figure 5, Figure 6 and Table 15).

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

Figure 6 — Loop Initialization Sequence AL_PA bit map

Except for the L_bit, each bit in Figure 6 represents a valid AL_PA (according to Tables 1 and 15). The L_bit shall only be set by the FL_Port or F/NL_Port to indicate that a new Fabric Login is required.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The Loop Initialization Sequences are updated as follows (see Figure 5, Figure 6 and Table 15):

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

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

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

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

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

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

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

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

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

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

6) Select final AL_PA and exit Loop Initialization

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

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

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

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

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

— the NL_Port has not previously executed a Fabric Login.

Normal responses to a Fabric Login request are:

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

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

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

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

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

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

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

10.5.4 Loop Initialization state diagram

10.5.4.0 State diagram overview

The following text provides a detailed state diagram of the INITIALIZATION process. In clause 10, in case of conflicts between text and figures, the following precedence shall be used: figures and then text. 10.5.4 takes precedence over clause 9, Table 12 and Table 14, which takes precedence over 8.4.3, item 21 and item 23, which takes precedence over 10.5.3.

All state diagrams in this subclause use the style shown in Figure 7.

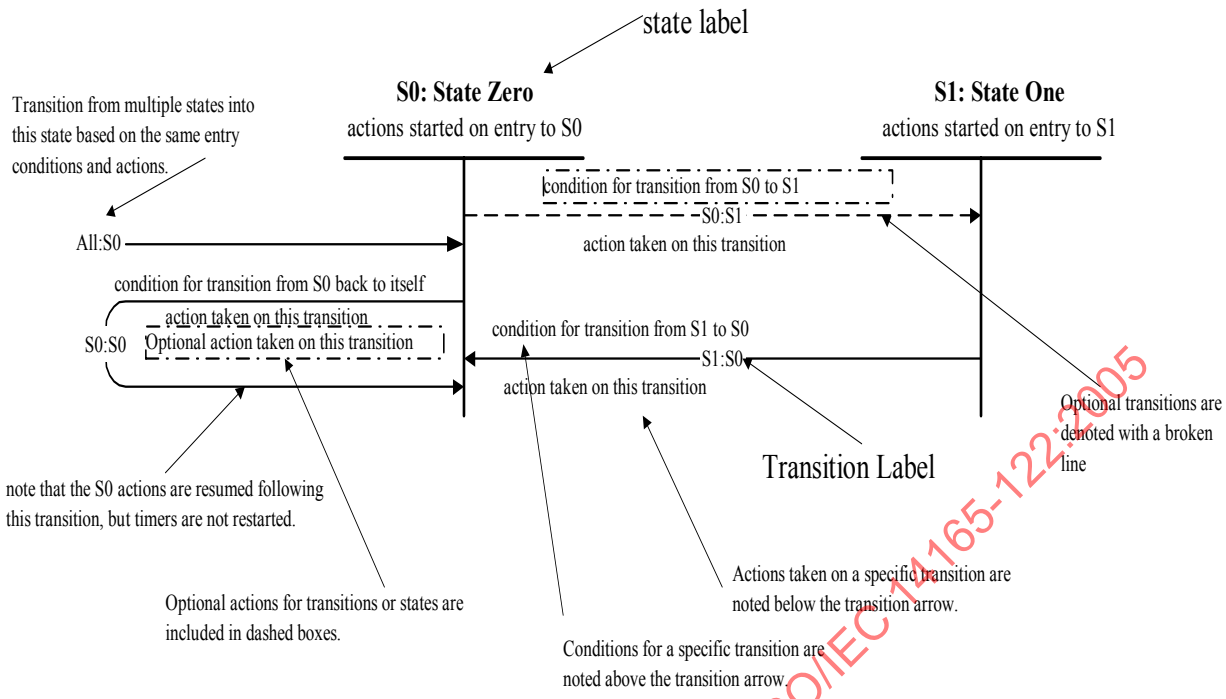


Figure 7 — Loop Initialization state diagram example

These state diagrams are represented using vertical staffs to represent states and horizontal arrows to represent state transitions.

Time elapses only within discrete states with instantaneous transitions between states. Transitions are illustrated with triggering conditions located above the transition arrow and any actions on transition located below the transition arrow. Transition actions are performed while remaining in the previous state, before entry into the new state. The state name appears above the vertical staff representing the state, immediately followed by entry actions if any.

Entry actions are executed every time a state is started. This means that a transition that points back to the same state shall repeat the actions from the beginning. All the actions started upon entry complete before any tests are made to exit the state.

Optional states are depicted with a broken box around the state. Entries to an optional state are shown as solid lines to depict that they are mandatory if the state is implemented. If there is a broken transition line into an optional state this is an optional transition even when the state is implemented.

Broken line transitions into required states, indicate optional transitions.

The following event-processing sequence is assumed:

a) Evaluate all transition conditions from the current state.

b) If a transition condition is satisfied, then

   1) perform the associated transition actions in the current state,

   2) enter the new state and

   3) perform entry actions, if any for the new state.

There are four additional memory elements used in this state diagram definition:

lip_type This is an 8-bit byte that contains a hexadecimal value of the next LIP to be generated.

lip_addr This is an 8-bit byte that contains the hexadecimal value of the address field for the next LIP to be sent.

prev_addr This is an 8 bit byte that contains the previously acquired address if one exists. This is used during the address selection phase of Loop Initialization. It is set to hex 'FF' when it is not valid.

my_addr This is the 8 bit address that this L_Port uses when comparing LIP, LPB and LPE addresses, and when originating LIPs. It is set to hex 'F7' when it is not valid.

The following state diagrams only define the states necessary to perform Loop Initialization. They do not attempt to document the normal Loop Port State Machine (LPSM) that is documented in a state diagram in clause 8 and in table format in clause 9 of this specification. These state diagrams define the operation of the INITIALIZATION process and the OLD-PORT state.

Transitions notes with the state diagrams, describe transitions that enter or leave state diagrams or need additional clarification.

10.5.4.1 Validity of AL_PA

During Loop Initialization, the AL_PA that an L_Port had previously acquired becomes unstable. This subclause defines the point where the AL_PA is valid for various uses.

Initially, an L_Port does not have an AL_PA. This means that the L_Port cannot respond to addressed LPB, LPE or LIPyx. Additionally, this L_Port is not participating in normal Loop operation. When Loop Initialization begins, the L_Port may attempt to acquire an AL_PA during the Hard Assigned or Soft Assigned phases of Loop Initialization.

After an L_Port has completed Loop Initialization once with PARTICIPATE is TRUE(1), it has an acquired AL_PA (this value shall be stored in prev_addr and my_addr upon the transition to MONITORING state). This AL_PA also becomes a fabric assigned AL_PA if FLOGI is completed. At this point in time the L_Port may respond to addressed LPB, LPE, LIPyx, in addition to participating in normal Loop operation.

If LIP occurs, all AL_PAs must be revalidated. Because AL_PAs may change during this process, the following rules apply:

1) An L_Port may respond to addressed LPB, LPE, and LIPyx until it has forwarded an ARB(F0). The L_Port shall use the value in my_addr to validate the address in these Primitive Sequences. After an ARB(F0) has been forwarded, the AL_PA is considered unstable, my_addr is set to hex 'F7', and shall therefore only allow the L_Port to validate the address of LPB, LPE or LIPyx to all L_Ports (i.e., y = hex 'FF') until Loop Initialization has been completed with the CLS being both received and transmitted, at which point the AL_PA acquired during this Loop Initialization is to be placed in my_addr, and may be used to validate the addresses in recognized LPB, LPE, and LIPyx Primitive Sequences.

2) An L_Port may attempt to regain its current AL_PA, which is stored in prev_addr, during either the Fabric Assigned (if FLOGI has been completed) or Previously Acquired phase of Loop Initialization. This AL_PA may be used for regaining the current AL_PA until some other L_Port claims it during Loop Initialization. At this point in time prev_addr is set to hex 'FF' to indicate that it is no longer valid. When Loop Initialization is completed, either the L_Port has a new acquired AL_PA, or is Non-Participating with no AL_PA. If the L_Port is Non-Participating, it shall not recognize any addressed Primitive Signal or Primitive Sequences.

If REQ(nonparticipating) is asserted in the INITIALIZATION process, an immediate transition to the MONITORING state shall be made with the transition actions being: PARTICIPATE is FALSE(0); my_addr = hex 'F7'; and, prev_addr = hex 'FF'.

If REQ(bypass L_Port) is asserted or if LPBfx is recognized in the INITIALIZATION process (except in the POWER-ON-INIT (P0) and the OLD-PORT (OP0) states) and the state transitions are not explicitly identified, an immediate transition to the MONITORING state shall be made with the transition actions being: BYPASS is TRUE(1); PARTICIPATE is FALSE(0); my_addr = hex 'F7'; and, prev_addr = hex 'FF'.

In all states of the INITIALIZATION process, except P0, Loop Failure causes a transition to the F0 state as defined by All:F0. REQ(initialize) shall be removed upon entry into the NORMAL-INITIALIZE state, unless the L_Port is attempting to bypass or enable other L_Ports. REQ(old port) shall be removed upon entry into OLD-PORT state.

10.5.4.2 POWER-ON state diagram

Figure 8 shows the POWER-ON state diagram. This state diagram maintains the transmitter off until the L_Port is capable of initializing and accomplishes the transition to the NORMAL-INITIALIZE or MONITORING state.

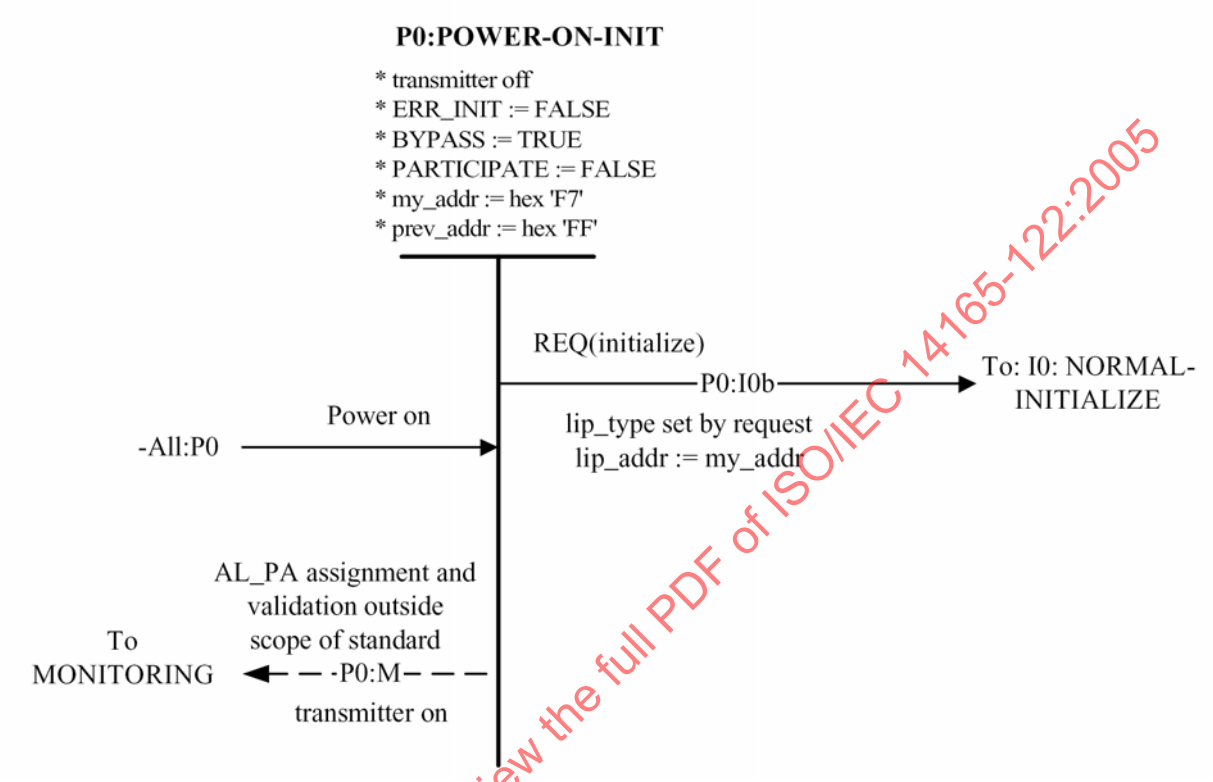


Figure 8 — POWER-ON state diagram

Transmitter off requires that for copper links the transmitter is either tri-stated, or driven to a constant value. For optical links, the optical output should be zero. It is not sufficient for optical links to drive a constant intermediate power level, as this may cause the receiver which has very high gain to mistakenly perceive that the remote L_Port is actually transmitting data.

Transition All:P0 This transition is taken from any state within the LPSM at power-on.

Transition P0:I0b This transition is taken to the NORMAL-INITIALIZE state when REQ(initialize) is asserted.

Transition P0:M This optional transition is taken by an L_Port that does not use Loop Initialization to acquire and verify an AL_PA. If Loop Initialization is not used to acquire and verify AL_PAs, my_addr and prev_addr shall be assigned by a method outside the scope of this standard. It is the responsibility of the implementation to assure AL_PAs are valid and that there are no conflicts. Even if the L_Port does not use Loop Initialization to acquire and verify an AL_PA, it shall participate in Loop Initializations initiated by other L_Ports on the Loop.

10.5.4.3 OLD-PORT state diagram

Figure 9 shows the OLD-PORT state diagram. In the OLD-PORT state, the LPSM is not running except to respond to recognized LIPs. The Port State Machine is operating as defined in FC-FS.

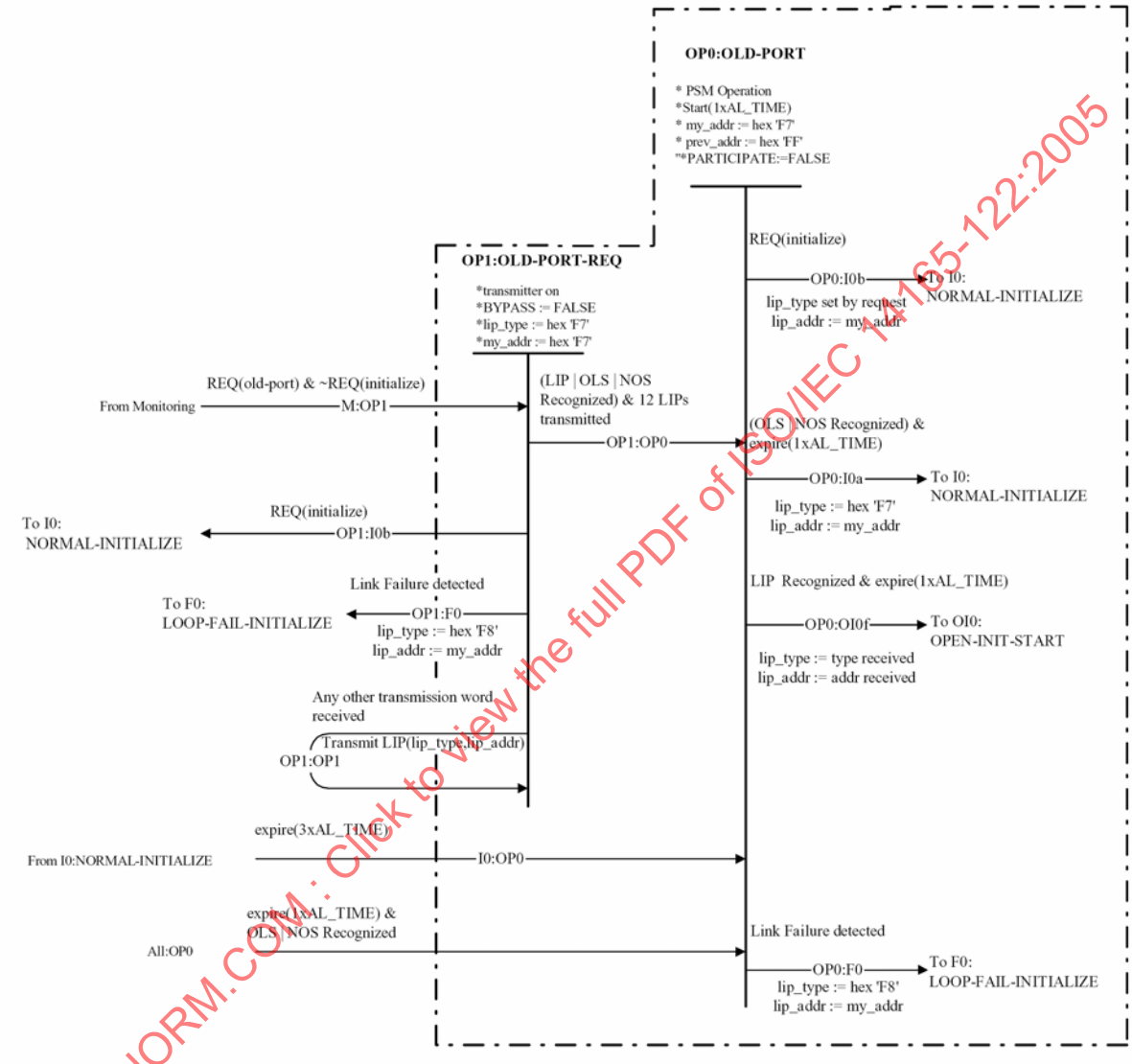


Figure 9 — OLD-PORT state diagram

OLD-PORT state diagram notes:

Transition I0:OP0 This transition is taken from the NORMAL-INITIALIZE state to the optional OLD-PORT state after minimum(3xAL_TIME).

Transition AII:OP0 This transition is taken from the OPEN-INIT-SELECT-MASTER or SLAVE-WAIT-FOR-MASTER state to the optional OLD-PORT state after minimum(1xAL_TIME) if OLS or NOS is recognized.

Transition M:OP1 This transition is taken when REQ(old-port) is asserted and REQ(initialize) is not asserted in the MONITORING state.

Transition OP0:I0b This transition shall be taken to the NORMAL-INITIALIZE state when REQ(initialize) is asserted.

Transition OP0:I0a This transition is taken to the NORMAL-INITIALIZE state when NOS or OLS are recognized, after expire(1xAL_TIME).

Transition OP0:OI0f This transition to the OPEN-INIT-START state is taken when a LIP is recognized after expire(1xAL_TIME).

Transition OP0:F0 This transition is taken when BYPASS is FALSE(0) and a Link Failure is detected as defined in FC-FS.

Transition OP1:I0b This transition is taken to the NORMAL-INITIALIZE state when REQ(initialize) is asserted.

Transition OP1:OP1 On any Transmission Word other than LIP, OLS or NOS, when REQ(initialize) is not asserted, LIP(F7,F7) shall be transmitted.

10.5.4.4 Loop Failure Initialization state diagram

Figure 10 shows the Loop Failure Initialization state diagram. This state diagram attempts to initialize the Loop when a Loop Failure has been detected.

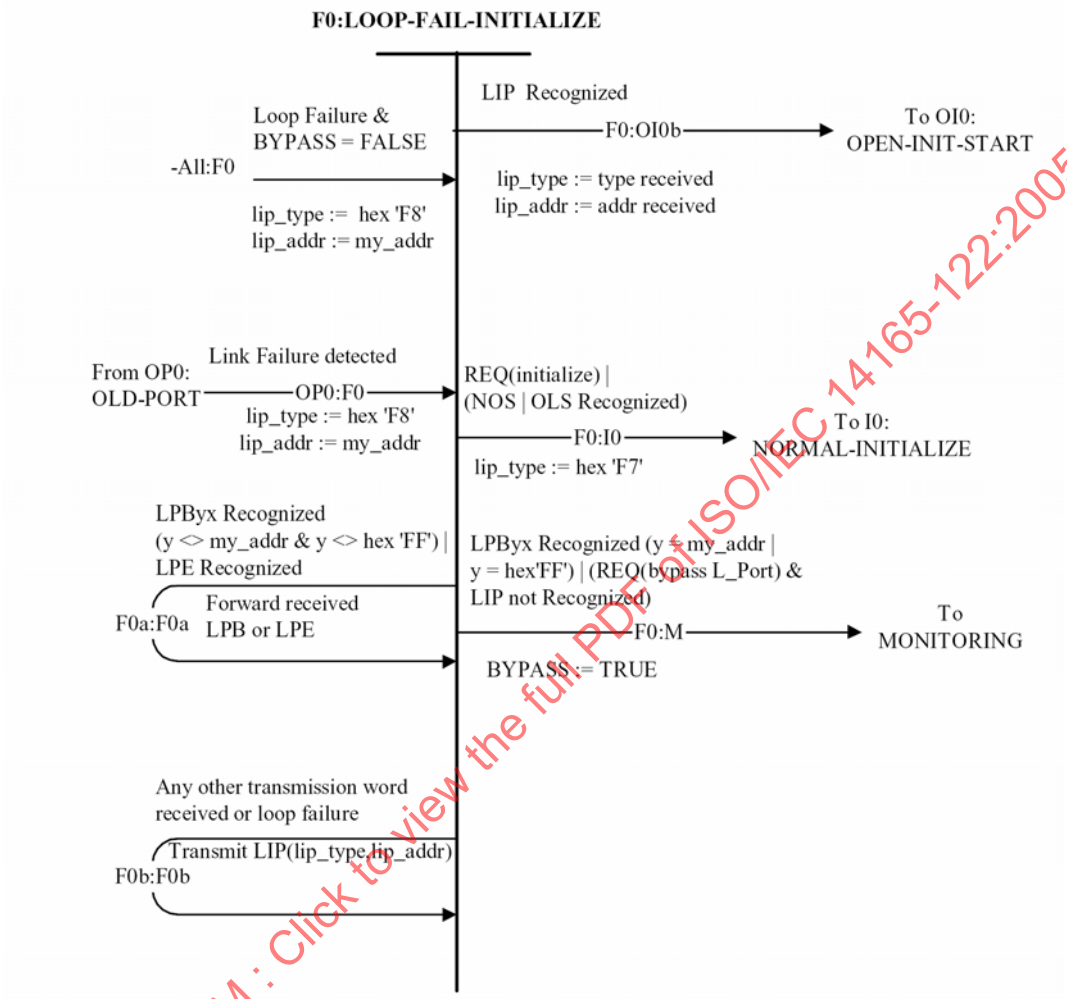


Figure 10 — Loop Fail Initialization state diagram

Loop Fail Initialization state diagram notes:

Transition All:F0 This transition is taken from any state other than P0:POWER-ON-INIT when a Loop Failure is detected and BYPASS is FALSE(0).

Transition F0:I0 This transition is taken from the LOOP-FAIL-INITIALIZE state to the NORMAL-INITIALIZE state when REQ(initialize) is asserted.

Transition F0:OI0b This transition is taken from the LOOP-FAIL-INITIALIZE state to the OPEN-INIT-START state when a LIP has been recognized by this L_Port, and the remainder of the INITIALIZATION process may be attempted.

Transition F0:I0 This transition is taken from the LOOP-FAIL-INITIALIZE state to the NORMAL-INITIALIZE state when NOS or OLS are recognized. This allows for waiting expire(3xAL_TIME) before transitioning to the optional OLD-PORT state.

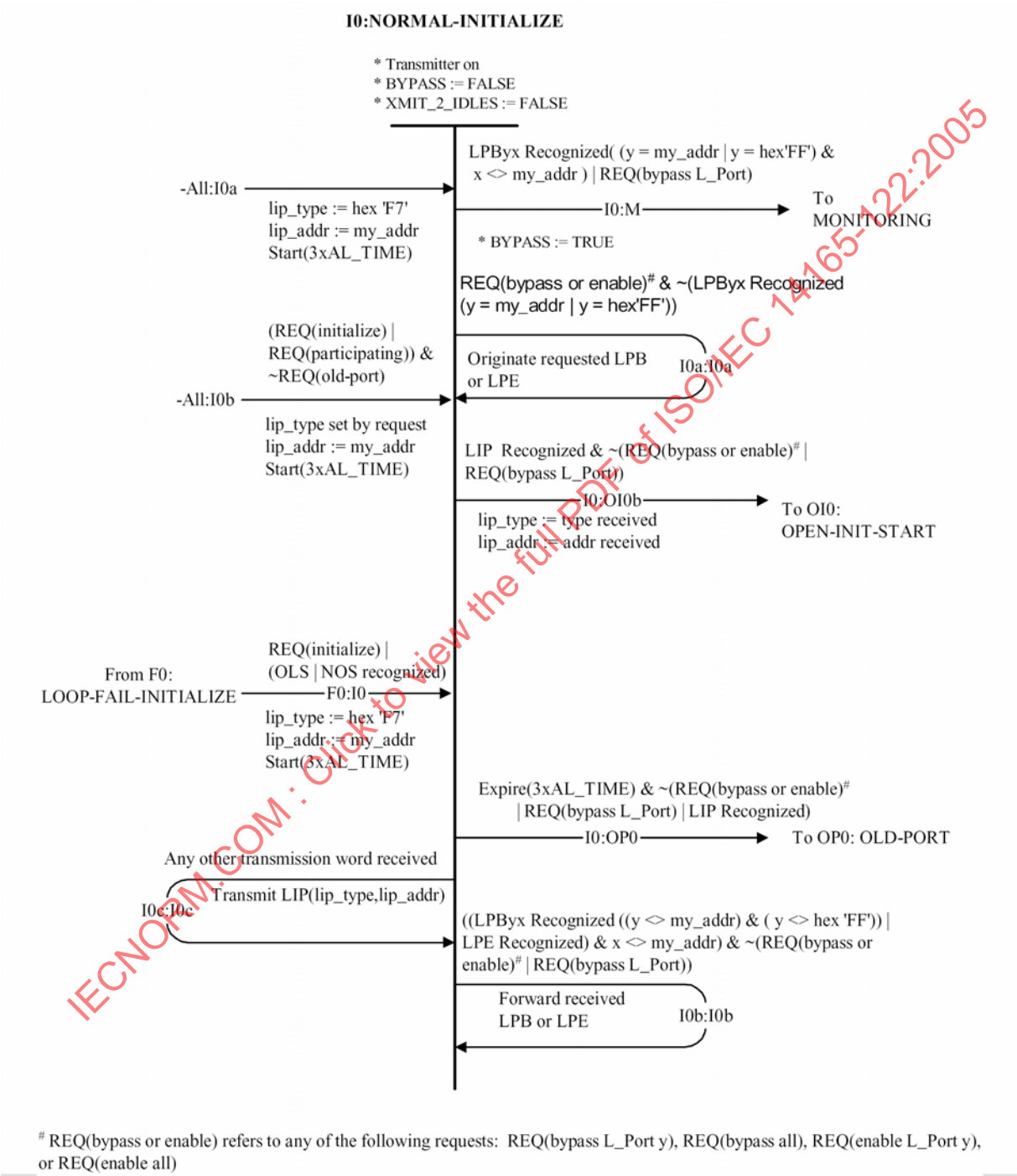
Transition F0:M This transition is taken from the LOOP-FAIL-INITIALIZE state to the MONITORING state when LPB for this AL_PA or LPBfx is recognized. The L_Port shall remain in the MONITORING state until it is enabled on the Loop and recognizes LIP, or REQ(initialize) is asserted.

Transition F0a:F0a When LPB that is not addressed to this L_Port or LPE is recognized, it shall be forwarded and the LPSM shall remain in the LOOP-FAIL-INITIALIZE state.

Transition F0b:F0b If any Transmission Word other than those explicitly identified in the state transitions from LOOP-FAIL-INITIALIZE is recognized, LIP(lip_type, lip_addr) shall be originated and the LPSM shall remain in the LOOP-FAIL-INITIALIZE state.

10.5.4.5 Normal Initialization state diagram

Figure 11 shows the normal initialization state diagram. This state diagram attempts to circulate LIP around the Loop before the address selection phase of initialization is begun.



REQ(bypass or enable) refers to any of the following requests: REQ(bypass L_Port y), REQ(bypass all), REQ(enable L_Port y), or REQ(enable all)

Figure 11 — Normal Initialization state diagram

Normal Initialization state diagram notes:

Unless this L_Port is attempting to manage other L_Port's Port Bypass Circuits, REQ(initialize) shall be removed upon entering the NORMAL-INITIALIZE state.

Transition All:I0a This transition is made from subsequent states of the INITIALIZATION process when an error in initialization is detected. The errors that are detected are protocol errors or LP_TOV timeout. The transition may actually be controlled from outside the LPSM, and therefore, may take an extended time.

Transition All:I0b This transition is taken from any state when REQ(participating) or REQ(initialize) is asserted. The type of LIP to be generated is indicated in the request.

Transition I0:OI0b This transition from the NORMAL-INITIALIZE state to the OPEN-INIT-START state is taken when LIP has been recognized, and the address selection phase of initialization can be started.

Transition I0:OP0 This transition from the NORMAL-INITIALIZE state to the optional OLD-PORT state is taken when LIP has been transmitted until expire(3xAL_TIME) without LIP being recognized and none of the listed REQs being asserted.

Transition I0:M This transition from the NORMAL-INITIALIZE state to the MONITORING state is taken when LPByx (where y = AL_PA of the L_Port) or LPBfx is recognized; or when REQ(bypass L_Port) is asserted. The L_Port shall remain in the MONITORING state until it is enabled on the Loop and recognizes LIP or REQ(initialize) is asserted while REQ(old-port) is not asserted.

Transition I0a:I0a When LPB that is not addressed to this L_Port or LPE is recognized, it shall be forwarded and the LPSM shall remain in the NORMAL-INITIALIZE state if none of the specified REQs are asserted.

Transition I0b:I0b If a request to bypass or enable (REQ(bypass L_Port y), REQ(bypass all), REQ(enable L_Port y), or REQ(enable all)) is asserted, and if LPB addressed to this L_Port is not recognized, the L_Port shall originate the requested LPB or LPE and remain in the NORMAL-INITIALIZE state.

Transition I0c:I0c If any Transmission Word other than those explicitly identified in the state transitions from NORMAL-INITIALIZE is recognized while none of the specified REQs are asserted, a LIP(lip_type, lip_addr) shall be originated and the LPSM shall remain in the NORMAL-INITIALIZE state.

10.5.4.6 OPEN-INIT state diagram

Figure 12 shows the OPEN-INIT state diagram. The OPEN-INIT state diagram is where the Loop Initialization Master (LIM) is determined for this Loop Initialization.

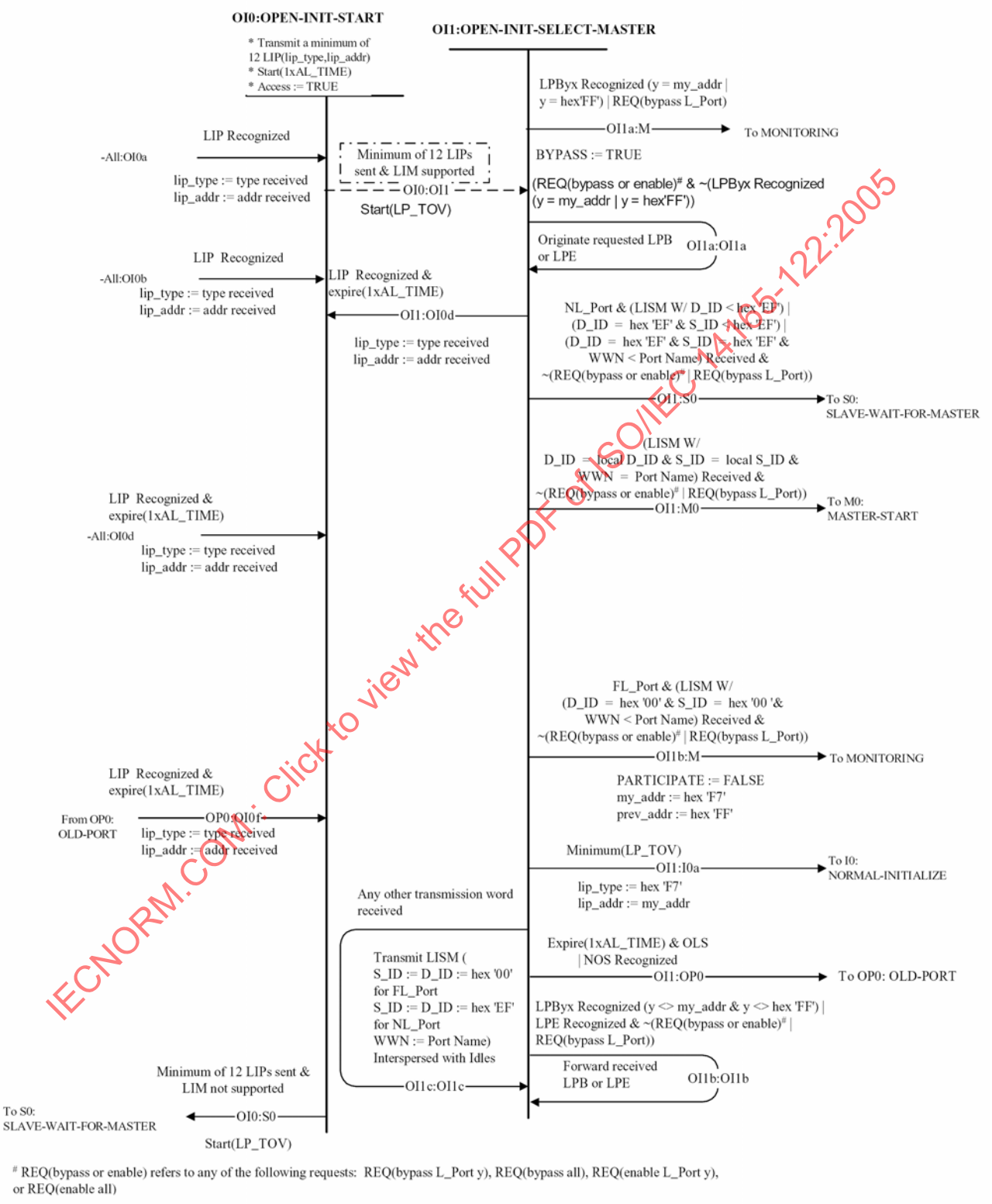


Figure 12 — OPEN-INIT state diagram

OPEN-INIT state diagram notes:

Transition AII:OI0a This transition is taken from any Loop Initialization state if LIP is recognized after the LIM has been selected or from any state outside the INITIALIZATION process if LIP is recognized.

Transition AII:OI0b This transition is the normal entrance to the OPEN-INIT-START state upon recognition of a (NonF8) LIP.

Transition AII:OI0d This is the re-entrance to the OPEN-INIT-START state upon the recognition of a after expire(1xAL_TIME) without successfully selecting a LIM.

Transition OP0:OI0f This is the entrance to the OPEN-INIT-START state upon the recognition of a LIP after expire(1xAL_TIME) in OLD-PORT state.

Transition OI0:S0 This transition from the OPEN-INIT-START state to the SLAVE-WAIT-FOR-MASTER state is taken after forwarding at least 12 received LIP(lip_type,lip_addr) if the L_Port is not capable of performing the LIM functions.

Transition OI1a:M This transition from the OPEN-INIT-SELECT-MASTER state to the MONITORING state is taken when LPByx (where y = AL_PA of the L_Port) or LPBfx is recognized; or when REQ(bypass L_Port) is asserted. The L_Port shall remain in the MONITORING state until it is enabled on the Loop and recognizes LIP or REQ(initialize) is asserted while REQ(old-port) is not asserted.

Transition OI1b:M This transition from the OPEN-INIT-SELECT-MASTER state to the MONITORING state is taken when an FL_Port does not become a LIM.

Transition OI1:I0a This transition from the OPEN-INIT-SELECT-MASTER state to the NORMAL-INITIALIZE state is taken when the L_Port after minimum(LP_TOV) while trying to select a LIM. This takes the L_Port back to attempting initialization.

Transition OI1:S0 This transition is taken when this L_Port has determined that it is not the LIM.

Transition OI1:M0 This transition is taken when this L_Port has determined that it is the LIM.

Transition OI1:OP0 This transition from the OPEN-INIT-SELECT-MASTER state to the optional OLD-PORT state is taken when OLS or NOS is recognized after expire(1xAL_TIME).

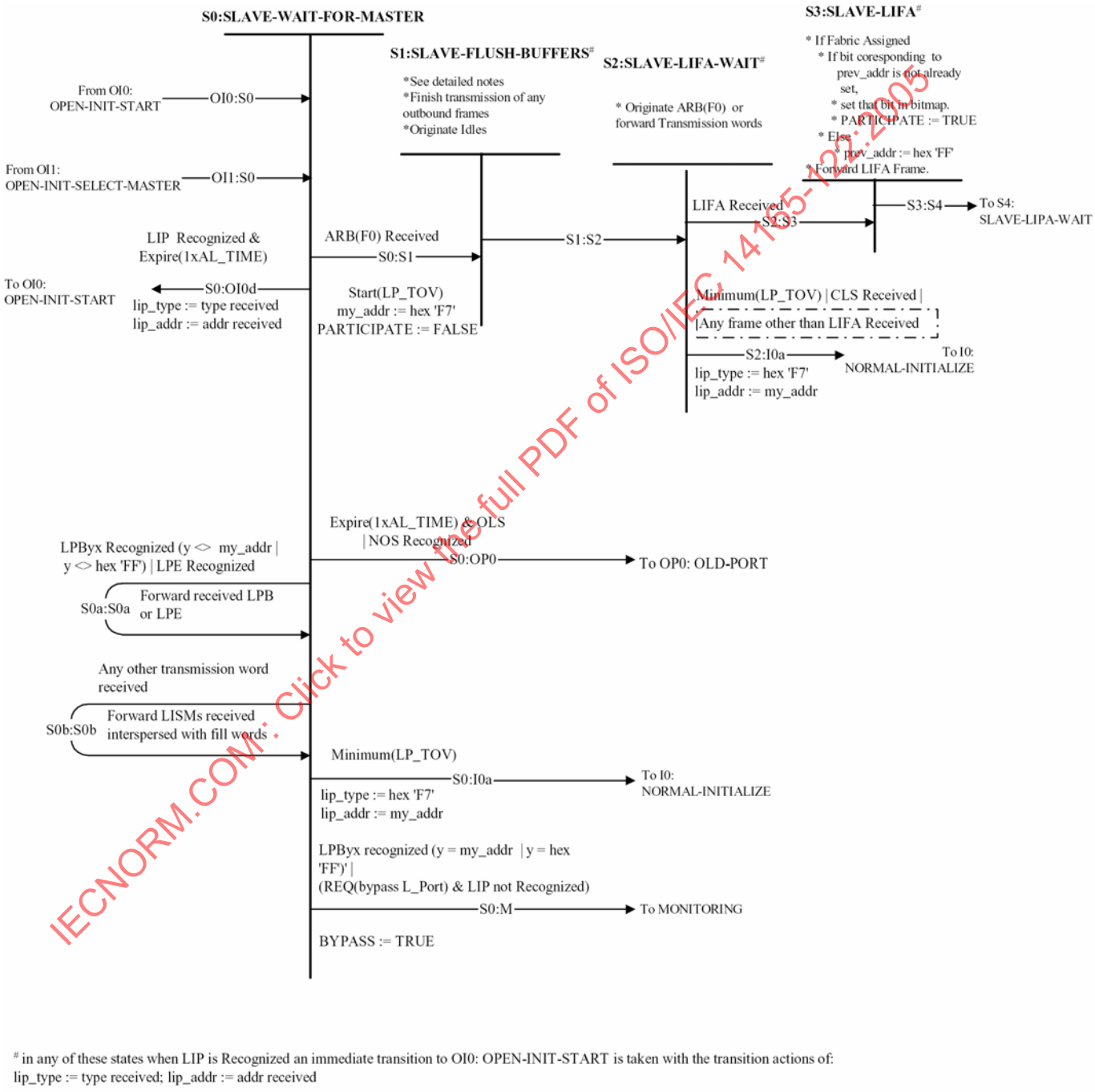
Transition OI1a:OI1a If a request to bypass or enable (REQ(bypass L_Port y), REQ(bypass all), REQ(enable L_Port y), or REQ(enable all)) is asserted, and if LPB addressed to this L_Port is not recognized, the L_Port shall originate the requested LPB or LPE and remain in the OPEN-INIT-SELECT-MASTER state.

Transition OI1b:OI1b When an LPB that is not addressed to this L_Port or an LPE is recognized, it shall be forwarded and the LPSM shall remain in the OPEN-INIT-SELECT-MASTER state, if none of the REQs are asserted.

Transition OI1c:OI1c If any Transmission Word other than those explicitly identified in the state transitions from the OPEN-INIT-SELECT-MASTER state is recognized while none of the specified REQs are asserted, LISM shall be originated and the LPSM shall remain in the OPEN-INIT-SELECT-MASTER state.

10.5.4.7 Slave Initialization state diagram

Figure 13 shows the Slave Initialization state diagram. This machine is where the L_Port acquires an address if it is not the LIM. The L_Port enters this state when it has determined that there is another L_Port with higher priority on the Loop or the LIM function is not supported by this L_Port.



in any of these states when LIP is Recognized an immediate transition to OI0: OPEN-INIT-START is taken with the transition actions of:
lip_type := type received; lip_addr := addr received

Figure 13 — Slave Initialization state diagram

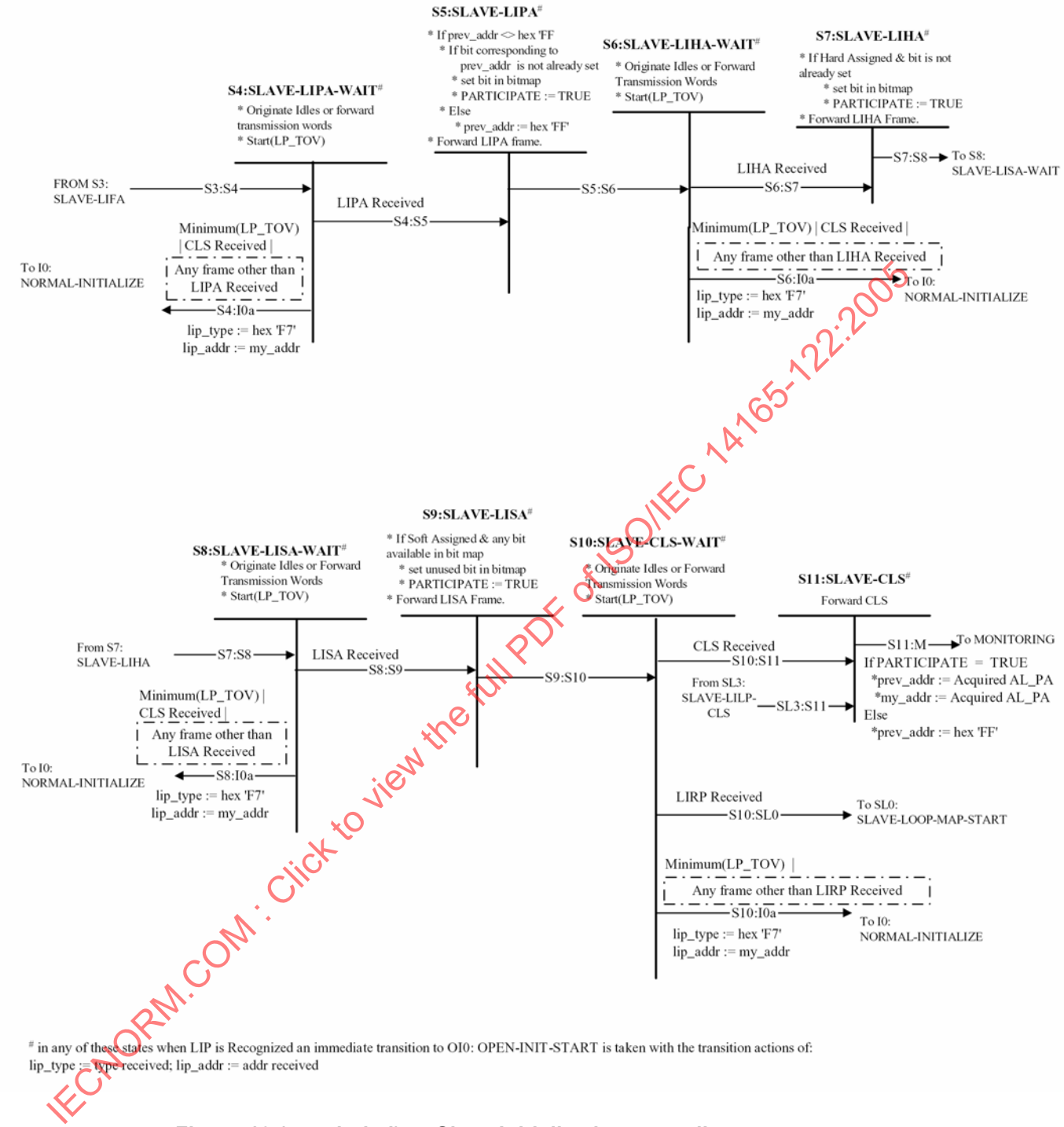


Figure 13 (concluded) — Slave Initialization state diagram

Slave Initialization state diagram notes:

State S0: SLAVE-WAIT-FOR-MASTER is used to forward received LISM frames. Implementations may directly forward frames as they are received; however, not all LISM frames will necessarily be received due to buffer limitations. Additionally, the L_Port may store the most recently received LISM frame and transmit it interspersed with Idles until another LISM is received and stored in the buffer. This may yield LISM transmission at a different rate than reception.

NOTE Because this state is being timed by LP_TOV, it is important to update the LISM being forwarded. If every device takes more than 15 ms to forward each LISM received, and there are 128 devices on the Loop, the delay to win LIM would exceed LP_TOV.

State S1: SLAVE-FLUSH-BUFFERS is used to ensure that the L_Port is ready to receive the LIFA frame that may be received shortly after forwarding the ARB(F0) around the Loop. In this state, the L_Port should complete sending any frame that it was transmitting when ARB(F0) was received and transmit Idle until it has a buffer to receive the LIFA frame. Upon reception of LIP in this state, the transition All:OI0a shall be taken back to the OPEN-INIT-START state.

NOTE Because this state and the next is being timed by LP_TOV, it is important that the buffers be flushed quickly, and ARB(F0) forwarded quickly. If each device takes more than 15 ms to flush buffers and originate or forward the LIFA frame, the total delay, would exceed LP_TOV.

Transition OI0:S0 This transition is the starting point for the Slave Initialization state diagram when the LIM is not supported. It is reached directly from the OPEN-INIT-START state.

Transition OI1:S0 This transition is the starting point for the Slave Initialization state diagram when the LIM is supported (but the L_Port yielded to another L_Port). It is reached when the L_Port determines in the OPEN-INIT-SELECT-MASTER state that it is not the LIM.

Transition S0:OI0d This transition is taken when a LIP is recognized in the SLAVE-WAIT-FOR-MASTER state after expire(1xAL_TIME). This causes the L_Port to return to the OPEN-INIT-START state.

Transition All:OI0a This transition is taken when any LIP is recognized in the SLAVE-FLUSH-BUFFERS, SLAVE-LIFA-WAIT, SLAVE-LIFA, SLAVE-LIPA-WAIT, SLAVE-LIPA, SLAVE-LIHA-WAIT, SLAVE-LIHA, SLAVE-LISA-WAIT, SLAVE-LISA, SLAVE-CLS-WAIT, SLAVE-CLS states. This causes the L_Port to return to the OPEN-INIT-START state.

Transition S0:OP0 This transition from the SLAVE-WAIT-FOR-MASTER state to the optional OLD-PORT state is taken when OLS or NOS is recognized after expire(1xAL_TIME).

Transition All:I0a This transition occurs in any of the indicated states when an error is detected in the address selection machine. The errors that are detected are protocol errors or LP_TOV timeouts. The transition may actually be controlled from outside the LPSM and therefore may take an extended time. These transitions go back to the NORMAL-INITIALIZE state and attempt to begin initialization again using LIP(F7).

Transition S10:SL0 This transition is made after all addresses have been assigned and all L_Ports are capable of generating an AL_PA position map. This transition is to the SLAVE-LOOP-MAP-START state in the Slave AL_PA position map generation state diagram.

Transition S11:M This transition to the MONITORING state is the completion of Loop Initialization. At this point the L_Port is participating in the Loop if PARTICIPATE is TRUE(1), or it remains in the MONITORING state and forgets any previously acquired AL_PA as a Non-Participating L_Port if PARTICIPATE is FALSE(0).

Transition SL3:S11 This transition is from the SLAVE-LILP-CLS state in the Slave AL_PA position map state diagram after it has received CLS. It transitions here, generates CLS and finishes the INITIALIZATION process.

Transition S0a:S0a When an LPB that is not addressed to this L_Port or an LPE is recognized, it shall be forwarded, and the LPSM shall remain in the SLAVE-WAIT-FOR-MASTER state.

Transition S0b:S0b If any Transmission Word other than those explicitly identified in the state transitions from SLAVE-WAIT-FOR-MASTER is recognized, received LISMs are forwarded and the LPSM shall remain in the SLAVE-WAIT-FOR-MASTER state. If no LISM frames have been received, the CFW shall be transmitted.

10.5.4.8 Slave AL_PA position map state diagram

Figure 14 shows the Slave AL_PA position map state diagram. All FC-AL-2 L_Ports must implement this; however, if there are FC-AL-1 L_Ports on the Loop this state diagram may not be used.

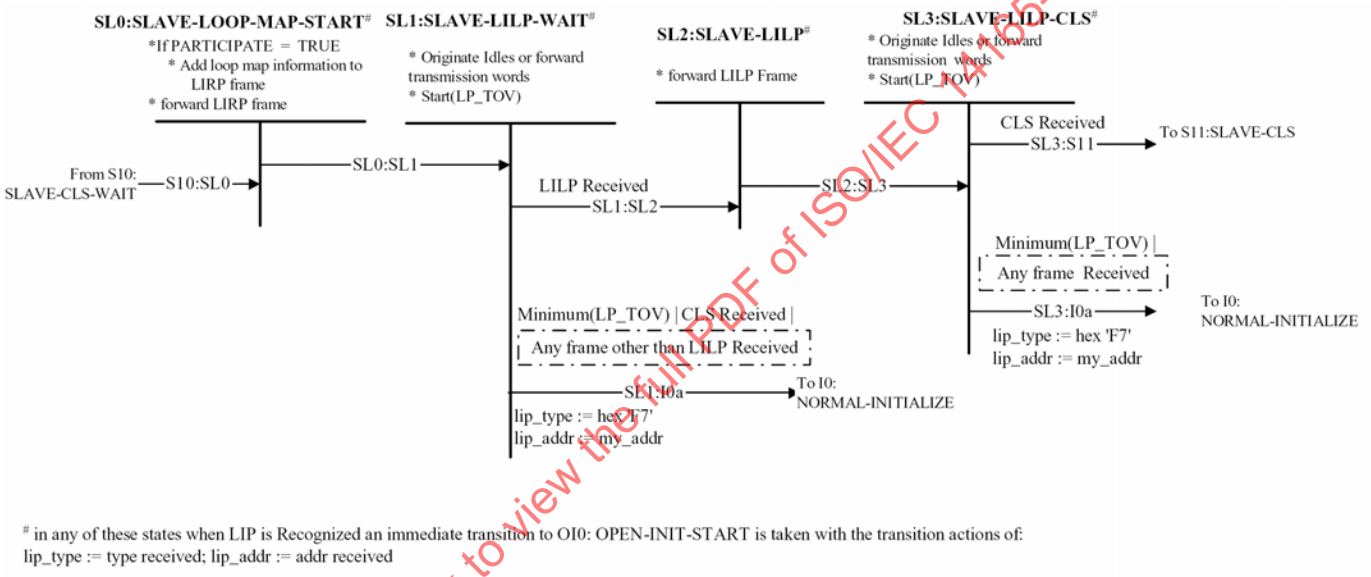


Figure 14 — Slave AL_PA position map state diagram

Slave AL_PA position map state diagram notes:

Transition S10:SL0 This is the main entry point to the Slave AL_PA position map state diagram from the Slave Initialization state diagram.

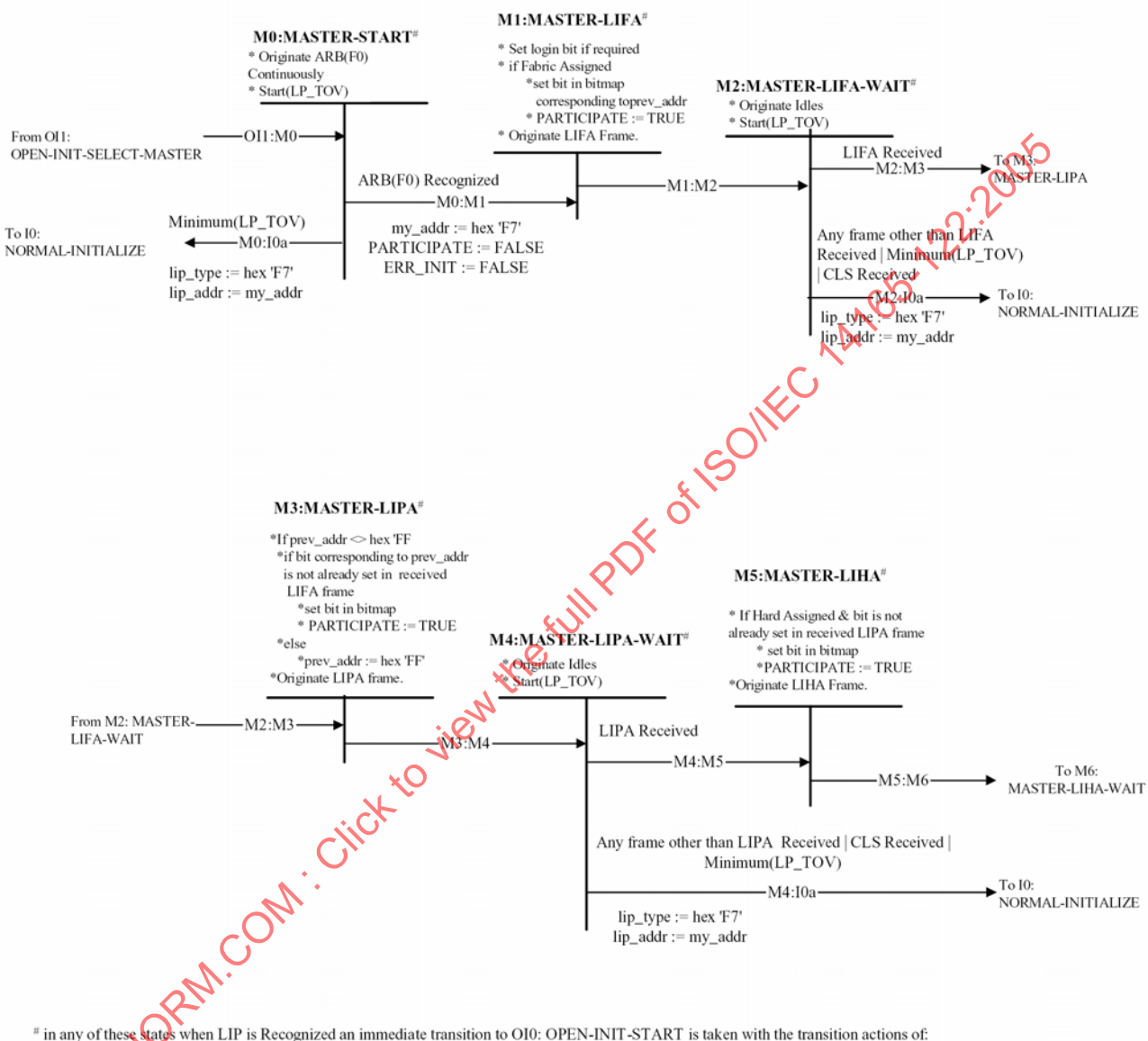
Transition All:Ol0a This transition is taken when LIP is recognized in the SLAVE-LOOP-MAP-START, SLAVE-LILP-WAIT, SLAVE-LILP, SLAVE-LILP-CLS states. This causes the L_Port to return to the OPEN-INIT-START state.

Transition All:I0a This transition is taken when an error is detected in any of the indicated states of the address selection machine. The errors that are detected are protocol errors or LP_TOV timeouts. The transition may actually be controlled from outside the LPSM and therefore may take an extended time. These transitions go back to the NORMAL-INITIALIZE state and attempt to begin initialization again.

Transition SL3:S11 This transition is taken to S11:SLAVE-CLS, when the AL_PA position map process is completed and CLS has been received. It is the final transition before Loop Initialization is completed in the Slave Initialization state diagram.

10.5.4.9 Master Initialization state diagram

Figure 15 shows the Master Initialization state diagram. This machine is where the L_Port acquires an AL_PA if it is the LIM. The L_Port enters this state when it has received its own LISM frame back.



in any of these states when LIP is Recognized an immediate transition to OI0: OPEN-INIT-START is taken with the transition actions of:
lip_type := type received; lip_addr := addr received

Figure 15 — Master Initialization state diagram

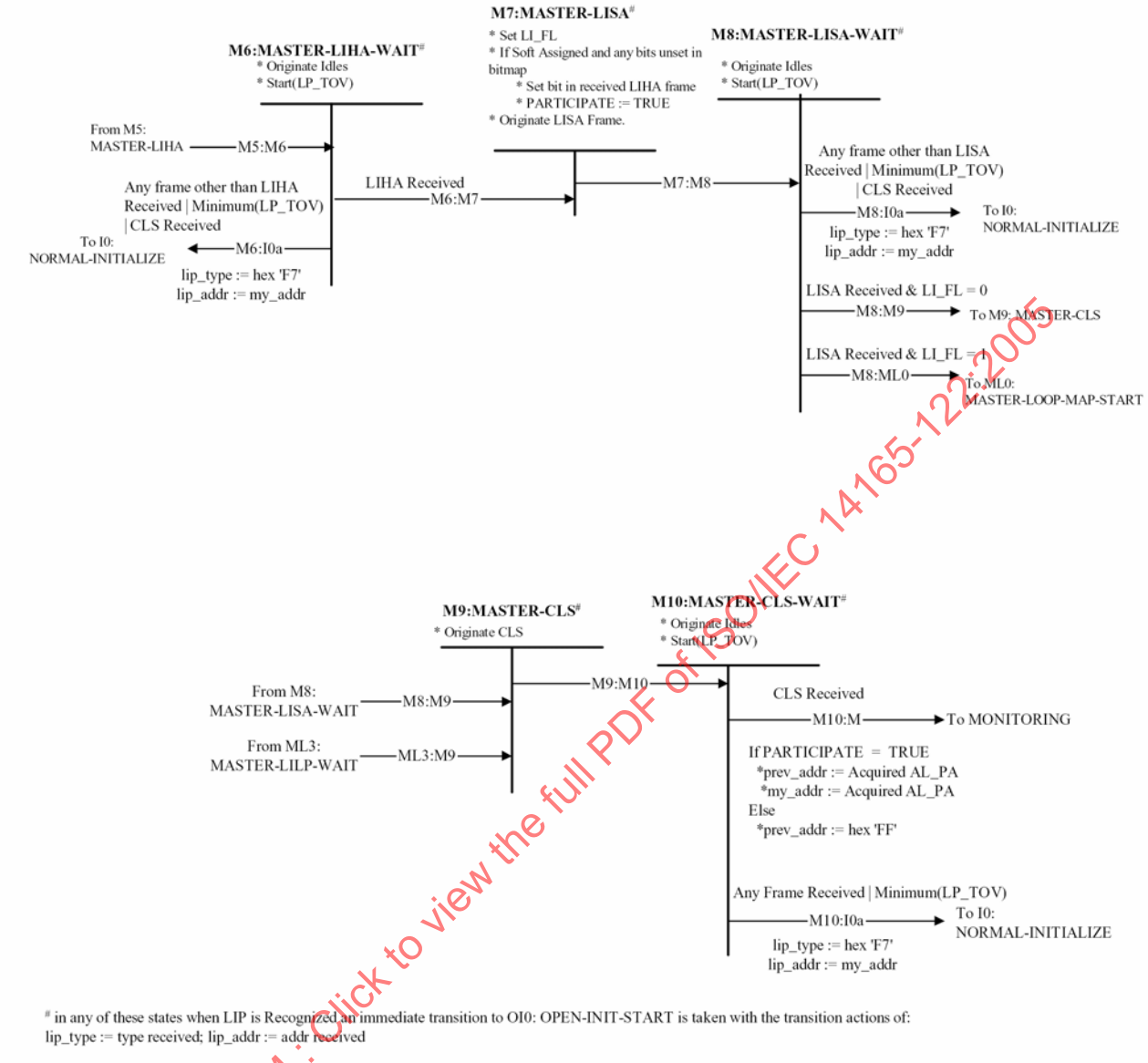


Figure 15 (concluded) — Master Initialization state diagram

Master Initialization state diagram notes:

Transition OI1:M0 This transition is the starting point for the Master Initialization state diagram. It is reached when the L_Port determines in the OPEN-INIT-SELECT-MASTER state that it is the LIM.

Transition All:OI0a This transition is taken when LIP is recognized in the MASTER-LIFA, MASTER-LIFA-WAIT, MASTER-LIPA, MASTER-LIPA-WAIT, MASTER-LIHA, MASTER-LIHA-WAIT, MASTER-LISA, MASTER-LISA-WAIT, MASTER-CLS, MASTER-CLS-WAIT states. This causes the L_Port to return to the OPEN-INIT-START state.

Transition AII:I0a This transition is taken when an error is detected in any of the indicated states of the address selection machine. The errors that are detected are protocol errors or LP_TOV timeouts. The transition may actually be controlled from outside the LPSM and therefore may take an extended time. These transitions go back to the NORMAL-INITIALIZE state and attempt to begin initialization again.

Transition M8:ML0 This transition to the MASTER-LOOP-MAP-START state is taken after all addresses have been assigned and all L_Ports are capable of generating an AL_PA position map.

Transition M10:M This transition to the MONITORING state is the completion of Loop Initialization. At this point the L_Port is participating in the Loop if PARTICIPATE is TRUE(1), or it remains in the MONITORING state and forgets any previously acquired AL_PA as a Non-Participating L_Port if PARTICIPATE is FALSE(0).

Transition ML3:M9 This transition is taken from the MASTER-LILP-WAIT state, and indicates that Loop Initialization is ready to be completed. The LIM shall generate CLS to complete Loop Initialization.

10.5.4.10 Master AL_PA position map state

Figure 16 shows the Master AL_PA position map state diagram. All FC-AL-2 L_Ports shall implement this; however, if there are FC-AL-1 L_Ports on the Loop this state diagram may not be used.

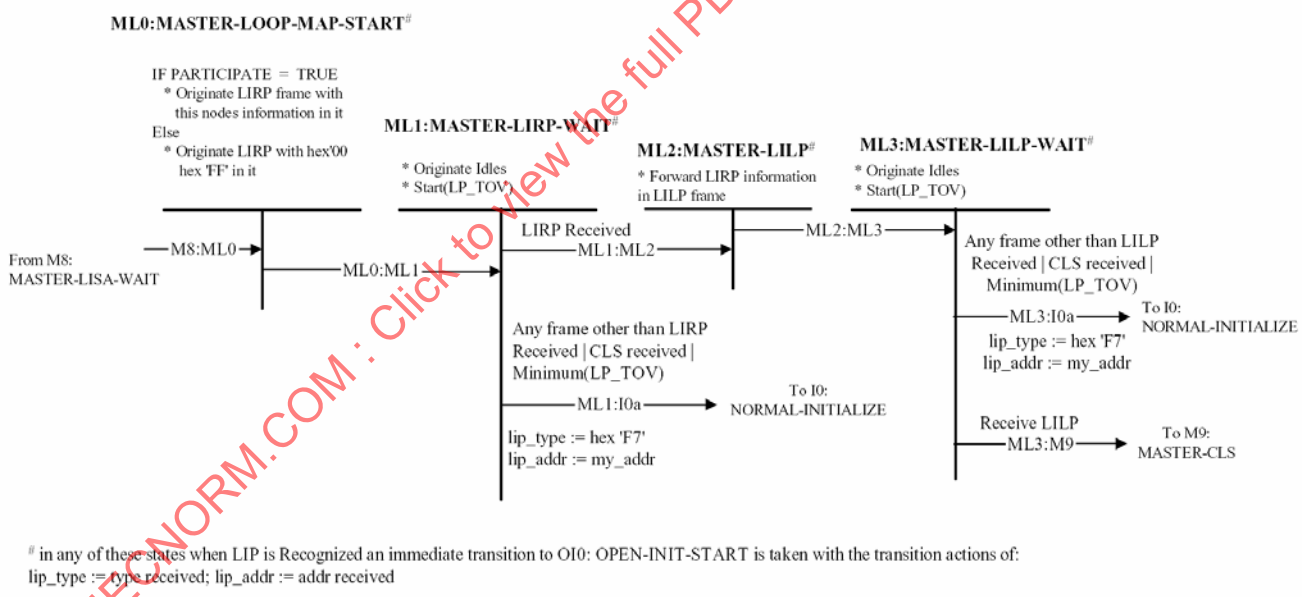


Figure 16 — Master AL_PA position map state diagram

Master AL_PA position map state diagram notes:

Transition M8:ML0 This is the main entry point to the Master AL_PA position map state diagram from the Master Initialization state diagram.

Transition AII:OI0a This transition is taken when LIP is recognized in the MASTER-LOOP-MAP-START, MASTER-LIRP-WAIT, MASTER-LILP, MASTER-LILP-WAIT states. This causes the L_Port to return to the OPEN-INIT-START state.

Transition AII:I0a This transition is taken when an error is detected in any of the indicated states of the address selection machine. The errors that are detected are protocol errors or LP_TOV timeouts. The transition may actually be controlled from outside the LPSM and therefore may take an extended time. These transitions go back to the NORMAL-INITIALIZE state and attempt to begin initialization again.

Transition ML3:M9 This transition is taken to MASTER-CLS, when the AL_PA position map process is completed.

Annex A
(normative)

L_Port Elasticity buffer management

A.0 Overview

This annex defines the L_Port elastic buffer and the clock skew management rules for inserting and deleting Fill Words. The elasticity buffer provides buffering between the receiver input and the transmitter to prevent over-run and under-run conditions at the transmitter. To prevent L_Ports from being starved of opportunities to delete and minimize buffer requirements in all L_Ports, a two priority algorithm for clock skew management delete operations is incorporated. For a description of the elasticity buffer function and example, see annex G.

A.1 L_Port elasticity buffer implementation

The elasticity buffer shall be implemented as a first-in-first-out (FIFO) device with the input coming from the receiver logic and the output going to the transmitter logic. An example of the elasticity buffer is illustrated in Figure A.1. Buffering required for clock resynchronization is not shown.

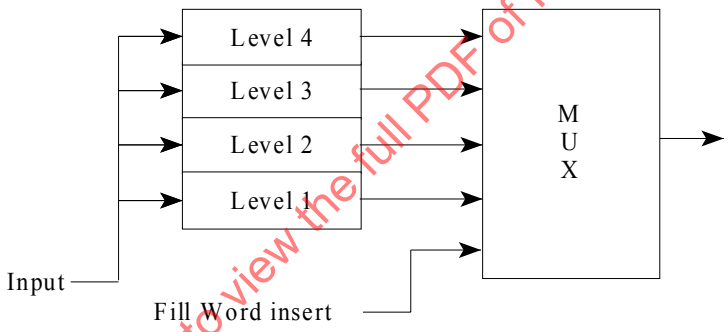


Figure A.1 — Elasticity buffer

Only valid Transmission Words are entered into the elasticity buffer. The information content of the valid Transmission Words entered into the buffer is called *valid information*. The buffer is divided into four levels. Each level represents buffering for a clock skew management state.

The buffer may be implemented either in bit, character, half word (two characters) or word wide units. The amount of valid information in the buffer is a count of the units entered into the buffer minus the units removed at the output.

NOTE It is recommended to use either a character or half word wide buffer. Bit wide implementations require a very high logic speeds for implementation. Word wide implementations do not provide a fine enough unit of measuring the level of valid information in the buffer.

A.2 Clock skew management

Clock skew management inserts and deletes Transmission Words to control the amount of valid information in the buffer. These operations are performed outside of FC-2 frames to allow the frames to be forwarded without modification. For clock skew management, Fill Words or any Ordered Set defined for use as a Primitive Sequence shall be treated equally.

The insertion of Fill Words is required when the receive clock is slower than the transmit clock to prevent buffer under-run. Deletion of Fill Words is required when the receive clock is faster than the transmit clock to prevent buffer over-run.

A.3 Clock skew management states

A.3.0 Clock skew overview

Clock skew management has 4 states as shown in Figure A.2: insertion pending, quiescent, low priority deletion pending, and high priority deletion pending. The current state is determined by the amount of valid information in the buffer.

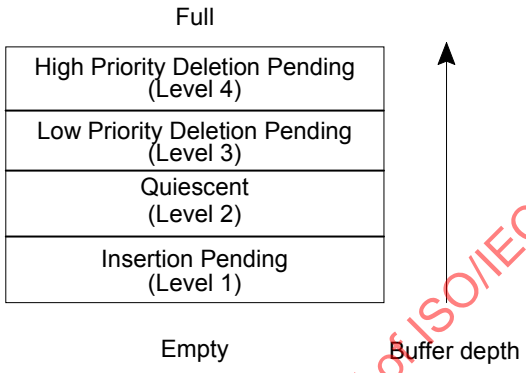


Figure A.2 — Clock skew management states

A.3.1 Insertion pending

To allow an FC-2 frame to be transmitted unmodified, at least a minimum amount of valid information must be in the buffer to ensure buffer under-run will not occur during the frame transmission.

Rule for insertion: When the amount of valid information is less than level 2, in level 1, the L_Port shall insert the current Fill Word immediately after any Fill Word.

A.3.2 Quiescent

When the amount of valid information in the buffer is greater than the level requiring an insertion and less than the level requiring a deletion, in level 2, the clock skew management algorithm is in the quiescent state. No requests to change the buffer depth are pending. This is the nominal state of the clock skew management.

A.3.3 Deletion pending

At least a minimum of free space must be in the buffer to prevent buffer over-run during frame retransmission. Transmission Words may be deleted to reduce the depth of the buffer to provide free space.

Deletion of Transmission Words is more difficult than insertion. When a deletion is required between frames, e.g., in an inter-frame gap, a minimum number of Primitive Signals shall be maintained between frames to meet FC-FS requirements and the Idle Primitive Signals which reset the fairness window shall be propagated.

When frames are originated onto the Loop, the FC-PH requirement of six Primitive Signals between frames is followed. The inter-frame gap may be reduced to 2 Fill Words plus other Primitive Signals by clock skew management. To prevent Ports from being starved of opportunities to delete, a two priority algorithm shall be used.

A.3.3.1 Low priority deletion pending

When the amount of valid information in the buffer triggers a request to delete, the low priority rules for deletion are followed. The low priority rules protect some Fill Words in inter-frame gaps for L_Ports requiring a more critical delete after all the low priority deletes have occurred.

Low Priority Rules: When the amount of valid information reaches level 3, the L_Port shall:

— after 4 Fill Words with no intervening non-Ordered Set (data words), delete the next Fill Word,

NOTE 1 A non-Ordered Set indicates an intervening frame. By detecting non-Ordered Sets, the L_Port is not required to detect SOF and EOF frame delimiters. New frame delimiters may be defined in the future.

— if the CFW changes to Idle while a delete is pending, not delete the first Idle.

NOTE 2 This Idle is protected to allow the LPSM to manage the fairness window as required.

After a low priority Transmission Word delete, the L_Port shall

— enter the low priority state and wait 4 Fill Words before another delete or

— enter the quiescent state with no delete pending.

Examples:

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

Legend:
EOF = End of Frame                    AR  = ARByx
SOF = Start of Frame                  ID  = Idle
FW  = Fill Word (ARByx or Idle)       RR  = R_RDY
```

A.3.3.2 High priority deletion pending

L_Ports that are close to over-running their buffers follow the high priority rules for deletion.

High Priority Rules: When the amount of valid information reaches level 4, the L_Port shall:

— after 2 Fill Words with no intervening non-Ordered Set (data words), the L_Ports deletes the next Fill Word and

— if the CFW changes to Idle while a delete is pending, the L_Port shall not delete the first Idle.

After a high priority Transmission Word delete, the L_Port shall

— enter the low priority state and wait 4 Fill Words before another delete or

— re-enter the high priority state and wait 2 Fill Words before another delete depending on buffer free space.

Examples:

```
EOF FW FW FW FW FW FW SOF
          ^  ^  ^  ^              May delete one for high priority

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

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

EOF FW FW FW FW FW FW SOF
          ^        ^          May delete both (third and sixth) for high priority
```

Legend

EOF = End of Frame
SOF = Start of Frame
FW = Fill Word (ARByx or Idle)
AR = ARByx
ID = 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 0,25 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

B.0 Overview

The two examples in this annex use seven of the nine states in the LPSM and twelve of the twenty-two state transitions as defined in 8.4. Of the ten unused state transitions, four are for rare events or error handling. Therefore, much of the Loop protocol is covered in these two simple examples.

B.1 L_Port initialization example

The general, error free procedure for taking the LPSM of a Public NL_Port through Loop Initialization to the point of Fabric Login follows (see clause 8 for reference items and LPSM transitions):

— the NL_Port powers on and attempts to join the Loop;

— the NL_Port may use a trusted AL_PA (since it does not have a valid AL_PA) to instruct the LPSM to arbitrate (REQ(arb own AL_PA)) and to initialize (REQ(initialize));

the LPSM makes transition (01);

— the LPSM, now in the ARBITRATING state, begins to replace any Idle or ARB(val) (where val is higher than the trusted AL_PA) with ARB(val) (where val is a trusted AL_PA). The LPSM monitors its inbound fibre for the ARB(val) which it transmitted. (See 8.4.3 item 14 for details);

the LPSM makes transition (12);

— the LPSM, now in the ARBITRATION WON state, detects (REQ(initialize)) (see 8.4.3 item 15 for details);

the LPSM makes transition (28);

— the LPSM, now in the INITIALIZATION process, performs Loop Initialization as described in clause 10 and waits for CLS to indicate that the INITIALIZATION process has completed;

— the LPSM detects CLS;

the LPSM makes transition (80);

— the NL_Port, now in the MONITORING state, is ready to execute a Fabric Login if it is a Public NL_Port (see 10.5, step (5)).

The complete list of state transitions and states in the order used is: (01), 1: ARBITRATING; (12) 2: ARBITRATION WON; (28), 8: INITIALIZATION process; (80), 0: MONITORING.