Document Number: DSP0222

Date: 2009-07-21

Version: 1.0.0

# Network Controller Sideband Interface (NC-SI) Specification

**Document Type: Specification**

**Document Status: DMTF Standard**

**Document Language: E**

# CONTENTS

# Figures



# Tables

# Foreword

The *Network Controller Sideband Interface (NC-SI) Specification* (DSP0222) was prepared by the PMCI Working Group.

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability.

# Introduction

In out-of-band management environments, the interface between the out-of-band Management Controller and the Network Controller is critical. This interface is responsible for supporting communication between the Management Controller and external management applications. Currently there are multiple such proprietary interfaces in the industry, leading to inconsistencies in implementation of out-of-band management.

The goal of this specification is to define an interoperable sideband communication interface standard to enable the exchange of management data between the Management Controller and Network Controller. The Sideband Interface is intended to provide network access for the Management Controller, and the Management Controller is expected to perform all the required network functions.

The specification defines the following primary characteristics of the interface:

- the physical and electrical characteristics of the interface
- the protocol and commands necessary for the operation of the interface

The specification is primarily intended for architects and engineers involved in the development of network interface components and Management Controllers that will be used in providing out-of-band management.

# Network Controller Sideband Interface (NC-SI) Specification

## 1 Scope

This specification defines the functionality and behavior of the Sideband Interface responsible for connecting the Network Controller to the Management Controller. It also outlines the behavioral model of the network traffic destined for the Management Controller from the Network Controller.

This specification defines the following three aspects of the Network Controller Sideband Interface (NC-SI):

- behavior of the interface, which include its operational states as well as the states of the associated components
- the electrical characteristics of the interface
- the payloads and commands of the communication protocol supported over the interface

The scope of this specification is limited to addressing only a single Management Controller communicating with one or more Network Controllers.

Only the topics that may affect the behavior of the Network Controller or Management Controller, as it pertains to the Sideband Interface operations, are discussed in this specification.

## 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.

IEEE 802.3*, 802.3™ IEEE Standard for Information technology— Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications,* December 2005, http://www.ieee.org/portal/site

IEEE 802.1Q*, IEEE 802.1Q-2005 IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks*, http://www.ieee.org/portal/site. This standard defines the operation of Virtual LAN (VLAN) Bridges that permit the definition, operation and administration of Virtual LAN topologies within a Bridged LAN infrastructure.

IETF RFC2131, *Dynamic Host Configuration Protocol* (DHCP), March 1997, http://www.ietf.org/rfc/rfc2131.txt

IETF RFC2373, *IP Version 6 Addressing Architecture*, July 1998, http://www.ietf.org/rfc/rfc2373.txt

IETF RFC2461, *Neighbor Discovery for IP Version 6 (IPv6)*, December 1998, http://www.ietf.org/rfc/rfc2461.txt

IETF RFC2464, *Transmission of IPv6 Packets over Ethernet Networks*, December 1998, http://www.ietf.org/rfc/rfc2464.txt

IETF RFC3315, *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*, July 2003, http://www.ietf.org/rfc/rfc3315.txt

ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards,* http://isotc.iso.org/livelink/livelink?func=ll&objId=4230456&objAction=browse&sort=subtype

Reduced Media Independent Interface (RMII) Consortium, *RMII Specification*, revision 1.2, March 20, 1998, http://www.national.com/appinfo/networks/files/rmii_1_2.pdf

# 3 Terms and Definitions

For the purposes of this document, the following terms and definitions apply.

## 3.1 Requirement Term Definitions

This clause defines key phrases and words that denote requirement levels in this specification.

**3.1**
**conditional**
indicates that an item is required under specified conditions

**3.2**
**deprecated**
indicates that an element or profile behavior has been outdated by newer constructs

**3.3**
**mandatory**
indicates that an item is required under all conditions

**3.4**
**may**
indicates that an item is truly optional

NOTE: An implementation that does not include a particular option shall be prepared to interoperate with another implementation that does include the option, although perhaps with reduced functionality. An implementation that does include a particular option shall be prepared to interoperate with another implementation that does not include the option (except for the feature that the option provides).

**3.5**
**may not**
indicates flexibility of choice with no implied preference

**3.6**
**not recommended**
indicates that valid reasons may exist in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and carefully weighed before implementing any behavior described with this label

**3.7**
**obsolete**
indicates that an item was defined in prior specifications but has been removed from this specification

**3.8**
**optional**
indicates that an item is not mandatory, conditional, or prohibited

**3.9**
**recommended**
indicates that valid reasons may exist in particular circumstances to ignore a particular item, but the full implications should be understood and carefully weighed before choosing a different course

**3.10**
**required**
indicates that the item is an absolute requirement of the specification

**3.11**
**shall**
indicates that the item is an absolute requirement of the specification

**3.12**
**shall not**
indicates that the item is an absolute prohibition of the specification

**3.13**
**should**
indicates that valid reasons may exist in particular circumstances to ignore a particular item, but the full implications should be understood and carefully weighed before choosing a different course

**3.14**
**should not**
indicates that valid reasons may exist in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and carefully weighed before implementing any behavior described with this label

## 3.2 NC-SI Term Definitions

For the purposes of this document, the following terms and definitions apply.

**3.2.1**
**Frame**
a data packet of fixed or variable length that has been encoded for digital transmission over a node-to-node link

*Frame* is used in references to IEEE 802.3 Frames. *Packet* is used in all other references.

**3.2.2**
**Packet**
a formatted block of information carried by a computer network

*Frame* is used in references to IEEE 802.3 Frames. *Packet* is used in all other references.

**3.2.3**
**External Network Interface**
the interface of the Network Controller that provides connectivity to the external network infrastructure; also known as *port*

**3.2.4**
**Internal Host Interface**
the interface of the Network Controller that provides connectivity to the host operating system running on the platform

**3.2.5**

**Management Controller**

an intelligent entity composed of hardware/firmware/software that resides within a platform and is responsible for some or all of the management functions associated with the platform; also known as BMC and Service Processor

**3.2.6**

**Network Controller**

the component within a system that is responsible for providing connectivity to an external Ethernet network

**3.2.7**

**Remote Media**

a manageability feature that enables remote media devices to appear as if they are attached locally to the host

**3.2.8**

**Network Controller Sideband Interface**

**NC-SI**

the interface of the Network Controller that provides network connectivity to a Management Controller; also shown as *Sideband Interface* or *NC-SI* as appropriate in the context

**3.2.9**

**Integrated Controller**

a Network Controller device that supports two or more channels for the NC-SI that share a common NC-SI physical interface (for example, a Network Controller that has two or more physical network ports and a single NC-SI bus connection)

**3.2.10**

**Multi-drop**

refers to the situation in which multiple physical communication devices share an electrically common bus and a single device acts as the master of the bus and communicates with multiple “slave” or “target” devices

Related to NC-SI, a Management Controller serves the role of the master, and the Network Controllers are the target devices.

**3.2.11**

**Point-to-Point**

refers to the situation in which only a single Management Controller and single Network Controller package are used on the bus in a master/slave relationship, where the Management Controller is the master

**3.2.12**

**Channel**

the control logic and data paths that support NC-SI Pass-through operations through a single network interface (port)

A Network Controller that has multiple network interface ports can support an equivalent number of NC-SI channels.

**3.2.13**

**Package**

one or more NC-SI channels in a Network Controller that share a common set of electrical buffers and common electrical buffer controls for the NC-SI bus

Typically, a single, logical NC-SI package exists for a single physical Network Controller package (chip or module). However, this specification allows a single physical chip or module to hold multiple NC-SI logical packages.

**3.2.14**

**Control traffic**

**Control packets**

command, response, and asynchronous event notification packets transmitted between the Management Controller and Network Controllers for the purpose of managing the NC-SI

**3.2.15**

**Command**

control packet sent by the Management Controller to the Network Controller to request the Network Controller to perform an action, and/or return data

**3.2.16**

**Response**

control packet sent by the Network Controller to the Management Controller as a positive acknowledgement of a command received from the Management Controller, and to provide the execution outcome of the command, as well as to return any required data

**3.2.17**

**Asynchronous event notification**

control packet sent by the Network Controller to the Management Controller as an explicit notification of the occurrence of an event of interest to the Management Controller

**3.2.18**

**Pass-through traffic**

**Pass-through packets**

network packets passed between the external network and the Management Controller through the Network Controller

## 3.3 Numbers and Number Bases

Hexadecimal numbers are written with a "`0x`" prefix (for example, `0xFFF` and `0x80`). Binary numbers are written with a lowercase *b* suffix (for example, `1001b` and `10b`). Hexadecimal and binary numbers are formatted in the `Courier New` font.

## 3.4 Reserved Fields

Unless otherwise specified, reserved fields are reserved for future use and should be written as zeros and ignored when read.

# 4 Acronyms and Abbreviations

The following symbols and abbreviations are used in this document.

**4.1**
**AC**
alternating current

**4.2**
**AEN**
Asynchronous Event Notification

**4.3**
**BMC**
Baseboard Management Controller (often used interchangeably with MC)

**4.4**
**CRC**
cyclic redundancy check

**4.5**
**CRS_DV**
a physical NC-SI signal used to indicate Carrier Sense/Received Data Valid

**4.6**
**DC**
direct current

**4.7**
**DHCP**
Dynamic Host Configuration Protocol

**4.8**
**FCS**
Frame Check Sequence

**4.9**
**MC**
Management Controller

**4.10**
**NC**
Network Controller

**4.11**
**NC-SI**
Network Controller Sideband Interface

**4.12**
**NC-SI RX**
the direction of traffic on the NC-SI from the Network Controller to the Management Controller

**4.13**

**NC-SI TX**

the direction of traffic on the NC-SI to the Network Controller from the Management Controller

**4.14**

**RMII**

Reduced Media Independent Interface

**4.15**

**RX**

Receive

**4.16**

**RXD**

physical NC-SI signals used to transmit data from the Network Controller to the Management Controller

**4.17**

**RX_ER**

a physical NC-SI signal used to indicate a Receive Error

**4.18**

**SerDes**

serializer/deserializer; an integrated circuit (IC or chip) transceiver that converts parallel data to serial data and vice-versa. This is used to support interfaces such as 1000Base-X and others.

**4.19**

**TX**

Transmit

**4.20**

**TXD**

physical NC-SI signals used to transmit data from the Management Controller to the Network Controller

**4.21**

**VLAN**

Virtual LAN

# 5 NC-SI Overview

With the increasing emphasis on out-of-band manageability and functionality such as Remote Media (R-Media) and Remote Keyboard-Video-Mouse (R-KVM), the need for defining an industry standard Network Controller Sideband Interface (NC-SI) has become clear. This specification enables a common interface definition between different Management Controller and Network Controller vendors. This specification addresses not only the electrical and protocol specifications, but also the system-level behaviors for the Network Controller and the Management Controller related to the NC-SI.

The NC-SI is defined as the interface between a Management Controller and one or multiple Network Controllers. This interface, depicted in Figure 1, is responsible for providing external network connectivity for the Management Controller. Electrically, it is similar to the Reduced Media Independent Interface™ (RMII) operating in full-duplex, as described in Clause 10.

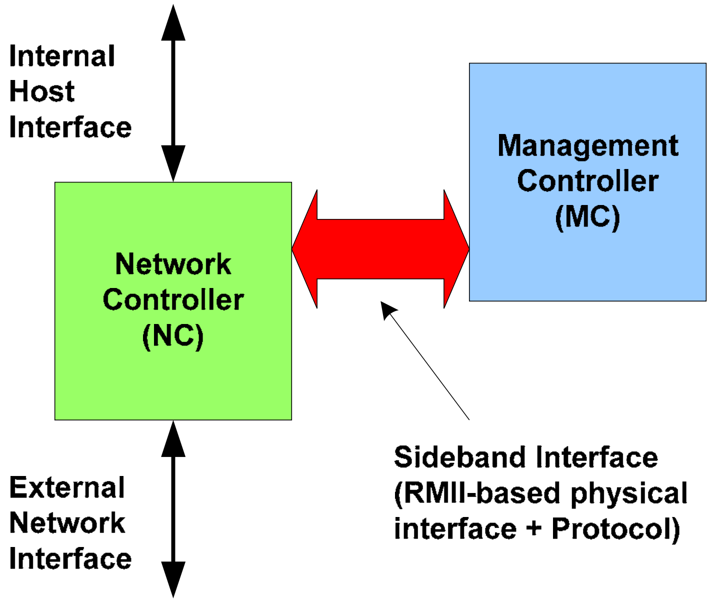



**Figure 1 – NC-SI Functional Block Diagram**

NC-SI traffic flow is illustrated in Figure 2. Two classes of packet data can be delivered over the Sideband Interface:

- "Pass-through" packets that are transferred between the Management Controller and the external network
- "Control" packets that are transferred between the Management Controller and Network Controllers for control or configuration functionality

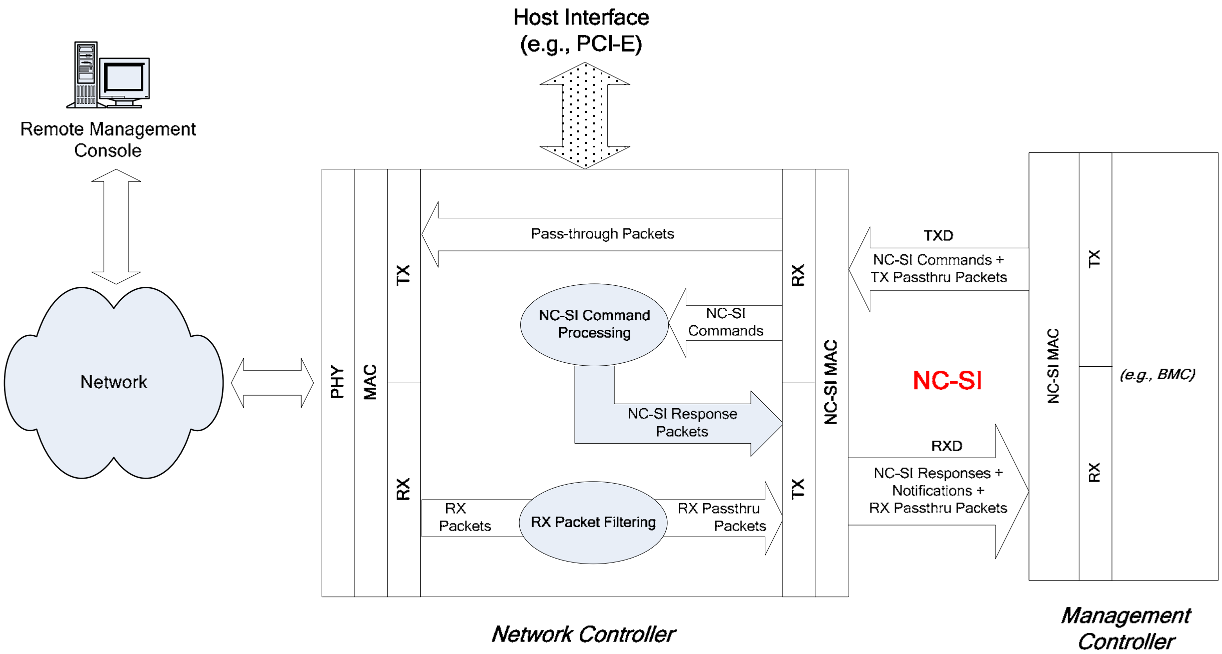



**Figure 2 – NC-SI Traffic Flow Diagram**

The NC-SI is intended to operate independently from the in-band activities of the Network Controller. As such, the Sideband Interface is not specified to be accessible through the host interface of the Network Controller. From the external world, this interface should behave and operate like a standard Ethernet Interface.

## 5.1 Defined Topologies

The topologies supported under this specification apply to the case in which a single Management Controller is actively communicating with one or more Network Controllers on the NC-SI. The electrical specification is targeted to directly support up to four physical Network Controller packages. The protocol specification allows up to eight Network Controller packages, with up to 31 channels per package.

Figure 3 illustrates some examples of Network Controller configurations supported by the NC-SI in the current release:

- Configuration 1 shows a Management Controller connecting to a single Network Controller with a single external network connection.
- Configuration 2 shows a Management Controller connecting to a Network Controller package that supports two NC-SI channels connections.
- Configuration 3 shows a Management Controller connecting to four discrete Network Controllers.

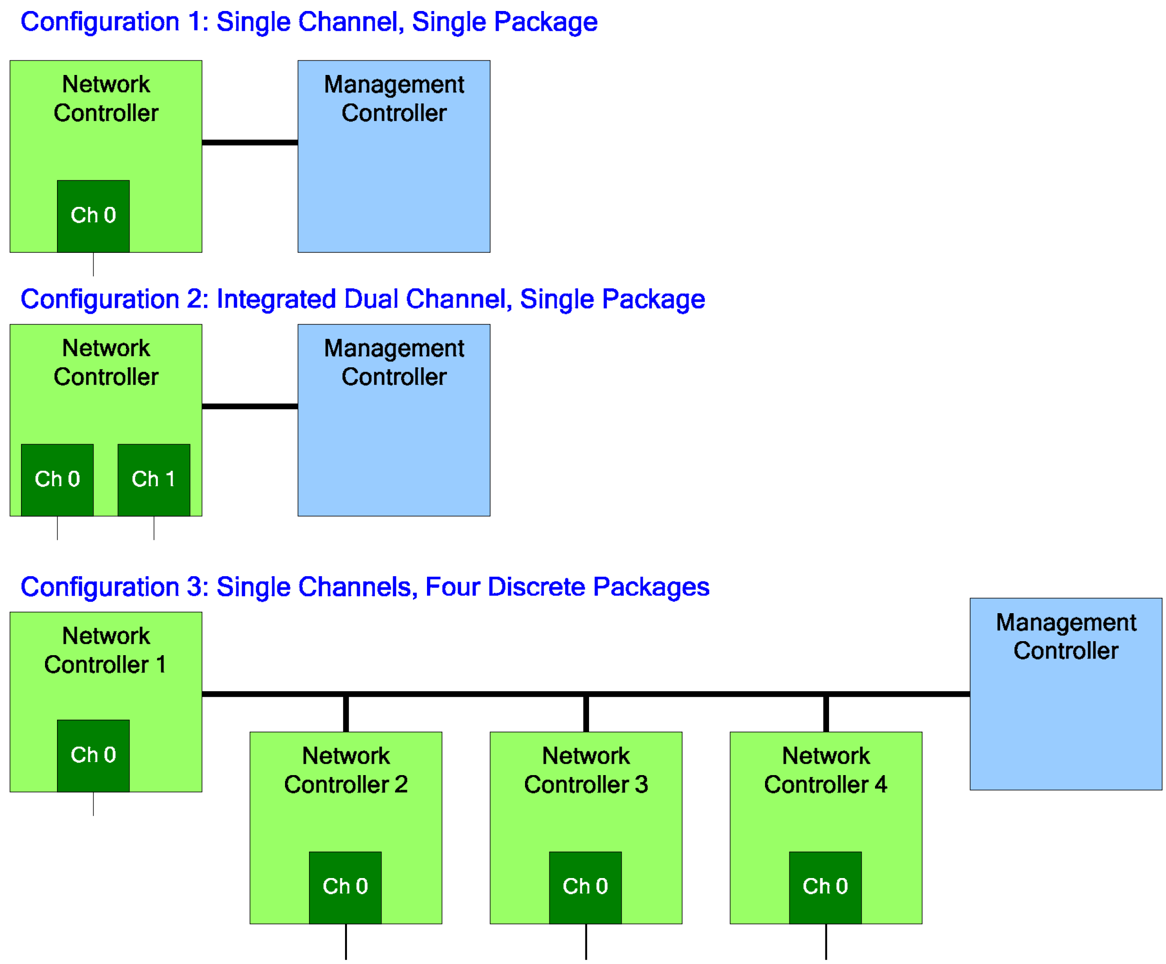



**Figure 3 – Example Topologies Supported by the NC-SI**

## 5.2 Single and Integrated Network Controller Implementations

This clause illustrates the general relationship between channels, packages, receive buffers, and bus buffers for different controller implementations.

An integrated controller is a Network Controller that connects to the NC-SI and provides NC-SI support for two or more network connections. A single controller is a controller that supports only a single NC-SI channel.

For the *NC-SI Specification*, an integrated controller can be logically implemented in one of three basic ways, as illustrated in Figure 4. Although only two channels are shown in the illustration, an integrated controller implementation can provide more than two channels. The example channel and package numbers (for example, channel 0, pkg 0) refer to the Internal Channel and Package ID subfields of the Channel ID. For more information, see 6.2.9.

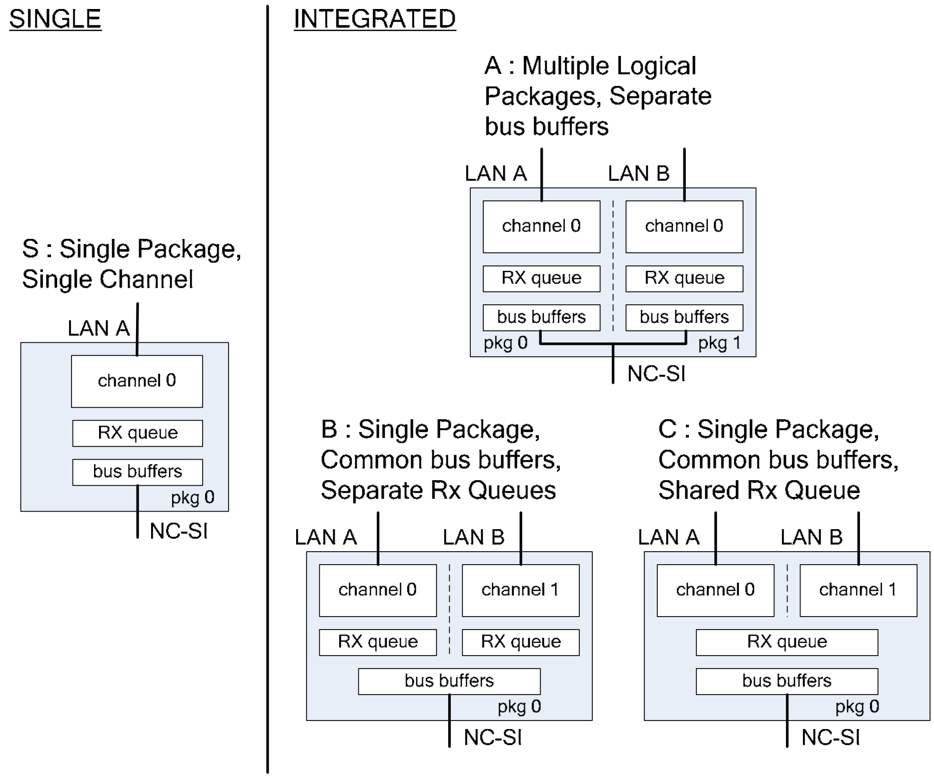



**Figure 4 – Network Controller Integration Options**

Packages that include multiple channels are required to handle internal arbitration between those channels and the NC-SI. The mechanism by which this occurs is vendor specific and not specified in this document. This internal arbitration is always active by default. No NC-SI commands are defined for enabling or disabling internal arbitration between channels.

The following classifications refer to a logical definition. The different implementations are distinguished by their *behavior* with respect to the NC-SI bus and command operation. The actual physical and internal implementation can vary from the simple diagrams. For example, an implementation can act as if it has separate RX queues without having physically separated memory blocks for implementing those queues.

- **S: Single Package, Single Channel**

  This implementation has a single NC-SI interface providing NC-SI support for a single LAN port, all contained within a package or module that has a single connection to the NC-SI physical bus.

- **A: Multiple Logical Packages, Separate Bus Buffers**

  This implementation acts like two physically separate Network Controllers that happen to share a common overall physical container. Electrically, they behave as if they have separate electrical buffers connecting to the NC-SI bus. This behavior may be accomplished by means of a passive internal bus or by separate physical pins coming from the overall package. From the point of view of the Management Controller and the NC-SI command operation, this implementation behaves as if the logical controllers were implemented as physically separate controllers.

This type of implementation may or may not include internal hardware arbitration between the two logical Network Controller packages. If hardware arbitration is provided external to the package, it shall meet the requirements for hardware arbitration described later in this specification. (For more information, see 7.2.)

- **B: Single Package, Common Bus Buffers, Separate RX Queues**

  In this implementation, the two internal NC-SI channels share a common set of electrical bus buffers. A single Deselect Package command will deselect the entire package. The Channel Enable and Channel Disable commands to each channel control whether the channel can transmit Pass-through and AEN packets through the NC-SI interface. The Channel Enable command also determines whether the packets to be transmitted through the NC-SI interface will be queued up in an RX Queue for the channel while the channel is disabled or while the package is deselected. Because each channel has its own RX Queue, this queuing can be configured for each channel independently.

- **C: Single Package, Common Bus Buffers, Shared RX Queue**

  This implementation is the same as described in the preceding implementation, except that the channels share a common RX Queue for holding Pass-through packets to be transmitted through the NC-SI interface. This queue may or may not also queue up AEN or Response packets.

## 5.3 Transport Stack

The overall transport stack of the NC-SI is illustrated in Figure 5. The lowest, physical-level interface is based on RMII, and the media-level interface is based on Ethernet. Above these interfaces are the two data-level protocols that are supported by the *NC-SI Specification*: NC-SI Command Protocol and the Network Data Protocol (for example, ARP, IP, DHCP, and NetBIOS) associated with Pass-through traffic.

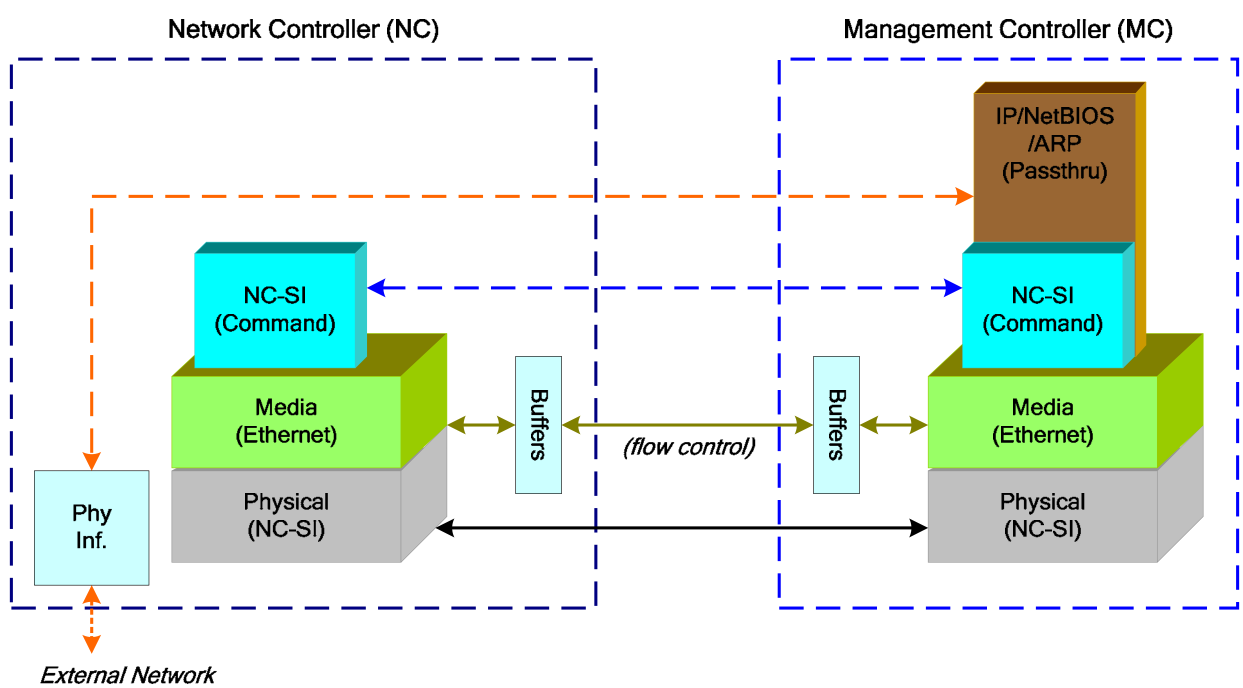



**Figure 5 – NC-SI Transport Stack**

This document defines the necessary NC-SI command set and interface specification that allows the appropriate configuration of the Network Controller parameters and operation to enable network traffic to flow to and from external networks to the Management Controller. As shown in Figure 5, the scope of the NC-SI Command Protocol is limited to the internal interface between the Network Controller and the Management Controller.

## 5.4 Transport Protocol

A simple transport protocol is used to track the reliable reception of command packets. The transport protocol is based upon a command/response paradigm and involves the use of unique Instance IDs (IIDs) in the packet headers to allow responses received to be matched to previously transmitted commands. The Management Controller is the generator of command packets sent to the Sideband Interface of one or more Network Controllers in the system, and it receives response packets from them. A response packet is expected to be received for every command packet successfully sent.

The transport protocol described here shall apply only to command and response packets sent between the Management Controller and the Network Controller.

# 6 Operational Behaviors

This clause describes the NC-SI operating states and typical system-level operation of the NC-SI.

## 6.1 Typical Operational Model

This clause describes the typical system-level operation of the NC-SI components.

The following tasks are associated with Management Controller use of the NC-SI:

- **Initial Configuration**

  When the NC-SI interface is first powered up, the Management Controller needs to discover and configure NC-SI devices in order to enable pass-through operation. This task includes setting parameters such as MAC addresses, configuring Layer 2 filtering, setting Channel enables, and so on.

- **Pass-through**

  The Management Controller handles transmitting and receiving Pass-through packets using the NC-SI. Pass-through packets can be delivered to and received from the network through the NC-SI based on the Network Controller's NC-SI configuration.

- **Asynchronous Event Handling**

  In certain situations, a status change in the Network Controller, such as a Link State change, can generate an asynchronous event on the Sideband Interface. These event notifications are sent to the Management Controller where they are processed as appropriate.

- **Error Handling**

  The Management Controller handles errors that may occur during operation or configuration. For example, a Network Controller may have an internal state change that causes it to enter a state in which it requires a level of reconfiguration (this condition is called the "Initial State," described in more detail in 6.2.4); or a data glitch on the NC-SI could have caused an NC-SI command to be dropped by the Network Controller, requiring the Management Controller to retry the command.

## 6.2 State Definitions

This clause describes NC-SI operating states.

### 6.2.1 General

Table 1 describes states related to whether and when the Network Controller is ready to handle NC-SI command packets, when it is allowed to transmit packets through the NC-SI interface, and when it has entered a state where it is expecting configuration by the Management Controller.

**Table 1 – NC-SI Operating State Descriptions**

| State | Applies to | Description |
|---|---|---|
| Interface Power Down | Package | The NC-SI is in the power down state. |
| Interface Power Up | Package | The NC-SI is in the power up state, as defined in Clause 10. |
| Package Selected (also referred to as the Selected state) | Package | A Selected package is allowed to turn on its electrical buffers and transmit through the NC-SI interface. |
| Package Deselected (also referred to as the Deselected state) | Package | A Deselected package is not allowed to turn on its electrical buffers and transmit through the NC-SI interface. |
| Hardware Arbitration Enabled | Package | When hardware arbitration is enabled, the package is allowed to transmit through the NC-SI interface only when it is Selected and has the TOKEN op-code. |
| Hardware Arbitration Disabled | Package | When hardware arbitration is disabled, the package is allowed to transmit through the NC-SI interface anytime that it is Selected, regardless of whether it has the TOKEN op-code. |
| Package Ready | Package | In the Package Ready state, the package is able to accept and respond to NC-SI commands for the package and be Selected. |
| Package Not Ready | Package | The Package Not Ready state is a transient state in which the package does not accept package-specific commands. |
| Channel Ready | Channel | In the Channel Ready state, a channel within the package is able to accept channel-specific NC-SI commands that are addressed to its Channel ID (Package ID + Internal Channel ID). |
| Channel Not Ready | Channel | The Channel Not Ready state is a transient state in which the channel does not accept channel-specific commands. |
| Initial State | Channel | In the Initial State, the channel is able to accept and respond to NC-SI commands, and one or more configuration settings for the channel need to be set or restored by the Management Controller (that is, the channel has not yet been initialized, or has encountered a condition where one or more settings have been lost and shall be restored). Refer to 6.2.4 for more information. |
| Channel Enabled | Channel | This is a sub-state of the Channel Ready state. When a channel is enabled, the channel is allowed to transmit unrequested packets (that is, packets that are not command responses—for example, AEN and Pass-through packets) through the NC-SI interface whenever the package is Selected. |
| Channel Disabled | Channel | This is a sub-state of the Channel Ready state. When a channel is disabled, the channel is not allowed to transmit unrequested packets (that is, packets that are not command responses—for example, AEN and Pass-through packets) through the NC-SI interface. |

### 6.2.2 NC-SI Power States

Only two power states are defined for the NC-SI:

- **NC-SI Interface Power Down State**

  In this state, the NC-SI Physical interface and the associated receive and transmit buffers in all devices on the NC-SI (that is, the NC-SI interfaces on the Network Controllers and Management Controller) are not powered up.

- **NC-SI Power Up State**

  In this state, the NC-SI Physical interface and the associated receive and transmit buffers in all devices on the NC-SI (that is, the Network Controller and Management Controller) are powered up. The Network Controller is expected to transition to the Initial State within T4 seconds after the Power Up state is entered.

### 6.2.3 Package Ready State

A Network Controller in the Package Ready state shall be able to respond to any NC-SI commands that are directed to the ID for the overall package (versus being directed to a particular channel within the package). Package-specific commands are identified by a particular set of Channel ID values delivered in the command header (see 6.2.9).

### 6.2.4 Initial State

The Initial State for a channel corresponds to a condition in which the NC-SI is powered up and is able to accept NC-SI commands, and the channel has one or more configuration settings that need to be set or restored by the Management Controller. Because this state may be entered at any time, the Initial State shall be acknowledged with a Clear Initial State command in order for the Initial State to be exited. This requirement helps to ensure that the Management Controller does not continue operating the interface unaware that the NC-SI configuration had autonomously changed in the Network Controller.

An NC-SI channel in the Initial State shall:

- be able to respond to NC-SI commands that are directed to the Channel ID for the particular channel (see 6.2.9)
- respond to all non-OEM command packets that are directed to the channel with a Response Packet that contains a Response Code of “Command Failed” and a Reason Code of “Initialization Required”

  NOTE: This requirement does not apply to commands that are directed to the overall package, such as the Select Package and Deselect Package commands.
- place the channel into the Disabled state

  NOTE: It shall not transmit AENs or Pass-through packets through the NC-SI interface.
- set hardware arbitration (if supported) to “enabled” on Interface Power Up only; otherwise, the setting that was in effect before entry into the Initial State shall be preserved (that is, the hardware arbitration enable/disable configuration is preserved across entries into the Initial State)
- set the enabled/disabled settings for the individual MAC and VLAN filters (typically set using the Set MAC Address and Set VLAN Filter commands) to “disabled”
- reset the counters defined in the Get NC-SI Statistics command and the Get NC-SI Pass-Through Statistics command to `0x0`

- disable transmission of Pass-through packets onto the network

  NOTE: Upon entry into the Initial State, the Channel Network TX setting is also set to “disabled”.

- reset the counters defined in the Get NC-SI Statistics command and the Get NC-SI Pass-Through Statistics command to `0x0` when entering into the Initial State
- clear any record of prior command instances received upon entry into the Initial State (that is, assume that the first command received after entering the Initial State is a new command and not a retried command, regardless of any Instance ID that it may have received before entering the Initial State)

Otherwise, there is no requirement that other NC-SI configuration settings be set, retained, or restored to particular values in the Initial State.

### 6.2.5 NC-SI Initial State Recovery

As described in 6.2.4, a channel in the Initial State shall receive the Clear Initial State command before other commands can be executed. This requirement ensures that if the Initial State is entered asynchronously, the Management Controller is made aware that one or more NC-SI settings may have changed without its involvement, and blocks the Management Controller from issuing additional commands under that condition. Until the channel receives the Clear Initial State command, the Management Controller shall respond to any other received command (except the Select Package and Deselect Package commands) with a Command Failed response code and Interface Initialization Required reason code to indicate that the Clear Initial State command shall be sent. See response and reason code definitions in 8.2.5.

NOTE: Package commands (for example, Select Package and Deselect Package) are always accepted and responded to normally regardless of whether the Channel is in the Initial State.

If the Management Controller, at any time, receives the response indicating that the Clear Initial State command is expected, it may interpret this response to mean that default settings have been restored for the channel (per the Initial State specification), and that one or more channel settings may need to be restored by the Management Controller.

### 6.2.6 State Transition Diagram

Figure 6 illustrates the general relationship between the package- and channel-related states described in Table 1 and the actions that cause transitions between the states. Each bubble in Figure 6 represents a particular combination of states as defined in Table 1.

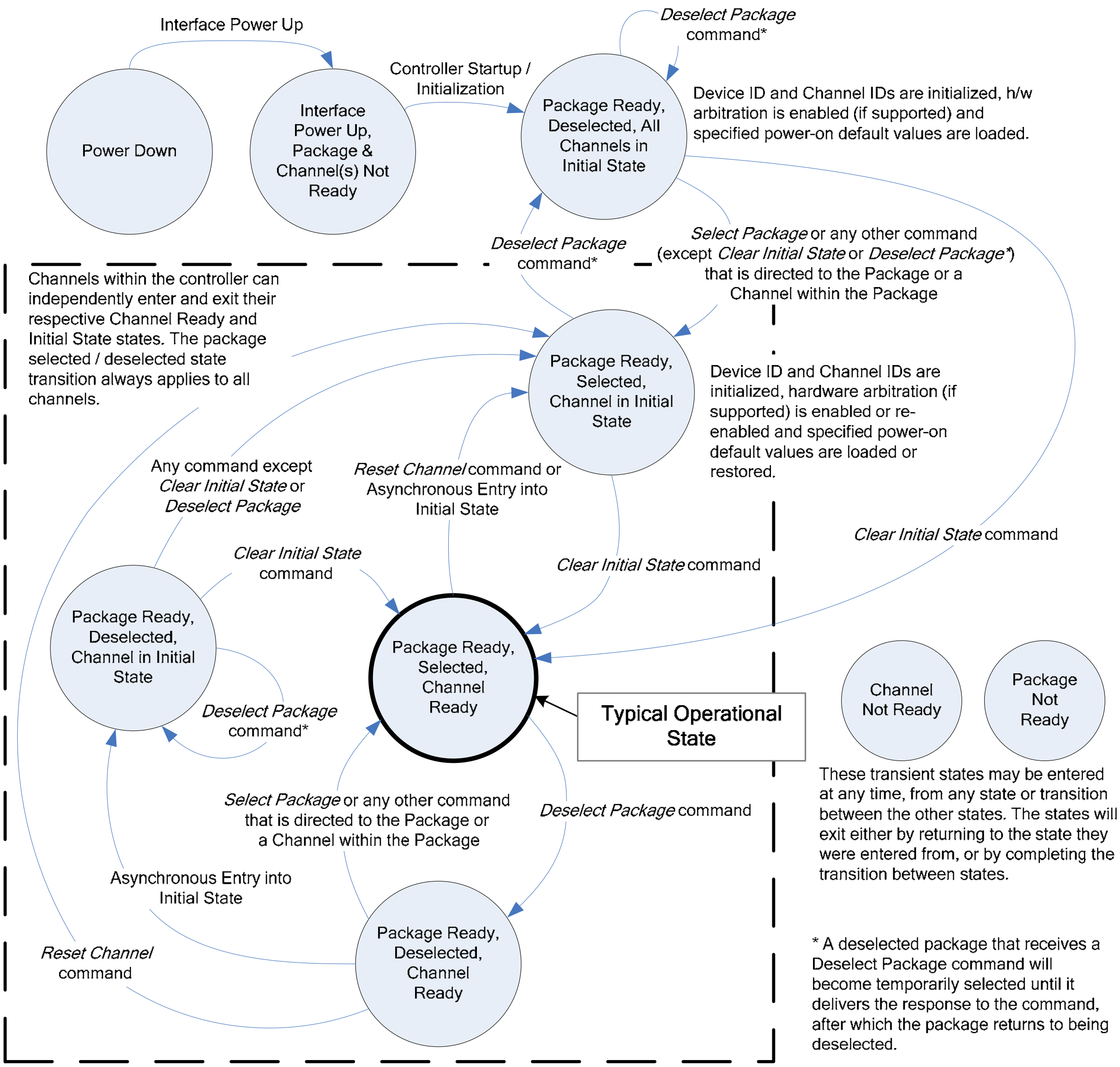



**Figure 6 – NC-SI Operational State Diagram**

### 6.2.7 State Diagram for NC-SI Operation with Hardware Arbitration

Figure 7 shows NC-SI operation in the hardware arbitration mode of operation. This is a sub-set of the general NC-SI Operational State Diagram (Figure 6) and has been included to illustrate the simplified sequence of package selection when this optional capability is used.

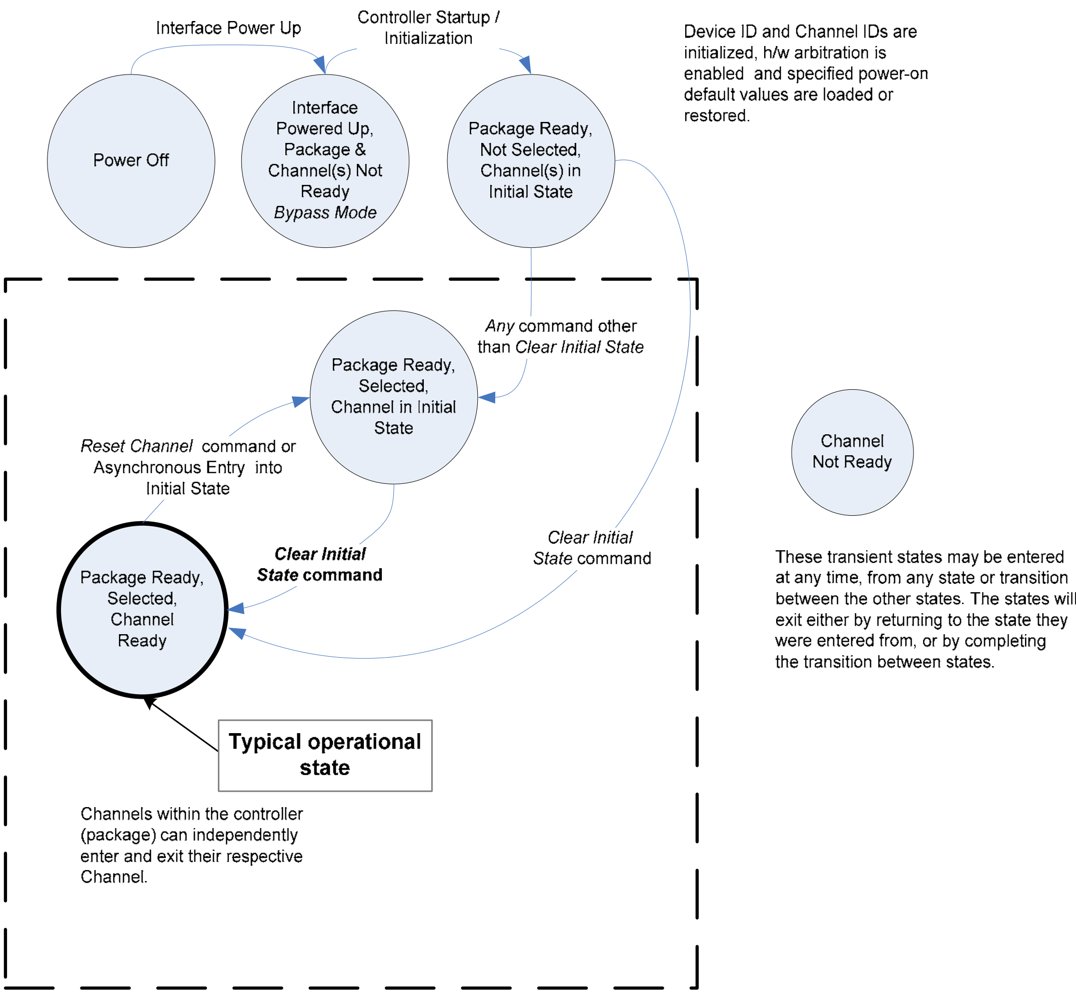



**Figure 7 – NC-SI Operational State Diagram for Hardware Arbitration Operation**

### 6.2.8 Resets

Two types of Reset events are defined for the NC-SI Channels:

- Asynchronous Entry into Initial State
- Synchronous Reset

NOTE: Resets that do not affect NC-SI operation are outside the scope of this specification.

#### 6.2.8.1 Asynchronous Entry into Initial State

An Asynchronous Reset event is defined as an event that results in a Channel asynchronously entering the Initial State. This event could occur as a consequence of powering up, a System Reset, a Driver Reset, an Internal Firmware error, loss of Configuration errors, Internal hardware errors, and so on.

Unless otherwise specified, NC-SI configuration settings beyond those required by the Initial State may or may not be preserved following asynchronous entry into the Initial State, depending on the Network Controller implementation.

There is no explicit definition of a Reset for an entire package. However, it is possible that an Asynchronous Reset condition may cause an Asynchronous Entry into the Initial State for all Channels in a package simultaneously.

#### 6.2.8.2 Synchronous Reset

A Synchronous Reset event on the NC-SI is defined as a Reset Channel command issued by a Management Controller to a Channel. Upon the receipt of this command, the Network Controller places the Channel into the Initial State.

Unless otherwise specified, NC-SI configuration settings beyond those required by the Initial State may or may not be preserved following a Synchronous Reset, depending on the Network Controller implementation.

### 6.2.9 Network Controller Channel ID

Each channel in the Network Controller shall be physically assigned a Network Controller Channel ID that will be used by the Management Controller to specify with which Network Controller channel, of possibly many, it is trying to communicate. The Network Controller Channel ID shall be physically assignable (configured) at system-integration time based on the following specification.

It is the system integrator’s or system designer’s responsibility to correctly assign and provide these identifier values in single- and multi-port Network Controller configurations, and to ensure that Channel IDs do not conflict between devices sharing a common NC-SI interconnect.

The Channel ID field comprises two subfields, Package ID and Internal Channel ID, as described in Table 2.

**Table 2 – Channel ID Format**

| Bits | Field Name | Description |
|---|---|---|
| [7..5] | Package ID | The Package ID is required to be common across all channels within a single Network Controller that share a common NC-SI physical interconnect.<br><br>The system integrator will typically configure the Package IDs starting from 0 and increasing sequentially for each physical Network Controller.<br><br>The Network Controller shall allow the least significant two bits of this field to be configurable by the system integrator, with the most significant bit of this field = `0b`. An implementation is allowed to have all 3 bits configurable. |
| [4..0] | Internal Channel ID | The Network Controller shall support Internal Channel IDs that are numbered starting from 0 and increasing sequentially for each Pass-through channel supported by the Network Controller that is accessible by the Management Controller through the NC-SI using NC-SI commands.<br><br>An implementation is allowed to support additional configuration options for the Internal Channel ID as long as the required numbering can be configured.<br><br>An Internal Channel ID value of `0x1F` applies to the entire Package. |

Channel IDs shall be completely decoded. Aliasing between values is not allowed (that is, the Network Controller is not allowed to have multiple IDs select the same channel on a given NC-SI).

Once configured, the settings of the Package ID and Internal Channel ID values shall be retained in a non-volatile manner. That is, they shall be retained across power-downs of the NC-SI and shall not be required to be restored by the Management Controller for NC-SI operation. This specification does not define the mechanism for configuring or retaining the Package ID or the Internal Channel ID (if configurable). Some implementations may use pins on the Network Controller for configuring the IDs, other implementations may use non-volatile storage logic such as electrically-erasable memory or FLASH, while others may use a combination of pins and non-volatile storage logic.

### 6.2.10 Configuration-Related Settings

This clause presents an overview of the different settings that the Management Controller may need to configure for NC-SI operation.

#### 6.2.10.1 Package-Specific Operation

Only two configuration settings are package-specific:

- the enable/disable settings for hardware arbitration
- NC-SI flow control

Hardware arbitration is enabled or disabled through a parameter that is delivered using the Select Package command. If hardware arbitration is enabled on all Network Controller packages on the NC-SI, more than one package can be in the Selected state simultaneously. Otherwise, only one package is allowed to be in the Selected state at a time in order to prevent electrical buffer conflicts (buffer fights) that can occur from more than one package being allowed to drive the bus.

NC-SI flow control is enabled or disabled using the Set NC-SI Flow Control command. The flow control setting applies to all channels in the package.

#### 6.2.10.2 Channel-Specific Operation

Table 3 shows the major categories of configuration settings that control channel operation when a channel is in the Channel Ready state.

**Table 3 – Channel Ready State Configuration Settings**

| Setting/Configuration Category | Description |
|---|---|
| “Channel Enable” settings | The Enable Channel and Disable Channel commands are used to control whether the channel is allowed to asynchronously transmit unrequested packets (AEN and Pass-through packets) through the NC-SI interface whenever the package is Selected. Note that channels are always allowed to transmit responses to commands sent to the channel. |
| Pass-through Transmit Enable settings | The Enable Channel Network TX command is used to enable the channel to transmit any Pass-through packets that it receives through the NC-SI onto the network, provided that the source MAC address in those packets matches the Network Controller settings. Correspondingly, the Disable Channel Network TX command is used to direct the controller not to transmit Pass-through packets that it receives onto the network. |
| AEN Enable settings | The AEN Enable command is used to enable and disable the generation of the different AENs supported by the Network Controller. |

| Setting/Configuration Category | Description |
|---|---|
| MAC Address Filter settings and control | The Set MAC Address command and Enable Global Multicast commands are used to configure the MAC Address Filter for unicast and multicast addresses that the controller uses in conjunction with the VLAN Filter settings for filtering incoming Pass-through packets. |
| VLAN Filter settings and control | The Set VLAN Filter command is used to configure VLAN Filters that the controller uses in conjunction with the MAC Address Filters for filtering incoming Pass-through packets. The Enable VLAN and Disable VLAN commands are used to configure VLAN filtering modes and enable or disable whether VLAN filtering is used. |

### 6.2.11 Transmitting Pass-through Packets from the Management Controller

Packets not recognized as command packets (that is, packets without the NC-SI Ethertype) that are received on the Network Controller's NC-SI interface shall be assumed to be Pass-through packets provided that they meet the source MAC Address setting for the channel in the Network Controller, and will be forwarded for transmission to the corresponding external network interface if Channel Network TX is enabled.

### 6.2.12 Receiving Pass-through Packets for the Management Controller

The Management Controller has control over and responsibility for configuring packet-filtering options, such as whether broadcast, multicast, or VLAN packets are accepted. Depending on the filter configurations, after the channel has been enabled, any packet that the Network Controller receives for the Management Controller shall be forwarded to the Management Controller through the NC-SI interface.

### 6.2.13 Startup Sequence Examples

The following sections show possible startup sequences that may be used by the Management Controller to start NC-SI operation. Depending upon the specific configuration of each system, there are many possible variations of startup sequences that may be used, and these examples are intended for reference only.

#### 6.2.13.1 Typical Non Hardware Arbitration Specific Startup Sequence

The following sequence is provided as an example of one way a Management Controller can start up NC-SI operation. This sequence assumes that the Management Controller has no prior knowledge of how many Network Controllers are hooked to its NC-SI, or what capabilities those controllers support. Note that this is not the only possible sequence. Alternative sequences can also be used to start up NC-SI operation. Some steps may be skipped if the Management Controller has prior knowledge of the Network Controller capabilities, such as whether Network Controllers are already connected and enabled for hardware arbitration.

1) **Power Up**

   The NC-SI is powered up (refer to 10.2.7 for the specification of this condition). The Network Controller packages are provided a Device Ready Interval during which they can perform internal firmware startup and initialization to prepare their NC-SI to accept commands. The Management Controller first waits for the maximum Device Ready Interval to expire (refer to Table 110). At this point, all the Network Controller packages and channels should be ready to accept commands through the NC-SI. (The Management Controller may also start sending commands before the Device Ready Interval expires, but will have to handle the case that Network Controller devices may be in a state in which they are unable to accept or respond to commands.)

2) **Discover Package**

The Management Controller issues a Select Package command starting with the lowest Package ID (see 8.4.5 for more information). Because the Management Controller is assumed to have no prior knowledge of whether the Network Controller is enabled for hardware arbitration, the Select Package command is issued with the Hardware Arbitration parameter set to ‘disable’.

If the Management Controller receives a response within the specified response time, it can record that it detected a package at that ID. If the Management Controller does not receive a response, it is recommended that the Management Controller retry sending the command. Three total tries is typical. (This same retry process should be used when sending all commands to the Network Controller and will be left out of the descriptions in the following steps.) If the retries fail, the Management Controller can assume that no Network Controller is at that Package ID and can immediately repeat this step 2) for the next Package ID in the sequence.

3) **Discover and Get Capabilities for Each Channel in the Package**

The Management Controller can now discover how many channels are supported in the Network Controller package and their capabilities. To do this, the Management Controller issues the Clear Initial State command starting from the lowest Internal Channel ID (which selects a given channel within a package). If it receives a response, the Management Controller can then use the Get Version ID command to determine NC-SI specification compatibility, and the Get Capabilities command to collect information about the capabilities of the channel. The Management Controller can then repeat this step until the full number of internal channels has been discovered. (The Get Capabilities command includes a value that indicates the number of channels supported within the given package.)

NOTE: The *NC-SI Specification* requires Network Controllers to be configurable to have their Internal Channel IDs be sequential starting from 0. If it is known that the Network Controller is configured this way, the Management Controller needs only to iterate sequentially starting from Internal Channel ID = 0 up to the number of channels reported in the first Get Capabilities response.

The Management Controller should temporarily retain the information from the Get Capabilities command, including the information that reports whether the overall package supports hardware arbitration. This information is used in later steps.

4) **Repeat Steps 2 and 3 for Remaining Packages**

The Management Controller repeats steps 2) and 3) until it has gone through all the Package IDs.

IMPORTANT: Because hardware arbitration has not been enabled yet, the Management Controller shall issue a Deselect Package command to the present Package ID before issuing the Select Package command to the next Package ID. If hardware arbitration is not being used, only one package can be in the Selected state at a time. Otherwise, hardware electrical buffer conflicts (buffer fights) will occur between packages.

5) **Initialize Each Channel in the Package**

Based on the number of packages and channels that were discovered, their capabilities, and the desired use of Pass-through communication, the Management Controller can initialize the settings for each channel. This process includes the following general steps for each package:

a) Issue the Select Package command.

b) For each channel in the package, depending on controller capabilities, perform the following actions. Refer to individual command descriptions for more information.

- Use the Set MAC Address command to configure which unicast and multicast addresses are used for routing Pass-through packets to and from the Management Controller.
- Use the Enable Broadcast Filter command to configure whether incoming broadcast Pass-through packets are accepted or rejected.
- Use the Enable Global Multicast Filter command to configure how incoming multicast Pass-through packets are handled based on settings from the Set MAC Address command.
- Use the Set VLAN Filter and Enable VLAN Filters commands to configure how incoming Pass-through packets with VLAN Tags are handled.
- Use the Set NC-SI Flow Control command to configure how Ethernet Pause Frames are used for flow control on the NC-SI.
- Use the AEN Enable command to configure what types of AEN packets the channel should send out on the NC-SI.
- Use the Enable Channel Network TX command to configure whether the channel is enabled to deliver Pass-through packets from the NC-SI to the network (based on the MAC address settings) or is disabled from delivering any Pass-through packets to the network.

c) Issue the Deselect Package command.

6) **Enable Hardware Arbitration for the Packages**

If only a single Network Controller package is discovered, the Management Controller does not need to enable hardware arbitration if the controller hardware supports it. In fact, the Management Controller may always elect to disable hardware arbitration, because then it does not need to be concerned with whether the implementation provided a 'loop back' of the hardware arbitration 'ARB_OUT' signal to the controller to the 'ARB_IN' signal.

If multiple packages are detected, and each package has reported that it supports hardware arbitration, then the hardware arbitration operation can be enabled by issuing a Select Package command, with the Hardware Arbitration parameter for the command set to 'enabled', to each package. Because hardware arbitration enables multiple packages to be selected simultaneously, sending Deselect Package commands is not necessary when hardware arbitration is being used.

NOTE: There is no status to indicate whether hardware arbitration is hooked up and operating correctly. The Management Controller shall have prior knowledge that the implementation routes the hardware arbitration signals between the packages.

7) **Start Pass-through Packet and AEN Operation on the Channels**

The channels should now have been initialized with the appropriate parameters for Pass-through packet reception and AEN operation. Pass-through operation can be started by issuing the Enable Channel command to each channel that is to be enabled for delivering Pass-through packets or generating AENs through the NC-SI interface.

NOTE: If hardware arbitration is not operational and it is necessary to switch operation over to another package, a Deselect Package command shall be issued to the presently selected package before a different package can be selected. Deselecting a package blocks all output from the package. Therefore, it is not necessary to issue Disable Channel commands before selecting another package. There is no restriction on enabling multiple channels *within* a package.

#### 6.2.13.2 Hardware Arbitration Specific Startup Sequence

The following is an example of the steps that a Management Controller may perform to start up NC-SI operation when Hardware Arbitration is specifically known to be used, present, and enabled on all Network Controllers. This example startup sequence assumes a high level of integration where the Management Controller knows the Network Controllers support and default to the use of Hardware Arbitration on startup, but does not have prior knowledge of how many Network Controllers are interfaced to the NC-SI, or the full set of capabilities those controllers support, so discovery is still required.

Although other startup examples may show a specific ordering of steps for the process of discovering, configuring and enabling channels, the Management Controller actually has almost total flexibility in choosing how these steps are performed once a channel in a package is discovered. In the end, it would be just as valid for a Management Controller to follow a breadth-first approach to discovery steps as it would be to follow a depth-first approach where each channel that is discovered is fully initialized and enabled before moving to the next.

1) **Power Up**

   No change from other startup scenarios.

2) **Discovery**

   The process of discovery consists of identifying the number of packages that are available, the number of channels that are available in each package, and for each channel, the capabilities that are provided for Management Controller use. Because, in this startup scenario, the Management Controller knows Hardware Arbitration is used, it is not required to use the ***Select Package*** and ***Deselect Package*** commands for discovery, but may elect to just use the ***Clear Initial State*** command for this purpose instead.

   In this startup scenario, Packages and Channels are discovered by sending the ***Clear Initial State*** command starting with the lowest Package ID and Channel ID, then waiting for, and recording, the response event as previously described. Internal channel IDs are required to be numbered sequentially starting with 0, so when the Management Controller does not receive a response to repeated attempts at discovery, it knows this means no additional channels exist in the current package. If this happens when the internal channel ID is 0, the Management Controller knows a package is not available at the current package ID, and it continues with the next package ID in sequence. If the Management Controller receives a response to the ***Clear Initial State*** command, it records that the channel and package are available, and continues discovery.

   During discovery, the Management Controller should interrogate the capabilities of each channel found to be available in each package by sending the ***Get Capabilities*** command appropriate package and channel ID values. However, it does not matter whether this is done as the very next step in the discovery process, or performed for each channel after all packages and channels have been discovered, just as long as the Management Controller does interrogate each channel.

3) **Configure each channel and enable pass-through**

   Once the existence of all packages and channels, and the capabilities of each channel, have been discovered and recorded, the Management Controller shall initialize and enable each channel as needed for use. The details of these steps remain essentially the same as have been previously stated, except to note that there are no restrictions on how they are performed. What this means is that the MC may perform these steps in any order across the channels in each package as it sees fit. The MC may fully initialize and enable each channel in each package one at a time, or perform the same step on each channel in sequence before moving on to the next, or in a different order. The specific order of steps is not dictated by this specification.

## 6.3 NC-SI Traffic Types

Two types of traffic are carried on the NC-SI: Pass-through traffic and Control traffic.

- Pass-through traffic consists of packets that are transferred between the external network interface and the Management Controller using the NC-SI.
- Control traffic consists of commands (requests) and responses that support the configuration and control of the NC-SI and Pass-through operation of the Network Controller, and AENs that support reporting various events to the Management Controller..

### 6.3.1 Command Protocol

Commands are provided to allow a Management Controller to initialize, control, and regulate Management Controller packet flow across the NC-SI, configure channel filtering, and to interrogate the operational status of the Network Controller. As interface master, the Management Controller is the initiator of all commands, and the Network Controller responds to commands.

#### 6.3.1.1 Instance IDs

The command protocol uses a packet field called the Instance ID (IID). IID numbers are 8-bit values that shall range from `0x01` to `0xFF`. IIDs are used to identify new instances of a command, to improve the robustness of matching responses to commands, and to differentiate between new and retried commands.

The Network Controller that receives a command handles the IID in the following ways:

- It returns the IID value from the command in the corresponding response.
- If the IID is the same as the IID for the previous command, it recognizes the command as a ‘retried’ command rather than as a new instance of the command.
- If a retried command is received, the Network Controller shall return the previous response. Depending on the command, the Network Controller can accomplish this either by holding the previous response data so that it can be returned, or, if re-executing the command has no side effects (that is, the command is idempotent), by re-executing the command operation and returning that response.
- When an IID value is received that is different from the one for the previous command, the Network Controller executes the command as a new command.
- When the Network Controller first enters the Initial State, it clears any record of any prior requests. That is, it assumes that the first command after entering the Initial State is a new command and not a retried command, regardless of any IID that it may have received before entering the Initial State.

Thus, for single-threaded operation with idempotent commands, a responding Network Controller can simply execute the command and return the IID in the response that it received in the command. If it is necessary to not execute a retried command, the responding controller can use the IID to identify the retried command and return the response that was delivered for the original command.

The Management Controller that generates a command handles the IID in the following ways:

- The IID changes for each new instance of a command.
- If a command needs to be retried, the Management Controller uses the same value for the IID that it used for the initial command.
- The Management Controller can optionally elect to use the IID as a way to provide additional confirmation that the response is being returned for a particular command.

Because an AEN is not a response, an AEN always uses a value of `0x00` for its IID.

NOTE: The Instance ID mechanism can be readily extended in the future to support multiple controllers and multiple outstanding commands. This extension would require having the responder track the IID on a per command and per requesting controller basis. For example, a retried command would be identified if the IID and command matched the IID and command for a prior command for the given originating controller's ID. That is, a match is made with the command, originating controller, and IID fields rather than on the IID field alone. A requester that generates multiple outstanding commands would correspondingly need to track responses based on both command and IID in order to match a given response with a given command. IIDs need to be unique for the number of different commands that can be concurrently outstanding.

#### 6.3.1.2 Single-Threaded Operation

The Network Controller is required to support NC-SI commands only in a single-threaded manner. That is, the Network Controller is required to support processing only one command at a time, and is not required to accept additional commands until after it has sent the response to the previous one.

Therefore, the Management Controller should issue NC-SI commands in a single-threaded manner. That is, the Management Controller should have only one command outstanding to a given Network Controller package at a time. Upon sending an NC-SI command packet, and before sending a subsequent command, the Management Controller should wait for the corresponding response packet to be received or a command timeout event to occur before attempting to send another command. For the full descriptions of command timeout, see 6.8.2.1.

#### 6.3.1.3 Responses

The Network Controller shall process and acknowledge each validly formatted command received at the NC-SI interface by formatting and sending a valid response packet to the Management Controller through the NC-SI interface.

To allow the Management Controller to match responses to commands, the Network Controller shall copy the IID number of the Command into the Instance ID field of the corresponding response packet.

To allow for retransmission and error recovery, the Network Controller may re-execute the last command or maintain a copy of the response packet most recently transmitted to the Management Controller through its NC-SI interface. This "previous" response packet shall be updated every time a new response packet is transmitted to the Management Controller by replacing it with the one just sent.

The Network Controller response shall return a "Command Unsupported" response code with an "Unknown Command Type" reason code for any command (standard or OEM) that the Network Controller does not support or recognize.

#### 6.3.1.4 Response and Post-Response Processing

Typically, a Network Controller completes a requested operation before sending the response. In some situations, however, it may be useful for the controller to be allowed to queue up the requested operation and send the response assuming that the operation will complete correctly (for example, when the controller is requested to change link configuration). The following provisions support this process:

- A Network Controller is allowed to send a response before performing the requested action if the command is expected to complete normally and all parameters that are required to be returned with the response are provided.
- Temporal ordering of requested operations shall be preserved. For example, if one command updates a configuration parameter value and a following command reads back that parameter, the operation requested first shall complete so that the following operation returns the updated parameter.

- Under typical operation of the Network Controller, responses should be delivered within the Normal Execution Interval (T5) (see Table 110).
- Unless otherwise specified, all requested operations shall complete within the Asynchronous Reset/Asynchronous Not Ready interval (T6) following the response.
- If the Network Controller channel determines that the requested operation or configuration change has not been completed correctly after sending the response, the channel shall enter the Initial State.

## 6.4 Link Configuration and Control

The Network Controller provides commands to allow the Management Controller to specify the auto-negotiation, link speed, duplex settings, and so on to be used on the network interface. For more information, see 8.4.21.

NOTE: The Management Controller should make link configuration changes only when the operating system (OS) is absent.

### 6.4.1 Link Status

The Network Controller provides a Get Link Status command to allow the Management Controller to interrogate the configuration and operational status of the primary Ethernet links. The Management Controller may issue the Get Link Status command regardless of OS operational status.

## 6.5 Frame Filtering for Pass-through Mode

The Network Controller provides the option of configuring various types of filtering mechanisms for the purpose of controlling the delivery of received Ethernet frames to the Management Controller. These options include VLAN Tag filter, L2 address filters, MAC address support, and limited frame filtering using L3, L4 protocol header fields. All frames that pass frame filtering are forwarded to the Management Controller over the NC-SI.

### 6.5.1 Multicast Filtering

The Network Controller may provide commands to allow the Management Controller to enable and disable global filtering of all multicast packets. The Network Controller may optionally provide one or more individual multicast filters, as well as DHCP v6, IPv6 Neighbor Advertisement, and IPv6 Router Advertisement filters.

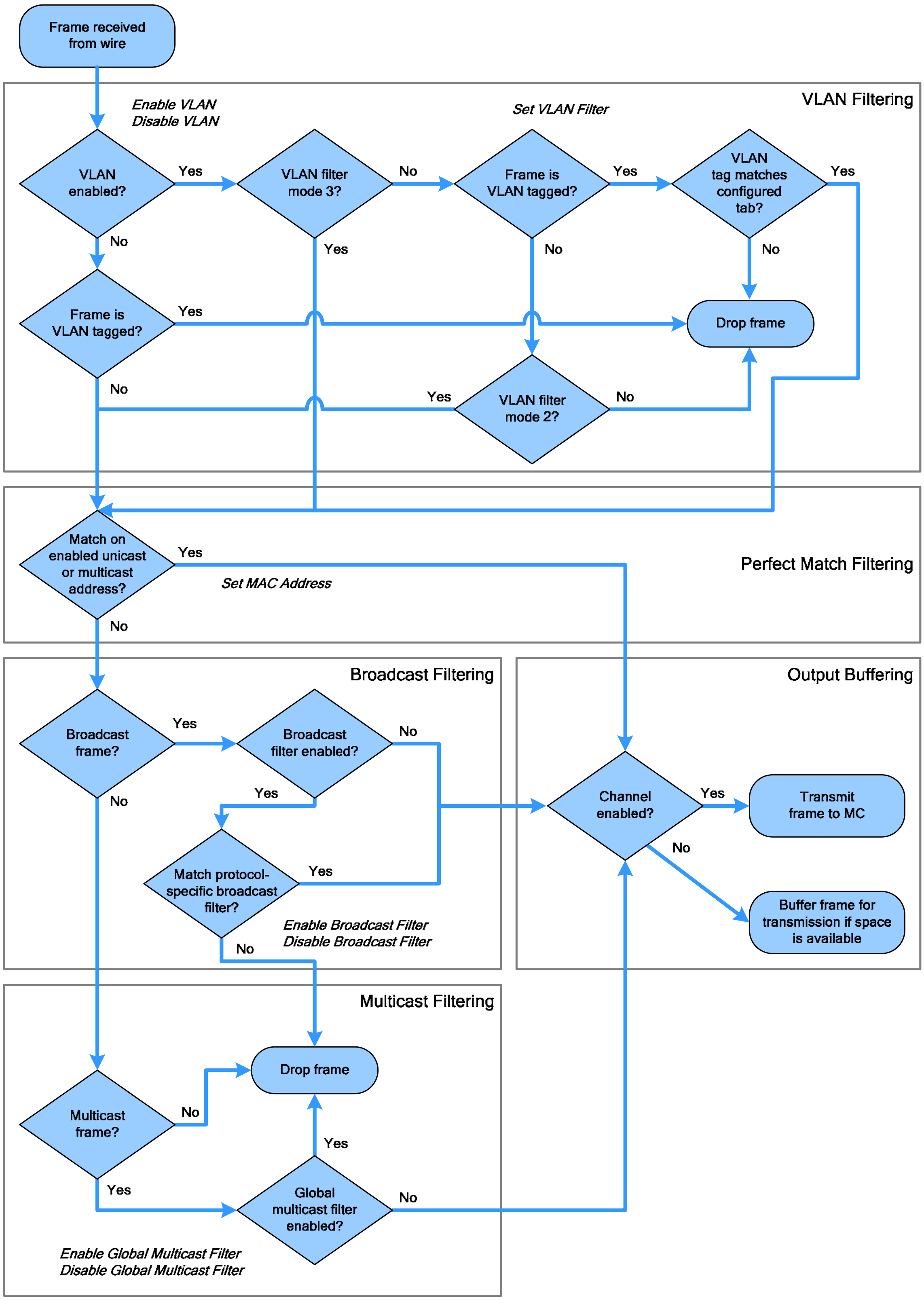
### 6.5.2 Broadcast Filtering

The Network Controller provides commands to allow the Management Controller to enable and disable forwarding of Broadcast and ARP packets. The Network Controller may optionally support selective forwarding of broadcast packets for specific protocols, such as DHCP and NetBIOS.

### 6.5.3 VLAN Filtering

The Network Controller provides commands to allow the Management Controller to enable and disable VLAN filtering, configure one or more VLAN Filters, and to configure VLAN filtering modes.

Figure 8 illustrates the flow of frame filtering. Italicized text in the figure is used to identify NC-SI command names.

1071 


1072 **Figure 8 – NC-SI Packet Filtering Flowchart**

## 6.6 NC-SI Flow Control

The Network Controller may provide commands to enable flow control on the NC-SI between the Network Controller and the Management Controller. The NC-SI flow control behavior follows the PAUSE frame behavior as defined in the IEEE 802.3 specification. Flow control is configured using the Set NC-SI Flow Command (see 8.4.41).

## 6.7 Asynchronous Event Notification

Asynchronous Event Notification (AEN) packets enable the Network Controller to deliver unsolicited notifications to the Management Controller when certain status changes that could impact interface operation occur in the Network Controller. Because the NC-SI is a small part of the larger Network Controller, its operation can be affected by a variety of events that occur in the Network Controller. These events include link status changes, OS driver loads and unloads, and chip resets. This feature defines a set of notification packets that operate outside of the established command-response mechanism.

Control over the generation of the AEN packets is achieved by control bits in the AEN Enable command. Each type of notification is optional and can be independently enabled by the Management Controller.

AENs are not acknowledged, and there is no protection against the possible loss of an AEN packet.

Each defined event has its own AEN packet. Because the AEN packets are generated asynchronously by the Network Controller, they cannot implement some of the features of the other Control packets. AEN packets leverage the general packet format of Control packets.

- The originating Network Controller channel shall fill in its Channel ID (Ch. ID) field in the command header to identify the source of notification.
- The IID field in an AEN shall be set to `0x00` to differentiate it from a response or command packet.
- The Network Controller shall copy the AEN MC ID field from the AEN Enable command into the MC ID field in every AEN sent to the Management Controller.

## 6.8 Error Handling

This clause describes the error-handling methods that are supported over the NC-SI. Two types of error-handling methods are defined:

- Synchronous Error Handling
- Errors that trigger Asynchronous Entry into the Initial State

Synchronous Error Handling occurs when an Error (non-zero) Response/Reason Code is received in response to a command issued by the Management Controller. For information about response and reason codes, see 8.2.5.

Asynchronous Entry into the Initial State Error Handling occurs when the Network Controller asynchronously enters the Initial State because of an error condition that affects NC-SI configuration or a failure of a command that was already responded to. For more information, see 6.2.8.1.

### 6.8.1 Transport Errors

Transport error handling includes the dropping of command packets. Data packet errors are out of the scope of this specification.

#### 6.8.1.1 Dropped Control Packets

The Network Controller shall drop command packets received on the NC-SI interface only under the following conditions:

- The packet has an invalid Frame Check Sequence (FCS) value.
- Frame length does not meet IEEE 802.3 requirements (except for OEM commands, where accepting larger packets may be allowed as a vendor-specific option).
- The packet checksum (if provided) is invalid.
- The NC-SI Channel ID value in the packet does not match the expected value.
- The Network Controller is congested and cannot accept the packet.
- The Network Controller receives a command packet with an incorrect header revision.

The Network Controller may also drop command packets if an event that triggers Asynchronous Entry into the Initial State causes packets to be dropped during the transition.

### 6.8.2 Missing Responses

There are two typical scenarios in which the Management Controller may not receive the response to a command:

- The Network Controller dropped the command and thus never sent the response.
- The response was dropped by the Management Controller (for example, because of a CRC error in the response packet).

The Management Controller can detect a missing response packet as the occurrence of an NC-SI command timeout event.

#### 6.8.2.1 Command Timeout

The Management Controller can detect missing responses by implementing a command timeout interval. The timeout value chosen by the Management Controller shall not be less than Normal Execution Interval, T5. Upon detecting a timeout condition, the Management Controller should not make assumptions on the state of the unacknowledged command (for example, the command was dropped or the response was dropped), but should retransmit (retry) the previous command using the same IID it used in the initial command.

The Management Controller should try a command at least three times before assuming an error condition in the Network Controller.

It is possible that a Network Controller could send a response to the original command at the same time a retried command is being delivered. Under this condition, the Management Controller could get more than one response to the same command. Thus, the Management Controller should be capable of determining that it has received a second instance of a previous response packet. Dropped commands may be detected by the Management Controller as a timeout event waiting for the response.

#### 6.8.2.2 Handling Dropped Commands or Missing Responses

To recover from dropped commands or missing responses, the Management Controller can retransmit the unacknowledged command packet using the same IID that it used for the initial command.

The Network Controller shall be capable of reprocessing retransmitted (retried) commands without error or undesirable side effects. The Network Controller can determine that the command has been retransmitted by verifying that the IID is unchanged from the previous command.

### 6.8.3 Detecting Pass-through Traffic Interruption

The Network Controller might asynchronously enter the Initial State because of a reset or other event. In this case, the Network Controller stops transmitting Pass-through traffic on the RXD lines. Similarly, Pass-through traffic sent to the Network Controller may be dropped. If the Management Controller is not in the state of sending or receiving Pass-through traffic, it may not notice this condition. Thus the Management Controller should periodically issue a command to the Network Controller to test whether the Network Controller has entered the Initial State. How often this testing should be done is a choice of the Management Controller.

# 7 Arbitration in Configurations with Multiple Network Controller Packages

More than one Network Controller package on an NC-SI can be enabled for transmitting packets to the Management Controller. This specification defines two mechanisms to accomplish Network Controller package arbitration operations. One mechanism uses software commands provided by the Network Controller for the Management Controller to control whose turn it is to transmit traffic. The other mechanism uses hardware arbitration to share the single NC-SI bus. Implementations are required to support command-based Device Selection operation; the hardware arbitration method is optional.

## 7.1 General

Figure 9 is a simplified block diagram of the Sideband Interface being used in a multi-drop configuration. The RMII (upon which NC-SI is based) was originally designed for use as a point-to-point interconnect. Accordingly, only one party can transmit data onto the bus at any given time. There is no arbitration protocol intrinsic in the RMII to support managing multiple transmitters.

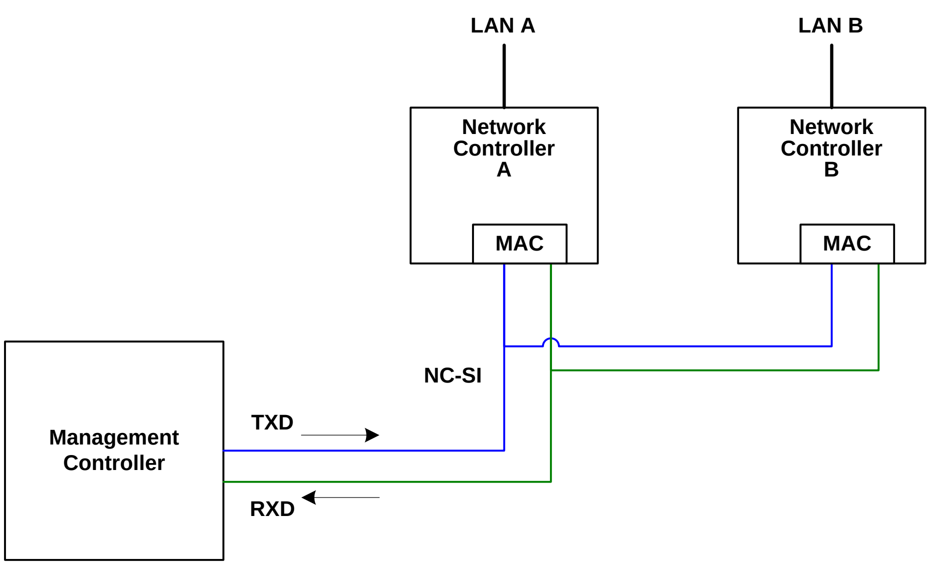



**Figure 9 – Basic Multi-Drop Block Diagram**

However, it is possible for multiple Network Controllers on the interface to be able to simultaneously *receive* traffic from the Management Controller that is being transmitted on the NC-SI TXD lines. The Network Controllers can receive commands from the Management Controller without having to arbitrate for the bus. This facilitates the Management Controller in delivering commands for setup and configuration of arbitration.

Arbitration allows multiple Network Controller packages that are attached to the interface to be enabled to share the RXD lines to deliver packets to the Management Controller.

This operation is summarized as follows:

- Only one Network Controller at a time can transmit packets on the RXD lines of the interface.
- Network Controllers can accept commands for configuring and controlling arbitration for the RXD lines.

## 7.2 Hardware Arbitration

To prevent two or more NC-SI packages from transmitting at the same time, a hardware-based arbitration scheme was devised to allow only one Network Controller package to drive the RX lines of the shared interface at any given time. This scheme uses a mechanism of passing messages (op-codes) between Network Controller packages to coordinate when a controller is allowed to transmit through the NC-SI interface.

### 7.2.1 General

Three conceptual modes of hardware arbitration exist: arbitration master assignment, normal operation, and bypass. After a package is initialized and has its Channel IDs assigned, it enters the arbitration master assignment mode. This mode assigns one package the role of an Arbitration Master (ARB_Master) that is responsible for initially generating a TOKEN op-code that is required for the normal operating mode. In the normal operating mode, the TOKEN op-code is passed from one package to the next in the ring. The package is allowed to use the shared RXD signals and transmit if the package has received the TOKEN op-code and has a packet to send. Bypass mode allows hardware arbitration op-codes to pass through a Network Controller package before it is initialized.

Hardware-based arbitration requires two additional pins (ARB_IN and ARB_OUT*)* on the Network Controller. The ARB_OUT pin of one package is connected to the ARB_IN pin of the next package to form a ring configuration, as illustrated in Figure 10. The timing requirements for hardware arbitration are designed to accommodate a maximum of four Network Controller packages. If the implementation consists of a single Network Controller package, the ARB_OUT pin may be connected to the ARB_IN pin on the same package, or may be left disconnected, in which case hardware arbitration should be disabled by using the Select Package command.

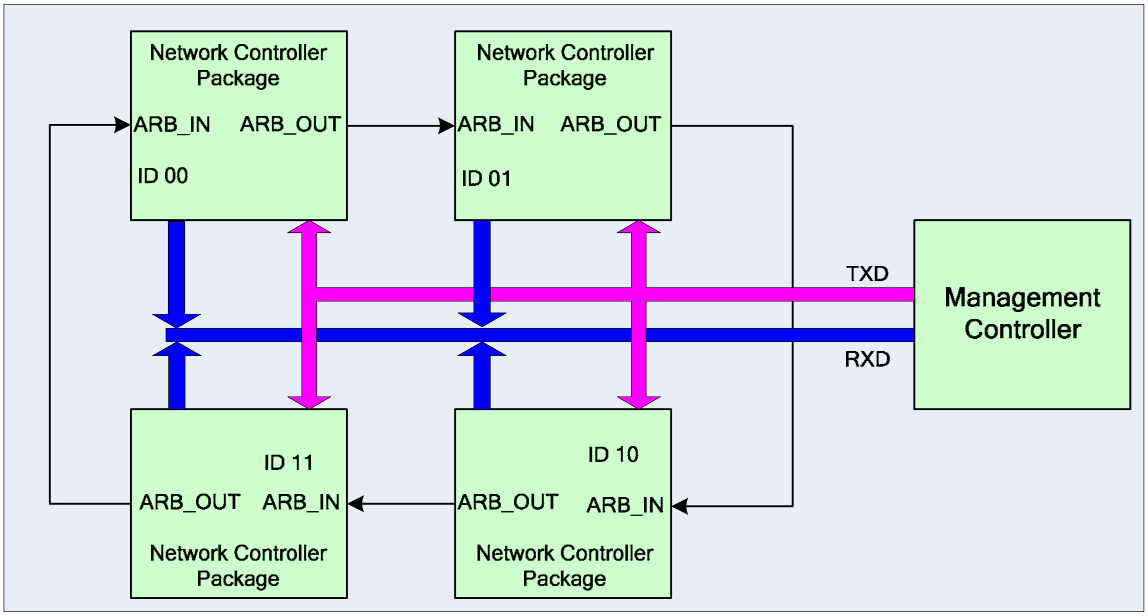



**Figure 10 – Multiple Network Controllers in a Ring Format**

Each Network Controller package sends out pulses on the ARB_OUT pin to create a series of symbols that form op-codes (commands) between Network Controllers. Each pulse is one clock wide and synchronized to REF_CLK. The hardware arbitration data bits follow the same timing specifications used for the TXD and RXD data bits (see 10.2.6). The pulses are di-bit encoded to ensure that symbols are correctly decoded. The symbols have the values shown in Table 4.

**Table 4 – Hardware Arbitration Di-bit Encoding**

| Symbol Name | Encoded Value |
|---|---|
| Esync | 11b |
| Ezero | 00b |
| Eone | 01b |
| Illegal symbol | 10b |

### 7.2.2 Hardware Arbitration Op-Codes

The hardware-based arbitration feature has five defined op-codes: IDLE, TOKEN, FLUSH, XON, and XOFF. Each op-code starts with an Esync symbol and is followed by either $E_{one}$ or $E_{zero}$ symbols. The legal op-codes are listed in Table 5.

**Table 5 – Hardware Arbitration Op-Code Format**

| Op-Code | Format |
|---|---|
| IDLE | $E_{sync}$ $E_{zero}$ $E_{zero}$ (110000b) |
| TOKEN | $E_{sync}$ $E_{one}$ $E_{zero}$ (110100b) |
| FLUSH | $E_{sync}$ $E_{one}$ $E_{one}$ $E_{zero}$ E(Package_ID[2:0]) $E_{zero}$ (11010100xxxxxx00b) |
| XOFF | $E_{sync}$ $E_{zero}$ $E_{one}$ $E_{zero}$ $E_{zero}$ $E_{zero}$ (110001000000b) |
| XON | $E_{sync}$ $E_{zero}$ $E_{one}$ $E_{one}$ $E_{zero}$ E(Package_ID[2:0]) $E_{zero}$ (1100010100uuuuuu00b) |

#### 7.2.2.1 Detecting Truncated Op-Codes

A truncated op-code is detected when the number of clocks between $E_{sync}$s is less than the number of bits required for the op-code. Note that any additional bits clocked in after a legitimate op-code is detected do not indicate an error condition and are ignored until the next $E_{sync}$.

#### 7.2.2.2 Handling Truncated or Illegal Op-Codes

When a Network Controller receives a truncated or illegal op-code, it should discard it.

#### 7.2.2.3 Relationship of Op-Codes Processing and Driving the RX Data Lines

A Network Controller package shall take no more than T9 REF_CLK times after receiving the last bit of the op-code to decode the incoming op-code and start generating the outgoing op-code. This time limit allows for decoding and processing of the incoming op-code under the condition that an outgoing op-code transmission is already in progress.

A package that has received a TOKEN and has packet data to transmit shall turn on its buffer and begin transmitting the packet data within T11 REF_CLK times of receiving the TOKEN, as illustrated in Figure 11. The package shall disable the RXD buffers before the last clock of the transmitted TOKEN.

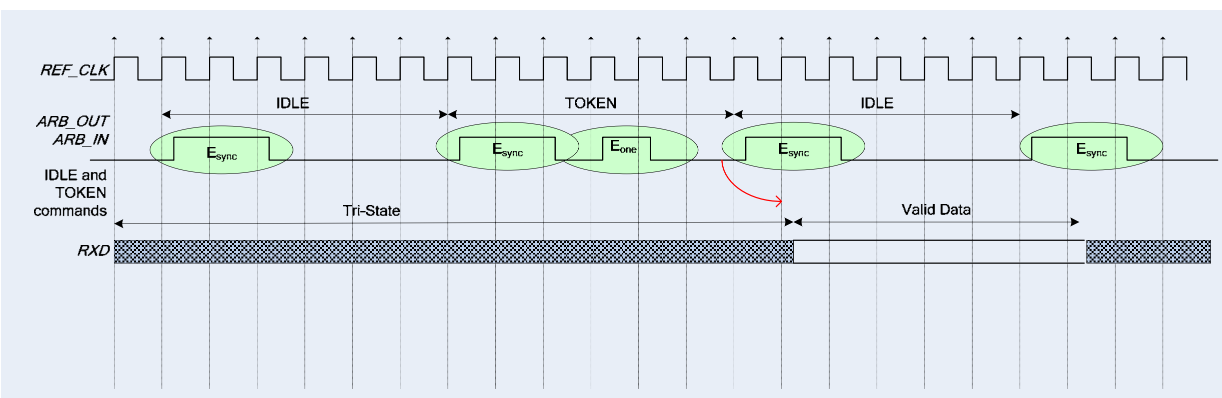



**Figure 11 – Op-Code to RXD Relationship**

### 7.2.3 Op-Code Operations

This clause describes the behavior associated with the five defined op-codes.

#### 7.2.3.1 TOKEN Op-Code

When a TOKEN op-code is received, the Network Controller package may drive the RXD signals to send only one of the following items: a Pass-through packet, a command response, or an AEN. One IEEE 802.3 PAUSE frame (XON or XOFF) may also be sent either before or after one of the previous packets, or on its own. While the Network Controller package is transmitting the data on the RXD signals of the interface, it shall generate IDLE op-codes on its ARB_OUT pin. Once a package completes its transmission, if any, it shall generate and send the TOKEN on its ARB_OUT pin.

#### 7.2.3.2 IDLE Op-Code

A package that has no other op-code to send shall continuously generate IDLE op-codes. Typically, a received IDLE op-code indicates that the TOKEN is currently at another package in the ring. This op-code is also used in the ARB_Master assignment process (for details, see 7.2.5).

#### 7.2.3.3 FLUSH Op-Code

A FLUSH op-code is used to establish an Arbitration Master for the ring when the package enters the Package Ready state or when the TOKEN is not received within the specified timeout, T8. This op-code is further explained in 7.2.5.

If the package receives a FLUSH op-code while it is in the middle of transmitting a packet onto NC-SI, it shall generate IDLE op-codes until the transmission is complete and then process the FLUSH op-code as described.

#### 7.2.3.4 Flow Control Op-Codes

The XON and XOFF op-codes are used to manage the generation of IEEE 802.3 PAUSE frames on the NC-SI. If the Network Controller supports flow control and flow control is enabled, the XOFF and XON op-codes behave as described in this clause. If the Network Controller does not support flow control or if flow control is not enabled, the Network Controller shall pass the op-codes to the next package.

Note: There is a maximum amount of time that the Network Controller may maintain a PAUSE. For more information, see 8.4.41.

##### 7.2.3.4.1 XOFF Op-Code

A Network Controller package that becomes congested while receiving packets from the NC-SI shall perform the following actions:

- If it does not have a TOKEN, it sends the XOFF op-code to the next package.
- If it has the TOKEN and has not previously sent an XOFF frame for this instance of congestion, it shall send a single XOFF frame (PAUSE frame with a pause time of `0xFFFF`) and will not generate an XOFF op-code.
- A package may also regenerate an XOFF frame or op-code if it is still congested and determines that the present PAUSE frame is about to expire.

When a package on the ring receives an XOFF op-code, it shall perform one of the following actions:

- If it does not have a TOKEN op-code, it passes the XOFF op-code to the next package in the ring.
- If it has the TOKEN, it shall send an XOFF frame (PAUSE frame with a pause time of `0xFFFF`) and will not regenerate the XOFF op-code. If it receives another XOFF op-code while sending the XOFF frame or a regular network packet, it discards the received XOFF op-code.

##### 7.2.3.4.2 XON Op-Code

XON frames (PAUSE frame with a pause time of `0x0000`) are used to signal to the Management Controller that the Network Controller packages are no longer congested and that normal traffic flow can resume. XON op-codes are used between the packages to coordinate XON frame generation. The package ID is included in this op-code to provide a mechanism to verify that every package is not congested before sending an XON frame to the Management Controller.

The XON op-code behaves as follows:

- When a package is no longer congested, it generates an XON op-code with its own Package ID. This puts the package into the ‘waiting for its own XON’ state.
- A package that receives the XON op-code takes one of the following actions:
  – If it is congested, it replaces the received XON op-code with the IDLE op-code. This action causes the XON op-code to be discarded. Eventually, the congested package generates its own XON op-code when it exits the congested state.
  – If the package is not congested and is not waiting for the XON op-code with own Package ID, it forwards the received XON op-code to the next package in the ring.

    NOTE: If the received XON op-code contains the package’s own Package ID, the op-code should be discarded.

  – If the package is not congested and is waiting for its own XON op-code, it performs one of the following actions:
    - If it receives an XON op-code with a Package ID that is higher than its own, it replaces the XON op-code with its own Package ID.
    - If it receives an XON op-code with a Package ID lower than its own, it passes that XON op-code to the next package and it exits the ‘waiting for its own XON’ state.
    - If it receives an XON op-code with the Package ID equal to its own, it sends an XON frame on the NC-SI when it receives the TOKEN op-code and exits the ‘waiting for its own XON’ state.

    NOTE: More than one XON op-code with the same Package ID may be received while waiting for the TOKEN and while sending the XON frame. These additional XON op-codes should be discarded.

- If a package originates an XON op-code but receives an XOFF op-code, it terminates its XON request so that it does not output an XON frame when it receives the TOKEN.

  NOTE: This behavior should not occur because the Management Controller will be in the Pause state at this point.

- A package that generated an XON op-code may receive its own XON op-code back while it has the TOKEN op-code. In this case, it may send a regular packet (Pass-through, command response, or AEN) to the Management Controller (if it has one to send), an XON frame, or both.

### 7.2.4 Bypass Mode

When the Network Controller package is in bypass mode, data received on the ARB_IN pin is redirected to the ARB_OUT pin within the specified clock delay. This way, arbitration can continue between other devices in the ring.

A package in bypass mode shall take no more than T10 REF_CLK times to forward data from the ARB_IN pin to the ARB_OUT pin. The transition in and out of bypass mode may result in a truncated op-code.

A Network Controller package enters into bypass mode immediately upon power up and transitions out of this mode after the Network Controller completes its startup/initialization sequence.

### 7.2.5 Hardware Arbitration Startup

Hardware arbitration startup works as follows:

1) All the packages shall be in bypass mode within $T_{pwrz}$ seconds of NC-SI power up.
2) As each package is initialized, it shall continuously generate FLUSH op-codes with its own Package ID.
3) The package then participates in the ARB_MSTR assignment process described in the following clause.

### 7.2.6 ARB_MSTR Assignment

ARB_MSTR assignment works as follows:

1) When a package receives a FLUSH op-code with a Package ID numerically smaller than its own, it shall forward on the received FLUSH op-code. If the received FLUSH op-code's Package ID is numerically larger than the local Package ID, the package shall continue to send its FLUSH op-code with its own Package ID. When a package receives a FLUSH op-code with its own Package ID, it becomes the master of the ring (ARB_MSTR).
2) The ARB_MSTR shall then send out IDLE op-codes until it receives an IDLE op-code.
3) Upon receiving the IDLE op-code, the ARB_MSTR shall be considered to be in possession of the TOKEN op-code (see 7.2.3.1).

   NOTE: If the package receives a FLUSH op-code while it is in the middle of transmitting a packet onto NC-SI, it shall generate IDLE op-codes until the transmission is complete and then process the FLUSH op-code as described.

### 7.2.7 Token Timeout Mechanism

Each Network Controller package that supports hardware-based arbitration control shall implement a timeout mechanism in case the TOKEN op-code is not received. When a package has a packet to send, it starts its timer. If it does not receive a TOKEN prior to the TOKEN timeout, the package shall send a FLUSH op-code. This restarts the arbitration process.

The timer may be programmable depending on the number of packages in the ring. The timeout value is designed to accommodate up to four packages, each sending the largest packet (1536 bytes) plus possible XON or XOFF frame transmission and op-code processing time. The timeout shall be no fewer than T8 cycles of the REF_CLK.

### 7.2.8 Timing Considerations

The ARB_OUT and ARB_IN pins shall follow the timing specifications outlined in Clause 10.

To improve the efficiency of the multi-drop NC-SI, TOKEN op-code generation may overlap the Inter Packet Gap (IPG) defined by the 802.3 specification, as shown in Figure 12. The TOKEN op-code shall be sent no earlier than the last T13 REF_CLK cycles of the IPG.

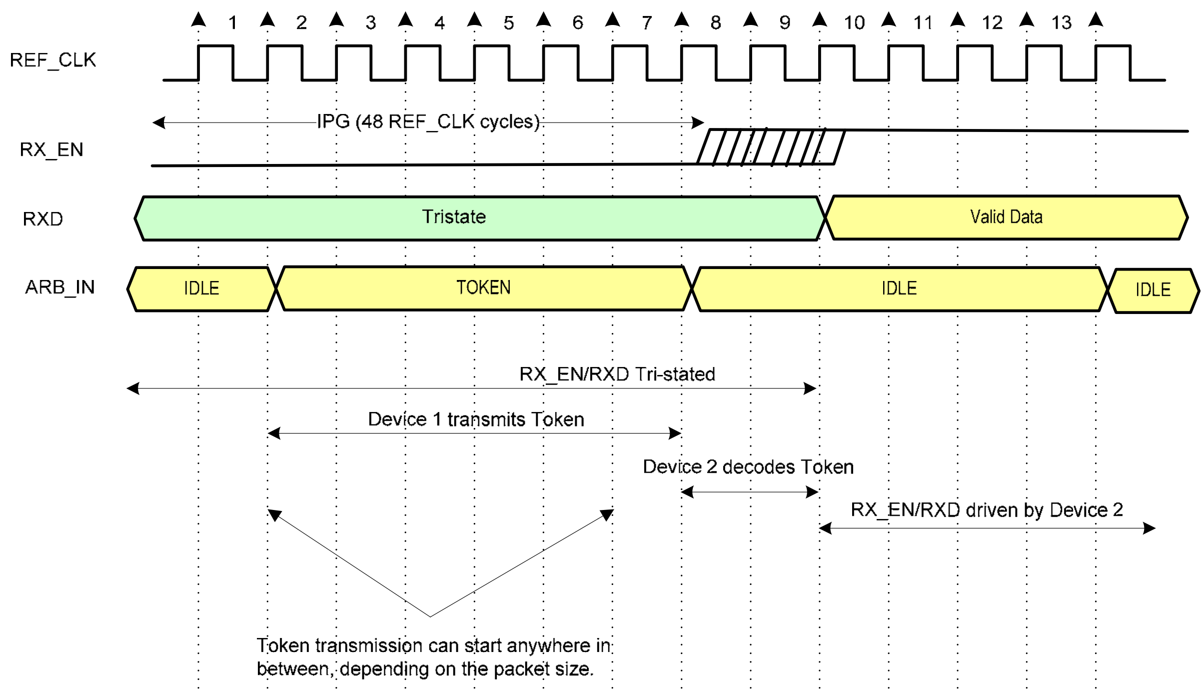
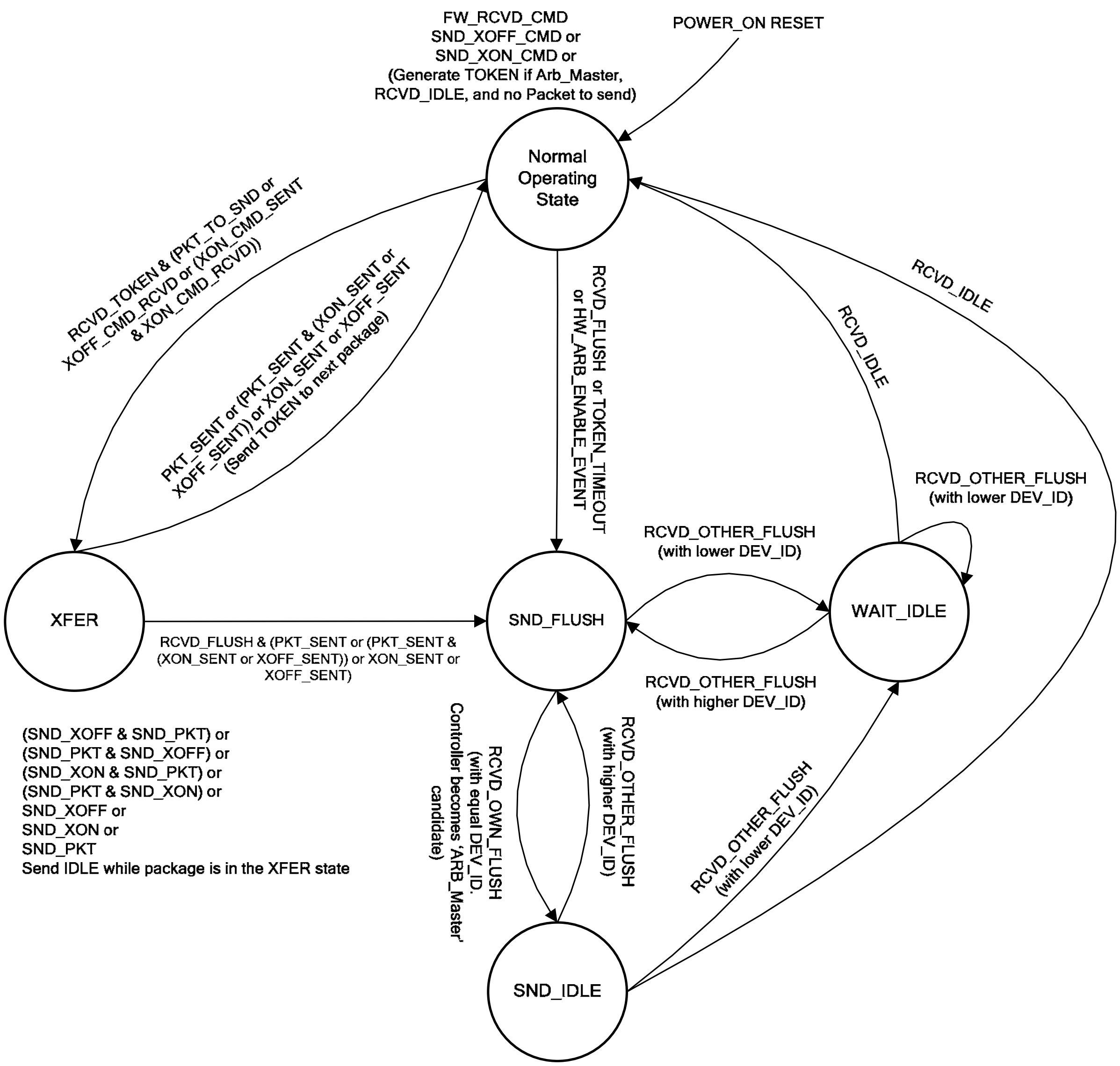



**Figure 12 – Example TOKEN to Transmit Relationship**

1358 ### 7.2.9 Example Hardware Arbitration State Machine

1359 The state machine diagram shown in Figure 13 is provided as a guideline to help illustrate the startup
1360 process and op-code operations described in the preceding clauses. Where Figure 13 may vary from the
1361 preceding specifications, the preceding specifications shall take precedence.




1362

1363 **Figure 13 – Hardware Arbitration State Machine**

1364 The states and events shown in Figure 13 are described in Table 6 and Table 7, respectively.

1365 **Table 6 – Hardware Arbitration States**

| State | Action |
|---|---|
| Normal Operating State | This state is the normal operating state for hardware arbitration. The following actions happen in this state:<br>• FW_RCVD_CMD: Forward received command. As op-codes are received and acted upon, the resulting op-code is sent to the next package. For example, the TOKEN op-code is received and no packet data is available to send, so the TOKEN op-code is sent to the next package in the ring.<br>• SND_XOFF_CMD: Send the XOFF op-code to the next package. This action happens when the specific conditions are met as described in 7.2.3.<br>• SND_XON_CMD: Send the XON op-code to the next package. This action happens when the specific conditions are met as described in 7.2.3.<br>• If the Network Controller is ARB_Master, it generates the TOKEN op-code upon receiving an IDLE op-code at the end of the FLUSH process.<br>• The RXD lines will be in a high-impedance condition in this state. |
| XFER | In this state, data is sent on the RXD lines. This data will be a Pass-through packet, response packet, XON (Pause Off) packet, XOFF (Pause On) packet, or AEN. (An XON or XOFF packet can be sent in addition to a Pass-through packet, response packet, or AEN.) IDLE op-codes are sent to the next package while the device is in the XFER state.<br>The following actions happen in this state:<br>• SND_XON: Transmit an XON frame (Pause Off) to the Management Controller.<br>• SND_XOFF: Transmit an XOFF frame (Pause On) to the Management Controller.<br>• SND_PKT: Transmit a Pass-through packet, response packet, or AEN to the Management Controller.<br>• The TOKEN op-code is sent to the next package upon completion of the transfer. |
| SND_FLUSH | This state is the entry point for determining the ARB_Master among the packages. In this state, the FLUSH op-code is continuously sent. This state is exited upon receiving a FLUSH op-code that has a DEV_ID that is equal to the package’s own DEV_ID. |
| SND_IDLE | This is the final state for determining the ARB_Master, entered when a device’s own FLUSH op-code is received. In this state, the IDLE op-code is continuously sent. |
| WAIT_IDLE | This state is entered when a FLUSH command is received from another package with a lower Device ID. When an IDLE op-code is received, the ARB_Master has been determined and the device transitions to the Normal Operating State. |

**Table 7 – Hardware Arbitration Events**

| Event | Description |
|---|---|
| RCVD_TOKEN | A TOKEN op-code was received. |
| RCVD_IDLE | An IDLE op-code was received. |
| XOFF_SENT | The Pause On frame was sent on the RXD interface. |
| XON_SENT | The Pause Off frame was sent on the RXD interface. |
| PKT_TO_SND | The Network Controller package has a Pass-through packet, command response packet, XON (Pause Off) frame, XOFF (Pause On) frame, or AEN to send. |
| XON_CMD_RCVD | A package received an XON op-code with its own Package ID. |
| XOFF_CMD_RCVD | An XOFF op-code was received. |
| XON_CMD_SENT | A package sent an XON op-code with its own Package ID. |
| RCVD_FLUSH | A FLUSH op-code was received. |
| TOKEN_TIMEOUT | The timeout limit expired while waiting for a TOKEN op-code. |
| HW_ARB_ENABLE_EVENT | This event begins ARB_MSTR assignment. This event occurs just after the Network Controller package initializes or when hardware arbitration is re-enabled through the Select Package command. |
| RCVD_OTHER_FLUSH | A package received a FLUSH op-code with a Package ID other than its own. |
| RCVD_OWN_FLUSH | A package received a FLUSH op-code with a Package ID equal to its own. |

## 7.3 Command-based Arbitration

If hardware arbitration is not being used, the Select Package and Deselect Package commands can be used to control which Network Controller package has the ability to transmit on the RXD lines. Because only one Network Controller package is allowed to transmit on the RXD lines, the Management Controller shall only have one package in the selected state at any given time. For more information, see 8.4.5 and 8.4.7.

# 8 Packet Definitions

This clause presents the formats of NC-SI packets and their relationship to frames used to transmit and receive those packets on NC-SI.

## 8.1 NC-SI Packet Encapsulation

The NC-SI is an Ethernet interface adhering to the standard IEEE 802.3 Ethernet frame format. Whether or not the Network Controller accepts runt packets is unspecified.

As shown in Figure 14, this L2, or data link layer, frame format encapsulates all NC-SI packets, including command and response packets, as the L2 frame payload data by adding a 14-byte header to the front of the data and appending a 4-byte Frame Check Sequence (FCS) to the end.

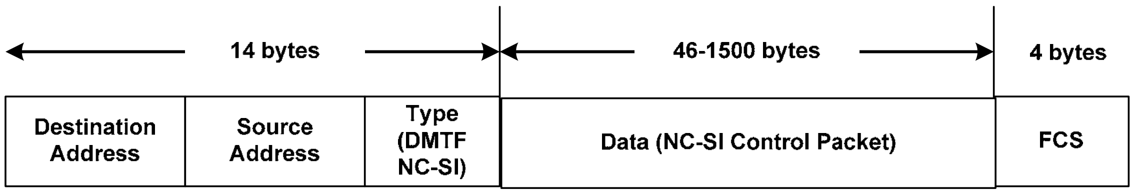



**Figure 14 – Ethernet Frame Encapsulation of NC-SI Packet Data**

### 8.1.1 Ethernet Frame Header

The Management Controller shall format the 14-byte Ethernet frame header so that when it is received, it shall be formatted in the big-endian byte order shown in Table 8.

Channels shall accept Pass-through packets that meet the IEEE 802.3 frame requirements.

**Table 8 – Ethernet Header Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..03** | DA5= `0xFF` | $DA_4$= `0xFF` | $DA_3$= `0xFF` | $DA_2$= `0xFF` |
| **04..07** | $DA_1$= `0xFF` | $DA_0$= `0xFF` | $SA_5$ | $SA_4$ |
| **08..11** | $SA_3$ | $SA_2$ | $SA_1$ | $SA_0$ |
| **12..13** | EtherType = `0x88F8 (DMTF NC-SI)` | | | |

#### 8.1.1.1 Destination Address (DA)

Bytes 0–5 of the header represent bytes 5–0 of the Ethernet Destination Address field of an L2 header.

The channel is not assigned a specific MAC address and the contents of this field are not interpreted as a MAC address by the Management Controller or the Network Controller. However, the DA field in all NC-SI control packets shall be set to the broadcast address (FF:FF:FF:FF:FF:FF) for consistency.

If the Network Controller receives a control packet with a Destination Address other than FF:FF:FF:FF:FF:FF, the Network Controller may elect to accept the packet, drop it, or return a response packet with an error response/reason code.

#### 8.1.1.2 Source Address (SA)

Bytes 6–11 of the header represent bytes 5–0 of the Ethernet Source Address field of the Ethernet header. The contents of this field may be set to any value. The Network Controller may use FF:FF:FF:FF:FF:FF as the source address for NC-SI Control packets that it generates.

#### 8.1.1.3 EtherType

The final two bytes of the header, bytes 12..13, represent bytes 1..0 of the EtherType field of the Ethernet header. For NC-SI Control packets, this field shall be set to a fixed value of `0x88F8` as assigned to the NC-SI by the IEEE. This value allows NC-SI Control packets to be differentiated from other packets in the overall packet stream.

### 8.1.2 Frame Check Sequence

The Frame Check Sequence (FCS) shall be added at the end of the frame to provide detection of corruption of the frame. Any frame with an invalid FCS shall be discarded.

## 8.2 Control Packet Data Structure

Each NC-SI Control packet is made up of a 16-byte packet header and a payload section whose length is specific to the packet type.

### 8.2.1 Control Packet Header

The 16-byte control packet header is used in command, response, and AEN packets, and contains data values intended to allow the packet to be identified, validated, and processed. The packet header is in big-endian byte order, as shown in Table 9.

**Table 9 – Control Packet Header Format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..03** | MC ID | Header Revision | Reserved | IID |
| **04..07** | Control Packet Type | Ch. ID | Reserved | Payload Length |
| **08..11** | Reserved | | | |
| **12..15** | Reserved | | | |

#### 8.2.1.1 Management Controller ID

In Control packets, this 1-byte field identifies the Management Controller issuing the packet. For this version of the specification, Management Controllers should set this field to `0x00` (zero). Network Controllers responding to command packets should copy the Management Controller ID field from the command packet header into the response packet header. For AEN packets, this field should be copied from the parameter that was set using the AEN Enable command.

#### 8.2.1.2 Header Revision

This 1-byte field identifies the version of the Control packet header in use by the sender. For this version of the specification, the header revision is `0x01`.

#### 8.2.1.3 Instance ID (IID)

This 1-byte field contains the IID of the command and associated response. The Network Controller can use it to differentiate retried commands from new instances of commands. The Management Controller can use this value to match a received response to the previously sent command. For more information, see 6.3.1.1.

#### 8.2.1.4 Control Packet Type

This 1-byte field contains the Identifier that is used to identify specific commands and responses, and to differentiate AENs from responses. Each NC-SI command is assigned a unique 7-bit command type value in the range `0x00..0x7F`. The proper response type for each command type is formed by setting the most significant bit (bit 7) in the original 1-byte command value. This allows for a one-to-one correspondence between 128 unique response types and 128 unique command types.

#### 8.2.1.5 Channel ID

This 1-byte field contains the Network Controller Channel Identifier. The Management Controller shall set this value to specify the package and internal channel ID for which the command is intended.

In a multi-drop configuration, all commands are received by all NC-SI Network Controllers present in the configuration. The Channel ID is used by each receiving Network Controller to determine if it is the intended recipient of the command. In Responses and AENs, this field carries the ID of the channel from which the response of AEN was issued.

#### 8.2.1.6 Payload Length

This 12-bit field contains the length, in bytes, of any payload data present in the command or response frame following the NC-SI packet header. This value does not include the length of the NC-SI header, the checksum value, or any padding that might be present.

#### 8.2.1.7 Reserved

These fields are reserved for future use and should be written as zeros and ignored when read.

### 8.2.2 Control Packet Payload

The NC-SI packet payload may contain zero or more defined data values depending on whether the packet is a command or response packet, and on the specific type. The NC-SI packet payload is always formatted in big-endian byte order, as shown in Table 10.

**Table 10 – Generic Example of Control Packet Payload**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..03** | Data0$_3$ | Data0$_2$ | Data0$_1$ | Data0$_0$ |
| **04..07** | Data1$_7$ | Data1$_6$ | Data1$_5$ | Data1$_4$ |
| **08..11** | Data1$_3$ | Data1$_2$ | Data1$_1$ | Data1$_0$ |
| **..** | | | | |
| **…** | DataN-1$_4$ | DataN-1$_3$ | DataN-1$_2$ | DataN-1$_1$ |
| **…** | DataN-1$_0$ | Payload Pad (as required) | | |
| **…** | 2s Complement Checksum Compensation | | | |
| **…** | Ethernet Packet Pad (as required) | | | |

#### 8.2.2.1 Data

As shown in Table 10, the bytes following the NC-SI packet header may contain payload data fields of varying sizes, and which may be aligned or require padding. In the case where data is defined in the payload, all data-field byte layouts (Data0–Data-1) shall use big-endian byte ordering with the most significant byte of the field in the lowest addressed byte position (that is, coming first).

#### 8.2.2.2 Payload Pad

If the payload is present and does not end on a 32-bit boundary, one to three padding bytes equal to `0x00` shall be present to align the checksum field to a 32-bit boundary.

#### 8.2.2.3 2's Complement Checksum Compensation

This 4-byte field contains the 32-bit checksum compensation value that may be included in each command and response packet by the sender of the packet. When it is implemented, the checksum compensation shall be computed as the 2's complement of the checksum, which shall be computed as the 32-bit unsigned sum of the NC-SI packet header and NC-SI packet payload interpreted as a series of 16-bit unsigned integer values. A packet receiver supporting packet checksum verification shall use the checksum compensation value to verify packet data integrity by computing the 32-bit checksum described above, adding to it the checksum compensation value from the packet, and verifying that the result is 0.

Verification of non-zero NC-SI packet checksum values is optional. An implementation may elect to generate the checksums and may elect to verify checksums that it receives. The checksum field is generated and handled according to the following rules:

- A checksum field value of all zeros specifies that a header checksum is not being provided for the NC-SI Control packet, and that the checksum field value shall be ignored when processing the packet.
- If the originator of an NC-SI Control packet is not generating a checksum, the originator shall use a value of all zeros for the header checksum field.
- If a non-zero checksum field is generated for an NC-SI Control packet, that header checksum field value shall be calculated using the specified algorithm.
- All receivers of NC-SI Control packets shall accept packets with all zeros as the checksum value (provided that other fields and the CRC are correct).
- The receiver of an NC-SI Control packet may reject (silently discard) a packet that has an incorrect non-zero checksum.
- The receiver of an NC-SI Control packet may ignore any non-zero checksums that it receives and accept the packet, even if the checksum value is incorrect (that is, an implementation is not required to verify the checksum field).
- A controller that generates checksums is not required to verify checksums that it receives.
- A controller that verifies checksums is not required to generate checksums for NC-SI Control packets that it originates.

#### 8.2.2.4 Ethernet Packet Pad

Per IEEE 802.3, all Ethernet frames shall be at least 64 bytes in length, from the DA through and including FCS. For NC-SI packets, this requirement applies to the Ethernet header and payload, which includes the NC-SI Control packet header and payload. Most NC-SI Control packets are less than the minimum Ethernet frame payload size of 46 bytes in length and require padding to comply with IEEE 802.3.

### 8.2.3 Command Packet Payload

Command packets have no common fixed payload format.

### 8.2.4 Response Packet Payload

Unlike command packets that do not necessarily contain payload data, all response packets carry at least a 4-byte payload. This default payload carries the response codes and reason codes (described in 8.2.5) that provide status on the outcome of processing the originating command packet, and is present in all response packet payload definitions.

The default payload occupies bytes `00..03` of the response packet payload, with any additional response-packet-specific payload defined to follow starting on the next word. All response packet payload fields are defined with big-endian byte ordering, as shown in Table 11.

**Table 11 – Generic Example of Response Packet Payload Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..03** | Response Code | | Reason Code | |
| **..** | ... | ... | ... | ... |
| **...** | DataN-$1_4$ | DataN-$1_3$ | DataN-$1_2$ | DataN-$1_1$ |
| **...** | DataN-$1_0$ | Word Pad (as required) | | |
| **...** | 2s Complement Checksum Compensation | | | |
| **...** | Ethernet Packet Pad (as required) | | | |

## 8.2.5 Response Codes and Reason Codes

Response codes and reason codes are status values that are returned in the responses to NC-SI commands. The response code values provide a general categorization of the status being returned. The reason code values provide additional detail related to a particular response code.

### 8.2.5.1 General

Response codes and reason codes are divided into numeric ranges that distinguish whether the values represent standard codes that are defined in this specification or are vendor/OEM-specific values that are defined by the vendor of the controller.

The response code is a 2-byte field where values from `0x00` through `0x7F` are reserved for definition by this specification. Values from `0x80` through `0xFF` are vendor/OEM-specific codes that are defined by the vendor of the controller.

The reason code is a 2-byte field. The ranges of values are defined in Table 12.

**Table 12 – Reason Code Ranges**

| MS-byte | LS-byte | Description |
|---|---|---|
| `00h` | `0x00–0x7F` | Standard generic reason codes<br>This range of values for the lower byte is used for reason codes that are not specific to a particular command but can be used as reason codes in responses for any command. The values in this range are reserved for definition by this specification. |
| | `0x80–0x7F` | Vendor/OEM generic reason codes<br>This range of values for the lower byte is used for reason codes that are not specific to a particular command but can be used as reason codes in responses for any command. Values in this range are defined by the vendor of the controller. |

| MS-byte | LS-byte | Description |
|---|---|---|
| Command Number | `0x00–0x7F` | Standard command-specific reason codes<br><br>This range of values for the lower byte is used for reason codes that are specific to a particular command. The upper byte holds the value of the command for which the reason code is defined. The values in this range are reserved for definition by this specification. |
| | `0x80–0xFF` | Vendor/OEM command-specific reason codes<br><br>This range of values for the lower byte is used for reason codes that are specific to a particular command. The upper byte holds the value of the command for which the reason code is defined. Values in this range are defined by the vendor of the controller. |

#### 8.2.5.2 Response Code and Reason Code Values

The standard response code values are defined in Table 13, and the standard reason code values are defined in Table 14. Command-specific values, if any, are defined in the clauses that describe the response data for the command. Unless otherwise specified, the standard reason codes may be used in combination with any response code.

**Table 13 – Standard Response Code Values**

| Value | Description | Comment |
|---|---|---|
| `0x0` | Command Completed | Returned for a successful command completion |
| `0x1` | Command Failed | Returned to report that a valid command could not be processed or failed to complete correctly |
| `0x2` | Command Unavailable | Returned to report that a command is temporarily unavailable for execution because the controller is in a transient state or busy condition |
| `0x3` | Command Unsupported | Returned to report that a command is not supported by the implementation |
| `0x8000–0xFFFF` | Vendor/OEM-specific | Response codes defined by the vendor of the controller |

**Table 14 – Standard Reason Code Values**

| Value | Description | Comment |
|---|---|---|
| `0x0000` | No Error/No Reason Code | When used with the Command Completed response code, indicates that the command completed normally. Otherwise this value indicates that no additional reason code information is being provided. |
| `0x0001` | Interface Initialization Required | Returned for all commands when the channel is in the Initial State, until the channel receives a Clear Initial State command |
| `0x0002` | Parameter Is Invalid, Unsupported, or Out-of-Range | Returned when a received parameter value is outside of the acceptable values for that parameter |
| `0x0003` | Channel Not Ready | May be returned when the channel is in a transient state in which it is unable to process commands normally |

| Value | Description | Comment |
|---|---|---|
| 0x0004 | Package Not Ready | May be returned when the package and channels within the package are in a transient state in which normal command processing cannot be done |
| 0x0005 | Invalid payload length | The payload length in the command is incorrect for the given command |
| 0x7FFF | Unknown / Unsupported Command Type | Returned when the command type is unknown or unsupported |
| 0x8000–0xFFFF | OEM Reason Code | Vendor-specific reason code defined by the vendor of the controller |

### 8.2.6 AEN Packet Format

AEN packets shall follow the general packet format of Control packets, with the IID field set to 0 because, by definition, the Management Controller does not send a response packet to acknowledge an AEN packet. The Control Packet Type field shall have the value 0xFF. The originating Network Controller shall fill in the Channel ID (Ch. ID) field with its own ID to identify itself as the source of notification. Currently, three AEN types are defined in the AEN Type field. Table 15 represents the general AEN packet format.

**Table 15 – AEN Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..03** | MC ID = 0x0 | 0x01 | Reserved | IID = 0x0 |
| **04..07** | Control Packet Type = 0xFF | Originating Ch. ID | Reserved | Payload Length = 0x4 |
| **08..11** | Reserved | | | |
| **12..15** | Reserved | | | |
| **16..19** | Reserved | | | AEN Type |
| **20..23** | OPTIONAL AEN Data | | | |
| **24..27** | Checksum | | | |

### 8.2.7 AEN Packet Data Structure

The AEN Type field (8-bit) has the values shown in Table 16.

**Table 16 – AEN Types**

| Value | AEN Type |
|---|---|
| 0x0 | Link Status Change |
| 0x1 | Configuration Required |
| 0x2 | Host NC Driver Status Change |
| 0x3..0x7F | Reserved |
| 0x80..0xFF | OEM-specific AENs |

## 8.3 Control Packet Type Definitions

Command packet types are in the range of `0x00` to `0x7F`. Table 17 describes each command, its corresponding response, and the type value for each.

Mandatory (M), Optional (O), and Conditional (C) refer to command support requirements for the Network Controller.

**Table 17 – Command and Response Types**

| Command Type | Command Name | Description | Response Type | Command Support Requirement |
|---|---|---|---|---|
| 0x00 | Clear Initial State | Used by the Management Controller to acknowledge that the Network Controller is in the Initial State | 0x80 | M |
| 0x01 | Select Package | Used to explicitly select a controller package to transmit packets through the NC-SI interface | 0x81 | M |
| 0x02 | Deselect Package | Used to explicitly instruct the controller package to stop transmitting packets through the NC-SI interface | 0x82 | M |
| 0x03 | Enable Channel | Used to enable the NC-SI channel and to cause the forwarding of bidirectional Management Controller packets to start | 0x83 | M |
| 0x04 | Disable Channel | Used to disable the NC-SI channel and to cause the forwarding of bidirectional Management Controller packets to cease | 0x84 | M |
| 0x05 | Reset Channel | Used to synchronously put the Network Controller back to the Initial State | 0x85 | M |
| 0x06 | Enable Channel Network TX | Used to explicitly enable the channel to transmit Pass-through packets onto the network | 0x86 | M |
| 0x07 | Disable Channel Network TX | Used to explicitly disable the channel from transmitting Pass-through packets onto the network | 0x87 | M |
| 0x08 | AEN Enable | Used to control generating AENs | 0x88 | C |
| 0x09 | Set Link | Used during OS absence to force link settings, or to return to auto-negotiation mode | 0x89 | M |
| 0x0A | Get Link Status | Used to get current link status information | 0x8A | M |
| 0x0B | Set VLAN Filter | Used to program VLAN IDs for VLAN filtering | 0x8B | M |
| 0x0C | Enable VLAN | Used to enable VLAN filtering of Management Controller RX packets | 0x8C | M |
| 0x0D | Disable VLAN | Used to disable VLAN filtering | 0x8D | M |
| 0x0E | Set MAC Address | Used to configure and enable unicast and multicast MAC address filters | 0x8E | M |
| 0x10 | Enable Broadcast Filtering | Used to enable full or selective broadcast packet filtering | 0x90 | M |

| Command Type | Command Name | Description | Response Type | Command Support Requirement |
|---|---|---|---|---|
| 0x11 | Disable Broadcast Filtering | Used to disable all broadcast packet filtering, and to enable the forwarding of broadcast packets | 0x91 | M |
| 0x12 | Enable Global Multicast Filtering | Used to disable forwarding of all multicast packets to the Management Controller | 0x92 | C |
| 0x13 | Disable Global Multicast Filtering | Used to enable forwarding of all multicast packets to the Management Controller | 0x93 | C |
| 0x14 | Set NC-SI Flow Control | Used to configure IEEE 802.3 flow control on the NC-SI | 0x94 | O |
| 0x15 | Get Version ID | Used to get controller-related version information | 0x95 | M |
| 0x16 | Get Capabilities | Used to get optional functions supported by the NC-SI | 0x96 | M |
| 0x17 | Get Parameters | Used to get configuration parameter values currently in effect on the controller | 0x97 | M |
| 0x18 | Get Controller Packet Statistics | Used to get current packet statistics for the Ethernet Controller | 0x98 | O |
| 0x19 | Get NC-SI Statistics | Used to request the packet statistics specific to the NC-SI | 0x99 | O |
| 0x1A | Get NC-SI Pass-through Statistics | Used to request NC-SI Pass-through packet statistics | 0x9A | O |
| 0x50 | OEM Command | Used to request vendor-specific data | 0xD0 | O |
| Key: M = Mandatory (required)<br>O = Optional<br>C = Conditional (see command description) | | | | |

1544 ## 8.4 Command and Response Packet Formats

1545 This clause describes the format for each of the NC-SI Commands and corresponding responses.

1546 The corresponding response packet format shall be mandatory when a given command is supported.

1547 ### 8.4.1 NC-SI Command Frame Format

1548 Table 18 illustrates the NC-SI frame format that shall be accepted by the Network Controller.

1549 **Table 18 – Example of Complete Minimum-Sized NC-SI Command Packet**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..03** | 0xFF | 0xFF | 0xFF | 0xFF |
| **04..07** | 0xFF | 0xFF | 0xXX | 0xXX |
| **08..11** | 0xXX | 0xXX | 0xXX | 0xXX |
| **12..15** | 0x88F8 | | MC ID | Header Revision |

| | Bits | | | | |
|---|---|---|---|---|---|
| **Bytes** | **31..24** | | **23..16** | **15..08** | **07..00** |
| **16..19** | Reserved | | IID | Command Type | Ch. ID |
| **20..23** | Reserved | Payload Length | | Reserved | |
| **24..27** | Reserved | | | Reserved | |
| **28..31** | Reserved | | | Checksum (3..2) | |
| **32..35** | Checksum (1..0) | | | Pad | |
| **36..39** | Pad | | | | |
| **40..43** | Pad | | | | |
| **44..47** | Pad | | | | |
| **48..51** | Pad | | | | |
| **52..55** | Pad | | | | |
| **56..59** | Pad | | | | |
| **60..63** | FCS | | | | |

### 8.4.2 NC-SI Response Packet Format

Table 19 illustrates the NC-SI response packet format that shall be transmitted by the Network Controller.

**Table 19 – Example of Complete Minimum-Sized NC-SI Response Packet**

| | Bits | | | | |
|---|---|---|---|---|---|
| **Bytes** | **31..24** | | **23..16** | **15..08** | **07..00** |
| **00..03** | `0xFF` | | `0xFF` | `0xFF` | `0xFF` |
| **04..07** | `0xFF` | | `0xFF` | `0xFF` | `0xFF` |
| **08..11** | `0xFF` | | `0xFF` | `0xFF` | `0xFF` |
| **12..15** | `0x88F8` | | | MC ID | Header Revision |
| **16..19** | Reserved | | IID | Response Type | Ch. ID |
| **20..23** | Reserved | Payload Length | | Reserved | |
| **24..27** | Reserved | | | Reserved | |
| **28..31** | Reserved | | | Response Code | |
| **32..35** | Reason Code | | | Checksum (3..2) | |
| **36..39** | Checksum (1..0) | | | Pad | |
| **40..43** | Pad | | | | |
| **44..47** | Pad | | | | |
| **48..51** | Pad | | | | |
| **52..55** | Pad | | | | |
| **56..59** | Pad | | | | |
| **60..63** | FCS | | | | |

### 8.4.3 Clear Initial State Command (`0x00`)

The Clear Initial State command provides the mechanism for the Management Controller to acknowledge that it considers a channel to be in the Initial State (typically because the Management Controller received an “Interface Initialization Required” reason code) and to direct the Network Controller to start accepting commands for initializing or recovering the NC-SI operation. When in the Initial State, the Network Controller shall return the “Interface Initialization Required” reason code for all commands until it receives the Clear Initial State command.

If the channel is in the Initial State when it receives the Clear Initial State command, the command shall cause the Network Controller to stop returning the “Interface Initialization Required” reason code. The channel shall also treat any subsequently received instance ID numbers as IDs for new command instances, not retries.

If the channel is not in the Initial State when it receives this command, it shall treat any subsequently received instance ID numbers as IDs for new command instances, not retries.

Table 20 illustrates the packet format of the Clear Initial State command.

**Table 20 – Clear Initial State Command Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.4 Clear Initial State Response (`0x80`)

Currently no command-specific reason code is identified for this response (see Table 21).

**Table 21 – Clear Initial State Response Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.5 Select Package Command (`0x01`)

A package is considered to be “selected” when its NC-SI output buffers are allowed to transmit packets through the NC-SI interface. Conversely, a package is “deselected” when it is not allowed to transmit packets through the NC-SI interface.

The Select Package command provides a way for a Management Controller to explicitly take a package out of the deselected state and to control whether hardware arbitration is enabled for the package. (Similarly, the Deselect Package command allows a Management Controller to explicitly deselect a package.)

The NC-SI package in the Network Controller shall also become selected if the package receives any other NC-SI command that is directed to the package or to a channel within the package.

The Select Package command is addressed to the package, rather than to a particular channel (that is, the command is sent with a Channel ID where the Package ID subfield matches the ID of the intended package and the Internal Channel ID subfield is set to `0x1F`).

More than one package can be in the selected state simultaneously if hardware arbitration is used between the selected packages and is active. The hardware arbitration logic ensures that buffer conflicts will not occur between selected packages.

If hardware arbitration is not active or is not used for a given package, only one package shall be selected at a time. To switch between packages, the Deselect Package command shall be issued to put the presently selected package into the deselected state before another package is selected.

A package shall also become selected if it receives any command that is directed to the package or to a channel within the package.

A package shall stay in the selected state until it receives a Deselect Package command, unless an internal condition causes all internal channels to enter the Initial State.

A package that is not using hardware arbitration may leave its output buffers enabled for the time that it is selected, or it may place its output buffers into the high-impedance state between transmitting packets through the NC-SI interface. (Temporarily placing the output buffers into the high-impedance state is not the same as entering the deselected state.)

For Type A integrated controllers: Because the bus buffers are separately controlled, a separate Select Package command needs to be sent to each Package ID in the controller that is to be enabled to transmit through the NC-SI interface. If the internal packages do not support hardware arbitration, only one package shall be selected at a time; otherwise, a bus conflict will occur.

For Type S single channel, and Types B and C integrated controllers: A single set of bus buffers exists for the package. Sending a Select Package command selects the entire package and enables all channels within the package to transmit through the NC-SI interface. (Whether a particular channel in a selected package starts transmitting Pass-through and AEN packets depends on whether that channel was enabled or disabled using the Enable or Disable Channel commands and whether the package may have had packets queued up for transmission.)

Table 22 illustrates the packet format of the Select Package command. Table 23 illustrates the disable byte for hardware arbitration.

**Table 22 – Select Package Command Packet Format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Reserved | | | Hardware Arbitration Disable |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

**Table 23 – Hardware Arbitration Disable Byte**

| Bits | Description |
|---|---|
| 7..1 | Reserved |
| 0 | `0b` = Hardware arbitration between packages is enabled.<br>`1b` = Disable hardware arbitration. Disabling hardware arbitration causes the package's arbitration logic to enter or remain in bypass mode.<br>In the case that the Network Controller does not support hardware arbitration, this bit is ignored; the Network Controller shall not return an error if the Select Package command can otherwise be successfully processed. |

### 8.4.6 Select Package Response (`0x81`)

Currently no command-specific reason code is identified for this response (see Table 24).

**Table 24 – Select Package Response Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.7 Deselect Package Command (`0x02`)

The Deselect Package command directs the controller package to stop transmitting packets through the NC-SI interface and to place the output buffers for the package into the high-impedance state.

The Deselect Package command is addressed to the package, rather than to a particular channel (that is, the command is sent with a Channel ID where the Package ID subfield matches the ID of the intended package and the Internal Channel ID subfield is set to `0x1F`).

The controller package enters the deselected state after it has transmitted the response to the Deselect Package command and placed its buffers into the high-impedance state. The controller shall place its outputs into the high-impedance state within the Package Deselect to Hi-Z Interval (T1). (This interval gives the controller being deselected time to turn off its electrical output buffers after sending the response to the Deselect Package command.)

It is recommended that a Network Controller should become deselected if it receives any command traffic directed to a different package ID as this suggests the Management Controller is attempting to communicate with another device.

If hardware arbitration is not supported or used, the Management Controller should wait for the Package Deselect to Hi-Z Interval (T1) to expire before selecting another controller.

For Type A integrated controllers: Because the bus buffers are separately controlled, putting the overall controller package into the high-impedance state requires sending separate Deselect Package commands to each Package ID in the overall package.

For Type S single channel, and Types B and C integrated controllers: A single set of bus buffers exists for the package. Sending a Deselect Package command deselects the entire NC-SI package and prevents all channels within the package from transmitting through the NC-SI interface.

Table 25 illustrates the packet format of the Deselect Package command.

**Table 25 – Deselect Package Command Packet Format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.8 Deselect Package Response (`0x82`)

The Network Controller shall always put the package into the deselected state after sending a Deselect Package Response.

No command-specific reason code is identified for this response (see Table 26).

**Table 26 – Deselect Package Response Packet Format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.9 Enable Channel Command (`0x03`)

The Enable Channel command allows the Management Controller to begin the flow of Network Controller packets, including Pass-through and AEN, through the NC-SI.

Table 27 illustrates the packet format of the Enable Channel command.

**Table 27 – Enable Channel Command Packet Format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

NOTE: It is currently unspecified whether the Enable Channel command by itself will cause the Network Controller to perform pass through from the Management Controller to the network, or if this can be enabled only by the Enable Channel Network TX command.

### 8.4.10 Enable Channel Response (0x83)

No command-specific reason code is identified for this response (see Table 28).

**Table 28 – Enable Channel Response Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.11 Disable Channel Command (0x04)

The Disable Channel command allows the Management Controller to disable the flow of packets, including Pass-through and AEN, to the Management Controller.

A Network Controller implementation is not required to flush pending packets from its RX Queues when a channel becomes disabled. If queuing is subsequently disabled for a channel, it is possible that a number of packets from the disabled channel could still be pending in the RX Queues. These packets may continue to be transmitted through the NC-SI interface until the RX Queues are emptied of those packets. The Management Controller should be aware that it may receive a number of packets from the channel before receiving the response to the Disable Channel command.

The 1-bit Allow Link Down (ALD) field can be used by the Management Controller to indicate that the specified channel will not be required to handle Pass-through traffic while disabled. The Network Controller is allowed to take down the external network physical link if no other functionality (for example, host OS or WoL [Wake-on-LAN]) is active.

Possible values for the 1-bit ALD field are as follows:

- `0b` = Keep link up for Pass-through management traffic
- `1b` = Allow link to be taken down

Table 29 illustrates the packet format of the Disable Channel command.

**Table 29 – Disable Channel Command Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Reserved | | | ALD |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

NOTE: It is currently unspecified whether this command will cause the Network Controller to cease the pass through of traffic from the Management Controller to the network, or if this can only be done using the Disable Channel Network TX command.

### 8.4.12 Disable Channel Response (`0x84`)

No command-specific reason code is identified for this response (see Table 30).

**Table 30 – Disable Channel Response Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.13 Reset Channel Command (`0x05`)

The Reset Channel command allows the Management Controller to put the channel into the Initial State. Packet transmission is not required to stop until the Reset Channel response has been sent. Thus, the Management Controller should be aware that it may receive a number of packets from the channel before receiving the response to the Reset Channel command.

Table 31 illustrates the packet format of the Reset Channel command.

**Table 31 – Reset Channel Command Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Reserved | | | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.14 Reset Channel Response (`0x85`)

Currently no command-specific reason code is identified for this response (see Table 32).

**Table 32 – Reset Channel Response Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.15 Enable Channel Network TX Command (`0x06`)

The Enable Channel Network TX command enables the channel to transmit Pass-through packets onto the network. After network transmission is enabled, this setting shall remain enabled until a Disable Channel Network TX command is received or the channel enters the Initial State.

The intention of this command is to control which Network Controller ports are allowed to transmit to the external network. The Network Controller compares the MAC address in outgoing Pass-through packets to the MAC addresses set using the Set MAC Address command. If a match exists, the packet is transmitted to the network.

Table 33 illustrates the packet format of the Enable Channel Network TX command.

**Table 33 – Enable Channel Network TX Command Packet Format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

NOTE: It is currently unspecified whether any dependencies exist between the Enable Channel and the Enable Channel Network TX commands, and whether the Enable Channel Network TX command, if sent before the Enable Channel command has been sent, should cause the Network Controller to immediately start forwarding pass thru packets received from the Management Controller to the network.

### 8.4.16 Enable Channel Network TX Response (`0x86`)

No command-specific reason code is identified for this response (see Table 34).

**Table 34 – Enable Channel Network TX Response Packet Format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.17 Disable Channel Network TX Command (`0x07`)

The Disable Channel Network TX command disables the channel from transmitting Pass-through packets onto the network. After network transmission is disabled, it shall remain disabled until an Enable Channel Network TX command is received.

Table 35 illustrates the packet format of the Disable Channel Network TX command.

**Table 35 – Disable Channel Network TX Command Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Checksum | | | |
| **20..23** | Pad | | | |

## 8.4.18 Disable Channel Network TX Response (`0x87`)

The NC-SI shall, in the absence of a checksum error or identifier mismatch, always accept the Disable Channel Network TX command and send a response.

Currently no command-specific reason code is identified for this response (see Table 36).

**Table 36 – Disable Channel Network TX Response Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

## 8.4.19 AEN Enable Command (`0x08`)

Network Controller implementations shall support this command on the condition that the Network Controller generates one or more standard AENs. The AEN Enable command enables and disables the different standard AENs supported by the Network Controller. The Network Controller shall copy the AEN MC ID field from the AEN Enable command into the MC ID field in every subsequent AEN sent to the Management Controller.

For more information, see 8.5 ("AEN Packet Formats") and 8.2.1.1 ("Management Controller ID").

Table 37 illustrates the packet format of the AEN Enable command.

**Table 37 – AEN Enable Command Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Reserved | | | AEN MC ID |
| **20..23** | AEN Control | | | |
| **24..27** | Checksum | | | |
| **28..45** | Pad | | | |

The AEN Control field has the format shown in Table 38.

**Table 38 – Format of AEN Control**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | Link Status Change AEN control | `0b` = Disable Link Status Change AEN<br>`1b` = Enable Link Status Change AEN |
| 1 | Configuration Required AEN control | `0b` = Disable Configuration Required AEN<br>`1b` = Enable Configuration Required AEN |
| 2 | Host NC Driver Status Change AEN control | `0b` = Disable Host NC Driver Status Change AEN<br>`1b` = Enable Host NC Driver Status Change AEN |
| 3..15 | Reserved | Reserved |
| 16..31 | OEM-specific AEN control | OEM-specific control |

### 8.4.20 AEN Enable Response (`0x88`)

Currently no command-specific reason code is identified for this response (see Table 39).

**Table 39 – AEN Enable Response Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.21 Set Link Command (`0x09`)

The Set Link command may be used by the Management Controller to configure the external network interface associated with the channel by using the provided settings. Upon receiving this command, the channel shall attempt to force the link to the configuration specified by the parameters. The channel shall send a response packet to the Management Controller within the required response time. However, the requested link state changes may take an unspecified amount of time to complete.

Table 40 illustrates the packet format of the Set Link command.

**Table 40 – Set Link Command Packet Format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Link Settings | | | |
| **20..23** | OEM Link Settings | | | |
| **24..27** | Checksum | | | |
| **28..45** | Pad | | | |

Table 41 and Table 42 describe the Set Link bit definitions. Refer to IEEE 802.3 for definitions of Auto Negotiation, Duplex Setting, Pause Capability, and Asymmetric Pause Capability.

**Table 41 – Set Link Bit Definitions**

| **Bit Position** | **Field Description** | **Value Description** |
|---|---|---|
| 00 | Auto Negotiation | `1b` = enable<br>`0b` = disable |
| 01..07 | Link Speed Selection<br>More than one speed can be selected when Auto Negotiation is set to 'enable'. If Auto Negotiation is not used, the channel attempts to force the link to the specified setting (in this case, if the setting is not supported or if multiple speeds are enabled, a Command Failed response code and Parameter Is Invalid, Unsupported, or Out-of-Range reason code shall be returned). | Bit 01: `1b` = enable 10 Mbps<br>Bit 02: `1b` = enable 100 Mbps<br>Bit 03: `1b` = enable 1000 Mbps (1 Gbps)<br>Bit 04: `1b` = enable 10 Gbps<br>Bits 05..07: RESERVED |
| 08..09 | Duplex Setting<br>(separate duplex setting bits)<br>More than one duplex setting can be selected when Auto Negotiation is set to 'enable'. If Auto Negotiation is not used, the channel attempts to force the link to the specified setting (in this case, if the setting is not supported or if multiple settings are enabled, a Command Failed response code and Parameter Is Invalid, Unsupported, or Out-of-Range reason code shall be returned). | Bit 08: `1b` = enable half-duplex<br>Bit 09: `1b` = enable full-duplex |
| 10 | Pause Capability | `1b` = disable<br>`0b` = enable |
| 11 | Asymmetric Pause Capability | `1b` = enable<br>`0b` = disable |
| 12 | OEM Link Settings Field Valid (see Table 42) | `1b` = enable<br>`0b` = disable |
| 13..31 | Reserved | 0 |

**Table 42 – OEM Set Link Bit Definitions**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 00..31 | OEM Link Settings | Vendor specified |

### 8.4.22 Set Link Response (0x89)

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Set Link command and send a response (see Table 43).

**Table 43 – Set Link Response Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

Table 44 describes the reason codes that are specific to the Set Link command. Returning the following command-specific codes is recommended, conditional upon Network Controller support for the related capabilities.

**Table 44 – Set Link Command-Specific Reason Codes**

| Value | Description | Comment |
|---|---|---|
| 0x1 | Set Link Host OS/ Driver Conflict | Returned when the Set Link command is received when the Host NC driver is operational |
| 0x2 | Set Link Media Conflict | Returned when Set Link command parameters conflict with the media type (for example, Fiber Media) |
| 0x3 | Set Link Parameter Conflict | Returned when Set Link parameters conflict with each other (for example, 1000 Mbps HD with copper media) |
| 0x4 | Set Link Power Mode Conflict | Returned when Set Link parameters conflict with current low-power levels by exceeding capability |
| 0x5 | Set Link Speed Conflict | Returned when Set Link parameters attempt to force more than one speed at the same time |
| 0x6 | Link Command Failed-Hardware Access Error | Returned when PHY R/W access fails to complete normally while executing the Set Link or Get Link Status command |

### 8.4.23 Get Link Status Command (0x0A)

The Get Link Status command allows the Management Controller to query the channel for potential link status and error conditions (see Table 45).

**Table 45 – Get Link Status Command Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.24 Get Link Status Response (0x8A)

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Get Link Status command and send a response (see Table 46).

**Table 46 – Get Link Status Response Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Link Status | | | |
| **24..27** | Other Indications | | | |
| **28..31** | OEM Link Status | | | |
| **32..35** | Checksum | | | |
| **36..45** | Pad | | | |

Table 47 describes the Link Status bit definitions.

**Table 47 – Link Status Field Bit Definitions**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 00 | Link Flag | 0b = Link is down<br>1b = Link is up<br>This field is mandatory. |

| Bit Position | Field Description | Value Description |
|---|---|---|
| 01..04 | Speed and duplex | 0x0 = Auto-negotiate not complete [per IEEE 802.3], SerDes Flag = 1b, or no Highest Common Denominator (HCD) from the following options (0x1 through 0x8) was found.<br>0x1 = 10BASE-T half-duplex<br>0x2 = 10BASE-T full-duplex<br>0x3 = 100BASE-TX half-duplex<br>0x4 = 100BASE-T4<br>0x5 = 100BASE-TX full-duplex<br>0x6 = 1000BASE-T half-duplex<br>0x7 = 1000BASE-T full-duplex<br>0x8 = 10G-BASE-T support<br>0x9 – 0xf = RESERVED<br>Except when SerDes = 1b, the value may reflect forced link setting. |
| 05 | Auto Negotiate Flag | 1b = Auto-negotiation is enabled.<br>This field always returns 0b if auto-negotiation is not supported.<br>This field is mandatory if supported by the controller. |
| 06 | Auto Negotiate Complete | 1b = Auto-negotiation has completed.<br>This includes if auto-negotiation was completed using Parallel Detection. Always returns 0b if auto-negotiation is not supported or is not enabled.<br>This field is mandatory if the Auto Negotiate Flag is supported. |
| 07 | Parallel Detection Flag | 1b = Link partner did not support auto-negotiation and parallel detection was used to get link.<br>This field contains 0b if Parallel Detection was not used to obtain link. |
| 08 | Reserved | None |
| 09 | Link Partner Advertised Speed and Duplex 1000TFD | 1b = Link Partner is 1000BASE-T full-duplex capable.<br>Valid when:<br>SerDes Flag = 0b<br>Auto-Negotiate Flag = 1b<br>Auto-Negotiate Complete = 1b<br>This field is mandatory. |
| 10 | Link Partner Advertised Speed and Duplex 1000THD | 1b = Link Partner is 1000BASE-T half-duplex capable.<br>Valid when:<br>SerDes Flag = 0b<br>Auto-Negotiate Flag = 1b<br>Auto-Negotiate Complete = 1b<br>This field is mandatory. |

| Bit Position | Field Description | Value Description |
| --- | --- | --- |
| 11 | Link Partner Advertised Speed 100T4 | 1b = Link Partner is 100BASE-T4 capable.<br>Valid when:<br>SerDes Flag = 0b<br>Auto-Negotiate Flag = 1b<br>Auto-Negotiate Complete = 1b<br>This field is mandatory. |
| 12 | Link Partner Advertised Speed and Duplex 100TXFD | 1b = Link Partner is 100BASE-TX full-duplex capable.<br>Valid when:<br>SerDes Flag = 0b<br>Auto-Negotiate Flag = 1b<br>Auto-Negotiate Complete = 1b<br>This field is mandatory. |
| 13 | Link Partner Advertised Speed and Duplex 100TXHD | 1b = Link Partner is 100BASE-TX half-duplex capable.<br>Valid when:<br>SerDes Flag = 0b<br>Auto-Negotiate Flag = 1b<br>Auto-Negotiate Complete = 1b<br>This field is mandatory. |
| 14 | Link Partner Advertised Speed and Duplex 10TFD | 1b = Link Partner is 10BASE-T full-duplex capable.<br>Valid when:<br>SerDes Flag = 0b<br>Auto-Negotiate Flag = 1b<br>Auto-Negotiate Complete = 1b<br>This field is mandatory. |
| 15 | Link Partner Advertised Speed and Duplex 10THD | 1b = Link Partner is 10BASE-T half-duplex capable.<br>Valid when:<br>SerDes Flag = 0b<br>Auto-Negotiate Flag = 1b<br>Auto-Negotiate Complete = 1b<br>This field is mandatory. |
| 16 | TX Flow Control Flag | 1b = Pause Flow Control enabled for TX on the external network interface.<br>This field is mandatory. |
| 17 | RX Flow Control Flag | 1b = Pause Flow Control enabled for RX on the external network interface.<br>This field is mandatory. |

| Bit Position | Field Description | Value Description |
|---|---|---|
| 18..19 | Link Partner Advertised Flow Control | `00b` = Link partner is not pause capable.<br>`01b` = Link partner supports symmetric pause.<br>`10b` = Link partner supports asymmetric pause toward link partner.<br>`11b` = Link partner supports both symmetric and asymmetric pause.<br>Valid when:<br>SerDes Flag = `0b`<br>Auto-Negotiate = `1b`<br>Auto-Negotiate Complete = `1b`<br>This field is mandatory. |
| 20 | SerDes Link | SerDes status (see 4.18")<br>`0b` = SerDes not used<br>`1b` = SerDes used<br>This field is mandatory. |
| 21 | OEM Link Speed Valid | `0b` = OEM link settings are invalid.<br>`1b` = OEM link settings are valid. |
| 22..31 | Reserved | 0 |

1761 Table 48 describes the Other Indications field bit definitions.

1762 **Table 48 – Other Indications Field Bit Definitions**

| Bits | Description | Values |
|---|---|---|
| 00 | Host NC Driver Status Indication | `0b` = The Network Controller driver for the host external network interface associated with this channel is not operational (not running).<br>`1b` = The Network Controller driver for the host external network interface associated with this channel is being reported as operational (running).<br>This field is optional, and always returns `0b` if the option is not supported. |
| 01..31 | Reserved | None |

1763 Table 49 describes the OEM Link Status field bit definitions.

1764 **Table 49 – OEM Link Status Field Bit Definitions (Optional)**

| Bits | Description | Values |
|---|---|---|
| 00..31 | OEM Link Status | OEM specific |

Table 50 describes the reason code that is specific to the Get Link Status command.

**Table 50 – Get Link Status Command-Specific Reason Code**

| Value | Description | Comment |
|---|---|---|
| `0x6` | Link Command Failed-Hardware Access Error | Returned when PHY R/W access fails to complete normally while executing the Set Link or Get Link Status command |

### 8.4.25 Set VLAN Filter Command (`0x0B`)

The Set VLAN Filter command is used by the Management Controller to program one or more VLAN IDs that are used for VLAN filtering.

Incoming packets that match both a VLAN ID filter and a MAC address filter are forwarded to the Management Controller. Other packets may be dropped based on the VLAN filtering mode per the Enable VLAN command.

The quantity of each filter type that is supported by the channel can be discovered by means of the Get Capabilities command. Up to 15 filters can be supported per channel. A Network Controller implementation shall support at least one VLAN filter per channel.

To configure a VLAN filter, the Management Controller issues a Set VLAN Filter command with the Filter Selector field indicating which filter is to be configured, the VLAN ID field set to the VLAN TAG values to be used by the filter, and the Enable field set to either enable or disable the selected filter.

The VLAN-related fields are specified per IEEE 802.1q. When VLAN Tagging is used, the packet includes a Tag Protocol Identifier (TPID) field and VLAN Tag fields, as shown in Table 51.

**Table 51 – IEEE 802.1q VLAN Fields**

| Field | Size | Description |
|---|---|---|
| TPI | 2 bytes | Tag Protocol Identifier<br>= `8100h` |
| VLAN TAG – user priority | 3 bits | User Priority<br>(typical value = `000b`) |
| VLAN TAG – CFI | 1 bit | Canonical Format Indicator = `0b` |
| VLAN TAG – VLAN ID | 12 bits | Zeros = no VLAN |

When checking VLAN field values, the Network Controller shall match against the enabled VLAN Tag Filter values that were configured with the Set VLAN Filter command. The Network Controller shall also match on the TPI value of 8100h, as specified by IEEE 802.1q. Matching against the User Priority/CFI bits is optional. An implementation may elect to ignore the setting of those fields.

Table 52 illustrates the packet format of the Set VLAN Filter command.

**Table 52 – Set VLAN Filter Command Packet Format**

<table>
<tr><th></th><th colspan="5">Bits</th></tr>
<tr><th>Bytes</th><th>31..24</th><th>23..16</th><th>15..08</th><th colspan="2">07..00</th></tr>
<tr><td>00..15</td><td colspan="5">NC-SI Header</td></tr>
<tr><td>16..19</td><td colspan="2">Reserved</td><td>User Priority/CFI</td><td colspan="2">VLAN ID</td></tr>
<tr><td>20..23</td><td colspan="2">Reserved</td><td>Filter Selector</td><td>Reserved</td><td>E</td></tr>
<tr><td>24..27</td><td colspan="5">Checksum</td></tr>
<tr><td>28..45</td><td colspan="5">Pad</td></tr>
</table>

Table 53 provides possible settings for the Filter Selector field. Table 54 provides possible settings for the Enable (E) field.

**Table 53 – Possible Settings for Filter Selector Field (8-Bit Field)**

| Value | Description |
|---|---|
| 1 | Settings for VLAN filter number 1 |
| 2 | Settings for VLAN filter number 2 |
| .. | |
| N | Settings for VLAN filter number *N* |

**Table 54 – Possible Settings for Enable (E) Field (1-Bit Field)**

| Value | Description |
|---|---|
| 0b | Disable this VLAN filter |
| 1b | Enable this VLAN filter |

### 8.4.26 Set VLAN Filter Response (0x8B)

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Set VLAN Filter command and send a response (see Table 55).

**Table 55 – Set VLAN Filter Response Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

Table 56 describes the reason code that is specific to the Set VLAN Filter command.

**Table 56 – Set VLAN Filter Command-Specific Reason Code**

| Value | Description | Comment |
|---|---|---|
| 0x7 | VLAN Tag Is Invalid | Returned when the VLAN ID is invalid (VLAN ID = 0) |

### 8.4.27 Enable VLAN Command (0x0C)

The Enable VLAN command may be used by the Management Controller to enable the channel to accept VLAN-tagged packets from the network for NC-SI Pass-through operation (see Table 57).

**Table 57 – Enable VLAN Command Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Reserved | | | Mode # |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

Table 58 describes the modes for the Enable VLAN command.

**Table 58 – VLAN Enable Modes**

| Mode | # | O/M | Description |
|---|---|---|---|
| Reserved | 0x00 | N/A | Reserved |
| VLAN only | 0x01 | M | Only VLAN-tagged packets that match the enabled VLAN Filter settings (and also match the MAC Address Filtering configuration) are accepted.<br>Non-VLAN-tagged packets are not accepted. |
| VLAN + non-VLAN | 0x02 | O | VLAN-tagged packets that match the enabled VLAN Filter settings (and also match the MAC Address Filtering configuration) are accepted.<br>Non-VLAN-tagged packets (that also match the MAC Address Filtering configuration) are also accepted. |
| Any VLAN + non-VLAN | 0x03 | O | Any VLAN-tagged packets that also match the MAC Address Filtering configuration are accepted, regardless of the VLAN Filter settings.<br>Non-VLAN-tagged packets (that also match the MAC Address Filtering configuration) are also accepted. |
| Reserved | 0x04 - 0xFF | N/A | Reserved |

### 8.4.28 Enable VLAN Response (`0x8C`)

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Enable VLAN command and send a response.

Currently no command-specific reason code is identified for this response (see Table 59).

**Table 59 – Enable VLAN Response Packet Format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.29 Disable VLAN Command (`0x0D`)

The Disable VLAN command may be used by the Management Controller to disable VLAN filtering. In the disabled state, only non-VLAN-tagged packets (that also match the MAC Address Filtering configuration) are accepted. VLAN-tagged packets are not accepted.

Table 60 illustrates the packet format of the Disable VLAN command.

**Table 60 – Disable VLAN Command Packet Format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.30 Disable VLAN Response (`0x8D`)

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Disable VLAN command and send a response.

Currently no command-specific reason code is identified for this response (see Table 61).

**Table 61 – Disable VLAN Response Packet Format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.31 Set MAC Address Command (`0x0E`)

The Set MAC Address command is used by the Management Controller to program the channel's unicast or multicast MAC address filters.

The channel supports one or more "perfect match" MAC address filters that are used to selectively forward inbound frames to the Management Controller. Assuming that a packet passes any VLAN filtering that may be active, it will be forwarded to the Management Controller if its 48-bit destination MAC address exactly matches an active MAC address filter.

MAC address filters may be configured as unicast or multicast addresses, depending on the capability of the channel. The channel may implement three distinct types of filter:

- **Unicast filters** support exact matching on 48-bit unicast MAC addresses.
- **Multicast filters** support exact matching on 48-bit multicast MAC addresses.
- **Mixed filters** support exact matching on both unicast and multicast MAC addresses.

The number of each type of filter that is supported by the channel can be discovered by means of the Get Capabilities command. The channel shall support at least one unicast address filter or one mixed filter, so that at least one unicast MAC address filter may be configured on the channel. Support for any combination of unicast, multicast, or mixed filters beyond this basic requirement is vendor specific. The total number of all filters shall be less than or equal to 8.

To configure an address filter, the Management Controller issues a Set MAC Address command with the Address Type field indicating the type of address to be programmed (unicast or multicast) and the MAC Address Num field indicating the specific filter to be programmed.

Filters are addressed using a 1-based index ordered over the unicast, multicast, and mixed filters reported by means of the Get Capabilities command. For example, if the interface reports four unicast filters, two multicast filters, and two mixed filters, then MAC Address numbers 1 through 4 refer to the interface's unicast filters, 5 and 6 refer to the multicast filters, and 7 and 8 refer to the mixed filters. Similarly, if the interface reports two unicast filters, no multicast filters, and six mixed filters, then MAC address numbers 1 and 2 refer to the unicast filters, and 3 through 8 refer to the mixed filters.

The filter type of the filter to be programmed (unicast, multicast, or mixed) shall be compatible with the Address Type being programmed. For example, programming a mixed filter to a unicast address is allowed, but programming a multicast filter to a unicast address is an error.

The Enable field determines whether the indicated filter is to be enabled or disabled. When a filter is programmed to be enabled, the filter is loaded with the 48-bit MAC address in the MAC Address field of the command, and the channel enables forwarding of frames that match the configured address. If the specified filter was already enabled, it is updated with the new address provided.

When a filter is programmed to be disabled, the contents of the MAC Address field are ignored. Any previous MAC address programmed in the filter is discarded and the channel no longer uses this filter in its packet-forwarding function.

Table 62 illustrates the packet format of the Set MAC Address command.

**Table 62 – Set MAC Address Command Packet Format**

<table>
<tr><th></th><th colspan="6">Bits</th></tr>
<tr><th>Bytes</th><th>31..24</th><th>23..16</th><th>15..08</th><th colspan="3">07..00</th></tr>
<tr><td>00..15</td><td colspan="6">NC-SI Header</td></tr>
<tr><td>16..19</td><td>MAC Address byte 5</td><td>MAC Address byte 4</td><td>MAC Address byte 3</td><td colspan="3">MAC Address byte 2</td></tr>
<tr><td>20..23</td><td>MAC Address byte 1</td><td>MAC Address byte 0</td><td>MAC Address Num</td><td>AT</td><td>Rsvd</td><td>E</td></tr>
<tr><td>24..27</td><td colspan="6">Checksum</td></tr>
<tr><td>28..45</td><td colspan="6">Pad</td></tr>
<tr><td colspan="7">NOTE: AT = Address Type, E = Enable.</td></tr>
</table>

Table 63 provides possible settings for the MAC Address Number field. Table 64 provides possible settings for the Address Type (AT) field. Table 65 provides possible settings for the Enable (E) field.

**Table 63 – Possible Settings for MAC Address Number (8-Bit Field)**

| Value | Description |
|---|---|
| `0x01` | Configure MAC address filter number 1 |
| `0x02` | Configure MAC address filter number 2 |
| .. | |
| N | Configure MAC address filter number *N* |

**Table 64 – Possible Settings for Address Type (3-Bit Field)**

| Value | Description |
|---|---|
| `0x0` | Unicast MAC address |
| `0x1` | Multicast MAC address |
| `0x2–0x7` | Reserved |

**Table 65 – Possible Settings for Enable Field (1-Bit Field)**

| Value | Description |
|---|---|
| `0b` | Disable this MAC address filter |
| `1b` | Enable this MAC address filter |

### 8.4.32 Set MAC Address Response (`0x8E`)

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Set MAC Address command and send a response (see Table 66).

**Table 66 – Set MAC Address Response Packet Format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

Table 67 describes the reason code that is specific to the Set MAC Address command.

**Table 67 – Set MAC Address Command-Specific Reason Code**

| Value | Description | Comment |
|---|---|---|
| `0x8` | MAC Address Is Zero | Returned when the Set MAC Address command is received with the MAC address set to 0 |

### 8.4.33 Enable Broadcast Filter Command (`0x10`)

The Enable Broadcast Filter command allows the Management Controller to control the forwarding of broadcast frames to the Management Controller. The channel, upon receiving and processing this command, shall filter all received broadcast frames based on the broadcast packet filtering settings specified in the payload. If no broadcast packet types are specified for forwarding, all broadcast packets shall be filtered out.

The Broadcast Packet Filter Settings field is used to specify those protocol-specific broadcast filters that should be activated. The channel indicates which broadcast filters it supports in the Broadcast Filter Capabilities field of the Get Capabilities Response frame defined in 8.4.46.

Table 68 illustrates the packet format of the Enable Broadcast Filter command.

**Table 68 – Enable Broadcast Filter Command Packet Format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Broadcast Packet Filter Settings | | | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

1880 Table 69 describes the Broadcast Packet Filter Settings field bit definitions.

1881 **Table 69 – Broadcast Packet Filter Settings Field**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | ARP Packets | `1b` = Forward this packet type to the Management Controller.<br>`0b` = Filter out this packet type.<br>For the purposes of this specification, an ARP broadcast packet is defined to be any packet that meets all of the following requirements:<br>• The destination MAC address field is set to the layer 2 broadcast address (`FF:FF:FF:FF:FF:FF`).<br>• The EtherType field set to `0x0806`.<br>This field is mandatory. |
| 1 | DHCP Client Packets | `1b` = Forward this packet type to the Management Controller.<br>`0b` = Filter out this packet type.<br>For the purposes of this filter, a DHCP client broadcast packet is defined to be any packet that meets all of the following requirements:<br>• The destination MAC address field is set to the layer 2 broadcast address (`FF:FF:FF:FF:FF:FF`).<br>• The EtherType field is set to `0x0800` (IPv4).<br>• The IP header's Protocol field is set to 17 (UDP).<br>• The UDP destination port number is set to 68.<br>This field is optional. If unsupported, broadcast DHCP client packets will be blocked when broadcast filtering is enabled. The value shall be set to 0 if unsupported. |
| 2 | DHCP Server Packets | `1b` = Forward this packet type to the Management Controller.<br>`0b` = Filter out this packet type.<br>For the purposes of this filter, a DHCP server broadcast packet is defined to be any packet that meets all of the following requirements:<br>• The destination MAC address field is set to the layer 2 broadcast address (`FF:FF:FF:FF:FF:FF`).<br>• The EtherType field is set to `0x0800` (IPv4).<br>• The IP header's Protocol field is set to 17 (UDP).<br>• The UDP destination port number is set to 67.<br>This field is optional. If unsupported, broadcast DHCP packets will be blocked when broadcast filtering is enabled. The value shall be set to 0 if unsupported. |

| Bit Position | Field Description | Value Description |
|---|---|---|
| 3 | NetBIOS Packets | `1b` = Forward this packet type to the Management Controller.<br>`0b` = Filter out this packet type.<br>For the purposes of this filter, NetBIOS broadcast packets are defined to be any packet that meets all of the following requirements:<br>• The destination MAC address field is set to the layer 2 broadcast address (`FF:FF:FF:FF:FF:FF`).<br>• The EtherType field is set to `0x0800` (IPv4).<br>• The IP header's Protocol field is set to 17 (UDP).<br>• The UDP destination port number is set to 137 for NetBIOS Name Service or 138 for NetBIOS Datagram Service, per the assignment of IANA well-known ports.<br>This field is optional. If unsupported, broadcast NetBIOS packets will be blocked when broadcast filtering is enabled. The value shall be set to 0 if unsupported. |
| 4..31 | Reserved | None |

### 8.4.34 Enable Broadcast Filter Response (0x90)

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Enable Broadcast Filter command and send a response.

Currently no command-specific reason code is identified for this response (see Table 70).

**Table 70 – Enable Broadcast Filter Response Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.35 Disable Broadcast Filter Command (0x11)

The Disable Broadcast Filter command may be used by the Management Controller to disable the broadcast filter feature and enable the reception of all broadcast frames. Upon processing this command, the channel shall discontinue the filtering of received broadcast frames.

Table 71 illustrates the packet format of the Disable Broadcast Filter command.

**Table 71 – Disable Broadcast Filter Command Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.36 Disable Broadcast Filter Response (`0x91`)

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Disable Broadcast Filter command and send a response.

Currently no command-specific reason code is identified for this response (see Table 72).

**Table 72 – Disable Broadcast Filter Response Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.37 Enable Global Multicast Filter Command (`0x12`)

The Enable Global Multicast Filter command is used to activate global filtering of multicast frames with optional filtering of specific multicast protocols. Upon receiving and processing this command, the channel shall deliver to the Management Controller only multicast frames that match protocol-specific multicast filters enabled using this command or specific, multicast addresses that have been configured and enabled using the Set MAC Address command.

The Multicast Packet Filter Settings field is used to specify optional, protocol-specific multicast filters that should be activated. The channel indicates which optional multicast filters it supports in the Multicast Filter Capabilities field of the Get Capabilities Response frame defined in 8.4.46. The Management Controller should not set bits in the Multicast Packet Filter Settings field that are not indicated as supported in the Multicast Filter Capabilities field.

IPv6 Neighbor Solicitation messages are not covered by the currently defined multicast filters. When multicast, Neighbor Solicitation messages are sent to a Solicited Node multicast address that is derived from the target node's IPv6 address. To enable forwarding of Solicited Node multicasts when global multicast filtering is active, the Management Controller would configure a multicast or mixed MAC address filter for the specific Solicited Node multicast address required, using the Set MAC Address command.

This command shall be implemented if the channel implementation supports accepting all multicast addresses. An implementation that does not support accepting all multicast addresses shall not implement these commands. Pass-through packets with multicast addresses can still be accepted depending on multicast address filter support provided by the Set MAC Address command. Multicast filter entries that are set to enabled in the Set MAC Address command are accepted; all others are rejected.

Table 73 illustrates the packet format of the Enable Global Multicast Filter command.

**Table 73 – Enable Global Multicast Filter Command Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Multicast Packet Filter Settings | | | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

Table 74 describes the bit definitions for the Multicast Packet Filter Settings field.

**Table 74 – Bit Definitions for Multicast Packet Filter Settings Field**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | IPv6 Neighbor Advertisement | `1b` = Forward this packet type to the Management Controller.<br>`0b` = Filter out this packet type.<br>For the purposes of this specification, an IPv6 Neighbor Advertisement multicast packet is defined to be any packet that meets all of the following requirements:<br>• The destination MAC address field is set to a layer 2 multicast address of the form `33:33:00:00:00:01`. This address corresponds to the All_Nodes (`FF02::1`) multicast address.<br>• The EtherType field is set to `0x86DD` (IPv6).<br>• The IPv6 header's Next Header field is set to 58 (ICMPv6).<br>• The ICMPv6 header's Message Type field is set to the following value: 136 – Neighbor Advertisement.<br>This field is optional. |
| 1 | IPv6 Router Advertisement | `1b` = Forward this packet type to the Management Controller.<br>`0b` = Filter out this packet type.<br>For the purposes of this specification, an IPv6 Router Advertisement multicast packet is defined to be any packet that meets all of the following requirements:<br>• The destination MAC address field is set to a layer 2 multicast address of the form `33:33:00:00:00:01`. This corresponds to the All_Nodes multicast address, `FF02::1`.<br>• The EtherType field is set to `0x86DD` (IPv6).<br>• The IPv6 header's Next Header field is set to 58 (ICMPv6).<br>• The ICMPv6 header's Message Type field is set to 134.<br>This field is optional. |

| Bit Position | Field Description | Value Description |
|---|---|---|
| 2 | DHCPv6 relay and server multicast | `1b` = Forward this packet type to the Management Controller.<br>`0b` = Filter out this packet type.<br>For the purposes of this filter, a DHCPv6 multicast packet is defined to be any packet that meets all of the following requirements:<br>• The destination MAC address field is set to the layer 2 multicast address `33:33:00:01:00:02` or `33:33:00:01:00:03`. These correspond to the IPv6 multicast addresses `FF02::1:2` (All_DHCP_Relay_Agents_and_Servers) and `FF05::1:3` (All_DHCP_Servers).<br>• The EtherType field is set to `0x86DD` (IPv6).<br>• The IPv6 header's Next Header field is set to 17 (UDP).<br>• The UDP destination port number is set to 547.<br>This field is optional. If unsupported, multicast DHCP packets will be blocked when multicast filtering is enabled, unless they are matched by an address filter configured using the Set MAC Address command. The value shall be set to 0 if unsupported. |
| 3..31 | Reserved | None |

### 8.4.38 Enable Global Multicast Filter Response (`0x92`)

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Enable Global Multicast Filter command and send a response.

Currently no command-specific reason code is identified for this response (see Table 75).

**Table 75 – Enable Global Multicast Filter Response Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.39 Disable Global Multicast Filter Command (`0x13`)

The Disable Global Multicast Filter command is used to disable global filtering of multicast frames. Upon receiving and processing this command, and regardless of the current state of multicast filtering, the channel shall forward all multicast frames to the Management Controller.

This command shall be implemented on the condition that the channel implementation supports accepting all multicast addresses. An implementation that does not support accepting all multicast addresses shall not implement these commands. Pass-through packets with multicast addresses can still be accepted depending on multicast address filter support provided by the Set MAC Address command. Packets with destination addresses matching multicast filter entries that are set to enabled in the Set MAC Address command are accepted; all others are rejected.

Table 76 illustrates the packet format of the Disable Global Multicast Filter command.

**Table 76 – Disable Global Multicast Filter Command Packet Format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

## 8.4.40 Disable Global Multicast Filter Response (0x93)

In the absence of any errors, the channel shall process and respond to the Disable Global Multicast Filter command by sending the response packet shown in Table 77.

Currently no command-specific reason code is identified for this response.

**Table 77 – Disable Global Multicast Filter Response Packet Format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

## 8.4.41 Set NC-SI Flow Control Command (0x14)

The Set NC-SI Flow Control command allows the Management Controller to configure IEEE 802.3 pause packet flow control on the NC-SI.

The Set NC-SI Flow Control command is addressed to the package, rather than to a particular channel (that is, the command is sent with a Channel ID where the Package ID subfield matches the ID of the intended package and the Internal Channel ID subfield is set to 0x1F).

When enabled for flow control, a channel may direct the package to generate and renew 802.3x (XOFF) PAUSE Frames for a maximum interval of T12 for a single congestion condition. If the congestion condition remains in place after a second T12 interval expires, the congested channel shall enter the Initial State and remove its XOFF request to the package. Note that some implementations may have shared buffering arrangements where all channels within the package become congested simultaneously. Also note that if channels become congested independently, the package may not immediately go into the XON state after T12 if other channels within the package are still requesting XOFF.

Table 78 illustrates the packet format of the Set NC-SI Flow Control command.

**Table 78 – Set NC-SI Flow Control Command Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Reserved | | | Flow Control Enable |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

Table 79 describes the values for the Flow Control Enable field.

**Table 79 – Values for the Flow Control Enable Field (8-Bit Field)**

| Value | Description |
|---|---|
| `0x0` | Disables NC-SI flow control |
| `0x1` | Enables Network Controller to Management Controller flow control frames (Network Controller generates flow control frames)<br>This field is optional. |
| `0x2` | Enables Management Controller to Network Controller flow control frames (Network Controller accepts flow control frames)<br>This field is optional. |
| `0x3` | Enables bi-directional flow control frames<br>This field is optional. |
| `0x4..0xFF` | Reserved |

### 8.4.42 Set NC-SI Flow Control Response (`0x94`)

The package shall, in the absence of a checksum error or identifier mismatch, always accept the Set NC-SI Flow Control command and send a response (see Table 80).

**Table 80 – Set NC-SI Flow Control Response Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

Table 81 describes the reason code that is specific to the Set NC-SI Flow Control command.

**Table 81 – Set NC-SI Flow Control Command-Specific Reason Code**

| Value | Description | Comment |
|---|---|---|
| `0x9` | Independent transmit and receive enable/disable control is not supported | Returned when the implementation requires that both transmit and receive flow control be enabled and disabled simultaneously |

### 8.4.43 Get Version ID Command (`0x15`)

The Get Version ID command may be used by the Management Controller to request the channel to provide the controller and firmware type and version strings listed in the response payload description.

Table 82 illustrates the packet format of the Get Version ID command.

**Table 82 – Get Version ID Command Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.44 Get Version ID Response (`0x95`)

The channel shall, in the absence of an error, always accept the Get Version ID command and send the response packet shown in Table 83. Currently no command-specific reason code is identified for this response.

**Table 83 – Get Version ID Response Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | NC-SI Version | | | |
| | Major | Minor | Update | Alpha1 |
| **24..27** | reserved | reserved | reserved | Alpha2 |
| **28..31** | Firmware Name String (11-08) | | | |
| **32..35** | Firmware Name String (07-04) | | | |
| **36..39** | Firmware Name String (03-00) | | | |
| **40..43** | Firmware Version | | | |
| | MS-byte (3) | Byte (2) | Byte (1) | LS-byte (0) |

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **44..47** | PCI DID | | PCI VID | |
| **48..51** | PCI SSID | | PCI SVID | |
| **52..55** | Manufacturer ID (IANA) | | | |
| **56..59** | Checksum | | | |

#### 8.4.44.1 NC-SI Version Encoding

The NC-SI Version field holds the version number of the NC-SI specification with which the controller is compatible. The version field shall be encoded as follows:

- The 'major', 'minor', and 'update' bytes are BCD-encoded, and each byte holds two BCD digits.
- The 'alpha' byte holds an optional alphanumeric character extension that is encoded using the ISO/IEC 8859-1 Character Set.
- The semantics of these fields follow the semantics specified in DSP4004.
- The value `0x00` in the Alpha1 or Alpha2 fields means that the corresponding alpha field is not used. The Alpha1 field shall be used first.
- The value `0xF` in the most-significant nibble of a BCD-encoded value indicates that the most-significant nibble should be ignored and the overall field treated as a single digit value.
- A value of `0xFF` in the update field indicates that the entire field is not present. `0xFF` is not allowed as a value for the major or minor fields.

EXAMPLE: Version 3.7.10a → `0xF3F7104100`
Version 10.01.7 → `0x1001F70000`
Version 3.1 → `0xF3F1FF0000`
Version 1.0a → `0xF1F0FF4100`
Version 1.0ab → `0xF1F0FF4142 (Alpha1 = 0x41, Alpha2 = 0x42)`

#### 8.4.44.2 Firmware Name Encoding

The Firmware Name String shall be encoded using the ISO/IEC 8859-1 Character Set. Strings are left-justified where the leftmost character of the string occupies the most-significant byte position of the Firmware Name String field, and characters are populated starting from that byte position. The string is null terminated if the string is smaller than the field size. That is, the delimiter value, `0x00`, follows the last character of the string if the string occupies fewer bytes than the size of the field allows. A delimiter is not required if the string occupies the full size of the field. Bytes following the delimiter (if any) should be ignored and can be any value.

#### 8.4.44.3 Firmware Version Encoding

To facilitate a common way of representing and displaying firmware version numbers across different vendors, each byte is hexadecimal encoded where each byte in the field holds two hexadecimal digits. The Firmware Version field shall be encoded as follows. The bytes are collected into a single 32-bit field where each byte represents a different 'point number' of the overall version. The selection of values that represent a particular version of firmware is specific to the Network Controller vendor.

Software displaying these numbers should not suppress leading zeros, which should help avoid user confusion in interpreting the numbers. For example, consider the two values `0x05` and `0x31`. Numerically, the byte `0x31` is greater that `0x05`, but if leading zeros were suppressed, the two displayed values would be "0.5" and "0.31", respectively, and a user would generally interpret 0.5 as representing a greater value than 0.31. Similarly, if leading zeros were suppressed, the value `0x01` and `0x10` would be displayed as 0.1 and 0.10, which could potentially be misinterpreted as representing the same version.

EXAMPLE: `0x00030217` → Version 00.03.02.17
`0x010100A0` → Version 01.01.00.A0

#### 8.4.44.4 PCI ID Fields

These fields (PCI DID, PCI VID, PCI SSID, PCI SVID) hold the PCI ID information for the Network Controller when the Network Controller incorporates a PCI or PCI Express™ interface that provides a host network interface connection that is shared with the NC-SI connection to the network.

If this field is not used, the values shall all be set to zeros (`0000h`). Otherwise, the fields shall hold the PCI ID information for the host interface as defined by the version of the PCI/PCI Express™ specification to which the device's interface was designed.

#### 8.4.44.5 Manufacturer ID (IANA) Field

The Manufacturer ID holds the IANA Enterprise Number for the manufacturer of the Network Controller as a 32-bit binary number. If the field is unused, the value shall be set to `0xFFFFFFFF`.

### 8.4.45 Get Capabilities Command (0x16)

The Get Capabilities command is used to discover additional optional functions supported by the channel, such as the number of unicast/multicast addresses supported, the amount of buffering in bytes available for packets bound for the Management Controller, and so on.

Table 84 illustrates the packet format for the Get Capabilities command.

**Table 84 – Get Capabilities Command Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.46 Get Capabilities Response (0x96)

In the absence of any errors, the channel shall process and respond to the Get Capabilities Command and send the response packet shown in Table 85. Currently no command-specific reason code is identified for this response.

**Table 85 – Get Capabilities Response Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Capabilities Flags | | | |
| **24..27** | Broadcast Packet Filter Capabilities | | | |
| **28..31** | Multicast Packet Filter Capabilities | | | |
| **32..35** | Buffering Capability | | | |
| **36..39** | AEN Control Support | | | |
| **40..43** | VLAN Filter Count | Mixed Filter Count | Multicast Filter Count | Unicast Filter Count |
| **44..47** | Reserved | | VLAN Mode Support | Channel Count |
| **48..51** | Checksum | | | |

#### 8.4.46.1 Capabilities Flags Field

The Capabilities Flags field indicates which optional features of this specification the channel supports, as described in Table 86.

**Table 86 – Capabilities Flags Bit Definitions**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | Hardware Arbitration | `0b` = Hardware arbitration is not supported by the package.<br>`1b` = Hardware arbitration is supported by the package. |
| 1 | OS Presence | `0b` = OS Presence status is not supported.<br>`1b` = OS Presence status is supported. |
| 2 | Network Controller to Management Controller Flow Control Support | `0b` = Network Controller to Management Controller flow control is not supported.<br>`1b` = Network Controller to Management Controller flow control is supported. |
| 3 | Management Controller to Network Controller Flow Control Support | `0b` = Management Controller to Network Controller flow control is not supported.<br>`1b` = Management Controller to Network Controller flow control is supported. |
| 4 | All multicast addresses support | `0b` = The channel cannot accept all multicast addresses. The channel does not support enable/disable global multicast commands.<br>`1b` = The channel can accept all multicast addresses. The channel supports enable/disable global multicast commands. |
| 5..31 | Reserved | Reserved |

#### 8.4.46.2 Broadcast Packet Filter Capabilities Field

The Broadcast Packet Filter Capabilities field defines the optional broadcast packet filtering capabilities that the channel supports. The bit definitions for this field correspond directly with the bit definitions for the Broadcast Packet Filter Settings field defined for the Enable Broadcast Filter command in Table 69. A bit set to 1 indicates that the channel supports the filter associated with that bit position; otherwise, the channel does not support that filter.

#### 8.4.46.3 Multicast Packet Filter Capabilities Field

The Multicast Packet Filter Capabilities field defines the optional multicast packet filtering capabilities that the channel supports. The bit definitions for this field correspond directly with the bit definitions for the Multicast Packet Filter Settings field defined for the Enable Global Multicast Filter command in Table 74. A bit set to 1 indicates that the channel supports the filter associated with that bit position; otherwise, the channel does not support that filter.

#### 8.4.46.4 Buffering Capability Field

The Buffering Capability field defines the amount of buffering in bytes that the channel provides for inbound packets destined for the Management Controller. The Management Controller may make use of this value in software-based Device Selection implementations to determine the relative time for which a specific channel may be disabled before it is likely to start dropping packets. A value of 0 indicates that the amount of buffering is unspecified.

#### 8.4.46.5 AEN Control Support Field

The AEN Control Support field indicates various standard AENs supported by the implementation. The format of the field is shown in Table 38.

#### 8.4.46.6 VLAN Filter Count Field

The VLAN Filter Count field indicates the number of VLAN filters, up to 15, that the channel supports, as defined by the Set VLAN Filter command.

#### 8.4.46.7 Mixed, Multicast, and Unicast Filter Count Fields

The Mixed Filter Count field indicates the number of mixed address filters that the channel supports. A mixed address filter can be used to filter on specific unicast or multicast MAC addresses.

The Multicast Filter Count field indicates the number of multicast MAC address filters that the channel supports.

The Unicast Filter Count field indicates the number of unicast MAC address filters that the channel supports.

The channel is required to support at least one unicast or mixed filter, such that at least one unicast MAC address can be configured on the interface. The total number of unicast, multicast, and mixed filters shall not exceed 8.

#### 8.4.46.8 VLAN Mode Support Field

The VLAN Mode Support field indicates various modes supported by the implementation. The format of field is defined in Table 87.

**Table 87 – VLAN Mode Support Bit Definitions**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | VLAN only | 1 = VLAN shall be supported in the implementation. |
| 1 | VLAN + non-VLAN | 0 = Filtering 'VLAN + non-VLAN' traffic is not supported in the implementation.<br>1 = Filtering 'VLAN + non-VLAN' traffic is supported in the implementation. |
| 2 | Any VLAN + non-VLAN | 0 = Filtering 'Any VLAN + non-VLAN' traffic is not supported in the implementation.<br>1 = Filtering 'Any VLAN + non-VLAN' traffic is supported in the implementation. |
| 3..7 | Reserved | 0 |

#### 8.4.46.9 Channel Count Field

The Channel Count field indicates the number of channels supported by the Network Controller.

### 8.4.47 Get Parameters Command (`0x17`)

The Get Parameters command can be used by the Management Controller to request that the channel send the Management Controller a copy of all of the currently stored parameter settings that have been put into effect by the Management Controller, plus "other" Host/Channel parameter values that may be added to the Get Parameters Response Payload.

Table 88 illustrates the packet format for the Get Parameters command.

**Table 88 – Get Parameters Command Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.48 Get Parameters Response (`0x97`)

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Get Parameters command and send a response. As shown in Table 89, each parameter shall return the value that was set by the Management Controller. If the parameter is not supported, 0 is returned. Currently no command-specific reason code is identified for this response.

The payload length of this response packet will vary according to how many MAC address filters or VLAN filters the channel supports. All supported MAC addresses are returned at the end of the packet, without any intervening padding between MAC addresses.

MAC addresses are returned in the following order: unicast filtered addresses first, followed by multicast filtered addresses, followed by mixed filtered addresses, with the number of each corresponding to those reported through the Get Capabilities command. For example, if the interface reports four unicast filters, two multicast filters, and two mixed filters, then MAC addresses 1 through 4 are those currently configured through the interface's unicast filters, MAC addresses 5 and 6 are those configured through the multicast filters, and 7 and 8 are those configured through the mixed filters. Similarly, if the interface reports two unicast filters, no multicast filters, and six mixed filters, then MAC addresses 1 and 2 are those currently configured through the unicast filters, and 3 through 8 are those configured through the mixed filters.

**Table 89 – Get Parameters Response Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | MAC Address Count | Reserved | | MAC Address Flags |
| **24..27** | VLAN Tag Count | Reserved | VLAN Tag Flags | |
| **28..31** | Link Settings | | | |
| **32..35** | Broadcast Packet Filter Settings | | | |
| **36..39** | Configuration Flags | | | |
| **40..43** | VLAN Mode | Flow Control Enable | Reserved | |
| **44..47** | AEN Control | | | |
| **48..51** | MAC Address 1 byte 5 | MAC Address 1 byte 4 | MAC Address 1 byte 3 | MAC Address 1 byte 2 |
| **52..55**[a] | MAC Address 1 byte 1 | MAC Address 1 byte 0 | MAC Address 2 byte 5 | MAC Address 2 byte 4 |
| **56..59** | MAC Address 2 byte 3 | MAC Address 2 byte 2 | MAC Address 2 byte 1 | MAC Address 2 byte 0 |
| **variable** | … | | | |
| | VLAN Tag 1 | | VLAN Tag 2 | |
| | … | | | |
| | … | | Pad (if needed) | |
| | Checksum | | | |

[a] Variable fields can start at this byte offset.

Table 90 lists the parameters for which values are returned in this response packet.

**Table 90 – Get Parameters Data Definition**

| Parameter Field Name | Description |
|---|---|
| MAC Address Count | The number of MAC addresses supported by the channel |
| MAC Address Flags | The enable/disable state for each supported MAC address<br>See Table 91. |
| VLAN Tag Count | The number of VLAN Tags supported by the channel |

| Parameter Field Name | Description |
|---|---|
| VLAN Tag Flags | The enable/disable state for each supported VLAN Tag<br>See Table 92. |
| Link Settings | The 32-bit Link Settings value as defined in the Set Link command |
| Broadcast Packet Filter Settings | The current 32-bit Broadcast Packet Filter Settings value |
| Configuration Flags | See Table 93. |
| VLAN Mode | See Table 58. |
| Flow Control Enable | See Table 79. |
| AEN Control | See Table 38. |
| MAC Address 1..24 | The current contents of up to eight 6-byte MAC address filter values |
| VLAN Tag 1..15 | The current contents of up to 15 16-bit VLAN Tag filter values |
| NOTE: The contents of the various configuration value fields, such as MAC Address, VLAN Tags, Link Settings, and Broadcast Packet Filter Settings, shall be considered valid only when the corresponding configuration bit is set (Enabled) in the Configuration Flags field. | |

2110 The format of the MAC Address Flags field is defined in Table 91.

2111 **Table 91 – MAC Address Flags Bit Definitions**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | MAC address 1 status | 0b = Default or unsupported or disabled<br>1b = Enabled |
| 1 | MAC address 2 status, or Reserved | 0b = Default or unsupported or disabled<br>1b = Enabled |
| 2 | MAC address 3 status, or Reserved | 0b = Default or unsupported or disabled<br>1b = Enabled |
| … | … | … |
| 7 | MAC address 8 status, or Reserved | 0b = Default or unsupported or disabled<br>1b = Enabled |

2112 The format of the VLAN Tag Flags field is defined in Table 92.

2113 **Table 92 – VLAN Tag Flags Bit Definitions**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | VLAN Tag 1 status | 0b = Default or unsupported or disabled<br>1b = Enabled |
| 1 | VLAN Tag 2 status, or Reserved | 0b = Default or unsupported or disabled<br>1b = Enabled |
| 2 | VLAN Tag 3 status, or Reserved | 0b = Default or unsupported or disabled<br>1b = Enabled |
| … | … | … |
| 14 | VLAN Tag 15 status, or Reserved | 0b = Default or unsupported or disabled<br>1b = Enabled |

2114 The format of the Configuration Flags field is defined in Table 93.

2115 **Table 93 – Configuration Flags Bit Definitions**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | Broadcast Packet Filter status | `0b` = Disabled<br>`1b` = Enabled |
| 1 | Channel Enabled | `0b` = Disabled<br>`1b` = Enabled |
| 2 | Channel Network TX Enabled | `0b` = Disabled<br>`1b` = Enabled |
| 3 | Global Multicast Packet Filter Status | `0b` = Disabled<br>`1b` = Enabled |
| 4..31 | Reserved | Reserved |

2116 **8.4.49 Get Controller Packet Statistics Command (`0x18`)**

2117 The Get Controller Packet Statistics command may be used by the Management Controller to request a
2118 copy of the aggregated packet statistics that the channel maintains for its external interface to the LAN
2119 network. The statistics are an aggregation of statistics for both the host side traffic and the NC-SI Pass-
2120 through traffic.

2121 **Table 94 – Get Controller Packet Statistics Command Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.50 Get Controller Packet Statistics Response (`0x98`)

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Get Controller Packet Statistics command and send the response packet shown in Table 95.

The Get Controller Packet Statistics Response frame contains a set of statistics counters that monitor the LAN traffic in the Network Controller. Implementation of the counters listed in Table 96 is optional. The Network Controller shall return any unsupported counter with a value of `0xFFFFFFFF`.

**Table 95 – Get Controller Packet Statistics Response Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Counters Cleared From Last Read (MS Bits) | | | |
| **24..27** | Counters Cleared From Last Read (LS Bits) | | | |
| **28..35** | Total Bytes Received | | | |
| **36..43** | Total Bytes Transmitted | | | |
| **44..51** | Total Unicast Packets Received | | | |
| **52..59** | Total Multicast Packets Received | | | |
| **60..67** | Total Broadcast Packets Received | | | |
| **68..75** | Total Unicast Packets Transmitted | | | |
| **76..83** | Total Multicast Packets Transmitted | | | |
| **84..91** | Total Broadcast Packets Transmitted | | | |
| **88..91** | FCS Receive Errors | | | |
| **92..95** | Alignment Errors | | | |
| **96..99** | False Carrier Detections | | | |
| **100..103** | Runt Packets Received | | | |
| **104..107** | Jabber Packets Received | | | |
| **108..111** | Pause XON Frames Received | | | |
| **112..115** | Pause XOFF Frames Received | | | |
| **116..119** | Pause XON Frames Transmitted | | | |
| **120..123** | Pause XOFF Frames Transmitted | | | |
| **124..127** | Single Collision Transmit Frames | | | |
| **128..131** | Multiple Collision Transmit Frames | | | |
| **132..135** | Late Collision Frames | | | |
| **136..139** | Excessive Collision Frames | | | |
| **140..143** | Control Frames Received | | | |
| **144..147** | 64-Byte Frames Received | | | |
| **148..151** | 65–127 Byte Frames Received | | | |

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **152..155** | 128–255 Byte Frames Received | | | |
| **156..159** | 256–511 Byte Frames Received | | | |
| **160..163** | 512–1023 Byte Frames Received | | | |
| **164..167** | 1024–1522 Byte Frames Received | | | |
| **168..171** | 1523–9022 Byte Frames Received | | | |
| **172..175** | 64-Byte Frames Transmitted | | | |
| **176..179** | 65–127 Byte Frames Transmitted | | | |
| **180..183** | 128–255 Byte Frames Transmitted | | | |
| **184..187** | 256–511 Byte Frames Transmitted | | | |
| **188..191** | 512–1023 Byte Frames Transmitted | | | |
| **192..195** | 1024–1522 Byte Frames Transmitted | | | |
| **196..199** | 1523–9022 Byte Frames Transmitted | | | |
| **200..207** | Valid Bytes Received | | | |
| **208..211** | Error Runt Packets Received | | | |
| **212..215** | Error Jabber Packets Received | | | |
| **216..219** | Checksum | | | |

2129 **Table 96 – Get Controller Packet Statistics Counter Numbers**

| Counter Number | Name | Meaning |
|---|---|---|
| 0 | Total Bytes Received | Counts the number of bytes received |
| 1 | Total Bytes Transmitted | Counts the number of bytes transmitted |
| 2 | Total Unicast Packets Received | Counts the number of good (FCS valid) packets received that passed L2 filtering by a specific MAC address |
| 3 | Total Multicast Packets Received | Counts the number of good (FCS valid) multicast packets received |
| 4 | Total Broadcast Packets Received | Counts the number of good (FCS valid) broadcast packets received |
| 5 | Total Unicast Packets Transmitted | Counts the number of good (FCS valid) packets transmitted that passed L2 filtering by a specific MAC address |
| 6 | Total Multicast Packets Transmitted | Counts the number of good (FCS valid) multicast packets transmitted |
| 7 | Total Broadcast Packets Transmitted | Counts the number of good (FCS valid) broadcast packets transmitted |
| 8 | FCS Receive Errors | Counts the number of receive packets with FCS errors |

| Counter Number | Name | Meaning |
|---|---|---|
| 9 | Alignment Errors | Counts the number of receive packets with alignment errors |
| 10 | False Carrier Detections | Counts the false carrier errors reported by the PHY |
| 11 | Runt Packets Received | Counts the number of received frames that passed address filtering, were less than minimum size (64 bytes from <Destination Address> through <FCS>, inclusively), and had a valid FCS |
| 12 | Jabber Packets Received | Counts the number of received frames that passed address filtering, were greater than the maximum size, and had a bad FCS |
| 13 | Pause XON Frames Received | Counts the number of XON packets received from the network |
| 14 | Pause XOFF Frames Received | Counts the number of XOFF packets received from the network |
| 15 | Pause XOFF Frames Transmitted | Counts the number of XON packets transmitted to the network |
| 16 | Pause XOFF Frames Transmitted | Counts the number of XOFF packets transmitted to the network |
| 17 | Single Collision Transmit Frames | Counts the number of times that a successfully transmitted packet encountered a single collision |
| 18 | Multiple Collision Transmit Frames | Counts the number of times that a transmitted packet encountered more than one collision but fewer than 16 |
| 19 | Late Collision Frames | Counts the number of collisions that occurred after one slot time (defined by IEEE 802.3) |
| 20 | Excessive Collision Frames | Counts the number of times that 16 or more collisions occurred on a single transmit packet |
| 21 | Control Frames Received | Counts the number of MAC control frames received that are *not* XON or XOFF flow control frames |
| 22 | 64 Byte Frames Received | Counts the number of good packets received that are exactly 64 bytes (from <Destination Address> through <FCS>, inclusively) in length |
| 23 | 65–127 Byte Frames Received | Counts the number of good packets received that are 65–127 bytes (from <Destination Address> through <FCS>, inclusively) in length |
| 24 | 128–255 Byte Frames Received | Counts the number of good packets received that are 128–255 bytes (from <Destination Address> through <FCS>, inclusively) in length |
| 25 | 256–511 Byte Frames Received | Counts the number of good packets received that are 256–511 bytes (from <Destination Address> through <FCS>, inclusively) in length |
| 26 | 512–1023 Byte Frames Received | Counts the number of good packets received that are 512–1023 bytes (from <Destination Address> through <FCS>, inclusively) in length |

| Counter Number | Name | Meaning |
|---|---|---|
| 27 | 1024–1522 Byte Frames Received | Counts the number of good packets received that are 1024–1522 bytes (from <Destination Address> through <FCS>, inclusively) in length |
| 28 | 1523–9022 Byte Frames Received | Counts the number of received frames that passed address filtering and were greater than 1523 bytes in length |
| 29 | 64 Byte Frames Transmitted | Counts the number of good packets transmitted that are exactly 64 bytes (from <Destination Address> through <FCS>, inclusively) in length |
| 30 | 65–127 Byte Frames Transmitted | Counts the number of good packets transmitted that are 65–127 bytes (from <Destination Address> through <FCS>, inclusively) in length |
| 31 | 128–255 Byte Frames Transmitted | Counts the number of good packets transmitted that are 128–255 bytes (from <Destination Address> through <FCS>, inclusively) in length |
| 32 | 256–511 Byte Frames Transmitted | Counts the number of good packets transmitted that are 256–511 bytes (from <Destination Address> through <FCS>, inclusively) in length |
| 33 | 512–1023 Byte Frames Transmitted | Counts the number of good packets transmitted that are 512–1023 bytes (from <Destination Address> through <FCS>, inclusively) in length |
| 34 | 1024–1522 Byte Frames Transmitted | Counts the number of good packets transmitted that are 1024–1522 bytes (from <Destination Address> through <FCS>, inclusively) in length |
| 35 | 1523–9022 Byte Frames Transmitted | Counts the number of transmitted frames that passed address filtering and were greater than 1523 in length |
| 36 | Valid Bytes Received | Counts the bytes received in all packets that did not manifest any type of error |
| 37 | Error Runt Packets Received | Counts the number of invalid frames that were less than the minimum size (64 bytes from <Destination Address> through <FCS>, inclusively) |
| 38 | Error Jabber Packets Received | Counts Jabber packets, which are defined as packets that exceed the programmed MTU size *and* have a bad FCS value |

The Network Controller shall also indicate in the Counters Cleared from Last Read fields whether the corresponding field has been cleared (possibly by the host) since it was last read by means of the NC-SI. The Counters Cleared from Last Read fields have the format shown in Table 97.

Currently no command-specific reason code is identified for this response.

**Table 97 – Counters Cleared from Last Read Fields Format**

| Field | Bits | Mapped to Counter Numbers |
|---|---|---|
| MS Bits | 0..6 | 32..38 |
| | 7..31 | Reserved |
| LS Bits | 0..31 | 0..31 |

## 8.4.51 Get NC-SI Statistics Command (`0x19`)

In addition to the packet statistics accumulated on the LAN network interface, the channel separately accumulates a variety of NC-SI specific packet statistics for the channel. The Get NC-SI Statistics command may be used by the Management Controller to request that the channel send a copy of all current NC-SI packet statistic values for the channel. The implementation may or may not include statistics for commands that are directed to the package.

Table 98 illustrates the packet format of the Get NC-SI Statistics command.

**Table 98 – Get NC-SI Statistics Command Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

## 8.4.52 Get NC-SI Statistics Response (`0x99`)

In the absence of any error, the channel shall process and respond to the Get NC-SI Statistics command by sending the response packet and payload shown in Table 99.

**Table 99 – Get NC-SI Statistics Response Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | NC-SI Commands Received | | | |
| **24..27** | NC-SI Control Packets Dropped | | | |
| **28..31** | NC-SI Command Type Errors | | | |
| **32..35** | NC-SI Command Checksum Errors | | | |
| **36..39** | NC-SI Receive Packets | | | |
| **40..43** | NC-SI Transmit Packets | | | |
| **44..47** | AENs Sent | | | |
| **48..51** | Checksum | | | |

The Get NC-SI Statistics Response frame contains a set of statistics counters that monitor the NC-SI traffic in the Network Controller. Counters that are supported shall be reset to `0x0` when entering into the Initial State and after being read. Implementation of the counters shown in Table 100 is optional. The Network Controller shall return any unsupported counter with a value of `0xFFFFFFFF`. Counters may wraparound or stop if they reach `0xFFFFFFFE`.

Currently no command-specific reason code is identified for this response.

**Table 100 – Get NC-SI Statistics Response Counters**

| Counter Number | Name | Meaning |
|---|---|---|
| 1 | NC-SI Commands Received | Counts the number of NC-SI frames received and identified as valid NC-SI commands (commands that generate a response packet) |
| 2 | NC-SI Control Packets Dropped | Counts the number of NC-SI Control packets that were received and dropped |
| 3 | NC-SI Command Type Errors | Counts the number of NC-SI commands that had a Command Unsupported response code |
| 4 | NC-SI Command Checksum Errors | Counts the number of NC-SI commands that had a checksum invalid error (if checksum is supported by the channel) |
| 5 | NC-SI Receive Packets | Counts the total number of NC-SI Control packets received |
| 6 | NC-SI Transmit Packets | Counts the total number of NC-SI Control packets transmitted to the Management Controller |
| 7 | AENs Sent | Counts the total number of AEN packets transmitted to the Management Controller |

### 8.4.53 Get NC-SI Pass-through Statistics Command (`0x1A`)

The Get NC-SI Pass-through Statistics command may be used by the Management Controller to request that the channel send a copy of all current NC-SI Pass-through packet statistic values.

Table 101 illustrates the packet format of the Get NC-SI Pass-through Statistics command.

**Table 101 – Get NC-SI Pass-through Statistics Command Packet Format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.54 Get NC-SI Pass-through Statistics Response (`0x9A`)

In the absence of any error, the channel shall process and respond to the Get NC-SI Pass-through Statistics command by sending the response packet and payload shown in Table 102.

**Table 102 – Get NC-SI Pass-through Statistics Response Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..27** | Pass-through TX Packets Received on NC-SI Interface (Management Controller to Network Controller) | | | |
| **28..31** | Pass-through TX Packets Dropped | | | |
| **32..35** | Pass-through TX Packet Channel State Errors | | | |
| **36..39** | Pass-through TX Packet Undersized Errors | | | |
| **40..43** | Pass-through TX Packet Oversized Errors | | | |
| **44..47** | Pass-through RX Packets Received on LAN Interface | | | |
| **48..51** | Total Pass-through RX Packets Dropped | | | |
| **52..55** | Pass-through RX Packet Channel State Errors | | | |
| **56..59** | Pass-through RX Packet Undersized Errors | | | |
| **60..63** | Pass-through RX Packet Oversized Errors | | | |
| **64..67** | Checksum | | | |

The Get NC-SI Statistics Response frame contains a set of statistics counters that monitor the NC-SI Pass-through traffic in the Network Controller. Supported counters shall be reset to `0x0` when entering into the Initial State and after being read. Implementation of the counters shown in Table 103 is optional. The Network Controller shall return any unsupported counter with a value of `0xFFFFFFFF`. Counters may wraparound or stop if they reach `0xFFFFFFFE`.

**Table 103 – Get NC-SI Statistics Response**

| Counter Number | Name | Meaning |
|---|---|---|
| 1 | Total Pass-through TX Packets Received (Management Controller to Channel) | Counts the number of Pass-through packets forwarded by the channel to the LAN |
| 2 | Total Pass-through TX Packets Dropped (Management Controller to Channel) | Counts the number of Pass-through packets from the Management Controller that were dropped by the Network Controller |
| 3 | Pass-through TX Packet Channel State Errors (Management Controller to Channel) | Counts the number of egress management packets (Management Controller to Network Controller) that were dropped because the channel was in the disabled state when the packet was received |
| 4 | Pass-through TX Packet Undersized Errors (Management Controller to Channel) | Counts the number of Pass-through packets from the Management Controller that were undersized (under 64 bytes, including FCS) |

| Counter Number | Name | Meaning |
|---|---|---|
| 5 | Pass-through TX Packet Oversized Errors (Management Controller to Channel) | Counts the number of Pass-through packets from the Management Controller that were oversized (over 1522 bytes, including FCS) |
| 6 | Total Pass-through RX Packets Received On the LAN Interface (LAN to Channel) | Counts the number of Pass-through packets that were received on the LAN interface of the channel. This counter does not necessarily count the number of packets that were transmitted to the Management Controller, because some of the packets might have been dropped due to RX queue overflow. |
| 7 | Total Pass-through RX Packets Dropped (LAN to Channel) | Counts the number of Pass-through packets that were received on the LAN interface of the channel but were dropped and not transmitted to the Management Controller |
| 8 | Pass-through RX Packet Channel State Errors (LAN to Channel) | Counts the number of ingress management packets (channel to Management Controller) that were dropped because the channel was in the disabled state when the packet was received |
| 9 | Pass-through RX Packet Undersized Errors (LAN to Channel) | Counts the number of Pass-through packets from the LAN that were undersized (under 64 bytes, including FCS) |
| 10 | Pass-through RX Packet Oversized Errors (LAN to Channel) | Counts the number of Pass-through packets from the LAN that were oversized (over 1522 bytes, including FCS) |

Currently no command-specific reason code is identified for this response.

### 8.4.55 OEM Command (`0x50`)

The OEM command may be used by the Management Controller to request that the channel provide vendor-specific information. The Vendor Enterprise Number is the unique MIB/SNMP Private Enterprise number assigned by IANA per organization. Vendors are free to define their own internal data structures in the vendor data fields.

Table 104 illustrates the packet format of the OEM command.

**Table 104 – OEM Command Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Manufacturer ID (IANA) | | | |
| **20…** | Vendor-Data<br>Note: The optional checksum is unspecified for the OEM command. OEMs supporting checksum validation for NC-SI commands may include the checksum in the OEM specific payload for the command and response. | | | |

### 8.4.56 OEM Response (`0xD0`)

The channel shall return the "Unknown Command Type" reason code for any unrecognized enterprise number, using the packet format shown in Table 105. If the command is valid, the response, if any, is allowed to be vendor-specific. The `0x8000` range is recommended for vendor-specific code.

Currently no command-specific reason code is identified for this response.

**Table 105 – OEM Response Packet Format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Manufacturer ID (IANA) | | | |
| **24…** | Return Data (Optional)<br>Note: The optional checksum is unspecified for the OEM command. OEMs supporting checksum validation for NC-SI commands may include the checksum in the OEM specific payload for the command and response. | | | |

## 8.5 AEN Packet Formats

This clause defines the formats for the different types of AEN packets. For a list of the AEN types, see Table 16.

### 8.5.1 Link Status Change AEN

The Link Status Change AEN indicates to the Management Controller any changes in the channel's external interface link status.

This AEN should be sent if any change occurred in the link status (that is, the actual link mode was changed). The Link Status and OEM Link Status fields reproduce the bit definitions defined in the Get Link Status Response Packet (see Table 47).

Table 106 illustrates the packet format of the Link Status Change AEN.

**Table 106 – Link Status Change AEN Packet Format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | AEN Header | | | |
| **16..19** | Reserved | | | AEN Type = 0x00 |
| **20..23** | Link Status | | | |
| **24.27** | OEM Link Status | | | |
| **28..31** | Checksum | | | |

### 8.5.2 Configuration Required AEN

The Configuration Required AEN indicates to the Management Controller that the channel is transitioning into the Initial State. (This AEN is not sent if the channel enters the Initial State because of a Reset Channel command.)

NOTE: This AEN may not be generated in some situations in which the Network Controller goes into the Initial State. For example, some types of hardware resets may not accommodate generating the AEN.

Table 107 illustrates the packet format of the Configuration Required AEN.

**Table 107 – Configuration Required AEN Packet Format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | AEN Header | | | |
| **16..19** | Reserved | | | AEN Type = 0x01 |
| **20..23** | Checksum | | | |

### 8.5.3 Host Network Controller Driver Status Change AEN

This AEN indicates a change of the Host Network Controller Driver Status. Table 108 illustrates the packet format of the AEN.

**Table 108 – Host Network Controller Driver Status Change AEN Packet Format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | AEN Header | | | |
| **16..19** | Reserved | | | AEN Type = 0x02 |
| **20..23** | Host Network Controller Driver Status | | | |
| **24..27** | Checksum | | | |

The Host Network Controller Driver Status field has the format shown in Table 109.

**Table 109 – Host Network Controller Driver Status Format**

| Bit Position | Name | Description |
|---|---|---|
| 0 | Host Network Controller Driver Status | `0b` = The Network Controller driver for the host external network interface associated with this channel is not operational (not running).<br>`1b` = The Network Controller driver for the host external network interface associated with this channel is being reported as operational (running). |
| 1..31 | Reserved | Reserved |

# 9 Packet-Based and Op-Code Timing

Table 110 presents the timing specifications for a variety of packet-to-electrical-buffer interactions, inter-packet timings, and op-code processing requirements.

**Table 110 – NC-SI Packet-Based and Op-Code Timing Parameters**

| Name | Symbol | Value | Description |
|---|---|---|---|
| Package Deselect to Hi-Z Interval | T1 | 200 µs, max | Maximum time interval from when a Network Controller completes transmitting the response to a Deselect Package command to when the Network Controller outputs are in the high-impedance state<br>Measured from the rising edge of the first clock that follows the last bit of the packet to when the output is in the high-impedance state as defined in clause 10 |
| Package Output to Data | T2 | 2 clocks, min | Minimum time interval after powering up the output drivers before a Network Controller starts transmitting a packet through the NC-SI interface<br>Measured from the rising edge of the first clock of the packet |
| Network Controller Power Up Ready Interval | T4 | 2 s, max | Time interval from when the NC-SI on a Network Controller is powered up to when the Network Controller is able to respond to commands over the NC-SI (that is, when it enters the Initial State)<br>Measured from when $V_{ref}$ becomes available |
| Normal Execution Interval | T5 | 50 ms, max | Maximum time interval from when a controller receives a command to when it delivers a response to that command, unless otherwise specified<br>Measured from the rising edge of the first clock following the last bit of the command packet to the rising edge of the clock for the first bit of the response packet |
| Asynchronous Reset Interval | T6 | 2 s, max | Interval during which a controller is allowed to not recognize or respond to commands due to an Asynchronous Reset event<br>For a Management Controller, this means that a Network Controller could become unresponsive for up to T6 seconds if an Asynchronous Reset event occurs. This is not an error condition. The Management Controller retry behavior should be designed to accommodate this possibility. |
| Synchronous Reset Interval | T7 | 2 s, max | Interval during which a controller may not recognize or respond to requests due to a Synchronous Reset event<br>Measured from the rising edge of the first clock following the last bit of the Reset Channel response packet |
| Token Timeout | T8 | 32,000 REF_CLK min | Number of REF_CLKs before timing out while waiting for a TOKEN to be received |

| Name | Symbol | Value | Description |
|---|---|---|---|
| Op-Code Processing | T9 | 32 REF_CLK max | Number of REF_CLKs after receiving an op-code on ARB_IN to decode the op-code and generate the next op-code on ARB_OUT<br>Measured from the falling edge of the last bit of the op-code received on ARB_IN to the rising edge of the next op-code on ARB_OUT |
| Op-Code Bypass Delay | T10 | 32 REF_CLK max | Number of REF_CLK delays between a bit received on ARB_IN and the corresponding bit passed on to ARB_OUT while in Bypass Mode<br>Measured from the falling edge of the last bit of the op-code received on ARB_IN to the rising edge of the next op-code on ARB_OUT |
| TOKEN to RXD | T11 | T2 min, 32 REF_CLK max | Number of REF_CLKs after receiving TOKEN to when packet data is driven onto the RXD lines<br>Measured from the falling edge of the last bit of the op-code received on ARB_IN to the rising edge of the next op-code on ARB_OUT |
| Max XOFF Renewal Interval | T12 | 50,331,648 REF_CLK max | Maximum time period (3 XOFF Frame timer cycles) during which a channel within a package is allowed to request and renew a single XOFF condition after requesting the initial XOFF |
| IPG to TOKEN Op-code Overlap | T13 | 6 REF_CLK max | Maximum number of REF_CLKs that the beginning of TOKEN transmission can precede the end of the Inter Packet Gap. For more information, see 7.2.8. |

NOTE: If hardware arbitration is in effect, the hardware arbitration output buffer enable/disable timing specifications take precedence.

# 10 Electrical Specification

This clause provides background information about the NC-SI specification, describes the NC-SI topology, and defines the electrical, timing, signal behavior, and power-up characteristics for the NC-SI physical interface.

## 10.1 Topologies

The electrical specification defines the NC-SI electrical characteristics for one management processor and one to four Network Controller packages in a bussed “multi-drop” arrangement. The actual number of devices that can be supported may differ based on the trace characteristics and routing used to interconnect devices in an implementation.

Figure 15 shows an example topology.

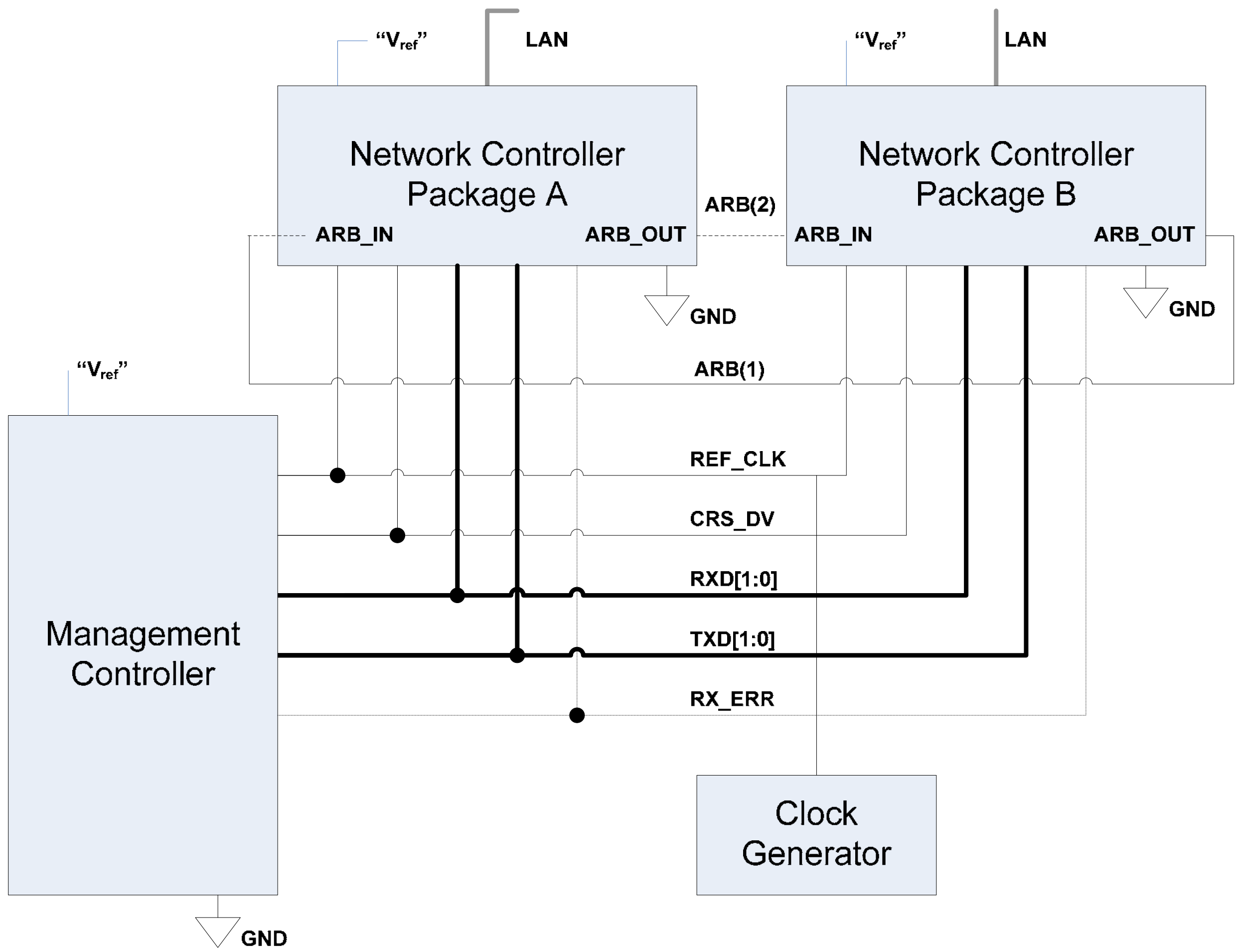



**Figure 15 – Example NC-SI Signal Interconnect Topology**

## 10.2 Electrical and Signal Characteristics and Requirements

This clause defines the electrical, timing, signal behavior, and power-up characteristics for the NC-SI physical interface.

### 10.2.1 Companion Specifications

Implementations of the physical interface and signaling for the NC-SI shall meet the specifications in RMII and IEEE 802.3, except where those requirements differ or are extended with specifications provided in this document, in which case the specifications in this document shall take precedence.

### 10.2.2 Full-Duplex Operation

The NC-SI is specified only for full-duplex operation. Half-duplex operation is not covered by this specification.

### 10.2.3 Signals

Table 111 lists the signals that make up the NC-SI physical interface.

Unless otherwise specified, the high level of an NC-SI signal corresponds to its asserted state, and the low level represents the de-asserted state. For data bits, the high level represents a binary '1' and the low level a binary '0'.

**Table 111 – Physical NC-SI Signals**

| Signal Name | Direction (with respect to the Network Controller) | Direction (with respect to the Management Controller MAC) | Use | Mandatory or Optional |
|---|---|---|---|---|
| REF_CLK [a] | Input | Input | Clock reference for receive, transmit, and control interface | M |
| CRS_DV [b] | Output | Input | Carrier Sense/Receive Data Valid | M |
| RXD[1:0] | Output | Input | Receive data | M |
| TX_EN | Input | Output | Transmit enable | M |
| TXD[1:0] | Input | Output | Transmit data | M |
| RX_ER | Output | Input | Receive error | O |
| ARB_IN | Input [c] | N/A | Network Controller hardware arbitration Input | O [c] |
| ARB_OUT | Output [c] | N/A | Network Controller hardware arbitration Output | O [c] |

[a] A device may provide an additional option to allow it to be configured as the source of REF_CLK, in which case the device is not required to provide a separate REF_CLK input line, but it can use REF_CLK input pin as an output. The selected configuration shall be in effect at NC-SI power up and remain in effect while the NC-SI is powered up.

[b] In the *RMII Specification Specification*, the MII Carrier Sense signal, CRS, was combined with RX_DV to form the CRS_DV signal. When the NC-SI is using its specified full-duplex operation, the CRS aspect of the signal is not required; therefore, the signal shall provide only the functionality of RX_DV as defined in IEEE 802.3. (This is equivalent to the CRS_DV signal states in *RMII Specification* when a carrier is constantly present.) The Carrier Sense aspect of the CRS_DV signal is not typically applicable to the NC-SI because it does not typically detect an actual carrier (unlike an actual PHY). However, the Network Controller should emulate a carrier-present status on CRS_DV per IEEE 802.3 in order to support Management Controller MACs that may require a carrier-present status for operation.

[c] If hardware arbitration is implemented, the Network Controller package shall provide both ARB_IN and ARB_OUT connections. In some implementations, ARB_IN may be required to be tied to a logic high or low level if it is not used.

### 10.2.4 High-Impedance Control

Shared NC-SI operation requires Network Controller devices to be able to set their NC-SI outputs (RXD[1:0], CRS_DV, and, if implemented, RX_ER) into a high-impedance state either upon receipt of a command received through NC-SI, or, if hardware-based arbitration is in effect, as a result of hardware-based arbitration.

Network Controller packages shall leave their NC-SI outputs in the high-impedance state on interface power up and shall not drive their NC-SI outputs until selected. For additional information about Network Controller packages, see 8.4.5.

For NC-SI output signals in this specification, unless otherwise specified, the high-impedance state is defined as the state in which the signal leakage meets the Iz specification provided in 10.2.5.

### 10.2.5 DC Characteristics

This clause defines the DC characteristics of the NC-SI physical interface.

#### 10.2.5.1 Signal Levels

CMOS 3.3 V signal levels are used for this specification.

The following characteristics apply to DC signals:

- Unless otherwise specified, DC signal levels and $V_{ref}$ are measured relative to Ground (GND) at the respective device providing the interface, as shown in Figure 16.
- Input specifications refer to the signals that a device shall accept for its input signals, as measured at the device.
- Output specifications refer to signal specifications that a device shall emit for its output signals, as measured at the device.

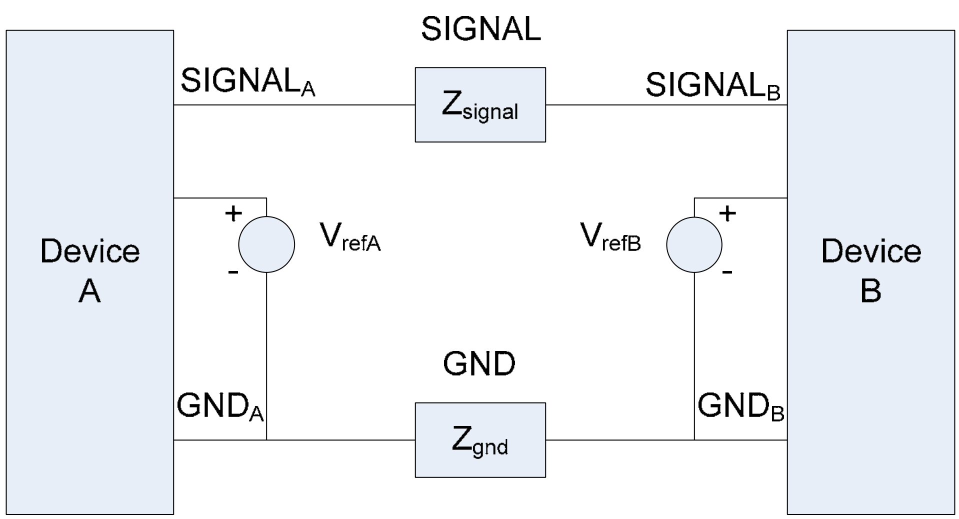



**Figure 16 – DC Measurements**

Table 112 provides DC specifications.

**Table 112 – DC Specifications**

| Parameter | Symbol | Conditions | Minimum | Typical | Maximum | Units |
|---|---|---|---|---|---|---|
| IO reference voltage | $V_{ref}$ [a] | | 3.0 | 3.3 | 3.6 | V |
| Signal voltage range | $V_{abs}$ | | -0.300 | | 3.765 | V |
| Input low voltage | $V_{il}$ | | | | 0.8 | V |
| Input high voltage | $V_{ih}$ | | 2.0 | | | V |
| Input high current | $I_{ih}$ | $V_{in} = V_{ref} = V_{ref}$,max | 0 | | 200 | µA |
| Input low current | $I_{il}$ | $V_{in}$ = 0 V | -20 | | 0 | µA |
| Output low voltage | $V_{ol}$ | $I_{ol}$ = 4 mA, $V_{ref}$ = min | 0 | | 400 | mV |
| Output high voltage | $V_{oh}$ | $I_{oh}$ = -4 mA, $V_{ref}$ = min | 2.4 | | $V_{ref}$ | V |
| Clock midpoint reference level | $V_{ckm}$ | | | | 1.4 | V |
| Leakage current for output signals in high-impedance state | $I_z$ | $0 \le V_{in} \le V_{ref}$ at $V_{ref} = V_{ref}$,max | -20 | | 20 | µA |

[a] $V_{ref}$ = Bus high reference level (typically the NC-SI logic supply voltage). This parameter replaces the term *supply voltage* because actual devices may have internal mechanisms that determine the operating reference for the NC-SI that are different from the devices' overall power supply inputs.

$V_{ref}$ is a reference point that is used for measuring parameters (such as overshoot and undershoot) and for determining limits on signal levels that are generated by a device. In order to facilitate system implementations, a device shall provide a mechanism (for example, a power supply pin, internal programmable reference, or reference level pin) to allow $V_{ref}$ to be set to within 20 mV of any point in the specified $V_{ref}$ range. This approach enables a system integrator to establish an interoperable $V_{ref}$ level for devices on the NC-SI.

## 10.2.6 AC Characteristics

This clause defines the AC characteristics of the NC-SI physical interface.

### 10.2.6.1 Rise and Fall Time Measurement

Rise and fall time are measured between points that cross 10% and 90% of $V_{ref}$ (see Table 112). The middle points (50% of $V_{ref}$) are marked as $V_{ckm}$ and $V_m$ for clock and data, respectively.

### 10.2.6.2 REF_CLK Measuring Points

In Figure 17, REF_CLK duty cycle measurements are made from $V_{ckm}$ to $V_{ckm}$. Clock skew $T_{skew}$ is measured from $V_{ckm}$ to $V_{ckm}$ of two NC-SI devices and represents maximum clock skew between any two devices in the system.

### 10.2.6.3 Data, Control, and Status Signal Measuring Points

In Figure 17, all timing measurements are made between $V_{ckm}$ and $V_m$. $T_{co}$ is measured with a capacitive load between 10 pF and 50 pF. Propagation delay $T_{prop}$ is measured from $V_m$ on the transmitter to $V_m$ on the receiver.

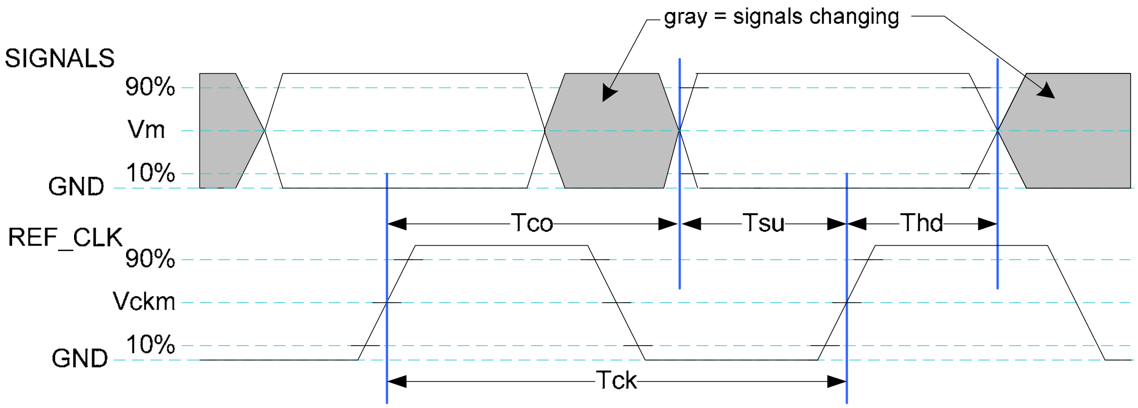



**Figure 17 – AC Measurements**

Table 113 provides AC specifications.

**Table 113 – AC Specifications**

| Parameter | Symbol | Minimum | Typical | Maximum | Units |
|---|---|---|---|---|---|
| REF_CLK Frequency | | | 50 | 50+100 ppm | MHz |
| REF_CLK Duty Cycle | | 35 | | 65 | % |
| Clock-to-out [a]<br>(10 pF ≤ $c_{load}$ ≤ 50 pF) | $T_{co}$ | 2.5 | | 12.5 | ns |
| Skew between clocks | $T_{skew}$ | | | 1.5 | ns |
| TXD[1:0], TX_EN, RXD[1:0], CRS_DV, RX_ER Data Setup to REF_CLK rising edge | $T_{su}$ | 3 | | | ns |
| TXD[1:0], TX_EN, RXD[1:0], CRS_DV, RX_ER data hold from REF_CLK rising edge | $T_{hd}$ | 1 | | | ns |
| Signal Rise/Fall Time | $T_r/T_f$ | 0.5 | | 6 | ns |
| REF_CLK Rise/Fall Time | $T_{ckr}/T_{ckf}$ | 0.5 | | 3.5 | ns |
| Interface Power-Up High-Impedance Interval | $T_{pwrz}$ | 2 | | | µs |
| Power Up Transient Interval (recommendation) | $T_{pwrt}$ | | | 100 | ns |
| Power Up Transient Level (recommendation) | $V_{pwrt}$ | -200 | | 200 | mV |
| Interface Power-Up Output Enable Interval | $T_{pwre}$ | | | 10 | ms |
| EXT_CLK Startup Interval | $T_{clkstrt}$ | | | 100 | ms |

[a] This timing relates to the output pins, while $T_{su}$ and $T_{hd}$ relate to timing at the input pins.

#### 10.2.6.4 Timing Calculation (Informative)

This clause presents the relationships between the timing parameters and how they are used to calculate setup and hold time margins.

##### 10.2.6.4.1 Setup Calculation

$$T_{su} \leq T_{clk} - (T_{skew} + T_{co} + T_{prop})$$

##### 10.2.6.4.2 Hold Calculation

$$T_{hd} \leq T_{co} - T_{skew} + T_{prop}$$

#### 10.2.6.5 Overshoot Specification

Devices shall accept signal overshoot within the ranges specified in Figure 18, measured at the device, without malfunctioning.

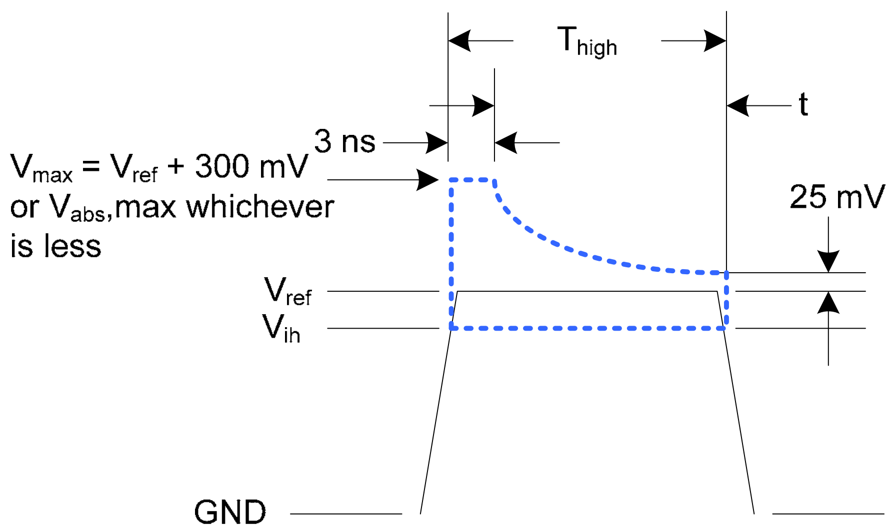



**Figure 18 – Overshoot Measurement**

The signal is allowed to overshoot up to the specified $V_{max}$ for the first 3 ns following the transition above $V_{ih}$. Following that interval is an exponential decay envelope equal to the following:

$$V_{ref} + V_{os} * e\text{^}[- K * ( [t - 3 \text{ ns}] /T_d)]$$

Where, for t = 3 to 10 ns:

t = 0 corresponds to the leading crossing of $V_{ih}$, going high.

$V_{ref}$ is the bus high reference voltage (see 10.2.5).

$V_{abs}$,max is the maximum allowed signal voltage level (see 10.2.5).

$V_{os} = V_{max} - V_{ref}$

$K = l_n(25 \text{ mV}/V_{os})$

$T_d = 7$ ns

For t > 10 ns, the $V_{ref}$ + 25 mV limit holds flat until the conclusion of $T_{high}$.

#### 10.2.6.6 Undershoot Specification

Devices are required to accept signal undershoot within the ranges specified in Figure 19, measured at the device, without malfunctioning.

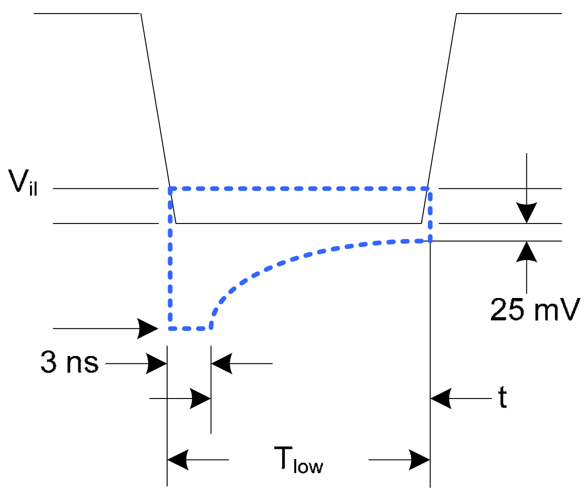



**Figure 19 – Undershoot Measurement**

The signal is allowed to undershoot up to the specified $V_{abs}$,min for the first 3 ns following the transition above $V_{il}$. Following that interval is an exponential envelope equal to the following:

$* ([t - 3\ ns]/T_d)]$

Where, for t = 3 to 10 ns:

t = 0 corresponds to the leading crossing of $V_{il}$, going low.

$V_{abs}$,min is the minimum allowed signal voltage level (see 10.2.5).

$K = l_n(25\ mV/V_{os})$

$T_d$ = 7 ns

For t > 7 ns, the GND – 25 mV limit holds flat until the conclusion of $T_{low}$.

### 10.2.7 Interface Power-Up

To prevent signals from back-powering unpowered devices, it is necessary to specify a time interval during which signals are not to be driven until devices sharing the interface have had time to power up. To facilitate system implementation, the start of this interval shall be synchronized by an external signal across devices.

#### 10.2.7.1 Power-Up Control Mechanisms

The device that provides the interface shall provide one or more of the following mechanisms to enable the system integrator to synchronize interface power-up among devices on the interface:

- **Device Power Supply Pin**

  The device has a power supply pin that the system integrator can use to control power-up of the interface. The device shall hold its outputs in a high-impedance state (current < $I_z$) for at least

$T_{pwrz}$ seconds after the power supply has initially reached its operating level (where the power supply operating level is specified by the device manufacturer).

- **Device Reset Pin or Other Similar Signal**

  The device has a reset pin or other signal that the system integrator can use to control the power-up of the interface. This signal shall be able to be driven asserted during interface power-up and de-asserted afterward. The device shall hold its outputs in a high-impedance state (current < $I_z$) for at least $T_{pwrz}$ seconds after the signal has been de-asserted, other than as described in 10.2.7.2. It is highly recommended that a single signal be used; however, an implementation is allowed to use a combination of signals if required. Logic levels for the signals are as specified by the device manufacturer.

- **REF_CLK Detection**

  The device can elect to detect the presence of an active REF_CLK and use that for determining whether NC-SI power up has occurred. It is recommended that the device should count at least 100 clocks and continue to hold its outputs in a high-impedance state (current < $I_z$) for at least $T_{pwrz}$ seconds more (Informational: 100 clocks at 50 MHz is 2 us).

#### 10.2.7.2 Power-Up Transients

It is possible that a device may briefly drive its outputs while the interface or device is first receiving power, due to ramping of the power supply and design of its I/O buffers. It is recommended that devices be designed so that such transients, if present, are less than $V_{pwrt}$ and last for no more than $T_{pwrt}$.

### 10.2.8 REF_CLK Startup

REF_CLK shall start up, run, and meet all associated AC and DC specifications within $T_{clkstrt}$ seconds of interface power up.

# ANNEX A
# (normative)

# Extending the Model

This annex explains how the model can be extended to include vendor-specific content.

## A.1 Commands Extension

A Network Controller vendor may implement extensions and expose them using the OEM command, as described in 8.4.55.

## A.2 Design Considerations

This clause describes certain design considerations for vendors of Management Controllers.

### A.2.1 PHY Support

Although not a requirement of this specification, a Management Controller vendor may want to consider designing an NC-SI in such a manner that it could also be configured for use with a conventional RMII PHY. This would enable the vendor’s controller to also be used in applications where a direct, non-shared network connection is available or preferred for manageability.

### A.2.2 Multiple Management Controllers Support

Currently, there is no requirement for Management Controllers to be able to put their TXD output lines and other output lines into a high-impedance state, because the present definition assumes only one Management Controller on the bus. However, component vendors may want to consider providing such control capabilities in their devices to support possible future system topologies where more than one Management Controller shares the bus to enable functions such as Management Controller fail-over or to enable topologies where more than one Management Controller can do NC-SI communications on the bus. If a vendor elects to make such provision, it is recommended that the TXD line and the remaining output lines be independently and dynamically switched between a high-impedance state and re-enabled under firmware control.

# ANNEX B
# (informative)

# Relationship to RMII Specification

## B.1 Differences with the *RMII Specification*

The following list presents key differences and clarifications between the *NC-SI Specification* and sections in the *RMII Specification*. (Section numbers refer to the *RMII Specification*.)

- General: Where specifications from IEEE 802.3 apply, this specification uses the version specified in clause 2, rather than the earlier IEEE 802.3u version that is referenced by RMII.
- Section 1.0:
  - The *NC-SI Specification* requires 100 Mbps support, but it does not specify a required minimum. (10 Mbps support is not required by NC-SI.)
  - Item 4. (Signals may or may not be considered to be TTL. NC-SI is not 5-V tolerant.)
- Section 2.0:
  - Comment: NC-SI chip-to-chip includes considerations for multi-drop and allows for non-PCB implementations and connectors (that is, not strictly point-to-point).
- Section 3.0:
  - Note/Advisory: The NC-SI clock is provided externally. An implementation can have REF_CLK provided by one of the devices on the bus or by a separate device.
- Section 5.0:
  - For NC-SI, the term *PHY* is replaced by *Network Controller*.
- Table 1:
  - The information in Table 1 in the *RMII Specification* is superseded by tables in this specification.
- Section 5.1, paragraph 2:
  - The *NC-SI Specification* allows 100 ppm. This supersedes the *RMII Specification*, which allows 50 ppm.
- Section 5.1, paragraph 3:
  - The NC-SI inherits the same requirements. The NC-SI MTU is required only to support Ethernet MTU with VLAN, as defined in the IEEE 802.3 version listed in clause 2.
- Section 5.1 paragraph 4:
  - The *RMII Specification* states: "During a false carrier event, CRS_DV shall remain asserted for the duration of carrier activity." This statement is not applicable to full-duplex operation of the NC-SI. CRS_DV from the Network Controller is used only as a data valid (DV) signal. Because the Carrier Sense aspect of CRS_DV is not used for full-duplex operation of the NC-SI, the Network Controller would not generate false carrier events for the NC-SI. However, it is recommended that the MAC in the Management Controller be able to correctly detect and handle these patterns if they occur, as this would be part of enabling the Management Controller MAC to also be able to work with an RMII PHY.

- Section 5.2:
  - The NC-SI does not specify a 10 Mbps mode. The Carrier Sense aspect of CRS_DV is not used for full-duplex operation of NC-SI.
- Section 5.3.1:
  - While the NC-SI does not specify Carrier Sense usage of CRS_DV, it is recommended that a Management Controller allow for CRS_DV toggling, in which CRS_DV toggles at 1/2 clock frequency, and that Management Controller MACs tolerate this and realign bit boundaries correctly in order to be able to work with an RMII PHY also.
- Section 5.3.2:
  - There is no 10 Mbps mode specified for the NC-SI.
- Section 5.3.3:
  - Generally there is no expectation that the Network Controller will generate these error conditions for the NC-SI; however, the MAC in the Management Controller should be able to correctly detect and handle these patterns if they occur.
- Section 5.3.3:
  - The NC-SI does not specify or require support for RMII Registers.
- Section 5.5.2:
  - Ignore (N/A) text regarding 10 Mbps mode. The NC-SI does not specify or require interface operation in 10 Mbps mode.
- Section 5.6:
  - The Network Controller will not generate collision patterns for the specified full-duplex operation of the NC-SI; however, the MAC in the Management Controller should be able to detect and handle these patterns if they occur in order to be able to work with an RMII PHY also.
- Section 5.7:
  - NC-SI uses the IEEE 802.3 version listed in clause 2 instead of 802.3u as a reference.
- Section 5.8:
  - Loopback operation is not specified for the NC-SI.
- Section 7.0:
  - The NC-SI electrical specifications (clause 10) take precedence. (For example, section 7.4.1 in the *RMII Specification* for capacitance is superseded by *NC-SI Specification* 25 pF and 50 pF target specifications.)
- Section 8.0:
  - NC-SI uses the IEEE 802.3 version listed in clause 2 as a reference, instead of 802.3u.

# ANNEX C
# (informative)

# Change Log

| Version | Date | Editor | Description |
|---|---|---|---|
| 1.0.0 | 07/21/2009 | | DMTF Standard Release |
| | | | |
| | | | |
| | | | |

# Bibliography

IANA, Internet Assigned Numbers Authority (www.iana.org). A body that manages and organizes numbers associated with various Internet protocols.

DMTF DSP4004, *DMTF Release Process*, January 2007, http://www.dmtf.org/standards/published_documents/DSP4004_2.1.0.pdf