**Document Number: DSP0256**

**Date: 2010-07-21**

**Version: 1.0.0**

# Management Component Transport Protocol (MCTP) Host Interface Specification

**Document Type: Specification**

**Document Status: DMTF Standard**

**Document Language: en-US**

# CONTENTS

## Tables

# Forward

The *Management Component Transport Protocol (MCTP) KCS Transport Binding Specification* (DSP0256) was prepared by the PMCI Subgroup of the Pre-OS Working Group.

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. For information about the DMTF, see http://www.dmtf.org.

# Introduction

The Management Component Transport Protocol (MCTP) defines a communication model intended to facilitate communication between:

- Management controllers and other management controllers
- Management controllers and management devices

The communication model includes a message format, transport description, message exchange patterns, and configuration and initialization messages. The *MCTP Base Specification* (DSP0236) describes the protocol and commands used for communication within and initialization of an MCTP network

This *MCTP Host Interface Specification* (DSP0256) describes how MCTP packets are delivered over host interfaces (interfaces between a management controller and host software).

# Management Component Transport Protocol (MCTP) KCS Transport Binding Specification

## 1 Scope

This document provides the specifications for the Management Component Transport Protocol (MCTP) host interface.

## 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.

DMTF DSP0134, *System Management BIOS Reference Specification 2.6,* http://www.dmtf.org/standards/published_documents/DSP0134_2.6.pdf

DMTF DSP0236, *Management Component Transport Protocol (MCTP) Base Specification 1.0,* http://www.dmtf.org/standards/published_documents/DSP0236_1.0.pdf

DMTF DSP0239, *Management Component Transport Protocol (MCTP) IDs and Codes Specification 1.0,* http://www.dmtf.org/standards/published_documents/DSP0239_1.0.pdf

Hewlett-Packard, Intel, Microsoft, Phoenix, and Toshiba, *Advanced Configuration and Power Interface Specification, 4.0a*, http://www.acpi.info/spec.htm

IPMI Consortium*, Intelligent Platform Management Interface Specification*, v1.5 Revision 1.1 February 20, 2002, http://download.intel.com/design/servers/ipmi/IPMIv1_5rev1_1.pdf

PCI-SIG, *PCI Express Base Specification v1.1*, PCIe v1.1, March 28, 2005, http://www.pcisig.com/login?back=http%3A//www.pcisig.com/members/downloads/specifications/pciexpress/PCI_Express_Base_11.pdf

PCI-SIG, *PCI Express Base Specification v2.0*, PCIe v2.0, December 20, 2006, http://www.pcisig.com/login?back=http%3A//www.pcisig.com/members/downloads/specifications/pciexpress/PCI_Express_Base_Rev_2.0_20Dec06a.pdf

PCI-SIG, *PCI Local Bus Specification v3.0*, PCI v3.0, February 3, 2004, http://www.pcisig.com/login?back=http%3A//www.pcisig.com/members/downloads/specifications/conventional/PCI_LB3.0-2-6-04.pdf

## 3 Terms and Definitions

Refer to DSP0236 for terms and definitions that are used across the MCTP specifications. For the purposes of this document, the following terms and definitions apply.

# 4 Symbols and Abbreviated Terms

Refer to DSP0236 for symbols and abbreviated terms that are used across the MCTP specifications. For the purposes of this document, the following additional symbols and abbreviated terms apply.

# 5 Conventions

The conventions described in the following clauses apply to this specification.

## 5.1 Reserved and Unassigned Values

Unless otherwise specified, any reserved, unspecified, or unassigned values in enumerations or other numeric ranges are reserved for future definition by the DMTF.

Unless otherwise specified, numeric or bit fields that are designated as reserved shall be written as 0 (zero) and ignored when read.

## 5.2 Byte Ordering

Unless otherwise specified, byte ordering of multi-byte numeric fields or bit fields is "Big Endian" (that is, the lower byte offset holds the most significant byte, and higher offsets hold lesser significant bytes).

# 6 MCTP Host Interface

The MCTP Host Interface defines how MCTP packets are delivered over a host interface. This specification describes the host interface discovery and commands for registering software endpoints such as BIOS, UEFI or system software.

## 6.1 Host Interface Discovery

Host Interface discovery applies to industry standard architecture (PC) systems. Having a single “active” management controller is a restriction from the defined discovery mechanisms and not a restriction on the host interface definition per se.

A Management Controller may make available multiple host interfaces, but only one management controller is allowed to be the “active” Management Controller that provides Management controller functionality for the system (in the case of a “partitioned” system, there can only one active Management Controller per partition). Only the host interface(s) for the active Management Controller are allowed to respond to the *Get MCTP Version Support* command as defined in the *MCTP Base Specification* (DSP0236). If other Management Controller devices are present, but not being used, they must not respond to the *Get MCTP Version Support* command.

MCTP host interface can be discovered with PCI/PCIe class codes, ACPI or SMBIOS structure tables. Maintaining consistency between these structures is outside the scope of this specification.

When multiple ways of discovering host interfaces are available, the driver can discover the MCTP host interface using the approach described in this section.

Drivers should not switch host interfaces during system operation or else unexpected results could occur. The *Get MCTP Version Support* command is required to execute correctly across multiple interfaces to a management controller, but other commands are not. Once the driver has chosen to use a given interface, all commands beyond *Get MCTP Version Support* should be delivered to that interface. If it is desired to change the choice of host interfaces, a warm or cold reset of the platform should be done to ensure that the system can re-initialize the management controller operation.

It is recommended that run-time drivers support the MCTP Host interfaces in the following order:

- A driver should preferentially use the management controller on PCI/PCIe, via the OS's native support, if available. (A "Plug and Play" OS will typically locate and load the appropriate driver for devices it finds on PCI/PCIe.) Clause 8 summarizes the PCI/PCIe Class codes for MCTP Host interfaces.
- If the desired interface is not available on PCI/PCIe, or the system is in a state where OS support for PCI/PCIe is unavailable the next step should be to look for the host interface as a static resource described in ACPI using the control methods described in clause 9.
- If the operating environment does not include a mechanism to support executing ACPI control methods, then look for the host interface at the location described by the MCHI (Management Controller Host Interface) Table(s) through the ACPI Description Table mechanisms. (There is one instance of MCHI table per host interface. The MCHI Table approach supports systems that include more than one host interface. Therefore, there can be more than one instance of the MCHI Table.) The MCHI Table is described in clause 9*.*
- If the MCHI Table is not present, the driver should look for the SMBIOS Type 42 table (see clause 7) and use the interface described there. There is one instance of the Type 42 record per physical interface. Therefore, in order to discover whether there are multiple non-plug-and-play host interfaces, the driver will need to see if the SMBIOS table has multiple Type 42 records.

## 6.2 Multiple Host Interfaces

Multiple host interfaces may exist in a given system implementation. Refer to the "MCTP Overview" clause in DSP0236 for details on identifying whether multiple host interfaces connect to common or separate MCTP networks.

## 6.3 Transport-Specific Commands

In order for a transport to be an MCTP host interface it must have a transport-specific command to register system firmware or system software in order to obtain a MCTP EID. The details of this command are described in the transport-binding specifications.

# 7 Locating MCTP Host Interfaces via SMBIOS Tables

The *System Management BIOS Reference Specification* (DSP0134) includes the following optional record for identifying the initial location of MCTP host interfaces and interrupts. This is summarized in Table 1. See DSP0134 for other application information on SMBIOS.

Note that the settings that this structure reports may be over-ridden by "Plug-and-Play" reassignment by the OS. Therefore, this structure should be used only when the interface cannot be discovered via "Plug-and-Play" discovery mechanisms incorporated in interfaces such as PCI and ACPI.

**Table 1 – SMBIOS Type 42 Management Controller Host Interface Structure**

| Offset | Name | Length | Value | Description |
|---|---|---|---|---|
| 00h | Type | BYTE | 42 | Management Controller Host Interface structure indicator |
| 01h | Length | BYTE | Varies | Length of the structure, a minimum of 0x09h. |
| 02h | Handle | WORD | Varies | |
| 04h | Interface type | BYTE | ENUM | Management Controller (MC) interface type |
| 05h | Interface specific data length (n) | BYTE | Varies | Number of bytes of interface specific data |
| 06h | Interface specific data | n BYTEs | Varies | Defined by interface specification |
| 06h + n | Protocol count (m) | BYTE | ENUM | Number of protocols supported on this interface |
| 07h + n | Protocol records | M Bytes | Varies | Protocol record data (see Table 2) |

## 7.1 Protocol Identifier and Protocol-Specific Data

The protocol identifier describes the protocol used over the Management Controller Host Interface. These identifiers are defined in DSP0239, *Management Component Transport Protocol (MCTP) IDs and Codes Specification.*

The protocol specific data describes certain data that is required by the protocol, such as protocol revision numbers. This protocol-specific data for MCTP is defined in DSP0236, *Management Component Transport Protocol (MCTP) Base Specification* Management Controller Host Interface Protocol Specific Data Requirements.

Table 2 shows the Type 42 protocol record data.

**Table 2 – SMBIOS Type 42 Protocol Record Data**

| Offset | Name | Length | Value | Description |
|---|---|---|---|---|
| X | Protocol Identifier | BYTE | ENUM | Protocol Identifier |
| X + 1 | Protocol specific data length | BYTE | Varies | Number of bytes of protocol specific data |
| X + 2 | Protocol specific data | n BYTEs | Varies | Defined by protocol specification |

# 8 Locating MCTP Host Interfaces with PCI / PCIe

The MCTP host interface uses the PCI SIG (http://www.pcisig.com) class codes for IPMI Host interfaces defined in Appendix D of the *PCI Local Bus Specification*. PCI-based implementations of the IPMI Host interfaces should use the appropriate PCI configuration space and the class code definition there to report the presence and type of host interface for driver loading purposes.

The first base address register of the PCI function that implements the MCTP host interface holds the base address for the MCTP Host Interface registers. The MCTP Host Interfaces can be I/O or memory mapped, as indicated by read-only bits in the base address register.

Unless otherwise specified, MCTP Host interfaces on PCI must be byte aligned and located at offset 0 with respect to the base address register.

# 9 Locating MCTP Host Interfaces with ACPI

This specification introduces the option of describing the presence of the MCTP Host Interface as a static (non-“Plug and Play”) resource using ACPI. The MCTP Host interface can also be implemented as a relocate-able resource on PCI (refer to clause 8).

There are two ACPI-based mechanisms that work together when the MCTP Host interface is implemented as a static resource: the Management Controller Host Interface (MCHI) Description Table and ACPI Control Methods.

## 9.1 MCHI Description Table and ACPI Control Methods

The MCHI Description Table is an optional table that describes the processor-relative, translated, fixed resources of a host interface at system boot time. The purpose of the MCHI Description Table is to provide a mechanism that can be used by the OSPM (an ACPI term for “OS Operating System-directed configuration and Power Management” essentially meaning an ACPI-aware OS or OS loader) very early in the boot process, for example, before the ability to execute ACPI control methods in the OS is available.

The MCHI Description Table is similar to the SMBIOS Type 42 (MC Device Information) record. The main difference between the two is that the MCHI Table is identified in the ACPI Specification as a table that has the reserved signature “MCHI”. The SMBIOS Type 42 record type is from the DMTF SMBIOS specifications. See DSP0134.

The MCHI Description Table can be used to describe the location of either fixed resource or PCI implementations of the host interface. For host interfaces on PCI, the table can only describe the location of the host interface at the time that the boot process is initiated. An OS may relocate these resources. Therefore, whether or not a PCI-based host interface remains at the MCHI addresses is OS-dependent. During normal run-time operation, software should locate the host interface directly on PCI and/or use the OS’s support for PCI instead of the MCHI Table.

A management controller device may present more than one host interface for messaging to the Management Controller. For example, a Management Controller may simultaneously support the KCS and the Serial interfaces. A unique MCHI Table should be provided for each of these interfaces. This allows the OS to select an interface that it is able to communicate and hence maximize the supportability.

Per ACPI, unless otherwise specified, numeric values for the table and any blocks or structures are always encoded in little Endian format. Signature values are stored as fixed-length strings.

**Table 3 – Management Controller Host Interface Description Table Format**

| Field | Byte Length | Byte Offset | Description |
|---|---|---|---|
| Header | | | |
| Signature | 4 | 0 | ‘MCHI’. Signature for the Management Controller Host Interface Table. |
| Length | 4 | 4 | Length, in bytes, of the entire Management Controller Host Interface Table. |
| Revision | 1 | 8 | 1 |
| Checksum | 1 | 9 | Two’s complement of the 16 bit sum of all the bytes in the table excluding the checksum byte, modulo 256 |
| OEMID | 6 | 10 | OEM ID. Per the ACPI Specification. An OEM-supplied string that identifies the OEM. |

| Field | Byte Length | Byte Offset | Description |
|---|---|---|---|
| OEM Table ID | 8 | 16 | For the Management Controller Host Interface Table, the table ID is the manufacturer model ID (assigned by the OEM identified by "OEM ID"). |
| OEM Revision | 4 | 24 | OEM revision of Management Controller Host Interface Table for supplied the given OEM Table ID. Per the ACPI Specification, this is "An OEM-supplied revision number. Larger numbers are assumed to be newer revisions." |
| Creator ID | 4 | 28 | Vendor ID of utility that created the table. For the tables containing Definition Blocks, this is the ID for the ASL Compiler. |
| Creator Revision | 4 | 32 | Revision of utility that created the table. For the tables containing Definition Blocks, this is the revision for the ASL Compiler. |
| Interface Type | 1 | 36 | Indicates the type of Host interface:<br>0 Reserved<br>1 Reserved<br>2 Keyboard Controller Style (KCS)<br>3 Serial (8250 Compatible)<br>4 Serial (16450 Compatible)<br>5 Serial (16450/16550A Compatible)<br>6 Serial (16550/16C550 Compatible)<br>7 Serial (16650/16C650 Compatible)<br>8 Serial (16750/16C750 Compatible)<br>9-255 Reserved |
| Protocol Identifier | 1 | 37 | Indicates the protocol of Host Interface:<br>0 Unspecified<br>1 Management Controller Transport Protocol (MCTP)<br>2 Intelligent Platform Management Interface (IPMI)<br>3-254 Reserved<br>255 OEM defined (OEM is identified by OEM ID field) |
| Protocol Specific Data | 8 | 38 | Indicates protocol specific data such as specification revision of version plus other protocol specific data defined by the protocol specification |
| Interrupt Type | 1 | 46 | Interrupt type(s) used by the interface:<br>[7:2] – Reserved (must be 0)<br>[1] – I/O APIC/SAPIC interrupt (Global System Interrupt)<br>[0] – SCI triggered through GPE<br>0 = not supported<br>1 = supported |

| Field | Byte Length | Byte Offset | Description |
|---|---|---|---|
| GPE | 1 | 47 | The bit assignment of the SCI interrupt within the GPEx_STS register of a GPE described if the FADT that the interface triggers. (Note: This field is valid only if Bit[0] of the Interrupt Type field is set. Otherwise set to 00h.) |
| PCI Device Flag | 1 | 48 | [7:1] – Reserved<br><br>[0] – PCI Device Flag. For PCI devices, this bit is set. For non-PCI devices, this bit is cleared. When this bit is cleared, the PCI Segment Group, Bus, Device and Function Number fields combined corresponds to the ACPI _UID value of the device whose _HID or _CID contains DMT0001 plug and play ID. _UID must be an integer. Byte 60 contains the least significant byte of the _UID value. Set to 0b for SMBus. |
| Global System Interrupt | 4 | 49 | The I/O APIC or I/O SAPIC Global System Interrupt[1] used by the interface. (Note: This field is valid only if Bit[1] of the Interrupt Type field is set. Otherwise set to 00h.) |
| Base Address | 12 | 53 | The base address of the interface register set described using the Generic Address Structure (GAS, see ACPI for the definition). The Address_Space_ID field in the GAS can only be of the value of 0 (System Memory), 1 (System IO), and 4 (SMBus). All other values are not permitted.<br><br>For SMBus:<br><br>The Address_Space_ID = 4 and the address field of the GAS holds the 7-bit slave address of the MC on the host SMBus in the least significant byte. Note that the slave address is stored with the 7-bit slave address in the least significant 7-bits of the byte, and the most significant bit of the byte set to 0b.<br><br>Register_Bit_Width = 0<br><br>Register_Bit_Offset = 0<br><br>Address_Size field = 1 (Byte access)<br><br>Address = 7-bit SMBus address of BMC SSIF |
| PCI Segment Group Number / UID byte 1 | 1 | 65 | PCI Segment Group Number, if the device is a PCI device. Otherwise, this field is byte 1 of a UID. See description for PCI Device Flag, above. |
| PCI Bus Number / UID byte 2 | 1 | 66 | PCI Bus Number, if the device is a PCI device.<br><br>Otherwise, this field is byte 2 of a UID. See description for PCI Device Flag, above. |
| PCI Device Number / UID byte 3 | 1 | 67 | PCI Device fields or byte 3 of a UID. Per PCI Device Flag, above.<br><br>For PCI Device Flag = 1b:<br><br>[7:5] – Reserved<br><br>[4:0] – PCI Device Number: The PCI device number if the device is a PCI device.<br><br>For PCI Device Flag = 0b:<br><br>[7:0] – byte 3 of UID |

| Field | Byte Length | Byte Offset | Description |
|---|---|---|---|
| PCI Function Number / UID byte 4 | 1 | 68 | PCI Device fields or byte 4 of a UID. Per PCI Device Flag, above.<br>For PCI Device Flag = 1b:<br>[7] – Reserved<br>[6] – Interrupt Flag:<br>0b = interrupt not supported<br>1b = interrupt supported<br>[5:3] – Reserved<br>[2:0] – PCI Function Number: The PCI function number if the device is a PCI device.<br>For PCI Device Flag = 0b:<br>[7:0] – byte 4 of UID |

1. ACPI represents all interrupts as “flat” values known as global system interrupts. Therefore, to support APICs or SAPICs on an ACPI-enabled system, each used APIC or SAPIC interrupt input must be mapped to the global system interrupt value used by ACPI. See the “Global System Interrupts” section in ACPI for a description of Global System Interrupts.

## 9.2 Locating MCTP Host Interfaces in ACPI Name Space

The MCHI Description Table provides a mechanism that can be used before the ability to execute ACPI control methods in the OS is available. This table is not, however, intrinsically supported in the OS as a way of discovering and reporting system resources. Therefore, it is recommended that non-PCI MCTP Host interfaces on the baseboards be described in the ACPI name space. This makes it possible for the OS to enumerate the IPMI Host interface as a device. In addition, the ACPI name space description is more flexible and friendly in hot-plug scenarios.

Note that to be ACPI compatible, the fixed resources for MCTP Host interfaces must still be accounted for in accordance with the ACPI Specification. If the device is not formally described in the ACPI Name Space, its resources must be described as fixed system resources or the resources appended to some other fixed resource system device in order to ensure that the OS does not attempt to allocate those resources to some other device.

To formally describe the MCTP Host interface in ACPI Name Space, an MCTP device is created using the named device object. The MCTP device object can have the following elements:

257 **Table 4 – MCTP Device Object Control Methods**

| Object | Description | Support Level |
|---|---|---|
| _ADR | Named object that evaluates to the interface's address on its parent bus. _ADR is a standard device configuration control method defined in the ACPI Specification. | Required only for devices on a bus that has standard enumeration mechanism. |
| _HID | Named object that provides the interface's Plug and Play identifier. This value can be vendor specific but must set to DMT0001[1] if no CID object is provided. _HID is a standard device configuration control method defined in the ACPI Specification. | Required |
| _CID | Named object that provides the interface's compatible Plug and Play identifier. This object is required and contains the value of DMT0001 if _HID contains vendor specific identifier. Otherwise, this object is optional. | See the "Description" column for this object |
| _STR | Named object that evaluates to a Unicode string that may be used by an OS to provide information to an end user describing the device. __STR is a standard device configuration control method defined in the ACPI Specification. | Required |
| _UID | Named object that specifies a device's unique persistent ID, or a control method that generates it. _UID is a standard device configuration control method defined in the ACPI Specification. | Required if more than one device |
| _CRS | Named object that returns the interface's current resource settings. System Processor Management Interfaces are considered static resources; hence only return their defined resources. The address region definition is interface type/subtype dependent. _CRS is a standard device configuration control method defined in the ACPI Specification. | Required |
| _STA | Object that returns the status of the device: enabled, disabled or removed, as defined in the ACPI Specification. If this method is not present, the device is assumed to be enabled. | Recommended |
| _IFT | Object that specifies the interface type, as defined in the MCHI Table. (Note: _IFT and _SRV, following, have been reserved in ACPI 3.0 as names for control methods defined for MCHI.) | Required |
| _SRV | Object that specifies the specification revision, as defined in the MCHI Table. | Required |
| _GPE | Named object that evaluates to either an integer or a package. If _GPE evaluates to an integer, the value is the bit assignment of the SCI interrupt within the GPEx_STS register of a GPE block described in the FADT that the Service Processor Management Interface will trigger.<br><br>If _GPE evaluates to a package, then that package contains two elements. The first is an object reference to the GPE Block device that contains the GPE register that will be triggered by the interface. The second element is numeric (integer) that specifies the bit assignment of the SCI interrupt within the GPEx_STS register of the GPE Block device referenced by the first element in the package.<br><br>(Note: This object is only provided if the interface supports a GPE.) | Required if interrupt through GPE is supported |

258 NOTE: Normally PCI based devices are not described in ACPI name space. OS / driver should use the PCI enumeration
259 mechanism to locate MCTP interfaces (see clause 8).

[1] DMTF has registered the DMT0001 PNP ID with Microsoft for describing all DMTF related devices. DMTF has granted the use of DMT0001 to describe the generic Management Controller Device as defined in this specification.

If the MCTP interface supports interrupts, the interrupt descriptor in _CRS is used if the interrupt is supported via IO (S)APIC, while _GPE object is used if the interrupt is supported through the GPE register. Having both the interrupt descriptor in _CRS and the _GPE object in the MCTP device scope is not permitted by this specification.

If the MCTP interface does not support interrupts, neither the interrupt descriptor in the _CRS nor the _GPE object will be present.

In a multi-node system where there may be more than one MCTP device in an OS domain, it is highly recommended that all MCTP devices be described in the ACPI name space with the _STA returning enabled for the active MCTP device(s).

## 9.3 Example MCTP Definition ASL Code

Example ASL code that defines MCTP Host interfaces is provided in the following subclauses.

### 9.3.1 Example 1: KCS Interface in 64-bit Address Space

Example ASL for describing a memory-mapped KCS host interface, located in a 64-bit address space at address 0x80000FFFFC020CA2:

```
Device(MI0) {
    Name(_HID, EISAID("DMT0001"))
    Name(_STR, Unicode("MCTP_KCS"))// Optional, but recommended
                        // for identifying MCTP host interface.
                        // The strings "MCTP_KCS" and "MCTP_Serial",
                        // are recommended for
                        // identifying the KCS and Serial
                        // interfaces, respectively.

    Name(_UID, 0)     // UID for the primary MCTP host interface in the system

    // Returns the "Current Resources"
    Name(_CRS,
    ResourceTemplate() {
        QWordMemory(
        ResourceConsumer,      //
        PosDecode,             //
        MinFixed,              //
        MaxFixed,              //
        NonCacheable,          //
        ReadWrite,             //
            0xFFFFFFFFFFFFFFFF,   // _GRA, Address granularity.
                                  // E.g. All 64-bits decoded.
            0x80000FFFFC020CA2,   // _MIN, Address range minimum
                                  // (System I/F base addr.)
            0x80000FFFFC020CA4,   // _MAX, Address range max
            0x0000000000000000,   // _TRA, Translation.
                                  // 0 for non-bridge devices
            0x0000000000000002,   // _LEN, Address range length
                ,                 // Resource Source Index
                ,                 // Resource Source Name
                ,                 // A name to refer back to this resource
                ,                 // _MTP, Nothing=>AddressRangeMemory
                ,                 // _TTP, Translation. Nothing=>TypeStatic
```

```
                                    // TypeTranslation: This resource, which is memory
                                    // on the secondary side of the bridge is I/O on
                                    // the primary side of the bridge.
                                    // TypeStatic: This resource, which is memory on
                                    // the secondary side of the bridge is also memory
                                    // on the primary side of the bridge.
             )
             }
             )

             // Returns the interface type
             Method(_IFT) {
                 Return(0x01) // MCTP KCS
             }

             // Returns the interface specification revision
             Method(_SRV) {
                 Return(0x0100)    // MCTP Base Specification Revision 1.0
             }

             // This interface does not support interrupt

}
```

# ANNEX A
(informative)

## Notation and Conventions

Examples of notations used in this document are as follows:

| | |
|---|---|
| • 2:N | In field descriptions, this will typically be used to represent a range of byte offsets starting from byte two and continuing to and including byte N. The lowest offset is on the left, the highest is on the right. |
| • (6) | Parentheses around a single number can be used in message field descriptions to indicate a byte field that may be present or absent. |
| • (3:6) | Parentheses around a field consisting of a range of bytes indicates the entire range may be present or absent. The lowest offset is on the left, the highest is on the right. |
| • PCIe | Underlined, blue text is typically used to indicate a reference to a document or specification called out in clause 2 or to items hyperlinked within the document. |
| • rsvd | Abbreviation for “reserved.” Case insensitive. |
| • [4] | Square brackets around a number are typically used to indicate a bit offset. Bit offsets are given as zero-based values (that is, the least significant bit [LSb] offset = 0). |
| • [7:5] | A range of bit offsets. The most significant bit is on the left, the least significant bit is on the right. |
| • `1b` | The lower case "b" following a number consisting of `0`s and `1`s is used to indicate the number is being given in binary format. |
| • `0x12A` | A leading "`0x`" is used to indicate a number given in hexadecimal format. |

## ANNEX B
(informative)

## Change Log

| Version | Date | Description |
|---|---|---|
| 1.0.0 | 2010-07-21 | Released as DMTF Standard |
|  |  |  |