



1
2
3
4

Document Number: DSP0237

Date: 28 July 2009

Version: 1.0.0

5 **Management Component Transport Protocol**
6 **(MCTP) SMBus/I2C Transport Binding**
7 **Specification**

8 **Document Type: Specification**
9 **Document Status: DMTF Standard**
10 **Document Language: E**
11

12 Copyright Notice

13 Copyright © 2009 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

14 DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems
15 management and interoperability. Members and non-members may reproduce DMTF specifications and
16 documents, provided that correct attribution is given. As DMTF specifications may be revised from time to
17 time, the particular version and release date should always be noted.

18 Implementation of certain elements of this standard or proposed standard may be subject to third party
19 patent rights, including provisional patent rights (herein "patent rights"). DMTF makes no representations
20 to users of the standard as to the existence of such rights, and is not responsible to recognize, disclose,
21 or identify any or all such third party patent right, owners or claimants, nor for any incomplete or
22 inaccurate identification or disclosure of such rights, owners or claimants. DMTF shall have no liability to
23 any party, in any manner or circumstance, under any legal theory whatsoever, for failure to recognize,
24 disclose, or identify any such third party patent rights, or for such party's reliance on the standard or
25 incorporation thereof in its product, protocols or testing procedures. DMTF shall have no liability to any
26 party implementing such standard, whether such implementation is foreseeable or not, nor to any patent
27 owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is
28 withdrawn or modified after publication, and shall be indemnified and held harmless by any party
29 implementing the standard from any and all claims of infringement by a patent owner for such
30 implementations.

31 For information about patents held by third-parties which have notified the DMTF that, in their opinion,
32 such patent may relate to or impact implementations of DMTF standards, visit
33 <http://www.dmtf.org/about/policies/disclosures.php>.

34 I²C is a trademark of Philips Semiconductors.

35 PCI-SIG, PCIe, and the PCI HOT PLUG design mark are registered trademarks or service marks of PCI-
36 SIG.

37 All other marks and brands are the property of their respective owners.

38

CONTENTS

40	Foreword	5
41	Introduction	6
42	1 Scope	7
43	2 Normative References.....	7
44	2.1 Approved References	7
45	2.2 Other References.....	7
46	3 Terms and Definitions	7
47	3.1 Requirement Term Definitions.....	7
48	3.2 MCTP Term Definitions.....	9
49	4 Symbols and Abbreviated Terms.....	15
50	5 Conventions	17
51	5.1 Reserved and Unassigned Values	17
52	5.2 Byte Ordering.....	18
53	6 MCTP over SMBus/I ² C Transport	18
54	6.1 Terminology	18
55	6.2 Transport Binding Use with I ² C.....	19
56	6.3 MCTP Packet Encapsulation	19
57	6.4 Bridges and Packet Formatting	20
58	6.5 MCTP Support Discovery	21
59	6.6 Support for Fixed-Address Devices.....	21
60	6.7 Supported Media.....	22
61	6.8 Physical Address Format for MCTP Control Messages	22
62	6.9 Get Endpoint ID Medium-Specific Information	22
63	6.10 Bus Owner Address.....	22
64	6.11 Bus Address Assignment.....	23
65	6.12 SMBus/I ² C Considerations for MCTP Messages	26
66	6.13 Fairness Arbitration.....	27
67	6.14 Fairness Arbitration Requirements for MCTP Bridges	28
68	6.15 Fairness Arbitration Requirements for Non-Bridge Endpoints.....	29
69	6.16 Fairness Arbitration Timing.....	31
70	6.17 MCTP Packet Timing Requirements	32
71	6.18 MCTP Control Message Timing Requirements.....	34
72	6.19 "Stuck 0" Condition Handling	35
73	6.20 MCTP over SMBus/I ² C Protocol Anti-Aliasing.....	36
74	6.21 Well-Known and Reserved Slave Addresses	37
75	6.22 Fixed Address Allocation	38
76	6.23 Recommended Address Range Allocation for Computer Systems.....	39
77	Annex A (informative) Notation	41
78	Annex B (informative) Change Log.....	42
79		

80 Figures

81	Figure 1 – MCTP over SMBus/I ² C Packet Format.....	19
82	Figure 2 – Address Assignment Flow	25
83	Figure 3 – Allowed Byte Range for First NACK'd Byte	29
84	Figure 4 – Fairness Arbitration Timing Measurement for SMBus and I ² C.....	31
85	Figure 5 – Example System Configuration	39

86

87 **Tables**

88	Table 1 – Packet Header Field Descriptions.....	19
89	Table 2 – Supported Media.....	22
90	Table 3 – Physical Address Format.....	22
91	Table 4 – Medium-Specific Information	22
92	Table 5 – Fairness Arbitration Timing Values for 100 kHz SMBus/I ² C.....	31
93	Table 6 – Fairness Arbitration Timing Values for 400 kHz I ² C	32
94	Table 7 – Timing Specifications for MCTP Packets on SMBus/I ² C.....	32
95	Table 8 – Timing Specifications for MCTP Control Messages on SMBus.....	34
96	Table 9 – Well-Known and Reserved Slave Addresses	37
97	Table 10 – Slave Address Allocation for Computer Systems	40
98		

Foreword

100 The *Management Component Transport Protocol (MCTP) SMBus/I2C Transport Binding Specification*
101 (DSP0237) was prepared by the PMCI Subgroup of the Pre-OS Working Group.

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

104

Introduction

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

- 107 • Management controllers and other management controllers
- 108 • Management controllers and management devices

109 The communication model includes a message format, transport description, message exchange
110 patterns, and configuration and initialization messages.

111 The *MCTP Base Specification* ([MCTP](#)) describes the protocol and commands used for communication
112 within and initialization of an MCTP network. Associated with the *MCTP Base Specification* are transport
113 binding specifications that define how the MCTP base protocol and MCTP control commands are
114 implemented on a particular physical transport type and medium, such as SMBus/I²C, PCI Express™
115 (PCIe) Vendor Defined Messaging, and so on.

116

117 Management Component Transport Protocol (MCTP) 118 SMBus/I2C Transport Binding Specification

119 1 Scope

120 This document provides the specifications for the Management Component Transport Protocol (MCTP)
121 transport binding for SMBus/I²C.

122 2 Normative References

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

126 2.1 Approved References

127 DMTF DSP0004, *CIM Infrastructure Specification 2.5*,
128 http://www.dmtf.org/standards/published_documents/DSP0004_2.5.pdf

129 DMTF DSP0136, *Alert Standard Format Specification 2.0*,
130 <http://www.dmtf.org/standards/documents/ASF/DSP0136.pdf>

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

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

135 2.2 Other References

136 IPMI Consortium, *Intelligent Platform Management Interface Specification, Second Generation 2.0*, 2006,
137 ftp://download.intel.com/design/servers/ipmi/IPMIv2_0rev1_0.pdf

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

140 Philips Semiconductors, *The I²C-Bus Specification v2.0*, I²C, December 1998
141 http://www.nxp.com/acrobat_download/literature/9398/39340011_20.pdf

142 SBS Implementers Forum, *System Management Bus (SMBus) Specification v2.0*, SMBus, August 2000,
143 <http://www.smbus.org/specs/smbus20.pdf>

144 3 Terms and Definitions

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

146 3.1 Requirement Term Definitions

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

- 148 **3.1.1**
149 **can**
150 used for statements of possibility and capability, whether material, physical, or causal
- 151 **3.1.2**
152 **cannot**
153 used for statements of possibility and capability, whether material, physical or causal
- 154 **3.1.3**
155 **conditional**
156 indicates requirements to be followed strictly to conform to the document when the specified conditions
157 are met
- 158 **3.1.4**
159 **deprecated**
160 indicates that an element or profile behavior has been outdated by newer constructs
- 161 **3.1.5**
162 **mandatory**
163 indicates requirements to be followed strictly to conform to the document and from which no deviation is
164 permitted
- 165 **3.1.6**
166 **may**
167 indicates a course of action permissible within the limits of the document
168 NOTE: An implementation that does *not* include a particular option shall be prepared to interoperate with another
169 implementation that *does* include the option, although perhaps with reduced functionality. An implementation that
170 *does* include a particular option shall be prepared to interoperate with another implementation that does *not* include
171 the option (except for the feature that the option provides).
- 172 **3.1.7**
173 **may not**
174 indicates flexibility of choice with no implied preference
- 175 **3.1.8**
176 **need not**
177 indicates a course of action permissible within the limits of the document
- 178 **3.1.9**
179 **not recommended**
180 indicates that valid reasons may exist in particular circumstances when the particular behavior is
181 acceptable or even useful, but the full implications should be understood and carefully weighed before
182 implementing any behavior described with this label
- 183 **3.1.10**
184 **obsolete**
185 indicates that an item was defined in prior specifications but has been removed from this specification

- 186 **3.1.11**
187 **optional**
188 indicates a course of action permissible within the limits of the document
- 189 **3.1.12**
190 **recommended**
191 indicates that valid reasons may exist in particular circumstances to ignore a particular item, but the full
192 implications should be understood and carefully weighed before choosing a different course
- 193 **3.1.13**
194 **required**
195 indicates that the item is an absolute requirement of the specification
- 196 **3.1.14**
197 **shall**
198 indicates requirements to be followed strictly to conform to the document and from which no deviation is
199 permitted
- 200 **3.1.15**
201 **shall not**
202 indicates requirements to be followed strictly to conform to the document and from which no deviation is
203 permitted
- 204 **3.1.16**
205 **should**
206 indicates that among several possibilities, one is recommended as particularly suitable, without
207 mentioning or excluding others, or that a certain course of action is preferred but not necessarily required
- 208 **3.1.17**
209 **should not**
210 indicates that a certain possibility or course of action is deprecated but not prohibited
- 211 **3.2 MCTP Term Definitions**
212 For the purposes of this document, the following terms and definitions apply.
- 213 **3.2.1**
214 **Address Resolution Protocol**
215 **ARP**
216 refers to the procedure used to dynamically determine the addresses of devices on a shared
217 communication medium
- 218 **3.2.2**
219 **Alert Standard Format**
220 **ASF**
221 alerting and remote control standard published by the DMTF

- 222 **3.2.3**
223 **baseline transmission unit**
224 indicates the required common denominator size of a transmission unit for packet payloads that are
225 carried in an MCTP packet
226 Baseline transmission unit-sized packets are guaranteed to be routable within an MCTP network.
- 227 **3.2.4**
228 **baseboard management controller**
229 **BMC**
230 a term coined by the IPMI specifications for the main management controller in an IPMI-based platform
231 management subsystem
232 This term is also sometimes used as a generic name for a motherboard resident management controller
233 that provides motherboard-specific hardware monitoring and control functions for the platform
234 management subsystem.
- 235 **3.2.5**
236 **bridge**
237 the circuitry and logic that connects one computer bus or interconnect to another, allowing an agent on
238 one to access the other
239 Within this document, the term *bridge* shall refer to MCTP bridge, unless otherwise indicated.
- 240 **3.2.6**
241 **bus**
242 a physical addressing domain shared between one or more platform components that share a common
243 physical layer address space
- 244 **3.2.7**
245 **bus owner**
246 the party responsible for managing address assignments (can be logical or physical addresses) on a bus
247 (for example, in MCTP, the bus owner is the party responsible for managing endpoint ID assignments for
248 a given bus)
249 A bus owner may also have additional media-specific responsibilities, such as assignment of physical
250 addresses.
- 251 **3.2.8**
252 **byte**
253 an 8-bit quantity. Also referred to as an *octet*.
254 NOTE: PMCI specifications shall use the term *byte*, not *octet*.
- 255 **3.2.9**
256 **endpoint**
257 see [MCTP endpoint](#)
- 258 **3.2.10**
259 **endpoint ID**
260 **EID**
261 see [MCTP endpoint ID](#)

- 262 **3.2.11**
263 **Inter-Integrated Circuit**
264 **I²C**
265 a multi-master, two-wire, serial bus originally developed by Philips Semiconductor
- 266 **3.2.12**
267 **Intelligent Platform Management Bus**
268 **IPMB**
269 the name for the architecture, protocol, and implementation of an I²C bus that provides a communications
270 path between management controllers in IPMI-based systems
- 271 **3.2.13**
272 **Intelligent Platform Management Interface**
273 **IPMI**
274 a set of specifications defining interfaces and protocols originally developed for server platform
275 management by the IPMI Promoters Group: Intel, Dell, HP, and NEC
- 276 **3.2.14**
277 **managed entity**
278 the physical or logical entity that is being managed by management parameters
279 Examples of *physical* entities include fans, processors, power supplies, circuit cards, chassis, and so on.
280 Examples of *logical* entities include virtual processors, cooling domains, system security states, and so
281 on.
- 282 **3.2.15**
283 **management controller**
284 a microcontroller or processor that aggregates management parameters from one or more management
285 devices and makes access to those parameters available to local or remote software, or to other
286 management controllers, through one or more management data models
287 Management controllers may also interpret and process management-related data, and initiate
288 management-related actions on management devices. While a native data model is defined for PMCI, it is
289 designed to be capable of supporting other data models, such as CIM, IPMI, and vendor-specific data
290 models. The microcontroller or processor that serves as a management controller can also incorporate
291 the functions of a management device.
- 292 **3.2.16**
293 **management device**
294 any physical device that provides protocol terminus for accessing one or more management parameters.
295 A management device responds to management requests, but does not initiate or aggregate
296 management operations except in conjunction with a management controller (that is, it is a *satellite*
297 device that is subsidiary to one or more management controllers). An example of a simple management
298 device would be a temperature sensor chip. A management controller that has I/O pins or built-in analog-
299 to-digital converters that monitor state and voltages for a managed entity would also be a management
300 device.
- 301 **3.2.17**
302 **management parameter**
303 a particular datum representing a characteristic, capability, status, or control point associated with a
304 managed entity. Example management parameters include temperature, speed, volts, on/off, link state,
305 uncorrectable error count, device power state, and so on.

306 3.2.18**307 Management Component Transport Protocol****308 MCTP**

309 a transport protocol for intercommunication between management controllers and management devices.

310 3.2.19**311 MCTP bridge**

312 an MCTP endpoint that can route MCTP messages (not destined for itself) that it receives on one
313 interconnect onto another without interpreting them

314 The ingress and egress media at the bridge may be either homogeneous or heterogeneous. Also referred
315 to in this document as a "bridge".

316 3.2.20**317 MCTP bus owner**

318 responsible for endpoint ID assignment for MCTP or translation on the buses of which it is a master

319 The MCTP bus owner may also be responsible for physical address assignment. For example, for SMBus
320 bus segments, the MCTP bus owner is also the ARP master. This means the bus owner assigns dynamic
321 SMBus addresses to those devices requiring it.

322 3.2.21**323 MCTP control command**

324 commands (defined under the MCTP *control* message type) that are used for the initialization and
325 management of MCTP communications (for example, commands to assign endpoint IDs, discover device
326 MCTP capabilities, and so on).

327 3.2.22**328 MCTP endpoint**

329 an MCTP communication terminus

330 An MCTP endpoint is a terminus or origin of MCTP packets or messages (that is, the combined
331 functionality within a physical device that communicates using the MCTP transport protocol and handles
332 MCTP control commands). This includes MCTP-capable management controllers and management
333 devices. Also referred to in this document as an "endpoint".

334 3.2.23**335 MCTP endpoint ID**

336 the logical address used to route MCTP messages to a specific MCTP endpoint

337 A numeric handle (logical address) that uniquely identifies a particular MCTP endpoint within a system for
338 MCTP communication and message routing purposes. Endpoint IDs are unique among MCTP endpoints
339 that comprise an MCTP communication network within a system. MCTP endpoint IDs are only unique
340 within a particular MCTP network. That is, they can be duplicated or overlap from one MCTP network to
341 the next. Also referred to in this document as "endpoint ID" and abbreviated "EID".

342 3.2.24**343 MCTP management controller**

344 a management controller that is an MCTP endpoint

345 Unless otherwise indicated, the term "management controller" refers to an "MCTP management
346 controller" in this document.

347 3.2.25**348 MCTP management device**

349 a management device that is an MCTP endpoint

- 350 Unless otherwise indicated, the term "management device" refers to an "MCTP management device" in
351 this document.
- 352 **3.2.26**
353 **MCTP message**
354 a unit of communication based on the message type that is relayed through the MCTP Network using one
355 or more MCTP packets
- 356 **3.2.27**
357 **MCTP network**
358 a collection of MCTP endpoints that communicate using MCTP and share a common MCTP endpoint ID
359 space
- 360 **3.2.28**
361 **MCTP packet**
362 the unit of data transfer used for MCTP communication on a given physical medium
- 363 **3.2.29**
364 **MCTP packet payload**
365 refers to the portion of the message body of an MCTP message that is carried in a single MCTP packet
- 366 **3.2.30**
367 **message**
368 see [MCTP message](#)
- 369 **3.2.31**
370 **message assembly**
371 the process of receiving and linking together two or more MCTP packets that belong to a given MCTP
372 message to allow the entire message header and message data (payload) to be extracted
- 373 **3.2.32**
374 **message body**
375 the portion of an MCTP message that carries the message type field and any message type-specific data
376 associated with the message
377 An MCTP message spans multiple MCTP packets when the message body needs is larger than what can
378 fit in a single MCTP packet. Thus, the message body portion of an MCTP message can span multiple
379 MCTP packets.
- 380 **3.2.33**
381 **message disassembly**
382 the process of taking an MCTP message where the message's header and data (payload) cannot be
383 carried in a single MCTP packet, and generating the sequence of two or more packets required to deliver
384 that message content within the MCTP network
- 385 **3.2.34**
386 **message originator**
387 the original transmitter (source) of a message targeted to a particular message terminus

- 388 **3.2.35**
389 **message terminus**
390 the name for a triplet of fields consisting of fields called the MCTP Source Endpoint ID, Tag Owner bit
391 value, and Message Tag value
392 Together these fields identify the packets for an MCTP message within an MCTP network for the purpose
393 of message assembly. The message terminus itself can be thought of as identifying a set of resources
394 within the recipient endpoint that is handling the assembly of a particular message.
- 395 **3.2.36**
396 **most significant byte**
397 **MSB**
398 refers to the highest order byte in a number consisting of multiple bytes
- 399 **3.2.37**
400 **nibble**
401 a four-bit aggregation, or half of a byte
- 402 **3.2.38**
403 **packet**
404 see [MCTP packet](#)
- 405 **3.2.39**
406 **packet payload**
407 see [MCTP packet payload](#)
- 408 **3.2.40**
409 **payload**
410 the information-bearing fields of a message
411 This is separate from those fields and elements that are used to transport the message from one point to
412 another, such as address fields, framing bits, checksums, and so on. In some instances, a given field may
413 be both a payload field and a transport field.
- 414 **3.2.41**
415 **physical transport binding**
416 refers to specifications that define how the MCTP Base Protocol and MCTP control commands are
417 implemented on a particular physical transport type and medium, such as SMBus/I²C, PCI Express™
418 (PCIe) Vendor Defined Messaging, and so on
- 419 **3.2.42**
420 **point-to-point**
421 refers to the case where only two physical communication devices are interconnected through a physical
422 communication medium. The devices may be in a master/slave relationship, or could be peers
- 423 **3.2.43**
424 **Platform Management Component Intercommunications**
425 **PMCI**
426 name for a working group under the Distributed Management Task Force's Pre-OS Working Group that is
427 chartered to define standardized communication protocols, low-level data models, and transport
428 definitions that support communications with and between management controllers and management
429 devices that form a platform management subsystem within a managed computer system

430 **3.2.44**
431 **simple endpoint**
432 an MCTP endpoint that is not associated with either the functions of an MCTP bus owner or an MCTP
433 bridge

434 **3.2.45**
435 **transport binding**
436 see [physical transport binding](#)

437 **3.2.46**
438 **transmission unit**
439 for MCTP, this refers to the size of the portion of the MCTP packet payload, which is the portion of the
440 message body carried in an MCTP packet

441 **4 Symbols and Abbreviated Terms**

442 The following symbols and abbreviations are used in this document.

443 **4.1**
444 **ACK**
445 acknowledge

446 **4.2**
447 **ARP**
448 Address Resolution Protocol

449 **4.3**
450 **ASF**
451 Alert Standard Format

452 **4.4**
453 **BMC**
454 baseboard management controller

455 **4.5**
456 **EEPROM**
457 Electrically Erasable Programmable Read-Only Memory

458 **4.6**
459 **EID**
460 endpoint identifier

461 **4.7**
462 **I²C**
463 Inter-Integrated Circuit

464 **4.8**
465 **I/O**
466 input/output
467

468	IPMB
469	Intelligent Platform Management Bus
470	4.9
471	IPMI
472	Intelligent Platform Management Interface
473	4.10
474	kHz
475	kilohertz
476	4.11
477	LSb
478	least significant bit
479	4.12
480	LSB
481	least significant byte
482	4.13
483	max
484	maximum
485	4.14
486	MCTP
487	Management Component Transport Protocol
488	4.15
489	min
490	minimum
491	4.16
492	ms
493	millisecond
494	4.17
495	MSB
496	most significant byte
497	4.18
498	MTU
499	Maximum Transmission Unit
500	4.19
501	NACK
502	not acknowledge
503	4.20
504	PCI
505	peripheral component interconnect

- 506 **4.21**
507 **PCIe®**
508 PCI Express™
- 509 **4.22**
510 **PEC**
511 packet error code
- 512 **4.23**
513 **PMCI**
514 Platform Management Component Intercommunications
- 515 **4.24**
516 **PSA**
517 persistent slave address
- 518 **4.25**
519 **rsvd**
520 reserved (not case sensitive)
- 521 **4.26**
522 **SCL**
523 serial clock
- 524 **4.27**
525 **SDA**
526 serial data
- 527 **4.28**
528 **sec**
529 second
- 530 **4.29**
531 **SEEPROM**
532 serial EEPROM
- 533 **4.30**
534 **SMBus**
535 System Management Bus
- 536 **4.31**
537 **UDID**
538 unique device identifier

539 **5 Conventions**

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

541 **5.1 Reserved and Unassigned Values**

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

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

546 5.2 Byte Ordering

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

549 6 MCTP over SMBus/I²C Transport

550 The MCTP over SMBus/I²C transport binding defines how MCTP packets are delivered over a physical
551 SMBus or I²C medium using SMBus transactions. This includes how physical addresses are used, how
552 fixed addresses are accommodated, how physical address assignment is accomplished for hot-plug or
553 other devices that require dynamic physical address assignment, and how MCTP support is discovered.
554 Timing specifications for bus and MCTP control operations are also given, and a "fairness" protocol is
555 defined for the purpose of avoiding deadlock and starvation/lockout situations among MCTP endpoints.

556 The binding has been designed to be able to share the same bus as devices communicating using earlier
557 SMBus/I²C management protocols such as Alert Standard Format (ASF) and IPMI, and with vendor-
558 specific devices using SMBus/I²C protocols. The specifications can also allow a given device to
559 incorporate non-MCTP SMBus functions alongside MCTP. This is described in more detail in 6.20.

560 6.1 Terminology

561 According to SMBus, SMBus devices are categorized as follows, where Address Resolution Protocol
562 (ARP) refers to the SMBus Address Resolution Protocol (a dynamic slave address assignment protocol)
563 and UDID refers to a "unique device identifier", a 128-bit value that a device uses during the ARP process
564 to uniquely identify itself. Because these protocols are implemented with command transactions that are
565 run on top of the SMBus physical specification, it is possible to use these protocols on devices that
566 support an I²C physical interface.

567 • ARP-capable

568 [SMBus](#) term indicating a device that supports all [SMBus](#) ARP commands with the exception of
569 the optional Host Notify command. The slave address is assignable. The device supports both
570 Reset commands.

571 • Fixed and Discoverable

572 [SMBus](#) term indicating a device supports the Prepare to ARP, directed Get UDID, general Get
573 UDID, and Assign Address commands. The slave address is fixed; the device will accept the
574 Assign Address command but will not allow address reassignment. The device supports both
575 Reset commands.

576 • Fixed - Not Discoverable

577 [SMBus](#) term indicating a device supports the directed Get UDID command. The slave address
578 is fixed.

579 • Non-ARP-capable

580 [SMBus](#) term indicating a device does not support any ARP commands. The slave address is
581 fixed.

582 • **Fixed Address**

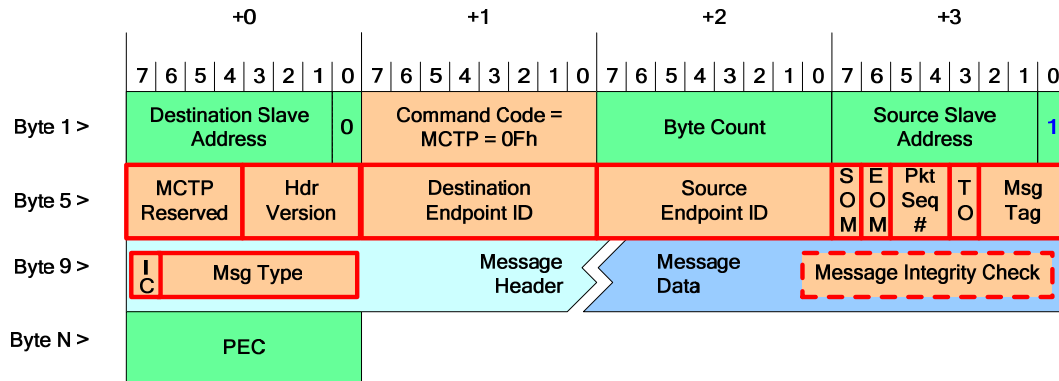
583 For this specification, this term is be used to refer to any device that uses a fixed slave address,
 584 without distinguishing whether it is "Fixed and Discoverable", "Fixed, not Discoverable", or
 585 "Non-ARP-capable".

586 **6.2 Transport Binding Use with I²C**

587 The transport binding defined in this specification has also been designed to be able to work with
 588 standard-mode and fast-mode (400 kHz) I²C buses that use 7-bit addressing; 10-bit addressing is not
 589 supported. This binding has not been specified for use with high-speed I²C specifications.

590 **6.3 MCTP Packet Encapsulation**

591 All MCTP transactions are based on the SMBus Block Write bus protocol. The first 8 bytes make up the
 592 packet header. The first three fields—Destination Slave Address, Command Code, and Length—map
 593 directly to SMBus functional fields. The remaining header and payload fields map to SMBus Block Write
 594 "Data Byte" fields, as indicated in Figure 1. Hence, the inclusion of the Source Slave Address in the
 595 header is specified by [MCTP](#) rather than [SMBus](#). This is done to facilitate addressing required for
 596 establishing communications back to the message originator.



597

598 **Figure 1 – MCTP over SMBus/I²C Packet Format**

599 **Table 1 – Packet Header Field Descriptions**

Byte	Block Write Field(s)	Description
1	Slave Address Wr	[7:1] SMBus Destination Slave Address: The slave address of the target device for the local SMBus link [0]: SMBus R/W# bit: Shall be set to 0b as all MCTP messages use SMBus write transactions.
2	Command Code	Command Code: SMBus Command Code All MCTP over SMBus messages use a command code of 0x0F.

Byte	Block Write Field(s)	Description
3	Byte Count	<p>Byte Count: Byte count for the SMBus Block Write protocol transaction that is carrying the MCTP packet content.</p> <p>This value is the count of bytes that follow the Byte Count field up to, but not including, the PEC byte. For example, if the MCTP packet payload length (starting with byte 9) is 64 bytes, the value in the Byte Count field would be 69. (The count of 69 accounts for 64 bytes of MCTP packet payload plus the five bytes [bytes 4 through 8, inclusive] that comprise the bytes of the SMBus-specific header and MCTP header that follow the Byte Count field.)</p>
4	Data Byte 1	<p>SMBus Source Slave address</p> <p>[7:1] : For the local SMBus link, the slave address of the source device.</p> <p>[0]: This bit shall be set to 1b. The value enables MCTP to be differentiated from IPMI over SMBus and IPMB (IPMI over I²C) protocols.</p>
5	Data Byte 2	<p>[7:4] MCTP reserved: This nibble is reserved for definition by the MCTP Base Specification.</p> <p>[3:0] MCTP header version:</p> <p>Set to 0001b for MCTP devices that are conformant to the MCTP Base Specification 1.0 and this version of the SMBus transport binding.</p> <p>All other values = Reserved.</p>
6	Data Byte 3	Destination endpoint ID (*)
7	Data Byte 4	Source endpoint ID (*)
8	Data Byte 5	<p>[7] SOM: Start Of Message flag (*)</p> <p>[6] EOM: End Of Message flag (*)</p> <p>[5:4] Packet sequence number (*)</p> <p>[3] Tag Owner (TO) bit (*)</p> <p>[2:0] Message tag (*)</p>
9	Data Byte 6	<p>[7] IC: Integrity Check bit (*)</p> <p>[6:0] Message type (*)</p>
10:N-1	Data Bytes 7:M	Message header and data (*)
N	PEC	<p>Packet error code (PEC): The PEC as defined in the SMBus 2.0 Specification. All MCTP transactions shall include a PEC byte. The PEC byte shall be transmitted by the source and checked by the destination.</p>

(*) Indicates a field that is defined by the [MCTP Base Specification](#).

600 6.4 Bridges and Packet Formatting

601 As an MCTP packet travels through a bridge from one SMBus/I²C port to another, the bridge leaves all
602 packet header and message header and data fields alone with the exception of the source and
603 destination slave address, which shall be modified to route across the intended bus/link. When an MCTP
604 bridge forwards a message from an input port to an output port, it replaces the destination slave address
605 with the targeted slave on the destination bus, and replaces the source slave address with the bridge's
606 slave address.

607 The MCTP SMBus/I²C bridge shall re-calculate the PEC byte to account for changes in the source and
608 destination slave address fields.

609 A similar process is used when bridging between different media. The physical addressing and header
610 information gets changed by the bridge to match the requirements of the target bus, and any packet-level
611 integrity check information is also updated.

612 **6.5 MCTP Support Discovery**

613 All SMBus devices that support an MCTP endpoint and the SMBus Get UDID command for a particular
614 SMBus/I²C interface (that is, devices with ARP-capable, fixed and discoverable, or fixed-not discoverable
615 interfaces) are required to have their MCTP support discoverable through the Get UDID command. To do
616 this, endpoints shall return a value of 1b in bit 5 (the ASF bit) in the Interface field in the Get UDID
617 command.

618 Once support for ASF has been indicated, an MCTP control message (for example, Get MCTP Version
619 Support) can be issued to the device to determine whether it supports MCTP. The SMBus command byte
620 for MCTP packets uses a value that has been allocated by the DMTF for MCTP use and does not overlap
621 values used for ASF. This enables older devices that indicate ASF support to be queried for MCTP
622 support without conflict. This is described in more detail in 6.6. Devices that do not support the Get UDID
623 command will need to have their support for MCTP configured into the bus owner as described in 6.6.

624 I²C devices can also support the SMBus protocols and commands for being an ARP-able device that is
625 also discoverable as an MCTP device. This is required for hot-plug I²C devices using MCTP.

626 **6.6 Support for Fixed-Address Devices**

627 MCTP bus owners shall include non-volatile options to record the addresses used by fixed-address
628 devices on SMBus/I²C buses that they own, and which of those devices support MCTP.

629 For non-MCTP devices, the MCTP bus owner needs this information to know which fixed addresses to
630 avoid when performing SMBus ARP for the bus. (Alternatively, the bus owner could be configured with a
631 range of SMBus slave addresses that the bus owner is allowed to allocate from.)

632 For MCTP devices, the bus owner needs this information to perform EID assignment and, if the bus
633 owner is also an MCTP bridge, routing table initialization and operation.

634 For fixed-address MCTP devices that do not support the Get UDID command (that is, non-ARP-capable
635 devices), the bus owner needs to also be configured with information that identifies the device as
636 supporting MCTP.

637 For fixed-address devices that support the [SMBus](#) Get UDID command (that is, devices with ARP-
638 capable, Fixed and Discoverable, or Fixed-Not Discoverable SMBus interfaces) the bus owner can either
639 discover whether the device supports MCTP by using the discovery approach described in 6.5, or it could
640 have this information configured at the same time that the slave address information for the fixed-address
641 device is provided.

642 It is recommended that general-purpose devices that act as MCTP bus owners allow being configured to
643 support at least 16 different fixed-address devices for each SMBus/I²C bus they own. This number would
644 include both MCTP and non-MCTP devices.

645 6.7 Supported Media

646 This physical transport binding has been designed to work with the media specified in Table 2. Use of this
647 binding with other types of physical media is not covered by this specification.

648 **Table 2 – Supported Media**

Physical Media Identifier	Description
0x01	SMBus 100 kHz compatible
0x02	SMBus + I²C 100 kHz compatible
0x03	I²C 100 kHz compatible
0x04	I²C 400 kHz compatible

649 6.8 Physical Address Format for MCTP Control Messages

650 The address format shown in Table 3 is used for MCTP control commands that require a physical
651 address parameter to be returned for a bus that uses this transport binding with one of the supported
652 media types listed in 6.7. This includes commands such as the Resolve Endpoint ID, Routing Information
653 Update, and Get Routing Table Entries commands.

654 **Table 3 – Physical Address Format**

Format Size	Layout and Description
1 byte	[7:1] slave address bits [0] 0b

655 6.9 Get Endpoint ID Medium-Specific Information

656 The definition shown in Table 4 is used for the medium-specific Information field returned in the response
657 to the Get Endpoint ID MCTP control message.

658 **Table 4 – Medium-Specific Information**

Description	
[7:1]	reserved
[0]	fairness arbitration support (see 6.13)
	0b = not supported
	1b = supported

659 6.10 Bus Owner Address

660 In order to be the target of the SMBus Notify ARP Master protocol transaction the MCTP bus owner shall
661 be configurable to be accessed at the SMBus host slave address. This configuration does not need to be
662 used if the bus implementation does not include any MCTP devices that require dynamic address
663 assignment of their slave address. For more information, see 6.11.4.

664 The bus owner may use a different, second slave address for all other MCTP communication functions.

665 **6.11 Bus Address Assignment**

666 This clause describes the configuration, setup, and operation of communication between MCTP
667 endpoints using SMBus/I²C as the communication medium.

668 **6.11.1 Slave Addresses**

669 Each device on SMBus/I²C shall have a slave address to be the target of transactions by bus masters.
670 The MCTP transport protocol solely utilizes Master Write transactions to transfer MCTP packets between
671 MCTP endpoints. For endpoint "A" to send an MCTP packet to endpoint "B", endpoint A shall master the
672 bus and issue a Block-Write transaction to the slave address of endpoint B. Similarly, for endpoint B to
673 send an MCTP packet to endpoint A, it shall master the bus and issue a Block-Write transaction to the
674 slave address of endpoint A. Thus, bi-directional transfer of MCTP packets requires that both sides of the
675 communication have slave addresses.

676 Device support for slave addresses can be of two general types: fixed or assignable. Devices with
677 assignable addresses (also referred to as "ARP-capable" or "ARP-able") can use the [SMBus](#) ARP. The
678 entity that assigns slave addresses to ARP-able devices is referred to as the "ARP master".

679 A bus can include a mix of fixed-address and ARP-able devices. Most fixed-address devices do not
680 include a discovery mechanism, and neither [SMBus](#) nor [I²C](#) require one. Therefore, for a generic bus
681 implementation that support ARP-able devices (such as SMBus to PCI/PCIe connectors) the ARP master
682 needs to know what ranges of addresses are being used for fixed-address devices so that it doesn't give
683 an ARP-able device an address that conflicts with a fixed-address device.

684 This transport binding allows for non-MCTP devices (both fixed address and ARP-able) to reside on the
685 same bus segment used for MCTP devices. The use and assignment of slave addresses shall therefore
686 be compatible with pre-existing devices. To accomplish this, the following approach is used for managing
687 devices on a bus that supports MCTP.

688 **6.11.2 Well Known and Reserved Slave Addresses**

689 The [SMBus](#) and [I²C](#) specifications define certain slave addresses that should either be avoided by
690 devices or are reserved (not to be used as a general device slave address) because those addresses are
691 related to functions that are used by MCTP. These addresses are listed in Table 9.

692 **6.11.3 Fixed Address Recommendations for Device Manufacturers**

693 MCTP may be used within a typical computer system application where the motherboard/baseboard may
694 come from one supplier, the chassis from another supplier, and possibly add-in modules from yet
695 another.

696 Referring to Table 10, it is thus recommended that devices that use fixed addresses and are targeted for
697 uses that can include baseboard (B), chassis/system (C), and add-in (A) applications are configurable to
698 cover for at least three different "B" addresses, at least three different "C" addresses, and at least two
699 different "A" addresses to help avoid address conflicts in those applications.

700 **6.11.4 Dynamic Address Assignment (SMBus ARP) Support**

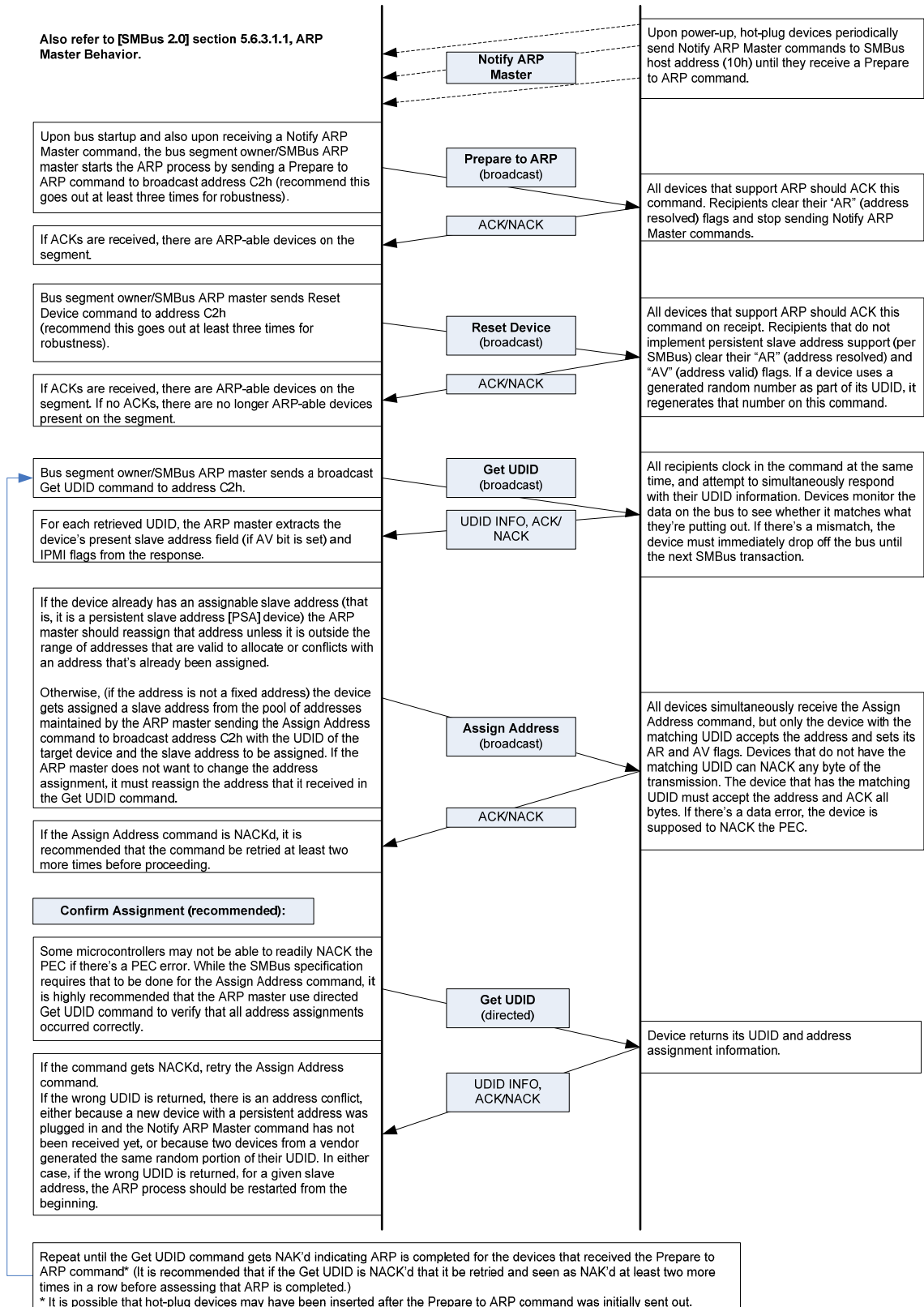
701 MCTP buses that support connections to standard PCI/PCIe add-in cards are required by the PCI
702 specifications to support SMBus ARP (be ARP-capable) to allow the devices to be dynamically assigned
703 addresses to avoid address conflicts and eliminate the need for manual configuration of addresses.
704 Figure 2 presents an overview of the address assignment process.

705 **6.11.5 Devices Supporting Multiple Interfaces**

706 Devices that support multiple, separate [SMBus](#) or [I²C](#) interfaces where the interfaces are intended to be
707 connected to the same bus shall meet the following requirements:

- 708 • The interfaces shall be either be ARP-capable or be fixed-address interfaces that are configured
709 to use a different slave address for each interface.
- 710 • If the interfaces support SMBus ARP, (as either ARP-able or ARP-enumerable devices) a
711 different SMBus UDID shall be used for each SMBus ARP-able interface.

712 NOTE: Devices that have internal hardware interfaces that may be implemented as separate blocks but are
713 designed to share a slave address are not considered to have separate interfaces in this context.



714

715

Figure 2 – Address Assignment Flow

716 **6.11.6 MCTP Requirements on SMBus ARP Master Support**

717 If the bus supports ARP-able devices, MCTP requires that each bus shall have a controller that operates
718 as the ARP master and assigns slave addresses to all ARP-able devices on the segment. Because the
719 MCTP bus owner shall know the physical addresses of ARP-able devices that support MCTP, the ARP
720 master role will typically be handled by the same device that serves as the MCTP bus owner.

721 If a different physical device than the device holding the bus owner functions as the ARP master, there
722 shall be a mechanism to communicate the address assignment information to the bus owner function.
723 The mechanism for this is not specified by MCTP.

724 Only one controller is allowed to function as the ARP master for the segment at a given time. The ARP
725 master function is allowed to *fail-over* or be transferred to another controller. The mechanism for this
726 capability, if provided, is not specified by MCTP.

727 **6.11.7 Recommendations on ARP Master Allocation of Slave Addresses**

728 For PCI and PCI Express™ (PCIe) bus implementations, it is recommended that, by default, the ARP
729 master only assigns addresses to ARP-able devices from the "B" range. This is because the PCI
730 slots/connectors themselves are most commonly implemented as part of the board set.

731 Device manufacturers of controllers that function as ARP masters should provide a mechanism to enable
732 system integrators to either configure which fixed addresses that ARP should avoid, or a pool of non-
733 conflicting addresses from which ARP can draw.

734 For PCI and PCIe SMBus implementations, the ARP master should be able to assign at least two
735 addresses for each PCI connector on the segment.

736 **6.11.8 MCTP Requirements on Hot-Pluggable Bridges Using SMBus**

737 Hot-pluggable MCTP devices that include bridging functionality are required to have static, pre-assigned,
738 SMBus UDIDs. This is because it is considered a more robust and reliable mechanism than randomly
739 generated UDIDs, and because it simplifies tracking and managing MCTP device hot-add and hot-
740 removal.

741 If devices regenerate their UDIDs on hot-plug, the MCTP bus owner/ARP master cannot rely on the UDID
742 to determine whether a device was newly added to the system. When a hot-plug device includes MCTP
743 bridging functionality, the bus owner shall be able to allocate the device a range of EIDs from a fixed pool
744 of IDs. Thus, it is important for the bus owner to be able to determine which devices have been removed
745 so that any EIDs it had given out can be returned to the pool.

746 It is straightforward for the ARP master to re-enumerate the UDIDs on the bus and determine which
747 UDIDs (if any) are no longer present (re-enumeration is a natural fallout of the ARP process). If there are
748 MCTP devices without fixed UDIDs in the mix, however, the bus owner would need to take additional
749 steps to check to see which devices had already been allocated EIDs to determine by elimination which
750 ranges, if any, had become freed. With fixed UDID, the bus owner can track which EIDs have been
751 allocated to which UDIDs and thereby determine which have been freed by a hot swap by just re-
752 enumerating the UDIDs.

753 **6.12 SMBus/I²C Considerations for MCTP Messages**

754 The following applies to MCTP messages on SMBus regardless of their message type. Note that MCTP
755 messages require Block Write byte count sizes that exceed limits specified by [SMBus](#). Additional
756 restrictions on MCTP packets over what the [SMBus](#) and [I²C](#) allow are given in 6.3 and 6.17.

757 6.12.1 Slave Address ACKs/NACKs

758 Per [SMBus](#) and [I²C](#), the NACK of a slave address indicates the physical absence of the device interface.

- 759 • Devices are therefore required to always ACK their slave addresses. This includes ACK'ing
760 slave addresses used for ARP if the device is ARP-able or ARP-enumerable.
- 761 • An MCTP device shall ACK its slave address(es) when the R/W bit on the slave address is 0.

762 6.12.2 Clock Stretching for Non-Addressed Devices

763 MCTP devices that are monitoring the bus as slaves and do not have a slave address that matches the
764 transaction shall not clock stretch past the ACK bit for the slave address byte. This requirement only
765 applies to MCTP packet transactions. It does not apply to non-MCTP-defined messages or transactions,
766 such as those used for SMBus ARP.

767 6.13 Fairness Arbitration

768 6.13.1 General

769 The following clauses describe an extension to the SMBus/I²C arbitration mechanism for device ports that
770 are used with MCTP. The extensions define a 'fairness' mechanism that helps ensure that ports that are
771 arbitrating for access to the bus will eventually get access and will not be locked out of access by other
772 MCTP ports that are using the bus.

773 NOTE: Fairness arbitration only applies for messages using the MCTP base protocol. SMBus messages such as
774 Host Notify are not required to use fairness arbitration.

775 This mechanism works as follows:

- 776 • An MCTP port that wins bus arbitration (per [SMBus](#) or [I²C](#)) for a given transaction shall wait
777 until it detects a particular bus idle interval before the device can again attempt to arbitrate for
778 the bus. This is referred to as the device waiting to detect the "FAIR_IDLE" condition.
- 779 • Once the port has succeeded in detecting the FAIR_IDLE condition, it can attempt to get on the
780 bus and no longer needs to wait to detect the FAIR_IDLE condition. The port can continue to
781 attempt to access the bus without waiting for FAIR_IDLE until the next time the port wins
782 arbitration. After winning arbitration, the port shall again wait to detect the FAIR_IDLE condition
783 before it can attempt to get on the bus.

784 With this approach, all ports that lose arbitration will eventually get a turn at accessing the bus, because
785 any ports that win arbitration will need to wait until a bus idle interval is detected, while those that have
786 lost arbitration will not need to wait.

787 For this to work, endpoints shall be able to do two things:

- 788 1) Be able to recognize the FAIR_IDLE condition. Ports that are waiting to detect a FAIR_IDLE
789 condition shall recognize that no other port has made the bus become busy within a particular
790 window of time (T_{IDLE_WINDOW}) after the bus becomes free.
- 791 2) Ports that have not won arbitration shall be able to issue a START condition soon enough after
792 the bus becomes free so that a bus busy condition is seen by ports that are waiting to detect a
793 FAIR_IDLE condition. To ensure this condition is met, START shall be issued by the port within
794 a particular window of time (T_{START_WINDOW}) after the bus becomes free.

795 NOTE: There is actually no explicit indication in [SMBus](#) or [I²C](#) that arbitration has been won. Instead, what the
796 master detects is that it was able to access the bus and did not have a collision (lose arbitration) with another master.
797 For this specification, this is referred to as *winning arbitration*. Because of the way arbitration works, an MCTP
798 endpoint that is transmitting as a master onto the bus will know that it has won arbitration if it is able to transmit from
799 the destination slave address byte through the end of the source slave address byte (byte 4) without receiving a
800 collision or NACK.

801 **6.13.2 Deadlock Avoidance with Fairness Arbitration**

802 A device that wins arbitration but is subsequently NACK'd for its write transaction shall return to waiting
803 for the FAIR_IDLE period before it can attempt the transaction again.

804 **6.13.3 Fairness Arbitration Support**

805 Bridges and endpoints should support fairness arbitration. An endpoint's support for fairness arbitration
806 shall be reported through the medium-specific Information field in the response to the Get Endpoint ID
807 MCTP control message.

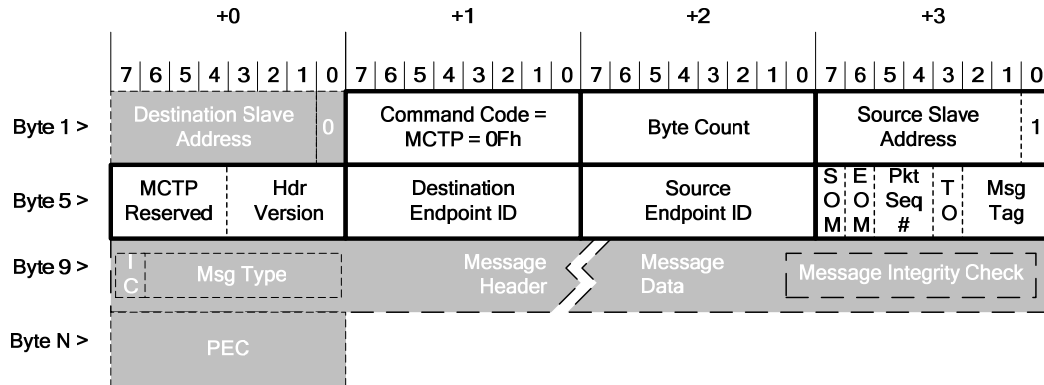
808 **6.13.4 Bus Busy Sampling Requirements for Fairness Arbitration**

809 It is atypical and unlikely that the bus will go busy and then free again within T_{IDLE_WINDOW} . This is because
810 T_{IDLE_WINDOW} is shorter than the time required to send one byte on the bus. Thus, this condition would only
811 occur on an error or under a usage of the bus that is not legal within the specifications. Therefore, an
812 implementation is not required to continuously check the bus busy status during the entire duration of
813 T_{IDLE_WINDOW} (though this is recommended). An implementation is allowed to check the bus busy status
814 only at the conclusion of the T_{IDLE_WINDOW} interval that is measured by the device.

815 **6.14 Fairness Arbitration Requirements for MCTP Bridges**

816 MCTP bridges that support fairness arbitration shall meet the following requirements:

- 817 • The bridge shall support FAIR_IDLE detection and implement the corresponding fairness policy
818 separately for each port on the bridge.
- 819 • Upon device power up or initialization, a port does not need to detect a FAIR_IDLE condition
820 before first attempting to access the bus.
- 821 • A bridge that loses arbitration when attempting to transmit shall continue to retry the transaction
822 when the bus becomes free for up to PN2 retries (see Table 7). If the retry limit is reached, the
823 bridge shall drop the packet data.
- 824 • A bridge that receives a NACK when attempting to transmit to a given physical address shall
825 continue to retry the transaction when the bus becomes free for up to PN2 retries. The bridge
826 will return to attempting to arbitrate for the bus as described in the preceding requirement,
827 restarting its number of arbitration retries. If the retry limit is reached, the bridge shall drop the
828 packet data.
- 829 • An MCTP bridge shall provide dedicated input buffer space per port. The minimum input buffer
830 size is large enough to store one full baseline MTU-sized MCTP packet. It is recommended, but
831 not required, that a bridge also implement a dedicated output buffer per port, sized to store at
832 least one full baseline MTU-sized MCTP packet.
- 833 • If the MCTP bridge is the target of an MCTP packet and it does not have enough buffer space in
834 its input buffer to store the full packet, it shall NACK the packet. If the bridge has an output
835 packet to transmit on that same port, it shall be able to issue a START within T_{START_WINDOW} after
836 issuing the retry NACK.
- 837 • A bridge is required to NACK an incoming packet if the bridge does not have input buffer space
838 available for the packet. For the NACK to be recognized by the transmitter as the NACK for a
839 packet retry, the first NACK bit shall be issued no earlier than byte two (that is, the Command
840 Code byte) and no later than byte 8 (the MCTP flags byte). These bytes are represented by the
841 bold outlined bytes in the Figure 3. After the first NACK has been issued any subsequent bytes
842 that are received for the packet shall also be NACK'd until a START, STOP, or bus free
843 condition is detected.



844

845

Figure 3 – Allowed Byte Range for First NACK'd Byte

846
847

- A bridge is required to drop a received packet if it finds that the packet error code (PEC) byte for the transaction is incorrect.

848
849
850

- An MCTP bridge is not allowed to perform "connected" transactions where the decision to ACK or NACK an incoming packet is dependent on the bridge's ability to acquire the destination bus prior to accepting the packet.

851
852
853
854
855

- MCTP bridges are required to implement "store and forward" packet processing. That is, once a bridge has accepted a packet for routing, it shall retain that packet until it can successfully transmit it onto the target bus (except when running out of retries when trying to access the target bus, or upon receiving a packet for a bus that is unavailable or an endpoint that is not present.)

856
857
858
859
860
861

- A bridge cannot make the acceptance of a receive packet on its upstream port (port that connects to a bus that is not owned by the bridge itself) conditional on its ability to transmit a packet on its upstream port. This requirement does not apply to a downstream port on a bridge (that is, a downstream port may elect to NACK an incoming packet to allow the bridge to transmit from that port). This requirement is to help avoid deadlock situations if a bridge is required to route a packet back onto the bus from which the packet came.

862
863

- A bridge that NACKs a packet shall continue to NACK any remaining bytes for the transaction until it recognizes the next START or STOP condition on the bus.

864
865
866
867
868
869
870

- A bridge that receives a NACK while it is performing a Master Write operation is not required to immediately conclude the Master Write operation and drop off the bus. The bridge may continue the write operation through its conclusion. In either case, the master shall always conclude its transaction with a STOP condition, unless some other device on the bus first produces a START or STOP condition. The latter situation is an erroneous condition on the bus, but bridges shall be able to handle it. Devices shall always recognize START and STOP conditions regardless of the transaction or bit position on which they occur.

871 **6.15 Fairness Arbitration Requirements for Non-Bridge Endpoints**

872 Non-bridge/bus owner endpoints ("simple endpoints") are required to implement the MCTP fairness
873 arbitration extensions (when enabled) as follows:

874
875

- The endpoint's port shall support FAIR_IDLE detection and implement the corresponding fairness policy.

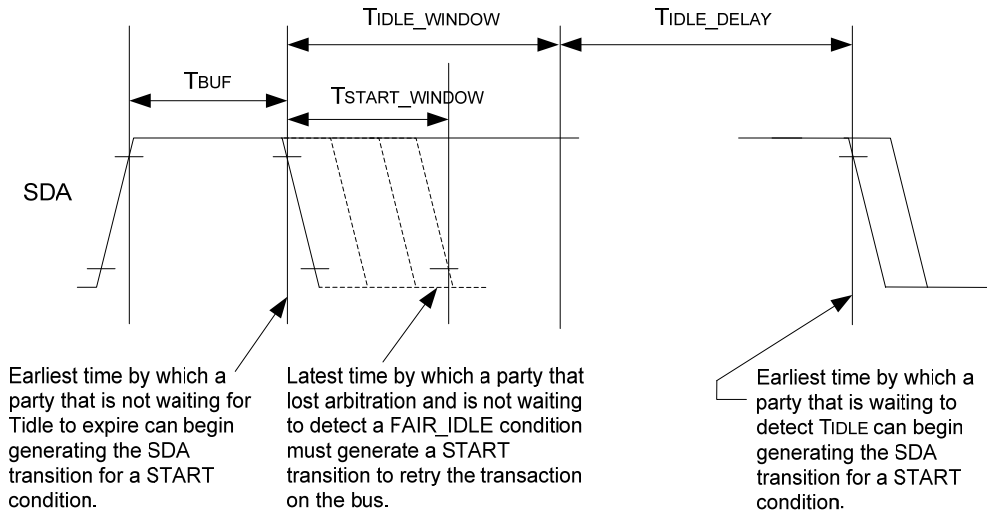
876
877

- Upon device power up or initialization, the endpoint does not need to detect a FAIR_IDLE condition before first attempting to access the bus.

- 878 • The endpoint cannot make the acceptance of a receive packet conditional on its ability to
879 transmit a packet (that is, a simple endpoint shall not NACK incoming packets because it is
880 trying to send an outgoing packet).
- 881 Meeting this requirement may require the endpoint to have separate transmit and receive
882 buffers. This is the recommended implementation.
- 883 If a device is severely limited in buffer space and cannot allocate separate space for both
884 transmit and received data, options are for the endpoint to allow its buffer to be over-written by
885 the receive packet, or in some cases the endpoint may elect to do a dummy receive of the
886 incoming packet (that is, ACK the incoming bytes, but internally drop them as they are coming
887 in.)
- 888 • Higher layer protocols shall be used to handle the case when the endpoint is targeted by more
889 messages than it can process. The buffering requirement for the MCTP Control Protocol
890 messages is defined in the MCTP Base Specification. Buffering requirements for other message
891 types are defined in the respective specifications for the message type.
- 892 • An endpoint is allowed to NACK a packet if it is temporarily unable to accept it (for example,
893 because of an *input* buffer-full condition). This should typically only occur if the endpoint is the
894 target of packets from more than one source endpoint.
- 895 • There is no direct limit of how long a non-bridge endpoint is allowed to successively NACK
896 incoming packets. However, there are limits on how many packet-level retries a transmitter will
897 attempt before it drops the transmitted packet, as well as message type-specific limits on how
898 long and how many times a given message will be retried.
- 899 • An endpoint that does NACK an incoming packet is required to issue the first NACK no earlier
900 than byte 2 (that is, the Command Code byte) and no later than byte 8 (the MCTP flags byte).
901 These bytes are represented by the bold outlined bytes in Figure 3. After the first NACK has
902 been issued, any subsequent bytes that are received for the packet shall also be NACK'd until a
903 STOP or bus free condition is detected.
- 904 • If an endpoint has an output transmit packet and it NACKs an input receive packet from lack of
905 input buffer space, it shall be able to issue a START condition to transmit the output packet
906 within T_{START_WINDOW} after the bus becomes free, unless the endpoint is waiting to detect T_{IDLE} .
- 907 • An endpoint that NACKs a packet shall continue to NACK any remaining bytes for the
908 transaction until it recognizes the next START or STOP condition on the bus.
- 909 • An endpoint that receives a NACK while it is performing a Master Write operation is not required
910 to immediately conclude the Master Write operation and drop off the bus. The endpoint may
911 continue the write operation through its conclusion. In either case, the master shall always
912 conclude its transaction with a STOP condition, unless some other device on the bus first
913 produces a START or STOP condition. The latter situation is an erroneous condition on the bus,
914 but bridges shall be able to handle it. Devices shall always recognize START and STOP
915 conditions regardless of the transaction or bit position on which they occur.
- 916 • Endpoints that are NACK'd or lose arbitration shall retry transaction for PN1 retries (see
917 Table 7).

918 **6.16 Fairness Arbitration Timing**

919 Figure 4, Table 5, and Table 6 present the specifications for the timing intervals for fairness arbitration on
 920 SMBus and I²C relative to the data (SDA) signal. Refer to [SMBus](#) and [I²C](#) for the additional specifications
 921 on the relationship between SCL and SDA for STOP, bus idle, and START conditions.



922

923

Figure 4 – Fairness Arbitration Timing Measurement for SMBus and I²C

924

Table 5 – Fairness Arbitration Timing Values for 100 kHz SMBus/I²C

Symbol	Min	Max	Unit	Notes
T _{BUF}	4.7	–	µs	Per SMBus 100 kHz specification
T _{START_WINDOW}	–	20	µs	Window of time within which a device that is not waiting to detect a FAIR_IDLE condition shall generate START if the device is retrying to gain bus access after losing arbitration.
T _{IDLE_WINDOW}	30	60	µs	Window of time within which a device that is waiting to detect a FAIR_IDLE condition shall not detect a bus busy condition. A FAIR_IDLE condition exists when bus busy is not detected within this interval.
T _{IDLE_DELAY}	31	–	µs	A device that detects FAIR_IDLE condition shall wait this delay before attempting to generate START. This delay accommodates the difference between the T _{IDLE_WINDOW} intervals implemented by different devices on the bus, plus additional time to accommodate bus skews between devices that are generating START and devices that are monitoring for it. This guarantees that one party that has detected T _{IDLE_WINDOW} does not generate START before other devices that are detecting FAIR_IDLE have completed checking for their T _{IDLE} window. Otherwise, the other devices would not see a FAIR_IDLE condition even though one occurred. (Therefore T _{IDLE_DELAY} shall be greater than the difference between the T _{IDLE_WINDOW} maximum and minimum.)

925

Table 6 – Fairness Arbitration Timing Values for 400 kHz I²C

Symbol	Min	Max	Unit	Notes
T _{BUF}	1.3	–	µs	Per I ² C 400 kHz specification
T _{START_WINDOW}	–	4	µs	Window of time within which a device that is waiting to detect a FAIR_IDLE condition shall not detect a bus busy condition. A FAIR_IDLE condition exists when bus busy is not detected within this interval.
T _{IDLE_WINDOW}	5	20	µs	A device that detects FAIR_IDLE condition shall wait for this delay before attempting to generate START. This delay accommodates the difference between the T _{IDLE_WINDOW} intervals implemented by different devices on the bus, plus additional time to accommodate bus skews between devices that are generating START and devices that are monitoring for it. This guarantees that one party that has detected T _{IDLE} does not generate START before other devices that are detecting T _{IDLE} have completed their T _{IDLE} window. Otherwise, the other devices would not see a FAIR_IDLE condition even though one occurred. (Therefore T _{IDLE_DELAY} shall be greater than the difference between the T _{IDLE_WINDOW} maximum and minimum.)
T _{IDLE_DELAY}	16	–	µs	Window of time within which a device that is waiting to detect a FAIR_IDLE condition shall not detect a bus busy condition. A FAIR_IDLE condition exists when bus busy is not detected within this interval.

926 **6.17 MCTP Packet Timing Requirements**

927 The timing specifications shown in Table 7 are specific to MCTP packet transfers on SMBus. Timing is
 928 specified for a "point-to-point" connection. That is, timing is specified as if there were only two endpoints
 929 in direct communication on the bus. In particular, the timing specifications assume that there is no clock
 930 stretching that occurs due to other parties on the bus.

931

Table 7 – Timing Specifications for MCTP Packets on SMBus/I²C

Timing Specification	Symbol	Value	Description
Endpoint packet level retries	PN1	8	Number of times a non-bridge endpoint shall retry sending an MCTP packet upon receiving a NACK during the specified window (see Figure 3). An endpoint that gets successive NACKs shall do one retry for each NACK up to at least this number of retries. This also includes bridges when bridges are transmitting as an endpoint (as opposed to a bridge transmitting from its routing functionality).
Bridge packet level retries	PN2	12	Number of times an MCTP bridge (when transmitting packet for routing) shall retry sending an MCTP packet upon receiving a NACK during the specified window (see Figure 3). A bridge shall do one retry on each NACK up to this number.
Packet transaction originator duration	PT1a	250 µs per byte ^[1]	The overall duration shall be less than the specified interval times the number of bytes in the packet, starting from the byte following the slave byte through and including the PEC byte. Individual data byte transmissions may exceed the specification provided the cumulative

Timing Specification	Symbol	Value	Description
			duration for the packet is met.
Originator slave address byte duration	PT1b	250 μ s ^[1]	The amount of time, including any clock stretching, used to transmit the slave address, Wr, and ACK bits on the bus.
Slave-induced clock stretching	PT1c	250 μ s per byte ^[1]	MCTP devices that are receiving MCTP packets shall not clock stretch the overall packet more than the specified amount. Note that MCTP devices may share the bus with non-MCTP SMBus devices that cause clock stretching that exceeds this specification.
<p>The PT2 parameters are intended to help guide a controller in determining when it is acceptable to initiate a Master Write transaction if the controller powers up or initializes itself on a bus segment that may already be active. It also helps controllers know when it is acceptable to continue under conditions where a STOP condition may have been lost because a controller dropped off the bus due to an error condition. An implementation shall meet at least one of specifications PT2a or PT2b.</p>			
Time-out waiting for bus free without seeing a STOP condition (Bus free determined by not detecting START or STOP)	PT2a	100 ms	<p>For controllers that have hardware that can only detect bus-free/busy-busy status by monitoring for START and STOP conditions, the controller can assume the bus is free if PT2a seconds goes by without detecting a START or STOP condition.</p> <p>If a START condition is detected, the time-out interval restarts.</p> <p>If a STOP condition is detected, the controller can assuming the bus is free following the TBUF interval specified in SMBus.</p> <p>NOTE: This interval effectively places an upper limit on the duration of a single transaction. The byte count in an MCTP packet limits the size of the transaction to 260 bytes. 100 ms is more than sufficient to cover this transfer.</p>
Time-out waiting for bus free without seeing a STOP condition (Bus free determined by data/clock activity)	PT2b	50 μ s	<p>The SMBus specification defines a bus-free (idle) condition as TBUF seconds after a STOP condition, or by the data and clock lines being high for PT2b seconds (where the value for PT2b is taken from T_{HIGH, max} as defined in SMBus).</p> <p>If a controller has appropriate hardware support, monitoring PT2b and TBUF can be used to determine the bus-free (idle) condition in lieu of PT2a. This is generally the most efficient and highest performance way to detect bus free on SMBus.</p> <p>SYSTEM IMPLEMENTATION NOTE: If "bit banded" I²C devices may be used on the same segment, it is important to ensure that those devices do not drive the clock and data high for more than T_{HIGH, max} seconds during transactions.</p>
SDA Low Timeout	PT3	2 sec min, 5 sec max	Time for a bus owner to monitor the SDA low level for a "Stuck 0" before attempting to clear the condition. (See 6.19.)
<p>NOTE 1: Intervals include the ACK bit associated with the byte.</p>			

932 **6.18 MCTP Control Message Timing Requirements**

933 The following timing specifications are specific to MCTP control messages on SMBus/I²C. Timing is
 934 specified for a "point-to-point" connection. That is, timing is specified as if there were only two endpoints
 935 in direct communication on the bus. In particular, the timing specifications assume that there is no clock
 936 stretching occurs due to other parties on the bus.

937 Response specifications are given assuming that the requester is able to operate at full speed on the bus.
 938 That is, clock stretching, if any, is solely generated by the requester.

939 Responses are not retried. A "try" or "retry" of a request is defined as a complete transmission of the
 940 MCTP control message.

941 **Table 8 – Timing Specifications for MCTP Control Messages on SMBus**

Timing Specification	Symbol	Min	Max	Description
Endpoint ID reclaim	T _{RECLAIM}	5 sec	–	Minimum time that a bus owner shall wait before reclaiming the EID for a non-responsive hot-plug endpoint.
Number of request retries	MN1	2	See descr.	Total of three tries, minimum: the original try plus two retries. The maximum number of retries for a given request is limited by the requirement that all retries shall occur within MT4, max of the initial request.
Request-to-response time	MT1	–	100 ms	This interval is measured at the responder from the end of the reception of the MCTP Control Protocol request to the beginning of the transmission of the response. This requirement is tested under the condition where the responder can successfully transmit the response on the first try.

Time-out waiting for a response	MT2	MT1 max+ 2*MT3 max	MT4, min ^[1]	This interval is measured at the requester from the end of the successful transmission of the MCTP Control Protocol request to the beginning of the reception of the corresponding MCTP Control Protocol response. This interval at the requester sets the minimum amount of time that a requester should wait before retrying an MCTP Control Protocol request. Note: This specification does not preclude an implementation from adjusting the minimum time-out waiting for a response to a smaller number than MT2 based on measured response times from responders. The mechanism for doing so is outside the scope of this specification.
Transmission Delay	MT3	-	100 ms	Time to take into account transmission delay of an MCTP Control Protocol Message. Measured as the time between the end of the transmission of an MCTP Control Protocol message at the transmitter to the beginning of the reception of the MCTP Control Protocol message at the receiver.
Instance ID expiration interval	MT4	5 sec ^[2]	6 sec	Interval after which the instance ID for a given response will expire and become reusable if a response has not been received for the request. This is also the maximum time that a responder tracks an instance ID for a given request from a given requester.
NOTE 1: Unless otherwise specified, this timing applies to the mandatory and optional MCTP commands.				
NOTE 2: If a requester is reset, it may produce the same sequence number for a request as one that was previously issued. To guard against this, it is recommended that sequence number expiration be implemented. Any request from a given requester that is received more than MT4 seconds after a previous, matching request should be treated as a new request, not a retry.				

942

943 **6.19 "Stuck 0" Condition Handling**

944 A possible condition exists in SMBus and I²C where a slave device that is being read or is driving ACK
 945 could be left driving a low (0) level onto the data line (SDA) of the bus. The bus uses a "wire OR'd"
 946 approach, where the low (0) level takes precedence over the high (1) level. Therefore, if one party drives
 947 a low (0) level onto the bus, the bus cannot go to a high (1) level until the low level is released.

948 This means that no other transactions can occur until this condition is cleared (because generating a
 949 START or STOP condition on the bus requires being able to drive a high-to-low or low-to-high transition
 950 on the data line, respectively).

951 This condition can occur due to the premature termination of a transaction from the master (as could
952 happen on device resets, power cycles, or firmware restarts, for example) or could occur due to the loss
953 of a clock due to electrical noise.

954 Effectively, what happens is that the device that was being accessed does not recognize that the
955 transaction has been terminated or that a clock was missed. The device continues to drive the 0 onto the
956 bus because it is waiting to get more clocks from the master to conclude the transaction, but those clocks
957 will never come unless some bus master takes steps to generate them.

958 The solution to this condition is to have a master clock the bus until the SDA line goes high, at which point
959 the master can issue a START or STOP condition to get the bus back in synchronization.

960 To accomplish this, the master needs to be able to access and clock the bus without paying attention to
961 the present state of the SDA line.

962 Many microcontrollers have the ability to have firmware dynamically reconfigure their SMBus pins as
963 general purpose I/O pins. If this is supported, it is straightforward for firmware to generate the necessary
964 clocks on the SCL line by bypassing the SMBus controller hardware and using programmed I/O to control
965 the pins instead. The firmware would then simply clock the bus until it sees a "1" condition on the SDA
966 line and then a new SMBus transaction can be launched.

967 NOTE: It is recommended that MCTP bus owners include a provision to detect and clear Stuck 0 conditions on
968 SMBus buses that they own. The controller should do this if it can detect that a constant 0 condition has existed on
969 the SDA line for more than PT3 seconds.

970 **6.20 MCTP over SMBus/I²C Protocol Anti-Aliasing**

971 MCTP over SMBus has been designed to allow one endpoint to support multiple protocols, such as ASF,
972 IPMI, or legacy device-specific protocols with a single slave address. The following clauses describe
973 provisions that can help support implement MCTP over SMBus in devices that also need to support other
974 SMBus or I²C protocols.

975 **6.20.1 IPMI**

976 The IPMI protocols for SMBus (IPMI over SMBus) and I²C (Intelligent Platform Management Bus, IPMB)
977 use the fourth byte of the transaction as a Source Slave Address byte, as does MCTP over SMBus.
978 However, the IPMI protocols require the least significant bit of that byte to be 0b, whereas MCTP over
979 SMBus requires the bit to be 1b. Thus, a device that needs to differentiate between MCTP over SMBus
980 and the IPMI SMBus/I²C protocols can do so using that bit.

981 **6.20.2 ASF**

982 MCTP over SMBus uses the ASF specification reserved value of 0x0F for the command byte. Thus, the
983 ASF-defined commands that use SMBus block-write protocol can be differentiated from MCTP over
984 SMBus block-write using the command byte value. If necessary, other ASF SMBus write transactions,
985 such as those for legacy sensor and control access can be differentiated from MCTP packets based on
986 the length of the transaction. The ASF transactions are all shorter.

987 **6.20.3 Integrating MCTP with Legacy SMBus Functions**

988 This clause describes some possible options if MCTP is being added to a device that shall also support
989 functions using a non-MCTP SMBus interface.

990 In general, there should be no problems having those functions co-exist with MCTP provided that the
991 legacy SMBus operations do not require generating or accepting write transactions that use the MCTP
992 value of 0x0F.

993 If the SMBus device currently uses the 0x0F MCTP command value for a device-specific purpose and it
 994 wants to use the same slave address, the following can be done:

- 995 • The device-specific command can be moved to a different command value. This is generally the
 996 most straightforward approach if it can be supported.
- 997 • Depending on the device-specific command definition, it may be possible to differentiate
 998 between the command and MCTP packets based on other differences, such as the overall
 999 length of the command or differences between the values in the fourth or fifth bytes of the
 1000 command. (MCTP always uses 1b as the least significant bit of the fourth byte, and the fifth
 1001 byte holds a fixed 4-bit value for the Header Version.)
- 1002 • The device can implement MCTP over SMBus on a separate slave address from the legacy
 1003 functions.

1004 **6.21 Well-Known and Reserved Slave Addresses**

1005 For bus segments that support ARP-able devices, Table 9 summarizes addresses that are generally
 1006 reserved by SMBus or I²C and should either be avoided by devices. In addition, some are reserved (not
 1007 to be used as a general device slave address) because those addresses are related to functions that are
 1008 used by MCTP.

1009 **Table 9 – Well-Known and Reserved Slave Addresses**

Slave Address bits [7:1]	R/W# bit [0]	Hex ^[7]	Comment	Disposition
0000 000	0	0x00	I ² C general call address, IPMI broadcast	avoid ^[1]
0000 000	1	0x01	START byte	avoid ^[2]
0000 001	X	0x02 , 0x03	CBUS address	avoid ^[3]
0000 010	X	0x04 , 0x05	Address reserved for different bus format	avoid
0000 011	X	0x06 , 0x07	Reserved for future use by I ² C specifications	avoid
0000 1XX	X	0x08 – 0x0F	I ² C specification, high-speed mode master code	avoid ^[4]
0001 000	X	0x10	SMBus host	rsvd
0001 100	X	0x18 , 0x19	SMBus Alert Response address	rsvd
0010 000	X	0x20 , 0x21	IPMI BMC address	avoid ^[5]
0101 000	X	0x50 , 0x51	Reserved for ACCESS.bus host	avoid (ACCESS.bus defunct)
0110 111	X	0x6E , 0x6F	Reserved for ACCESS.bus default address	avoid
1111 0XX	X	0xF0 – 0xF7	I ² C 10-bit slave addressing ^[1]	avoid ^[6]
1111 1XX	X	0xF8 – 0xFF	Reserved for future use by I ² C specifications	avoid
1100 001	X	0xC2 , 0xC3	SMBus Device Default address	rsvd. Used for SMBus ARP with MCTP

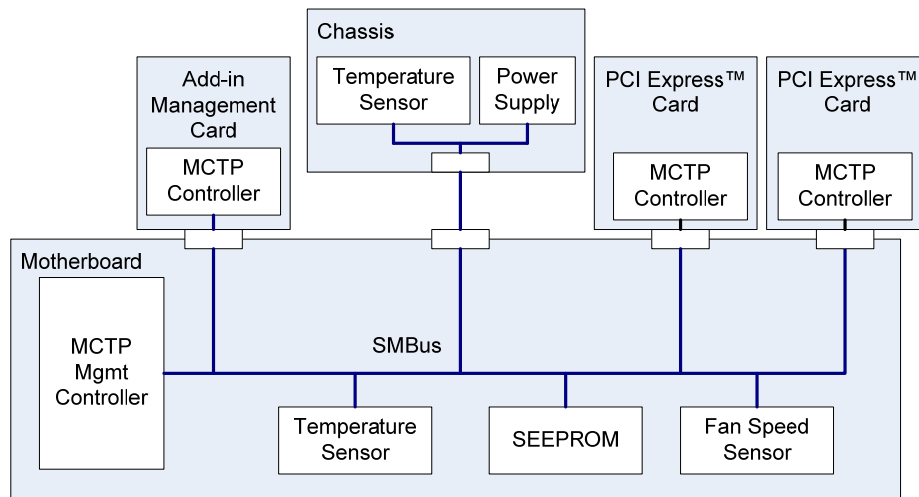
Slave Address bits [7:1]	R/W# bit [0]	Hex ^[7]	Comment	Disposition
NOTE 1.			This address is used as a broadcast address in IPMI and I ² C. It should be avoided if IPMI management controllers may be used on the same bus segment. In I ² C, it is reserved for two purposes: to broadcast a portion of an address that is used for devices that have a portion of their address that is configurable, and as an optional mechanism for a device to master and broadcast its slave address onto the bus. MCTP does not support the use of this address for the I ² C address assignment or slave address broadcast purposes.	
NOTE 2.			The I ² C START byte is a pre-amble to the slave address that is intended to provide time for firmware driven I ² C interfaces to shift into polling of I ² C clock and data lines after a START condition has been detected. This is a very rarely used option in I ² C. MCTP does not support the use of the START byte with MCTP or non-MCTP devices.	
NOTE 3.			CBUS is an ancestor of I ² C, developed by Philips Semiconductor. It uses a data and clock signal similar to I ² C, but with a third signal (SEN) used to generate the START and STOP conditions on the bus. This address range was reserved by the I ² C specification to enable a degree of backward compatibility with CBUS devices sharing the I ² C SCL and SDA lines as the CBUS clock and data lines, respectively. While listed as a reserved address in the I ² C specification, few SMBus/I ² C implementations using MCTP will have any need to also support CBUS devices.	
NOTE 4.			MCTP is not defined to support I ² C high-speed mode operation.	
NOTE 5.			This address is the "well known address" for an IPMI BMC. This address should be avoided if an IPMI BMC may be used on the same MCTP segment.	
NOTE 6.			Used in conjunction with the R/W# bit position to deliver the most-significant three address bits for I ² C 10-bit addressing. MCTP protocols and data structures do not support 10-bit addressing on SMBus or I ² C segments. MCTP only supports 7-bit addresses for MCTP and non-MCTP devices on a bus segment.	
NOTE 7.			By convention, when the 7-bit slave address field is represented as a two-digit hexadecimal number, it is treated as an 8-bit value where the 7-bit address occupies the upper 7 bits and the least significant bit is 0b or 1b according to the value of the SMBus/I2C Read-Write bit associated with the slave address.	

1010 6.22 Fixed Address Allocation

1011 One of the problems that an implementer often faces is choosing which slave address to use. For the
 1012 PCI™ and PCI Express™ bus specifications, the specifications require that devices on standard
 1013 connectors defined by those specifications have their addresses set through SMBus ARP. Therefore,
 1014 fixed address allocation is not an option for PCI add-in cards themselves. In fixed bus implementations,
 1015 however, there are many situations where it is desired or necessary to utilize fixed-address devices.

1016 From a practical point-of-view, SMBus and I²C do not have an effective central registry or other
 1017 mechanism for avoiding conflicts in the assignment and use of slave addresses among device vendors.
 1018 While there are potential registries of device slave address usage for SMBus (under the System
 1019 Management Interface Forum) and I²C (from Philips Semiconductor), these have not generally been used
 1020 by device vendors and there is no group or standard that works to enforce conformance to those
 1021 registries.

1022 Most device vendors provide a configurable range of three or more addresses to enable an implementer
 1023 to reconcile address conflicts on a single segment. Because typically only a small number of fixed-
 1024 address devices are used on a given segment, it is frequently possible to configure devices so they do
 1025 not have overlapping addresses. This approach is problematic, however, in situations where a component
 1026 that is attached to that segment in the platform may come from several sources. Clause 6.22 provides
 1027 guidelines to allocating fixed addresses that are designed to reduce the number of conflicts that could
 1028 occur when multiple suppliers provide different elements of a computer system (see Figure 5 for an
 1029 example).



1030

1031

Figure 5 – Example System Configuration

1032 **6.23 Recommended Address Range Allocation for Computer Systems**

1033 This clause provides a recommended allocation of SMBus addresses between board, chassis, and add-in
 1034 uses that help avoid address conflicts when fixed addresses are used. It also serves as a general
 1035 guideline of what addresses a generic ARP master should use for allocation to PCI/PCIe add-in cards.

1036 There might be cases when MCTP is used within a typical computer system application where the
 1037 motherboard may come from one supplier, the chassis from another supplier, and possibly add-in
 1038 modules from yet another supplier. To facilitate the mix-and-match of these elements and to help avoid
 1039 the need for every system manufacturer to set up their own address allocation conventions with suppliers,
 1040 MCTP recommends that system manufacturers follow the address allocation approach initially defined by
 1041 the IPMI specifications (see Table 10). This approach splits the available fixed addresses (addresses
 1042 other than reserved addresses) into four main usage areas:

1043 **B Board:** An area reserved for board set manufacturer use (where *board set* would be the
 1044 motherboard and other boards that accompany that motherboard from the same vendor).

1045 **C Chassis:** An area reserved for use by vendors that make chassis in which a third-party board
 1046 set would be used.

1047 **A Add-in:** For third-party add-in devices (for example, modules or add-in cards that used fixed
 1048 addresses and would be used in combination with a motherboard or chassis where there is a
 1049 connection to a SMBus segment implementing MCTP).

1050 NOTE: PCI/PCIe add-in cards that use standard PCI connectors are required to support SMBus ARP
 1051 and fixed addresses are not used.

1052 **R Reserved** for IPMI, I²C, SMBus, or MCTP uses. Includes the *avoid* addresses from Table 9.

1053 By following this convention, future motherboards can offer connections to chassis elements and third-
 1054 party modules where those devices can use fixed addresses, if required. It also provides a convention to
 1055 avoid conflicts if legacy non-MCTP devices share the same SMBus segment.

1056

Table 10 – Slave Address Allocation for Computer Systems

Use	Address	Typical Device	Use	Address	Typical Device
R	0x00	I ² C, IPMB broadcast	C	0x44	8574
	0x20	IPMB uC (BMC)		0x46	8574
	0x01	I ² C		0x48	8574
	0x02	I ² C		0x4A	8574
	0x04-0x0E	I ² C		0x4C	8574
	0x50	ACCESS.bus		0x4E	8574
	0x6E	ACCESS.bus		0x52-0x6C	58h, 5Ah, 5Ch = Heceta
	0xF0-0xF6	I ² C		0x74	8574A
	0xF8-0xFE	I ² C		0x76	8574A
A	0x10	SMBus host (B)	0x78	8574A	
	0x12-0x16		0x7A	8574A	
	0x18	SMBus Alert Response address (B)	0x7Ch	8574A	
	0x1A-0x1E		0x7E	8574A	
	0x30-0x3E		0x98	LM75, DS1624, DS1621	
	0xB0-0xBE	power-supply monitoring	0x9A	LM75, DS1624, DS1621	
	0xD0-0xDE		0x9C	uC (pri. HSC), DS1624, DS1621	
B	0x22	uC (FPC, ICMB) ^[1]	0x9E	uC (sec. HSC), DS1624, DS1621	
	0x24	uC (PBC) ^[1]	0xA4	SEEPROM	
	0x26		0xA6	SEEPROM	
	0x28	SM Card ^[1]	0xAC	SEEPROM	
	0x2A-0x2E		0xAE	SEEPROM	
	0x40	8574	NOTE 1: Term from IPMI usage. FPC = front panel controller, PBC = Power Backplane Controller, ICMB = Intelligent Chassis Management Bus bridge, SM Card = System Management Card		
	0x42	8574			
	0x70	8574A			
	0x72	8574A			
	0x80-0x8E				
	0x90	LM75, DS1624, DS1621, 8591			
	0x92	LM75, DS1624, DS1621, 8591			
	0x94	LM75, DS1624, DS1621			
	0x96	LM75, DS1624, DS1621			
	0xA0	SEEPROM			
	0xA2	SEEPROM			
	0xA8	SEEPROM			
	0xAA	SEEPROM			
	0xC0				
	0xC2	SMBus Device Default address			
	0xC4-0xCE				
	0xE0-0xEE				

Annex A (informative)

Notation

1057
1058
1059
1060
1061

1062 A.1 Notations

1063 Examples of notations used in this document are as follows:

- 1064 • 2:N In field descriptions, this will typically be used to represent a range of byte offsets
1065 starting from byte two and continuing to and including byte N. The lowest offset is on
1066 the left, the highest is on the right.
- 1067 • (6) Parentheses around a single number can be used in message field descriptions to
1068 indicate a byte field that may be present or absent.
- 1069 • (3:6) Parentheses around a field consisting of a range of bytes indicates the entire range
1070 may be present or absent. The lowest offset is on the left, the highest is on the right.
- 1071 • [PCle](#) Underlined, blue text is typically used to indicate a reference to a document or
1072 specification called out in 2, "Normative References" or to items hyperlinked within the
1073 document.
- 1074 • rsvd Abbreviation for "reserved." Case insensitive.
- 1075 • [4] Square brackets around a number are typically used to indicate a bit offset. Bit offsets
1076 are given as zero-based values (that is, the least significant bit [LSb] offset = 0).
- 1077 • [7:5] A range of bit offsets. The most significant bit is on the left, the least significant bit is
1078 on the right.
- 1079 • 1b The lower case "b" following a number consisting of 0s and 1s is used to indicate the
1080 number is being given in binary format.
- 1081 • 0x12A A leading "0x" is used to indicate a number given in hexadecimal format.
- 1082

Annex B
(informative)**Change Log**

Version	Date	Author	Description
1.0.0	7/28/2009		DMTF Standard Release

1083
1084
1085
1086
1087

1088