Specification for
I3C Basic^{SM}

Improved Inter Integrated Circuit

Version 1.0 – 19 July 2018

MIPI Board Adopted 8 October 2018

* NOTE TO IMPLEMENTERS *

This page intentionally left blank.

Specification for
I3C Basic^{SM}

Improved Inter Integrated Circuit – Basic

Version 1.0
19 July 2018

MIPI Board Adopted 8 October 2018

NOTICE OF DISCLAIMER

The material contained herein is provided on an "AS IS" basis. To the maximum extent permitted by applicable law, this material is provided AS IS AND WITH ALL FAULTS, and the authors and developers of this material and MIPI Alliance Inc. ("MIPI") hereby disclaim all other warranties and conditions, either express, implied or statutory, including, but not limited to, any (if any) implied warranties, duties or conditions of merchantability, of fitness for a particular purpose, of accuracy or completeness of responses, of results, of workmanlike effort, of lack of viruses, and of lack of negligence. ALSO, THERE IS NO WARRANTY OR CONDITION OF TITLE, QUIET ENJOYMENT, QUIET POSSESSION, CORRESPONDENCE TO DESCRIPTION OR NON-INFRINGEMENT WITH REGARD TO THIS MATERIAL.

IN NO EVENT WILL ANY AUTHOR OR DEVELOPER OF THIS MATERIAL OR MIPI BE LIABLE TO ANY OTHER PARTY FOR THE COST OF PROCURING SUBSTITUTE GOODS OR SERVICES, LOST PROFITS, LOSS OF USE, LOSS OF DATA, OR ANY INCIDENTAL, CONSEQUENTIAL, DIRECT, INDIRECT, OR SPECIAL DAMAGES WHETHER UNDER CONTRACT, TORT, WARRANTY, OR OTHERWISE, ARISING IN ANY WAY OUT OF THIS OR ANY OTHER AGREEMENT RELATING TO THIS MATERIAL, WHETHER OR NOT SUCH PARTY HAD ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

The material contained herein is not a license, either expressly or impliedly, to any IPR owned or controlled by any of the authors or developers of this material or MIPI. Any license to use this material is granted separately from this document. This material is protected by copyright laws, and may not be reproduced, republished, distributed, transmitted, displayed, broadcast or otherwise exploited in any manner without the express prior written permission of MIPI Alliance. MIPI, MIPI Alliance and the dotted rainbow arch and all related trademarks, service marks, tradenames, and other intellectual property are the exclusive property of MIPI Alliance Inc. and cannot be used without its express prior written permission. The use or implementation of this material may involve or require the use of intellectual property rights ("IPR") including (but not limited to) patents, patent applications, or copyrights owned by one or more parties, whether or not members of MIPI. MIPI does not make any search or investigation for IPR, nor does MIPI require or request the disclosure of any IPR or claims of IPR as respects the contents of this material or otherwise.

Without limiting the generality of the disclaimers stated above, users of this material are further notified that MIPI: (a) does not evaluate, test or verify the accuracy, soundness or credibility of the contents of this material; (b) does not monitor or enforce compliance with the contents of this material; and (c) does not certify, test, or in any manner investigate products or services or any claims of compliance with MIPI specifications or related material.

Questions pertaining to this material, or the terms or conditions of its provision, should be addressed to:

MIPI Alliance, Inc.
c/o IEEE-ISTO
445 Hoes Lane, Piscataway New Jersey 08854, United States
Attn: Managing Director

Contents

Figures

Tables

Release History

Date	Version	Description
2018-10-08	v1.0	Initial Board Adopted release.

This page intentionally left blank.

Preface to the I3C Basic Specification

1. What is MIPI I3C Basic?

MIPI I3C Basic is a feature-reduced, lower-complexity version of the powerful, flexible, and efficient MIPI I3C interface *[MIPI02]*, suitable for a broad range of device interconnect applications including sensor and memory interfacing.

2. Motivation

MIPI Alliance considers I3C technology to have significant advantages over the widely adopted legacy interfaces currently used for similar applications, and wishes to see I3C technology broadly deployed in the industry. Being aware that some advanced I3C v1.x features are not needed for many common applications, and that some potential users might regard MIPI Alliance membership as a barrier, the MIPI Board of Directors decided to make the I3C Basic Specification freely available without requiring a MIPI Alliance membership, and to facilitate a royalty free licensing environment for all implementers, as described further below.

3. IPR Status

MIPI Alliance's regular intellectual property rights-related terms apply only by and among members. To address this issue, MIPI Alliance created a set of supplemental patent licensing terms for the current and future versions of the I3C Basic Specification, attached to this document as *Annex E*. MIPI required that all members participating in the development of the I3C Basic Specification agree to these additional terms. These terms include an obligation to license applicable patent claims to both member and non-member implementers, for uses both in mobile devices and otherwise, on "RAND-Z" terms: that is, under royalty free (the Z references zero royalty) and otherwise reasonable and non-discriminatory terms.

As described in *Annex E*, any implementer that wishes to benefit from the RAND-Z obligation must also commit to license other implementers under the same RAND-Z terms. This reciprocity requirement is intended to expand royalty free license obligations through the broader I3C Basic ecosystem.

The MIPI member companies that joined the I3C Basic Specification development effort and manifest agreement to the terms in *Annex E* are listed in *Annex F*. *Annex F* also notes if and when any party terminates their agreement to these terms (subject to certain ongoing license obligations, as described in the terms).

The I3C Basic IPR terms are intended to create a robust royalty free environment for implementers. However, no set of IPR terms can comprehensively address all potential risks. The terms apply only to those parties that agree to them, and the scope of application is limited to what is described in the terms. Implementers must ultimately make their own risk assessment, and the disclaimers described elsewhere in this document remain in full force and effect.

4. Relationship to MIPI I3C v1.x Specifications

The MIPI I3C Basic v1.0 Specification is a subset of the MIPI I3C v1.0 Specification, but also incorporates certain elements of the draft MIPI I3C v1.1 Specification. In the I3C Basic Specification, the terms "I3C," "I3C Device," and "I3C Bus" should be interpreted as referring to both I3C v1.x and I3C Basic v1.x.

4.1 I3C v1.0 Functions Not Included in I3C Basic

The following functions of the I3C v1.0 Specification have been removed from the I3C Basic v1.0 Specification. Companies interested in implementing these functions should join MIPI Alliance to enjoy member IPR licensing.

- Timing Control (***Section 5.1.8***)
- HDR Double Data Rate Mode (HDR-DDR) (***Section 5.2.2***)
- HDR Ternary Modes (HDR-TSP and HDR-TSL) (***Section 5.2.3***)

4.2 I3C v1.1 Functions Included in I3C Basic

The following I3C Basic functions do not appear in MIPI I3C v1.0, but are expected in the forthcoming MIPI I3C v1.1 Specification.

- Direct Read/Write CCC Capability (***Section 5.1.9.1***, Category 4)
- Get Max Data Speed (GETMXDS) Refinement (***Section 5.1.9.3.18***)
- SETAASA CCC (***Section 5.1.9.3.22***)
- Low Voltage / High Capacitive Load IO (***Section 6***, ***Table 55***)

5. How I3C Basic Devices Work with I3C v1.x Devices

MIPI I3C Basic Devices, whether Main Master, Secondary Master, or Slave, are intended to be interoperable with I3C v1.x Devices for the set of optional features that are mutually supported by the connected Devices. It is easiest to view I3C Basic as a subset of I3C, with a specific set of additional, optional features that are only offered to MIPI Alliance Members.

1 Introduction

57 The proliferation of sensors in mobile wireless and mobile-influenced products has created significant design
58 challenges. Because there are no consistent methods for interfacing physical sensors, Device and platform
59 designers are faced with digital interface fragmentation that includes I^2C, SPI, and UART among others.

60 In addition to the main interface other signals may be needed, such as dedicated interrupts, chip select signals,
61 and enable and sleep signals. This increases the required number of Host GPIOs, and that in turn drives up
62 system cost with more Host package pins and more PCB layers.

63 As time passes and the number of sensors increases, this situation is becoming increasingly difficult to
64 support and manage.

65 The MIPI I3C interface has been developed to ease sensor system design architectures in mobile wireless
66 products by providing a fast, low cost, low power, two-wire digital interface for sensors.

67

68
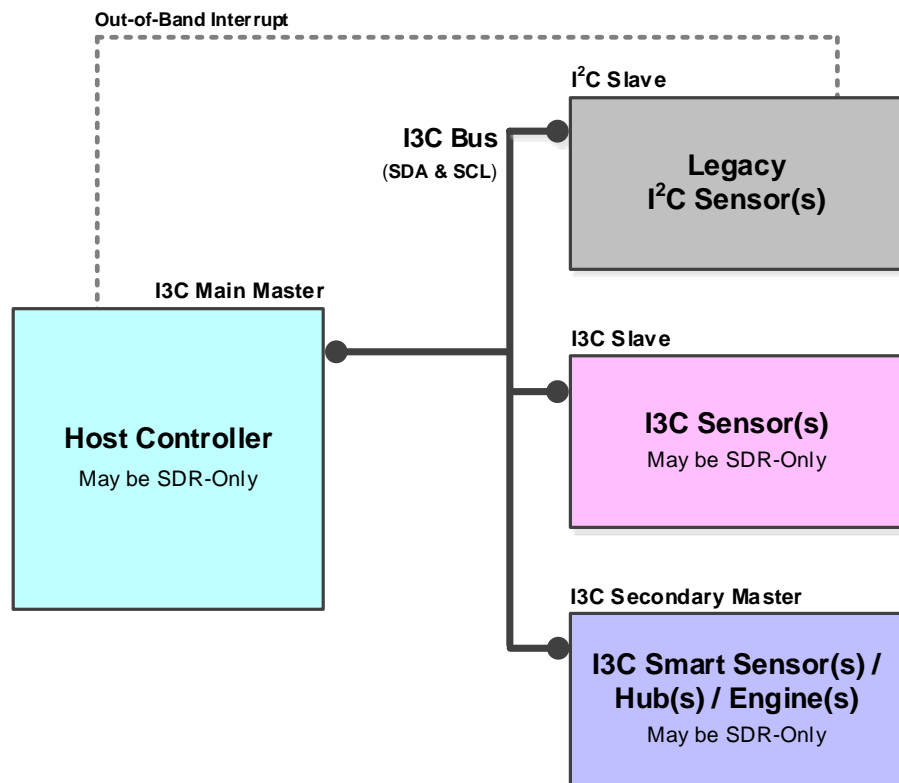
Figure 1 I3C System Diagram

69   Example classes of sensor addressed by I3C are listed in *Table 1*.

70   **Table 1 Sensor Classes Addressed by I3C**

Mechanical / Motion	**Biometrics/Health**
• Compass/Magnetometer • Gyro • Accelerometer • Proximity • Touch screen • Grip • Time of Flight (gestures) • Audio/Ultrasonic (events)	• Fingerprint • Glucometer • Heart rate • Olfactory (e.g. breathalyzer) • EKG • GSR (galvanic skin response)
Environmental Sensing	**Other**
• Ambient light • Barometric pressure / altimeter • Temperature • Carbon monoxide / pollutants • Humidity	• NFC (Near Field Communication) • Haptic feedback • IR (smart TV remote) • UV/RGB

1.1   Scope

71   The following topics are in scope for this Specification:

72   • I3C interface protocols and commands leveraged for I3C Basic

73   • Electrical specifications, such as timing and voltage levels

74   • Support for specific classes of sensors and other Devices

75   The following topics are out of scope for this Specification:

76   • Mechanical, system, and implementation details within an I3C Device

77   • ESD (Electrostatic Discharge) structures

78   • System power management

79   • Use case specific data or format definitions besides Bus management command codes

1.2   I3C Purpose

80   The I3C interface is intended to improve upon the features of the I^{2}C interface *[NXP01]*, preserving backward
81   compatibility. This Specification defines a standard Multi-Drop interface between Host processors and
82   peripheral sensors.

83   Implementing the I3C Specification greatly increases the flexibility mobile terminal system designers have
84   to support an ever-expanding sensor subsystem as efficiently and at as low a cost as possible.

1.3   I3C Key Features

85   Two main concerns are paramount for the I3C interface: The use of as little energy as possible in transporting
86   data and control, while reducing the number of physical pins used by the interface.

87   Therefore, the I3C interface features:

88   • Two wire serial interface up to 12.5 MHz using Push-Pull
89   • Legacy I²C Device co-existence on the same Bus (with some limitations)
90   • Dynamic Addressing while supporting Static Addressing for Legacy I²C Devices
91   • Legacy I²C messaging
92   • I²C-like Single Data Rate messaging (SDR)
93   • **NOT SUPPORTED IN I3C BASIC:** Optional High Data Rate messaging Modes (HDR)
94   • Multi-Drop capability
95   • Multi-Master capability
96   • In-Band Interrupt support
97   • Hot-Join support
98   • **NOT SUPPORTED IN I3C BASIC:** Synchronous Timing Support and Asynchronous Time
99   Stamping

100   The I3C interface provides major efficiencies in Bus power while providing greater than 10x speed
101   improvements over I²C. *Figure 2* and *Figure 3* show different Bus Mode options that provide tradeoffs for
102   performance/power vs. target Device complexity.

The bar chart on the left shows energy consumption for a given amount of data for the different I3C modes, compared to I2C. The bar chart on the right shows the same comparison for data throughput.
Both charts show a significant advantage for I3C.

103
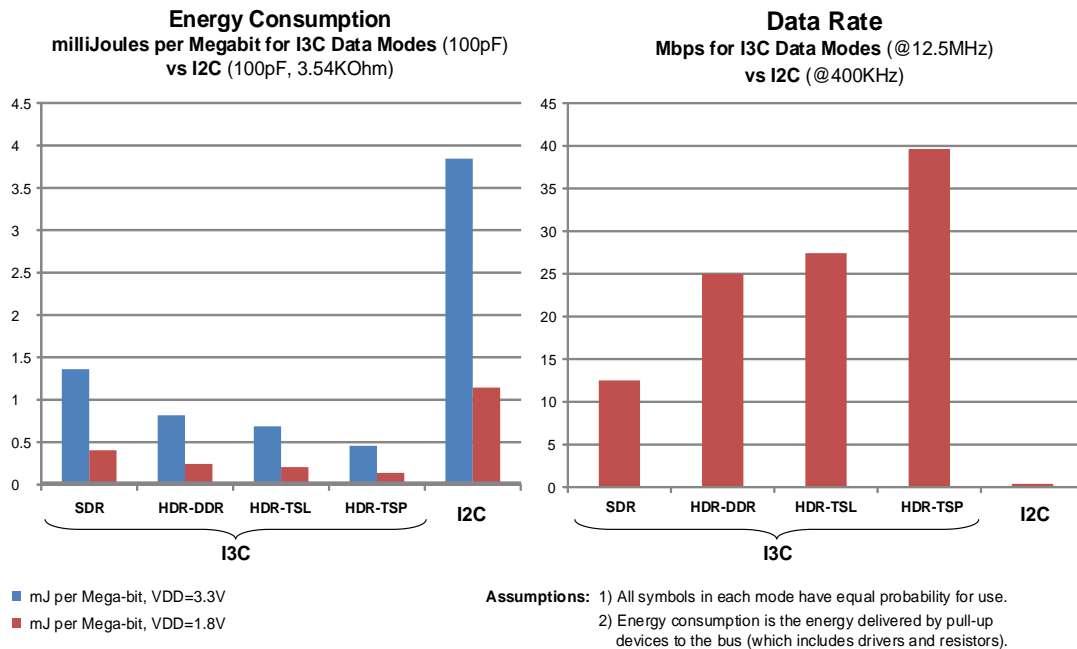
104   **Figure 2 Energy Consumption and Raw Data Rate: I3C vs. I²C**

105   The bar chart in *Figure 2* shows energy consumption for a given amount of data for the different I3C Modes
106   compared to I²C (units are milli-Joules per megabit) whereas on the right is data-throughput. Both show a
107   significant advantage for I3C.

108   *Figure 3* shows an expanded view of energy consumption for different I3C Modes.

Energy Consumption: mJ per mega bit for I3C data modes

Assumptions: 1) All symbols in each mode have equal probability for use.
109   2) Energy consumption is the energy delivered by pull up devices to the bus (which includes drivers and resist
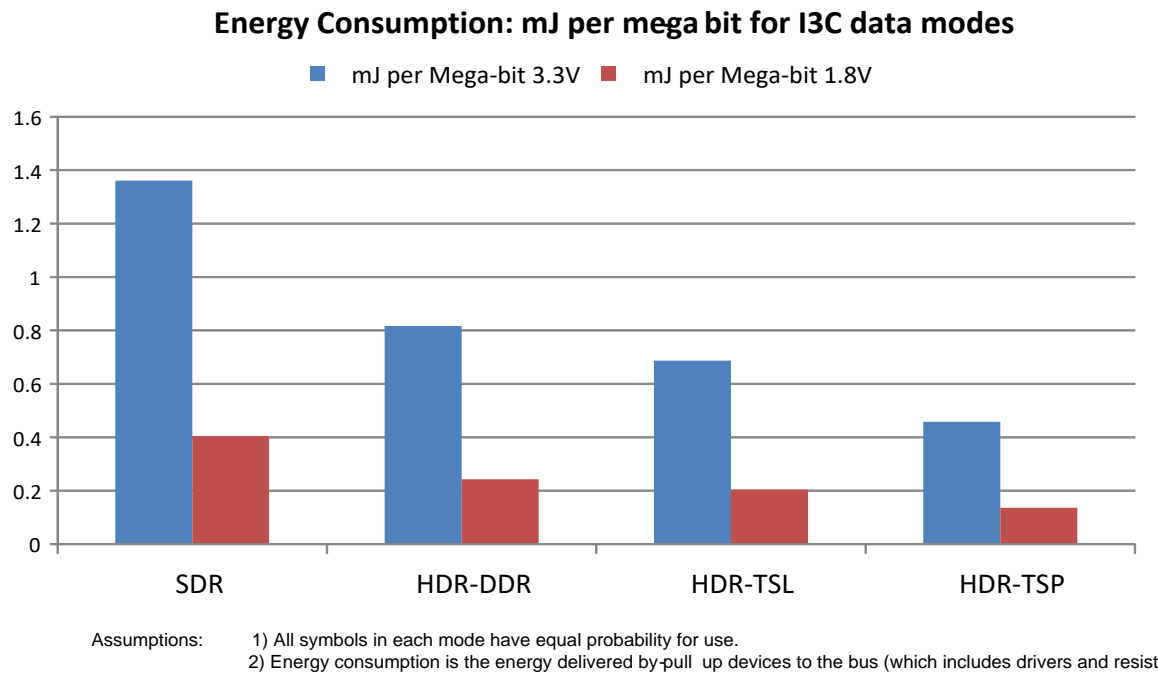
110   **Figure 3 Energy Consumption Comparison for I3C Data Modes**

2 Terminology

2.1 Use of Special Terms

111 The MIPI Alliance has adopted Section 13.1 of the *IEEE Standards Style Manual*, which dictates use of the
112 words "shall", "should", "may", and "can" in the development of documentation, as follows:

113 The word *shall* is used to indicate mandatory requirements strictly to be followed in order
114 to conform to the Specification and from which no deviation is permitted (*shall* equals *is*
115 *required to*).

116 The use of the word *must* is deprecated and shall not be used when stating mandatory
117 requirements; *must* is used only to describe unavoidable situations.

118 The use of the word *will* is deprecated and shall not be used when stating mandatory
119 requirements; *will* is only used in statements of fact.

120 The word *should* is used to indicate that among several possibilities one is recommended
121 as particularly suitable, without mentioning or excluding others; or that a certain course of
122 action is preferred but not necessarily required; or that (in the negative form) a certain
123 course of action is deprecated but not prohibited (*should* equals *is recommended that*).

124 The word *may* is used to indicate a course of action permissible within the limits of the
125 Specification (*may* equals *is permitted to*).

126 The word *can* is used for statements of possibility and capability, whether material,
127 physical, or causal (*can* equals *is able to*).

128 All sections are normative, unless they are explicitly indicated to be informative.

129 All quoted voltage and frequency values in the informative sections represent their typical values.

2.2 Definitions

130 **ACK:** Short for "acknowledge". See also **NACK**.

131 **Address Arbitration:** Process for determining arbitrated Addresses to resolve contention.

132 **Address:** A set of bits designating a Device or the location of a register.

133 **Arbitrable:** Subject to decision by Arbitration.

134 **Arbitration:** If two Devices start transmission at the same time, then Arbitration is required to determine
135 Bus control. Arbitration could also be required during a Slave transmission if a Master addresses multiple
136 Slaves.

137 **Bridge Device:** Device on the I3C Bus that allows conversion from the native I3C Bus protocol to another
138 protocol (such as SPI, UART etc.).

139 **Broadcast:** A command intended for multiple Slave Devices, using the Broadcast Address 7'h7E.

140 **Bus:** The physical and logical implementation of the SCL and SDA lines.

141 **Bus Available Condition:** State on the I3C Bus where both the SCL line and the SDA line are High for at
142 least t_{AVAL} (see *Table 58*), and a Device is able to initiate a transaction on the Bus.

143 **Bus Free Condition:** State on the I3C Bus after a STOP and before a START for at least t_{CAS} (see *Table 58*).

144 **Bus Idle Condition:** An extended duration of the Bus Free Condition that indicates that Devices may attempt
145 to Hot-Join the I3C Bus.

146 **Bus Turnaround:** When a transmitting Device sends a command, and then the receiving Device takes over
147 the I3C Bus in order to respond.

148 **Characteristics:** Quantification of a Device's available features and capabilities.

149 **Clock to Data Turnaround Time:** The time duration between reception of an SCL edge and the start of
150 driving an SDA change. See t_{SCO} in **Table 59**.

151 **CRC5:** Cyclic Redundancy Check with fifth-order polynomial length.

152 **Current Master:** The I3C Device that presently has Master control of the I3C Bus.

153 **Device:** A Master or Slave.

154 **Device ID:** Defines a Device's characteristic or function within a sensor system.

155 **Dynamic Address:** A Device Address that is assigned or allocated during initialization of the Bus. Usually
156 occurs after power up.

157 **Failsafe:** An I3C Device Bus pad is considered Failsafe if its leakage does not increase when it is unpowered.
158 A pad may be unpowered because the Device is unpowered, because its IO rail is unpowered or clamped, or
159 both. Failsafe only matters for some Hot-Join Devices.

160 **Frame:** A Frame begins with a START, followed by the Address of the targeted Slave(s), Data, and finally a
161 STOP.

162 **High:** Defines a signal level that is a logical "1".

163 **High Data Rate (HDR):** High Data Rate Modes that achieve higher speed by transferring data on both clock
164 edges.

165 **High-Keeper:** A weak Pull-Up type Device used when SDA, and sometimes SCL, is in High-Z with respect
166 to all Devices.

167 **Host:** Hardware and software that provides the core functionality of a mobile device.

168 **Hot-Join:** Slaves that join the Bus after it is already started, whether because they were not powered
169 previously or because they were physically inserted into the Bus; the Hot-Join mechanism allows the Slave
170 to notify the Master that it is ready to get a Dynamic Address.

171 **In-Band Interrupt (IBI):** A method whereby a Slave Device emits its Address into the arbitrated Address
172 header on the I3C Bus to notify the Master of an interrupt.

173 **I²C Device:** A Master or Slave that meets the requirements of the I²C Specification *[NXP01]*.

174 **I3C Basic Device:** A Master or Slave that meets the requirements of the I3C Basic Specification (this
175 document).

176 **I3C v1.x Device:** A Master or Slave that meets the requirements of the MIPI I3C Specification, version 1.0
177 *[MIPI02]* or any subsequently developed MIPI I3C Specification version 1.1, 1.2, etc.

178 **I3C Slave:** See Slave.

179 **Legacy I²C:** I3C maintains the industry standard architecture of I²C and supports existing I²C Slave Devices.
180 I3C does not support I²C Bus Masters.

181 **Low:** Defines a signal level that is a logical "0".

182 **Main Master:** Master that has overall control of the I3C Bus. Including control and hand off to Secondary
183 Masters.

184 **Master:** A reference to the I3C Bus Device that is controlling the Bus.

185 **Mastership:** Control of the I3C Bus, in a Master role.

186 **Message:** A packetized communication between Devices.

187 **Minimal Bus:** An I3C Bus with one Master Device (potentially with reduced functionality), and one active
188 Slave Device with a fixed and reserved Slave Address value of 7'h01. Additional Read-only Slave Devices
189 may optionally also be present in a Minimal Bus, but there can be no additional Read-Write Slave Devices.

190 **MIPI Manufacturer ID:** A two byte/16 bit unique identifier for a vendor of a MIPI compliant Device
191 *[MIPI01]*.

192 **Mixed Fast Bus:** I3C Bus topology with both I^{2}C and I3C Devices present on the I3C Bus, where the I^{2}C
193 Devices have a true I^{2}C 50 ns Spike Filter on the SCL line.

194 **Mixed Slow/Limited Bus:** I3C Bus topology with both I^{2}C and I3C Devices present on the I3C Bus, where
195 the I^{2}C Devices do not have a true I^{2}C 50 ns Spike Filter on the SCL line.

196 **Mode:** Distinguishes different data transfer methods used in I3C including Legacy I^{2}C Mode, Single Data
197 Rate Mode (SDR), and High Data Rate Mode (HDR).

198 **Multi-Drop:** A Bus that communicates through a process of Arbitration to determine which Device sends
199 information at any point. The other Devices listen for data they are intended to receive.

200 **Multi-Master:** Multiple Bus Masters present on the Bus. Used when multiple nodes on the Bus need to
201 initiate a transfer.

202 **NACK**: Short for "not acknowledge", which means No ACK was asserted. See also **ACK**.

203 **Offline Capable:** An Offline Capable Device is able to disconnect from the physical I3C Bus and/or is able
204 to ignore I3C traffic on the I3C Bus. A Device's Offline capability is one of the capabilities reflected in its
205 Bus Characterization Register.

206 **Open Drain:** High-Z with an active Pull-Down. Typically used in conjunction with a passive Pull-Up.

207 **Park:** Logic level High set by the Master (or Slave on Read) before turning around the Bus to allow the other
208 to drive to Logic level Low or not (will be held High by weak Pull-Up).

209 **Pull-Down:** Active mechanism used to pull the Bus to a logical Low state.

210 **Pull-Up:** Mechanism used to pull the Bus to a logical High state. The mechanism may be either active or
211 passive.

212 **Pure Bus:** A Bus topology with only I3C Devices present. No I^{2}C Devices are permitted on a Pure Bus.

213 **Push-Pull:** Active Pull-Down and active Pull-Up on output driver.

214 **Repeated START:** Two or more instances of a START in a row without an intervening STOP. A Repeated
215 START is used in circumstances where the Master wishes to continue communicating on the I3C Bus without
216 having to first generate a STOP. In this Specification, a Repeated START is abbreviated as "Sr". This is
217 equivalent to repeated START in I^{2}C *[NXP01]*.

218 **SDR-Only:** An SDR-Only Device supports only SDR Mode, i.e. does not support HDR Mode.

219 **Secondary Master:** A Secondary Master controls the I3C Bus only after receiving permission from the Main
220 Master. Control of the Bus is temporary, and always ends with passing control back to the Main Master.

221 **Single Data Rate (SDR):** Single Data Rate transfers data on only one edge of the clock.

222 **Slave:** A Slave Device can only respond to either Common or individual commands from a Master. A Slave
223 Device cannot generate a clock.

224 **Spike Filter:** A filter that removes SCL (and SDA) spikes shorter than 50 ns in duration. See Input Filter (t$_{SP}$
225 parameter) in the I^{2}C Specification *[NXP01]*. Also known as a glitch filter.

226 **Stall:** The act of the I3C Master holding the SCL line Low under specific transitory conditions.

227 **START:** START is the I3C Bus condition of a High to Low transition on the SDA line while the SCL line
228 remains High. In this Specification, a START is abbreviated as "S".

229 **START Request:** A method for a Slave to force the Master to issue a START on an idled I3C Bus.

230 **Static Address:** A Device Address that is fixed and cannot be changed.

231 **STOP:** STOP is the I3C Bus condition of a Low to High transition on the SDA line while the SCL line
232 remains High. In this Specification, a STOP is abbreviated as "P".

233 **T-Bit:** Transition bit, an alternative to the ACK/NACK mechanism.

234 **Word:** Transmission containing 16 payload bits and two parity bits.

235 **Word Types:** Four Word Types are used: Command Word, User Data Word, CRC Word, and Reserved Word.

2.3 Abbreviations

236 ACK   Acknowledge

237 e.g.   For example (Latin: exempli gratia)

238 i.e.   That is (Latin: id est)

239 High-Z   An output driver that is set to high impedance mode (cannot source or sink current)

240 NACK   Not Acknowledge

241 P   STOP

242 PHY   Physical Layer

243 S   START

244 Sr   Repeated START

245 T   Transition Bit

2.4 Acronyms

246 BCR   Bus Characteristics Register

247 BER   Bit Error Rate

248 BMB   Bus Management Block

249 CCC   Common Command Code

250 CRC   Cyclic Redundancy Check

251 DAR   Dynamic Address Request

252 DCR   Device Characteristics Register

253 DDR   Double Data Rate

254 ESD   Electro Static Discharge

255 FSM   Finite State Machine

256 HDR   High Data Rate

257 HDR-DDR   HDR Double Data Rate Mode

258 IBI   In-Band Interrupt

259 ISTO   Industry Standards and Technology Organization

260 LCR   Legacy Characteristics Register

261 LSb   Least Significant Bit

262 Mbps   Megabits per second

263 MHz   Mega Hertz

264 MID   MIPI Manufacturer Identification *[MIPI01]*

265 MSb   Most Significant Bit

266 NVMEM   Non-Volatile Memory

Confidential

267	OD	Open Drain
268	PICS	Protocol Implementation Conformance Statement
269	PUR	Pull-Up Resistor
270	SCL	Serial Clock
271	SDA	Serial Data
272	SDR	Single Data Rate
273	SPI	Serial Peripheral Interface
274	SWG	Sensor Working Group, part of MIPI Alliance
275	TSL	Ternary Symbol Legacy
276	TSP	Ternary Symbol for Pure Bus (no I^{2}C Devices)
277	UART	Universal Asynchronous Receiver Transmitter

3   References

3.1   Normative References

278   [MIPI01]   MIPI Alliance, Inc., "MIPI Alliance Manufacturer ID Page", http://mid.mipi.org, last
279   accessed 19 July 2018.

3.2   Informative References

280   [MIPI02]   *Specification for I3C*, version 1.0, MIPI Alliance, Inc., adopted 31 December 2016.

281   [NXP01]   UM10204, *I²C Bus Specification and User Manual*, Rev. 6 – 4, NXP Corporation, April
282   2014.

283   [USB01]   *Universal Serial Bus 3.1 Specification*, Revision 1.0,
284   http://www.usb.org/developers/docs/usb_31_072715.zip, Hewlett-Packard Company *et*
285   *al.*, 26 July 2013.

4   Technical Overview (Informative)

286   This Section generally describes the I3C Bus, the I3C interface, and I3C Master and Slave Devices.

287   I3C is a two-wire bidirectional serial Bus, optimized for multiple sensor Slave Devices and controlled by
288   only one I3C Master Device at a time. I3C is backward compatible with many Legacy I²C Devices, but I3C
289   Devices also support significantly higher speeds, new communication Modes, and new Device roles,
290   including an ability to change Device Roles over time (i.e., the initial Master can cooperatively pass the
291   Master role to another I3C Device on the Bus, if the second I3C Device supports that feature).

292   I3C Basic includes:

293   • Support for many Legacy I²C Slave Devices and messages

294   • **I3C Single Data Rate (SDR) Mode:** New I3C enhanced version of the I²C protocol supporting
295   private messages, and adding two kinds of standard built-in messages:

296   • **Broadcast messages**, which are sent to all I3C Slaves on the Bus

297   • **Direct messages**, which are addressed to specific Slaves

4.1   I3C Fundamental Principles

298   I3C supports several communication formats, all sharing a two-wire interface.

299   The two wires are designated SDA and SCL:

300   • SDA (Serial Data) is a bidirectional data pin

301   • SCL (Serial Clock) is a clock pin

302   An I3C Bus supports the mixing of various Message types:

303   1.   I²C-like SDR Messages, with SCL clock speeds up to 12.5MHz

304   2.   Broadcast and Direct Common Command Code (CCC) Messages that allow the Master to
305   communicate to all or one of the Slaves on the I3C Bus, respectively

306   3.   **NOT SUPPORTED IN I3C BASIC:** HDR Mode Messages, which achieve higher data rates per
307   equivalent clock cycle

308   4.   I²C Messages to Legacy I²C Slaves

309   5.   Slave-initiated START Request to the Master, for example to send an In-Band Interrupt or to
310   request the Master role

311   An example traffic pattern on the I3C Bus is shown in *Figure 4*.

312
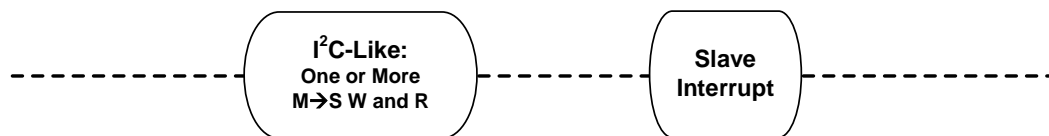
313   **Figure 4 Example of Data Traffic on the I3C Bus**

314   *Figure 5* illustrates how I3C communication is initiated:

315   • All I3C communication occurs within a Frame. The Frame begins with a START, followed by one
316   or more transfers, and a STOP.

317   • For the HDR Modes that are not supported in I3C Basic, but are tolerated by I3C Basic Devices:

318   • First the dedicated Broadcast I3C Address (7'h7E) is issued to all Slaves on the I3C Bus.

319   • Then one of the EnterHDR CCCs is issued, indicating that the Master is entering an HDR
320   Mode. Each HDR Mode has its own EnterHDR CCC.

321   • This is followed by one or more HDR transfers.

322   • HDR Mode is ended by using the HDR Exit Pattern protocol.
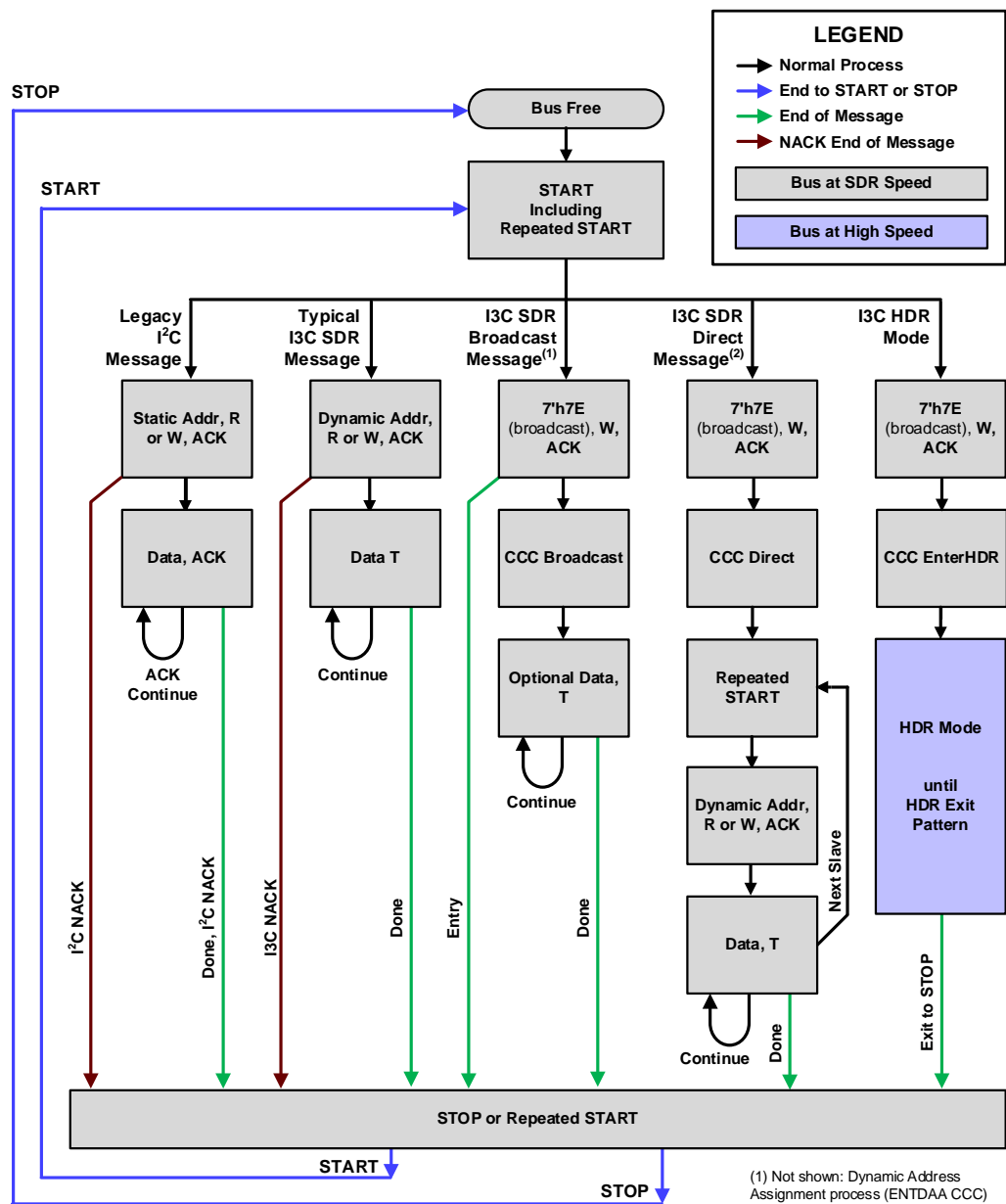
323
324   **Figure 5 I3C Communication Flow**

325 I3C is based on a Frame encapsulation approach. A Frame includes a Data Payload.

326 The I3C Basic Frame always includes at least the START, the Header, the Data, and the STOP.

327 The Header following a START allows for Bus Arbitration. The Master uses the Header to address Slave
328 Device(s). Slave Devices(s) may use the Header Arbitration for multiple purposes: for In-Band Interrupt, for
329 Hot-Join, and for Secondary Master functionality.

330 I3C allows only one Master to have control of the I3C Bus at a time. Mechanisms for handoff of the Master
331 role from one Device to another Device are provided.

4.2   I3C Master and Slave Devices

332 A given I3C Bus always has one Master and one or more Slaves. This Section generally describes I3C Master
333 Devices and I3C Slave Devices.

334 A given I3C Device can be designed to function either solely as an I3C Master, solely as an I3C Slave, or
335 with both I3C Master and I3C Slave capabilities.

336 An I3C Device with both I3C Master and I3C Slave capabilities cannot function as both Master and Slave at
337 the same time, instead it must be configured either as an I3C Slave Device or as an I3C Master Device. Such
338 an I3C Device can be initially configured (initialized) on an I3C Bus either as the Master of that I3C Bus, or
339 as a Slave on that I3C Bus. However for that I3C Bus to function properly, only one of the multiple I3C
340 Devices on the Bus can be initially configured (initialized) as an I3C Master Device. That I3C Device will
341 have the 'Main Master' Device Role, and will be the first I3C Device on the Bus to serve as Current Master;
342 all other I3C Devices and Legacy I^{2}C Devices on the I3C Bus will be initially configured (initialized) as
343 Slaves.

344 I3C introduces the concept of Current Master, defined as the I3C Master Device on the I3C Bus that is
345 functioning as Master (i.e., the one that is controlling the Bus) at the present time. Only one I3C Device on
346 an I3C Bus can serve as Current Master at a time. However after initial Bus configuration the Current Master
347 function can be cooperatively passed from the Current Master to any other I3C Device on the Bus with I3C
348 Master Device capability, using provided I3C commands (CCCs).

349 I3C defines several Master and Slave Device Roles (see *Table 2* and *Table 3*) to reflect the functional
350 capabilities of a given I3C Master or Slave Device. A given I3C Device must support at least one Device
351 Role, and can be designed to support multiple Device Roles. Every I3C Device exposes the Device Roles it
352 supports via its Bus Characteristics Register (BCR, see *Section 5.1.1.2.1*).

353   **Table 2 Roles for I3C Compatible Devices**

Device Type	Device Role	Description
I3C Master[1]	I3C Main Master	Initially configures I3C Bus, has HDR support
	SDR-Only Main Master (I3C Basic)	Initially configures I3C Bus, no HDR support
	I3C Secondary Master	Can Master but currently functioning as Slave
	SDR-Only Secondary Master (I3C Basic)	Can Master but currently functioning as Slave, no HDR support
I3C Slave[2]	I3C Slave	Ordinary I3C Slave, no Master capability
	I^{2}C Slave	No I3C Master or I3C Slave capabilities

Note:

1) *Applies to Master-only Devices. In a Multi-Master context a Master Device may also implement functionality to join the Bus acting in a Slave role.*

2) *Applies to Slave-only Devices. In a Multi-Master context a Slave Device may also implement functionality to join the Bus acting in a Master role.*

4.2.1   I3C Master Device

354 An I3C Bus requires there to be exactly one I3C Device at a time functioning as an I3C Master Device. In
355 I3C terms, this I3C Master Device is the Current Master at that time. In typical applications, the Current
356 Master is the I3C Device on the Bus that sends the majority of the I3C Commands (CCC), addressing either
357 all Slaves (Broadcast CCCs) or specific individual Slaves (Directed CCCs). The Current Master is also the
358 only Device on the I3C Bus allowed to send I²C Messages.

359 In addition to sending I3C Commands and I²C Messages, an I3C Master Device also:

360 - Generates the Bus clock
361 - Manages Pull-Up structures
362 - Manages the Dynamic Address Assignment procedure (including Hot-Join events) while acting as
363   the Main Master
364 - Manages START Requests from I3C Slave Devices on the Bus along with Address Arbitration
365   requests:
366   - Generate In-Band Interrupts
367   - For Hot-Join events
368   - To become Current Master
369 - Supports I²C Legacy Slave Devices
370 - Supports I3C SDR Mode

371 *Figure 6* is a block diagram of a typical generic I3C Master Device.
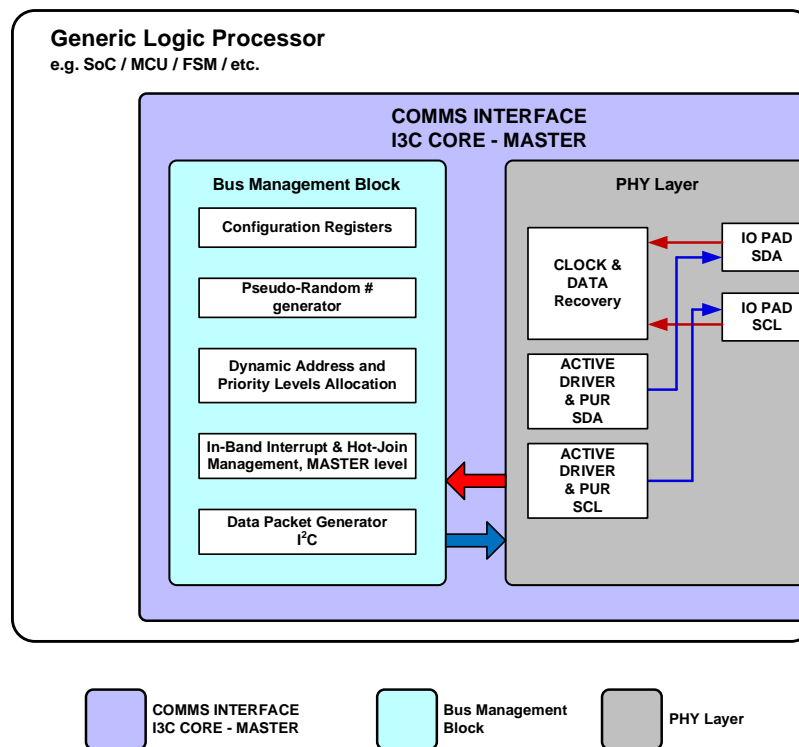
372
373 **Figure 6 I3C Master Device Block Diagram**

4.2.1.1   I3C Master Device Roles

All I3C Master Devices support one of the two Main Master Device Roles, and may also support one of the two Secondary Master Device Roles.

Main Master Device Roles:

- **Main Master:** The I3C Master Device on the I3C Bus that initially configures the I3C Bus and serves as the first Current Master. Only one I3C Device on a given I3C Bus can take the Main Master role, i.e. the role cannot be passed on to any other I3C Device on the I3C Bus.
- **SDR-Only Main Master:** A Main Master that only supports I3C's SDR Mode, i.e. does not support any of the HDR Modes.

Secondary Master Device Roles:

- **I3C Secondary Master:** Any I3C Device on the I3C Bus, other than the Current Master, with I3C Master Capability. There can be multiple Secondary Masters on an I3C Bus at the same time. By definition, a Secondary Master functions as an I3C Slave Device until and unless it eventually becomes Current Master.
- **SDR-Only Secondary Master:** A Secondary Master that only supports I3C's SDR Mode, i.e. does not support any of the HDR Modes.

See also *Table 2* and *Table 3*.

Note:

*Current Master is not formally defined as an I3C Device Role, and is not exposed in the I3C Device's Bus Characteristics Register (BCR, see **Section 5.1.1.2.1**).*

4.2.2   I3C Slave Device

393 An I3C Bus supports up to 11 I3C Slave Devices, though the maximum number of Devices will depend on
394 trace length, capacitive load per Device, and the types of Devices (I^2C vs. I3C) present on the Bus, because
395 these factors affect clock frequency requirements.

396 An I3C Slave Device listens to the I3C Bus for relevant I3C Commands (CCCs) sent by the Current Master,
397 and responds accordingly. This includes all Broadcast Commands (CCC), and any Directed Commands
398 (CCC) addressed specifically to that I3C Slave Device and supported by that I3C Slave Device.

399 In addition to responding to I3C Commands, an I3C Slave Device always supports I3C SDR Mode.

400 In addition, an I3C Slave Device can optionally:

401 • Request In-Band Interrupts
402 • Generate Hot-Join events
403 • Request to become Current Master, if the I3C Slave Device also has I3C Master Device capability

404 While functioning as a Slave, an I3C Slave Device functions in one of the Slave Device Roles detailed in
405 *Section 4.2.2.1*.

406 *Figure 7* is a block diagram of a typical generic I3C Slave Device.
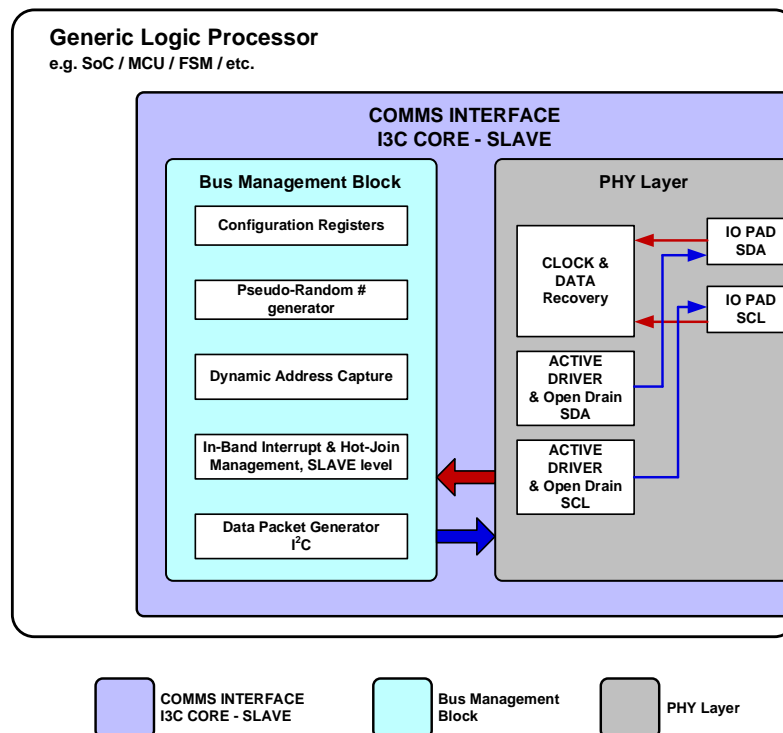
407
408 **Figure 7 I3C Slave Device Block Diagram**

4.2.2.1   I3C Slave Device Roles

409 All I3C Slave Devices support one of the two I3C Slave Device Roles:

410   • **I3C Slave:** An ordinary I3C Slave Device without Master capability.

411   • **SDR-Only I3C Slave:** An I3C Slave without Master capability that only supports I3C's SDR
412   Mode (i.e., does not support any of the HDR Modes).

413 ***Note:***

414   *An additional Slave Device Role is defined for I²C Slave, however this is not relevant for an I3C Slave*
415   *Device.*

416 See also ***Table 2*** and ***Table 3***.

This page intentionally left blank.

5 I3C Protocol

417 This Section specifies the communication protocols for all defined I3C Modes:

418 • **Single Data Rate (SDR) Mode:** See *Section 5.1*

419 • **NOT SUPPORTED IN I3C BASIC: High Data Rate (HDR) Modes:** See *Section 5.2*

420 • **HDR Ternary Symbol Pure-bus (HDR-TSP) Mode**

421 • **HDR Ternary Symbol Legacy-inclusive-bus (HDR-TSL) Mode**

422 • **HDR Double Data Rate (HDR-DDR) Mode**

423 It is important to note that the I3C Bus is always initialized and configured in SDR Mode, never in any of
424 the HDR Modes. (The procedure for entering an HDR Mode from SDR Mode is detailed in *Section 5.2*.)

425 As a result, most of the essential basic I3C protocol specifications are found in *Section 5.1*, including:

426

Subject	Section
Bus Configuration	5.1.1
Bus Communication	5.1.2
Bus Free Condition	5.1.3.2
Bus Idle Condition	5.1.3.4
Bus Initialization and Dynamic Address Assignment Mode	5.1.4
Hot-Join Mechanism	5.1.5
In-Band Interrupt	5.1.6
Secondary Master Functions	5.1.7
Timing Control	5.1.8
Common Command Codes (CCC)	5.1.9
Error Detection and Recovery Methods	5.1.10
High Data Rate (HDR) Modes	5.2

5.1 Single Data Rate (SDR) Mode

427 This Section specifies the communication protocols for Single Data Rate (SDR) Mode.

428 SDR Mode is the default Mode of the I3C Bus, and is primarily used for private messaging from the Current
429 Master Device to Slave Devices. SDR Mode is also used to enter other Modes, sub-Modes, and states (as
430 described in *Section 5.1* and *Section 5.2*); and for built-in features such as Common Commands (CCCs), In-
431 Band Interrupts, and transition from I²C to I3C by assignment of a Dynamic Address.

432 I3C SDR Mode is significantly similar to the I²C protocol *[NXP01]* in terms of procedures and conditions,
433 and as a result I3C Devices and many Legacy I²C Slave Devices (but not I²C Master Devices) can coexist on
434 the same I3C Bus. However SDR Mode also includes numerous new features not present in I²C. For the
435 procedures and conditions that I3C shares with I²C, SDR Mode closely follows the definitions in the I²C
436 Specification. I²C traffic from an I3C Master to an I²C Slave will be properly ignored by all I3C Slaves,
437 because the I3C protocol is designed to allow I²C traffic. I3C traffic from an I3C Master to an I3C Slave will
438 not be seen by most Legacy I²C Slave Devices, because the I²C Spike Filter is opaque to I3C's higher clock
439 speed.

5.1.1   Bus Configuration

440   The I3C Bus can be configured as the link among several clients, in a flexible and efficient manner. At the
441   system architecture level, eight roles are defined for I3C compatible Devices (see *Table 2*).

442   An example block diagram of I3C interconnections is shown in *Figure 8*. In this diagram the color blue
443   indicates Devices with a Master role, the color pink indicates Devices with an I3C Slave role, and the color
444   purple indicates Devices with an I^{2}C Slave role. Note that I3C Secondary Master Devices are shaded from
445   blue to pink, illustrating their ability to function in both Master and Slave roles (at different times).

446
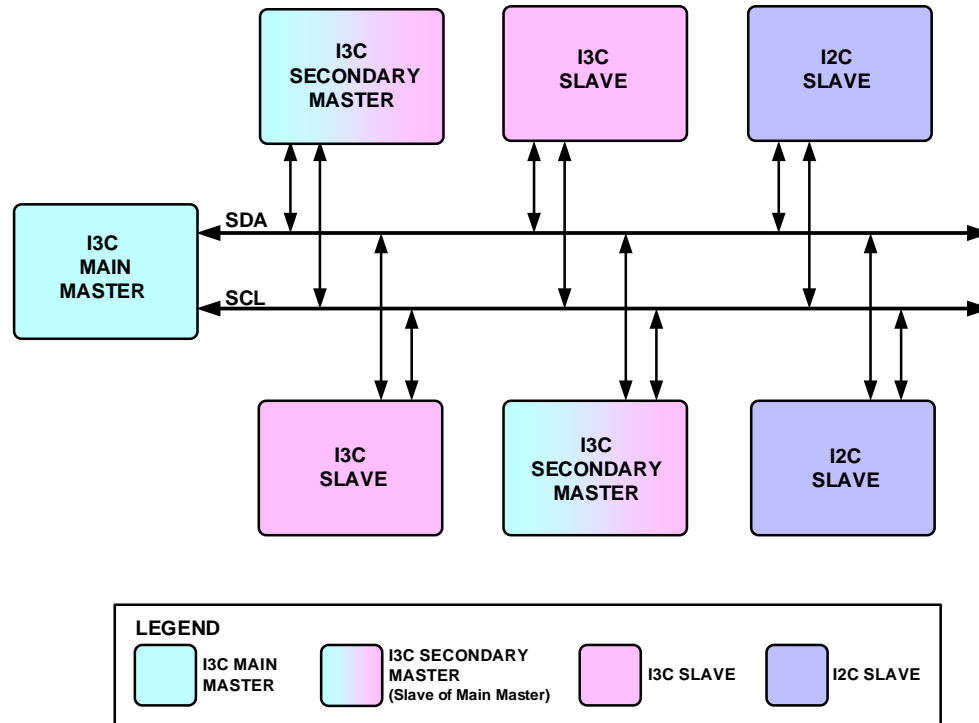
447   **Figure 8 I3C Bus with I^{2}C Devices and I3C Devices**

448   I3C compatible Devices may have diverse features, as appropriate for their function within the I3C Bus.
449   Depending on the I3C Bus' system design, it may not be necessary for all features of a given Device to be
450   enabled for any particular Bus instantiation. However, the enabled features of every I3C compatible Device
451   shall be described in Characteristics Registers associated with the Device, as described in *Section 5.1.1.2*.
452   The I3C Main Master shall obtain the Characteristics of any Legacy I^{2}C Devices on the I3C Bus before power
453   up (e.g. the fixed Address of each Legacy I^{2}C Device present on the Bus).

454   At every start-up from a powered-down state, the Main Master shall assign a unique Dynamic Address to
455   every Device on the Bus, including itself. The Dynamic Address assignment procedure is described in
456   *Section 5.1.4*. Dynamic Addresses create a priority ranking of the Devices' In-Band Interrupts. Any
457   Secondary Masters present on the I3C Bus shall be made aware of the Dynamic Address assignments and
458   Characteristic Registers associated with each I3C compatible Device on the Bus via Common Command
459   Codes as described in *Section 5.1.9*.

5.1.1.1 I3C Device Characteristics

460 The configuration of an I3C Bus will depend upon the Characteristics of the I3C Devices intended to be active on that I3C Bus. Therefore, an active I3C
461 Device playing a given Role in a given I3C Bus instantiation shall fulfill all responsibilities for that Role, as detailed in *Table 3*.

462

Table 3 I3C Devices Roles vs Responsibilities

Responsibilities / Features	Comments	Roles					
		Main Master	Secondary Master	SDR Only Main Master	SDR Only Secondary Master	Slave	SDR Only Slave
Manages SDA Arbitration	For Address Arbitration, In-Band Interrupt, Hot-Join, Dynamic Address, as appropriate	Y	Y	Y	Y	N	N
Dynamic Address Assignment	Master assigns Dynamic Address	Y	N	Y	N	N	N
Hot-Join Dynamic Address Assignment	Master capable of assignment Dynamic Address after Hot-join	Y	Optional	Y	Optional	N	N
Self Dynamic Address Assignment	Only Main Master can self-assign a Dynamic Address	Y	N	Y	N	N	N
Static I²C Address[1]	–	N/A	Optional	N/A	Optional	Optional	Optional
Memory for Slaves' Addresses and Characteristics	Retaining registers	Y	Y	Y	Y	N	N
HDR Exit Pattern Generation capable[2]	Able to generate the HDR Exit Pattern on the Bus for error recovery	Y	Y	Y	Y	N	N
HDR Tolerant	Recognizes HDR Exit Pattern	Y	Y	Y	Y	Y	Y
Note: 1) A Static Address may be used to more quickly assign a Dynamic Address. See *Section 5.1.4*. 2) All Slaves require an HDR Exit Pattern Detector, even Slaves that are not HDR capable							

463

464 The I3C protocol supports a subset of I²C Slave features. For example, an I3C Slave can have a Static Address
465 but also support Dynamic Addressing. A Device shall not have a 50 ns filter enabled when used in an I3C
466 Bus operating at full clock speed. These differences are summarized in **Table 4**. When used in an I3C system,
467 I3C Slaves shall enable or disable appropriate I²C features as shown in **Table 4**.

468   **Table 4 I²C Features Allowed in I3C Slaves**

I²C Feature When Used on an I3C Bus	Required on I3C	Desirable on I3C	Not Used on I3C	Not Allowed on I3C	Note
Fm Speed	–	–	X	–	3
Fm+ Speed	X	–	–	–	–
HS Speed	–	–	X	–	3
UFm Speed	–	–	X	–	3
Static I²C Address	–	X	–	–	–
50ns Spike Filter	–	–	X	X (Shall disable)	3
Clock Stretch	–	–	–	X	–
20mA Open Drain Driver	–	–	X	–	1, 3
Matches I²C AC Timing	–	–	X	–	2, 3
I²C Extended Address (10 bit)	–	–	X	–	3
I3C Reserved Addresses	–	–	–	X	–

Note:

1) *See* **Table 54** *and* **Table 57**

2) *I3C drive and timing requirements are different from I²C*

3) *If an I3C Slave has I²C features intended for use on an I²C Bus, then they will not be used on an I3C Bus. As stated in* **Section 5.1.1.1**, *once the Slave sees a 7'h7E, it will disable I²C features that are not used by I3C.*

469 Performance of an I3C Bus is heavily dependent upon any I²C-only Devices that may be connected to that
470 Bus. Consequently, all I²C-only Devices permitted on any instantiation of an I3C Bus must be compliant with
471 one of the categories detailed in **Table 5**. Furthermore (and as referenced in **Table 56**), no I²C or I3C Device
472 present on an I3C Bus shall have a fixed I²C Address that matches any of the Addresses associated with Error
473 Type S0 (see **Section 5.1.10.1.1**).

474   **Table 5 Legacy I²C-Only Slave Categories and Characteristics**

Index Specific	I²C-Only Devices Index 0	I²C-Only Devices Index 1	I²C-Only Devices Index 2
50ns IO Spike Filter[1]	Y	N	N
Max SCL clock frequency (f$_{SCL}$) tolerant[2]	N/A	Y	N

Note:

1) *Allows tolerance of HDR Modes and SDR at SCL High periods of t$_{DIG\_H\_MIXED}$ or less*

2) *Allows compliance up to maximum SDR SCL clock frequency (f$_{SCL}$)*

5.1.1.2   I3C Characteristics Registers

475 I3C Characteristics Registers describe and define an I3C compatible Device's capabilities and functions on
476 the I3C Bus, as the Device services a given system. Devices without I3C Characteristics Registers shall not
477 be connected to a common I3C Bus.

478 There are three Characteristics Register types:

479 • **Bus Characteristics Register** (BCR, see *Section 5.1.1.2.1*)

480 • **Device Characteristics Register** (DCR, see *Section 5.1.1.2.2*)

481 • **Legacy Virtual Register** (LVR, see *Section 5.1.1.2.3*)

482 Every I3C compatible Device shall have associated Characteristics Registers, depending on the Device type
483 as described below:

484 • Every I3C compliant Device (as defined in *Table 3*) shall have one Bus Characteristics Register,
485   and one Device Characteristics Register.

486 • Every Legacy I^{2}C Device to be connected to an I3C Bus shall have one associated Legacy Virtual
487   Register. Since these are Legacy Devices, it is understood that this register will exist virtually, for
488   example as part of the Device's driver.

5.1.1.2.1   Bus Characteristics Register (BCR)

489   Each I3C Device that is connected to the I3C Bus shall have an associated read-only Bus Characteristics
490   Register (BCR). This read-only register describes the I3C compliant Device's role and capabilities for use in
491   Dynamic Address assignment and Common Command Codes. The bits within the BCR shall conform to the
492   Descriptions presented in in *Table 6*.

493   **Table 6 Bus Characteristics Register (BCR)**

BIT	Name	Description
BCR [7]	**Device Role [1]**	2'b00 – I3C Slave 2'b01 – I3C Master[1]
BCR [6]	**Device Role [0]**	2'b10 – Reserved for future definition by MIPI 2'b11 – Reserved for future definition by MIPI
BCR [5]	**MIPI Reserved**	0 – Default
BCR [4]	**Bridge Identifier[2]**	0 – Not a Bridge Device 1 – Is a Bridge Device
BCR [3]	**Offline Capable[3]**	0 – Device will always respond to I3C Bus commands 1 – Device will not always respond to I3C Bus commands
BCR [2]	**IBI Payload**	0 – No data byte follows the accepted IBI 1 – Mandatory one or more data bytes follow the accepted IBI. Data byte continuation is indicated by T-Bit, as described in *Section 5.1.2.3.4*
BCR [1]	**IBI Request Capable**	0 – Not Capable 1 – Capable
BCR [0]	**Max Data Speed Limitation[4]**	0 – No Limitation 1 – Limitation

Note:

1) *For an I3C Device acting as I3C Main Master, the BCR Device Role bits will contain the value 2'b01.*

2) *Bridge Devices are required to comply with the MIPI Specification for I3C v 1.0 **[MIPI02]**.*

3) *Offline Capable Devices retain the Dynamic Address, and are specified in **Section 2.2**.*

4) *Master shall use the GETMXDS CCC to interrogate the Slave for specific limitation.*

5.1.1.2.2　Device Characteristics Register (DCR)

Each I3C Device that is connected to the I3C Bus shall have an associated read-only Device Characteristics Register (DCR). This read-only register describes the I3C compliant Device type (e.g. accelerometer, gyroscope, etc.) for use in Dynamic Address assignment and Common Command Codes. The bits within the DCR shall conform to the Descriptions presented in *Table 7*.

Table 7 I3C Device Characteristics Register (DCR)

Bit	Name	Description
DCR [7]	**Device ID [7]**	255 available codes for describing the type of sensor, or Device.
DCR [6]	**Device ID [6]**	Examples: Accelerometer, gyroscope, composite Devices.
DCR [5]	**Device ID [5]**	
DCR [4]	**Device ID [4]**	Default value is 8'b0: Generic Device
DCR [3]	**Device ID [3]**	
DCR [2]	**Device ID [2]**	
DCR [1]	**Device ID [1]**	
DCR [0]	**Device ID [0]**	

5.1.1.2.3　Legacy Virtual Register (LVR)

Each Legacy I^{2}C Device that can be connected to the I3C Bus shall have an associated read-only Legacy Virtual Register (LVR) describing the Device's significant features. Since these are Legacy I^{2}C Devices, it is understood that this register will exist virtually, for example as part of the Device's driver. When Legacy I^{2}C Devices are present on an I3C Bus, LVR data determines allowed Modes and maximum SCL clock frequency. The bits within the LVR shall conform to the descriptions in *Table 8*.

All LVRs shall be established by the higher-level entity controlling the I3C Bus, and transferred to the I3C Bus Main Master prior to Bus configuration. The LVR content for all I^{2}C Devices is always known by the Main Master. The LVR information can be transferred to Secondary Masters by using the DEFSLVS CCC (see *Section 5.1.9.3.7*).

Table 8 Legacy I^{2}C Virtual Register (LVR)

Bit	Name	Description
LVR [7]	**Legacy I^{2}C only [2]** (Indexed in *Table 5*)	3'b000 – Index 0 3'b001 – Index 1 3'b010 – Index 2
LVR [6]	**Legacy I^{2}C only [1]** (Indexed in *Table 5*)	3'b011 – Index 3 (Reserved) 3'b100 – Index 4 (Reserved) 3'b101 – Index 5 (Reserved)
LVR [5]	**Legacy I^{2}C only [0]** (Indexed in *Table 5*)	3'b110 – Index 6 (Reserved) 3'b111 – Index 7 (Reserved)
LVR [4]	**I^{2}C Mode Indicator**	0 – I^{2}C Fm+ 1 – I^{2}C Fm
LVR [3]	MIPI Reserved	
LVR [2]	MIPI Reserved	15 available codes for describing the Device capabilities and function on the sensors' system.
LVR [1]	MIPI Reserved	
LVR [0]	MIPI Reserved	

5.1.2   Bus Communication

509   The primary protocol and Mode of I3C is SDR (Single Data Rate) Mode. The SDR protocol is based on the
510   I²C standard protocol *[NXP01]*, with a few notable variations:

511   • The I3C START and STOP (as shown in *Figure 37* and *Figure 39*, respectively) are identical to
512   the I²C START and STOP in their signaling, but they may vary from I²C in their timing. Compare
513   *Table 58* vs. *Table 57*.

514   • The I3C Address Header is identical to the I²C Address Header in bit form and in signaling, but it
515   may vary from I²C in its timing. See *Section 5.1.2.2*, *Table 58*, and *Table 59*.

516   • The Data 9-bit Words use the same bit count as I²C, but differ in the ninth bit, as explained in
517   *Section 5.1.2.3*.

518   • The I3C Data is normally sent using Push-Pull signaling, whereas I²C uses Open Drain signaling.
519   There are exceptions, including use in DAA (see *Section 5.1.4.2*) and an allowance for the Master
520   and Slave to agree to let the Slave to return Read operations using Open Drain. The use of Push-
521   Pull impacts transitions between Master and Slave. See *Section 5.1.2.3* and its sub-sections.

522   • In I3C, the SCL line is only driven by the Master. Normally this drive is Push-Pull, but it can also
523   be Open Drain.

524   Because the bit count is the same for Address Header and Data Word, the I3C Slave only needs to know
525   whether a Message is an I3C Message (vs. is an I²C Message) if the Message is addressed to that Slave (either
526   directly or by Broadcast).

527   An I3C Message is defined as everything from the initial START (or Repeated START) to the next Repeated
528   START or STOP.

529   An I3C Message is an SDR Message if:

530   • The Address in the Address Header is 7'h7E (the I3C Broadcast Address). All I3C Slaves shall
531   match Address value 7'h7E. No I²C Slave will match Address 7'h7E, because that value is
532   reserved and unused in I²C.

533   • The Address in the Address Header matches the Slave's Dynamic Address (as assigned by the I3C
534   Master per *Section 5.1.4*). All I3C Slaves shall match their own Dynamic Address. (It is permitted
535   to then NACK the Header if needed.)

536   All I3C Slaves shall ignore all Messages with Addresses other than 7'h7E or the I3C Master Assigned
537   Address, and await either the Repeated START or the STOP. I3C Slaves shall not transmit on the Bus in
538   response to a non-matching Address.

539   ***Note:***

540   *Legacy I²C Slaves will ignore any Message not addressed to them, and will await the next START or*
541   *STOP. Per **Section 5.1.2.4**, Legacy I²C Slaves may also not see some or all I3C Messages and*
542   *Modes due to the speed of SCL signaling.*
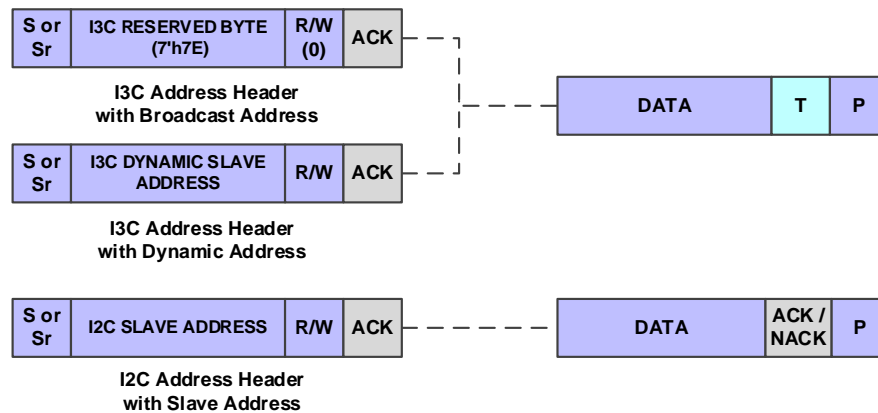
Figure 9 Address Header Comparison

5.1.2.1   Role of I3C Slave

The I3C Slave does not have to know whether it is on a Legacy I²C Bus or an I3C Bus. If it has a Legacy I²C Static Address, then it may participate using that Address up until (and if) it is assigned a Dynamic Address. Once assigned a Dynamic Address, unless asked to Reset, it shall only operate as an I3C Slave.

An I3C Capable Slave may act as an I²C Device before it gets its Dynamic Address (DA) assigned. However, the Slave shall ACK the START with address 7'h7E. (The only exception would be if the Slave is choosing to remain an I²C-only Device on a given bus or use, in which case it would leave its 50ns Spike Filter enabled.)

Slaves that do recognize START and the 7'h7E address may see any CCC, not just ENTDAA (see *Section 5.1.9.3.4*), and shall behave as follows:

- The Slave shall appropriately process all required Broadcast CCCs, including ENTDAA, RSTDAA (see *Section 5.1.9.3.3*), ENEC (see *Section 5.1.9.3.1*), and DISEC (see *Section 5.1.9.3.1*).

    Examples of appropriate processing:

    - RSTDAAA has no effect, since no Dynamic Address is assigned
    - If the Slave would never issue a Master Request, then DISEC for Master Request can be ignored.

- The Slave shall recognize the CCCs ENTHDR0 through ENTHDR7 (see *Section 5.1.9.3.9*), and then wait for the HDR Exit Pattern.

- The Slave may choose to understand and process the SETDASA CCC (see *Section 5.1.9.3.10*) when its Static Address matches.

- The Slave may choose to understand and process the SETAASA CCC (see *Section 5.1.9.3.22*).

- The Slave shall disregard all Directed CCC commands, but shall properly recognize the ends of Directed CCCs (either repeated START followed by 7'h7E, or STOP).

- When no Dynamic Address has been assigned yet, the Slave may either support or ignore non-Required and Conditionally Required Broadcast CCCs.

    This is true even if the Slave supports any of these CCCs after being assigned a Dynamic Address. For example, the Slave may choose to only support Test Mode only when no Dynamic Address is assigned.

- The I3C Slave shall ignore S0 type errors related to incorrect addresses only (see *Section 5.1.10.1.1*).

574   The role of an I3C Slave shall be as follows:

575   1.   Following a START or a Repeated START, at any speed conforming to *Section 6* of this
576   Specification, the I3C Slave shall attempt to match the Address to the I3C Broadcast Address
577   (7'h7E) or to its own Dynamic Address, once assigned. If a match is found, then the I3C Slave
578   shall treat that Message as I3C SDR.

579   2.   If the Message is addressed to the Slave's Dynamic Address, then the Slave may ACK or passively
580   NACK the Address Header:

581   a.   If the Slave ACKs the Address Header, then the Slave shall process the Message as I3C SDR,
582   following all rules as outlined in this Section.

583   b.   If the Slave NACKs the Address Header (does not drive the ACK bit Low), then the Slave
584   may disregard any bits that follow, up until the next Repeated START or STOP.

585   3.   If the Message is addressed to the I3C Broadcast Address (7'h7E), with a Write (RnW bit is 0),
586   then the Slave shall process that Message at least through the first byte of data (if any data is
587   present in the Message):

588   a.   If there is a byte of data in a 7'h7E Broadcast Message, then the Message is a CCC (Common
589   Command Code) Command, per *Section 5.1.9*.

590   b.   The I3C Slave shall process all applicable Required CCC commands, per *Section 5.1.9.3*. A
591   command may be either Always Required, or only Contextually Required, per *Table 9*.

592   c.   If the CCC command changes the Mode of the I3C Bus, then the I3C Slave shall handle the
593   new Mode in one of two ways.

594   **Either:**

595   i.   If the new Mode is Dynamic Address Assignment Mode (see *Section 5.1.4*), and required
596   for all I3C Slaves, then the Slave shall participate if it does not have a current Dynamic
597   Address; otherwise, the Slave shall await the STOP that indicates exit from Dynamic
598   Address Assignment Mode.

599   **Or:**

600   ii. If the new Mode is HDR (High Data Rate) Mode, then the Slave may either enter into
601   HDR Mode if it supports that specific HDR Mode, or else enable its HDR Exit Pattern
602   detector (per *Section 5.2.1* and *Section 5.2.1.3*) to await the exit from HDR Mode.

603   4.   If the Message is not addressed to the I3C Broadcast Address (7'h7E) or to the Slave's Dynamic
604   Address, then the I3C Slave shall await either a Repeated START or a STOP. The Slave may
605   record/monitor the bits as they pass (if desired), but the only obligation is to wait for either a
606   Repeated START or a STOP:

607   •   A Repeated START is defined by the SDA line changing from High to Low while the SCL line
608   is High, per *Section 6*.

609   •   A STOP is indicated by the SDA line changing from Low to High while the SCL line is High,
610   per *Section 6*.

611   In both cases, I3C timing may or may not be the same as with I^{2}C.

Copyright © 2016-2018 MIPI Alliance, Inc.
                                        All rights reserved.
                                        Confidential

5.1.2.2   I3C Address Header

612 The I3C Address Header follows either a START, or a Repeated START. The format is the same as I²C: 7
613 bits of Address, 1 bit of RnW, and 1 bit of ACK/NACK.

614 The Address Header following a START is an Arbitrable Address Header, as explained in *Section 5.1.2.2.1*.
615 This means the START and at least the first Address bit and ACK/NACK are issued on SDA using Open
616 Drain Bus drive, similar to I²C. However, some of the Arbitrable Address Header may be driven on SDA
617 using Push-Pull and higher speed (see *Section 5.1.2.2.2*).

618 The Address Header following a Repeated START is always driven on SDA using Push-Pull, with the
619 exception of the ACK/NACK (see *Section 5.1.2.2.4*).

620 Using the I3C Arbitrable Address Header, I3C Slaves may transmit any of three requests to the I3C Master:

621 1. An In-Band Interrupt, per *Section 5.1.6*. This is equivalent to toggling a wire to get the Master's
622   attention. The In-Band Interrupt request shall be made using the Slave's Dynamic Address with a
623   RnW bit of 1.

624 2. A Secondary Master request, per *Section 5.1.7*. An I3C Slave shall not make such a request unless
625   it is marked as a Secondary Master in its BCR register, per *Section 5.1.1.2.1*. The Secondary
626   Master request shall be made using the Slave's Dynamic Address with a RnW bit of 0.

627 3. A Hot-Join request, per *Section 5.1.5*. An I3C Slave shall only make such a request when
628   becoming available after the I3C Bus is operational. The Hot-Join request shall be made using the
629   special Hot-Join Address of 7'h02.

630 The I3C Slaves shall make these requests to the I3C Master in only two Bus conditions:

631 1. A START (but not a Repeated START) is issued on the Bus following a Bus Available Condition,
632   per *Section 5.1.3*. The Slave may transmit its Dynamic Address or the Hot-Join Address (7'h02)
633   following the START, by adhering to the I3C Address Arbitration rules per *Section 5.1.2.2.1*.

634 2. The Bus is in a Bus Available Condition, per *Section 5.1.3*, so the Slave may issue a START by
635   pulling the SDA Low.

636   a. If the Slave pulls the SDA Low, then the Master shall pull SCL Low within a best-efforts
637   period of time, where that time is not explicitly defined.

638   b. The Master shall also pull SDA Low (overlapping the Slave pulling it Low).

639   c. Once the Master has pulled the SCL Low, the Slave shall control the SDA line in Open Drain
640   mode (i.e., either pull Low, or release High).

641   d. The Slave may then issue its Address in the normal way (condition 1 above).

5.1.2.2.1   I3C Address Arbitration

642 An Address Header following a START (but not a Repeated START) is subject to Arbitration, meaning both
643 the Master and one or more Slaves may attempt to drive an Address onto the Bus, using SDA. Such Address
644 Headers are defined as Arbitrable Address Headers.

645 The Arbitration model follows the common Open Drain approach. All Devices (whether Master or Slave)
646 that are transmitting an Address shall then follow the same rule:

647 1.   If the current bit to transmit is a 0, then the Device shall drive SDA Low after the falling edge of
648   SCL and hold Low until the next falling edge of SCL.

649   ***Note:***

650   *Other Devices may also be driving SDA Low, but that is acceptable.*

651 2.   If the current bit to transmit is a 1, then the Device shall not drive SDA, but rather shall High-Z
652   SDA on the falling edge of SCL.

653   a.   Additionally, the Device shall monitor the SDA on the rising edge of SCL to determine
654   whether another Device has driven SDA Low.

655   b.   If another Device has driven the SDA Low, then the Device has "lost" the Arbitration and
656   shall not further participate in this Address Header. That is, the Device shall not transmit any
657   more bits, but may wait for a future START Condition (but not a Repeated START
658   Condition).

5.1.2.2.2   I3C Address Arbitration Optimization

659 I3C Address Arbitration may optionally be optimized, as detailed in this Section.

660 As previously described in ***Section 5.1.2.2***, an I3C Master Device assigns 7-bit Dynamic Addresses with
661 values in the range 7'h03 to 7'h7B. However because the I3C Master treats the entire 9-bit Arbitrable Address
662 Header as Open Drain, it has no way to detect whether a Slave Device might be transmitting its own Address
663 during some (or all) of the Address Header.

664 Note that for I3C Secondary Masters and I3C In-Band Interrupt Slaves, the I3C Master is free to restrict
665 assigned Dynamic Addresses to the lower half of the available range (7'h03 to 7'h3F), thus leaving the value
666 of Address bit A6 (the first Address bit after the START) as value 0 in all assigned Dynamic Addresses.

667 Having restricted Dynamic Addresses in this manner, the I3C Master may then optionally optimize the
668 Arbitrable Address Header as follows:

669 • If the I3C Master is transmitting a 1 value (i.e. High-Z on SDA), as it would when transmitting
670   7'h7E, then it can monitor SDA on the rising edge of SCL. If SDA has the value 1 (i.e. if SDA is
671   not being driven by any Slave), then the I3C Master may optionally transmit the remainder of the
672   Address Header (up until the ACK/NACK) in Push-Pull mode. See ***Figure 10***, upper waveform.

673 • If the I3C Master is transmitting a 0 value (i.e. is driving SDA Low), or if SDA was driven Low by
674   a Slave, then the I3C Master shall transmit the remainder of the Address Header using Open Drain.

675 • If the I3C Master intends to transmit solely to other I3C Devices (i.e. not to any I^{2}C Devices), then
676   it may optionally maintain the SCL pulse width below 50ns, with the result that any I^{2}C Devices
677   present on the Bus will see only a 0 value. This shorter pulse width produces a higher data rate,
678   because for Open Drain Arbitration only the Low time is extended. See ***Figure 10***, middle
679   waveform.

680 • If the I3C Master intends to transmit to any I^{2}C Devices, then it must use the slower I^{2}C timing.
681   See ***Figure 10***, lower waveform.

682 The upper waveform of *Figure 10* illustrates this optimization. When the I3C Master sees a value of 1 for
683 Address bit A6, but previously made sure that all Dynamic Addresses assigned to I3C In-Band Interrupt
684 Slaves have a 0 in bit A6, then the I3C Master knows that the line was not being driven by any such Slave.

I3C Address Header (A5:A0; R/W) In Push-Pull
I^{2}C on Bus, but excluded from transaction

I3C Address Header (A5:A0; R/W) In Open-Drain
I^{2}C on Bus, but excluded from transaction

I3C Address Header (A5:A0; R/W) In Open-Drain
Includes I^{2}C in transaction

685
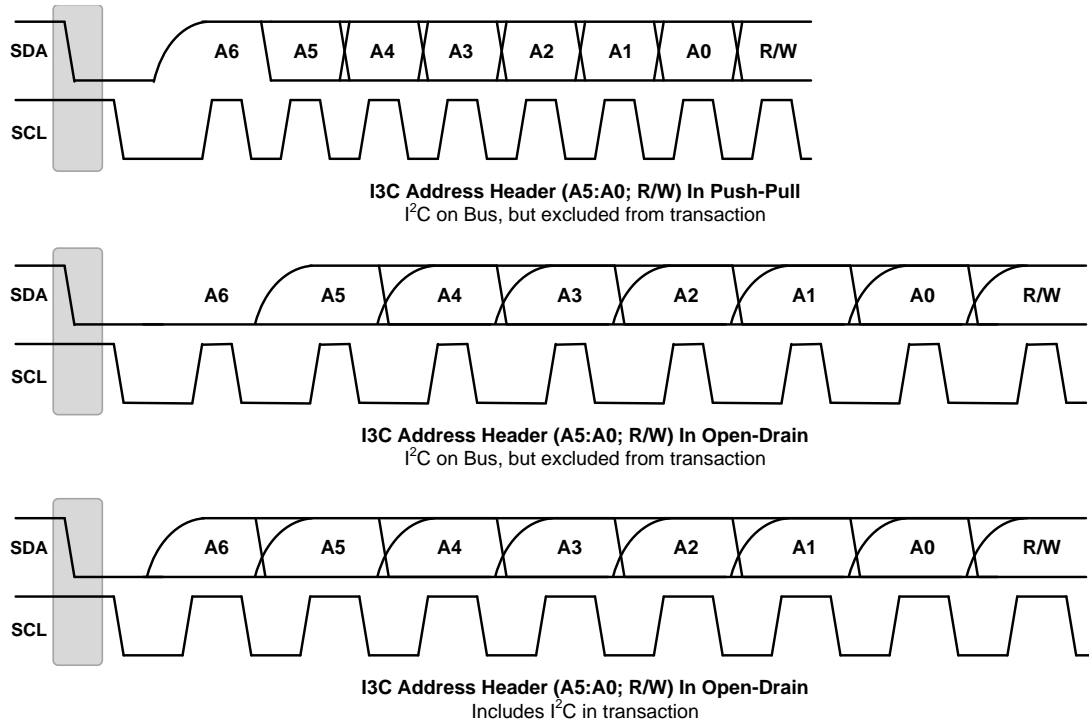
686 **Figure 10 Address Arbitration During Header**

5.1.2.2.3   Consequence of Master Starting a Frame with an I3C Slave Address

687 The I3C Master normally should start a Frame with 7'h7E (for all I3C Messages) or an I^{2}C Static Address
688 (when sending only to a Legacy I^{2}C Slave).

689 In both cases the Address may be arbitrated, and so the Master shall monitor to see whether an IBI, Mastership
690 request (Slave requests to become the Master), or Hot-Join request has been made.

691   • If not, then the Master may proceed normally.

692   • If so, then the Master may ACK or NACK that request and then proceed accordingly.

693 If the Master chooses to start an I3C Message with an I3C Dynamic Address, then special provisions shall
694 be made because that same I3C Slave may be initiating an IBI or Mastership request. So, one of three things
695 may happen:

696 1. The Addresses match, but the difference is caught on the RnW bit, and the Master was Writing
697   (RnW=0); so the Master wins (IBI with RnW=1 loses), and proceeds normally.

698 2. The Addresses match, but the difference is caught on the RnW bit, and the Master was Reading
699   (RnW=1); so the Master loses, and must ACK or NACK the Mastership request (RnW=0).

700 3. The Addresses match as do the RnW and so neither Master nor Slave shall ACK since both are
701   expecting the other side to.

702   a. This is a problem since the Master cannot tell whether the NACK was due to this condition,
703   or simply because the Slave may have chosen to NACK it.

704   b. The Master shall transmit the Slave Address again after a Repeated START (the next one or
705   any one prior to a STOP in the Frame). This allows it to determine which condition occurred
706   and to avoid a deadlock.

5.1.2.2.4　　Address Header Following a Repeated START is Push-Pull

The Address transmitted by the I3C Master following a Repeated START shall not be arbitrated. That is, no I3C Slave shall attempt to transmit is own Dynamic Address nor the Hot-Join Address following a Repeated START.

As a result, the Address Header (i.e., the 7 bits of Address plus the RnW bit) shall be transmitted on SDA using Push-Pull mode when the Message is not to a Legacy I²C Slave. The ACK/NACK bit that follows the RnW bit is always Open Drain to allow the Slave to ACK or passively NACK its Address.

5.1.2.2.5　　I3C Slave Address Restrictions

The I3C Slave Address space is dependent on decisions by the Master. That is, the Master may choose the Dynamic Addresses from a set of values, observing optional and non-optional restrictions as follows. These restrictions are also illustrated in *Table 9*.

- The I3C Master shall not use any of 7'h00, 7'h01, 7'h02, 7'h7E, 7'h7F. All are reserved for I3C.
- The I3C Master shall not use any of 7'h3E, 7'h5E, 7'h6E, 7'h76, 7'h7A, 7'h7C, 7'h7F. All are prohibited for detecting an error in the Broadcast Address (7'h7E).
- The I3C Master may choose to not use 7'h03, marked in I²C as reserved.
- The I3C Master shall not use 7'h04, 7'h05, 7'h06, 7'h07 if any Legacy I²C Devices are present on the Bus that support I²C "High-Speed Mode".
- The I3C Master shall not use 7'h7C or 7'h7D if any Legacy I²C Devices are present on the Bus that support I²C Device ID Mode.
- The I3C Master shall not use 7'h78, 7'h79, 7'h7A, 7'h7B if any Legacy I²C Devices are present on the Bus that support I²C Extended Address Mode and have an Extended Address or would be impacted by the Extended Address mechanism.

727

Table 9 I3C Slave Address Restrictions

Slave Dynamic Address		Restriction	Description
Binary	**Hex**		
000 0000	7'h00	**Shall not use**	I3C Reserved
000 0001	7'h01	**Shall not use**	I3C Reserved: For use with SETDASA CCC in special Point-to-Point Communication. See *Section 5.1.9.3.10*.
000 0010	7'h02	**Shall not use**	I3C Reserved: Hot-Join Address
000 0011	7'h03	**Optional**	Marked 'Reserved' by I²C
000 0100	7'h04	**Conditional**	Available for use only if no Legacy I²C Devices supporting I²C "High-Speed Mode" are present on the Bus
000 0101	7'h05		
000 0011	7'h06		
000 0011	7'h07		
000 1000 – 011 1101	7'h08 – 7'h3D	**Available for use**	54 Addresses
011 1110	7'h3E	**Shall not use**	I3C Reserved: Broadcast Address single bit error detect
011 1111 – 101 1101	7'h3F – 7'h5D	**Available for use**	31 Addresses
101 1110	7'h5E	**Shall not use**	I3C Reserved: Broadcast Address single bit error detect
101 1111 – 110 1101	7'h5F – 7'h6D	**Available for use**	15 Addresses
110 1110	7'h6E	**Shall not use**	I3C Reserved: Broadcast Address single bit error detect
110 1111 – 111 0101	7'h6F – 7'h75	**Available for use**	7 Addresses
111 0110	7'h76	**Shall not use**	I3C Reserved: Broadcast Address single bit error detect
111 0111	7'h77	**Available for use**	1 Address
111 1000	7'h78	**Conditional**	Available for use only if no Legacy I²C Devices are present on the Bus that both a) support I²C "Extended Address Mode", and b) either have an Extended Address, or would be impacted by the Extended Address mechanism
111 1001	7'h79		
111 1010	7'h7A	**Shall not use**	I3C Reserved: Broadcast Address single bit error detect
111 1011	7'h7B	**Conditional**	Available for use only if no Legacy I²C Devices are present on the Bus that both a) support I²C "Extended Address Mode", and b) either have an Extended Address, or would be impacted by the Extended Address mechanism
111 1100	7'h7C	**Shall not use**	I3C Reserved: Broadcast Address single bit error detect (Also not available for use if any Legacy I²C Devices supporting I²C "Device ID Mode" are present on the Bus.)
111 1101	7'h7D	**Conditional**	Available for use only if no Legacy I²C Devices supporting I²C "Device ID Mode" are on the Bus
111 1110	7'h7E	**Shall not use**	I3C Reserved: Broadcast Address
111 1111	7'h7F	**Shall not use**	I3C Reserved: Broadcast Address single bit error detect

5.1.2.3   I3C SDR Data Words

728   In I3C SDR, the Data Words match I²C only in the sense that they are both 9 bits long. I3C SDR Data Words
729   differ from I²C in three ways, as detailed in *Sub-Sections 5.1.2.3.1*, *5.1.2.3.2*, and *5.1.2.3.4*.

730   In summary:

731   1. **Handoff from Address ACK to SDR Master Write Data:** When performing an SDR Write, the
732   handoff from the Slave's Address Header ACK to the Master's first Data bit is different in I3C.
733   I²C is Open Drain, so overlap from the ACK Low into the first bit is harmless. By contrast, I3C is
734   Push-Pull and so this handoff is specified (see *Section 5.1.2.3.1*).

735   2. **Ninth Bit of SDR Master Written Data as Parity:** In I²C, the ninth Data bit written by the
736   Master is an ACK by the Slave. By contrast, in I3C the ninth Data bit written by the Master is the
737   Parity of the preceding eight Data bits. Therefore, in I3C the Slave shall not drive the SDA line for
738   Data written by the Master in SDR. In SDR terms, the ninth bit of Write data is referred to as the
739   T-Bit (for 'Transition') (see *Section 5.1.2.3.2*).

740   3. **Ninth Bit of SDR Slave Returned (Read) Data as End-of-Data:** In I²C, the ninth Data bit from
741   Slave to Master is an ACK by the Master. By contrast, in I3C this bit allows the Slave to end a
742   Read, and allows the Master to Abort a Read. In SDR terms, the ninth bit of Read data is referred
743   to as the T-Bit (for 'Transition') (see *Section 5.1.2.3.4*).

5.1.2.3.1   Transition from Address ACK to SDR Master Write Data

744   The end of any Address Header (whether Arbitrated or not) is an ACK or NACK by the one or more addressed
745   Slaves, using Open Drain on SDA:

746   • If 7'h7E, then it is the ACK of all I3C Slaves on the Bus.

747   • If a single Slave Address, then it is the ACK (or NACK) of the addressed Slave, or a NACK if no
748   such Slave is on the Bus.

749   When the Address Header results in an ACK, and the Message is SDR Write from Master, the SDA line has
750   to be turned from Open Drain to Push-Pull for the first data bit. To do that safely, I3C SDR specifies how the
751   handoff is to occur. This is summarized below and shown in *Figure 32*.

752   1. The I3C Slave shall hold the SDA line Low during the ACK (while SCL is Low).

753   • This is an Open Drain SCL Low period.

754   2. After the I3C Slave sees the rising edge of SCL, it releases the SDA line to High-Z.

755   • The I3C Slave shall release the SDA line using normal (Push-Pull) timing (release the SDA line
756   as soon as it sees SCL rising).

757   3. After the rising edge of SCL, the I3C Master shall drive the SDA line Low.

758   • As a result, both Master and Slave will be driving the SDA line Low for a short overlap (which
759   is safe).

760   • The SCL High period may be as short as the minimal t_{DIG_H}, per *Section 6.2*.

761   4. On the falling edge of SCL the I3C Master shall begin driving data on the SDA line, using Push-
762   Pull drive as shown in *Figure 32*.

763   When the Address Header results in a NACK, the Master may choose to either:

764   1. Continue the transaction, by generating a Repeated START

765   **or**

766   2. Relinquish the Bus, by generating a STOP as shown in *Figure 33*.

5.1.2.3.2   Transition from Address ACK to Mandatory Byte during IBI

767 The end of any IBI Address Header during an IBI request (whether the Address Header is Arbitrated or not)
768 is either an ACK or a NACK by the Master, using Open Drain on SDA.

769   If the Master detects one of the Slave Addresses, and if the Master chooses to receive the request,
770   then the Master will ACK the request per *Section 5.1.6.2*.

771 • **ACK:** When the IBI Slave Address Header results in an ACK, and the Slave Device is capable of
772 sending a Mandatory Byte, then the SDA line has to be turned from Open Drain to Push-Pull for
773 the first data bit. To do that safely, I3C SDR specifies how the handoff is to occur. This is
774 summarized below.

775   1. The I3C Master shall hold the SDA line Low during the ACK (i.e., while the SCL line is Low).
776   This is an Open Drain SCL Low period.

777   2. After the rising edge of the SCL line, the I3C Slave shall drive the SDA line Low as soon as
778   possible. As a result, both Master and Slave will be driving the SDA line Low for a short overlap
779   (which is safe).

780   3. After (A) a minimum of the t_{SCO} time reported by the Slave Device, or (B) a predetermined safe
781   time that could guarantee the takeover by all Slaves, the I3C Master shall release the SDA line.
782   The SCL High period may be as short as the minimal t_{DIG_H}, per *Section 6.2*.

783   4. On the falling edge of SCL, the I3C Slave shall begin driving data on the SDA line, using Push-
784   Pull drive.

785 • **NACK:** When the Address Header results in a NACK, the Master may choose to either:

786   • Continue the transaction, by generating a Repeated START, or

787   • Relinquish the Bus by generating a STOP.

5.1.2.3.3   Ninth Bit of SDR Master Written Data as Parity

788 The ninth data bit of each SDR Data Word written by the I3C Master (also referred to as the T-Bit) is a Parity
789 bit, calculated using odd parity. Parity can help in detecting noise-caused errors on the line. The value of this
790 Parity bit shall be the XOR of the 8 Data bits with 1, i.e.: XOR(Data [7:0], 1).

791 **Examples:**

792 • If all eight data bits are 1's (0xFF), then the Parity bit value will be 1.

793 • If all eight data bits are 0's (0x00), then the Parity bit value will be 1.

794 • If an odd number of bits in the Data Word are 1's (e.g. 0xFE or 0x01), then the Parity bit value will
795 be 0.

796 T (Parity) bit writes shall always be kept valid through the SCL High period. In the case of a T-Bit
797 representing the last data byte, the write is therefore kept valid through the SCL High period, and the next
798 SCL Low can then be used to either change the SDA, or not change the SDA, in preparation for the Repeated
799 START or STOP that follows.

5.1.2.3.4 Ninth Bit of SDR Slave Returned (Read) Data as End-of-Data

800 In I²C, Read from Slave has the issue that only the Master ends the Read, so the Slave has no ability to control
801 the amount of data it returns. In I3C SDR, by contrast, the Slave controls the number of data Words it returns;
802 but it also allows the I3C Master to abort the Read prematurely when necessary.

803 This mechanism is controlled purely by the ninth (T) Data bit of each SDR Data Word returned by the I3C
804 Slave. The ninth bit is returned by the Slave in one of three ways, as explained below.

805 • The I3C Slave returns the ninth bit as 0 (SDA Low) to end the Message:

806 • The Slave shall set SDA Low on the falling edge of SCL.

807 • On the following rising edge of SCL, the Slave shall set SDA to High-Z.

808 • The I3C Master shall drive SDA Low on the rising edge of SCL, thereby overlapping with the
809 Slave.

810 • The I3C Master then shall issue either a STOP as shown in ***Figure 39***, or a Repeated START as
811 shown in ***Figure 40*** (on the next clock, or one after, per the normal I²C procedure for setting
812 up SDA to issue a Repeated START).

813 • The I3C Slave returns the ninth bit as 1 (SDA High) to continue the Message (and permit the
814 Master to abort the Message):

815 • The Slave shall set SDA High on the falling edge of SCL.

816 • On the following rising edge of SCL, the Slave shall set SDA to High-Z, thereby Parking the
817 Bus for the SCL High period:

818 • If the I3C Master is able to continue the reply from the Slave, then it shall do nothing. The
819 weak Pull-Up resistor on SDA will keep SDA High during the SCL High period, as shown
820 in ***Figure 41***.

821 • If the I3C Master wants to abort the Message, then it shall drive SDA Low after the rising
822 edge of SCL, thereby terminating the Message with a Repeated START. The I3C Master
823 then takes control starting with the falling edge of SCL. The Master shall ensure enough
824 delay after SCL rising before driving SDA Low to ensure no contention. In order to
825 achieve this delay, the Master might have to extend the SCL High period.
826 Since the transition of SDA Low when SCL is High is a Repeated START, the Master
827 may begin a new Address (see ***Figure 43***), or it may issue a STOP in the next cycle (see
828 ***Figure 42***). However in a Mixed Bus the Master should extend the SCL Low period, in
829 order to ensure that any Spike Filters for Legacy I²C Devices properly reset.

830 • The Slave shall monitor the SDA on the falling edge of SCL:

831 • If SDA is High, then the Slave shall continue with the next data value.

832 • If SDA is Low (i.e., if there has been a Repeated START), then the Message has been
833 aborted, and the Slave shall not drive SDA after that.

5.1.2.4   Use of Clock Speed to Prevent Legacy I^{2}C Devices from Seeing I3C Traffic

834   In a system, there are three possible I3C Bus Configurations:

835   1. **Pure Bus:** Only I3C Devices are present on the Bus.

836   2. **Mixed Fast Bus:** Both I3C Devices and Legacy I^{2}C Devices are present on the Bus, such that the
837   Legacy I^{2}C Devices are restricted to ones that are generally permissible (i.e., Slave-only, and no
838   Slave clock stretching), and that have a true I^{2}C 50 ns Spike Filter on SCL. (I.e., I^{2}C Devices that
839   do not "see" the SCL line as High when the High duration is less than 50 ns, across all
840   temperatures and processes.)

841   3. **Mixed Slow/Limited Bus:** Both I3C Devices and Legacy I^{2}C Devices are present on the Bus,
842   such that the Legacy I^{2}C Devices are restricted to ones that are generally permissible (i.e., Slave-
843   only, and no Slave clock stretching), but that do not have a true I^{2}C 50 ns Spike Filter on SCL.

844   The Bus designer's choice of Bus Configuration affects what speed options are available for I3C SDR, as
845   well as what HDR Modes are possible at various clock speeds. *Table 10* shows the possible options for each
846   Bus Configuration.

847   **Table 10 Available Options for Bus Operating Parameters, Per I3C Bus Configuration**

Bus Operating Parameter	Available Options Per Bus Configuration		
	Pure Bus	**Mixed Fast Bus**	**Mixed Slow and Limited Bus**
SDR Mode Speed	f_{SCL} (Min) to f_{SCL} (Max)	I^{2}C messages: Fm or Fm+, I3C messages: SCL High from $t_{DIG\_H\_MIXED}$ (Min) to $t_{DIG\_H\_MIXED}$ (Max) with SCL low up to t_{DIG_L} (Max) (see *Section 5.1.2.4.1*)	Fm or Fm+ only[1]
Note:			
1) *May be faster if all Legacy I^{2}C Devices are index 1, per* **Table 5**			

5.1.2.4.1   Use of Duty Cycle to Achieve Lower Effective Speed in a Mixed Fast Bus

848   A pure I3C Bus may simply change the clock speed to any frequency in the allowed speed range, as outlined
849   in *Section 6*. By contrast, a Mixed Fast Bus wanting to clock faster than the slowest Legacy I²C Device needs
850   to take advantage of the Spike Filter, i.e, shall ensure that the SCL High period is shorter than the Spike Filter,
851   in order to prevent the Legacy I²C Devices from seeing the SCL High period (see $t_{DIG\_H\_MIXED}$ in *Table 59*).
852   As a result, the Legacy I²C Device 'sees' the SCL as staying Low the whole period.

853   However, a Mixed Fast Bus I3C Master can change the effective Bus frequency by varying the duty-cycle of
854   SCL. This allows running the data rate slower, for example to accommodate Slaves which need a lower rate
855   as defined by GETMXDS CCC (see *Section 5.1.9.3.18*). It may also be necessary to accommodate Legacy
856   I²C Devices with Spike Filters that need a longer Low period in order to function properly.

857   In this model the SCL High period shall never exceed $t_{DIG\_H\_MIXED}$, thus staying below the 50ns required by
858   the I²C Spike Filter; however, the Low period is free to be any length permitted by I3C's allowed clock
859   frequency range. For example, depending on the clock generation capability of the I3C Master, the SCL Low
860   period may be a multiple of the High period, or it may be any multiple of some higher frequency clock
861   capable of providing a High period less than or equal to $t_{DIG\_H\_MIXED}$, but greater than the minimum SCL High
862   period as defined in *Table 57*.

863   **Example 1:** An SCL High period of 40ns, plus a Low period of 280ns, yields a total clock period of 320ns
864   which corresponds to a frequency of 3.125MHz.

865   **Example 2:** In order to ensure that the Legacy I²C Device Spike Filters continue tracking SCL as Low, an
866   SCL High period of 40ns could be used with a Low period of 80ns, yielding a total clock period of 120ns
867   which corresponds to a frequency of 8.3MHz.

868   Such adjustment of the clock Duty Cycle brings three benefits:

869   • It remains hidden from Legacy I2C Devices, while still significantly increasing the SDR data rate.

870   • It ensures a longer Low period, so that Legacy I²C Device Spike Filters continue to track SCL as
871   Low.

872   • It is easy for a Master to generate, and has no impact on I3C Slaves (which just react to clock
873   edges).

874   This model works because all I3C Slaves must be able to accommodate 12.5MHz as a clock rate, so the
875   shorter High period will not impact them.

876   Note that this Duty Cycle technique does not resolve issues of long Buses with high line capacitance. This is
877   because the short High period may be insufficient for SDA propagation, even if the Low period is long
878   enough.

5.1.2.5 Master Clock Stalling

879 In SDR Mode the I3C Master may Stall the I3C Bus during the SCL Low period, but only under the specific,
880 transitory conditions described in this Section.

881 Stalling may be necessary for either of two reasons:

882 1. The absolute or relative timing of a Message to a specific Slave, or to all Slaves, needs to be
883 carefully controlled. Clock Stalling provides the Master with fine-grained data timing control.

884 2. The I3C Master needs to internally synchronize data. This may be due to parts of the Master's
885 system waking up in response to data, to changing state, or to otherwise needing time during a
886 transaction.

887 Note that Stalling impacts Bus performance. For example, it will reduce the Bus capacity and increase the
888 latency of any Devices issuing In-Band Interrupts.

889 To Stall the Bus, the I3C Master shall hold SCL Low while the Bus is in one of the four following conditions,
890 each of which is detailed below:

891 1. I3C/I²C Transfer, ACK/NACK Phase

892 2. Write Data Transfer, Parity Bit

893 3. I3C Read Transfer, Transition Bit

894 4. Dynamic Address Assignment, First Bit of Assigned Address

895 The maximum Stall time (SCL Low period) shall be 15 ms; note, this is an absolute maximum. The Master
896 should use the shortest Stall duration possible given current circumstances, as per *Table 11*.

897 **Table 11 Master Clock Stall Times**

Condition	Section	Maximum Stall Time
I3C/I²C Transfer, ACK/NACK Phase	*5.1.2.5.1*	100 µs
Write Data Transfer, Parity Bit	*5.1.2.5.2*	100 µs
I3C Read Transfer, Transition Bit	*5.1.2.5.3*	100 µs
Dynamic Address Assignment, First Bit of Assigned Address	*5.1.2.5.4*	15 ms

898 In all cases, after Stalling the SCL Low period, the continuation of the clock (i.e., the first SCL rising edge
899 after the extended Low period) shall follow the normal rules for I3C SDR (for I3C SDR Messages), or for
900 I²C (for I²C Messages) including rise time, change in SDA (if any) before SCL rise, next bit, etc. For example,
901 the I3C Master might choose to terminate the Frame after this Stalled bit with a STOP; but in this case the
902 Master is required to observe the requirements for STOP timing, as appropriate.

903 It is recommended to use Master Clock Stalling only when necessary and unavoidable. In all other
904 circumstances the Master should avoid Master Clock Stalling because of its negative impacts on Bus
905 performance. In order to help guide system designers, Data Sheets for I3C Master Devices should include
906 appropriately detailed SCL Stalling parameters.

5.1.2.5.1   I3C/I²C Transfer, ACK/NACK Phase

Master Clock Stalling during the ACK/NACK phase of the I3C/I²C transfer (see *Figure 11*) indicates one of the following:

- The Master can allow the I3C/I²C Slave to prepare to receive data (for write transfers) or transmit data (for read transfers)
- The Master can Stall SCL during write or read transfers to Legacy I²C Slaves in case of underrun and overflow situations
- The Master can Stall the clock during the ACK/NACK phase of the incoming In-Band Interrupt (Slave Interrupt Request or Mastership Request), to decide whether to ACK or NACK depending upon the incoming In-Band Interrupt
- The Master can allow the Slave to respond with data for the directed GET CCC commands if the Slave is not ready with the CCC data to be returned
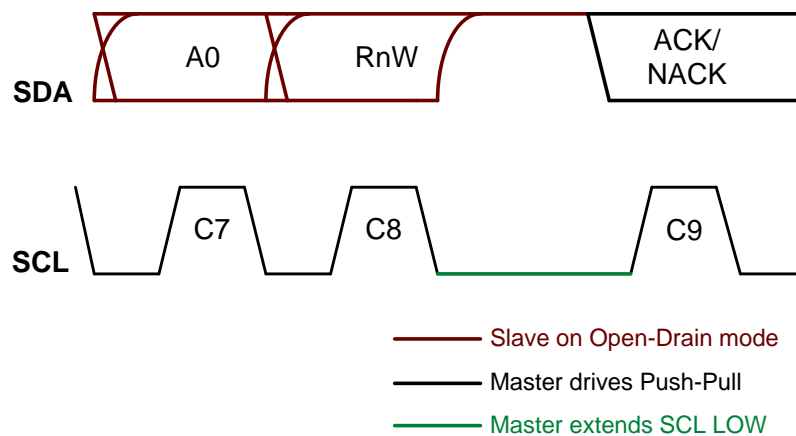
Figure 11 Master Clock Stalling in ACK Phase

5.1.2.5.2   Write Data Transfer, Parity Bit

920   The Master can Stall SCL between data bytes of an I3C Write Data transfer, by Stalling the clock during the
921   Low period of the Parity Bit, in case of underrun situations. See *Figure 12*.

922
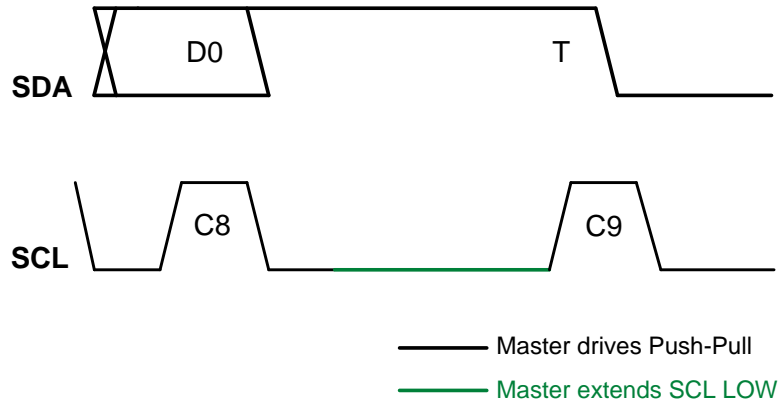
923   **Figure 12 Master Clock Stalling in Write Parity Bit**

5.1.2.5.3   I3C Read Transfer, Transition Bit

924   The Master can Stall SCL between data bytes of an I3C Read Data transfer, or before terminating, by stalling
925   the clock during the Low period of the Transition Bit, in case of overflow situations. See *Figure 13* through
926   *Figure 17*.

927
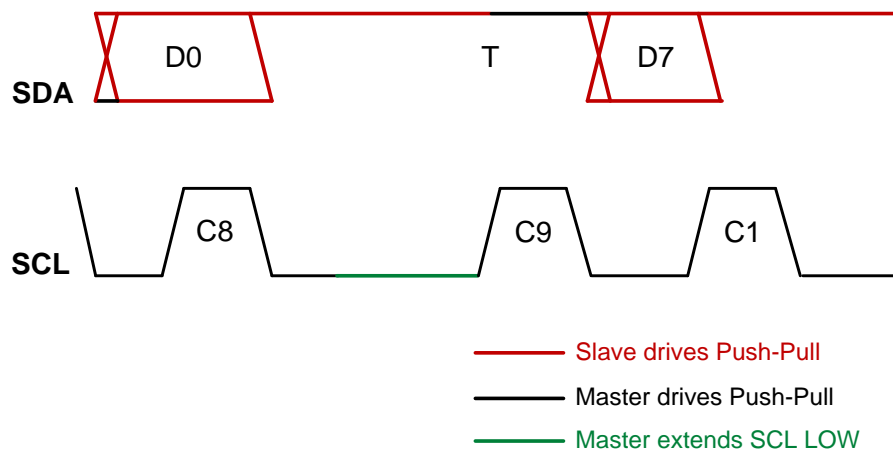
928   **Figure 13 Master Clock Stalling in T-Bit Before Next Read Data**
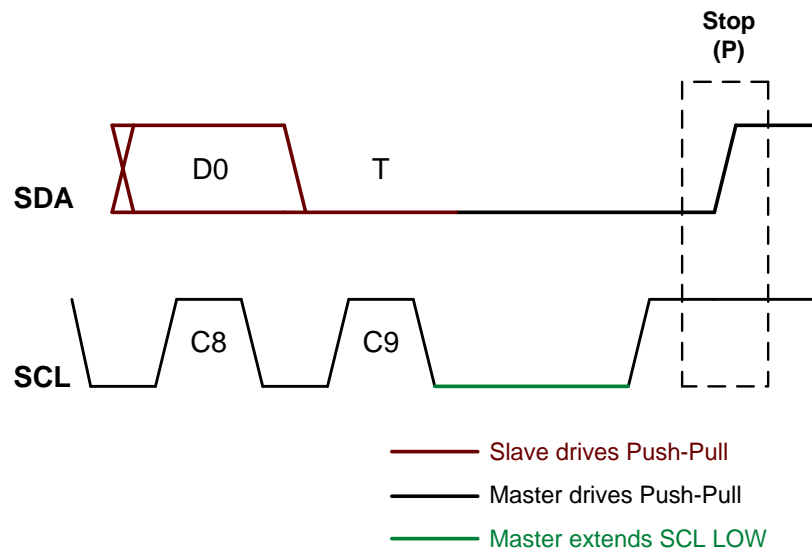
Figure 14 Master Clock Stalling in T-Bit Before STOP
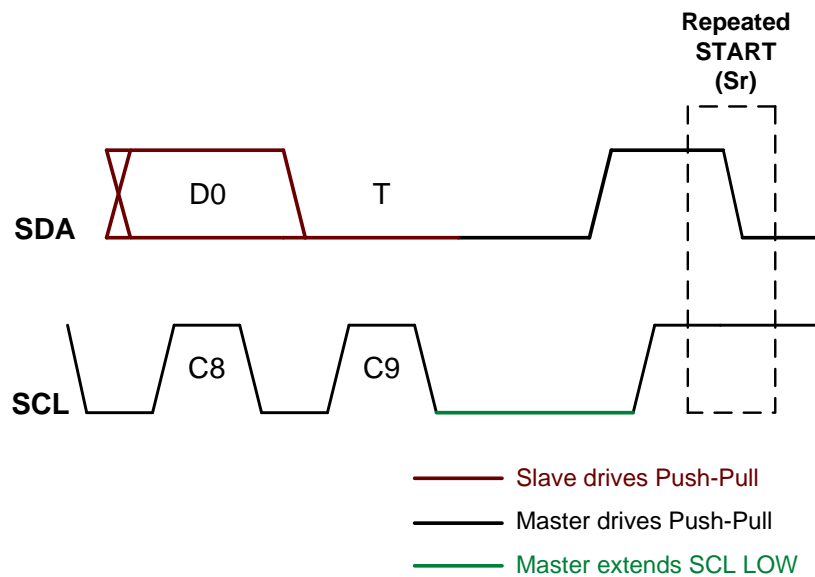
Figure 15 Master Clock Stalling in Low T-Bit Before Repeated Start
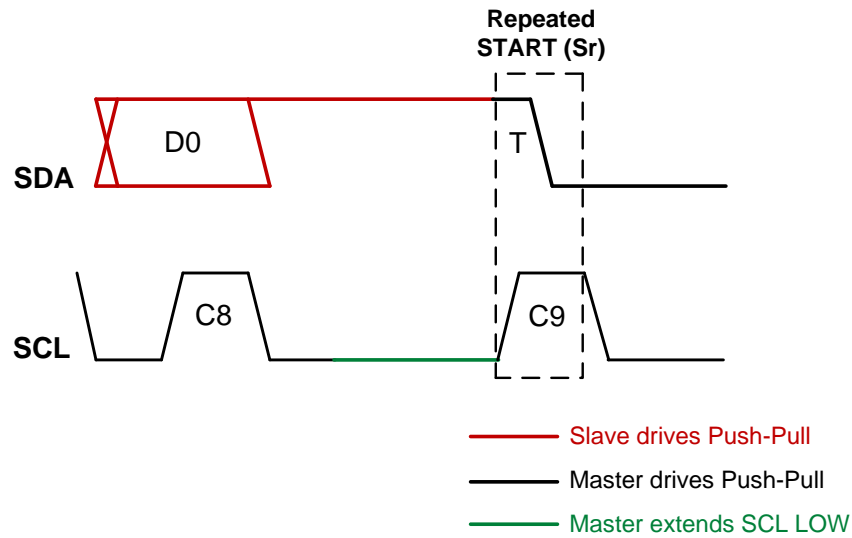
Figure 16 Master Clock Stalling in High T-Bit Before Repeated START
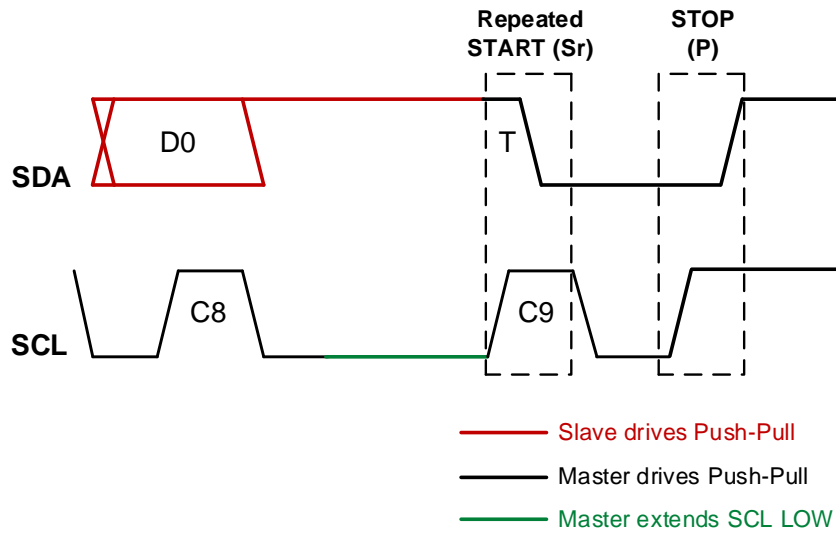
Figure 17 Master Clock Stalling in High T-Bit Before Repeated START and STOP

5.1.2.5.4 Dynamic Address Assignment, First Bit of Assigned Address

937 The Master can stall SCL during the Low period of the first bit of the Assigned Address phase of the Enter
938 Dynamic Address Assignment CCC command (ENTDAA, see *Section 5.1.9.3.4*), for example to gain time
939 to assign the Dynamic Address to the Device based on the Slave's Bus Characteristics Register BCR (see
940 *Section 5.1.1.2.1*) and Device Characteristics Register DCR (see *Section 5.1.1.2.2*) bytes. See *Figure 18*.

941
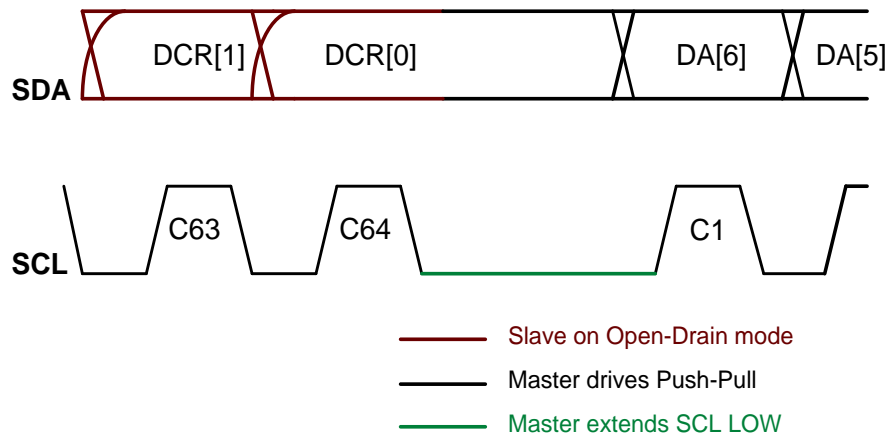
942 **Figure 18 Master Clock Stalling in Dynamic Address First Bit**

5.1.3 Bus Conditions

943 This Specification defines Open Drain Pull-Up and High-Keeper, as well as three distinct conditions in which
944 the I3C Bus shall be considered inactive: Bus Free, Bus Available, and Bus Idle.

5.1.3.1 Open Drain Pull-Up and High-Keeper

945 I3C Master Devices shall provide an active Open Drain class Pull-Up, to be engaged whenever the Bus is in
946 Open Drain mode (with exceptions, detailed below, where a weak Pull-Up may be used).

947 This active Pull-Up shall be implemented either:

948 1. As a passive resistance from V_{DD}, or

949 2. As a passive resistance from a current source, or

950 3. In any other method that both:

951   a. Balances its current sourcing in order to ensure that SDA rises within t_{rDA} (see **Table 57**), and

952   b. Is not so strong as to prevent a Slave with the minimum I_{OL} driver (see **Table 54**) from driving
953   SDA Low within t_{rDA} (see **Table 58**).

954 In addition to the active Open Drain class Pull-Up, a High-Keeper is also required on the Bus. A High-Keeper
955 is generally used for a Master-to-Slave or Slave-to-Master handoff, and for optional termination uses where
956 the Master can signal a termination by driving SDA Low while parked High.

957 The High-Keeper on the Bus shall be strong enough to prevent system leakage (i.e., the sum of the leakage
958 of all Devices on the Bus) from pulling SDA, and sometimes SCL, Low. The High-Keeper on the Bus shall
959 also be weak enough that a Slave with the minimum I_{OL} driver (see **Table 54**) is able to pull SDA, SCL, or
960 both Low within the Minimum t_{DIG_L} period.

961 The Master may provide the High-Keeper on the Bus. If the Master does so, then the Master shall also provide
962 a way to disable its High-Keeper. Reasons to disable a Master-provided High-Keeper might include that the
963 High-Keeper is not strong enough for the present system leakage, or other reasons.

964 A Master-provided High-Keeper may optionally be implemented using a single, common Pull-Up Device
965 capable of supporting both the active Open Drain class Pull-Up function and the weak High-Keeper class
966 Pull-Up function.

967 Whether implemented as a single combined Pull-Up, or as two separate Pull-Ups, the Master should switch
968 SCL and SDA, each independently, between three Pull-Up states as needed, based on Bus state:

969 1. No Pull-Up (High-Z)

970 2. High-Keeper Pull-Up

971 3. Open Drain Pull-Up

972 The Bus shall have High-Keepers on SDA and SCL. When adequate High-Keepers cannot be provided by
973 the Master, then the system designer is required to handle this externally. Reasons why the Master would be
974 unable to provide adequate High-Keepers might include that the Master does not support High-Keepers, that
975 the Master-provided High-Keepers are not strong enough for present needs, or that the Bus is too long to use
976 the Master-provided High-Keepers.

977 The system High-Keepers on SCL and SDA may be implemented as one or more passive resistors tied to
978 V_{DD}, or they may be active Bus-Keeper Devices that turn off when the respective line is pulled below some
979 threshold. The system High-Keepers on SCL and SDA shall be sized so as to balance between system leakage
980 and the requirement that I3C Devices be able to pull the corresponding line Low within the t_{DIG_L} period.

981 ***Note:***

982   *The Master may choose to not use the Open Drain class Pull-Up in cases of Open Drain mode where*
983   *the SDA happens to be already High (i.e., is coming into the SCL Falling edge). In that case, the*
984   *Master may choose to rely solely upon the High-Keeper, in order to use less power if and when a*
985   *Slave drives SDA Low.*

5.1.3.2   Bus Free Condition

986 The Bus Free Condition is defined as a period occurring after a STOP and before a START, and with the
987 following duration:

988   • **For Pure Bus:** A duration of at least t_{CAS} (see *Table 58*)
989   • **For Mixed Bus** (i.e., at least one Legacy I^{2}C Device is present on the I3C Bus)**:** A duration of at
990   least t_{BUF} (see *Table 57*)

5.1.3.3   Bus Available Condition

991 The Bus Available Condition is defined as a period during which the Bus Free Condition is sustained
992 continuously for a duration of at least **t_{AVAL}** (see *Table 58*). A Slave may only issue a START Request (e.g.,
993 for an In-Band Interrupt, or for a Master Handoff Request) after a Bus Available Condition.

5.1.3.4   Bus Idle Condition

994 The I3C Bus Idle Condition is defined in order to help ensure Bus stability during Hot-Join events. The Bus
995 Idle Condition is defined as a period during which the Bus Available Condition (see *Section 5.1.3.3*) is
996 sustained continuously for a duration of at least t_{IDLE} (see *Table 58*).

997 ***Note:***

998   *If a Hot-Join Device is powered up onto the I3C Bus at the same time as the Main Master, then the*
999   *Hot-Join Device may pull SDA Low after 1 ms if (1) the Main Master has SCL and SDA pulled up,*
1000   *and (2) the Master does not act on the I3C Bus within the same Idle period.*

5.1.3.5   Activity States

1001 I3C provides a mechanism for the Master to inform Slaves about expected upcoming levels of activity on the
1002 I3C Bus, in order to help the Slaves better manage their internal states. Four Activity State levels from 0
1003 through 3 are defined (see *Table 12*).

1004   **Table 12 Activity States**

Activity State	Activity Interval	CCC
0	1 μ second	ENTAS0
1	100 μ second	ENTAS1
2	2 millisecond	ENTAS2
3	50 millisecond	ENTAS3

1005 The Activity State number serves as a hint to the Slave, indicating how long it will be before the Master
1006 directs Bus activity to that Slave, and the likely latencies to expect in response to the Slave pulling SDA Low
1007 (i.e., for the START Request to then generate an In-Band Interrupt Request or a Master Handoff Request).

1008 The Master uses CCC commands ENTAS0, ENTAS1, ENTAS2, and ENTAS3 (see *Section 5.1.9.3.2*) to
1009 communicate the four expected Bus Activity states to Slaves. Each ENTASx CCC has both a Broadcast
1010 version and a Directed (per-Slave) version. The Master may switch to a different Activity State in any way,
1011 and at any time, by issuing the appropriate ENTSAx CCC command.

1012 The Slave may use the received Activity State hint to adjust internal settings such as power savings, FIFO
1013 trigger levels, timestamp counters and clock rates, and other suitable operating parameters. However Slaves
1014 are not required to support the Activity States CCCs (ENTASx), and as a result could even ignore them
1015 completely.

1016 The Activity States mechanism is a basis for a general agreement between Master and Slave, i.e., that the
1017 Slave may NACK any access occurring sooner than the general time factor. For example: If the Master sends
1018 the ENTAS2 CCC, meaning the Master is unlikely to initiate a request sooner than 2 ms from the time the
1019 CCC is sent, then a request arriving only 1 ms later could result in a NACK (although it will be ACKed if the

Copyright © 2016-2018 MIPI Alliance, Inc.
                              Confidential

1020 request is repeated 50us later, i.e., after the Slave re-awakens). As a result, the Slave shall wake up either
1021 upon any Bus activity, or upon matching 7'h7E and its own Dynamic Address.

1022 The Activity State CCCs also adjust the maximum value for I3C Bus timing parameter t_{CAS} (Clock after
1023 START; see *Table 58*), the maximum amount of time that the Master may take to generate the SCL clock
1024 (drive SCL Low) in response to the Slave pulling SDA Low. Note that t_{CAS} is only a worst-case number; it
1025 does not indicate whether or not the Master will ACK the In-Band Interrupt Request or Master Request.
1026 Further, the selected Activity State does not necessarily indicate the time at which the Master will read
1027 additional data from a Slave in response to an In-Band Interrupt; that is a private contract between Master
1028 and Slave. A Slave that does not support the ENTASn CCCs shall have a t_{CAS} maximum value of 50
1029 milliseconds (the ENTAS3 value).

1030 Activity States are not intended as a substitute for a more precise power mode, nor for any other mechanism
1031 that might be supported by private contract between Master and Slave. For example, if a Slave has a device
1032 power mode setting, then the Master should use that mechanism to put the Slave into the desired state.
1033 Likewise, a Slave could provide a FIFO trigger level setting, relating to the amount of time that the Master
1034 will have to read the FIFO contents in response to an In-Band Interrupt Request; if so, then the Master should
1035 use that setting to match its internal latencies.

5.1.4 Bus Initialization and Dynamic Address Assignment Mode

1036 As detailed in this Section, the Main Master is responsible for performing a Dynamic Address Assignment
1037 procedure, in order to provide a unique Dynamic Address to each Device connected to the I3C Bus.

1038 The Main Master shall provide a Dynamic Address to a Device:

1039 1. Upon any initialization of the I3C Bus, and

1040 2. When the Device is connected to an already configured I3C Bus.

1041 Once a Device receives a Dynamic Address, that Dynamic Address shall be used in all of that Device's
1042 subsequent transactions on the I3C Bus, until and unless the Master changes the Device's Dynamic Address.
1043 The only way for the Master to change the Device's Dynamic Address is by using either the RSTDA CCC
1044 command (see *Section 5.1.9.3.3*) or the SETNEWDA CCC command (see *Section 5.1.9.3.11*). The Master
1045 might choose to change the Device's Dynamic Address due to re-prioritization.

1046 The Main Master controls the Dynamic Address Assignment process. This process includes an Address
1047 Arbitration procedure similar to I²C's (see *[NXP01]* at *Sections 3.1* and *3.1.8*). The I3C Arbitration procedure
1048 differs from I²C by using the values of the 48-bit Provisional ID and the Device's I3C Characteristic Registers
1049 (that is, BCR and DCR), concatenated. The Device on the I3C Bus with the lowest concatenated value wins
1050 each Arbitration round in turn, and the Main Master assigns a unique Dynamic Address to each winning
1051 Device.

5.1.4.1   Device Requirements for Dynamic Address Assignment

5.1.4.1.1   Unique Identifiability

1052 In order to support the Dynamic Address Assignment procedure, each I3C Device to be connected to an I3C
1053 Bus shall be uniquely identifiable in one of two ways, before starting the procedure.

1054 **Either:**

1055 1.   The Device may have a Static Address, in which case the Master may use that Static Address
1056   (presuming it is known to the Master).

1057   For example, an Address similar to what I²C specifies *[NXP01]*.

1058 **Or:**

1059 2.   The Device shall in all cases have a 48-bit Provisional ID. The Master shall use this 48-bit
1060   Provisional ID, unless the Device has a Static Address and the Master uses the Static Address.

1061   The 48-bit Provisional ID is composed of three parts:

1062   1.   **Bits [47:33]**: MIPI Manufacturer ID *[MIPI01]* (15 bits)

1063   *Note:* The Most Significant Bit of the MIPI Manufacturer ID is discarded, i.e.
1064   only the 15 Least Significant Bits are used.

1065   2.   **Bit [32]:** Provisional ID Type Selector (One bit, 1'b1: Random Value, 1'b0: Vendor Fixed
1066   Value)

1067   3.   **Bits [31:0]:** 32 bits containing either a Vendor Fixed Value or a Random Value, depending on
1068   the value of Bit [32]:

1069   **If the value of Bit [32] is 1'b0: Vendor Fixed Value:**

1070   • **Bits [31:16]: Part ID:** The meaning of this 16-bit field is left to the Device vendor to
1071   define.

1072   • **Bits [15:12]: Instance ID:** The value in this 4-bit field should identify the individual
1073   Device, using a method selected by the system designer. For example: straps, fuses, non-
1074   volatile memory, or another appropriate method.

1075   • **Bits [11:0]:** The meaning of this 12-bit field is left for definition with additional meaning.
1076   For example: deeper Device Characteristics, which could optionally include Device
1077   Characteristic Register values.

1078   **If the value of Bit [32] is 1'b1: Random Value:**

1079   • Bits [31:0]: 32-bit value randomly generated by the Device. This value can be queried via
1080   General Test Mode, using the Command Code **Enter Test Mode (ENTTM)** (see
1081   *Section 5.1.9.3.8*).

1082 *Note:*

1083   *Under Vendor Test Mode, the Device may provide a fully random or pseudo-random 48-bit*
1084   *Provisional ID (see **Section 5.1.9.3.8**).*

5.1.4.2   Bus Initialization Sequence with Dynamic Address Assignment

1085 Bus Initialization and Dynamic Address Assignment shall be executed according to the following sequence.
1086 See also *Figure 59 (Dynamic Address Assignment FSM)*.

1087 The Dynamic Address Assignment process shall be performed in Open Drain mode, except that the Repeated
1088 START and 7'h7E/R may be either Open Drain or Push-Pull. For Open Drain the Main Master shall drive
1089 the SCL line with clocks at the appropriate Open Drain speed for the Devices present on the I3C Bus. For
1090 Repeated START, the Main Master may optionally choose to actively drive the SDA line High, but only after
1091 the Slave has safely released the SDA line.

1. The Main Master shall begin in an appropriately configured state, and shall have, either in its own non-volatile memory or as a result of having received it from the Application Host, the following data:

   a. The number of I3C compliant Devices that need to receive a Dynamic Address,

   b. The data for any I3C Devices resident on the I3C Bus that already have Static Addresses, and

   c. The data for any Legacy I²C Devices resident on the I3C Bus.

2. The Main Master shall assign Dynamic Addresses to any I3C Devices with a known Static Address, using the Command Code **Set Dynamic Address from Static Address** (**SETDASA**) (see *Section 5.1.9.3.10*), or by assigning all I3C Devices their known I²C Static Address using the Command Code **Set All Addresses to Static Address** (**SETAASA**) (see *Section 5.1.9.3.22*)

   Under these pre-conditions, Slaves supporting the SETAASA CCC and Slaves only supporting the SETDASA CCC (see *Section 5.1.9.3.10*) can be mixed on the I3C Bus. Via SETDASA, the Master will individually assign a Dynamic Address to each Slave not supporting SETAASA.

   In a system that mixes I²C-capable Devices and non-I²C-capable Devices, the Master shall send the SETAASA CCC before sending the ENTDAA CCC (see *Section 5.1.9.3.4*), in order to assign Dynamic Addresses to the I3C-only Slaves. In addition, any Slave having a restricted Address (see *Table 9*) shall not support SETAASA.

3. The Main Master shall send the Broadcast Command Code **Enter Dynamic Address Assignment** (**ENTDAA**) (see *Section 5.1.9.3.4*). There is no data associated with this Command Code.

4. The Main Master shall send a Repeated START, and the I3C Broadcast Address 7'h7E with RnW bit High (i.e. Read). Every I3C Device on the I3C Bus that does not yet have an assigned Dynamic Address, and that is not a Hot-Join Device, shall acknowledge the I3C Broadcast Address. (I.e., Hot-Join Devices that do not yet have an assigned Dynamic Address shall not acknowledge the I3C Broadcast Address in this step.)

   At least one I3C Device on the I3C Bus will acknowledge the I3C Broadcast address in this step.

   Note:

   *This use of the I3C Broadcast Address (7'h7E) with RnW bit High (i.e. Read) is specific to Dynamic Address Assignment Mode. It is also acceptable per the I²C Specification **[NXP01]**.*

5. The Main Master shall drive only the SCL line. The Main Master shall release the SDA line to a High-Z state, allowing SDA to go to High level via the Bus Pull-Up resistor.

6. Every I3C Device that responds to the I3C Broadcast Address sent in Step 4 shall drive the SDA line with its own 48-bit Provisional ID (using Big Endian bit order), until it loses Dynamic Address Arbitration.

   a. The 48-bit Provisional ID shall be transferred continuously, starting with the most significant bit (bit [47]), with no delimitation or ACK/NACK pulse.

   b. The Hot-Join Devices shall participate in the Dynamic Address Assignment procedure only after requesting it via an In-Band Interrupt using the Reserved Slave Address of 7'b0000_010 and RnW bit Low (i.e. Write).

7. The Main Master shall continue to drive the SCL line with the same clock, while still releasing the SDA line. The I3C Device that did not yet lose the Arbitration shall then transfer its Bus Characteristics Register (BCR) and Device Characteristic Register(s) (DCR) per *Section 5.1.1.2.2*, until it eventually loses Dynamic Address Arbitration. See also *Section 5.1.4.3*.

8. The Device whose concatenated Provisional ID, BCR, and DCR has the lowest value will win the Arbitration round, due to the nature of Arbitration.

Note:

*It is possible for multiple Devices to have the same concatenated value, although this is highly improbable. See **Section 5.1.4.3**.*

1139   9.   The Main Master shall transfer a 7-bit wide Dynamic Address for the winning Device, in Open Drain
1140   Mode. This Dynamic Address shall incorporate the priority level that the Main Master assigns to the
1141   Device, per ***Section 5.1.6.2***. The Arbitration-winning Device shall acknowledge the assigned
1142   Dynamic Address. This Dynamic Address transfer procedure shall have the following steps:

1143   • The Main Master shall drive the 7-bit Dynamic Address, followed by a parity bit (PAR), which
1144   is calculated as odd parity. Odd parity is the inverse of the XOR of the 7 bits. Therefore
1145   ~XOR(dynamic_address [7:1]) is placed in position 0.

1146   • If the parity is valid, then the Slave shall acknowledge receipt of the Dynamic Address on the
1147   next SCL clock. If the parity is invalid, then the Slave shall passively NACK on the next SCL.

1148
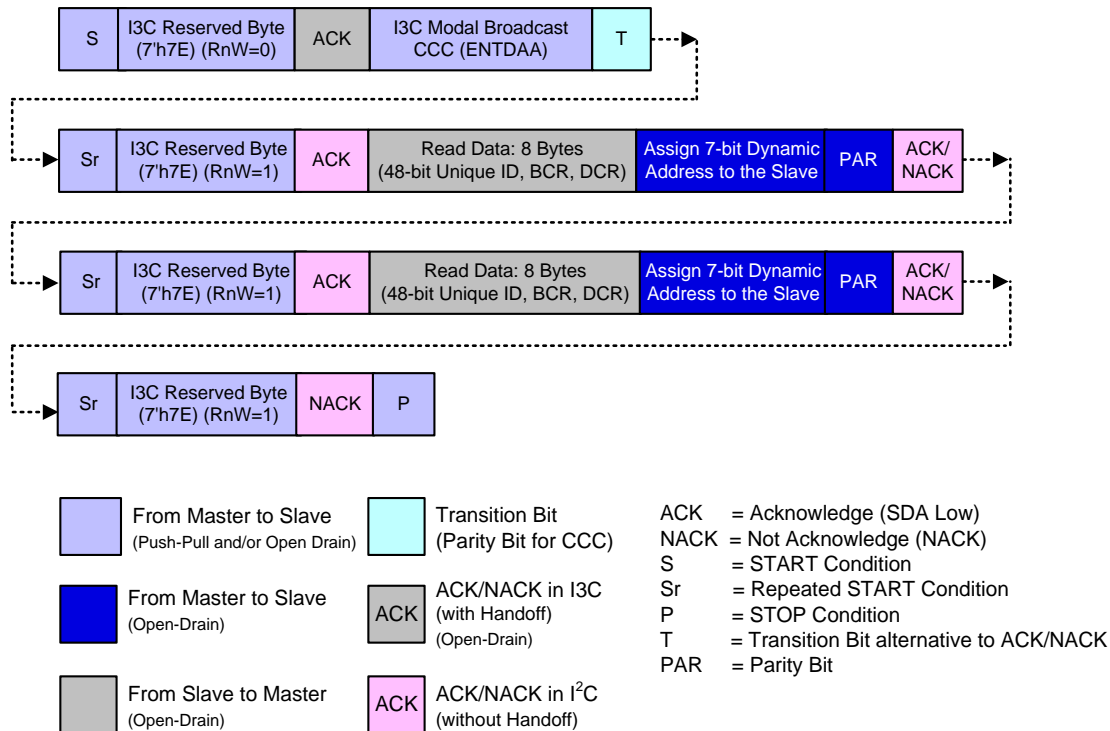
1149   **Figure 19 Dynamic Address Assignment Transaction**

1150   10. The Main Master shall repeat this procedure, jumping back to step 4 until there is no ACK from
1151   any Device present on the I3C Bus.

1152   11. This Dynamic Address Assignment procedure shall be ended by the Main Master issuing a STOP.

1153   Regarding this Dynamic Address Assignment procedure, note that:

1154   • The Main Master is able to end the Dynamic Address Assignment procedure at any time, even if
1155   some of the pre-established I3C Devices have not yet received their Dynamic Addresses.

1156   • The Dynamic Address Assignment procedure can be started again any time the I3C Bus is in Bus
1157   Available Condition, using the Command Code **Enter Dynamic Address Assignment (ENTDAA)**
1158   (see ***Section 5.1.9.3.4***).

1159   • If a given Slave does not acknowledge its assigned Dynamic Address, then the procedure requires
1160   the Main Master to continue from step 4. The Slave will then participate in the Address Arbitration
1161   using the same 48-bit Provisional ID, and as a result the Slave will win the Arbitration round. If the
1162   Slave does not ACK the Dynamic Address a second time, then the Main Master shall exit the
1163   Dynamic Address Assignment procedure and execute an error management procedure provided by
1164   the I3C Bus designer.

1165 • Any Secondary Masters wishing to receive the Dynamic Addresses assigned to all Devices present
1166 on the I3C Bus can do one, or both, of the following:

1167 • Follow the Dynamic Address Allocation procedure. A Secondary Master that follows the same
1168 procedure will have received all the same data.

1169 • Use the Command Code **Define List of Slaves (DEFSLVS)** (see *Section 5.1.9.3.7*) to acquire
1170 (eventually selectively) the Addresses and Characteristics of all Devices on the I3C Bus from
1171 the Main Master.

1172 After the Dynamic Address Assignment process is complete, the Main Master knows which Device on the
1173 I3C Bus is the Secondary Master. The Main Master then addresses the Secondary Masters with the Command
1174 Code **Define List of Slaves (DEFSLVS)** (see *Section 5.1.9.3.7*) and transfers the data for any and all Legacy
1175 I²C Devices on the I3C Bus. In order to keep the I3C Bus under control, this could all be done within the
1176 same transaction (i.e. the Main Master continues the communication by issuing a Repeated START each
1177 time, with no intervening STOP).

5.1.4.3 Provisional ID Collision Detection and Correction

1178 In the Dynamic Address Assignment procedure described in *Section 5.1.4.2*, multiple I3C compliant Devices
1179 will receive the same Dynamic Address if they provide the Main Master with both identical 48-bit Provisional
1180 IDs, and identical Device Characteristic Register values. Although the probability of such a coincidence is
1181 quite small the potential does exist, and the I3C Bus will not operate properly under such conditions (at least
1182 the Instance IDs must be different, see *Section 5.1.4.1.1*). As a result the Main Master must test for this
1183 collision condition, and resolve it if necessary, before the I3C Devices present on the I3C Bus can all be
1184 safely used together.

1185 The I3C Main Master knows the number of Devices resident on the I3C Bus requiring Dynamic Address
1186 Assignment (per procedure step 1.a. in *Section 5.1.4.2*), which enables detection of collision errors. At
1187 completion of the Dynamic Address Assignment procedure the Main Master shall compare this expected
1188 number to the final count of Static Addresses and Dynamic Addresses that were actually assigned. If fewer
1189 Dynamic Addresses were assigned than expected, then multiple Devices must have received the same
1190 Dynamic Address, i.e. a collision has occurred.

1191 If this collision condition is detected, then the Main Master shall resolve it using either of the following
1192 methods:

1193 • The Main Master directs a Reset Dynamic Address request to the Dynamic Address most likely to
1194 have been duplicated, using the Command Code **Reset Dynamic Address Assignment**
1195 **(RSTDAA)** (see *Section 5.1.9.3.3*).

1196 • Alternatively, the Main Master resets the Dynamic Address Assignment process by using the
1197 Broadcast Command Code **Reset Dynamic Address Assignment (RSTDAA)** (per
1198 *Section 5.1.9.3.3*), and then proceeding from step 3 of the procedure in *Section 5.1.4.2* from step 8,
1199 until the sooner of either:

1200 a. The expected number of I3C Devices is discovered, or
1201 b. A set maximum number of attempts fail. If this limit is reached, then a system error message
1202 shall be sent to the Host. To avoid freezing the entire system, a limit of three (3) attempts is
1203 recommended.

5.1.5   Hot-Join Mechanism

1204 The I3C protocol supports a Hot-Join mechanism, to allow Slaves to join the I3C Bus after it is already
1205 configured.

1206 *Note:*

1207 *Hot-Join does not allow Slaves to join the I3C Bus before the I3C Bus has been configured.*

1208 Hot-Join may be used for:

1209 • I3C Devices mounted on the same board, but de-powered until needed. Such Devices shall not
1210 violate electrical limits for Slaves when depowered (or while transitioning) as defined in *Section 6*,
1211 including capacitive load.

1212 • I3C Devices mounted on a module/board that is physically inserted after the I3C Bus has already
1213 been configured. This specification does not attempt to address how that physical insertion is
1214 handled, however such insertion shall not disrupt the SCL and SDA lines, including respecting all
1215 electrical limits defined in *Section 6*.

1216 Hot-Joining Slaves may be any valid Slave type, including Secondary Master. After a Hot-Join, the Current
1217 Master shall use the **DEFSLVS** CCC as described in *Section 5.1.9.3.7*. To ensure that any Secondary Masters
1218 are aware of all of the available Slaves, the Master shall immediately notify the I3C Bus if it discovers that
1219 any previously joined Slaves are no longer present on the Bus (e.g. due either to non-response, or to
1220 mechanisms outside of the Bus).

1221 Since the Master is not inherently aware when new Slaves join the I3C Bus, the Hot-Join mechanism provides
1222 the following procedure for informing the Master about new Slaves:

1223 1. The Slave shall wait for at least t_{IDLE} of Bus Available Condition (see *Table 58*).

1224 2. After the Slave determines that the Bus is Idle it may either issue a START, or else wait for a
1225 START. (To issue a START, pull the SDA line Low and hold it Low until the SCL line goes Low.)

1226 *Note:*

1227 *The Hot-Join Device may power-up at the same time as the Main Master (e.g. may be connected to*
1228 *the I3C Bus when power is applied to the system). In that case, the Main Master might pull SDA Low*
1229 *even before the I3C Bus has been started, assuming that SCL and SDA are being pulled up. If a*
1230 *Main Master needs 1 ms or more in order to start acting on the I3C Bus, then that Main Master shall*
1231 *be able to accommodate the situation of SDA being held Low when it is ready to initialize the I3C*
1232 *Bus, or it may instead delay pulling SDA High, and/or pulling SCL High, until it is ready.*

1233 Hot-Join capable Devices should implement some method that allows the system designer to
1234 prevent the Device from attempting to Hot-Join when the Device is not being used as a Hot-Join
1235 Device. For example: use of NVMEM, or else a pin strap could be used when the Device is
1236 soldered onto the board and powered with the rest of the I3C Bus.

1237 3. The Slave issues the Reserved Slave Address of 7'b0000_010 as an IBI, using W (write) after the
1238 START. This requests a Dynamic Address Assignment process.

1239 4. The Current Master shall perform one of 3 actions:

1240 a. NACK the request. The Slave shall try again at the next START.

1241 b. ACK the request, but issue a Broadcast CCC to disable Hot-Join by setting the DISHJ bit in
1242 the Command Code **Enable/Disable Slave Events Command (ENEC/DISEC)** (see
1243 *Section 5.1.9.3.1*). If another Device attempts to Hot-Join before the Master is ready to assign
1244 Dynamic Addresses to the Slaves, then the Master might need to repeat this procedure.

1245 c. ACK the request and then issue a Broadcast Command Code **Enter Dynamic Address**
1246 **Assignment (ENTDAA)** (see *Section 5.1.9.3.4*) to start the Dynamic Address Assignment
1247 process. Since only Slaves with no Dynamic Address shall participate, this allows the newly
1248 joined Slave to receive its Dynamic Address.

1249 5. If an **Enter Dynamic Address Assignment (ENTDAA)** Command Code is issued as described in
1250 *Section 5.1.9.3.4*, then the Slave shall transmit its 48-bit Address as per the normal mechanism.

1251   Hot-Join Slaves required to receive the same assigned Dynamic Address every time they join can
1252   use a fixed-value ID, per *Section 5.1.4.1.1*.

1253 If a Slave drops off the I3C Bus due to being detached or depowered, then the Master might not be aware of
1254 this. The Master may determine this condition by repeated attempts to contact the Slave using a safe (shall
1255 respond) Command Code such as **Get Device Status (GETSTATUS)** (see *Section 5.1.9.3.15*). If the Master
1256 determines that the Slave is no longer part of the I3C Bus, then it shall either recycle that Slave's Dynamic
1257 Address, or else reserve that Slave's Dynamic Address in case that Slave later re-joins the Bus.

5.1.5.1   Failsafe Device Pads

1258 Hot-Join Devices shall be Failsafe, unless there is a protection method outside of the pad. Failsafe means that
1259 when the SCL and SDA pads are unpowered but wire connected to the I3C Bus, they must not draw extra
1260 leakage current from the Bus.

1261 The Device SCL and SDA pads shall be designed so as to avoid drawing current from the system SCL and
1262 SDA lines, both when they are being held High and when they are being held Low. Such leakage may be
1263 avoided by using special connectors, or other isolation methods (e.g., physical connection order in a plug).
1264 This excess current should be avoided whether due to diode effect, rail tie for ESD, or clamps. The leakage
1265 current variation between unpowered and powered shall not exceed the normal variation range of an active
1266 pad, I_i, as specified in *Section 6*, *Table 54*.

5.1.6   In-Band Interrupt

1267 This Section specifies I3C's priority-based In-Band Interrupt mechanism.

5.1.6.1   Priority Level

1268 In I3C, Priority Level controls the order in which Slaves' In-Band Interrupt requests and Master requests are
1269 processed. The Priority Level of each Slave in an I3C Bus instantiation is encoded in its Slave Address, with
1270 lower Addresses having higher Priority. That is, Slaves with lower value Addresses and higher Priority Levels
1271 have their In-Band Interrupts and Master requests processed sooner than Slaves with higher value Addresses
1272 and lower Priority Levels. This Priority Level ordering is a natural outcome of the I3C Address Arbitration
1273 behavior specified at *Section 5.1.2.2.1*, where Address bits with value 0 prevail bits with value 1.

1274 As a result, during each Dynamic Address Assignment operation (see *Section 5.1.4*) the Main Master should
1275 assign lower Addresses to Slaves for higher Priority In-Band Interrupt requests.

5.1.6.2   I3C Slave Interrupt Request

1276 In order to request an interrupt, an I3C Slave shall emit its Address into the arbitrated Address header
1277 following a START (but not following a Repeated START). If no START is forthcoming within the Bus
1278 Available Condition, then the I3C Slave may issue a START Condition by pulling the SDA line Low, and
1279 then wait for the Current Master to pull the SCL Low.

1280 If the Current Master sets the SCL line Low, thus completing the START Condition, and then provides clocks
1281 on the SCL line, then the Slave shall drive the SDA line with its own Address, followed by an RnW bit with
1282 value 1'b1. If more than one Slave has issued an IBI request after the same START Condition, then the
1283 Current Master shall process those IBIs in Priority Level order, per *Section 5.1.6.1*. The Slave(s) that lost the
1284 arbitration may issue another IBI request, but shall not do so until after the Bus Available Condition.

1285 At that point, the Current Master shall perform one of the following three actions:

1286 1. Accept the IBI by providing the ACK bit. The actions available to the Current Master depend upon
1287   the value of the Slave's BCR [2] bit (in the Slave's BCR register):

1288   a. If the I3C Slave's BCR [2] bit is set to 1, then, per *Section 5.1.1.2.1*, the Current Master shall
1289   read the Mandatory Data Byte that follows the accepted IBI request at any "read" clock speed
1290   allowable by the Slave. This operation is similar to a "read" from the Slave and all the related
1291   rules apply. Note that the Current Master cannot avoid receiving the Mandatory Data Byte,
1292   since it is transmitted in Push-Pull mode.

1293   After reading the Mandatory Data Byte, the Current Master may take any other valid I3C
1294   action. For example, the Current Master could issue a STOP, or issue a Repeated START,
1295   or it could continuing reading additional Data Bytes from the Slave (if, for example, a
1296   private contract between the two Devices has been established in advance).

1297   If the Current Master completes this transfer before all additional Data Bytes are read,
1298   then the Slave shall not repeat the as yet unserviced IBI request; the Current Master has
1299   assessed the request, and will service it at a later time. Depending upon the character of
1300   the data, the Master can either:

1301   i.   Attempt to read either the full data (e.g. for short data packets), or
1302   ii. Continue from the position at which the transfer was interrupted (e.g. for FIFO transfers),
1303   or
1304   iii. Abort the IBI service (e.g. for Timing Control information).

1305   In all cases the specific follow-up actions shall be established by private contract between
1306   the devices, bearing in mind the possibility for data to be corrupted in the meantime.

1307   One conceptual time diagram of this sequence is shown in *Figure 20* below:

Open Drain	Open Drain	Open Drain	Hand Off	Push-Pull	Drive High or Low, and then High-Z	Push-Pull
S	Slave_addr_as_IBI/R	Master_ACK	SCL High	Slave_byte	T	Sr

1308 **Figure 20 IBI Sequence with Mandatory Data Byte**

1309   b. If the I3C Slave's BCR [2] bit is set to 0, then the Current Master may take any other valid
1310   I3C action. For example, the Current Master could issue a STOP, or issue a Repeated START,
1311   or it could continue reading additional Data Bytes from the Slave at any allowable "read"
1312   clock speed (if, for example, a private contract between the two Devices has been established
1313   in advance).

1314 2. Refuse the IBI without disabling interrupts. To do this, the Current Master simply passively
1315   NACKs to deny the IBI. The Slave shall try again after the Bus Available Condition.

1316 3. Refuse the IBI and disable interrupts. To do this, the Current Master NACKs to deny the IBI, then
1317   sends a Repeated START, and finally sets the DISINT bit in the Command Code **Disable Slave**
1318   **Events Command (DISEC)** to the interrupting Slave as described in *Section 5.1.9.3.1*. (The
1319   Current Master can set the ENINT bit in the Command Code **Enable Slave Events Command**
1320   **(DISEC)** at a later time.)

1321 In cases where the Main Master anticipates that the I3C Bus might be needed for a higher Priority Level
1322 transaction before the newly requested transaction would complete, the Master could either deny the access
1323 (by not acknowledging the request), or else drive the higher-priority Device Address instead of the Address
1324 of the Device sending the IBI request. The Main Master would then hold the SCL line Low until the expected
1325 higher-priority transaction begins.

1326 Bus contention occurs during Address evaluation. As a result, if multiple I3C Devices simultaneously attempt
1327 to win the Bus, then all but one will lose the Arbitration. These Devices will have the opportunity, but are not
1328 required, to repeat the attempt upon the next Bus Available Condition. Note that Slave Devices have the
1329 option to choose to not try again.

5.1.6.3   I3C Secondary Master Requests to be Current Master

1330 When the I3C Secondary Master requests to be a Current Master:

1331 1. To perform the request, the I3C Secondary Master shall wait for either a START (not a Repeated
1332   START), or a Bus Available Condition and issue its own START.

1333 2. After a START, the I3C Secondary Master shall issue its own Dynamic Address on the I3C Bus,
1334   followed by the RnW bit of 0 to request to become Current Master.

1335 3. If the I3C Secondary Master wins the Arbitration by having the lower Dynamic Address, and if the
1336   Current Master is currently willing to relinquish Mastership to the requester, then the Current
1337   Master shall respond with ACK.

1338 4. After the ACK, the Current Master may issue one or more commands, and shall then issue a
1339   GETACCMST CCC (see *Section 5.1.9.3.16*) followed by a STOP, whereupon it releases control
1340   of the SCL line, and therefore also releases Master control of the I3C Bus.

1341 5. Following the STOP and Bus Available Condition, the I3C Secondary Master assumes the role of
1342   the Current Master and takes control of the I3C Bus.

1343 *Figure 44* presents a simplified and generalized picture of the Master to Master Handoff procedure. While
1344 there are many possible variations to this process, the most significant states are as follows:

1345 1. At the end of data transfer, the Current Master (indicated by 'CM' in the Figure) controls SCL and
1346   SDA, and the New Master ('NM' in the Figure) has SCL and SDA in High-Z state.

1347 2. The Current Master provides the rising edge of SCL, while keeping SDA Low

3. After t_{SU_STO} elapses, the Current Master drives SDA High using either an active drive, or the Open-Drain class Pull-Up. However, once SDA is High the Open Drain class Pull-Up shall be used.

4. After the New Master determines that SDA is High, and after both its Clock to Data Turnaround time and the time of flight elapse, it then takes two actions:

   a. The New Master Actively drives SCL High, overlapping with the Current Master's active drive; and

   b. The New Master enables its Open-Drain class Pull-Up, in parallel with the Current Master's Open-Drain class Pull-Up

   Neither of these actions conflicts with the Current Master.

   Although t_{SCO} generally applies to a Slave Device, in this step it applies to the New Master because prior to the Bus transfer the New Master performs in the Slave role.

   In the Figure, the t_{SCO} is shown starting as if the CM has used the Open-Drain drive of SDA High; this depicts the worst condition for this stage of the handoff, showing the latest possible moment when the NM starts overlapping the line controls with the CM.

5. After a time delay of $t_{MMoverlap}$, the Current Master:

   a. Releases SCL to High-Z; and

   b. Disables its Open-Drain class Pull-Up on SDA, and sets SDA to High-Z

   In the Figure, $t_{MMoverlap}$ is shown in the worst case: on the Open-Drain drive of SDA, and where the CM starts counting it from the lower side of the SDA rising edge. The CM must control the lines until the NM could safely take over, and that is certain after $t_{DIG\_OD\_L}$ **Min**. As a result, $t_{MMoverlap}$ shall be greater than or equal to $t_{DIG\_OD\_L}$ **Min**.

6. After t_{MMlock} (the time interval during which the New Master shall not drive SDA Low), the New Master could actively drive SDA Low, producing a START condition.

   The Slaves may also drive SDA Low at this point. This is allowed because SDA is held by an Open-Drain class Pull-Up. The t_{MMlock} period timing parameter is similar to I3C's t_{AVAL} and I^{2}C's t_{BUF} and, as such, requires different values for Pure Bus and for Mixed Bus, respectively.

   In the Figure, t_{MMlock} is shown as referred to the SDA rising edge in Open-Drain driving mode, similar to I3C's t_{AVAL} and I^{2}C's t_{BUF}.

7. After t_{CAS} expires, the New Master may drive SCL Low, producing the first SCL falling edge.

Following this SCL falling edge, the New Master shall actively drive SCL, while also driving SDA in Open-Drain mode with the arbitrable address.

Note:

*The way the I3C Secondary Master indicates that it is a Secondary Master is by properly returning the value 2'b01 for Device Role in its BCR (see **Section 5.1.1.2.1**).*

*The I3C Secondary Master records the Dynamic Addresses of all Devices on the I3C Bus. The preferred method is to record the Address from the Command Code **Define List of Slaves (DEFSLVS)** (see **Section 5.1.9.3.7**).*

*The Master determines that this is a Current Master request by returning the value 2'b01 for Device Role in its BCR (see **Section 5.1.1.2.1**).*

5.1.6.4   I3C Main Master Initiating a Transaction

1388   When initiating an I3C transaction, an I3C Current Master shall perform Dynamic Address Arbitration during
1389   the Address call. Any other I3C Device attempting to interrupt shall win the Arbitration. Interrupting Devices
1390   with lower-priority level shall wait for the next Bus Available Condition.

1391   ***Note:***

1392   *In order to allow the lower-priority Slaves to perform an In-Band Interrupt, Masters may do the*
1393   *following:*

1394   *1)   Wait more than the minimum "Bus Available" period before starting a new communication.*

1395   *2)   Transmit the reserved I3C Broadcast Address, followed by a Repeated START and normal*
1396   *Messages. (The reserved I3C Broadcast Address has lower priority than any interrupts.)*

5.1.7 Secondary Master Functions

1397 Once granted control of the Bus, the Secondary Master maintains control until another Master is granted Bus
1398 control. After the Secondary Master transitions to the Current Master role it could encounter Bus management
1399 activities besides the data transfers that it itself initiates. Some examples are the In-Band Interrupt, or the
1400 Hot-Join request. One optional possibility, shown at *Section 5.1.7.2*, is that the Secondary Master performs
1401 the Current Master's actions with the full capabilities of the Main Master. Another optional possibility is that
1402 the Secondary Master, while serving in the Current Master role, could defer some actions to a more capable
1403 Master, as described in *Section 5.1.7.3*.

1404 A Secondary Master shall support the following scenarios according to its capabilities.

5.1.7.1 Hardware and Software Requirements

1405 The Secondary Master capable of performing the Main Master tasks shall have the following minimum
1406 hardware and software capabilities:

1407 1. Memory for:
1408   a. All Bus Devices' capabilities and functions at system level
1409   b. Retaining the Dynamic Addresses of all I3C Bus Devices
1410   c. Retaining the data transfer protocol that the I3C Bus Devices are capable of
1411 2. Link requirements for Slaves that are intended to transmit In-Band Interrupt requests (IBI), e.g. the
1412   clock speed and the maximum length of data transfer
1413 3. Data transfer protocol capability needed for communicating to all I3C Bus Devices

5.1.7.2 Bus Management Procedures

1414 The Secondary Master capable of performing the Main Master tasks shall perform the following minimum
1415 Bus Management procedures:

1416 1. During Bus initialization, the Secondary Master shall obtain Characteristics of Devices on the Bus
1417   by either using the Command Code **Define List of Slaves (DEFSLVS)** (see *Section 5.1.9.3.7*) sent
1418   by the Main Master, or by monitoring the Bus during Dynamic Address assignment.
1419 2. Perform the Dynamic Address Assignment for Hot-Join Devices
1420   a. The Secondary Master needs to know all Bus Devices' functionality at system level, and such
1421   in order to be able to assign the correct priority level to the Hot-Join Device
1422   b. The Current Master shall issue the Command Code **Define List of Slaves (DEFSLVS)** (see
1423   *Section 5.1.9.3.7*) to ensure that other Masters have the same knowledge
1424 3. Manage the In-Band Interrupt procedure

1425 The Secondary Master needs to know the meaning of the interrupt from the Slave, and needs to
1426 service the interrupt appropriately.

1427 4. Procedure for requesting transfer of Bus control to another Master, including the Main Master

1428 The Secondary Master uses the Command Code **Get Accept Mastership (GETACCMST)** (see
1429 *Section 5.1.9.3.16*).

5.1.7.3 Reduced Functionality Secondary Masters

Secondary Masters are not required to implement and be able to perform all the tasks required of the Main Master. The Secondary Master may defer or transfer the Bus control to a Master when needed; see *Section 5.1.7.1* and *Section 5.1.7.2*.

A Secondary Master with reduced functionality shall have the following minimum capabilities:

1. Memory sufficient for retaining Device Addresses and Characteristics of any Slaves it supports

2. Memory sufficient for retaining the Device Address and Characteristics of the Master to which it will return Bus control

5.1.7.4 In-Band Interrupt Handling

The Secondary Master can either service the In-Band Interrupt request as described in *Section 5.1.6*, or else defer In-Band Interrupt handling to a capable Master by setting the DISINT bit in the Command Code **Disable Slave Events Command (DISEC)** to the interrupting Slave (see *Section 5.1.9.3.1*).

5.1.7.5 Hot-Join Management

The Secondary Master can either manage the Hot-Join event as described in *Section 5.1.5*, or else defer management to a capable Master by issuing a Broadcast Command Code **Disable Slave Events Command (DISEC)** to disable Hot-Join by setting the DISHJ bit (see *Section 5.1.9.3.1*).

5.1.8 Timing Control

1443　This section is not included in the I3C Basic Specification. To gain access to this capability, please contact
1444　MIPI Alliance.

1445　I3C includes optional Timing Control and Timestamping of events generated by I3C Devices resident on the
1446　I3C Bus. This Timing Control framework allows uncertainties affecting the transmission or reception of
1447　timing information (for example: Bus activity, busy Master or Slave, operation latency, system jitter, etc.) to
1448　be nullified.

1449　The I3C Timing Control framework provides:

1450　　• Flexible implementation with significant capability enhancements
1451　　• Means for synchronizing the timing references/clocks, and subsequently the events, of Slave
1452　　　Devices on the I3C Bus to the timing reference/clock of the Master
1453　　　　This can result in reduced energy consumption at the system level.
1454　　• Transmission of timing information, with minimal complexity for the Slave Devices
1455　　• Controllable timing accuracy, suitable for the target use cases

1456　I3C defines two forms of systems and events for Timing Control. Only one form can be used on the I3C Bus
1457　at any time:

1458　　• **Synchronous Time Control** (see *Section 5.1.8.2*): The Master emits a periodic time sync, which
1459　　　allows all Slaves to set their sampling time relative to this time sync. This avoids drift of individual
1460　　　Slaves' clocks. It also ensures that samples occur close together in time, permitting data collected
1461　　　at the same time to be fused. It also provides a calibration method, enabling more accurate
1462　　　sampling between the time syncs.
1463　　• **Asynchronous Time Control** (see *Section 5.1.8.3*): Slaves timestamp the moment at which they
1464　　　acquire sampled data, permitting the Master to time-correlate samples received from multiple,
1465　　　different sensors in an easier and more accurate manner. Timestamping ensures that the Master
1466　　　always has the time at which sensor data acquisition occurred (subject to the timing granularity
1467　　　supported), even if there are latencies in moving the data from the Slave to the Master. Four
1468　　　different Asynchronous Time Control versions ('Modes') provide different methods for measuring
1469　　　time, and/or calibrating timing, with increasing accuracy.

5.1.8.1 General Principles

1470　This section is not included in the I3C Basic Specification. To gain access to this capability, please contact
1471　MIPI Alliance.

1472　The exchange of timing information is based on agreements between the Master and Slaves.

1473　The elements of this agreement are:

1474　1. Two I3C Common Command Codes:
1475　　• Set Exchange Timing Information (SETXTIME), see *Section 5.1.9.3.20*, and
1476　　• Get Exchange Timing Information (GETXTIME), see *Section 5.1.9.3.21*.
1477　　These CCCs:
1478　　• Identify Bus events and markers (i.e., specific SDA edge and/or SCL edge),
1479　　• Transfer timing or control information (e.g., 'abort' command),
1480　　• Specify which timing control procedure is in effect, and
1481　　• Query the Slave Device for Exchange Timing support details, such as which Timing Control
1482　　　Mode(s) are supported, the current Timing Control state, frequency, inaccuracy, etc.
1483　2. One defined and Mode-specific marker is sent by the Master, and used to synchronize the Slaves
1484　3. One data collection event is sent by the Slave. It includes the time information previously
1485　　requested by the Master.

5.1.8.2 Synchronous Systems and Events

1486 This section is not included in the I3C Basic Specification. To gain access to this capability, please contact
1487 MIPI Alliance.

1488 In systems where multiple sensors (or other Slave Devices) provide periodically sampled data, it is
1489 advantageous to instruct the Slaves to be able to collect the data at essentially synchronized times, so that the
1490 Master can read several Slaves' data in a single system awake period.

1491 In a typical system, different Slaves will sample their data at different, uncorrelated times. This is true even
1492 if all Slaves are set to the same sampling frequency, because Slave-to-Slave oscillator accuracy differences
1493 will cause drift over time. This method permits all Slave data sampling to occur very close together in time,
1494 so that Slaves can prepare and activate their individual sampling mechanisms, even if there are variances
1495 across their clocks.

5.1.8.3 Asynchronous Systems and Events

1496 This section is not included in the I3C Basic Specification. To gain access to this capability, please contact
1497 MIPI Alliance.

1498 This Section defines four variations, called Asynchronous Timing Control Modes, on a common procedure
1499 for increasing the precision of an acquired time-of-occurrence of an event in a sensor or other I3C Slave
1500 connected to an I3C Master. The four Asynchronous Modes are presented in order of increasing accuracy
1501 and/or precision.

1502 The actual event itself may or may not be periodic, and the Master and Slave devices need not use the same
1503 time base clock (source, frequency or accuracy). The procedure can help address the large
1504 inaccuracies/offsets to the event's acquired time-of-occurrence that IBI launch/acknowledge latencies can
1505 introduce.

1506
Table 13 Asynchronous Timing Control Modes

Mode	Description
Async Mode 0: Asynchronous Basic Mode	Simple for I3C Master and I3C Slave Allows all I²C out-of-band interrupt usages to be replaced with I3C In-Band Interrupt
Async Mode 1: Asynchronous Advanced Mode	Extension to Async Mode 0 If clock used by Slave's counter drifts, or is not accurate enough, then the Master must send an aperiodic event to limit the running time of the Slave's timing counter.
Async Mode 2: Asynchronous High-Precision Low-Power Mode	Minimizes the time for which the clock driving the Slave's timer counter needs to run. Devices can use burst clock sources, at the cost of Master overhead to measure time and correlate time scales. Time reference is falling SCL when Master ACK's IBI
Async Mode 3: Asynchronous High-Precision Triggerable Low-Power Mode	Precision controlled trigger followed by precision time-stamp of detected event. Time stamp similar in operation to Mode 2, except time reference is to sync signal preceding trigger.

5.1.8.3.1   Async Mode 0: Asynchronous Basic Mode

This section is not included in the I3C Basic Specification. To gain access to this capability, please contact MIPI Alliance.

5.1.8.3.2   Async Mode 1: Asynchronous Advanced Mode

This section is not included in the I3C Basic Specification. To gain access to this capability, please contact MIPI Alliance.

5.1.8.3.3   Async Mode 2: Async High-Precision Low-Power Mode

This section is not included in the I3C Basic Specification. To gain access to this capability, please contact MIPI Alliance.

5.1.8.3.4   Async Mode 3: Async High-Precision Triggereable Mode

This section is not included in the I3C Basic Specification. To gain access to this capability, please contact MIPI Alliance.

5.1.9   Common Command Codes (CCC)

1515 Common Command Codes (CCCs) are globally supported commands that can be transmitted either directly
1516 to a specific I3C Slave Device, or to all I3C Slave Devices simultaneously. This Section specifies how CCCs
1517 are transmitted on the I3C Bus, how each CCC functions, and which CCCs I3C Devices are required to
1518 support.

5.1.9.1   CCC Command Format

1519 The CCC Command Protocol is only formatted using I3C SDR, and always starts with the I3C Broadcast
1520 Address (7'h7E). That is, after a START or Repeated START, the Address of a CCC Command shall always
1521 be a 7'h7E and the RnW bit shall always be a Write.

1522 All I3C Slaves shall recognize both the 7'h7E Broadcast Address, and their own Dynamic Address once it
1523 has been assigned. The I3C Master shall issue a CCC Command both before and after I3C Dynamic
1524 Addresses are assigned.

1525 *Note:*

1526 *Because the I²C Specification (**[NXP01] Section 3.1.12**) reserves the Address value 7'h7E, no*
1527 *Legacy I²C Slave will match the I3C Broadcast Address.*

1528 There are four categories of CCC Command:

1529 1. **Broadcast Write:** A Broadcast Write CCC is seen by all I3C Slaves. All Slaves shall inspect every
1530 received Broadcast command, even if the Slave then ignores the Broadcast command.

1531 Every Broadcast Write CCC Command ends with a Repeated START or a STOP, except
1532 ENTDAA (see *Section 5.1.9.3.4*) which always ends with a STOP.

1533 *Note:*

1534 *If the CCC Command enters a Mode, then the new Mode ends according to its own rules.*

1535 2. **Direct Read/Write:** A Direct Read/Write CCC may alternatively read or write data from/to one or
1536 more specific I3C Slaves, one Slave at a time, selected by the Slave Dynamic Address(es). For
1537 example, it may Write to, and then Read back from, the same Slave to verify that a change was
1538 accepted.

1539 Every Direct Read/Write CCC Command ends with a STOP or a Repeated START,
1540 followed by the I3C Broadcast Address (7'h7E).

1541 *Note:*

1542 *This type may also have data both provided with the CCC (code), as well as with each write.*

1543 3. **Direct Write:** A Direct Write CCC is directed to one or more specific I3C Slaves, selected by the
1544 Slave Dynamic Address(es).

1545 Every Direct Write CCC Command ends with a STOP or a Repeated START, followed by
1546 the I3C Broadcast Address (7'h7E).

1547 4. **Direct Read:** A Direct Read CCC reads data from one or more specific I3C Slaves, one Slave at a
1548 time, selected by the Slave Dynamic Address(es).

1549 Every Direct Read CCC Command ends with a STOP or a Repeated START, followed by
1550 the I3C Broadcast Address (7'h7E).

1551 All CCC Commands share the same general Frame format, which is shown in *Figure 21* for Broadcast CCCs,
1552 and in *Figure 22* for Direct CCCs. Each CCC Command has a unique Command Code. The fields within
1553 this Frame format are detailed in *Table 14*.
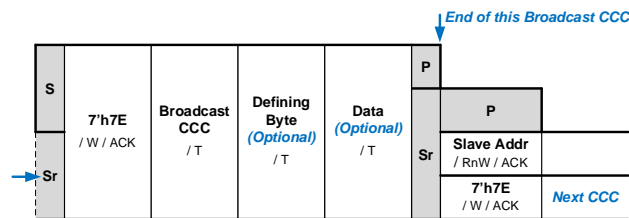
1554
1555
Figure 21 CCC Broadcast General Frame Format
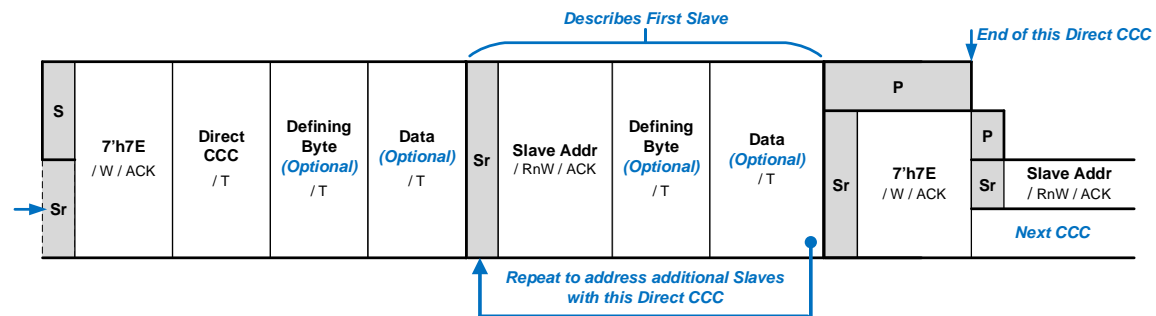
1556
1557
Figure 22 CCC Direct General Frame Format

1558
Table 14 CCC Frame Field Definitions

Field	Definition	
S or **Sr**	A CCC always begins with either START, or Repeated START.	
7'h7E / W / ACK	This field has three parts:	
	7'h7E	The CCC Frame starts with the global Broadcast Address, so that all I3C Slaves on the Bus will see the CCC Code that follows.
	W	The Write Bit is clear (value 1'b0), indicating that the Master is writing a Message to the Slaves.
		This Message always includes the CCC Code, and may optionally include further Data, depending upon the value of the CCC Code.
	ACK	The collective ACK (SDA driven Low) by 1 or more I3C Slaves.
Command Code / T	An 8-bit value indicating which command is being sent, followed by a T-Bit. All defined Command Code values are specified in *Section 5.1.9.3*.	
Defining Byte (Optional) / T	This optional field is used with Broadcast and Direct Read/Write CCC Messages as needed. It is followed by a T-Bit. *Section 5.1.9.3* specifies, for each defined CCC Code the use of a Defining Byte as a sub command, extending its capability.	
Data (Optional) / T	This optional field is used with Broadcast and Direct Read/Write CCC Messages as needed. It is followed by a T-Bit. *Section 5.1.9.3* specifies, for each defined CCC Code, how much Data (if any) appears here.	
Sr / Broadcast Address or **P**	A CCC always ends with either a STOP, or a Repeated START and Broadcast Address.	

5.1.9.2   Broadcast CCCs vs Direct CCCs

1559   The Command Code space is divided into Broadcast Commands and Direct Commands:

1560   • Broadcast Commands are Command Codes 0x00 to 0x7F

1561   • Direct Commands are Command Codes from 0x80 to 0xFE

1562   • Command Code 0xFF is reserved.

1563   As a result, a Slave can easily determine whether a received Command is being Broadcast to all Slaves on
1564   the I3C Bus (1'b0), or is a Direct Command (1'b1) intended only for the particular Slave, by inspecting Bit
1565   7 (MSb) of the Command Code.

5.1.9.2.1   End of a CCC Command

1566   A CCC Command shall end in one of the following three I3C Bus conditions:

1567   • A STOP after the Command or Data

1568   • For a Broadcast Command, a Repeated START (for any Address value)

1569   • For a Direct Command, a Repeated START followed by 7'h7E (which may be the start of a new
1570   CCC, or may be followed by a Repeated START)

1571   If the CCC Command enters a Mode, then the Mode ends according to its own rules.

1572   A 7'h7E following a Repeated START to end a Direct CCC may start another CCC, or may be followed by
1573   a Repeated START.

1574   Although not a normal use, the Master may terminate a Direct CCC Command without addressing any Slave.

1575   If the Master invalidly terminates the data associated with a CCC prematurely, then the Slave shall use best
1576   efforts to handle the termination and ascertain the proper course of action. That is, the Slave may choose to
1577   disregard the event with no effect, or the Slave may process a partial impact from the incomplete data.

5.1.9.2.2   Framing Model for Direct CCC Commands

1578   In Direct CCC Commands the Frame contains first the Command Code, then one or more Repeated STARTs,
1579   then the Address of the targeted Slave Device, followed by Data, and finally either a STOP, or a Repeated
1580   START and 7'h7E.

1581   A single Direct CCC Command may also optionally address more than one Slave Device. To do this, one
1582   additional block is inserted before the final 7'h7E for each additional desired Slave Device (see *Figure 22*).
1583   Each such block consists of Repeated START, the Slave Address of the additional desired Slave Device, and
1584   the Data to be sent to that Slave Device.

1585   With Direct Read/Write Direct CCC Commands, the command code may be followed by data, such as a
1586   defining byte. This will be explained in the details for each CCC. Further, each Slave access may be a Write
1587   with 0 or more data bytes following from the Master, or it may be a Read with 1 or more data bytes following
1588   from the Slave, if it has ACKed the Read. Each Direct CCC will explain whether and how it uses the Write
1589   and Read.

1590   Each addressed Slave Device may either ACK or NACK the Direct CCC Command. For a Read request, the
1591   Slave Device then returns the requested data in accordance with the SDR Standard Read model.

1592   Slaves terminate the Read in I3C, and future Direct Get CCC definitions could be extended to include
1593   additional data bytes, beyond those specified in this version of the I3C Specification. As a result, all I3C
1594   Masters Devices shall ignore any additional, unrecognized data bytes (i.e. more data bytes than this
1595   Specification defines for the given CCC Command) that the Slave Device might return in response to a Direct
1596   Get CCC.

5.1.9.2.3 Retry Model for Direct GET CCC Commands

1597 I3C mandates a single-retry model for Direct GET CCC Commands.

1598 Recall that a Direct GET CCC Command first sends an overall GET command to all Slave Devices using the
1599 Broadcast Address 7'h7E, and then sends targeted Slave Address(es) in order to stimulate per-Slave
1600 response(s). This requires the Slave to be in a state allowing an immediate response if its Address is
1601 transmitted, but also to be prepared for its Address not to be transmitted (and therefore for the Slave not to
1602 have to actually respond). For Direct GET CCC Commands that the Slave supports in hardware, such as
1603 those dealing with information locally known within the Slave (like GETBCR), this requirement generally
1604 presents no issues. But for Direct GET CCCs supported by the Slave's system, or those supported by software
1605 in the Slave, it's possible that the Slave might not be able to respond to the Direct GET CCC in time.

1606 Such situations are handled with the following single-retry model, which applies to Direct GET CCC
1607 Commands only.

1608 If a Slave cannot provide a response to a Direct GET CCC Command in time, then:

1609 1. The Slave shall NACK its Address.

1610 2. The Master shall then emit a Repeated START, followed by the Slave Address. Note that this is a
1611 second transmission of the Slave Address. This gives the Slave extra time to prepare its response.
1612 As a further measure to give the Slave even more time to prepare its response, the Master may
1613 optionally also employ a clock delay (as per *Section 5.1.2.5*) after the Slave's NACK and before
1614 the Master's Repeated START.

1615 3. The Slave should respond to the retry attempt in step 2 with an ACK, and then transmit the results
1616 requested by the Direct GET CCC.

1617 This Retry Model is limited to a single retry. Any Slave still unable (for any reason) to respond to the retry
1618 attempt from step 2 will NACK it. If a Slave NACKs the second attempt in step 2, then the Master shall not
1619 attempt further retries to that Slave.

1620 Note that step 2 only occurs if the Slave NACKs the original Direct GET CCC request. As a result, this retry
1621 model never imposes a time penalty, except in cases where the Slave actually needs the extra time.

5.1.9.3 CCC Command Definitions

Every I3C Common Command Code (CCC) marked as 'Required' in *Table 15* shall be supported by all I3C
Master Devices and by all I3C Slave Devices. When a non-'Required' CCC is supported by an I3C Device,
it shall be implemented as defined in this Specification.

Table 15 I3C Common Command Codes

Command Code	CCC Type	Required	Command Name	Default	Section	Brief Description
0x00	Broadcast	Y	**ENEC** Enable Events Command	Y	*5.1.9.3.1*	Enable Slave event driven interrupts
0x01	Broadcast	Y	**DISEC** Disable Events Command	N	*5.1.9.3.1*	Disable Slave event driven interrupts
0x02	Broadcast	Y[1]	**ENTAS0** Enter Activity State 0	Y	*5.1.9.3.2*	Set Activity Mode to State 0 (normal operation)
0x03	Broadcast	N[1]	**ENTAS1** Enter Activity State 1	N	*5.1.9.3.2*	Set Activity State 1
0x04	Broadcast	N[1]	**ENTAS2** Enter Activity State 2	N	*5.1.9.3.2*	Set Activity State 2
0x05	Broadcast	N[1]	**ENTAS3** Enter Activity State 3	N	*5.1.9.3.2*	Set Activity State 3
0x06	Broadcast	Y	**RSTDAA** Reset Dynamic Address Assignment	–	*5.1.9.3.3*	Forget current Dynamic Address and wait for new assignment
0x07	Broadcast	Y	**ENTDAA** Enter Dynamic Address Assignment	–	*5.1.9.3.4*	Entering Master initiation of Slave Dynamic Address Assignment. Don't participate if the Slave already has an Address assigned.
0x08	Broadcast	N	**DEFSLVS** Define List of Slaves	–	*5.1.9.3.7*	Master defines Dynamic Address, DCR Type, and Static Address (or 0) per Slave
0x09	Broadcast	Y[6]	**SETMWL** Set Max Write Length	–	*5.1.9.3.5*	Maximum write length in a single command
0x0A	Broadcast	Y[7]	**SETMRL** Set Max Read Length	–	*5.1.9.3.6*	Maximum read length in a single command
0x0B	Broadcast	N	**ENTTM** Enter Test Mode	–	*5.1.9.3.8*	Master has entered Test Mode
0x0C – 0x1F	–	–	*MIPI Reserved*	–	*–*	*Reserved for future use by MIPI Alliance*
0x20	Broadcast	N[3]	**ENTHDR0** Enter HDR Mode 0	–	*5.1.9.3.9*	Master has entered HDR–DDR Mode *This Mode is not supported in the I3C Basic Specification. To gain access to this capability, please contact MIPI Alliance.*

Command Code	CCC Type	Required	Command Name	Default	Section	Brief Description
0x21	Broadcast	N[3]	**ENTHDR1** Enter HDR Mode 1	–	*5.1.9.3.9*	Master has entered HDR–TSP Mode *This Mode is not supported in the I3C Basic Specification. To gain access to this capability, please contact MIPI Alliance.*
0x22	Broadcast	N[3]	**ENTHDR2** Enter HDR Mode 2	–	*5.1.9.3.9*	Master has entered HDR–TSL Mode *This Mode is not supported in the I3C Basic Specification. To gain access to this capability, please contact MIPI Alliance.*
0x23	*Broadcast*	*N[3]*	*ENTHDR3 Enter HDR Mode 3*	*–*	*–*	*Master has entered HDR Mode 3*
0x24	*Broadcast*	*N[3]*	*ENTHDR4 Enter HDR Mode 4*	*–*	*–*	*Master has entered HDR Mode 4*
0x25	*Broadcast*	*N[3]*	*ENTHDR5 Enter HDR Mode 5*	*–*	*–*	*Master has entered HDR Mode 5*
0x26	*Broadcast*	*N[3]*	*ENTHDR6 Enter HDR Mode 6*	*–*	*–*	*Master has entered HDR Mode 6*
0x27	*Broadcast*	*N[3]*	*ENTHDR7 Enter HDR Mode 7*	*–*	*–*	*Master has entered HDR Mode 7*
0x28	Broadcast	N	**SETXTIME** Exchange Timing Information	–	*5.1.9.3.20*	This CCC is not included in the I3C Basic Specification. To gain access to this capability please contact MIPI Alliance.
0x29	Broadcast	–	**SETAASA** Set Static Address as Dynamic Address	–	*5.1.9.3.22*	All slaves use their known Static Addresses as their Dynamic Addresses
0x2A – 0x48	*–*	*–*	*MIPI Reserved*	*–*	*–*	*Reserved for future use by MIPI*
0x49 – 0x57	*Broadcast*	*–*	*MIPI Reserved for other WG's – Broadcast CCCs*	*–*	*–*	*Reserved for future use by other MIPI Working Groups*
0x58 – 0x5B	*Broadcast*	*–*	*MIPI Debug WG Reserved – Broadcast CCCs*	*–*	*–*	*Reserved for use by MIPI Debug Working Group*
0x5C – 0x60	*Broadcast*	*–*	*MIPI RIO WG Reserved – Broadcast CCCs*	*–*	*–*	*Reserved for use by MIPI RIO Working Group*
0x61 – 0x7F	Broadcast	–	Vendor Extension – Broadcast CCCs	–	–	For Vendor use
0x80	Direct	Y	**ENEC** Enable Events Command	Y	*5.1.9.3.1*	Enable Slave event driven interrupts
0x81	Direct	Y	**DISEC** Disable Events Command	N	*5.1.9.3.1*	Disable Slave event driven interrupts
0x82	Direct	Y[1]	**ENTAS0** Enter Activity State 0	Y	*5.1.9.3.2*	Set Activity Mode to State 0 (normal operation)

Command Code	CCC Type	Required	Command Name	Default	Section	Brief Description
0x83	Direct	N[1]	**ENTAS1** Enter Activity State 1	N	*5.1.9.3.2*	Set activity State 1
0x84	Direct	N[1]	**ENTAS2** Enter Activity State 2	N	*5.1.9.3.2*	Set activity State 2
0x85	Direct	N[1]	**ENTAS3** Enter Activity State 3	N	*5.1.9.3.2*	Set activity State 3
0x86	Direct	Y	**RSTDAA** Reset Dynamic Address Assignment	–	*5.1.9.3.3*	Forget current Dynamic Address and wait for new assignment
0x87	Direct Set	N	**SETDASA** Set Dynamic Address from Static Address	–	*5.1.9.3.10*	Master assigns a Dynamic Address to a Slave with a known Static Address.
0x88	Direct Set	Y	**SETNEWDA** Set New Dynamic Address	–	*5.1.9.3.11*	Master assigns a new Dynamic Address to any I3C Slave
0x89	Direct Set	Y[2]	**SETMWL** Set Max Write Length	–	*5.1.9.3.5*	Maximum write length in a single command
0x8A	Direct Set	Y[2]	**SETMRL** Set Max Read Length	–	*5.1.9.3.6*	Maximum read length in a single command
0x8B	Direct Get	Y[2]	**GETMWL** Get Max Write Length	–	*5.1.9.3.5*	Get Slave's maximum possible write length
0x8C	Direct Get	Y[2]	**GETMRL** Get Max Read Length	–	*5.1.9.3.6*	Get a Slave's maximum possible read length
0x8D	Direct Get	Y	**GETPID** Get Provisional ID	–	*5.1.9.3.12*	Get a Slave's Provisional ID
0x8E	Direct Get	Y	**GETBCR** Get Bus Characteristics Register	–	*5.1.9.3.13*	Get a Device's Bus Characteristic Register (BCR)
0x8F	Direct Get	Y	**GETDCR** Get Device Characteristics Register	–	*5.1.9.3.14*	Get a Device's Device Characteristics Register (DCR)
0x90	Direct Get	Y	**GETSTATUS** Get Device Status	–	*5.1.9.3.15*	Get a Device's operating status
0x91	Direct Get	N	**GETACCMST** Get Accept Mastership	–	*5.1.9.3.16*	Current Master is requesting and confirming a Bus Mastership from a Secondary Master

Command Code	CCC Type	Required	Command Name	Default	Section	Brief Description
0x92	–	–	*MIPI Reserved*	–	–	*Reserved for future use by MIPI*
0x93	Direct Set	N	**SETBRGTGT** Set Bridge Targets	–	*5.1.9.3.17*	Master tells Bridge (to/from I²C, SPI, UART, etc.) what endpoints it is talking to (by Dynamic Address and type/ID).
0x94	Direct Get	N[4]	**GETMXDS** Get Max Data Speed	–	*5.1.9.3.18*	Master asks Slave for its SDR Mode max. Read and Write data speeds (& optionally max. Read Turnaround time)
0x95	Direct Get	N[5]	**GETHDRCAP** Get HDR Capability	–	*5.1.9.3.19*	This CCC is not included in the I3C Basic Specification. To gain access to this capability please contact MIPI Alliance.
0x96 – 0x97	–	–	*MIPI Reserved*	–	–	*Reserved for future use by MIPI*
0x98	Direct	N	**SETXTIME** Set Exchange Timing Information	–	*5.1.9.3.20*	This CCC is not included in the I3C Basic Specification. To gain access to this capability please contact MIPI Alliance.
0x99	Direct	N	**GETXTIME** Get Exchange Timing Information	–	*5.1.9.3.21*	This CCC is not included in the I3C Basic Specification. To gain access to this capability please contact MIPI Alliance.
0x9A – 0xBF	*Direct*	–	*MIPI Reserved – Direct CCCs*	–	–	*Reserved for future use by MIPI*
0xC0 – 0xD6	*Direct*	–	*MIPI Reserved for other WG's – Direct CCCs*	–	–	*Reserved for future use by MIPI*
0xD7 – 0xDA	*Direct*	–	*MIPI Debug WG Reserved – Direct CCCs*	–	–	*Reserved for use by MIPI Debug WG*
0xDB – 0xDF	*Direct*	–	*MIPI RIO WG Reserved – Direct CCCs*	–	–	*Reserved for use by MIPI RIO WG*
0xE0 – 0xFE	Direct	–	Vendor Extension – Direct CCCs	–	–	For Vendor use
0xFF	–	–	*MIPI Reserved*	–	–	*Reserved for future use by MIPI*

Note:

  1) *Slave Devices shall be permitted to self-power-manage based on this information.*

  2) *This CCC shall be supported by Devices capable of transporting 16 or more sequential bytes.*

  3) *HDR Modes are not supported in I3C Basic, however in order to maintain compatibility with I3C v1.x, Devices must detect the HDR Enter, HDR Restart, and HDR Exit patterns.*

  4) *This CCC shall be supported by Slave Devices with Bus Control Register (BCR) Bit [0] set to 1'b1.*

  5) *This CCC shall be supported by Slave Devices that support any HDR Mode.6) See* **Section 5.1.9.3.5 Set/Get Max Write Length (SETMWL/GETMWL)**.

  7) *See* **Section 5.1.9.3.6** *Set/Get Max Read Length (SETMRL/GETMRL).*

5.1.9.3.1 Enable/Disable Slave Events Command (ENEC/DISEC)

1626 These four Direct (*Table 16*) or Broadcast (*Table 17*) CCCs allows the Master to control when Slave-initiated
1627 traffic is (Enable) vs. is not (Disable) allowed on the I3C Bus. This control governs a Slave's attempts to
1628 request an Interrupt (ENINT/DISINT), to request Mastership (ENMR/DISMR), or to signify a Hot-Join event
1629 (ENHJ/DISHJ).

1630
Table 16 ENEC/DISEC Format 1: Direct

S	7'h7E / W / ACK	Direct ENEC/DISEC CCC / T	Sr	Slave Addr / W / ACK	En/Dis Slave Event Byte / T	Sr	7'h7E
							P

1631
Table 17 ENEC/DISEC Format 2: Broadcast

S Sr	7'h7E / W / ACK	Broadcast ENEC/DISEC CCC / T	Enable/Disable Slave Event Byte / T	Sr	7'h7E
					P

1632 *Table 18* and *Table 19* show the bits to set in the Slave Event Byte to enable and disable, respectively, the
1633 four supported I3C Slave/Secondary Master event types. Reserved bits are for future MIPI use.

1634
Table 18 Enable Slave Events Command Byte Format

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved				ENHJ	Reserved	ENMR	ENINT

1635
Table 19 Disable Slave Events Command Byte Format

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved				DISHJ	Reserved	DISMR	DISINT

1636 The supported I3C Slave/Secondary Master event types are:

1637 • **Slave Interrupt Requests:** Enable (ENINT) / Disable (DISINT)

1638 These bits allow the Master to control when Slave-initiated Interrupts are (ENINT) and are not
1639 (DISINT) allowed on the I3C Bus.

1640 • **Mastership Requests:** Enable (ENMR) / Disable (DISMR)

1641 These bits allow the Current Master to control when Mastership requests from Secondary Masters
1642 are (ENMR) and are not (DISMR) allowed on the I3C Bus.

1643 • **Hot-Join Event:** Enable (ENHJ) / Disable (DISHJ)

1644 These bits allow the Master to control when Slave-initiated Hot-Join is (ENHJ) and is not (DISHJ)
1645 allowed on the I3C Bus. The Master can Broadcast this CCC in order to command Devices to
1646 refrain from making Dynamic Address Assignment requests until later authorized by the Master, in
1647 case the Master isn't ready to service the Hot-Joining Device(s). (Hot-Join events are
1648 asynchronous.)

5.1.9.3.2   Enter Activity State 0–3 (ENTAS0–ENTAS3)

These four Direct (*Table 20*) and four Broadcast (*Table 21*) CCCs allow the Master to inform one or all Slave Devices that it will not be active on the I3C Bus for an approximated amount of time, so that the Slave Devices may use a lower power state during that period. There is one Direct CCC and one Broadcast CCC per Activity State. All I3C Slave Devices shall maintain I3C communications capabilities in all Activity States.

Table 20 ENTASx Format 1: Direct

S	7'h7E / W / ACK	Direct ENTASx CCC / T	Sr	Slave Addr / W / ACK	Sr	7'h7E
						P

Table 21 ENTASx Format 2: Broadcast

S / Sr	7'h7E / W / ACK	Broadcast ENTASx CCC / T	Sr	7'h7E
				P

Table 22 below gives the expected minimum Bus activity interval for each Activity State.

Table 22 Enter Activity State CCCs (ENTASx)

CCC	Activity State	Minimum Bus Activity Interval
ENTAS0	Activity State 0	1 µSec: Latency-free operation
ENTAS1	Activity State 1	100 µSec
ENTAS2	Activity State 2	2 mSec
ENTAS3	Activity State 3	50 mSec: Lowest-activity operation

5.1.9.3.3   Reset Dynamic Address Assignment (RSTDAA)

This Direct (*Table 23*) or Broadcast (*Table 24*) CCC indicates to one or all I3C Devices that the Master requires them to clear/reset their Master-assigned Dynamic Address. After clearing their Dynamic Address, the Devices are ready to participate in a Dynamic Address Assignment procedure, per *Section 5.1.4*.

Table 23 RSTDAA Format 1: Direct

S	7'h7E / W / ACK	Direct RSTDAA CCC / T	Sr	Slave Addr / W / ACK	Sr	7'h7E
						P

Table 24 RSTDAA Format 2: Broadcast

S / Sr	7'h7E / W / ACK	Broadcast RSTDAA CCC / T	Sr	7'h7E
				P

5.1.9.3.4 Enter Dynamic Address Assignment (ENTDAA)

This Broadcast CCC (*Table 25*) indicates to all I3C Devices that the Master requires them to enter the Dynamic Address Assignment procedure described in *Section 5.1.4*. Slave Devices that already have a Dynamic Address assigned shall not respond to this command.

To exit from ENTDAA mode, the Master shall issue a STOP, and shall not use a Repeated START.

Table 25 ENTDAA Format

S / Sr	7'h7E / W / ACK	ENTDAA CCC / T	P

5.1.9.3.5 Set/Get Max Write Length (SETMWL/GETMWL)

These Direct and Broadcast CCCs (Direct Set or Get in *Table 26*, Broadcast Set in *Table 27*) allow the I3C Master to Set or Get a maximum data write length in bytes for one Slave Device. This Max Write Length does not affect data write lengths for Broadcast CCCs. The Set/Get Max Write Length value is transmitted over two bytes, with the most significant byte (MSb) transmitted first. The minimum value that Max Write Length can be set to is 8.

This CCC is required if (and only if) any private Write Message(s) and/or any extended Write CCC(s) implemented by the Slave Device support a variable limit on the maximum number of data bytes per Message, and this limit is greater than 8 bytes. This allows the Slave to limit the number of bytes the Master sends. A Slave Device with no such settable limit may optionally support this CCC, but is not expected to do so.

Table 26 Direct SETMWL/GETMWL Format

S	7'h7E / W / ACK	SETMWL or GETMWL CCC / T	Sr	Slave Addr / RnW / ACK	MWL MSb / T	MWL LSb / T	Sr	7'h7E
								P

Table 27 Broadcast SETMWL Format

S	7'h7E / W / ACK	SETMWL CCC / T	MWL MSb / T	MWL LSb / T	Sr	7'h7E
						P

5.1.9.3.6   Set/Get Max Read Length (SETMRL/GETMRL)

1680   These Direct and Broadcast CCCs (Direct Set or Get in *Table 28*, Broadcast Set in *Table 29*) allow the I3C
1681   Master to Set or Get a maximum data read length, and optionally a maximum IBI payload size.

1682   The Set/Get Max Read Length value is transmitted over the first two bytes, with most significant byte (MSb)
1683   transmitted first. The minimum value to which Max Read Length can be set is 16.

1684   For devices with BCR bit 2 set to 1'b1, the Max IBI payload size value is added as a third byte, where a value
1685   of 0 indicates an unlimited payload size. The minimum IBI payload size is one (one byte).

1686   This CCC is optional for the Slave, with two exceptions:

1687   1. This CCC is required if both (a) any private Read Request Message(s) and/or any extended Read
1688   Request CCC(s) implemented by the Slave support a variable limit on the maximum number of
1689   data bytes that the Slave may return per Message, and (b) this limit is greater than 16 bytes.

1690   2. This CCC is required if the Slave both (a) supports an IBI Payload (as indicated with BCR bit 1),
1691   and (b) will transmit more than one byte of private payload.

1692   **Table 28 Direct SETMRL/GETMRL Format**

S	7'h7E / W / ACK	SETMRL or GETMRL CCC / T	Sr	Slave Addr / RnW / ACK	MRL MSb / T	MRL LSb / T	IBI Payload Size / T	Sr	7'h7E
									P

1693   **Table 29 Broadcast SETMRL Format**

S	7'h7E / W / ACK	SETMRL CCC / T	MRL MSb / T	MRL LSb / T	IBI Payload Size / T	Sr	7'h7E
							P

5.1.9.3.7 Define List of Slaves (DEFSLVS)

1694 This Broadcast CCC (*Table 30*) is only relevant to Secondary Masters, which may independently elect either
1695 to respond to this CCC, or to ignore it. This CCC tells Secondary Master Devices what Slaves are present on
1696 the I3C Bus, via four consecutive Data Bytes per Slave. First the Current Master identifies itself by
1697 transmitting its own data in the first set of four data bytes, using the value 7'h7E as the Static Address. Then
1698 each additional Slave on the I3C Bus Slave is represented by a further set of four data bytes.

1699

Table 30 DEFSLVS Format

S Sr	7'h7E / W Sr / ACK	DEFSLVS CCC / T	Count / T	Describes Current Master				Describes First Slave (Any additional Slaves will follow)				Sr	7'h7E
				Dynamic Addr Master / T	DCR Type Master / T	BCR Type Master / T	Static Addr 7'h7E / T	Dynamic Addr 0 / T	DCR Type 0 / T	BCR Type 0 / T	Static Addr 0 / T		P

1700 Count is the number of Slaves present on the I3C Bus. Each Slave is represented by a set of four data bytes:

1701 1. **Dynamic Address:** The 7 most significant bits (Bits [7:1]) contain the current value of the Slave's
1702   Main Master-assigned 7-bit Dynamic Address, and the least significant bit (Bit [0]) is filled with
1703   the value 1'b0. For Legacy I^{2}C Devices, the value of the Dynamic Address shall be 7'h00.

1704 2. **DCR Type:** The Slave's Device Characteristics Register value, or 0x00 if unknown. For Legacy
1705   I^{2}C Devices the value of the DCR Type shall be the Device's LVR value.

1706 3. **BCR Type:** The Slave's Bus Characteristics Register value, or 0x00 if unknown.

1707 4. **Static Address:** The Slave's original 7-bit static I^{2}C Address in the 7 most significant bits
1708   (Bits [7:1]) with the least significant bit (Bit [0]) filled with the value 1'b0. If no Static Address,
1709   then the value shall be 7'h00.

1710 For a Legacy I^{2}C Device, the value of the Dynamic Address field will be 7'h00, and the DCR field will
1711 contain the value of the Device's Legacy Virtual Register (LVR).

1712 The Master shall not send the DEFSLVS CCC unless at least one Secondary Master Device is present on the
1713 I3C Bus. The Master shall send the DEFSLVS CCC following every "Hot-Join" event.

5.1.9.3.8 Enter Test Mode (ENTTM)

1714 This Broadcast CCC (*Table 31*) informs all I3C Devices that the Master is entering a specified Test Mode
1715 during manufacturing or Device test. The Enter Test Mode command Frame format includes a byte that
1716 specifies which Test Mode to enter. Supporting I3C Devices shall enter the indicated Test Mode upon receipt
1717 of the Enter Test Mode CCC. *Table 32* lists the defined Test Mode byte values.

1718 **Table 31 ENTTM Format**

S	7'h7E / W / ACK	ENTTM CCC / T	Test Mode Byte / T	Sr	7'h7E
				P	

1719 **Table 32 ENTTM Test Mode Byte Values**

Byte Value	I3CTest Mode	Description
0x00	Exit Test Mode	This value removes all I3C Devices from Test Mode
0x01	Vendor Test Mode	This value indicates that I3C Devices shall return a random 32-bit value in the Provisional ID during the Dynamic Address Assignment procedure
0x02 – 0xFF	MIPI Reserved	Reserved for future use by the MIPI Alliance

5.1.9.3.9 Enter HDR Mode 0–7 (ENTHDR0–ENTHDR7)

1720 These eight Broadcast CCCs inform all I3C Devices that the Bus is being switched into the indicated HDR
1721 Mode. These Modes are not supported in the I3C Basic Specification. To gain access to this capability, please
1722 contact MIPI Alliance. See *Section 5.2* for further details on the HDR Modes.

1723 **Table 33 Enter HDR Mode CCCs (ENTHDRx)**

CCC	Enter HDR Mode	Mode Name	See Section
ENTHDR0	HDR Mode 0	HDR-DDR	*Section 5.2.2*
ENTHDR1	HDR Mode 1	HDR-TSP	*Section 5.2.3*
ENTHDR2	HDR Mode 2	HDR-TSL	*Section 5.2.3*
ENTHDR3 to ENTHDR7	HDR Modes 3-7	Reserved for future definition by MIPI Alliance	

5.1.9.3.10 Set Dynamic Address from Static Address (SETDASA)

1724 This CCC (**Table 34** and **Table 35** illustrate the two distinct command formats) allows the Master to assign a
1725 Dynamic Address to one Slave using the Slave's Static Address. This is faster than the ENTDAA Dynamic
1726 Address Assignment procedure (see **Section 5.1.9.3.4**). The SETDASA CCC should be used before the
1727 ENTDAA CCC is used; all Slaves without assigned Dynamic Addresses will respond to the ENTDAA CCC.

1728 ***Note:***

1729 *The SETDASA Command Code is unusual in that it addresses the desired I3C Device by its I²C*
1730 *Static Address, rather than by its Dynamic Address. As a result, this CCC can only be sent to a Slave*
1731 *that has an I²C Static Address.*

1732 The newly assigned Address is transmitted in the Dynamic Address Byte shown in **Table 34**, where the 7
1733 most significant bits (Bits [7:1]) contain the 7-bit Dynamic Address, and the least significant bit (Bit [0]) is
1734 filled with the value 1'b0.

1735 **Table 34 SETDASA Format 1: Primary**

S	7'h7E / W / ACK	SETDASA CCC / T	Sr	Slave Addr / W / ACK	7-bit Dynamic Address / 0 / T	Sr	7'h7E
							P

SETDASA Minimal Bus Point-to-Point Communication

1736 The SETDASA CCC may also be specially used for simple point-to-point communication in I3C Minimal
1737 Bus use cases. An I3C Minimal Bus is an I3C Bus with one I3C Master Device (potentially with reduced
1738 functionality), and one I3C Slave Device. In this special usage of the SETDASA CCC, the Master Device
1739 both uses the fixed value 7'h01 as the Static Address, and uses the fixed (and reserved) value of 7'h01 as the
1740 Dynamic Address (see **Table 35**). This special usage of the SETDASA CCC allows for simpler Master
1741 Devices, and optionally simpler Slave Devices intended for use specifically in Minimal Bus configurations.

1742 I3C Slave Devices should support this special usage of the SETDASA CCC unless such support would make
1743 the Slave Device unusable in a Minimal Bus use case. A Slave Device supporting this special Minimal Bus
1744 usage of the SETDASA CCC shall match the Static Address 7'h01, and then accept 7'h01 as its new Dynamic
1745 Address (same as the natural result of SETDASA). The Slave Device may choose to behave differently after
1746 receiving its Dynamic Address in this manner.

1747 An I3C Master shall not use this special form of the SETDASA CCC unless the I3C Bus is known to have
1748 precisely one Master and either:

1749 • One active I3C Slave Device that is capable of supporting this special usage of the SETDASA
1750   CCC, or

1751 • One or more I3C Slave Devices that act strictly as receivers, i.e., that do not use In-Band Interrupt
1752   and that will not be sent Read requests. This arrangement permits a single Master Device to, in
1753   effect, Broadcast to the Slave Devices.

1754 **Table 35 SETDASA Format 2: Point-to-Point**

S	7'h7E / W / ACK	SETDASA CCC / T	Sr	7'h01 / W / ACK	7'h01 / 0 / T	Sr	7'h7E
							P

5.1.9.3.11 Set New Dynamic Address (SETNEWDA)

1755 This Direct CCC (*Table 36*) allows the I3C Master to assign a new Dynamic Address to one I3C Slave Device.
1756 In the Dynamic Address field, the 7 most significant bits (Bits [7:1]) contain the 7-bit Dynamic Address, and
1757 the least significant bit (Bit [0]) is filled with the value 1'b0.

1758

Table 36 SETNEWDA Format

S	7'h7E / W / ACK	SETNEWDA CCC / T	Sr	Slave Addr / W / ACK	7-bit Dynamic Address / 0 / T	Sr	7'h7E
							P

5.1.9.3.12 Get Provisional ID (GETPID)

1759 This Direct CCC (see *Figure 19* and *Table 37*) is a Get request for one I3C Slave Device to return its 48-bit
1760 Provisional ID to the Master, as described in *Section 5.1.4*. Following transmission of the GETPID CCC, the
1761 48-bit value is transmitted as 6 bytes, with MSb first.

1762

Table 37 GETPID Format

S	7'h7E / W / ACK	GETPID CCC / T	Sr	Slave Addr / R / ACK	GETPID Byte 5 / T	GETPID Byte 4 / T	GETPID Byte 3 / T	GETPID Byte 2 / T	GETPID Byte 1 / T	GETPID Byte 0 / T	Sr	7'h7E
												P

5.1.9.3.13 Get Bus Characteristics Register (GETBCR)

1763 This Direct CCC (*Table 38*) is a Get request for one I3C Slave Device to return its Bus Characteristics
1764 Register (BCR) to the Master, as described in *Section 5.1.1.2.1*. The BCR value is transmitted in one byte,
1765 with the MSb transmitted first.

1766

Table 38 GETBCR Format

S	7'h7E / W / ACK	GETBCR CCC / T	Sr	Slave Addr / R / ACK	GETBCR Byte / T	Sr	7'h7E
							P

5.1.9.3.14 Get Device Characteristics Register (GETDCR)

1767 This Direct CCC (*Table 39*) is a Get request for one I3C Slave Device to return its Device Characteristics
1768 Register (DCR) to the Master, as described in *Section 5.1.1.2.2*. The DCR value is transmitted in one byte,
1769 with the MSb transmitted first.

1770

Table 39 GETDCR Format

S	7'h7E / W / ACK	GETDCR CCC / T	Sr	Slave Addr / R / ACK	GETDCR Byte / T	Sr	7'h7E
							P

5.1.9.3.15   Get Device Status (GETSTATUS)

1771   This Direct CCC (*Table 40*) is a Get request for one I3C Slave Device to return its current Status, in the two-
1772   byte format detailed in *Table 41*. Note that byte 0 is the LSb, and byte 1 is the MSb.

1773   **Table 40 GETSTATUS Format**

S	7'h7E / W / ACK	GETSTATUS CCC / T	Sr	Slave Addr / R / ACK	GETSTATUS MSb / T	GETSTATUS LSb / T	Sr	7'h7E P

1774   **Table 41 GETSTATUS MSb-LSb Format**

Bits	Field	Description
15:8	**Vendor Reserved**	Reserved for vendor-specific meaning.
7:6	**Activity Mode**	Contains the two-bit ID of the Slave Device's current Activity Mode (readiness to support data read of sensor or related information).
5	**Protocol Error**	If set to 1'b1, then the Slave detected a protocol error since the last Status read. The Slave might or might not be able to check for such errors. Note that this value self-clears upon every successful completion of a Master read of the Slave's Status.
4	**Reserved**	Reserved for future definition by the MIPI.
3:0	**Pending Interrupt**	Contains the interrupt number of any pending interrupt, or 0 if no interrupts are pending. This encoding allows for up to 15 numbered interrupts. If more than one interrupt is set, then the highest priority interrupt shall be returned.

5.1.9.3.16   Get Accept Mastership (GETACCMST)

1775 This Direct Get CCC (*Table 42*) is used both to verify a Master Request, and to allow the Current Master to
1776 offer Mastership to an I3C Secondary Master. The Get is used to confirm acceptance; to do so, the Secondary
1777 Master returns its 7-bit Dynamic Address as shown in *Table 42*. The value of the 7-bit Dynamic Address will
1778 be the same as the value of the Slave Address. If the Secondary Master does not accept, then it shall NACK
1779 the Get request as shown in *Table 43*.

1780 A Secondary Master requesting Mastership will gain Mastership only if all four of the following steps occur
1781 in the indicated order. For a Current Master offering Mastership, only steps 2 through 4 apply.

1782 1.   The Current Master ACKs the Mastership request

1783 2.   The Current Master transmits a GETACCMST command to the Secondary Master

1784 3.   The Secondary Master correctly replies with its 7-bit Dynamic Address. The value of the 7-bit
1785   Dynamic Address will be the same as the value of the Slave Address.

1786   If the Secondary Master instead responds with NACK (*Table 43*), or with an incorrect 7-
1787   bit Address (*Table 44*), then:

1788   • The Secondary Master will not acquire Mastership

1789   • The GETACCMST command is canceled, and

1790   • The Current Master can then either (a) Retry the CCC, or (b) End the transaction by
1791   providing a STOP.

1792 4.   The Current Master issues a STOP

1793   If the Current Master instead issues a Repeated START, then:

1794   • The Secondary Master will not gain Mastership

1795   • The GETACCMST command is canceled (*Table 44*), and

1796   • The Current Master can then either (a) Retry the CCC, or (b) End the transaction by
1797   providing a STOP.

1798 **Table 42 GETACCMST Format 1: Accepted**

S	7'h7E / W / ACK	GETACCMST CCC / T	Sr	Slave Addr / R / ACK	7-Bit Dynamic Address / PAR / T	P

1799 ***Note:***

1800 *The value of the 7-Bit Dynamic Address will be the same as the value of the Slave Addr.*

1801 **Table 43 GETACCMST Format 2: Not Accepted**

S	7'h7E / W / ACK	GETACCMST CCC / T	Sr	Slave Addr / R / NACK	Sr	7'h7E or **Retry** P

1802 **Table 44 GETACCMST Format 3: Incorrect Cancel**

S	7'h7E / W / ACK	GETACCMST CCC / T	Sr	Slave Addr / R / ACK	Incorrect 7-Bit Dynamic Address / PAR / T	Sr	7'h7E or **Retry** P

5.1.9.3.17 Set Bridge Targets (SETBRGTGT)

This Direct CCC (*Table 45*) is only relevant to Bridging Devices, i.e., I3C Devices that the Master knows in advance (not by inspecting BCR bits) to be Bridging Devices. An I3C Master shall only send this CCC to known Bridging Devices that accept it.

Table 45 SETBRGTGT Format

S / Sr	7'h7E / W / ACK	SETBRGTGT CCC / T	Sr	Slave Addr / W / ACK	Count / T	Describes Bridged Slave 0			Describes Bridged Slave 1 (Any additional Slaves will follow)			Sr	7'h7E
						Dynamic Addr 0 / T	ID0 / T	ID0 / T	Dynamic Addr 1 / T	ID1 / T	ID1 / T		P

Slave Addr is either the Slave's original 7-bit static I²C Address, or else 7'h00 if the Slave had no such static I²C Address.

Count is the number of Bridged "Slaves".

Each described Bridged "Slave" is represented by two fields:

- **Dynamic Address:** The 7 most significant bits (Bits [7:1]) contain the current value of the Slave's Main Master-assigned 7-bit Dynamic Address, and the least significant bit (Bit [0]) is filled with the value 1'b0. For Legacy I²C Devices, the value of the Dynamic Address shall be 7'h00.
- **ID:** A 16-bit unambiguous identifier for the Bridged Device (two bytes)

The meaning of the ID field is not defined by this Specification, and should be a contract between the Bridge and the Master (or any other entity that supplies such information to the Master). For example, the upper nibble could indicate the communication type (e.g. I²C, SPI, UART, LIN, etc.), and the lower 12 bits could be the port (as port number and Device selector).

5.1.9.3.18 Get Max Data Speed (GETMXDS)

1819 The Master uses this Direct CCC (there are two formats: *Table 46* and *Table 47*) to determine the SDR Mode
1820 data speed limitations of one Slave Device. The Master is required to use this CCC only when Bit [0] of the
1821 addressed Slave Device's Bus Control Register (BCR) is set to 1'b1 (see *Table 6*). A Slave Device is required
1822 to support this CCC only when Bit [0] of its Bus Control Register (BCR) is set to 1'b1 (see *Table 6*).

1823 Bit [0] of the Bus Characteristics Register shall be set in any Slave Device with a Clock-to-Data turnaround
1824 time greater than 12 ns. Slave Devices within the t_{SCO} delay range should not set the limitation bit (i.e., Bit [0])
1825 unless other limitations exist. But Devices should support this CCC to allow better factoring of each number
1826 when computing the maximum effective frequency for reads, since the Master/System-designer can be told
1827 the t_{SCO} parameter along with line length (propagation time), line capacitance, number of Slaves, and stubs
1828 if present.

1829 The t_{SCO} delay is the measurement of the total internal delay from the SCL input to the SDA output, excluding
1830 other factors like line capacitance and path delay that are factored out of the device computation and put into
1831 the broader computation. The t_{SCO} delay is applicable to both Master and Slave Devices, since Master Devices
1832 may behave as Slave device in a multi-Master system based on the system configuration.

t1: time from CLK FLOP to Q to PAD
t2: time through output pad (PFET/NFET)(tested over a 90 Ohm, 50pF line capacitance)
t3: time over wires pad-pad: master drive + Line Cap + Tpath
t4: time through input pad of slave
t5: time through output pad of slave (tested over a 90 Ohm, 50pF line capacitance)
t6: tSCO : time inside slave from SCL input to SDA reply out (including pad delays t4 and t5)
t7: time over wires pad-pad : Tpath + slave drive+ line cap
t8: time through iData input pad (Schmitt input)
t9: time from iData pad to D input for serializer

1833
1834 **Figure 23 Components of Clock-to-Data Turnaround Delay (t_{SCO})**

1835 Any Slave Devices with a Clock-to-Data turnaround (t_{SCO}) delay greater than 12 ns shall set the Clock to
1836 Data Turnaround field of the maxRD Byte to 3'b111 and report this value to the Master by private agreement.

1837 Maximum Read Turnaround Time is used to notify the Master how long to wait before reading the data it
1838 requested. This allows the Master to get the Slave Device's turnaround delay time for data read requests
1839 (excluding In-Band Interrupt responses). This is useful because some Slaves may exhibit data read delays (as
1840 contrasted with CCC reads) due to clock-starting effects, bridging, slower inner processors, etc. The Master
1841 can use the returned delay factor to avoid NACKs that would result from reading the Device too soon. The
1842 Slave may have the data ready before the time reported, and the Master can also estimate the time and retry
1843 reading before the maximum Read Turnaround time. The Master should be aware that any NACKs for read
1844 requests from the Slave before the maximum time is a normal condition, and no recovery methods are
1845 required; by contrast, NACKs after the maximum Data Turnaround time shall be dealt with accordingly.

₁₈₄₆ ***Note:***

₁₈₄₇ *This CCC relates to bit data rates and Read Turnaround times, not data sizes. Data sizes are the*
₁₈₄₈ *subject of the CCCs Get Max Read Length (GETMRL, see **Section 5.1.9.3.6**) and Get Max Write*
₁₈₄₉ *Length (GETMWL, see **Section 5.1.9.3.5**).*

₁₈₅₀ The Slave Device shall return either:

₁₈₅₁ • **GETMXDS Format 1:** Two data bytes containing the Slave Device's Maximum Write Speed and
₁₈₅₂ Maximum Read Speed (see ***Table 46***), or

₁₈₅₃ • **GETMXDS Format 2:** Five data bytes containing the Slave Device's Maximum Write Speed,
₁₈₅₄ Maximum Read Speed, and a three-byte Maximum Read Turnaround Time (see ***Table 47***).

₁₈₅₅ The Slave signals which Format it is returning via the number of returned data bytes: Two bytes indicates
₁₈₅₆ Format 1, and five bytes indicates Format 2.

₁₈₅₇ The Slave's selection of Format 1 vs. Format 2 should be based upon whether Maximum Read Turnaround
₁₈₅₈ Time needs to be communicated to the Master Device.

₁₈₅₉ Interpretation of the returned fields maxWr, maxRd, and (for Format 2 only) maxRdTurn is shown in ***Table***
₁₈₆₀ ***48***, ***Table 49***, and ***Table 50*** respectively.

₁₈₆₁ **Table 46 GETMXDS Format 1: Without Turnaround**

S	7'h7E / W / ACK	GETMXDS CCC / T	Sr	Slave Addr / R / ACK	maxWr / T	maxRd / T	Sr	7'h7E
								P

₁₈₆₂ **Table 47 GETMXDS Format 2: With Turnaround**

S	7'h7E / W / ACK	GETMXDS CCC / T	Sr	Slave Addr / R / ACK	maxWr / T	maxRd / T	maxRdTurn / T	Sr	7'h7E
									P

₁₈₆₃ **Table 48 maxWr Byte Format**

Bits	Field	Description
7:3	Reserved	Reserved for future use by the MIPI Alliance
2:0	**Maximum Sustained Data Rate for non-CCC messages sent by Master Device to Slave Device**	0: f_{SCL} Max (default value) 1: 8 MHz 2: 6 MHz 3: 4 MHz 4: 2 MHz 5–7: Reserved for future use by the MIPI Alliance

¹⁸⁶⁴
Table 49 maxRd Byte Format

Bits	Field	Description
7:6	Reserved	Reserved for future use by the MIPI Alliance
5:3	**Clock to Data Turnaround Time (T_{SCO})**	0: ≤ 8 ns (default value) 1: ≤ 9 ns 2: ≤ 10 ns 3: ≤ 11 ns 4: ≤ 12 ns 5-6: Reserved for future use by the MIPI Alliance 7: t_{SCO} is > 12 ns, and is reported by private agreement
2:0	**Maximum Sustained Data Rate for non-CCC messages sent by Slave Device to Master Device**	0: f_{SCL} Max (default value) 1: 8 MHz 2: 6 MHz 3: 4 MHz 4: 2 MHz 5–7: Reserved for future use by the MIPI Alliance

¹⁸⁶⁵
Table 50 maxRdTurn Format

Byte	Field	Description
0	**Least Significant Byte**	Maximum Read Turnaround Time in µSeconds.
1	**Middle Byte**	24-bit field can encode turnaround times from 0.0 seconds
2	**Most Significant Byte**	to 16 seconds.

5.1.9.3.19   Get HDR Capability (GETHDRCAP)

¹⁸⁶⁶ This section is not included in the I3C Basic Specification. To gain access to this capability, please contact
¹⁸⁶⁷ MIPI Alliance.

5.1.9.3.20   Set Exchange Timing Information (SETXTIME)

¹⁸⁶⁸ This section is not included in the I3C Basic Specification. To gain access to this capability, please contact
¹⁸⁶⁹ MIPI Alliance.

¹⁸⁷⁰ As detailed in *Section 5.1.8*, the SETXTIME CCC provides the framework for Master(s) and Slave(s) to
¹⁸⁷¹ exchange event timing information for purposes such as synchronizing controls, collecting or reconstructing
¹⁸⁷² timestamps, and specifying the timing data procedure. The SETXTIME CCC is available as both a Broadcast
¹⁸⁷³ Command and as a Direct Command.

5.1.9.3.21   Get Exchange Timing Support Information (GETXTIME)

¹⁸⁷⁴ This section is not included in the I3C Basic Specification. To gain access to this capability, please contact
¹⁸⁷⁵ MIPI Alliance.

¹⁸⁷⁶ This Directed CCC, whose use is further detailed in *Section 5.1.8*, provides the framework for the Master to
¹⁸⁷⁷ query the Exchange Timing capabilities supported by the I3C Slaves.

5.1.9.3.22   Set All Addresses to Static Address (SETAASA)

1878 This CCC (*Table 51*) allows the Master to request that all connected Slaves use their known Static Address
1879 as their Dynamic Address. This is the fastest method to assign I3C Dynamic Addresses to all Slaves with
1880 Static Addresses (see *Section 5.1.4.2*).

1881   **Table 51 SETAASA Format**

S	7'h7E	SETAASA CCC	Sr	7'h7E	Sr Continuation
Sr	/ W / ACK	/ T	P		…

5.1.10   Error Detection and Recovery Methods for SDR

1882 The error detection and recovery methods specified in this Section are provided in order to avoid fatal
1883 conditions when errors occur. A set of required methods is specified for I3C Slave Devices, and a separate
1884 set of required methods is specified for I3C Master Devices. Origins for all of these SDR Error Types are
1885 shown in *Figure 53* through *Figure 56*.

5.1.10.1   SDR Error Detection and Recovery Methods for I3C Slave Devices

1886 The seven Error Types summarized in *Table 52* shall be supported for all I3C Slave Devices. Each Error Type
1887 is further explained below the table.

1888   **Table 52 SDR Slave Error Types**

Error Type	Description	Error Detection Method	Error Recovery Method
S0	**Invalid Broadcast Address/W** **(=7'h7E/W)** or **Dynamic Address/RW after DA assignment**	Detect any of the following: 7'h3E / W 7'h5E / W 7'h6E / W 7'h76 / W 7'h7A / W 7'h7C / W 7'h7F / W 7'h7E / R	a. Enable HDR EXIT Detector and ignore all other patterns b. (Optional) If both SCL and SDA stay at High level for a period greater than 60 µs, then enable STOP or START detector to exit from the S0/S1 error situation.
S1	**CCC Code**	Parity Check, using T-Bit	a. Enable HDR EXIT detector and neglect other patterns. b. (Optional) If both SCL/SDA stay at High level for a period greater than 60 µs, then enable STOP or START detector to exit from the S0/S1 error situation.
S2	**Write Data**	Parity Check, using T-Bit	Enable STOP or Repeated START detector and neglect other patterns.
S3	**Assigned Address during Dynamic Address Arbitration**	Parity Check, using PAR Bit	Generate NACK (after PAR), then wait for another Repeated START and 7E/R to re-transmit the Provisional ID.
S4	**7'h7E/R missing after Sr during Dynamic Address Arbitration**	Detect 7'h7E/R missing after Sr during Dynamic Address Arbitration	Generate NACK (after 7'h7E/R), then enable STOP or Repeated START Detector and ignore all other patterns
S5	**Transaction after detecting CCC**	Detect illegally formatted CCC	Generate NACK (after Slave Address), then enable STOP or Repeated START Detector and ignore all other patterns
S6 (optional)	**Monitoring Error**	Slave detects (through monitoring) that transmitted Data differs from what it intended to transmit (Does not apply during Dynamic Address Arbitration)	Stop the transmission, then enable STOP or Repeated START Detector and ignore all other patterns
Note:			
1. *In the ENTDAA mode, "7'hE / R" shall be excluded from the S0 error definition.*			

5.1.10.1.1 Error Type S0

1889 If an error occurs during Broadcast Address/W or Dynamic Address/RnW after a Dynamic Address is
1890 assigned, then the Slave will be unable to distinguish whether the transfer is a CCC transfer or a Private RnW
1891 transfer. For example, in the case of an ENTHDR CCC transfer the Slave is not able to know that the I3C
1892 Bus has changed to HDR Mode. A potentially fatal situation could ensue if this case is not detected and
1893 handled, because the Slave might become confused by seeing many STARTs and STOPs as illustrated in
1894 *Figure 24*, and might attempt to interpret the HDR transfer as though the I3C Bus were still in SDR Mode.
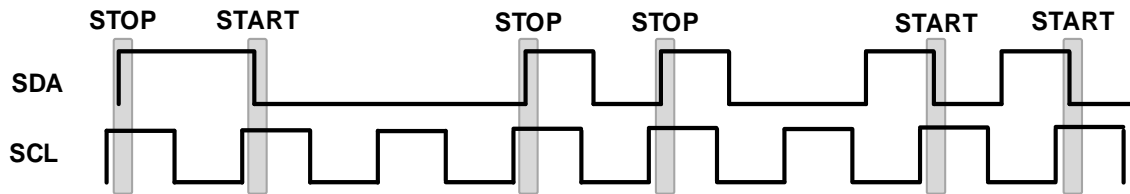
1895
1896 **Figure 24 Example Waveform for Error Type S0**

1897 In order to avoid this situation, the Master shall not use any of the possible error case Addresses: 7'h7F,
1898 7'h7C, 7'h7A, 7'h76, 7'h6E, 7'h5E and 7'h3E. Except during a Dynamic Address Arbitration procedure, the
1899 Slave shall consider receipt of any of these restricted Addresses, or the receipt of 7'h7E/R, as an error and
1900 shall then ignore the rest of the signal until the HDR EXIT pattern is detected.

5.1.10.1.2 Error Type S1

1901 If the Slave detects a parity error during a CCC code, then the Slave will not be able to know that the I3C
1902 Bus has changed to HDR Mode if the CCC is ENTHDR. This is similar to the situation in Error Type S0. In
1903 order to avoid this situation, if the Slave detects a parity error during a CCC code, then the Slave shall ignore
1904 the rest of the signal, until the HDR EXIT pattern is detected.

5.1.10.1.3 Error Type S2

1905 If the Slave detects a parity error during Write Data, then the Slave shall wait for STOP or Repeated START.

1906 If the Slave detects this error after receiving a CCC, then the Slave shall either:

1907 1. Retain the CCC state until the Slave detects the end of CCC command (i.e., when the Slave is
1908   recovered by the Repeated START condition), or else

1909 2. Stop the CCC when the Slave is recovered by the STOP condition.

5.1.10.1.4 Error Type S3

1910 If the Slave detects a parity error in the PAR Bit of the Assigned Address during a Dynamic Address
1911 Arbitration procedure, then the Slave shall generate NACK (after PAR) and then wait for another Repeated
1912 START and 7E/R to re-transmit the Provisional ID.

5.1.10.1.5 Error Type S4

1913 During a Dynamic Address Arbitration procedure, if the Slave detects any value other than 7'h7E/R following
1914 Repeated START, then the Slave shall generate NACK (after 7'h7E/R) and then wait for STOP to exit the
1915 ENTDAA mode.

1916 If the Slave detects this error after receiving a CCC, then the Slave shall either:

1917 1. Retain the CCC state until the Slave detects the end of CCC command (i.e., when the Slave is
1918   recovered by the Repeated START condition), or else

1919 2. Stop the CCC when the Slave is recovered by the STOP condition.

5.1.10.1.6   Error Type S5

1920 If the Slave detects an illegally formatted CCC, then the Slave shall generate NACK (after the Dynamic
1921 Address) and then wait for STOP or Repeated START. An example of an illegally formatted CCC would be
1922 if the Slave receives the Dynamic Address/Write during the GETBCR CCC.

1923 If the Slave detects this error after receiving a CCC, then the Slave shall either:

1924 1. Retain the CCC state until the Slave detects the end of CCC command (i.e., when the Slave is
1925   recovered by the Repeated START condition), or else
1926 2. Stop the CCC when the Slave is recovered by the STOP condition.

5.1.10.1.7   Error Type S6 (Optional)

1927 If an error occurs in a RnW Bit in a Private Read transfer, then the Write Data from the Master might conflict
1928 with the Read Data from the Slave.

1929 The Slave should always monitor the Data it transmits. If the Slave does so, then if the monitored Data differs
1930 from the Data the Slave intended to transmit (except for Data transferred during a Dynamic Address
1931 Arbitration procedure), the Slave shall consider that to be an error. If the Slave detects this error, it shall stop
1932 the transmission and then wait for STOP or Repeated START.

1933 If the Slave detects this error after receiving a CCC, then the Slave shall either:

1934 1. Retain the CCC state until the Slave detects the end of CCC command (i.e., when the Slave is
1935   recovered by the Repeated START condition), or else
1936 2. Stop the CCC when the Slave is recovered by the STOP condition.

5.1.10.1.8   Optional Recovery Method for Error Types S0 and S1

1937 An I3C Slave can recover from an Error Type S0 or Error Type S1 situation not only by detecting the HDR
1938 EXIT pattern, but also by monitoring the SCL and SDA lines. If the Slave detects that both lines stay at High
1939 level for a period exceeding 60 µs, then the I3C Slave can regard the bus as operating in non-HDR mode.
1940 The I3C Slave can then recover from the S0 or S1 situation, and wait for a STOP or a START condition to
1941 resume normal operation.

1942 ***Note:***

1943 *Regarding timing, HDR's slowest clock rate is 10 kHz (100 µs total cycle). The period 60 µs is derived*
1944 *by assuming that HDR (like DDR and TSP) will always keep an approximately even duty cycle,*
1945 *especially at very slow clock rates (since the only reason to go so slow is for long lines and/or large*
1946 *capacitive load on bus lines). 60 µs represents 60% of the duty cycle and therefore is a safe duration*
1947 *to wait when seeing both lines High.*

1948 *The Slave can start measuring the period whenever it detects that both SCL and SDA are at High*
1949 *level. There is no need to start timing this period at any particular signal pattern (for example, at the*
1950 *STOP condition).*

5.1.10.2   SDR Error Detection and Recovery Methods for I3C Master Devices

1951 The two Error Types summarized in *Table 53* shall be supported for all I3C Master Devices. Each Error Type
1952 is further explained below the table.

1953   **Table 53 SDR Master Error Types**

Error Type	Description	Error Detection Method	Error Recovery Method
M0	**Transaction after sending CCC**	Detect illegally formatted CCC	Stop the transmission, then send STOP and retry the transmission.
M1 (optional)	**Monitoring Error**	Master detects (through monitoring) transmitted Data different from what it intended to transmit (Does not apply during Dynamic Address Arbitration)	Stop the transmission, then send STOP and retry the transmission.
M2	**No response to Broadcast Address** (7'h7E)	Master detects NACK after Broadcast Address (7'h7E) transmission	Upon detection of NACK, Master transmits HDR Exit Pattern followed by STOP

5.1.10.2.1   Error Type M0

1954 If the Master detects an illegally formatted CCC, then the Master shall stop the transmission, send STOP, and
1955 retry the transmission. An example of an illegally formatted CCC would be the Master receiving just one
1956 byte from the Slave in a GETMWL CCC code, since the Master expects two bytes.

5.1.10.2.2   Error Type M1 (Optional)

1957   ***Note:***

1958   *Although this section describes methods to mitigate or handle the Type M1 error, this error is not an*
1959   *expected behavior.*

1960   Error Type M1 occurs when the I3C Master detects that the transmitted data differs from what the Master
1961   intended to transmit. This might happen when the I3C Master or I3C Slaves misinterpret the RnW bit or the
1962   ACK/NACK during transactions (including IBI) defined by the I3C specification. This Section describes
1963   only two kinds of occurrence conditions for Error Type M1, and the recovery method for each kind. Note
1964   that Type M1 errors are not an expected behavior.

1965   If an error occurs in a RnW Bit in a Private Write transfer, then the Write Data from the Master might conflict
1966   with the Read Data from the Slave. For example, the Slave could misinterpret a Private Write transfer as a
1967   Read transfer if there is an error in the RnW Bit, and as a result Write Data from the Master would conflict
1968   with Read Data from the Slave.

1969   The Master should always monitor the Data it transmits. If the Master does so, then if the monitored Data
1970   differs from the Data the Master intended to transmit (except for Data transferred during a Dynamic Address
1971   Arbitration procedure), the Master shall consider that to be an error. If the Master detects this error, it shall
1972   stop the transmission, then send STOP and retry the transmission.

1973   If the Slave does not reply, then the Master shall transmit a pattern including HDR EXIT and STOP. This
1974   requirement applies regardless of whether the I3C Bus is in SDR Mode or in HDR Mode at the time.

1975   **Example:**

1976   The Master should try the following sequence of steps when attempting to recover the Slave:

1977   1.   The Master transmits the Private Read several times.

1978   • If the Slave responds (including ACK), then the Slave temporarily did not have the Data
1979   needed for a response.

1980   • If there is no response from the Slave, then proceed to the next step.

1981   2.   The Master transmits a pattern including HDR EXIT and STOP.

1982   • If the Slave responds (including ACK), then the Slave either detected an illegal format, or
1983   detected an error.

1984   • If there is no response from the Slave, then no Slave with that Address is present on the
1985   I3C Bus.

5.1.10.2.3   Error Type M2

1986   If the Master does not receive an ACK of a transmitted Broadcast Address (7'h7E), then it shall transmit the
1987   HDR Exit Pattern followed by STOP in order to recover any Slave after S0, S1, S2, S5, and S6 errors.

5.1.10.2.4   Master Error Detection and Escalation Handling

If the Master does not receive an ACK of a transmitted private Message to a Slave, and if the following conditions are all true:

- Activity State is 0 (and has been)
- Either the Slave Device has not indicated read-turnaround delays via GETMXDS, or the Slave Device has indicated a GETMXDS period and a period longer than GETMXDS has elapsed
- The Slave Device has not notified the Master that it will be going into a lower Activity State via a private contract In-Band Interrupt (and either GETSTATUS or a private activity state status),

then the Master has the following escalation options to recover the system:

1. If the Master is aware that the Slave might sometimes need extra time, then the Master can choose to try again after a short delay.
2. If that fails, then the Master shall check whether the Slave is responsive by issuing GETSTATUS to the Slave. The idea here is that the Slave might be forced to NACK a private Message due to its inner system being unready, whereas GETSTATUS is a lightweight request that does not necessarily involve the inner system.
3. If that fails, then the Master shall use M2 error handling (see *Section 5.1.10.2.3*) to emit the sequence:

<p align="center">S | 7'h7E (W) | ACK/NACK* | HDR Exit Pattern | STOP</p>

   Note:

   * The I3C Master doesn't care whether the I3C Slaves' response is ACK vs. NACK.

4. If that fails, and if the Slave is still not responsive, then the next step depends on the value of the BCR "Offline-Capable" bit (bit [3]):
   a. If the Slave is not Offline-Capable, then:
      i. The Master can try slowing the SCL clock rate, or try slowing its effective rate by using duty cycle.
      ii. If that fails, then any further escalation is outside the scope of the I3C Specification.
   b. If the Slave is Offline-capable, then:
      i. If the Slave is known (i.e., via a private contract) to have a long wake period, and also known to monitor the I3C Bus during that wake period, then the Master simply delays and tries later, after a delay based on the known or expected wake up time. Escalation and the GETSTATUS ensure that the Slave's Dynamic Address was issued; that should serve as the wake trigger. Either the Slave may issue an In-Band Interrupt at a later time to notify the Master that it is ready, or else the Master may initiate the retry.
      ii. If Slave is not known to have a long wake period and to always monitor the I3C Bus, then the Master marks the Slave as offline. The Master may then either retain the Slave's Dynamic Address for re-use, or else discard it.

      The next action will be for the Slave to issue a Hot-Join request, in order to be re-joined to the I3C Bus. In the Hot-Join operation the Master will assign the Slave a Dynamic Address; the new Address may be either the same Address that the Slave used before being marked as offline, or a different Address.

This page intentionally left blank.

5.2   High Data Rate (HDR) Modes

I3C Basic does not support the HDR Modes specified in the I3C Specification version 1.x *[MIPI02]*, but per *Table 3*, I3C Basic Devices must detect the HDR Enter, HDR Exit, and HDR Restart patterns in order to be compatible with I3C v1.x Devices. I3C v1.x's HDR Modes are designed to transfer more data at the same Bus frequency.

Note:

> *It is important to note that the I3C Bus is always initialized and configured in SDR Mode, never in any of the HDR Modes.*

I3C v1.x's HDR Modes have Bus-wide effect. That is, the whole I3C Bus can be put into a given HDR Mode, and once entered that HDR Mode shall remain in effect until the end of that transaction.

An HDR Mode period on the I3C Bus involves five steps:

1. The I3C v1.x Master Broadcasts an Enter HDR Mode CCC, indicating which particular HDR Mode to enter. (See the Command Codes **Enter HDR Mode 1** through **Enter HDR Mode 8 [ENTHDR1–ENTHDR8]** in *Section 5.1.9.3.9*).

2. The I3C v1.x Devices on the Bus switch to the requested HDR Mode.

3. The I3C v1.x Master issues an HDR Command, followed by HDR data sent by the I3C v1.x Master or I3C v1.x Slave Device.

4. An HDR Restart Pattern or HDR Exit Pattern (see *Section 5.2.1*) is sent.
   - If an HDR Restart Pattern, then a new HDR Command is sent.

5. An I3C STOP, which ends with the Bus Free Condition.

5.2.1   HDR Exit Pattern and HDR Restart Pattern

Once an HDR Mode is entered, the HDR Exit Pattern is used to leave it, always exiting back to SDR Mode. The same HDR Exit Pattern is used to exit all HDR Protocols; that Pattern does not appear in any HDR Protocol's normal data or command flow. All I3C Slaves shall detect and respond to the HDR Exit Pattern, irrespective of whether the Slave supports any particular HDR Mode. The HDR Exit Pattern Detector is specified in *Section 5.2.1.3*.

As an alternative to the HDR Exit Pattern, the HDR Restart Pattern is also available. The HDR Restart Pattern allows multiple Messages to be sent while in HDR Mode, without forcing intervening exits to SDR Mode. That is: While the I3C Bus is in a given HDR Mode, an HDR Command can be sent to or from a Slave, and then the HDR Restart Pattern can be used to immediately send another HDR Command to or from the Slave (or a different Slave), without needing to exit the current HDR Mode between the HDR Commands. All I3C Slaves shall detect and respond to the HDR Restart Pattern while operating in any HDR Mode that the Slave supports. The HDR Restart Pattern Detector is specified in *Section 5.2.1.4*. Note that unlike the HDR Exit Pattern, the HDR Restart Pattern is only detected by I3C Slaves that support the current HDR Mode.

5.2.1.1   HDR Exit Pattern

2059   The HDR Exit Pattern is defined thus:

2060   • SDA starts High, SCL starts Low

2061   • SDA falls (from High to Low) 4 times, while SCL remains Low (for the whole time)

2062   • Each SDA transition is separated by a time interval of at least t_{DIG_H} (see *Section 6.2*)

2063   • At the end of the HDR Exit Pattern, first SCL rises and then SDA rises. This is a normal I3C STOP.

2064   *Figure 25* illustrates an HDR Exit Pattern in the upper diagram (with edges to set SCL and SDA up into
2065   correct state), and an HDR Exit Pattern with appended I3C STOP in the lower diagram (SCL being High
2066   while SDA rises).
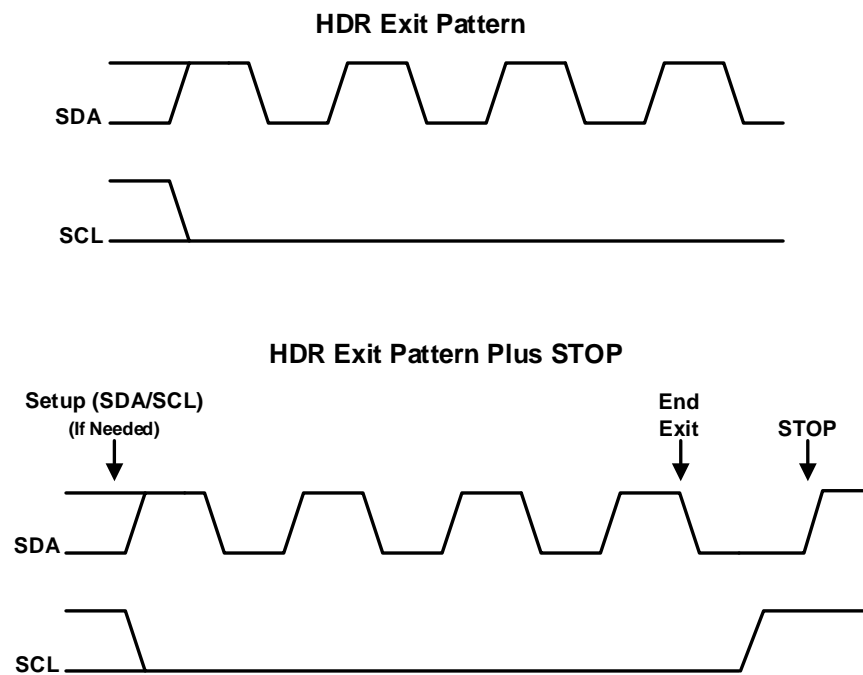
2067
2068   **Figure 25 HDR Exit Pattern and Exit Plus STOP**

Copyright © 2016-2018 MIPI Alliance, Inc.
All rights reserved.
Confidential

5.2.1.2   HDR Restart Pattern

2069   The HDR Restart Pattern is based on a subset of the HDR Exit Pattern. It is defined thus:

2070   • SDA starts High, SCL starts Low (same as HDR Exit Pattern)

2071   • SDA toggles 4 times (fall, rise, fall, rise)

2072   • The next edge is SCL rising. SDA may change with SCL rising, but SCL shall rise.

2073   *Figure 26* illustrates an HDR Restart Pattern (with edges to set SCL and SDA up into correct state) along
2074   with the necessary SCL ending edge.

2075
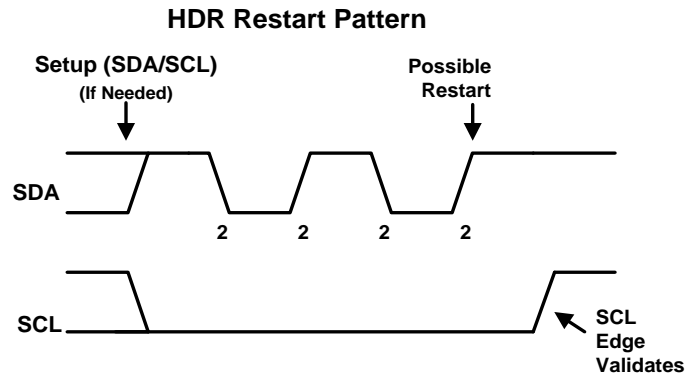
2076   **Figure 26 HDR Restart with Next Edge**

5.2.1.3 HDR Exit Pattern Detector

2077 All I3C Slaves shall include an HDR Exit Pattern Detector. The HDR Exit Pattern Detector shall be enabled
2078 only after an HDR Mode is entered, and shall be disabled after an HDR Exit pattern is detected. The HDR
2079 Exit Pattern Detector may be implemented either in digital logic, or in software (bit banged).

2080 The remainder of this Section presents an example digital logic implementation in both schematic view and
2081 in RTL code.

2082 The basic logic model is that SCL is held Low (0), and so SCL High (1) shall reset the detector. It also only
2083 uses falling edges of SDA, hence treats SDA as a clock. The HDR Exit Pattern Detector schematic is shown
2084 in *Figure 27*.

2085
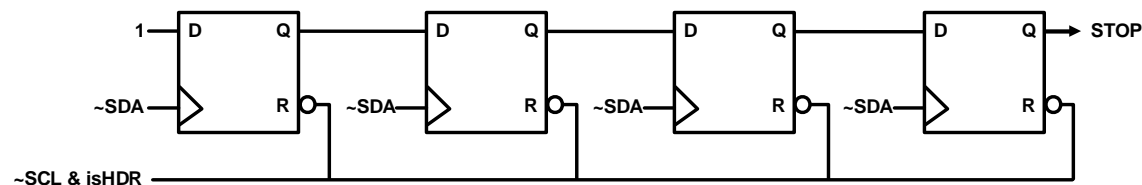
2086 **Figure 27 Example HDR Exit Pattern Detector (Schematic)**

2087 The Detector uses an inverted version of SDA as a clock (so positive edge logic, but can use negative edge
2088 logic), and is reset when SCL is High (1), or when the block is not in HDR Mode. The asynchronous nature
2089 of the reset makes this safe. This is shown in *Figure 28*. Due to the nature of SCL vs. SDA changing at the
2090 same time with HDR Modes, the Bus Slave may see SCL change after on SDA and before the next. If the
2091 HDR Exit Pattern Detector were only using clocking logic, then it would not see any change at all (SDA
2092 posedge would always see SCL Low in this example). Because the Detector uses an asynchronous reset on
2093 SCL, a change in SCL will impact the counter, even in case (b) above. Note that SCL and SDA will still be
2094 approximately 50ns between changes. So, as shown, if SCL rises at any time, then the HDR Exit Pattern
2095 Detector shall be reset, and therefore will not mistakenly signal a false Exit.

2096
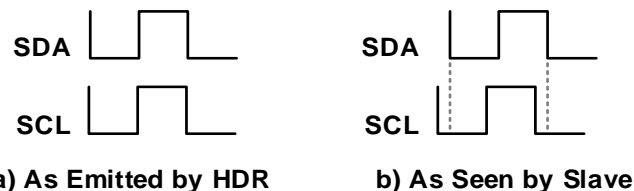
2097 **Figure 28 Metastable Changes on SCL and SDA Do Not Break the Exit Pattern Detector**

2098 An example RTL implementation of the above schematic is shown in *Listing 1*.

2099   **Listing 1 Example RTL Code for HDR Exit Pattern Generator**

```
2100    wire          scl_rst_n;
2101    wire          SDA_clk_n;       // ~SDA as clock
2102    reg[3:0]      stp_cnt;         // HDR STOP counter (shift chain)
2103
2104    assign scl_rst_n = ~iSCL & iIsInHDR; // SCL=1 resets
2105    // next uses glitch free XOR for SDA clock. Only inverts
2106    // SDA as clock if not in scan. Could add a clock mux
2107    // so uses one common clock in scan.
2108    SAFE_CLK_XOR sda_neg_clk(iSDA, ~scan_mode, SDA_clk_n);
2109
2110    // counter for 3 and 4 rising SDA when SCL is High
2111    // (else SCL resets). Whole thing held in reset if
2112    // not in HDR Mode
2113    always @ (posedge SDA_clk_n or negedge scl_rst_n)
2114      if (~scl_rst_n)
2115        stp_cnt <= 4'd0;            // SCL High or not HDR
2116      else
2117        stp_cnt <= {stp_cnt[2:0], 1'b1}; // shift chain counter
2118
2119    assign oHDR_STOP = stp_cnt[3]; // detects Exit (STOP)
```

5.2.1.4   HDR Restart and Exit Pattern Detector

2120   Any I3C Slave that supports at least one HDR Mode shall include HDR Restart Pattern detection. While this
2121   function is easily incorporated into the required HDR Exit Pattern Detector, or may be part of HDR Mode
2122   support, the particular design is not mandated by this Specification (i.e. is up to the manufacturer). The HDR
2123   Restart Pattern Detector may be implemented either in digital logic or in software (bit banged).

2124   The remainder of this Section presents an example digital logic implementation in both schematic view and
2125   in RTL code, building upon the HDR Exit Pattern Detector presented in ***Section 5.2.1.3***.

2126   The basic logic model is that SCL is held Low (0) and so SCL High (1) will reset the main detector. It also
2127   uses falling edges of SDA primarily, hence treats SDA as a clock. The HDR Restart is then detected only
**2128**   when two falling edges have been seen, and verifies the rising edge and then SCL change that is required for
2129   an HDR Restart.

2130   The combined HDR Restart and Exit Pattern Detector schematic is shown in ***Figure 29***.

2131

2132   **Figure 29 Combined HDR Restart and Exit Pattern Detector (Schematic)**
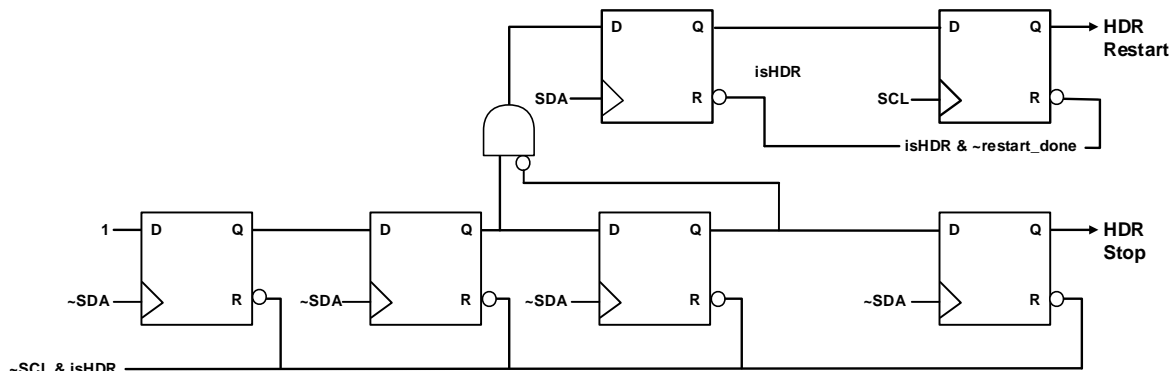
2133   This Detector design builds on the HDR Exit Pattern Detector. After exactly two SDA falling edges are seen,
2134   followed by a rising edge, the HDR Restart Pattern Detector checks for the rising edge of SCL. It does not
2135   matter whether SDA also falls at the same time; the rising SCL is the key. This works because even if SDA

2136 were falling (and therefore triggering the next flop in the Exit Pattern Detector), the upper-left flop will still
2137 hold 1 because it has not yet seen a rising edge on SDA.

2138 An example RTL implementation of the above schematic is shown in *Listing 2*.

2139   **Listing 2 Example RTL Code for Combined HDR Pattern Detector: Exit and Reset**

```
2140    wire          scl_rst_n;
2141    wire          restart_rst_n;
2142    wire          SDA_clk_n;       // ~SDA as clock
2143    reg[3:0]      stp_cnt;         // HDR STOP counter (shift chain)
2144    reg           poss_restart;    // possible restart
2145    reg           is_restart;
2146
2147    assign scl_rst_n     = ~iSCL & iIsInHDR; // SCL=1 resets
2148    assign restart_rst_n = ~iRestartDone & iIsInHDR;
2149
2150    // next uses glitch free XOR for SDA clock. Only inverts
2151    // SDA as clock if not in scan. Could add a clock mux
2152    // so uses one common clock in scan.
2153    SAFE_CLK_XOR sda_neg_clk(iSDA, ~scan_mode, SDA_clk_n);
2154
2155    // counter for 3 and 4 rising SDA when SCL is High
2156    // (else SCL resets). Whole thing held in reset if
2157    // not in HDR Mode
2158    always @ (posedge SDA_clk_n or negedge scl_rst_n)
2159      if (~scl_rst_n)
2160        stp_cnt <= 4'd0;           // SCL High or not HDR
2161      else
2162        stp_cnt <= {stp_cnt[2:0], 1'b1}; // shift chain counter
2163
2164    assign oHDR_STOP = stp_cnt[3];// detects Exit (STOP)
2165
2166    // Possible Restart means exactly 2 falling edges
2167    // of SDA and then rising edge of SDA. Actual
2168    // Restart is from SCL then rising
2169    always @ (posedge iSDA or negedge restart_rst_n)
2170      if (~restart_rst_n)
2171        poss_restart <= 1'b0;      // Restart ACK or not HDR
2172      else if (stp_cnt[1] & ~stp_cnt[2]) // after 2nd fall only
2173        poss_restart <= 1'b1;      // SDA rise after 2 falls
2174      else
2175        poss_restart <= 1'b0;      // else not possible restart
2176
2177    always @ (posedge SCL or negedge restart_rst_n)
2178      if (~restart_rst_n)
2179        is_restart <= 1'b0;        // Restart ACK or not HDR
2180      else if (poss_restart)
2181        is_restart <= 1'b1;        // SCL rises after SDA does
2182      else
2183        is_restart <= 1'b0;        // else not restart
```

5.2.1.5   Compatibility of HDR Pattern Detection and Ternary Modes

2184 This section is not included in the I3C Basic Specification. To gain access to this capability, please contact
2185 MIPI Alliance.

5.2.2　　　HDR Double Data Rate Mode (HDR-DDR)

2186　This section is not included in the I3C Basic Specification. To gain access to this capability, please contact
2187　MIPI Alliance.

2188　Like SDR Mode, HDR-DDR Mode uses SCL as a clock; however unlike SDR, Data and Commands change
2189　SDA on both SCL edges (when High and when Low), effectively doubling the data rate. By contrast, in SDR
2190　Mode SDA is changed only when SCL is Low. Since SDR Mode defines it as a START or a STOP for SDA
2191　to change while SCL remains High, HDR-DDR Mode is classified as an HDR Mode in order to prevent
2192　confusion.

5.2.2.1　　　HDR-DDR Overview

2193　This section is not included in the I3C Basic Specification. To gain access to this capability, please contact
2194　MIPI Alliance.

5.2.2.2　　　HDR-DDR Command Coding

2195　This section is not included in the I3C Basic Specification. To gain access to this capability, please contact
2196　MIPI Alliance.

5.2.2.3　　　HDR-DDR Bus Turnaround

2197　This section is not included in the I3C Basic Specification. To gain access to this capability, please contact
2198　MIPI Alliance.

5.2.2.3.1　　　Command to Read Data from Slave

2199　This section is not included in the I3C Basic Specification. To gain access to this capability, please contact
2200　MIPI Alliance.

5.2.2.3.2　　　End of a Read Command Message

2201　This section is not included in the I3C Basic Specification. To gain access to this capability, please contact
2202　MIPI Alliance.

5.2.2.3.3　　　Master Termination of a Read Command Message

2203　This section is not included in the I3C Basic Specification. To gain access to this capability, please contact
2204　MIPI Alliance.

5.2.2.4　　　HDR-DDR Error Detection

2205　This section is not included in the I3C Basic Specification. To gain access to this capability, please contact
2206　MIPI Alliance.

5.2.2.5　　　HDR-DDR CRC5 Algorithm

2207　This section is not included in the I3C Basic Specification. To gain access to this capability, please contact
2208　MIPI Alliance.

5.2.3   HDR Ternary Modes (HDR-TSP and HDR-TSL)

2209 This section is not included in the I3C Basic Specification. To gain access to this capability, please contact
2210 MIPI Alliance.

2211 I3C defines two HDR Modes that use Ternary Coding:

2212   • HDR-TSP: Ternary Symbol for Pure Bus (no I^{2}C Devices)

2213   • HDR-TSL: Ternary Symbol Legacy-inclusive-Bus

2214 These HDR Ternary Modes are entered in the standard way (see *Section 5.1.9.3.9*), followed by a Command
2215 and then zero or more Data Words. In HDR Ternary Modes, Commands shall be issued only by a Master.
2216 Data Words may be issued by a Master or by a Slave, depending on the particular Command (Write or Read).

5.2.3.1   HDR Ternary Signaling and Coding Protocol

2217 This section is not included in the I3C Basic Specification. To gain access to this capability, please contact
2218 MIPI Alliance.

5.2.3.1.1   Ternary Signaling

2219 This section is not included in the I3C Basic Specification. To gain access to this capability, please contact
2220 MIPI Alliance.

5.2.3.1.2   Ternary Coding Protocol

2221 This section is not included in the I3C Basic Specification. To gain access to this capability, please contact
2222 MIPI Alliance.

5.2.3.2   HDR Ternary Command Coding

2223 This section is not included in the I3C Basic Specification. To gain access to this capability, please contact
2224 MIPI Alliance.

6   I3C Electrical Specifications

6.1   DC I/O Characteristics

This Section describes the DC operating parameters of the I3C interface in two modes: Push-Pull Mode and Open Drain Mode. In Push-Pull Mode, the SDA pin drives at higher speeds with a totem-pole driver.

Two important parameters should be noted for I3C:

- The I3C interface targets nominal operating voltages of 1.2V, 1.8V, and 3.3V or less. The I3C interface is not characterized for 5V systems, but could be extended to support 5V if sufficient driver strength, reduced diameter, and/or reduced speed is used in the system.

- For peak speeds the capacitance loading allowed is reduced to 50 pF, which is much lower than Legacy I^{2}C. With I3C higher capacitance busses are possible, but only at reduced speeds and with reduced features (e.g. no I^{2}C Devices are supported).

Optionally, the I3C Basic interface also targets a nominal operating voltage of 1.0V and a 100 pF capacitive loading for new usages, such as Serial Presence Detect (SPD) in DDR5.

2236

Table 54 I3C I/O Stage Characteristics Common to Push-Pull Mode and Open Drain Mode

Parameter	Symbol	Conditions	Min	Typ	Max	Unit	Notes
Operating Voltage	V_{DD}	–	1.10	1.20	1.30	V	1
			1.65	1.80	1.95		
			2.97	3.30	3.63		
Low-Level Input Voltage	V_{IL}	–	-0.3	–	$0.3 * V_{DD}$	V	
High-Level Input Voltage	V_{IH}	–	$0.7 * V_{DD}$	–	$V_{DD} + 0.3$	V	
Schmitt Trigger Inputs Hysteresis	V_{hys}	–	$0.1 * V_{DD}$	–	–	V	
Output Low Level	V_{OL}	For $V_{DD} < 1.4V$: $I_{OL} = 2$ mA	–	–	0.18	V	2
		For $V_{DD} \geq 1.4V$: $I_{OL} = 3$ mA	–	–	0.27	V	2
Input Current (per Input-Only I/O Pin)	I_i	-100 mV < V_i < V_{DD} + 100 mV for ≥ 1.8V nominal	-10	–	10	µA	2
		-100 mV < V_i < V_{DD} + 100 mV for < 1.8V nominal	-5	–	5	µA	2
Capacitance (per I/O Pin)	C_i	For < 1.8V nominal	–	–	5	pF	5
		For ≥ 1.8V nominal	–	–	10	pF	5
Capacitance Mismatch Between Pins	ΔC	Difference between SDA and SCL capacitance $C_I \leq 5pF$	–	–	1.5	pF	5
		Difference between SDA and SCL capacitance $C_I > 5pF$	–	–	3	pF	5
Push-Pull Only							
Output High Level	V_{OH}	For $V_{DD} < 1.4V$: $I_{OH} = -2$ mA	$V_{DD} - 0.18$	–	–	V	2
		For $V_{DD} \geq 1.4V$: $I_{OH} = -3$ mA	$V_{DD} - 0.27$	–	–	V	2
Legacy Mode with Pull-Up							
Pull-Up for Open Drain	R_p	t_r = Max rise time C_b = Bus Capacitance V_{DD} = 1.2V, 1.8V, or 3.3V	$\dfrac{V_{DD} - V_{OL}}{3\ mA}$ for $V_{DD} \geq 1.4V$	$\dfrac{t_r}{0.8473 * Cb}$		Ω	3, 4, 6, 7
		t_r = 120 ns C_b = 50 pF	$\dfrac{V_{DD} - V_{OL}}{2\ mA}$ for $V_{DD} < 1.4V$	2833			

Note:

1) *This Specification considered only 1.2V, 1.8V, and 3.3V, with associated tolerances. Other voltage ranges are not prohibited. Any I3C Bus implementation shall ensure the correct operation of the Devices resident on the Bus, notably the inter-compatibility of their voltage ranges.*

2) *Negative sign for currents indicates current direction.*

3) $V(t1) = 0.3 * V_{DD} = V_{DD} (1 - e^{-t1/RC})$; *then* $t1 = 0.3566749 * RC$
    $V(t2) = 0.7 * V_{DD} = V_{DD} (1 - e^{-t2/RC})$; *then* $t2 = 1.2039729 * RC$
    $T = t2 - t1 = 0.8473 * RC$

4) *Pull-Up for Open Drain shall be switched off during I3C Push-Pull operation. May be implemented as: A Pull-Up internally; A current source internally; or an external Pull-Up resistor driven by a pin.*

5) *For Devices that support both 1.8V and 3.3V, the larger capacitance may be used.*

6) *Open-Drain never occurs on SCL*

7) *A weak pull-up needs to be applied as well for Master handoff*

Table 55 I3C Low Voltage / High Capacitive Load I/O Stage Characteristics for Push-Pull Mode and Open Drain Mode

Parameter	Symbol	Conditions	Min	Typ	Max	Unit	Notes
Operating Voltage	VDD	–	1.0	1.1	1.2	V	–
Low-Level Input Voltage	VIL	–	-0.2	–	0.3 * VDD	V	–
High-Level Input Voltage	VIH	–	0.7 * VDD	–	VDD	V	1
Schmitt Trigger Inputs Hysteresis	Vhys	–	0.1 * VDD	–	0.4 * VDD	V	–
Output Low Level	VOL	IOL = 4 mA	–	–	0.25 * VDD	V	2
Input Current (per Input-Only I/O Pin)	Ii	-100 mV < Vi < VDD	-10	–	10	µA	2
Capacitance (per I/O Pin)	Ci	-	–	–	5	pF	–
Capacitance Mismatch Between Pins	ΔC	-	–	–	1.5	pF	–
Push-Pull Only							
Output High Level	VOH	IOH = -4 mA	0.75 * VDD	–	–	V	2
Legacy Mode with Pull-Up							
Pull-Up for Open Drain	Rp	tr = Max rise time (100 ns) Cb = Bus Capacitance (100 pf) VDD = 1V	$\dfrac{V_{DD} - V_{OL}}{4\ mA}$	$\dfrac{t_r}{0.8473 * Cb}$		Ω	3, 4, 6, 7, 8

Note:

    1) V_{DD} with respect to Typical V_{DD}

    2) Negative sign for currents indicates current direction.

    3) $V(t1) = 0.3 * V_{DD} = V_{DD} (1 - e^{-t1/RC})$; then $t1 = 0.3566749 * RC$

        $V(t2) = 0.7 * V_{DD} = V_{DD} (1 - e^{-t2/RC})$; then $t2 = 1.2039729 * RC$

        $T = t2 - t1 = 0.8473 * RC$

    4) Pull-Up for Open Drain shall be switched off during I3C Push-Pull operation. May be implemented as: A Pull-Up internally; A current source internally; or an external Pull-Up resistor driven by a pin.

    6) Open-Drain never occurs on SCL

    7) A weak pull-up needs to be applied as well for Master handoff

    8) **R_p Min should be decided based on the V_{IL} specification**

2238

I3C supports Legacy I²C Slaves. An important feature of an I²C Slave is the 50 ns Spike Filter on the SDA and SCL pads. If a Spike Filter is implemented on all I²C Devices present on the I3C Bus, then the I3C Bus may operate at maximum rated clock frequency. If any I²C Device does not have a Spike Filter, then the I3C Bus speed is determined by the slowest Legacy I²C Device without a Spike Filter. Other requirements of an I²C Legacy Slave are listed in *Table 56*.

Table 56 Legacy I²C Device Requirements When Operating on I3C

Feature	Required	Desirable	Not Used	Not Allowed	Notes
Fm Speed	X	–	–	–	–
Fm+ Speed	–	X	–	–	–
HS and UFm	–	–	X	–	2
Static I²C Address	X	–	–	–	–
50 ns Spike Filter	–	X	–	–	1
Clock Stretching by Slave	–	–	–	X	–
I²C Extended Address (10 bit)	–	–	X	–	2
I3C Reserved Address	–	–	–	X	–
Note:					
1) *Lack of Spike Filter will severely degrade Bus performance and eliminate certain I3C Bus features*					
2) *"Not Used" means that the I3C Master will not make use of the I²C feature, however if the Slave supports the feature, then it will not interfere with I3C Bus operation.*					

6.2   Timing Specification

2245   A key feature of I3C is to maximize the speed of data transfer and minimize the time required for low-power
2246   Devices to remain in Active Mode. In Single Data Rate Mode this is accomplished by keeping Bus
2247   capacitance low and clock speed high. For large packets of data an additional speed improvement is possible
2248   using HDR Modes, where data is transferred on every clock edge. I3C supports Legacy I^{2}C Fm and Fm+
2249   Modes. *Table 57* gives reference timing requirements used in Legacy Mode.

Table 57 I3C Timing Requirements When Communicating With I²C Legacy Devices

Parameter	Symbol	Timing Diagram	Legacy Mode 400kHz / Fm		Legacy Mode 1MHz / Fm+		Units	Notes
			Min	Max	Min	Max		
SCL Clock Frequency	f_{SCL}	–	0	0.4	0	1.0	MHz	–
Setup Time for a Repeated START	t_{SU_STA}	*Figure 30*	600	–	260	–	ns	–
Hold Time for a (Repeated) START	t_{HD_STA}	*Figure 30*	600	–	260	–	ns	–
SCL Clock Low Period	t_{LOW}	*Figure 30*	1300	–	500	–	ns	–
	t_{DIG_L}	*Figure 30* *Figure 45*	$t_{LOW} + t_{rCL}$	–	$t_{LOW} + t_{rCL}$	–	ns	–
SCL Clock High Period	t_{HIGH}	*Figure 30*	600	–	260	–	ns	–
	t_{DIG_H}	*Figure 30* *Figure 45*	$t_{HIGH} + t_{rCL}$	–	$t_{HIGH} + t_{rCL}$	–	ns	–
Data Setup Time	t_{SU_DAT}	*Figure 30*	100	–	50	–	ns	–
Data Hold Time	t_{HD_DAT}	*Figure 30*	–	–	–	–	ns	–
SCL Signal Rise Time	t_{rCL}	*Figure 30*	20	300	–	120	ns	–
SCL Signal Fall Time	t_{fCL}	*Figure 30*	20 * (V_{DD}/ 5.5V)	300	20 * (V_{DD}/ 5.5V)	120	ns	–
SDA Signal Rise Time	t_{rDA}	*Figure 30*	20	300	–	120	ns	–
SDA Signal Fall Time	t_{fDA}	*Figure 30*	20 * (V_{DD}/ 5.5V)	300	20 * (V_{DD}/ 5.5V)	120	ns	–
Setup Time for STOP	t_{SU_STO}	*Figure 30*	600	–	260	–	ns	–
Bus Free Time Between a STOP Condition and a START Condition	t_{BUF}	–	1.3	–	0.5	–	µs	–
Pulse Width of Spikes that the Spike Filter Must Suppress	t_{SPIKE}	*Figure 45*	0	50	0	50	ns	–

2250

2251 During an I3C communication the drive on the SDA pin shall have the ability to dynamically switch between
2252 Push-Pull and Open Drain.

2253   **Table 58 I3C Open Drain Timing Parameters**

Parameter	Symbol	Timing Diagram	I3C Open Drain Mode Min	I3C Open Drain Mode Max	Units	Notes
Low Period of SCL Clock	t_{LOW_OD}	*Figure 34*	200	–	ns	1, 2
	$t_{DIG\_OD\_L}$	*Figure 34*	t_{LOW_ODmin} + t_{fDA_ODmin}	–	ns	–
High Period of SCL Clock	t_{HIGH}	*Figure 31*	–	41	ns	3, 4
	t_{DIG_H}	*Figure 31* *Figure 45*	–	t_{HIGH} + t_{CF}	ns	–
Fall Time of SDA Signal	t_{fDA_OD}	*Figure 34*	t_{CF}	12	ns	–
SDA Data Setup Time During Open Drain Mode	t_{SU_OD}	*Figure 32* *Figure 34*	3	–	ns	1
Clock After START (S) Condition	t_{CAS}	*Figure 34*	38.4 nano	For ENTAS0: 1 µ For ENTAS1: 100 µ For ENTAS2: 2 milli For ENTAS3: 50 milli	seconds	5, 6
Clock Before STOP (P) Condition	t_{CBP}	*Figure 35*	t_{CASmin} / 2	–	seconds	–
Current Master to Secondary Master Overlap time during handoff	$t_{MMOverlap}$	*Figure 44*	$t_{DIG\_OD\_Lmin}$	–	ns	–
Bus Available Condition	t_{AVAL}	–	1	–	µs	7
Bus Idle Condition	t_{IDLE}	–	200	–	µs	–
Time Internal Where New Master Not Driving SDA Low	t_{MMLock}	*Figure 44*	$t_{AVALmin}$	–	us	–

Note:

1) This is approximately equal to t_{LOWmin} + t_{DS_ODmin} + t_{rDA_ODtyp} + t_{SU_ODmin}

2) The Master may use a shorter Low period if it knows that this is safe, i.e., that SDA is already above V_{IH}

3) Based on t_{SPIKE}, rise and fall times, and interconnect

4) This maximum High period may be exceeded when the signals can be safely seen by Legacy I²C Devices, and/or in consideration of the interconnect (e.g., a short Bus)

5) On a Legacy Bus where I²C Devices need to see Start, the t_{CAS} Min value is further constrained (see **Section 5.1.3.5**)

6) Slaves that do not support the optional ENTASx CCCs (see **Section 5.1.9.3.2**) shall use the t_{CAS} Max value shown for ENTAS3

7) On a Mixed Bus with Fm Legacy I²C Devices, t_{AVAL} is 300 ns shorter than the Fm Bus Free Condition time (t_{BUF})

2254

Table 59 I3C Push-Pull Timing Parameters for SDR Mode

Parameter		Symbol	Timing Diagram	Min	Typ	Max	Units	Notes
SCL Clock Frequency		f_{SCL}	–	0.01	12.5	12.9	MHz	1
SCL Clock Low Period		t_{LOW}	*Figure 30*	24	–	–	ns	–
		t_{DIG_L}	*Figure 31*	32	–	–	ns	2, 4
SCL Clock High Period for Mixed Bus		t_{HIGH_MIXED}	*Figure 31*	24	–	–	ns	–
		$t_{DIG\_H\_MIXED}$	*Figure 31*	32	–	45	ns	2, 3
SCL Clock High Period		t_{HIGH}	*Figure 30*	24	–	–	ns	–
		t_{DIG_H}	*Figure 31* *Figure 30*	32	–	–	ns	2
Clock in to Data Out for Slave		t_{SCO}	*Figure 37*	–	–	12	ns	7, 8
SCL Clock Rise Time		t_{CR}	*Figure 30*	–	–	150e06 * 1 / f_{SCL} (capped at 60)	ns	5
SCL Clock Fall Time		t_{CF}	*Figure 30*	–	–	150e06 * 1 / f_{SCL} (capped at 60)	ns	5
SDA Signal Data Hold in Push-Pull Mode	Master	t_{HD_PP}	*Figure 36*	t_{CR} + 3 and t_{CF} + 3	–	–	–	4, 6
	Slave	t_{HD_PP}	*Figure 38*	0	–	–	–	6
SDA Signal Data Setup in Push-Pull Mode		t_{SU_PP}	*Figure 36* *Figure 37*	3	–	N/A	ns	–
Clock After Repeated START (Sr)		t_{CASr}	*Figure 42*	t_{CASmin}	–	N/A	ns	–
Clock Before Repeated START (Sr)		t_{CBSr}	*Figure 42*	t_{CASmin} / 2	–	N/A	ns	–
Capacitive Load per Bus Line (SDA/SCL)		C_b	–	–	–	50	pF	–

Note:

1) $F_{SCL} = 1 / (t_{DIG\_L} + t_{DIG\_H})$

2) t_{DIG_L} and t_{DIG_H} are the clock Low and High periods as seen at the receiver end of the I3C Bus using V_{IL} and V_{IH} (see **Figure 30**)

3) When communicating with an I3C Device on a mixed Bus, the $t_{DIG\_H\_MIXED}$ period must be constrained in order to make sure that I²C Devices do not interpret I3C signaling as valid I²C signaling.

4) As both edges are used, the hold time needs to be satisfied for the respective edges; i.e., t_{CF} + 3 for falling edge clocks, and t_{CR} + 3 for rising edge clocks.

5) The clock maximum rise/fall time is capped at 60 ns. For lower frequency rise and fall the maximum value is limited at 60 ns, and is not dependent upon the clock frequency.

6) t_{HD_PP} is a Hold time parameter for Push-Pull Mode that has a different value for Master mode vs. Slave mode. In SDR Mode the Hold time parameter is referred to as t_{HD_SDR}.

7) Devices with more than 12ns of t_{SCO} delay shall set the limitation bit in the BCR, and shall support the GETMXDS CCC to allow the Master to read this value and adjust computations accordingly. For purposes of system design and test conformance, this parameter should be considered together with pad delay, bus capacitance, propagation delay, and clock triggering points.

8) Pad delay based on 90 Ω / 4 mA driver and 50 pF load. Note that Master may be a Slave in a multi-Master system, and thus shall also adhere to this requirement

2255   The timing diagram in **Figure 30** depicts I3C Legacy Mode for I²C Devices. The timing parameters
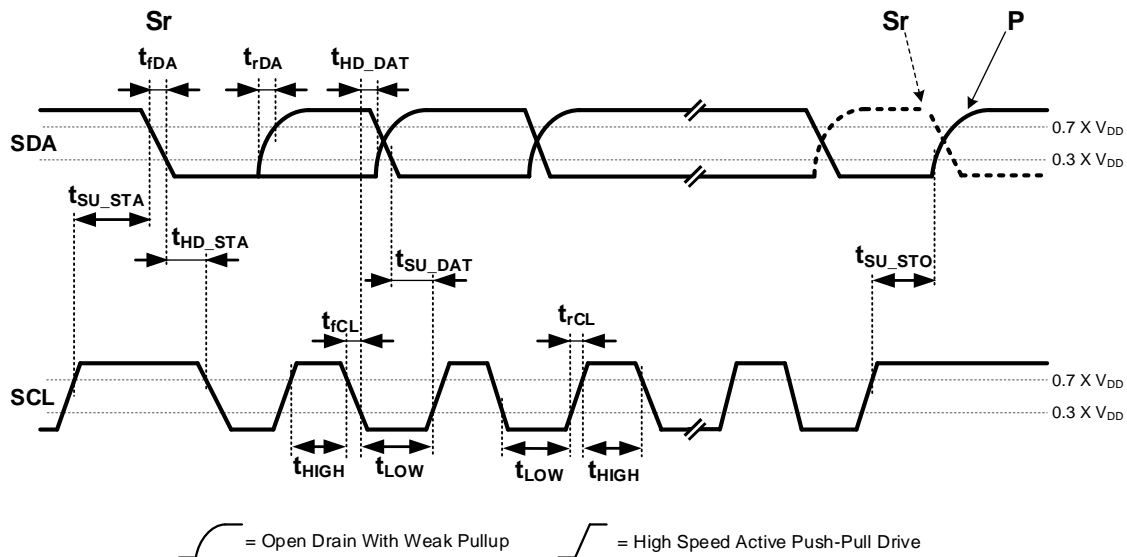2256   referenced in this diagram appear in **Table 57**.

2257

2258   **Figure 30 I3C Legacy Mode Timing**

2259
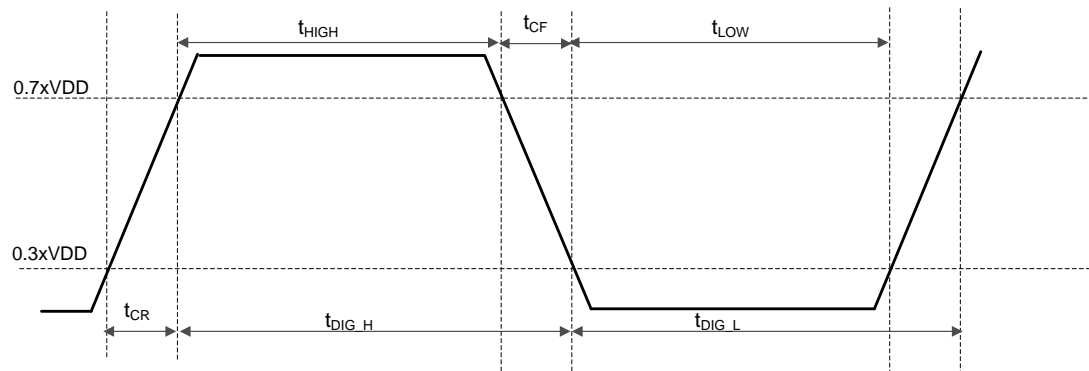
2260   **Figure 31 t$_{DIG\_H}$ and t$_{DIG\_L}$**

2261   The start of a typical I3C communication in SDR Mode is shown in **Figure 32**. The initial communication
2262   looks very similar to I²C, with additional commands that follow as described in **Section 4**. The key difference
2263   is that higher clock speed is supported, up to 12.5 MHz. The higher clock speed allows Legacy I²C Devices
2264   with 50 ns Spike Filters to ignore the communications. The Master can then run in SDR Mode for I3C
2265   Devices using full 12.5 MHz timing, though it will have to slow down in order to communicate with Legacy
2266   I²C Devices. **Figure 32** shows the beginning of a transaction, where the Slave acknowledges its address.

2267   **Figure 33** shows the possible continuation of an I3C SDR communication, in the case where the Slave does
2268   not acknowledge its address. The Master may either STOP the communication, or else continue with a
2269   Repeated START.

Figure 32 I3C Data Transfer – ACK by Slave

SDA

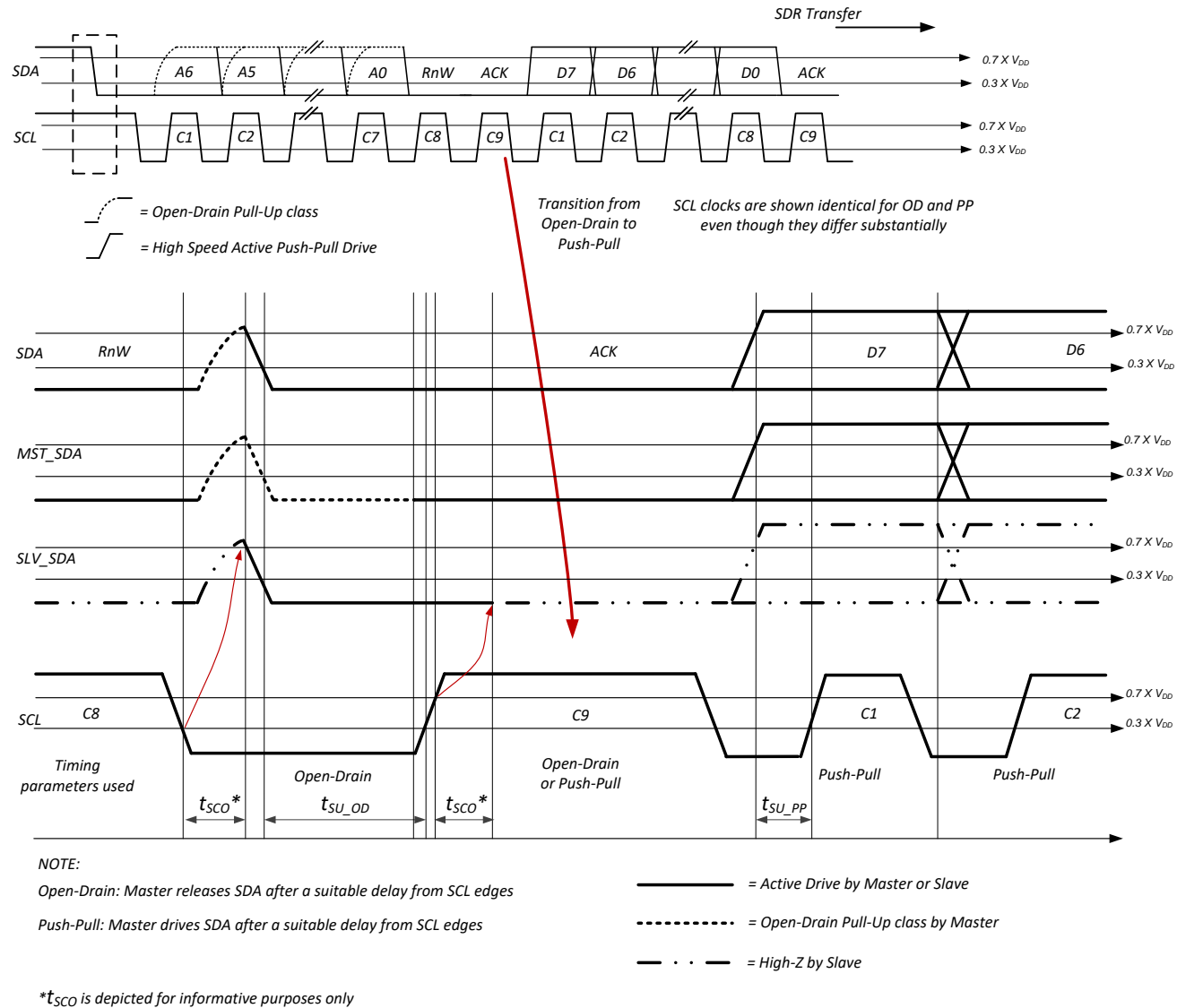A6   A5   A0   RnW   NACK   STOP
                                                        0.7 X V_{DD}
                                                        0.3 X V_{DD}
                                                        Repeated START

SCL   C9   0.7 X V_{DD}
                                                        0.3 X V_{DD}

⌐ = Open-Drain Pull-Up class

⌐ = High Speed Active Push-Pull Drive

Transition from
Open-Drain to
Push-Pull

SDA   0.7 X V_{DD}
                                                        0.3 X V_{DD}

MST_SDA   0.7 X V_{DD}
                                                        0.3 X V_{DD}

SLV_SDA   0.7 X V_{DD}
                                                        0.3 X V_{DD}

SCL   C9   0.7 X V_{DD}
                                                        0.3 X V_{DD}

Timing
parameters used   Open-Drain   Push-Pull   Push-Pull
                                               t_{SU_PP}

NOTE:

Open-Drain: Master releases SDA after a suitable delay   ——— = Active Drive by Master or Slave   ——— = Master drives Push-Pull HIGH,
from SCL edges   prepares for Repeated START

Push-Pull: Master drives SDA after a suitable delay from   ········ = Open-Drain Pull-Up class by Master   ——— = Master drives Push-Pull LOW,
SCL edges   prepares for STOP

                                                           —·· — = High-Z by Slave
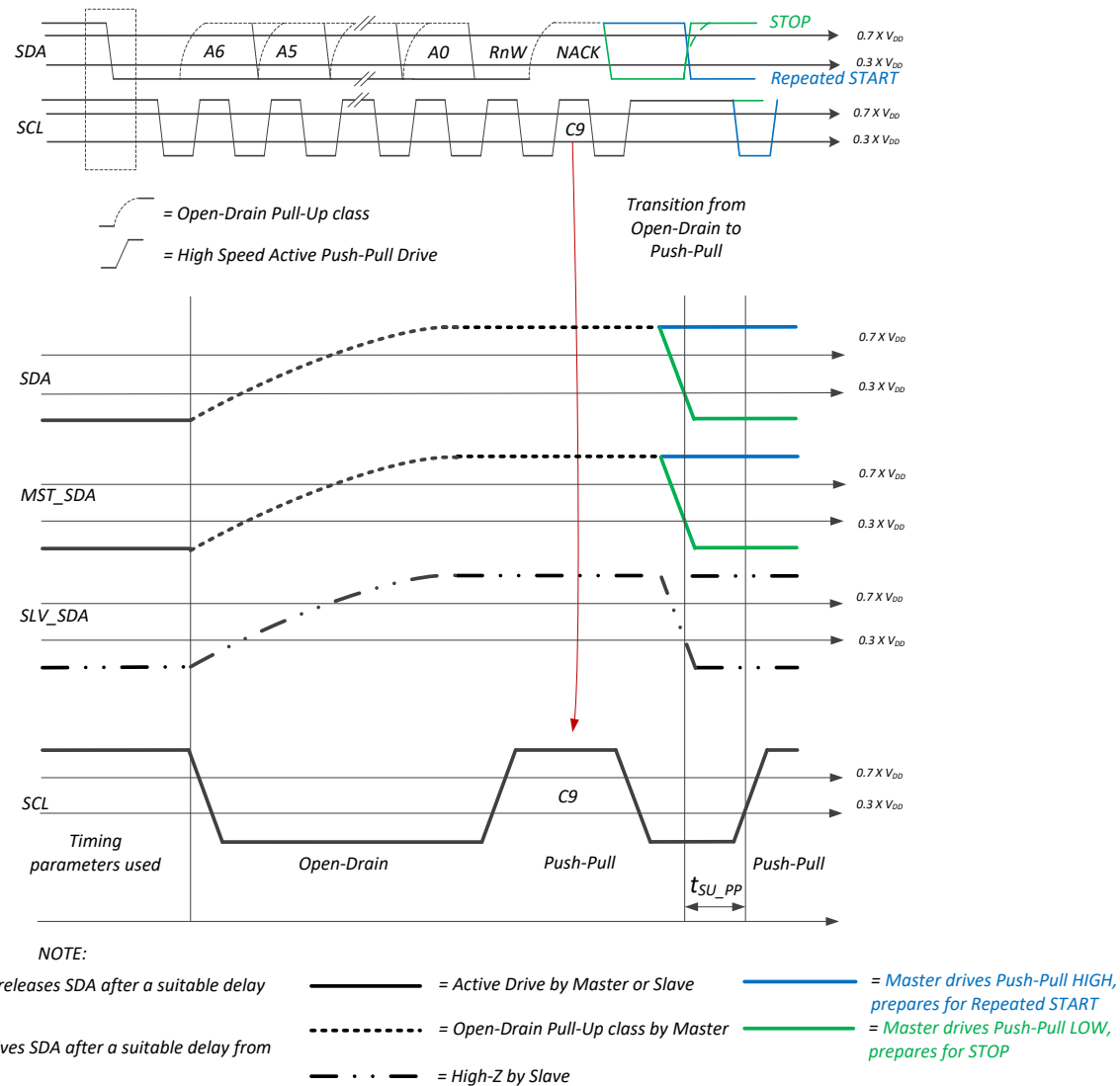
Figure 33 I3C Data Transfer – NACK by Slave

2274 *Figure 34* shows the timing parameters for an I3C START Condition, and *Figure 35* shows the timing
2275 parameters for an I3C STOP Condition. Notice for both the START and the STOP, the SDA pin is in Open
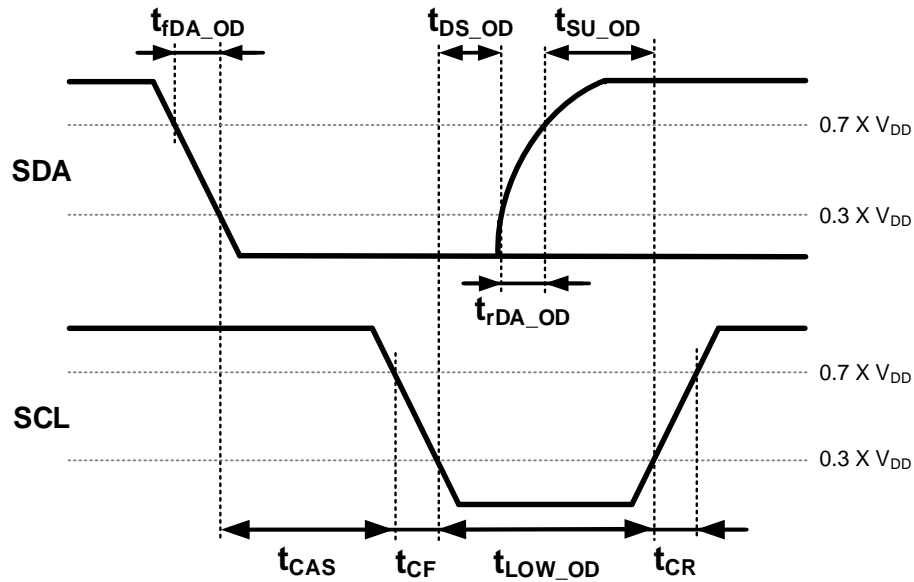**2276** Drain mode, as indicated by **t**$_{\text{DV}}$'s slow rise time.

2277

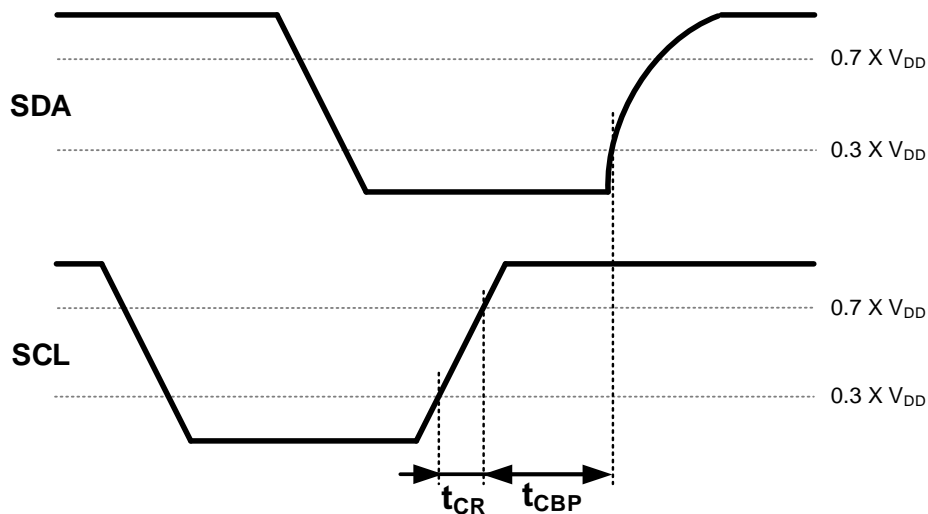**2278** **Figure 34 I3C START Condition Timing**

2279

2280 **Figure 35 I3C STOP Condition Timing**

2281 *Figure 36* and *Figure 37* illustrate the timing parameters that are unique to the Master Device and to the
2282 Slave Device, as specified in *Table 59*. The primary difference between the two is that a Master always
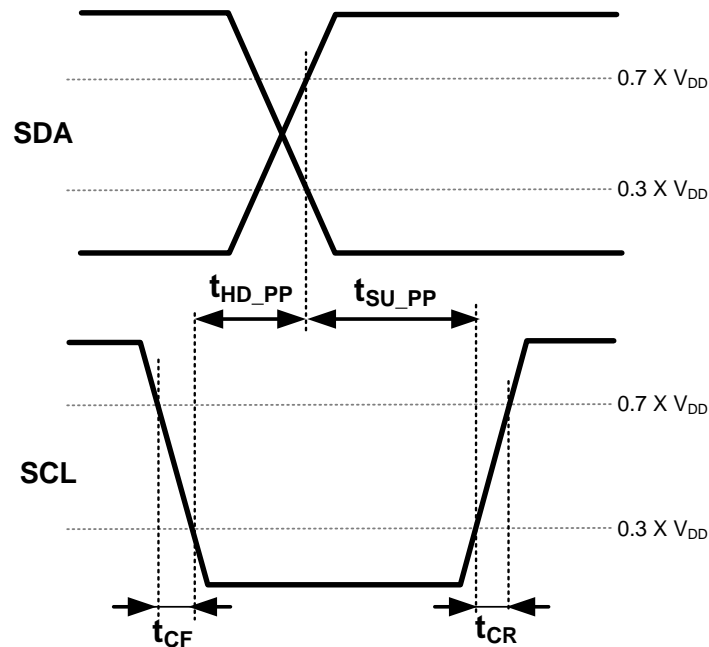2283 transmits the clock (SCL), whereas the Slave is receiver-only on the SCL pin.
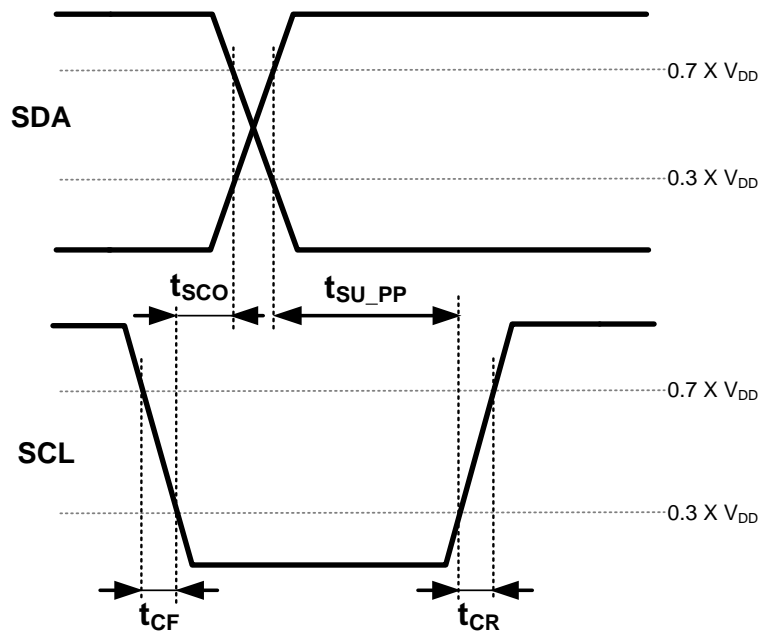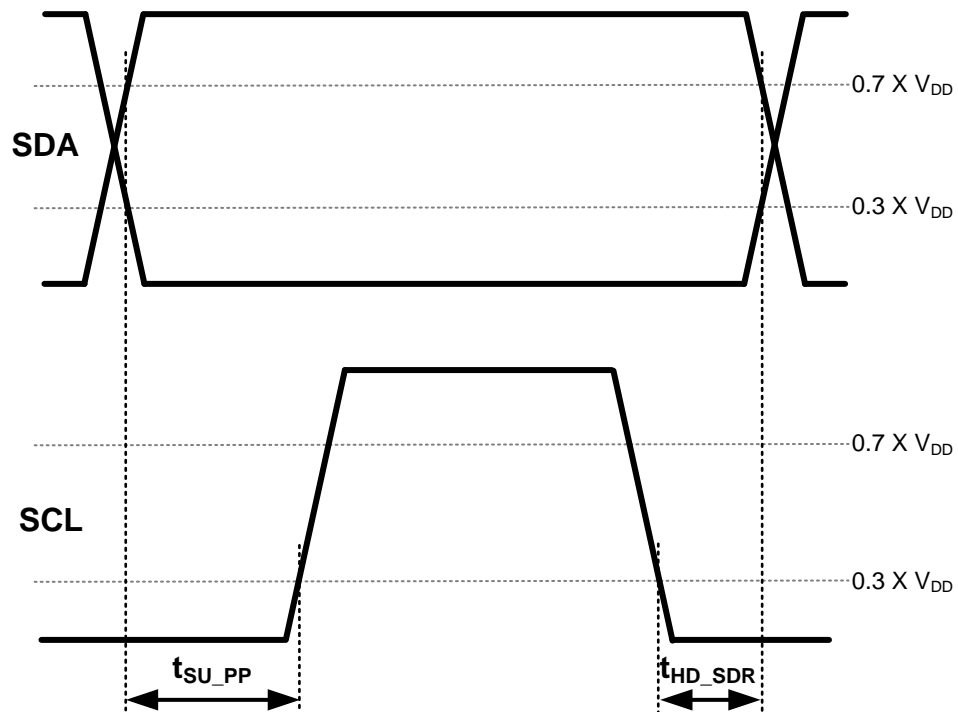
2284

2285 **Figure 36 I3C Master Out Timing**

2286

2287 **Figure 37 I3C Slave Out Timing**

Figure 38 Master SDR Timing

Figure 39 T-Bit When Slave Ends Read and Master Generates STOP

Repeated START (Sr)

SDA

D7 D6 D0 T

0.7 X V_{DD}
0.3 X V_{DD}

SCL

C1 C8 C9

0.7 X V_{DD}
0.3 X V_{DD}

= High Speed Active Push-Pull Drive

Repeated START (Sr)

SDA

0.7 X V_{DD}
0.3 X V_{DD}

MST_SDA

0.7 X V_{DD}
0.3 X V_{DD}

SLV_SDA

0.7 X V_{DD}
0.3 X V_{DD}

SCL

C9

0.7 X V_{DD}
0.3 X V_{DD}

t_{SCO} t_{SCO}

Slave terminates the Read transfer by pulling SDA Low in 9th bit and allowing Master Device to generate either STOP or RESTART

——————— = Active Drive by Master or Slave

— — — = High-Z by Master

············· = Open-Drain class Pull-Up by Master
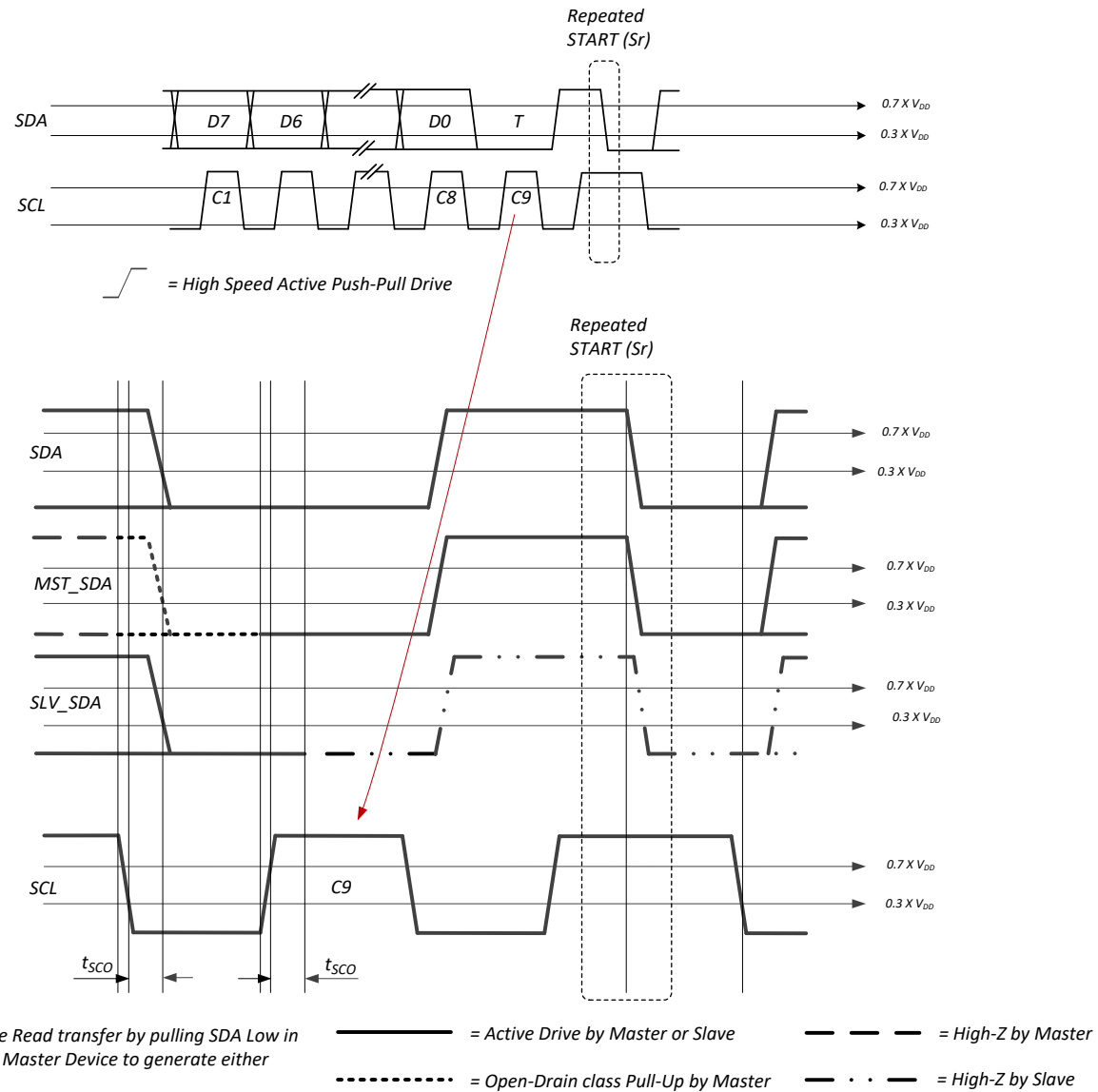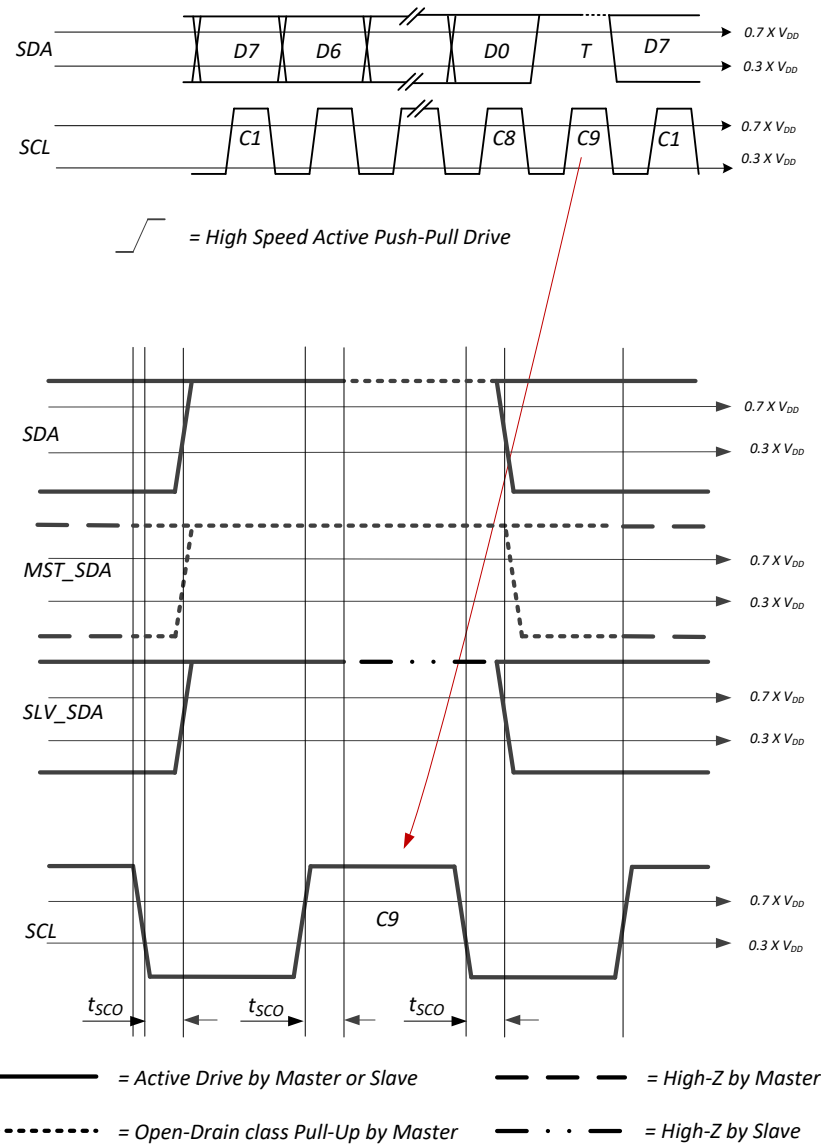
— · · — = High-Z by Slave

Figure 40 T-Bit When Slave Ends Read and Master Generates Repeated START

2292

2293

Figure 41 T-Bit When Slave and Master Agree to Continue Read Message

Figure 42 T-Bit When Master Ends Read with Repeated START and STOP
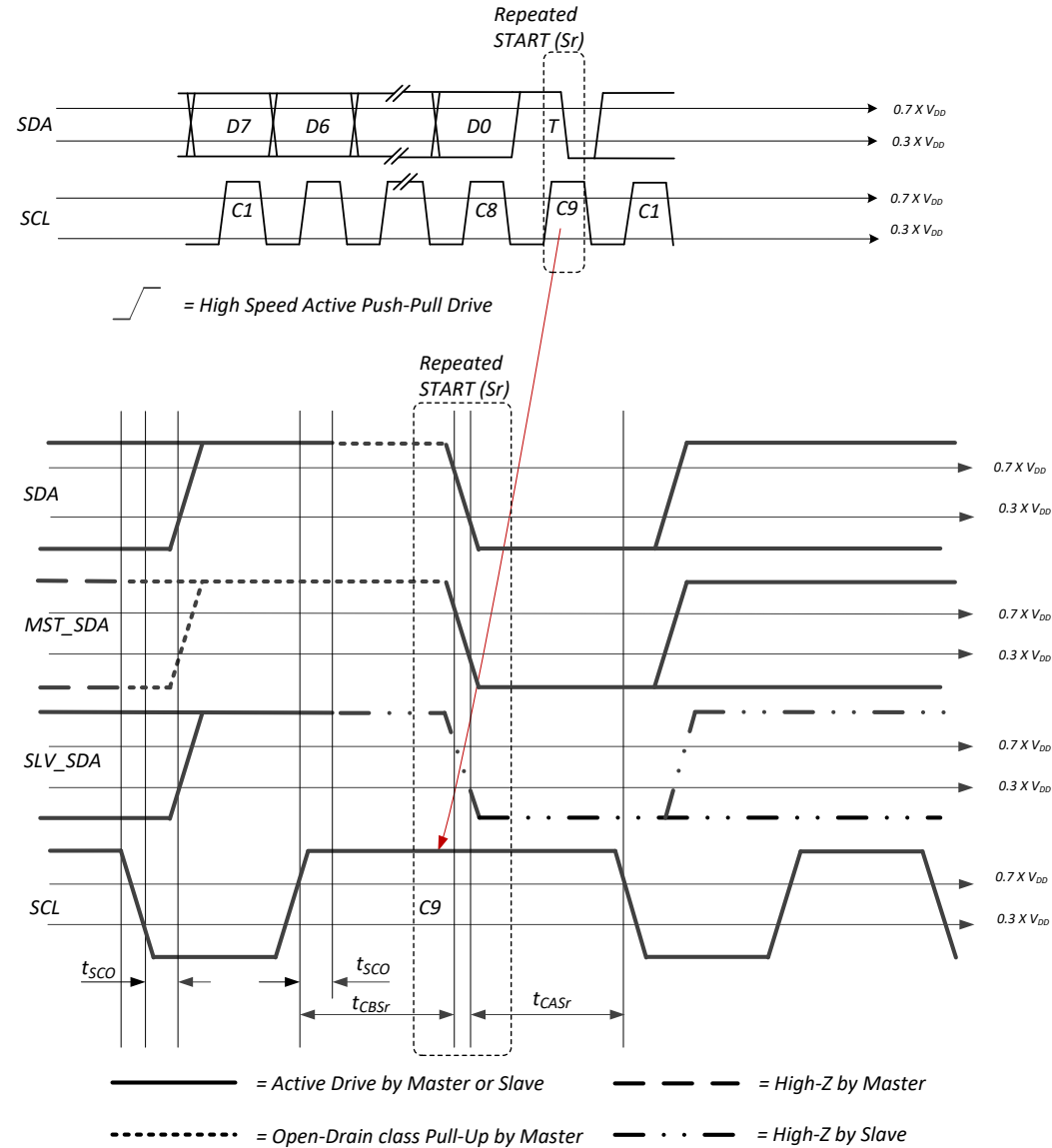
Figure 43 T-Bit When Master Ends Read via Repeated START and Further Transfer
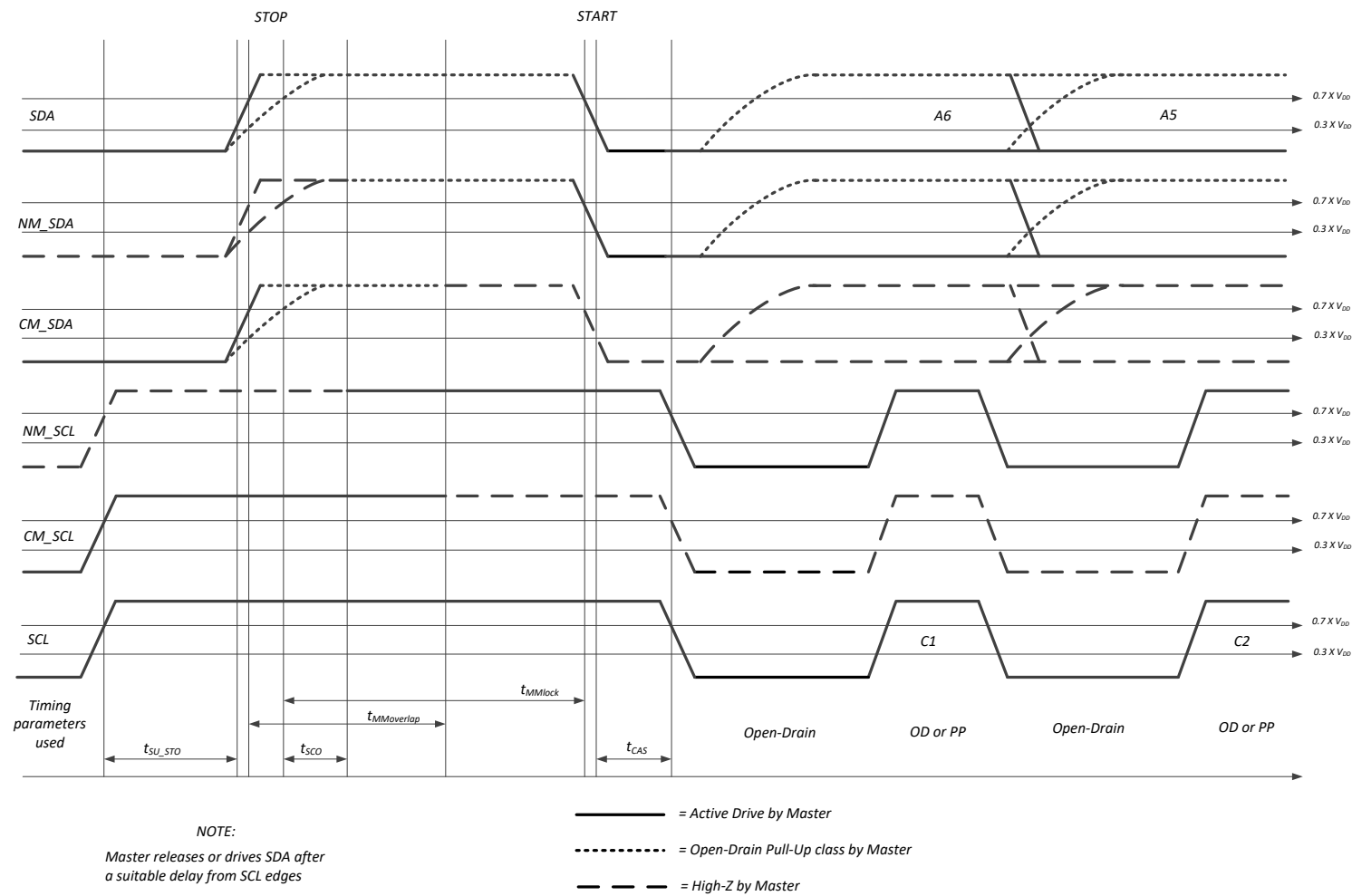
Figure 44 Master to Master Bus Handoff

Figure 45 shows the timing parameters related to the Spike Filter on a Legacy I²C Device. This timing shall be met, in order to ensure that Legacy I²C Devices properly ignore I3C High Speed Mode. It is recommended that both the SCL pin and the SDA pin have a Spike Filter. The timing is specified in *Table 59*.

Input to SDA/SCL Pins

0.7 X V$_{DD}$

0.3 X V$_{DD}$

Legacy I²C SDA/SCL Pins After Spike Filter

Suppressed by 50ns filter

0.7 X V$_{DD}$

0.3 X V$_{DD}$

I3C SDA/SCL Pins Without Spike Filter

0.7 X V$_{DD}$

0.3 X V$_{DD}$

t$_{SPIKE}$ **t$_{DIG\_H}$**   **t$_{DIG\_L}$**   **t$_{SPIKE}$**

Figure 45 I²C Spike Filter Behavior

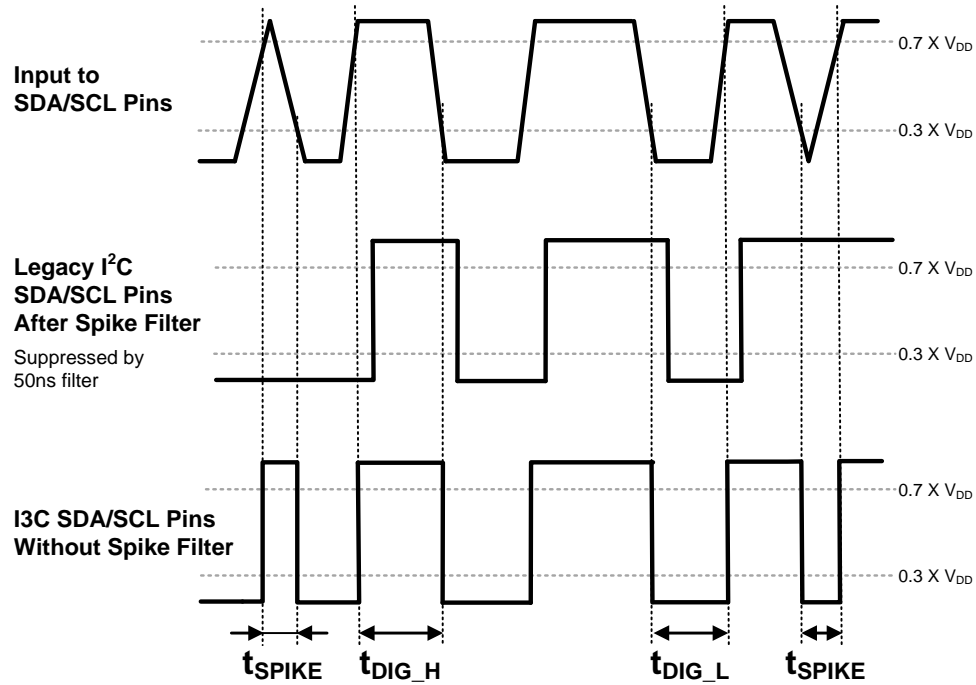2304   *Table 60* through *Table 62* detail how timing and drive strength are adjusted during transmission of an I3C
2305   Message.

2306

Table 60 Timing and Drive for Start of New Frame: No Contention on A7

	S	Header				Data	P
	START	ArbBit	Addr [5:0]	RnW	ACK	N Data + Parity	STOP
SDA Mode	Open Drain	Open Drain	Push-Pull	Push-Pull	Open Drain	Push-Pull	Push-Pull
Clocks	1	1	6	1	1	9 * N	1

2307

Table 61 Timing and Drive for Start of New Frame: With Contention on A7

	S	Header				Data	P
	START	ArbBit	Addr [5:0]	RnW	ACK	N Data + Parity	STOP
SDA Mode	Open Drain	Open Drain	Open Drain	Open Drain	Open Drain	Push-Pull	Push-Pull
Clocks	1	1	6	1	1	9 * N	1
Note: *Contention on A6, so Arbitrated over A5 to A0 and RnW.*							

2308

Table 62 Timing and Drive for Continuation of Frame Using Repeated START

	S	Header / Data…	Sr	Header		Data	P
	START	…	START	Addr/RW	ACK	N Data + Parity	STOP
SDA Mode	Open Drain	…	Push-Pull	Push-Pull	Open Drain	Push-Pull	Push-Pull
Clocks	1	…	1	8	1	9 * N	1
Note: *There may be any number of Repeated STARTs before the STOP.*							

This page intentionally left blank.

Annex A I3C Communication Format Details

A.1 I3C CCC Transfers

I3C Broadcast CCC Write

I3C Directed CCC Write

I3C Directed CCC Read

From Master to Slave	ACK With Handoff	ACK = Acknowledge (SDA Low)
		NACK = Not Acknowledge (NACK)
From Slave to Master	ACK Without Handoff	S = START Condition
		Sr = Repeated START Condition
		P = STOP Condition
Transition Bit Without Handoff (Parity Bit for CCC/Write Data)		T = Transition Bit alternative to ACK/NACK
Transition Bit With Handoff (End-of-Data for Read Data)		

NOTE: **1.** Optional Write Data can be 0/1/n bytes, based on the CCC
2. Read Data can be 1/n bytes based on the CCC

2309

2310

Figure 46 I3C CCC Transfers

A.2   I3C Private Write and Read Transfers

| S | I3C Reserved Byte (7'h7E) (RnW=0) | ACK | Sr | I3C Dynamic Slave Address (RnW=0) | ACK | Write Data-1 | T | **•••** | Write Data-N | T | Sr or P |

I3C Private Write Transfer Initiated with START Condition

| Sr | I3C Dynamic Slave Address (RnW=0) | ACK | Write Data-1 | T | **•••** | Write Data-N | T | Sr or P |

I3C Private Write Transfer Initiated with Repeated START Condition

| S | I3C Reserved Byte (7'h7E) (RnW=0) | ACK | Sr | I3C Dynamic Slave Address (RnW=1) | ACK | Read Data-1 | T | **•••** | Read Data-N | T | Sr or P |

I3C Private Read Transfer Initiated with START Condition

| Sr | I3C Dynamic Slave Address (RnW=1) | ACK | Read Data-1 | T | **•••** | Read Data-N | T | Sr or P |

I3C Private Read Transfer Initiated with Repeated START Condition

From Master to Slave

From Slave to Master

ACK   ACK With Handoff

ACK   ACK Without Handoff

Transition Bit without Handoff
(Parity Bit for Write Data)

Transition Bit with Handoff
(End-of-Data for Read Data)

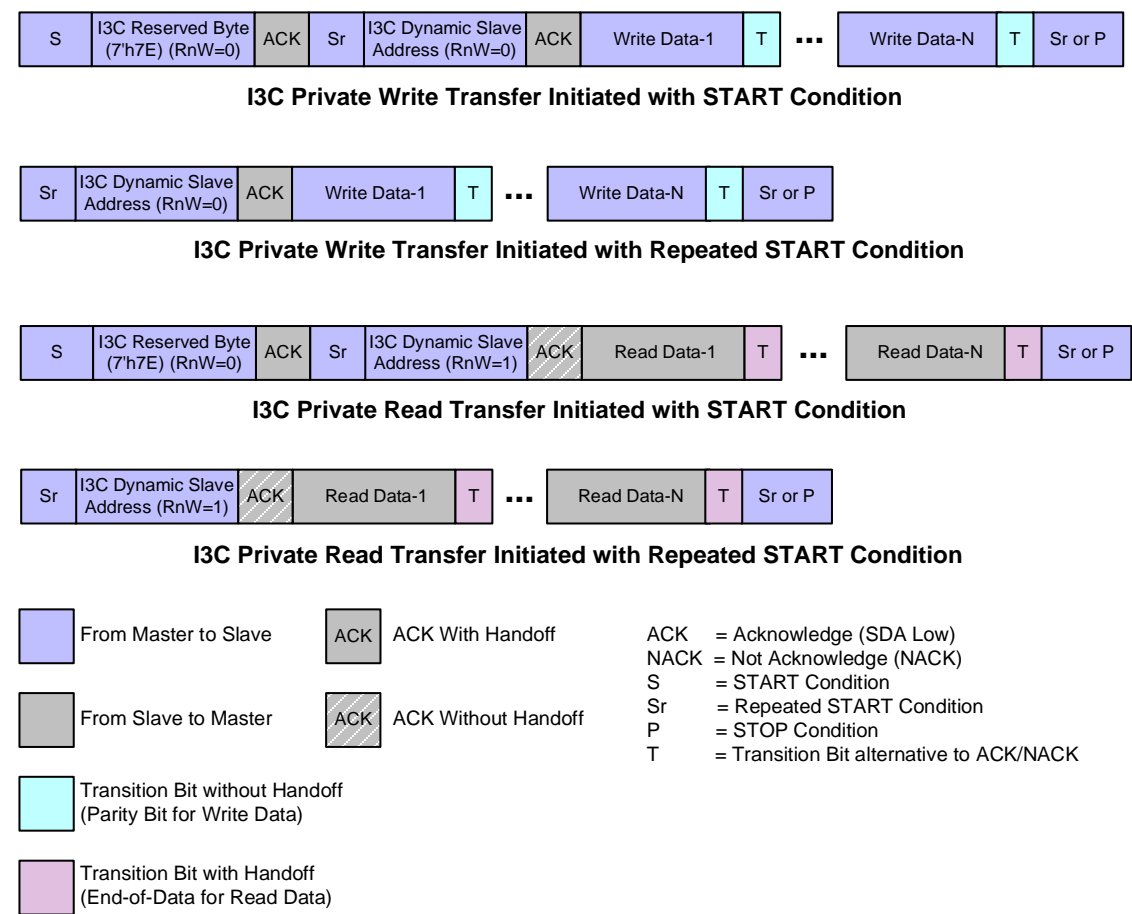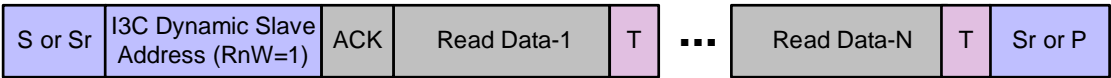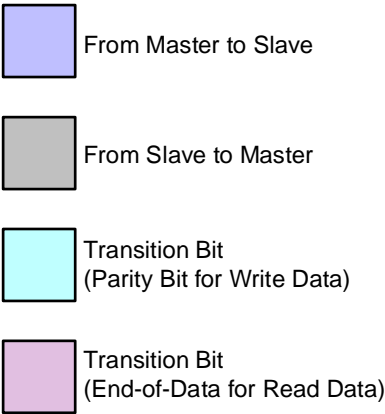| ACK | = Acknowledge (SDA Low) |
NACK	= Not Acknowledge (NACK)
S	= START Condition
Sr	= Repeated START Condition
P	= STOP Condition
T	= Transition Bit alternative to ACK/NACK

2311

2312

Figure 47 I3C Private Write and Read Transfers with 7'h7E Address

I3C Private Write Transfer

I3C Private Read Transfer Initiated with Repeated START Condition

From Master to Slave

From Slave to Master

Transition Bit
(Parity Bit for Write Data)

Transition Bit
(End-of-Data for Read Data)

ACK = Acknowledge (SDA Low)
NACK = Not Acknowledge (NACK)
S = START Condition
Sr = Repeated START Condition
P = STOP Condition
T = Transition Bit alternative to ACK/NACK

2313

2314 **Figure 48 I3C Private Write and Read Transfers without 7'h7E Address**

A.3 Legacy I²C Write and Read Transfers on the I3C Bus

Legacy I²C Write Transfer Initiated with START Condition

Legacy I²C Write Transfer Initiated with Repeated START Condition

Legacy I²C Read Transfer Initiated with START Condition

Legacy I²C Read Transfer Initiated with Repeated START Condition

	ACK = Acknowledge (SDA Low)
From Master to Slave	NACK = Not Acknowledge (NACK)
	S = START Condition
	Sr = Repeated START Condition
From Slave to Master	P = STOP Condition

Figure 49 Legacy I²C Write and Read Transfers in I3C Bus with 7'h7E Address

Legacy I²C Write Transfer

Legacy I²C Read Transfer

	ACK = Acknowledge (SDA Low)
From Master to Slave	NACK = Not Acknowledge (NACK)
	S = START Condition
	Sr = Repeated START Condition
From Slave to Master	P = STOP Condition
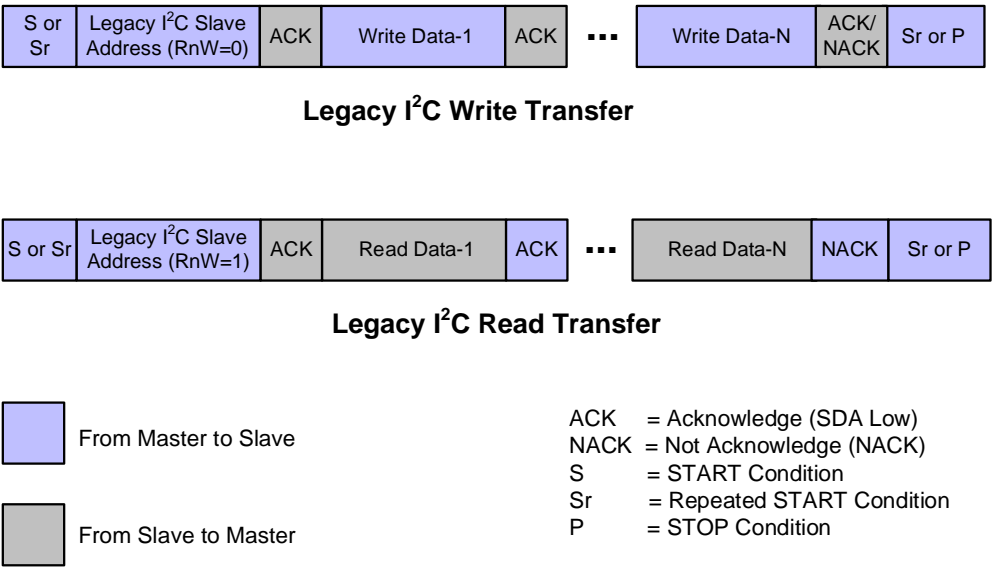
Figure 50 Legacy I²C Write and Read Transfers in I3C Bus without 7'h7E Address

A.4 Dynamic Address and Enter HDR
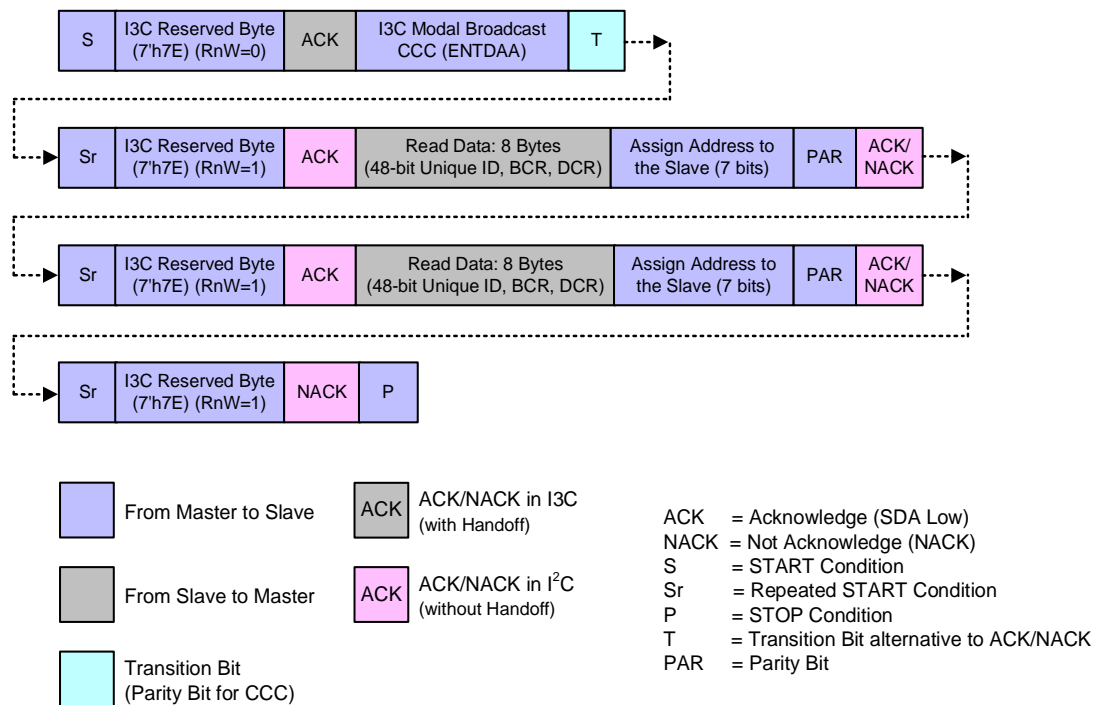
Figure 51 Dynamic Address Allocation ENTDAA CCC Bus Modal
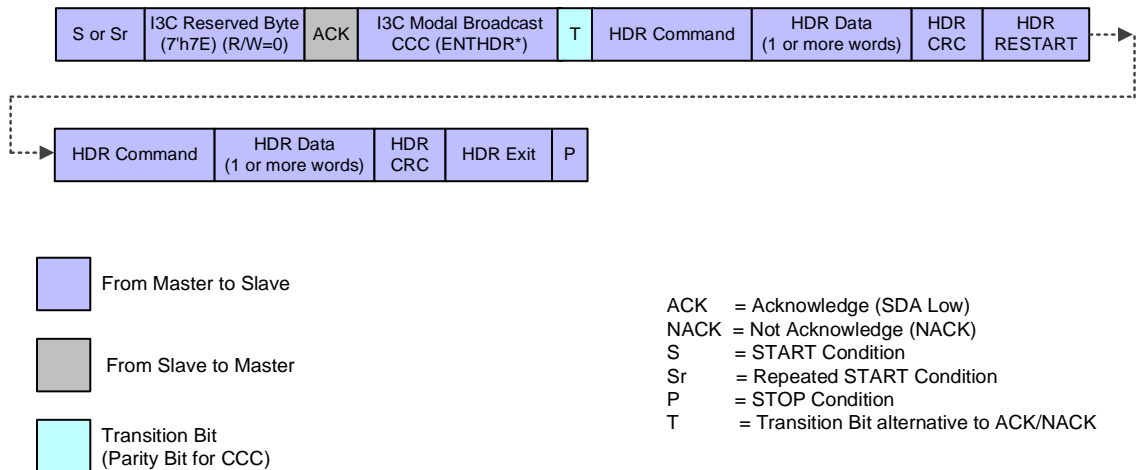
Figure 52 Enter HDR Mode CCC Bus Modal

This page intentionally left blank.

Annex B SDR Mode Error Type Origins
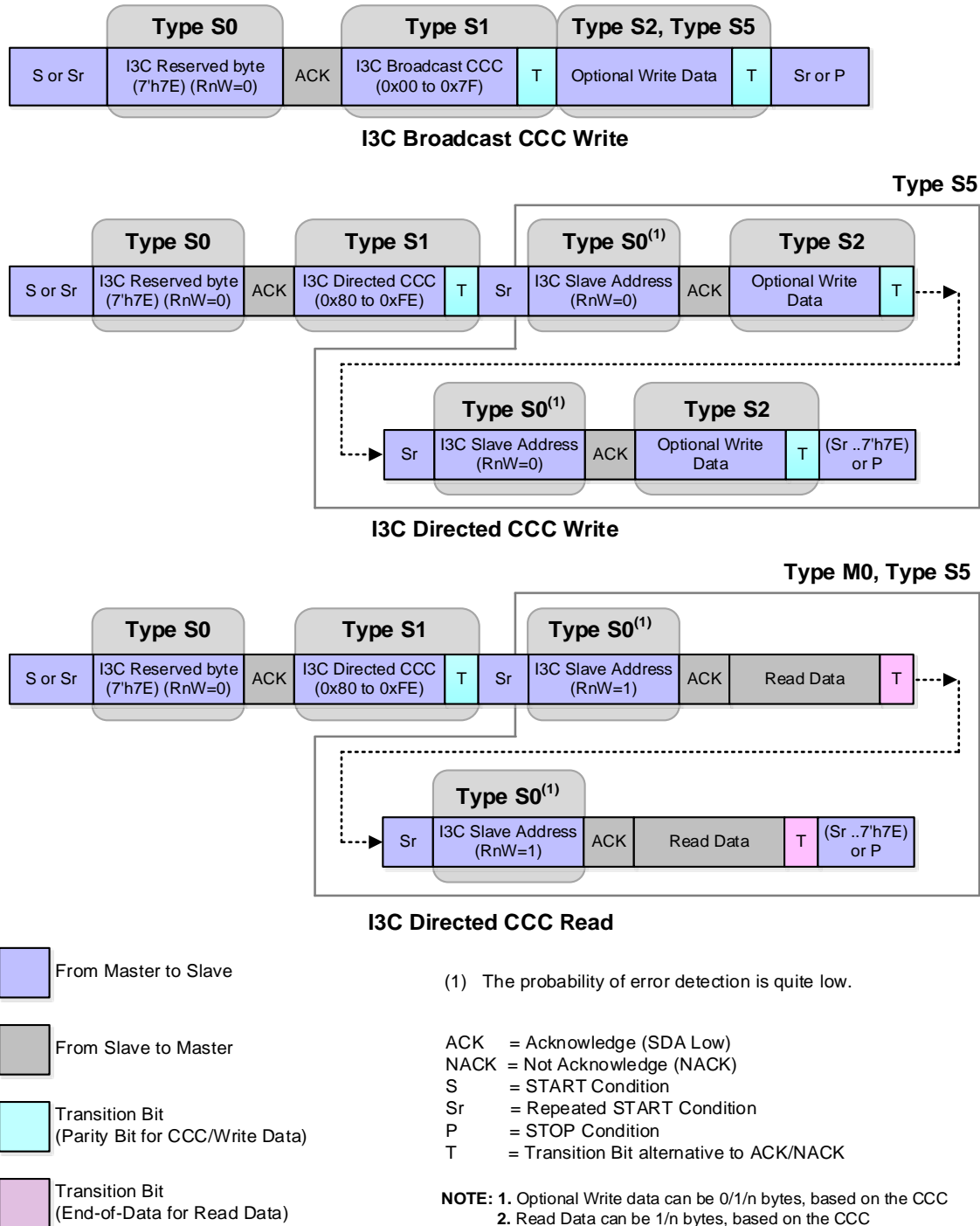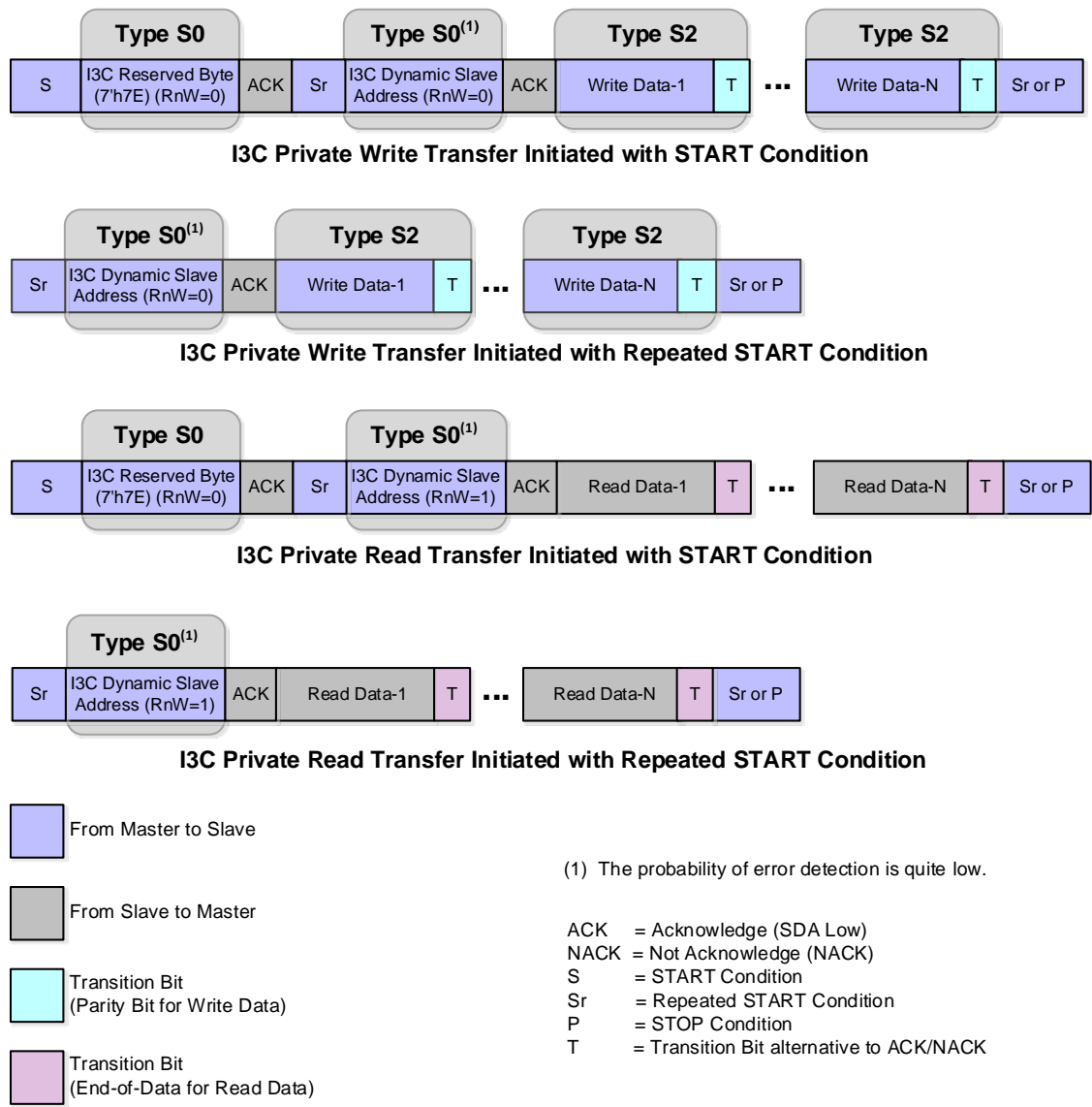
B.1   Error Types in I3C CCC Transfers

Figure 53 Error Type Origins: I3C CCC Transfers

B.2   Error Types in I3C Private Read and Write Transfers

I3C Private Write Transfer Initiated with START Condition

I3C Private Write Transfer Initiated with Repeated START Condition

I3C Private Read Transfer Initiated with START Condition

I3C Private Read Transfer Initiated with Repeated START Condition

From Master to Slave

From Slave to Master

Transition Bit
(Parity Bit for Write Data)

Transition Bit
(End-of-Data for Read Data)

(1) The probability of error detection is quite low.

ACK   = Acknowledge (SDA Low)
NACK = Not Acknowledge (NACK)
S   = START Condition
Sr   = Repeated START Condition
P   = STOP Condition
T   = Transition Bit alternative to ACK/NACK

Figure 54 Error Type Origins: I3C CCC Private Write & Read Transfers with 7'h7E Address
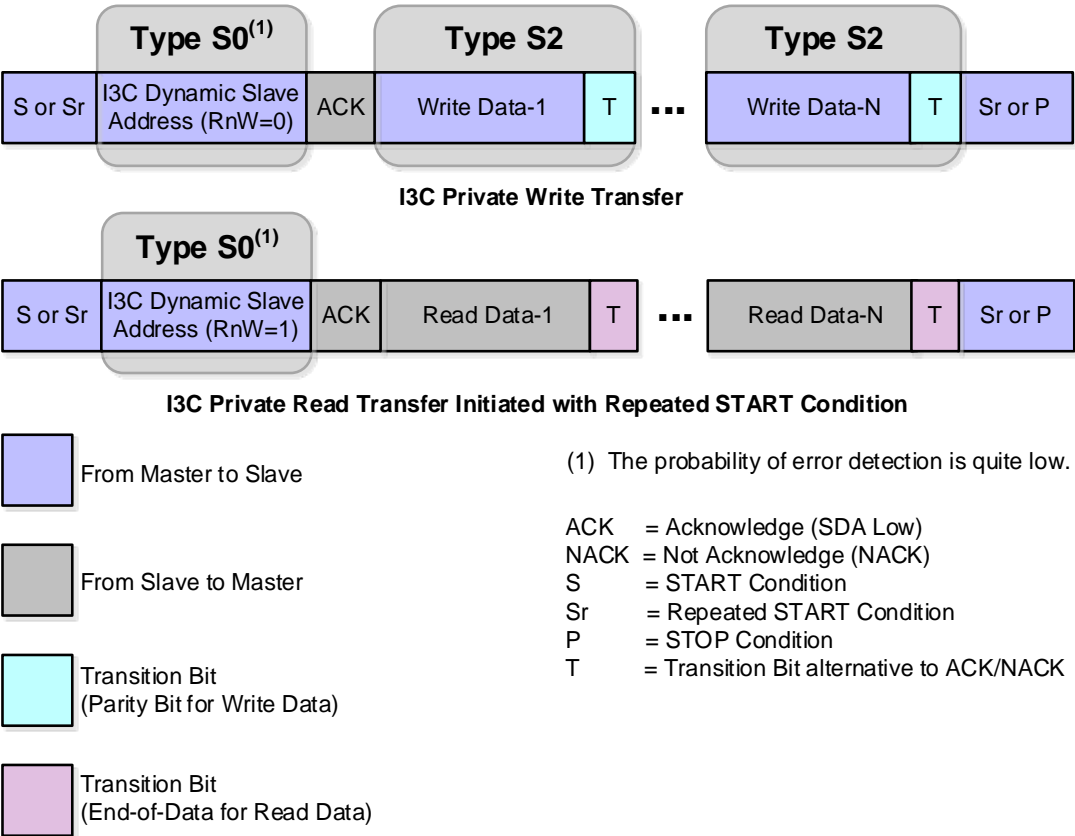
Figure 55 Error Type Origins: I3C Private Read Transfers without 7'h7E Address

B.3   Error Types in Dynamic Address Arbitration
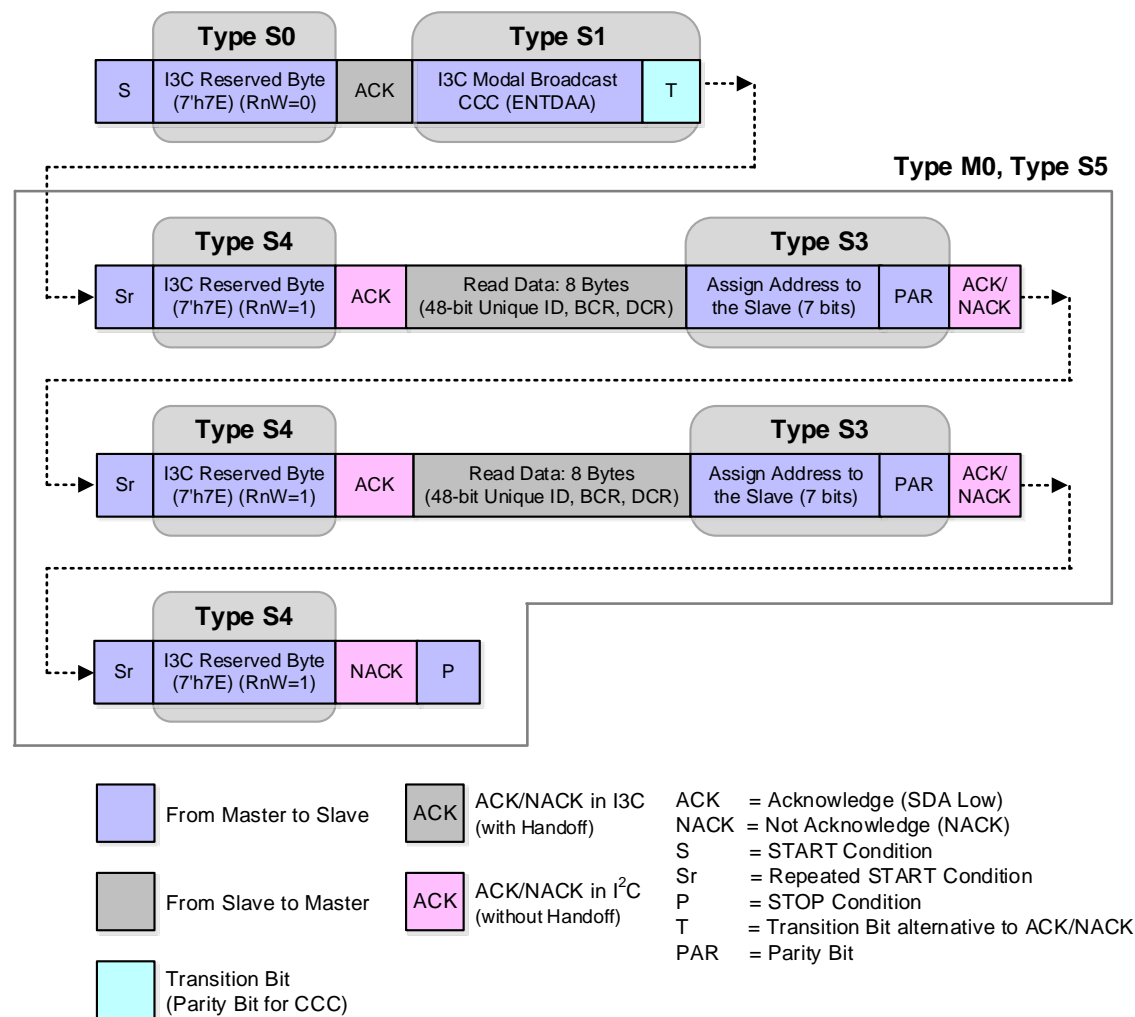
Figure 56 Error Type Origins: Dynamic Address Arbitration

Annex C I3C Master FSMs

2331 *Figure 57* shows the key transmission Modes and their entry, resume and exit sequences. It includes the SDR
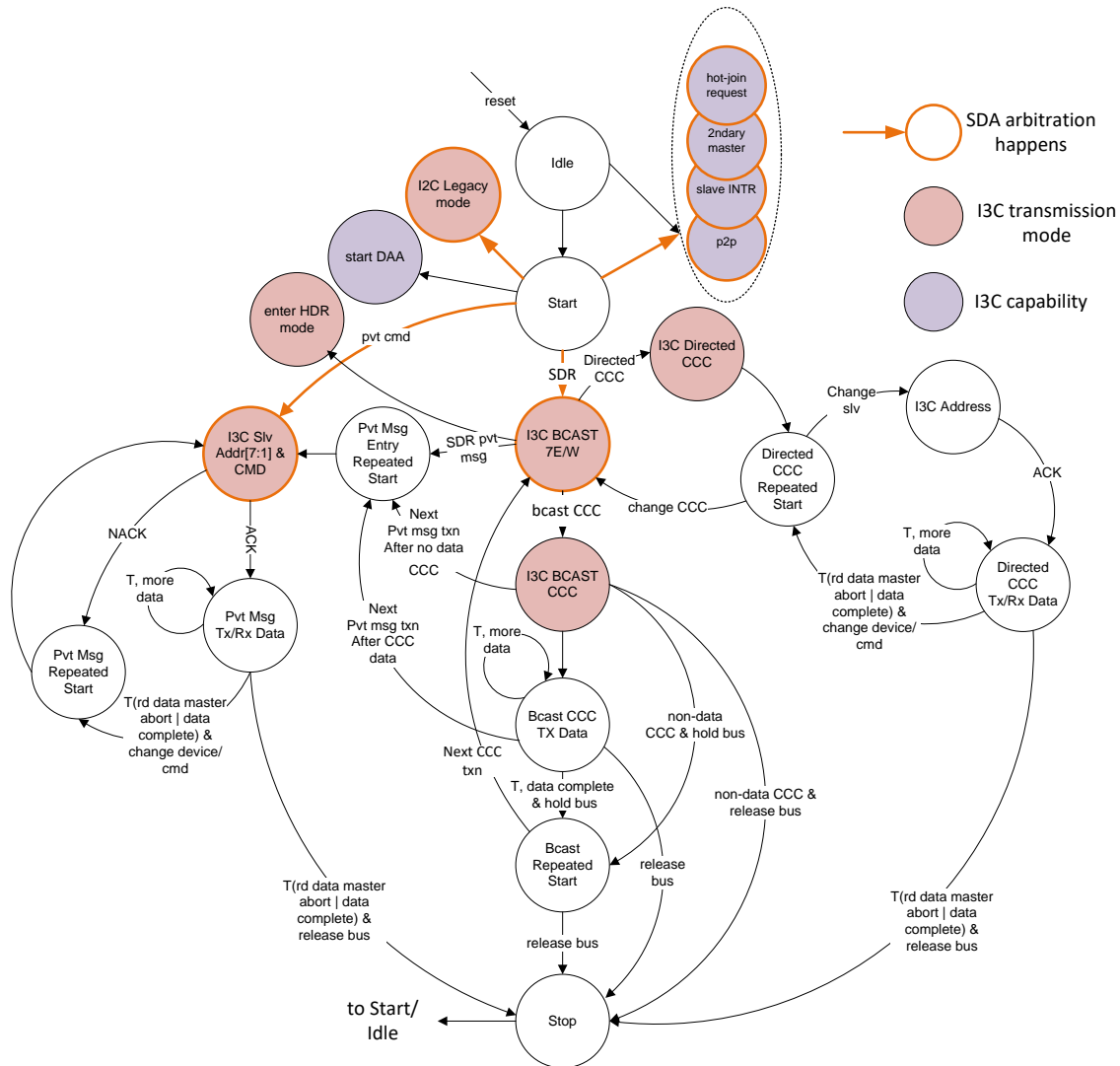2332 transmission states, and differentiates these from other capabilities and Modes. It captures the states at which
2333 Arbitration happens as well.

2334

2335 **Figure 57 I3C Main Master FSM**

2336 *Figure 58* through *Figure 62* represent I3C features including In-Band Interrupts, Secondary Master,
2337 Dynamic Address Assignment, and Hot-Join.

2338

Slave Interrupt request

2339
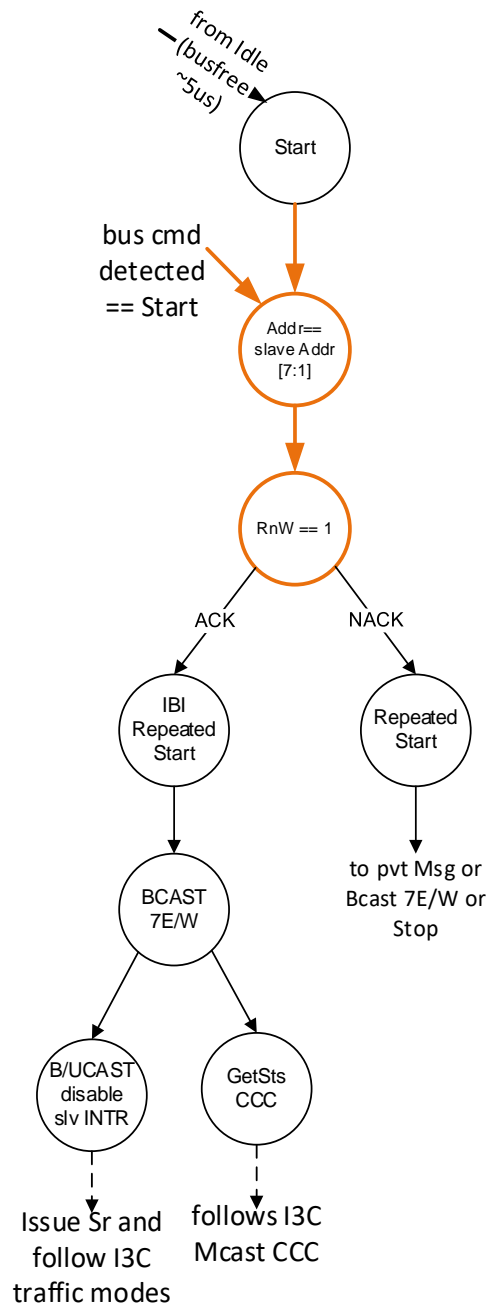
2340 **Figure 58 Slave Interrupt Request FSM**

DAA

from Start

2341

2342   **Figure 59 Dynamic Address Assignment FSM**

HotJoin

from Idle
(busfree
~1ms)

Start

bus cmd
detected
== Start

slave Addr
== 0x01

CMD
RnW == 0

ACK

NACK

Repeated
Start

Repeated
Start

disable
Hot Join

BCAST
7E/W

BCAST
Disable
Hot Join
CCC

to pvt Msg or
Bcast 7E/W or
Stop

ENTDAA
CCC

follow DAA
Flow
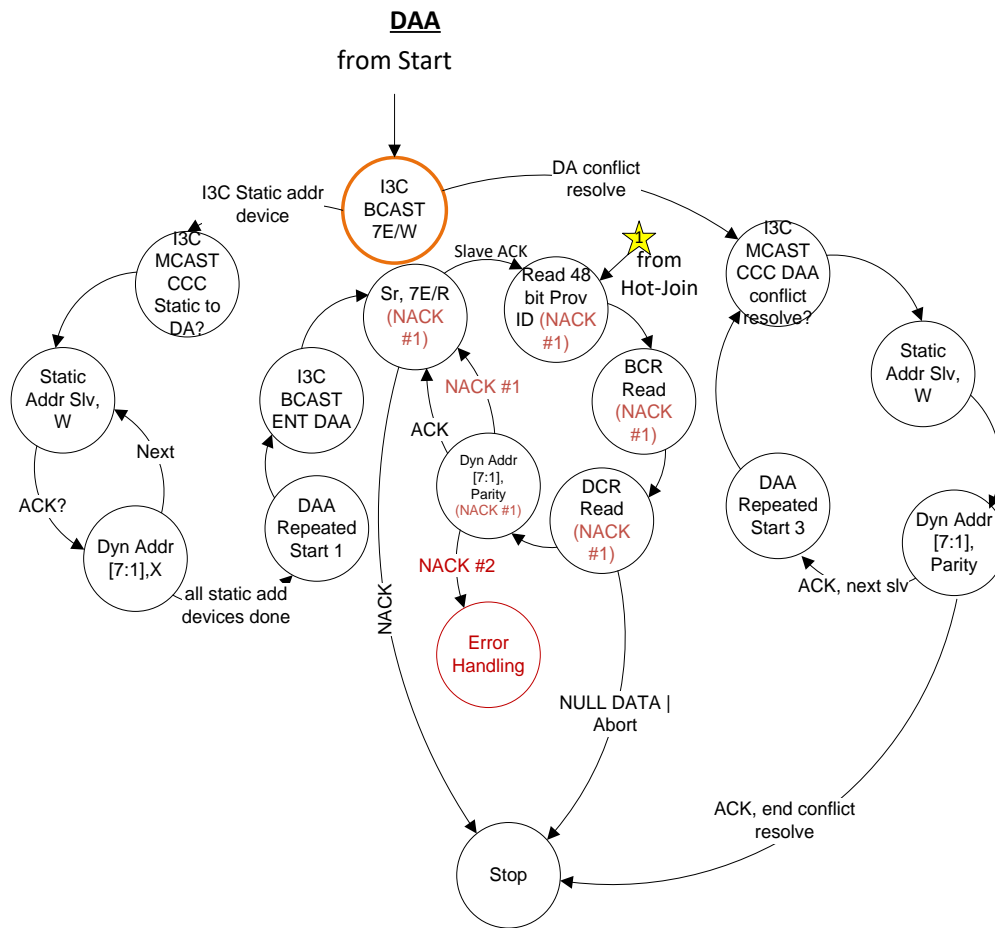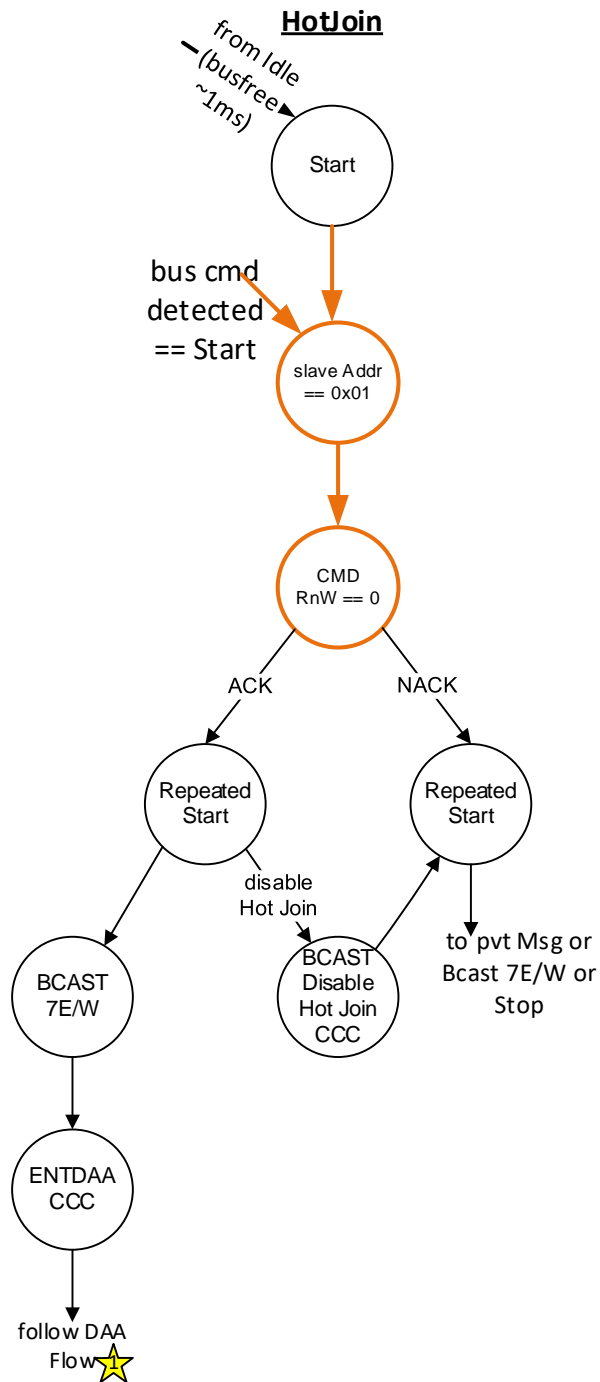
Figure 60 Hot-Join FSM

`2343`

`2344`
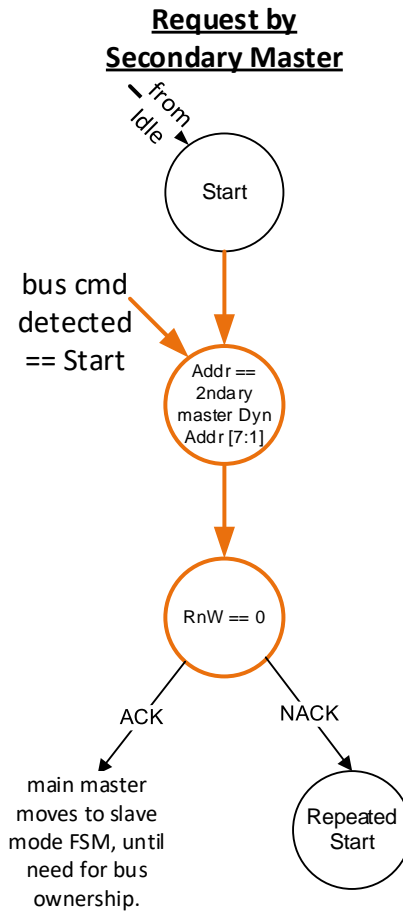
Figure 61 Secondary Master Request FSM
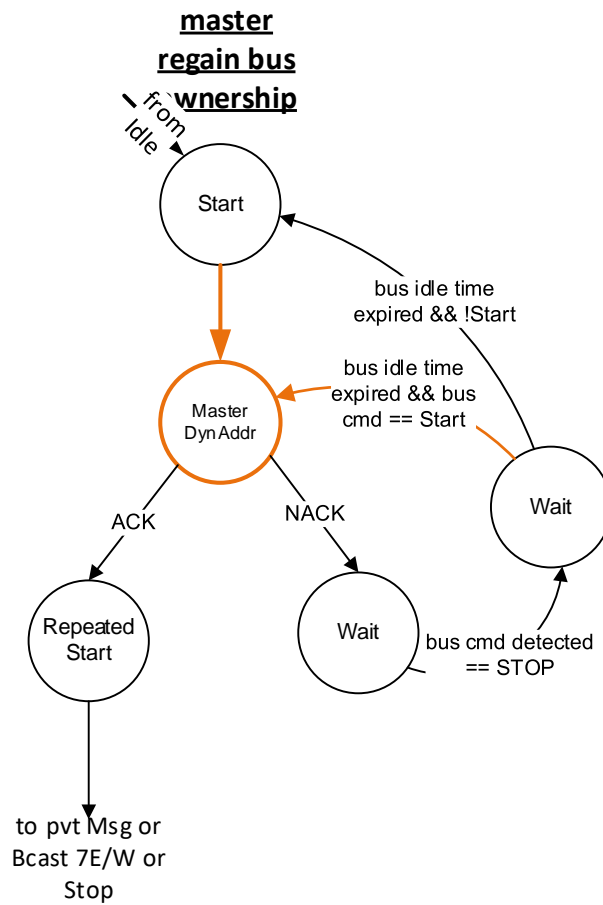
Figure 62 Master Regaining Bus Ownership FSM

2347
2348

2349   *Figure 63* is for reference only.

I2C legacy Master FSM

2350

2351   **Figure 63 I²C Legacy Master FSM**

This page intentionally left blank.

Annex D Typical I3C Basic Protocol Communications

Figure 64 through *Figure 66* each illustrate a typical communication for the I3C SDR Protocol. While these diagrams do not exhaustively illustrate all possible I3C SDR communications, they do serve as useful introductions to the signaling and transmission formatting used in the I3C SDR Protocol.

Figure 64 illustrates example communication using I3C Single Data Rate (SDR) coding. It shows the Master reading a byte of data from the Slave at Address 0x2B in SDR Mode.

From the Bus Free Condition, the Master issues a START by driving the SDA line Low while keeping the SCL line High. It then issues the Broadcast Address (7'h7E) followed by RnW (0 for Write). Then the Master turns on a pull-up resistor and goes to Open Drain. All Slaves ACK by pulling the SDA line Low (in the Figure, pink fill means the Slave is in control of the SDA line at this time). The Master then issues a Repeated START, then the Address of the Slave (0x2B) it wants to read followed by RnW (1 for Read). The Master then turns on a pull-up resistor and goes to Open Drain, allowing the Slave to acknowledge by pulling the SDA line Low. At this point, the Master continues to toggle the SCL line and release the SDA line, allowing the Slave to drive SDA to send one byte of data (0x4A) followed by 'T'. T=1 informs the Master that there is additional data, whereas T=0 signals the end. Here there is additional data, so the Slave drives SDA High until SCL goes High, at which time it releases SDA. The Master has the option of holding SDA High with a weak pull-up, which signals to the Slave that the Master allows another byte to be transmitted, or to pull SDA Low (while SCL is High – hence a Repeated START), which would signal to the Slave that the Master has terminated the Read and is taking over.

SDR Mode is backwards compatible with Legacy I²C Devices, because the High time of an SCL pulse is always less than 50ns and therefore SCL will always appear to be Low because of the I²C 50ns Spike Filter.
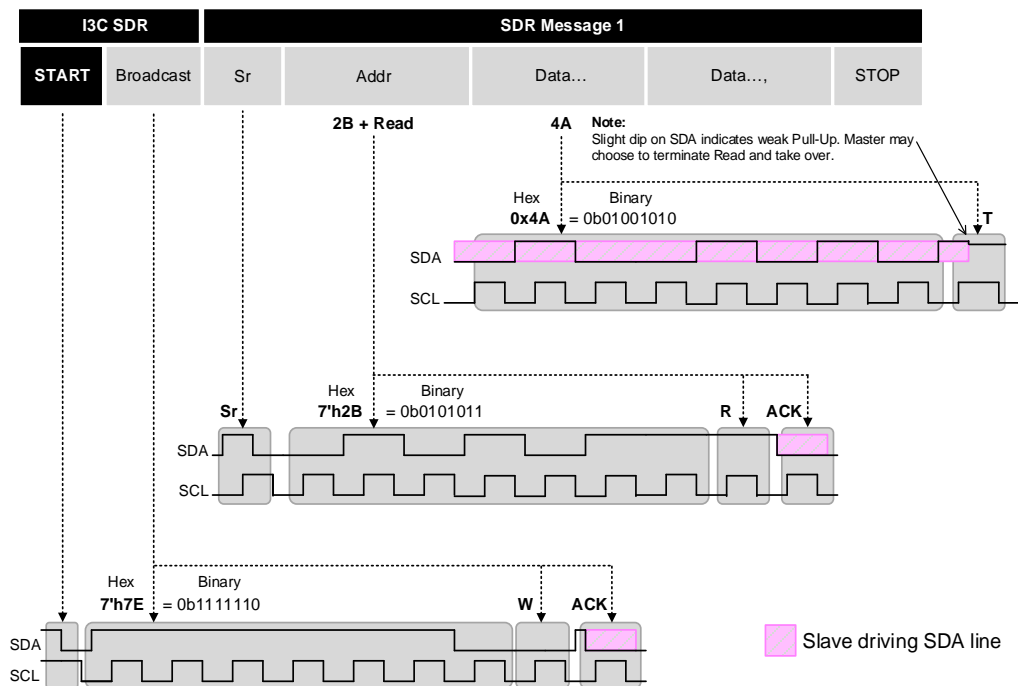

2352

2353

Figure 64 Example Communication Using I3C Coding SDR

Figure 65 shows the Master issuing a CCC Direct Command to a single Slave. This particular command (GETPID) reads the Provisional ID of a Slave.

From the Bus Free Condition, the Master issues a START by driving the SDA line Low while keeping the SCL line High. It then issues the Broadcast Address (7'h7E) followed by RnW (0 for Write). Then the Master turns on a pull-up resistor and goes to Open Drain. All Slaves ACK by pulling SDA Low (in the Figure, pink fill means the Slaves are in control of SDA at this time). The Master then issues the Direct Common Command Code for GETPID (0x8C) followed by parity bit 'T' (odd parity = 0 for 0x8C) then the 7-bit Dynamic Address of the Slave (chosen arbitrarily here to be 0x2B) followed by a RnW bit (1 for Read). Then the Master turns on a pull-up resistor and goes to Open Drain, allowing the Slave at Address 0x2B to ACK by pulling SDA Low, which tells the Master that the Slave Acknowledges the command and will comply. (Alternatively, the Slave may NACK by not pulling SDA Low, which would inform the Master that the Slave will not comply – in this case, that an error occurred.) Following the ACK the Slave outputs its 48-bit PID one byte at a time, and then the Master issues a Repeated START (this part of the waveform sequence is not shown in the Figure).
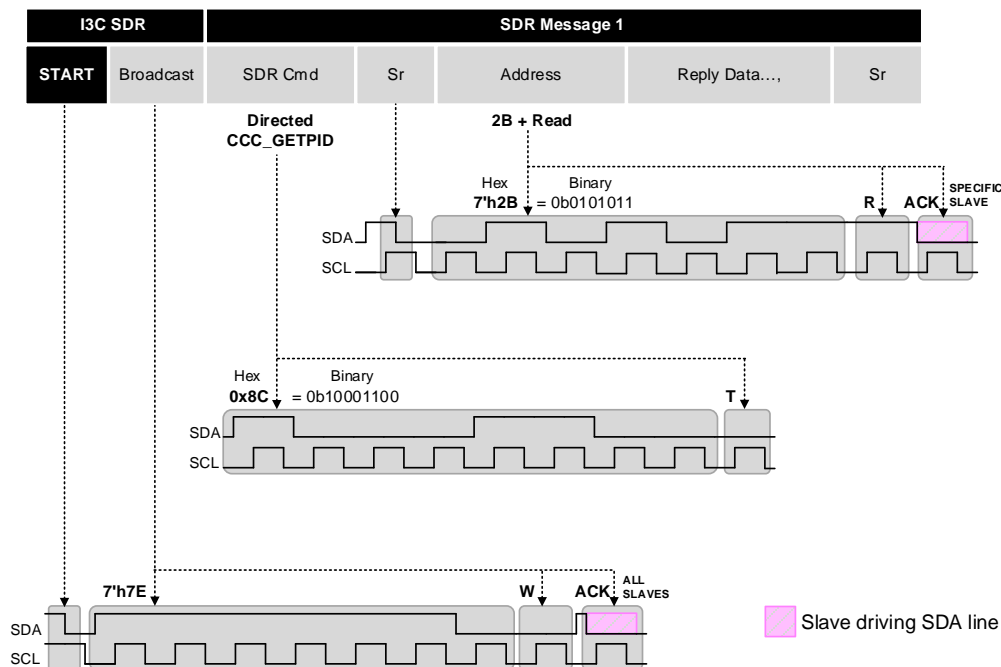

2354
2355 **Figure 65 Example Communication Using I3C Coding SDR with CCC Direct Addressing**

Figure 66 illustrates example SDR communication with a CCC Broadcast command. The command used in this example sets the Maximum Read Length of all Slaves to 43 bytes (0x002B).

From the Bus Free Condition, the Master issues a START by driving the SDA line Low while keeping the SCL line High. It then issues the Broadcast Address (7'h7E) followed by RnW (0 for Write). Then the Master turns on a pull-up resistor and goes to Open Drain. All Slaves ACK by pulling SDA Low (in the Figure, pink fill means the Slaves are in control of SDA at this time). The Master then issues the Broadcast Common Command Code for SETMRL (0x09) followed by parity bit 'T' (odd parity = 1 for 0x09), and then 2 data bytes (MSb first) to define the maximum number of bytes which can be read from a Slave in a single read operation. Each data byte is followed by a 'T' bit (parity bit – odd parity). After this the Master issues a Repeated START.
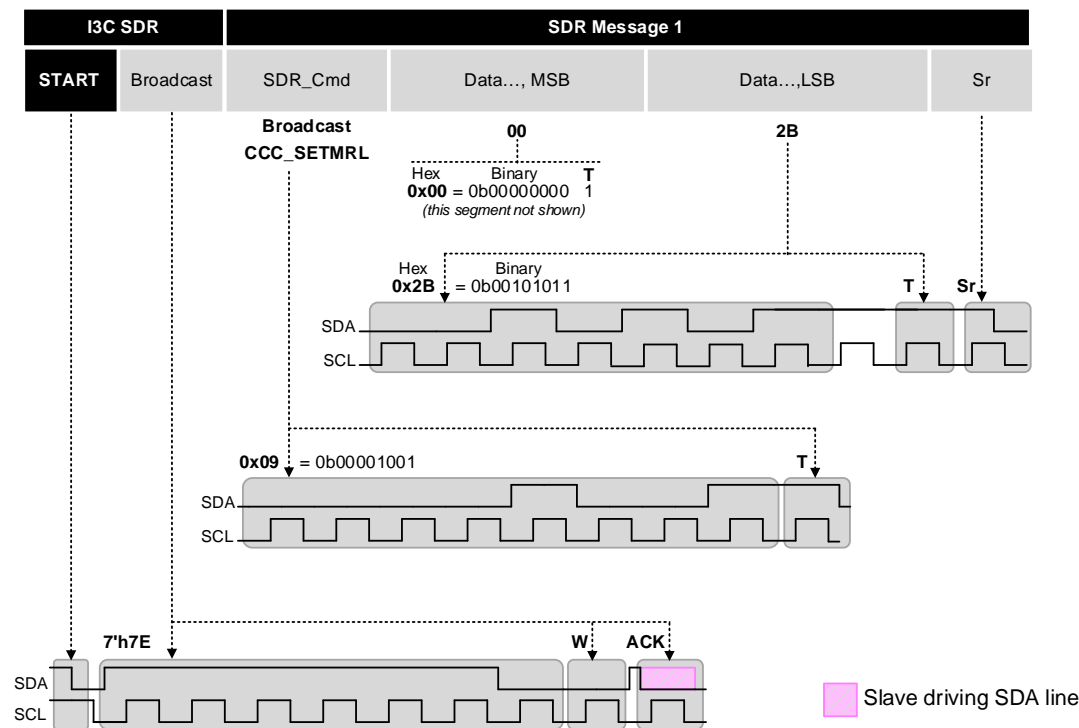

2356

2357   **Figure 66 Example Communication Using I3C Coding SDR with CCC Broadcast**

This page intentionally left blank.

Annex E MIPI I3C Basic Specification Supplemental Patent Licensing Terms

2358 This agreement (the "Agreement") between MIPI Alliance Inc. ("MIPI") and each MIPI member or other
2359 party that has manifest agreement to these terms (each a "Licensor" and collectively the "Licensors") is
2360 effective as of the date the I3C Basic Specification (defined below) is first approved by the MIPI Board (the
2361 "Effective Date"). Capitalized terms used in this Agreement that are not expressly defined here have the
2362 meaning identified in the MIPI Membership Agreement or MIPI Bylaws, as applicable. For convenience,
2363 key definitions are reproduced in Attachment A. For the avoidance of doubt: (a) in connection with this
2364 Agreement, any reference to a "MIPI Specification" means the MIPI I3C Basic Specification as described in
2365 Section 2, and (b) any rights or obligations created under this Agreement are independent of any rights or
2366 obligations created under the MIPI Membership Agreement, Bylaws or any other agreements, and nothing in
2367 this Agreement is intended to alter rights or obligations established elsewhere.

2368 **1. Background.** Typically, MIPI specifications are implemented only by MIPI members. MIPI
2369 members make certain intellectual property licensing commitments to other members under the MIPI
2370 Membership Agreement, with different rules applying to "Mobile Terminals" and "Accessories," in contrast
2371 to other types of implementations. The MIPI Board intends to make the MIPI I3C Basic Specification
2372 available for implementation by parties who are non-members of MIPI, however. Further, both MIPI
2373 members and non-members may use this specification inside and outside of Mobile Terminals or Accessories.
2374 The Licensors contributed to the development of the MIPI I3C Basic Specification, and desire to see it widely
2375 used. MIPI and the Licensors believe that making licenses available (as set forth in this Agreement) to both
2376 member and non-member implementers, for all types of implementations, will facilitate widespread adoption
2377 of the MIPI I3C Basic Specification, to the benefit of MIPI, the Licensors, and the broader community that
2378 MIPI serves.

2379 **2. MIPI I3C Basic Specification**. "MIPI I3C Basic Specification" means the specification titled "I3C
2380 Basic 1.0" as approved by the MIPI Board, and all subsequent versions of such specification approved by the
2381 MIPI Board after the Effective Date. Any party implementing the MIPI I3C Basic Specification, whether or
2382 not a MIPI member, is an "Implementer." For the avoidance of doubt: the MIPI I3C Basic Specification is
2383 distinct from the MIPI I3C Specification version 1.0 approved by the MIPI Board on Dec. 31, 2016. The
2384 terms of this Agreement apply exclusively to the MIPI I3C Basic Specification.

2385 **3. License commitment.**

2386 *a. RAND-Z license obligation.* For the MIPI I3C Basic Specification only, Licensor hereby
2387 agrees to grant, and to cause its Affiliates to grant, to any requesting Implementer a worldwide,
2388 non-exclusive, non-sublicensable license under the Necessary Claims of Licensor or its Affiliates,
2389 with zero royalties or other compensation, under terms and conditions that are reasonable and
2390 nondiscriminatory, to make, have made, use, import, offer to sell, lease, sell, promote and
2391 otherwise distribute Compliant Portions. Licensor shall not be obligated to license any part or
2392 function of a product in which a Compliant Portion is incorporated that is not itself a Compliant
2393 Portion.

2394 *b. Reciprocity; defensive suspension.* Licensor shall not be obligated to license any Implementer
2395 if that Implementer does not agree to make patent licenses available under any Necessary Claims
2396 of that Implementer and its Affiliates to Licensors and all other Implementers under terms
2397 substantially identical to the terms described in this Agreement. Further, a Licensor may suspend
2398 any license granted under this Agreement to any Implementer if that Implementer or its Affiliate
2399 initiates against any party litigation that alleges infringement of a Necessary Claim of Implementer
2400 or its Affiliate in connection with the MIPI I3C Basic Specification. Additionally, subject to
2401 Section 3.1(f) of the MIPI Membership Agreement as between MIPI Members, a Licensor may
2402 terminate, ab initio, any license granted pursuant to this Agreement to any Implementer that
2403 initiates litigation against the Licensor alleging infringement of any patent claim of the
2404 Implementer or its Affiliate. For the purposes of this Agreement, a party that files a suit which is

2405 defensive based on a patent infringement claim or suit by another party will not be deemed to have
2406 initiated litigation.

2407 *c.   Circumvention or transfer.* Licensor agrees that it has not transferred and will not transfer any
2408 patent having Necessary Claims solely for the purpose of circumventing the obligations described
2409 in this Agreement. In addition, Licensor agrees that any transfers by Licensor to a third party of a
2410 patent having relevant Necessary Claims, whether or not recognized as Necessary Claims at the
2411 time of transfer, shall be subject to (i) the terms and conditions of this Agreement, and (ii) the
2412 agreement that the third party shall grant licenses under said Necessary Claims to Implementers
2413 pursuant to the terms of this Agreement in like manner and to the same extent as the third party
2414 would be required to do if it had executed this Agreement. A transfer of ownership in a business
2415 entity which owns or has the right to license a patent having Necessary Claims shall be considered
2416 a transfer of such patent.

2417 **4.   Termination of obligation to license future versions.** Each Licensor may terminate its
2418 participation in this Agreement at any time by providing written notice to the MIPI Managing Director;
2419 termination will be effective 30 days after such notice is actually received, subject to the survival points
2420 below. Promptly upon receipt of such notice, the MIPI Managing Director will alert the MIPI Board and will
2421 use reasonable efforts to notify all Licensors of such termination. After termination, the agreement to grant a
2422 license as provided in Section 3 shall survive in full force and effect only: (a) for versions of the MIPI I3C
2423 Basic Specification which the Board had approved before the effective date of termination; (b) for Necessary
2424 Claims relating to any version of the MIPI I3C Basic Specification approved after the effective date of
2425 termination that are used in a substantially similar manner and to a substantially similar extent with a
2426 substantially similar result as the Necessary Claims that were used in a prior version for which the Licensor
2427 is obligated to grant licenses under this Agreement; and (c) for those Necessary Claims that directly result
2428 from inclusion of Licensor-provided material in the draft version of the specification that existed immediately
2429 prior to Licensor's withdrawal. Termination of this Agreement by one Licensor does not impact the
2430 Agreement among MIPI or the other Licensors, not does it not impact Licensor's MIPI Membership
2431 Agreement, which will remain in full force and effect.

2432 **5.   Third party beneficiaries.** All Implementers, whether or not they are MIPI members, are intended
2433 third party beneficiaries of this Agreement.

2434 **6.   Counterparts; additional Licensors.** This Agreement may be signed in any number of
2435 counterparts. If additional parties manifest agreement to terms substantially identical to this Agreement, those
2436 parties will be deemed Licensors under this Agreement.

Attachment A

1.3. **"Compliant Portions"** means only those specific portions of products (hardware, software or combinations thereof) that: (i) both implement and are compliant with the relevant portions of the MIPI Specification, (ii) are qualified pursuant to the MIPI qualification process (if available), (iii) meet the requirements set forth in any compliance requirements set forth by the Corporation, applied to all Members on a nondiscriminatory basis, and (iv) are within the bounds of the Scope of IPR (defined below).

1.5 **"Interface"** means the protocols, signaling characteristics, commands, clocking signals, register models, application program interfaces and data structures to the extent they enable interoperation, interconnection or communication between integrated circuits (even if located on the same die).

1.7. **"Necessary Claims"** mean those claims of all patents and patent applications, other than design patents and design registrations, throughout the world which (i) a Member or its Affiliates has the right, at any time during the term of this Agreement, to grant licenses of the nature granted or agreed to be granted herein without such grant resulting in payment of royalties or other consideration to third parties (except for payments to Affiliates or to employees within the scope of their employment); (ii) are within the Scope of IPR; and (iii) are necessarily infringed by an implementation of a MIPI Specification, wherein such infringement could not have been avoided by another commercially reasonable non-infringing implementation of such MIPI Specification. Necessary Claims do not include (i) any claims other than those set forth above even if contained in the same patent as Necessary Claims, or (ii) any claims that read on any implementation of the Specification to the extent that such implementation is not within the bounds of the MIPI Specification.

1.8 **"Scope of IPR"** means Interfaces, solely to the extent disclosed with particularity in a MIPI Specification, where the purpose and sole licensed (under this agreement) use of such disclosure is to define, implement, and utilize an interface that enables interoperation, interconnection or communication in accordance with a MIPI Specification. Notwithstanding the foregoing, the Scope of IPR shall not include (i) any enabling technologies that may be necessary to make or use any product or portion thereof that complies with a MIPI Specification, but are not themselves expressly set forth in a MIPI Specification; (ii) semiconductor manufacturing technology, DSP architecture, processor architecture/microarchitecture, wireless communication technology, compiler technology, integrated circuit packaging technology, security technology, internal architectures of integrated circuits, applications which run on integrated circuits, audio coding technology, video coding technology or basic operating system technology; (iii) SDO Standards, whether in whole or significant part, not developed by or for the Corporation, but referred to or incorporated in a MIPI Specification, or (iv) any portions of any product and any combination except for that portion or portions which are required solely in order to achieve an interface that is compliant with a MIPI specification; (v) any methods or processes practiced, in whole or in part, over an Interface that are not expressly set forth in a MIPI Specification.

"Affiliate," means any corporation, partnership, or other entity that, directly or indirectly, owns, is owned by, or is under common ownership with, such Member hereto, for so long as such ownership exists. For the purposes of the foregoing, "own," "owned," or "ownership" shall mean ownership of more than fifty (50%) of the stock or other equity interests entitled to vote for the election of directors or an equivalent governing body of an entity that is directly or indirectly controlled by, under common control with or that controls the subject party.

"Board" means the Board of Directors of MIPI.

This page intentionally left blank.

Annex F I3C Basic Development Companies

2444   Analog Devices, Inc.

2445   Analogix Semiconductor, Inc.

2446   Avery Design Systems, Inc.

2447   Cadence Design Systems, Inc.

2448   Intel Corporation

2449   Introspect Test Technology Inc.

2450   InvenSense, Inc.

2451   L&T Technology Services

2452   Lattice Semiconductor Corp.

2453   MediaTek Inc.

2454   NXP Semiconductors

2455   Qualcomm Incorporated (provided notice of termination effective August 13, 2018)

2456   Robert Bosch GmbH

2457   SmartDV Technologies India Private Limited

2458   STMicroelectronics

2459   Synaptics

2460   Synopsys, Inc.

2461   Toshiba Memory Corporation

2462   Valens Semiconductor

This page intentionally left blank.

Participants

The list below includes those persons who participated in the Ad Hoc Working Group that developed this Specification and who consented to appear on this list.

Eugen Becker, Robert Bosch GmbH

Rajesh Bhaskar, Intel Corporation

Enrico Carrieri, Intel Corporation

Geraud Cheenne, STMicroelectronics

Ladvine D Almeida, Synopsys, Inc.

Kenneth Foust, Intel Corporation

Chris Grigg, MIPI Alliance

Paul Kimelman, NXP Semiconductors

Abinaya Kubendran, L&T Technology Services

Peter Lefkin, MIPI Alliance

Satwant Singh, Lattice Semiconductor Corp.

Przemyslaw Sroka, Cadence Design Systems, Inc.

Greg Stewart, Analogix Semiconductor, Inc.

Eyuel Zewdu Teferi, STMicroelectronics

Suresh Venkatachalam, Synopsys, Inc.

James Wang, InvenSense, Inc.

Miles Williams, Introspect Test Technology Inc.

This page intentionally left blank.